AppViewX Setup Guides

Version: 2023.1.0 FP3

Copyright AppViewX, Inc.

Trademarks

The trademarks, logos, and service marks displayed in this manual are the property of AppViewX or other third parties. Users are not permitted to use these marks without the prior written consent of AppViewX or such third party which may own the mark.

External Reference Links

This product includes software developed by the CentOS Project (www.centos.org).

This product includes software developed by Red Hat, Inc. (www.redhat.com).

This product includes software developed by VMware, Inc. (www.vmware.com).

All other trademarks mentioned in this document are the property of their respective owners.

Contact Information

AppViewX, Inc.

222 Broadway, FL 19

New York, NY 10038

Email: info@appviewx.com

Web: www.appviewx.com

Contents

Preface

Revision History

Revision	Description	Date
5.0	Updated draft of document for release v2023.10 FP3 HF2	July 2024
4.0	Updated draft of document for release v2023.10 FP3	June 2004
3.0	Updated draft of document for release v2023.10 FP2	February 2024
2.0	Updated draft of document for release v2023.1.0 FP1	November 2023
1.0	Initial draft of document for release v2023.1.0	September 2023

About this Guide

This guide outlines the steps for installing the AppViewX Windows Gateway for enabling communication between AppViewX and Windows. It also includes the steps for installing and using the AppViewX validator to validate the accessibility of the target machine on which the AppViewX Windows Gateway will be installed.

Audience

This guide is intended for AppViewX's customers deploying its products on Windows-based machines.

Text Conventions

The following text conventions are used in this document:

Convention	Description
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`codeblock`	Indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Chapter 1: AppViewX On-Prem Setup Guides

Install, Upgrade, and Maintenance Guide

This document covers the installation, maintenance, and upgrade activities for AppViewX.

Overview

AppViewX’s offering is a modular, low-code software application that enables the automation and orchestration of network infrastructure using an intuitive, context-aware, visual workflow. Leveraging a vast library of pre-built tasks and workflows, AppViewX enables the operations teams to quickly and easily translate business requirements into automation workflows that improve agility, enforce compliance, eliminate errors, and reduce cost. AppViewX is closed-loop and state-aware, capable of verifying that intent has been achieved and providing actionable insights and automated remediation.

AppViewX is a web based application that offers:

- CERT+, which lets you:
  - Discover, monitor, analyze, orchestrate and fully automate certificate lifecycle management and key management solutions.
  - Make a shift from reactive mode and be more proactive as you get a complete view of your entire certificate infrastructure.
  - Manage certificates as a service with pre-built integrations and extensible APIs that plugin to your enterprise applications, web servers, microservices, and multi-cloud environments.
  - Analyze certificates for crypto standards like key size, cipher strength, and allowed protocol versions.
  - Setup policies for enforcing high crypto standards.
  - Update certificates as per new policies.
  - Provision certificates for devices and applications.
  - Save resources, time, and effort of installation and maintenance.

  For details, refer the CERT+ User Guide.

- ADC+, which lets you:
  - Efficiently distribute network load or client requests across servers.
  - Send requests to the available servers, ensuring high application availability.
  - Scale the number of servers (up or down) based on the traffic.

  For details, refer the ADC+ User Guide.

- PKI+, which lets you:
  - Create root CAs and subordinate CAs and enroll them to the AppViewX PKIaaS certificate authority.
  - Onboard custodians to add root CAs and subordinate CAs to the PKI+ system.
  - Manage custodians for approving PKI+-related actions.

  For details, refer the PKI+ User Guide.

- SSH+, which lets you:
  - Discover and display SSH certificates alongside SSH keys, offering a more comprehensive overview of your security credentials.
  - Download keys for key-based access control, ensuring streamlined access management.
  - Specify access duration in either hours or days when requesting access to an infrastructure group, providing enhanced access management control.
  - Use a dynamic access flow that adapts to either key or certificate-based access, depending on the user's selected 'Access Mode' during host addition.
  - Rotate host certificates effortlessly, directly from the host inventory, promoting secure host certificate management.
  - Revoke SSH certificates directly, thus enhancing security control.

  - Choose between 'Key' and 'Certificate' access modes during host addition, with the 'Certificate' option being pre-selected by default.
  - Rotate and delete keys from hosts with multiple keys through the user and host key age report.

  For details, refer the SSH+ User Guide.

- SIGN+, which lets you:
  - Simplify Code Signing Certificate enrollment and Certificate Lifecycle Management (CLM) operations.
  - Customize signing policies according to your requirements
  - Integrate with AppViewX's customized Cryptographic Service Provider (CSP) and PKCS#11 for enhanced security.
  - Manage your code signing inventory with a full suite of tools and features.
  - Sign your code effortlessly using a variety of tools including SignTool, JSign, JarSigner, APKSigner, Mage, and Nuget.
  - Ensure compatibility with third-party Timestamp Authorities (TSA) for a wider range of options.

  For details, refer the SIGN+ User Guide.

- KUBE+, which lets you:
  - Simplify Certificate Lifecycle Management for Kubernetes workloads.
  - Get real-time visibility, central audit, and governance over K8’s Certs.
  - Achieve end-to-end automated certificate enrollment process.
  - Have secure and compliant PKI across K8s workloads (secrets, pods, and service mesh).

  For details, refer the KUBE+ User Guide.

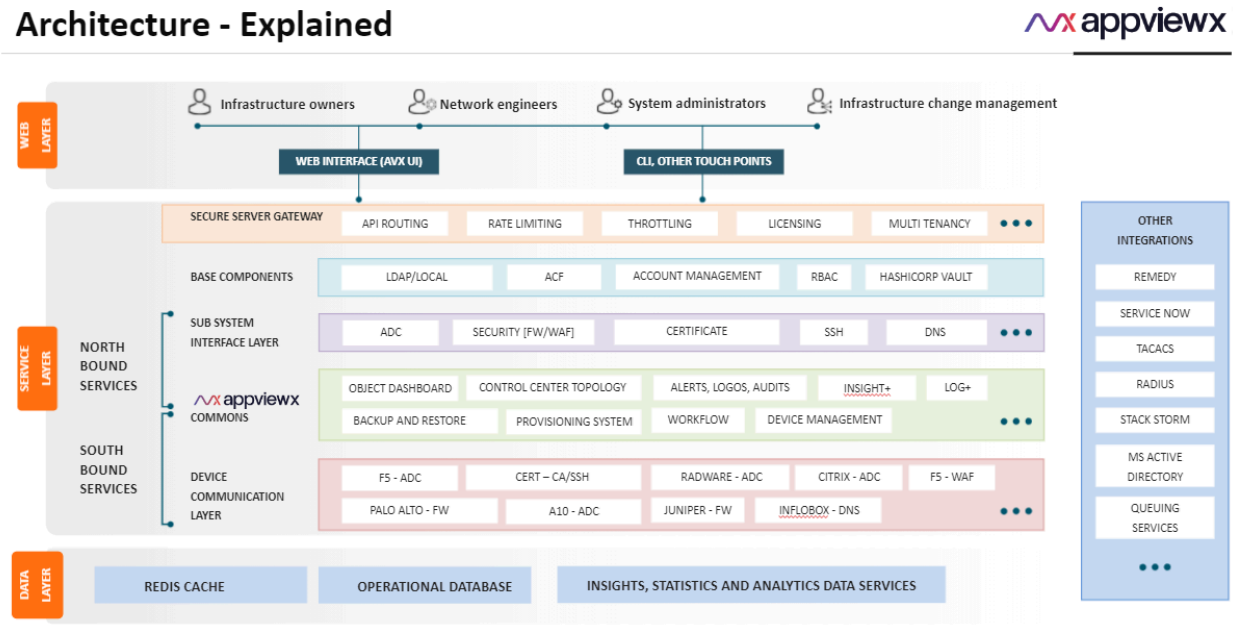
AppViewX is built on the microservice architecture. A microservice is a program that runs on a server or a virtual computing instance. The main task of this program is to respond to network requests.

- AppViewX Architecture
- Supported Deployment Methods and Types
- Understanding the Installation Steps

AppViewX Architecture

AppViewX is built on Kubernetes, an open-source platform for deploying and managing containers. It provides a container runtime, container orchestration, self-healing mechanisms, service discovery and load balancing. It’s used for the deployment, scaling, management, and composition of application containers across clusters of hosts.

AppViewX is designed based on microservice architecture making it easier to move to containerized workloads and the containers being orchestrated using Kubernetes. The following diagram depicts the deployment architecture:



In the diagram:

- **Presentation/ Web Layer** - houses the AppViewX user interface related files and interacts with the service layer
- **Service Layer** - contains the Northbound & Southbound services that can be further classified into:
  - **Business Layer**:
    - Houses AppViewX specific business logic
    - Interacts with the Data layer for persisting the input data
  - **Device Communication Layer**:
    - Low code
    - Stateless layer
    - Routes communication to the respective vendor through APIs or SSH
    - Houses vendor specific business logic
- **Data Layer**:
  - Houses data persistence and retrieval logic
  - Redis caching is available

Benefits of AppViewX

In order to optimally utilize the resources, AppViewX has adopted Kubernetes to achieve higher security by adopting a zero trust network model. The features of AppViewX coupled with Kubernetes are given below.

- **Auto scaling**

  AppViewX services can have a custom throttling capability based on pre-configured memory configuration per API. This will enable AppViewX services to utilize (scale up) resources optimally as the demand surges and scale down when not in use. This will help to horizontally scale the vendor components on demand and optimize the resource usage.

- **Resiliency**

  There is no guarantee that the services will run without any interruption and they are bound to failure. Kubernetes keeps deployments healthy by restarting containers that have failed, killing and replacing unresponsive containers based on health checks. This helps to mitigate the common pain point of the application upkeep process.

- **Security**

  AppViewX architecture is designed around the concept of zero trust network model to enforce tighter security within the Kubernetes cluster. This means no one is trusted by default and required verification to gain access to the services.

Supported Deployment Methods and Types

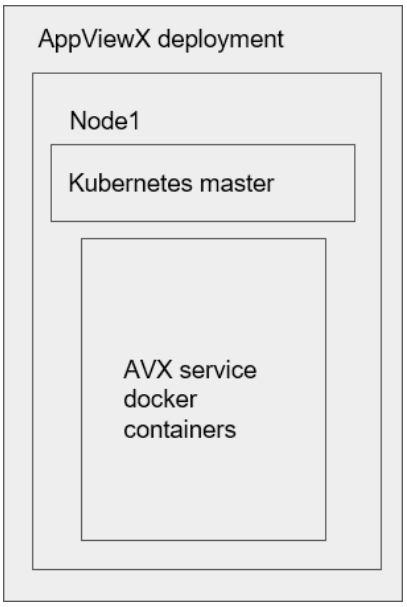
This section explains the types and methods in which you can deploy AppViewX.

> ⚠ **Warning:** Hybrid cloud management deployment is not supported in AppViewX.

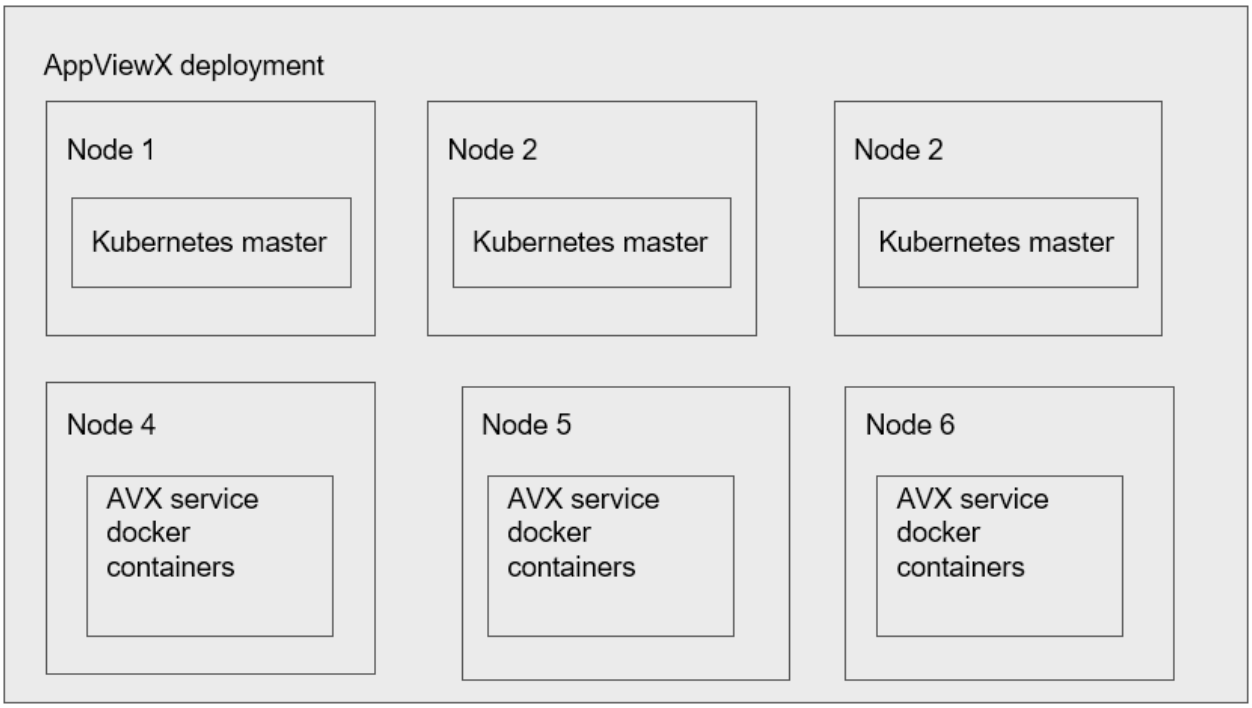
First, AppViewX can be deployed in the following modes:

1. **Single Node** - is used to host all the services on a single setup.
   - Single-node setups may have lower performance because of a lack of resources.
   - Node resiliency and HA are not supported in single-node deployment.
2. **Multi node** - is used to host the services across multiple nodes to ensure high availability.

The following diagrams depict AppViewX deployment on a single node and a multi node mode:



Single Node Deployment



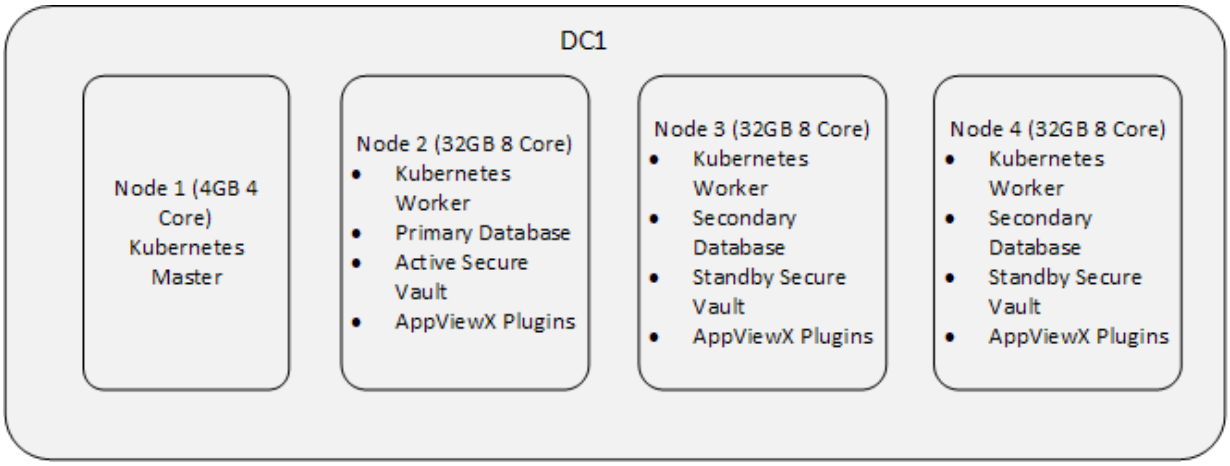
Multi Node Deployment

Once the deployment mode is finalized, AppViewx can be installed using any one of the following methods:

- **OVA Installation** - stands for Open Virtual Appliance that contains a compressed and installable version of a virtual machine. When you use an OVA-based installer, the installation-related artifacts are pre-bundled as part of the OVA.
- **Native Installation** - uses the standard command line interface to execute installation commands.

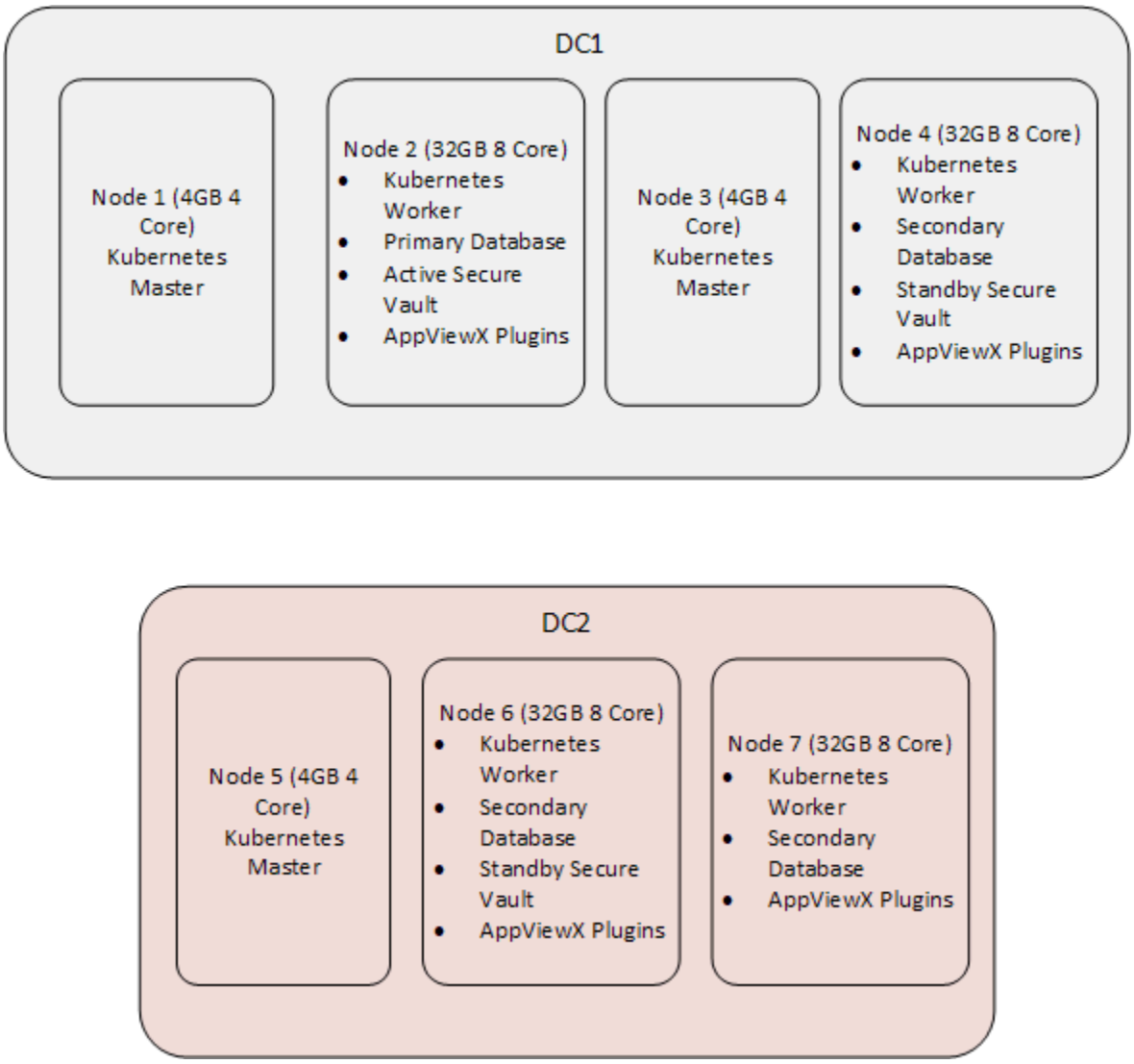
AppViewX supports the following deployment types/scenarios:

- One Data Center and Four Nodes



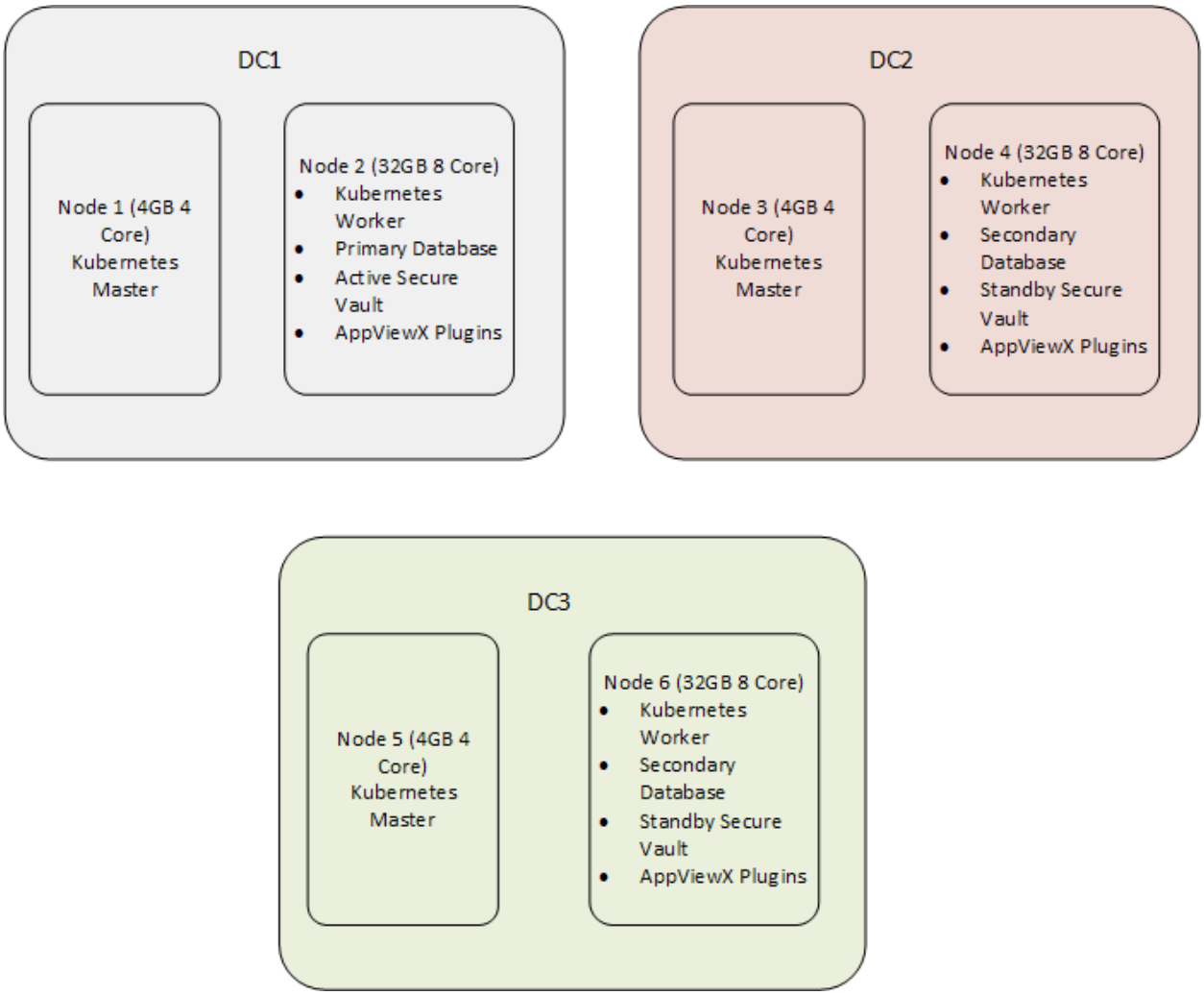
This deployment is recommended for customers who need only HA at the application level. This deployment does not support HA; neither for the Kube nor for the DC. This deployment is best suited for less than 50 ADC devices having a total of 100,000 objects and 10,000 certificates.

- Two Data Centers and Seven Nodes



This deployment is recommended for customers who require HA at the Application, Kube, and DC level. This deployment supports HA for the Kube, Application as well as the DC. This deployment is best suited for 50 to 100 ADC devices having a total of 300,000 objects and 10,000 certificates.

• Three Data Centers and Six Nodes



This deployment is recommended for customers who require HA at the Application, Kube, and DC level. This deployment supports HA for the Kube, Application as well as the DC. This deployment is best suited for 50 to 100 ADC devices having a total of 300,000 objects and 10,000 certificates.

The table below summarizes the different deployments supported by AppViewX.

Table - Deployments supported by AppViewX

Model	Load	HA		
		Kube	DC	Application
1 DC 4N	Less than 50 ADC devices having a total of 100,000 objects and 10,000 certificates.	No	No	Yes
2 DC 7N	50 to 100 ADC devices having a total of 300,000 objects and 10,000 certificates.	Yes	Yes	Yes

Table - Deployments supported by AppViewX (continued)

Model	Load	HA		
		Kube	DC	Application
3 DC 6N	50 to 100 ADC devices having a total of 300,000 objects and 10,000 certificates.	Yes	Yes	Yes

Note: Apart from the deployments mentioned here, AppViewX can customize the deployment based on the needs and requirements.

Understanding the Installation Steps

This section outlines the various mandatory and optional steps in the process of installing AppViewX.

Table - Sequence of Installation Steps

Step No	Step Name	Mandatory	Optional
1	Working with Prerequisites	Yes	No
2	Configuring Firewall	Yes	No
3	Configuring Elevated Access	Yes	No
4	Downloading AppViewX Packages	Yes	No
5	Running the Prerequisite Tool	Yes	No
6	Deploying the AppViewX Virtual Appliance	No	Yes
7	Performing a Single Node or Standalone Installation	No	Yes
8	Performing a Multi-node or High Availability Installation	No	Yes
9	Configuring the appviewx.conf file	No	Yes
10	Configuring the POD and Service IP CIDR	No	Yes
11	Verifying the Installation	Yes	No
12	Uploading the License Key	Yes	No
13	Accessing the AppViewX Graphical User Interface	Yes	No

Table - Sequence of Installation Steps (continued)

Step No	Step Name	Mandatory	Optional
14	Adding Third-party Libraries	No	Yes

Working with Prerequisites

This section covers all the prerequisites required to install AppViewX on the system.

- Understanding Hardware and Software Requirements
- Configuring Elevated Access
- Configuring Firewall Ports
- Configuring YUM
- Configuring Calico before Deployment
- Configuring SELinux
- Configuring NTP
- Configuring Ulimit
- Increasing vm.max_map_count
- Enabling IP Forwarding
- Enabling Bridging
- Enabling the IP in IP Protocol
- Downloading AppViewX Packages
- Running the Prerequisite Tool

Understanding Hardware and Software Requirements

Hardware Requirements

Before proceeding with the installation, ensure that you have, at minimum, the following hardware with the specifications given below:

• Single Node Requirements

Node	Bare Minimum		
	CPU	RAM	Hard Disk Space
Single Node	8	32 GB	500 GB

• Multi Node Requirements

Node	Bare Minimum		
	CPU	RAM	Hard Disk Space
Multi-node (master node)	4	4 GB	100 GB
Node	**Bare Minimum**		
	CPU	**RAM**	**Hard Disk Space**
Note: One node for a single master installation and a minimum of three nodes for multi-master installation.			
Node	**Bare Minimum**		
	CPU	**RAM**	**Hard Disk Space**
Multi-node (worker node)	8	32 GB	500 GB

Note: For more information on the nodes, refer to the Supported Deployment Methods and Types section.

For deploying the OVA, ensure that you have all the prerequisites as mentioned below.

• Platform Bare Minimum Requirements

Supported Virtualization Platforms	Versions	VCPU	RAM	HDD
VM Server, VMware ESXi	5.5 or later	8v	32 GB	1 TB

Software Requirements

- RHEL 8.7
- RHEL 8.8
- RHEL 8.10
- RHEL 9.2
- RHEL 9.3
- RHEL 9.4
- Ubuntu 20.04
- Ubuntu 22.04

Ensure the following packages are installed before proceeding with the AppViewX installation:

For RHEL:

- nmap
- curl
- bindutils
- net-tools
- rsync
- nmapncat
- fontconfig
- zip
- unzip
- sysstat
- tcpdump

For UBUNTU:

- nmap
- curl
- ebtables
- sysstat
- zip
- unzip
- rsync
- dnsutils
- fontconfig
- net-tools
- tcpdump

Configuring Elevated Access

AppViewX is installed on top of a Kubernetes engine and to install the underlying Kubernetes engine and other dependent packages like docker, we would require the user to have sudo access and executable permission for the tmp folder. Refer to the Understanding Commands Executed during Installation section to get the details on the commands that the sudo user needs access to.

Note: If you are using an OVA-based installer, a user named "appviewx" is already available with Super user privileges.

Configuring Firewall Ports

The following ports must be opened between the nodes to install AppViewX. Users can configure it in a firewall device, firewalld, or using iptables.

Table - Firewall Ports

Sr No	Source		Destination		Protocol Used	TCP/ UDP	Type of Information Communicated
	IP	Port	IP	Port			
1	All Nodes	Any	All Nodes*	22	SSH	TCP	Required for AppViewX installation and prerequisite checks.
2	All Nodes	Any	All Nodes*	179	BGP	TCP	To establish a common routing table for the overlay network.
3	All Nodes	Any	All Nodes*	6443	HTTPS	TCP	Kubernetes API server for communication between Kubernetes master and worker nodes.
4	All Nodes	Any	All Nodes*	10250	HTTPS	TCP	Used by Kubelet Agent which exposes Rest endpoints for the Kubernetes API Server.
5	Load Balancer (for ex, F5, GCP, etc.)	Any	ISTIO Ingress Proxy IP (Kube Worker)	31443	HTTPS	TCP	To access the AppViewX web user interface.

Table - Firewall Ports (continued)

Sr No	Source		Destination		Protocol Used	TCP/ UDP	Type of Information Communicated
	IP	Port	IP	Port			
6	Load Balancer (for ex, F5, GCP, etc.)	Any	Kube Master IP	6443		TCP	To allow communication between the F5 load balancer and the pool members (master nodes).
7	All Nodes	Any	F5 VIP	6443		TCP	To allow all the nodes to communicate with the Kube Master for Kubernetes Control plane traffic.
8	AppViewX Admin network #	Any	ISTIO Ingress Proxy IP (Kube Worker)	30190	HTTPS	TCP	To access the AppViewX management console.
9	All Nodes	-	All Nodes*	-	IP-IP IP Protocol 4	NA	Overlay network established with IP-IP tunnels. Information over this tunnel is encrypted using mTLS.
10	Master	Any	Kube Master	2379	HTTPS	TCP	Required for etcd server communication in a multi-master setup.
11	Master	Any	Kube Master	2380	HTTPS	TCP	Required for etcd server communication in a multi-master setup.
12	All Nodes	Any	All Nodes*	9100	HTTP	TCP	Required for monitoring the node metrics.
13	All Nodes	Any	All Nodes*	4789	VXLAN	UDP	To establish a common routing table for the overlay network.

* (asterisk) indicates all the nodes present in the cluster i.e. master nodes, secondary master nodes, and worker nodes.

- indicates the network/machines/nodes of users who want to manage AppViewX Infra using the management console (actions include create, delete pods, and/or services.

Note:

- The system will require 1 IP per node.
- The externally exposed services will all use the nodes IP address to communicate within the network.
- Port 22 is used for administration of the node for example to log into the linux CLI. Need SSH access the nodes to other nodes.
- We would need an external Load Balancer to distribute user/API traffic to all Kube master nodes. We can open firewall ports depending on the network setup.
- Ensure that the external endpoints that you want to access from the AppViewX worker nodes are accessible., e.g. Microsoft CA. Ensure that the corresponding ports and URLs are opened for communication.

Configuring Firewall Ports for External Integrations

Table - Firewall Ports for External Integrations

S. No	Source		Destination		Protocol Used	TCP/ UDP	Type of Information Communicated
	IP	Port	IP	Port			
1	AppViewX Worker Nodes	Any	ADC		SSH		
2	AppViewX Worker Nodes	Any	ADC		HTTPS		To execute REST APIs
3	AppViewX Worker Nodes	Any	MSCA Agent		HTTPS		AppViewX to MSCA agent communication
4	AppViewX Worker Nodes	Any	CA		HTTPS		To execute REST APIs

Configuring YUM

This section guides users to configure AppViewX nodes to the YUM repository hosted by AppViewX. Yum will sync only AppViewX repositories to get the OS package updates. This task is required to update the OS security patching on AppViewX supplied OVAs.

Note: For information regarding the best practices on rebooting the operating system after security patching, refer to the Understanding the Best Practices on Reboot Sequence.

Warning: This will remove all the other repositories configured in the system.

Before you configure yum, ensure that:

1. AppViewX nodes have access to the following URL https://repos.appviewx.com
2. The user has root/sudo access to configure yum.

To configure YUM:

1. Download the **appviewx.repo** file from the release portal.
2. Login as a root user.
3. To take a backup of existing yum repositories, execute the following command:

   ```
   mv /etc/yum.repos.d /etc/yum.repos.d_backup
   ```

   This is to ensure that we have a backup of the existing yum repository configurations.
4. To create a yum repository, execute the following command:

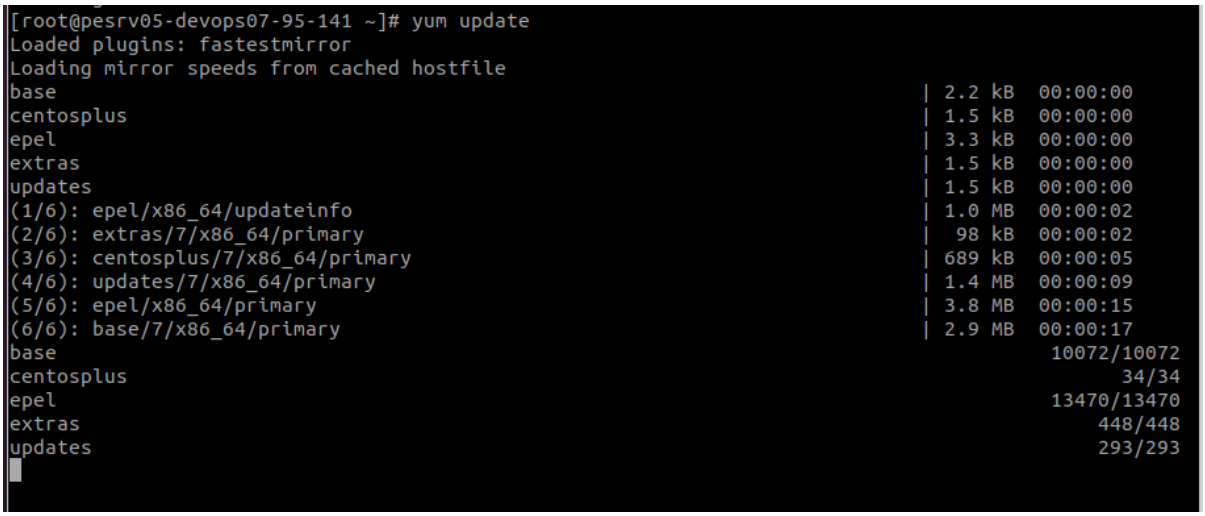
   ```
   mkdir -p /etc/yum.repos.d
   ```
5. To copy the appviewx.repo to yum.repos.d, execute the following command:

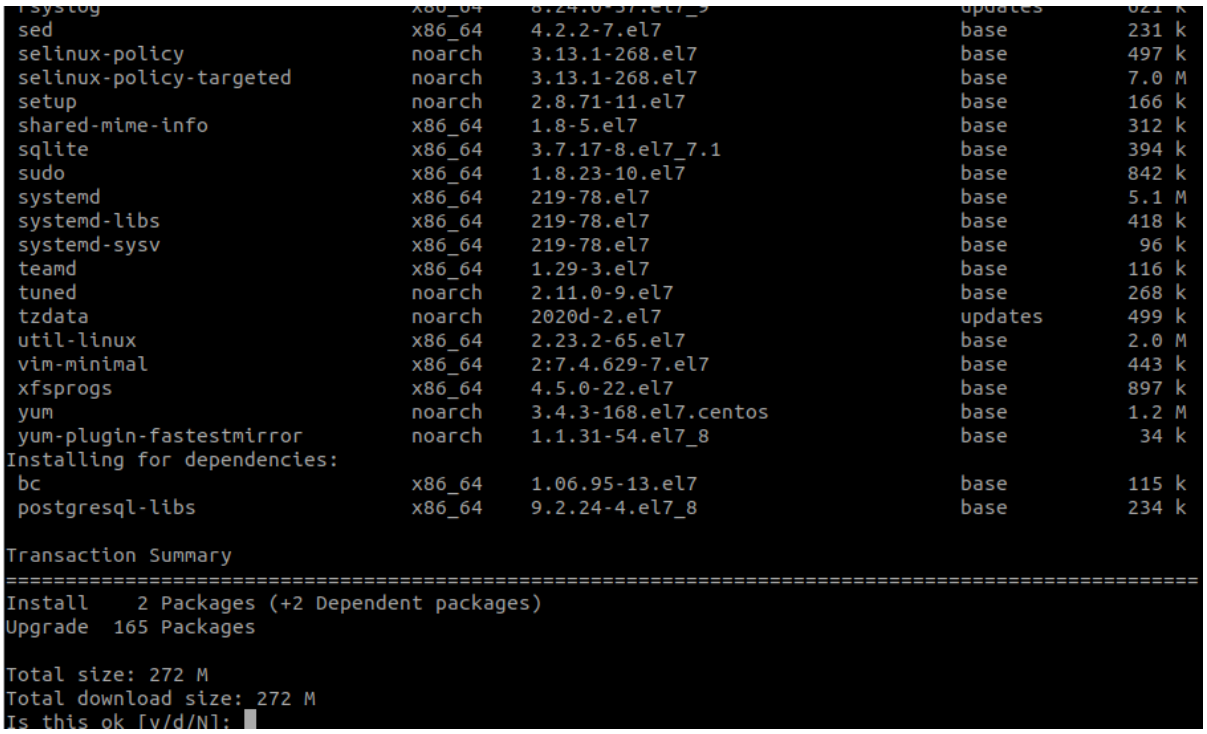
   ```
   cp appviewx.repo /etc/yum.repos.d/
   ```
6. To clean the yum repository, execute the following command:

   ```
   yum clean all
   ```
7. To get the latest updates from repos.appviewx.com, execute the following command:

   ```
   yum update
   ```

   The command will connect to the AppViewX repository and update the packages. Reference images are given below:





Configuring Calico before Deployment

This section provides instructions on configuring calico before deploying AppViewX on Azure.

> ⚠ **Warning:** Follow these instructions ONLY if you are deploying AppViewX on Azure.

1. Navigate to the /home/appviewx/appviewx_kubernetes/configs/kube directory.
2. Open the **calico.yaml** file in edit mode.
3. Change the value of the **CALICO_IPV4POOL_VXLAN** parameter from **CrossSubnet** to **Always**.
4. Change the value of the **CALICO_IPV4POOL_IPIP** parameter from **Always** to **Never**.
5. Save the changes to the **calico.yaml** file.
6. Close the editor.

Configuring SELinux

To configure SELinux

1. Open the file /etc/selinux/config file
2. Configure the parameters, **SELINUX=permissive** or **SELINUX=disabled**
3. Reboot the node by the command

```
sudo reboot
```

4. Verify that the command output below should be permissive

```
getenforce
```

Configuring NTP

To configure NTP

1. Install the NTP service by the command

```
sudo yum install ntp
```

2. Update the NTP server details in **/etc/ntp.conf** or **/etc/chrony.conf**
3. Restart the NTPD/chronyd service by the command

```
sudo systemctl start ntpd
```

4. Verify the NTP status using command

```
ntpstat
```

Configuring Ulimit

To set or verify the ulimit values on Linux:

1. Edit the **/etc/security/limits.conf** file and specify the following values:
   - **<USERNAME> soft nofile 65536**
   - **<USERNAME> hard nofile 65536**
2. Exit and login again to verify the changes.
3. Verify the Ulimit using the command

```
ulimit -n
```

Increasing vm.max_map_count

To increase the vm.max_map_count

1. Execute the command

```
sudo sysctl -w vm.max_map_count=262144
```

2. Verify the value using the command

```
cat /proc/sys/vm/max_map_count
```

Enabling IP Forwarding

1. In the **/etc/sysctl.conf** file, add the parameter **net.ipv4.ip_forward=1**
2. Execute the command

```
sudo sysctl -p
```

3. Verify the IP_Forwarding using the command

```
sysctl net.ipv4.ip_forward
```

Enabling Bridging

To enable bridging

1. In **/etc/sysctl.conf** file, add the following parameters:
   - **net.bridge.bridge-nf-call-ip6tables = 1**
   - **net.bridge.bridge-nf-call-iptables = 1**

2. Execute the following commands:

```
sudo modprobe br_netfilter
```

```
sudo sysctl -p
```

3. Verify the bridging using the command

```
sysctl net.bridge.bridge-nf-call-iptables
```

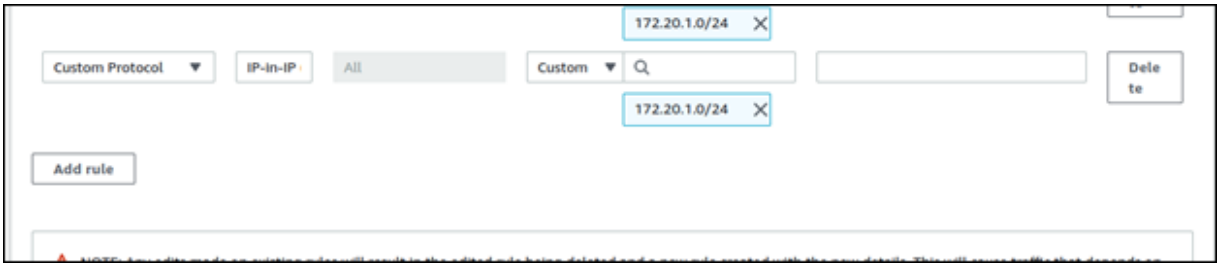
```
sysctl net.bridge.bridge-nf-call-ip6tables
```

Enabling the IP in IP Protocol

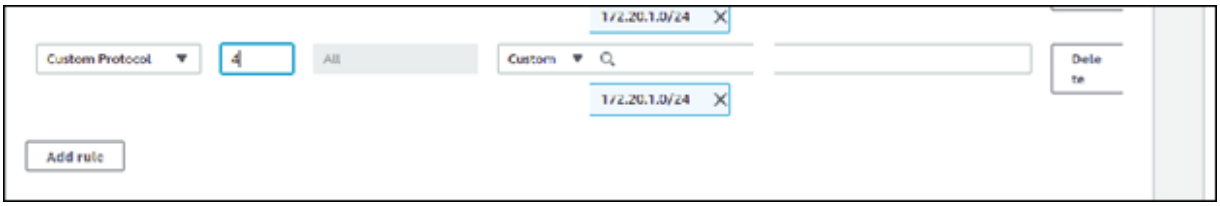
> ⚠ **Warning:** Follow these steps ONLY if you want to deploy AppViewX on AWS.

You must enable the IP in IP protocol between the nodes in the AWS security group before deploying AppViewX.

1. Log in to the AWS console.
2. Navigate to the security group that needs to be modified.
3. Click **Edit inbound rules**.
4. Click **Add rule**.
5. From the **Add rule** list, select **Custom Protocol**.

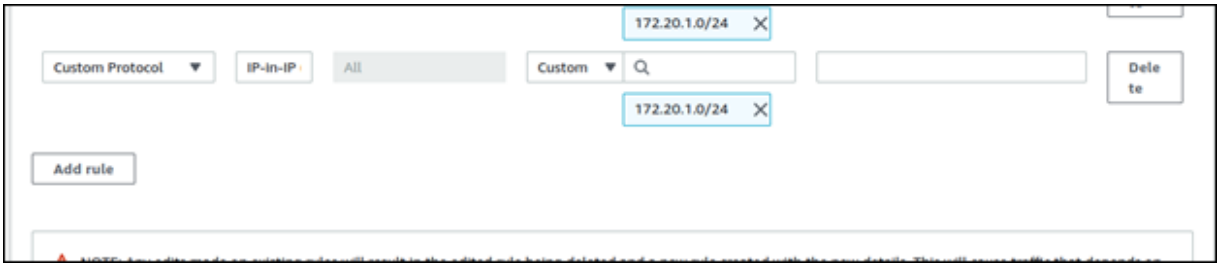


6. Enter the protocol value as **4**.



7. Enter the subnet across which IP in IP needs to be enabled.
8. Click **Save rule**.

The protocol automatically changes to IP in IP.



Downloading AppViewX Packages

To install AppViewX, download the following packages from the AppViewX Release Portal.

Note: To get the release portal credentials, contact help@appviewx.com.

File Name	Mandatory	Description	Purpose
appviewx_kubernetes_2023.1.2.0.tar.gz	Yes	AppViewX core installer	Core installer that has the AppViewX package from which the installation is triggered.
appviewx_kubernetes_addons_2023.1.2.0.tar.gz	Yes	To install AppViewX addons	Additional software to support the functionalities of AppViewX. This is mandatory for the installation.
appviewx_kubernetes_elk_2023.1.2.0.tar.gz	Optional	ELK stack to monitor logs	Additional package to install a GUI-based log collector to troubleshoot and Graffana-based UI to monitor the application performance.
appviewx_kubernetes_insight_2023.1.2.0.tar.gz	Optional	Insight for AppViewX Insight module	The insight package is an additional package to enable AppViewX to collect the statistical information of devices

File Name	Mandatory	Description	Purpose
			managed by AppViewX and generate it as a report.
upgrade.tar.gz		To upgrade from the existing version	This package is required to upgrade from older versions of AppViewX to 2023.1.0. FP2
prerequisite_utils.tar.gz		To check whether all the components are available.	The tool checks whether all the required prerequisites are present on the system.

Note: All OVA related updates are maintained by AppViewX and are available on the release site.

Running the Prerequisite Tool

The prerequisite tool

- **validates** for environment readiness and also **configures** some of the system configurations to ensure a smooth and successful AppViewX deployment.
- is a secure and reliable way to avoid any adverse effects on the environment.
- logs all changes made to the environment configuration for auditing and troubleshooting purposes.

Sudo permissions are required to execute the tool. This utility can be executed from any of the nodes; either worker or master. It is available in github - prerequisite utility.

The table below lists all the validations and the possible configurations performed by the prerequisite tool.

System Configuration	Validation	Configuration	Reason
Validating the system architecture - x86_64	Yes	No	System level configuration

System Configuration	Validation	Configuration	Reason
Validating RAM in worker nodes and master nodes	Yes	No	System level configuration
Validating CPU cores in worker nodes and master nodes	Yes	No	System level configuration
Validating disk space	Yes	No	System level configuration
Verifying ports communication	Yes	No	System level configuration
Cleaning up the process running on ports	Yes	Yes	NA
Validating OS	Yes	No	OS - RHEL/ CentOS/ Ubuntu
Validating OS version	Yes	No	Validating version of OS
Installing rpm/ deb dependency packages	Yes	Yes	NA
Validation for chrony and chrony sync status	Yes	Yes	NA
Validating NTP and NTP sync status	Yes	Yes	NA
Validate Runc version	Yes	Yes	NA
Validating Selinux status	Yes	Yes	NA
Validating IP of node	Yes	No	Validating the provided Ip of the nodes
Validating user id value	Yes	No	User id check
Validating umask value	Yes	Yes	NA
Validating ulimit value	Yes	Yes	NA
Validating Openssl version	Yes	No	Openssl version validation
Validating time difference between the servers	Yes	Yes	NA
Validating ftype in xfs_info	Yes	No	Checking supported file system

System Configuration	Validation	Configuration	Reason
noexec for /tmp	Yes	No	Checking permission for the /tmp directory
Validate IPV6 service	Yes	Yes	NA
Check vm.max_map_count value	Yes	Yes	NA
Server latency check	Yes	No	Latency check in and between the servers
Validate loopback IP - 127.0.0.1	Yes	Yes	NA
Validate Firewalld service	Yes	Yes	NA
Network interface lookup	Yes	No	Validating the network interface
Packet analyser	Yes	No	capture, inspect, and analyze network traffic
Analysing received packets	Yes	No	capture, inspect, and analyze network traffic
Validation of Crontab of user	Yes	Yes	NA
Validate puppet status	Yes	Yes	NA
Validate IP Tables	Yes	Yes	NA
Validate ip_forwarding	Yes	Yes	NA
Validate bridging	Yes	Yes	NA
Validating proxy variable values	Yes	Yes	NA
Cross validating SSH communication	Yes	No	Validation of ssh to nodes
Validating the GID	Yes	No	Verify that the GID we're validating corresponds to an existing group on the system
Validating User Id between the user of all servers	Yes	No	Ensuring user id assignment across servers is same

System Configuration	Validation	Configuration	Reason
Validating group Id between the user of all servers	Yes	No	Ensuring GID assignment across servers is same

To run the prerequisite tool:

1. Create a folder to download the **prerequisite.tar.gz** Eg:

   ```
   mkdir -p <folder_name>
   ```

2. Download and extract the **prerequisite.tar.gz** file to the created folder. To extract the file use the command below.

   ```
   tar -xvf prerequisite.tar.gz
   ```

3. To update the **hosts_template** execute the command below. Specify the AppViewX IP address of the VMs (master and worker nodes), DNS servers and gateway address, and users in the file.

   ```
   vi hosts_template
   ```

   Add the following values in the host_template parameters below.

Parameters	Description
NODES	It is the IP addresses of all the nodes. You can enter multiple comma separated values. Example:`NODE = 192.111.222.333`
USERS	The username for the nodes and must have sudo privileges. Example:`USERS = appviewx` In case of multi-node setup, enter the sudo username with comma separated value the number of times equivalent to number of AppViewX nodes (including masters & workers). for example, for a 2 DC 7 node setup, we need to enter the username 7 times - appviewx,appviewx,appviewx,appviewx,appviewx,appviewx,appviewx.

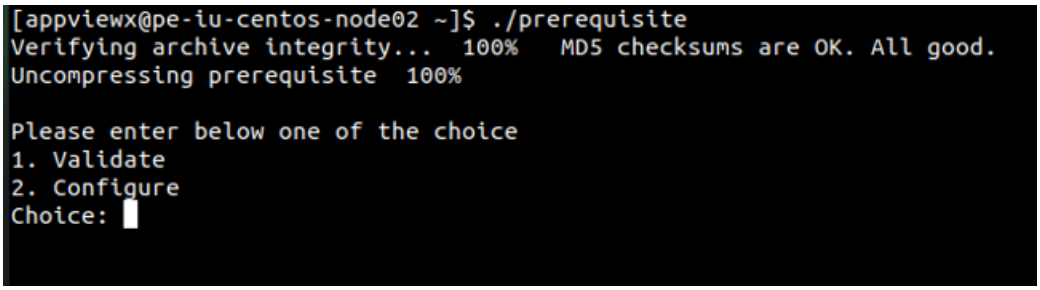
Parameters	Description
	Note: Usually for a multi-node setup the username for all nodes (master and slave) must be the same.
MASTER_NODES	IP address of the master node Example:`MASTER_NODES = 192.111.222.444`
WORKER_NODES	IP address of the master node Example:`WORKER_NODES = 192.111.222.555`
NEW_INSTALLATION_PATH	Example:`NEW_INSTALLATION_PATH = /home/appviewx/appviewx_cluster`
USER_GENERATED_PEM	If you require to do a password-less installation of AppViewX, enter the value as True. If set to False, then you will be prompted to enter the password after you execute the `./prerequisite` command Example:`USER_GENERATED_PEM = TRUE`
PRIVATE_KEY_FILE_PATH	Specify the path where the pem file is saved. Example:`PRIVATE_KEY_FILE_PATH = /tmp/user_generated_private.pem`
CHRONY_SYNC	If you require the time sync in the nodes enter the value as true. Example:`CHRONY_SYNC = TRUE`
CHRONY_SERVER	Specify the chrony server to be used for time sync. Example:`CHRONY_SERVER = abcs.appviewx.net`
HTTP_PROXY	If internet is not present in the nodes then use the http proxy to install the rpm/deb packages in the nodes. Example:`HTTP_PROXY = https://192.999.888.777:1234`

Parameters	Description
HTTPS_PROXY	If internet is not present in the nodes then use the http proxy to install the rpm/deb packages in the nodes. Example:`HTTPS_PROXY = https://192.999.888.777:1234`

4. Execute the following command:

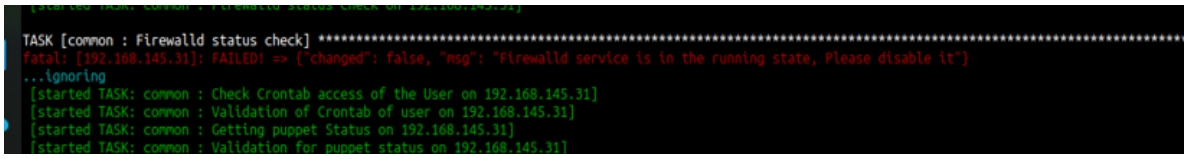
```
./prerequisite
```

The following options are displayed.

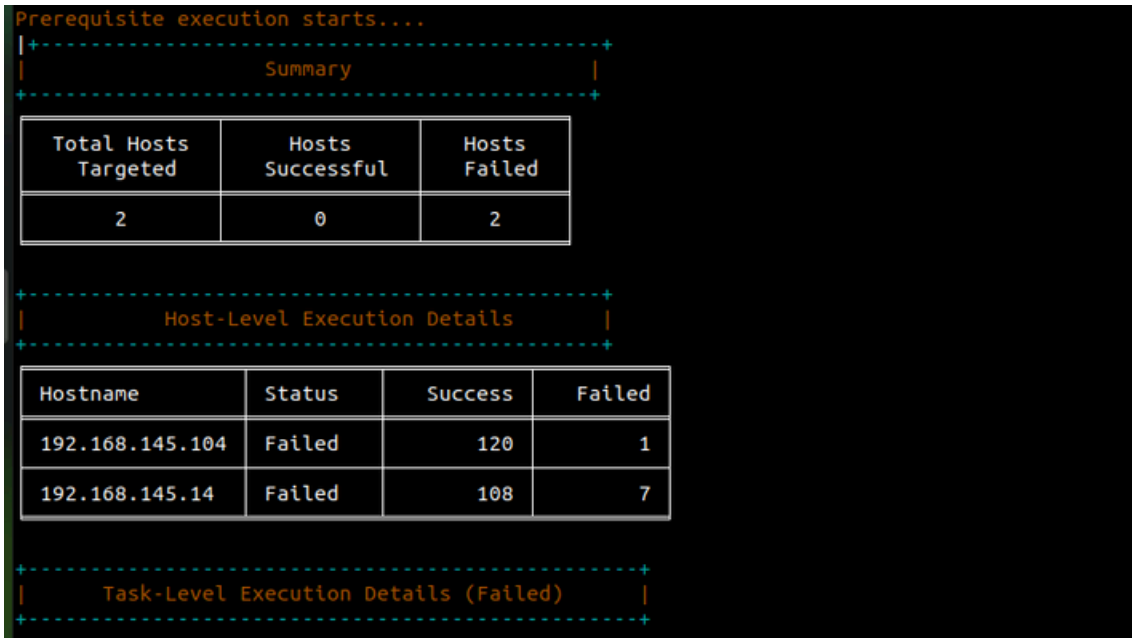


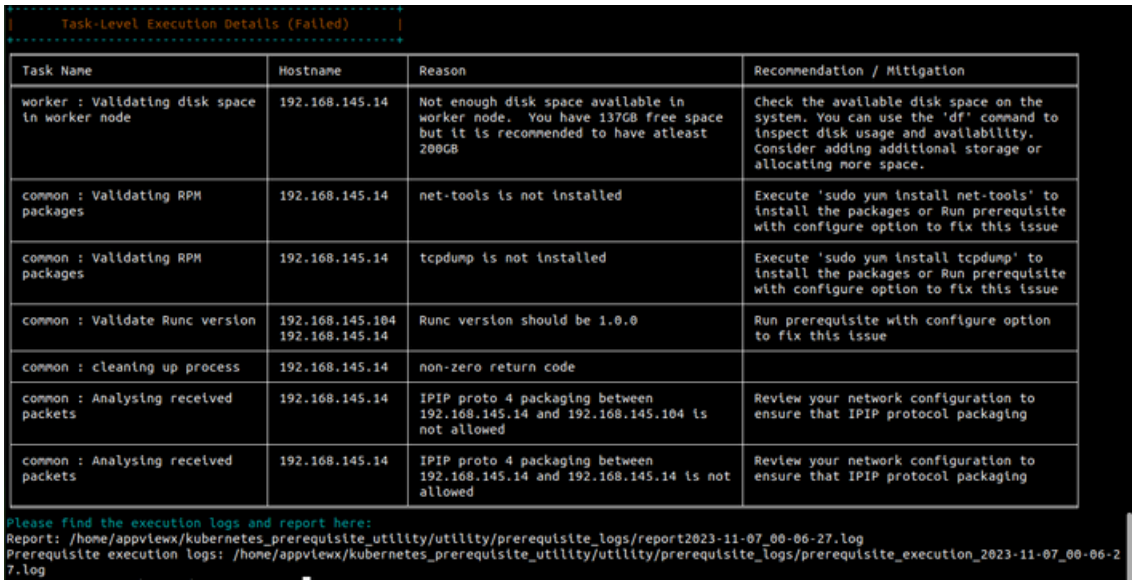
a. Enter the choice as 1 or 2

b. If you enter 1, then the tool validates all the system configurations specified in the table. If the configurations are not as per expectations the errors/failures will be displayed as shown below.



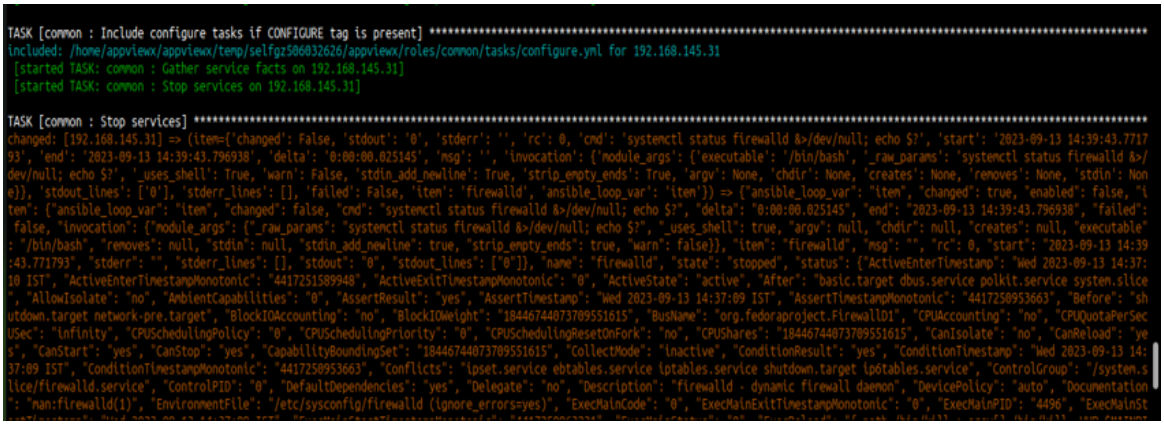
Post the validation the results are displayed as report in the format below. It shows the Summary, Host-Level Execution Status, and the Task-Level Execution Status (Failed). (*The screenshots shown are for a multi-node setup.*)



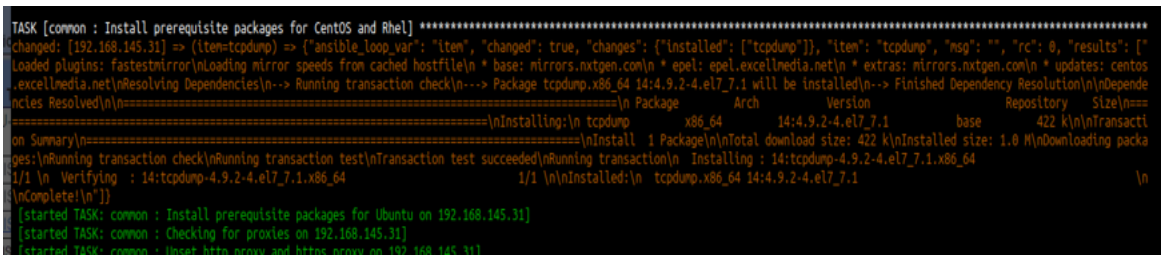


c. If you enter 2, then the tool will install the configurations as specified in the table and also validates all the system configurations.

- It stops the firewall service.



- It installs the rpm/deb package if not already installed.



d. To view the execution logs for both validate and configure, refer the **prerequisite_execution.log** file that is present in the location where the **prerequisite** file is present.

Deploying the AppViewX Virtual Appliance

An OVA file is an open virtualization appliance that contains a compressed and installable version of a virtual machine. If you are using an OVA-based installer, the installation-related artifacts are pre-bundled as part of the OVA.

The following packages are pre-bundled as part of the OVA:

- **appviewx_kubernetes_2023.1.3.0.tar.gz**
- **appviewx_kubernetes_elk_2023.1.3.0.tar.gz**
- **appviewx_kubernetes_insight_2023.1.3.0.tar.gz**
- **appviewx_kubernetes_addons_2023.1.3.0.tar.gz**
- **upgrade.tar.gz**

To deploy an OVA file, refer the sections below.

Downloading the Release Package

To download the release package,

1. Visit the AppViewX download URL at https://release.appviewx.com.
2. Download the release package in <.ova> format into the Downloads folder or the Desktop in your environment.
   For example, setting up a VM for a master or worker required either of the OVA
   - **appviewx_2023.1.0_prod_ubuntu_master.ova**
   - **appviewx_2023.1.0_prod_ubuntu_1TB_Worker.ova**
   - **appviewx_2023.1.0_prod_ubuntu_500GB_Worker.ova**
3. Validate the md5sum of the downloaded file
4. Open a terminal window.
5. To display the md5sum value of the downloaded file, execute the command:
   a. To display the md5sum value of the downloaded file, execute the following command:

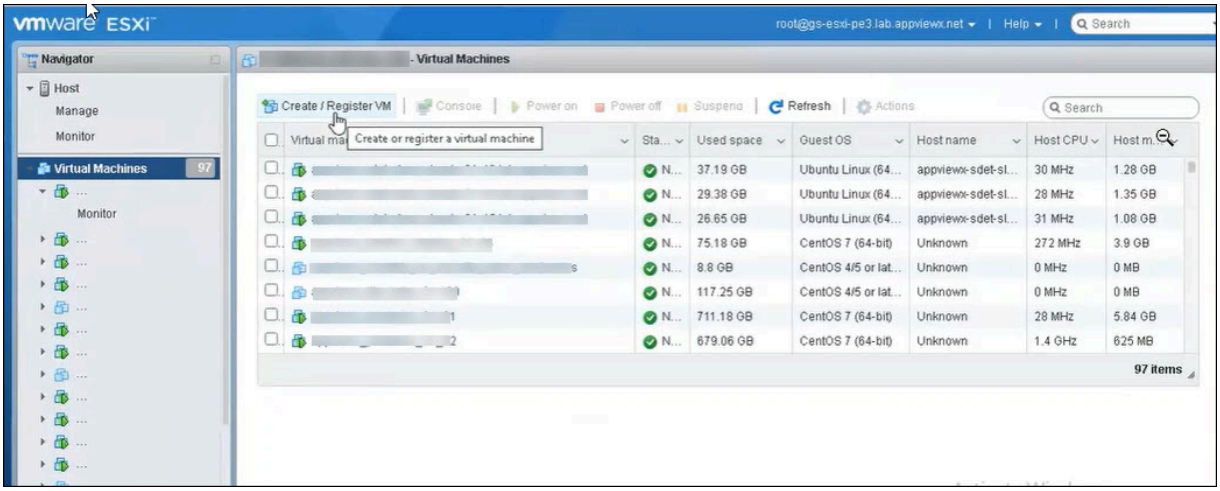
```
md5sum <filename>
```

   b. Match the displayed value against the original value from the release portal.

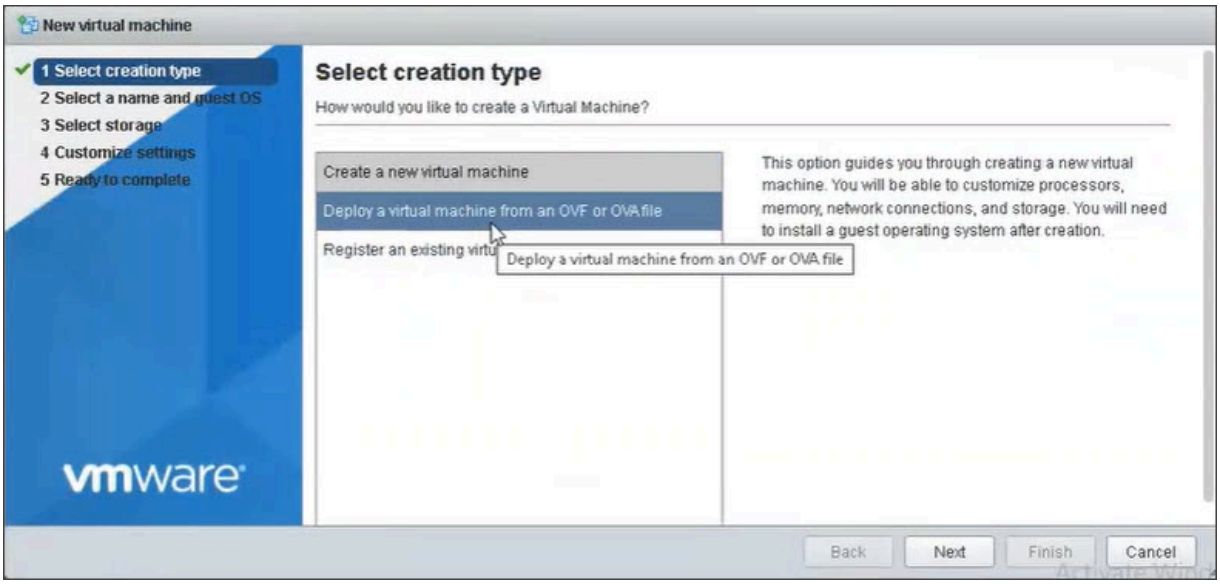
Installing the AppViewX OVA

This section covers the procedures for installing the AppViewX master and worker OVA.

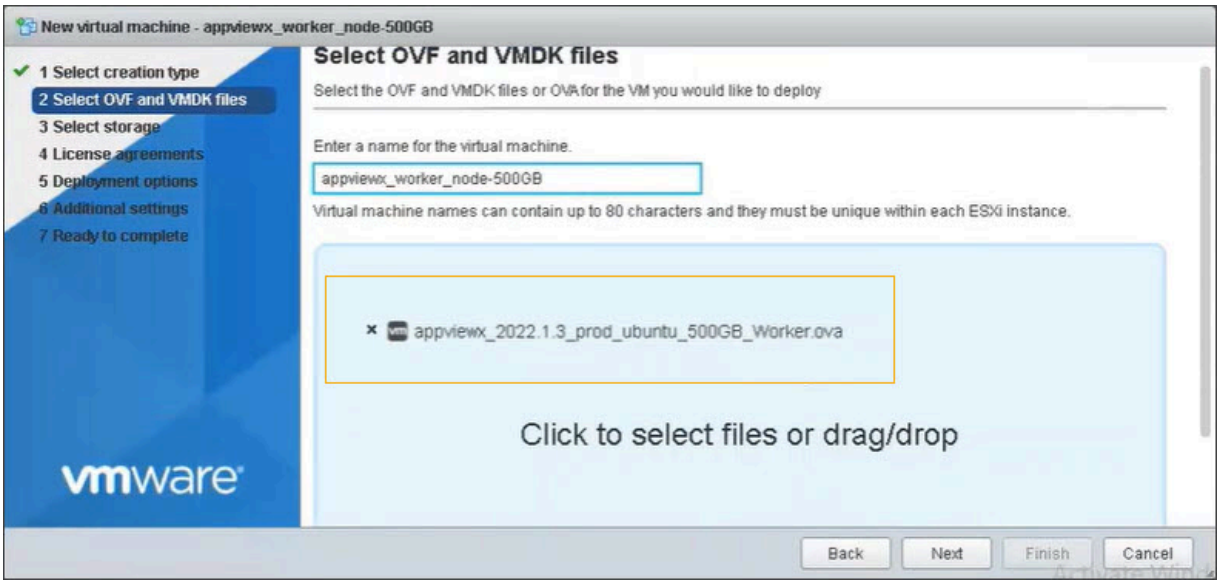
1. Log into the ESXI Server, go to **Virtual Machines > Create/Register VM**.



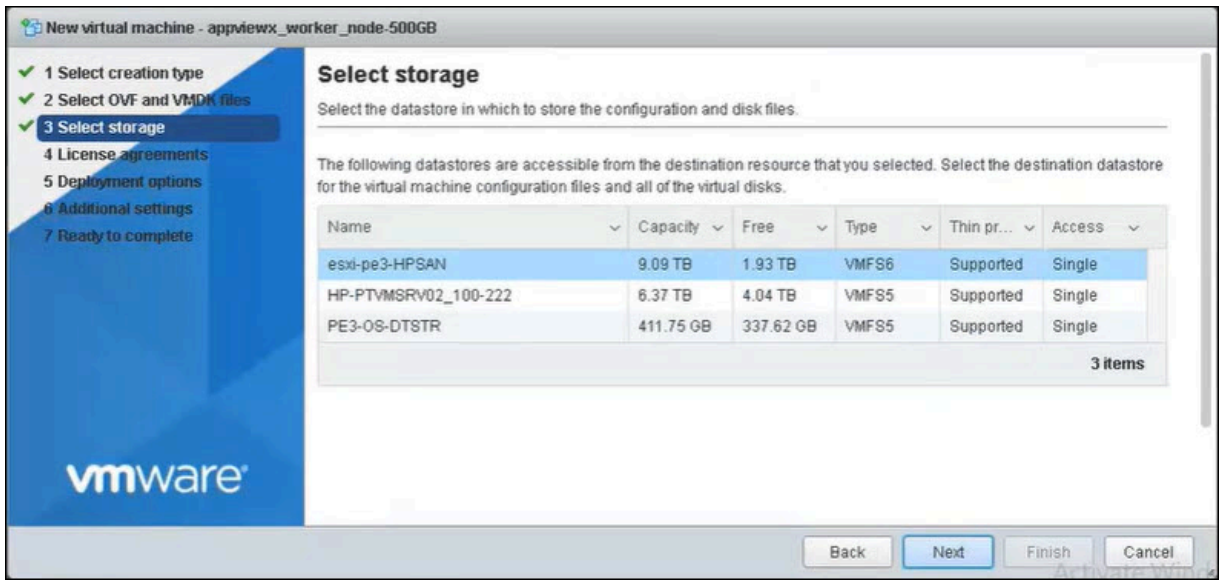
2. Select **Deploy a virtual machine from an OVF or OVA file** and click **Next**



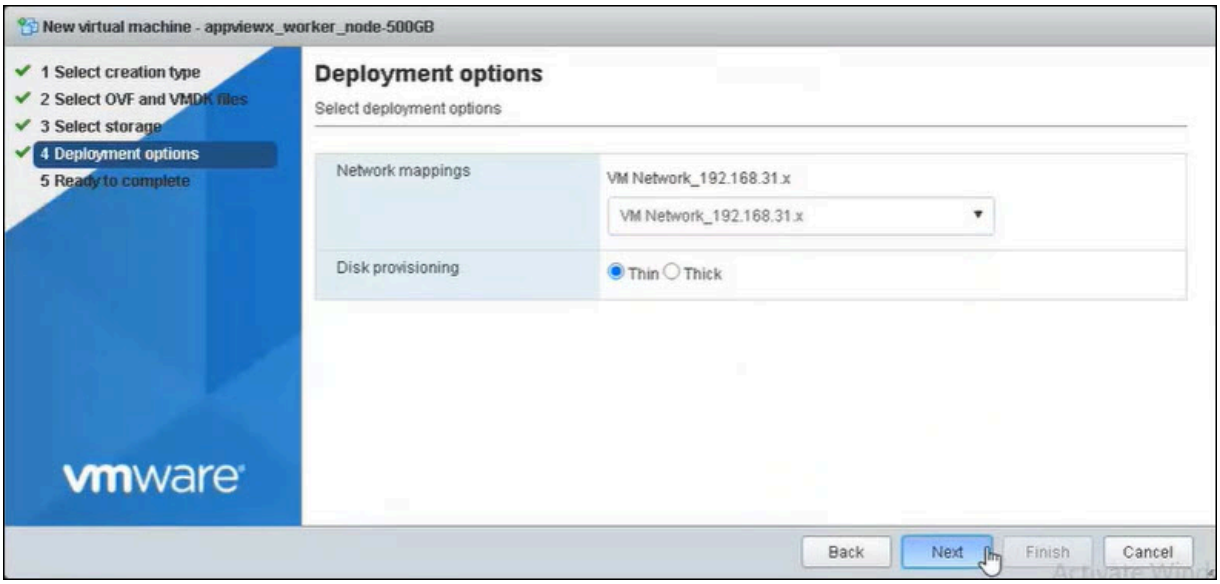
3. On the Source screen, *Enter a name for the virtual machine* in the text field provided and select **Click to select files or drag/drop option**. Click **Next**



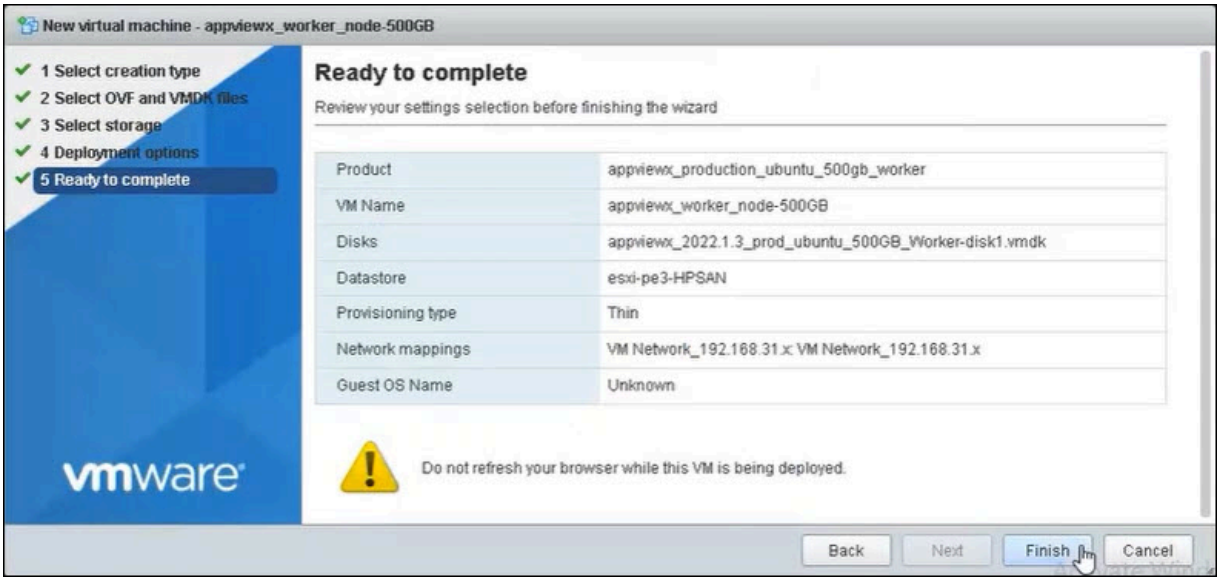
4. On the Select Storage screen, select a storage location and click **Next**.



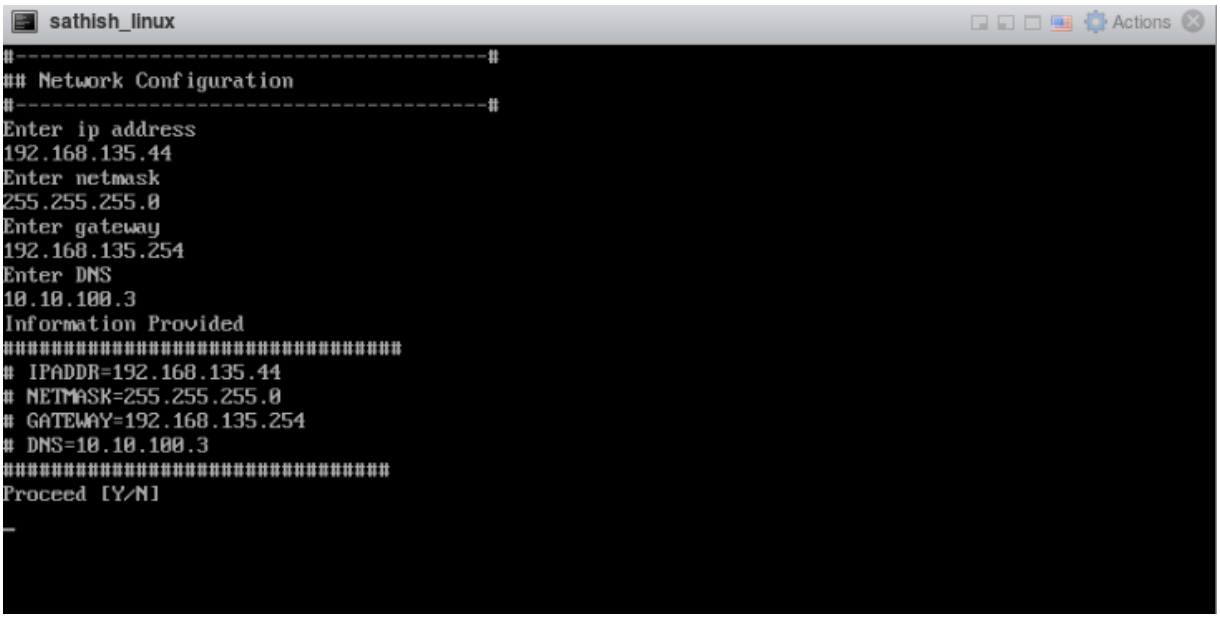
5. On the Deployment options screen, choose a network adapter in the *Network mappings* dropdown list and select the *Disk provisioning* as **Thin**. Click **Next**.



6. On the Ready to Complete screen click **Finish** to complete the OVA deployment process.

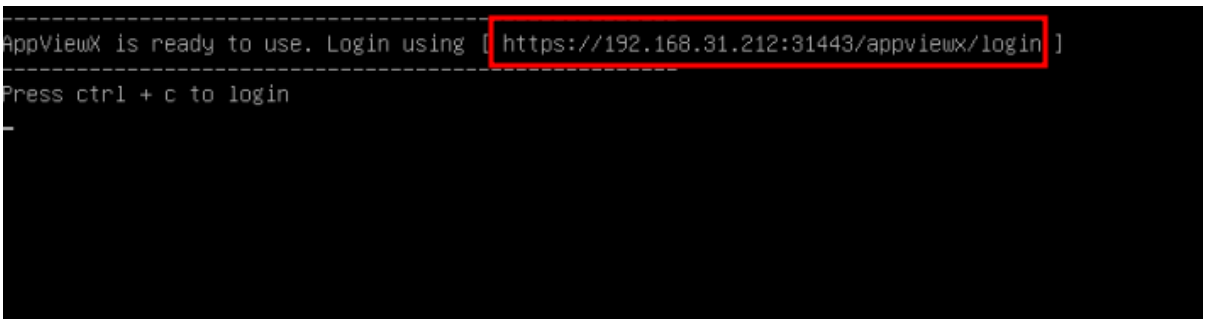


When the deployment wizard finishes, the AppViewX user interactive provisioning console opens within the ESXI server console section. You can use this console to set up your basic network configuration.



- Type **Y** on the console screen to proceed with the network configuration.

7. After the basic network configuration process finishes, the installation starts automatically. Once the installation process completes, you can access the application by opening the browser on the host machine and entering: https://<**ip:31443/appviewx/login**>.



Installing AppViewX

This section covers the process to install AppViewX on Linux servers in a single node as well as a multi-node environment. Once AppViewX is installed, users can verify the installation, upload the license key and integrate third party libraries with AppViewX.

Note: If you do not have a deployment model defined yet, contact help@appviewx.com

Warning:

- It is critical that you execute the prerequisite tool before installing AppViewX.
- Before you start the installation, ensure that the node password does not contain special characters such as single quote (**'**), double quote (**"**), and back slash (****), ampersand (**&**), comma (**,**) semicolon (**;**) or a combination of special characters such as **%{** and **${**.
- Upgrading from earlier versions is not supported in v2021.1.0. A new install is the only option.

Performing a Single Node or Standalone Installation

Prior to performing the installation, ensure the prerequisites success result is received after running the prerequisite tools. For running the prerequisites tool, see section Running the Prerequisite Tool.

1. Copy all the downloaded packages to the server.

   Note: The AppViewX installation must start from the node that is selected for the primary MongoDB host. For example, the first node specified under the MONGODB_HOST property in the **appviewx.conf** file.

2. SSH to the server in which packages are copied.
3. Open the terminal.
4. To extract the contents of the **appviewx_kubernetes_2023.1.3.0.tar.gz** file, execute the following command:

```
tar -xvf appviewx_kubernetes_2023.1.3.0.tar.gz
```

5. To move the **appviewx_kubernetes_addons_2023.1.3.0.tar.gz** file to the **appviewx_kubernetes** folder, execute the following command:

```
mv appviewx_kubernetes_addons_2023.1.3.0.tar.gz appviewx_kubernetes/
```

> **Note:** Refer to the Configuring POD and Service IP CIDR section before proceeding with the install to change the IP addresses/range used for pods and services.

6. To navigate to the <InstallerLocation>/appviewx_kubernetes/scripts directory, execute the following command:

```
cd <InstallerLocation>/appviewx_kubernetes/scripts
```

> **Note:** If you have received the **appviewx.conf** file already from AppViewX support, you can skip steps 6 through 9. Copy the provided **appviewx.conf**file into InstallerLocation/appviewx_kubernetes/scripts/ and continue to Step 10.

7. To copy the **appviewx.conf.template** file to the **appviewx.conf** file, execute the following command:

```
cp appviewx.conf.template appviewx.conf
```

> **Note:** The entire installation process is driven by the values mentioned in the appviewx.conf file.

8. To open the **appviewx.conf** file, execute the following command:

```
vi appviewx.conf
```

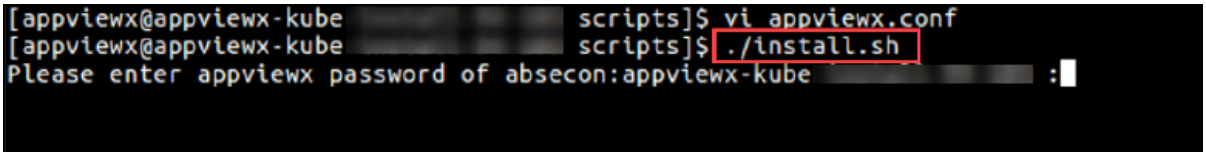
9. Enter the configuration values.

> **Note:** For more information, refer to the Configuring the appviewx.conf File to Install Appviewx section. Refer to the deployment diagram provided from help@appviewx.com or use the reference architecture provided by AppViewX

10. Save the changes to the file and exit the editor.
11. In the <InstallerLocation>/appviewx_kubernetes/scripts/ directory, execute the following command

```
/install.sh
```

12. Enter the user credentials for the respective nodes.



> **Note:** The installer location is the path where the installer file is extracted. After you enter the credentials, the installation process starts and takes about 15 to 20 minutes to complete.

After the AppViewX installation is complete, a success message is displayed on the command prompt with the Web and Gateway URLs.

> **Note:**
> - Take a backup of the below files and copy it to a secure location. Then, remove it from the installer location. The files are
>   - <installer location>/infra/.vault_key_for_reference
>   - <installer location>/.appviewx_configuration
> - Users can also find the AppViewX Web and Gateway URLs in the appviewx.conf file in the installation location.
> - Users can
>   - verify the installation by following the instructions provided in the section Verifying the Installation
>   - upload the license by referring to the instructions provided in the section
>   - For troubleshooting issues, please refer to the Troubleshooting section.

Performing a Multi-node or High Availability Installation

This section explains the procedure to install AppViewX in a multi-node environment. The installation procedure is identical to the single node installation with the only difference being the cluster configuration and the POD and Service IP CIDR configuration.

Prior to performing the installation, ensure the prerequisites success result is received after running the prerequisite tools. For running the prerequisites tool, see section Running the Prerequisite Tool.

Recommendations:

- MongoDB is CPU and disk intensive. Therefore, it is recommended to run MongoDB on a worker node.
- The hostnames or IP addresses present in the configuration should be a subset of `SSH_HOSTS`.

- The items in the `SSH` list and the `SSH_HOSTS` list should be in the same order. In other words, if the index of the IP address is 3 in the `SSH` list, it should also be 3 in the `SSH_HOSTS` list.
- It is recommended to assign a data center to a plugin once it is enabled.
- For production environments, a single node deployment is **NOT** recommended because:
  - Single node does not support log monitoring using Kibana and Grafana.
  - Unavailability of HA.
  - Syslog and statistics not available.

Configuring the appviewx.conf File to Install Appviewx

The installation of the application is driven by the **appviewx.conf** file available within the release package. For more information refer to the configuration file available in the following location:

<InstallerLocation>/appviewx_kubernetes/scripts/

The following parameters must be configured to install the application:

Table - AppViewX Conf File Parameters and its Description

Parameter	Description

MULTINODE	Specifies the boolean value to describe if the installation is in a single node/multi-node environment.
	
	Example:
	`MULTINODE=TRUE (For multi-node)`
	`MULTINODE=FALSE (For single node)`

SAAS_ENABLED	Specifies the flag to enable SAAS model deployment **Note:** This parameter is only for SaaS installations. Example: `SAAS_ENABLED=false`

Table - AppViewX Conf File Parameters and its Description (continued)

Parameter	Description
SAAS_DOMAIN	Specifies the domain name for SaaS installations **Note:** This parameter is only for SaaS installations. Example: `SAAS_DOMAIN=appvx.com`
VAULT_ENABLED	Specifies the flag to enable or disable the vault - for SAAS model deployment **Note:** This parameter is only for SaaS installations. Example: `VAULT_ENABLED=true`
PROVISIONING_ENABLED	Specifies the flag to be enabled for provisoning only the cluster **Note:** This parameter is only for SaaS installations. Example: `PROVISIONING_ENABLED=false`
TENANT_DEPLOYMENT_TYPE	Specifies the flag to set tenant_deployment_type. Expected values (any one of) - customer-prod, customer-non-prod, customer-additional, free-trial, poc-free, free-partner, free-training, internal-dev, internal-qa, internal-se, internal-training Example:

Table - AppViewX Conf File Parameters and its Description (continued)

Parameter	Description
	TENANT_DEPLOYMENT_TYPE=customer-prod
SSH	Specifies the comma (,) separated values of node IPs in which the application is set to be deployed. Example: SSH=192.168.XXX.XXX, 192.168.XXX.XXX, 192.168.XXX.XXX
SSH_HOST	Specifies the comma (,) separated values of node hostnames in which the application is set to be deployed. **Note:** Execute the command hostname in the node and add that output to this field. The hostname of a node must be the output of the command "hostname". Ensure to give the IPs provided in the SSH and host name provided in the SSH_HOST must be in the same order. Example: SSH_HOST=master:appviewx-kube-95.214.appviewx.net,master:appviewx-kube-95.215.appviewx.net,master:appviewx-kube-95.216.appviewx.net,dc1:appviewx-kube-95.217.appviewx.net,appviewx-kube-95.218.appviewx.net,appviewx-kube-95.219.appviewx.net **Note:** For the master nodes, the recommendation is to have the hostname as master:hostname. Ensure that the SSH_HOST and SSH are in the same order.
CLOUD_CONNECTOR_DC	Comma separated values of DC names which will communicate via cloud connector (avx_vendors, avx_vendor_cert_network_discovery)

Table - AppViewX Conf File Parameters and its Description (continued)

Parameter	Description
	Example: DC1, DC2 `CLOUD_CONNECTOR_DC=absecon`
INGRESS_HOST	To access AppViewX's Web UI, the `INGRESS_HOST` parameter must be configured. It can be configured with comma (,) separated values of Kubernetes worker node IP addresses where AppViewX needs to be accessed. **Note:** For single node AppViewX deployments, ensure that it is the IP address of the instance. To ensure high availability of the multiple DC deployments, it is recommended to add a minimum of one host per DC. Example: `INGRESS_HOST=192.168.XXX.XXX,192.168.XXX.XXX,192.168.XXX.XXX` **Note:** It is recommended to add the Kubernetes worker node IP addresses in this field. **Warning:** If the `INGRESS_HOST` parameter does not contain a host IP address, the AppVIewX UI will not be accessible.
INGRESS_LB_URL INGRESS_LB_PORT	In case the load balancer is used for ingress gateway service, provide the URL of the load balancer service and its port.
HSM_HOST	Comma separated values of node hostnames in which HSM pods will be scheduled **Note:** Execute the command "hostname" in the node and add that output to this field

Table - AppViewX Conf File Parameters and its Description (continued)

Parameter	Description
	• For single node AppViewX deployments add the IP address of the instance where AppViewX is installed. • To ensure high availability in multiple DC deployments, It is recommended to add a minimum of one host per DC. Example: `HSM_HOST=$(hostname)`
INSTALLATION_PATH	Specifies the path in which AppViewX is installed. Example: `INSTALLATION_PATH=/home/appviewx/appviewx/`
ENABLE_IPV6	Specifies whether IP v6 is enabled. Example: `ENABLE_IPV6=False`
MONITORING	Specifies whether monitoring is enabled or not. When you set the value to `TRUE`, set the value of the `PROMETHEUS_HOST` and `GRAFANA_HOST` to one of the worker node for multinodes. Example: `MONITORING=TRUE`
PROMETHEUS_HOST	Specifies the hostname or IP address of the Prometheus node.
GRAFANA_HOST	Specifies the hostname or IP address of the Grafana node.
LOKI_HOST	Specifies the hostname or IP address of the Loki node.

Table - AppViewX Conf File Parameters and its Description (continued)

Parameter	Description
ENABLED_PLUGINS	Specifies the list of plugins that needs to be enabled in the AppViewX installation. ENABLED_PLUGINS=appviewx_dependencies,avx_pkiaas_cert_ocsp_generator, avx_pkiaas_cert_ocsp_server,avx_commons,avx_crontab,avx_config_server, avx_platform_core,avx_platform_queue,avx_platform_gateway,avx_platform_web, avx_subsystems,avx_vendors,avx_subsystems_sync,avx_python_sandbox, avx_python_sandbox_sync,avx_platform_report_generator,avx_visual_page_builder, avx_platform_logforwarding,avx_vendor_cert_network_discovery,avx_platform_hsm, avx_ssh_server,avx_subsystem_codesigning
PLUGINS	Specifies the plugins to be installed in the datacenters. Example: avx_commons=absecon avx_config_server=absecon avx_platform_core=absecon avx_platform_queue=absecon avx_pkiaas_cert_ocsp_server=absecon avx_pkiaas_cert_ocsp_generator=absecon avx_platform_hsm=absecon avx_subsystems=absecon avx_subsystems_sync=absecon avx_python_sandbox=absecon avx_python_sandbox_sync=absecon avx_vendors=absecon avx_platform_gateway=absecon avx_platform_web=absecon avx_platform_report_generator=absecon avx_visual_page_builder=absecon avx_platform_logforwarding=absecon avx_vendor_cert_network_discovery=absecon avx_ssh_server=absecon avx_vendor_cert_cmp_agent=absecon avx_subsystem_codesigning=absecon

Table - AppViewX Conf File Parameters and its Description (continued)

Parameter	Description
SSH_OTHER_USER SSH_OTHER_GROUP SSH_PORT	Specifies the Linux user account with which AppViewX is installed. **Note:** AppViewX can be installed only as a Sudo user. Refer to the document Commands executed during AppViewX installation to get the details of commands that the Sudo user needs access to. Example: SSH_OTHER_USER=appviewx SSH_OTHER_GROUP=appviewx SSH_PORT=22
MONGODB_MIN_REPLICA	This parameter is used for enabling 2DC,3 Nodes Setup. A maximum 2 nodes needs to be added in MONGODB_HOST. It is mandatory to update ARBITER_HOST.
MONGODB_HOST	Specifies the comma (,) separated values of node hostnames in which the MongoDB is set to be deployed. This parameter is applicable only in a multi-node installation. **Note:** Add the output of hostname command in each node in this field. Do not add the output of hostname –f. A minimum of three nodes must be added. Example: MONGODB_HOST=appviewx-kube-95.217.appviewx.net, appviewx-kube-95.218.appviewx.net, appviewx-kube-95.219.appviewx.net

Table - AppViewX Conf File Parameters and its Description (continued)

Parameter	Description
	Note: A minimum of three nodes for MongoDB across three data centers are required to achieve HA at the data center level. It is recommended to run MongoDB only in the worker nodes.
ARBITER_HOST	This parameter is applicable only when AppViewX is deployed with two data centers. Arbiters are MongoDB instances that are part of a replica set but do not hold data. Arbiters participate in elections to break ties. Recommended to enable Arbiters only in AppViewX deployment with two data centers (DC) for high availability. In two DC environments, select the DC that has one Kubernetes master node, configure one of the Kubernetes worker nodes as an Arbiter node. **Note:** If AppViewX deployment is not in two DC environments, this parameter can be blank. Example: `ARBITER_HOST=192.168.XXX.XXX` **Note:** Do not add multiple IP addresses. Only one IP address is allowed.
VAULT_HOST	This parameter is valid only in multi-node installations. This parameter is comma (,) separated values of node hostnames in which the vault is set to be installed. **Note:** Add the output of `hostname` command in each node to this field. A minimum of three nodes must be added.

Table - AppViewX Conf File Parameters and its Description (continued)

Parameter	Description
	Example: `VAULT_HOST=appviewx-kube-95.217.appviewx.net, appviewx-kube-95.218.appviewx.net, appviewx-kube-95.219.appviewx.net` **Note:** A minimum of three nodes for a vault across three data centers is required to achieve HA at the data center level. It is recommended to run the vault only in the worker hosts.
MASTER_HOST	Specifies the hostname of the node which you want to run as a Kubernetes Master. The total number of masters can be 1, 3, 5, 7, and so on. For example, for a three-master installation, enter one node in the master host and the other two nodes in the secondary master_host. **Note:** Add the output of <hostname> command in this parameter. Example: `MASTER_HOST=appviewx-kube-install-94-179`
SECONDARY_MASTER_HOST	Specifies the list of nodes that are designated to run as secondary masters. The total number of masters can be 1, 3, 5, 7, and so on. For example, for a three-master installation, enter one node in the master host and the other two nodes in the secondary master_host. This parameter is applicable only in multi-node installations. **Note:** For deployments with a single master, comment out the SECONDARY_MASTER_HOST section. Example:

Table - AppViewX Conf File Parameters and its Description (continued)

Parameter	Description
	SECONDARY_MASTER_HOST=appviewx-kube-install-94-180, appviewx-kube-install-94-181
WORKER_HOST	Specifies the hostname of the list of Kubernetes worker nodes. **Note:** This parameter can be empty in a three-node setup. Example: WORKER_HOST=appviewx-kube-install-94-180, appviewx-kube-install-94-181 **Note:** Do not add the value given in the master_host in the worker_host. The worker and master nodes cannot be the same. This is again applicable only in multi-node installations.
TENANT_DB_S3BUCKET TENANT_MIGRATION_S3BUCKET CC_BINARY_S3BUCKET MONGO_S3BUCKET	The following parameters specify the S3 bucket to be mounted for Tenant DB, Tenant migration, CC Binary, Mogo-backup in a SaaS multitenancy provisioning cluster. **Note:** These parameters are only used for SaaS Installations.
EST_SERVER_ACCESS_CERT	Specifies the location for the digital enrollment certificate.
EST_SERVER_ACCESS_KEY	Specifies the location of the access key for the digital enrollment certificate.
EST_TRUSTED_CA_CERTS	Specifies the location of trusted certificate authorities for the EST server.
ELK	Specifies whether the `ELK` stash is enabled or not. You must specify a value for the `ELASTICSEARCH_HOST` parameter when you set `ELK` to `TRUE` for multinodes.

Table - AppViewX Conf File Parameters and its Description (continued)

Parameter	Description
	Example: `ELK=FALSE`
ELASTICSEARCH_HOST	Specifies the hostname or IP address of the elastic search host node.
PUBSUB_ENABLED=false PUBSUB_PROJECT_ID= PUBSUB_TOPIC= PUBSUB_JSON_PATH=	Specifies the flags to enable the Google Pub/Sub for SaaS model deployment. **Note:** These labels below will be considered when ELK is set to true
SPLUNK_HEC_ENABLED=false SPLUNK_HEC_HTTPS_ENABLED=false SPLUNK_HEC_URL= SPLUNK_HEC_CERT= SPLUNK_HEC_TOKEN=	Specifies the flags to enable the Splunk for SaaS model deployment. **Note:** These labels below will be considered when ELK is set to true
XSS_PROTECTION	This parameter is used to enable the XSS Sanitisation in the API Gateway, and avoids any XSS related exploits. Example: `XSS_PROTECTION=true`
API_ADDRESS	Specifies the hostname of the API server.
INSIGHT	Specifies whether the Insight module is enabled or not. Example: `INSIGHT=TRUE`
SYSLOG	Specifies whether the Syslog module is enabled or not. Example:

Table - AppViewX Conf File Parameters and its Description (continued)

Parameter	Description
	`SYSLOG=TRUE`
INSIGHT_ELASTICSEARCH_HOST	Specifies the hostname or IP address of the insight elastic search host node.
USER_GENERATED_PEM and PRIVATE_KEY_FILE_PATH	Set the value of the `USER_GENERATED_PEM` variable to `TRUE` if you want to perform a password-less installation. **Note:** • Update the value of the `PRIVATE_KEY_FILE_PATH` and set the value of the `USER_GENERATED_PEM` variable to `TRUE`. Otherwise, leave it empty. • Ensure that the value of the `PRIVATE_KEY_FILE_PATH` variable points to the private key file and not the directory. For example: `/tmp/user_generated_private.pem`.
REDIS_HOST	• The REDIS_HOST parameter is configured and applicable only in a multi-node setup. • Use only comma separated values of node hostnames in which the REDIS is to be deployed. **Note:** Add the output of the hostname command in each nodes to this field. Do not add output of hostname -f. • It is recommended to add only worker node(s) as the REDIS hosts but not the master hosts. • In case of two REDIS instances to be deployed on one node, add that node's hostname twice (e.g.: hostname1, hostname1, hostname2). • Add only two REDIS instances for a two-DC setup.

Table - AppViewX Conf File Parameters and its Description (continued)

Parameter	Description
	Example: `REDIS_HOST=$(hostname)`
SENTINEL_DC	• The parameter, SENTINEL_DC is only needed for a two-DC setups. • It is preferred to be in the secondary DC (i.e. DC with less kubernetes Master) • The REDIS Sentinel will spin up only on the DC mentioned in this parameter. Example: `SENTINEL_DC=absecon`
SYSLOG_LOGSTASH_HOST	Specifies the hostname of the node where the syslog logstash needs to be deployed. Enter only one hostname, as shown below. `SYSLOG_LOGSTASH_HOST=$(hostname)`
ENABLE_LOWER_TLS	Set ENABLE_LOWER_TLS=True to enable TLSv1.0, TLSv1.1 in the application to manage devices with lower TLS versions.

OPTIMISE_ROUTING_FOR_LATENCY
 	
PREFERRED_DEFAULT_DC
	This parameter is used mainly if the application is installed across multiple DCs and the latency between the DCs is high. The local routing between the pods can be enabled by setting OPTIMISE_ROUTING_FOR_LATENCY=True and specifying the preferred DC name in PREFERRED_DEFAULT_DC to increase the application performance.
	
	Example:
	`OPTIMISE_ROUTING_FOR_LATENCY=FALSE`
	`PREFERRED_DEFAULT_DC=absecon`

Table - AppViewX Conf File Parameters and its Description (continued)

Parameter	Description
MTU_VALUE	This option is used to change the MTU value for the calico during the appviewx installation. **Note:** This value should be changed before the application installation. Example: `MTU_VALUE=1350`

IPV4POOL_IPIP
 	
IPV4POOL_VXLAN
	This option is used to enable the IPinIP/VXLAN tunneling for calico.
	
	Note:
	• 'Always' should be for any one of the protocols (IPIP or VXLAN), it should not be added for both.
	• This value should be changed before the application installation.
	
	Example:
	`IPV4POOL_IPIP=Always`
	`IPV4POOL_VXLAN=Never`

SERVICE_SUBNET POD_SUBNET	This option is used to configure the pod and service default IP subnet ranges. **Note:**

Table - AppViewX Conf File Parameters and its Description (continued)

Parameter	Description
	• The IP range should not conflict with any of the internal IP ranges. • This value should be changed before the application installation. Example: SERVICE_SUBNET=10.96.0.0/12 POD_SUBNET=10.244.0.0/16
CALICO_PORT	This option is used to configure the default calico port. **Note:** This value should be changed before the application installation. Example: CALICO_PORT=179
SFTP_TRANSFER REMOTE_BACKUP_SERVER REMOTE_BACKUP_SERVER_SSH_PORT REMOTE_BACKUP_SERVER_USER MONGO_BACKUP_PATH VAULT_BACKUP_PATH	This option is used to configure the external SFTP Transfer for Mongo and Vault backup. It enables Passwordless communication between the remote backup server and the appviewx nodes. ***Pre-installation:*** Set SFTP_TRANSFER to true and configure the below listed variables ***Post-installation:*** Set SFTP_TRANSFER to true and configure the below listed variables and execute ./ sftp_transfer.sh script from the <appviewx-installer-location>/appviewx_kubernetes/scripts directory. The parameter description with examples is as follows: • *SFTP_TRANSFER=FALSE* – Enables SFTP transfer • *REMOTE_BACKUP_SERVER=* – Updates the SFTP server IP to store the vault and mongo backups

Table - AppViewX Conf File Parameters and its Description (continued)

Parameter	Description
	• *REMOTE_BACKUP_SERVER_SSH_PORT=22* – Updates the External SFTP server's SSH port in case of a custom SSH port • *REMOTE_BACKUP_SERVER_USER=appviewx* – Contains the username of the remote backup server • *MONGO_BACKUP_PATH=/home/appviewx/* – Updates the External SFTP location to store the mongodb backup • *VAULT_BACKUP_PATH=/home/appviewx/* – Updates the External SFTP location to store the vault backup

ENABLE_CUSTOM_CA_CERTS
 	
CERTIFICATE_PATHS
	This option is used to enable custom certs for outbound site communication. Enter the absolute path of the certificate to add to java truststore in the comma delimited format.
	
	Note:
	• It is recommended to use CA-signed certificates for better security.
	• If you still want to go ahead and add any internal CA's or Self-signed ones, do so at your own risk.
	
	Example:
	`ENABLE_CUSTOM_CA_CERTS=FALSE`
	`CERTIFICATE_PATHS=/home/appviewx/appviewx/ca-bundle.crt`

DB_MIGRATION_JOB_TIMEOUT	This parameter is used to configure the timeout (in minutes) for the DB Migration job. The default value is 60. If there is a higher volume of certs in the system and a migration/upgrade has to be carried out, then change this to a higher value, preferably 3x or 5x of the default value.

Table - AppViewX Conf File Parameters and its Description (continued)

Parameter	Description
	Example: `DB_MIGRATION_JOB_TIMEOUT=60`

ISTIO_SECRET_TT	The istio secret TTL value is used to extend the workload certificates. The secret TTL value should always be in minutes. Execute the utility command to the configuration:
	`./appviewx.sh --update-secret-ttl`
	Example:
	`ISTIO_SECRET_TTL=8640m`

ENFORCE_TLS_1_3	This flag to used to enable the enforcing of TLS 1.3. If the value is • True - enforce TSL1.3 • False - both TLS1.2 and TLS1.3 can be used
MONGODB_PORT	This parameter is used to configure the MongoDB port. By default the port number is 27017 used at the time of a fresh install. Customers with specific port requirements can use custom value by editing this parameter. For custom values the port range should be in between 10000 to 65535. Example: `MONGODB_PORT=27017`
API_ADDRESS_LISTNER_PORT	The custom port for Kube API server load balancer *Example:* `API_ADDRESS_LISTNER_PORT=6443`

Table - AppViewX Conf File Parameters and its Description (continued)

Parameter	Description
ENABLE_STRICT_ROUTING	This flag determines if the STRICT_ROUTING_DC is to be applied. If true, then the DC:plugin mentioned in the STRICT_ROUTING_DC parameter will be applied. *Example:* `ENABLE_STRICT_ROUTING=false`
STRICT_ROUTING_DC	This parameter contains the comma separated list of DC:plugins where strict routing is to be enabled. *Example:* If strict routing is to be enabled for avx_vendors in absecon DC, then set the value as mentioned below. `STRICT_ROUTING_DC=absecon:avx_vendors`
LOGMON_HOST	This parameter is used to deploy the logmon logstash onto a specific node. Provide a single hostname as the input which belongs to the worker node. `LOGMON_HOST=$(hostname)`
ELASTICSEARCH_BACKUP_HOST	This parameter is only applicable for multi-node and contains the comma separated hostnames of the nodes where you want to take the elasticsearch_insight_backup *Example:* `ELASTICSEARCH_BACKUP_HOST=xe-au-node99.lab.appviewx.net`
ELASTIC_BACKUP_PATH	This parameter contains the custom path in the AppViewX nodes where elastic backups are to be stored. *Example:* `ELASTIC_BACKUP_PATH=/home/appviewx/elastic_backup`
MSP_MODE	This a boolean flag (true/false) used to enable or disable the MSP mode. *Example:*

Table - AppViewX Conf File Parameters and its Description (continued)

Parameter	Description
	MSP_MODE=false
EXTERNAL_GATEWAY_HOST	This parameter contains the hostname of the node in which the external gateway is to be deployed. Enter one of the ingress host's hostname. *Example:* EXTERNAL_GATEWAY_HOST=$(hostname)
BACKUP_CRONJOB_SCHEDULE BACKUP_CRONJOB_RETENTION	This parameter is used to configure the backup cronjob frequency and retention. The BACKUP_CRONJOB_SCHEDULE used to specify the timing of a recurring schedule task. The value should be enclosed within double quotes; the value is entered in the format "<Minute> <Hour> <Day of Month> <Month> <Day of Week>". The below example states that cron job will run at 4:00 AM every day. BACKUP_CRONJOB_SCHEDULE="0 4 * * *" The BACKUP_CRONJOB_RETENTION variable determines how many backup files should be kept. The below example means that the system will keep the last 5 backups and delete any older ones. BACKUP_CRONJOB_RETENTION=5
SECONDARY_DB_BACKUP	This a boolean flag (true/false) used to configure if DB backup has to be taken from the secondary shared DB. *Example:* SECONDARY_DB_BACKUP=true

Configuring POD and Service IP CIDR

This section explains how to configure the number of POD/Service IP CIDRs that can run on a node. The Pods that run on a node are allocated an IP address from the node's Pod CIDR range.

Note: It is recommended to use the default settings for the POD and Service IP CIDR.

To configure POD/Service IP CIDRs:

1. Navigate to the <InstallerLocation>/appviewx_kubernetes/configs/kube/ directory.
2. To open the file, execute the following command:

```
vi kubeadm-config.yaml.tpl
```

```
-bash-4.2$ cd  /home/appviewx/appviewx_kubernetes/configs/kube/
-bash-4.2$ vi kubeadm-config.yaml.tpl
-bash-4.2$
```

Note: The service_subnet and pod_subnet can be configured in **appviewx.conf** file. During installation, the install script will read from there and update the **kubeadm-config.yaml.tpl** yaml file.

3. Under the networking section, check for serviceSubnet and change it as per requirements.`CIDR networking: serviceSubnet: <value> <change this default value to the desired CIDR>`
4. Under the networking section, check for podSubnet and change it as per requirements.`podSubnet: <value> <change this default value to the desired CIDR>`

```
apiVersion: kubeadm.k8s.io/v1beta2
kind: ClusterConfiguration
kubernetesVersion: v1.18.1
controlPlaneEndpoint: "${api_address}:6443"
networking:
  serviceSubnet: "[redacted]"
  podSubnet: "[redacted]"
  dnsDomain: "cluster.local"
apiServer:
  certSANs:
  - "${api_address}"
  extraArgs:
    service-account-signing-key-file: /etc/kubernetes/pki/sa.key
    service-account-key-file: /etc/kubernetes/pki/sa.pub
    service-account-issuer: api
    service-account-api-audiences: api,vault,factors
    authorization-mode: "Node,RBAC"
~
```

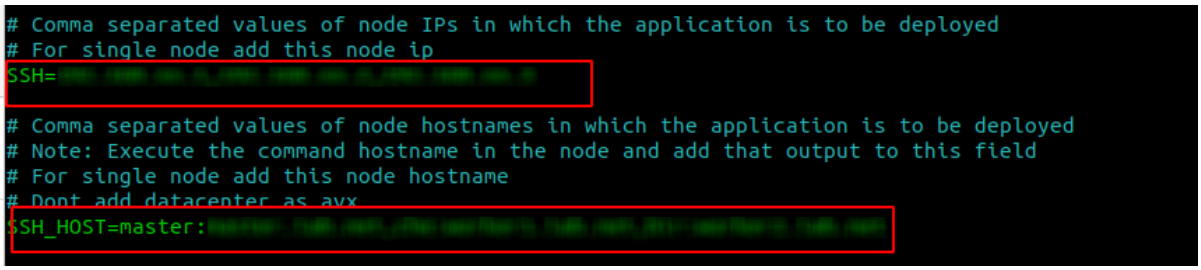
5. Save the changes and close the editor.
6. Once the above steps are complete, proceed with the AppViewX installation as mentioned in the Installing AppViewX section.

Note:

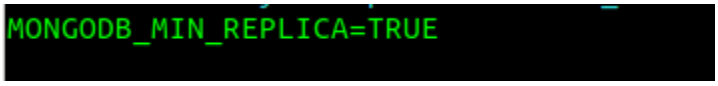
- After installation completes, take a backup of the below files and copy it to a secure location. Then, remove it from the installer location. The files are
  - <installer location>/infra/.vault_key_for_reference
  - <installer location>/.appviewx_configuration
- After the successful installation, you can access the .appviewx_configuration file by following the procedure given in the Accessing the Management Console section.
- Users can upload the license by referring to the instructions provided in the section Uploading the License Key.

Installation Support for 3 Nodes and 2 Datacenters

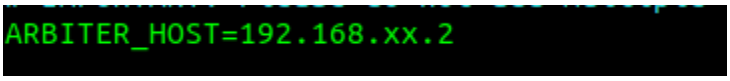
1. In the **appviewx.conf** file, set the value for the **Multinode** parameter as "**TRUE**".
2. Update the **SSH** and **SSH_HOST** parameters with the 1 master and min 2 workers as shown below.



3. Set the value of the **MONGODB_MIN_REPLICA** parameter as **TRUE**.



4. Ensure that you add a minimum of 2 hosts to the **MONGODB_HOST** parameter. It is mandatory to add any one of the IP addresses of the mongodb host to the **ARBITER_HOST** parameter.



5. Retain the **VAULT_HOST** parameter as is, because the system will automatically assign a vault host from each of the datacenters.
6. Update the **MASTER_HOST** and the **WORKER_HOST** parameters appropriately with the hostnames.
7. To navigate to the <installer location./appviewx_kubernetes/scripts

```
cd <installer location>/appviewx_kubernetes/scripts
```

8. To run the installation script, execute the following command:

```
./install.sh
```

Enabling the Load Balancer for the Kube API Server

Given below is an example configuration done on F5 devices and is needed only when we need to balance the load between multiple kube api servers in the case of multi DC support.

Prerequisite:

Create the TCP load balancer for Kube master apiserver.

> **Note:** This section is applicable only when the load balancer for the kube apiserver is not installed during the installation.

Sample Configuration:

Load balancer Configuration for Kube Master:

```
ltm virtual vs-appviewxmasterapi {
destination <IP Address>:sun-sr-https
ip-protocol tcp
mask XXX.XXX.XXX.XXX
pool pool-avxmasterapi
profiles {
fastL4 { }
}
serverssl-use-sni disabled
source 0.0.0.0/0
source-address-translation {
type automap
}
translate-address enabled
translate-port enabled
}
```

Pool Member Configuration for Kube Master

```
ltm pool pool-avxmasterapi {
 members {
 <Master Node IP Address>:sun-sr-https {
 address XXX.XXX.XXX.XXX
 session monitor-enabled
 state up
 }
 <Master Node IP Address>:sun-sr-https {
 address XXX.XXX.XXX.XXX
 session monitor-enabled
 state up
 }
 <Master Node IP Address>:sun-sr-https {
 address XXX.XXX.XXX.XXX
 session monitor-enabled
 state up
 }
 }
 monitor gateway_icmp
 }
```

To enable the load balancer for Kube Master:

1. To verify whether the load balancer is functioning normally, execute the following command:

```
curl -k https://loadbalancer-ip:6443/version
```

2. Apply the latest script patch from the release portal.

3. Navigate to the <installerLocation>/appviewx_kubernetes/scripts/ directory.
4. Open the **appviewx.conf** file.
5. Search for the **API_ADDRESS** parameter.
6. Modify the value of the **API_ADDRESS** parameter to reflect the IP Address or the FQDN of the load balancer.

```
#API ADDRESS - by default it will be MASTER IP; # If the cluster has a single master, use the IP
#of that master as the api_address. If the cluster has 3 masters, the api_address var needs
#to point to the IP of the load balancer
API_ADDRESS=tpsv-01.appvx.com
```

7. Navigate to the <installerLocation>/appviewx_kubernetes/scripts/loadbalancer/ directory.
8. To run the load balancer script, execute the following command:

```
./loadbalancer.sh
```

```
appviewx_loadbalancer.tf  loadbalancer.sh  sshkeyless  terraform.tfstate
-bash-4.2$ ./loadbalancer.sh
Please enter appviewx password of master:pesrv02-[redacted] lab.appviewx.net :
Please enter appviewx password of master:[redacted] et :
Please enter appviewx password of master:[redacted] 2-94-206 :
Please enter appviewx password of dc1:gs-[redacted] :
```

9. Enter the password of the nodes when prompted.
10. To verify the changes, execute the following command:

```
kubectl cluster-info
```

The output should contain the updated load balancer URL (IP Address or FQDN) of the kube API server.

Note: Once the LoadBalancer has been verified, proceed to delete the backup of the older control plane certificates using the following command:

```
rm -rf <actual_directory_path>/kubernetes_API_LB_backup_*
```

Replace `<actual_directory_path>` with the actual directory path.

Verifying the Installation

This section provides information on verifying whether the installation of AppViewX is successful. There are a few commands that will help you verify the installation. The commands are listed below.

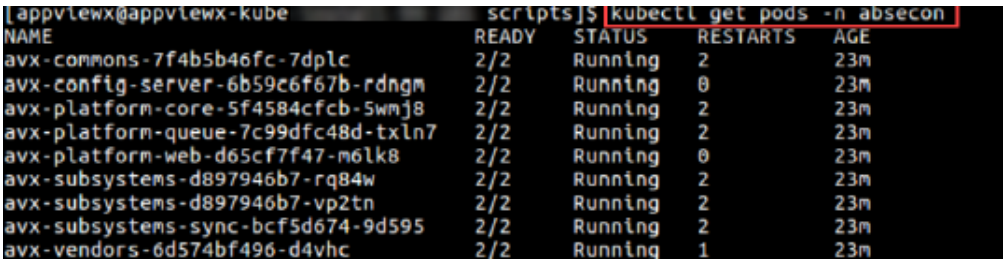
1. To check the status of the pods, execute the following command:

```
kubectl get pods -A
```

   If any of the pods show a different status, the application might not function as expected.

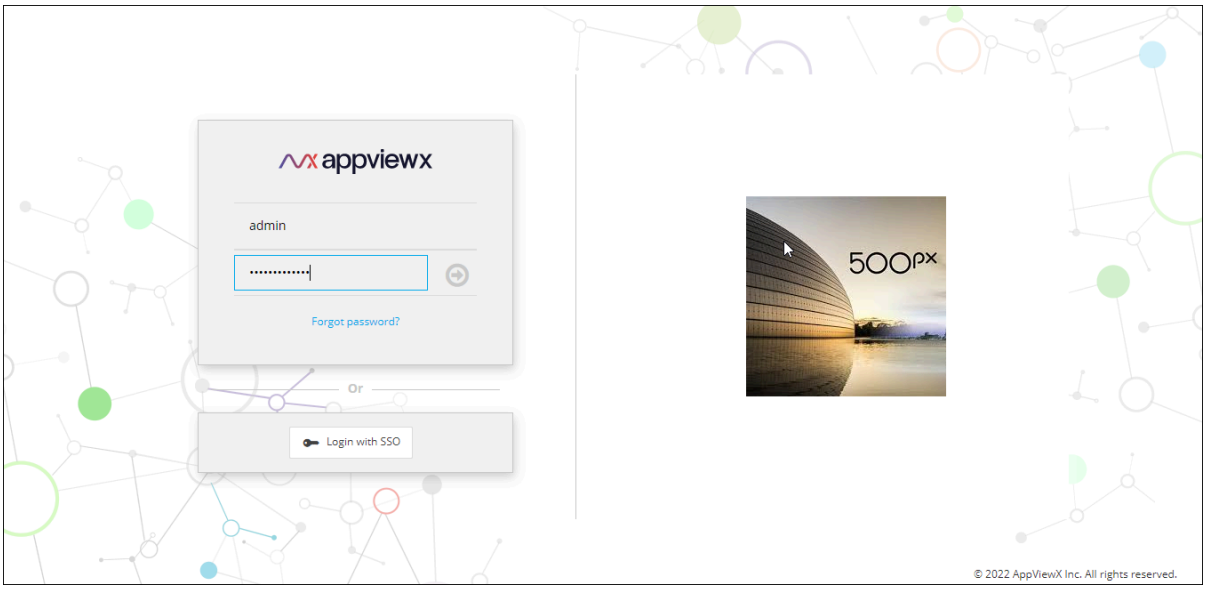
2. To restart the pod, execute the following command:

```
kubectl delete pods -n <dcname> <podname>
```



3. Access the GUI using the AppViewX Web URL with valid credentials. (AppViewX provides the default credentials).

> **Note:** Refer to the **appviewx_configuration** file, available for the URL. The file is available in the <InstallerLocation>/appviewx_kubernetes/scripts/ directory.



> **Note:** Multi-node installations come with a Redis cluster out-of-the-box. For single-node installations, there is a single Redis instance available that is enabled for PubSub only.

What to do next:

- After verifying the installation, take backups of the following files for future reference and housekeeping activities and delete them from the AppViewX nodes:
  - <INSTALLER_DIR_LOCATION>/scripts/../infra/.vault_key_for_reference
  - <INSTALLATION_DIR>/.appviewx_configuration
- For troubleshooting issues, refer to the Troubleshooting section.

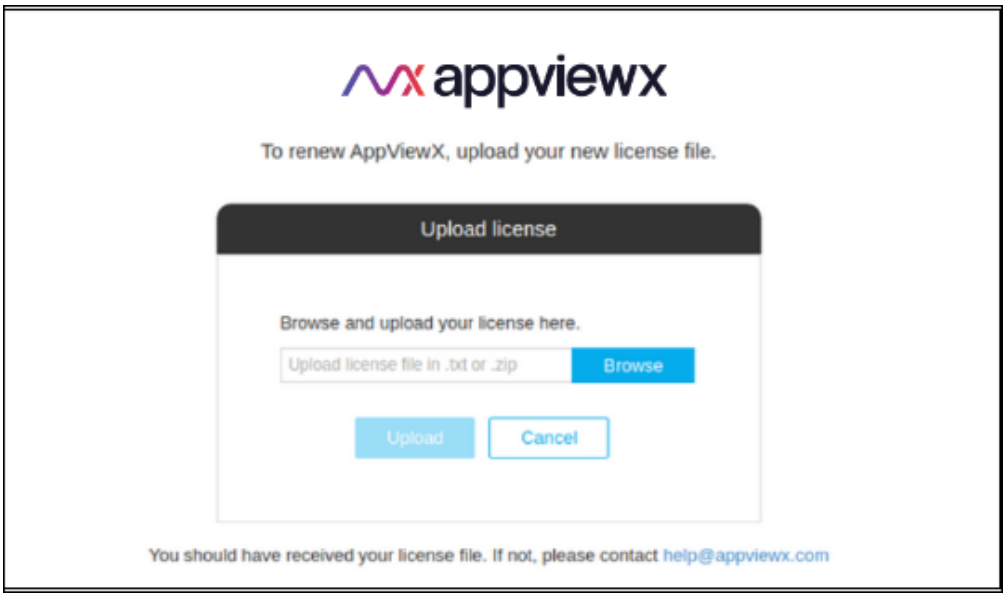
Uploading the License Key

License Management Software tracks software installed throughout the enterprise and ensures legal licenses for its usage. The software helps you to obtain the license key, upload the license key, and troubleshoot the license issues. License management is an essential element of software asset management (SAM).

To access the application, the user needs to upload a license. If you do not have a license key, send an email to help@appviewx.com with the hostname of the node in which the application is installed.

To upload the license key:

1. Log in to the application using the Web URL displayed in the success message of the installation. You are prompted to enter the username and password of the AppViewX admin account.
2. Click **Browse** and upload the license file.



A confirmation message is displayed after uploading a valid license.

Troubleshooting:

- If the license upload fails, ensure that the uploaded file is in the proper <.txt> (or) <.zip> format.
- If the license upload fails while activating the license, ensure that the output of the hostname command is provided during the generation of the license.
- If the license upload fails, trigger the following URL from your browser and try again after a few minutes. - https://AppViewX GATEWAY URL/refresh

Adding Third-party Libraries

AppViewX requires specified libraries to manage and control the devices. These libraries are specified by the manufacturers of the devices. AppViewX will be able to communicate with the devices only when these libraries are installed.

Please follow the steps in this section to add external proprietary jars in AppViewX.

- If the customer wants to use any third party integrations with earlier versions of AppViewX, ensure that the **.jar** files for these integrations are downloaded and extracted in the Installer/external_lib directory before the migration/installation process.
- If the customer wants to use any third party integrations with earlier versions of AppViewX after migration or installation, ensure that the corresponding **.jar** files are downloaded and extracted to the /home/appviewx/appviewx/external_libs directory.

- iControl F5 Integration
- Thales
- Safenet/Gemalto

iControl F5 Integration

iControl is an open API that enables applications to work in sync with the network based on the software integration. iControl uses SOAP/XML to ensure an open communication between dissimilar systems. It helps F5 customers, independent software vendors (ISVs), and solution providers leverage efficiency in automation and management of network objects and devices.

Users who want to use third party integrations to control their devices can integrate the required .jar file. The process begins with the user downloading the .jar file from the respective vendor. After downloading, the contents of the .jar file must be extracted into the external_libs directory. Finally, the plugin must be restarted for the changes to take effect.

1. To integrate the iControl library into the required project, copy the library and paste it into the <user_home_dir>/installer/external_libs/ directory (create a directory if it does not exist).
2. Visit devcentral f5 download page URL: iControl Library For Java With Source.
3. Download the latest iControl integration library file from the list of libraries.
4. Extract the downloaded zip file to: iControlAssembly_13_1_0-Java.
5. Download the axis.jar from the axis library.
6. Copy the **iControl.jar** and the **axis.jar** files from the extracted package to the **external_libs** directory.
7. If AppViewX is already installed or upgraded from an earlier version of AppViewX, move the **iControl-13.1.0.jar** and **axis.jar** file to external_libs directory using command below

```
cp -r /lib/iControl-13.1.0.jar <user_home_dir>/appviewx_dependencies/external_libs/ directory
```

```
cp -r axis-1.4.jar <user_home_dir>/appviewx_dependencies/external_libs/ directory
```

8. If AppViewX is not installed, move the **iControl-13.1.0.jar** and **axis.jar** file to external_libs directory using command

```
cp -r /lib/iControl-13.1.0.jar /home/appviewx/Installer/external_libs
```

```
cp -r axis-1.4.jar /home/appviewx/Installer/external_libs
```

9. In case of a multi node environment, copy the **iControl-13.1.0.jar** and **axis.jar** file to all the servers where the **avx_vendors** plugin is running.

> **Note:** To restart the avx_vendors plugin followed by the gateway plugin, refer to the Restarting a plugin section.

Thales

Users who want to use third party integrations to control their devices can integrate the required .jar file. The process begins with the user downloading the .jar file from the respective vendor. After downloading, the contents of the .jar file must be extracted into the **external_libs** directory. Finally, the plugin must be restarted for the changes to take effect.

> **Note:** Install the Thales client only on the node where AppViewX is installed.

1. To navigate to the directory where Thales client is installed, execute the following command:

```
cd /opt/nfast/java/classes
```

2. Copy the jutils, kmjava, and nfjava jars from the directory and paste it to the external libs directory in AppViewX.
   - If AppViewX is already installed /migrated, execute the following command:

     ```
     cp <jar_name>.jar <user_home_dir>/external_libs/
     ```

   - If AppViewX is not installed/migrated, to copy the jar in the installer directory, execute the following command:

     ```
     cp <jar_name.jar /home/appviewx/Installer/external_libs
     ```

3. Restart the **avx_vendors** plugin followed by the gateway plugin.

> **Note:** For more information on how to restart the plugin, refer to the Restarting a plugin section.

Safenet/Gemalto

Users who want to use third party integrations to control their devices can integrate the required .jar file. The process begins with the user downloading the .jar file from the respective vendor. After downloading, the contents of the .jar file must be extracted into the external_libs directory. Finally, the plugin must be restarted for the changes to take effect.

> **Note:** Install the Safenet client only on the node where AppViewX is installed.

1. To navigate to the directory where Safenet is installed, execute the following command:

   ```
   cd /usr/safenet/lunaclient/jcprov/lib
   ```

2. Copy the **jcprov jar** from the directory and paste it to the **external_lib** directory in AppViewX.
   - If AppViewX is already installed /migrated:

     ```
     cp jcprov.jar <user_home_dir>/appviewx_dependencies/external_libs/
     ```

   - If AppViewX is not installed/migrated, copy the jar in the installer directory:

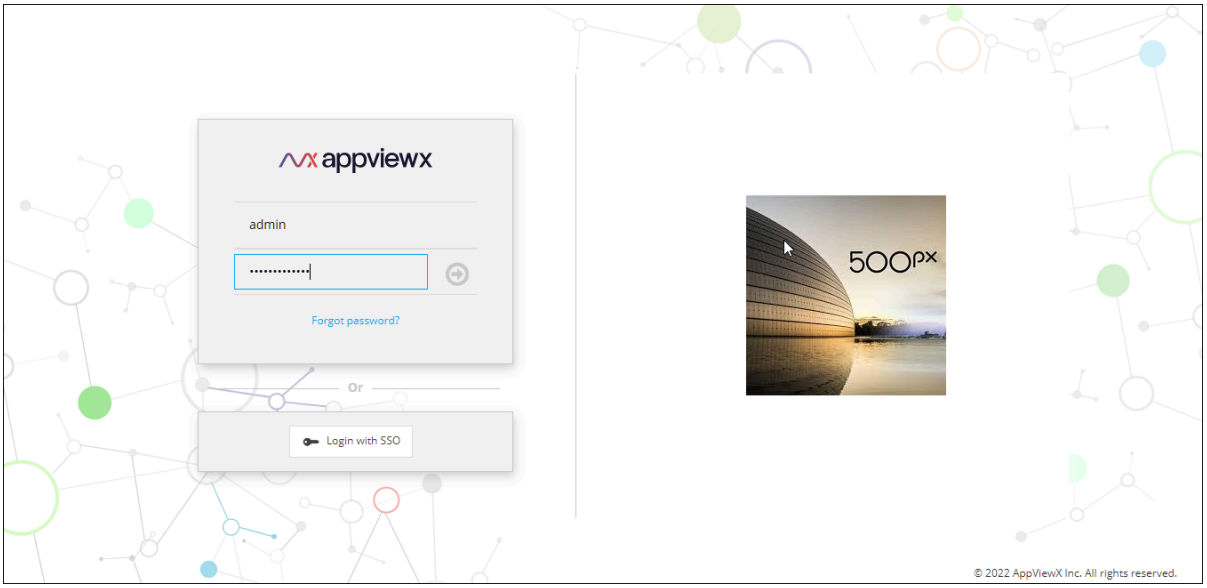
     ```
     cp jcprov.jar /home/appviewx/Installer/external_libs
     ```

3. Restart the **avx_vendors** plugin followed by the gateway plugin.

> **Note:** For more information on how to restart the plugin, refer to the Restarting a plugin section.

Accessing the AppViewX Graphical User Interface

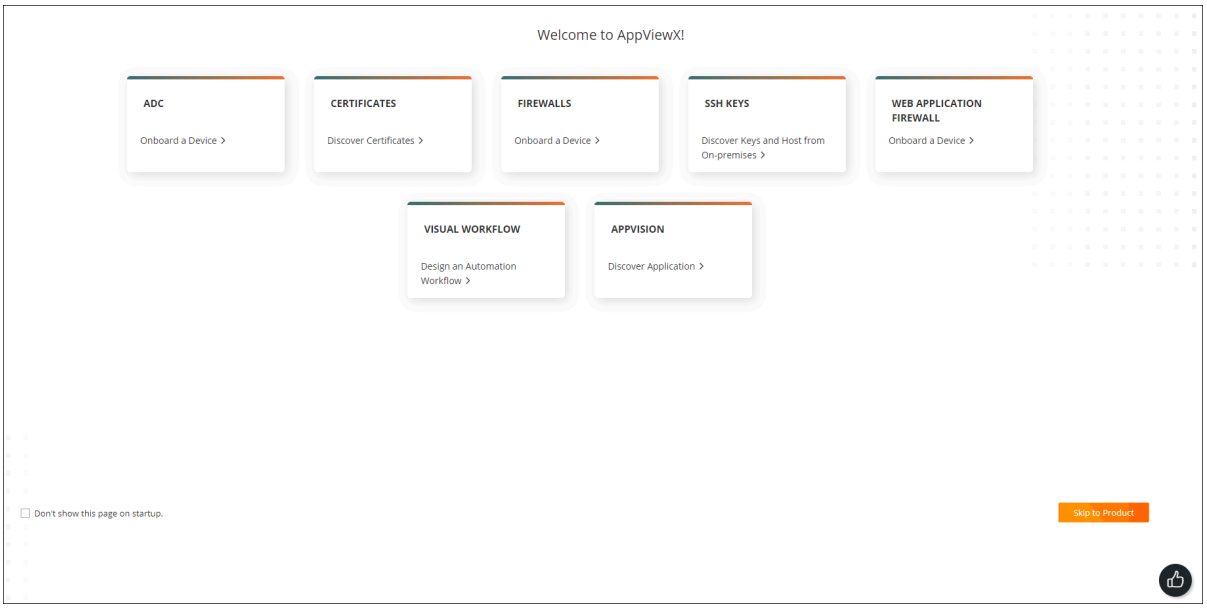
1. Access the graphical user interface (GUI) using the AppViewX Web URL with valid credentials.



Note: AppViewX provides default credentials to access the GUI.

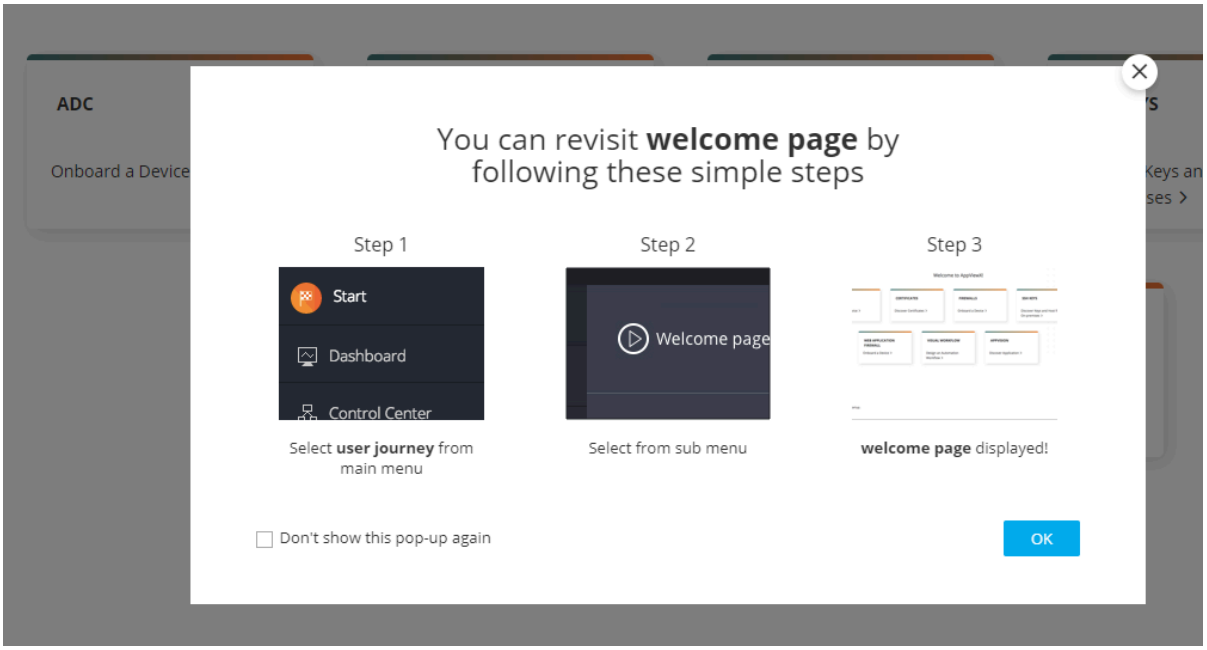
Note: Refer to the appviewx_configuration file, available for the URL. The file is available in the <InstallerLocation>/appviewx__kubernetes/scripts/ directory

Upon successful login, the **Welcome to AppViewX** page is displayed.



2. Click **Skip to Product**.

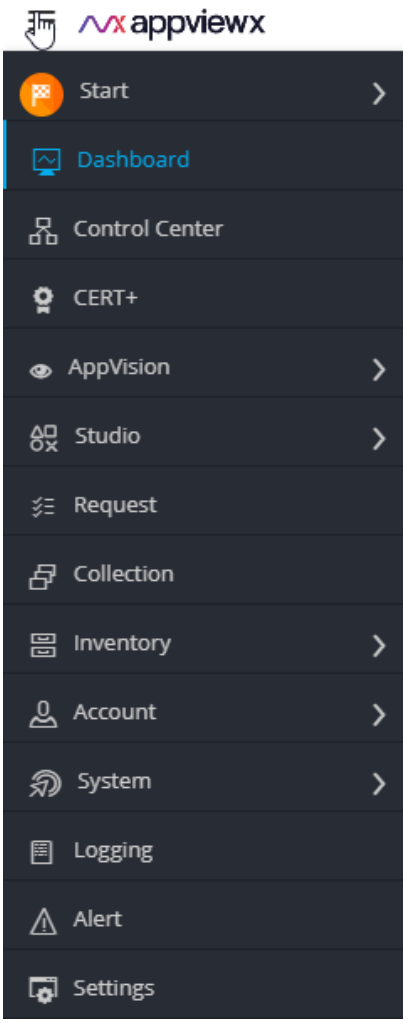
   The **Revisit Welcome** page is displayed.



3. Click **OK**.

   The system loads the dashboard page.

4. To access the different modules, click the icon to view the menu.



Using this menu, you can navigate to the different modules and access the features of the application.

Installing a Fix Pack

This section provides instructions for applying patches on AppViewX v2023.1.0.

Downloading the Patch

Before installing the fix pack, ensure that you have downloaded the patch plugins and addons from the release portal.

Installing the Patch

The process of installing the fix pack is executed by a script.

Note: For more information and detailed steps, please refer to the respective **Patch Deployment Guides** in the AppViewX On-Prem Setup Guides section.

Infra Readiness

The infra readiness must be performed before the application upgrade. Refer to the sections below.

- Automatic Diagnosis and Remediation Tool
- Backups and VMSnapshots

Automatic Diagnosis and Remediation Tool

Overview

The automatic remediation tool is used to check the health of the Kubernetes cluster after the installation, patch deployments or upgrades. You can choose to perform only diagnosis or a remediation along with the diagnosis of the various Kubernetes components and the services.

In diagnosis only the logs of the issues encountered are collected and saved in a file, while in remediation, the utility attempts to fix the issues encountered during the diagnostic process.

This tool can also be used before the patch and upgrade process to determine if users can proceed with the deployment process.

Diagnosis and Remediation Process

To initiate the remediation process

1. Navigate to the appviewx_kubernetes/scripts folder and execute the command below.

```
./appviewx.sh --remediation
```

You will be prompted to perform a diagnosis or diagnosis with remediation.

```
[Tue Feb 21 17:26:40 IST 2023 ~/fp10/appviewx_kubernetes/scripts]
[RPK-appviewx@192.168.94.96]$ ./appviewx.sh --remediation
Do you want to do only diagnostic or diagnostic with remediation?
Enter D for diagnostic and R for diagnostic with remediation - R
Enter password for appviewx@192.168.224.120:
Enter password for appviewx@192.168.224.113:
Enter password for appviewx@192.168.94.96:
```

2. Enter D for diagnosis only and R for diagnosis with remediation.

> **Note:** To check for infra readiness use only D.

3. Enter the password if required and continue (for passwordless applications no passwords will be asked).

```
[Tue Feb 21 17:25:17 IST 2023 ~/fp10/appviewx_kubernetes/logs/auto_remediation_log_files]
[RPK-appviewx@192.168.94.96]$ cat Auto_Remediation_Tool_Logs_Tue_Feb_21_17_22_25_IST_2023.txt
INFO:root:################### GETTING NODE PASSWORDS ###################
INFO:root:Password is correct for all the nodes
INFO:root:other_user_internal.pem is working for all the nodes
INFO:root:Time is in sync across the nodes.
INFO:root:Operation basic_cluster_checks succeeded.
INFO:root:################### CHECK FIREWALLD STATUS ###################
INFO:root:Firewalld is inactive in all the nodes.
INFO:root:Operation check_firewalld_status succeeded.
INFO:root:################### CHECK CONTAINERD STATUS ###################
INFO:root:Containerd is active in all the nodes.
INFO:root:Operation check_containerd_status succeeded.
INFO:root:################### CHECK KUBELET STATUS ###################
INFO:root:Kubelet is active in all the nodes.
INFO:root:Operation check_kubelet_status succeeded.
INFO:root:################### CHECK KUBECTL COMMAND ###################
INFO:root:Kubectl command is working in all the nodes.
INFO:root:Operation check_kubectl_command succeeded.
INFO:root:################### CHECK HDD STATUS ###################
DEBUG:Avx Commons:Following nodes have occupied more than 70% HDD space, kindly free some space          192.168.94.96
INFO:root:Operation check_hdd_space succeeded.
INFO:root:################### CHECK NAMESPACES ###################
INFO:root:Operation check_namespaces succeeded.
INFO:root:################### CHECK KUBE-SYSTEM PODS STATUS ###################
INFO:root:Pod: calico-kube-controllers-5477bbd996-8mv9r , Pod status: Running , Container status: 1/1
INFO:root:Pod: calico-node-8t8bf , Pod status: Running , Container status: 1/1
INFO:root:Pod: calico-node-kflx5 , Pod status: Running , Container status: 1/1
INFO:root:Pod: calico-node-w4hhw , Pod status: Running , Container status: 1/1
INFO:root:Pod: coredns-66bb66bbff-rgnl6 , Pod status: Running , Container status: 1/1
INFO:root:Pod: coredns-66bb66bbff-wvb5v , Pod status: Running , Container status: 1/1
INFO:root:Pod: etcd-pe-iu-node20.lab.appviewx.net , Pod status: Running , Container status: 1/1
INFO:root:Pod: kube-apiserver-pe-iu-node20.lab.appviewx.net , Pod status: Running , Container status: 1/1
INFO:root:Pod: kube-controller-manager-pe-iu-node20.lab.appviewx.net , Pod status: Running , Container status: 1/1
INFO:root:Pod: kube-metrics-adapter-75766454d7-jh8rv , Pod status: Running , Container status: 1/1
INFO:root:Pod: kube-proxy-579tg , Pod status: Running , Container status: 1/1
```

4. After the process is completed the log files are stored in the location $installer_directory/appviewx_kubernetes/logs/auto_remediation_log_files

Auto Remediation Tool Validations

The following validations are added in auto-remediation tool:

1. **Firewalld status check**

   The firewalld status is checked in all the nodes. By default it should be disabled, but if found in the running state in any node, then the script throws an error.

2. **Containerd status check**

   The containerd status is checked in all the nodes. By default it should be running, but if found in the not running state, then the script throws an error.

3. **Kubelet status check**

   The kubelet status is checked in all the nodes. By default it should be running, but if found in the not running state, then the script throws an error.

4. **Kubectl command check**

   The Kubectl command checked to see if they are working as expected in all the nodes.

5. **Hard disk space check**

   The hard disk usage is checked in all the nodes. If the hard disk usage is more than 70% in any node then the script throws an warning message to free some space.

6. **Namespace check**

   This check is used to find if the avx and dc namespaces are present in the cluster.

7. **Kube-system pod status check**

   The pods of the kube-system namespace are checked to see if the are in the running state.

8. **Mongodb pod status check**

   The mongodb pods are checked to see if the are in the running state.

9. **Istio-system pod status check**

   The pods of the Istio-system namespace are checked to see if the are in the running state.

10. **Config-server pod status check**

    The config-server pods are checked to see if the are in the running state.

11. **Consul server status check**

    The consul server pods are checked to see if the are up; if they are up it then checks if the consul server leader is present.

12. **Active vault status check**

The active vault pods are checked to see if the are up; if they are up it then checks if the active vault is present.

13. **Ephemeral vault status check**

    The ephemeral vault pods are checked to see if the are up; if they are up it then checks if the active vault is present.

14. **DC namespace's pod status check**

    This checks the pod status of all DC namespaces to see if are up and running.

15. **Calico status check**

    It checks to see if the calico is working as expected.

16. **Istio proxy status check**

    It checks to see if the istio proxy is working as expected.

17. **SElinux status check**

    This checks to see if the SElinux status is as expected. It should be either permissive or disabled

18. **Proxy check**

    This checks for the proxy status. It should disabled by default.

19. **Plugin helm chart check**

    This checks if helm charts are present for the plugins which are added in the ENABLED_PLUGINS parameter of appviewx.conf file.

20. **Infra helm chart check**

    It checks if helm charts are present for the required infra components.

21. **Mongo URL check**

    It checks if the Mongo URLs are properly updated in *avx-common-config* config map of the avx and DC namespaces. If they are not then should be updated with the proper values.

22. **Vault URL check**

    It checks if the Vault URLs are properly updated in *avx-common-config* config map of the avx and DC namespaces.

23. **Database password check**

It checks if the database password is properly updated in *avx-common-config* config map of the avx and DC namespaces.

24. **Super User password check**

    It checks if the super user password is properly updated in *avx-common-config* config map of the avx and DC namespaces.

25. **Collect TCPDUMP logs**

    It collects the TCPDUMP logs for all the servers in the cluster.

26. **Checking Registered VMs in gateway config**

    It checks if the VMs (Pod URL) are accurate in the Registered VMs parameter of the Gateway config map.

The following checks can be automatically remediated with the command

```
appviewx.sh --remediation -R
```

- Firewalld status check
- Containerd status check
- Kubelet status check
- Kubectl command check
- Mongo URL check

Backups and VMSnapshots

1. To take backup of mongo and vault, navigate to the installer node trigger appviewx_kubernets/scripts and execute the commands below.

   ```
   /bin/bash mongo_backup.sh
   ```

   ```
   /bin/bash vault_backup.sh
   ```

   Refer to Backup MongoDB and Vault for more details

2. Securely copy files tar files to an external server using the command below.

   ```
   scp <backups-tar-location> <remote-username>@<remote-hostname>:/<remote-location>
   ```

3. Navigate to the installer node trigger appviewx_kubernets/scripts and stop all the services using the command below.

```
./appviewx.sh --stop -all
```

4. Take VM snapshots
5. Start all the services using the command below.

```
./appviewx.sh --start -all
```

Upgrading to 2023.1.0FP3

- You can now upgrade to AppViewX 2023.1.0 FP3 from any of the legacy applications or older versions, mainly the
  - 2020.3.0 FP10-FP11
  - 2021.1.0 FP3
  - 2022.1.0 FP3
  - 2023.1.0 and FP1-FP2

  Refer to the Application Upgrade Guide.

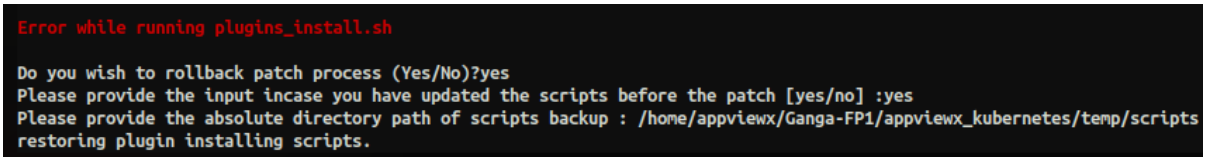
> **Note:** Execute the Prerequisite tool before performing any upgrade.

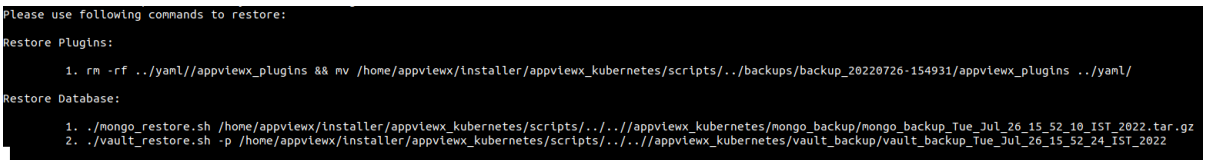
Rollback

In case of any failure during the patch deployment, an rollback can be initiated by the two following ways.

Instant Rollback

The instant rollback is performed at the very instance the patch fails. Executing the commands shown in the image below.





Anytime Rollback

The anytime rollback functionality is meant to be used in case of patch failure only after the entire patching process. It is integrated within the appviewx utility. Please find the steps to implement the functionality.

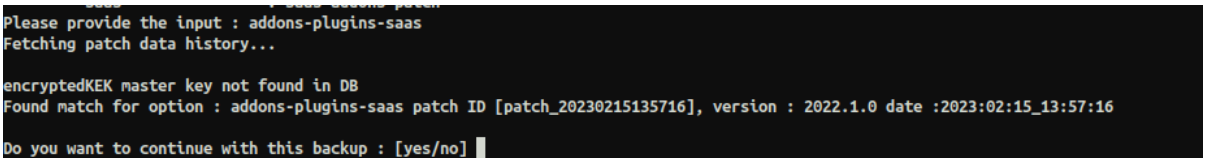
1. To initiate the rollback, execute the command.

```
./appviewx.sh --rollback
```

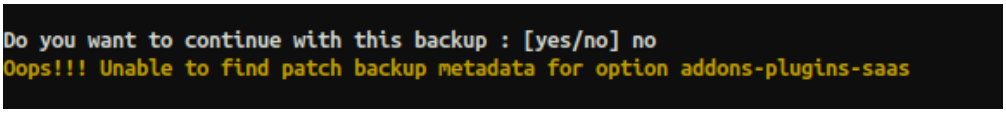
2. Once the rollback command is triggered you will be prompted for the type of backup you want to restore.

> **Note:**
> - The application maintains a backup based on the previous patch installed
> - The type of patch and rollback are tightly coupled, hence the backup data will be looked at based on the input provided

3. Once the application locates the backup file, you will be prompted to confirm if it is the exact backup you want to use.



4. The application will search for the most recent backup on the basis of the input provided. If the input value is “No” it searches for the next recent backup of the patch type.



Monitoring and Maintaining AppViewX

A system monitor is a component that is used to gauge resources and performance. It enables users to gather data and manage the health of the system. Although monitoring does not fix issues, it ensures that the system is stable and reliable.

In the case of AppViewX, during installation, the ELK stack is installed. This stack is a combination of three open-source products; Elasticsearch, Logstash, and Kibana. These are used to analyse the log files and ensure a stable system performance.

Installing ELK Components

Elasticsearch, Logstash, and Kibana (ELK) provide centralized logging to identify problems in servers or applications. It allows you to search all the logs and find issues that occur in multiple servers by connecting their logs during a specific time frame. To enable log management of the present and historical CLI logs of AppViewX from a GUI, an ELK-based utility is utilized.

To install ELK:

1. Download the **appviewx_kubernetes_elk_23.1.3.0.tar.gz** file.
2. To extract the contents of the file, execute the following command:

```
tar -xvf appviewx_kubernetes_elk_23.1.3.0.tar.gz
```

3. Move the contents of the extracted file to <InstallerLocation>/appviewx_kubernetes/yaml/appviewx_monitoring directory using the command,

```
cp -r appviewx_monitoring/* <InstallerLocation>/appviewx_kubernetes/yaml/appviewx_monitoring
```

4. Edit the **appviewx.conf** file.
5. Update the following parameters with their corresponding values:

a. **ELK=TRUE**

b. /appviewx_kubernetes/yaml/appviewx_monitoring directory

6. Navigate to <InstallerLocation>/appviewx_kubernetes/scripts/ directory.
7. To run the script, execute the following command:

```
./elk_install.sh
```

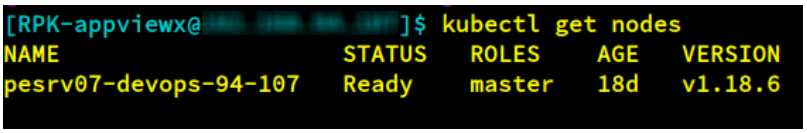
You can find the URLs to access the ELK stack in the command prompt after the installation is complete.

Executing Commands for Maintenance

AppViewX is installed based on the Kubernetes engine. We can use a few of the basic Kubernetes commands to manage the components in AppViewX. The list below contains the basic commands to manage the Kubernetes cluster.

1. View all the nodes in the cluster

```
kubectl get nodes
```



2. View all the pods of AppViewX services

```
kubectl get pods -n avx -o wide
```

```
[RPK-appviewx@ ]$ kubectl get pods -n avx -o wide
NAME                                    READY   STATUS              RESTARTS   AGE
IP              NODE                    NOMINATED NODE   READINESS GATES
avx-platform-gateway-586cdccd79-s7fl9   0/2     Pending             0          14d
<none>          <none>                  <none>           <none>
avx-platform-gateway-586cdccd79-xlxcd   1/2     Terminating         793        18d
<none>          pesrv07-devops-94-107   <none>           <none>
avx-platform-web-5c4595b87-cd2ml        2/2     Running             0          12d
                pesrv07-devops-94-107   <none>           <none>
mongo-configdb-0                        2/2     Running             0          12d
                pesrv07-devops-94-107   <none>           <none>
mongo-configdb-1                        2/2     Running             0          12d
                pesrv07-devops-94-107   <none>           <none>
mongo-configdb-2                        2/2     Running             0          12d
                pesrv07-devops-94-107   <none>           <none>
```

3. View all the services

```
kubectl get services -n avx
```

```
[RPK-appviewx@ ]$ kubectl get services -n avx
NAME                    TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)              AGE
avx-platform-gateway    ClusterIP                    <none>        5300/TCP             18d
avx-platform-web        ClusterIP                    <none>        5004/TCP,5555/TCP    18d
mongo-configdb-service  ClusterIP                    <none>        27017/TCP            18d
mongo-routerdb-service  ClusterIP                    <none>        27017/TCP            18d
mongo-shareddb-service  ClusterIP                    <none>        27017/TCP            18d
vault                   ClusterIP                    <none>        8200/TCP,8201/TCP    18d
vault-internal          ClusterIP                    <none>        8200/TCP,8201/TCP    18d
```

4. Log in to a particular container of the pod

```
kubectl exec -it avx-platform-web-5c4595b87-cd2ml -n avx /bin/sh
```

```
[RPK-appviewx@ ]$ kubectl exec -it avx-platform-web-5c4595b87-cd2ml -n avx -- /bin/sh
Defaulting container name to avx-platform-web.
Use 'kubectl describe pod/avx-platform-web-5c4595b87-cd2ml -n avx' to see all of the containers in this pod.
sh-4.2#
sh-4.2#
```

5. List all the namespaces

```
kubectl get namespaces
```

```
[RPK-appviewx@ ]$ kubectl get namespaces
NAME                   STATUS   AGE
absecon                Active   18d
avx                    Active   18d
avx-jobs               Active   18d
default                Active   18d
external-system        Active   18d
istio-operator         Active   18d
istio-system           Active   18d
kube-node-lease        Active   18d
kube-public            Active   18d
kube-system            Active   18d
kubernetes-dashboard   Active   18d
lens-metrics           Active   16d
```

6. List all the configuration maps. This is used to view configuration related details.

```
kubectl get configmaps -n avx
```

```
[RPK-appviewx@ ]$ kubectl get configmaps -n avx
NAME                           DATA   AGE
avx-common-config              6      18d
avx-platform-gateway-config    2      18d
avx-platform-web-config        1      18d
avx-vault-configmap            3      18d
istio-ca-root-cert             1      18d
vault-config                   1      18d
```

7. List all the deployments

```
kubectl get deployments -n avx
```

```
[RPK-appviewx@ ]$ kubectl get deployments -n avx
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
avx-platform-gateway  0/1     1            0           18d
avx-platform-web      1/1     1            1           18d
```

8. Stop a deployment

```
kubectl scale --replicas=0 deployment/avx-vendor-haproxy -n avx
```

```
[RPK-appviewx@ ]$ kubectl scale --replicas=0 deployment/avx-platform-gateway -n avx
deployment.apps/avx-platform-gateway scaled
```

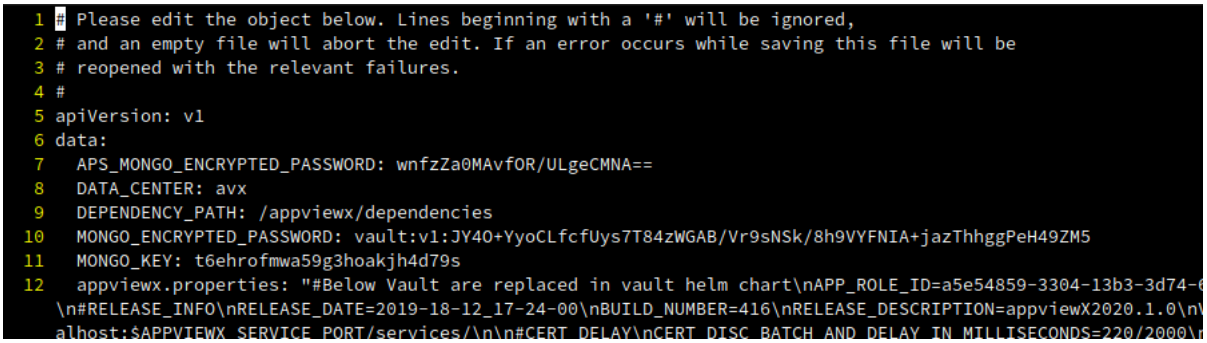
9. Start a deployment

```
kubectl scale --replicas=1 deployment/avx-vendor-haproxy -n avx
```

```
[RPK-appviewx@ ]$ kubectl scale --replicas=1 deployment/avx-platform-gateway -n avx
deployment.apps/avx-platform-gateway scaled
```

10. Edit a configuration

```
kubectl edit configmaps -n avx avx-common-config
```

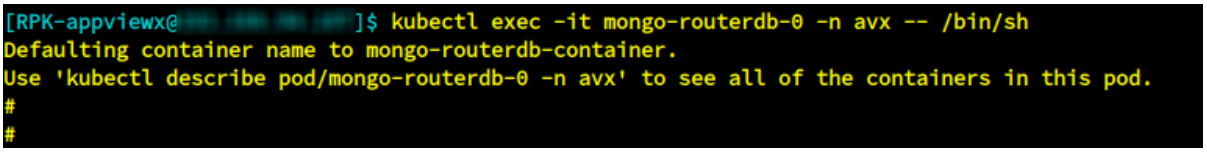


11. Describe the pods

```
kubectl describe pods -n avx <plugin name>
```

12. Log in to the database

```
kubectl exec -it mongo-routerdb-0 -n avx -- /bin/sh
```



Installing Trusted Certificate for GUI/API Access

The steps to install a trusted certificate for GUI/API access is follows:

1. To create a secret external-tls-credential of type tls, execute the following command:

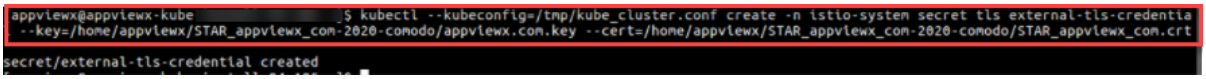
```
kubectl --kubeconfig=~/.kube/config create -n istio-system secret tls external-tls-credential --key=/etc/qualys/ssl/appviewx.com.key
 --cert=/etc/qualys/ssl/ssl-bundle.crt
```

For example:

```
kubectl --kubeconfig=~/.kube/config create -n istio-system secret tls external-tls-credential --key=/etc/qualys/ssl/appviewx.com.key
 --cert=/etc/qualys/ssl/ssl-bundle.crt
```

where:

- *~/.kube/config* should be present in each node
- *~/.kube* will be present in the home folder of the installing user

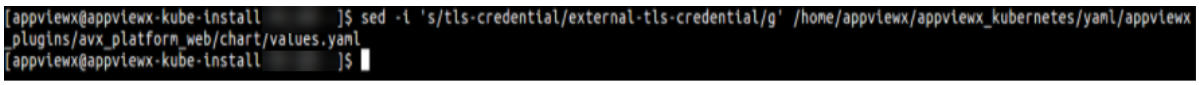


2. Replace secret name tls-credential with external-tls-credential in the `values.yaml` file.

> **Note:** The `values.yaml` file is available at installerLocation/appviewx_kubernetes/yaml/appviewx_plugins/avx_platform_web/chart/

- To replace, execute the following command:

```
sed -i 's/tls-credential/external-tls-credential/g' <installerLocation>/appviewx_kubernetes/yaml/appviewx_plugins/avx_platform_web/chart/values.yaml
```



3. Update the Gateway to consume the latest changes:

a. To navigate to the <installerLocation>/appviewx_kubernetes/yaml/appviewx_plugins/avx_platform_web directory, execute the following command:

```
cd <installerLocation>/appviewx_kubernetes/yaml/appviewx_plugins/avx_platform_web
```

b. To upgrade the avx-platform-web package to reflect changes, execute the following command:

```
helm upgrade avx-platform-web ./chart
```

```
[appviewx@appviewx-kube-install                                    ]$ helm upgrade avx-platform-web ./chart
Release "avx-platform-web" has been upgraded. Happy Helming!
NAME: avx-platform-web
LAST DEPLOYED: Wed Sep  9 10:49:09 2020
NAMESPACE: default
STATUS: deployed
REVISION: 2
TEST SUITE: None
[appviewx@appviewx-kube-install                                    ]$
```

4. Verify the application URL to check SSL is enabled.
5. Verify the certificate by launching the Appviewx portal.

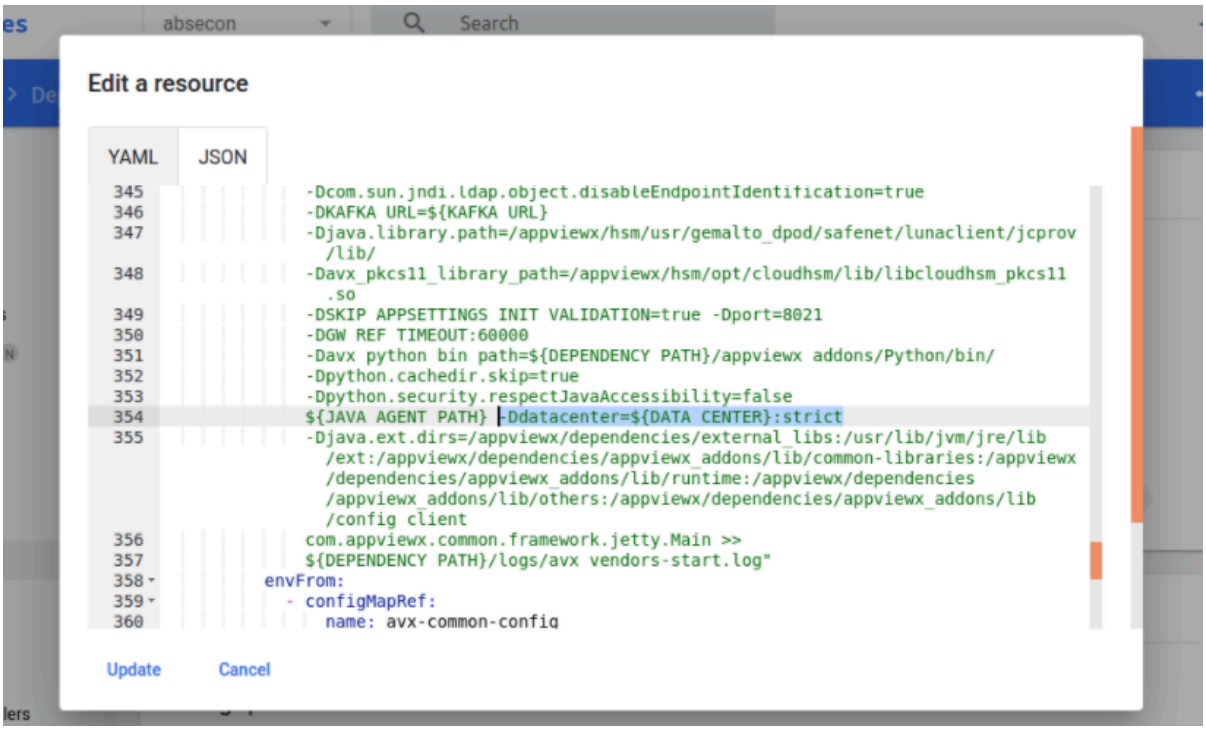
   The URL is `https://<Service URL>:Port>/appviewx`

Enabling Strict Data Center Routing

Strict data center routing is used to ensure that calls from AppViewX to a plugin/device through one data center are not routed to any other data center if there are no plugins available to serve traffic in the same data center.

To enable strict data center routing:

1. Log in to the Kubernetes dashboard of AppViewX.
2. On the left pane, under **Workloads**, click **Deployments**.
3. Search for the respective deployment to modify it.
4. Click **Edit**.
5. Add the argument **:strict** in **-Ddatacenter** jvm argument as shown below:



6. Click **Update**.

Enabling Device Syslog Processing

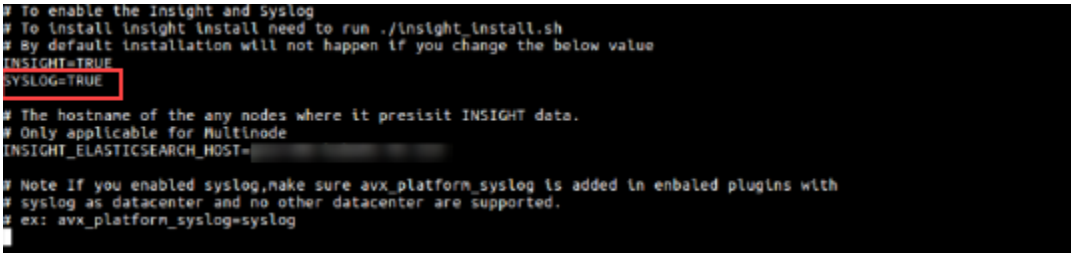
The Syslog module in AppViewX is used to receive syslogs from the device and update the necessary changes made in the device into the AppViewX database.

To enable Syslog parsing for the devices managed by AppViewX:

1. Navigate to the /home/appviewx/appviewx_kubernetes/scripts directory.
2. To open the **appviewx.conf** file, execute the following command:

```
vi appviewx.conf
```

3. Search for the SYSLOG parameter.
4. Set the value of the SYSLOG parameter to TRUE.



5. Search for **Enabled Plugins**.
6. Add the following plugins:
   - **appviewx_dependencies**
   - **avx_platform_syslog**
   - **avx_platform_gateway**

> **Note:** Gateway must be added to register the new APIs from the plugins that are installed.

7. Update the data center as **syslog** for the parameter **avx_platform_syslog** plugin.

8. Save and exit the **appviewx.conf** file.
9. From the /home/appviewx/appviewx_kubernetes/scripts directory, execute the following command:

```
./insight_install.sh
```

10. Execute the following command:

```
./plugins_install.sh
```

11. Execute the following command:

```
kubectl get services -n syslog
```

It displays the results as shown in the image below. Fetch the Syslog port from the service logstash-syslog-service. Here, the Syslog port is 30336.

This Syslog port changes for every installation/upgrade.

12. Connect to the MongoDB and open the **avx_app_metadata** collections. Edit this file by searching the parameter SYSLOG_RECEIVER_ENABLED and set it to TRUE. Save the file and move out of the DB.
13. To configure Syslog as TRUE, execute the following command:

```
kubectl edit configmaps -n "data center name" Set SYSLOG_RECEIVER_ENABLED=True,SYSLOG_HOST=192.168.XXX.XXX (Node IP where Syslog is installed)),SYSLOG_PORT=30047 (fetch the ports from point 8)
```

14. Save and exit the **<configmaps>** file.
15. To get the Pod name, execute the following command:

```
kubectl get pods -n "data center name"
```

16. To restart subsystems and vendors, execute the following command:

```
kubectl delete pods “Pod name” -n ‘’data center name’’
```

For example: You may restart multiple pods and the config servers by entering the name of the pod and the config server in the command below with space.

```
kubectl delete pods avx-subsystems-7666cfb459-6q4rn avx-vendors-99c69cd69-jtr4w avx-config-server-85ff9dd46d-h5qnr-n ''data center name’’
```

Enabling the Insight Module

The Insight module allows you to collect statistics from the devices that are managed by AppViewX. Also, it displays historical statistics on demand for users.

To install Insight for statistics collection:

1. Open the terminal.
2. Navigate to the /home/appviewx/appviewx_kubernetes/yaml directory.
3. Download the **appviewx_kubernetes_insight_2023.1.0.tar.gz** file.
4. To extract the file, execute the following command:

```
tar -xvf appviewx_kubernetes_insight_2023.1.0.tar.gz
```

5. Navigate to the /home/appviewx/appviewx_kubernetes/scripts directory.
6. To open the **appviewx.conf** file in the editor mode, execute the following command:

```
vi appviewx.conf
```

7. Set the following parameters as follows:
   - INSIGHT = TRUE
   - ELASTICSEARCH_BACKUP_HOST = <node1-hostname>,<node2-hostname>
   - ELASTIC_BACKUP_PATH = <path to store backup e.g.: /home/appviewx/elastic_backup>
8. Search for **Enabled Plugins** and add the following plugins:
   - **appviewx_dependencies**
   - **avx_insight_subsystem_adc**
   - **avx_insight_statistics_bot**
   - **avx_platform_gateway**
9. Update the data center for insight plugins as shown in the image below:

```
SSH_OTHER_USER=appviewx

avx_commons=dc1
avx_config_server=dc1
avx_platform_core=dc1
avx_platform_queue=dc1
avx_subsystems=dc1
avx_subsystems_sync=dc1
avx_vendors=dc1
avx_platform_gateway=dc1
avx_platform_web=dc1
avx_insight_subsystem_adc=dc1
avx_insight_statistics_bot=dc1
```

10. Save and exit the **appviewx.conf** file.
11. To install Insight, navigate to the /home/appviewx/appviewx_kubernetes/scripts directory.
12. Execute the following command:

```
./insight_install.sh
```

13. Execute the following command:

```
./plugins_install.sh
```

14. To restart subsystems and vendors, execute the following command:

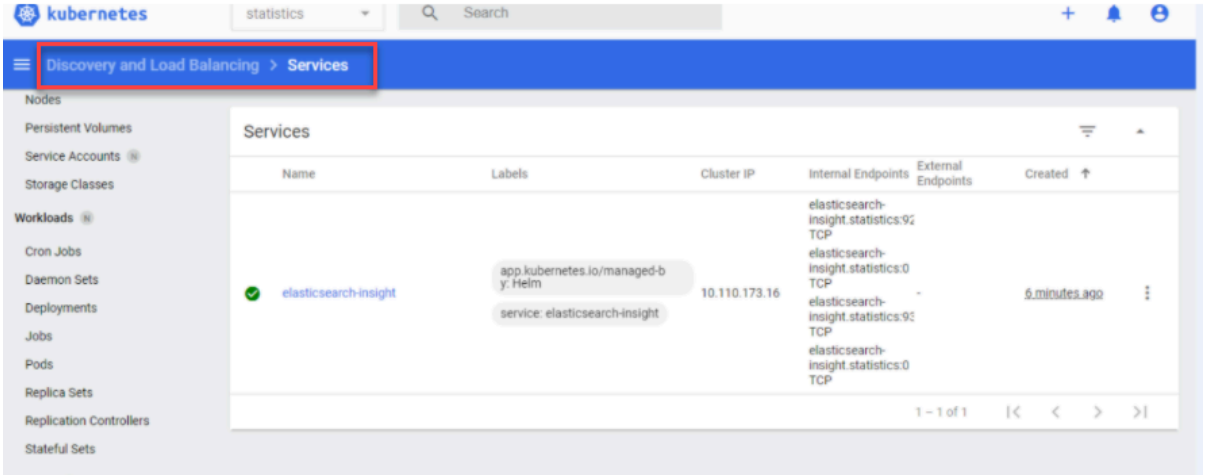
```
kubectl delete pods "Pod name" -n "datacenter name"
```

For example:

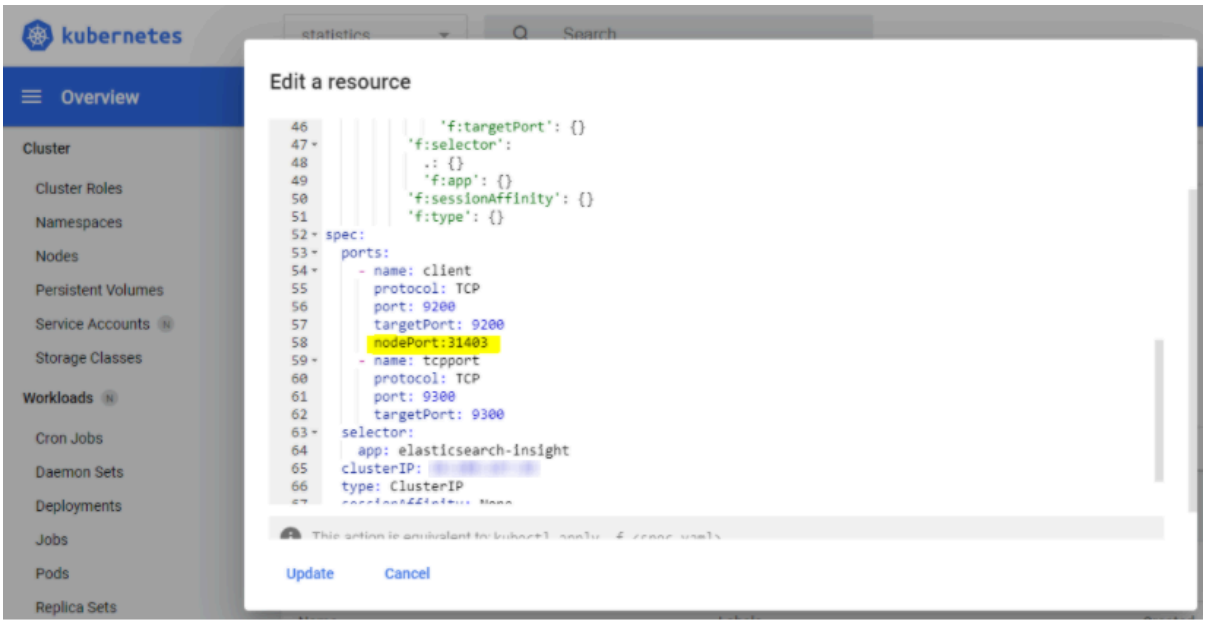
```
kubectl delete pods avx-insight-statsitics-bot-3499c69cd6-4sdfs,avx-insight-subsystem-adc-4399c69ed6-4sdfs,avx-subsystems-7666cfb459-6q4rn -n
absecon
```

To restart multiple Pods, enter the name of the pod in the above command with space.

15. In the case of Insight migration, continue till point 11.
16. Log in to the Kubernetes dashboard, enter statistics as the namespace.
17. Select services and search for **elasticsearch-insight**.

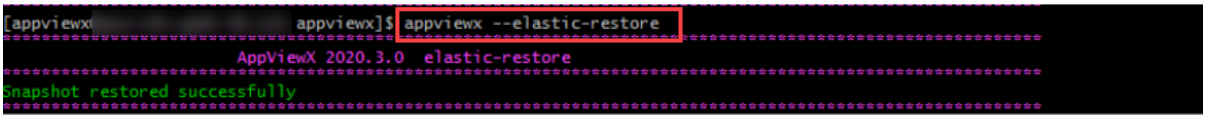


18. On the Pod, click **Edit**.
19. Enter the port details as **nodePort: 31403** and save as shown in the image below:



20. To restore the elastic data, go to the old installation path of AppViewX, and execute the following command:

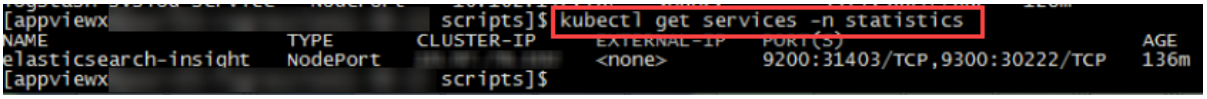
```
appviewx --elastic-restore
```



21. To connect to the elastic database, execute the following command:

```
kubectl get services -n statistics
```

It displays the results as shown in the image below:



Understanding Commands Executed during Installation

The section lists the commands executed by the AppViewX installer that requires Sudo access.

To restrict the commands that a Sudo user has access to, enable the following commands:

- `sudo kubeadm`
- `sudo kubectl`
- `sudo yum remove`
- `sudo yum install`
- `sudo systemctl daemon-reload`
- `sudo rpm -ivh --force *.rpm`
- `sudo modprobe br_netfilter`
- `sudo swapoff -a`
- `sudo iptables -F`
- `sudo date --set`
- `sudo -S setenforce 0`
- `sudo -S sysctl -w net.bridge.bridge-nf-call-iptables=1`

Apart from the above commands, Sudo user must be able to read/write/execute in the following directories:

- `/etc/`
- `/root/`
- `/var/lib`
- `/tmp`
- `/usr/local/bin`
- `/home/SSH_OTHER_USER` (Other user is user-defined in /scripts/appviewx.conf)

Enabling Sudo Access

To enable Sudo access and grant access to all commands:

1. Log in as an Administrator.
2. Log in to the node with root credentials.

• Creating a New Sudo User

• Adding Users to the Sudo Group

• Verifying if the Wheel Group is Enabled

Creating a New Sudo User

To create a new Sudo user:

1. Open the terminal.
2. Execute the following command:

```
adduser <UserName>
```

> **Note:** Replace the UserName with the new user's name.

3. To create a password for the new user, execute the following command:

```
passwd <Password>
```

The system prompts you to set and confirm a password for your new user account. If successful, the system responds with “all authentication tokens updated successfully.”

> **Note:** A strong secure password has more characters and a few special characters (such as numbers, symbols, or capitals). Ensure that you are choosing an appropriately strong password for your system.

Adding Users to the Sudo Group

For CentOS

By default, CentOS 7 has a user group called the Wheel group. Members of the wheel group are automatically granted Sudo privileges. Adding a user to this group grants Sudo privileges to the user.

To add a user to the wheel group, refer to Adding a User to the Wheel Group.

For Ubuntu

In Ubuntu OS, a user in “sudo” group is granted sudo permissions.

To add a user to the sudo group:

Execute the following command:

```
usermod –aG scudo UserName
```

Note: Replace the UserName with the new user's name to grant Sudo privileges.

Verifying if the Wheel Group is Enabled

To verify whether CentOS 7 installation has the wheel group enabled or disabled:

1. To open the configuration file, execute the following command:

```
vi sudo
```

2. Search for the following entry in the configuration file:

```
## Allows people in group wheel to run all commands %wheel ALL=(ALL) AL
```

   If the second line begins with the # sign, it indicates that the line is marked as a comment and the feature is disabled.

3. Delete the # sign at the beginning of the second line as given below.

```
%wheel ALL=(ALL) ALL
```

4. Save the file and exit the editor.

Note: If there is no # sign at the beginning of the line, do not make any changes. The wheel group is already enabled.

Adding a User to the Wheel Group

Adding a user to the wheel group is applicable for CentOS.

To add a user to the wheel group:

Execute the following command:

```
usermod –aG wheel UserName
```

Note: Replace the UserName with the new user's name to grant Sudo privileges.

Switching to the Sudo User

To switch to the new (or newly-elevated) user account with the su (substitute user):

1. Execute the following command:

```
su - UserName
```

2. Enter the password if prompted.
3. To list the contents of the /root directory, execute the following command:

```
sudo ls -la /root
```

4. Enter the password if prompted.
   The terminal displays the list of directories. Since listing the contents of the /root directory requires Sudo privileges, this is an easy way to prove that the new user can use the Sudo command.

Understanding the Best Practices on Reboot Sequence

This section provides information on the best practices to be followed for rebooting the operating system after security patching.

Note: Before you perform these steps, ensure that all prerequisites are complied with as mentioed in the Configuring YUM section.

The steps are to be executed in the order given below.

1. Log in into the AppViewX worker node from where the installation has been initiated.
2. Navigate to <installer directory path>/appviewx_kubernetes/scripts
3. Take a backup of the scripts directory from /appviewx_kubernetes/scripts
4. Download the latest **scripts.tar.gz** from the release portal.
5. Copy the existing **appviewx.conf** file from the older scripts folder to the newly downloaded scripts folder from the release portal.
6. Execute the commands from the installer location/scripts folder.

   Note: The Stop all and Start all commands are applicable only for a multi node setup.

7. To drain all the pods, execute the following command:

```
./appviewx.sh --stop -all
```

   The command will drain the pods in the nodes in the following order; Worker, Secondary master(if any), Master.
8. Shut down the nodes in the order mentioned in step 7.

9. Start the nodes in the reverse order; Primary master, Secondary masters, and Workers from the primary mongo as per **appviewx.conf** entries.
10. To start all the pods in the nodes, execute the following command:

    ```
    ./appviewx.sh --start -all
    ```

    This command will start the pods in the nodes in the following order; Master, Secondary master(if any), Worker.

Adding a Node to the Cluster

This section explains the steps to be followed to add a new node to an existing cluster.

1. Login to the node from where the AppViewX deployment was triggered (installer node).
2. (Optional) Execute this step ONLY when the new node is cloned from an existing node. In this case, you will have to remove the existing Kubernetes configuration files. To delete the existing kubelet config files:

    a. Log in as a root user.

    b. Execute the following command:

    ```
    kubeadm reset -f
    ```

    c. To navigate to the /var/lib/kubelet directory, execute the following command:

    ```
    cd /var/lib/kubelet/
    ```

    d. To delete all the files inside the kubelet directory, execute the following command:

    ```
    rm -r *
    ```

    e. To navigate to the /etc/kubernetes directory, execute the following command:

    ```
    cd /etc/kubernetes/
    ```

    f. To delete all the files inside the Kubernetes directory, execute the following command:

    ```
    rm -r *
    ```

3. Login to the release portal.
4. Download the **new_node_addition.tar.gz** file from the portal.
5. To extract the contents of the file, execute the following command:

    ```
    tar -xvzf new_node_addition.tar.gz
    ```

The command extracts the contents to a directory named **new_node_addition**.

6. Navigate to the **new_node_addition** directory.

   The directory contains a package named
   - **appviewx_adding_node.tar.gz**
   - **node_addition.tar.gz**

   .
7. To extract the contents of the **appviewx_adding_node.tar.gz** file, execute the following command:

   ```
   tar -xvzf appviewx_adding_node.tar.gz
   ```

   The command extracts the contents to a directory named **appviewx_adding_node**.
8. Navigate to the **appviewx_adding_node** directory.
9. Copy the files the respective directories
   - To copy all the files and folders to the <installer_location>/appviewx_kubernetes/scripts directory, execute the following command:

     ```
     chart

     joining_steps.sh

     appviewx_add_node.sh

     derive_configmap.py

     cp -r * <installer_location>/appviewx_kubernetes/scripts/
     ```
   - Manually copy/move the file **node_addition.tar.gz** into the <installer_location>/appviewx_kubernetes folder.

Adding a Master Node

1. For the nodes that will be added using the node addition script, ensure the following prerequisites are met.

   a. Execute/run the prerequisite tool and ensure it meets the tool execution criteria.

   b. Check if the IP forwarding is enabled and firewalld is disabled. If they are not then execute steps c and d.

   c. Enable ip_forward, using the steps below:

      i. Execute the command

      ```
      sudo vi  /etc/sysctl.conf
      ```

ii. In the conf file add the value `net.ipv4.ip_forward=1`

iii. Execute the command

```
sudo sysctl -p
```

d. To disable the firewalld, use the command below.

```
sudo systemctl stop firewalld
```

2. To add the master node, execute the respective command:

• Run the command:

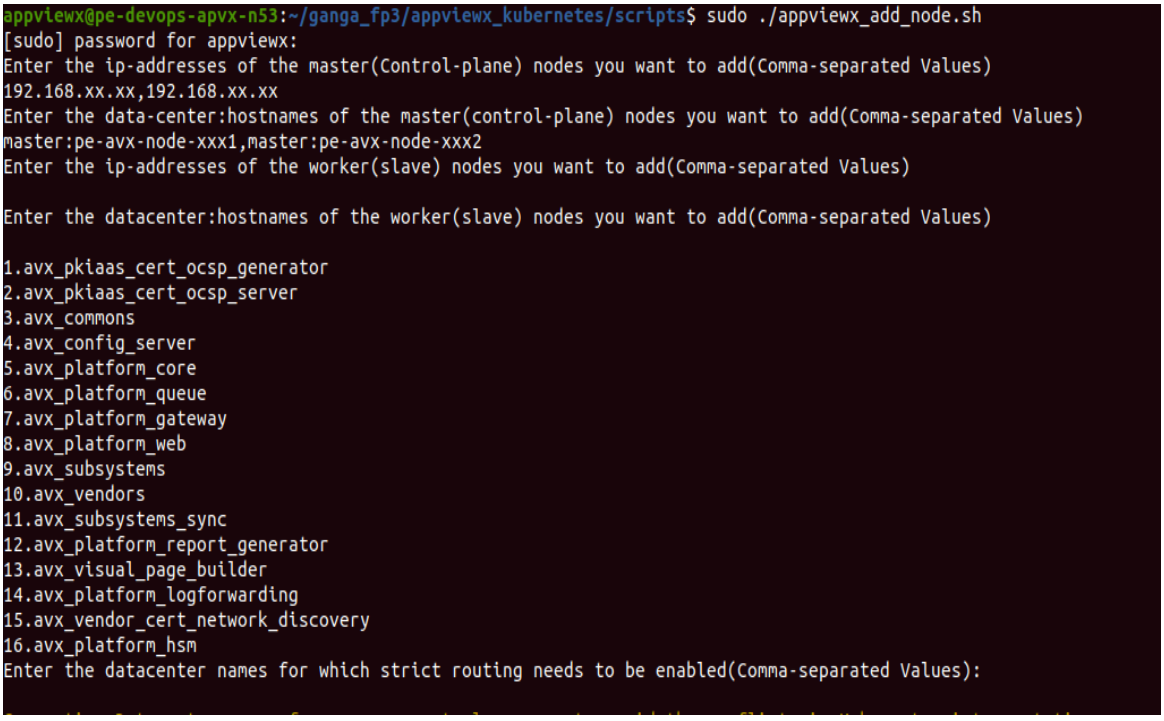
```
sudo ./appviewx_add_node.sh
```

• The command will prompt you to enter the following details:

a. IP addresses of new master nodes.

> **Note:** The total of master nodes must be an odd number (e.g., 3, 5, etc.). Add the master nodes accordingly. To enter multiple IP addresses, separate them with commas (e.g., IP1, IP2).

b. datacentername:hostname of the new master nodes.



> **Note:** If you are prompted to enter the worker node details, hit the enter key and proceed.

3. To verify whether the pods are up and running in the new node, execute the following command:

```
kubectl get pods -n <dcname> -o wide
```

Working with Alerts

Alerts are used to notify users when a predefined target or a condition is met. For example, if the memory usage for a cluster exceeds 90%, you can set an email notification to be sent to the users. This type of notification helps in mitigating the dangers of application downtime that might occur when parameters or go unnoticed.

The following alerts are available:

- Application Alerts
- System Alerts
- Enabling an email Alert
- Troubleshooting Alerts

Enabling an email Alert

AppViewX enables the administrator to send out an email to designated email addresses if the **appviewx.conf** file is modified.

To enable an email alert when the **appviewx.conf** file is modified:

1. Open the terminal.
2. Navigate to the <avx_installed_path>/conf directory.
3. To open the **appviewx.conf** file, execute the following command:

```
vi appviewx.conf
```

4. Update the following SMTP fields in the **appviewx.conf** file.
   - SMTP_SERVER = <email server>:<port>
   - SMTP_SENDER_USER = <sender email address>
   - SMTP_RECEIVER_USER = <sender email address>
5. To get an email alert if the file is tampered, execute the following command:

```
./appviewx --conf_change_alert cron
```

6. To set the command in crontab, complete the following steps:

```
crontab -e

<cron freq> cd /home/appviewx/appviewx/scripts && ./appviewx --conf_change_alert

cron 2>>/home/appviewx/appviewx/logs/cron_logs 1>/dev/null
```

Troubleshooting Alerts

Note: For troubleshooting issues, please refer to the Troubleshooting section.

Working with Backup and Restore

The application level backups are no longer supported in AppViewX. You can back up the mongodb and vault and restore the same in the event of any failure. To facilitate this process, there are scripts available for mongodb and vault backup and restore. You can download them from the release portal.

- Downloading the Scripts
- Backup MongoDB and Vault
- Restore MongoDB and Vault
- Troubleshooting Backup and Restore Operations
- Elastic Backup and Restore

Downloading the Scripts

The scripts are used to trigger the backup and restore operations. The backup files will be created under the directory mentioned in the scripts.

Download the following scripts from the release portal:

- **mongo_backup.sh**
- **vault_backup.sh**
- **vault_restore.sh**
- **mongo_restore.sh**

Copy all the files to the <appviewx_installer_location_path>/appviewx_kubernetes/scripts directory.

Backup MongoDB and Vault

MongoDB Backup

To take a backup of the MongoDB follow the steps below:

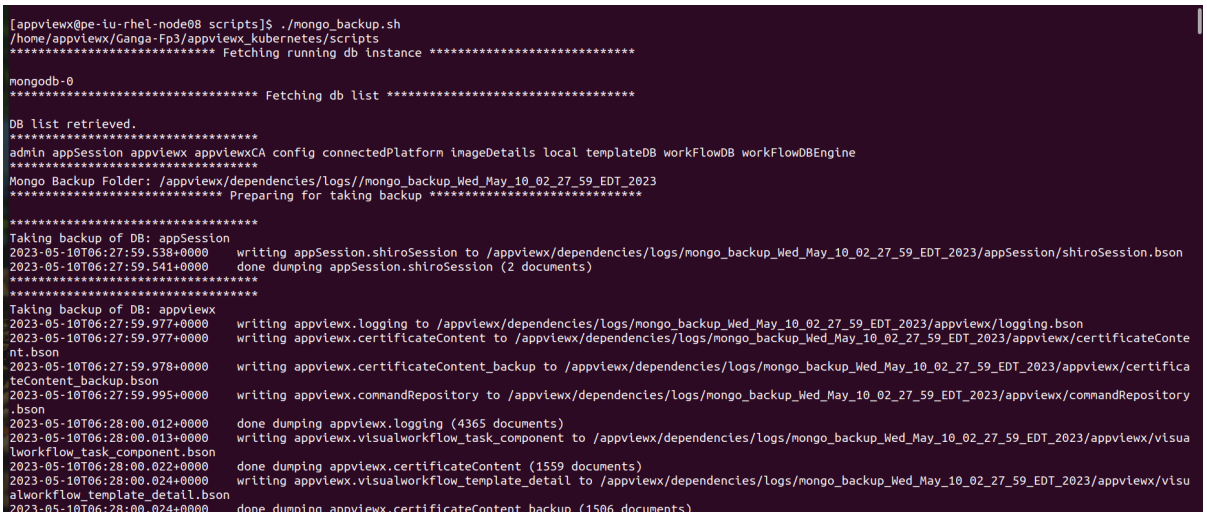
1. Navigate to the scripts folder using the command below.

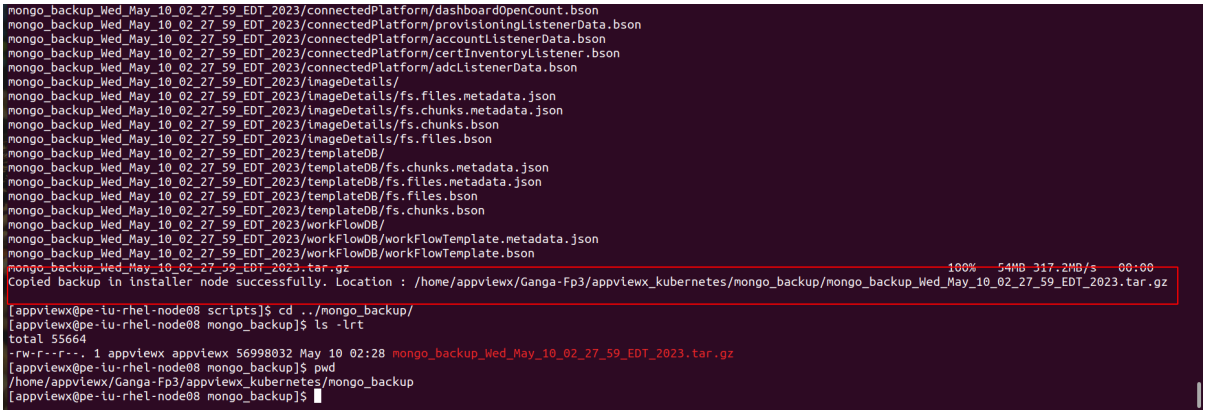
```
cd <appviewx-kubernetes-path>/scripts
```

2. Execute the mongo backup script using the command below.

```
/bin/bash mongo_backup.sh
```

The backup file is created and stored in the location /home/appviewx/Ganga-Fp3/appviewx_kubernetes/mongo_backup/mongo_backup_Wed_May_10_02_40_20_EDT_2023.





Vault Backup

To take a backup of the Vault follow the steps below:

1. Navigate to the scripts folder using the command below.

```
cd <appviewx-kubernetes-path>/scripts
```

2. Execute the mongo backup script using the command below.

```
/bin/bash vault_backup.sh
```

The backup file is created and stored in the location /home/appviewx/Ganga-Fp3/appviewx_kubernetes/vault_backup/vault_backup_Wed_May_10_02_40_20_EDT_2023.

```
[appviewx@pe-iu-node36 scripts]$ ./vault_backup.sh
/home/appviewx/Hudson/appviewx_kubernetes/scripts
Vault Backup File: /home/appviewx/Hudson/appviewx_kubernetes/vault_backup/vault_backup_Fri_Jul_7_10_15_34_IST_2023
```

Restore MongoDB and Vault

MongoDB Restore

To restore the MongoDB follow the steps below:

1. Navigate to the scripts folder using the command below.

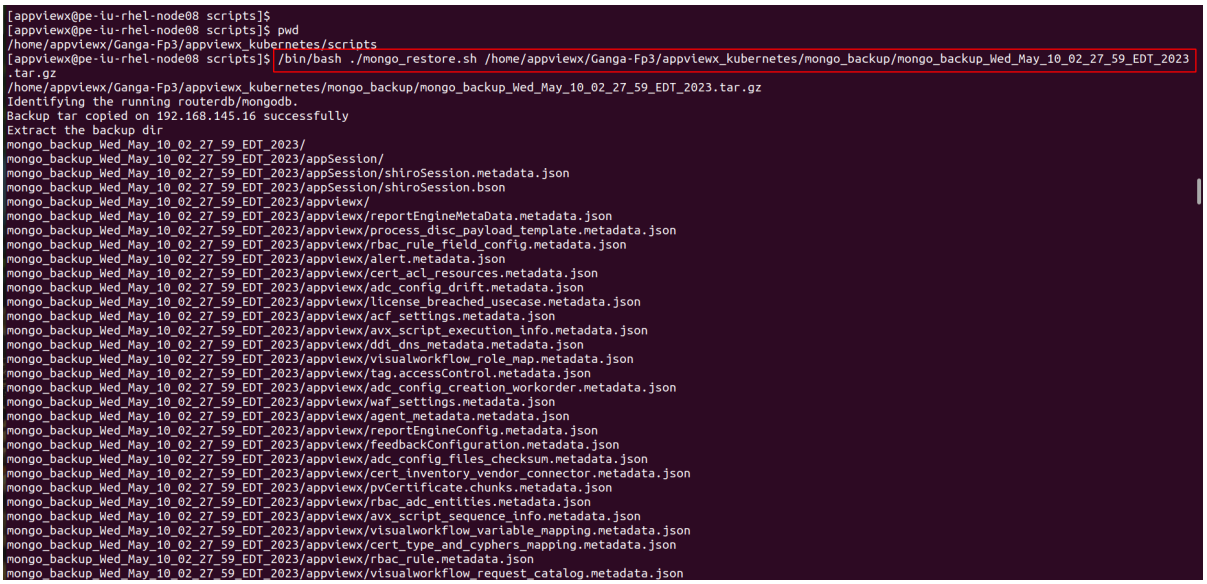
```
cd <appviewx-kubernetes-path>/scripts
```

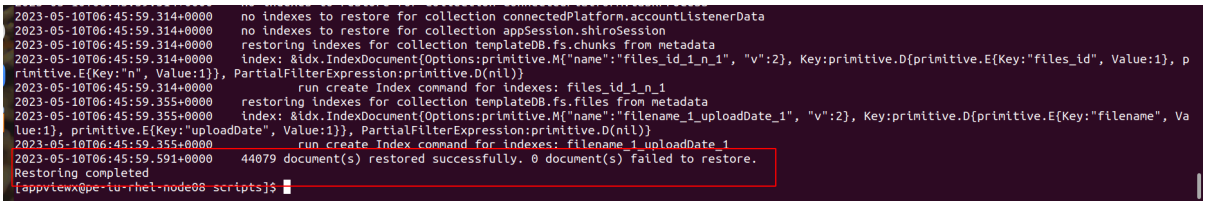
2. Execute the script to restore the backup file using the command below.

```
/bin/bash mongo_restore.sh <appviewx-kubernetes-path>/mongo_backup/<mongo-backup-file>
```

Example

```
/bin/bash mongo_restore.sh /home/appviewx/Ganga-Fp3/appviewx_kubernetes/mongo_backup/mongo_backup_Wed_May_10_02_27_59_EDT_2023.tar.gz
```





Vault Restore

To restore the Vault follow the steps below:

1. Navigate to the scripts folder using the command below.

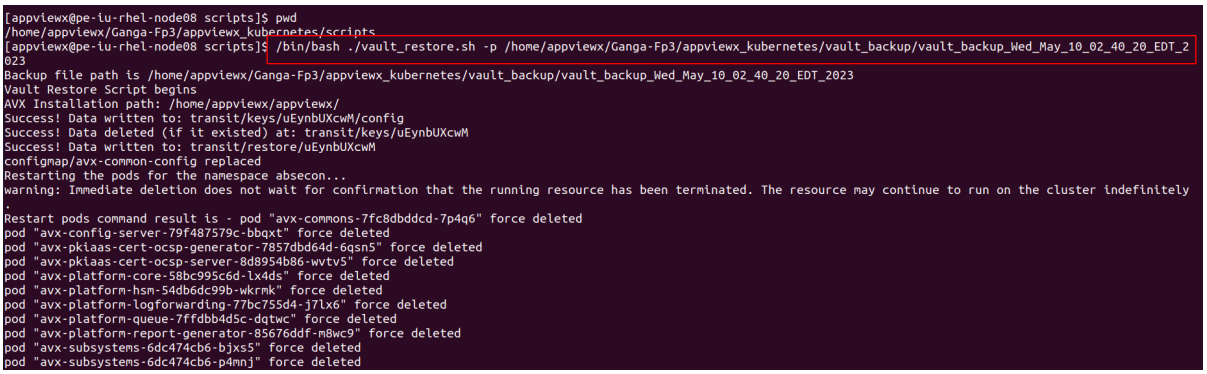
```
cd <appviewx-kubernetes-path>/scripts
```

2. Execute the script to restore the backup file using the command below.

```
/bin/bash vault_restore.sh -p <appviewx-kubernetes-path>/vault_backup/<vault-backup-file>
```

Example

```
/bin/bash vault_restore.sh -p /home/appviewx/Ganga-Fp3/appviewx_kubernetes/vault_backup/vault_backup_Wed_May_10_02_40_20_EDT_2023
```



Troubleshooting Backup and Restore Operations

Note: For troubleshooting issues, please refer to the Troubleshooting section.

Elastic Backup and Restore

Elastic Backup

The script **elastic_backup.py** is for elastic backup. To manually perform the elastic backup,

1. Navigate to the scripts directory.
2. Run the script using the application provided python binary, example

```
/home/appviewx/appviewx/appviewx_dependency/appviewx_addons/Python_Linux/bin/python
```

3. Before taking the backup we have to set up the Elasticsearch repo for storing the snapshots.

```
/home/appviewx/appviewx/appviewx_dependency/appviewx_addons/Python_Linux/bin/python elastic_backup.py --setup elasticsearch_insight
```

4. To take the backup run the command

```
/home/appviewx/appviewx/appviewx_dependency/appviewx_addons/Python_Linux/bin/python elastic_backup.py --backup elasticsearch_insight
```

5. Perform the following steps after taking the backup

a. Navigate to the clusters node where the elasticsearch-insight is deployed
b. Navigate to the installer path and create the tar of the elastic-insight-backup directory
c. Copy the backup tar into the installer node.

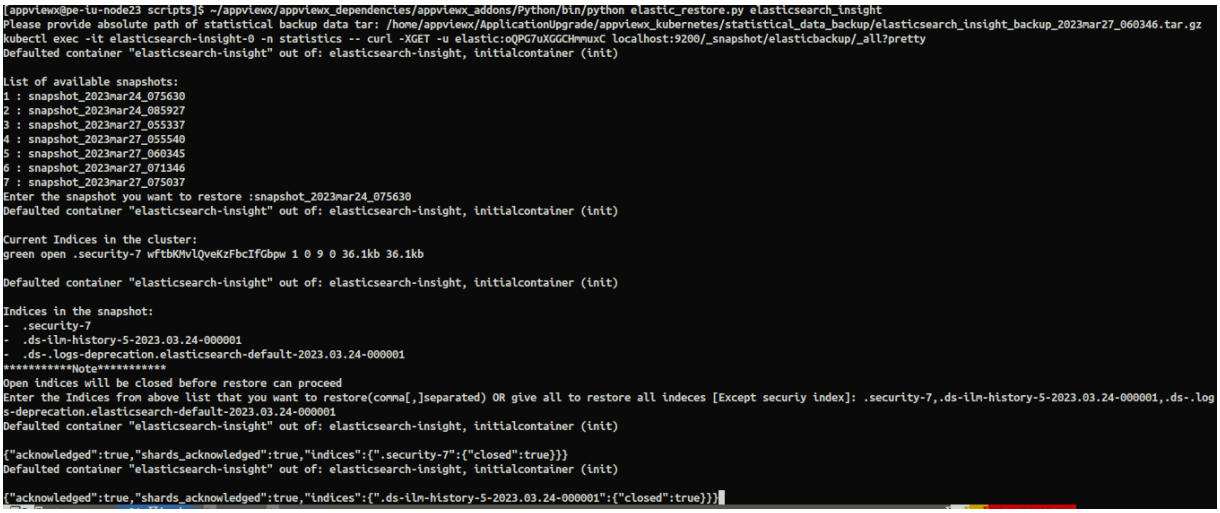
Elastic Restore

The script **elastic_restore.py** is used for restore. To manually perform the elastic restore,

1. Navigate to the scripts directory.
2. Run the **elastic_restore.py** script to restore the backup

```
/home/appviewx/appviewx/appviewx_dependency/appviewx_addons/Python_Linux/bin/python elastic_restore.py elasticsearch_insight
```

3. Script will ask for the backup tar which was created manually. Provide the absolute path of the backup tar.



4. Script will list all the available snapshots that has date and time in the naming. Select the backup which you want to restore.
5. Provide the details of the indices you want to restore (follow the screenshot above).

Working with Logs

In any application, log files are used to record all events. It provides information about the customer usage patterns, the names of modules that are used frequently. In addition, they also help users analyse the issues depending on the events.

In any application, there are mainly two types of logs that are collected. One of them is application logs that are required to monitor the performance. Another type of log that is maintained is infrastructure logs.

These logs are used to monitor the status of the hardware infrastructure like memory usage, disk usage, and CPU usage.

In AppViewX, the log files are collected and maintained for plugins. To manage logs, AppViewX uses Kibana.

- Managing Logs using Kibana
- Managing Logs using AppViewX Nodes
- Automatic Log Collection
- Log Analyser Tool

Managing Logs using Kibana

Kibana is an open user interface that enables you to graphically represent the log files and monitor system performance.

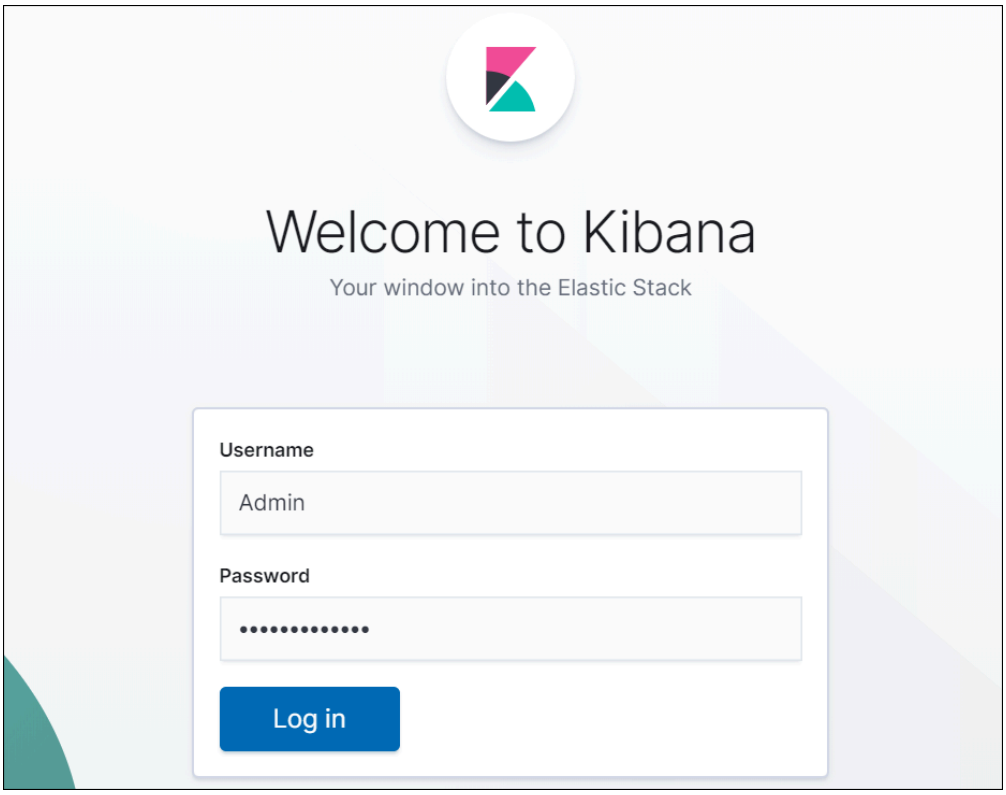
Before using Kibana, ensure that you have the following:

- **Kibana Web URL** - mentioned in the <INSTALLATION_PATH>/.appviewx_configuration file
- **Kibana Username**- mentioned in the <INSTALLATION_PATH>/.appviewx_configuration file
- **Kibana Password** - mentioned in the <INSTALLATION_PATH>/.appviewx_configuration file

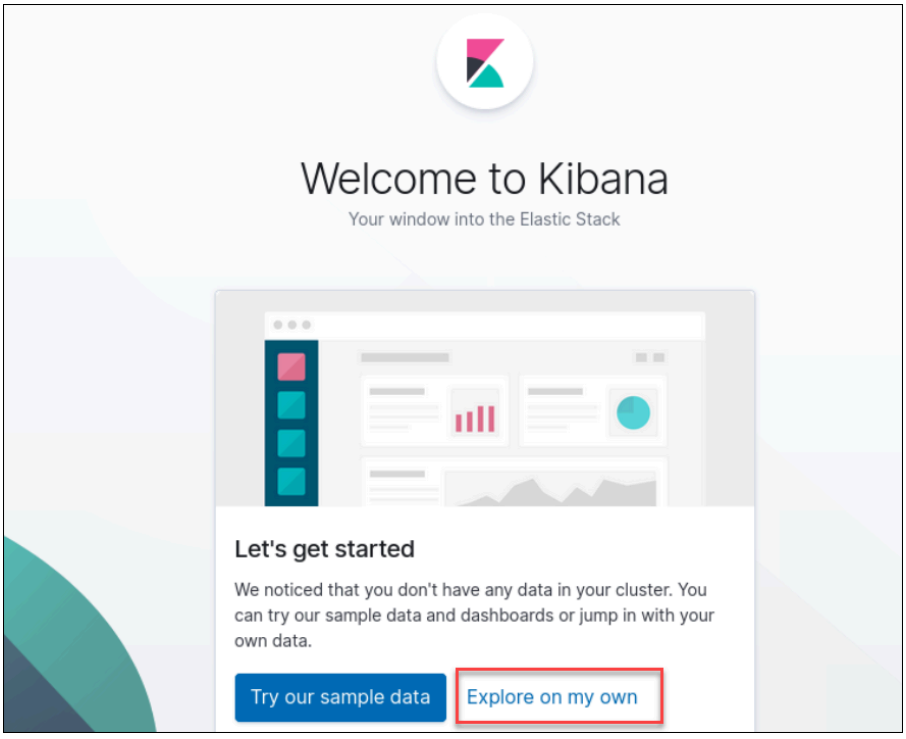
- Accessing Kibana
- Creating an Index Pattern
- Viewing Logs
- Generating a Report

Accessing Kibana

1. Open the Kibana Web URL.
2. Enter the credentials.
3. Click **Log in**.

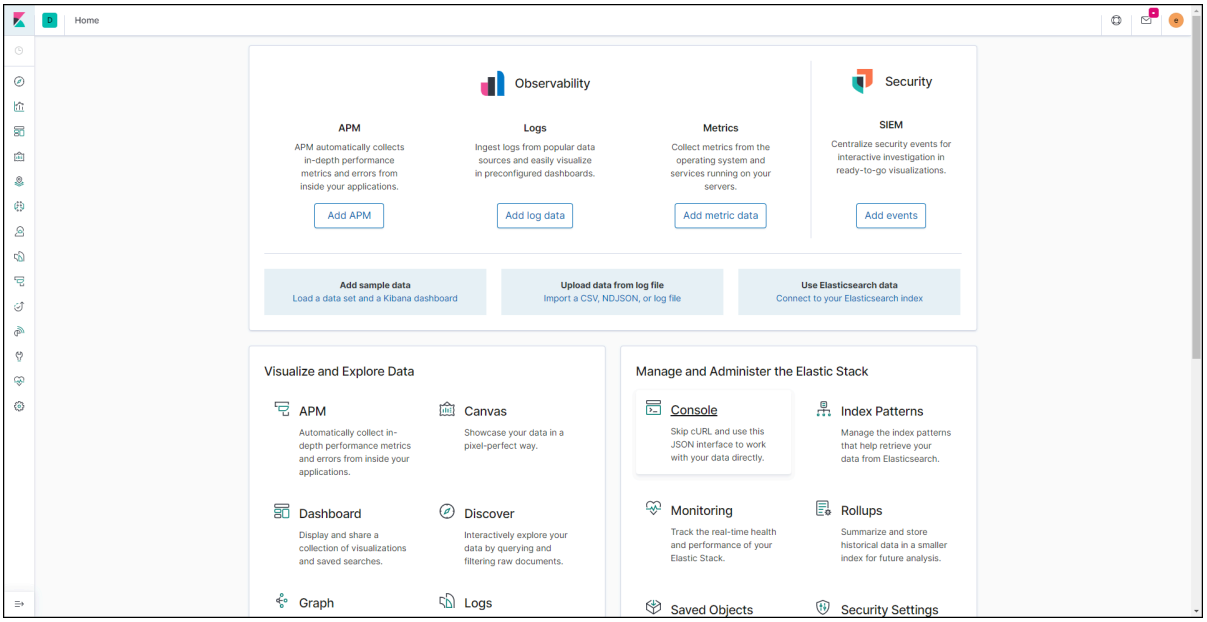


The **Let's get started** page is displayed.



4. Click **Explore on my own**.

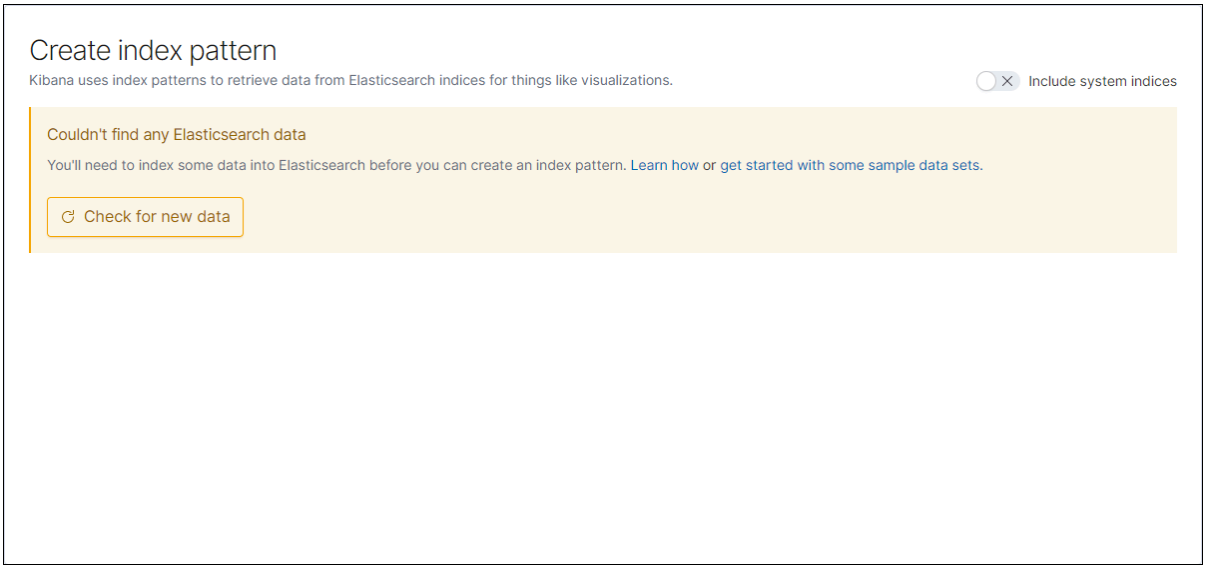
The **Home page** is displayed.



Creating an Index Pattern

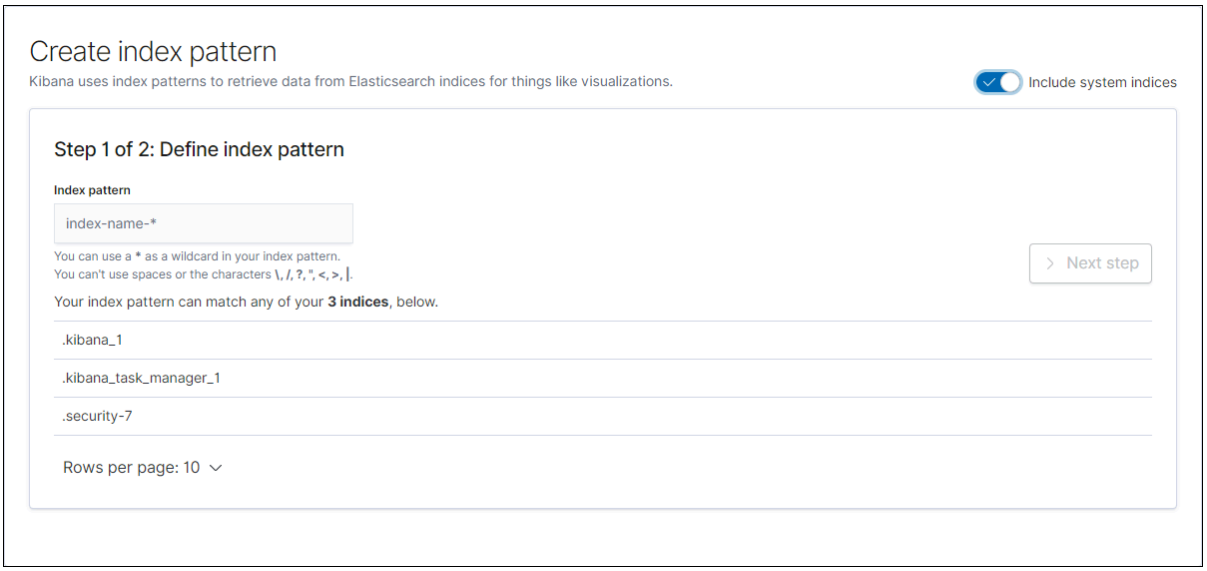
1. Login to Kibana.
2. Under **Visualize and Explore Data**, click **Visualize**.

   The **Create index pattern** page is displayed.



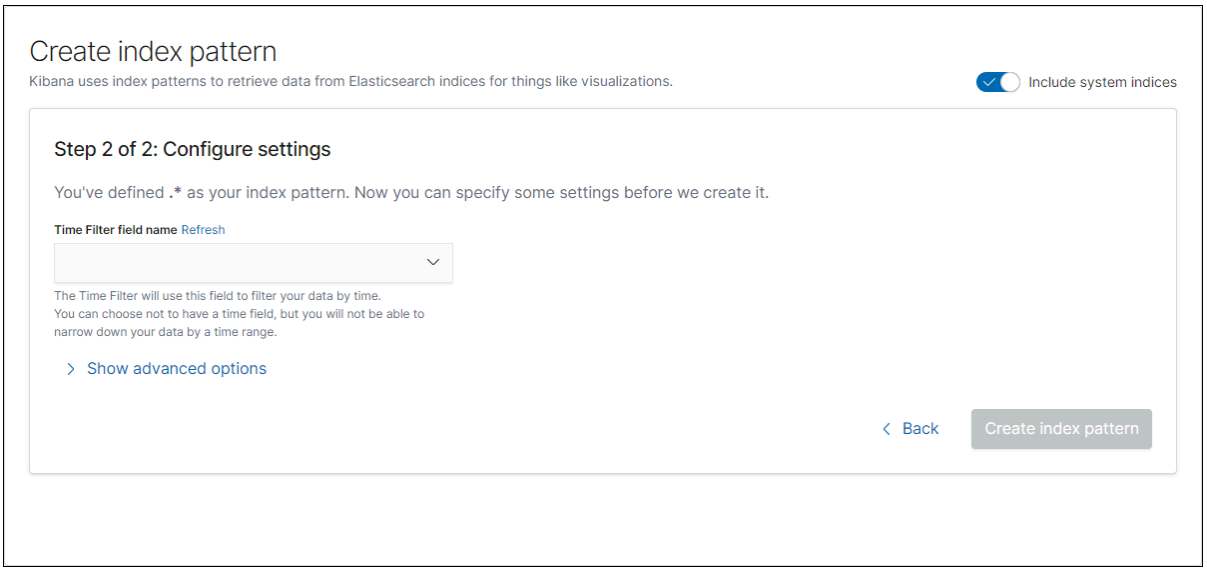
3. Enable the **Include system indices** option.

   The **Define index pattern** page is displayed.



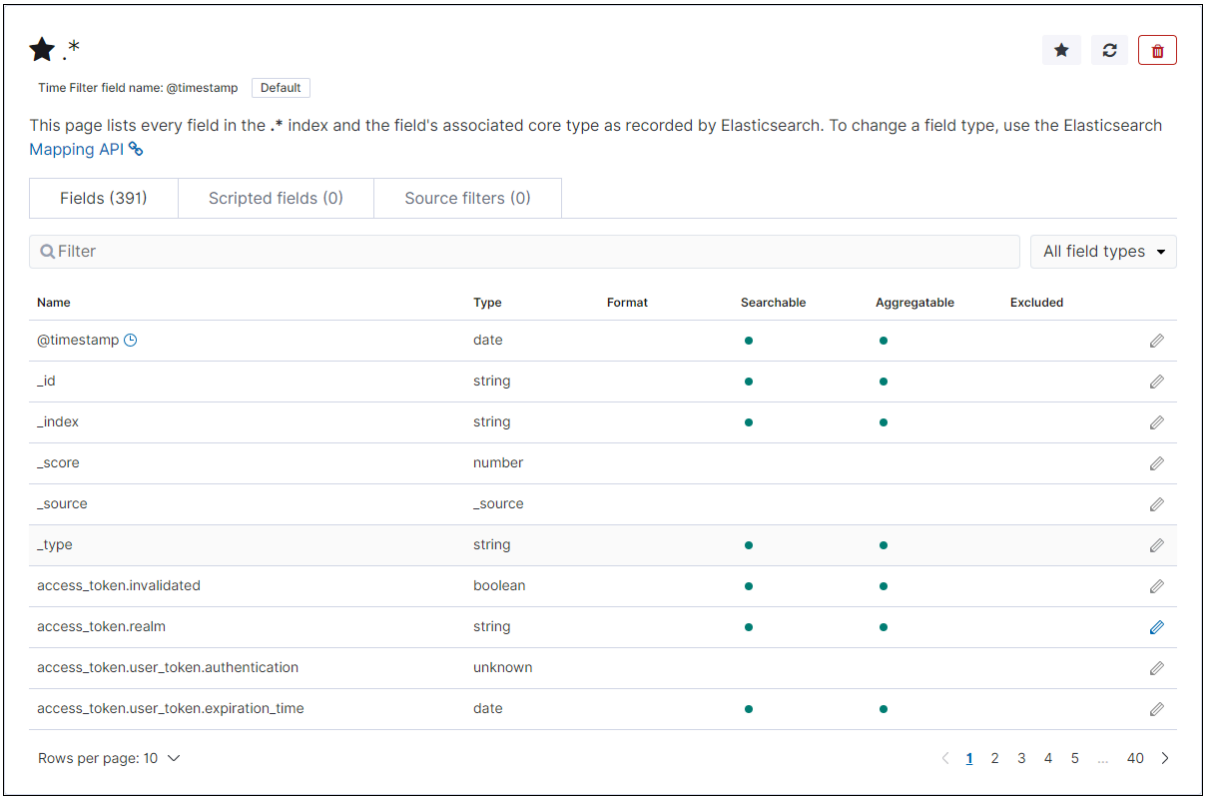
4. Under Index pattern, enter .*.
5. Click **Next step**.

   The **Configure settings** page is displayed.



6. From the **Time Filter field name** list, select **@timestamp**.
7. Click **Create Index Pattern**.

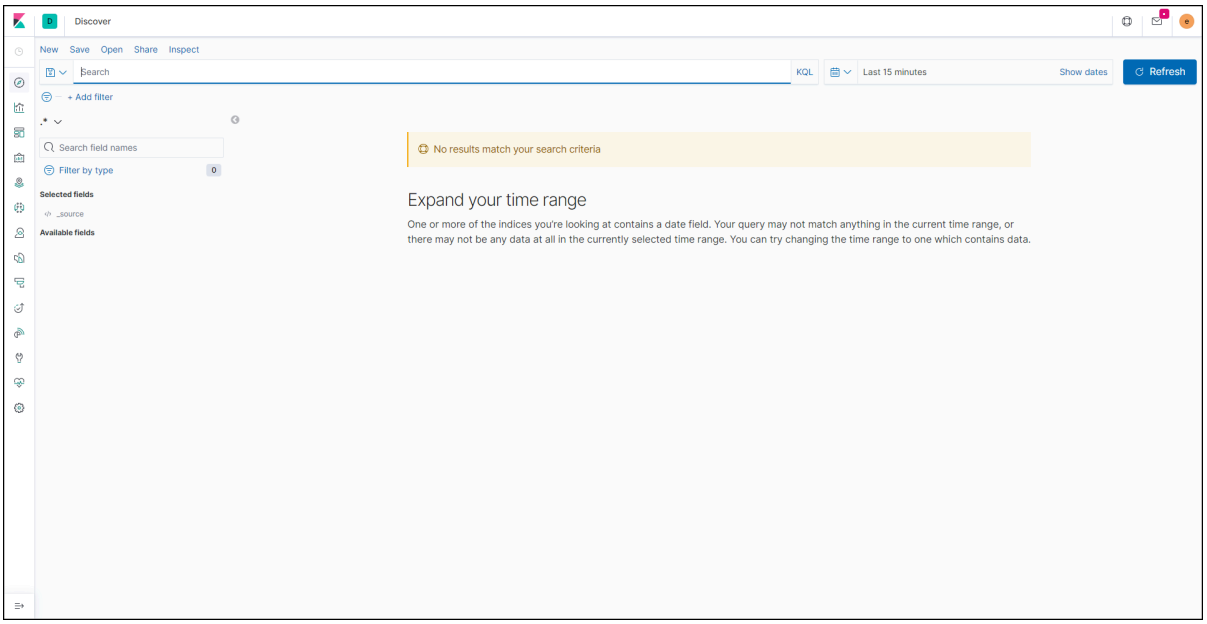
The system creates an index pattern.



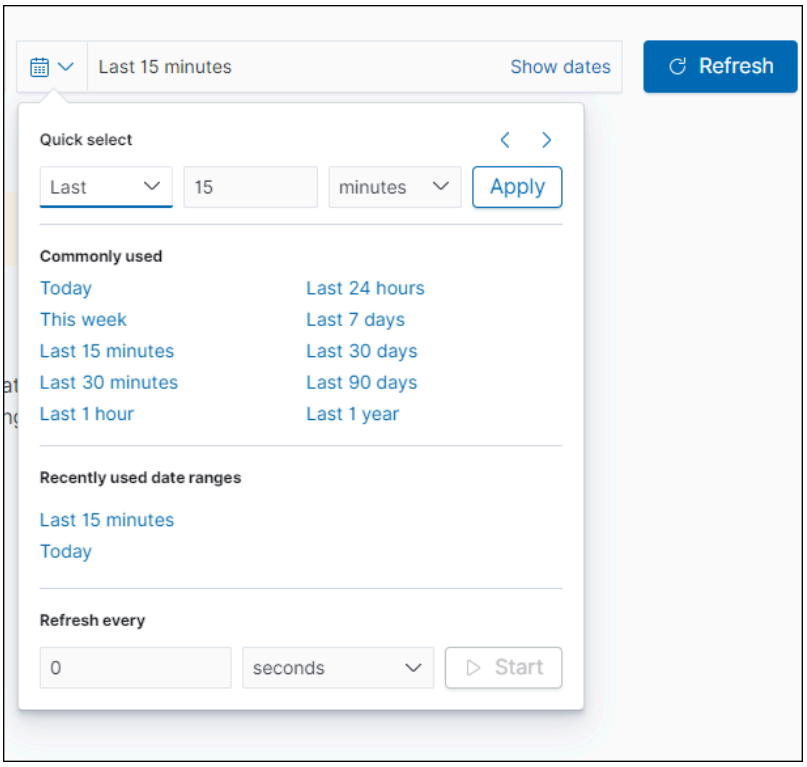
Viewing Logs

1. Login to Kibana.
2. Under **Visualize and Explore Data**, click **Discover**.

   The **Discover** page is displayed.



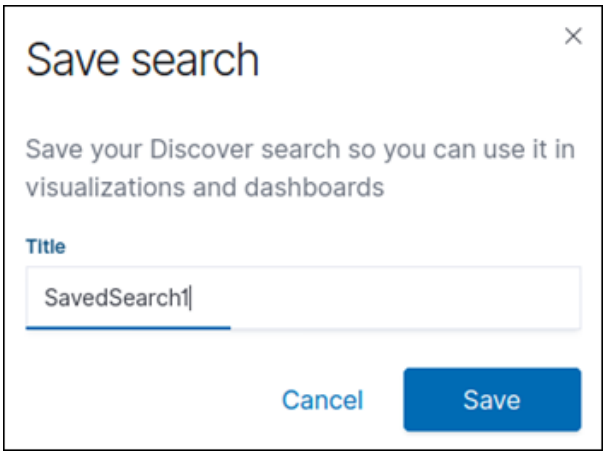
3. In the time frame section, select the time frame within which the logs need to be captured.



4. To view the updated logs, click **Refresh**.
5. To save the search:

a. Click **Save**.

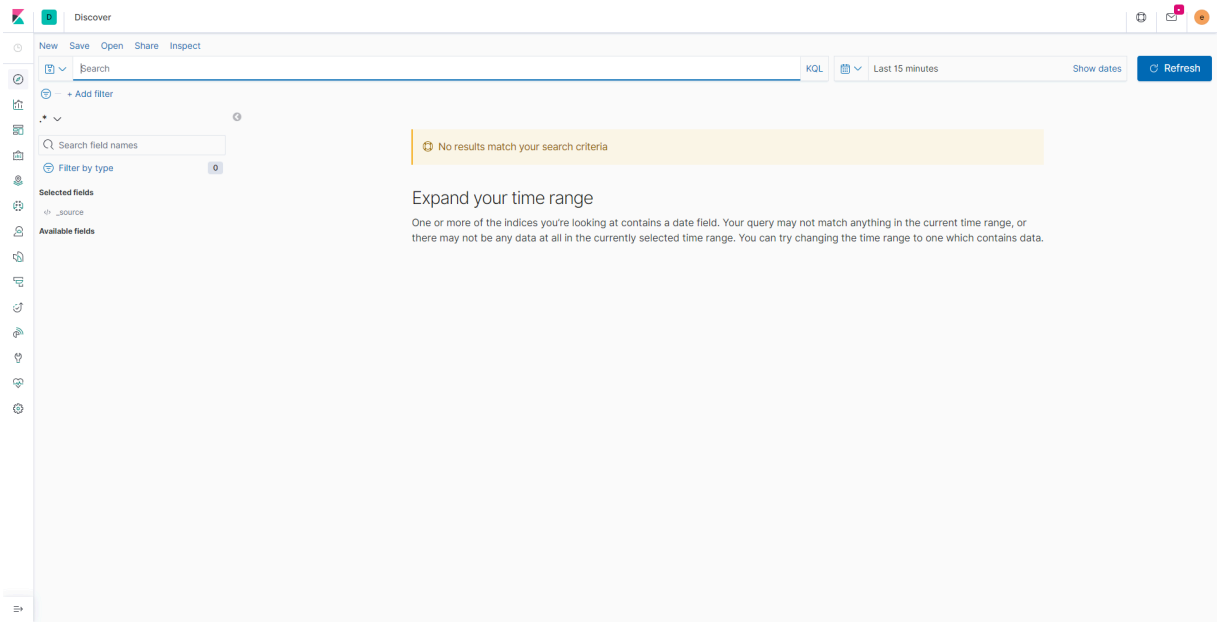
b. Enter a valid name to save the search.



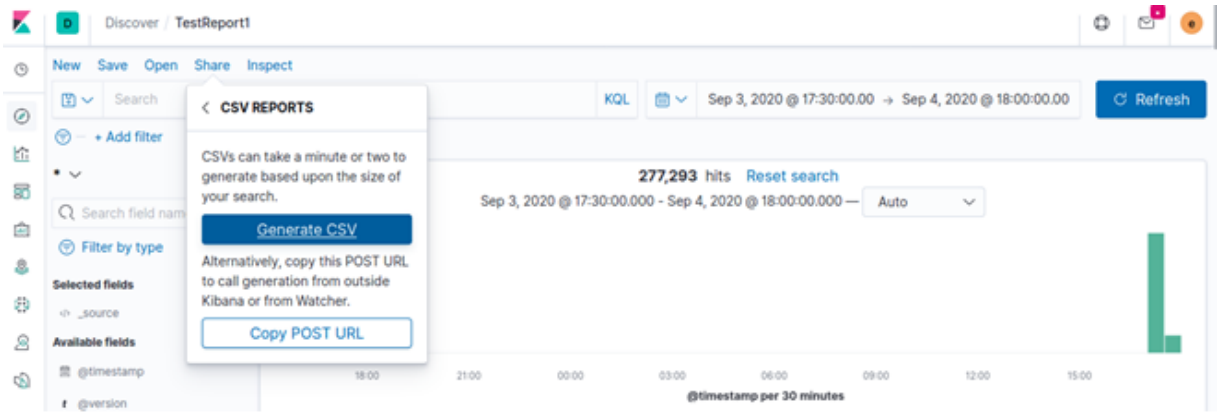
Generating a Report

Kibana enables you to generate a report in CSV format. In order to generate the report, you must copy the <.ndjson> ext files from the <InstallerLocation>/appviewx_kubernetes/yaml/appviewx_monitoring/kibana/deploy location and import into the import section (for example, <gateway.ndjson> and <platform.ndjson>).

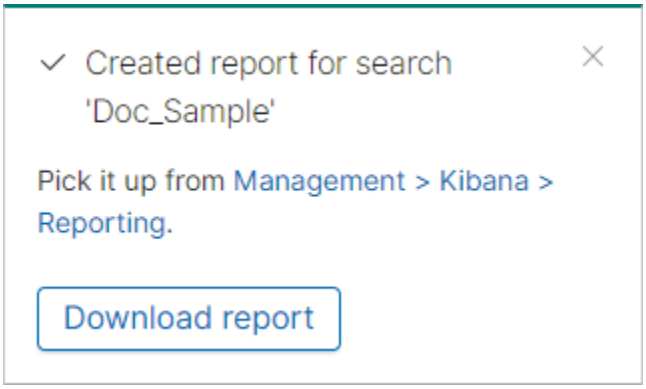
1. Login to Kibana.
2. Under **Visualize and Explore Data**, click **Discover**.
   The **Discover** page is displayed.



3. Select **Share > CSV reports > Generate CSV**.



The system generates the report and prompts to download the same.



4. To download the report, click **Download report**.
   The system downloads the report to the default download location.

Managing Logs using AppViewX Nodes

You can also view and manage the log files even if you do not have Kibana installed. In this case, you can use the AppViewX nodes to view and manage the log files. You can also view logs using the command line interface before you install the ELK.

Note: Logs are maintained as per the retention policy. Any log exceeding 30 MB will be rotated and archived as part of the data retention policy.

To view the logs:

1. Log in to the respective node.
2. Navigate to the appviewx/dependencies/logs directory. You can view the CLI logs for pods in the same node.

To view the logs from the AppViewX nodes:

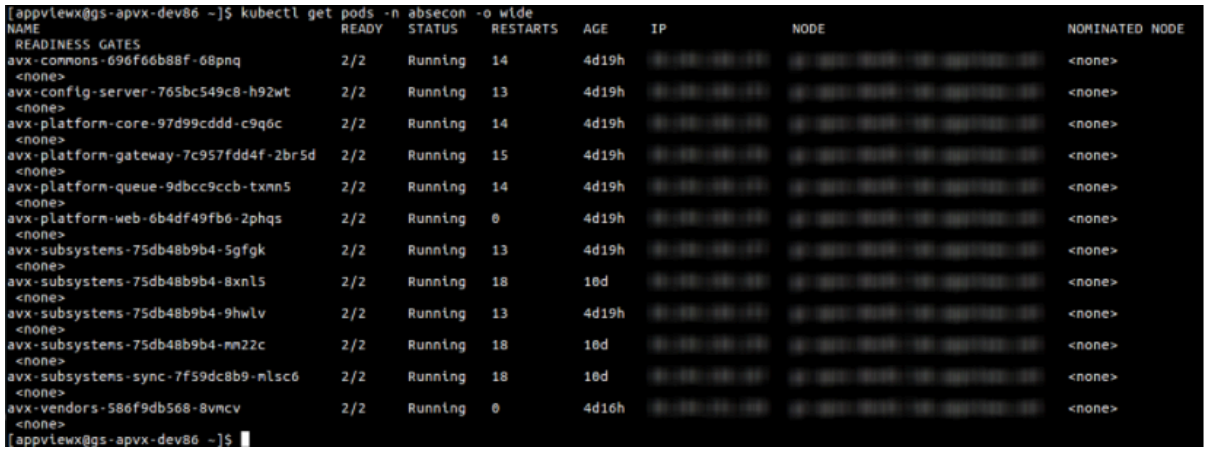
1. Using the command line interface, log in to the AppViewX node.
2. To fetch the node name in which the pod is running, execute the following command:

```
kubectl get pods- n <dc> -o wide
```

3. Log in to the respective node using SSH.
4. Navigate to `<INSTALLATION_PATH>/logs` for all log files.

   For example, If you want to view the logs for the subsystem plugin in the datacenter DC1, execute the following command to get the node name of the pod:

```
kubectl get pods -n DC1 -o wide
```



```
[appviewx@gs-apvx-dev86 ~]$ kubectl get pods -n absecon -o wide
NAME                                    READY   STATUS    RESTARTS   AGE     IP              NODE                          NOMINATED NODE
 READINESS GATES
avx-commons-696f66b88f-68pnq            2/2     Running   14         4d19h   [redacted]      [redacted]                    <none>
 <none>
avx-config-server-765bc549c8-h92wt      2/2     Running   13         4d19h   [redacted]      [redacted]                    <none>
 <none>
avx-platform-core-97d99cddd-c9q6c       2/2     Running   14         4d19h   [redacted]      [redacted]                    <none>
 <none>
avx-platform-gateway-7c957fdd4f-2br5d   2/2     Running   15         4d19h   [redacted]      [redacted]                    <none>
 <none>
avx-platform-queue-9dbcc9ccb-txmn5      2/2     Running   14         4d19h   [redacted]      [redacted]                    <none>
 <none>
avx-platform-web-6b4df49fb6-2phqs       2/2     Running   0          4d19h   [redacted]      [redacted]                    <none>
 <none>
avx-subsystems-75db48b9b4-5gfgk         2/2     Running   13         4d19h   [redacted]      [redacted]                    <none>
 <none>
avx-subsystems-75db48b9b4-8xnl5         2/2     Running   18         10d     [redacted]      [redacted]                    <none>
 <none>
avx-subsystems-75db48b9b4-9hwlv         2/2     Running   13         4d19h   [redacted]      [redacted]                    <none>
 <none>
avx-subsystems-75db48b9b4-mm22c         2/2     Running   18         10d     [redacted]      [redacted]                    <none>
 <none>
avx-subsystems-sync-7f59dc8b9-nlsc6     2/2     Running   18         10d     [redacted]      [redacted]                    <none>
 <none>
avx-vendors-586f9db568-8vmcv            2/2     Running   0          4d16h   [redacted]      [redacted]                    <none>
 <none>
[appviewx@gs-apvx-dev86 ~]$
```

Note: For troubleshooting issues, please refer to the Troubleshooting section.

Automatic Log Collection

The **collect_logs.sh** script (under the *scripts_util* directory) has been updated to collect all logs from all the nodes in a cluster setup (single node and multi node). After all the logs are collected, the log dump directory is cleaned up and the tar ball (tar file) is created with all the generated logs and available for download.

Prerequisite

1. Ensure that the AppViewX application is updated with latest 2022.1.0 FP3 patch or upgraded to 2023.1.0 (FP1-FP3)
2. Input (sudo) passwords for **message** and **mongodb** logs.

To obtain the all the logs

1. Go to the terminal prompt and navigate to the **scripts** folder by executing

```
cd /home/<folder_location>/appviewx_kubernetes/scripts
```

2. To collect all logs, execute the following command,

```
./appviewx.sh - -collect-logs all-logs
```

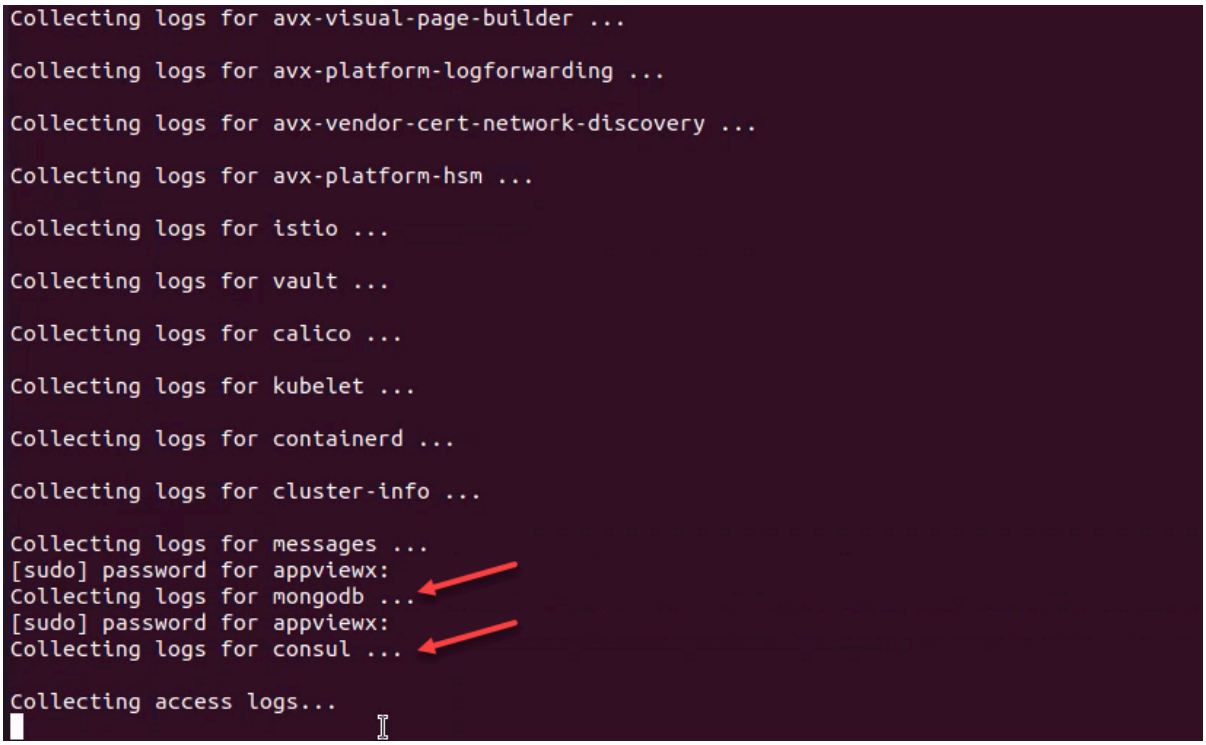
```
bash-4.2$ ./appviewx.sh --collect-logs all-logs
Input Validation Completed Successfully !
Please enter appviewx password of master:pe-devops-appvx-node10.lab.appviewx.net :
Please enter appviewx password of absecon:pe-devops-appvx-node7.lab.appviewx.net :
Please enter appviewx password of absecon:pe-devops-appvx-node8.lab.appviewx.net :
Please enter appviewx password of absecon:pe-devops-appvx-node9.lab.appviewx.net :
```

The message "*Input validation completed successfully*" is displayed. Since the command for all-logs include logs from mongodb and message, you will be prompted to enter the passwords.

3. Enter the respective sudo passwords and hit the Enter key.

The below messages are displayed once the log collection starts. If there are no logs for any system then 'No logs present for…." Is displayed.

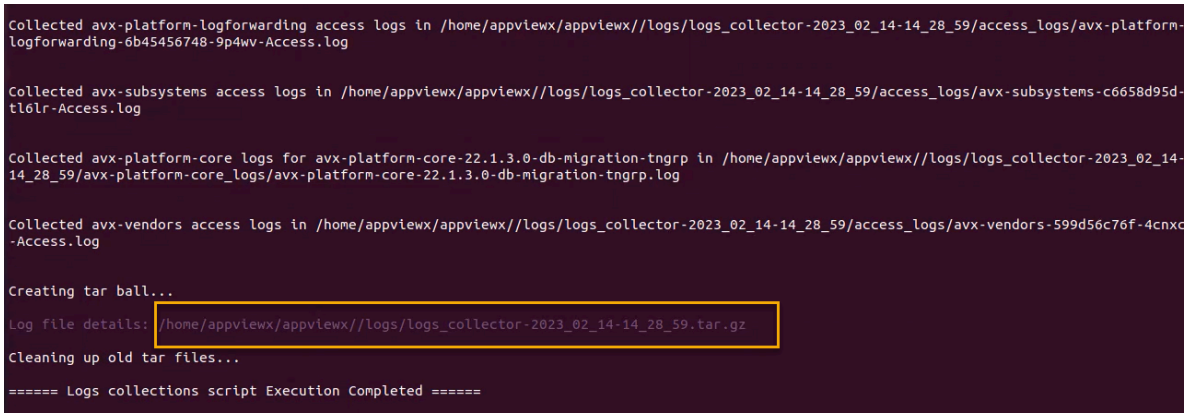
```
Starting files copy...

No logs present for appviewx-dependencies ...

Collecting logs for avx-pkiaas-cert-ocsp-generator ...

Collecting logs for avx-pkiaas-cert-ocsp-server ...

Collecting logs for avx-commons ...

Collecting logs for avx-crontab ...

Collecting logs for avx-config-server ...

Collecting logs for avx-platform-core ...

Collecting logs for avx-platform-queue ...

Collecting logs for avx-platform-gateway ...

Collecting logs for avx-platform-web ...

Collecting logs for avx-subsystems ...

Collecting logs for avx-vendors ...

Collecting logs for avx-subsystems-sync ...

Collecting logs for avx-platform-report-generator ...

Collecting logs for avx-visual-page-builder ...

Collecting logs for avx-platform-logforwarding ...
```



Note: The passwords during the log collection process will be taken automatically as they were entered at the beginning of the log collection process.

4. The log collection process continues with the following steps
   a. Logs files are created for each application (plugins, DBs) and are saved in a folder location as indicated.
   b. The final process is creation of the tar ball. The tar ball contains all the logs, its location is displayed in the end.
   c. Old tar files (up to two days old) are deleted from the folder.

d. Process ends with the message "*Log collection script execution completed.*"



Individual logs can also be collected by executing for each system as mentioned below:

```
./appviewx.sh - -collect-logs all-plugins
```

```
./appviewx.sh - -collect-logs avx-platform-gateway
```

```
./appviewx.sh - -collect-logs mongo
```

```
./appviewx.sh - -collect-logs istio
```

```
./appviewx.sh - -collect-logs vault
```

```
./appviewx.sh - -collect-logs calcio
```

```
./appviewx.sh - -collect-logs consul
```

```
./appviewx.sh - -collect-logs kubelet
```

```
./appviewx.sh - -collect-logs containerd
```

```
./appviewx.sh - -collect-logs cluster-info
```

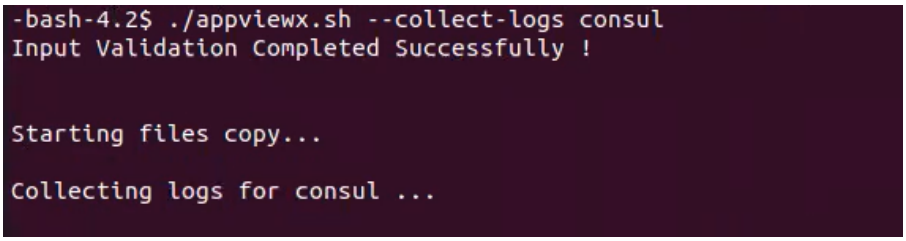
```
./appviewx.sh - -collect-logs messages
```

```
./appviewx.sh - -collect-logs access
```

```
/appviewx.sh - -collect-logs <deployment-name> [supported:

 avx-subsystems,avx-vendors,avx-platform-core,avx-platform-queue,kubelet,containerd,messages]
```

Note: <deployment-name> will recognize only one argument from the supported list - [supported: avx-subsystems,avx-vendors,avx-platform-core,avx-platform-queue,kubelet,containerd,messages]

An example of execution for collection of logs for consul is shown below. This execution will process without the password input.



Note: In case you encounter any problems while using this tool, kindly capture a screenshot of the CLI output and reach out to the support team for assistance.

Log Analyser Tool

This topic contains the installation steps to for the log analyzer tool (logmon). A new component has been added in the existing **scripts** files and a parameter LOGMON_HOST added to the **appview.conf** file to support this feature.

To intall the log analyser tool

1. Login to the release portal and download the following filesfile. executing
   a. **scripts.tar.gz**
   b. **appviewx_kubernetes_logmon_20xx.x.x_FPx.tar.gz**
2. Copy both the above files to the installer node.
3. Sync updated scripts with existing installer scripts using the commands below.

```
tar -xf scripts.tar.gz
```

```
cp -r scripts/* <INSTALLER_PATH>/appviex_kuberetes/scripts
```

4. Navigate to installer scripts folder.
5. Sync the **appviewx.conf** file using the command below.

```
./appviewx.sh --conf-merge
```

6. Install the logmon components using the commands below.

```
chmod +x logmon_install.sh
```

```
./logmon_install.sh
```

Working with Plugins

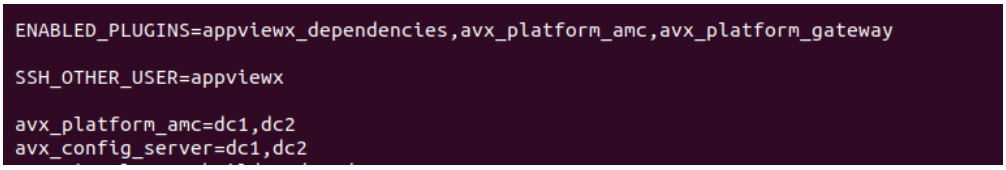
Adding a New Plugin

During the AppViewX installation, the user may not enable all the plugins that are required. Therefore, the user can enable those plugins after the AppViewX installation.

To enable a plugin after installation:

1. Navigate to the /home/appviewx/appviewx_kubernetes/scripts directory.
2. Open the **appviewx.conf** file.
3. Modify the **ENABLED_PLUGINS** as new plugins that need to be installed.

> **Warning:** It is not recommended to delete the appviewx_dependencies in the ENABLED_PLUGINS value. For example, ENABLED_PLUGINS= avx_dependencies,avx_vendors.



4. Enter the data center value in which the plugin needs to be installed.
   For example, **avx_vendors=dc1**.

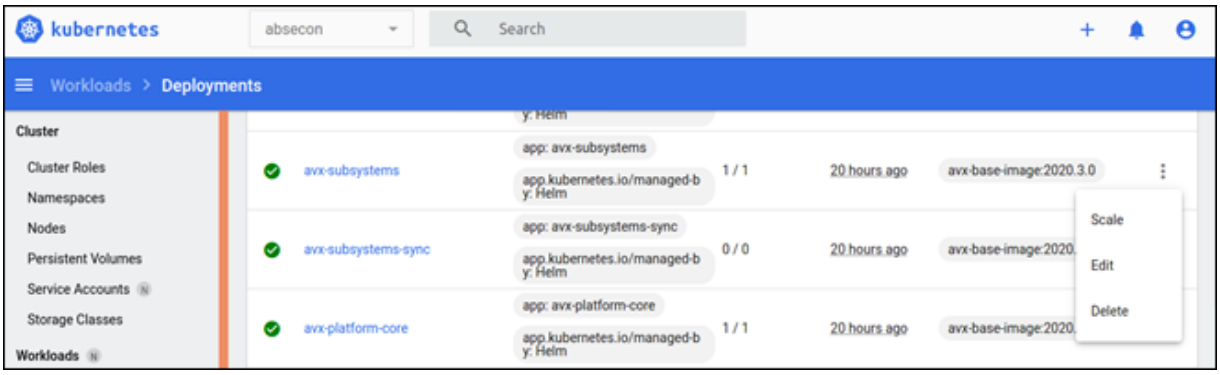
5. Save and exit the **appviewx.conf** file.
6. Navigate to the **scripts** directory.
7. In the scripts directory, execute the following command:

```
script plugins_install.sh
```

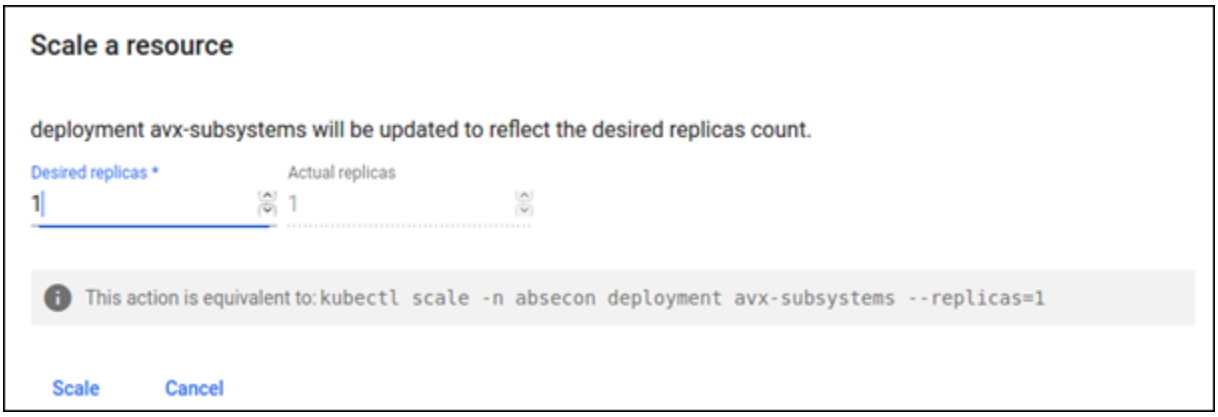
Removing a Plugin

To remove a plugin for maintenance purposes:

1. Log in into the kubernetes management console.
2. From the top list, select the required namespace or datacenter.
3. From the left pane, click **Deployments**.
4. Search for the specific deployment/plugin that needs to be stopped.
5. Against the name of the pod, click the three dots and select **Scale**.



The **Scale a Resource** page is displayed.

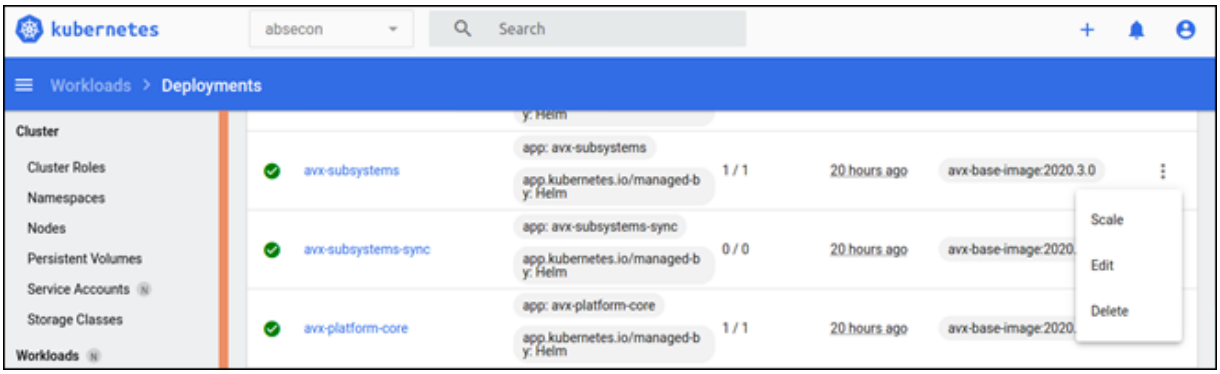


6. Set the value for **Desired replicas** to 0.

This will delete all the pods and does not spin any new pod for that plugin.

Restarting a Plugin

1. Log in into the kubernetes management console.
2. From the top list, select the required namespace or datacenter.
3. From the left pane, click **Deployments**.
4. Search for the specific deployment/plugin that needs to be restarted.
5. Against the name of the pod, click the three dots and select **Delete**.
   This will stop the current pod and create a new pod.

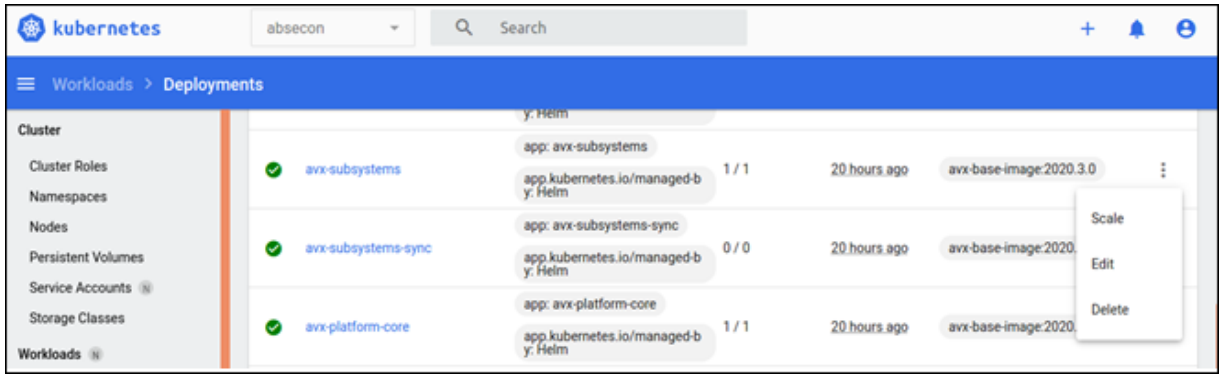


Scaling a Plugin

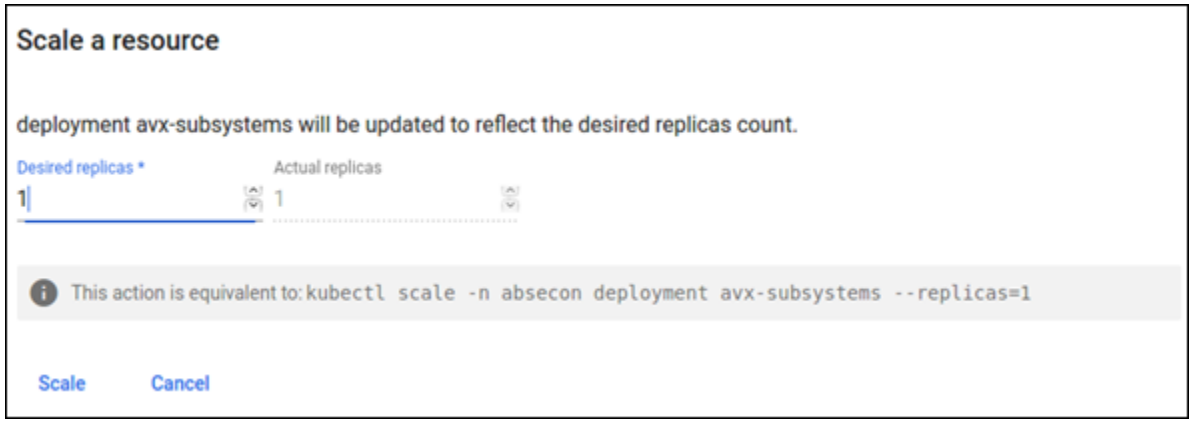
Scale refers to an increase or decrease in the number of plugins manually. You have an option to scale it from the Kubernetes management console.

To increase/decrease the number of plugins of a specific type:

1. Log in into the kubernetes management console.
2. From the top list, select the required namespace or datacenter.
3. From the left pane, click **Deployments**.
4. Search for the specific deployment/plugin that needs to be scaled.
5. Against the name of the pod, click the three dots and select **Scale**.



The **Scale a Resource** page is displayed.



6. Update the value of the **Desired replicas** parameter to increase or decrease the number of pods for a plugin.
7. Click **Scale**.

Changing the Memory for a Plugin

Every plugin inside the node runs on a dedicated memory. It can be adjusted to the maximum and minimum memory that a pod can use.

To increase or decrease the plugins memory:

1. Log in to the Kubernetes dashboard of AppViewX.
2. From the left pane, under **Workloads**, click **Deployments**.
3. Search for the respective deployment to modify it.
4. Click **Edit**.
5. Modify the xmx and xms values to the required values as shown below.

Working with the Management Console

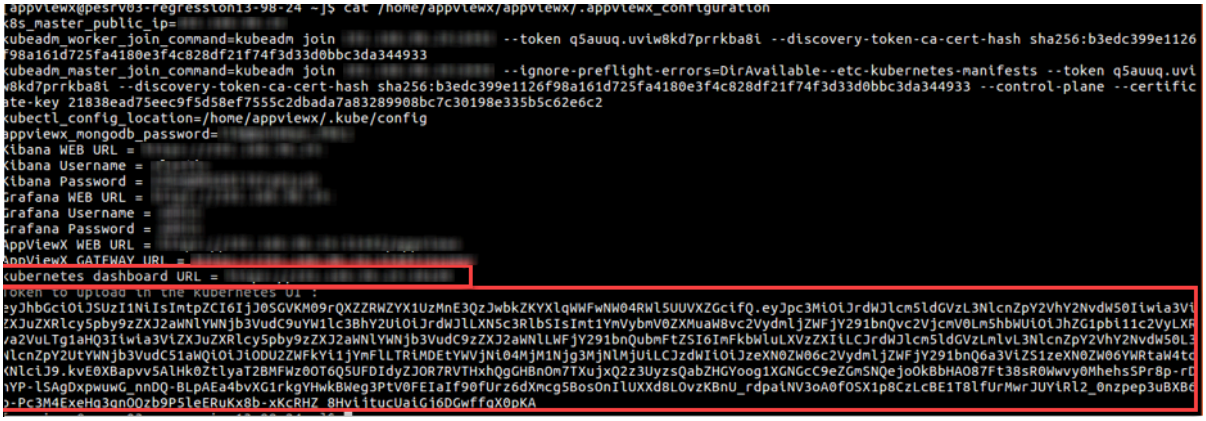
The management console allows you to monitor, maintain, and manage the application as well as the performance. The console provides a graphical interface to view and monitor the application instance.

- Accessing the Management Console
- Viewing the POD Status
- Accessing the POD Console
- Accessing the Database Command Line
- Exporting a Database Collection

Accessing the Management Console

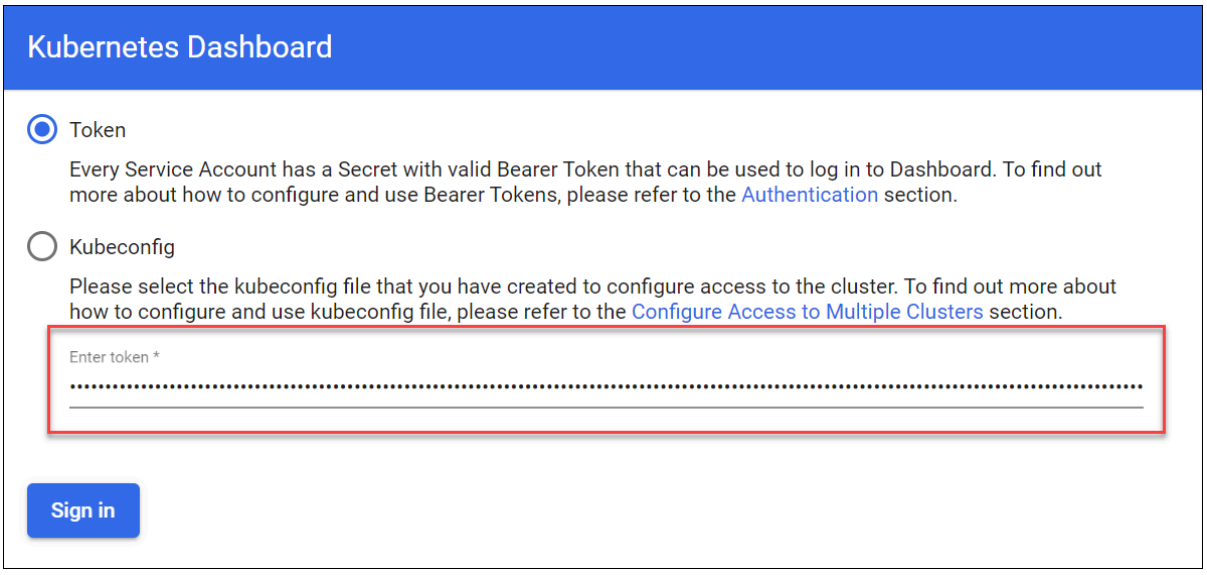
1. Navigate to the installation directory.
2. Open the **.appviewx_configuration** file.

   For example: cat/home/appviewx/appviewx/.appviewx_configuration
3. Search for Kubernetes dashboard URL and copy the URL to a web browser.

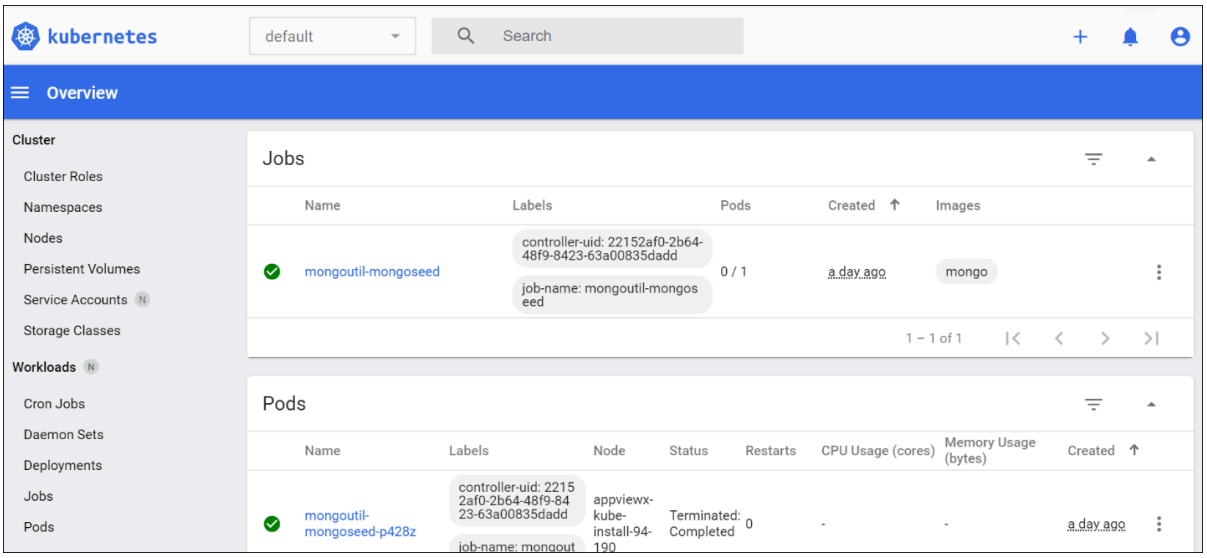


4. On the **Kubernetes Dashboard** login screen, enter the token value
5. Click **Sign In**.

> **Note:** You can find the token value from the **.appviewx_configuration** file.

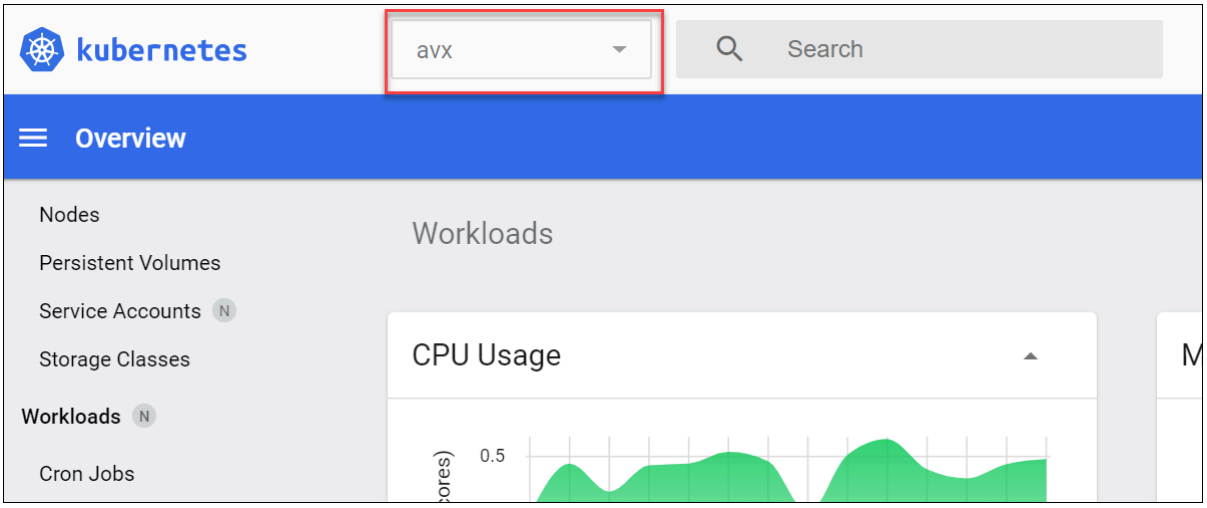


After you log in, you can access the Kubernetes management console and manage AppViewX components.



Viewing the POD Status

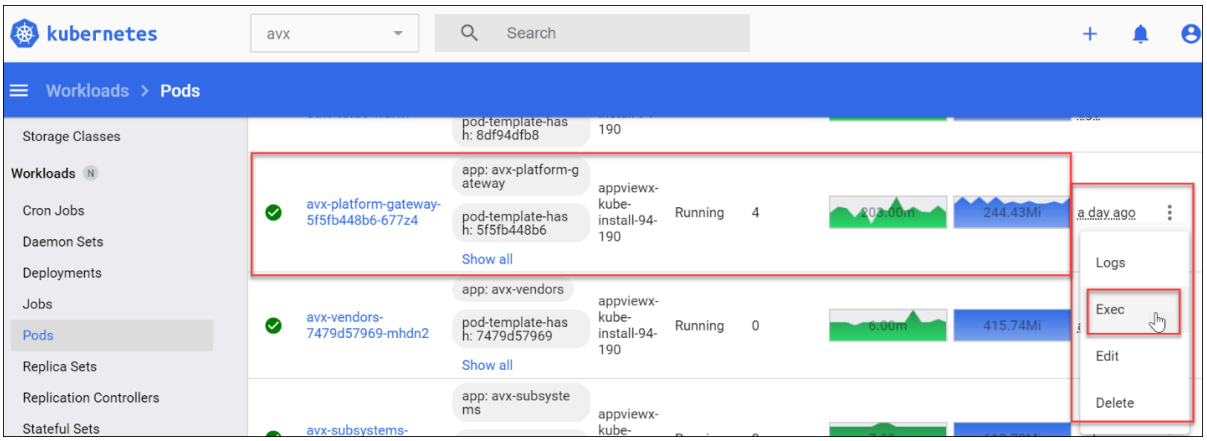
1. Open the Kubernetes management console.
2. Select a namespace from the top list.
3. Select **Pods** on the left menu.



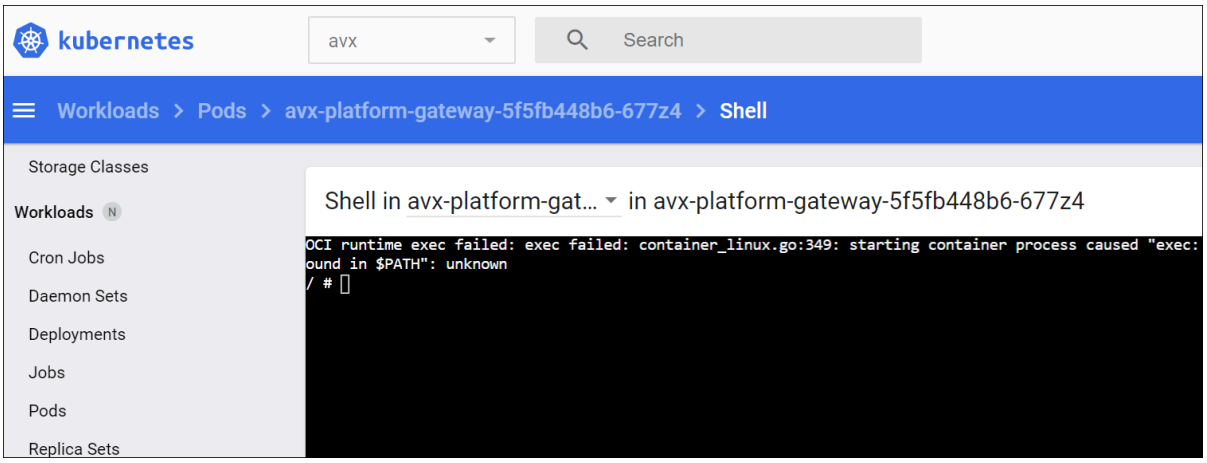
Accessing the POD Console

1. Open the Kubernetes management console.
2. Select the required namespace.
3. Under **Workloads**, click **Pods**.

   The **Pods** page is displayed.
4. Click on the three dots next to the pod and select **Exec**.

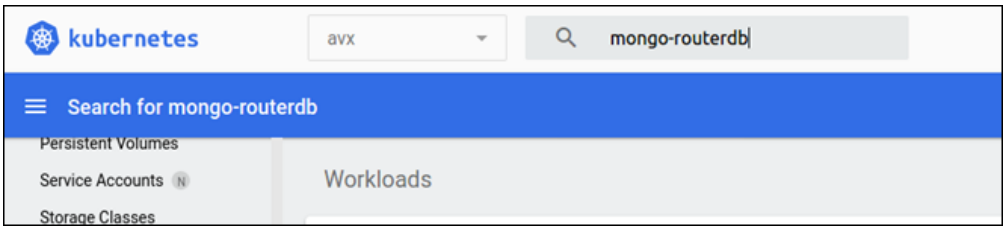


The Pod command line shell is displayed.

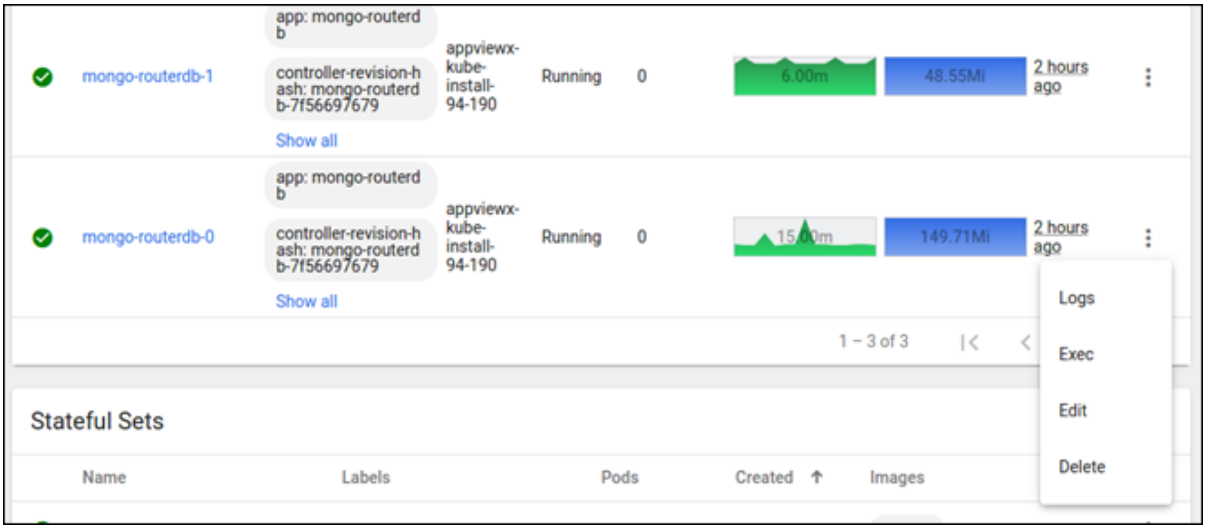


Accessing the Database Command Line

1. Open the Kubernetes management console.
2. Select **avx** in the namespace.
3. Search for **mongo-routerdb**.



4. Click on the three dots next to mongo-routerdb-0 pod and select **Exec**.



5. To launch the mongo db prompt, execute the following command: `<mongo>`
6. Execute the following command:

```
<use admin>
```

7. Execute the following command:

```
db.auth("admin","<mongodbpasssword>")
```

> **Note:** The password can be taken from the value of the `appviewx_mongodb_password` variable from the `<INSTALLATION_PATH>/.appviewx_configuration` file.

Exporting a Database Collection

Collections serve as generic repositories that hold any data in key-value pair format. It acts as interfaces to enter and modify data into the AppViewX Mongo database. Data from collections is consumed as a part of the provisioning request process or by any other scripts that are triggered by AppViewX. The structure of the collections is based on the Mongo database.

To export a mongo database collection:

1. Login to Kubernetes dashboard UI with the token.
2. From the top section, select the **avx** namespace.
3. Click pods and search for **mongo-routerdb-0**.
4. Click on the three dots icon and select **Exec**.
5. To navigate to the logs directory, execute the following command:

   ```
   cd /appviewx/dependencies/logs
   ```

6. Check if export_collection directory is available. Otherwise, to create the directory, execute the following command:

   ```
   mkdir export_collection
   ```

7. To navigate to the export_collection directory, execute the following command:

   ```
   cd /appviewx/dependencies/logs/export_collection
   ```

8. To export the database collection, execute the following command: Change the fields highlighted in bold with the desired values according to your setup.

   ```
   mongoexport --username admin --password <password> --db=appviewx --collection=<collectionName> --out=<fileName>.json --authenticationDatabase admin
   ```

   This command will export the collection and the file will be available at the following location: /appviewx/dependencies/logs/export_collection

   Note: The exported file is also available at the following location on the host where the mongodb pod is running: INSTALLATION_PATH/appviewx/logs/export_collection

Applying Custom Pod Configurations

Objective

The objective of this chapter is to implement custom changes related to pod specifications and prevent any issues with overridden changes after upgrades.

Feature Specifications and Commands

The features and the respective commands are as follows:

1. Update pod affinity

```
./appviewx.sh --apply-affinities
```

2. To update node labels.

```
./appviewx.sh --apply-labels
```

3. Update memory allocations for particular application pod.

```
./appviewx.sh --apply-allocate-memory
```

4. Change the pipeline worker count for logstash conf

```
./appviewx.sh --apply-logstashCustoms
```

5. Change the replica count

```
./appviewx.sh --apply-replicas
```

6. Change the HPA value of deployment

```
./appviewx.sh --apply-hpa
```

Steps to Apply Custom Changes

1. Navigate to <installer>/scripts location.
2. Copy the **custom_changes.yaml.template** to **custom_changes.yaml**
3. Find the formatting below for all custom configurations.
   a. Pod affinities

```
affinities:
  <Plugin Name>:
    nodeAffinity:
      enable: false
      type: "required"
```

```
        key: "dummy"
        values:
          - "dummy"
      podAffinity:
        enable: false
        key: "app"
        values:
          - "dummy"
        namespaces:
          - "avx"
        topologyKey: "kubernetes.io/hostname"
      podAntiAffinity:
        enable: false
        key: "app"
        values:
          - "dummy"
        topologyKey: "kubernetes.io/hostname"
```

Note: All three types of affinities should be mentioned during the configuration. It is necessary to set the enable field to *true* or *false* according to use case.

b. Node labels

```
node_labels:
  <node name>:
    ingress: "true"
```

c. Pod memory allocation

```
memoryAllocations:
    avx_subsystems:
      xms: "1g"
      xmx: "2g"
      memoryRequest: "100Mi"
      cpuRequest:  "100"
```

Note: xms and **xmx** fields are mandatory in case of custom memory allocations. After applying changes on the plugin you must re-deploy the same plugin in your setup.

d. Pipeline worker count for logstash conf

```
logstash_customs:
    pipelineWorkerCount: "5"
```

e. Replica count

```
replication:
    <pod_name>:
      count: "2"
```

f. HPA value of deployment

```
horizontalPodAutoScalling:
    avx_subsystems_sync:
      maxReplicas: "3"
      minReplicas: "1"
      cpu: "300"
      memory: "300"
```

Note: memory for HPA is optional parameter, you may skip it during configuration.

At the time modifying the custom_changes.yaml file for custom configurations, the rules of yaml code should be intact.

1. Before you start writing the custom configurations, ensure the three dashes “---” are present on the top of very first line of the custom configuration.
2. Field and its scope must be accurate.

Post the changes in **custom_changes.yaml**, installing the plugins will apply all custom changes.

External Certificate for Kubernetes

The section describes the steps to update the external certificate authority for the Kubernetes kubeadm. It contains the certificate specifications for the different certificates to be generated since .p12 is the only file format that is supported.

Certificate Specifications

Certificates must be generated individually for each of the common names listed in the table below. All master nodes (IP address and hostname) listed in the table must be added in the SAN of the certificates for a multi-node environment.

Table - Common name and IP address

Common Name	Type	O (in Subject)	SAN (refer notes below)	Parent CA	Cert and Location
kube-etcd	server	-	<master_hostnames>, <master_Host_IPs>, <kube_api_address>, localhost, 127.0.0.1,<service_ip>	etcd-ca	etcd/server.crt,etcd/server.key
kube-etcd-peer	server	-	<master_hostnames>, <master_Host_IPs>, <kube_api_address>, localhost, 127.0.0.1, <service_ip>	etcd-ca	etcd/peer.crt,etcd/peer.key
kube-etcd-healthcheck-client	client	-	-	etcd-ca	etcd/healthcheck-client.crt,etcd/healthcheck-client.key
kube-apiserver-etcd-client	client	system:masters	-	etcd-ca	pki/apiserver-etcd-client.key, pki/apiserver-etcd-client.crt
kube-apiserver	server	-	<master_hostnames>, <master_Host_IPs>,	kubernetes-ca	pki/apiserver.key,

Table - Common name and IP address (continued)

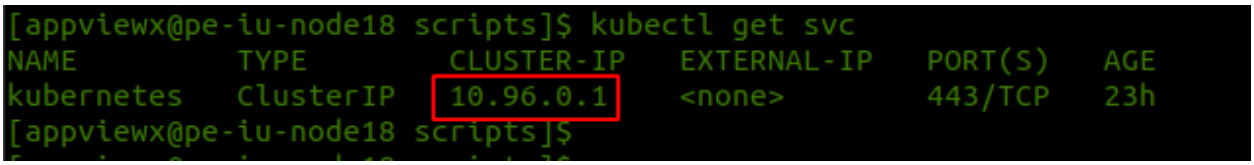
Common Name	Type	O (in Subject)	SAN (refer notes below)	Parent CA	Cert and Location
			<kube_api_address>, localhost, 127.0.0.1,<service_ip>, kubernetes, kubernetes.default, kubernetes.default.svc, kubernetes.default. svc.cluster, kubernetes.default. svc.cluster.local		pki/ apiserver.crt
kube-apiserver-kubelet-client	client	system:masters	-	kubernetes-ca	pki/apiserver-kubelet-client.key, pki/apiserver-kubelet-client.crt
front-proxy-client	client	-	-	kubernetes-front-proxy-ca	client.key,pki/front-proxy-client.crt
kubernetes-admin	client	system:masters	-	kubernetes-ca	admin.crt, admin.key
system:kube-controller-manager	client	-	-	kubernetes-ca	controller-manager.crt, controller-manager.key
system:kube-scheduler	client	-	-	kubernetes-ca	scheduler.crt, scheduler.ke
system:node:<hostname>	client	system:nodes	-	kubernetes-ca	kubelet.crt, kubelet.key

SAN values are as follows:

<master_hostnames> Hostname of the master.

<master_Host_IPs> IPs of the master.

<service_ip> The service IP can be obtained by executing the `kubectl get svc` from any node.



<kube_api_address> Load balancer for kube apiserver (if configured).

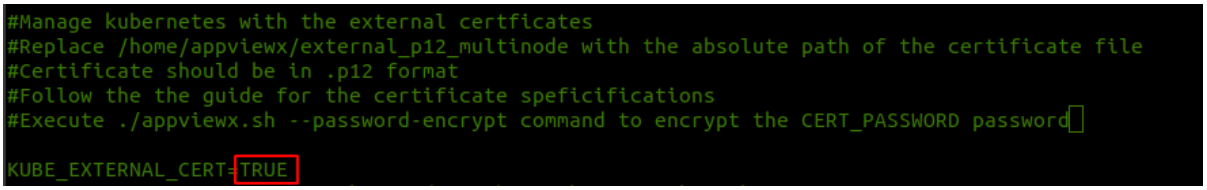
system:node:<hostname>The *kubelet.crt* should be generated for all the servers (master and worker) separately. For example, if the setup consists of three nodes, the certificates must be generated for all three nodes. The value of the **<hostname>** should be entered from the output of the `hostname` command.

Note:

- Enter only the `hostname` command output in SAN; do not enter the `hostname -f` command output.
- A general rule for the SAN of the certificate is to add the IP address in the *IP Address* field and the hostnames in the *DNS* field.
- An example of a wrong entry is shown below:

Entering All Certificates in the appviewx.conf File

1. Navigate to the <appviewx_installed_location>/appviewx_kubernetes/scripts and open *appviewx.conf* file.
2. To enable the external CA for kubeadm, set the value KUBE_EXTERNAL_CERT=TRUE

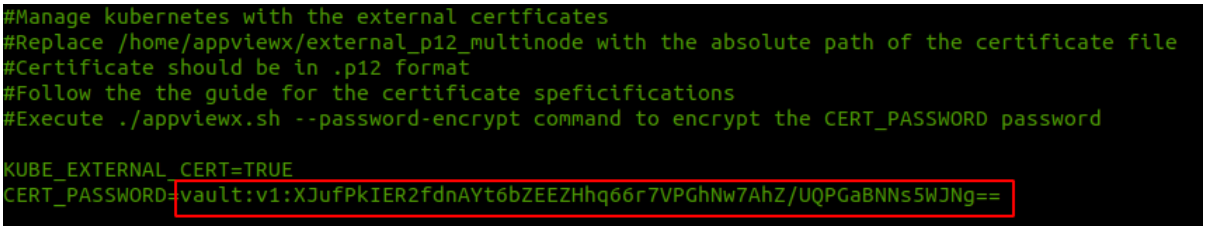


3. Enter the encrypted certificate password in the `CERT_PASSWORD` key. To encrypt the password,

a. Navigate to <appviewx_installer_location>/appviewx_kubernetes/scripts

b. Execute the command

```
./appviewx.sh --password-encrypt
```



4. Enter the absolute path of the certificate which is generated for the common name **kube-etcd** in KUBE_ETCD_PATH
5. Enter the absolute path of the certificate which is generated for the common name **kube-etcd-peer** in KUBE_ETCD_PEER_PATH
6. Enter the absolute path of the certificate which is generated for the common name **kube-etcd-healthcheck-client** in KUBE_ETCD_HEALTHCHECK_CLIENT_PATH
7. Enter the absolute path of the certificate which is generated for the common name **kube-apiserver-etcd-client** in KUBE_APISERVER_ETCD_CLIENT_PATH
8. Enter the absolute path of the certificate which is generated for the common name **kube-apiserver** in KUBE_APISERVER_PATH
9. Enter the absolute path of the certificate which is generated for the common name **kube-apiserver-kubelet-client** in KUBE_APISERVER_KUBELET_CLIENT_PATH
10. Enter the absolute path of the certificate which is generated for the common name **front-proxy-client** in FRONT_PROXY_CLIENT_PATH
11. Enter the absolute path of the certificate which is generated for the common name **kubernetes-admin** in KUBERNETES_ADMIN_PATH
12. Enter the absolute path of the certificate which is generated for the common name **system:kube-controller-manager** in KUBE_CONTROLLER_MANAGER_PATH
13. Enter the absolute path of the certificate which is generated for the common name **system:kube-scheduler** in KUBE_SCHEDULER_PATH

```
KUBE_ETCD_PATH=/home/appviewx/external_p12_multinode/kube-etcd_17_BA_FA_51_75_3A_CE_0D_E5_86_9B_20_A5_5A_4D_14_00_35_89_DD.p12
KUBE_ETCD_PEER_PATH=/home/appviewx/external_p12_multinode/kube-etcd-peer_51_A3_CE_5F_51_35_9A_72_3C_15_1B_54_BE_83_5C_25_ED_94_CB_C4.p12
KUBE_ETCD_HEALTHCHECK_CLIENT_PATH=/home/appviewx/external_p12_multinode/kube-etcd-healthcheck-client_31_54_F6_E1_3E_68_AB_B1_65_EC_02_99_E2_FB_A9_A7_5D_0C_D5_D3.p12
```

```
KUBE_APISERVER_ETCD_CLIENT_PATH=/home/appviewx/external_p12_multinode/kube-apiserver-etcd-client_27_FC_1E_94_84_0A_A8_90_D8_5D_99_5F_98_BB_B9_10_BF_E8_B5_4A.p12
KUBE_APISERVER_PATH=/home/appviewx/external_p12_multinode/kube-apiserver_19_33_6A_BE_B7_5E_F0_90_E6_2A_A8_F8_5D_C3_A0_2C_2A_78_BD_D1.p12
KUBE_APISERVER_KUBELET_CLIENT_PATH=/home/appviewx/external_p12_multinode/kube-apiserver-kubelet-client_7D_5F_B2_78_2C_51_03_D1_39_17_BF_FD_26_6E_A2_1A_60_93_1C_BF.p12
FRONT_PROXY_CLIENT_PATH=/home/appviewx/external_p12_multinode/front-proxy-client_61_97_2B_D9_E8_13_2B_24_3F_7E_85_B3_1A_F9_3A_AF_10_4C_5F_45.p12
KUBERNETES_ADMIN_PATH=/home/appviewx/external_p12_multinode/kubernetes-admin_2D_A0_1B_5E_A0_CF_27_2E_6B_9C_34_02_D9_E0_CA_60_95_BD_92_E0.p12
KUBE_CONTROLLER_MANAGER_PATH=/home/appviewx/external_p12_multinode/system_kube-controller-manager_31_32_15_2E_5F_4A_9C_B9_0E_2A_11_9B_CE_15_AA_59_5D_B7_FC_D1.p12
KUBE_SCHEDULER_PATH=/home/appviewx/external_p12_multinode/system_kube-scheduler_6A_FF_10_E1_F1_C9_9F_3C_0F_9D_82_88_18_38_EB_01_FB_3D_02_70.p12
```

14. Enter the **Kubelet certificates** in a colon ':' separated format, such as *<hostname>:<kubelet_certificate.p12>*. There should not be any spaces and also no colon (:) in the certificate file name.

Note:

a. If the kubelet certificate is generated for the host **pe-iu-node20.lab.appviewx.net**, the entry should be in the format **KUBELET_CERT_PATH=<hostname>:<absolute certificate file path>**. The entry for the host would be `KUBELET_CERT_PATH=pe-iu-node20.lab.appviewx.net:/home/appviewx/external_p12_multinode/system_node_pe-iu-node20.lab.appviewx.net.p12`

b. Enter all certificates that match the hosts in a comma-separated format, as given in the example below:

```
KUBELET_CERT_PATH=pe-iu-node20.lab.appviewx.net:/home/appviewx/external_p12_multinode/system_node_pe-iu-node20.lab.appviewx.net.p12,pe-iu-node16.lab.appviewx.net:/home/appviewx/external_p12_multinode/system_node_pe-iu-node16.lab.appviewx.net.p12,pe-iu-node17.lab.appviewx.net:/home/appviewx/external_p12_multinode/system_node_pe-iu-node17.lab.appviewx.net.p12,pe-iu-node18.lab.appviewx.net:/home/appviewx/external_p12_multinode/system_node_pe-iu-node18.lab.appviewx.net.p12,pe-iu-node19.lab.appviewx.net:/home/appviewx/external_p12_multinode/system_node_pe-iu-node19.lab.appviewx.net.p12,pe-iu-node20.lab.appviewx.net:/home/appviewx/external_p12_multinode/system_node_pe-iu-node20.lab.appviewx.net.p12,pe-iu-node21.lab.appviewx.net:/home/appviewx/external_p12_multinode/system_node_pe-iu-node21.lab.appviewx.net.p12
```

> **Warning:** Entering wrong certificates in the paths mentioned above will compromise the functioning of the application.

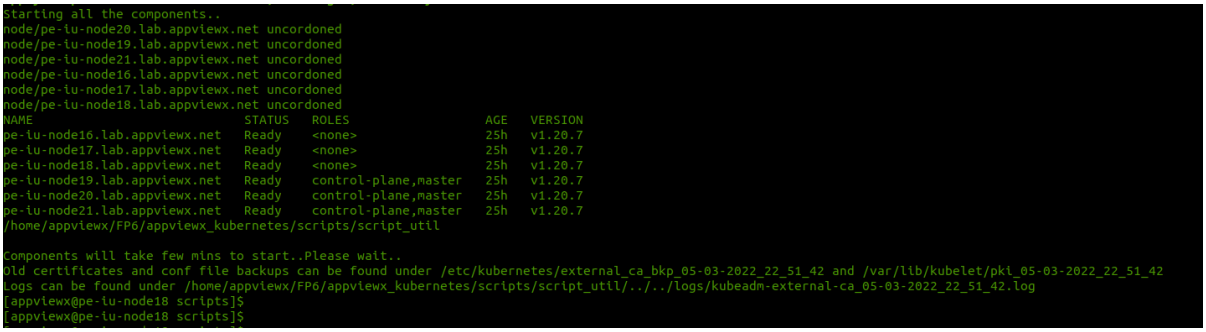
15. After adding all the certificate entries in the *appviewx.conf*
    a. Navigate to the **<appviewx_installer_location>/appviewx_kubernetes/scripts**
    b. Execute the command `./appviewx.sh --enable-kube-external-ca`

```
[appviewx@pe-iu-node18 scripts]$ ./appviewx.sh --enable-kube-external-ca
MAC verified OK
MAC verified OK
MAC verified OK
MAC verified OK
MAC verified OK
MAC verified OK
MAC verified OK
MAC verified OK
MAC verified OK
MAC verified OK
MAC verified OK
MAC verified OK
MAC verified OK
```

16. The command prompt for the passwords of all the nodes once the validations are completed. Enter the passwords, and hit the keyboard **Enter** key to proceed further.

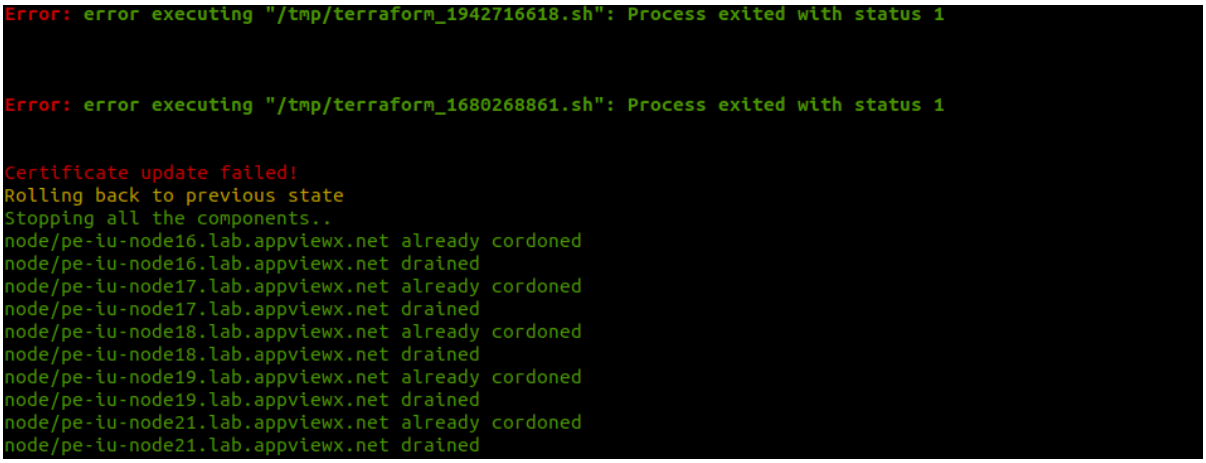
```
node/pe-iu-node20.lab.appviewx.net drained
NAME                          STATUS                     ROLES                  AGE   VERSION
pe-iu-node16.lab.appviewx.net   Ready,SchedulingDisabled   <none>                 25h   v1.20.7
pe-iu-node17.lab.appviewx.net   Ready,SchedulingDisabled   <none>                 25h   v1.20.7
pe-iu-node18.lab.appviewx.net   Ready,SchedulingDisabled   <none>                 25h   v1.20.7
pe-iu-node19.lab.appviewx.net   Ready,SchedulingDisabled   control-plane,master   25h   v1.20.7
pe-iu-node20.lab.appviewx.net   Ready,SchedulingDisabled   control-plane,master   25h   v1.20.7
pe-iu-node21.lab.appviewx.net   Ready,SchedulingDisabled   control-plane,master   25h   v1.20.7
/home/appviewx/FP6/appviewx_kubernetes/scripts/script_util
Please enter appviewx password of master:pe-iu-node20.lab.appviewx.net :
Please enter appviewx password of master:pe-iu-node21.lab.appviewx.net :
Please enter appviewx password of master:pe-iu-node19.lab.appviewx.net :
Please enter appviewx password of absecon:pe-iu-node16.lab.appviewx.net :
Please enter appviewx password of antartica:pe-iu-node17.lab.appviewx.net :
Please enter appviewx password of antartica:pe-iu-node18.lab.appviewx.net :
null_resource.ssh_connectivity[5]: Creating...
null_resource.ssh_connectivity[3]: Creating...
null_resource.ssh_connectivity[2]: Creating...
null_resource.ssh_connectivity[4]: Creating...
null_resource.ssh_connectivity[1]: Creating...
null_resource.ssh_connectivity[0]: Creating...
null_resource.ssh_connectivity[3]: Provisioning with 'remote-exec'...
null_resource.ssh_connectivity[2]: Provisioning with 'remote-exec'...
```

The following message is displayed on the successful completion of the execution:



Rollback Steps For Failure in Certificate Updates

This section describes the commands that can be executed to restore the certificates and config files to their previous state, in the event of a certificate update failure.



Note: The pods can either be in the **Init:CrashLoopBackOff** state or the **Pending** state.

1. **Init:CrashLoopBackOff:** If the pod is in this state, delete the pods by executing the command

   ```
   kubectl delete pod <podname> -n <namespace> –force
   ```

2. **Pending:** If the pod is in this state, execute the commands in the order mentioned below:

   a.
   ```
   kubectl scale –replicas=0 deploy/<component name> -n <namespace>
   ```

   b.
   ```
   kubectl get pods --all-namespaces | awk '{if ($4=="Terminating") print "kubectl delete pod " $2 " -n " $1 " --force --grace-period=0 ";}' | sh > /dev/null 2>&1
   ```

   c.
   ```
   kubectl scale –replicas=3 deploy/<component name> -n <namespace>
   ```

   Replicas can be changed based on the initial setup.

Uninstalling AppViewX

Users can uninstall AppViewX when they want to migrate into another environment. They can also uninstall AppViewX when it is no longer required.

To uninstall an application package safely:

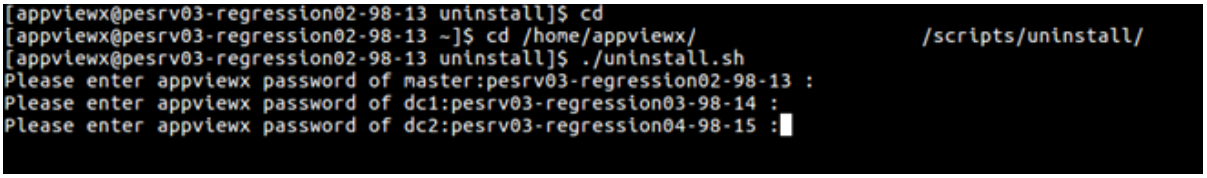
1. Open the terminal window.
2. To navigate to the **appviewx_kubernetes** directory, execute the following command:

```
cd /home/appviewx/appviewx_kuberbetes/scripts/uninstall
```

3. To start the uninstallation process, execute the following command:

```
/uninstall.sh
```

4. Enter the node's credentials when prompted.



5. Reboot all the nodes after completion of the AppViewX uninstallation.

• Uninstalling AppViewX

Uninstalling AppViewX

Users can uninstall AppViewX when they want to migrate into another environment. They can also uninstall AppViewX when it is no longer required.

To uninstall an application package safely:

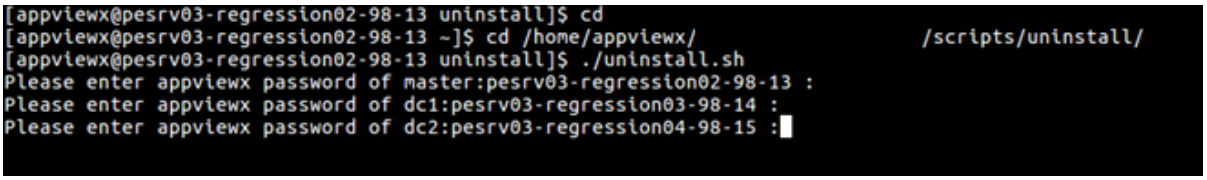
1. Open the terminal window.
2. To navigate to the **appviewx_kubernetes** directory, execute the following command:

```
cd /home/appviewx/appviewx_kuberbetes/scripts/uninstall
```

3. To start the uninstallation process, execute the following command:

```
/uninstall.sh
```

4. Enter the node's credentials when prompted.



5. Reboot all the nodes after completion of the AppViewX uninstallation.

• Uninstalling AppViewX

Troubleshooting

This section lists the issues encountered with AppViewX.

Whenever the AppViewX installation fails, you will get an error stating that some script execution failed.

- **Pre requisites not met**

  Please check for all the items below.

  - port not opened
  - insufficient disk/CPU
  - time not in sync
  - packages not found
  - hostname incorrect in configuration
  - etc
- **Error while installing the docker**

  If a customer brings in a custom OS, the Linux packages that AppViewX includes with the installer may not be compatible with the OS. In such situations, you may need to install the appropriate package to continue. This can be observed from the log messages that indicate an error while installing a package.
- **Error while installing the docker**

  Occasionally, we have observed intermittent errors from the OS during the installation of Docker. If you encounter an error at this stage, please attempt to uninstall the application, reboot all nodes, and then proceed with the installation.
- **Docker gets uninstalled from the CAGateway**

  Root cause: Although we removed the "uninstall docker" commands from our scripts, we discovered that Docker relies on containerd, which is used as a runtime in the product. The scripts also include steps to remove containerd in the install, uninstall, and upgrade scripts, which cannot be avoided. This

ultimately results in the removal of Docker as well. Additionally, the containerd version used in the product conflicts with the pre-existing containerd version of Docker on the server.

Docker and the AppViewX application cannot co-exist in the same server as it is tightly coupled with containerd. The manually installed docker will be removed during every maintenance activity such as install, uninstall and infra upgrade.

- **Context deadline exceeded in consul after the FP3 patching process**

  For setups with high network latency or slow I/O, after the FP3 patch process, the consul may be stuck in 1/2 stage, causing the vault to go in a crash loop back. If you encounter this, check the consul logs using the command

  ```
  kubectl logs consul-consul-server-0 -n avx
  ```

  If the logs specify “context deadline exceeded,” then increase the timeout in consul by the following steps:

  1. Navigate to <installer location>/appviewx_kubernetes/yaml/appviewx_vaut/consul/chart/vaules.yaml
  2. Edit **consulAPITimeout: 5s** (old value) to **consulAPITimeout: 10s** (new value)
  3. Save the changes.

  Note: Increase this timeout only based on the latency.

- **Error while initializing the kube master/worker**

  In certain cases, when uninstallation does not clean up the data properly, we may observer errors while initializing kube master and worker. In such cases, perform an uninstall, reboot all the nodes and then go ahead with the install. Additionally, there are cases where the installation fails due to port connectivity issues. If a failure occurs in this stage, check if ports 6443, 10250, 2379 and 2380 are opened properly.

- **Error while initializing the mongodb chart**

  This specific error occurs after a timeout of 5 minutes to initialize the mongodb charts. This error occurs when the pods are not able to communicate between themselves. Use the following commands to verify that:

  ```
  kubectl describe statefulset -n avx mongo-shareddb
  ```

  For any connectivity issues, the output of this command will display the specific error stating connection timed out.

- **Node is enabled with IPv6 but the application is not.**

Verify the output of the command:

```
ifconfig | grep -i inet6
```

If an IPv6 address is displayed, it is necessary to enable IPv6 in the appviewx.conf file. Failure to do so may result in communication issues.

- **IP in IP tunnelling is not enabled**

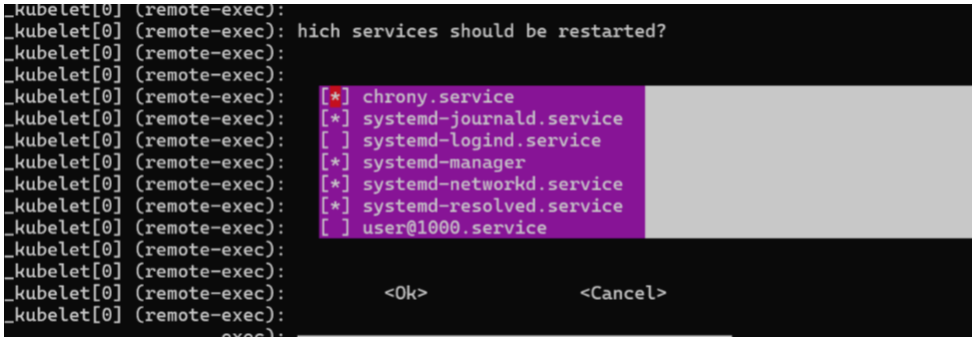
  If the IP in IP traffic is disabled, which means that the IPv4 protocol is not permitted, we will encounter the same problem. The prerequisite check script does not identify this, so we need to verify it separately to confirm.

- **Error while installing the AppViewX plugins**

  If an error occurs during the installation of AppViewX plugins, it is likely due to an error in the configuration file. You may observe and error such as `Upload failed: scp`, in such cases re-trigger plugins_install.sh to install the plugins. Likewise, ensure to review the configuration file carefully and proceed with the execution of plugins_install.sh to install only the plugins.

- **Issue with uninstall script hanging on Ubuntu 22.04 due to needrestart command**

  Root cause: It is observed that in some Ubuntu setups, the uninstall process gets stuck due to a prompt that waits indefinitely, causing the script to hang. This issue is new in Ubuntu 22.04 and is related to the needrestart command, which is now part of the **apt-get** upgrade process. By default, needrestart is set to "*interactive*" mode, leading to interruptions in scripts.



Remediation: To change this behavior, edit the /etc/needrestart/needrestart.conf file by modifying the line from

```
#$nrconf{restart} = 'i';
```

to,

```
$nrconf{restart} = 'a';
```

After making this change, subsequent runs should not encounter this issue.

Steps to Change Mongodb Password

This section walks you through the steps to be taken to change the MongoDB Password.

- Untar Scripts Tarball

Untar Scripts Tarball

- Command for changing MongoDB Password
- User command

Command for changing MongoDB Password

- For single-node setup
- For multi-node setup/For FP5

For single-node setup

Use command

```
echo "db.getSiblingDB(\"admin\").changeUserPassword(\"admin\", \"<newpass>\")" |
        kubectl exec -it mongodb-0 -n avx -- mongo --authenticationDatabase admin -u admin –p
        <currentPass>
```

For multi-node setup/For FP5

Use command

```
echo "db.getSiblingDB(\"admin\").changeUserPassword(\"admin\", \"<newpass>\")" |
kubectl exec -it routerdb-0 -n avx -- mongo --authenticationDatabase admin -u admin –p <currentPass>
```

Note: It is necessary to cross-check whether your password is changed or not. Copy **other_user_internal.pem** file into scripts directory from **appviewx_kubernetes/scripts** directory.

User command

- For single-node setup
- For multi-node setup/For FP5
- Trigger Script

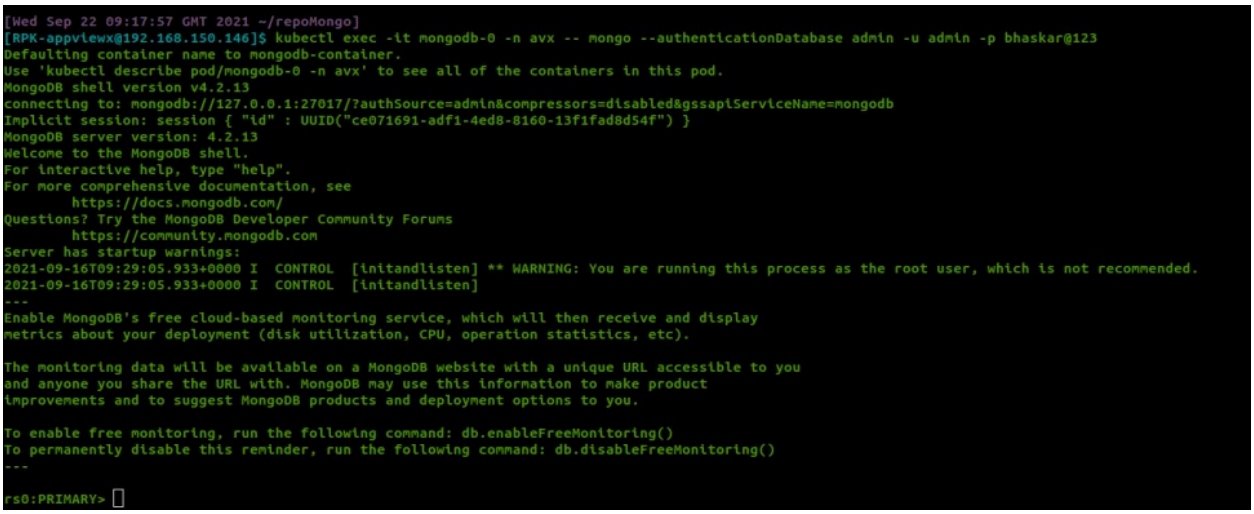
For single-node setup

```
kubectl exec -it mongodb-0 -n avx -- mongo --authenticationDatabase admin -u admin -p

<newpass>
```

For multi-node setup/For FP5

```
kubectl exec -it routerdb-0 -n avx -- mongo --authenticationDatabase admin -u admin –p

<newpass>
```

If you are successfully logged in, it will be displayed as shown in the below image:



Trigger Script

```
./scripts/appviewx.sh --password
```

After successful execution of script, delete all pods.

Note: Make sure the vault is up and running.

Disable Kex Algorithm Guide

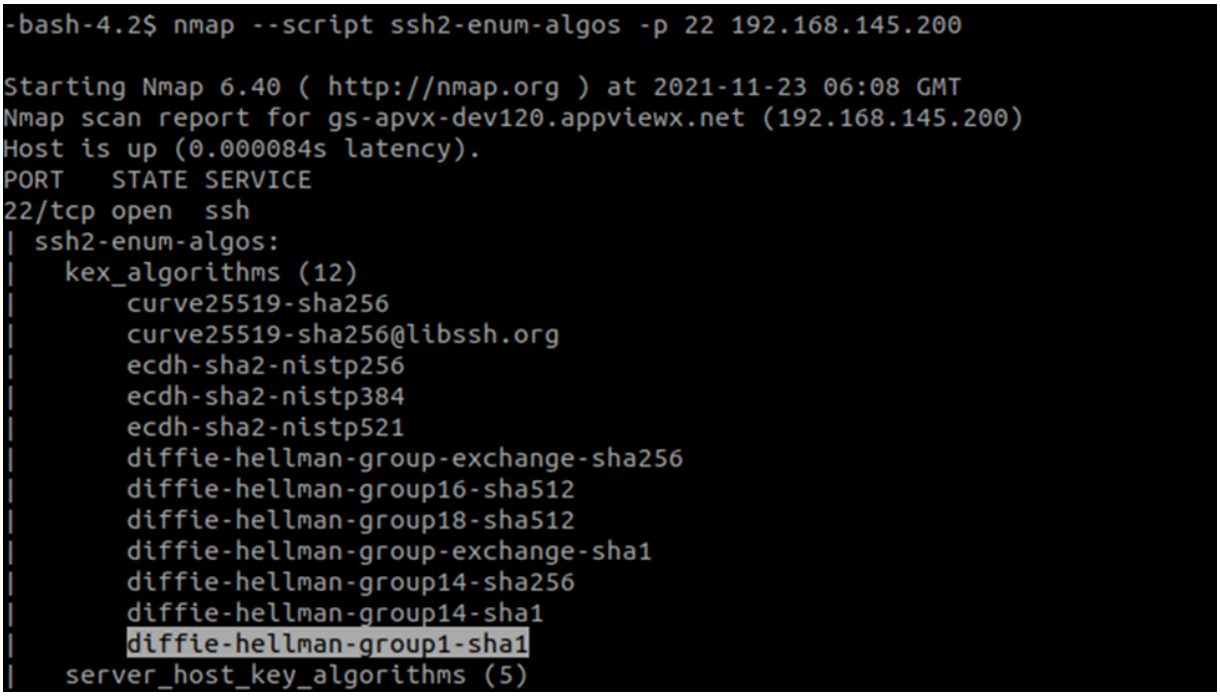
This guide is designed to help disable kex Algorithm `diffie-hellman-group1-sha1` in systems with 2022.1.0 ovas.

Steps to Disable Kex Algorithm

1. Run command: `nmap --script ssh2-enum-algos -p 22 <ip address>`

   Note: Replace `<ip address>` in the command with the actual IP.

   The deprecated algorithm `diffie-hellman-group1-sha1` will be active. Refer the following image for the same.



2. Run command: `sudo vi /etc/ssh/sshd_config`
   a. `sshd_config` file will open.

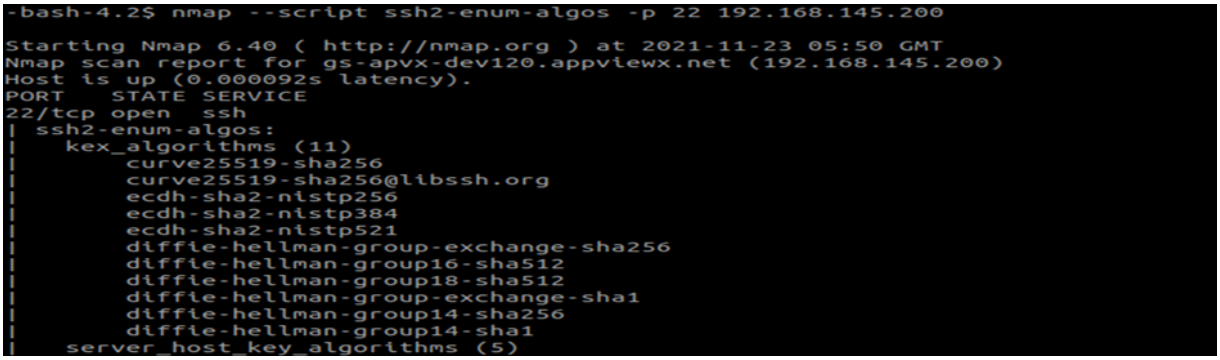
   Note: Make sure you have a KexAlgorithms list. This list should not include `diffie-hellman-group1-sha1` entry.

> **Note:** Below mentioned is the reference text used in the above image:`KexAlgorithms curve25519-sha256, curve25519-sha256@libssh.org,ecdh-sha2-nistp256,ecdh-sha2-nistp384,ecdh-sha2-nistp521,diffie-hellman-group-exchange-sha256,diffie-hellman-group16-sha512,diffie-hellman-group18-sha512,diffie-hellman-group-exchange-sha1,diffie-hellman-group14-sha256,diffie-hellman-group14-sha1`

b. Save the changes and exit from the file.

3. Run command: `udo systemctl restart sshd`
4. Execute command: `nmap --script ssh2-enum-algos -p 22 <ip address>`

> **Note:**
>
> a. Replace `<ip address>` with the actual IP.
>
> b. Confirm that `diffie-hellman-group1-sha1` is disabled. Refer the following image for the same.



Migrating CentOS to Ubuntu/RHEL

> **Attention:** This document is only for customers with CentOS on AppViewX v2020.3.0 FP10 and FP11 and upgrading to v2023.1.0 FP2. You may ignore this if you are currently using AppViewX v2022.1.0 (FP1-FP3), v2023.1.0 and v2023.1.0 FP1.

AppViewX would like to notify their customers about an important update regarding the Operating System (OS) support for AppViewX installations. Presently with AppViewX 2020.3.10 and 2022.1.3, we support three Operating Systems: CentOS & RHEL 7, Ubuntu 20.04 LTS, RHEL 8x

However, as some may be aware, CentOS is rapidly approaching its end-of-life (EOL) in June 2024, we need to adjust our AppViewX deployment strategy.

Starting from 2022.1.3 (Ganga FP3), we have transitioned from providing the OVAs with CentOS 7 to Ubuntu. This change ensures that our new customers receive continuous support without any complications. However, please be aware that existing customers can still utilize the CentOS OVA until it reaches its EOL in June 2024 but, we strongly encourage customers from CentOS 7 as soon as they are able to.

For a seamless transition, we advise our current customers who are using CentOS to migrate to either the Ubuntu OVA or bring their own Ubuntu/RHEL OS before CentOS reaches its end-of-life (EOL). Detailed migration steps can be found in the sections below.

Prerequisites

- New Nodes must have any of the following OS:
  - RHEL 8.5-8.8, 9.2, or 9.3
  - Ubuntu 20.04
- To list the nodes in the cluster, execute the command below and save the outputs for further reference

```
kubectl get nodes --show-labels
```

Note: If custom labels are detected add them to the custom_changes.yaml file. Refer to the chapter *Adding Custom Pod Configuration* of the section **Monitoring and Maintaining AppViewX** in the **Install, Upgrade, and Maintenance Guide**.

- To get the status of HPA, execute command below and save the outputs for further reference

```
kubectl get hpa
```

- Keep a backup of the iControl jar files available at location -/home/appviewx/appviewx/appviewx_dependencies/external_libs/iControl-13.1.0.jar for iControl to be done after the migration.
- Check for any other custom changes that may have been done specifically for the customer
- Check the enabled plugins in appviewx.conf file of the old installer (previous setup), in case there are plugins that are not present but are required to be installed in the new setup, please add them, example

- avx_pkiaas_cert_ocsp_generator
- avx_pkiaas_cert_ocsp_server, avx_pkiaas_cert_ocsp_server
- avx_platform_hsm

AppViewX 2020.3.0 FP10 (CentOS) to v2023.1.0 FP3 (Ubuntu/RHEL)

Overview

1. Take the mongodb and vault backup from the old CentOS environment.
2. Install AppViewX v2023.1.0 FP3 in the new environment with Ubuntu/RHEL.
3. Restore mongodb in the new environment.
4. Restore the vault in the new environment.
5. Trigger the plugins install.

The detailed steps are as follows:

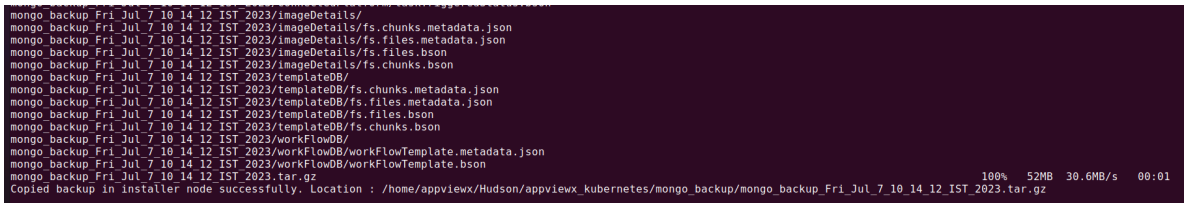
1. In the CentOS environment, take mongodb and vault backup as follows

   a. Navigate to installer node's scripts directory - <appviewx_installer_location>/appviewx_kubernetes/scripts/

   b. Initiate a backup of the AppViewX Database using the following command

   ```
   ./mongo_backup.sh
   ```

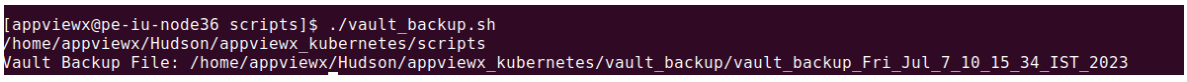
   After the backup is taken successfully, the database's backup file and its location are displayed on the screen. Copy this backup file to a safe location for future reference.

   c. Initiate a backup of the Secrets Vault using the command

   ```
   ./vault_backup.sh
   ```

   After the backup is taken successfully, the vault's backup file and its location are displayed on the screen. Copy this backup file to a safe location for future reference.

2. If you have completed the backup process, proceed to install the new server with the chosen version on the supported operating system.

   a. Choose the preferred OS or OVA from the supported list.
      i. RHEL (8.7, 8.8, 8.10, 9.2, 9.3 or 9.4)
      ii. Ubuntu 20.04, 22.04 LTS
      iii. Ubuntu OVA (with Ubuntu 20.04 LTS)

   b. Once you have decided on the OS or OVA, install the AppViewX server by following the steps below:
      i. Install 2023.1.0 FP3 with OVA - Click here or
      ii. Install 2023.1.0 FP3 with installer - Click here

3. To restore mongodb in the new environment,

   a. Copy the backed up file(s) to the new environment's installer node.

   b. Navigate to the installer location's scripts directory - <appviewx_installer_location>/appviewx_kubernetes/scripts/

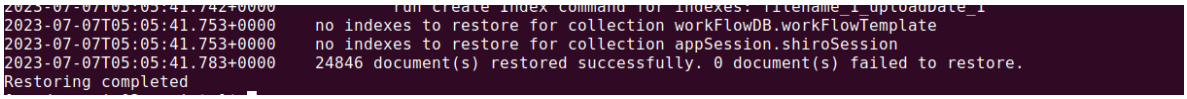
   c. Restore the database by triggering database restoration script. Execute the following command:

```
./mongo_restore.sh <location of mongo backup file>
```

   Example:

```
./mongo_restore.sh /home/appviewx/mongo_backup/mongo_backup_Fri_Jul_7_10_14_12_IST_2023.tar.gz
```

   d. Wait for the successful completion message.



4. Restore the Secrets Vault by triggering the vault restoration script by using the following command:

```
./vault_restore.sh -p <location of vault backup file> --removedek
```

   Example:

```
./vault_restore.sh -p /home/appviewx/vault_backup/vault_backup_Wed_Jul_28_05_50_40_UTC_2021 –removedek
```

```
Vault Restore Completed
node/iu03.lab.appviewx.net not labeled
configmap/avx-common-config patched
configmap/avx-common-config patched
configmap/avx-common-config patched
configmap/avx-common-config patched
configmap/avx-common-config patched (no change)
configmap/avx-common-config patched (no change)
configmap/avx-common-config patched (no change)
configmap/avx-common-config patched (no change)
node/iu03.lab.appviewx.net not labeled
NAME: cryptutilencrypt
LAST DEPLOYED: Fri Jul  7 10:39:55 2023
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
Successfully Updated DB with hash
Successfully restarted the pods
None
configmap/avx-common-config patched
configmap/avx-common-config patched
configmap/avx-common-config patched
configmap/avx-common-config patched
configmap/avx-common-config patched
configmap/avx-common-config patched
configmap/avx-common-config patched
configmap/avx-common-config patched
```

5. To install plugins,

   a. Navigate to the appviewx_kubernetes/scripts folder.

   b. Execute the following command

```
./plugins_install.sh
```

AppViewX 2020.3.0 FP10 (CentOS) to v2020.3.0 FP10 (Ubuntu/RHEL)

1. Take the mongodb and vault backup from the old environment with CentOS.
2. Install AppViewX v2023.3.0 FP10 in the new environment with Ubuntu/RHEL.
3. Restore mongodb in the new environment.
4. Restore the vault in the new environment.

Refer the detailed steps in the section above (only steps 1 to 4).

Post Upgrade Steps

The following actions must be taken to avoid any post-migration errors listed below.

1. **Loss of Mongo replica set priority configurations**

   During installation, mongodb is freshly set up with the latest upgraded versions. The previous replicaset configurations such as replicaset priorities will not be taken ahead and hence have to be re-configured. High latency customers must perform the following step:

   a. Configure the parameters OPTIMISE_ROUTING_FOR_LATENCY and PREFERRED_DEFAULT_DC in the appviewx.conf
   b. Re-trigger the `plugins_install.sh` to change the configurations.

2. **Custom changes**

   If the custom_changes.yaml, custom_vm_args.conf are present and updated in the custom changes, then the custom changes will be persistent. Any of the custom changes that may have been done specifically for the customer as noted in the prerequisites will not be present if the above mentioned files are not updated with this configuration.

3. **External web cert is not upgraded from 2022.1.0 to 2023.1.0**

   To update the external CA web certificate, execute the command below:

   ```
   ./appviewx.sh --update-web-cert
   ```

   The following prompts will be displayed:

   • Enter the absolute path of external cert file:
   • Enter the absolute path of external key file:

   Enter both the values to proceed. Once the cert upgrade is completed, restart the gateway and web.

4. Set the ELASTIC_ENABLE as True in the **Statistics Configuration**. There are two ways to do it, choose from either the command prompt (a) or from the management console UI (b).

   Note: Elastic insight must be reinstalled. Application upgrade will take care of installing insight if it's already there in the existing setup, but this won't be taken care automatically in the fresh install. Hence, install insight before step b.

   a. Execute the command

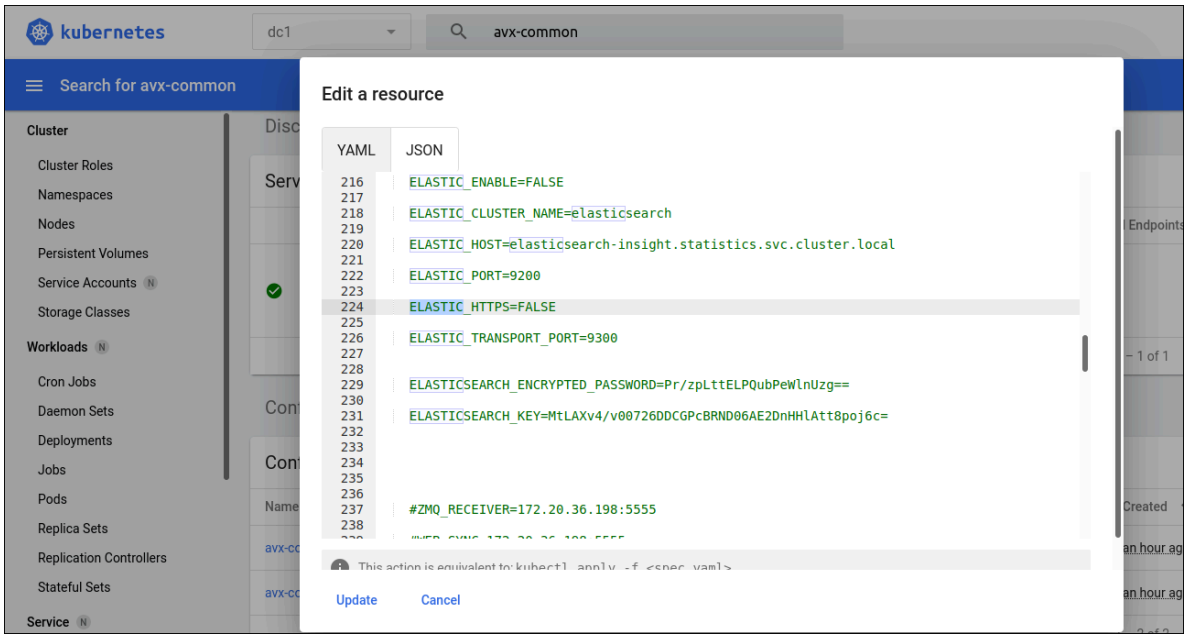
      ```
      kubectl edit configmaps -n <Datacenter1>
      ```

      Search for the keyword as Elastic and set ELASTIC_ENABLE as True and below params should have default values as below

      ```
      ELASTIC_ENABLE=TRUE

      ELASTIC_CLUSTER_NAME=elasticsearch
      ```

```
ELASTIC_HOST=elasticsearch-insight.statistics.svc.cluster.local

ELASTIC_PORT=9200

ELASTIC_HTTPS=FALSE

ELASTIC_TRANSPORT_PORT=9300
```

b. Login to management console >> Search for the namespace with the configured DC >> Search for avx-common-config in config maps >> Click on Edit and search for Elastic >> Set as True and give as update as shown below.



- Enabling HSM
- Elastic Restore

Enabling HSM

This is a required step if you have upgraded from v2020.3.0 to the latest v2023.1.0 FP2.

Prerequisites

- To configure Fortanix and Utimaco, the **.so** file and **config** file must be present in the current Appviewx version.
  - The **.so** file is essential for communicating with the HSM using the PKCS11 interface.
  - The **config** file facilitates communication between the HSM and Appviewx.

After the successful upgrade, proceed with the steps below to enable HSM.

1. Ensure the HSM pod is operational and running in the required datacenters and that the HSM node is specified in the appviewx.conf file, execute the following command:

```
kubectl get pods -A -o wide |grep hsm
```

2. From the command line interface, navigate to the properties folder path {APPVIEWX_INSTALLATION_PATH}/appviewx_dependencies/properties

For Fortanix

a. Open the HSM file using the following command:

```
vi hsm
```

b. Check and confirm if the HSM file has the following lines. If not, uncomment the following lines:

```
export FORTANIX_PKCS11_CONFIG_PATH= /appviewx/dependencies/hsm/fortanix/pkcs11.conf
```

```
echo "FORTANIX Config Path : $FORTANIX_PKCS11_CONFIG_PATH"
```

c. If the file is edited, restart the **avx-platform-hsm** pod, using the following commands:

```
kubectl get pods -n <namespace>
```

```
kubectl delete pods -n <namespace> <PodName> --force
```

For Utimaco

a. Open the HSM file using the following command:

```
vi hsm
```

b. Check and confirm if the HSM file has the following lines. If not, uncomment the following lines:

```
export CS_PKCS11_R2_CFG=/appviewx/dependencies/hsm/utimaco/cs_pkcs11_R2.cfg
```

```
echo "UTIMACO Config Path : $CS_PKCS11_R2_CFG"
```

c. If the file is edited, restart the **avx-platform-hsm** pod, using the following commands:

```
kubectl get pods -n <namespace>
```

```
kubectl delete pods -n <namespace> <PodName> --force
```

3. Once the HSM pod is back to running state, login to AppViewX and navigate to **Platform** > **Vault & Security** > **HSM**.
4. Access the required HSM.

Configuring Fortanix

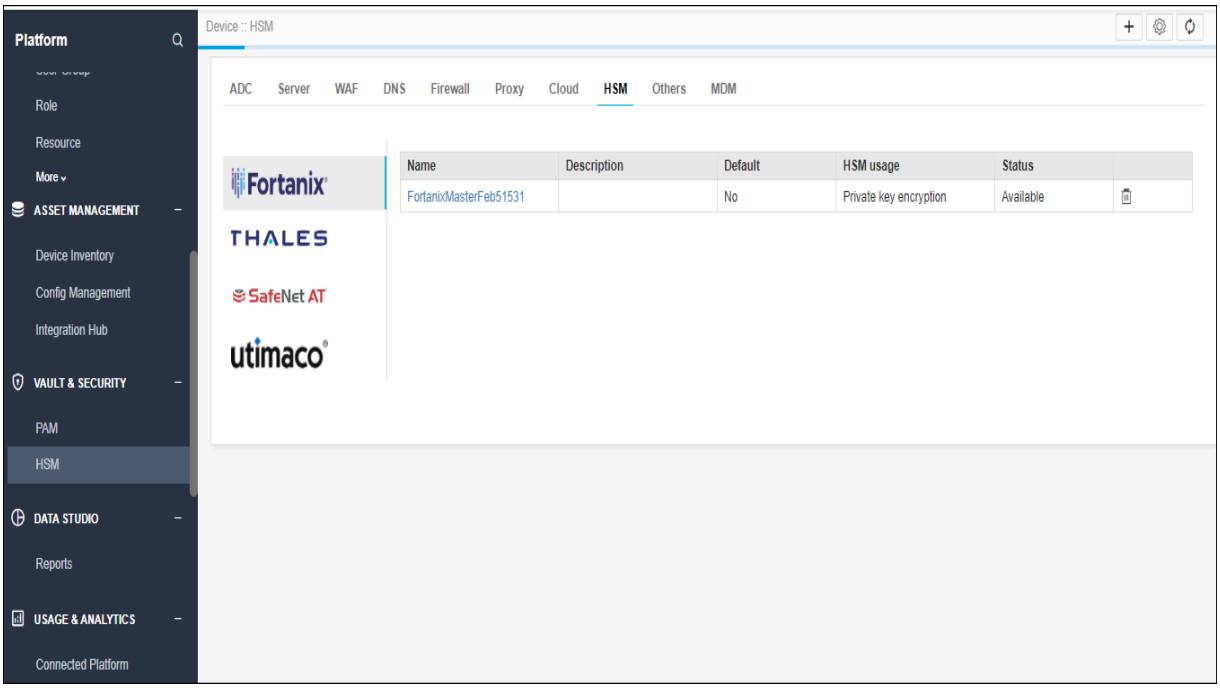
1. If the added HSM is Fortanix, upload the **.so** file and **config** file as follows:
   To upload the **.so** file,
   a. Click **Browse**.
   b. Navigate to the location of the .so file.
   c. Select the .so file and click **Open**.
   To upload the **.cfg** (config) file,
   a. Click **Browse**.
   b. Navigate to the location of the .cfg file.
   c. Select the .cfg file and click **Open**.
2. Choose the respective datacenter.
3. Update each HSM available in the inventory one by one, ensuring that the HSM is moved to the **Available** status.

Configuring Utimaco

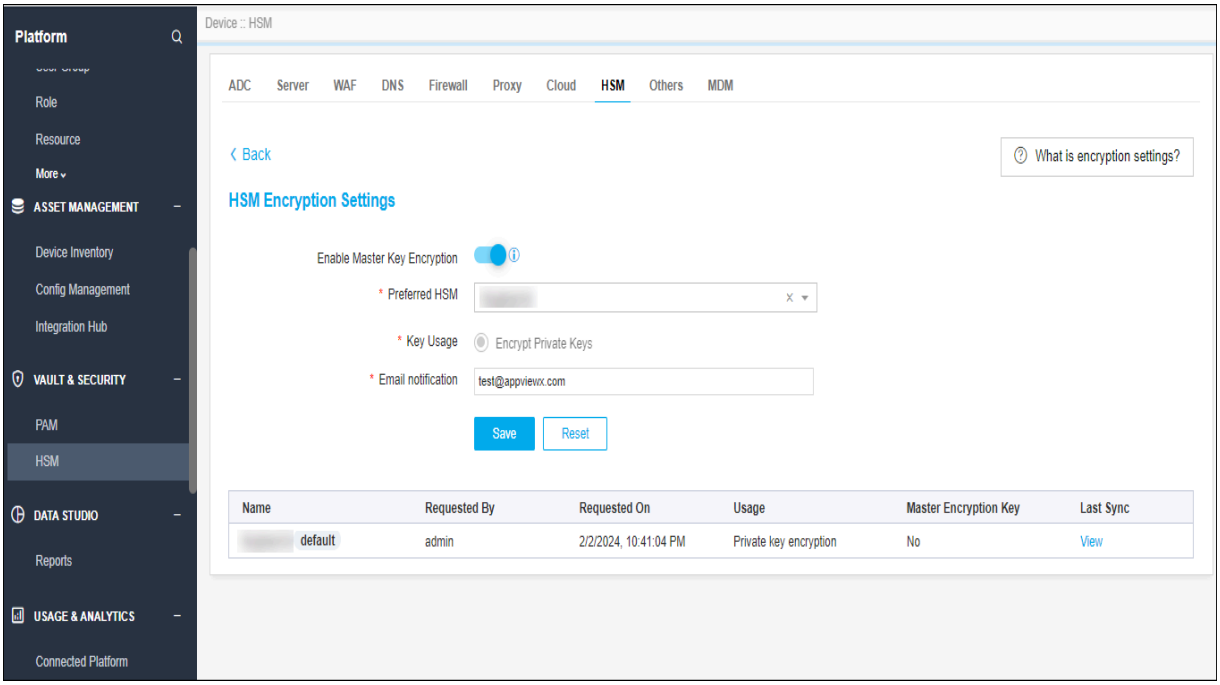
1. If the added HSM is Utimaco, upload the **.so** file and **config** file as follows:
   To upload the **.so** file,
   a. Click **Browse**.
   b. Navigate to the location of the .so file.
   c. Select the .so file and click **Open**.
   To upload the **.cfg** (config) file,
   a. Click **Browse**.
   b. Navigate to the location of the .cfg file.
   c. Select the .cfg file and click **Open**.
2. Choose the respective datacenter.
3. Update each HSM available in the inventory one by one, ensuring that the HSM is moved to the **Available** status.

Verifying/Modifying HSM Configuration for Private Key Encryption

1. On the top-right corner of the HSM inventory page, click the master encryption settings icon.



2. If the already added HSM has implementation type **private key generation** or **Both** and Set as “default” in 2020.3.0 Appviewx version then that HSM will be available in the Master encryption settings page as default HSM.



3. To receive emails for default HSM health status, configure the email address in the settings page.

> **Note:** For email use case SMTP settings should be available in **Platform** > **System Administration** > **SMTP**.

4. To change the new default HSM, add the new HSM with HSM usage as **Master key encryption** or **both**.
5. Once the HSM is moved to **Available** status it will be present in the **Preferred HSM** dropdown field of the master key encryption settings page.
6. Choose the new HSM and click **Save**.

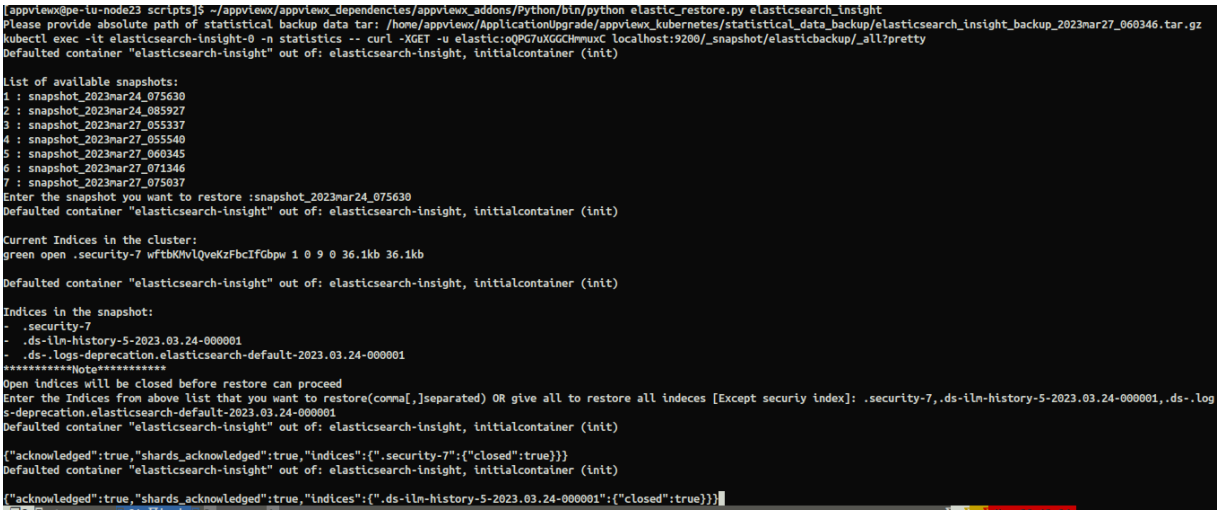
Elastic Restore

The script **elastic_restore.py** is used for restore. To manually perform the elastic restore,

1. Navigate to the scripts directory.
2. Run the elastic_restore.py script to restore the backup.

```
/home/appviewx/appviewx/appviewx_dependency/appviewx_addons/Python_Linux/bin/python elastic_restore.py elasticsearch_insight
```

3. Script will ask for the backup tar which was created manually. Provide the absolute path of the backup tar.



4. Script will list all the available snapshots that has date and time in the naming. Select the backup which you want to restore.
5. Provide the details of the indices you want to restore (follow the screenshot above).

OS Patching on Ubuntu for Multi-Node Environment

This section provides a step-by-step guide for applying OS patches in an Ubuntu environment while minimizing downtime in a multi-node, multi-data center (DC) Kubernetes setup. By following the rolling update procedure, OS patches can be applied without causing significant disruption to the overall environment. Ensure the GUI load balancer is properly configured to distribute traffic during the patching process to maintain high availability. As per our high-availability (HA) claim, a brief downtime of approximately 5 minutes is expected when the secondary DC nodes are down.

Prerequisites

- Ensure your system has access to the required package repositories by either enabling APT Proxy or whitelisting the AppViewX repository (**repo.appviewx.com**).
- For multi-master Kubernetes environments, configure the TCP load balancer to handle failover effectively, ensuring minimal downtime in case the primary master becomes unavailable during the upgrade.
- Verify that the installed user has **sudo** privileges.
- Take VM snapshots before proceeding with the OS patching.

OS Patching Procedure for Minimal Downtime

To minimize application downtime in environments spanning multiple data centers, it is recommended to perform a rolling update. This process involves stopping the application on one node at a time, applying the OS patches, and then restarting the node before moving to the next one.

1. Log in to the installer node and navigate to <appviewx installer directory>/scripts.
2. *Stop the application on the node* - Before applying the patch, stop the application on the specific node. This ensures that the node is not running any critical services during the patching process.
   Run the following command to stop the application:

   ```
   ./appviewx.sh --stop <node_name>
   ```

   Note: Replace **<node_name>** with the actual name of the node you are stopping.

3. *Apply the OS patch/security update* - Once the application is stopped, proceed with updating the operating system. Run the following commands to apply the necessary patches:
   Run the following command to stop the application:

   ```
   sudo apt update -y
   ```

   ```
   sudo apt upgrade -y
   ```

   This will update the package lists and apply all available OS security patches.
4. *Reboot the node* - After the patching is complete, reboot the node to ensure that all changes are applied properly.

   ```
   sudo reboot
   ```
5. *Start the application on the node* - After the node has rebooted, start the application back up using the following command:

   ```
   ./appviewx.sh --start <node_name>
   ```

   Note: Replace **<node_name>** with the actual name of the node.
6. *Repeat for the remaining nodes in the data center* - Repeat the above steps for all the nodes in the first data center. Ensure that each node is patched one at a time to avoid any application downtime within the data center.
7. *Move to the next data center* - Once all nodes in the first data center have been patched and verified to be working, proceed to the second data center and follow the same steps for each node.

Upgrading RHEL to v8.10/v9.4

This section provides instructions on how to perform an in-place upgrade from Red Hat Enterprise Linux (RHEL) 8.x to versions 8.10 and 9.4.

Prerequisites

Before you upgrade the OS version, perform the steps below.

1. **Take snapshot of existing setup** - VM snapshots of the current setup should be taken as a backup.
2. **Stop the application and services that are running** - Execute the commands in the order below in all the nodes, if you have a multi-node setup.

```
systemctl stop appviewx.service
```

```
systemctl stop kubelet.service
```

```
systemctl stop containerd.service
```

3. **Disable the following services** - Execute the commands in the order below in all the nodes, if you have a multi-node setup.

```
systemctl disable appviewx.service
```

```
systemctl disable kubelet.service
```

```
systemctl disable containerd.service
```

Upgrading RHEL to v8.10

This section provides instructions on how to perform an in-place upgrade from RHEL 8.x to 8.10.

1. Set the target OS Version using subscription-manager command below.

```
sudo subscription-manager release --set=8.10
```

2. Clear cache using the command below.

```
sudo dnf clean all
```

3. Check if any updates are available for the system. Run the command below.

```
sudo dnf check-update
```

4. To update package to the latest version that are available and resolvable, run the command below.

```
sudo dnf upgrade -y
```

5. Reboot the system using the command below.

```
sudo reboot
```

6. To check the upgraded version of the OS, run the command

```
cat /etc/os-release
```

7. (*Optional step*) If there are any kernel specific error messages write the kernel messages in Linux operating systems to standard output using the **dmesg** command.

```
dmesg | grep -i 'err|warn|cri'
```

Upgrading RHEL to v9.4

Important:

- If your current OS version is 8.x, upgrade to version 8.10 first, and then follow the steps to upgrade to version 9.4.
- If you've already upgraded to version 8.10, follow the prerequisites and then proceed with the upgrade to version 9.4.

Refer to the Upgrading from RHEL 8 to RHEL 9section on the Red Hat documentation website.

Post-Upgrade Steps and Troubleshooting

Prerequisite for RHEL 9.4

Note: After you have upgraded the OS to RHEL 9.4 and before you start using AppViewX, perform the steps below to update the **sysctl.conf** file.

1. To update the **sysctl.conf** file, execute the command below.

```
sudo vi /etc/sysctl.conf
```

2. Add these two lines (parameters) to the end of the file.

```
fs.inotify.max_user_instances=8192
```

```
fs.inotify.max_user_watches=524288
```

3. Save and exit.
4. Now, run the command below.

```
sudo sysctl -p
```

Starting the Services and Pods

Perform these steps after you upgrade to RHEL 8.10 or 9.4

1. Start the infra and application services using the following commands

   a. To enable the services, run the command below.

   ```
   systemctl enable containerd.service
   ```

   ```
   systemctl enable kubelet.service
   ```

   ```
   systemctl enable appviewx.service
   ```

   b. To start the services, run the command below.

   ```
   systemctl start containerd.service
   ```

   ```
   systemctl start kubelet.service
   ```

   ```
   systemctl start appviewx.service
   ```

2. Restart all the pods by executing the command below.

   ```
   kubectl delete po –all -A –force
   ```

   Note: Wait until all the pods are in the running state.

3. Verify the application status.

   a. To check if all pods are up and running, run the command below.

   ```
   kubectl get pods -A
   ```

   b. Ensure the UI is accessible and functioning as expected.

Troubleshooting

1. **Handling istio issues (if present).**

   a. If the **istio-ingressgateway** pods are stuck in 0/1 Running state, run the command below.

   ```
   sudo iptables -P INPUT ACCEPT; sudo iptables -P FORWARD ACCEPT; sudo iptables -P OUTPUT ACCEPT; sudo iptables -F; sudo iptables -X; sudo iptables -Z; sudo iptables -t nat -F; sudo iptables -t nat -X; sudo iptables -t mangle -F; sudo iptables -t mangle -X; sudo iptables -t raw -F; sudo iptables -t raw -X;
   ```

This command flushes the iptables rules to resolve any networking issues related to istio.

b. Restart all pods (after flushing the iptables).

```
kubectl delete pods --all -A --force
```

2. **Handling pod specific issues (if present)** - If vault, mongo, or other pods start crashing with error message as xtables parameter problem: iptables-restore: unable to initialize table 'nat', do the following:
   a. Manually create the **iptable_filter.conf** file under **/etc/modules-load.d/**.
   b. To ensure the changes take effect after creating the above file, reboot the node.

Application Upgrade Guide 2023.1.0 FP3

The document describes the steps to upgrade AppViewX from the versions

- 2020.3.0 FP10-FP11
- 2021.1.0 FP3
- 2022.1.0 FP1-FP3
- 2023.1.0 FP1-FP2

to the latest v2023.1.0 FP3

AppViewX Supported Upgrade Paths

Purpose

The purpose of this document is to set the right expectations when upgrading AppViewX from a lower version to its latest version.

Supported Upgrade Table

Version Upgrade

Note: Ensure you follow the Prerequisites before proceeding with the upgrade.

From AppViewX Version	To Appviewx Version	Upgrade Mechanism	Guide Name
2020.3.0 FP10, FP11 with Ubuntu/ RHEL	2023.1.0 FP3	Application Upgrade	Application Upgrade Guide
2021.1.0 FP3 with Ubuntu/RHEL			
2022.1.0, FP1, FP2, FP3 with Ubuntu/RHEL			
2023.1.0, FP1, and FP2 with Ubuntu/ RHEL			

On-Prem Kubernetes to Managed Kubernetes Upgrade

From AppViewX Version	To Appviewx Version	Upgrade Mechanism	Guide Name
2020.3.0 FP10, FP11	Managed Kubernetes	Managed Kubernetes - AppViewX Install and Upgrade	1. Managed Kubernetes - AppViewX Install and Upgrade Guide for AKS
2021.1.0 FP3			2. Managed Kubernetes - AppViewX Install and Upgrade Guide for EKS
2022.1.0 FP1, FP2, FP3			3. Managed Kubernetes - AppViewX Install and Upgrade Guide for GKE

Prerequisites

General Prerequisites

1. If you are currently using CentOS operating system, refer to the CentOS Migration Guide.
2. Nodes must have the following OS:
   - RHEL 8.7, 8.8, 9.2, or 9.3
   - Ubuntu 20.04, 22.04
3. Keep the following file locations ready -

- old installer file location, for example - /home/appviewx/FP2/appviewx_kubernetes
- installed location, for example - /home/appviewx/appviewx_cluster

4. Location to save the new installer file - /home/appviewx/appviewx/Application_upgrade (Assign the folder name as required).
5. To list the nodes in the cluster, execute the command below and save the output for further reference.

```
kubectl get nodes --show-labels
```

> **Note:** If custom labels are detected add them to the custom_changes.yaml file. Refer to the chapter *Adding Custom Pod Configuration* of the section **Monitoring and Maintaining AppViewX** in the **Install, Upgrade, and Maintenance Guide**.

6. To get the status of HPA, execute the command below and save the output for further reference

```
kubectl get hpa
```

7. Keep a backup of the iControl jar files available at location - /home/appviewx/appviewx/appviewx_dependencies/external_libs/iControl-13.1.0.jar for iControl to be done after the upgrade.
8. Check for any other custom changes that may have been done specifically for the customer.
9. Check the enabled plugins in appviewx.conf file of the old installer (previous version), in case there are plugins that are not present but are required to be installed in the latest versions, please add them, example
   - avx_pkiaas_cert_ocsp_generator
   - avx_pkiaas_cert_ocsp_serveravx_pkiaas_cert_ocsp_server
   - avx_platform_hsm

Points to Remember when Upgrading from 2022.1.0 (FP1, FP2, FP3) to 2023.1.0 FP3

Know the following before proceeding with the FP3 upgrade.

1. If an external CA certificate is configured for kubernetes, the infra upgrade will overwrite the certificates.
2. A manual elastic restore must be performed post the upgrade. Post the application upgrade the back of the elastic search will be stored at the following location - INSTALLER_PATH/appviewx_kubernetes/statistical_data_backup. The steps for Elastic Restore are explained in the section elastic restore

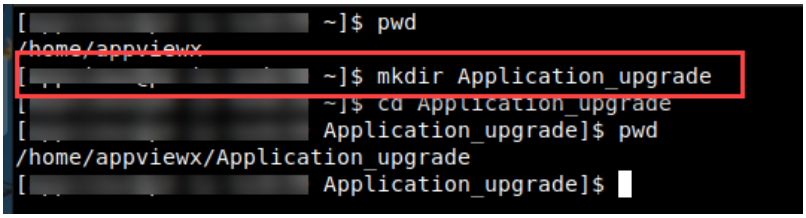
Upgrading AppViewX to v2023.1.0 FP3

You can now upgrade to AppViewX version 2023.1.0 FP3 if you are currently using the following versions of AppViewX in RHEL (8.7-8.8 and 9.2, 9.3) or Ubuntu (20.04, 22.04) OS:

- 2020.3.0 FP10-F11
- 2021.1.0 FP3
- 2022.1.0 FP1-FP3
- 2023.1.0 FP1-FP2

1. Log in to the release portal and download the installer and addons file –
   - **appviewx_kubernetes_2023.1.3.0.tar.gz**
   - **appviewx_kubernetes_addons_2023.1.3.0.tar.gz**
2. Create a new folder in the same location as the existing installer directory.

   Example: /home/appviewx/Application_upgrade/



3. Copy the installer file **appviewx_kubernetes_2023.1.3.0.tar.gz** into the new folder location /home/appviewx/Application_upgrade/.
4. Untar the installer file using the command below:

   ```
   tar –xf appviewx_kubernetes_2023.1.3.0.tar.gz
   ```

   After the command is executed, the **appviewx_kubernetes** folder is created: /home/appviewx/Application_upgrade/appviewx_kubernetes/
5. Copy the **appviewx_ kubernetes_addons_2023.1.3.0.tar.gz** file to the **appviewx_kubernetes** folder using the command:

   ```
   mv appviewx_kubernetes_addons_2023.1.3.0.tar.gz appviewx_kubernetes/
   ```

6. Navigate to the **scripts** directory in the **appviewx_kubernetes** folder (/home/appviewx/Application_upgrade/appviewx_kubernetes) using the command:

   ```
   cd /home/appviewx/Application_upgrade/appviewx_kubernetes/scripts
   ```

   The **scripts** folder contains the **appviewx.conf.template** file.
7. Update the conf parameters in the appviewx.conf file as mentioned in the Install and Upgrade Maintenance guide after copying the **appviewx.conf.template** file as **appviewx.conf**

   To copy, use the command:

   ```
   cp appviewx.conf.template appviewx.conf
   ```

   OR

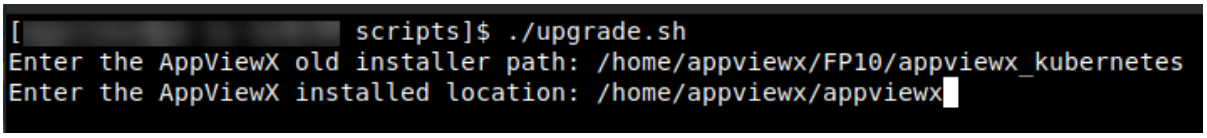
Skip the above step to use the conf merge feature as part of step 10b.

> **Note:**
> • For the complete list of the appviewx.conf file parameters refer the *Configuring the appviewx.conf File to Install Appviewx* section in the **Install and Upgrade Maintenance Guide**.

8. From the /home/appviewx/ApplicationUpgrade/appviewx_kubernetes/scripts directory execute the upgrade command below:

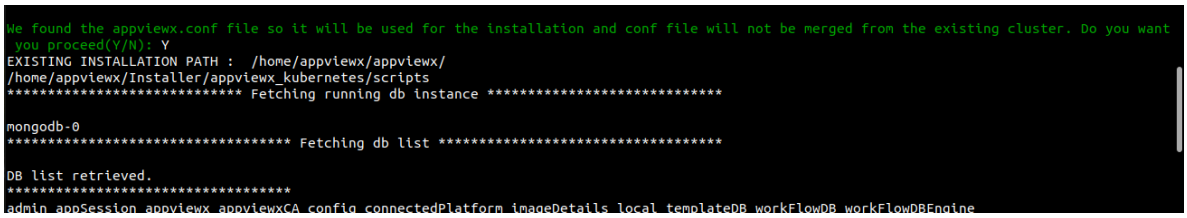
```
./upgrade.sh
```

9. Provide the input of the older installer directory and the directory where the application is currently installed.

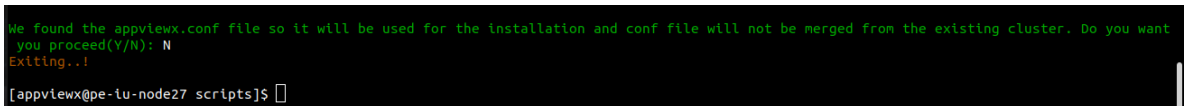


After entering both inputs, the system checks for newly introduced conf file parameters.

10. You will now be prompted with the message about the presence of the conf file, answer Y/N as follows:
    a. If the updated conf file is available in the installer folder, and you choose **Y**, the upgrade proceeds.



    b. If the updated conf file is available in the installer folder, and you choose **N**, the upgrade stops/exits.



    To continue with the upgrade
    • Edit the conf file and resume the upgrade.
    • Delete the conf file from the installer location and resume the upgrade (the upgrade script will handle the merging of the new conf parameters).

11. Enter the appropriate value to alter the default value OR hit the enter key (*recommended*) to use the default value. An example is shown below.

```
Checking for newly introduced conf parameters...
================================================================================================================================
Please provide the appropriate input for SAAS_ENABLED
# Flag to check if saas enabled or on-prem
############################
# DO NOT CHANGE FOR ON-PREM  #
############################
Default value for the parameter is : false

Please enter the value to alter the default value according to the above instruction. Kindly press enter to use default value :
```

a. To enable msp, the default value is False. Hit the enter key to select the default value and continue.

b. For the parameters HSM_HOST and REDIS_HOST enter the value as follows:

```
================================================================================================================================
Please provide the appropriate input for HSM_HOST
# Comma separated values of node hostnames in which HSM pods will be scheduled
# Note: Execute the command "hostname" in the node and add that output to this field
# IMPORTANT: (i) For single node AppViewX deployments add the IP address of the instance where AppViewX is installed.
# (ii) To ensure high availability in multiple DC deployments, It is recommended to add a minimum of one host per DC.
Default value for the parameter is : $(hostname)

Please enter the value to alter the default value according to the above instruction. Kindly press enter to use default value :
```

   i. If you have a single node, hit Enter for the default value or the IP address of the instance where AppViewX is installed.
   ii. If you have a multi-node setup, you must enter one hostname per DC of the worker nodes.

c. To configure the BACKUP_CRONJOB_SCHEDULE, enter the values in double quotes. For example, “0 4 * * * states that the cron job will run at 4:00 AM every day.

d. To configure BACKUP_CRONJOB_RETENTION, enter an inter value. For example, 5, which means that the system will keep the last 5 backups and delete any older ones.

e. To configure the SECONDARY_DB_BACKUP, set the value to true, if DB backup has to be taken from the secondary shared DB.

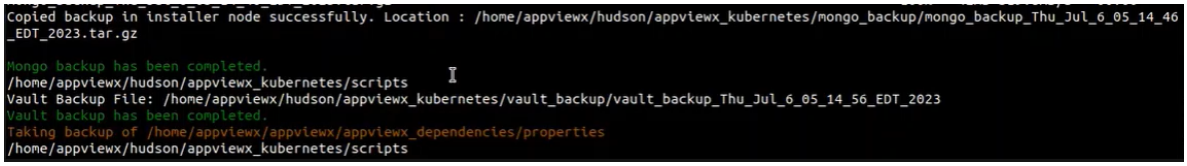
f. To configure EXTERNAL_GATEWAY_HOST, enter one of the ingress host's hostname in which the external gateway is to be deployed.

g. For SENTINAL_DC enter the value as follows (only for 2-DC setup):
   i. If it's not a 2-DC setup, enter any one of the DCs.
   ii. If you have a multi-node 2-DC setup, enter the DC which has less number of redis instances than the other DCs.

> **Note:** Ensure you read all the instructions specified in the conf parameters before entering the values.

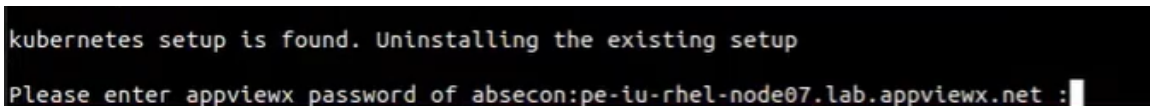
12. The upgrade continues and the following operations are carried out during the process.

a. **Taking backups of mongo and vault**

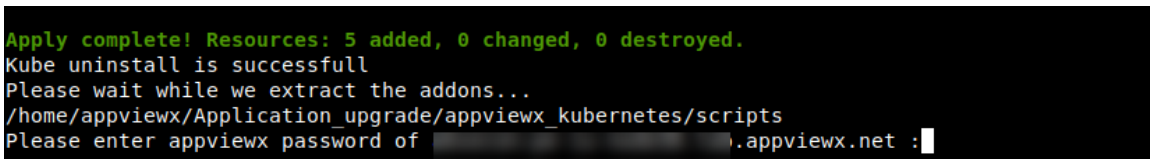


b. **Uninstalling the old version**

i. You will be prompted to enter the node password.



ii. After the uninstall is complete, you will be prompted to enter the password for the DC host.

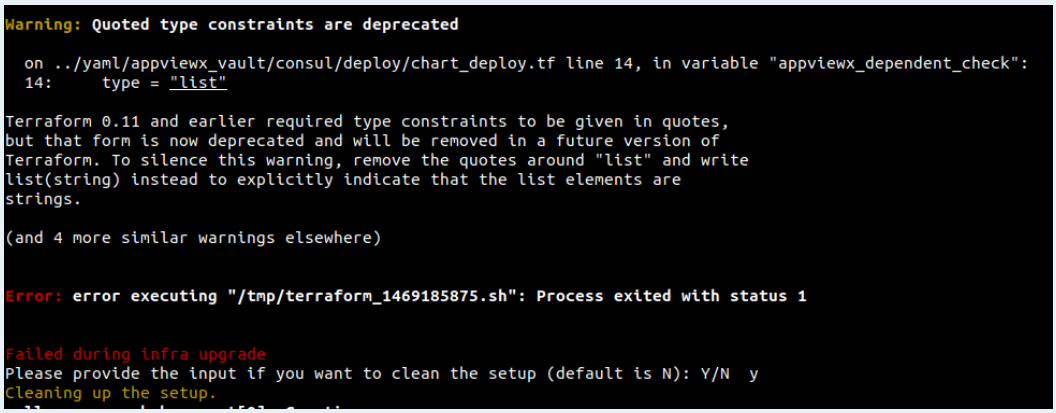


Note:

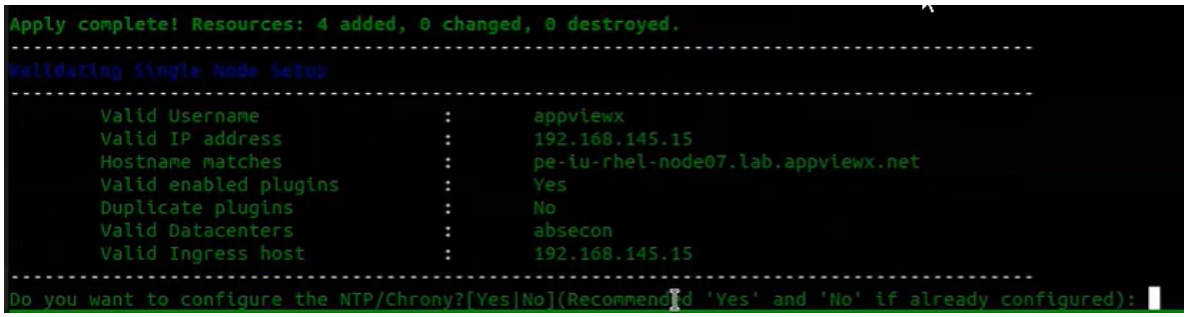
i. In case of upgrade failures, resume the upgrade by executing the command:

```
/upgrade.sh
```

ii. In case of Infra upgrade failure, the script will prompt a question to clean the setup as shown below. Enter ‘Y’ (yes) to proceed with the clean-up.

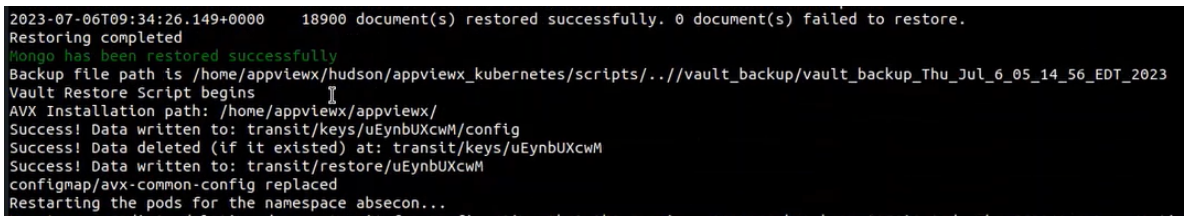


c. **Time Sync (NTP/Chrony)**



- For a single node - Enter **No** as we do not have to sync time.
- For multi-node - If time sync is already configured before the upgrade then enter **No**. If the time sync for nodes has to be configured then enter **Yes**.

d. **Installing the new version and restoring the backups**



Note: In the case of Mongo restore, if the restore operation is stuck or takes more time than usual, then stop the installation process and increase Mongo's **wiredTiger** cachesize of the **mongodb** or **mongo-shareddb statefulset**. (The **mongodb** is for single node setup and **mongo-shareddb statefulset** is for multi-node setup). Use the commands below.

Single Node:

```
kubectl edit statefulset mongodb -n avx
```

Multi node:

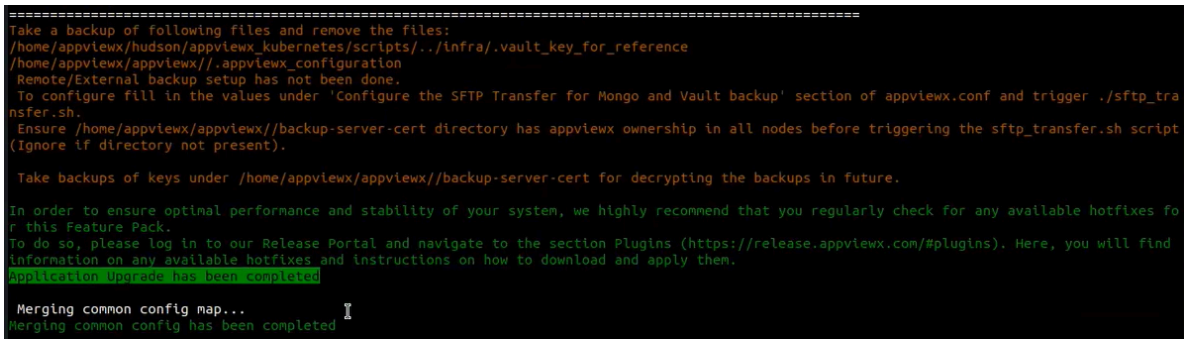
```
kubectl edit statefulset mongo-shareddb -n avx
```

Navigate to **MONGO_CACHE_SIZE** key value of the **env** field and increase the cache size value by 1 to 2 GB

```
        env:
        - name: MONGO_CACHE_SIZE
          value: "0.25"
        image: mongo:4.2.18
        imagePullPolicy: Never
```

After making the required changes run command `./upgrade.sh` to resume the upgrade.

e. **Merging the common config map**



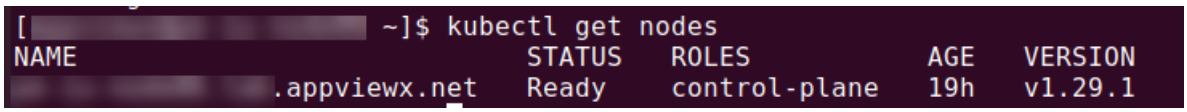
f. After the installation is complete, take a backup of the below files and copy it to a secure location. Then, remove it from the installer location. The files are

- <installer location>/infra/.vault_key_for_reference
- <installer location>/.appviewx_configuration

13. Check the upgraded version and the pods running status.

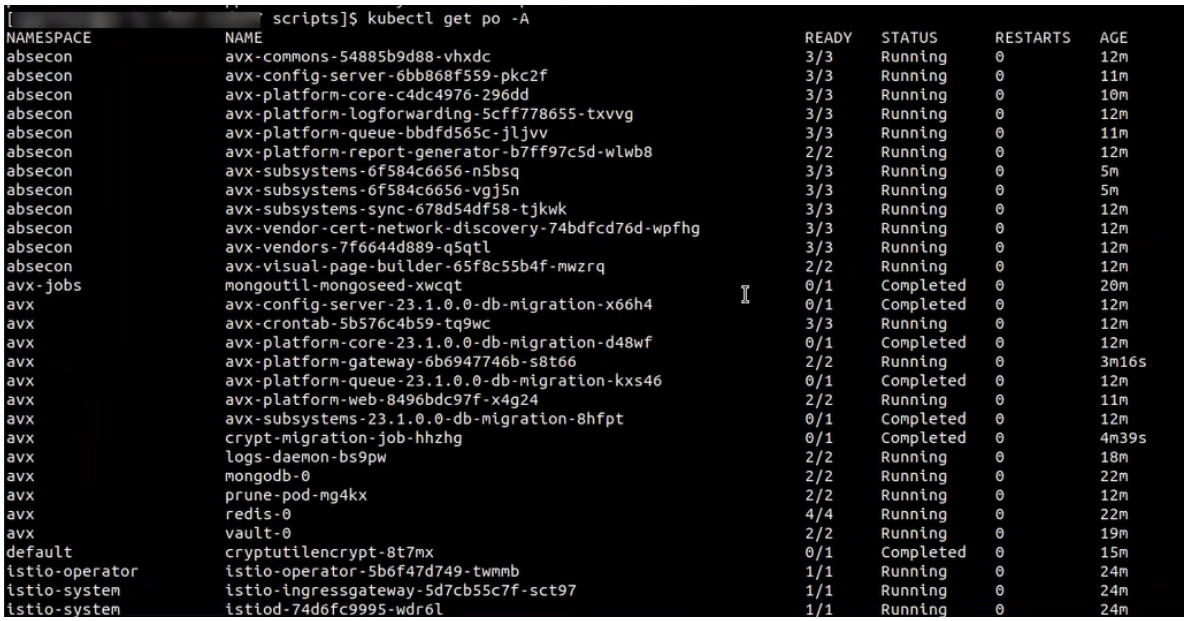
a. To check the upgraded version, run the following:

```
kubctl get no
```



b. To check the pods running status, run the following:

```
kubctl get po -A
```



```
[                    scripts]$ kubectl get po -A
NAMESPACE             NAME                                                READY   STATUS      RESTARTS   AGE
absecon               avx-commons-54885b9d88-vhxdc                        3/3     Running     0          12m
absecon               avx-config-server-6bb868f559-pkc2f                  3/3     Running     0          11m
absecon               avx-platform-core-c4dc4976-296dd                    3/3     Running     0          10m
absecon               avx-platform-logforwarding-5cff778655-txvvg         3/3     Running     0          12m
absecon               avx-platform-queue-bbdfd565c-jljvv                  3/3     Running     0          11m
absecon               avx-platform-report-generator-b7ff97c5d-wlwb8       2/2     Running     0          12m
absecon               avx-subsystems-6f584c6656-n5bsq                     3/3     Running     0          5m
absecon               avx-subsystems-6f584c6656-vgj5n                     3/3     Running     0          5m
absecon               avx-subsystems-sync-678d54df58-tjkwk                3/3     Running     0          12m
absecon               avx-vendor-cert-network-discovery-74bdfcd76d-wpfhg   3/3     Running     0          12m
absecon               avx-vendors-7f6644d889-q5qtl                        3/3     Running     0          12m
absecon               avx-visual-page-builder-65f8c55b4f-mwzrq            2/2     Running     0          12m
avx-jobs              mongoutil-mongoseed-xwcqt                           0/1     Completed   0          20m
avx                   avx-config-server-23.1.0.0-db-migration-x66h4       0/1     Completed   0          12m
avx                   avx-crontab-5b576c4b59-tq9wc                        3/3     Running     0          12m
avx                   avx-platform-core-23.1.0.0-db-migration-d48wf       0/1     Completed   0          12m
avx                   avx-platform-gateway-6b6947746b-s8t66               2/2     Running     0          3m16s
avx                   avx-platform-queue-23.1.0.0-db-migration-kxs46      0/1     Completed   0          12m
avx                   avx-platform-web-8496bdc97f-x4g24                   2/2     Running     0          11m
avx                   avx-subsystems-23.1.0.0-db-migration-8hfpt          0/1     Completed   0          12m
avx                   crypt-migration-job-hhzhg                           0/1     Completed   0          4m39s
avx                   logs-daemon-bs9pw                                   2/2     Running     0          18m
avx                   mongodb-0                                           2/2     Running     0          22m
avx                   prune-pod-mg4kx                                     2/2     Running     0          12m
avx                   redis-0                                             4/4     Running     0          22m
avx                   vault-0                                             2/2     Running     0          19m
default               cryptutilencrypt-8t7mx                              0/1     Completed   0          15m
istio-operator        istio-operator-5b6f47d749-twmmb                     1/1     Running     0          24m
istio-system          istio-ingressgateway-5d7cb55c7f-sct97               1/1     Running     0          24m
istio-system          istiod-74d6fc9995-wdr6l                             1/1     Running     0          24m
```

Post Upgrade Steps

The following actions must be taken to avoid any post-upgrade errors, if they are applicable in your respective setup:

1. **Loss of Mongo replica set priority configurations**

   During application upgrade, mongodb is freshly set up with the latest upgraded versions. The existing replicaset configurations such as replicaset priorities will not be taken ahead and hence have to be re-configured. High latency customers must perform the following step:

   a. Configure the parameters OPTIMISE_ROUTING_FOR_LATENCY and PREFERRED_DEFAULT_DC in the appviewx.conf
   b. Re-trigger the `plugins_install.sh` to change the configurations.

2. **Custom changes**

   If the custom_changes.yaml, custom_vm_args.conf are present and updated in the custom changes, then the custom changes will be persistent. Any of the custom changes that may have been done specifically for the customer as noted in the prerequisites will not be present if the above mentioned files are not updated with this configuration.

3. **External web cert is not upgraded**

To update the external CA web certificate, execute the command below:

```
./appviewx.sh --update-web-cert
```

The following prompts will be displayed:

- Enter the absolute path of external cert file:
- Enter the absolute path of external key file:

Enter both the values to proceed. Once the cert upgrade is completed, restart the gateway and web.

4. Download the **icontrol.jar** and the **axis.jar** files and copy them to the **external_libs**directory. For details, refer to the section iControl F5 Integration of the *AppViewX Install, Upgrade and Maintenance Guide*.
5. **Setting the ELASTIC_ENABLE as True in the Statistics Configuration**

   There are two ways to do it, choose from either the command prompt (a) or from the management console UI (b).

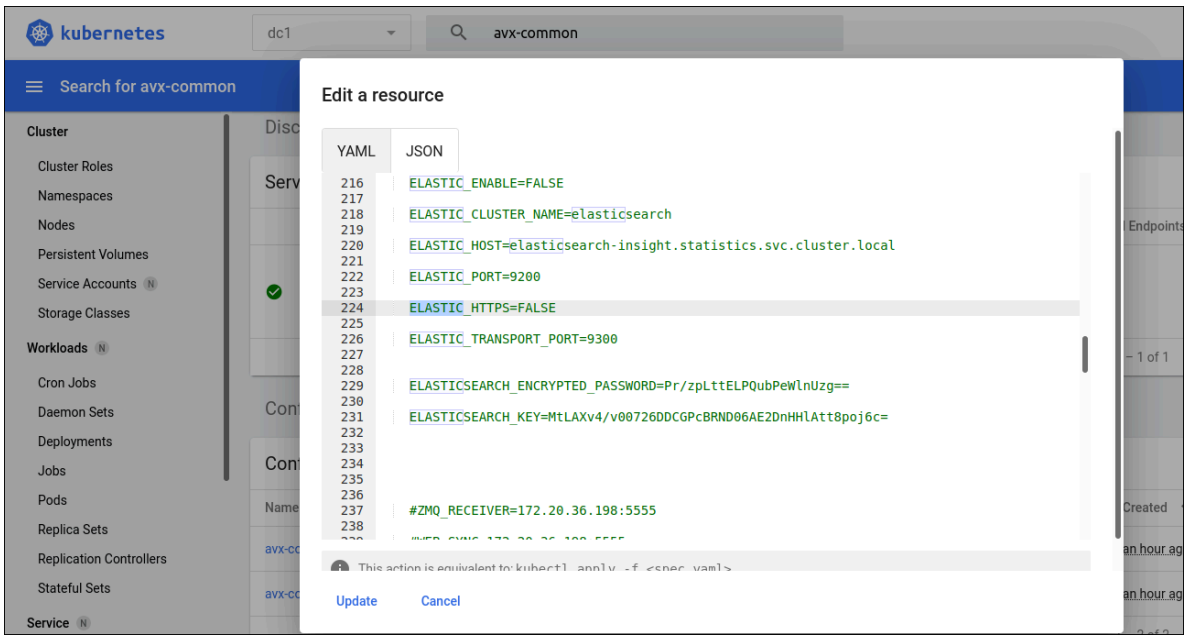
   a. Execute the command

   ```
   kubectl edit configmaps -n <Datacenter1>
   ```

   Search for the keyword as Elastic and set ELASTIC_ENABLE as True and below params should have default values as below

   ```
   ELASTIC_ENABLE=TRUE
   ELASTIC_CLUSTER_NAME=elasticsearch
   ELASTIC_HOST=elasticsearch-insight.statistics.svc.cluster.local
   ELASTIC_PORT=9200
   ELASTIC_HTTPS=FALSE
   ELASTIC_TRANSPORT_PORT=9300
   ```

b. Login to management console >> Search for the namespace with the configured DC >> Search for avx-common-config in config maps >> Click on Edit and search for Elastic >> Set as True and give as update as shown below.



- Elastic Restore
- Enabling HSM

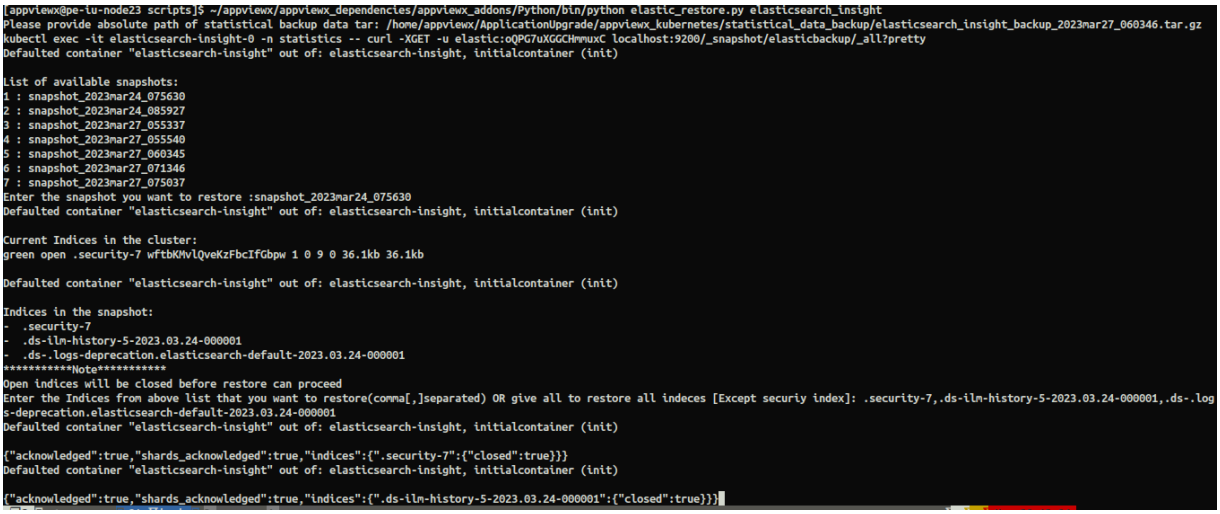
Elastic Restore

The script **elastic_restore.py** is used for restore. To manually perform the elastic restore,

1. Navigate to the scripts directory.
2. Run the elastic_restore.py script to restore the backup.

```
/home/appviewx/appviewx/appviewx_dependency/appviewx_addons/Python_Linux/bin/python elastic_restore.py elasticsearch_insight
```

3. Script will ask for the backup tar which was created manually. Provide the absolute path of the backup tar.



4. Script will list all the available snapshots that has date and time in the naming. Select the backup which you want to restore.
5. Provide the details of the indices you want to restore (follow the screenshot above).

Enabling HSM

Prerequisites

- To configure Fortanix and Utimaco, the **.so** file and **config** file must be present in the current Appviewx version.
  - The **.so** file is essential for communicating with the HSM using the PKCS11 interface.
  - The **config** file facilitates communication between the HSM and Appviewx.

After the successful upgrade, proceed with the steps below to enable HSM.

1. Ensure the HSM pod is operational and running in the required datacenters and that the HSM node is specified in the appviewx.conf file, execute the following command:

   ```
   kubectl get pods -A -o wide |grep hsm
   ```

2. From the command line interface, navigate to the properties folder path {APPVIEWX_INSTALLATION_PATH}/appviewx_dependencies/properties

   For Fortanix

   a. Open the HSM file using the following command:

   ```
   vi hsm
   ```

b. Check and confirm if the HSM file has the following lines. If not, uncomment the following lines:

```
export FORTANIX_PKCS11_CONFIG_PATH= /appviewx/dependencies/hsm/fortanix/pkcs11.conf
```

```
echo "FORTANIX Config Path : $FORTANIX_PKCS11_CONFIG_PATH"
```

c. If the file is edited, restart the **avx-platform-hsm** pod, using the following commands:

```
kubectl get pods -n <namespace>
```

```
kubectl delete pods -n <namespace> <PodName> --force
```

For Utimaco

a. Open the HSM file using the following command:

```
vi hsm
```

b. Check and confirm if the HSM file has the following lines. If not, uncomment the following lines:

```
export CS_PKCS11_R2_CFG=/appviewx/dependencies/hsm/utimaco/cs_pkcs11_R2.cfg
```

```
echo "UTIMACO Config Path : $CS_PKCS11_R2_CFG"
```

c. If the file is edited, restart the **avx-platform-hsm** pod, using the following commands:

```
kubectl get pods -n <namespace>
```

```
kubectl delete pods -n <namespace> <PodName> --force
```

3. Once the HSM pod is back to running state, login to AppViewX and navigate to **Platform** > **Vault & Security** > **HSM**.
4. Access the required HSM.

• Configuring Fortanix

• Configuring Utimaco

• Verifying/Modifying HSM Configuration for Private Key Encryption

Configuring Fortanix

1. If the added HSM is Fortanix, upload the **.so** file and **config** file as follows:

   To upload the **.so** file,

   a. Click **Browse**.

   b. Navigate to the location of the .so file.

   c. Select the .so file and click **Open**.

   To upload the **.cfg** (config) file,

a. Click **Browse**.
b. Navigate to the location of the .cfg file.
c. Select the .cfg file and click **Open**.

2. Choose the respective datacenter.
3. Update each HSM available in the inventory one by one, ensuring that the HSM is moved to the **Available** status.

Configuring Utimaco

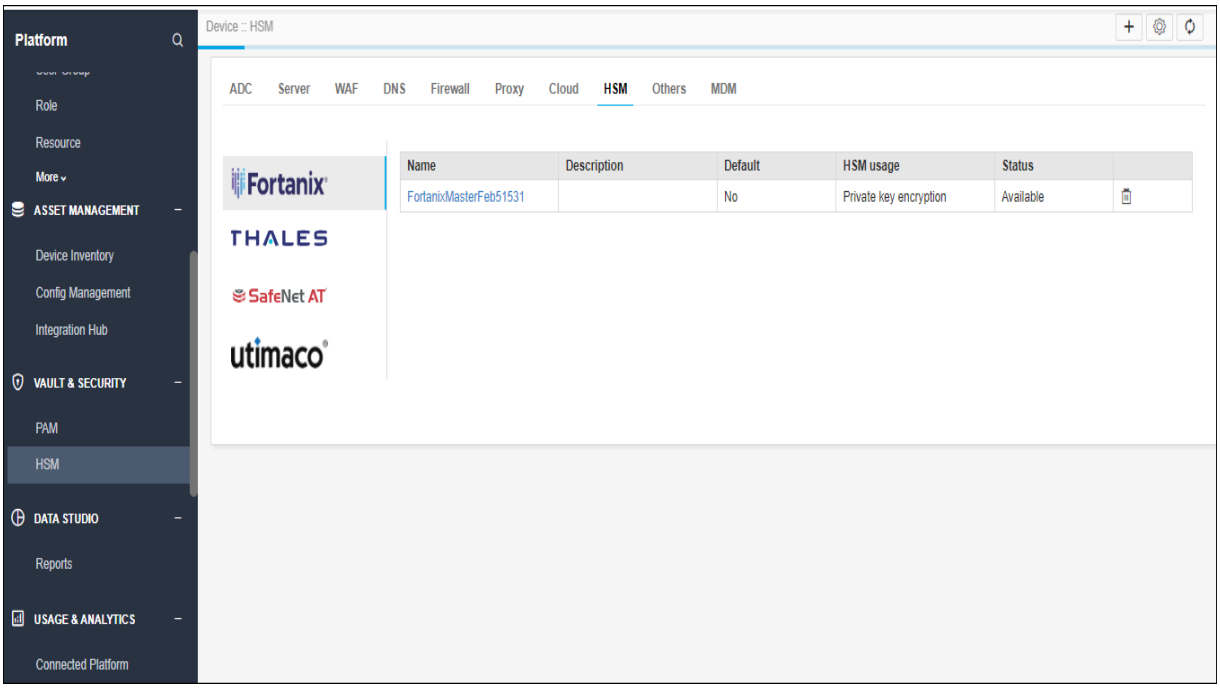
1. If the added HSM is Utimaco, upload the **.so** file and **config** file as follows:

   To upload the **.so** file,
   a. Click **Browse**.
   b. Navigate to the location of the .so file.
   c. Select the .so file and click **Open**.

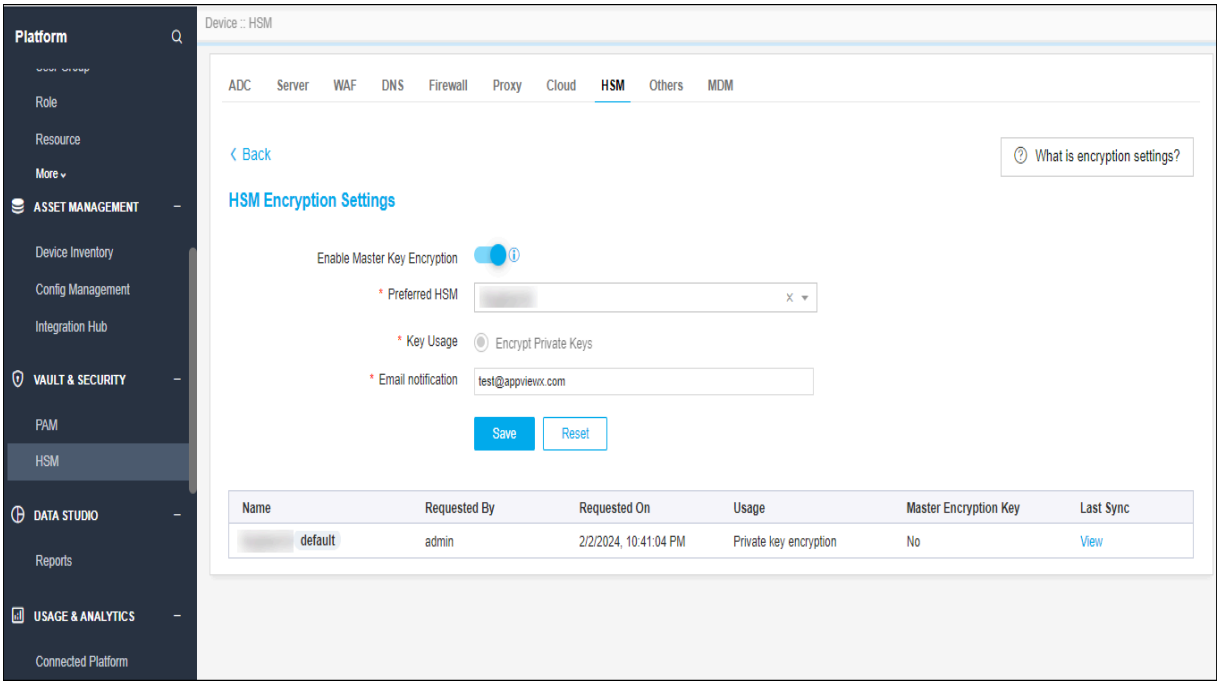
   To upload the **.cfg** (config) file,
   a. Click **Browse**.
   b. Navigate to the location of the .cfg file.
   c. Select the .cfg file and click **Open**.
2. Choose the respective datacenter.
3. Update each HSM available in the inventory one by one, ensuring that the HSM is moved to the **Available** status.

Verifying/Modifying HSM Configuration for Private Key Encryption

1. On the top-right corner of the HSM inventory page, click the master encryption settings icon.



2. If the already added HSM has implementation type **private key generation** or **Both** and Set as “default” in 2020.3.0 Appviewx version then that HSM will be available in the Master encryption settings page as default HSM.



3. To receive emails for default HSM health status, configure the email address in the settings page.

> **Note:** For email use case SMTP settings should be available in **Platform** > **System Administration** > **SMTP**.

4. To change the new default HSM, add the new HSM with HSM usage as **Master key encryption** or **both**.
5. Once the HSM is moved to **Available** status it will be present in the **Preferred HSM** dropdown field of the master key encryption settings page.
6. Choose the new HSM and click **Save**.

Steps to Achieve High Availability

In Hudson FP1, MongoDB has been updated to version 5.x. With MongoDB 5.0, the default 'WriteConcern' is set to 'majority,' impacting MongoDB deployments with Arbiters in high availability setups. To resolve this, we're reverting the 'WriteConcern' value back to 1, the default value in previous MongoDB versions. Run the "high_availability_setup.sh" script to set the WriteConcern for MongoDB, set node affinity for the platform-web, and establish Redis HA.

1. Login to AppViewX software release portal 2023.1.0 page: https://release.appviewx.com/Login#overview/AppViewX_2023.1.0.
2. Download the HA files from the following location:
   high_availability_setup.sh (md5sum - f3f8c83cf6f02c2b529cf3090be36a35)

   Link: https://release.appviewx.com/downLoadArtifact?id=1200
3. Navigate to the folder: /home/appviewx/Install20231110/appviewx_kubernetes/scripts.

```
$ cd /home/appviewx/<install_dir>/appviewx_kubernetes/scripts
```

4. Copy the downloaded **high_availability_setup.sh** file into the scripts folder location:

```
cp <file downloaded location>/high_availability_setup.sh <installation_dir>/scripts
```

5. Execute the file by running the following command:

```
$ chmod +x high_availability_setup.sh
```

```
$ ./high_availability_setup.sh
```

Troubleshooting for Setup Limitations

Consul Stuck in 1/2 State

The consul may be stuck in 1/2 state in case of a hard restart. If you encounter this check the consul logs using the command:

```
kubectl logs <consul-consul-server-0> -n avx
```

where <consul-consul-server-0> is the name of the consul pod which is stuck in 1/2 state.

In such scenario run the following command::

1. Scale down consul server to zero replicas.

   ```
   kubectl scale -- replicas=0 consul-consul-server -n avx
   ```

2. Wait for the consul-consul server pods to be terminated.

3. Scale the consul server to three replicas.

   ```
   kubectl scale -- replicas=3 consul-consul-server -n avx
   ```

Failure in decryption within the pods

This failure arises for instances where both active vault and ephemeral vaults are configured. If the keys in the vaults are not in sync the decryption within the pods will fail causing the pods to crash. In such a case re-sync the vaults by the steps below

- Navigate to <installer >/appviewx_kubernetes/yaml/appviewx_vault_ha/
- Execute the command

  ```
  ./uninstall_vault_ha.sh
  ```

- Once completed trigger the script

  ```
  ./run.sh
  ```

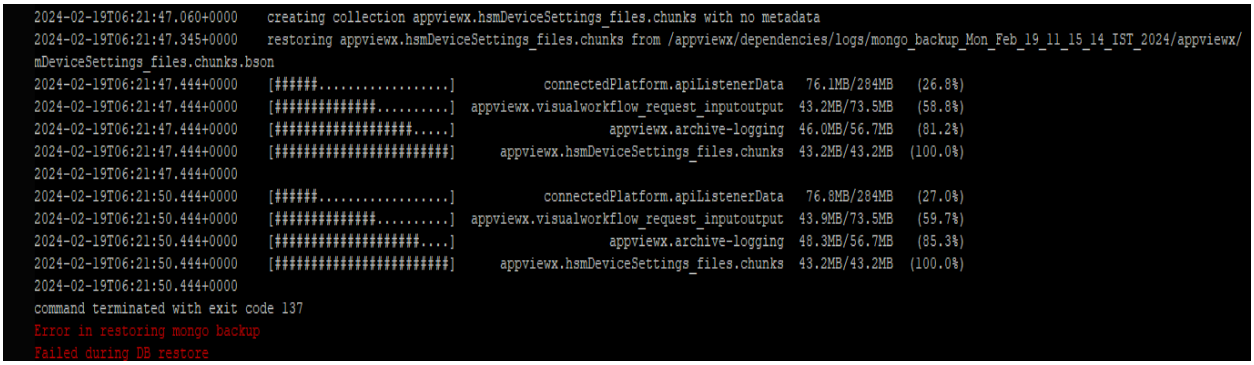
Post completion the keys in the vault will be in sync and the vault will be up and running.

Error while installing the AppViewX plugins

If an error occurs during the installation of AppViewX plugins, it is likely due to an error in the configuration file. You may observe an error such as `Upload failed: scp`, in such cases re-trigger `plugins_install.sh` to install the plugins. Likewise, ensure to review the configuration file carefully and proceed with the execution of plugins_install.sh to install only the plugins.

Pod Out of Memory

During the mongo restore step of the application upgrade process, the pod may go into an out of memory state (exit code 137 as in the screenshot below). In this case resume the upgrade by rerunning the `upgrade.sh` command.

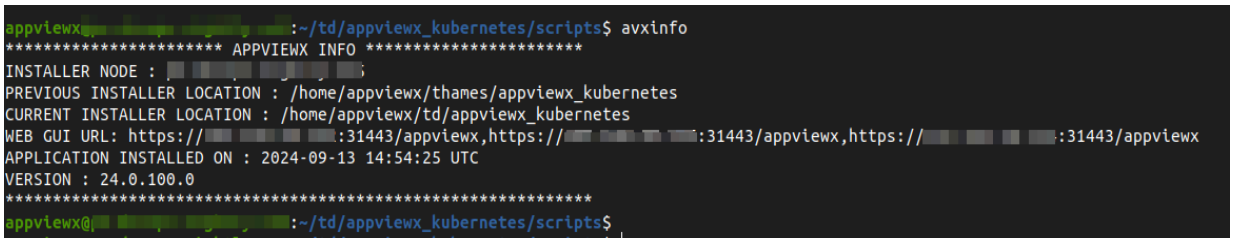


Interactive-Based Installation with Terminal UI

This document outlines several new features designed to simplify the AppViewX installation process and enhance the overall user experience. The most notable addition is the introduction of an interactive terminal-based installation UI, which enables users to generate the *appviewx.conf* file without manual steps. Other key improvements include enhanced prerequisite configuration, encrypted backup support, and better log management.

Key Features

- *User-Friendly Installation*: The interactive terminal UI guides users through the installation process step-by-step, making it easier for users of all technical backgrounds.
- *Automated Prerequisite Configuration*: A prerequisite check and configure option helps users avoid common dependency issues, creating a more seamless installation experience.
- *Secure Backup*: MongoDB and Vault backups now feature encryption, ensuring sensitive data remains secure during transfers to external servers such as Windows SFTP. Additionally, backup success or failure alerts will be visible in the AppViewX infrastructure alerts section, with options to configure email notifications.
- *Enhanced Log Management*: Configurable *logrotate* has been introduced for MongoDB and plugin logs, streamlining log management and preventing disk space overuse through automatic log cleanup.
- *Avxinfo*: A new command, avxinfo, has been introduced to retrieve details such as the installer node and installation location. This command can be executed from any node within the cluster.



- *Auto Remediation*: This feature automatically conducts checks on the Kubernetes platform, identifies and resolves issues, and collects log files to determine the root cause of each problem. It empowers customer success teams and customers to rectify issues themselves without the need of reaching out to the engineering team. Newly identified issues will be incorporated into this tool for future handling.
- *Collect Logs*: This feature automates the process of collecting and archiving critical data, such as application logs, system logs, and TCP dump files.
- *Apply Patch*: This feature facilitates updating the existing AppViewX setup by applying plugin patches, add-on patches, or both, ensuring the system stays up-to-date and functional.

Prerequisites

Before beginning the installation, ensure that the following are prepared:

- Sudo access to the configured for the appviewx installation user
- If the user opts for a key-based installation, passwordless sudo must be configured. As the root user, add the following lines to the **/etc/sudoers** file:

```
appviewx ALL=(ALL) NOPASSWD:ALL
```

- Load balancer (LB) for the AppViewX GUI (*optional - refer note below*)
- A .p12 certificate for the GUI (*optional - refer note below*)
- NTP server details
- Nameserver details

- Kubernetes master L4 load balancer (*optional - refer note below*)
- Proxy or internet access is required on all the nodes if any of the following OS prerequisite packages mentioned below are to be installed during the prerequisite check. Ignore, if the packages are already installed on the nodes.
  - **Ubuntu** - curl, net-tools, nmap, zip, unzip, sysstat, rsync, tcpdump, chrony, bind9-utils, dnsutils, ebtables, netcat, netcat-openbsd
  - **RHEL** - curl, net-tools, nmap, zip, unzip, sysstat, rsync, tcpdump, chrony, bind-utils, nmap-ncat, (ebtables, iptables-ebtables, iptables-nft) any one based on OS version.

Note: The Load Balancers and the **.p12** certificate prerequisites are optional during the installation process but are recommended to be enabled for High Availability (HA).

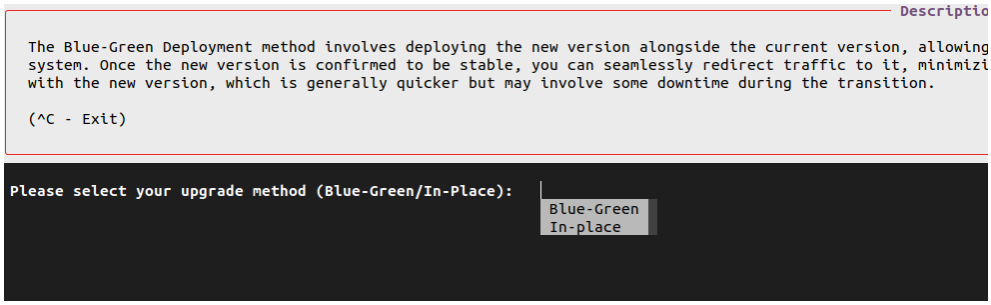
This will help complete the installation or upgrade smoothly in one attempt.

If a custom workflow is being used, please contact AppViewX technical support before proceeding with the upgrade.

Keyboard Shortcuts for Install and Upgrade Operations

- *Ctrl* + *Z*: Go to the previous question
- *Ctrl* + *L*: Collect the logs
- *Ctrl* + *W*: Scroll up
- *Ctrl* + *S*: Scroll down
- *Ctrl* + *U*: Unpause scrolling
- *Ctrl* + *C*: Cancel the installation or upgrade
- *Ctrl* + *T*: Show or hide the password
- *Ctrl* + *E*: Show the error message
- *Ctrl* + *P*: To rollback during apply patch failures
- *Ctrl* + *R*: To resume
- *V* : View verbose logs
- *Tab*: Press the tab key for suggestions to questions

Example:



Terminal UI Installation

1. To start the installation process, run the command:

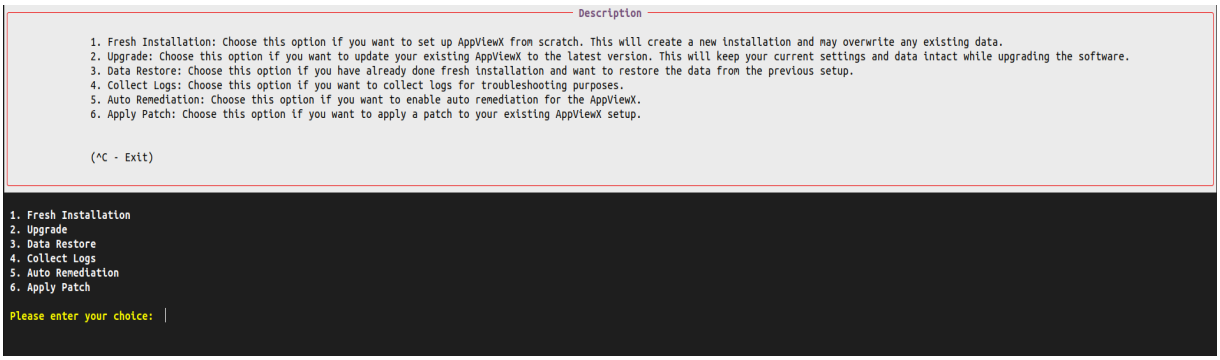
   ```
   ./install.sh
   ```

   The following welcome screen is displayed.

2. Select **Continue**.

   The screen shows three options

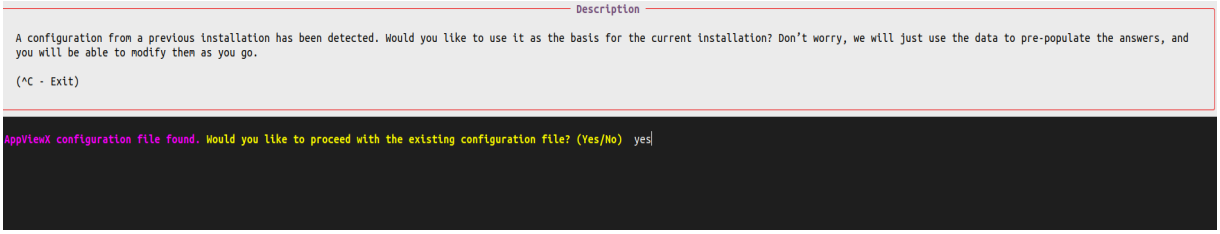
   - 1. Fresh Installation
   - 2. Upgrade
   - 3. Data Restore
   - 4. Collect Logs
   - 5. Auto Remediation
   - 6. Apply Patch



3. To install the AppViewX application from scratch, Enter **1** (Fresh Installation).

Note: In case the application is already installed, you will be prompted to uninstall and the proceed with the installation.

4. Follow the interactive steps provided by the Terminal UI to configure the settings for generating the appviewx.conf file.



Note: If an appviewx.conf file already exists, you will be prompted to continue with the existing file. However, validations will occur for each configuration step.

5. Ensure all required inputs are provided, and the system will automatically generate the configuration file, ready for use.
6. Once all the questions are answered, you will be prompted with a final table displaying the provided details. Review the information carefully, and type '**y**' to proceed with the prerequisites validation.

Description	Value
Installation Type	Multi Node
AppViewX will be installed using this username	appviewx
SSH port	22
Authentication method for SSH connection	password
Total number of data centers	2
All Data Center Name(s)	dc1,dc2
Hostname(s) / IP address(es) of master nodes	[redacted]
Names of Datacenter(s) with Master VM(s)	dc1, dc2
Master VM(s) in dc1	[redacted]
Master VM(s) in dc2	[redacted]
Worker VM(s) in dc1	[redacted]
Worker VM(s) in dc2	[redacted]
All VM(s) in dc1	[redacted]
All VM(s) in dc2	[redacted]
Hostname / IP address of the TCP Load Balancer	[redacted]
TCP Load balancer port	6443
Hostname(s) / IP address(es) of all Database nodes	[redacted]
Hostname(s) / IP address(es) of all Vault nodes	[redacted]
Hostname(s) / IP address(es) in which the AppViewX GUI will be accessible	[redacted]
External certificate for the AppViewX web interface	/home/appviewx/on-premise.p12
Hostname(s) / IP address(es) of all HSM nodes	[redacted]
The directory where AppViewX will be installed	/home/appviewx/appviewx_app
Remote Server IP or Hostname for SFTP Transfer for Database and Vault backup	[redacted]

o you wish to continue (y/n)?

^S to scroll down and ^W to scroll up

7. **Prerequisites validation** - The prerequisites validation will be performed based on the information provided.

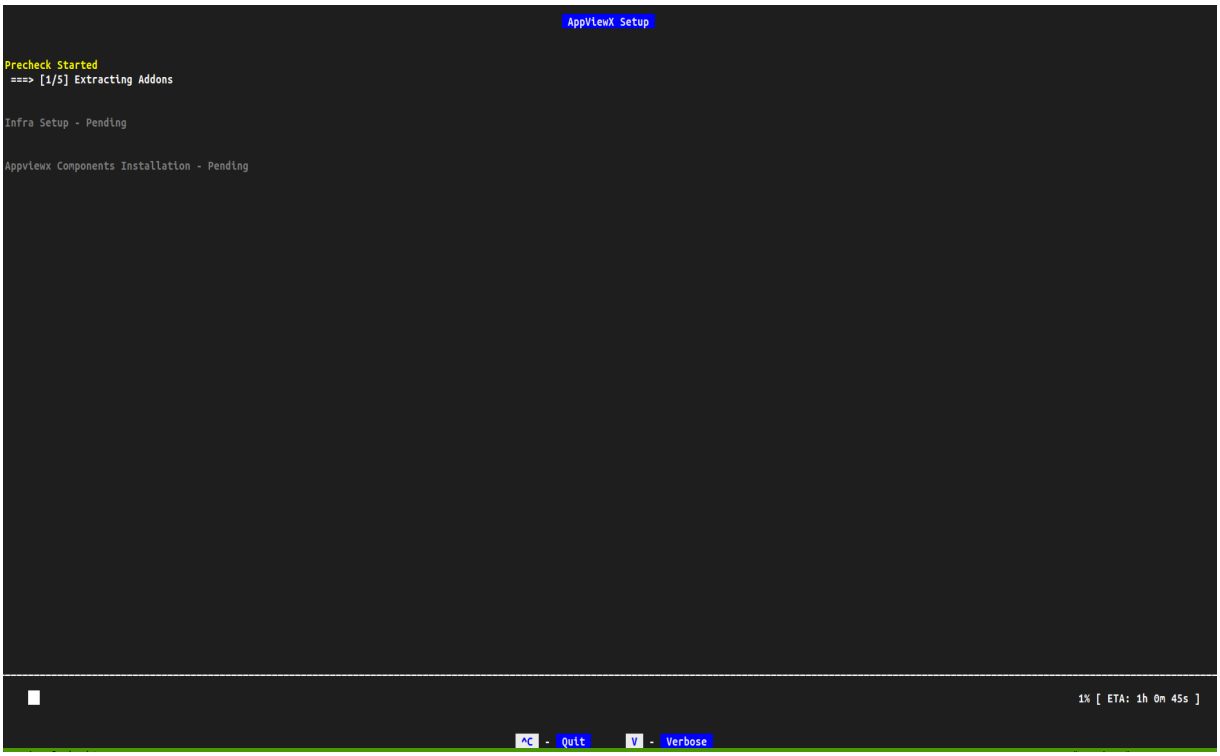
8. **Configuring Prerequisites** - If any prerequisites are missing, the Prerequisite Configure option allows the installer to automatically install or configure the necessary packages. Press '**y**' to continue with the configuration.

AppViewX Prerequisite Report

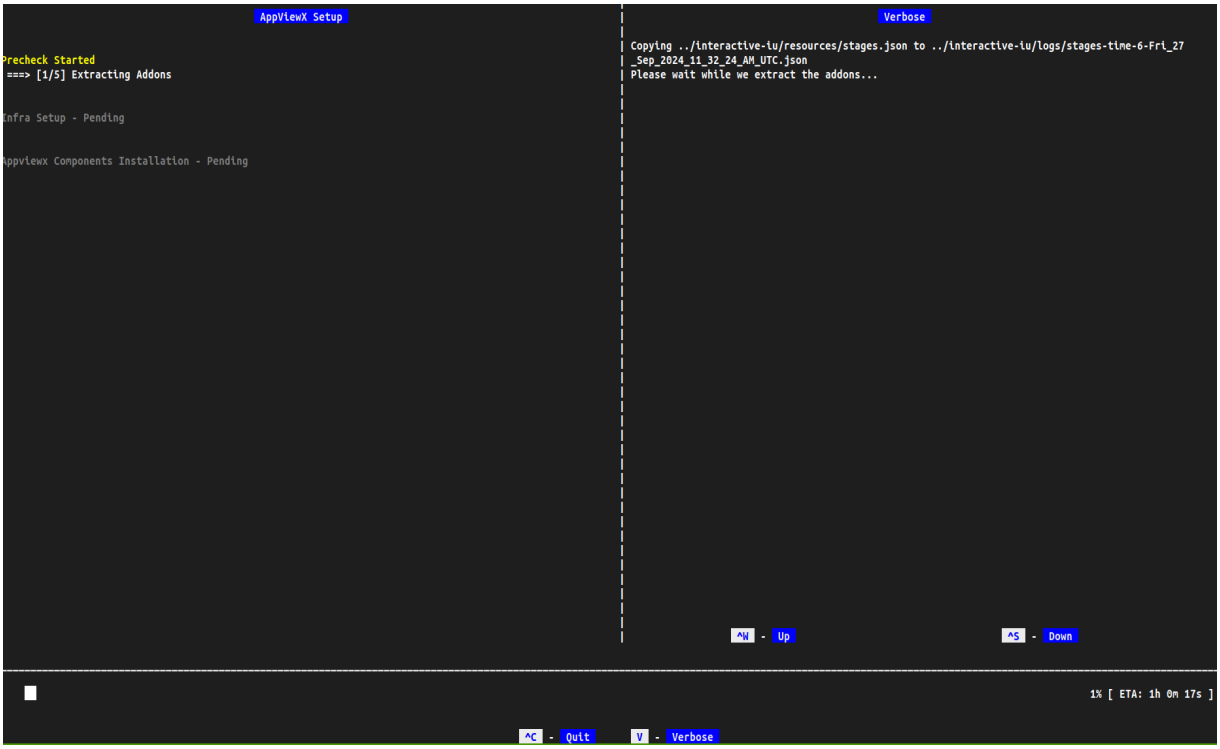
Validation	Nodes	Reason	Mitigation
Validating Disk Space: ⚠(Warning)		Insufficient disk space on installation path (103 GB available). Minimum recommended: 250GB.	Insufficient disk space available on worker node Recommended minimum: 250GB
Validating Disk Space: ⚠(Warning)		Insufficient disk space on installation path (124 GB available). Minimum recommended: 250GB.	Insufficient disk space available on worker node Recommended minimum: 250GB
Validating Disk Space: ⚠(Warning)		Insufficient disk space on installation path (105 GB available). Minimum recommended: 250GB.	Insufficient disk space available on worker node Recommended minimum: 250GB

Note: Installing these prerequisite packages requires internet access or proxy access to download and install the missing packages.

9. Once all the prerequisites are validated, the AppViewX installation will be triggered.



10. To view the verbose logs, press '**v**.'



Terminal UI Upgrade

Note: Ensure that a valid **other_user_internal.pem** file is present in the <appviewx installer directory>/scripts directory of the old installer for the in-place and blue-green deployments, where the existing setup is live.

1. To start the upgrade process, use the installer command:

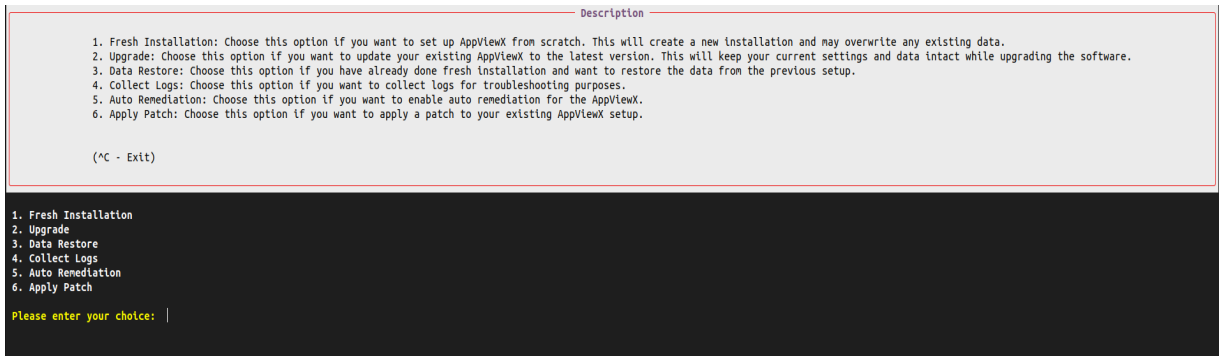
```
./install.sh
```

A welcome screen is displayed with options to Continue with installation or Exit.

2. Select **Continue**.

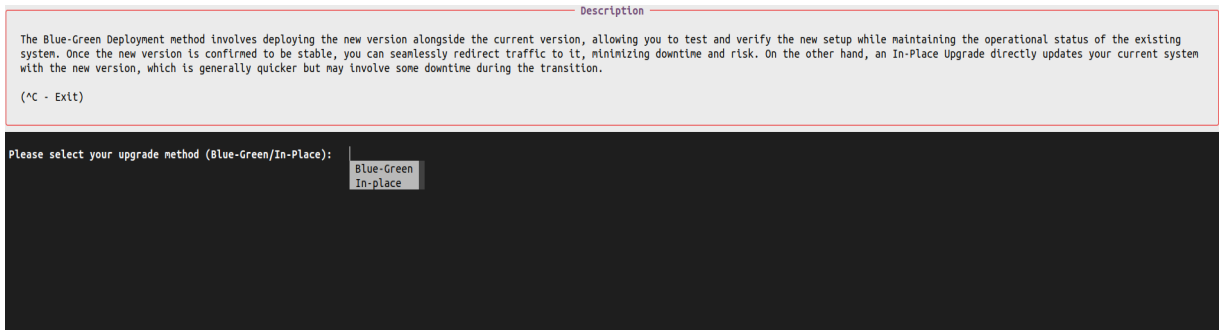
   The screen shows three options

   - 1. Fresh Installation
   - 2. Upgrade
   - 3. Data Restore
   - 4. Collect Logs
   - 5. Auto Remediation
   - 6. Apply Patch



3. To upgrade the AppViewX application, choose option **2**.

   The screen displays a prompt "*Please select your upgrade method (Blue-Green/In-Place):*"



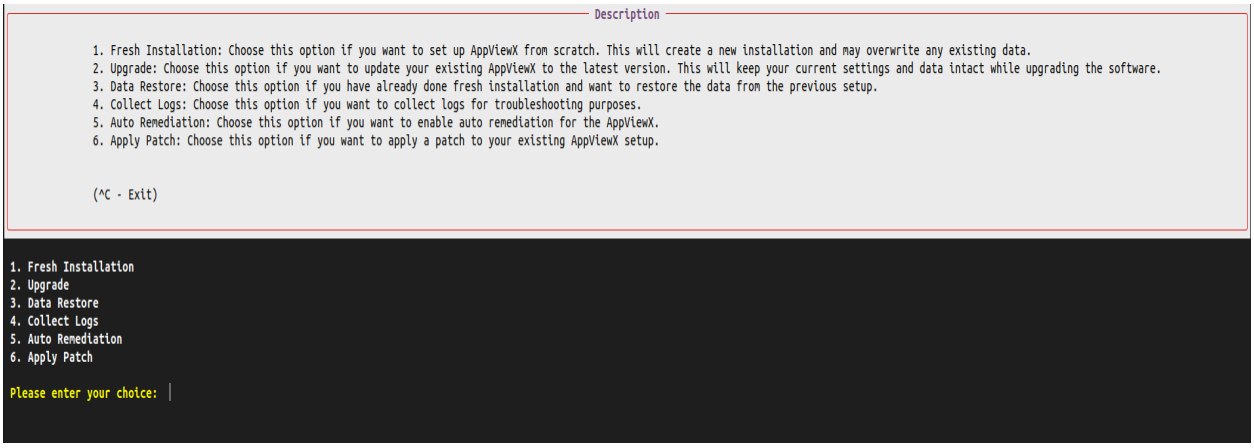
4. Now, choose between two types: Blue-Green Upgrade and In-Place Upgrade.
   Choose the upgrade option based on the current transition type:

> **Note:** All prerequisite validations will be carried out during the upgrade, and the `appviewx.conf` file can be modified during the interactive question phase.

- **Blue-Green**: If this option is selected, prepare the previous installer node details, installer path, and decide whether to take new hot backups or use an existing backup for the migration. There are two options
  - Select **Yes** if you are retaining the same design, where both the existing and new clusters have an identical number of nodes, to migrate without modifying the configuration.
  - Select **No** to allow the system to automatically generate the `appviewx.conf` file based on the inputs provided.
- **In-Place**: This option involves downtime, so ensure that the necessary downtime is planned and approved before proceeding with the in-place upgrade. The in-place upgrade retains the existing configuration, but validations will take place during the interactive questions and confirmations. You may add or remove nodes during the interactive questions phase.

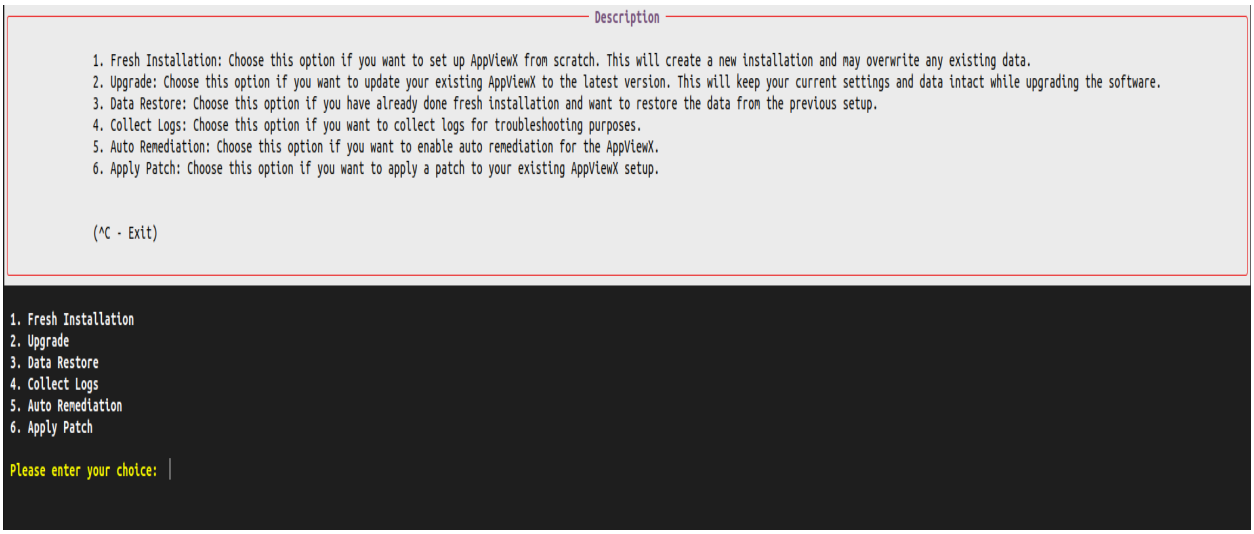
Terminal UI Data Restore

This option should be used when the customer is ready to proceed with the final database and vault restoration in preparation for the cut-over and go-live phase. Choose option 3 for Data Restore.



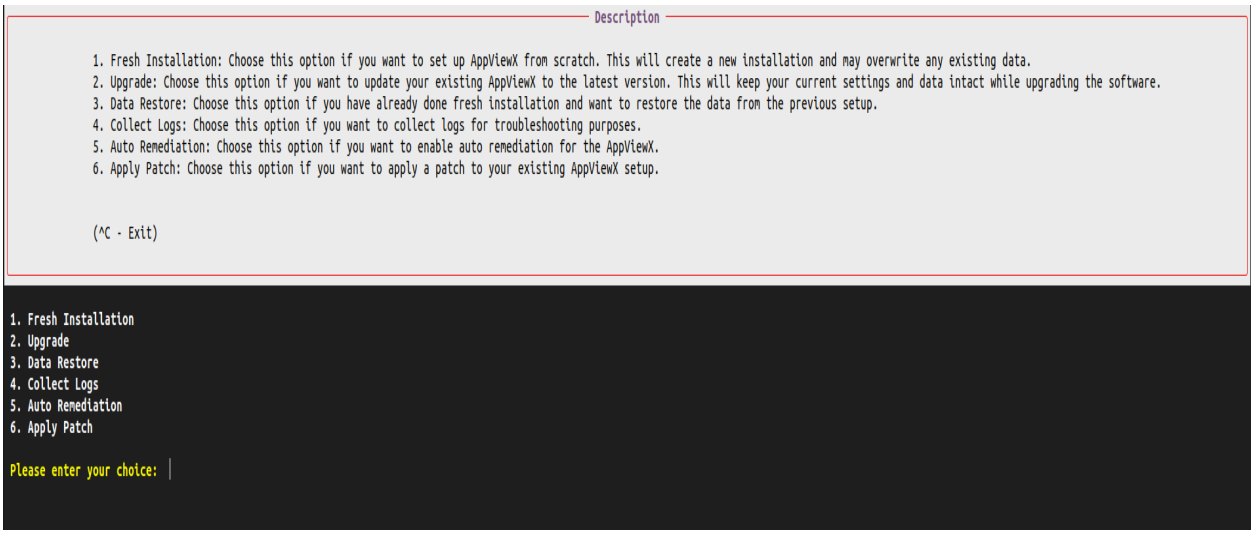
Terminal UI Collect Logs for Troubleshooting

This option should be used to gather application logs, system logs, and other relevant logs for troubleshooting in case of any issues with the application. Choose option 4 for collecting logs.



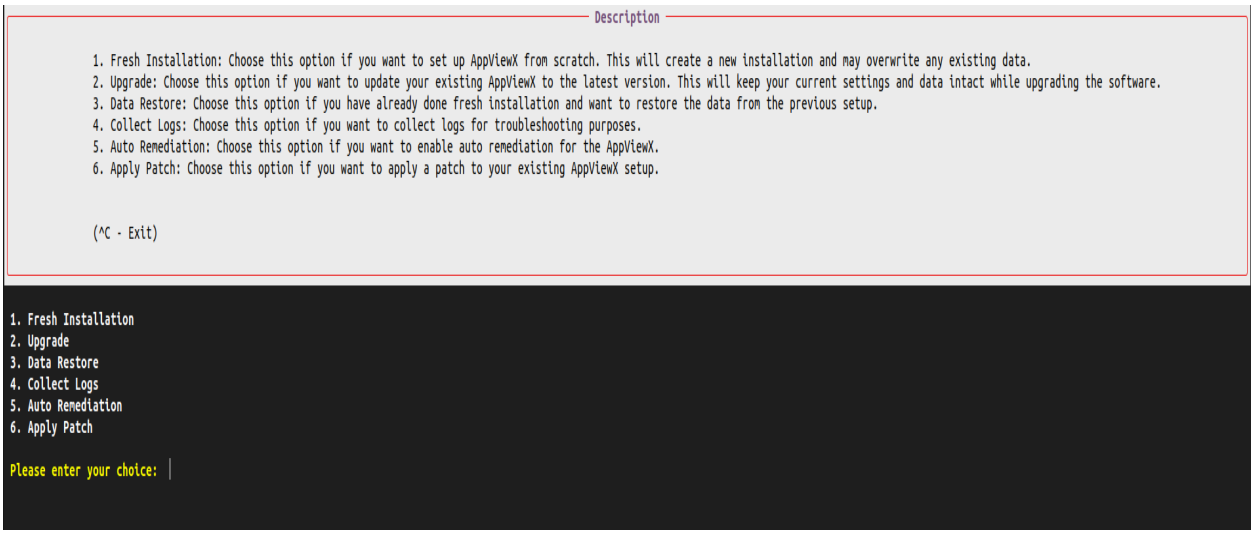
Terminal UI Auto-Remediation

This option should be used during an outage in the customer environment. It scans the entire cluster, performs remediation if any issues are detected, and collects logs for root cause analysis (RCA). Choose option 5 for Auto-Remediation.



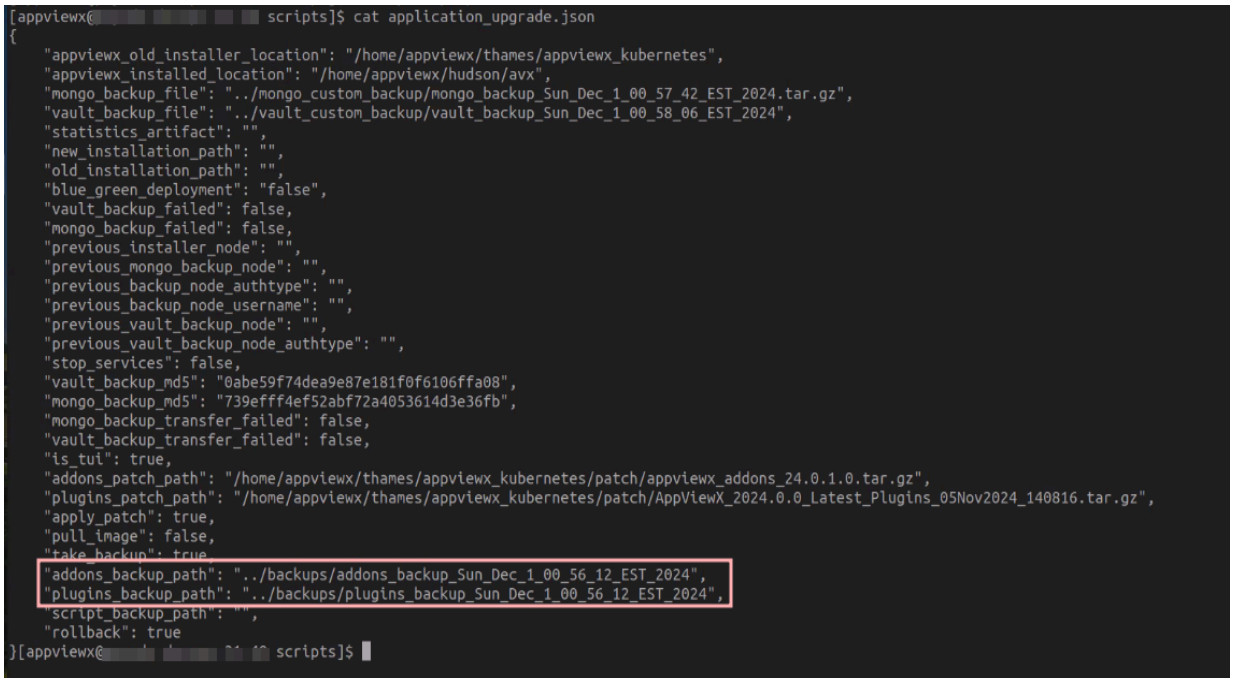
Terminal UI Apply Patch

This option should be selected when the customer is ready to apply the latest plugins and addons patch, incorporating the newest bug fixes and application enhancements. Choose option 6 for applying the patch.

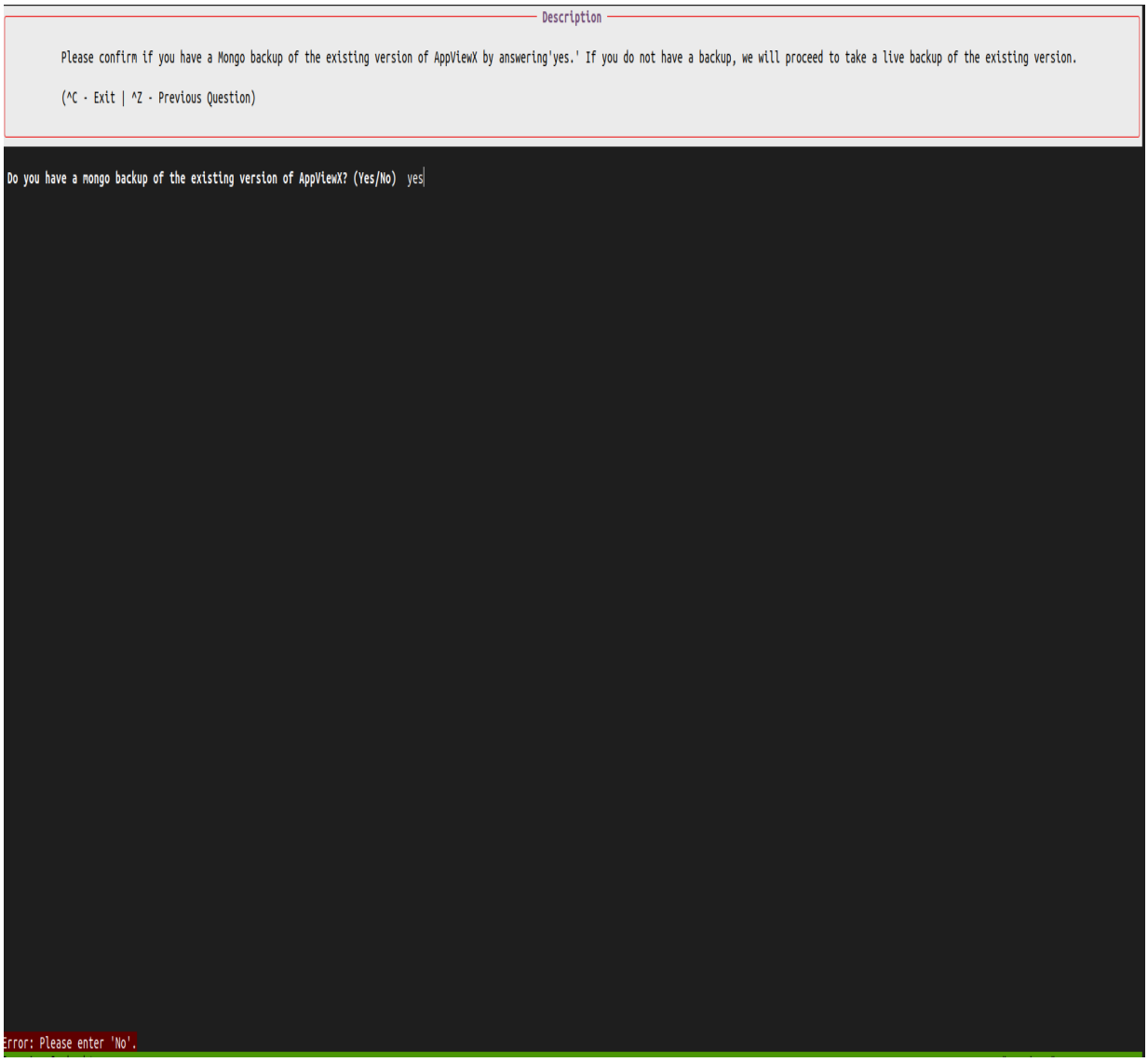


Points to Remember

- Do not use the appviewx_kubernetes installer if it was previously used during the upgrade process. During the upgrade, the initial seed dump file is replaced with the user-provided backup file, which could lead to issues if the same installer is reused for a fresh installation. To ensure a smooth process, always use a new appviewx_kubernetes installer for fresh installations.
- The file <installer path>/interactive-iu/resources/requirements.txt located inside the **interactive-iu** folder is just a reference for prerequisites validation during the upgrade. Do not use it for package installation.
- If auto-enrolment plugins are selected during the installation phase, the external gateway must be installed manually.
- Navigating back to previous questions after reaching the additional questions requires exiting and restarting the process.
- Latency warnings during prerequisite checks can be ignored. However, it is recommended to manually verify latency using the `ping` command for confirmation.
- For a patch rollback, update the `addons_backup_path` and `plugins_backup_path` in the `application_upgrade.json` file located in the `<appviewx_installer>/scripts` folder if multiple backups are present. Backup paths can be found in the `<appviewx_installer>/backups` directory.



- Additional questions have been introduced during the production cut-over; however, they are consistently restricted to "No," ensuring they do not impact the cut-over process.



Chapter 2: AppViewX SaaS Setup Guides

- SaaS Architecture Guide
- AppViewX SaaS Onboarding and Getting Started Guide
- AppViewX Cloud Connector User Guide

SaaS Architecture Guide

This guide will walk you through the architecture used by AppViewX to implement SaaS. It covers multi-tenant architecture, network architecture, and cluster architecture. Information with respect to scaling of clusters, DB isolation for each tenant, and high availability of AppViewX with the help of this architecture has been touched upon in this guide.

- Key Highlights of AppViewX Software as a Service
- AppViewX Architecture
- Multi-Tenancy Architecture
- SaaS Deployment Architecture

Key Highlights of AppViewX Software as a Service

The AppViewX Security Automation and Orchestration Platform is a centralized control plane to automate tasks, orchestrate workflows and gain visibility to manage identities at scale, reduce security and compliance risk and ensure secure application availability

The AppViewX SaaS platform offers the following products:

- CERT+, which lets you:
  - Discover, monitor, analyze, orchestrate and fully automate certificate lifecycle management and key management solutions.
  - Make a shift from reactive mode and be more proactive as you get a complete view of your entire certificate infrastructure.
  - Manage certificates as a service with pre-built integrations and extensible APIs that plugin to your enterprise applications, web servers, microservices, and multi-cloud environments.
  - Analyze certificates for crypto standards like key size, cipher strength, and allowed protocol versions.
  - Setup policies for enforcing high crypto standards.
  - Update certificates as per new policies.

- Provision certificates for devices and applications.
- Save resources, time, and effort of installation and maintenance.

For details, refer the CERT+ User Guide.

- ADC+, which lets you:
  - Efficiently distribute network load or client requests across servers.
  - Send requests to the available servers, ensuring high application availability.
  - Scale the number of servers (up or down) based on the traffic.

  For details, refer the ADC+ User Guide.
- PKI+, which lets you:
  - Create root CAs and subordinate CAs and enroll them to the AppViewX PKIaaS certificate authority.
  - Onboard custodians to add root CAs and subordinate CAs to the PKI+ system.
  - Manage custodians for approving PKI+-related actions.

  For details, refer the PKI+ User Guide.
- SSH+, which lets you:
  - Discover and display SSH certificates alongside SSH keys, offering a more comprehensive overview of your security credentials.
  - Download keys for key-based access control, ensuring streamlined access management.
  - Specify access duration in either hours or days when requesting access to an infrastructure group, providing enhanced access management control.
  - Use a dynamic access flow that adapts to either key or certificate-based access, depending on the user's selected 'Access Mode' during host addition.
  - Rotate host certificates effortlessly, directly from the host inventory, promoting secure host certificate management.
  - Revoke SSH certificates directly, thus enhancing security control.
  - Choose between 'Key' and 'Certificate' access modes during host addition, with the 'Certificate' option being pre-selected by default.
  - Rotate and delete keys from hosts with multiple keys through the user and host key age report.

  For details, refer the SSH+ User Guide.
- SIGN+, which lets you:
  - Simplify Code Signing Certificate enrollment and Certificate Lifecycle Management (CLM) operations.
  - Customize signing policies according to your requirements
  - Integrate with AppViewX's customized Cryptographic Service Provider (CSP) and PKCS#11 for enhanced security.
  - Manage your code signing inventory with a full suite of tools and features.

- Sign your code effortlessly using a variety of tools including SignTool, JSign, JarSigner, APKSigner, Mage, and Nuget.
- Ensure compatibility with third-party Timestamp Authorities (TSA) for a wider range of options.

For details, refer the SIGN+ User Guide.

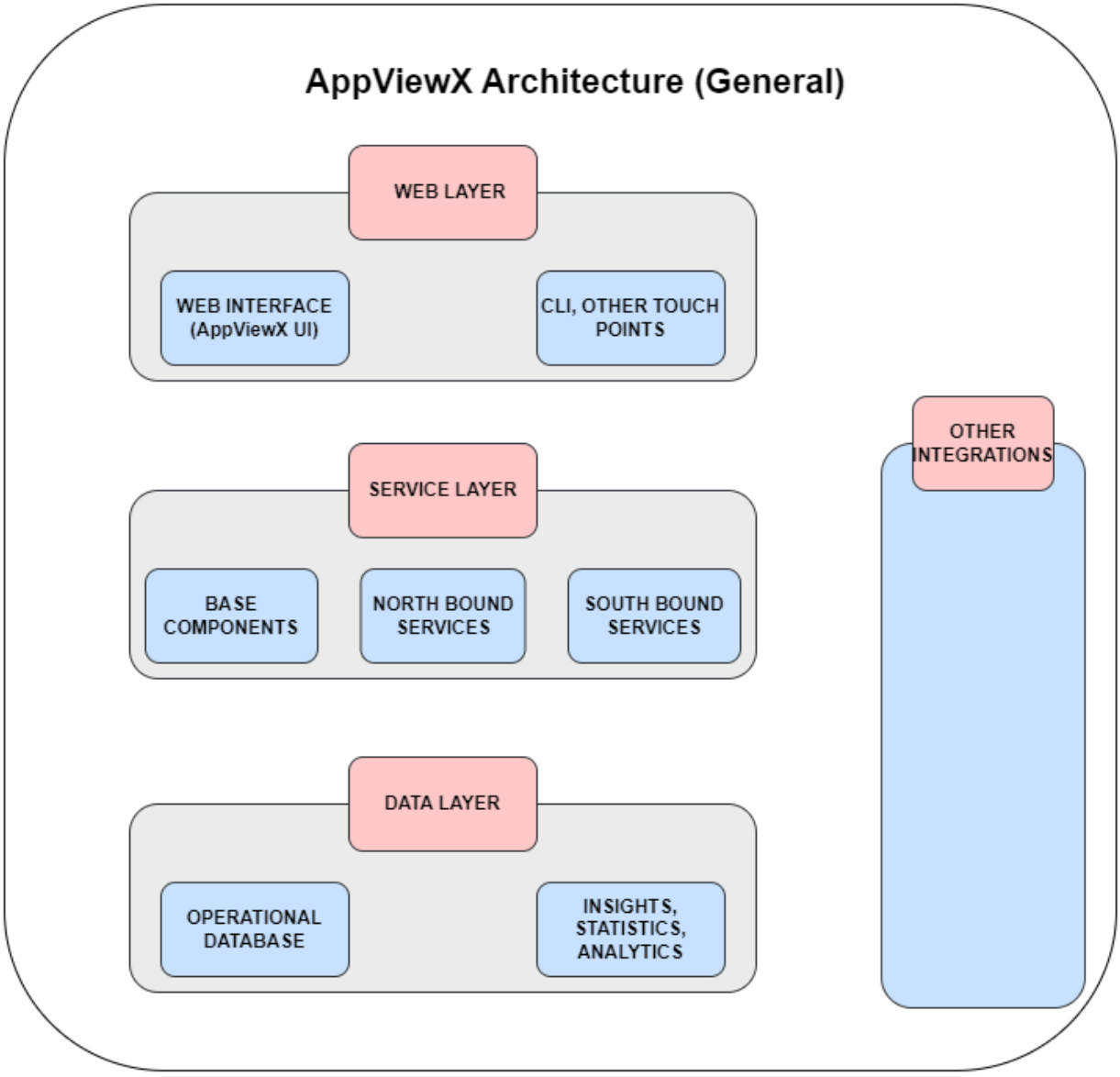
- KUBE+, which lets you:
  - Simplify Certificate Lifecycle Management for Kubernetes workloads.
  - Get real-time visibility, central audit, and governance over K8's Certs.
  - Achieve end-to-end automated certificate enrollment process.
  - Have secure and compliant PKI across K8s workloads (secrets, pods, and service mesh).

For details, refer the KUBE+ User Guide.

AppViewX Architecture

AppViewX is designed based on microservice architecture and its deployed on Kubernetes, an open-source platform for deploying and managing containers. The microservice architecture of AppViewX makes it easier to move to containerized workloads and the containers being orchestrated using Kuberneters. Kubernetes provides container runtime, orchestration, self-healing mechanisms, service discovery and load balancing and its used for the deployment, scaling, management, and composition of application containers across clusters.

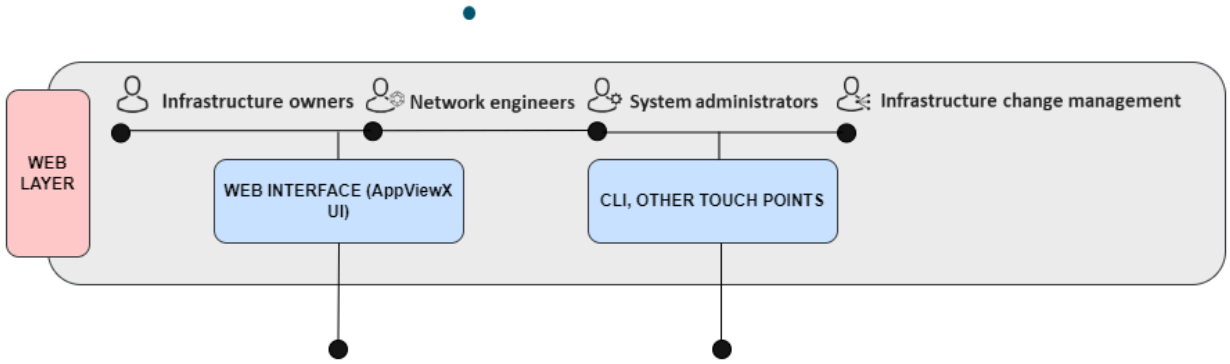
High Level Architecture



Understanding the AppViewX Architecture

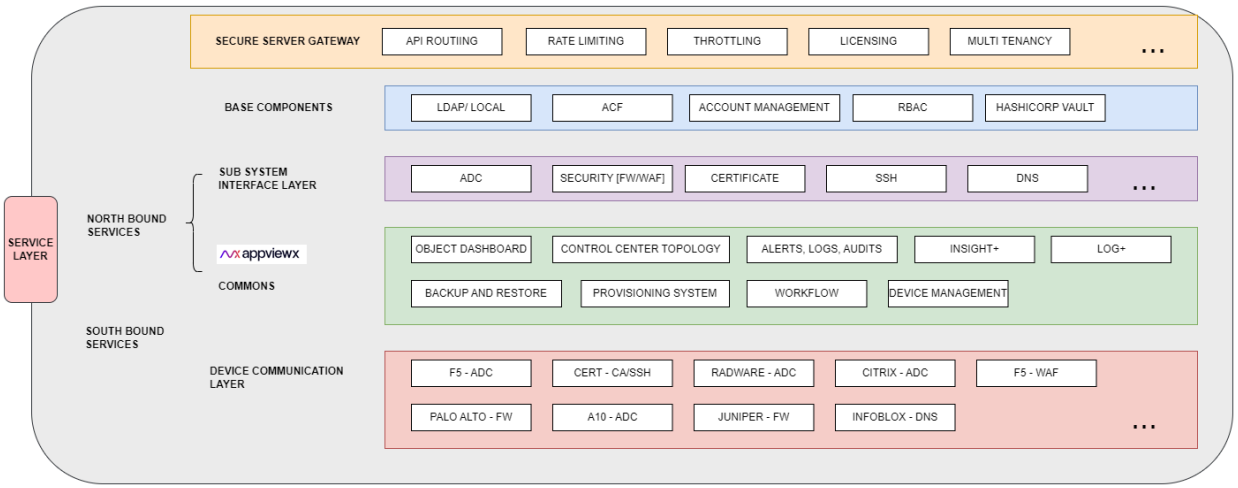
Web Layer

The web layer includes services for user interaction.



Service Layer

The service layer houses the core AppViewX business logic that is responsible for fetching user inputs from the UI. The AppViewX application then uses these inputs to perform CLM operations for the end devices. The responses thus received are persisted in the database.

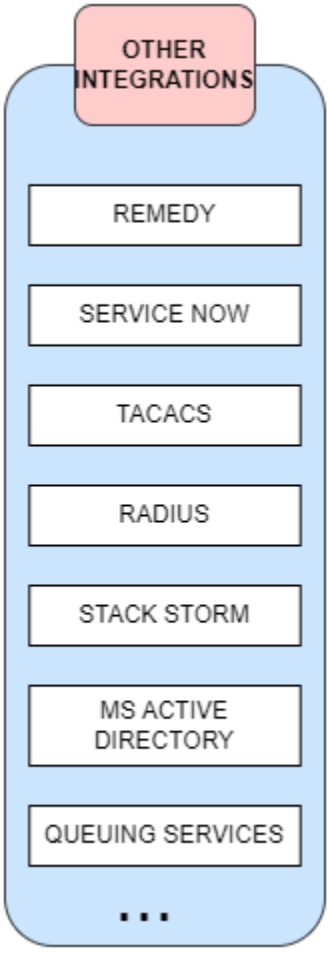


Data Layer

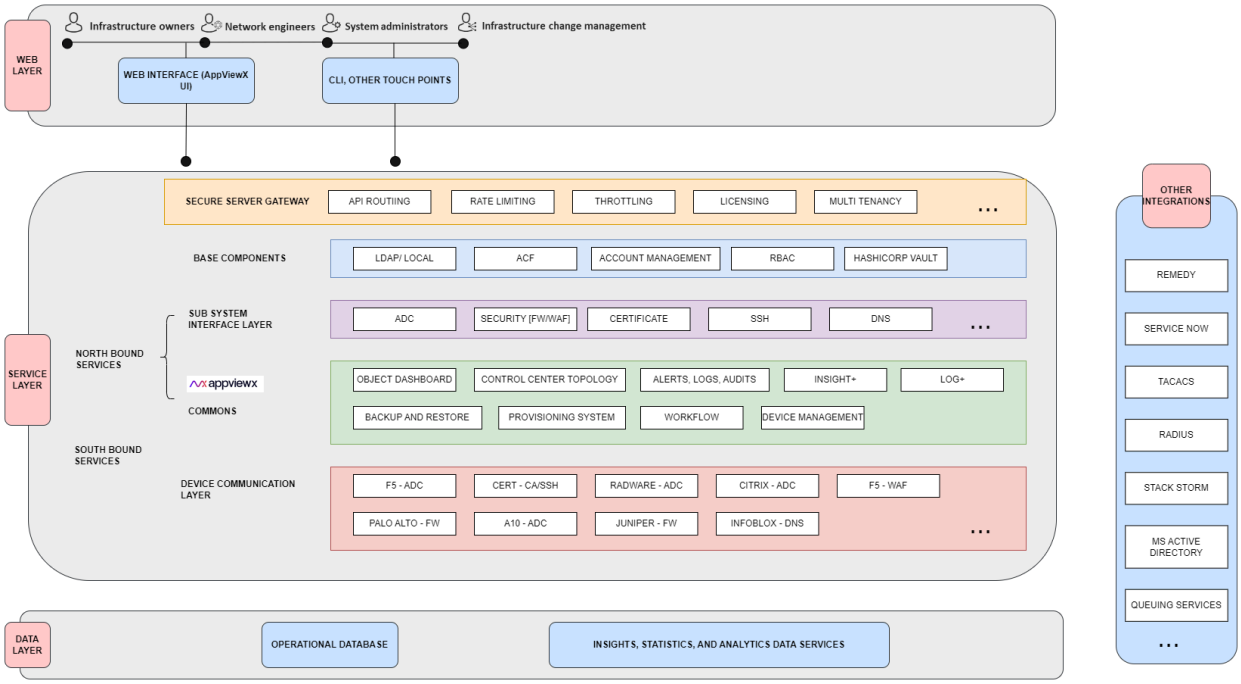
The data layer houses the persistence logic for the application. The data persistence logic is responsible for backing up the data in the application's file system. In events of a data loss, the data can then be retrieved from the file system.

Other Integrations

Other Integrations are out-of-the-box ticketing, authorization, and authentication tools that AppViewX supports integration with.



Low Level Architecture

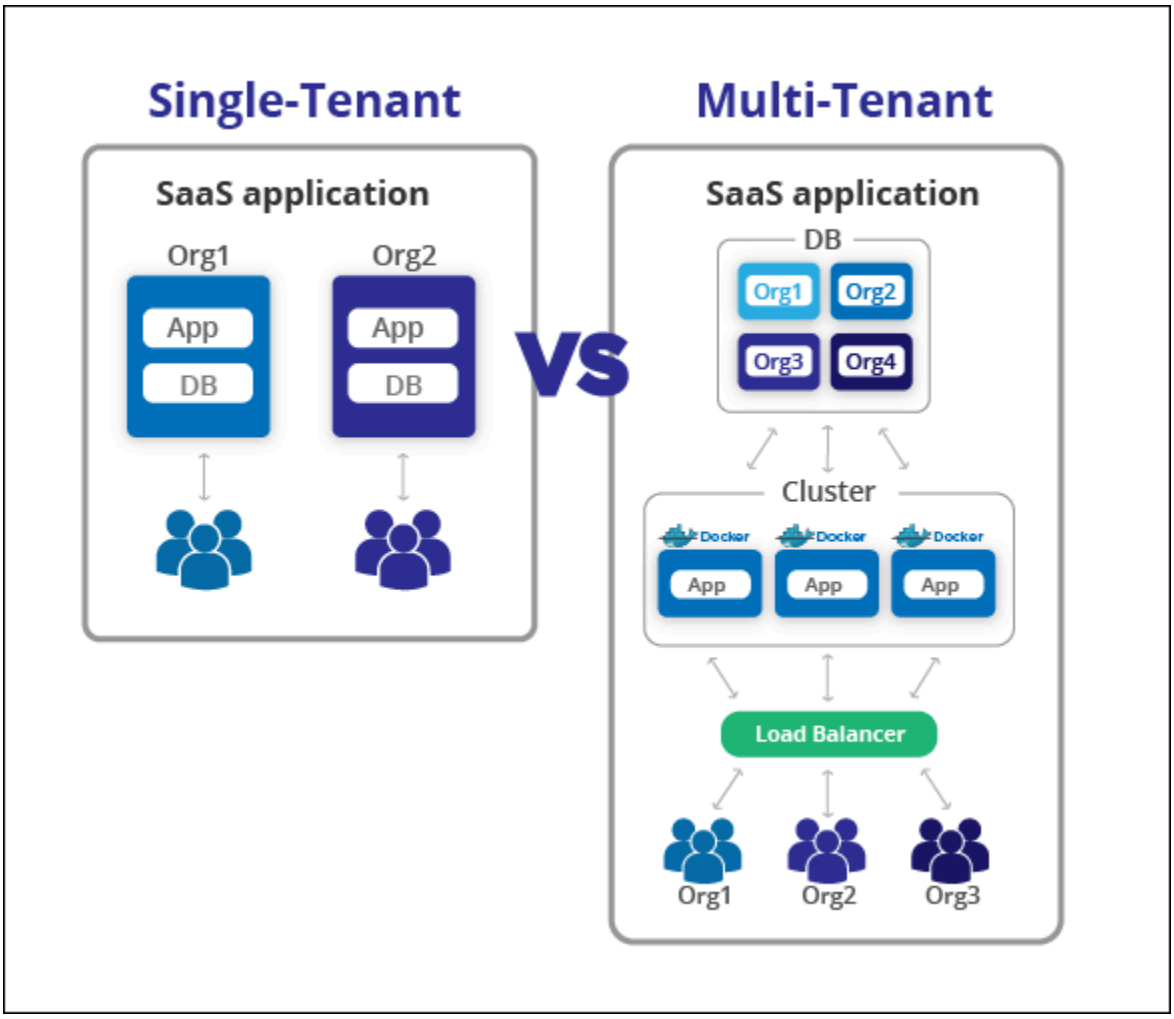


Multi-Tenancy Architecture

In a multi-tenancy architecture, a single instance of the software serves multiple accounts/customers. In this setup, the same resources -compute, networking and storage - are shared on the cloud among tenants.

In this ecosystem, a single environment can serve multiple tenants utilising a scalable, available, and resilient architecture. The underlying infrastructure is completely shared, logically isolated, and with fully centralised services.

AppViewX multi-tenant architecture is enabled by a shared compute cluster or a workload cluster where the workloads run and a database cluster where the actual tenant isolation happens by allocating a dedicated schema for each and every tenant. The diagram below depicts the multi-tenant implementation.



Multi-Tenant Architecture

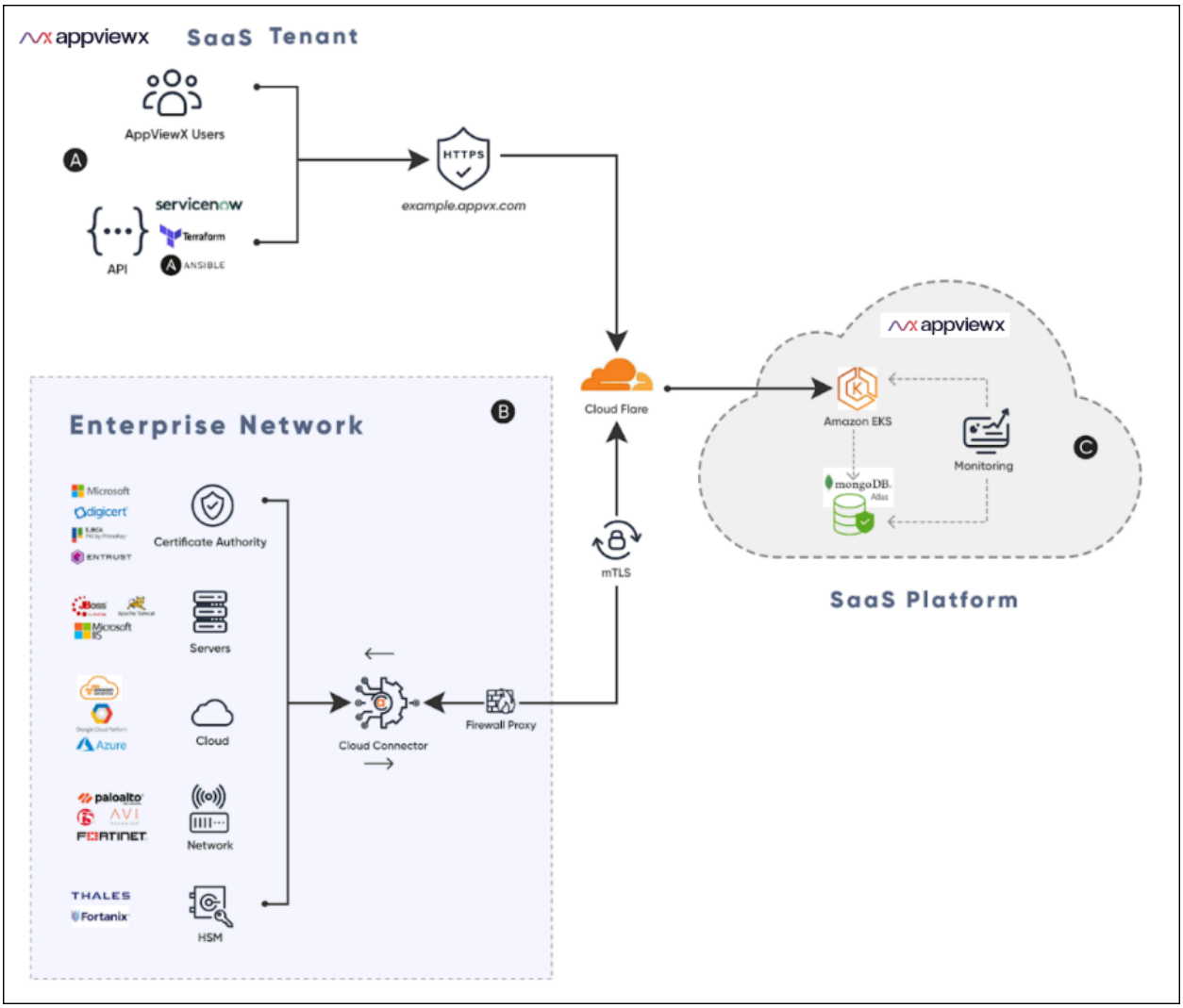
SaaS Deployment Architecture

SaaS Deployment Architecture

The AppViewX SaaS deployment architecture is a cloud-based deployment with the following benefits.

• Lower cost of ownership (TCO), significantly reduced maintenance.
• Guaranteed availability (SLA), and enhanced data security

- Faster release cycles and upgrades to access new offerings.
- Avoid installation of the entire AppViewX infrastructure in the tenant network.



AppViewX Multi-Tenancy Architecture

At a high level, the SaaS deployment architecture consists of:

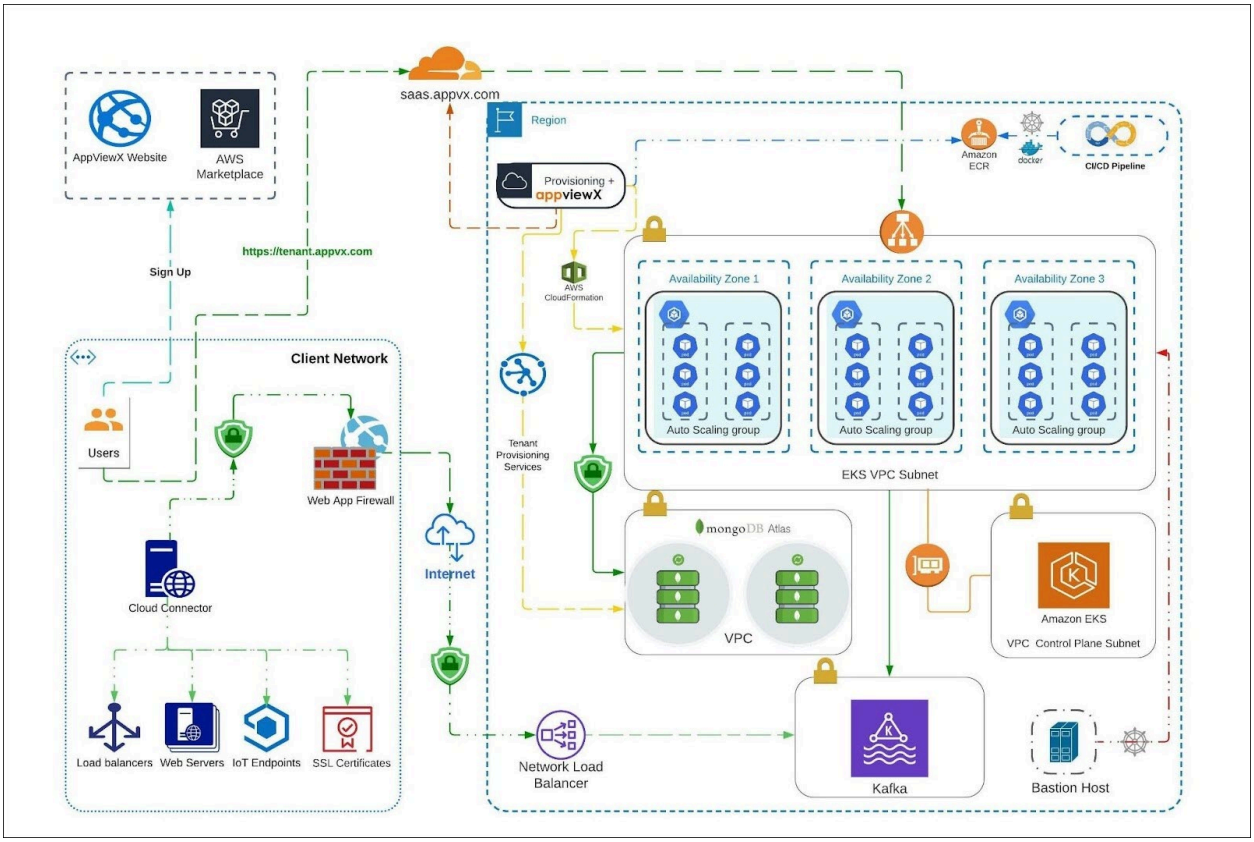
- A public access to the AppViewX SaaS products' tenant secured via https.
- AppViewX Cloud Connector (a lightweight proxy) deployment that enables connectivity between the SaaS platform to the Enterprise network thereby ensuring faster value realisation of critical Certificate Lifecycle Management (CLM) functions such as Discovery, visibility, Automation and self-servicing of SSL / TLS certificates from the tenants infrastructure.
- AppViewX SaaS platform that enables the server-side components such as database, compute, monitoring and tenant provisioning (which includes install and upgrades).

Architecture Components Overview

The AppViewX SaaS platform is enabled with the help of Provisioning, Compute, Database, Monitoring clusters and an AppViewX Cloud Connector.

The key tenets of the platform include:

1. Provisioning Cluster
2. Compute Cluster
3. Database Cluster
4. Monitoring Cluster
5. AppViewX Cloud Connector



Deployment Architecture

- Provisioning Cluster (SaaS Management Portal)
- Compute Cluster
- Database Cluster

- Monitoring Cluster
- The AppViewX Cloud Connector

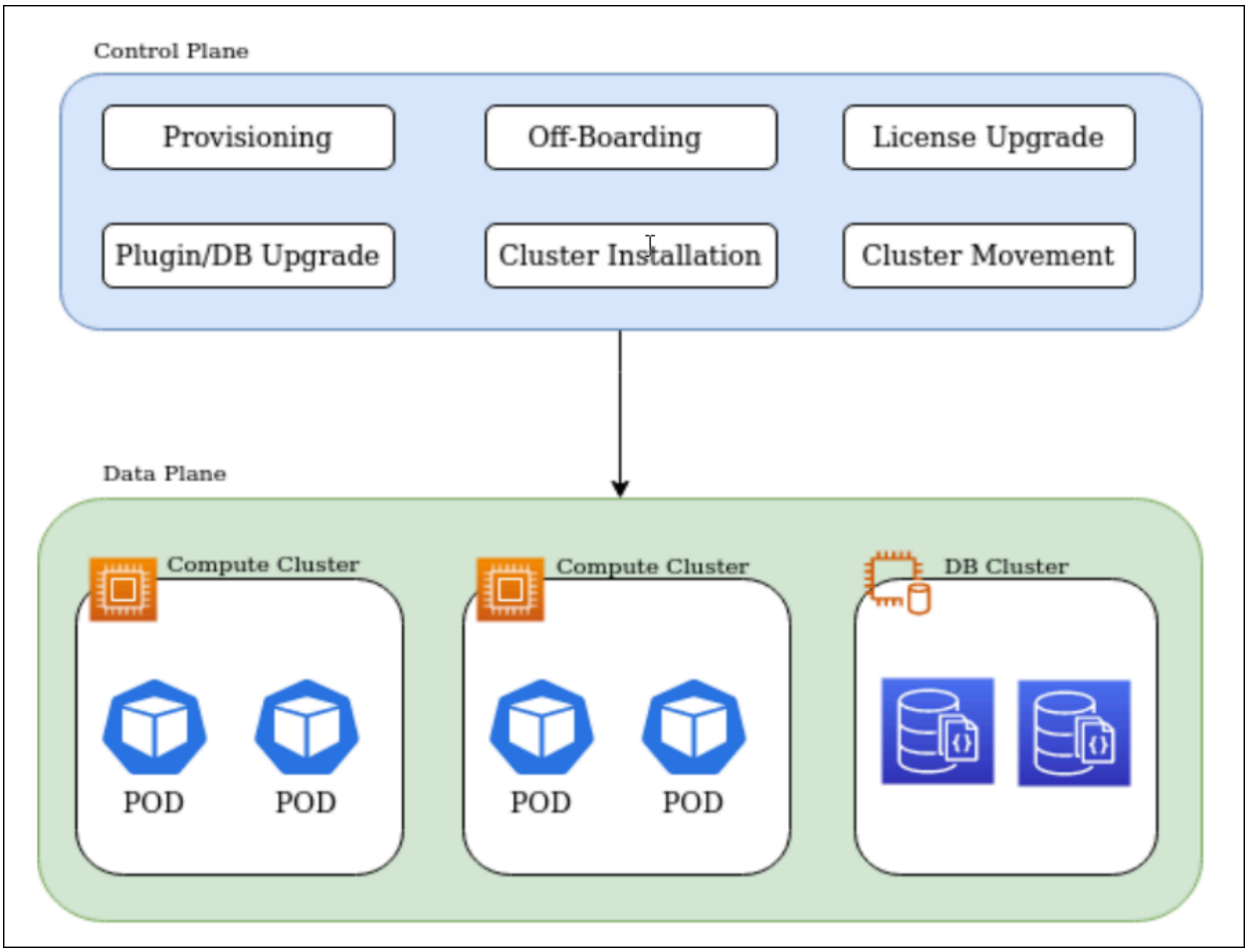
Provisioning Cluster (SaaS Management Portal)

Provisioning Cluster (aka SaaS Management portal) is a cluster used to orchestrate the compute and database instances that powers the AppViewX SaaS. The cluster provides key capabilities like management and visibility into tenants and clusters which includes licensing, upgrading the SaaS tenants and so on, from a Single pane of glass. The granular features and capabilities of the portal are explained below.

- The Provisioning cluster is an internal AppViewX platform for SaaS lifecycle management which can be deployed cross zone or cross region for high availability.

> **Note:** The cluster can be deployed in a dedicated AWS account (not necessary to be deployed in the AWS account where the actual compute cluster is deployed).

- Key features of the provisioning cluster include:
  1. Tenant Life Cycle Management - Onboarding, Offboarding, Licence upgrade.
  2. Cluster Management - Create , Delete , Modify, Upgrade Compute clusters
  3. AppViewX SaaS Life Cycle Management - Install AppViewX on Compute, Upgrade AppViewX (via Canary upgrades), Install Infrastructure components (Istio , ELK etc.,)
- The Provisioning Cluster uses cloudformation templates for creating the Compute cluster and AppViewX automation workflows for mapping the DB clusters with the compute and the other process tied to AppViewX SaaS life cycle management.



AppViewX Provisioning Cluster Architecture

Compute Cluster

AppViewX compute cluster is a managed-compute infrastructure that runs the AppViewX business logic. The compute cluster is powered by Amazon Elastic Kubernetes Service (Amazon EKS) which is a managed AWS Kubernetes service that scales, manages, and deploys containerized applications.

The compute cluster is deployed via the Provisioning cluster using the cloud formation template. Compute cluster encompasses the below.

- EKS
- AppViewX workloads
- Infrastructure components
- Bastion Host

EKS Cluster

EKS clusters are composed of the following main components—a control plane and worker nodes. Each cluster runs in its own, fully managed Virtual Private Cloud (VPC).

The control plane is composed of three master nodes, each running in a different AZ to ensure AWS high availability. Incoming traffic directed to the Kubernetes API passes through the AWS network load balancer (NLB).

Worker nodes run on Amazon EC2 instances located in a VPC. EKS provides managed node groups with automated lifecycle management. This lets users automatically create, update, or shut down nodes with one operation. EKS uses Amazon's latest Linux AMIs optimised for use with EKS. When nodes are terminated, EKS gracefully drains them to make sure there is no interruption of service.

High Availability

Amazon EKS runs and scales the Kubernetes control plane across multiple AWS Availability Zones to ensure high availability. Amazon EKS automatically scales control plane instances based on load, detects and replaces unhealthy control plane instances, and automatically patches the control plane.

The EKS cluster consists of EC2 instances deployed in multiple availability zones within the region. Each instance has replicas of the services and nodes which exist across all the EC2 instances.

Each zone or instance has an active pod listening to other instances. In case of a failure of any instance, the active pod ensures seamless functioning of the application by activating the nodes from any other working cluster.

Note: EKS clusters are deployed within specific regions and each region has multiple availability zones. Example - Region : us-east-1 and the respective zones : us-east-1a, us-east-1b, us-east-1c.

AppViewX Workloads & Infrastructure Components

AppViewX workloads are containerized workloads running as microservices and these containers are orchestrated using Amazon Elastic Kubernetes Service (Amazon EKS).

The workloads are a mix of AppViewX Business logics that enable communication from User Interface to AppViewX core services and AppViewX SaaS services which is used for enabling the SaaS communication from the AppViewX's SaaS compute to the customer network.

The infrastructure components encompasses third party components that are used for the purpose of service mesh, log aggregation and monitoring the utilisation of the application workloads and so on.

All these workloads, infrastructure components are deployed from the provisioning cluster.

Bastion Host

A bastion host is another EC2 instance based on Linux OS which is created on the same VPC of the wokernodes and this is used for cluster admin operations and troubleshooting the application if required.

The bastion host is accessed via SSH keys generated during the EKS cluster creation and each and every cluster have their own SSH key and the key is downloaded only from the AppViewX SaaS provisioning cluster.

Database Cluster

AppViewX Database cluster is a managed database infrastructure of AppViewX SaaS which holds the customer data. The database cluster is powered by MongoDB Atlas which brings together capabilities that are critical to a modern, cloud-native, microservice-aligned database architecture, including scalability, availability, and uptime.

The AppViewX Database Cluster is enabled via MongoDB Atlas which is a global cloud document database service. The Atlas service MongoDB ensures availability, scalability, and security compliance. The granular features of this cluster are:

- A single database with multiple schemas.
- Individual schemas are generated for each Licensed tenant.
- Snapshots are created for the licensed tenant in the DB cluster and each of the snapshots contains mandatory schemas such as :
  - appSession
  - appviewx
  - appviewxCA

- These schemas are created before the tenants are onboarded.
- Apart from the three mandatory schemas, Snapshot Ids are created for the following schemas:
  - connectedPlatform
  - imageDetails
  - templateDB
  - workFlowDB
  - workFlowDBEn..
- The tenant data is secured and isolated due to this segregation of schemas.
- It also ensures the singularity of data for each licensed tenant.

The AppViewX Database Cluster is made highly available by enabling the cluster deployment on multiple zones or even more resilient by enabling the cluster deployment on multiple regions. Each of these Clusters have a unique URL and credential associated with it.

Monitoring Cluster

Monitoring Cluster is a managed infrastructure of AppViewX SaaS which caters to monitoring, and understanding the performance of the application and machine critical services which is a condensed form of metadata, metrics, and events about the application and its underlying services. This is enabled with a monitoring stack comprising Prometheus, Grafana, Loki, Promtail, and AlertManager.

The monitoring cluster has a Status dashboard and is deployed in a separate cluster which is again a subset of AppViewX powered with AppViewX monitoring capability enabled via Prometheus, Grafana, Loki, Promtail, and AlertManager and it is deployed on AWS (like an onprem AppViewX deployment) with its own database, compute etc which can be deployed cross zone or cross region for high availability.

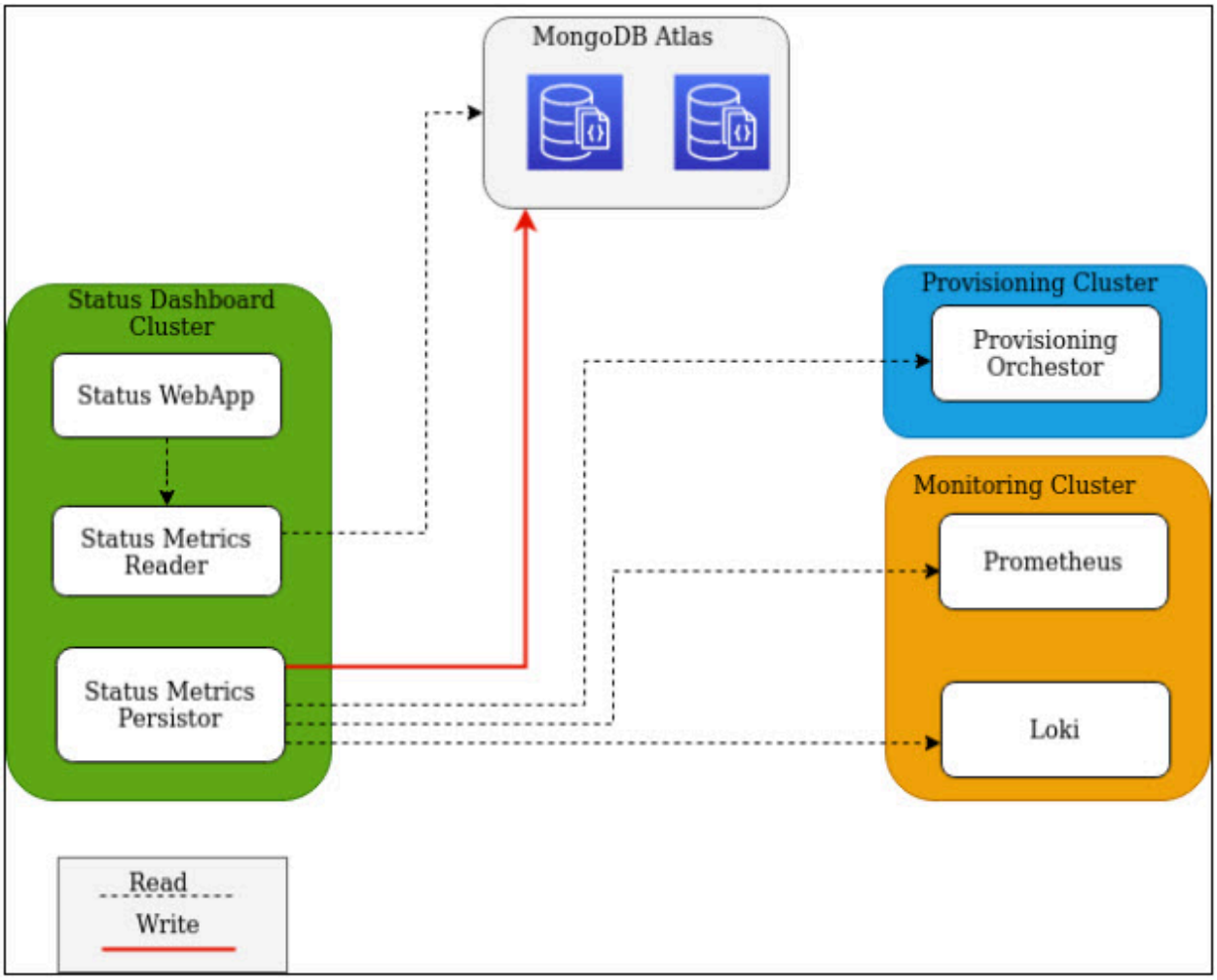
The status dashboard application will have three microservices

- Status dashboard Webapp
- Status Metrics Reader
- Status Metrics Persistor

The Webapp talks to Status metrics Reader and displays AppViewX services uptime details.

The Status Metrics Reader reads data from a dedicated instance residing in MongoDB Atlas.

The Status Metrics Persitor aggregates data from Monitoring and Provisioning clusters and save them in the dedicated instance which resides in MongoDB Atlas.



Monitoring Cluster Architecture

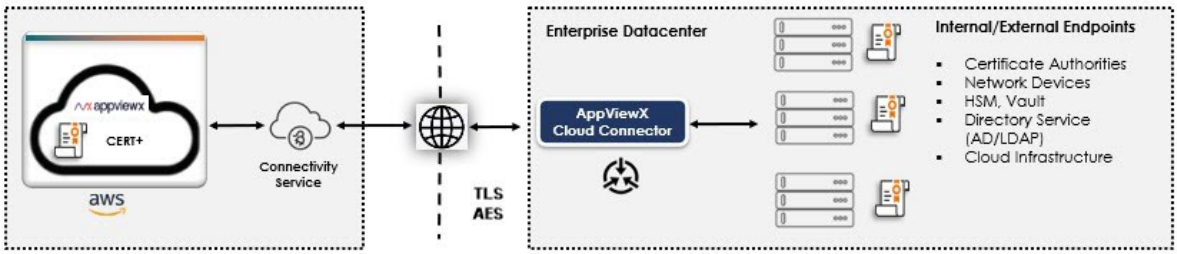
The AppViewX Cloud Connector

AppViewX Cloud Connector is a lightweight plug-in that establishes connectivity between AppViewX Cloud and the Enterprise Network. The cloud connector serves as a secure channel for communication between AppViewX SaaS and your enterprise network without requiring any complex network or infrastructure configuration.

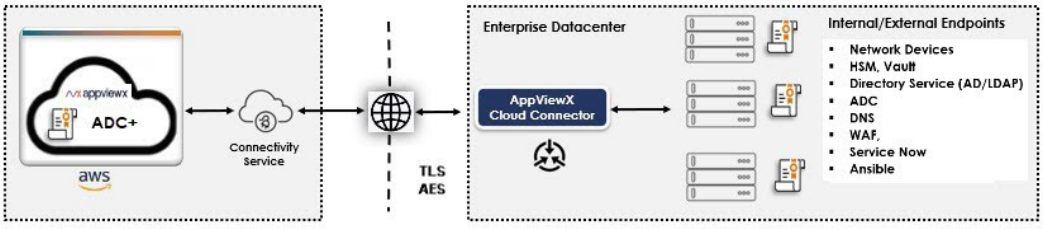
Services that require the AppViewX Cloud Connector for using the AppViewX products (examples):

- **CERT+**
  - Discovering certificates from an endpoint within the enterprise network via Smart Network Scan and Managed Device Scan.
  - Discovering certificates from Certificate Authorities (CAs) that are internal to the enterprise. For example : EJBCA.
  - Discovering certificates from public Certificate Authorities (CAs)

    In this case, AppViewX provides a default instance of the Cloud Connector called **cloud-dc**.



- **ADC+**:
  - Communicating with ADC devices and discover the Application Services from the ADC infrastructure
  - Gain Visibility and to fetch the real time state/status of the Applications discovered
  - Self Service the Applications to allow/deny traffic
  - Backup the configuration of the ADC devices
  - Restore the configuration of the ADC devices
  - Automate and Orchestrate the ADC configuration within and across devices



Key features of the AppViewX Cloud Connector:

- A self-serviceable, Linux-based lightweight setup
- Secure communication between the AppViewX SaaS and the AppViewX Cloud Connector using TLS and AES encryption

- Connectivity from the AppViewX SaaS to the enterprises' network endpoints
- No complex network setup (Inbound Firewall Whitelisting, VPN setup, and so on)
- Features of the AppViewX Cloud Connector

Features of the AppViewX Cloud Connector

- AppViewX Cloud Connector DataCenter Significance
- Cloud Connector High Availability
- Integrated Gateway
- Custom Certificates for Core Communication
- Communication Authentication and Encryption
- Auto Enrollment with the AppViewX Cloud Connector
- Enabling Proxy for End Point Communication

AppViewX SaaS Onboarding and Getting Started Guide

This guide outlines the steps for onboarding customers to the AppViewX SaaS platform and enables them to get started with the AppViewX SaaS products.

- Key Highlights of AppViewX Software as a Service
- Introduction to the AppViewX Cloud Connector
- Prerequisites for Setting up AppViewX Cloud Connector
- Getting Started with the AppViewX Free Trial
- Signing Up for the Free Trial via the AppViewX Website
- Signing Up for the Free Trial via the AWS Marketplace

Key Highlights of AppViewX Software as a Service

The AppViewX Security Automation and Orchestration Platform is a centralized control plane to automate tasks, orchestrate workflows and gain visibility to manage identities at scale, reduce security and compliance risk and ensure secure application availability

The AppViewX SaaS platform offers the following products:

- CERT+, which lets you:
  - Discover, monitor, analyze, orchestrate and fully automate certificate lifecycle management and key management solutions.
  - Make a shift from reactive mode and be more proactive as you get a complete view of your entire certificate infrastructure.
  - Manage certificates as a service with pre-built integrations and extensible APIs that plugin to your enterprise applications, web servers, microservices, and multi-cloud environments.
  - Analyze certificates for crypto standards like key size, cipher strength, and allowed protocol versions.
  - Setup policies for enforcing high crypto standards.
  - Update certificates as per new policies.
  - Provision certificates for devices and applications.
  - Save resources, time, and effort of installation and maintenance.

  For details, refer the CERT+ User Guide.

- ADC+, which lets you:
  - Efficiently distribute network load or client requests across servers.
  - Send requests to the available servers, ensuring high application availability.
  - Scale the number of servers (up or down) based on the traffic.

  For details, refer the ADC+ User Guide.

- PKI+, which lets you:
  - Create root CAs and subordinate CAs and enroll them to the AppViewX PKIaaS certificate authority.
  - Onboard custodians to add root CAs and subordinate CAs to the PKI+ system.
  - Manage custodians for approving PKI+-related actions.

  For details, refer the PKI+ User Guide.

- SSH+, which lets you:
  - Discover and display SSH certificates alongside SSH keys, offering a more comprehensive overview of your security credentials.
  - Download keys for key-based access control, ensuring streamlined access management.
  - Specify access duration in either hours or days when requesting access to an infrastructure group, providing enhanced access management control.
  - Use a dynamic access flow that adapts to either key or certificate-based access, depending on the user's selected 'Access Mode' during host addition.
  - Rotate host certificates effortlessly, directly from the host inventory, promoting secure host certificate management.
  - Revoke SSH certificates directly, thus enhancing security control.

- Choose between 'Key' and 'Certificate' access modes during host addition, with the 'Certificate' option being pre-selected by default.
- Rotate and delete keys from hosts with multiple keys through the user and host key age report.

For details, refer the SSH+ User Guide.

- SIGN+, which lets you:
  - Simplify Code Signing Certificate enrollment and Certificate Lifecycle Management (CLM) operations.
  - Customize signing policies according to your requirements
  - Integrate with AppViewX's customized Cryptographic Service Provider (CSP) and PKCS#11 for enhanced security.
  - Manage your code signing inventory with a full suite of tools and features.
  - Sign your code effortlessly using a variety of tools including SignTool, JSign, JarSigner, APKSigner, Mage, and Nuget.
  - Ensure compatibility with third-party Timestamp Authorities (TSA) for a wider range of options.

  For details, refer the SIGN+ User Guide.
- KUBE+, which lets you:
  - Simplify Certificate Lifecycle Management for Kubernetes workloads.
  - Get real-time visibility, central audit, and governance over K8’s Certs.
  - Achieve end-to-end automated certificate enrollment process.
  - Have secure and compliant PKI across K8s workloads (secrets, pods, and service mesh).

  For details, refer the KUBE+ User Guide.

Introduction to the AppViewX Cloud Connector

AppViewX Cloud Connector is a lightweight plug-in that establishes connectivity between AppViewX Cloud and the Enterprise Network. The cloud connector serves as a secure channel for communication between AppViewX SaaS and your enterprise network without requiring any complex network or infrastructure configuration.

Key features of the AppViewX Cloud Connector include:

- Data center-based routing
- High availability
- Integrated gatway functionality
- Custom certificates for core communication
- Communication authentication and encryption

Refer to the AppViewX Cloud Connector User Guide, to read more on the features of the AppViewX Cloud Connector and the services it supports.

Prerequisites for Setting up AppViewX Cloud Connector

The AppViewX Cloud Connector can be set up in two ways: using the **virtual image** and via the **native OS**.

For the complete list of system requirements that are minimum prerequisites for setting up and operating the AppViewX Cloud Connector, click here.

Important: For installation via the virtual image, only the **hardware** and **server and network** prerequisites have to be ensured. The operating system and Docker prerequisites are packaged as part of the OVA.

Note: AppViewX provides you with a script for checking if the hostname meets all the installation prerequisites. For instructions on how you can download and execute this script, click here.

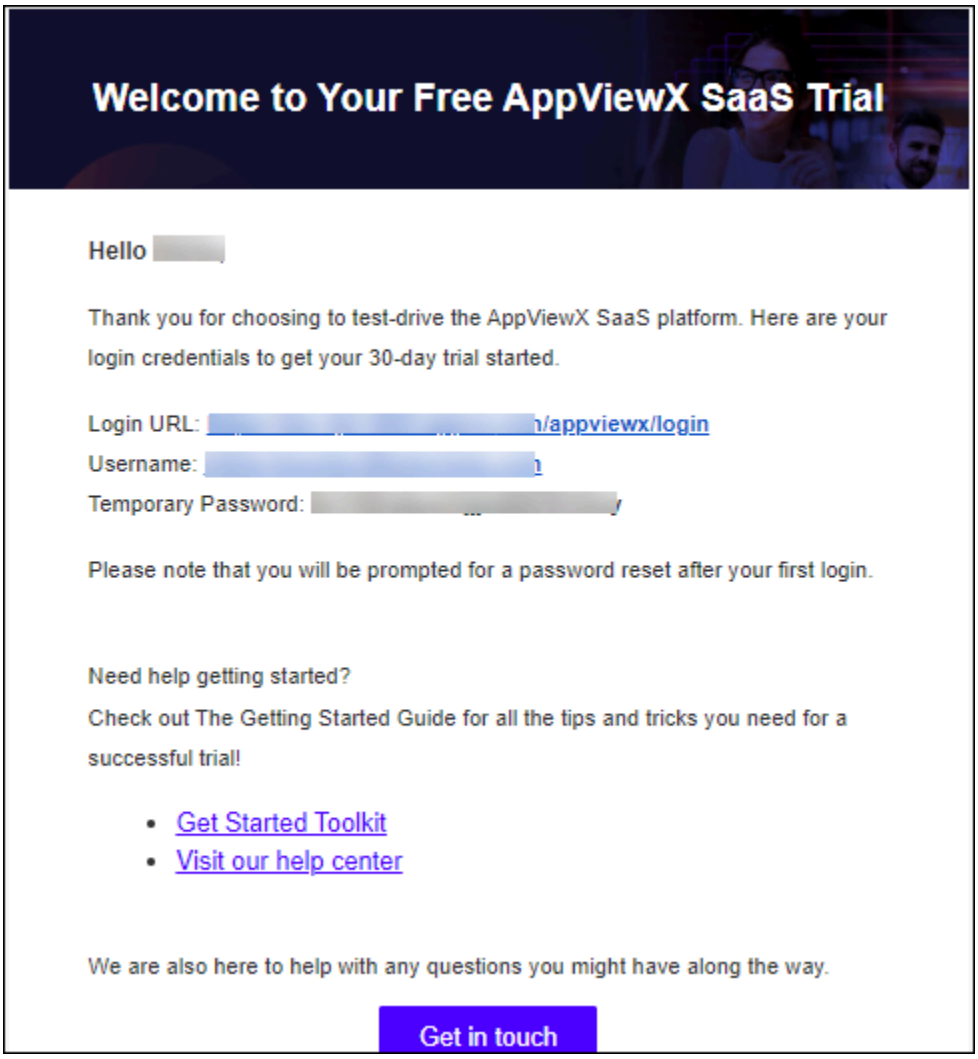
Getting Started with the AppViewX Free Trial

For users evaluating the AppViewX SaaS solution, which enables turnkey Certificate Lifecycle Management, ADC management and automation, and PKI, AppViewX enables two channels to onboard you for a free trial of the product:

- via the AppViewX website
- via the AWS Marketplace

Signing Up for the Free Trial via the AppViewX Website

1. Go to the SaaS free trial page.
2. On the landing page, enter your details and click **Get Stared**.
   A confirmation message that you will receive an email from AppViewX on your registered email ID shortly is displayed. You will receive a verification email shortly.The welcome email you receive will include your login URL and temporary password.
3. In the verification email, click **Verify your Email Address**.
   A welcome email, with your login URL and temporary credentials, is sent to your registered email ID.



Troubleshooting: If you have not received the welcome email, please get in touch with **sales@appviewx.com**.

4. Go to the login URL.
5. On the login page, enter the credentials you received in the welcome email.
   The OTP verification screen is displayed and an OTP is sent to your registered email ID.
6. Enter the OTP and click **Continue**.
   The **Change password** screen is displayed.
7. Enter and reenter your new password in the **Enter New Password** and **Confirm New Password** fields respectively, and click **Continue**.

Note: Your password must:

- Have at least one uppercase character
- Have at least one lowercase character
- Have one special character such as ~!@#$^&*_-+=|()
- Have minimum of 6 characters and maximum of 24 characters
- Not contain user name
- Not contain more than 3 same characters continuously, for example, aaa
- Not contain blank space

A message notifying successful password change is displayed.

You will be redirected to the login page.

8. Login to the application with your new credentials.
   An OTP is sent to your registered email ID.
9. On the **OTP Verification** screen, enter the OTP received and click **Continue**.
10. On the **Terms of Service** screen, select the **I accept the terms and conditions** checkbox and click **Continue**.
    The AppViewX Platform landing page is displayed.
11. To try a product, click **Try Now** to start your 30-day trial of the product.
    You will be redirected to the **GET STARTED** page of the selected product.

Note: The 30-day trial period starts from the day you receive the welcome email. The trial period can be extended by 60 more days (which makes the trial duration 90 days). For more details on how you can extend your trial, please reach out to AppViewX Support.

What to do next:

- For instructions on installing the AppViewX Cloud Connector, refer to the corresponding documentation here.
- To simplify your interaction with the product features, refer to AppViewX's exhaustive documentation in the form of the following guides:
  - AppViewX Cloud Connector User Guide
  - CERT+ User Guide
  - CERT+ Admin Guide
  - Platform User Guide
  - ADC+ User Guide
  - ADC+ Admin Guide
  - PKI+ User Guide

- SIGN+ User Guide
- SIGN+ Admin Guide
- KUBE+ User Guide

Signing Up for the Free Trial via the AWS Marketplace

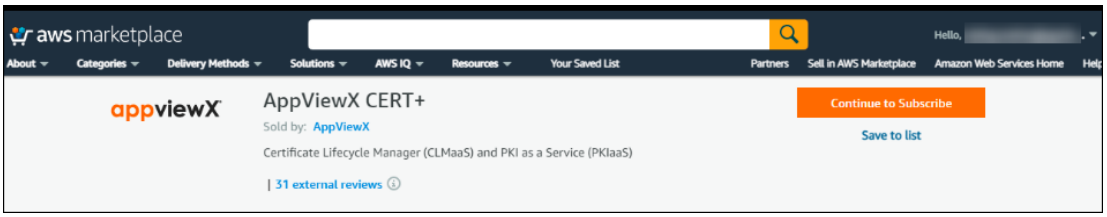
Note: Currently, you can sign up only for the CERT+ SaaS trial via the AWS Marketplace.

To get started with the CERT+ SaaS free trial, you can sign up via the AWS Marketplace and set up the SaaS by following the steps given below:

Step 1: Accessing the AWS Marketplace Sign Up Page

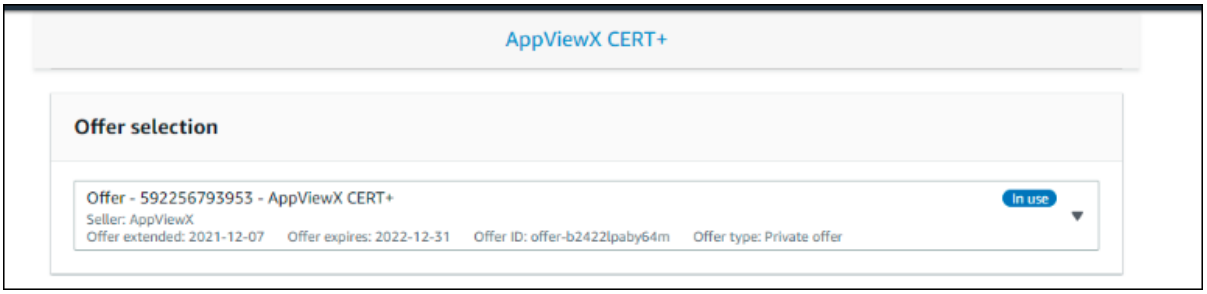
1. Navigate to the AWS Marketplace page.

   The **AppViewX CERT+** page is displayed.

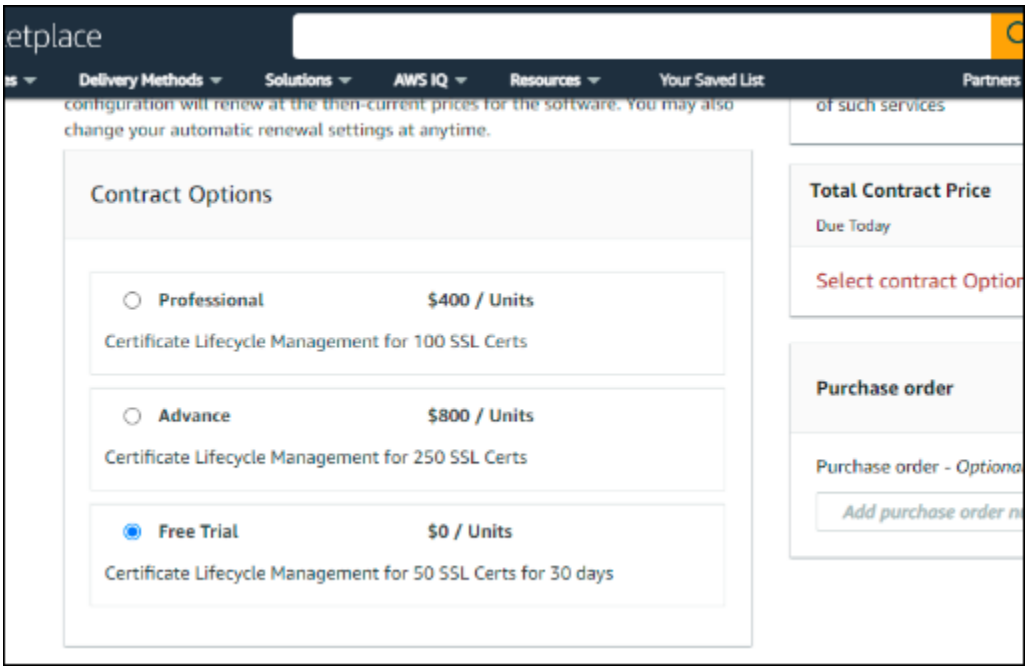


2. Sign into your account or create your account if you are new to AWS Marketplace.
3. Click **Continue to Subscribe.**

   In case you have a private offer from AppViewX, it will be listed on top of the next page. Make sure you select the private offer and not the public offers.

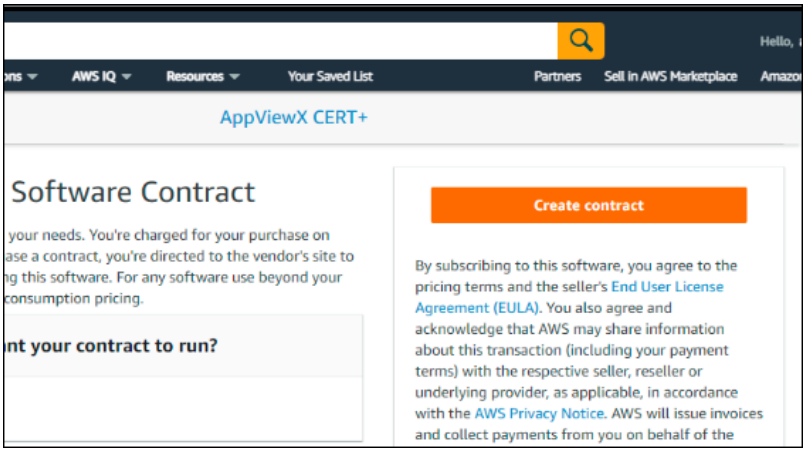


   In case you are here for the first time and do not have a private offer listed, select **Free Trial** option at bottom of the page from the **Pricing** menu.



4. Set **Auto Renew** to **No.**

Once all the aforesaid configuration is complete, the **Create contract** button is enabled.



5. Click **Create contract** to create the contract for AppViewX CERT+

A confirmation dialog box is displayed.

6. In the confirmation dialog box, click **Setup your account** to complete the signup.

   You will be redirected to the AppViewX SaaS registration page.

Step 2: Filling the Sign Up Form

1. To get started with your free trail, enter the following details:

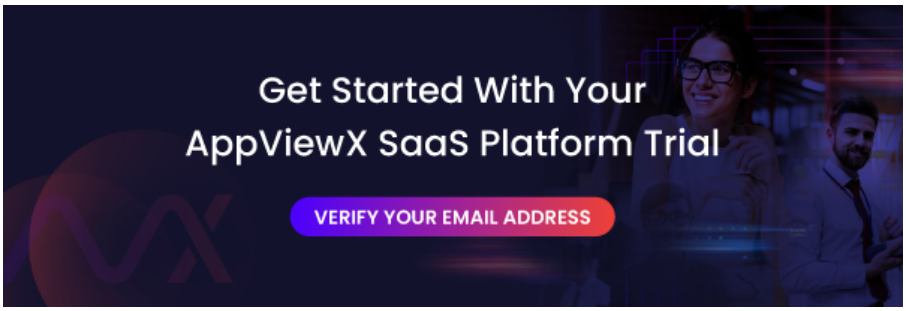
Field	Description
First Name*	Enter your first name.
Last name*	Enter your last name.
Business Email*	Enter your business email address.
Company Name*	Enter your company name.
Enter Custom Domain*	By default, the company name is auto-filled. Enter a custom domain if you want to.
Select Service Region*	The service region is where your SaaS account will be set up and localized. You cannot migrate data between regions. Select from one of the service regions: • US (Americas) • EMEA • APAC
Select Country*	Select the country from the dropdown list.

Note:
- Fields marked with the asterisk (*) symbol are mandatory.
- If you are creating a free or trial account, there are email restrictions put in place for security reasons. Email addresses from Gmail.com, Outlook.com, Yahoo.com, and other personalized email addresses are restricted and may not be used for trial account creation purposes.

2. From the **What are you trying to solve** list, select the corresponding checkboxes for your requirements.
3. To acknowledge that you have read and reviewed AppViewX's Terms of Service and their Privacy Policy, select the **By checking this box, I acknowledge...** checkbox.
4. Click **Get Started.** The message, *Thank you for signing up for the free trial! You will receive an email from us shortly,* is displayed.

Step 3: Verifying your Email

On clicking **Get Started,** you will get a verification email to your registered email address. Click **Verify Email Address** to get your SaaS account set-up.



Note:

- If you do not see the email in your inbox, then check the Junk/Spam folder. Whitelist the email address so you receive all AppViewX emails in your inbox.
- Confirm your email address within 48hours.

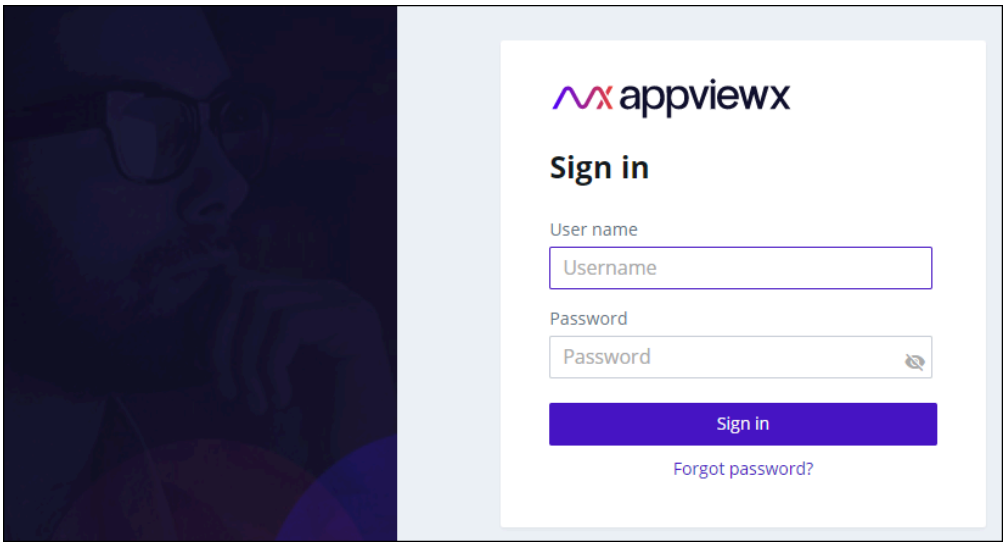
Wait for a couple of minutes until your email address is successfully verified.

Step 4: Logging in to your SaaS Account

Based on the details entered, you will receive a welcome email on your registered email ID. The email includes your login URL and temporary credentials.

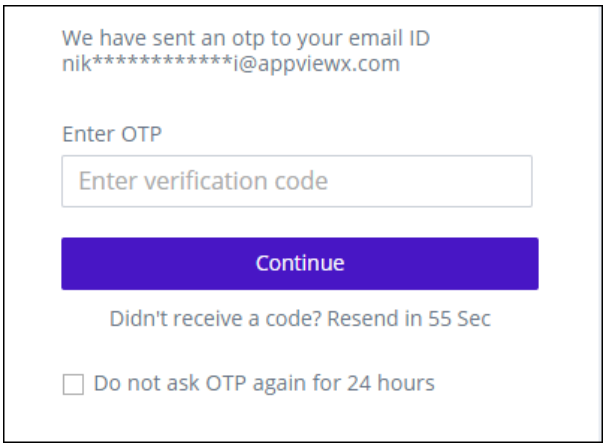
1. Navigate to the login URL.

   The login page is displayed.



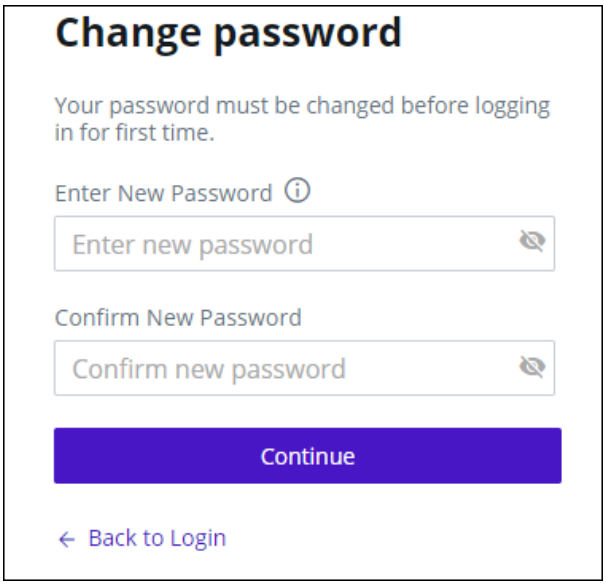
2. From the welcome email, login using the credentials provided.

   On successful login, the OTP verification screen is displayed. You will receive the OTP on your registered email address.



3. Enter the OTP received.

   On entering the correct OTP, the **Change password** screen is displayed.



4. Enter and reenter your new password in the **Enter New Password** and **Confirm New Password** fields respectively.

> **Note:** The password must:
> - Have at least one uppercase character
> - Have at least one lowercase character
> - Have one special character such as ~!@#$^&*_-+=|()
> - Have minimum of 6 characters and maximum of 24 characters
> - Not contain user name
> - Not contain more than 3 same characters continuously, for example, aaa
> - Not contain blank space

5. Click **Continue**.

   A message notifying successful password change is displayed.

   You will be redirected to the login page again.

6. Sign in with your new credentials.
7. n the **OTP Verification** screen, enter the OTP received on your registered email and click **Continue**.
8. On the **Terms of Service** screen, select the **I accept the terms and conditions** checkbox and click **Continue**.

   The AppViewX Platform landing page is displayed.

9. To try a product, click **Try Now** to start your 30-day trial of the product.

You will be redirected to the **GET STARTED** page of the selected product.

> **Note:** The 30-day trial period starts from the day you receive the welcome email. The trial period can be extended by 60 more days (which makes the trial duration 90 days). For more details on how you can extend your trial, please reach out to AppViewX Support.

Step 5: Setting up the AppViewX Cloud Connector

The AppViewX Cloud Connector can be set up in two different ways:

- Using the virtual machine

  For instructions on setting up the AppViewX Cloud Consumer using a virtual image, click here.

- Via the native OS

  For instructions on setting up the AppViewX Cloud Consumer using the native OS, click here.

To understand the difference between the two methods, click here.

Step 6: Getting Started with AppViewX SaaS

To simplify your interaction with the product's features, AppViewX offers exhaustive documentation in the form of the following guides:

- AppViewX Cloud Connector User Guide
- CERT+ User Guide
- CERT+ Admin Guide
- Platform User Guide
- ADC+ User Guide
- ADC+ Admin Guide
- PKI+ User Guide
- SIGN+ User Guide
- SIGN+ Admin Guide
- KUBE+ User Guide

You can access the complete AppViewX documentation here.

AppViewX Cloud Connector User Guide

The guide introduces you to the features of the AppViewX Cloud Connector, the component that facilitates a SaaS deployment of AppViewX's flagship products.

The guide includes steps for installing, configuring, managing, and troubleshooting your AppViewX Cloud Connector instance.

AppViewX Software as a Service

The AppViewX Security Automation and Orchestration Platform is a centralized control plane to automate tasks, orchestrate workflows and gain visibility to manage identities at scale, reduce security and compliance risk and ensure secure application availability.

AppViewX Cloud Connector is a lightweight plug-in that establishes connectivity between AppViewX Cloud and the Enterprise Network, and is the enabler for AppViewX's SaaS-based deployment. The cloud connector serves as a secure channel for communication between AppViewX SaaS and your enterprise network without requiring any complex network or infrastructure configuration.

The AppViewX SaaS platform offers the following three products:

- CERT+, which lets you:
  - Discover, monitor, analyze, orchestrate and fully automate certificate lifecycle management and key management solutions.
  - Make a shift from reactive mode and be more proactive as you get a complete view of your entire certificate infrastructure.
  - Manage certificates as a service with pre-built integrations and extensible APIs that plugin to your enterprise applications, web servers, microservices, and multi-cloud environments.
  - Analyze certificates for crypto standards like key size, cipher strength, and allowed protocol versions.
  - Setup policies for enforcing high crypto standards.
  - Update certificates as per new policies.
  - Provision certificates for devices and applications.
  - Save resources, time, and effort of installation and maintenance.

  For details, refer the CERT+ User Guide.

- ADC+, which lets you:
  - Efficiently distribute network load or client requests across servers.
  - Send requests to the available servers, ensuring high application availability.
  - Scale the number of servers (up or down) based on the traffic.

  For details, refer the ADC+ User Guide.

- PKI+, which lets you:
  - Create root CAs and subordinate CAs and enroll them to the AppViewX PKIaaS certificate authority.
  - Onboard custodians to add root CAs and subordinate CAs to the PKI+ system.
  - Manage custodians for approving PKI+-related actions.

  For details, refer the PKI+ User Guide.

- SSH+, which lets you:
  - Discover and display SSH certificates alongside SSH keys, offering a more comprehensive overview of your security credentials.
  - Download keys for key-based access control, ensuring streamlined access management.
  - Specify access duration in either hours or days when requesting access to an infrastructure group, providing enhanced access management control.
  - Use a dynamic access flow that adapts to either key or certificate-based access, depending on the user's selected 'Access Mode' during host addition.
  - Rotate host certificates effortlessly, directly from the host inventory, promoting secure host certificate management.
  - Revoke SSH certificates directly, thus enhancing security control.

- Choose between 'Key' and 'Certificate' access modes during host addition, with the 'Certificate' option being pre-selected by default.
- Rotate and delete keys from hosts with multiple keys through the user and host key age report.

For details, refer the SSH+ User Guide.

- SIGN+, which lets you:
  - Simplify Code Signing Certificate enrollment and Certificate Lifecycle Management (CLM) operations.
  - Customize signing policies according to your requirements
  - Integrate with AppViewX's customized Cryptographic Service Provider (CSP) and PKCS#11 for enhanced security.
  - Manage your code signing inventory with a full suite of tools and features.
  - Sign your code effortlessly using a variety of tools including SignTool, JSign, JarSigner, APKSigner, Mage, and Nuget.
  - Ensure compatibility with third-party Timestamp Authorities (TSA) for a wider range of options.

  For details, refer the SIGN+ User Guide.
- KUBE+, which lets you:
  - Simplify Certificate Lifecycle Management for Kubernetes workloads.
  - Get real-time visibility, central audit, and governance over K8’s Certs.
  - Achieve end-to-end automated certificate enrollment process.
  - Have secure and compliant PKI across K8s workloads (secrets, pods, and service mesh).

  For details, refer the KUBE+ User Guide.
- The AppViewX Cloud Connector

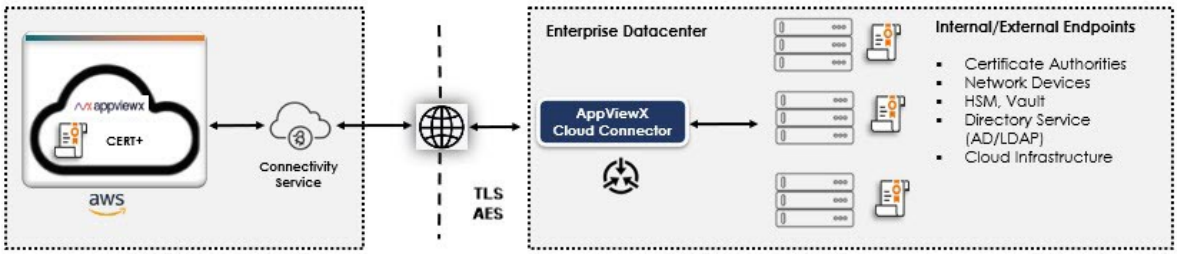
The AppViewX Cloud Connector

AppViewX Cloud Connector is a lightweight plug-in that establishes connectivity between AppViewX Cloud and the Enterprise Network. The cloud connector serves as a secure channel for communication between AppViewX SaaS and your enterprise network without requiring any complex network or infrastructure configuration.

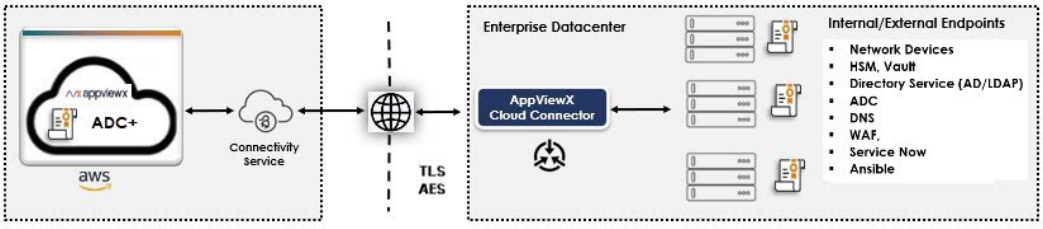
Services that require the AppViewX Cloud Connector for using the AppViewX products (examples):

- **CERT+**
  - Discovering certificates from an endpoint within the enterprise network via Smart Network Scan and Managed Device Scan.
  - Discovering certificates from Certificate Authorities (CAs) that are internal to the enterprise. For example : EJBCA.
  - Discovering certificates from public Certificate Authorities (CAs)

    In this case, AppViewX provides a default instance of the Cloud Connector called **cloud-dc**.



- **ADC+**:
  - Communicating with ADC devices and discover the Application Services from the ADC infrastructure
  - Gain Visibility and to fetch the real time state/status of the Applications discovered
  - Self Service the Applications to allow/deny traffic
  - Backup the configuration of the ADC devices
  - Restore the configuration of the ADC devices
  - Automate and Orchestrate the ADC configuration within and across devices



Key features of the AppViewX Cloud Connector:

- A self-serviceable, Linux-based lightweight setup
- Secure communication between the AppViewX SaaS and the AppViewX Cloud Connector using TLS and AES encryption

- Connectivity from the AppViewX SaaS to the enterprises' network endpoints
- No complex network setup (Inbound Firewall Whitelisting, VPN setup, and so on)
- Features of the AppViewX Cloud Connector

Features of the AppViewX Cloud Connector

- AppViewX Cloud Connector DataCenter Significance
- Cloud Connector High Availability
- Integrated Gateway
- Custom Certificates for Core Communication
- Communication Authentication and Encryption
- Auto Enrollment with the AppViewX Cloud Connector
- Enabling Proxy for End Point Communication

AppViewX Cloud Connector DataCenter Significance

AppViewX provides a **default** Cloud Connector DataCenter called **cloud-dc** within the AppViewX SaaS infrastructure for connectivity to the public endpoints such as Cloud Accounts (Cloud service providers - AWS, Azure, Google Cloud) and External Certificate Authorities (Digicert, Entrust, Commodo etc) directly thereby eliminating complex configurations.

Alternatively, users can also set up **dedicated** cloud connectors within the enterprises's cloud infrastructure should there be any source connection restrictions or a need for dedicated communication to the respective public endpoints - Cloud Accounts (Cloud service providers - AWS, Azure, Google Cloud) and External Certificate Authorities (Digicert, Entrust, Commodo, and so on). The cloud connectors can be mapped to a unique DataCenter name.

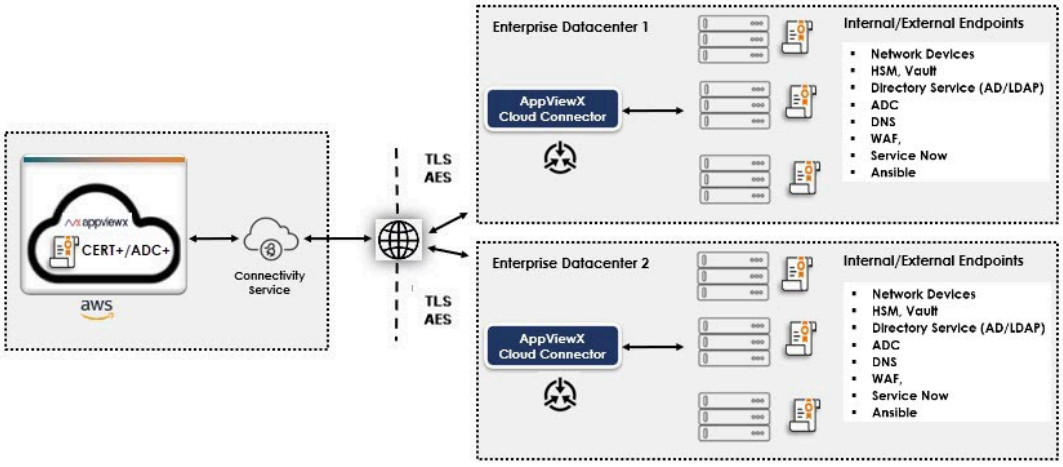
You can map the DataCenter to the AppViewX Cloud Connector instance at the time of adding the cloud connector.

You can also choose to have a dedicated cloud connector instance for specific endpoints. To do this, at the time of onboarding the endpoint (Managed Devices, Network Scan, Certificate Authority, and so on), select the AppViewX Cloud Connector DataCenter Name from the **DataCenter** field in each of the endpoint onboarding pages.

Data Center-based Routing

The AppViewX Cloud Connector instances that need to connect to the network endpoints are deployed inside a specific DataCenter in an enterprise's premises. Based on the DataCenter in which the cloud connector is added, the calls to manage the end points are routed to the specific cloud connector inside a DataCenter.

Figure 1. Typical deployment of the AppViewX Cloud Connector across multiple data centers



AppViewX supports the following two types of data center routing:

- Non strict routing (Default)
- Strict routing

Non strict routing (Default)

In this mode of routing, when a user selects a specific DC when performing an action (like discovery, device addition, cert push etc), the specific action will be routed to the AppViewX Cloud Connector in the selected DC. However, when there are no healthy AppViewX Cloud Connector instances available in the selected DC, the request will be routed to the next available healthy instance in a different DC.

This is a preferred method of deployment when you do not have a restriction in communication across your data centers.

Strict routing

When you want the requests to an endpoint in a DC to be routed only to the AppViewX Cloud Connector instance in the same DC, enable strict routing. This method ensures that when there are no healthy AppViewX Cloud Connectors in the selected DC to perform the action, the request does not get routed to any other available AppViewX Cloud Connector instance in a different DC. This method is most suitable when you are trying to manage devices within restricted DMZ zones and high latency between DCs.

Cloud Connector High Availability

To deploy AppViewX cloud connectors with high availability, it is recommended that you deploy:

- more than one cloud connector across all data centers, in the case of non-strict routing (default)
- more than one cloud connector per datacenter, in the case of strict routing

Integrated Gateway

To eliminate the need for a separate installation, AppViewX has introduced the integrated gateway functionality (tech preview). The integrated gateway is a Windows Gateway Agent that is packaged along with the AppViewX Cloud Connector.

Important:

- This is only a **tech preview**, which means the feature is under development. The tech preview has been released for testing and evaluation by the general audience. Necessary improvements will be made before the official release. Currently, this feature is **not recommended** for production.
- Ensure that you upgrade your AppViewX Cloud Connector to be able to apply this functionality.
- The integrated CC approach is recommended for customers managing fewer than 500,000 certificates in MSCA.

The integrated gateway is currently implemented for the following vendors:

- Devices
  - Microsoft IIS
  - Microsoft PC
  - Microsoft Server
  - Microsoft SQL

- Apache (Windows)
- Tomcat (Windows)

For instructions on using the internal gateway for the supported device vendors, click the vendor name from the above list.

- CAs
  - Microsoft Enterprise CA
  - Microsoft Standalone CA

For instructions on using the internal gateway for the supported Certificate Authorities, click the CA name from the above list.

Custom Certificates for Core Communication

By default, you can provision existing AppViewX self-signed certificates for the communication between the AppViewX Cloud Connector and the AppViewX SaaS. In addition to this, you can also push your custom certificates created using external CAs.

Note: This is explained in detail as part of the insrtuctions for setting up the AppViewX Cloud Connector.

Communication Authentication and Encryption

The Cloud Connector authenticates and encrypts all communications between the AppViewX SaaS and the DataCenter where the cloud connector is deployed. All connections are established from the Cloud Connector to the AppViewX SaaS using the standard HTTPS port (443) and the TCP protocol.

Auto Enrollment with the AppViewX Cloud Connector

The AppViewX Cloud Connector supports Auto Enrollment Protocols like EST, SCEP and ACME. IoT devices should be able to send auto enrollment requests to the AppViewX Cloud Connector using EST, SCEP, and ACME protocols. The cloud connector will be able to capture such information and route it to CERT+.

A gateway service will be running as a part of the AppViewX Cloud Connector to support Auto Enrollment Protocols. Ports that are exposed from this gateway are listed below:

- 30021 - port that will receive EST request
- 30022 - port that will receive SCEP request
- 30020 - port that will receive ACME request

Note: For complete documentation of the auto enrollment protocols supported in AppViewX, click here.

Enabling Proxy for End Point Communication

You can configure network proxy settings if the machine on which the Cloud Connector is deployed requires communication to external or internal endpoints (Certificate Authority, SSL / TLS endpoints).

Note: This is an optional enablement and should be performed only is a proxy setup is required. For instructions on configuring the network proxy settings, click here.

System Requirements for Setting up the AppViewX Cloud Connector

The following sections list the system requirements that are minimum prerequisites for setting up and operating the AppViewX Cloud Connector.

Note: If the host machine on which you want to set up the AppViewX Cloud Connector does not/ cannot fulfill the operating system, network, and Docker prerequisites (listed below), you can set up the AppViewX Cloud Connector via the AppViewX SaaS OVA, which is a virtual, remotely-accessible setup bundled with the OS, system, and Docker prerequisites for the AppViewX Cloud Connector.

To know more about the OVA and for instructions on setting up the AppViewX Cloud Connector using the AppViewX SaaS OVA, click here.

- Overview
- Hardware
- Operating System

Overview

The following sections list the system requirements that are minimum prerequisites for setting up and operating the AppViewX Cloud Connector.

Note: If the host machine on which you want to set up the AppViewX Cloud Connector does not/cannot fulfill the operating system, network, and Docker prerequisites (listed below), you can set up the AppViewX Cloud Connector via the AppViewX SaaS OVA, which is a virtual, remotely-accessible setup bundled with the OS, system, and Docker prerequisites for the AppViewX Cloud Connector.

To know more about the OVA and for instructions on setting up the AppViewX Cloud Connector using the AppViewX SaaS OVA, click here.

Hardware

Each AppViewX Cloud Connector instance requires the following minimum configuration:

- 4vCPU
- 8 GB memory
- 16 GB free disk space (must be available at all times)

  It is recommended to have a minimum of 32GB hard disk.
- x86 64 bit architecture

Note:

- If **/var/lib** is going to be a separate mount, ensure that it has minimum 5 GB of free space (must be available at all times).

  In case of restrictions in meeting this requirement, it is recommended to change the data root directory from **/var/lib** to another dedicated directory. For instructions on changing the data root directory, click here.
- For a RHEL8+ node, ensure that **/run** has minimum 3 GB of free space.

Operating System

- Ubuntu version 22.04
- RHEL versions 8.6, 8.7, 8.8, 9 (support for RHEL 9 validated for the version 9.2 and 9.3)

> **Note:** Versions of RHEL 9 other than 9.2 can be tested, as required. In case of any issues, please contact AppViewX Support.

- Amazon Linux 2

Server and Network Prerequisites

General Prerequisites

- Use dedicated machines for hosting the Cloud Connector and do not install any other components on these machines.
- Ensure the node on which the AppViewX Cloud Connector is installed has access to the enterprise's internal network devices.
- On the node on which the AppViewX Cloud Connector is installed, ensure that the node's clock is synchronized with the network time using NTP/PTP.

  For the **ntpd** package (for **CentOS**, **RHEL**, and **Amazon Linux 2**), execute the following sequence of commands:

```
sudo yum install -y ntp
sudo systemctl enable ntpd
sudo systemctl start ntpd
```

  For the **chronyd** package , execute the following sequence of commands:

  - For **CentOS**, **RHEL**, and **Amazon Linux 2**

```
sudo yum install -y chrony
sudo systemctl enable chronyd
sudo systemctl start chronyd
```

  OR

  - For **Ubuntu**

```
sudo dnf install -y chrony

sudo systemctl enable chronyd

sudo systemctl start chronyd
```

- Ensure that the AppViewX Cloud Connector can establish connectivity with the AppViewX SaaS server endpoints over HTTPS (port 443).

Note: In the instance a proxy being used, the proxy has to be configured as a pass-through.

Note: The Cloud Connector URL to be whitelisted for connectivity can be obtained from the Cloud Connector Settings Page of your SaaS account. Example of the AppViewX Cloud Connector URL:

```
https://<example-tenant>-cc.appvx.com:443/
```

Also, ensure that the cloud connector URL is not blocked by your puppet scripts, anti-virus software settings, and/or firewall rules.

Tip: : To verify connectivity with the AppViewX SaaS servers, use the **cURL** utility.

1. Install the **cURL** utility.
   - On **Ubuntu**: `sudo apt-get install curl`
   - On **CentOS**, **RHEL**, and **Amazon Linux 2**: `sudo yum install curl`
2. To check connectivity, execute the following command:
   - If proxy is not enabled:

```
curl -k --max-time 20 --connect-timeout 20 -s -o /dev/null -w "%{http_code}" "<<https://AppViewX SaaS server URL>>/socket.io/?EIO=3&transport=polling&t=O11wka_"
```

   - If proxy is enabled:

```
curl -k --proxy "<<http://proxyhost:proxyport>>" --max-time 20 --connect-timeout 20 -s -o /dev/null -w "%{http_code}" "<<https://AppViewX SaaS server URL>>/socket.io/?EIO=3&transport=polling&t=O11wka_"
```

   If connectivity has been established successfully, the command will return the HTTP code **200**. If the command returns any other code, it indicates that connectivity is not established.

- Disable the firewalld in the tenant's node (**Ubuntu**) where the AppViewX Cloud Connector is to be installed.

  To check the current status of firewalld, execute the command given below:

```
sudo ufw status
```

To permanently disable firewalld, execute the command given below:

```
sudo ufw disable
```

- Disable the firewalld in the tenant's node (**CentOS**, **RHEL**, and **Amazon Linux 2**) where the AppViewX Cloud Connector is to be installed.

  To check the current status of firewalld, execute the command given below:

  ```
  sudo systemctl status firewalld --now
  ```

  To permanently disable the firewalld, execute the command given below:

  ```
  sudo systemctl disable firewalld --now
  ```

  To restrict other devices from enabling the firewalld, execute the command given below:

  ```
  sudo systemctl mask firewalld --now
  ```

- Disable the **nftables** service.
  - For **Ubuntu**

    ```
    sudo apt purge -y nftables
    ```

  - Fo**CentOS**, **RHEL**, and **Amazon Linux 2**

    ```
    sudo yum remove -y nftables
    ```

- If you are utilizing the IP ranges **10.42.0.0/16** and **10.43.0.0/16**, modify them before installing the cloud connector.

  (These are the default CIDR ranges for the AppViewX Cloud Connector and should be modified to prevent IP conflicts).

  To adjust the IP range:

  1. To modify **install.sh**, execute the command: `vi install.sh`.
  2. **For a k3d installation**:
     a. In the **install.sh** file, search for the text **k3d cluster create**.
     b. At the end of this line, paste the following: **--k3s-arg '--cluster-cidr=<preferredcidr>/16' --k3s-arg '--service-cidr=<preferredcidr>/16'**.

        Here, replace **<preferredcidr>** with your preferred CIDR address.

     OR

For a standard k3s installation:

a. In the **install.sh** file, search for the text **--disable=metrics-server**.

b. Add a space and paste the following text: **--cluster-cidr=<preferredcidr>/16 --service-cidr=<preferredcidr>/16**.

Here, replace **<preferredcidr>** with your preferred CIDR address.

3. Save the file.
4. Execute the following command: `./install.sh`

Additional Prerequisites for Installation on RHEL8+

- Ensure that the user has access to perform any action (create, start, stop, and so on) on the **systemd** services through **systemctl**.
- The AppViewX Cloud Connector is installed on top of a Kubernetes engine. To install the underlying Kubernetes engine directly on the host, the user must have **sudo** access with **read/write/execute** permissions for the following directories at the least:
  - **/var/lib**
  - **/etc**
  - **/run**
  - **/usr/local/bin**
  - **/tmp**
- If **nm-cloud-setup** is enabled, disable it and reboot the node.

```
systemctl disable nm-cloud-setup.service nm-cloud-setup.timer

reboot
```

Note: Since RHEL8+ does not include Docker support, these additional prerequisites are necessary for a Docker-less installation of the AppViewX Cloud Connector.

Additional Prerequisites for Installation on Amazon Linux 2

- Deploy an EC2 instance with type C5.Xlarge 4vcpu and 8GB RAM.

Docker Prerequisites

Note: Since RHEL8+ does not include Docker support, Docker prerequisites are not applicable when the AppViewX Cloud Connector is being installed on a RHEL8+ node.

- Docker version 20.10.5 or above installed with non-sudo access with basic read and write permissions

Note: Support for rootless Docker is excluded.

For Docker installation instructions, refer to the links below:

- For installing the Docker Engine: https://docs.docker.com/engine/install/
- For post-installation steps for Linux: https://docs.docker.com/engine/install/linux-postinstall/

Important: In the event of a VM reboot, the Docker needs to be restarted. To configure the Docker to restart on boot, follow the instructions given here.

Note: If **/var/lib** is going to be a separate mount, ensure that it has minimum 5 GB of free space.

In case of restrictions in meeting this requirement, it is recommended to change the data root directory from **/var/lib** to another dedicated directory. For instructions on changing the data root directory, click here.

- Bash shell support in the node for the installation of the AppViewX Cloud Connector Connectivity Service
- Changing the Data Root Directory

Changing the Data Root Directory

Note: The following are one-time setup steps for Docker. Ensure that you execute these steps:

- as the **root** user
- are performed in each AppViewX Cloud Connector node to change the Docker data root

To change the data root directory (if **/var/lib** has insufficient space to hold the images):

1. View the current root directory using the following command: `docker info -f '{{ .DockerRootDir}}'`
   The name of the current root directory is displayed. For example: **/var/lib/docker**
2. Stop the processes that currently running using the following command: `systemctl stop docker`
3. Verify Docker status using the following command: `systemctl status docker`
   Since Docker processes were stopped in step 2, the status will be displayed as **Active: inactive (dead)**
4. Create the Docker directory, which will be your new data root directory, using the following command: `mkdir /new/path/docker-data-root`
   For example, you can choose **/home/appviewx/docker-data-root** as the new path for the data root directory.
5. Copy the content from **/var/lib** to the new data root directory using the following command:`rsync -avxP /var/lib/docker/ /new/path/docker-data-root`
6. To update the path of the Docker daemon file, create or edit the **/etc/docker/daemon.json** configuration file to add the following:

```
{
    "data-root": "/new/path/docker-data-root"
}
```

7. Restart Docker services using the following commands:

```
systemctl daemon-reload
systemctl start docker
```

8. Verify if the new data root directory has been set as the root directory using the following command: `docker info -f '{{ .DockerRootDir}}'`
   The output should display the new root directory (for example, **/new/path/docker-data-root**).

Setting Up the AppViewX Cloud Connector

The AppViewX Cloud Connector can be set up on two platforms (a virtual machine and a native operating system) and in two ways (using an automated installation and using the AppViewX GUI).

- **Via a Virtual Image**

  The AppViewX Virtual Image is an Open Virtual Appliance (OVA) that is bundled with the software, network, and Docker prerequisites for installing the AppViewX Cloud Connector without altering the OS configuration on their systems.

  When setting up the cloud connector via a virtual image, you will be required to download only the license file.

On a machine with a virtual image, you can install the cloud connector:

- using an automated script during and after the OVA deployment
- using the AppViewX GUI

- **Via the Native OS**
  - **With Docker runtime**

    Tenant/Administrator/User to provision a Linux machine with docker installed fulfilling prerequisites across the following categories: hardware, operating system, Docker, and server and network. If all prerequisites are met, you can install the AppViewX Cloud Connector via the Native OS.

    When setting up the cloud connector via the native OS, you will be required to download a package that contains the cloud connector installer and the license file.

  - **RHEL 8+ without Docker runtime**

    Tenant/Administrator/User to provision a RHEL8+ machine fulfilling prerequisites (generic as well as those exclusive for RHEL 8+) across the following categories: hardware, operating system, Docker, and server and network.

On a native operating system also, you can install the cloud connector:

- using an automated script
- using the AppViewX GUI

Methods to Set up the AppViewX Cloud Connector

The AppViewX Cloud Connector can be set up in two ways:

- **Via a Virtual Image**

  The AppViewX Virtual Image is an Open Virtual Appliance (OVA) that is bundled with the software, network, and Docker prerequisites for installing the AppViewX Cloud Connector without altering the OS configuration on their systems.

  When setting up the cloud connector via a virtual image, you will be required to download only the license file.

- **Via the Native OS**
  - **With Docker runtime**

    Tenant/Administrator/User to provision a Linux machine with docker installed fulfilling prerequisites across the following categories: hardware, operating system, Docker, and server and network. If all prerequisites are met, you can install the AppViewX Cloud Connector via the Native OS.

    When setting up the cloud connector via the native OS, you will be required to download a package that contains the cloud connector installer and the license file.

  - **RHEL 8+ without Docker runtime**

    Tenant/Administrator/User to provision a RHEL8+ machine fulfilling prerequisites (generic as well as those exclusive for RHEL 8+) across the following categories: hardware, operating system, Docker, and server and network.

Setting up the AppViewX Cloud Connector via a Virtual Image

> **Note:** The steps outlined in the following subsections are **specifically** for creating a virtual machine for OVA deployment for setting up the AppViewX Cloud Connector. For instructions on OVA deployment for setting up the AppViewX product, click here.

The AppViewX Virtual Image is an Open Virtual Appliance (OVA) that is bundled with the software, network, and Docker prerequisites for installing the AppViewX Cloud Connector without altering the OS configuration on their systems.

> **Note:** The AppViewX SaaS OVA is CIS benchmarked.

The AppViewX SaaS OVA offers the following advantages:

- Built with Ubuntu version 22.04
- Docker 20.10.5 pre installed with all required permissions
- Hardened OVA with all security issues addressed

> **Note:** Detailed instructions for updating the AppViewX virtual image from the AppViewX repository are documented here.

Note: If this AppViewX Cloud Connector installation requires configuing a proxy server, click here for instructions.

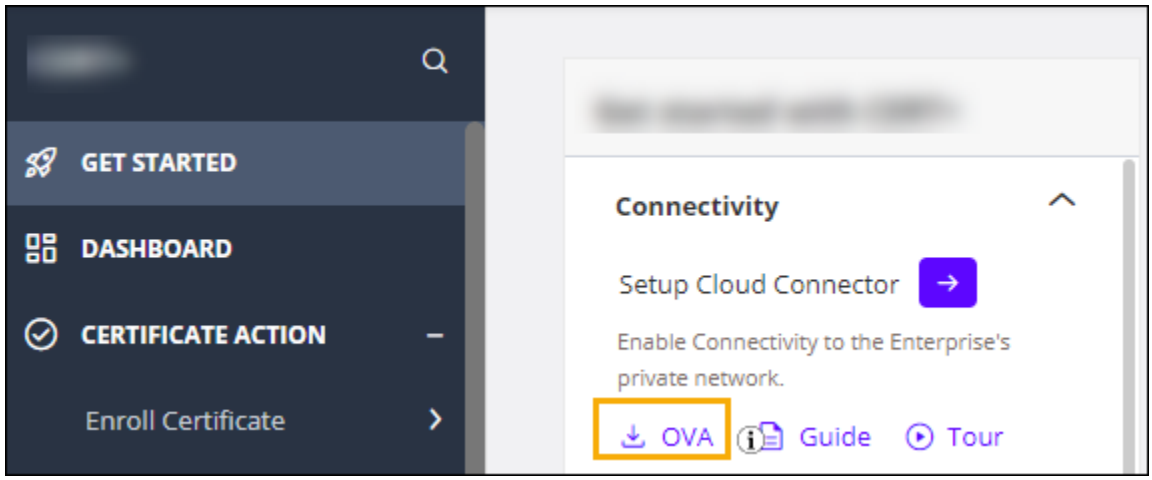
Setting up the AppViewX Cloud Connector via a Virtual Image using the Automated Script

The following sections will guide you through the steps for deploying the AppViewX virtual machine and installing the AppViewX Cloud Connector for

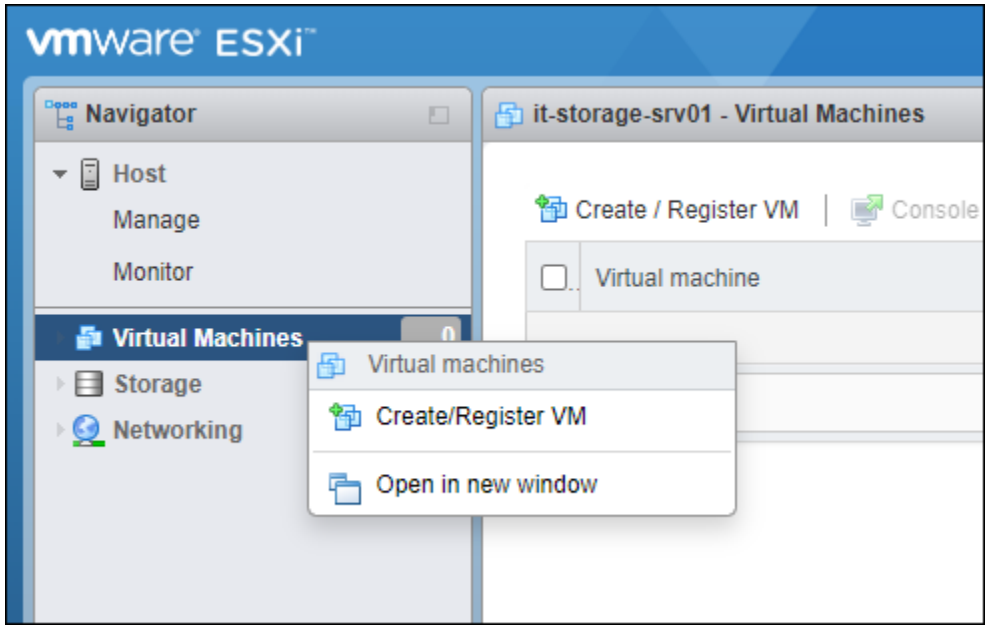
- an on-prem deployment
- AWS
- Azure
- GCP

Setting up the AppViewX Cloud Connector using a Virtual Image for an On-Prem Deployment

1. To download the release package in the OVA format, from the respective AppViewX's product line landing page, under **GET STARTED** menu > **Connectivity** section, click OVA.

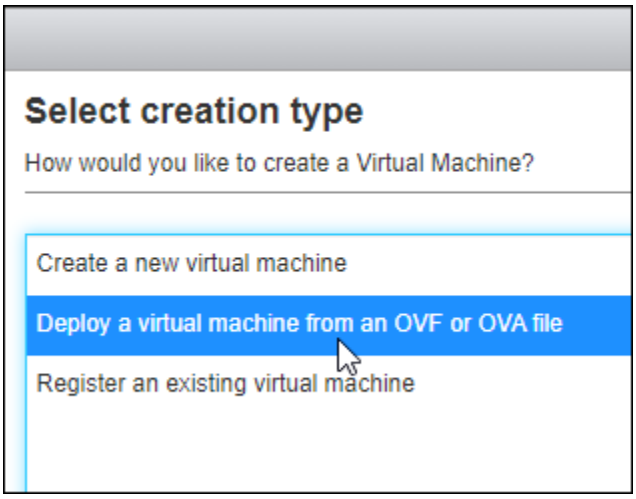


2. Log in to the **vmware** client.
3. From the **Navigation** pane on the left, right click **Virtual Machines**.
4. Click **Create/Register VM**.

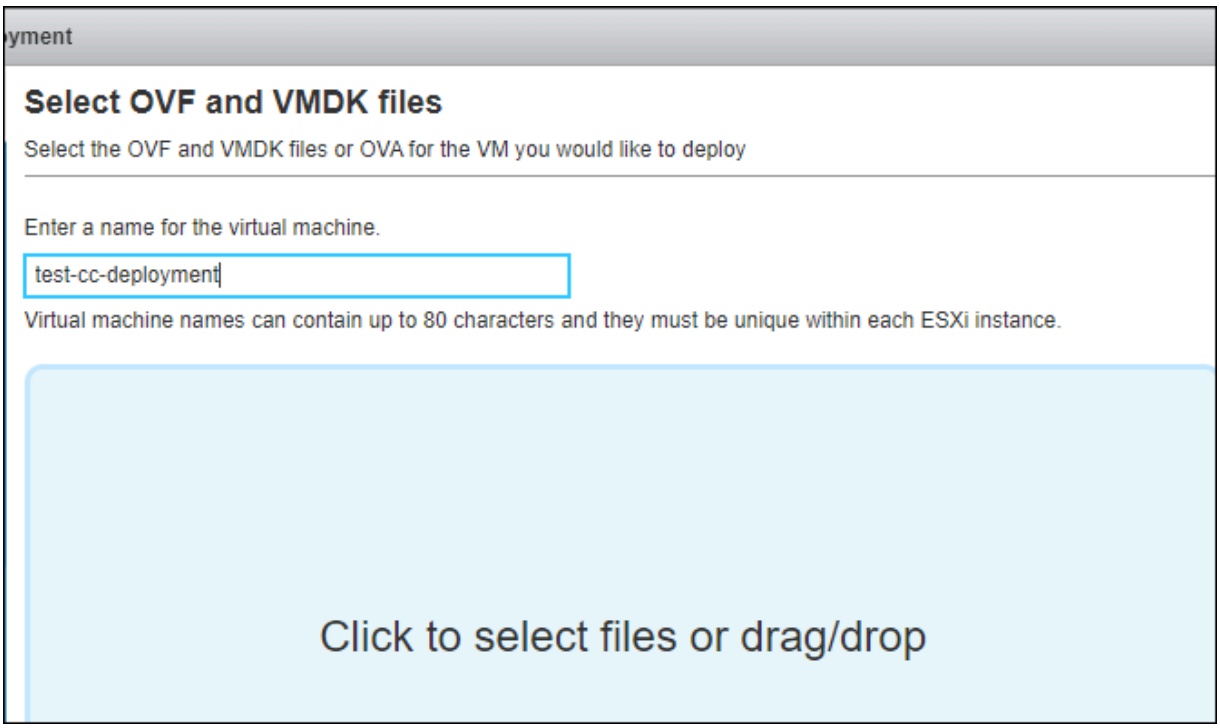


The **New Virtual machine** window is displayed.

5. From the navigation pane in the left, select **Select creation type**.
6. In the **Select creation type** window, select the **Deploy a virtual machine from an OVA or OVF file** option.



7. Click **Next**.
8. In the **Select OVF and VMDK files** window:

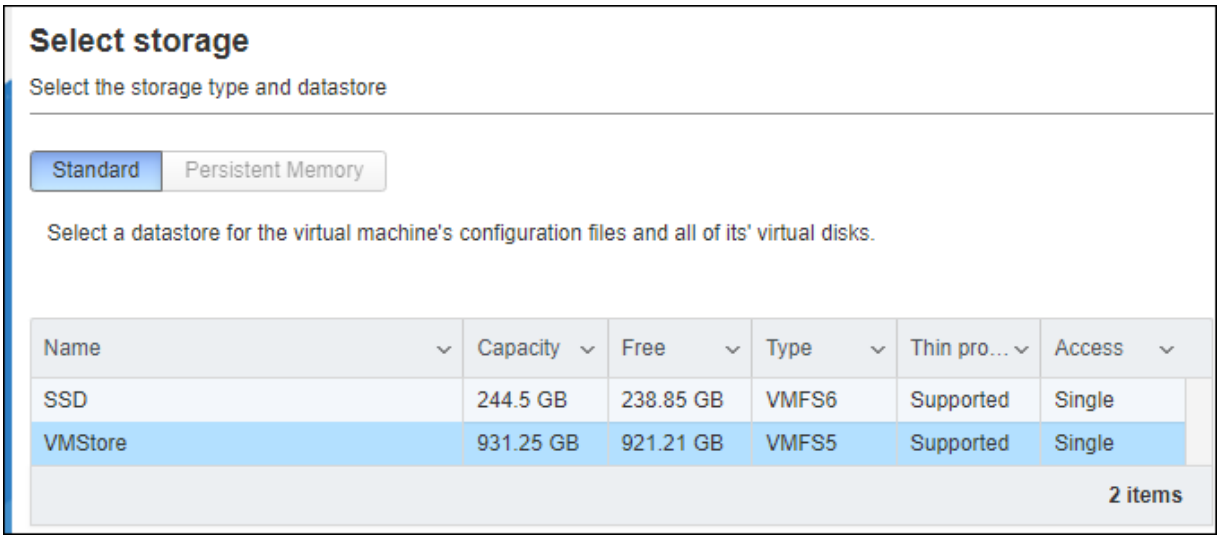


   a. Enter a name for the virtual machine.
      For the purpose of this document, we will name it **test-cc-deployment**.

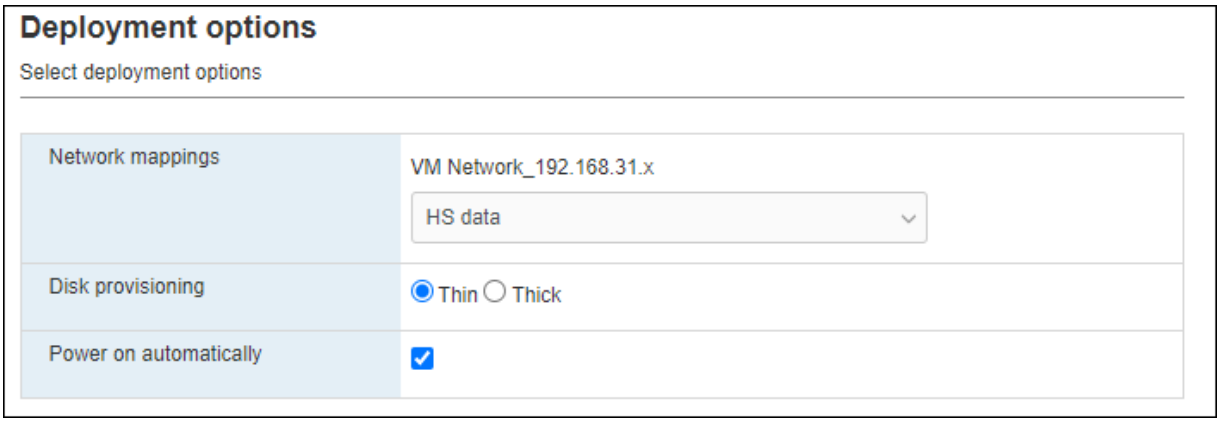
   b. In the **Click to select files or drag/drop** area, click and, from the file explorer, navigate to the location of the file, select the file, and click **Open**.

9. Click **Next**.

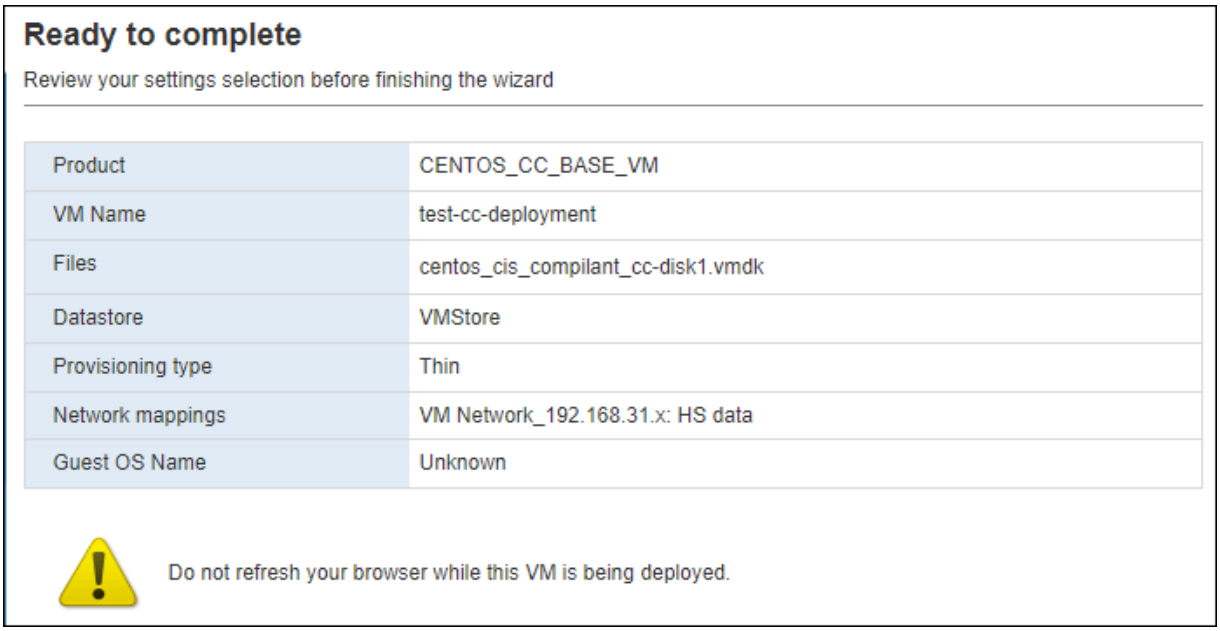
10. In the **Select storage** window, from the available options, select a datastore for storing the virtual machine's files and all of its virtual disks.



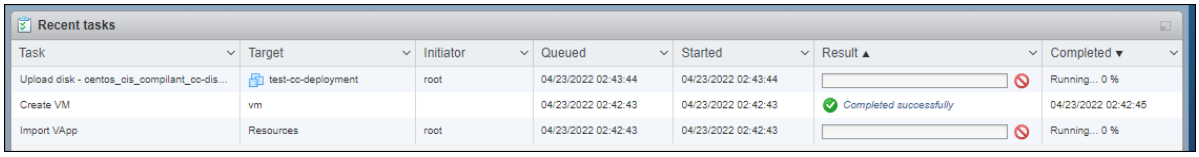
11. Click **Next**.
12. In the **Deployment options** window:

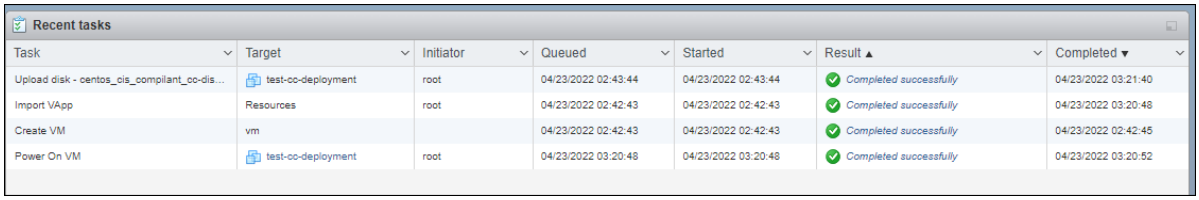


    a. Select the network mapping.
    b. Select the disk provisioning required.
    c. Select the **Power on automatically** checkbox.
13. Click **Next**.
14. In the **Ready to complete** window, review your settings.

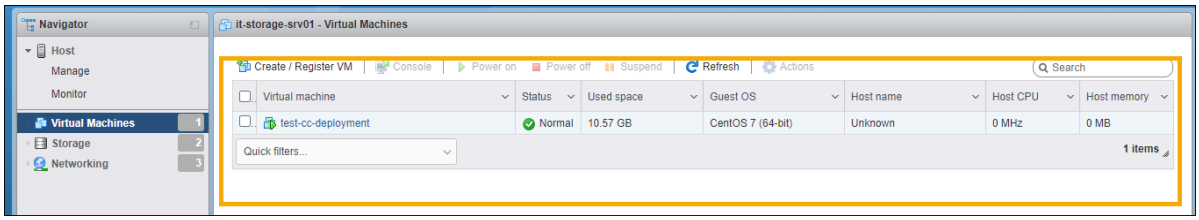


15. Click **Finish**.
    - The progress of the OVA deployment is shown in the **Recent Tasks** section.





    - On successful completion of the OVA deployment, the new virtual machine is displayed in the **Virtual Machines** inventory. For each virtual machine in the inventory, the following details are displayed:

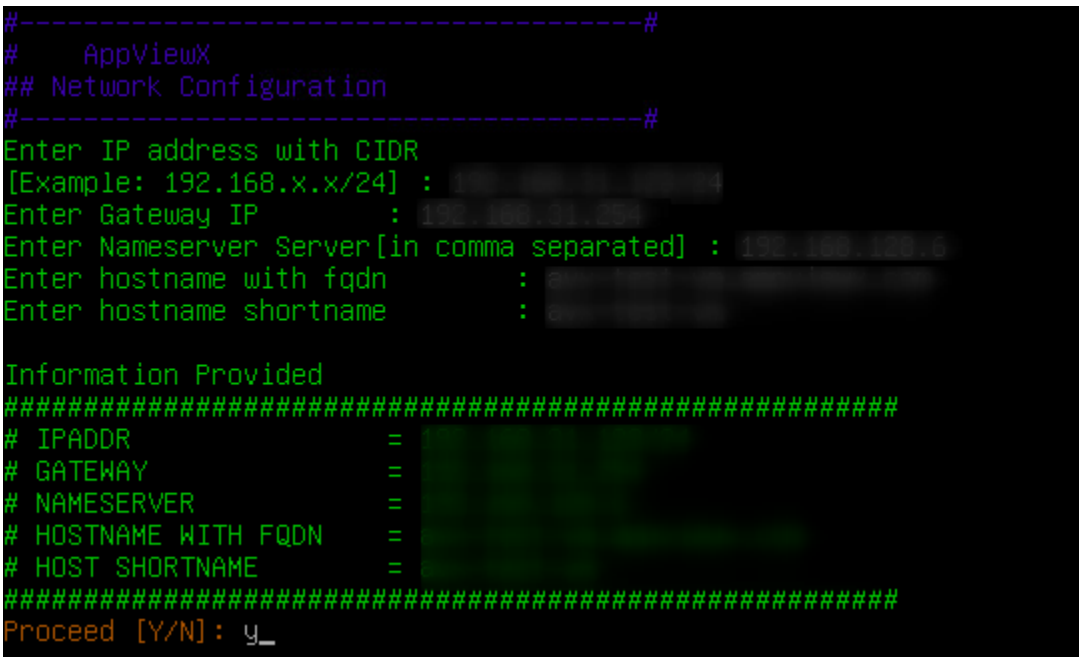


16. From the **Virtual Machines** inventory, click the virtual machine just added.
    The terminal window for the virtual machine is displayed. The script for configuring the network IP is executed automatically.

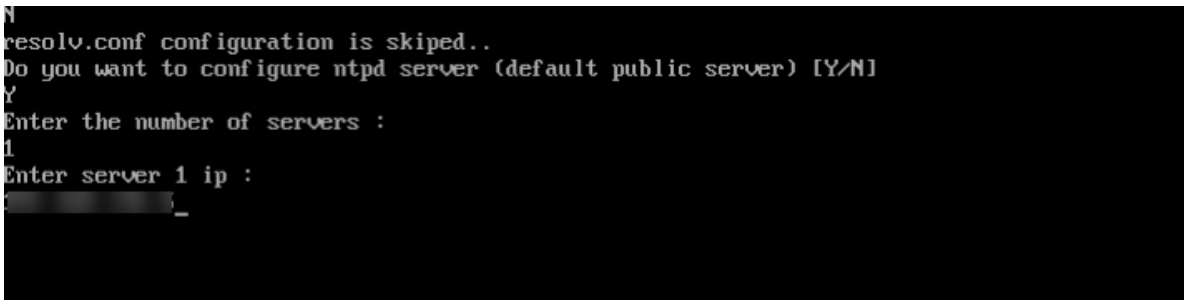
17. To configure the IP address, when prompted, enter the required values for the following requested parameters:

```
==========================================

IPADDR          = XXX.XXX.XXX.XXX

GATEWAY         = XXX.XXX.XXX.XXX

NAMESERVER      = XXX.XXX.XXX.XXX

HOSTNAME WITH FQDN  = server1.example.com

HOST SHORTNAME      = server1

==============================
```

For example, refer to the sample screenshot below:



18. To configure the hostname and the DNS, when prompted, press **Y**. If you prefer to configure the hostname and DNS manually, to skip this step, press **N**.
19. To configure an NTP server(s):
    a. When prompted **Do you want to configure ntpd server (default public server)**, enter **Y**.



b. Enter the number of NTP servers to be configured.
c. For the number of servers entered above, enter the IP address of each NTP server on a new line.
d. To update the **ntp.conf** file with the IP addresses provided above, press **Y**.

After the execution of this step, if the cloud connector does not exist on this host machine, the following prompt is displayed: **Would you like to opt for automated installation of the cloud connector? (y/n):**.

> **Note:** Proceeding with the cloud connector installation at this point will require you to enter the tenant ID and master key for the installation **manually** (you will not be able to copy the details and paste them in response to the prompt).

20. To install the AppViewX cloud connector at this point, press **y** and then press **Enter**.

> **Note:** If you enter **n** here (that is, you choose to **not** go ahead with the installation here), **skip steps 21 to 25**. You will be reprompted for the automated installation after you login to the VM as the appviewx user.

21. When prompted **Please enter your AppViewX Cloud tenant name or ID**, enter the required details.
    Internet connectivity on the host machine is validated. The installation proceeds only if the host machine has internet access.

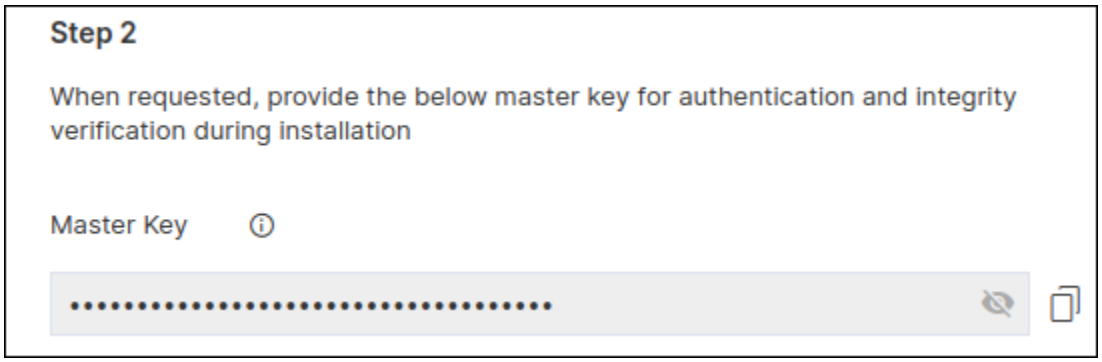
> **Note:** If internet access is unavailable, you will be prompted to proceed with installation using a proxy. Enter **y** and for instructions on proceeding with the installation with a proxy, click here.

22. When prompted to **Please provide the master key**, enter the required details.
    To retrieve the master key:

    a. Go to (**Menu**) > **Platform** > **Connectivity** > **Cloud Connector**.
    The **Settings :: Cloud Connector** inventory page is displayed.

    b. From the cloud connector details banner, under **Automate Cloud Connector Setup**, click **Steps to Automate Setup**.

    c. From the **Automated Cloud Connector Setup** window, under **Step 2**, copy the **Master Key**.



d. Paste the master key in the terminal window and press **Enter**.

23. When prompted to **Please enter Datacenter**, enter the name of the data center on which this cloud connector will be deployed.
    The cloud connector installation script will check for the prerequisites and trigger the cloud connector installation.

    When the cloud connector instance is successfully installed, a corresponding entry will be listed in the cloud connector inventory.

24. **Note:** Enabling auto-enrollment protocols is recommended only if the cloud connector is being installed in a demilitarized zone (DMZ) or devices in a restricted environment (that disables them from connecting to the <tenant>-aep directly.

    When prompted to enable auto-enrollment protocols, enter **y** and enter the protocol name(s) you want to enable. For instructions on enabling auto-enrollment protocols, click here.

    Note: By default, only the AppViewX certificate is enabled for auto-enrollment. To enable custom certificate for auto-enrollment:
    a. Execute the command `./avxctl upgrade gateway-cert`.
    b. When prompted, enter the location of the custom certificate.

    On successful completion of the setup, a corresponding instance of this cloud connector is displayed in the inventory.

    Note: The cloud connector installation on a OVA-based host machine will not prompt you to select if you want to manage F5 Big-IP devices. However, after the cloud connector has been

installed you can copy the **iControl jar** and **axis jar** in the deps/external_libs folder and restart the starter and platform pods (click here for instructions), to enable this feature.

25. After the script is executed, when prompted, login to the VM as the appviewx user, using the credentials shared by AppViewX's customer support team.

    Note: Root user access is required for maintaining the OS configuration and for patching security updates. Since direct root access is not provided, you can:
    a. Login as the **appviewx** user.
    b. Switch to the root user by executing the command `sudo -i`.

    Note: It is recommended to change the default credentials after the first login.

    If the cloud connector has not been installed already, you will be reprompted to proceed with the installation.
26. When prompted **Would you like to opt for automated installation of the cloud connector? (y/n):**, enter **y**.
    If you enter **n** here (that is, you choose to not go ahead with the installation here), **skip step 27**. You will be prompted again for the automated installation every time you login again to the VM as the appviewx user (until the cloud connector is installed).
27. Repeat steps **21** to **24** above.
28. When prompted **Are you sure you want to continue connecting**, press **y** and then press **Enter**.
29. To check if the Docker is up and running, execute the command: `systemctl status docker`.
    If the Docker status is **active (running)**, as shown in the screenshot below, it means that the OVA has been deployed successfully.

```
[appviewx@ccnode ~]$ systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
   Active: active (running) since Tue 2022-06-14 06:08:50 EDT; 6 days ago
     Docs: https://docs.docker.com
 Main PID: 1540 (dockerd)
   CGroup: /system.slice/docker.service
           └─1540 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

    Note: To check if the Docker is accessible to the appviewx user, execute the following command:

    ```
    docker image ls
    ```

If the command does not return an error, it means that the Docker is accessible to the appviewx user:

30. Login to the host machine on which the OVA has been deployed.
31. If the Cloud Connector has been installed, go to **step 39** to approve the cloud connector installation.
    If the AppViewX Cloud Connector has not been installed on the host machine till this point, the following prompt will be displayed again: **Would you like to opt for automated installation of the cloud connector? (y/n):**
32. Enter **y**.
33. When prompted **Please enter your AppViewX Cloud tenant name or ID**, enter the required details.
    Internet connectivity on the host machine is validated. The installation proceeds only if the host machine has internet access.

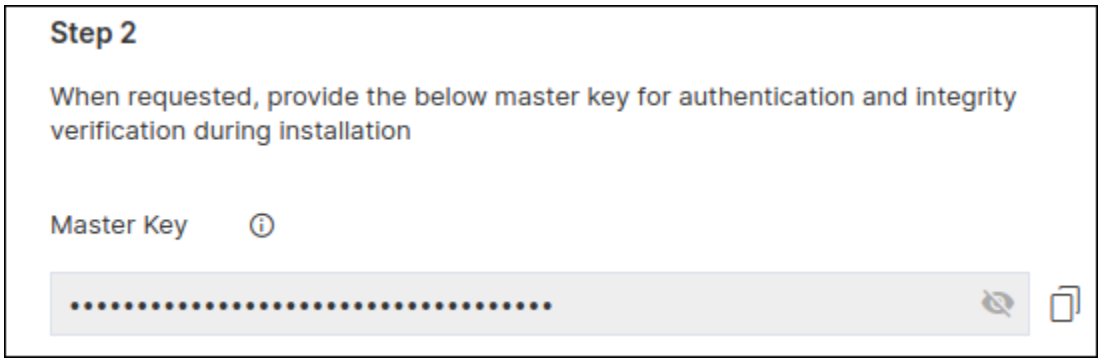
Note: If internet access is unavailable, you will be prompted to proceed with installation using a proxy. Enter **y** and for instructions on proceeding with the installation with a proxy, click here.

34. When prompted to **Please provide the master key**, enter the required details.
    To retrieve the master key:

    a. Go to (**Menu**) > **Platform** > **Connectivity** > **Cloud Connector**.
    The **Settings :: Cloud Connector** inventory page is displayed.

    b. From the cloud connector details banner, under **Automate Cloud Connector Setup**, click **Steps to Automate Setup**.

    c. From the **Automated Cloud Connector Setup** window, under **Step 2**, copy the **Master Key**.



    d. Paste the master key in the terminal window and press **Enter**.

The cloud connector installation script will check for the prerequisites and trigger the cloud connector installation.

When the cloud connector instance is successfully installed, a corresponding entry will be listed in the cloud connector inventory.

35. When prompted, **Please enter Datacenter**, enter the name of the data center on which this cloud connector will be deployed.

36. **Note:** Enabling auto-enrollment protocols is recommended only if the cloud connector is being installed in a demilitarized zone (DMZ) or devices in a restricted environment (that disables them from connecting to the <tenant>-aep directly.

    When prompted to enable auto-enrollment protocols, enter **y** and enter the protocol name(s) you want to enable. For instructions on enabling auto-enrollment protocols, click here.

    Note: By default, only the AppViewX certificate is enabled for auto-enrollment. To enable custom certificate for auto-enrollment:
    a. Execute the command `./avxctl upgrade gateway-cert`.
    b. When prompted, enter the location of the custom certificate.

    On successful completion of the setup, a corresponding instance of this cloud connector is displayed in the inventory.

    Note: The cloud connector installation on a OVA-based host machine will not prompt you to select if you want to manage F5 Big-IP devices. However, after the cloud connector has been installed you can copy the **iControl jar** and **axis jar** in the deps/external_libs folder and restart the starter and platform pods (click here for instructions), to enable this feature.

37. To approve the cloud connector installation:

a. Go to (**Menu**) > **Platform** > **Connectivity** > **Cloud Connector**.
The **Settings :: Cloud Connector** inventory page is displayed.

b. For the cloud connector instance installed, from the **Actions** column, click **Approve**.

Troubleshooting: If the AppViewX Cloud Connector instance has been approved but is not moved to the **Running** state, you can check the pod status and/or restart the pod(s), as required.

Setting up the AppViewX Cloud Connector using a Virtual Image on AWS

Note: For the AWS AMI, the following two operating systems are supported: **Ubuntu** and **Amazon Linux 2**.

Prerequisites

- Relay your requirements to your assigned AppViewX Solution Architect and finalize a deployment model.
- Share your AWS account number and region with your Solution Architect. The Solution Architect will use these details to create a custom AMI based on your account and region.

  When the AppViewX AMI is successfully shared with your customer account, AppViewX will notify you of this development via email.
- From your AppViewX Onboarding Engineer, get the default password for the **appviewx** user.

To install the AppViewX Cloud Connector on AWS, you will need a virtual machine that is preconfigured for the operating system and software stack prerequisites.AWS uses AMI to create pre-configured EC2 instances as per AppViewX standards and requirements.

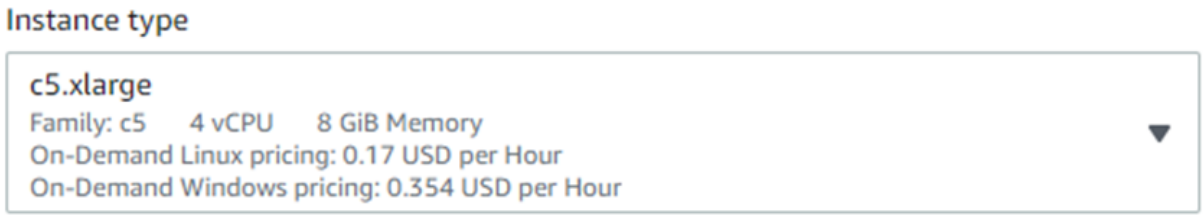
To create an EC2 instance using the AppViewX AMI:

1. Login to the AWS Management Console and go to **EC2** > **Images** > **AMIs**.
   The **Amazon Machine Images (AMIs)** page is displayed.
2. On the **Amazon Machine Images (AMIs)** page, from the **Owned by me** dropdown list, select **Private images**.
   All AMIs with visibility set to private are listed. This list will also have the AMI that is created and shared by AppViewX for your requirements.

3. From this list, select the checkbox for the AMI shared by AppViewX.
4. Click **Launch instance from AMI**.
   The **EC2 > Instances > Launch an instance** page is displayed.
5. Enter the **Name and tags** to be associated with this EC2 instance.

> **Note:** The **Application and OS Images (Amazon Machine Images)** section will show the configuration details of the AppViewX AMI.

6. For the master node, select the following hardware configuration:



7. To securely connect to the EC2 instance, in the **Key pair (login)** section:
   a. To use an existing key pair, from the **Key pair name** dropdown list, select the key pair you want to use.

   OR

   a. To create a new key pair, Click **Create new key pair**.
8. In the **Network settings** section, under **Firewall (security groups)**, as required, create a new security group or select an existing security group.
9. If you select **Select existing security group** in the previous step, from the **Common security groups** dropdown list, select the required security group.
10. From the bottom-right corner of the screen, click **Launch instance**.
    The **Launching instance** page is displayed, which shows you the progress of the launch. As soon as the launch is initiated, you will get a success message.
11. Under **Success**, click **Launch log** to review the instance details.
12. From the page name (**EC2 > Instances > Launch an instance**), click **Instances** to go back to the previous page.
13. From the list of instances, select the AWS instance just created.
14. To login to this AWS instance using the key pair .pem file:

    a. Execute one of the following commands:
       • For Ubuntu:

```
ssh -i newkey.pem ubuntu@<public ipaddress of the aws instance>
```

- For Amazon Linux 2:

```
ssh -i newkey.pem ec2-user<public ipaddress of the aws instance>
```

b. To switch to the **sudo** user, execute the following command: `sudo -i`
The following prompt is displayed: **Would you like to opt for automated installation of the cloud connector? (y/n):**

c. Enter **n**.
This question is asked here by default. However, since the nameserver has not been configured yet, it is **not recommended** to proceed with the installation because the connection with the cloud connector and mothership will not be established.

d. To add an entry for the nameserver in the **resolv.conf** file, execute the following command: `echo "nameserver <IP of nameserver>" > /etc/resolv.conf`

e. Update the /etc/hosts file for the IP and the hostname of the VM created, using the following commands:

```
vi /etc/hosts


hostnamectl set-hostname “hostname-of-the-vm”
```

f. To validate the update to the **/etc/hosts** file, execute the following commands:

```
hostname -i

hostname -f

hostname
```

g. To switch to the **appviewx** user, execute the following command: `sudo su – appviewx`
After the execution of this step, if the cloud connector does not exist on this host machine, the following prompt is displayed: **Would you like to opt for automated installation of the cloud connector? (y/n):**.

> **Note:** Proceeding with the cloud connector installation at this point will require you to enter the tenant ID and master key for the installation **manually** (you will not be able to copy the details and paste them in response to the prompt). It is, therefore, recommended to **not opt** for the installation at this point in the deployment process.

To install the AppViewX cloud connector at this point, press **y** and then press **Enter**.

After the execution of this step, a script is executed to validate if the cloud connector exists on this host machine and the following prompt is displayed: **Would you like to opt for automated installation of the cloud connector? (y/n):**.

> **Note:** Proceeding with the cloud connector installation at this point will require you to enter the tenant ID and master key for the installation **manually** (you will not be able to copy the details and paste them in response to the prompt). It is, therefore, recommended to **not opt** for the installation at this point in the deployment process.

To install the AppViewX cloud connector at this point, press **y** and then press **Enter**.

15. To install the AppViewX cloud connector at this point, enter **y**.

> **Note:** If you enter **n** here (that is, you choose to **not** go ahead with the installation here), **skip steps 16 to 20**. You will be reprompted for the automated installation after you login to the VM as the appviewx user.

16. When prompted **Please enter your AppViewX Cloud tenant name or ID**, enter the required details. Internet connectivity on the host machine is validated. The installation proceeds only if the host machine has internet access.

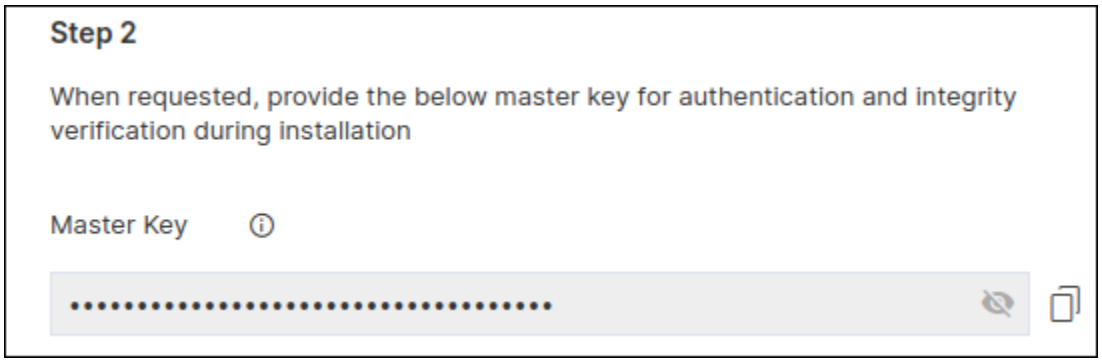
> **Note:** If internet access is unavailable, you will be prompted to proceed with installation using a proxy. Enter **y** and for instructions on proceeding with the installation with a proxy, click here.

17. When prompted to **Please provide the master key**, enter the required details.
    To retrieve the master key:

    a. Go to (**Menu**) > **Platform** > **Connectivity** > **Cloud Connector**.
    The **Settings :: Cloud Connector** inventory page is displayed.

    b. From the cloud connector details banner, under **Automate Cloud Connector Setup**, click **Steps to Automate Setup**.

    c. From the **Automated Cloud Connector Setup** window, under **Step 2**, copy the **Master Key**.



d. Paste the master key in the terminal window and press **Enter**.

The cloud connector installation script will check for the prerequisites and trigger the cloud connector installation.

When the cloud connector instance is successfully installed, a corresponding entry will be listed in the cloud connector inventory.

18. When prompted, **Please enter Datacenter**, enter the name of the data center on which this cloud connector will be deployed.

19. **Note:** Enabling auto-enrollment protocols is recommended only if the cloud connector is being installed in a demilitarized zone (DMZ) or devices in a restricted environment (that disables them from connecting to the <tenant>-aep directly.

When prompted to enable auto-enrollment protocols, enter **y** and enter the protocol name(s) you want to enable. For instructions on enabling auto-enrollment protocols, click here.

Note: By default, only the AppViewX certificate is enabled for auto-enrollment. To enable custom certificate for auto-enrollment:

a. Execute the command `./avxctl upgrade gateway-cert`.

b. When prompted, enter the location of the custom certificate.

On successful completion of the setup, a corresponding instance of this cloud connector is displayed in the inventory.

Note: The cloud connector installation on a OVA-based host machine will not prompt you to select if you want to manage F5 Big-IP devices. However, after the cloud connector has been installed you can copy the iControl jar in the deps/external_libs folder (click here for

instructions) and restart the starter and platform pods (click here for instructions), to enable this feature.

Note: Optional, required only for password authentication) In order to successfully execute the installation, AppViewX needs to run a script for which authentication via the **.pem** file needs to be bypassed. To do this, execute the following commands:

```
sudo sed -i 's/.*PasswordAuthentication.*/PasswordAuthentication yes/g' /etc/ssh/sshd_config

sudo systemctl restart sshd
```

20. To approve the cloud connector installation:

    a. Go to (**Menu**) > **Platform** > **Connectivity** > **Cloud Connector**.
    The **Settings :: Cloud Connector** inventory page is displayed.

    b. For the cloud connector instance installed, from the **Actions** column, click **Approve**.

Troubleshooting: If the AppViewX Cloud Connector instance has been approved but is not moved to the **Running** state, you can check the pod status and/or restart the pod(s), as required.

Setting up the AppViewX Cloud Connector using a Virtual Image on Azure

To install the AppViewX Cloud Connector in Azure, you need to create a virtual machine (VM) using the Azure Virtual Hard Disk (VHD). Microsoft Azure uses the Azure VHD file format to store the virtual machine (VM) disk images that are containers preloaded with the operating system, network, applications, and data requirements for setting up a virtual machine.

To deploy the AppViewX virtual machine for Azure:

1. Go to https://release.appviewx.com/Login and, from **Overview**, navigate to **2023.1.0 FP2**.
2. Scroll down to Production Images and download the latest artifact of Azure CC VHD, **AppViewX-2023.1.2.10-CC-Ubuntu-Azure-ddmmmyyy-vhd.tar.gz**.
3. Untar the downloaded artifact.
   ```
   tar -xvf AppViewX-2023.1.2.10-CC-Ubuntu-Azure-ddmmmyyy-vhd.tar.gz
   ```
4. Download the **Azure Storage Explorer** from here.

The **Azure Storage Explorer** is a desktop application that provides you with a GUI for easily managing your Azure resources.

Important: Install the Azure Storage Explorer at the same location as the downloaded Azure CC VHD artifact.

5. Using the Azure Storage Explorer, login to the Azure account for which the VM has to be created.
6. On successful login, go to the **disks** section and select the resource group.
   The resource group page is displayed.
7. Click **Upload**.
8. In the pop-up window displayed, enter/select the resource details.

Descriptions for the resources and their corresponding values

Resource	Value
Source VHD	<Disk file location>
Disk name	<Name of the disk>
OS type	**Linux**
Location	<region in which the VM is to be created>
Availability Zone	<zone name>
Account type	**Premium SSD**
Hyper-V Generation	**V1**
Architecture	**x64**

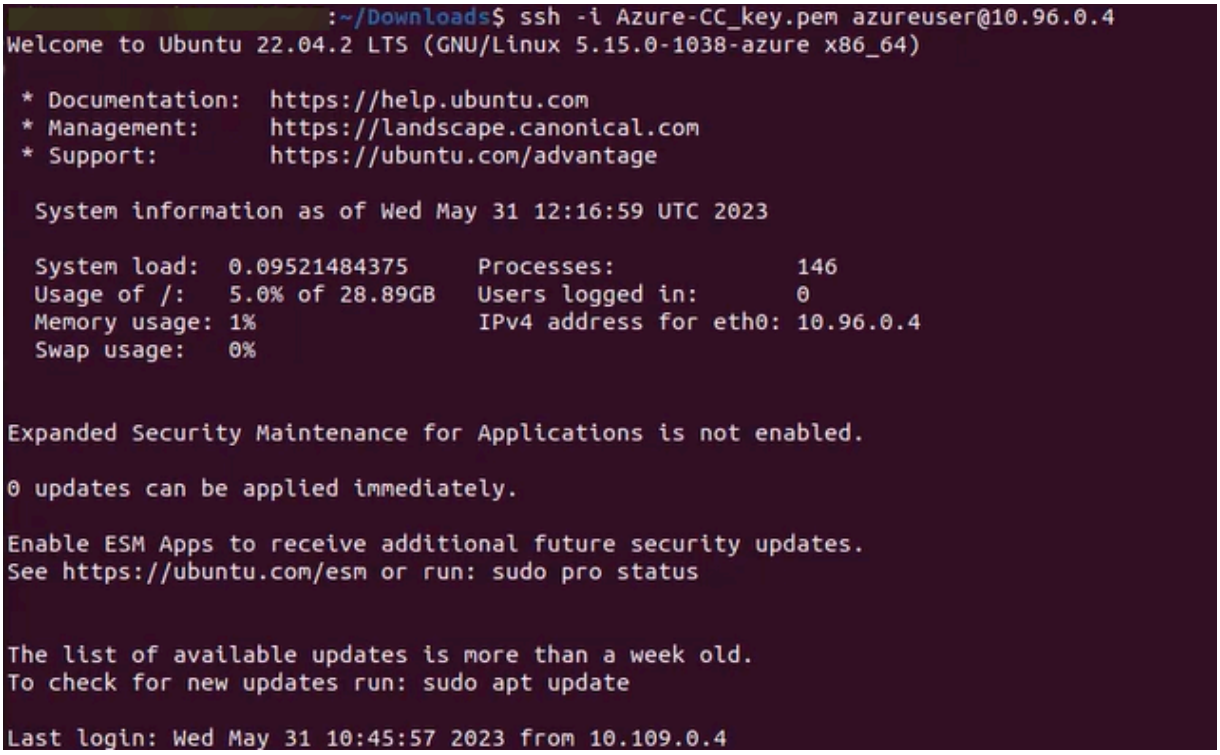
Note: All the values in bold, in the above table, are actual values and have to be assigned as is; values enclosed in angle brackets (**<>**) have to be assigned as per your specific configuration.

9. Click **Create**.
10. Once the disk is successfully uploaded to the Azure cloud, from the Azure portal, select the disk and click **+Create VM**.
11. On the **Create a virtual machine** page, configure the VM configurations based on your organization's standards and requirements.

12. Once the VM is successfully created, use the **.pem** file shared by AppViewX's SRE/TS team to login to the node (on which the cloud connector is to be installed). To do this, on the Command Line Terminal or Powershell, execute the following command:

    `ssh -i Azure-CC_key.pem azureuser@<ip_of_the_CC_VM>`

    Output:



13. To switch to the root user, execute the following command: `sudo -i`.

    The following prompt is displayed: **Would you like to opt for automated installation of the cloud connector? (y/n):**

14. Enter **n**.

    This question is asked here by default. However, since the nameserver has not been configured yet, it is **not recommended** to proceed with the installation because the connection with the cloud connector and mothership will not be established.

15. To enable login without the **.pem** file:

    a. Enable password authentication in the Azure node in which the AppViewX Cloud Connector is installed by executing the following commands:

    ```
    sudo sed -i 's/^\(PasswordAuthentication \).*/\1yes/' /etc/ssh/sshd_config

    sudo sed -i 's/^\(PasswordAuthentication \).*/\1yes/' /etc/ssh/sshd_config.d/50-cloud-init.conf
    ```

    b. Enable root password authentication by executing the following commands:

```
sudo sed -i '/^#PermitRootLogin/s/^#//' /etc/ssh/sshd_config

sudo sed -i 's/^PermitRootLogin.*/PermitRootLogin yes/' /etc/ssh/sshd_config
```

c. The SSH (Secure Shell) protocol is used for secure administration (login, command execution) over remote networks. For SSH changes to take effect, restart the SSH service. To do this, execute the following command: `sudo service ssh restart`

OR

To login without the **.pem** file, execute the following commands:

```
ssh appviewx@<ip/hostname>

ssh root@<ip/hostname>
```

16. To add an entry for the nameserver in the **resolv.conf** file, execute the following command: `echo "nameserver <IP of nameserver>" > /etc/resolv.conf`
17. Update the /etc/hosts file for the IP and the hostname of the VM created, using the following commands:

```
vi /etc/hosts


hostnamectl set-hostname “hostname-of-the-vm”
```

18. To validate the update to the **/etc/hosts** file, execute the following commands:

```
hostname -i

hostname -f

hostname
```

19. To switch to the **appviewx** user, execute the following command: `sudo su – appviewx`
    After the execution of this step, if the cloud connector does not exist on this host machine, the following prompt is displayed: **Would you like to opt for automated installation of the cloud connector? (y/n):**.
20. To install the AppViewX cloud connector at this point, enter **y**.

```
[x] - Cloud Connector does not exist on the server. Initiating the cloud connector setup on the server
[x] - Please make sure you have access to the tenant ID and master key for proceeding with the cloud connector setup
Would you like to opt for automated installation of the cloud connector? (y/n): y
Initiating the automated cloud connector installation...
Please enter your AppViewX Cloud tenant name or ID (e.g., my-tenant): kube-mar18-1
Checking the connectivity to the AppViewX Cloud Url...
[✓] - AppViewX Cloud Url is reachable. Downloading the cloud connector installer...
[✓] - Downloaded the installer script successfully.

Please provide the master key:

Checking the communication to AppViewX Cloud...

Downloading the cloud connector installer...
[✓] - Downloaded the installer successfully.

Untarring the installer...
Please enter the Datacenter: cbe
[✓] - Datacenter is valid!
```

21. When prompted **Please enter your AppViewX Cloud tenant name or ID**, enter the required details. Internet connectivity on the host machine is validated. The installation proceeds only if the host machine has internet access.

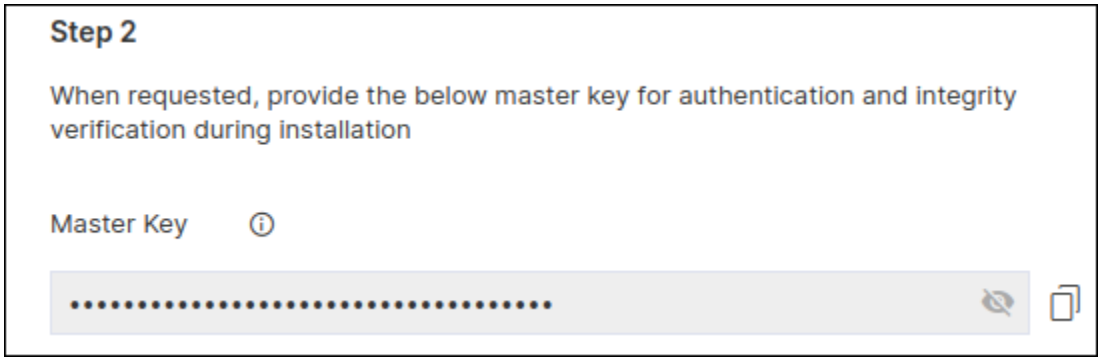
> **Note:** If internet access is unavailable, you will be prompted to proceed with installation using a proxy. Enter **y** and for instructions on proceeding with the installation with a proxy, click here.

22. When prompted to **Please provide the master key**, enter the required details.
    To retrieve the master key:

    a. Go to (**Menu**) > **Platform** > **Connectivity** > **Cloud Connector**.
    The **Settings :: Cloud Connector** inventory page is displayed.

    b. From the cloud connector details banner, under **Automate Cloud Connector Setup**, click **Steps to Automate Setup**.

    c. From the **Automated Cloud Connector Setup** window, under **Step 2**, copy the **Master Key**.



    d. Paste the master key in the terminal window and press **Enter**.

The cloud connector installation script will check for the prerequisites and trigger the cloud connector installation.

When the cloud connector instance is successfully installed, a corresponding entry will be listed in the cloud connector inventory.

23. When prompted, **Please enter Datacenter**, enter the name of the data center on which this cloud connector will be deployed.

24. **Note:** Enabling auto-enrollment protocols is recommended only if the cloud connector is being installed in a demilitarized zone (DMZ) or devices in a restricted environment (that disables them from connecting to the <tenant>-aep directly.

When prompted to enable auto-enrollment protocols, enter **y** and enter the protocol name(s) you want to enable. For instructions on enabling auto-enrollment protocols, click here.

Note: By default, only the AppViewX certificate is enabled for auto-enrollment. To enable custom certificate for auto-enrollment:

a. Execute the command `./avxctl upgrade gateway-cert`.

b. When prompted, enter the location of the custom certificate.

On successful completion of the setup, a corresponding instance of this cloud connector is displayed in the inventory.

Note: The cloud connector installation on a OVA-based host machine will not prompt you to select if you want to manage F5 Big-IP devices. However, after the cloud connector has been installed you can copy the iControl jar in the deps/external_libs folder (click here for instructions) and restart the starter and platform pods (click here for instructions), to enable this feature.

Note: Optional, required only for password authentication) In order to successfully execute the installation, AppViewX needs to run a script for which authentication via the **.pem** file needs to be bypassed. To do this, execute the following commands:

```
sudo sed -i 's/.*PasswordAuthentication.*/PasswordAuthentication yes/g' /etc/ssh/sshd_config

sudo systemctl restart sshd
```

25. To approve the cloud connector installation:

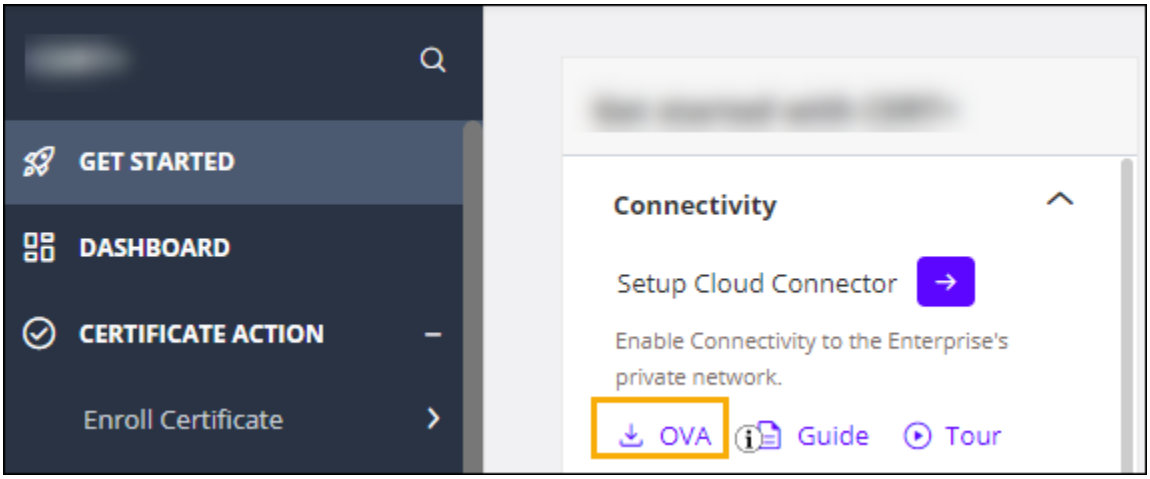
    a. Go to (**Menu**) > **Platform** > **Connectivity** > **Cloud Connector**.
    The **Settings :: Cloud Connector** inventory page is displayed.

    b. For the cloud connector instance installed, from the **Actions** column, click **Approve**.

> **Troubleshooting:** If the AppViewX Cloud Connector instance has been approved but is not moved to the **Running** state, you can check the pod status and/or restart the pod(s), as required.

Setting up the AppViewX Cloud Connector using a Virtual Image on GCP

1. Log in to the GCP console and start a **Cloud Shell** instance.
2. From the command line terminal, create a bucket with the default settings.
   `gsutil mb gs://my-virtual-appliances-bucket`
3. Download the GCP OVA from the AppViewX release portal.
   Alternatively, you can Download the release package in the OVA format, from the respective AppViewX's product line landing page, under **GET STARTED** menu > **Connectivity** section, click OVA.



4. Upload it to the Cloud Shell. To upload/copy the OVA to the GCP bucket, execute the following command:

```
gsutil cp ~/path-to-file/local

  gs://my-virtual-appliances-bucket/my-va-file.ova
```

5. Create a virtual instance from the OVA.

```
gcloud compute instances import <my-instance> \

  --source-uri=gs://my-virtual-appliances-bucket/my-va-file.ova \

  --zone southamerica-east1-a \

  --os=ubuntu-1804
```

6. Login to the GCP node using AppViewX credentials.
   `ssh appviewx@<node IP>`
7. Enter the password when prompted.
   The following prompt is displayed: **Would you like to opt for automated installation of the cloud connector? (y/n):**
8. Enter **n**.
   This question is asked here by default. However, since the nameserver has not been configured yet, it is **not recommended** to proceed with the installation because the connection with the cloud connector and mothership will not be established.
9. To switch to the root user, execute the following command: `sudo -i`.
   The following prompt is displayed: **Would you like to opt for automated installation of the cloud connector? (y/n):**
10. Enter **n**.
    This question is asked here by default. However, since the nameserver has not been configured yet, it is **not recommended** to proceed with the installation because the connection with the cloud connector and mothership will not be established.
11. To add an entry for the nameserver in the **resolv.conf** file, execute the following command: `echo "nameserver <IP of nameserver>" > /etc/resolv.conf`
12. Update the /etc/hosts file for the IP and the hostname of the VM created, using the following commands:

```
vi /etc/hosts



hostnamectl set-hostname “hostname-of-the-vm”
```

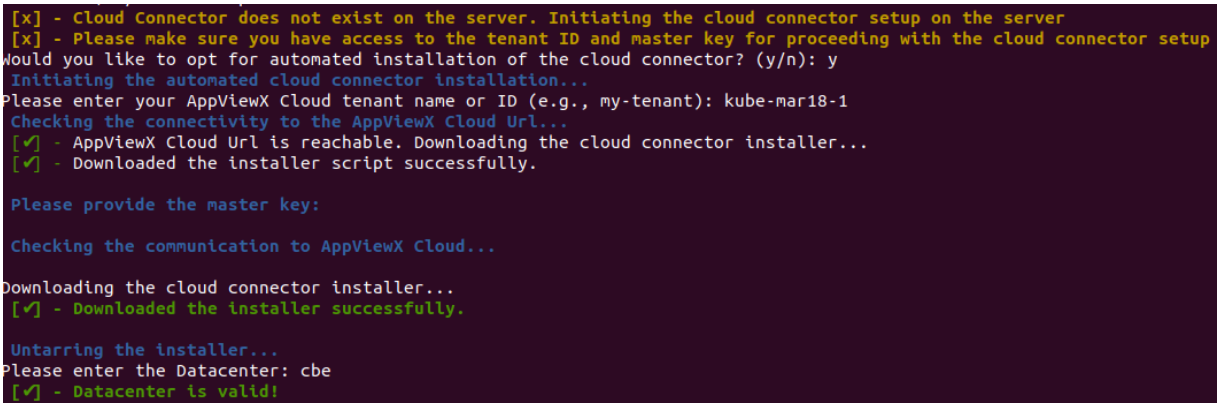
13. To validate the update to the **/etc/hosts** file, execute the following commands:

```
hostname -i

hostname -f

hostname
```

14. To switch to the **appviewx** user, execute the following command: `sudo su – appviewx`

On login, if the cloud connector does not exist on this host machine, the following prompt is displayed: **Would you like to opt for automated installation of the cloud connector? (y/n):**

15. To install the AppViewX cloud connector at this point, enter **y**.



16. When prompted **Please enter your AppViewX Cloud tenant name or ID**, enter the required details. Internet connectivity on the host machine is validated. The installation proceeds only if the host machine has internet access.

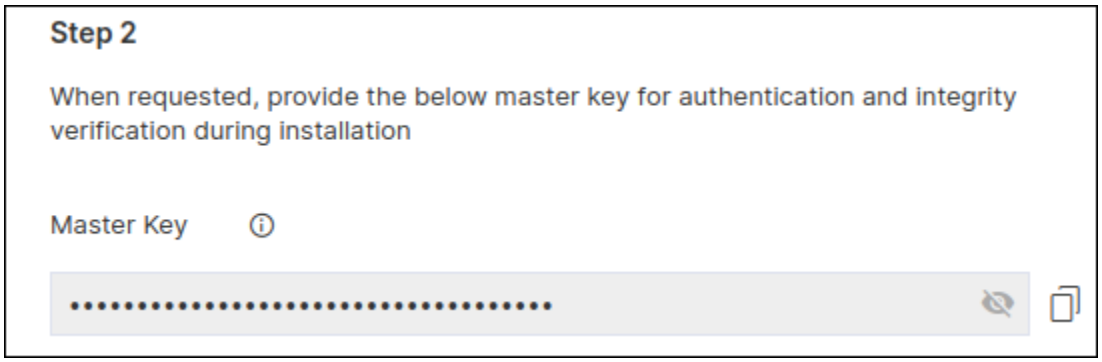
> **Note:** If internet access is unavailable, you will be prompted to proceed with installation using a proxy. Enter **y** and for instructions on proceeding with the installation with a proxy, click here.

17. When prompted to **Please provide the master key**, enter the required details.
    To retrieve the master key:

    a. Go to (**Menu**) > **Platform** > **Connectivity** > **Cloud Connector**.
    The **Settings :: Cloud Connector** inventory page is displayed.

    b. From the cloud connector details banner, under **Automate Cloud Connector Setup**, click **Steps to Automate Setup**.

    c. From the **Automated Cloud Connector Setup** window, under **Step 2**, copy the **Master Key**.



d. Paste the master key in the terminal window and press **Enter**.

The cloud connector installation script will check for the prerequisites and trigger the cloud connector installation.

When the cloud connector instance is successfully installed, a corresponding entry will be listed in the cloud connector inventory.

18. When prompted, **Please enter Datacenter**, enter the name of the data center on which this cloud connector will be deployed.

19. **Note:** Enabling auto-enrollment protocols is recommended only if the cloud connector is being installed in a demilitarized zone (DMZ) or devices in a restricted environment (that disables them from connecting to the <tenant>-aep directly.

When prompted to enable auto-enrollment protocols, enter **y** and enter the protocol name(s) you want to enable. For instructions on enabling auto-enrollment protocols, click here.

Note: By default, only the AppViewX certificate is enabled for auto-enrollment. To enable custom certificate for auto-enrollment:

a. Execute the command `./avxctl upgrade gateway-cert`.

b. When prompted, enter the location of the custom certificate.

On successful completion of the setup, a corresponding instance of this cloud connector is displayed in the inventory.

Note: The cloud connector installation on a OVA-based host machine will not prompt you to select if you want to manage F5 Big-IP devices. However, after the cloud connector has been installed you can copy the iControl jar in the deps/external_libs folder (click here for

instructions) and restart the starter and platform pods (click here for instructions), to enable this feature.

Note: Optional, required only for password authentication) In order to successfully execute the installation, AppViewX needs to run a script for which authentication via the **.pem** file needs to be bypassed. To do this, execute the following commands:

```
sudo sed -i 's/.*PasswordAuthentication.*/PasswordAuthentication yes/g' /etc/ssh/sshd_config

sudo systemctl restart sshd
```

20. To approve the cloud connector installation:

    a. Go to (**Menu**) > **Platform** > **Connectivity** > **Cloud Connector**.
    The **Settings :: Cloud Connector** inventory page is displayed.

    b. For the cloud connector instance installed, from the **Actions** column, click **Approve**.

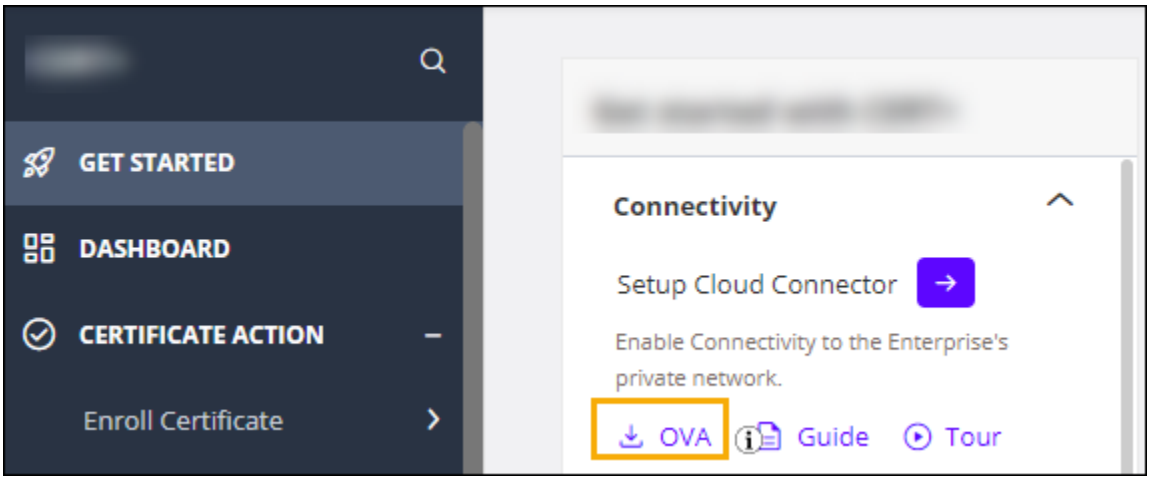
Troubleshooting: If the AppViewX Cloud Connector instance has been approved but is not moved to the **Running** state, you can check the pod status and/or restart the pod(s), as required.

Setting up the AppViewX Cloud Connector via a Virtual Image using the AppViewX User Interface

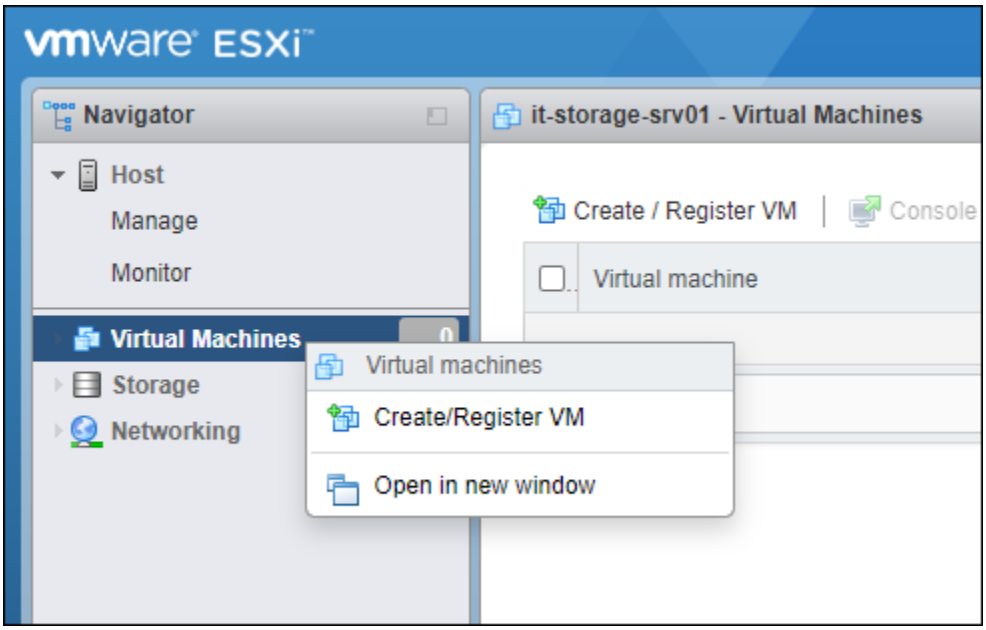
- Setting up the AppViewX Cloud Connector using a Virtual Image for an On-prem Deployment
- Setting up the AppViewX Cloud Connector using a Virtual Image on AWS
- Setting up the AppViewX Cloud Connector using a Virtual Image on Azure
- Setting up the AppViewX Cloud Connector using a Virtual Image for GCP

Setting up the AppViewX Cloud Connector using a Virtual Image for an On-prem Deployment

1. To download the release package in the OVA format, from the respective AppViewX's product line landing page, under **GET STARTED** menu > **Connectivity** section, click OVA.



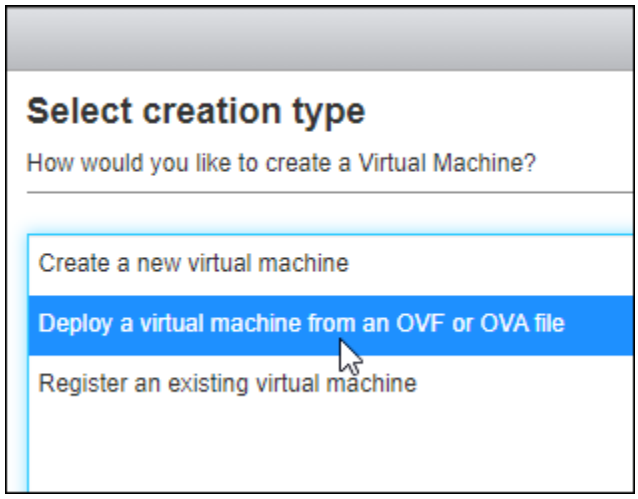
2. Log in to the **vmware** client.
3. From the **Navigation** pane on the left, right click **Virtual Machines**.
4. Click **Create/Register VM**.



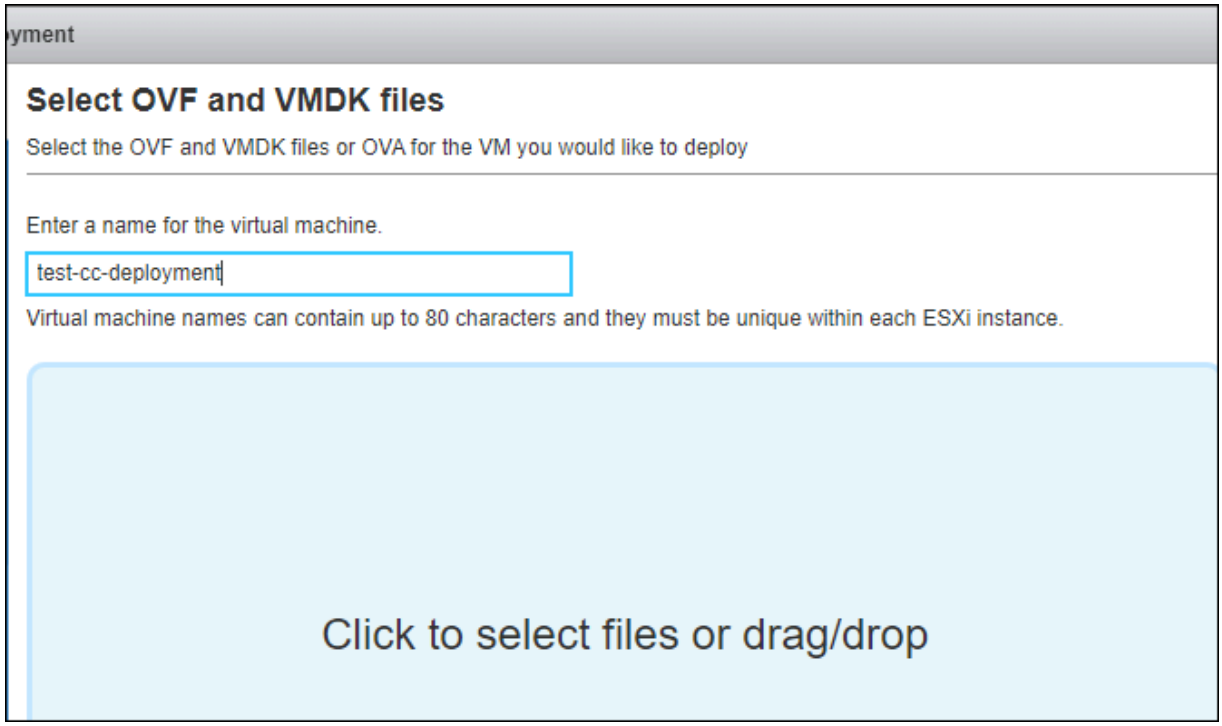
The **New Virtual machine** window is displayed.

5. From the navigation pane in the left, select **Select creation type**.

6. In the **Select creation type** window, select the **Deploy a virtual machine from an OVA or OVF file** option.



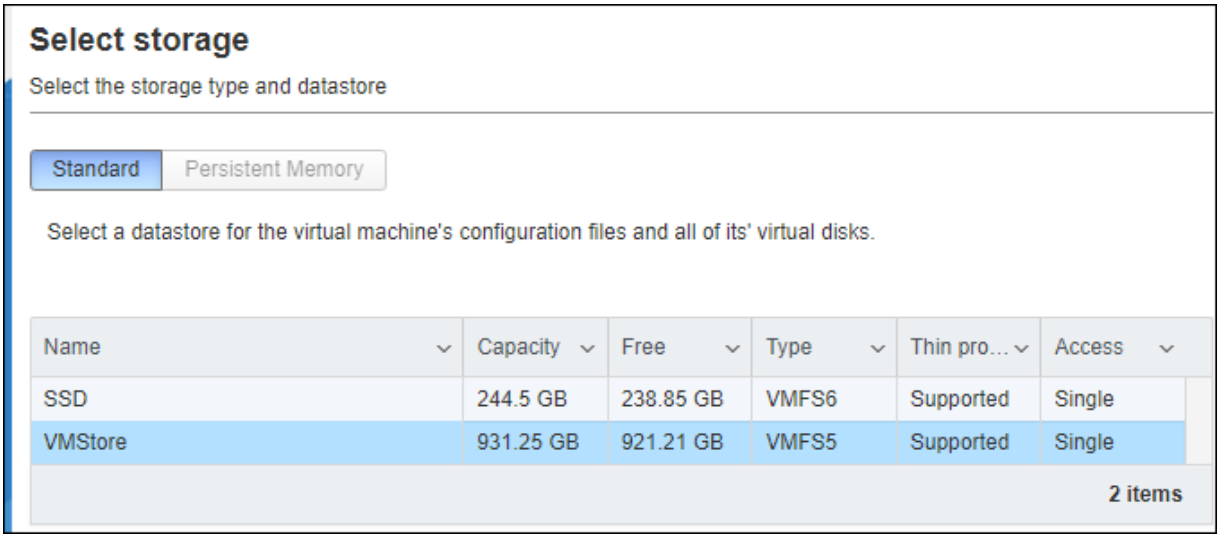
7. Click **Next**.
8. In the **Select OVF and VMDK files** window:



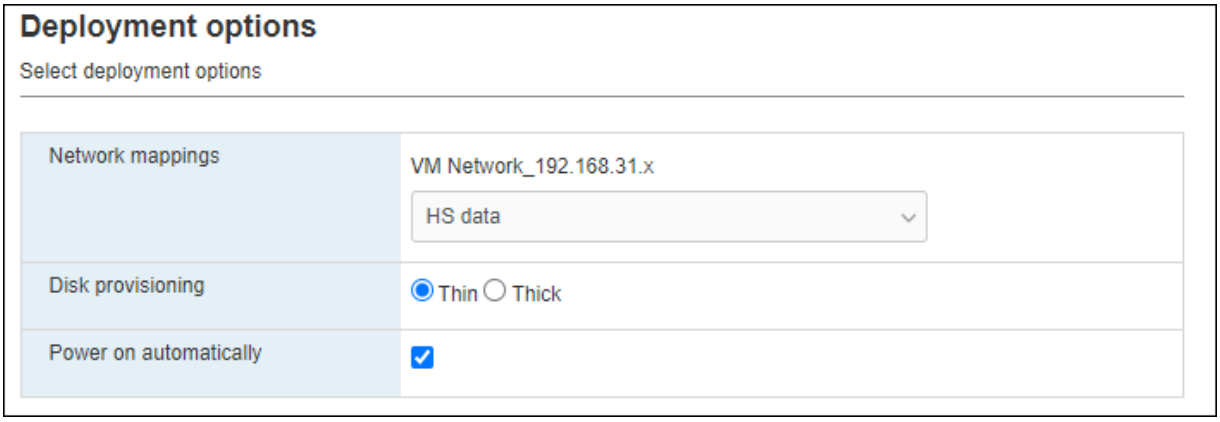
   a. Enter a name for the virtual machine.
      For the purpose of this document, we will name it **test-cc-deployment**.

   b. In the **Click to select files or drag/drop** area, click and, from the file explorer, navigate to the location of the file, select the file, and click **Open**.

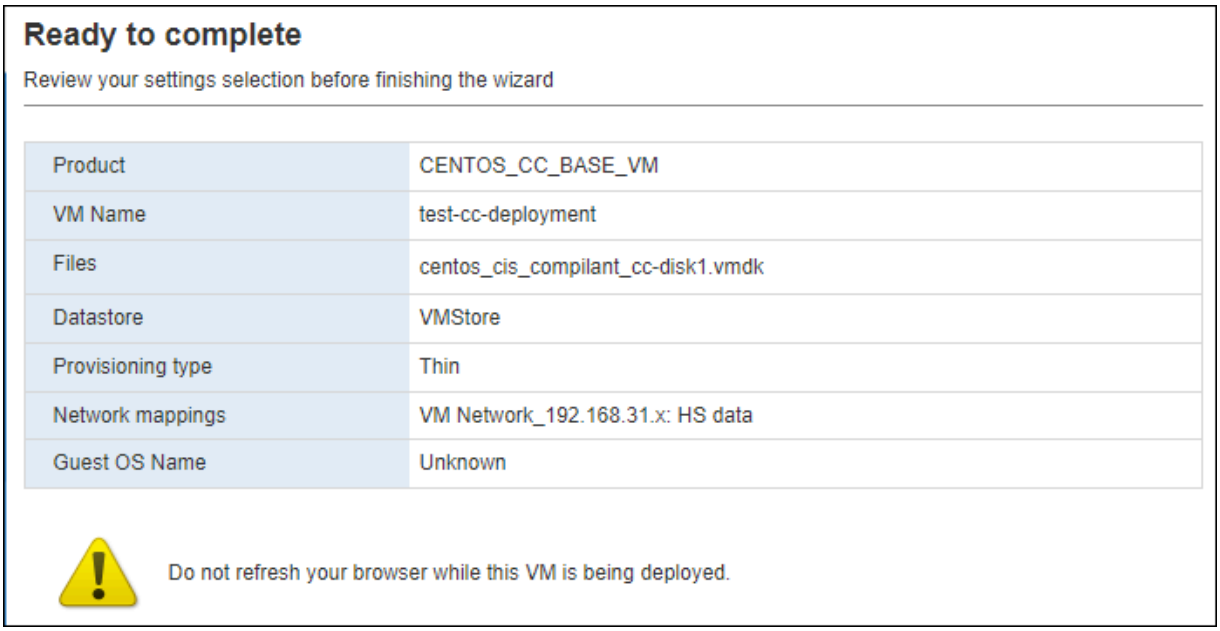
9. Click **Next**.
10. In the **Select storage** window, from the available options, select a datastore for storing the virtual machine's files and all of its virtual disks.



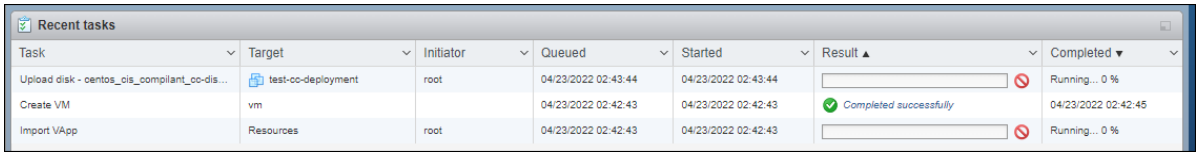
11. Click **Next**.
12. In the **Deployment options** window:

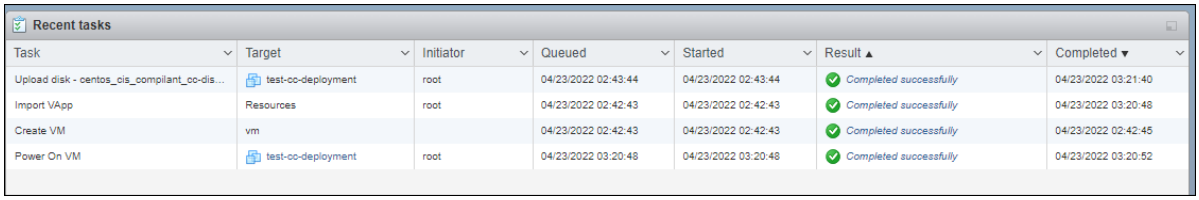


    a. Select the network mapping.
    b. Select the disk provisioning required.
    c. Select the **Power on automatically** checkbox.
13. Click **Next**.
14. In the **Ready to complete** window, review your settings.

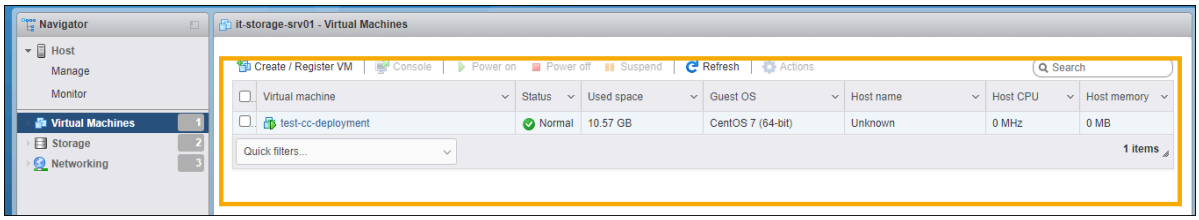


15. Click **Finish**.
    - The progress of the OVA deployment is shown in the **Recent Tasks** section.





    - On successful completion of the OVA deployment, the new virtual machine is displayed in the **Virtual Machines** inventory. For each virtual machine in the inventory, the following details are displayed:

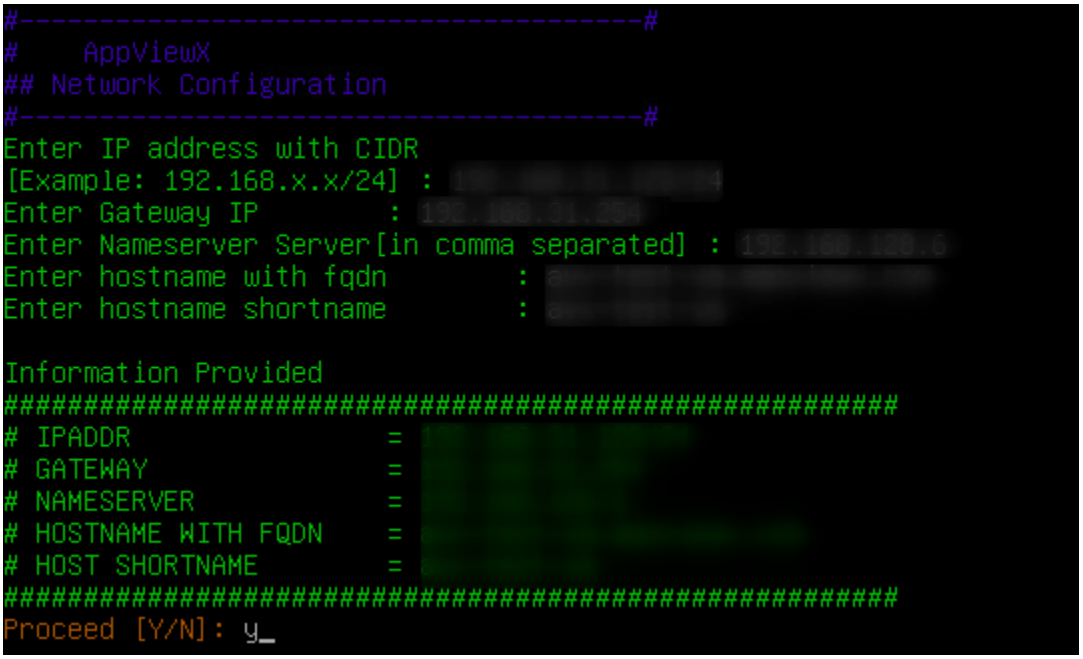


16. From the **Virtual Machines** inventory, click the virtual machine just added.
    The terminal window for the virtual machine is displayed. The script for configuring the network IP is executed automatically.

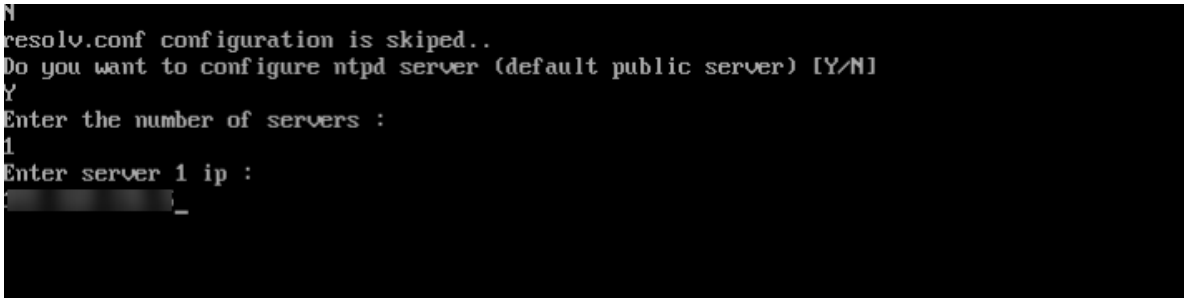
17. To configure the IP address, when prompted, enter the required values for the following requested parameters:

```
=========================================

IPADDR   = XXX.XXX.XXX.XXX

NETMASK  = XXX.XXX.XXX.XXX

GATEWAY  = XXX.XXX.XXX.XXX

================================
```

For example, refer to the sample screenshot below:



18. To configure the hostname and the DNS, when prompted, press **Y**. If you prefer to configure the hostname and DNS manually, to skip this step, press **N**.
19. To configure an NTP server(s):
    a. When prompted, **Do you want to configure ntpd server (default public server)**press **Y**.



    b. Enter the number of NTP servers to be configured.

c. For the number of servers entered above, enter the IP address of each NTP server on a new line.

d. To update the **ntp.conf** file with the IP addresses provided above, press **Y**.

After the execution of this step, if the cloud connector does not exist on this host machine, the following prompt is displayed: **Would you like to opt for automated installation of the cloud connector? (y/n):**.

20. Since these instructions are for setting up the cloud connector via the user interface, enter **n**.

```
Authorized uses only. All activity may be monitored and reported.
appviewx@192.168.31.143's password:
You have new mail.
Last login: Fri Apr 12 04:37:30 2024 from 192.168.236.254
 [x] - Cloud Connector does not exist on the server. Initiating the cloud connector setup on the server
 [x] - Please make sure you have access to the tenant ID and master key for proceeding with the cloud connector setup
Would you like to opt for automated installation of the cloud connector? (y/n): n
 [x] - Skipping Automated Cloud Connector installation. Refer Cloud Connector User Guide to set it up manually.
```

21. When prompted, login to the VM as the appviewx user, using the credentials shared by AppViewX's customer support team.

> **Note:** Root user access is required for maintaining the OS configuration and for patching security updates. Since direct root access is not provided, you can:
>
> a. Login as the **appviewx** user.
>
> b. Switch to the root user by executing the command `sudo -i`.

> **Note:** It is recommended that, after the first login, change the default credentials.

22. Since these instructions are for setting up the cloud connector via the user interface, when prompted **Would you like to opt for automated installation of the cloud connector? (y/n):**, enter **y**.
23. When prompted **Are you sure you want to continue connecting**, press **y** and then press **Enter**.

```
:~$ ssh appviewx@
The authenticity of host '1                ' can't be established.
ECDSA key fingerprint is SHA256:fNQG4/+CfA9BC/bBQpSnf9Ocke2JvUxWJP5rrPIfHww.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '               ' (ECDSA) to the list of known hosts.
Authorized uses only. All activity may be monitored and reported.
appviewx@                  's password:
```

24. To check if the Docker is up and running, execute the command: `systemctl status docker`.
    If the Docker status is **active (running)**, as shown in the screenshot below, it means that the OVA has been deployed successfully.

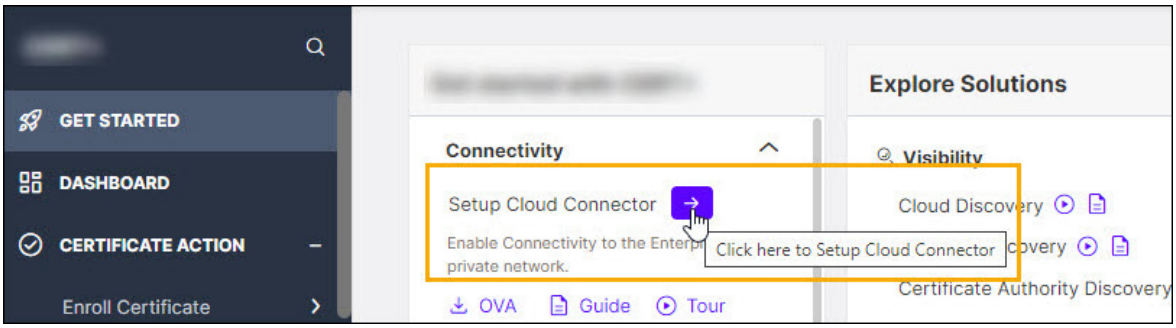
```
[appviewx@ccnode ~]$ systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
   Active: active (running) since Tue 2022-06-14 06:08:50 EDT; 6 days ago
     Docs: https://docs.docker.com
 Main PID: 1540 (dockerd)
   CGroup: /system.slice/docker.service
           └─1540 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

Note: To check if the Docker is accessible to the appviewx user, execute the following command:

```
docker image ls
```

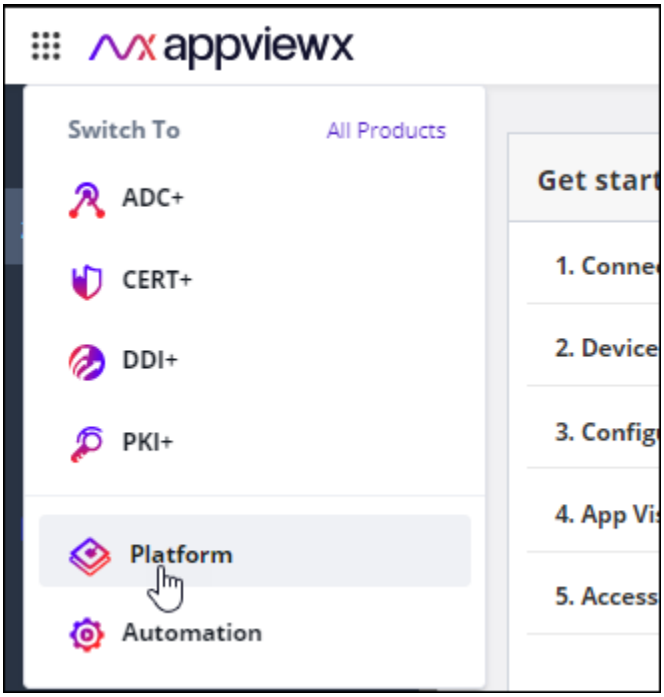
If the command does not return an error, it means that the Docker is accessible to the appviewx user:

25. Enter your account URL (for example, https://tenant-name.appvx.com/appviewx/login) in the address bar of your browser.
    The AppViewX login page is displayed.
26. Login to AppViewX.
27. Navigate to the cloud connector's setup interface.
    There are three ways you can access the interface for setting up the AppViewX Cloud Connector:
    - From the product landing page (that you will see as soon as you have logged in)
      - Expand the **Connectivity** section and click  .

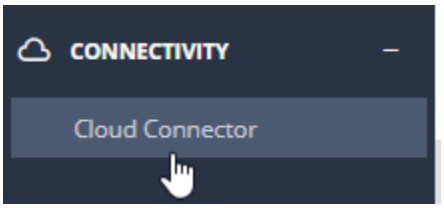


      You will be redirected to the **Settings :: Cloud Connector** page.
    - From the new navigation menu (displayed by default starting product version 2022.1.0 FP3 onwards):

a. From the menu in the top-right corner of the page, select **Platform**.



b. From the **Platform** menu, under **Connectivity**, click **Cloud Connector**.



The **Settings :: Cloud Connector** page is displayed.

• From the old navigation menu:

> **Note:** For instructions on switching between the new and the old navigation menus, click here.

a. From the top right corner of the landing page, click the menu icon.

b. From the menu displayed, navigate to **Settings** > **Cloud Connector**.

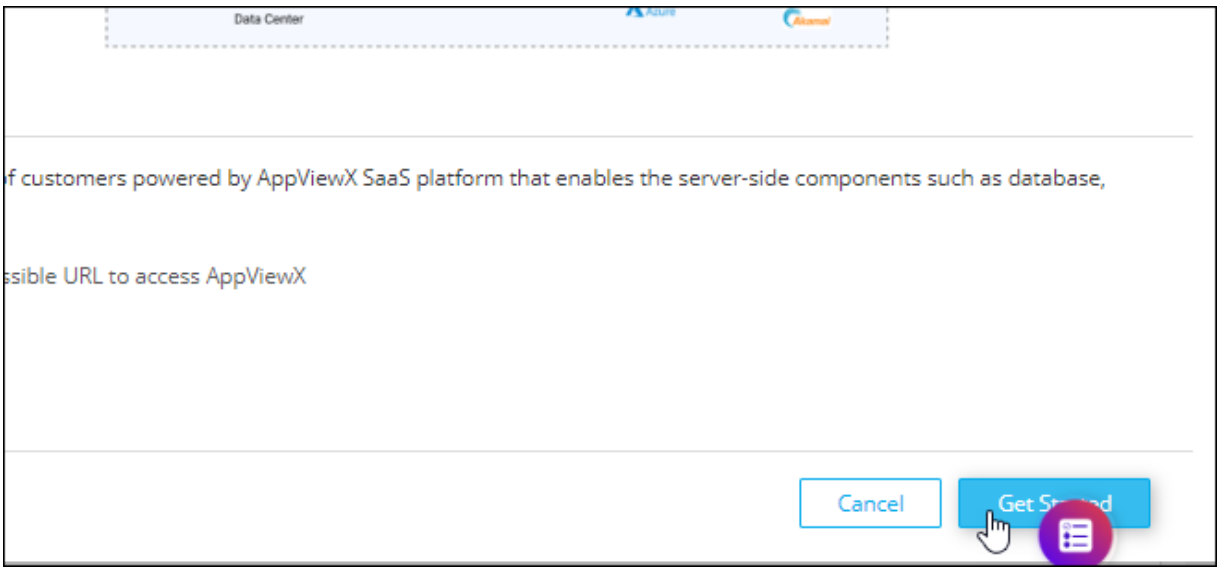
The **Settings :: Cloud Connector** page is displayed.

28. On the **Setting :: Cloud Connector** page, click **Add Cloud Connector**.

The **Cloud Connector Setup** screen is displayed.

The landing page gives you a quick introduction to the AppViewX Cloud Connector, with a graphical representation of how the infrastructure is deployed and works.

29. To start with the process of adding the cloud connector, from the bottom-right corner of the screen, click **Get Started**.



You will be redirected to the **Basic Information** page.

30. On the **Basic Information** page, configure the basic cloud connector settings.

    a. To install the cloud connector via the virtual image, from **Installation Type**, select **Virtual Image**.

    > **Note:** Click here to read how a virtual image-based installation is different from a native OS installation.

    b. In the **Cloud Connector Name (FQDN)** field, enter the hostname of the machine on which the AppViewX Cloud Connector will be installed.

    > **Tip:** To retrieve the hostname, from the command line terminal of the host machine, execute the following command: `hostname -f`:

    > **Note:** The hostname entered here is added to the license file that will be generated and downloaded as part of the installer. Therefore, the license file can be used to install the cloud connector only on the machine with the entered hostname and no other.

> **Tip:** The **Setup Cloud Connector** section to the right of the **Basic Information** screen lists hyperlinks to the prerequisites required for setting up the AppViewX Cloud Connector. To read more about what the AppViewX Cloud Connector offers, click **Learn More**.

c. Click **Next**.

31. [Optional] Execute a prerequisite check script.

> **Note:** This is an **optional** step. The prerequisite check script is executed automatically at the time of installing the AppViewX Cloud Connector and the results are shown as a part of the installation logs.

To simplify compliance to the AppViewX Cloud Connector installation prerequisites, you can execute a script to identify and rule out any deviations from the prerequisites.

a. On the **Basic Information** screen, under **Setup Cloud Connector**, you will see a list of the installation prerequisites.

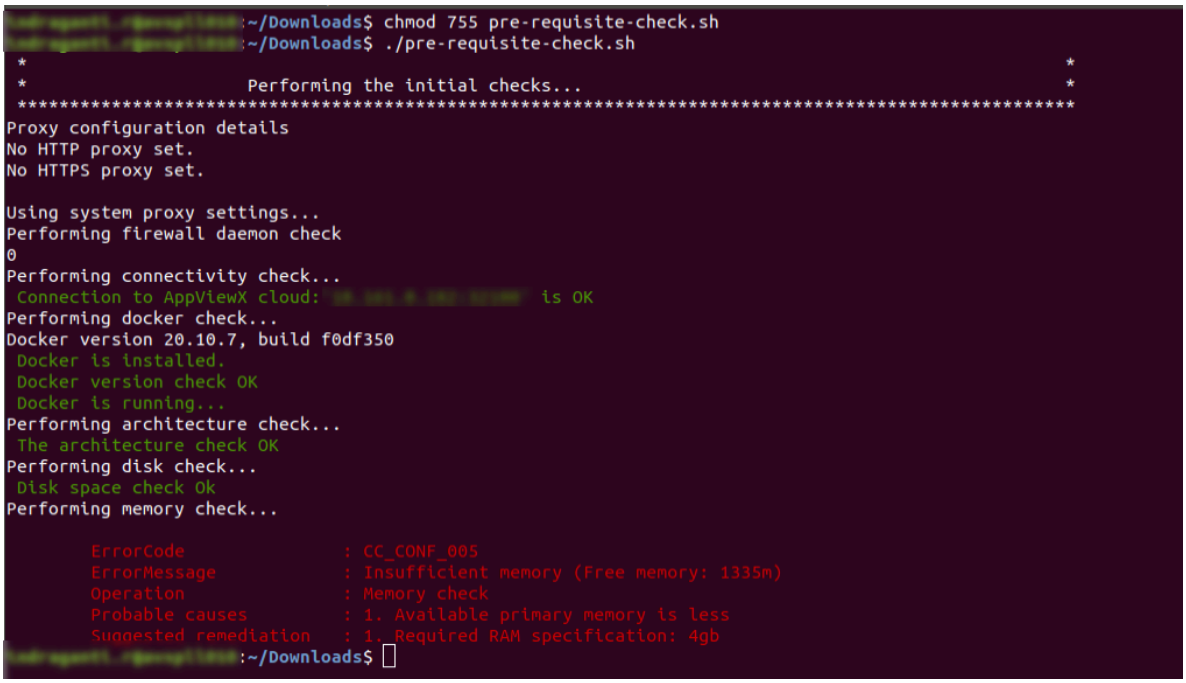
From this list, for **Executing the Prerequisites Check Script**, to download the script, click . The **pre-requisite-check.sh** script file is downloaded.

b. Securely copy the **pre-requisite-check.sh** via SCP/SFTP to the host machine where the AppViewX Cloud Connector is to be installed

c. Convert the downloaded script file into an executable file using the chmod command, as shown below:`chmod 755 pre-requisite-check.sh`

d. Execute the **.sh** prerequisite check script file: `./pre-requisite-check.sh`
If the node does not meet the prerequisites for the AppViewX Cloud Connector installation, the output of the command returns an error code and the corresponding error message, causes, and fixes, if any.

For example, as seen in the sample output in the image below, the prerequisite check for the memory requirement has failed.



> **Note:** For resolutions to the prerequisite check failure scenarios, click here.

32. Click **Next**.
    You will be navigated to the **AssignData Center** screen, where, for deploying the AppViewX Cloud Connector, you can either select an existing data center or add a new one.
33. To use an existing data center, select one from the options displayed on the **Assign Data Center** screen.

> **Tip:** Alternatively, you can use the **Search…** field on this screen to search for an existing data center.

To add a new data center:

a. Click **Add Data Center.**

b. In the **Add Data Center** dialog box, enter a name for the new data center.

c. Click **Save**.
   The new data center will now be displayed on the **Assign Data Center** screen along with the other existing data centers.

d. Select the required data center.

> **Tip:** The **Data Center based routing** section to the right of the **Assign Data Center** screen explains the concept of data center-based routing and how you can achieve high availability. To read more on this, click **Learn More** from the top-right corner of this screen.

34. Click **Next**.
    The **Advanced Configuration** screen is displayed.
35. On the **Advanced Configuration** page, to configure the TLS authentication and proxy server settings for your cloud connector:

    a. Enter/Select the advanced configuration settings for the AppViewX Cloud Connector.

> **Note:** The **Data center** field is auto-populated based on your selection on the **Assigning a Data Center** screen.

Field	Description
TLS Authentication	**Tip:** The section on the right of the screen gives you a brief context of what is **TLS Authentication**. To read more, click **Learn More** (next to the **TLS Authentication** heading). • To auto-generate a TLS certificate, select **Auto-generate** (default selection). By default, the certificate is generated using the AppViewX CA. **Note:** The created certificate is available in the certificate inventory. You can: • Assign this certificate to a certificate group • Configure a certificate expiry alert for this certificate group from the **Server Certificate** dashboard, using the **Certificate Summary Report** widget settings • To enter details of a custom TLS certificate, select **Custom**. The **TLS Certificate Password** and **Custom TLS Certificate** fields are displayed. The instructions for filling these fields are given below.

Field	Description
TLS Certificate Password*	**Note:** This field is displayed only if you have selected to enter details of a Custom TLS certificate in the **TLS Authentication** field. Password of the TLS certificate (that will be uploaded in the next step) **Note:** This is a mandatory field if a Custom TLS certificate is uploaded. AppViewX supports only password-protected Custom TLS certificates.
TLS Certificate	**Note:** This field is displayed only if you have selected to enter details of a Custom TLS certificate in the **TLS Authentication** field. To upload a custom TLS certificate: i. To navigate to the location of the custom TLS certificate, click within the field. ii. Select the certificate file. iii. Click **Open**. iv. To upload the custom TLS certificate selected, click **Upload**. **Note:** AppViewX supports only password-protected Custom TLS Certificates.
Use proxy	**Tip:** The section on the right of the screen gives you a brief context of what is **Proxy based routing**. To read more, click **Learn More** (next to the **Proxy based routing** heading). A proxy server is required if the AppViewX Cloud Connector is unable to connect to your endpoints available in the internet. To use a proxy server for the deployment:

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>
i. Select the Use proxy checkbox.

ii. To select a preconfigured proxy (for the selected data center), from the Select Proxy dropdown list, select a proxy server.

OR

To create a new proxy server setting:

i. Use the Click here option shown below the Select Proxy dropdown list.

The Add Proxy pop-up screen is displayed.

ii. Enter/Select the details required to add a proxy.

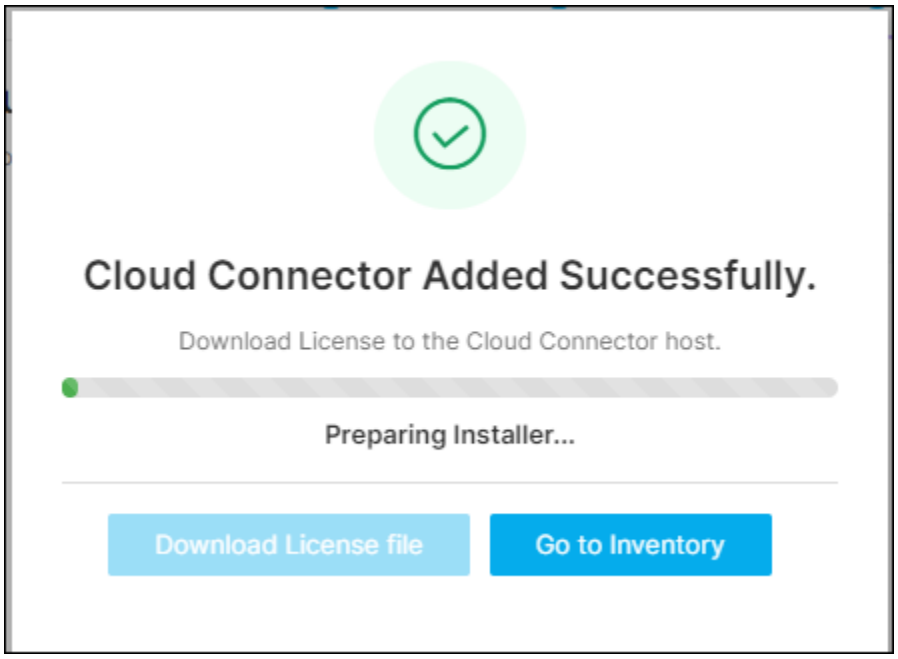
Field descriptions for the Add Proxy details
<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>*Proxy Name</td><td>Name of the proxy server</td></tr>
<tr><td>*Server IP</td><td>IP address/FQDN of the proxy server</td></tr>
<tr><td>*Port</td><td>Port number of the proxy server</td></tr>
<tr><td>URL</td><td>From the dropdown menu, select the URL.</td></tr>
<tr><td>Authentication</td><td>To enable authentication for accessing the proxy server, select this checkbox.</td></tr>
<tr><td>*Username</td><td>Note: This field is enabled only when Authentication is selected.

Enter the username required for accessing the proxy server.</td></tr>
</table>
</td></tr>
</table>

Field	Description
	<table><tr><th>Field</th><th>Description</th></tr><tr><td>***Password**</td><td>**Note:** This field is enabled only when **Authentication** is selected. Enter the password required for accessing the proxy server.</td></tr></table>

b. Click **Finish**.

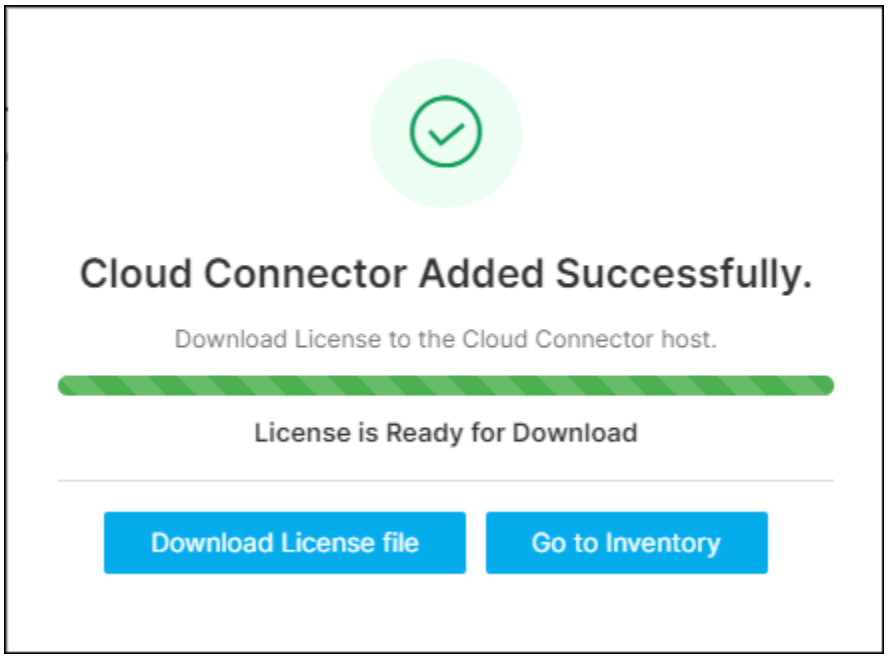
A confirmation message is displayed. AppViewX begins preparing the installer and the license file. Once the license file is ready, you can download it and proceed with the installation of the AppViewX Cloud Connector.



36. Download the license file.

Note: The installer is prepackaged with the OVA, so, for a virtual image-based installation, you only need to download the license file.

a. On the **Cloud Connector Added Successfully** dialog box, when the **License is Ready for Download**, click **Download License file**.



Tip: At this point, if the installer has been deleted or is not usable, and you wish to revert to a native installation, click **Go to Inventory**. It will take you back to the cloud connector inventory, from where you can download the license file and installer for the native OS download.

Tip: You can also choose to download the license file and the installer package individually. To do this:

i. Click the **Cloud Connector Name**.

   The selected Cloud Connector's details are shown in a pane to your right.

ii. To download the AppViewX Cloud Connector installer package, click **Download Cloud Connector**. This is useful in the event that the installer has been deleted or is no longer usable.

   To download the license file, click **Download License**.

Note: A installer download is made available even for a virtual-image based deployment, to help you with reconfiguration in case the existing OVA configuration is deleted.

b. Save the license file on the OVA node.

On the **Settings :: Cloud Connector** page, details of this AppViewX Cloud Connector are added in the inventory table, which is explained here.

37. Install the AppViewX Cloud Connector Agent.

> **Note:** The following steps assume that:
> • All system prerequisites are fulfilled by the host machine.
> • The AppViewX Cloud Connector installer (downloaded in the above step) is securely copied via SCP/SFTP to the host machine where the AppViewX Cloud Connector is to be installed.

a. On the node where the AppViewX Cloud Connector agent will be installed, from the extracted installation package, run the **./install.sh** script.

The script will check if the installation prerequisites for the AppViewX Cloud Connector have been fulfilled.

> **Note:** Ensure that the license file is placed in the same location as the **install.sh** script. If the license file is placed in another location, run the install.sh script using the following command:
>
> ```
> ./install.sh <complete path of the license file with the filename>
> ```

On successful verification of the prerequisites, you will be prompted to specify if you want to manage f5 BIG-IP devices and if you need auto-enrollment of the certificates.

```
Do you want to manage f5 BIG-IP devices? (y/n)?:n
Continuing with the installation

Do you need Auto-enrollment of the certificate using EST/SCEP/ACME? (y/n)?:y
Please choose one or more protocol (use comma separated numbers): 1)EST(MTLS)   2)SCEP(HTTP)  3)ACME(HTTPS)
1,2,3
Auto enrollment enabled successfully for protocol(s): MTLS HTTP HTTPS
Do you want to enable Syslog receiver for a near real time configuration updates from the devices. (y/n) n
syslog enabled  n
```

b. When prompted, enter the required input value(s):

> **Important:** If you choose to **not enable** any of the following features, to enable them later, you will have to reinstall the AppViewX Cloud Connector.

i. If you want manage f5 BIG-IP devices, enter **y**/**n** for yes/no, respectively.

ii. When prompted to enable auto-enrollment of the certificate using one of the following supported auto-enrollment protocols, enter **y** only if the cloud connector is being installed in a demilitarized zone (DMZ) or devices in a restricted environment (that disables them from connecting to the **<tenant>-aep** directly.

• If you choose **y** (yes) here, enter the required protocol(s) name.

> **Note:** By default, the AppViewX certificate is enabled for auto-enrollment. To enable custom certificate for auto-enrollment:
>
> 1. Execute the command `./avxctl upgrade gateway-cert`.
> 2. When prompted, enter the location of the custom certificate.

> **Note:** If you are a KUBE+ customer, the auto-enrollment gateway should be enabled as part of the installation for your KUBE+ use cases to work via the cloud connector.

iii. If you want to enable Syslog receiver for a near-real time configuration updates from the devices, enter **y/n** for yes/no, respectively. For **configuring Syslog reception**, refer to the Platform User guide section, Syslog Reception.

In case you have an older version of AppViewX on cloud and want to make use of Syslog capabilities for ADC, you must manually activate the Syslog flag by setting **SYSLOG_ENABLED**=**true** in the path **ccpath/deps/properties**.

c. Enter the sudo password.

After the relevant details have been entered, the installation proceeds. Installation logs, according to the outcome of the installation, are displayed.

Given below are sample installation logs:

```
Loaded image: rancher/k3s:v1.23.3-k3s1
Loaded image: rancher/k3d-tools:5.2.2
Loaded image: rancher/mirrored-pause:3.6
[36mINFO[0m[0000] [SimpleConfig] Hostnetwork selected - disabling injection of docker host into the cluster, server load balancer and setting the api
 port to the k3s default
[33mWARN[0m[0000] No node filter specified
[33mWARN[0m[0000] No node filter specified
[33mWARN[0m[0000] No node filter specified
[36mINFO[0m[0000] Prep: Network
```

```
[36mINFO[0m[0000] Re-using existing network 'host' (8bebb4ae61001f74487d0aa6b315396405d0127c938da1206614d113295ae139)
[36mINFO[0m[0000] Created volume 'k3d-cc-images'
[36mINFO[0m[0000] Starting new tools node...
[36mINFO[0m[0000] Starting Node 'k3d-cc-tools'
[36mINFO[0m[0001] Creating node 'k3d-cc-server-0'
[36mINFO[0m[0001] Using the k3d-tools node to gather environment information
[36mINFO[0m[0001] Starting cluster 'cc'
[36mINFO[0m[0001] Starting servers...
[36mINFO[0m[0001] Starting Node 'k3d-cc-server-0'
[36mINFO[0m[0033] All agents already running.
[36mINFO[0m[0033] All helpers already running.
[36mINFO[0m[0033] Cluster 'cc' created successfully!
[36mINFO[0m[0034] You can now use it like this:
kubectl cluster-info
Cluster setup is completed. Will start the deployment shortly...
Importing the required images...
[36mINFO[0m[0000] Importing image(s) into cluster 'cc'
[36mINFO[0m[0000] Importing images from 1 tarball(s)...
[36mINFO[0m[0000] Importing images '[/home/appviewx/CCTEST/deps/tools/mid-server-docker-image/avx-mid-server-base-22.1.0.0.tar]' into node
 'k3d-cc-server-0'...
[36mINFO[0m[0024] Successfully imported image(s)
[36mINFO[0m[0024] Successfully imported 1 image(s) into 1 cluster(s)
Import in progress...
[36mINFO[0m[0000] Importing image(s) into cluster 'cc'
[36mINFO[0m[0000] Importing images from 1 tarball(s)...
[36mINFO[0m[0000] Importing images '[/home/appviewx/CCTEST/deps/tools/mid-server-docker-image/k3d-tools-5.2.2.tar]' into node 'k3d-cc-server-0'...
[36mINFO[0m[0005] Successfully imported image(s)
[36mINFO[0m[0005] Successfully imported 1 image(s) into 1 cluster(s)
Import in progress...
[36mINFO[0m[0000] Importing image(s) into cluster 'cc'
[36mINFO[0m[0000] Importing images from 1 tarball(s)...
[36mINFO[0m[0000] Importing images '[/home/appviewx/CCTEST/deps/tools/mid-server-docker-image/rancher-mirrored-coredns-coredns-1.8.6.tar]' into
 node 'k3d-cc-server-0'...
[36mINFO[0m[0007] Successfully imported image(s)
[36mINFO[0m[0007] Successfully imported 1 image(s) into 1 cluster(s)
[36mINFO[0m[0000] Importing image(s) into cluster 'cc'
[36mINFO[0m[0000] Importing images from 1 tarball(s)...
```

```
[36mINFO[0m[0000] Importing images '[/home/appviewx/CCTEST/deps/tools/mid-server-docker-image/rancher-local-path-provisioner-v0.0.21.tar]' into
 node 'k3d-cc-server-0'...
[36mINFO[0m[0004] Successfully imported image(s)
[36mINFO[0m[0004] Successfully imported 1 image(s) into 1 cluster(s)
[36mINFO[0m[0000] Importing image(s) into cluster 'cc'
[36mINFO[0m[0000] Importing images from 1 tarball(s)...
[36mINFO[0m[0000] Importing images '[/home/appviewx/CCTEST/deps/tools/mid-server-docker-image/rancher-mirrored-pause-3.6.tar]' into node
 'k3d-cc-server-0'...
[36mINFO[0m[0003] Successfully imported image(s)
[36mINFO[0m[0003] Successfully imported 1 image(s) into 1 cluster(s)
Deploying the Cloud Connector...
NAME: avx-mid-server-starter
LAST DEPLOYED: Mon May 30 15:51:13 2022
NAMESPACE: cc
STATUS: deployed
REVISION: 1
NOTES:
1. It may take a couple of minutes for the Cloud Connector to be up.
  kubectl get pod --namespace cc
**************************************************************************
*    Congratulations!!! The installation completed successfully.   *
*    Please wait till the Cloud Connector is up and running.       *
**************************************************************************
(1%) Cloud Connector status: Running
[32m Cloud Connector is up and running. (B[m
```

Troubleshooting: For installation errors, refer to the Troubleshooting section.

The AppViewX Cloud Connector consists of two important components—the starter plugin and the platform. The starter plugin component is installed along with the AppViewX Cloud Connector, in the same installation process.

When installed, the starter plugin is used to initiate the download of the platform component. The platform component is used to host business use cases related to the AppViewX Cloud Connector.

When the platform component download is in progress, it is indicated by the symbol prefixed to the platform component version number in the AppViewX Cloud Connector inventory details

 . A completed download/upgrade is indicated by the  symbol in the same location  .

> **Note:** Based on the internet bandwidth and the number of cloud connectors being installed, the downloading of the cloud connector may vary between 5 to 15 minutes.

38. To approve the cloud connector installation:

    a. Go to  (**Menu**) > **Platform** > **Connectivity** > **Cloud Connector**.
    The **Settings :: Cloud Connector** inventory page is displayed.

    b. For the cloud connector instance installed, from the **Actions** column, click **Approve**.

> **Troubleshooting:** If the AppViewX Cloud Connector instance has been approved but is not moved to the **Running** state, you can check the pod status and/or restart the pod(s), as required.

Setting up the AppViewX Cloud Connector using a Virtual Image on AWS

> **Note:** For the AWS AMI, the following two operating systems are supported: **Ubuntu** and **Amazon Linux 2**.

Prerequisites

- Relay your requirements to your assigned AppViewX Solution Architect and finalize a deployment model.
- Share your AWS account number and region with your Solution Architect. The Solution Architect will use these details to create a custom AMI based on your account and region.

  When the AppViewX AMI is successfully shared with your customer account, AppViewX will notify you of this development via email.
- From your AppViewX Onboarding Engineer, get the default password for the **appviewx** user.

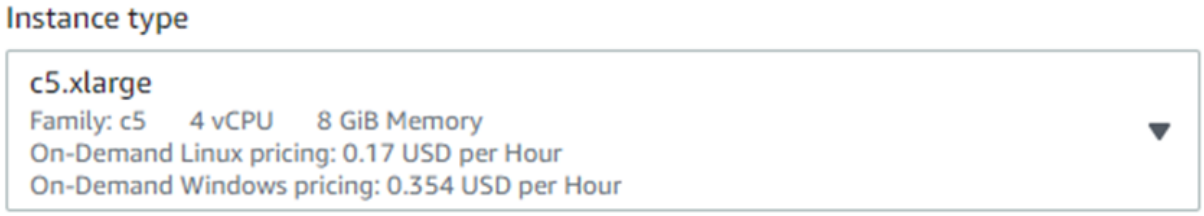
To install the AppViewX Cloud Connector on AWS, you will need a virtual machine that is preconfigured for the operating system and software stack prerequisites.AWS uses AMI to create pre-configured EC2 instances as per AppViewX standards and requirements.

To create an EC2 instance using the AppViewX AMI:

1. Login to the AWS Management Console and go to **EC2** > **Images** > **AMIs**.
   The **Amazon Machine Images (AMIs)** page is displayed.
2. On the **Amazon Machine Images (AMIs)** page, from the **Owned by me** dropdown list, select **Private images**.
   All AMIs with visibility set to private are listed. This list will also have the AMI that is created and shared by AppViewX for your requirements.
3. From this list, select the checkbox for the AMI shared by AppViewX.
4. Click **Launch instance from AMI**.
   The **EC2 > Instances > Launch an instance** page is displayed.
5. Enter the **Name and tags** to be associated with this EC2 instance.

   Note: The **Application and OS Images (Amazon Machine Images)** section will show the configuration details of the AppViewX AMI.

6. For the master node, select the following hardware configuration:

7. To securely connect to the EC2 instance, in the **Key pair (login)** section:
   a. To use an existing key pair, from the **Key pair name** dropdown list, select the key pair you want to use.

   OR

   a. To create a new key pair, Click **Create new key pair**.
8. In the **Network settings** section, under **Firewall (security groups)**, as required, create a new security group or select an existing security group.
9. If you select **Select existing security group** in the previous step, from the **Common security groups** dropdown list, select the required security group.
10. From the bottom-right corner of the screen, click **Launch instance**.

The **Launching instance** page is displayed, which shows you the progress of the launch. As soon as the launch is initiated, you will get a success message.

11. Under **Success**, click **Launch log** to review the instance details.
12. From the page name (**EC2 > Instances > Launch an instance**), click **Instances** to go back to the previous page.
13. From the list of instances, select the AWS instance just created.
14. To login to this AWS instance using the key pair .pem file:

    a. Execute one of the following commands:
    - For Ubuntu:

      `ssh -i newkey.pem ubuntu@<public ipaddress of the aws instance>`

    - For Amazon Linux 2:

      `ssh -i newkey.pem ec2-user<public ipaddress of the aws instance>`

    b. To switch to the **sudo** user, execute the following command: `sudo -i`
    After the execution of this step, if the cloud connector does not exist on this host machine, the following prompt is displayed: **Would you like to opt for automated installation of the cloud connector? (y/n):**.

    c. Since these instructions are for setting up the cloud connector via the user interface, enter **n**.

    d. To add an entry for the nameserver in the **resolv.conf** file, execute the following command: `echo "nameserver <IP of nameserver>" > /etc/resolv.conf`

    e. Update the /etc/hosts file for the IP and the hostname of the VM created, using the following commands:

    ```
    vi /etc/hosts


    hostnamectl set-hostname “hostname-of-the-vm”
    ```

    f. To validate the update to the **/etc/hosts** file, execute the following commands:

    ```
    hostname -i

    hostname -f

    hostname
    ```

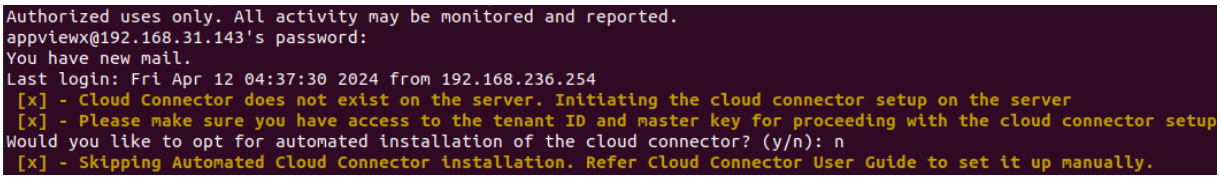
    g. To switch to the **appviewx** user, execute the following command: `sudo su – appviewx`

h. {Optional, required only for password authentication) In order to successfully execute the installation, AppViewX needs to run a script for which authentication via the **.pem** file needs to be bypassed. To do this, execute the following commands:

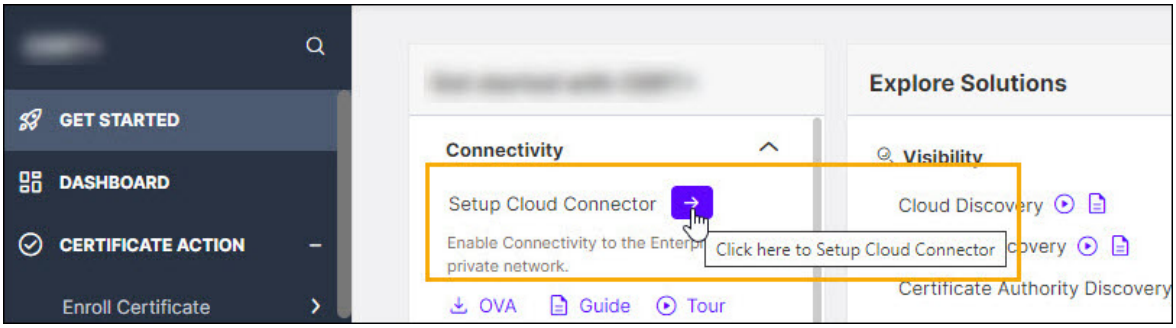
```
sudo sed -i 's/.*PasswordAuthentication.*/PasswordAuthentication yes/g' /etc/ssh/sshd_config

sudo systemctl restart sshd
```

After the execution of this step, a script is executed to validate if the cloud connector exists on this host machine and the following prompt is displayed: **Would you like to opt for automated installation of the cloud connector? (y/n):**.

15. Since these instructions are for setting up the cloud connector via the user interface, enter **n**.

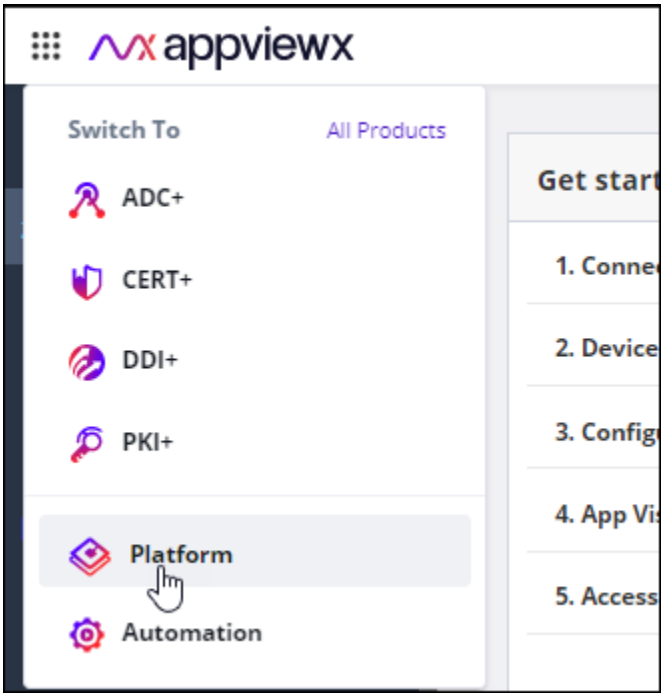


16. Enter your account URL (for example, https://tenant-name.appvx.com/appviewx/login) in the address bar of your browser.
    The AppViewX login page is displayed.
17. Login to AppViewX.
18. Navigate to the cloud connector's setup interface.
    There are three ways you can access the interface for setting up the AppViewX Cloud Connector:
    - From the product landing page (that you will see as soon as you have logged in)
      - Expand the **Connectivity** section and click  .

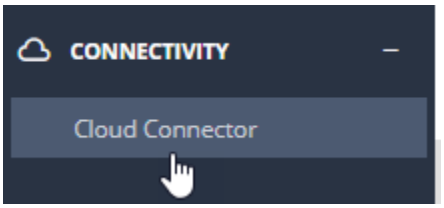


      You will be redirected to the **Settings :: Cloud Connector** page.
    - From the new navigation menu (displayed by default starting product version 2022.1.0 FP3 onwards):

a. From the menu in the top-right corner of the page, select **Platform**.



b. From the **Platform** menu, under **Connectivity**, click **Cloud Connector**.



The **Settings :: Cloud Connector** page is displayed.

• From the old navigation menu:

> **Note:** For instructions on switching between the new and the old navigation menus, click here.

a. From the top right corner of the landing page, click the menu icon.

b. From the menu displayed, navigate to **Settings** > **Cloud Connector**.

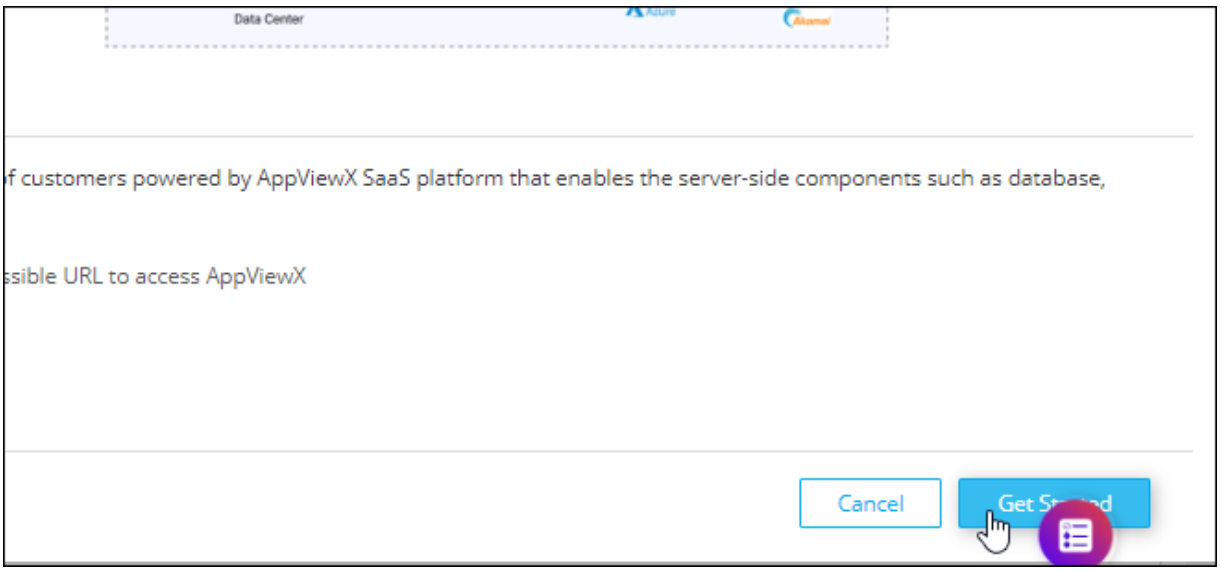
The **Settings :: Cloud Connector** page is displayed.

19. On the **Setting :: Cloud Connector** page, click **Add Cloud Connector**.

The **Cloud Connector Setup** screen is displayed.

The landing page gives you a quick introduction to the AppViewX Cloud Connector, with a graphical representation of how the infrastructure is deployed and works.

20. To start with the process of adding the cloud connector, from the bottom-right corner of the screen, click **Get Started**.



You will be redirected to the **Basic Information** page.

21. On the **Basic Information** page, configure the basic cloud connector settings.

    a. To install the cloud connector via the virtual image, from **Installation Type**, select **Virtual Image**.

    > **Note:** Click here to read how a virtual image-based installation is different from a native OS installation.

    b. In the **Cloud Connector Name (FQDN)** field, enter the hostname of the machine on which the AppViewX Cloud Connector will be installed.

    > **Tip:** To retrieve the hostname, from the command line terminal of the host machine, execute the following command: `hostname -f`:

    > **Note:** The hostname entered here is added to the license file that will be generated and downloaded as part of the installer. Therefore, the license file can be used to install the cloud connector only on the machine with the entered hostname and no other.

> **Tip:** The **Setup Cloud Connector** section to the right of the **Basic Information** screen lists hyperlinks to the prerequisites required for setting up the AppViewX Cloud Connector. To read more about what the AppViewX Cloud Connector offers, click **Learn More**.

c. Click **Next**.

22. [Optional] Execute a prerequisite check script.

> **Note:** This is an **optional** step. The prerequisite check script is executed automatically at the time of installing the AppViewX Cloud Connector and the results are shown as a part of the installation logs.

To simplify compliance to the AppViewX Cloud Connector installation prerequisites, you can execute a script to identify and rule out any deviations from the prerequisites.

a. On the **Basic Information** screen, under **Setup Cloud Connector**, you will see a list of the installation prerequisites.

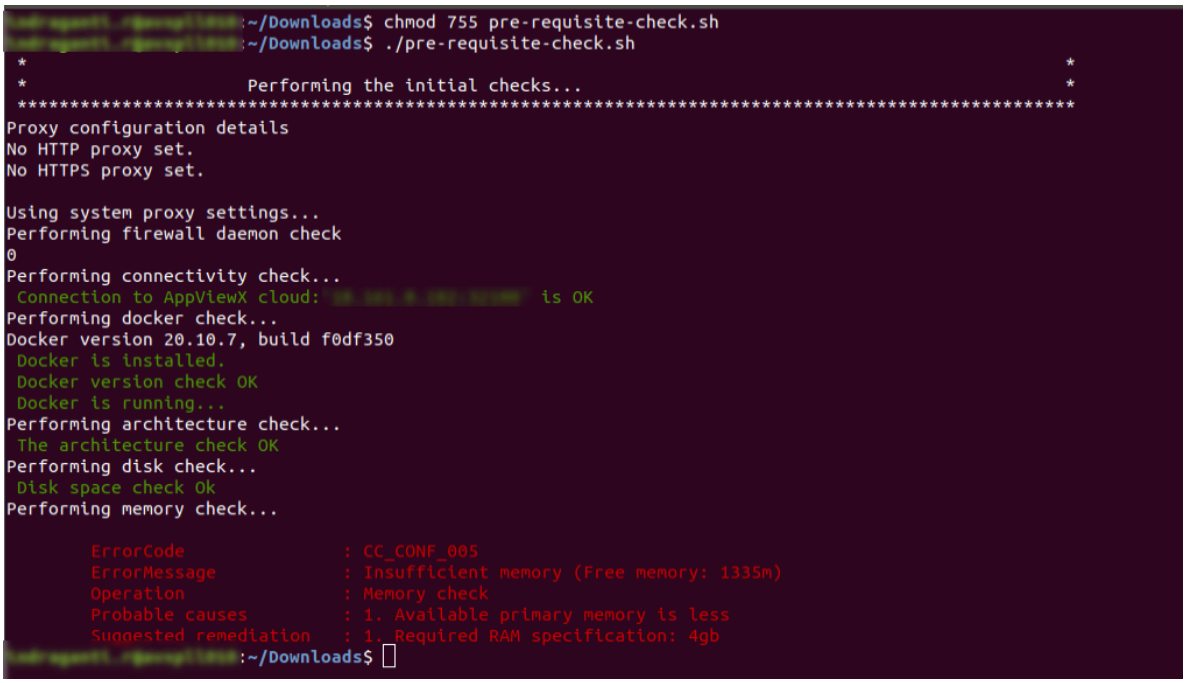
   From this list, for **Executing the Prerequisites Check Script**, to download the script, click . The **pre-requisite-check.sh** script file is downloaded.

b. Securely copy the **pre-requisite-check.sh** via SCP/SFTP to the host machine where the AppViewX Cloud Connector is to be installed

c. Convert the downloaded script file into an executable file using the chmod command, as shown below:`chmod 755 pre-requisite-check.sh`

d. Execute the **.sh** prerequisite check script file: `./pre-requisite-check.sh`
   If the node does not meet the prerequisites for the AppViewX Cloud Connector installation, the output of the command returns an error code and the corresponding error message, causes, and fixes, if any.

   For example, as seen in the sample output in the image below, the prerequisite check for the memory requirement has failed.



> **Note:** For resolutions to the prerequisite check failure scenarios, click here.

23. Click **Next**.
    You will be navigated to the **AssignData Center** screen, where, for deploying the AppViewX Cloud Connector, you can either select an existing data center or add a new one.
24. To use an existing data center, select one from the options displayed on the **Assign Data Center** screen.

> **Tip:** Alternatively, you can use the **Search…** field on this screen to search for an existing data center.

To add a new data center:

a. Click **Add Data Center.**

b. In the **Add Data Center** dialog box, enter a name for the new data center.

c. Click **Save**.
   The new data center will now be displayed on the **Assign Data Center** screen along with the other existing data centers.

d. Select the required data center.

> **Tip:** The **Data Center based routing** section to the right of the **Assign Data Center** screen explains the concept of data center-based routing and how you can achieve high availability. To read more on this, click **Learn More** from the top-right corner of this screen.

25. Click **Next**.
    The **Advanced Configuration** screen is displayed.
26. On the **Advanced Configuration** page, to configure the TLS authentication and proxy server settings for your cloud connector:

    a. Enter/Select the advanced configuration settings for the AppViewX Cloud Connector.

> **Note:** The **Data center** field is auto-populated based on your selection on the **Assigning a Data Center** screen.

Field	Description
TLS Authentication	**Tip:** The section on the right of the screen gives you a brief context of what is **TLS Authentication**. To read more, click **Learn More** (next to the **TLS Authentication** heading). • To auto-generate a TLS certificate, select **Auto-generate** (default selection). By default, the certificate is generated using the AppViewX CA. **Note:** The created certificate is available in the certificate inventory. You can: • Assign this certificate to a certificate group • Configure a certificate expiry alert for this certificate group from the **Server Certificate** dashboard, using the **Certificate Summary Report** widget settings • To enter details of a custom TLS certificate, select **Custom**. The **TLS Certificate Password** and **Custom TLS Certificate** fields are displayed. The instructions for filling these fields are given below.

Field	Description
TLS Certificate Password*	**Note:** This field is displayed only if you have selected to enter details of a Custom TLS certificate in the **TLS Authentication** field. Password of the TLS certificate (that will be uploaded in the next step) **Note:** This is a mandatory field if a Custom TLS certificate is uploaded. AppViewX supports only password-protected Custom TLS certificates.
TLS Certificate	**Note:** This field is displayed only if you have selected to enter details of a Custom TLS certificate in the **TLS Authentication** field. To upload a custom TLS certificate: i. To navigate to the location of the custom TLS certificate, click within the field. ii. Select the certificate file. iii. Click **Open**. iv. To upload the custom TLS certificate selected, click **Upload**. **Note:** AppViewX supports only password-protected Custom TLS Certificates.
Use proxy	**Tip:** The section on the right of the screen gives you a brief context of what is **Proxy based routing**. To read more, click **Learn More** (next to the **Proxy based routing** heading). A proxy server is required if the AppViewX Cloud Connector is unable to connect to your endpoints available in the internet. To use a proxy server for the deployment:

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>
i. Select the Use proxy checkbox.

ii. To select a preconfigured proxy (for the selected data center), from the Select Proxy dropdown list, select a proxy server.

OR

To create a new proxy server setting:

i. Use the Click here option shown below the Select Proxy dropdown list.

The Add Proxy pop-up screen is displayed.

ii. Enter/Select the details required to add a proxy.

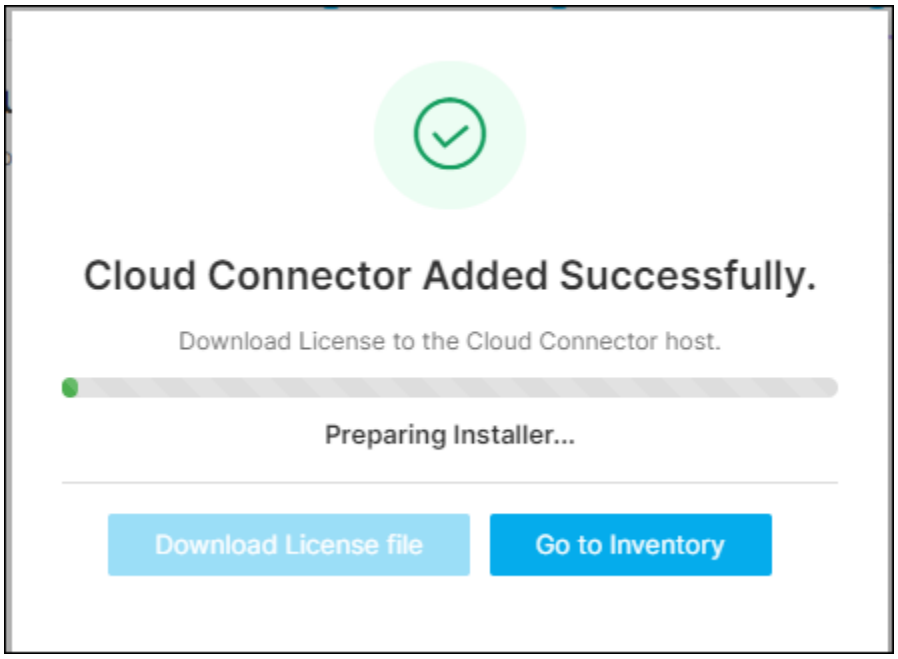
Field descriptions for the Add Proxy details
<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>*Proxy Name</td><td>Name of the proxy server</td></tr>
<tr><td>*Server IP</td><td>IP address/FQDN of the proxy server</td></tr>
<tr><td>*Port</td><td>Port number of the proxy server</td></tr>
<tr><td>URL</td><td>From the dropdown menu, select the URL.</td></tr>
<tr><td>Authentication</td><td>To enable authentication for accessing the proxy server, select this checkbox.</td></tr>
<tr><td>*Username</td><td>Note: This field is enabled only when Authentication is selected.

Enter the username required for accessing the proxy server.</td></tr>
</table>
</td></tr>
</table>

Field	Description
	<table><tr><th>Field</th><th>Description</th></tr><tr><td>***Password**</td><td>**Note:** This field is enabled only when **Authentication** is selected. Enter the password required for accessing the proxy server.</td></tr></table>

b. Click **Finish**.

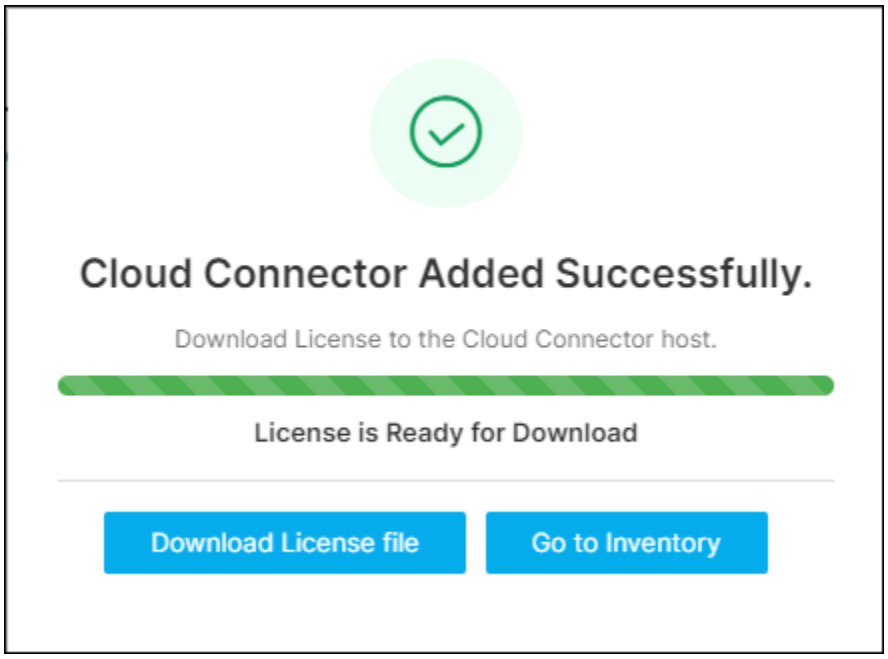
A confirmation message is displayed. AppViewX begins preparing the installer and the license file. Once the license file is ready, you can download it and proceed with the installation of the AppViewX Cloud Connector.



27. Download the license file.

Note: The installer is prepackaged with the OVA, so, for a virtual image-based installation, you only need to download the license file.

a. On the **Cloud Connector Added Successfully** dialog box, when the **License is Ready for Download**, click **Download License file**.



Tip: At this point, if the installer has been deleted or is not usable, and you wish to revert to a native installation, click **Go to Inventory**. It will take you back to the cloud connector inventory, from where you can download the license file and installer for the native OS download.

Tip: You can also choose to download the license file and the installer package individually. To do this:

i. Click the **Cloud Connector Name**.

   The selected Cloud Connector's details are shown in a pane to your right.

ii. To download the AppViewX Cloud Connector installer package, click **Download Cloud Connector**. This is useful in the event that the installer has been deleted or is no longer usable.

   To download the license file, click **Download License**.

Note: A installer download is made available even for a virtual-image based deployment, to help you with reconfiguration in case the existing OVA configuration is deleted.

b. Save the license file on the OVA node.

On the **Settings :: Cloud Connector** page, details of this AppViewX Cloud Connector are added in the inventory table, which is explained here.

28. Install the AppViewX Cloud Connector Agent.

> **Note:** The following steps assume that:
> • All system prerequisites are fulfilled by the host machine.
> • The AppViewX Cloud Connector installer (downloaded in the above step) is securely copied via SCP/SFTP to the host machine where the AppViewX Cloud Connector is to be installed.

a. On the node where the AppViewX Cloud Connector agent will be installed, from the extracted installation package, run the **./install.sh** script.

The script will check if the installation prerequisites for the AppViewX Cloud Connector have been fulfilled.

> **Note:** Ensure that the license file is placed in the same location as the **install.sh** script. If the license file is placed in another location, run the install.sh script using the following command:
>
> ```
> ./install.sh <complete path of the license file with the filename>
> ```

On successful verification of the prerequisites, you will be prompted to specify if you want to manage f5 BIG-IP devices and if you need auto-enrollment of the certificates.

```
Do you want to manage f5 BIG-IP devices? (y/n)?:n
Continuing with the installation

Do you need Auto-enrollment of the certificate using EST/SCEP/ACME? (y/n)?:y
Please choose one or more protocol (use comma separated numbers): 1)EST(MTLS)   2)SCEP(HTTP)  3)ACME(HTTPS)
1,2,3
Auto enrollment enabled successfully for protocol(s): MTLS HTTP HTTPS
Do you want to enable Syslog receiver for a near real time configuration updates from the devices. (y/n) n
syslog enabled  n
```

b. When prompted, enter the required input value(s):

> **Important:** If you choose to **not enable** any of the following features, to enable them later, you will have to reinstall the AppViewX Cloud Connector.

i. If you want manage f5 BIG-IP devices, enter **y**/**n** for yes/no, respectively.

ii. When prompted to enable auto-enrollment of the certificate using one of the following supported auto-enrollment protocols, enter **y** only if the cloud connector is being installed in a demilitarized zone (DMZ) or devices in a restricted environment (that disables them from connecting to the **<tenant>-aep** directly.

• If you choose **y** (yes) here, enter the required protocol(s) name.

> **Note:** By default, the AppViewX certificate is enabled for auto-enrollment. To enable custom certificate for auto-enrollment:
>
> 1. Execute the command `./avxctl upgrade gateway-cert`.
> 2. When prompted, enter the location of the custom certificate.

> **Note:** If you are a KUBE+ customer, the auto-enrollment gateway should be enabled as part of the installation for your KUBE+ use cases to work via the cloud connector.

iii. If you want to enable Syslog receiver for a near-real time configuration updates from the devices, enter **y/n** for yes/no, respectively. For **configuring Syslog reception**, refer to the Platform User guide section, Syslog Reception.

In case you have an older version of AppViewX on cloud and want to make use of Syslog capabilities for ADC, you must manually activate the Syslog flag by setting **SYSLOG_ENABLED**=**true** in the path **ccpath/deps/properties**.

c. Enter the sudo password.

After the relevant details have been entered, the installation proceeds. Installation logs, according to the outcome of the installation, are displayed.

Given below are sample installation logs:

```
Loaded image: rancher/k3s:v1.23.3-k3s1
Loaded image: rancher/k3d-tools:5.2.2
Loaded image: rancher/mirrored-pause:3.6
[36mINFO[0m[0000] [SimpleConfig] Hostnetwork selected - disabling injection of docker host into the cluster, server load balancer and setting the api
 port to the k3s default
[33mWARN[0m[0000] No node filter specified
[33mWARN[0m[0000] No node filter specified
[33mWARN[0m[0000] No node filter specified
[36mINFO[0m[0000] Prep: Network
```

```
[36mINFO[0m[0000] Re-using existing network 'host' (8bebb4ae61001f74487d0aa6b315396405d0127c938da1206614d113295ae139)
[36mINFO[0m[0000] Created volume 'k3d-cc-images'
[36mINFO[0m[0000] Starting new tools node...
[36mINFO[0m[0000] Starting Node 'k3d-cc-tools'
[36mINFO[0m[0001] Creating node 'k3d-cc-server-0'
[36mINFO[0m[0001] Using the k3d-tools node to gather environment information
[36mINFO[0m[0001] Starting cluster 'cc'
[36mINFO[0m[0001] Starting servers...
[36mINFO[0m[0001] Starting Node 'k3d-cc-server-0'
[36mINFO[0m[0033] All agents already running.
[36mINFO[0m[0033] All helpers already running.
[36mINFO[0m[0033] Cluster 'cc' created successfully!
[36mINFO[0m[0034] You can now use it like this:
kubectl cluster-info
Cluster setup is completed. Will start the deployment shortly...
Importing the required images...
[36mINFO[0m[0000] Importing image(s) into cluster 'cc'
[36mINFO[0m[0000] Importing images from 1 tarball(s)...
[36mINFO[0m[0000] Importing images '[/home/appviewx/CCTEST/deps/tools/mid-server-docker-image/avx-mid-server-base-22.1.0.0.tar]' into node
 'k3d-cc-server-0'...
[36mINFO[0m[0024] Successfully imported image(s)
[36mINFO[0m[0024] Successfully imported 1 image(s) into 1 cluster(s)
Import in progress...
[36mINFO[0m[0000] Importing image(s) into cluster 'cc'
[36mINFO[0m[0000] Importing images from 1 tarball(s)...
[36mINFO[0m[0000] Importing images '[/home/appviewx/CCTEST/deps/tools/mid-server-docker-image/k3d-tools-5.2.2.tar]' into node 'k3d-cc-server-0'...
[36mINFO[0m[0005] Successfully imported image(s)
[36mINFO[0m[0005] Successfully imported 1 image(s) into 1 cluster(s)
Import in progress...
[36mINFO[0m[0000] Importing image(s) into cluster 'cc'
[36mINFO[0m[0000] Importing images from 1 tarball(s)...
[36mINFO[0m[0000] Importing images '[/home/appviewx/CCTEST/deps/tools/mid-server-docker-image/rancher-mirrored-coredns-coredns-1.8.6.tar]' into
 node 'k3d-cc-server-0'...
[36mINFO[0m[0007] Successfully imported image(s)
[36mINFO[0m[0007] Successfully imported 1 image(s) into 1 cluster(s)
[36mINFO[0m[0000] Importing image(s) into cluster 'cc'
[36mINFO[0m[0000] Importing images from 1 tarball(s)...
```

```
[36mINFO[0m[0000] Importing images '[/home/appviewx/CCTEST/deps/tools/mid-server-docker-image/rancher-local-path-provisioner-v0.0.21.tar]' into
 node 'k3d-cc-server-0'...
[36mINFO[0m[0004] Successfully imported image(s)
[36mINFO[0m[0004] Successfully imported 1 image(s) into 1 cluster(s)
[36mINFO[0m[0000] Importing image(s) into cluster 'cc'
[36mINFO[0m[0000] Importing images from 1 tarball(s)...
[36mINFO[0m[0000] Importing images '[/home/appviewx/CCTEST/deps/tools/mid-server-docker-image/rancher-mirrored-pause-3.6.tar]' into node
 'k3d-cc-server-0'...
[36mINFO[0m[0003] Successfully imported image(s)
[36mINFO[0m[0003] Successfully imported 1 image(s) into 1 cluster(s)
Deploying the Cloud Connector...
NAME: avx-mid-server-starter
LAST DEPLOYED: Mon May 30 15:51:13 2022
NAMESPACE: cc
STATUS: deployed
REVISION: 1
NOTES:
1. It may take a couple of minutes for the Cloud Connector to be up.
  kubectl get pod --namespace cc
**************************************************************************
*    Congratulations!!! The installation completed successfully.   *
*    Please wait till the Cloud Connector is up and running.       *
**************************************************************************
(1%) Cloud Connector status: Running
[32m Cloud Connector is up and running. (B[m
```

Troubleshooting: For installation errors, refer to the Troubleshooting section.

The AppViewX Cloud Connector consists of two important components—the starter plugin and the platform. The starter plugin component is installed along with the AppViewX Cloud Connector, in the same installation process.

When installed, the starter plugin is used to initiate the download of the platform component. The platform component is used to host business use cases related to the AppViewX Cloud Connector.

When the platform component download is in progress, it is indicated by the  symbol prefixed to the platform component version number in the AppViewX Cloud Connector inventory details

 . A completed download/upgrade is indicated by the  symbol in the same location  .

> **Note:** Based on the internet bandwidth and the number of cloud connectors being installed, the downloading of the cloud connector may vary between 5 to 15 minutes.

29. To approve the cloud connector installation:

    a. Go to  (**Menu**) > **Platform** > **Connectivity** > **Cloud Connector**.
    The **Settings :: Cloud Connector** inventory page is displayed.

    b. For the cloud connector instance installed, from the **Actions** column, click **Approve**.

> **Troubleshooting:** If the AppViewX Cloud Connector instance has been approved but is not moved to the **Running** state, you can check the pod status and/or restart the pod(s), as required.

Setting up the AppViewX Cloud Connector using a Virtual Image on Azure

To install the AppViewX Cloud Connector in Azure, you need to create a virtual machine (VM) using the Azure Virtual Hard Disk (VHD). Microsoft Azure uses the Azure VHD file format to store the virtual machine (VM) disk images that are containers preloaded with the operating system, network, applications, and data requirements for setting up a virtual machine.

To deploy the AppViewX virtual machine for Azure:

1. Go to https://release.appviewx.com/Login and, from **Overview**, navigate to **2022.1.0**.
2. Scroll down to Production Images and download the latest artifact of Azure CC VHD, **AppViewX-2022.1.3-FP3-CC-Ubuntu-Azure-ddmmmyyy-vhd.tar.gz**.
3. Scroll down to Production Images and download the latest artifact of Azure CC VHD, **AppViewX-2023.1.2.10-CC-Ubuntu-Azure-ddmmmyyy-vhd.tar.gz**.
4. Untar the downloaded artifact.
   ```
   tar -xvf AppViewX-2023.1.2.10-CC-Ubuntu-Azure-ddmmmyyy-vhd.tar.gz
   ```
5. Download the **Azure Storage Explorer** from here.
   The **Azure Storage Explorer** is a desktop application that provides you with a GUI for easily managing your Azure resources.

Important: Install the Azure Storage Explorer at the same location as the downloaded Azure CC VHD artifact.

6. Using the Azure Storage Explorer, login to the Azure account for which the VM has to be created.
7. On successful login, go to the **disks** section and select the resource group.
   The resource group page is displayed.
8. Click **Upload**.
9. In the pop-up window displayed, enter/select the resource details.

Descriptions for the resources and their corresponding values

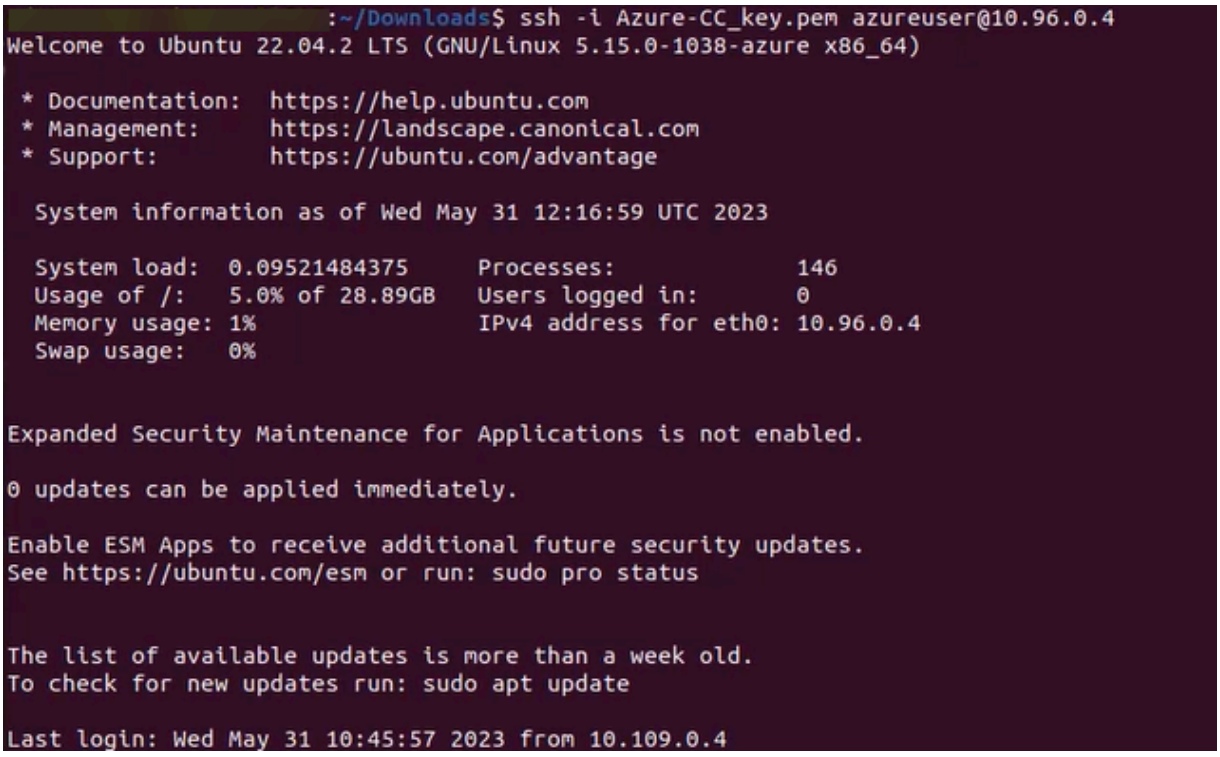
Resource	Value
Source VHD	<Disk file location>
Disk name	<Name of the disk>
OS type	**Linux**
Location	<region in which the VM is to be created>
Availability Zone	<zone name>
Account type	**Premium SSD**
Hyper-V Generation	**V1**
Architecture	**x64**

Note: All the values in bold, in the above table, are actual values and have to be assigned as is; values enclosed in angle brackets (**<>**) have to be assigned as per your specific configuration.

10. Click **Create**.
11. Once the disk is successfully uploaded to the Azure cloud, from the Azure portal, select the disk and click **+Create VM**.
12. On the **Create a virtual machine** page, configure the VM configurations based on your organization's standards and requirements.
13. Once the VM is successfully created, use the **.pem** file shared by AppViewX's SRE/TS team to login to the node (on which the cloud connector is to be installed). To do this, on the Command Line Terminal or Powershell, execute the following command:

`ssh -i Azure-CC_key.pem azureuser@<ip_of_the_CC_VM>`

Output:



14. To switch to the root user, execute the following command: `sudo -i`

    OR

    To switch to the appviewx user, execute the following command:

    `su - appviewx`

    OR

    [Optional] To enable login without the **.pem** file:

    a. Enable password authentication in the Azure node in which the AppViewX Cloud Connector is installed by executing the following commands:

    ```
    sudo sed -i 's/^\(PasswordAuthentication \).*/\1yes/' /etc/ssh/sshd_config

    sudo sed -i 's/^\(PasswordAuthentication \).*/\1yes/' /etc/ssh/sshd_config.d/50-cloud-init.conf
    ```

    b. Enable root password authentication by executing the following commands:

```
sudo sed -i '/^#PermitRootLogin/s/^#//' /etc/ssh/sshd_config

sudo sed -i 's/^PermitRootLogin.*/PermitRootLogin yes/' /etc/ssh/sshd_config
```

c. The SSH (Secure Shell) protocol is used for secure administration (login, command execution) over remote networks. For SSH changes to take effect, restart the SSH service. To do this, execute the following command: `sudo service ssh restart`

OR

To login without the **.pem** file, execute the following commands:

```
ssh appviewx@<ip/hostname>

ssh root@<ip/hostname>
```

15. Update the **/etc/hosts** file for the IP and the hostname of the VM created, using the following commands:

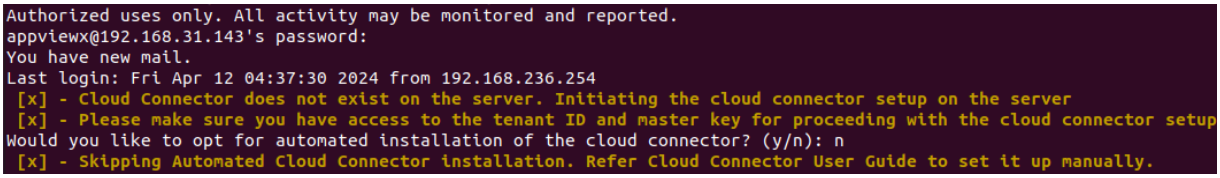
```
vi /etc/hosts

hostnamectl set-hostname “hostname-of-the-vm”
```

16. To validate the update to the **/etc/hosts** file, execute the following commands:

```
hostname -i

hostname -f

hostname
```

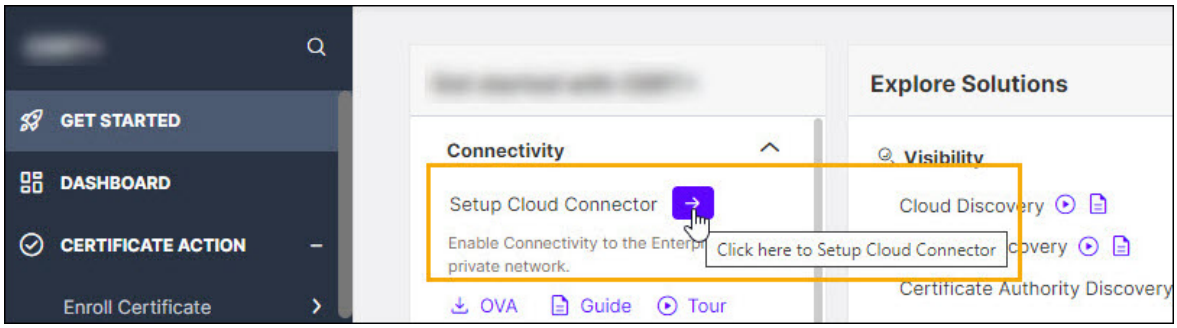
After the execution of this step, if the cloud connector does not exist on this host machine, the following prompt is displayed: **Would you like to opt for automated installation of the cloud connector? (y/n):**.

17. Since these instructions are for setting up the cloud connector via the user interface, enter **n**.



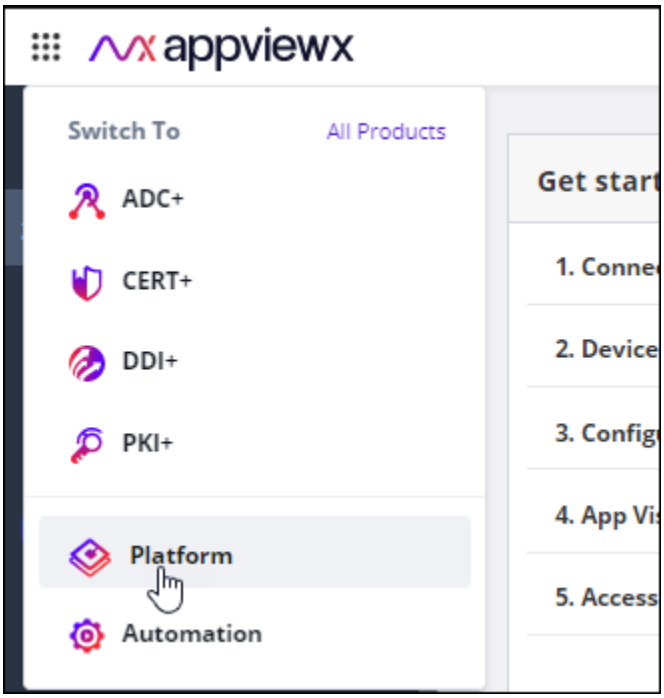
18. Enter your account URL (for example, https://tenant-name.appvx.com/appviewx/login) in the address bar of your browser.
    The AppViewX login page is displayed.
19. Login to AppViewX.
20. Navigate to the cloud connector's setup interface.
    There are three ways you can access the interface for setting up the AppViewX Cloud Connector:

- From the product landing page (that you will see as soon as you have logged in)
  - Expand the **Connectivity** section and click  .

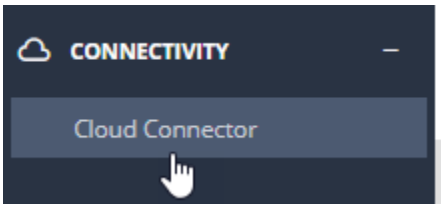


You will be redirected to the **Settings :: Cloud Connector** page.

- From the new navigation menu (displayed by default starting product version 2022.1.0 FP3 onwards):
  a. From the menu in the top-right corner of the page, select **Platform**.



  b. From the **Platform** menu, under **Connectivity**, click **Cloud Connector**.



The **Settings :: Cloud Connector** page is displayed.

- From the old navigation menu:

> **Note:** For instructions on switching between the new and the old navigation menus, click here.

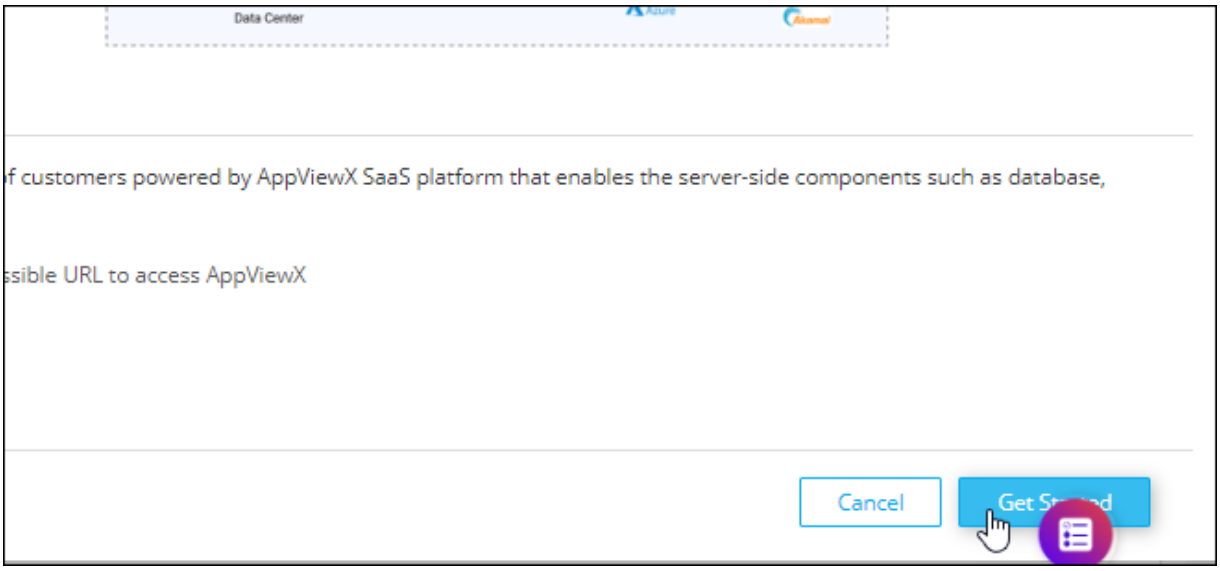
a. From the top right corner of the landing page, click the menu icon.
b. From the menu displayed, navigate to **Settings** > **Cloud Connector**.

The **Settings :: Cloud Connector** page is displayed.

21. On the **Setting :: Cloud Connector** page, click **Add Cloud Connector**.
The **Cloud Connector Setup** screen is displayed.

The landing page gives you a quick introduction to the AppViewX Cloud Connector, with a graphical representation of how the infrastructure is deployed and works.

22. To start with the process of adding the cloud connector, from the bottom-right corner of the screen, click **Get Started**.



You will be redirected to the **Basic Information** page.

23. On the **Basic Information** page, configure the basic cloud connector settings.

a. To install the cloud connector via the virtual image, from **Installation Type**, select **Virtual Image**.

> **Note:** Click here to read how a virtual image-based installation is different from a native OS installation.

b. In the **Cloud Connector Name (FQDN)** field, enter the hostname of the machine on which the AppViewX Cloud Connector will be installed.

> **Tip:** To retrieve the hostname, from the command line terminal of the host machine, execute the following command: `hostname -f`:

> **Note:** The hostname entered here is added to the license file that will be generated and downloaded as part of the installer. Therefore, the license file can be used to install the cloud connector only on the machine with the entered hostname and no other.

> **Tip:** The **Setup Cloud Connector** section to the right of the **Basic Information** screen lists hyperlinks to the prerequisites required for setting up the AppViewX Cloud Connector. To read more about what the AppViewX Cloud Connector offers, click **Learn More**.

c. Click **Next**.

24. [Optional] Execute a prerequisite check script.

> **Note:** This is an **optional** step. The prerequisite check script is executed automatically at the time of installing the AppViewX Cloud Connector and the results are shown as a part of the installation logs.

To simplify compliance to the AppViewX Cloud Connector installation prerequisites, you can execute a script to identify and rule out any deviations from the prerequisites.

a. On the **Basic Information** screen, under **Setup Cloud Connector**, you will see a list of the installation prerequisites.

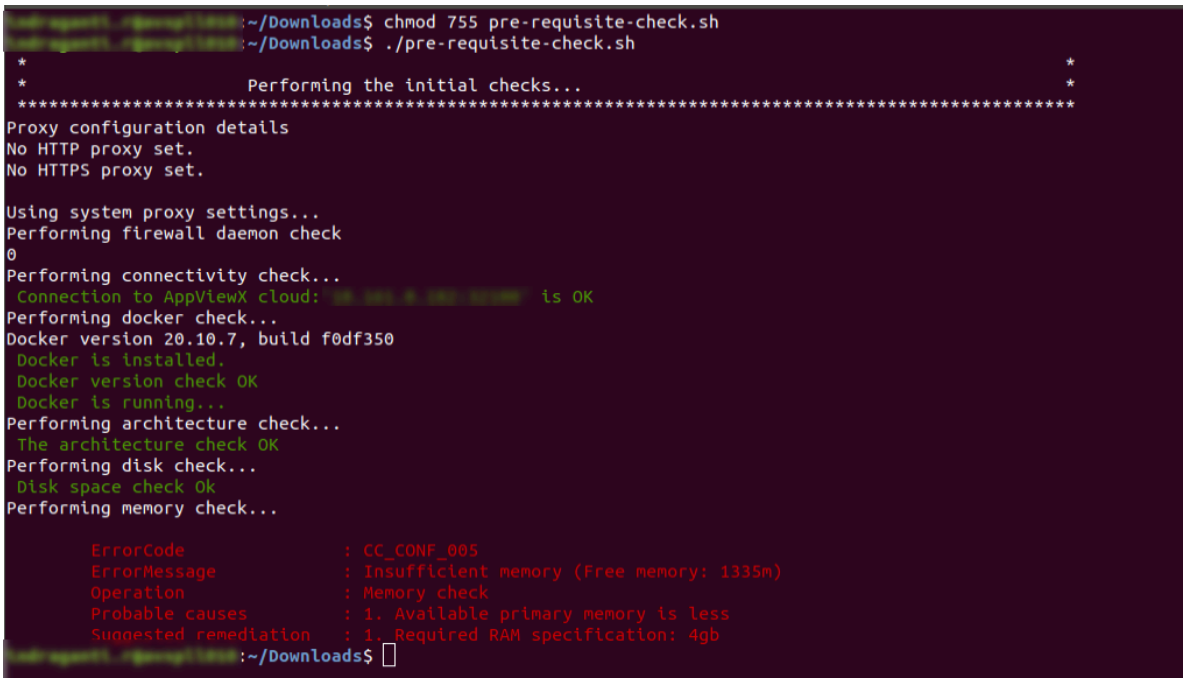
From this list, for **Executing the Prerequisites Check Script**, to download the script, click . The **pre-requisite-check.sh** script file is downloaded.

b. Securely copy the **pre-requisite-check.sh** via SCP/SFTP to the host machine where the AppViewX Cloud Connector is to be installed

c. Convert the downloaded script file into an executable file using the chmod command, as shown below:`chmod 755 pre-requisite-check.sh`

d. Execute the **.sh** prerequisite check script file: `./pre-requisite-check.sh`
If the node does not meet the prerequisites for the AppViewX Cloud Connector installation, the output of the command returns an error code and the corresponding error message, causes, and fixes, if any.

For example, as seen in the sample output in the image below, the prerequisite check for the memory requirement has failed.



Note: For resolutions to the prerequisite check failure scenarios, click here.

25. Click **Next**.
You will be navigated to the **AssignData Center** screen, where, for deploying the AppViewX Cloud Connector, you can either select an existing data center or add a new one.
26. To use an existing data center, select one from the options displayed on the **Assign Data Center** screen.

> **Tip:** Alternatively, you can use the **Search…** field on this screen to search for an existing data center.

To add a new data center:

a. Click **Add Data Center.**

b. In the **Add Data Center** dialog box, enter a name for the new data center.

c. Click **Save**.
The new data center will now be displayed on the **Assign Data Center** screen along with the other existing data centers.

d. Select the required data center.

> **Tip:** The **Data Center based routing** section to the right of the **Assign Data Center** screen explains the concept of data center-based routing and how you can achieve high availability. To read more on this, click **Learn More** from the top-right corner of this screen.

27. Click **Next**.
The **Advanced Configuration** screen is displayed.
28. On the **Advanced Configuration** page, to configure the TLS authentication and proxy server settings for your cloud connector:

a. Enter/Select the advanced configuration settings for the AppViewX Cloud Connector.

> **Note:** The **Data center** field is auto-populated based on your selection on the **Assigning a Data Center** screen.

Field	Description
TLS Authentication	**Tip:** The section on the right of the screen gives you a brief context of what is **TLS Authentication**. To read more, click **Learn More** (next to the **TLS Authentication** heading). • To auto-generate a TLS certificate, select **Auto-generate** (default selection). By default, the certificate is generated using the AppViewX CA.

Field	Description
	Note: The created certificate is available in the certificate inventory. You can: • Assign this certificate to a certificate group • Configure a certificate expiry alert for this certificate group from the **Server Certificate** dashboard, using the **Certificate Summary Report** widget settings • To enter details of a custom TLS certificate, select **Custom**. The **TLS Certificate Password** and **Custom TLS Certificate** fields are displayed. The instructions for filling these fields are given below.
TLS Certificate Password*	**Note:** This field is displayed only if you have selected to enter details of a Custom TLS certificate in the **TLS Authentication** field. Password of the TLS certificate (that will be uploaded in the next step) **Note:** This is a mandatory field if a Custom TLS certificate is uploaded. AppViewX supports only password-protected Custom TLS certificates.
TLS Certificate	**Note:** This field is displayed only if you have selected to enter details of a Custom TLS certificate in the **TLS Authentication** field. To upload a custom TLS certificate: i. To navigate to the location of the custom TLS certificate, click within the field. ii. Select the certificate file. iii. Click **Open**. iv. To upload the custom TLS certificate selected, click **Upload**.

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>

Note: AppViewX supports only password-protected Custom TLS Certificates.

</td></tr>
<tr><td>Use proxy</td><td>

Tip: The section on the right of the screen gives you a brief context of what is **Proxy based routing**. To read more, click **Learn More** (next to the **Proxy based routing** heading).

A proxy server is required if the AppViewX Cloud Connector is unable to connect to your endpoints available in the internet.

To use a proxy server for the deployment:

i. Select the **Use proxy** checkbox.

ii. To select a preconfigured proxy (for the selected data center), from the **Select Proxy** dropdown list, select a proxy server.

OR

To create a new proxy server setting:

i. Use the Click here option shown below the **Select Proxy** dropdown list.

The **Add Proxy** pop-up screen is displayed.

ii. Enter/Select the details required to add a proxy.

Field descriptions for the Add Proxy details

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>*Proxy Name</td><td>Name of the proxy server</td></tr>
<tr><td>*Server IP</td><td>IP address/FQDN of the proxy server</td></tr>
<tr><td>*Port</td><td>Port number of the proxy server</td></tr>
<tr><td>URL</td><td>From the dropdown menu, select the URL.</td></tr>
</table>

</td></tr>
</table>

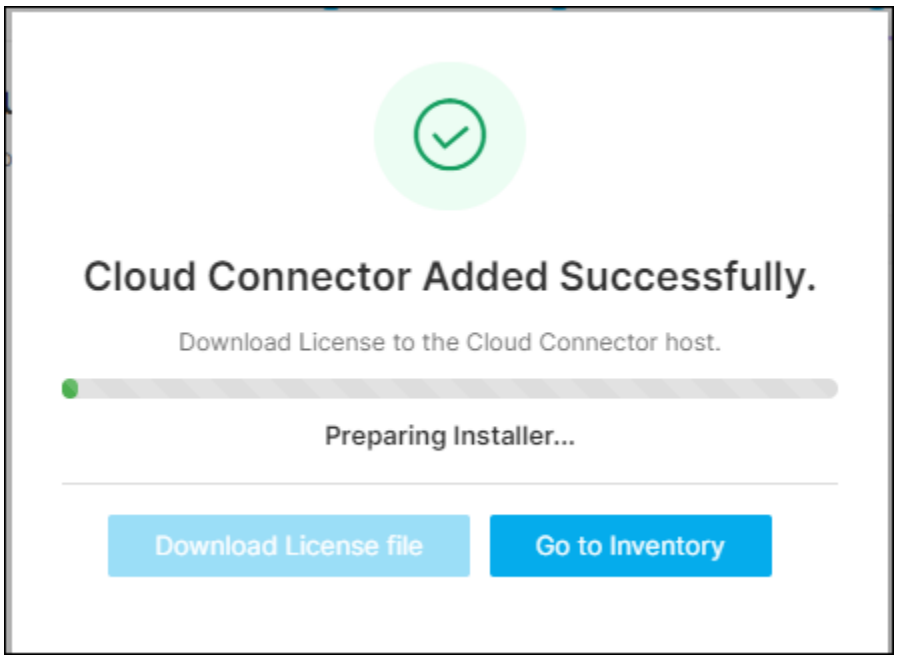
<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>Authentication</td><td>To enable authentication for accessing the proxy server, select this checkbox.</td></tr>
<tr><td>*Username</td><td>Note: This field is enabled only when Authentication is selected.

Enter the username required for accessing the proxy server.</td></tr>
<tr><td>*Password</td><td>Note: This field is enabled only when Authentication is selected.

Enter the password required for accessing the proxy server.</td></tr>
</table>
</td></tr>
</table>

b. Click **Finish**.

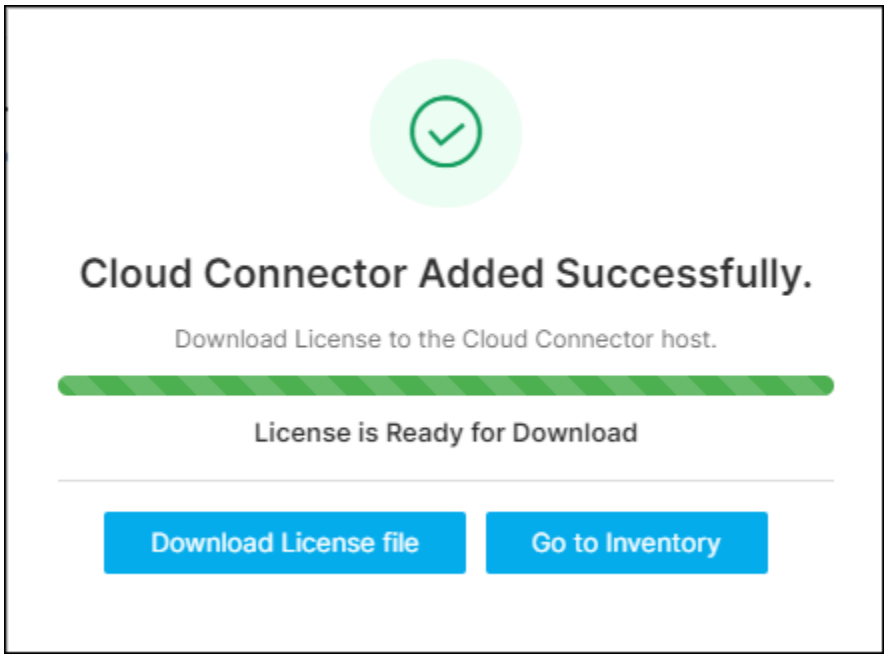
A confirmation message is displayed. AppViewX begins preparing the installer and the license file. Once the license file is ready, you can download it and proceed with the installation of the AppViewX Cloud Connector.



29. Download the license file.

Note: The installer is prepackaged with the OVA, so, for a virtual image-based installation, you only need to download the license file.

a. On the **Cloud Connector Added Successfully** dialog box, when the **License is Ready for Download**, click **Download License file**.



Tip: At this point, if the installer has been deleted or is not usable, and you wish to revert to a native installation, click **Go to Inventory**. It will take you back to the cloud connector inventory, from where you can download the license file and installer for the native OS download.

Tip: You can also choose to download the license file and the installer package individually. To do this:

i. Click the **Cloud Connector Name**.

   The selected Cloud Connector's details are shown in a pane to your right.

ii. To download the AppViewX Cloud Connector installer package, click **Download Cloud Connector**. This is useful in the event that the installer has been deleted or is no longer usable.

   To download the license file, click **Download License**.

Note: A installer download is made available even for a virtual-image based deployment, to help you with reconfiguration in case the existing OVA configuration is deleted.

b. Save the license file on the OVA node.

On the **Settings :: Cloud Connector** page, details of this AppViewX Cloud Connector are added in the inventory table, which is explained here.

30. Install the AppViewX Cloud Connector Agent.

> **Note:** The following steps assume that:
> • All system prerequisites are fulfilled by the host machine.
> • The AppViewX Cloud Connector installer (downloaded in the above step) is securely copied via SCP/SFTP to the host machine where the AppViewX Cloud Connector is to be installed.

a. On the node where the AppViewX Cloud Connector agent will be installed, from the extracted installation package, run the **./install.sh** script.

The script will check if the installation prerequisites for the AppViewX Cloud Connector have been fulfilled.

> **Note:** Ensure that the license file is placed in the same location as the **install.sh** script. If the license file is placed in another location, run the install.sh script using the following command:
>
> ```
> ./install.sh <complete path of the license file with the filename>
> ```

On successful verification of the prerequisites, you will be prompted to specify if you want to manage f5 BIG-IP devices and if you need auto-enrollment of the certificates.

```
Do you want to manage f5 BIG-IP devices? (y/n)?:n
Continuing with the installation

Do you need Auto-enrollment of the certificate using EST/SCEP/ACME? (y/n)?:y
Please choose one or more protocol (use comma separated numbers): 1)EST(MTLS)   2)SCEP(HTTP)  3)ACME(HTTPS)
1,2,3
Auto enrollment enabled successfully for protocol(s): MTLS HTTP HTTPS
Do you want to enable Syslog receiver for a near real time configuration updates from the devices. (y/n) n
syslog enabled  n
```

b. When prompted, enter the required input value(s):

> **Important:** If you choose to **not enable** any of the following features, to enable them later, you will have to reinstall the AppViewX Cloud Connector.

i. If you want manage f5 BIG-IP devices, enter **y**/**n** for yes/no, respectively.

ii. When prompted to enable auto-enrollment of the certificate using one of the following supported auto-enrollment protocols, enter **y** only if the cloud connector is being installed in a demilitarized zone (DMZ) or devices in a restricted environment (that disables them from connecting to the **<tenant>-aep** directly.

• If you choose **y** (yes) here, enter the required protocol(s) name.

> **Note:** By default, the AppViewX certificate is enabled for auto-enrollment. To enable custom certificate for auto-enrollment:
>
> 1. Execute the command `./avxctl upgrade gateway-cert`.
> 2. When prompted, enter the location of the custom certificate.

> **Note:** If you are a KUBE+ customer, the auto-enrollment gateway should be enabled as part of the installation for your KUBE+ use cases to work via the cloud connector.

iii. If you want to enable Syslog receiver for a near-real time configuration updates from the devices, enter **y/n** for yes/no, respectively. For **configuring Syslog reception**, refer to the Platform User guide section, Syslog Reception.

In case you have an older version of AppViewX on cloud and want to make use of Syslog capabilities for ADC, you must manually activate the Syslog flag by setting **SYSLOG_ENABLED**=**true** in the path **ccpath/deps/properties**.

c. Enter the sudo password.

After the relevant details have been entered, the installation proceeds. Installation logs, according to the outcome of the installation, are displayed.

Given below are sample installation logs:

```
Loaded image: rancher/k3s:v1.23.3-k3s1
Loaded image: rancher/k3d-tools:5.2.2
Loaded image: rancher/mirrored-pause:3.6
[36mINFO[0m[0000] [SimpleConfig] Hostnetwork selected - disabling injection of docker host into the cluster, server load balancer and setting the api
 port to the k3s default
[33mWARN[0m[0000] No node filter specified
[33mWARN[0m[0000] No node filter specified
[33mWARN[0m[0000] No node filter specified
[36mINFO[0m[0000] Prep: Network
```

```
[36mINFO[0m[0000] Re-using existing network 'host' (8bebb4ae61001f74487d0aa6b315396405d0127c938da1206614d113295ae139)
[36mINFO[0m[0000] Created volume 'k3d-cc-images'
[36mINFO[0m[0000] Starting new tools node...
[36mINFO[0m[0000] Starting Node 'k3d-cc-tools'
[36mINFO[0m[0001] Creating node 'k3d-cc-server-0'
[36mINFO[0m[0001] Using the k3d-tools node to gather environment information
[36mINFO[0m[0001] Starting cluster 'cc'
[36mINFO[0m[0001] Starting servers...
[36mINFO[0m[0001] Starting Node 'k3d-cc-server-0'
[36mINFO[0m[0033] All agents already running.
[36mINFO[0m[0033] All helpers already running.
[36mINFO[0m[0033] Cluster 'cc' created successfully!
[36mINFO[0m[0034] You can now use it like this:
kubectl cluster-info
Cluster setup is completed. Will start the deployment shortly...
Importing the required images...
[36mINFO[0m[0000] Importing image(s) into cluster 'cc'
[36mINFO[0m[0000] Importing images from 1 tarball(s)...
[36mINFO[0m[0000] Importing images '[/home/appviewx/CCTEST/deps/tools/mid-server-docker-image/avx-mid-server-base-22.1.0.0.tar]' into node
 'k3d-cc-server-0'...
[36mINFO[0m[0024] Successfully imported image(s)
[36mINFO[0m[0024] Successfully imported 1 image(s) into 1 cluster(s)
Import in progress...
[36mINFO[0m[0000] Importing image(s) into cluster 'cc'
[36mINFO[0m[0000] Importing images from 1 tarball(s)...
[36mINFO[0m[0000] Importing images '[/home/appviewx/CCTEST/deps/tools/mid-server-docker-image/k3d-tools-5.2.2.tar]' into node 'k3d-cc-server-0'...
[36mINFO[0m[0005] Successfully imported image(s)
[36mINFO[0m[0005] Successfully imported 1 image(s) into 1 cluster(s)
Import in progress...
[36mINFO[0m[0000] Importing image(s) into cluster 'cc'
[36mINFO[0m[0000] Importing images from 1 tarball(s)...
[36mINFO[0m[0000] Importing images '[/home/appviewx/CCTEST/deps/tools/mid-server-docker-image/rancher-mirrored-coredns-coredns-1.8.6.tar]' into
 node 'k3d-cc-server-0'...
[36mINFO[0m[0007] Successfully imported image(s)
[36mINFO[0m[0007] Successfully imported 1 image(s) into 1 cluster(s)
[36mINFO[0m[0000] Importing image(s) into cluster 'cc'
[36mINFO[0m[0000] Importing images from 1 tarball(s)...
```

```
[36mINFO[0m[0000] Importing images '[/home/appviewx/CCTEST/deps/tools/mid-server-docker-image/rancher-local-path-provisioner-v0.0.21.tar]' into
 node 'k3d-cc-server-0'...
[36mINFO[0m[0004] Successfully imported image(s)
[36mINFO[0m[0004] Successfully imported 1 image(s) into 1 cluster(s)
[36mINFO[0m[0000] Importing image(s) into cluster 'cc'
[36mINFO[0m[0000] Importing images from 1 tarball(s)...
[36mINFO[0m[0000] Importing images '[/home/appviewx/CCTEST/deps/tools/mid-server-docker-image/rancher-mirrored-pause-3.6.tar]' into node
 'k3d-cc-server-0'...
[36mINFO[0m[0003] Successfully imported image(s)
[36mINFO[0m[0003] Successfully imported 1 image(s) into 1 cluster(s)
Deploying the Cloud Connector...
NAME: avx-mid-server-starter
LAST DEPLOYED: Mon May 30 15:51:13 2022
NAMESPACE: cc
STATUS: deployed
REVISION: 1
NOTES:
1. It may take a couple of minutes for the Cloud Connector to be up.
  kubectl get pod --namespace cc
**************************************************************************
*    Congratulations!!! The installation completed successfully.   *
*    Please wait till the Cloud Connector is up and running.       *
**************************************************************************
(1%) Cloud Connector status: Running
[32m Cloud Connector is up and running. (B[m
```

Troubleshooting: For installation errors, refer to the Troubleshooting section.

The AppViewX Cloud Connector consists of two important components—the starter plugin and the platform. The starter plugin component is installed along with the AppViewX Cloud Connector, in the same installation process.

When installed, the starter plugin is used to initiate the download of the platform component. The platform component is used to host business use cases related to the AppViewX Cloud Connector.

When the platform component download is in progress, it is indicated by the  symbol prefixed to the platform component version number in the AppViewX Cloud Connector inventory details

 . A completed download/upgrade is indicated by the  symbol in the same location  .

> **Note:** Based on the internet bandwidth and the number of cloud connectors being installed, the downloading of the cloud connector may vary between 5 to 15 minutes.

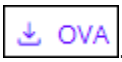
31. To approve the cloud connector installation:

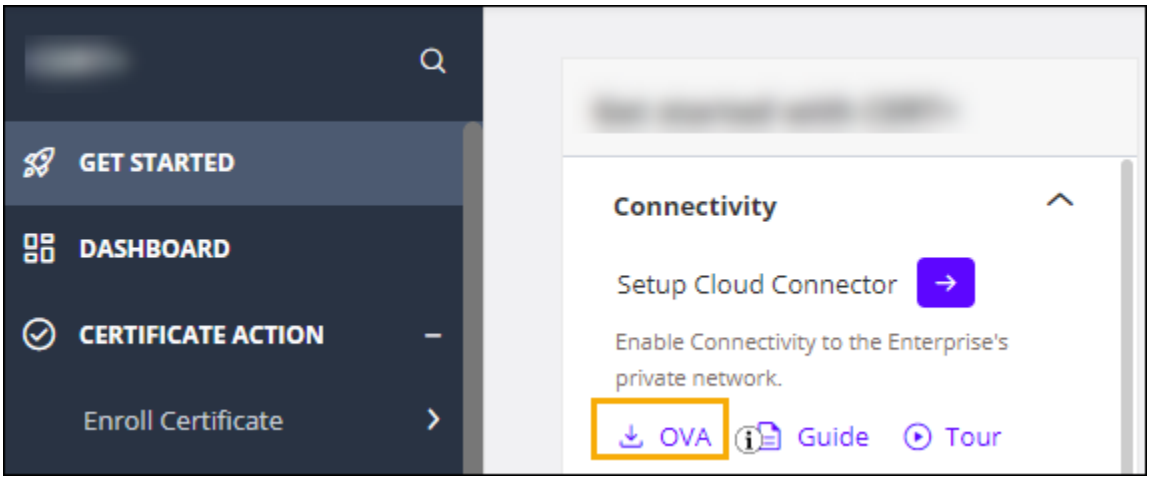
    a. Go to  (**Menu**) > **Platform** > **Connectivity** > **Cloud Connector**.
    The **Settings :: Cloud Connector** inventory page is displayed.

    b. For the cloud connector instance installed, from the **Actions** column, click **Approve**.

> **Troubleshooting:** If the AppViewX Cloud Connector instance has been approved but is not moved to the **Running** state, you can check the pod status and/or restart the pod(s), as required.

Setting up the AppViewX Cloud Connector using a Virtual Image for GCP

1. Log in to the GCP console and start a **Cloud Shell** instance.
2. From the command line terminal, create a bucket with the default settings.
   `gsutil mb gs://my-virtual-appliances-bucket`
3. Download the GCP OVA from the AppViewX release portal.
   Alternatively, you can Download the release package in the OVA format, from the respective AppViewX's product line landing page, under **GET STARTED** menu > **Connectivity** section, click  .



4. Upload it to the Cloud Shell. To upload/copy the OVA to the GCP bucket, execute the following command:

```
gsutil cp ~/path-to-file/local
  gs://my-virtual-appliances-bucket/my-va-file.ova
```

5. Create a virtual instance from the OVA.

```
gcloud compute instances import <my-instance> \
  --source-uri=gs://my-virtual-appliances-bucket/my-va-file.ova \
  --zone southamerica-east1-a \
  --os=ubuntu-1804
```

6. Login to the GCP node using AppViewX credentials.

   `ssh appviewx@<node IP>`

   After the execution of this step, a script is executed to validate if the cloud connector exists on this host machine and the following prompt is displayed: **Would you like to opt for automated installation of the cloud connector? (y/n):**.

7. Since these instructions are for setting up the cloud connector via the user interface, enter **n**.

```
Authorized uses only. All activity may be monitored and reported.
appviewx@192.168.31.143's password:
You have new mail.
Last login: Fri Apr 12 04:37:30 2024 from 192.168.236.254
 [x] - Cloud Connector does not exist on the server. Initiating the cloud connector setup on the server
 [x] - Please make sure you have access to the tenant ID and master key for proceeding with the cloud connector setup
Would you like to opt for automated installation of the cloud connector? (y/n): n
 [x] - Skipping Automated Cloud Connector installation. Refer Cloud Connector User Guide to set it up manually.
```

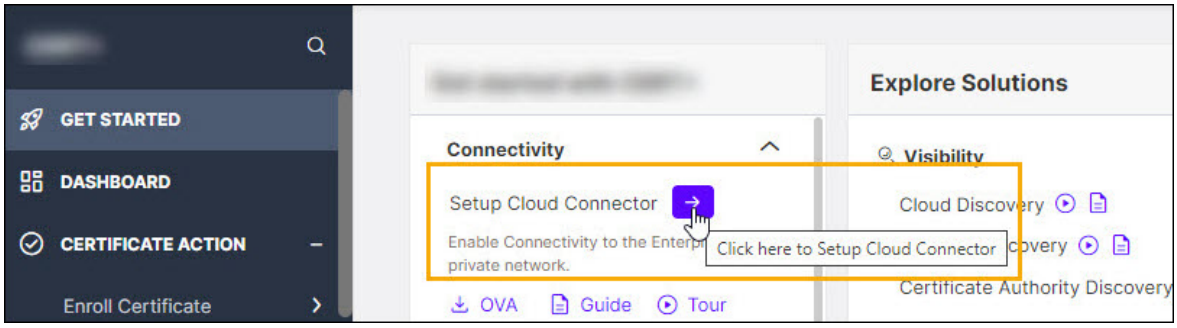
8. Enter your account URL (for example, https://tenant-name.appvx.com/appviewx/login) in the address bar of your browser.

   The AppViewX login page is displayed.

9. Login to AppViewX.
10. Navigate to the cloud connector's setup interface.

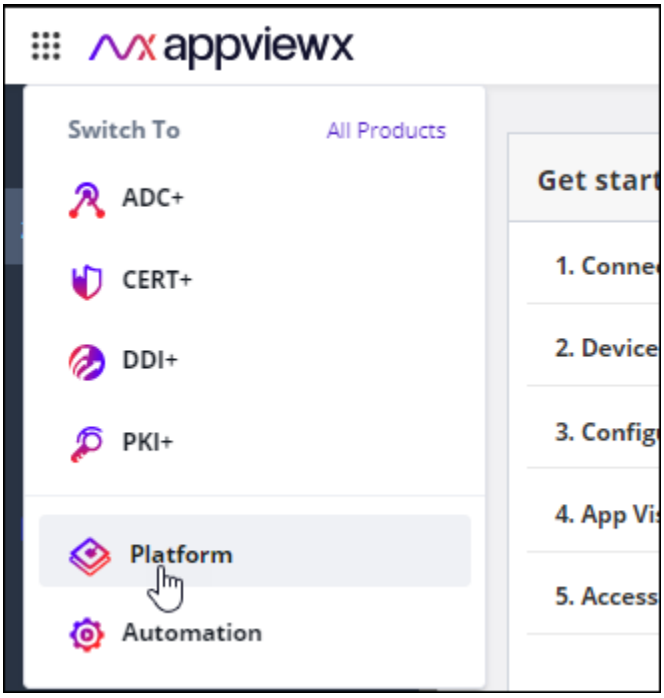
There are three ways you can access the interface for setting up the AppViewX Cloud Connector:

- From the product landing page (that you will see as soon as you have logged in)
  - Expand the **Connectivity** section and click  .

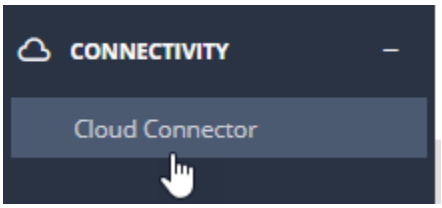


You will be redirected to the **Settings :: Cloud Connector** page.

- From the new navigation menu (displayed by default starting product version 2022.1.0 FP3 onwards):
  a. From the menu in the top-right corner of the page, select **Platform**.



  b. From the **Platform** menu, under **Connectivity**, click **Cloud Connector**.



The **Settings :: Cloud Connector** page is displayed.

- From the old navigation menu:

> **Note:** For instructions on switching between the new and the old navigation menus, click here.

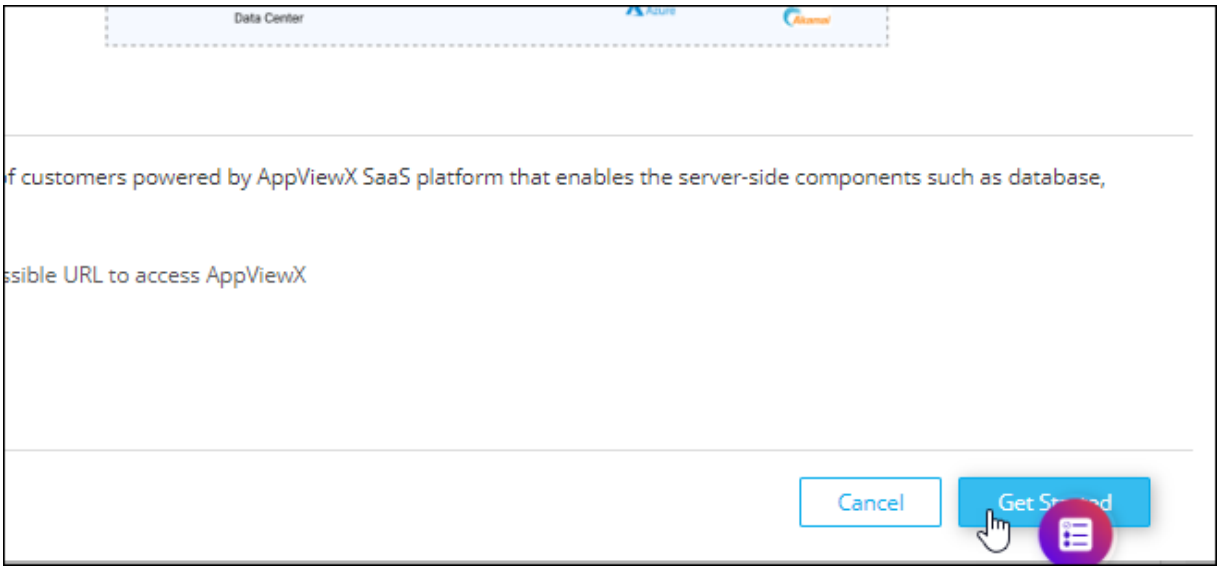
a. From the top right corner of the landing page, click the menu icon.
b. From the menu displayed, navigate to **Settings** > **Cloud Connector**.

The **Settings :: Cloud Connector** page is displayed.

11. On the **Setting :: Cloud Connector** page, click **Add Cloud Connector**.
    The **Cloud Connector Setup** screen is displayed.

    The landing page gives you a quick introduction to the AppViewX Cloud Connector, with a graphical representation of how the infrastructure is deployed and works.

12. To start with the process of adding the cloud connector, from the bottom-right corner of the screen, click **Get Started**.



You will be redirected to the **Basic Information** page.

13. On the **Basic Information** page, configure the basic cloud connector settings.

a. To install the cloud connector via the virtual image, from **Installation Type**, select **Virtual Image**.

> **Note:** Click here to read how a virtual image-based installation is different from a native OS installation.

b. In the **Cloud Connector Name (FQDN)** field, enter the hostname of the machine on which the AppViewX Cloud Connector will be installed.

> **Tip:** To retrieve the hostname, from the command line terminal of the host machine, execute the following command: `hostname -f`.

> **Note:** The hostname entered here is added to the license file that will be generated and downloaded as part of the installer. Therefore, the license file can be used to install the cloud connector only on the machine with the entered hostname and no other.

> **Tip:** The **Setup Cloud Connector** section to the right of the **Basic Information** screen lists hyperlinks to the prerequisites required for setting up the AppViewX Cloud Connector. To read more about what the AppViewX Cloud Connector offers, click **Learn More**.

c. Click **Next**.

14. [Optional] Execute a prerequisite check script.

> **Note:** This is an **optional** step. The prerequisite check script is executed automatically at the time of installing the AppViewX Cloud Connector and the results are shown as a part of the installation logs.

To simplify compliance to the AppViewX Cloud Connector installation prerequisites, you can execute a script to identify and rule out any deviations from the prerequisites.

a. On the **Basic Information** screen, under **Setup Cloud Connector**, you will see a list of the installation prerequisites.

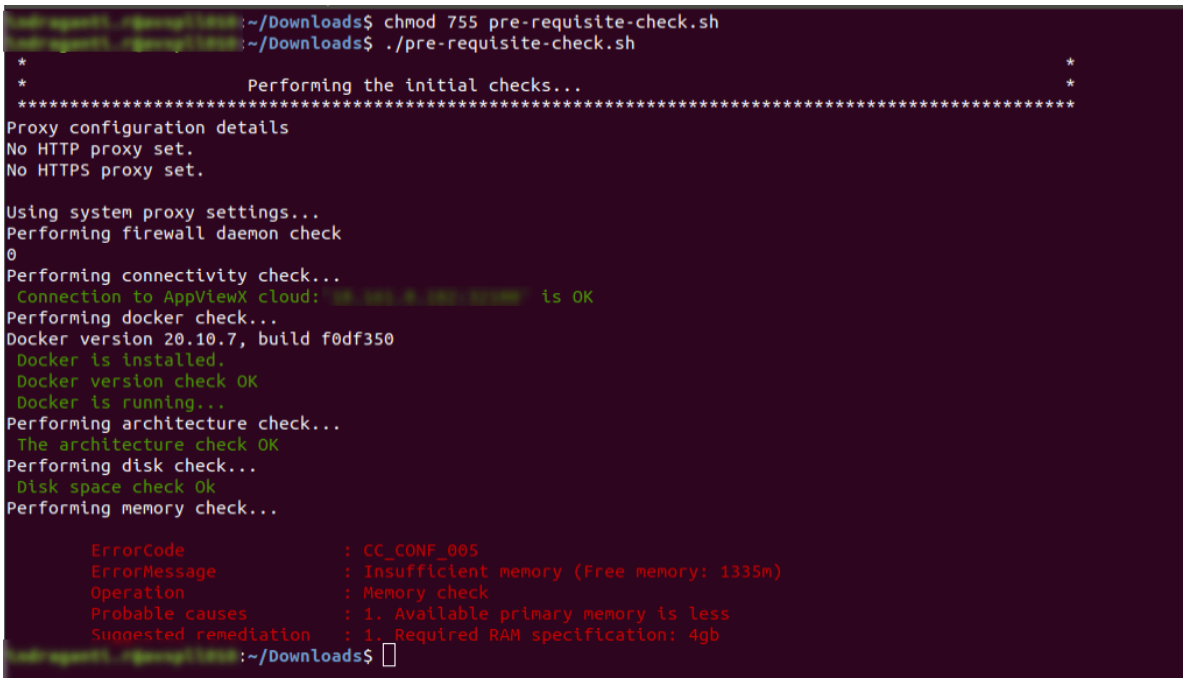
From this list, for **Executing the Prerequisites Check Script**, to download the script, click . The **pre-requisite-check.sh** script file is downloaded.

b. Securely copy the **pre-requisite-check.sh** via SCP/SFTP to the host machine where the AppViewX Cloud Connector is to be installed

c. Convert the downloaded script file into an executable file using the chmod command, as shown below:`chmod 755 pre-requisite-check.sh`

d. Execute the **.sh** prerequisite check script file: `./pre-requisite-check.sh`
If the node does not meet the prerequisites for the AppViewX Cloud Connector installation, the output of the command returns an error code and the corresponding error message, causes, and fixes, if any.

For example, as seen in the sample output in the image below, the prerequisite check for the memory requirement has failed.



Note: For resolutions to the prerequisite check failure scenarios, click here.

15. Click **Next**.
You will be navigated to the **AssignData Center** screen, where, for deploying the AppViewX Cloud Connector, you can either select an existing data center or add a new one.
16. To use an existing data center, select one from the options displayed on the **Assign Data Center** screen.

> **Tip:** Alternatively, you can use the **Search…** field on this screen to search for an existing data center.

To add a new data center:

a. Click **Add Data Center.**

b. In the **Add Data Center** dialog box, enter a name for the new data center.

c. Click **Save**.
The new data center will now be displayed on the **Assign Data Center** screen along with the other existing data centers.

d. Select the required data center.

> **Tip:** The **Data Center based routing** section to the right of the **Assign Data Center** screen explains the concept of data center-based routing and how you can achieve high availability. To read more on this, click **Learn More** from the top-right corner of this screen.

17. Click **Next**.
The **Advanced Configuration** screen is displayed.
18. On the **Advanced Configuration** page, to configure the TLS authentication and proxy server settings for your cloud connector:

a. Enter/Select the advanced configuration settings for the AppViewX Cloud Connector.

> **Note:** The **Data center** field is auto-populated based on your selection on the **Assigning a Data Center** screen.

Field	Description
TLS Authentication	**Tip:** The section on the right of the screen gives you a brief context of what is **TLS Authentication**. To read more, click **Learn More** (next to the **TLS Authentication** heading). • To auto-generate a TLS certificate, select **Auto-generate** (default selection). By default, the certificate is generated using the AppViewX CA.

Field	Description
	Note: The created certificate is available in the certificate inventory. You can: • Assign this certificate to a certificate group • Configure a certificate expiry alert for this certificate group from the **Server Certificate** dashboard, using the **Certificate Summary Report** widget settings • To enter details of a custom TLS certificate, select **Custom**. The **TLS Certificate Password** and **Custom TLS Certificate** fields are displayed. The instructions for filling these fields are given below.
TLS Certificate Password*	**Note:** This field is displayed only if you have selected to enter details of a Custom TLS certificate in the **TLS Authentication** field. Password of the TLS certificate (that will be uploaded in the next step) **Note:** This is a mandatory field if a Custom TLS certificate is uploaded. AppViewX supports only password-protected Custom TLS certificates.
TLS Certificate	**Note:** This field is displayed only if you have selected to enter details of a Custom TLS certificate in the **TLS Authentication** field. To upload a custom TLS certificate: i. To navigate to the location of the custom TLS certificate, click within the field. ii. Select the certificate file. iii. Click **Open**. iv. To upload the custom TLS certificate selected, click **Upload**.

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>

Note: AppViewX supports only password-protected Custom TLS Certificates.

</td></tr>
<tr><td>Use proxy</td><td>

Tip: The section on the right of the screen gives you a brief context of what is **Proxy based routing**. To read more, click **Learn More** (next to the **Proxy based routing** heading).

A proxy server is required if the AppViewX Cloud Connector is unable to connect to your endpoints available in the internet.

To use a proxy server for the deployment:

i. Select the **Use proxy** checkbox.

ii. To select a preconfigured proxy (for the selected data center), from the **Select Proxy** dropdown list, select a proxy server.

OR

To create a new proxy server setting:

i. Use the Click here option shown below the **Select Proxy** dropdown list.

The **Add Proxy** pop-up screen is displayed.

ii. Enter/Select the details required to add a proxy.

Field descriptions for the Add Proxy details

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>*Proxy Name</td><td>Name of the proxy server</td></tr>
<tr><td>*Server IP</td><td>IP address/FQDN of the proxy server</td></tr>
<tr><td>*Port</td><td>Port number of the proxy server</td></tr>
<tr><td>URL</td><td>From the dropdown menu, select the URL.</td></tr>
</table>

</td></tr>
</table>

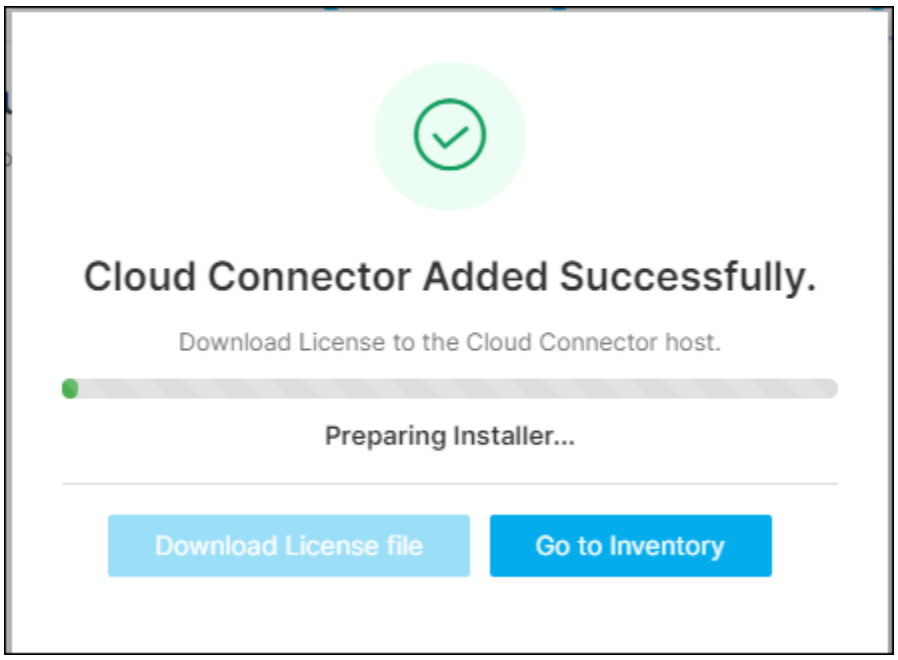
<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>Authentication</td><td>To enable authentication for accessing the proxy server, select this checkbox.</td></tr>
<tr><td>*Username</td><td>Note: This field is enabled only when Authentication is selected.

Enter the username required for accessing the proxy server.</td></tr>
<tr><td>*Password</td><td>Note: This field is enabled only when Authentication is selected.

Enter the password required for accessing the proxy server.</td></tr>
</table>
</td></tr>
</table>

b. Click **Finish**.

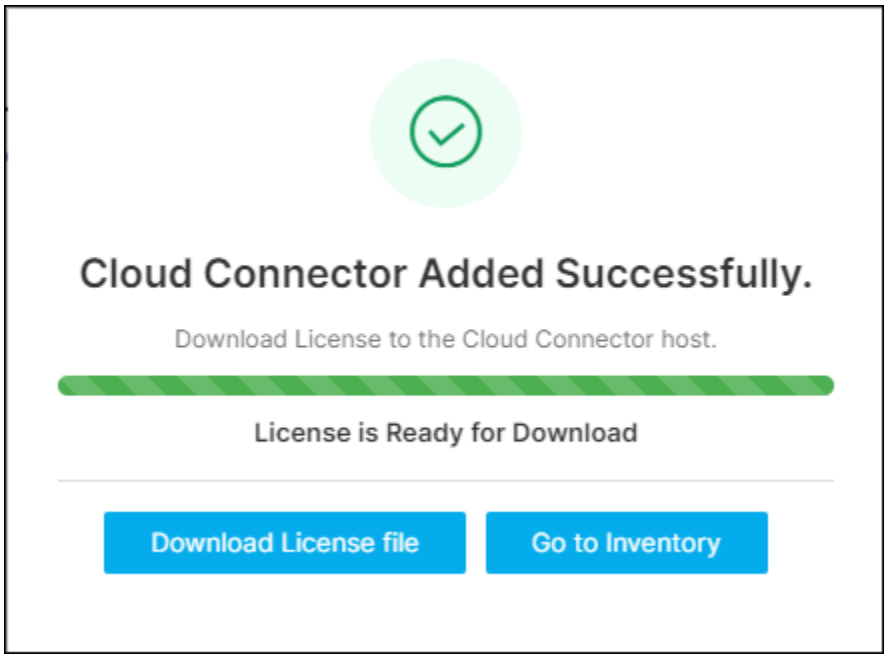
A confirmation message is displayed. AppViewX begins preparing the installer and the license file. Once the license file is ready, you can download it and proceed with the installation of the AppViewX Cloud Connector.



19. Download the license file.

> **Note:** The installer is prepackaged with the OVA, so, for a virtual image-based installation, you only need to download the license file.

a. On the **Cloud Connector Added Successfully** dialog box, when the **License is Ready for Download**, click **Download License file**.



Tip: At this point, if the installer has been deleted or is not usable, and you wish to revert to a native installation, click **Go to Inventory**. It will take you back to the cloud connector inventory, from where you can download the license file and installer for the native OS download.

Tip: You can also choose to download the license file and the installer package individually. To do this:

i. Click the **Cloud Connector Name**.

   The selected Cloud Connector's details are shown in a pane to your right.

ii. To download the AppViewX Cloud Connector installer package, click **Download Cloud Connector**. This is useful in the event that the installer has been deleted or is no longer usable.

   To download the license file, click **Download License**.

Note: A installer download is made available even for a virtual-image based deployment, to help you with reconfiguration in case the existing OVA configuration is deleted.

b. Save the license file on the OVA node.

On the **Settings :: Cloud Connector** page, details of this AppViewX Cloud Connector are added in the inventory table, which is explained here.

20. Install the AppViewX Cloud Connector Agent.

> **Note:** The following steps assume that:
> • All system prerequisites are fulfilled by the host machine.
> • The AppViewX Cloud Connector installer (downloaded in the above step) is securely copied via SCP/SFTP to the host machine where the AppViewX Cloud Connector is to be installed.

a. On the node where the AppViewX Cloud Connector agent will be installed, from the extracted installation package, run the **./install.sh** script.

The script will check if the installation prerequisites for the AppViewX Cloud Connector have been fulfilled.

> **Note:** Ensure that the license file is placed in the same location as the **install.sh** script. If the license file is placed in another location, run the install.sh script using the following command:
>
> ```
> ./install.sh <complete path of the license file with the filename>
> ```

On successful verification of the prerequisites, you will be prompted to specify if you want to manage f5 BIG-IP devices and if you need auto-enrollment of the certificates.

```
Do you want to manage f5 BIG-IP devices? (y/n)?:n
Continuing with the installation

Do you need Auto-enrollment of the certificate using EST/SCEP/ACME? (y/n)?:y
Please choose one or more protocol (use comma separated numbers): 1)EST(MTLS)   2)SCEP(HTTP)  3)ACME(HTTPS)
1,2,3
Auto enrollment enabled successfully for protocol(s): MTLS HTTP HTTPS
Do you want to enable Syslog receiver for a near real time configuration updates from the devices. (y/n) n
syslog enabled  n
```

b. When prompted, enter the required input value(s):

> **Important:** If you choose to **not enable** any of the following features, to enable them later, you will have to reinstall the AppViewX Cloud Connector.

i. If you want manage f5 BIG-IP devices, enter **y**/**n** for yes/no, respectively.

ii. When prompted to enable auto-enrollment of the certificate using one of the following supported auto-enrollment protocols, enter **y** only if the cloud connector is being installed in a demilitarized zone (DMZ) or devices in a restricted environment (that disables them from connecting to the **<tenant>-aep** directly.

• If you choose **y** (yes) here, enter the required protocol(s) name.

> **Note:** By default, the AppViewX certificate is enabled for auto-enrollment. To enable custom certificate for auto-enrollment:
>
> 1. Execute the command `./avxctl upgrade gateway-cert`.
> 2. When prompted, enter the location of the custom certificate.

> **Note:** If you are a KUBE+ customer, the auto-enrollment gateway should be enabled as part of the installation for your KUBE+ use cases to work via the cloud connector.

iii. If you want to enable Syslog receiver for a near-real time configuration updates from the devices, enter **y/n** for yes/no, respectively. For **configuring Syslog reception**, refer to the Platform User guide section, Syslog Reception.

In case you have an older version of AppViewX on cloud and want to make use of Syslog capabilities for ADC, you must manually activate the Syslog flag by setting **SYSLOG_ENABLED**=**true** in the path **ccpath/deps/properties**.

c. Enter the sudo password.

After the relevant details have been entered, the installation proceeds. Installation logs, according to the outcome of the installation, are displayed.

Given below are sample installation logs:

```
Loaded image: rancher/k3s:v1.23.3-k3s1
Loaded image: rancher/k3d-tools:5.2.2
Loaded image: rancher/mirrored-pause:3.6
[36mINFO[0m[0000] [SimpleConfig] Hostnetwork selected - disabling injection of docker host into the cluster, server load balancer and setting the api
 port to the k3s default
[33mWARN[0m[0000] No node filter specified
[33mWARN[0m[0000] No node filter specified
[33mWARN[0m[0000] No node filter specified
[36mINFO[0m[0000] Prep: Network
```

```
[36mINFO[0m[0000] Re-using existing network 'host' (8bebb4ae61001f74487d0aa6b315396405d0127c938da1206614d113295ae139)
[36mINFO[0m[0000] Created volume 'k3d-cc-images'
[36mINFO[0m[0000] Starting new tools node...
[36mINFO[0m[0000] Starting Node 'k3d-cc-tools'
[36mINFO[0m[0001] Creating node 'k3d-cc-server-0'
[36mINFO[0m[0001] Using the k3d-tools node to gather environment information
[36mINFO[0m[0001] Starting cluster 'cc'
[36mINFO[0m[0001] Starting servers...
[36mINFO[0m[0001] Starting Node 'k3d-cc-server-0'
[36mINFO[0m[0033] All agents already running.
[36mINFO[0m[0033] All helpers already running.
[36mINFO[0m[0033] Cluster 'cc' created successfully!
[36mINFO[0m[0034] You can now use it like this:
kubectl cluster-info
Cluster setup is completed. Will start the deployment shortly...
Importing the required images...
[36mINFO[0m[0000] Importing image(s) into cluster 'cc'
[36mINFO[0m[0000] Importing images from 1 tarball(s)...
[36mINFO[0m[0000] Importing images '[/home/appviewx/CCTEST/deps/tools/mid-server-docker-image/avx-mid-server-base-22.1.0.0.tar]' into node
 'k3d-cc-server-0'...
[36mINFO[0m[0024] Successfully imported image(s)
[36mINFO[0m[0024] Successfully imported 1 image(s) into 1 cluster(s)
Import in progress...
[36mINFO[0m[0000] Importing image(s) into cluster 'cc'
[36mINFO[0m[0000] Importing images from 1 tarball(s)...
[36mINFO[0m[0000] Importing images '[/home/appviewx/CCTEST/deps/tools/mid-server-docker-image/k3d-tools-5.2.2.tar]' into node 'k3d-cc-server-0'...
[36mINFO[0m[0005] Successfully imported image(s)
[36mINFO[0m[0005] Successfully imported 1 image(s) into 1 cluster(s)
Import in progress...
[36mINFO[0m[0000] Importing image(s) into cluster 'cc'
[36mINFO[0m[0000] Importing images from 1 tarball(s)...
[36mINFO[0m[0000] Importing images '[/home/appviewx/CCTEST/deps/tools/mid-server-docker-image/rancher-mirrored-coredns-coredns-1.8.6.tar]' into
 node 'k3d-cc-server-0'...
[36mINFO[0m[0007] Successfully imported image(s)
[36mINFO[0m[0007] Successfully imported 1 image(s) into 1 cluster(s)
[36mINFO[0m[0000] Importing image(s) into cluster 'cc'
[36mINFO[0m[0000] Importing images from 1 tarball(s)...
```

```
[36mINFO[0m[0000] Importing images '[/home/appviewx/CCTEST/deps/tools/mid-server-docker-image/rancher-local-path-provisioner-v0.0.21.tar]' into
 node 'k3d-cc-server-0'...
[36mINFO[0m[0004] Successfully imported image(s)
[36mINFO[0m[0004] Successfully imported 1 image(s) into 1 cluster(s)
[36mINFO[0m[0000] Importing image(s) into cluster 'cc'
[36mINFO[0m[0000] Importing images from 1 tarball(s)...
[36mINFO[0m[0000] Importing images '[/home/appviewx/CCTEST/deps/tools/mid-server-docker-image/rancher-mirrored-pause-3.6.tar]' into node
 'k3d-cc-server-0'...
[36mINFO[0m[0003] Successfully imported image(s)
[36mINFO[0m[0003] Successfully imported 1 image(s) into 1 cluster(s)
Deploying the Cloud Connector...
NAME: avx-mid-server-starter
LAST DEPLOYED: Mon May 30 15:51:13 2022
NAMESPACE: cc
STATUS: deployed
REVISION: 1
NOTES:
1. It may take a couple of minutes for the Cloud Connector to be up.
  kubectl get pod --namespace cc
**************************************************************************
*    Congratulations!!! The installation completed successfully.   *
*    Please wait till the Cloud Connector is up and running.       *
**************************************************************************
(1%) Cloud Connector status: Running
[32m Cloud Connector is up and running. (B[m
```

Troubleshooting: For installation errors, refer to the Troubleshooting section.

The AppViewX Cloud Connector consists of two important components—the starter plugin and the platform. The starter plugin component is installed along with the AppViewX Cloud Connector, in the same installation process.

When installed, the starter plugin is used to initiate the download of the platform component. The platform component is used to host business use cases related to the AppViewX Cloud Connector.

When the platform component download is in progress, it is indicated by the  symbol prefixed to the platform component version number in the AppViewX Cloud Connector inventory details

 . A completed download/upgrade is indicated by the  symbol in the same location  .

> **Note:** Based on the internet bandwidth and the number of cloud connectors being installed, the downloading of the cloud connector may vary between 5 to 15 minutes.

21. To approve the cloud connector installation:

    a. Go to  (**Menu**) > **Platform** > **Connectivity** > **Cloud Connector**.
    The **Settings :: Cloud Connector** inventory page is displayed.

    b. For the cloud connector instance installed, from the **Actions** column, click **Approve**.

> **Troubleshooting:** If the AppViewX Cloud Connector instance has been approved but is not moved to the **Running** state, you can check the pod status and/or restart the pod(s), as required.

Setting up the AppViewX Cloud Connector via the Native OS

> **Note:** The installation occurs with the privileges of the user who begins the installation.

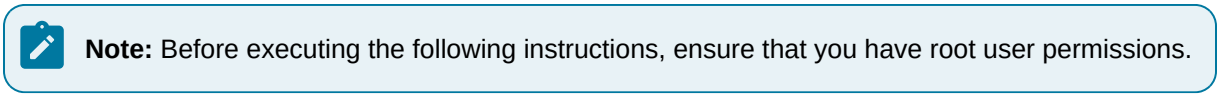
> **Note:** The steps for installing the AppViewX Cloud Connector via the native OS assume that you have gone through the system requirements across the following categories: hardware, operating system, Docker, and server and network.
> If the host machine does not/cannot fulfill the installation prerequisites, you can set up the AppViewX Cloud Connector via the AppViewX SaaS OVA. To know more about the OVA and for instructions on setting up the AppViewX Cloud Connector using the AppViewX SaaS OVA, click here.

> **Note:** If this AppViewX Cloud Connector installation requires configuring a proxy server, click here for instructions.

Setting up the AppViewX Cloud Connector via a Native OS using the Automated Script

To install the AppViewX Cloud Connector using the automated installation script, from the cloud connector details banner, you can either download the installer script or copy the `curl` command.

1. Configure the native OS for the cloud connector installation.



Note: Before executing the following instructions, ensure that you have root user permissions.

a. Create the **appviewx** user and directory.

```
useradd appviewx && mkdir /home/appviewx/ && chown appviewx:appviewx /home/appviewx && chmod 700 /home/appviewx
```

b. Uninstall the previous version of Docker.

- **For RHEL and Amazon Linux 2**:

```
sudo yum remove docker \
 docker-client \
 docker-client-latest \
 docker-common \
 docker-latest \
 docker-latest-logrotate \
 docker-logrotate \
 Docker-engine
```

- **For Ubuntu**

```
for pkg in docker.io docker-doc docker-compose podman-docker containerd runc; do sudo apt-get remove $pkg; done
```

c. Configure the hostname and the nameserver.

i. Configure the hostname: `sudo hostnamectl set-hostname "hostname"`

ii. Add the host IP address at the end of the **hosts** file.

```
sudo vi /etc/hosts
<ip> <hostname>
```

iii. Replace the default nameserver IP address.

```
sudo systemctl stop systemd-resolved.service
sudo systemctl disable systemd-resolved.service
sudo rm /etc/resolv.conf
sudo vi /etc/hosts/
<ip> <hostname>
```

iv. To configure nameserver, add the IP address of the nameserver to beginning of the **resolv.conf**

```
sudo vi /etc/resolv.conf
nameserver <nameserver ip>
```

d. Add Docker repo and install Docker.

- For **RHEL**

```
sudo yum install -y yum-utils && sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo sudo yum install -y
 docker-ce docker-ce-cli containerd.io
sudo systemctl start docker
sudo systemctl enable docker
sudo systemctl status docker
```

- For **Amazon Linux 2**

```
sudo yum install -y docker containerd
sudo systemctl start docker
sudo systemctl enable docker
sudo systemctl status docker
```

- For **Ubuntu**

```
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg
#Add Docker’s official GPG key for Ubuntu:
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg
#Use the following command to set up the repository:
echo \
 "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \  "$(. /etc/os-release &&
 echo "$VERSION_CODENAME")" stable" | \
 sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

To install Docker for Ubuntu

```
sudo apt-get update && sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin sudo systemctl start
 docker
sudo systemctl enable docker
sudo systemctl status docker
```

e. Add the appviewx user in the Docker group.

```
sudo groupadd docker
sudo usermod -aG docker appviewx
```

f. Install NTP.

- For **RHEL**

```
sudo yum update -y
sudo yum install ntp -y
sudo systemctl restart ntpd
sudo systemctl enable ntpd
ntpq -np
```

- For **Amazon Linux 2**

```
sudo yum install -y chrony
sudo systemctl enable chronyd
sudo systemctl start chronyd
sudo systemctl status chronyd
```

- For **Ubuntu**

```
sudo apt update -y
sudo apt install ntp -y
sudo systemctl restart ntp
sudo systemctl enable ntp
ntpq -np
```

g. Install the additional packages required for the AppViewX Cloud Connector to run.

- For **RHEL** and **Amazon Linux 2**

```
sudo yum install bind-utils net-tools telnet tcpdump curl -y
```

- For **Ubuntu**

```
sudo apt install bind9-utils net-tools telnet tcpdump curl -y
```

h. Limit the maximum number of processes that the appviewx user can create.

```
sudo vi /etc/security/limit.conf
appviewx soft nproc 65536
appviewx hard nproc 65536
appviewx soft nofile 65536
appviewx hard nofile 65536
```

i. Allow the **appviewx** user to run **sudo root**.

```
sudo vi /etc/sudoers
appviewx ALL=(ALL) ALL
```

j. If access to the internet is restricted, whitelist the AppViewX repository to perform OS patching.

Note: RHEL OS patching is now supported directly by Red Hat.

- For **Ubuntu**
  i. Create a file, **appviewx_repo.list**, and open it for editing.

```
vi /etc/apt/sources.list.d/appviewx_repo.list
```

  ii. Add the following to the **appviewx_repo.list** file:

```
#appviewx apt repo
deb https://repos.appviewx.com/repository/ubuntu/ jammy main restricted
deb https://repos.appviewx.com/repository/ubuntu/ jammy-updates main restricted
deb https://repos.appviewx.com/repository/ubuntu/ jammy universe
deb https://repos.appviewx.com/repository/ubuntu/ jammy-updates universe
deb https://repos.appviewx.com/repository/ubuntu/ jammy multiverse
deb https://repos.appviewx.com/repository/ubuntu/ jammy-updates multiverse
deb https://repos.appviewx.com/repository/ubuntu/ jammy-backports main restricted universe multiverse
deb https://repos.appviewx.com/repository/ubuntu/ jammy-security main restricted
deb https://repos.appviewx.com/repository/ubuntu/ jammy-security universe
deb https://repos.appviewx.com/repository/ubuntu/ jammy-security multiverse
```

  iii. Create an **auth.conf file** and reach out to the AppViewX Support team for the credentials that need to be added to the file.

```
vi /etc/apt/auth.conf
```

k. Create a new directory at the location **/home/appviewx** and name it **cc-installer**.

l. Ensure that the AppViewX Cloud Connector can establish connectivity with the AppViewX SaaS server endpoints over HTTPS (port 443).

m. To verify connectivity with the AppViewX SaaS servers, use the **cURL** utility.

```
curl -k --max-time 20 --connect-timeout 20 -s -o /dev/null -w "%{http_code}" "<<https://AppViewX SaaS server URL>>/socket.io/?EIO=3&transport=polling&t=O11wka_"
```

If connectivity has been established successfully, the command will return the HTTP code 200. If the command returns any other code, it indicates that connectivity is not established.

2. Access the AppViewX user interface.

In order to set up the AppViewX Cloud Connector instance, you will need to login to the connectivity service's user interface. The following steps will outline the navigation and steps required to access the AppViewX Cloud Connector's setup interface.

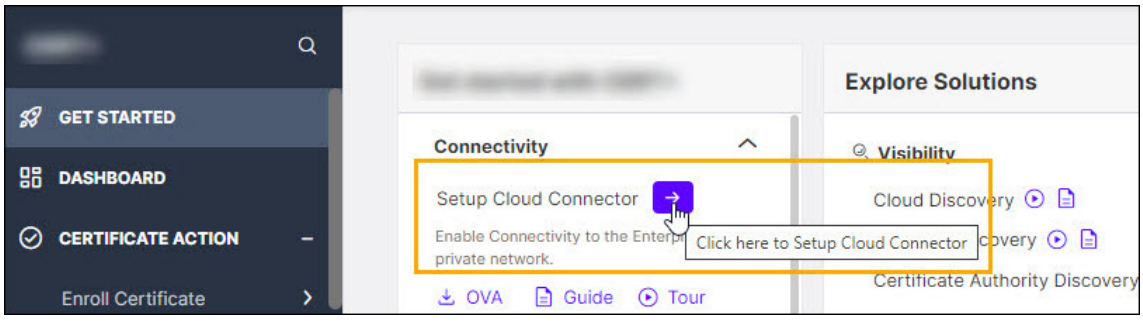
As an additional layer of security, AppViewX issues client certificates to access the AppViewX GUI. The client certificate will be made available as part of the onboarding process. Upload this client certificate to the browser to start accessing the product.

a. Enter your account URL (for example, https://tenant-name.appvx.com/appviewx/login) in the address bar of your browser.
The AppViewX login page is displayed.

b. Login to AppViewX.

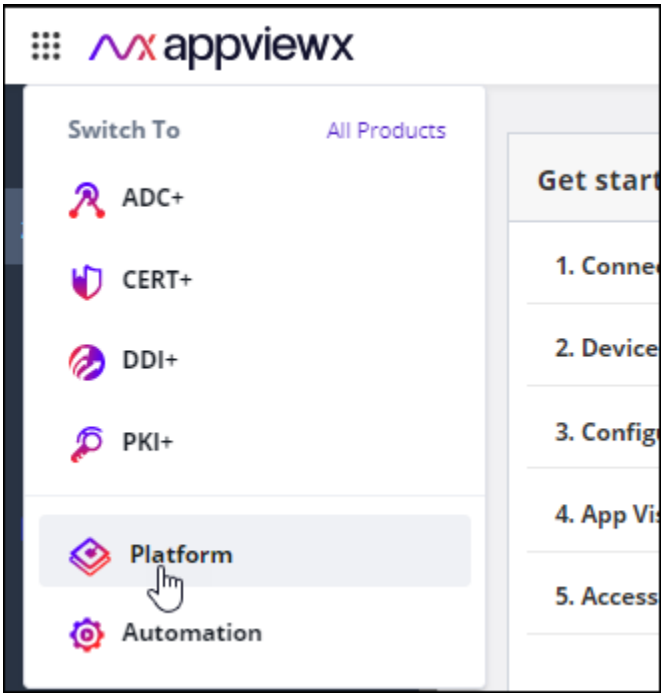
c. Navigate to the cloud connector's setup interface.
There are three ways you can access the interface for setting up the AppViewX Cloud Connector:
• From the product landing page (that you will see as soon as you have logged in)

• Expand the **Connectivity** section and click  .

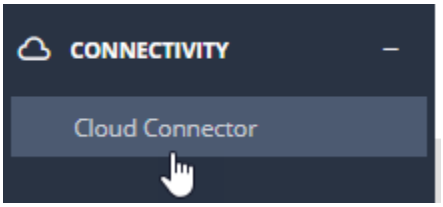


You will be redirected to the **Settings :: Cloud Connector** page.

- From the new navigation menu (displayed by default starting product version 2022.1.0 FP3 onwards):
  i. From the menu in the top-right corner of the page, select **Platform**.



  ii. From the **Platform** menu, under **Connectivity**, click **Cloud Connector**.



  The **Settings :: Cloud Connector** page is displayed.

- From the old navigation menu:

> **Note:** For instructions on switching between the new and the old navigation menus, click here.

  i. From the top right corner of the landing page, click the menu icon.
  ii. From the menu displayed, navigate to **Settings** > **Cloud Connector**.

  The **Settings :: Cloud Connector** page is displayed.

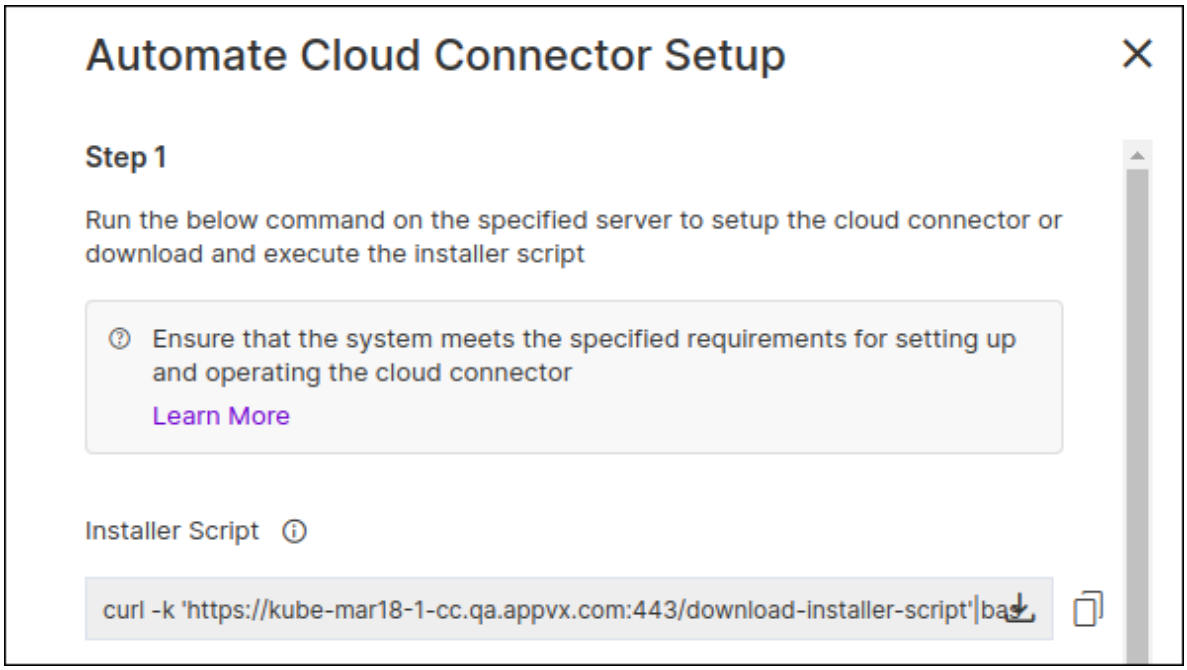
3. Install the AppViewX Cloud Connector using the automated script.

To install the AppViewX Cloud Connector using the automated installation script, from the cloud connector details banner, you can either download the installer script or copy the `curl` command.

a. From the cloud connector details banner, click **Download Installer Script**.

OR

Copy the `curl` command to run the installer script.



b. On the host machine:

If you have downloaded the installer script, execute the following commands:

```
chmod +x installer.sh

./installer.sh
```

OR

If you have copied the `curl` command, to execute the command, paste the command on the command line terminal and press **Enter**.

Internet connectivity on the host machine is validated. The installation proceeds only if the host machine has internet access.

> **Note:** For instructions on proceeding with the installation despite not having internet access, click here.

c. When prompted to **Please provide the master key**, enter the required details.

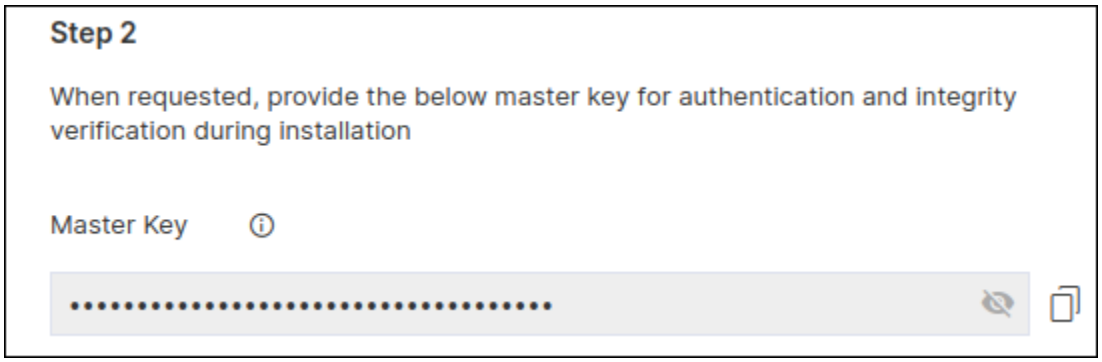
To retrieve the master key:

i. Go to (**Menu**) > **Platform** > **Connectivity** > **Cloud Connector**.

The **Settings :: Cloud Connector** inventory page is displayed.

ii. From the cloud connector details banner, under **Automate Cloud Connector Setup**, click **Steps to Automate Setup**.

iii. From the **Automated Cloud Connector Setup** window, under **Step 2**, copy the **Master Key**.



iv. Paste the master key in the terminal window and press **Enter**.

The cloud connector installation script will check for the prerequisites and trigger the cloud connector installation.

When the cloud connector instance is successfully installed, a corresponding entry will be listed in the cloud connector inventory.

d. When prompted, **Please enter Datacenter**, enter the name of the data center on which this cloud connector will be deployed.

e. **Note:** Enabling auto-enrollment protocols is recommended only if the cloud connector is being installed in a demilitarized zone (DMZ) or devices in a restricted environment (that disables them from connecting to the <tenant>-aep directly.

When prompted to enable auto-enrollment protocols, enter **y** and enter the protocol name(s) you want to enable. For instructions on enabling auto-enrollment protocols, click here.

Note: By default, only the AppViewX certificate is enabled for auto-enrollment. To enable custom certificate for auto-enrollment:

i. Execute the command `./avxctl upgrade gateway-cert`.

ii. When prompted, enter the location of the custom certificate.

On successful completion of the setup, a corresponding instance of this cloud connector is displayed in the inventory.

> **Note:** The cloud connector installation on a OVA-based host machine will not prompt you to select if you want to manage F5 Big-IP devices. However, after the cloud connector has been installed you can copy the iControl jar in the deps/external_libs folder (click here for instructions) and restart the starter and platform pods (click here for instructions), to enable this feature.

> **Note:** Optional, required only for password authentication) In order to successfully execute the installation, AppViewX needs to run a script for which authentication via the **.pem** file needs to be bypassed. To do this, execute the following commands:
>
> ```
> sudo sed -i 's/.*PasswordAuthentication.*/PasswordAuthentication yes/g' /etc/ssh/sshd_config
> sudo systemctl restart sshd
> ```

4. To approve the cloud connector installation:

   a. Go to (**Menu**) > **Platform** > **Connectivity** > **Cloud Connector**.
   The **Settings :: Cloud Connector** inventory page is displayed.

   b. For the cloud connector instance installed, from the **Actions** column, click **Approve**.

Setting up the AppViewX Cloud Connector via a Native OS using the AppViewX User Interface

1. Configure the native OS for the cloud connector installation.

> **Note:** Before executing the following instructions, ensure that you have root user permissions.

   a. Create the **appviewx** user and directory.

```
useradd appviewx && mkdir /home/appviewx/ && chown appviewx:appviewx /home/appviewx && chmod 700 /home/appviewx
```

   b. Uninstall the previous version of Docker.

- **For RHEL and Amazon Linux 2**:

```
sudo yum remove docker \
 docker-client \
 docker-client-latest \
 docker-common \
 docker-latest \
 docker-latest-logrotate \
 docker-logrotate \
 Docker-engine
```

- **For Ubuntu**

```
for pkg in docker.io docker-doc docker-compose podman-docker containerd runc; do sudo apt-get remove $pkg; done
```

c. Configure the hostname and the nameserver.

i. Configure the hostname: `sudo hostnamectl set-hostname "hostname"`

ii. Add the host IP address at the end of the **hosts** file.

```
sudo vi /etc/hosts
<ip> <hostname>
```

iii. Replace the default nameserver IP address.

```
sudo systemctl stop systemd-resolved.service
sudo systemctl disable systemd-resolved.service
sudo rm /etc/resolv.conf
sudo vi /etc/hosts/
<ip> <hostname>
```

iv. To configure nameserver, add the IP address of the nameserver to beginning of the **resolv.conf**

```
sudo vi /etc/resolv.conf
nameserver <nameserver ip>
```

d. Add Docker repo and install Docker.

- For **RHEL**

```
sudo yum install -y yum-utils && sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo sudo yum install -y
 docker-ce docker-ce-cli containerd.io
sudo systemctl start docker
sudo systemctl enable docker
sudo systemctl status docker
```

- For **Amazon Linux 2**

```
sudo yum install -y docker containerd
sudo systemctl start docker
sudo systemctl enable docker
sudo systemctl status docker
```

- For **Ubuntu**

```
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg
#Add Docker's official GPG key for Ubuntu:
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg
#Use the following command to set up the repository:
echo \
 "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \  "$(. /etc/os-release &&
 echo "$VERSION_CODENAME")" stable" | \
 sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

To install Docker for Ubuntu

```
sudo apt-get update && sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin sudo systemctl start
 docker
sudo systemctl enable docker
sudo systemctl status docker
```

e. Add the **appviewx** user in the Docker group.

```
sudo groupadd docker
sudo usermod -aG docker appviewx
```

f. Install NTP.

- For **RHEL**

```
sudo yum update -y
sudo yum install ntp -y
sudo systemctl restart ntpd
sudo systemctl enable ntpd
ntpq -np
```

- For **Amazon Linux 2**

```
sudo yum install -y chrony
sudo systemctl enable chronyd
sudo systemctl start chronyd
sudo systemctl status chronyd
```

- For **Ubuntu**

```
sudo apt update -y
sudo apt install ntp -y
sudo systemctl restart ntp
sudo systemctl enable ntp
ntpq -np
```

g. Install the additional packages required for the AppViewX Cloud Connector to run.

- For **RHEL** and **Amazon Linux 2**

```
sudo yum install bind-utils net-tools telnet tcpdump curl -y
```

- For **Ubuntu**

```
sudo apt install bind9-utils net-tools telnet tcpdump curl -y
```

h. Limit the maximum number of processes that the appviewx user can create.

```
sudo vi /etc/security/limit.conf
appviewx soft nproc 65536
appviewx hard nproc 65536
appviewx soft nofile 65536
appviewx hard nofile 65536
```

i. Allow the **appviewx** user to run **sudo root**.

```
sudo vi /etc/sudoers
appviewx ALL=(ALL) ALL
```

j. To download the required packages, enable the AppViewX repository.

Important: This step is not applicable for Amazon Linux 2.

k. If access to the internet is restricted, whitelist the AppViewX repository to perform OS patching.

- For **RHEL**

```
mv /etc/yum.repos.d /etc/yum.repos.d_backup
mkdir -p /etc/yum.repos.d
```

```
mv appviewx.repo /etc/yum.repos.d/appviewx.repo

yum clean all

yum update
```

- For **Ubuntu**

  i. Create a file, **appviewx_repo.list**, and open it for editing.

  ```
  vi /etc/apt/sources.list.d/appviewx_repo.list
  ```

  ii. Add the following to the **appviewx_repo.list** file:

  ```
  #appviewx apt repo

  deb https://repos.appviewx.com/repository/ubuntu focal universe

  deb https://repos.appviewx.com/repository/ubuntu focal-updates universe

  deb https://repos.appviewx.com/repository/ubuntu focal multiverse

  deb https://repos.appviewx.com/repository/ubuntu focal-updates multiverse

  deb https://repos.appviewx.com/repository/ubuntu focal-backports main restricted universe multiverse

  deb https://repos.appviewx.com/repository/ubuntu focal-security main restricted

  deb https://repos.appviewx.com/repository/ubuntu focal-security universe

  deb https://repos.appviewx.com/repository/ubuntu focal-security multiverse
  ```

  iii. Create a file, **auth.conf** and open it for editing.

  ```
  vi /etc/apt/auth.conf
  ```

  iv. Add the following to the **auth.conf** file:

  ```
  machine repos.appviewx.com

  login ereq2aYRsD4rR5KjD9H2wN4CBuwC7uVpkFaMq

  password zstGbbn7wBvHTWkREuTYpeL8fVgJEZxkJtsJM
  ```

l. Create a new directory at the location **/home/appviewx** and name it **cc-installer**.

m. Ensure that the AppViewX Cloud Connector can establish connectivity with the AppViewX SaaS server endpoints over HTTPS (port 443).

n. To verify connectivity with the AppViewX SaaS servers, use the **cURL** utility.

```
curl -k --max-time 20 --connect-timeout 20 -s -o /dev/null -w "%{http_code}" "<<https://AppViewX SaaS server

URL>>/socket.io/?EIO=3&transport=polling&t=O11wka_"
```

If connectivity has been established successfully, the command will return the HTTP code 200. If the command returns any other code, it indicates that connectivity is not established.

2. Access the AppViewX user interface.

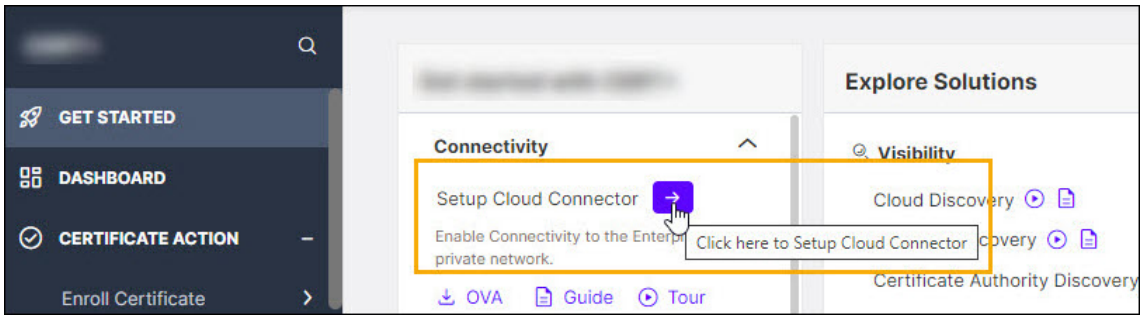
In order to set up the AppViewX Cloud Connector instance, you will need to login to the connectivity service's user interface. The following steps will outline the navigation and steps required to access the AppViewX Cloud Connector's setup interface.

As an additional layer of security, AppViewX issues client certificates to access the AppViewX GUI. The client certificate will be made available as part of the onboarding process. Upload this client certificate to the browser to start accessing the product.

a. Enter your account URL (for example, https://tenant-name.appvx.com/appviewx/login) in the address bar of your browser.
   The AppViewX login page is displayed.

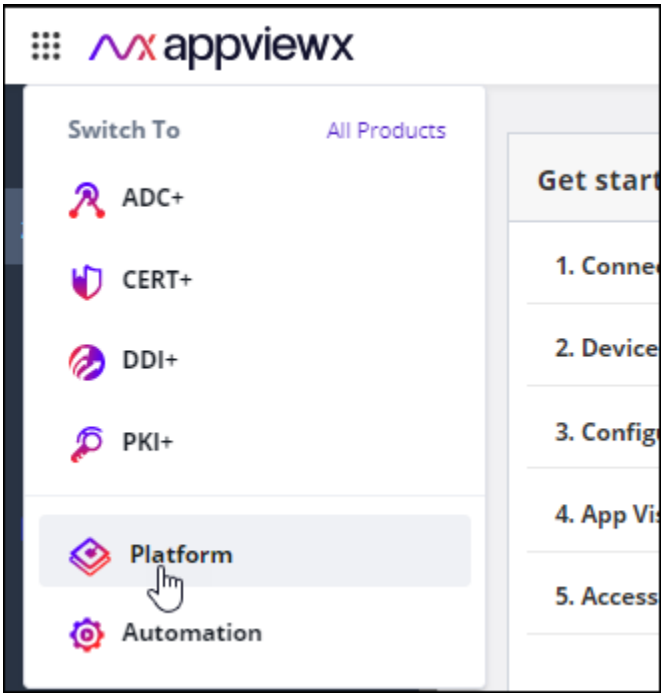
b. Login to AppViewX.

c. Navigate to the cloud connector's setup interface.
   There are three ways you can access the interface for setting up the AppViewX Cloud Connector:
   • From the product landing page (that you will see as soon as you have logged in)
   • Expand the **Connectivity** section and click  .

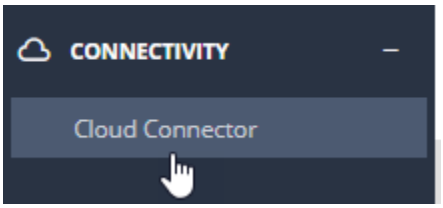


   You will be redirected to the **Settings :: Cloud Connector** page.
   • From the new navigation menu (displayed by default starting product version 2022.1.0 FP3 onwards):

i. From the menu in the top-right corner of the page, select **Platform**.



ii. From the **Platform** menu, under **Connectivity**, click **Cloud Connector**.



The **Settings :: Cloud Connector** page is displayed.

- From the old navigation menu:

> **Note:** For instructions on switching between the new and the old navigation menus, click here.

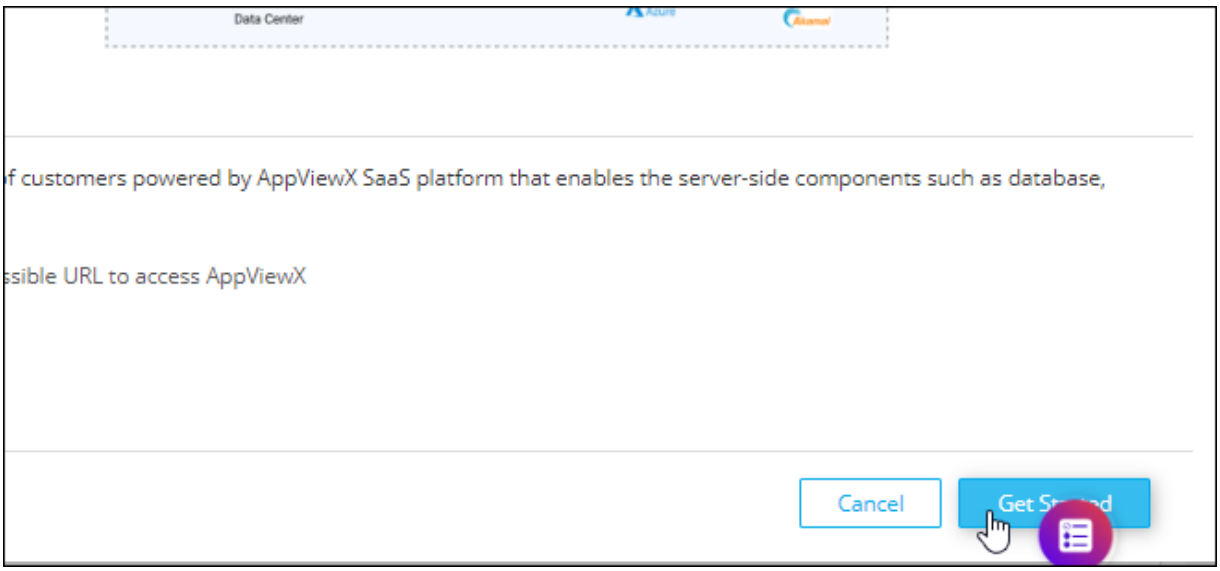
i. From the top right corner of the landing page, click the menu icon.
ii. From the menu displayed, navigate to **Settings** > **Cloud Connector**.

The **Settings :: Cloud Connector** page is displayed.

3. On the **Setting :: Cloud Connector** page, click **Add Cloud Connector**.
   The **Cloud Connector Setup** screen is displayed.

The landing page gives you a quick introduction to the AppViewX Cloud Connector, with a graphical representation of how the infrastructure is deployed and works.

4. To start with the process of adding the cloud connector, from the bottom-right corner of the screen, click **Get Started**.



You will be redirected to the **Basic Information** page.

5. On the **Basic Information** page, configure the basic cloud connector settings.

   a. From **Installation Type**, select **Native OS**.

   b. In the **Cloud Connector Name (FQDN)** field, enter the hostname of the machine on which the AppViewX Cloud Connector will be installed.

> **Tip:** To retrieve the hostname, from the command line terminal of the host machine, execute the following command: `hostname -f`

> **Note:** The hostname entered here is added to the license file that will be generated and downloaded as part of the installer. Therefore, the license file can be used to install the cloud connector only on the machine with the entered hostname and no other.

> **Tip:** The **Setup Cloud Connector** section to the right of the **Basic Information** screen lists hyperlinks to the prerequisites required for setting up the AppViewX Cloud Connector. To read more about what the AppViewX Cloud Connector offers, click **Learn More**.

c. Click **Next**.

6. [Optional] Execute a prerequisite check script.
   To simplify compliance to the AppViewX Cloud Connector installation prerequisites, you can execute a script to identify and rule out any deviations from the prerequisites.

> **Note:** This is an **optional** step. The prerequisite check script is executed automatically at the time of installing the AppViewX Cloud Connector and the results are shown as a part of the installation logs.

   a. On the **Basic Information** screen, under **Setup Cloud Connector**, you will see a list of the installation prerequisites.

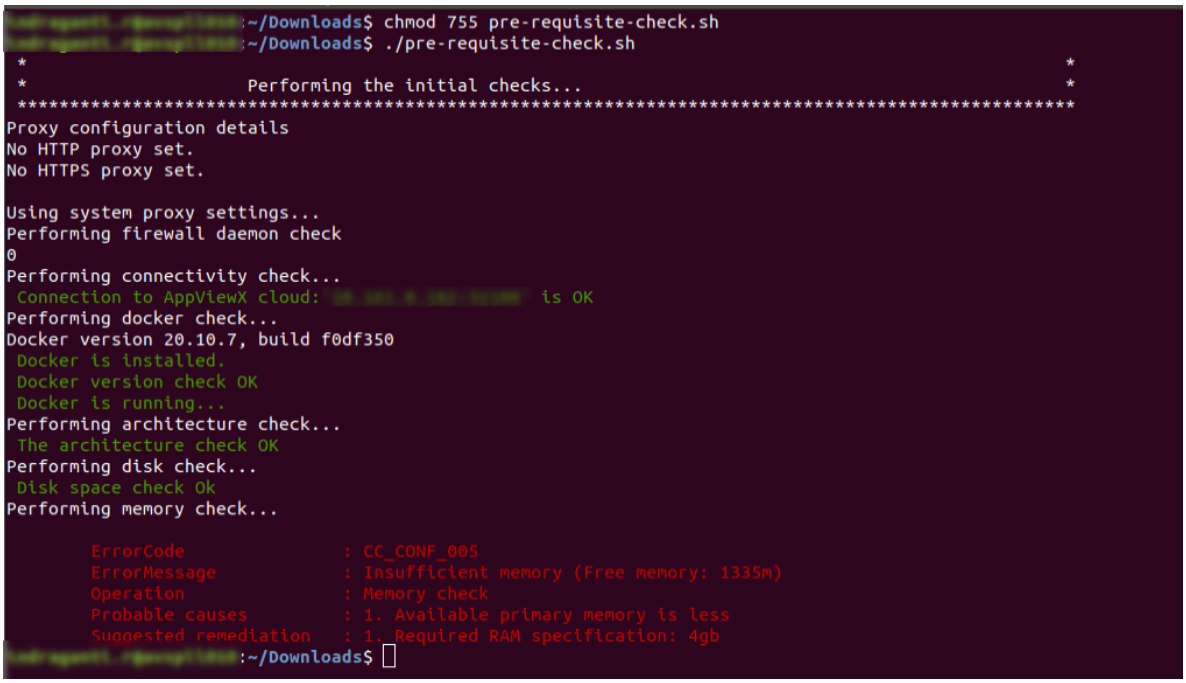
      From this list, for **Executing the Prerequisites Check Script**, to download the script, click .
      The **pre-requisite-check.sh** script file is downloaded.

   b. Securely copy the **pre-requisite-check.sh** via SCP/SFTP to the host machine where the AppViewX Cloud Connector is to be installed

   c. Convert the downloaded script file into an executable file using the chmod command, as shown below:`chmod 755 pre-requisite-check.sh`

   d. Execute the **.sh** prerequisite check script file: `./pre-requisite-check.sh`
      If the node does not meet the prerequisites for the AppViewX Cloud Connector installation, the output of the command returns an error code and the corresponding error message, causes, and fixes, if any.

      For example, as seen in the sample output in the image below, the prerequisite check for the memory requirement has failed.



Note: For resolutions to the prerequisite check failure scenarios, click here.

7. Click **Next**.
   You will be navigated to the **AssignData Center** screen, where, for deploying the AppViewX Cloud Connector, you can either select an existing data center or add a new one.
8. To use an existing data center, select one from the options displayed on the **Assign Data Center** screen.

   Tip: Alternatively, you can use the **Search…** field on this screen to search for an existing data center.

   To add a new data center:

   a. Click **Add Data Center.**

   b. In the **Add Data Center** dialog box, enter a name for the new data center.

   c. Click **Save**.
      The new data center will now be displayed on the **Assign Data Center** screen along with the other existing data centers.

d. Select the required data center.

> **Tip:** The **Data Center based routing** section to the right of the **Assign Data Center** screen explains the concept of data center-based routing and how you can achieve high availability. To read more on this, click **Learn More** from the top-right corner of this screen.

9. Click **Next**.
   The **Advanced Configuration** screen is displayed.
10. On the **Advanced Configuration** page, to configure the TLS authentication and proxy server settings for your cloud connector:

    a. Enter/Select the advanced configuration settings for the AppViewX Cloud Connector.

> **Note:** The **Data center** field is auto-populated based on your selection on the **Assigning a Data Center** screen.

Field	Description
TLS Authentication	**Tip:** The section on the right of the screen gives you a brief context of what is **TLS Authentication**. To read more, click **Learn More** (next to the **TLS Authentication** heading). • To auto-generate a TLS certificate, select **Auto-generate** (default selection). By default, the certificate is generated using the AppViewX CA. **Note:** The created certificate is available in the certificate inventory. You can: • Assign this certificate to a certificate group • Configure a certificate expiry alert for this certificate group from the **Server Certificate** dashboard, using the **Certificate Summary Report** widget settings • To enter details of a custom TLS certificate, select **Custom**. The **TLS Certificate Password** and **Custom TLS Certificate** fields are displayed. The instructions for filling these fields are given below.

Field	Description
TLS Certificate Password*	**Note:** This field is displayed only if you have selected to enter details of a Custom TLS certificate in the **TLS Authentication** field. Password of the TLS certificate (that will be uploaded in the next step) **Note:** This is a mandatory field if a Custom TLS certificate is uploaded. AppViewX supports only password-protected Custom TLS certificates.
TLS Certificate	**Note:** This field is displayed only if you have selected to enter details of a Custom TLS certificate in the **TLS Authentication** field. To upload a custom TLS certificate: i. To navigate to the location of the custom TLS certificate, click within the field. ii. Select the certificate file. iii. Click **Open**. iv. To upload the custom TLS certificate selected, click **Upload**. **Note:** AppViewX supports only password-protected Custom TLS Certificates.
Use proxy	**Tip:** The section on the right of the screen gives you a brief context of what is **Proxy based routing**. To read more, click **Learn More** (next to the **Proxy based routing** heading). A proxy server is required if the AppViewX Cloud Connector is unable to connect to your endpoints available in the internet. To use a proxy server for the deployment:

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>
i. Select the Use proxy checkbox.

ii. To select a preconfigured proxy (for the selected data center), from the Select Proxy dropdown list, select a proxy server.

OR

To create a new proxy server setting:

i. Use the Click here option shown below the Select Proxy dropdown list.

The Add Proxy pop-up screen is displayed.

ii. Enter/Select the details required to add a proxy.

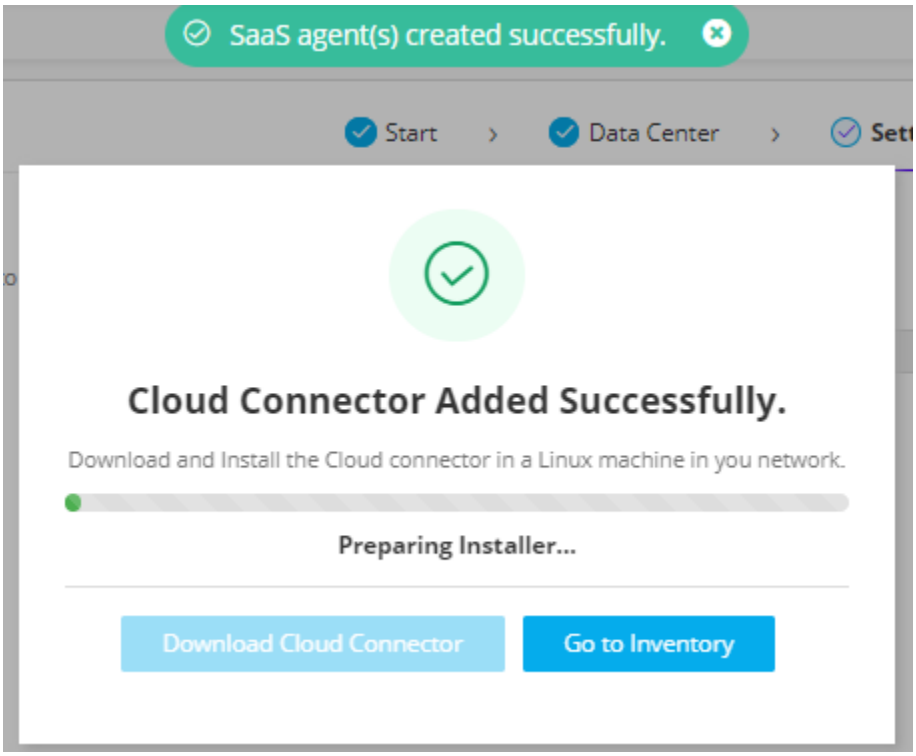
Field descriptions for the Add Proxy details
<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>*Proxy Name</td><td>Name of the proxy server</td></tr>
<tr><td>*Server IP</td><td>IP address/FQDN of the proxy server</td></tr>
<tr><td>*Port</td><td>Port number of the proxy server</td></tr>
<tr><td>URL</td><td>From the dropdown menu, select the URL.</td></tr>
<tr><td>Authentication</td><td>To enable authentication for accessing the proxy server, select this checkbox.</td></tr>
<tr><td>*Username</td><td>Note: This field is enabled only when Authentication is selected.

Enter the username required for accessing the proxy server.</td></tr>
</table>
</td></tr>
</table>

Field	Description
	<table><tr><th>Field</th><th>Description</th></tr><tr><td>***Password**</td><td>**Note:** This field is enabled only when **Authentication** is selected. Enter the password required for accessing the proxy server.</td></tr></table>

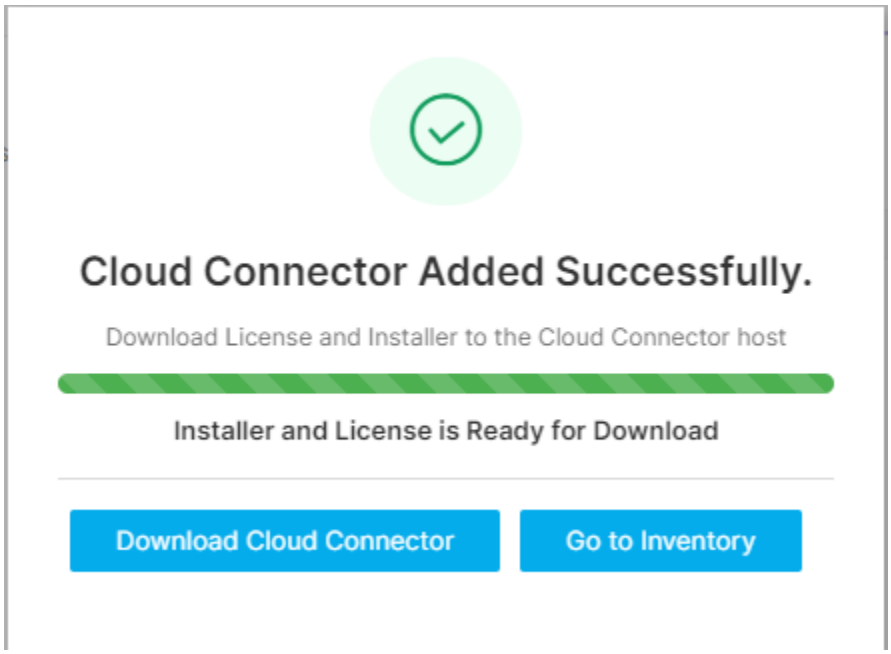
b. Click **Finish**.

A confirmation message is displayed. AppViewX begins preparing the installer that you can download and proceed with the installation of the AppViewX Cloud Connector.



11. Download the cloud connector installer and license file.

a. On the **Cloud Connector Added Successfully** dialog box, when the **Installer and License is Ready for Download**, click **Download Cloud Connector**.



> **Tip:** Alternatively, you can click **Go to Inventory** and download the installer from the AppViewX Cloud Connector inventory.

> **Tip:** You can also choose to download the license file and the installer package individually. To do this:
>
> i. From the cloud connector inventory, click the **Cloud Connector Name** (for which you want to download the license file and the installer package).
>
> The selected cloud connector's details are shown in a pane to your right.
>
> ii. To download the AppViewX Cloud Connector installer package, click **Download Cloud Connector**. This is especially useful in the event that the installer has been deleted or is no longer usable.
>
> To download the license file, click **Download License**.

b. Save the installer and the license file on the host machine.

On the **Settings :: Cloud Connector** page, details of this AppViewX Cloud Connector are added in the inventory table, which is explained here.

12. Install the AppViewX Cloud Connector Agent.

> **Note:** The following steps assume that:
> - All system prerequisites are fulfilled by the host machine.
> - The AppViewX Cloud Connector installer and license file are securely copied via SCP/SFTP to the host machine where the cloud connector is to be installed.
> - For installation on RHEL8+, the user must have **sudo** access with **read/write/execute** permissions for the following directories at the least:
>   - **/var/lib**
>   - **/etc**
>   - **/run**
>   - **/usr/local/bin**
>   - **/tmp**

a. To extract the installer, from the downloaded package, extract the tar.gz file using the command given below: `tar -zxvf <filename>.tar.gz`
For example: `tar - zxvf pesrv07-test-94-99-appviewx-appviewx-net-cloud-connector.tar.gz`

b. On the node where the AppViewX Cloud Connector agent will be installed, from the extracted installation package, run the **./install.sh** script.
The script will check if the installation prerequisites for the AppViewX Cloud Connector have been fulfilled.

> **Note:**
> Ensure that the license file is placed in the same location as the **install.sh** script. If the license file is placed in another location, run the install.sh script using the following command:
>
> ```
> ./install.sh <complete path of the license file with the filename>
> ```

On successful verification of the prerequisites, you will be prompted to specify if you want to manage f5 BIG-IP devices and if you need auto-enrollment of the certificates.

```
Do you want to manage f5 BIG-IP devices? (y/n)?:n
Continuing with the installation

Do you need Auto-enrollment of the certificate using EST/SCEP/ACME? (y/n)?:y
Please choose one or more protocol (use comma separated numbers): 1)EST(MTLS)   2)SCEP(HTTP)  3)ACME(HTTPS)
1,2,3
Auto enrollment enabled successfully for protocol(s): MTLS HTTP HTTPS
Do you want to enable Syslog receiver for a near real time configuration updates from the devices. (y/n) n
syslog enabled  n
```

c. When prompted, enter the required input value(s):

> **Important:** If you choose to **not enable** any of the following features, to enable them later, you will have to reinstall the AppViewX Cloud Connector.

i. If you want manage f5 BIG-IP devices, enter **y**/**n** for yes/no, respectively.

ii. When prompted to enable auto-enrollment of the certificate using one of the following supported auto-enrollment protocols, enter **y** only if the cloud connector is being installed in a demilitarized zone (DMZ) or devices in a restricted environment (that disables them from connecting to the **<tenant>-aep** directly.

iii. If you choose **y** (yes) here, enter the required protocol(s) name.

> **Note:** By default, the AppViewX certificate is enabled for auto-enrollment. To enable custom certificate for auto-enrollment:
>
> i. Execute the command `./avxctl upgrade gateway-cert`.
>
> ii. When prompted, enter the location of the custom certificate.

> **Note:** If you are a KUBE+ customer, the auto-enrollment gateway should be enabled as part of the installation for your KUBE+ usecases to work via the cloud connector.

d. If you want to enable Syslog receiver for a near-real time configuration updates from the devices, enter **y**/**n** for yes/no, respectively. For configuring Syslog reception, refer to Platform User guide section, Syslog Reception.
In case you have an older version of AppViewX on cloud and want to make use of Syslog capabilities for ADC, you must manually activate the Syslog flag by setting SYSLOG_ENABLED=`true` in the path ccpath/deps/properties.

e. Enter the sudo password.
After the relevant details have been entered, the installation proceeds. Installation logs, according to the outcome of the installation, are displayed.

Given below are the sample installation logs:

For installation on RHEL8+:

```
Moving images and k3s binary as required for k3s installation...


Triggering k3s install.....
```

```
[INFO]  Skipping k3s download and verify
[INFO]  Skipping installation of SELinux RPM
[WARN]  Failed to find the k3s-selinux policy, please install:
    dnf install -y container-selinux
    dnf install -y https://rpm.rancher.io/k3s/stable/common/centos/8/noarch/k3s-selinux-0.4-1.el8.noarch.rpm

[INFO]  Creating /usr/local/bin/kubectl symlink to k3s
[INFO]  Creating /usr/local/bin/crictl symlink to k3s
[INFO]  Creating /usr/local/bin/ctr symlink to k3s
[INFO]  Creating killall script /usr/local/bin/k3s-killall.sh
[INFO]  Creating uninstall script /usr/local/bin/k3s-uninstall.sh
[INFO]  env: Creating environment file /etc/systemd/system/k3s.service.env
[INFO]  systemd: Creating service file /etc/systemd/system/k3s.service
[INFO]  systemd: Enabling k3s unit
Created symlink /etc/systemd/system/multi-user.target.wants/k3s.service → /etc/systemd/system/k3s.service.
[INFO]  systemd: Starting k3s



k3s install success. Backing up the kubeconfig...
Importing CC base image...
unpacking docker.io/library/avx-mid-server-base:22.1.0.0 (sha256:5e1948b797dd19382f50faf06f921645337d53a1736016db81cd680c0afd7317)...done
************************************************
adding nameservers in coredns configmap

configmap/coredns configured
************************************************
Deploying the Cloud Connector...
NAME: avx-mid-server-starter
LAST DEPLOYED: Wed May 10 11:02:29 2023
NAMESPACE: cc
STATUS: deployed
REVISION: 1
NOTES:
1. It may take a couple of minutes for the Cloud Connector to be up.
  kubectl get pod --namespace cc
************************************************************************
```

```
*    Congratulations!!! The installation completed successfully.   *
*    Please wait till the Cloud Connector is up and running.      *
**********************************************************************
(1%) Cloud Connector status: Running
 Cloud Connector is up and running.
```

Note: If selinux is enabled on the node and is set to **enforcing**, the warning **Failed to find the k3s-selinux policy...** will show up in the logs. This warning can be ignored.

For installation on an OS other than RHEL8+:

```
Loaded image: rancher/k3s:v1.23.3-k3s1
Loaded image: rancher/k3d-tools:5.2.2
Loaded image: rancher/mirrored-pause:3.6
[36mINFO[0m[0000] [SimpleConfig] Hostnetwork selected - disabling injection of docker host into the cluster, server load balancer and setting the api
 port to the k3s default
[33mWARN[0m[0000] No node filter specified
[33mWARN[0m[0000] No node filter specified
[33mWARN[0m[0000] No node filter specified
[36mINFO[0m[0000] Prep: Network
[36mINFO[0m[0000] Re-using existing network 'host' (8bebb4ae61001f74487d0aa6b315396405d0127c938da1206614d113295ae139)
[36mINFO[0m[0000] Created volume 'k3d-cc-images'
[36mINFO[0m[0000] Starting new tools node...
[36mINFO[0m[0000] Starting Node 'k3d-cc-tools'
[36mINFO[0m[0001] Creating node 'k3d-cc-server-0'
[36mINFO[0m[0001] Using the k3d-tools node to gather environment information
[36mINFO[0m[0001] Starting cluster 'cc'
[36mINFO[0m[0001] Starting servers...
[36mINFO[0m[0001] Starting Node 'k3d-cc-server-0'
[36mINFO[0m[0033] All agents already running.
[36mINFO[0m[0033] All helpers already running.
[36mINFO[0m[0033] Cluster 'cc' created successfully!
[36mINFO[0m[0034] You can now use it like this:
kubectl cluster-info
Cluster setup is completed. Will start the deployment shortly...
Importing the required images...
[36mINFO[0m[0000] Importing image(s) into cluster 'cc'
[36mINFO[0m[0000] Importing images from 1 tarball(s)...
```

```
[36mINFO[0m[0000] Importing images '[/home/appviewx/CCTEST/deps/tools/mid-server-docker-image/avx-mid-server-base-22.1.0.0.tar]' into node
 'k3d-cc-server-0'...
[36mINFO[0m[0024] Successfully imported image(s)
[36mINFO[0m[0024] Successfully imported 1 image(s) into 1 cluster(s)
Import in progress...
[36mINFO[0m[0000] Importing image(s) into cluster 'cc'
[36mINFO[0m[0000] Importing images from 1 tarball(s)...
[36mINFO[0m[0000] Importing images '[/home/appviewx/CCTEST/deps/tools/mid-server-docker-image/k3d-tools-5.2.2.tar]' into node 'k3d-cc-server-0'...
[36mINFO[0m[0005] Successfully imported image(s)
[36mINFO[0m[0005] Successfully imported 1 image(s) into 1 cluster(s)
Import in progress...
[36mINFO[0m[0000] Importing image(s) into cluster 'cc'
[36mINFO[0m[0000] Importing images from 1 tarball(s)...
[36mINFO[0m[0000] Importing images '[/home/appviewx/CCTEST/deps/tools/mid-server-docker-image/rancher-mirrored-coredns-coredns-1.8.6.tar]' into
 node 'k3d-cc-server-0'...
[36mINFO[0m[0007] Successfully imported image(s)
[36mINFO[0m[0007] Successfully imported 1 image(s) into 1 cluster(s)
[36mINFO[0m[0000] Importing image(s) into cluster 'cc'
[36mINFO[0m[0000] Importing images from 1 tarball(s)...
[36mINFO[0m[0000] Importing images '[/home/appviewx/CCTEST/deps/tools/mid-server-docker-image/rancher-local-path-provisioner-v0.0.21.tar]' into
 node 'k3d-cc-server-0'...
[36mINFO[0m[0004] Successfully imported image(s)
[36mINFO[0m[0004] Successfully imported 1 image(s) into 1 cluster(s)
[36mINFO[0m[0000] Importing image(s) into cluster 'cc'
[36mINFO[0m[0000] Importing images from 1 tarball(s)...
[36mINFO[0m[0000] Importing images '[/home/appviewx/CCTEST/deps/tools/mid-server-docker-image/rancher-mirrored-pause-3.6.tar]' into node
 'k3d-cc-server-0'...
[36mINFO[0m[0003] Successfully imported image(s)
[36mINFO[0m[0003] Successfully imported 1 image(s) into 1 cluster(s)
Deploying the Cloud Connector...
NAME: avx-mid-server-starter
LAST DEPLOYED: Mon May 30 15:51:13 2022
NAMESPACE: cc
STATUS: deployed
REVISION: 1
NOTES:
1. It may take a couple of minutes for the Cloud Connector to be up.
```

```
 kubectl get pod --namespace cc
**********************************************************************
*    Congratulations!!! The installation completed successfully.   *
*    Please wait till the Cloud Connector is up and running.       *
**********************************************************************
(1%) Cloud Connector status: Running
[32m Cloud Connector is up and running. (B[m
```

Troubleshooting: For installation errors, refer to the Troubleshooting section.

The AppViewX Cloud Connector consists of two important components—the starter plugin and the platform. The starter plugin component is installed along with the AppViewX Cloud Connector, in the same installation process.

When installed, the starter plugin is used to initiate the download of the platform component. The platform component is used to host business use cases related to the AppViewX Cloud Connector.

When the platform component download is in progress, it is indicated by the symbol prefixed to the platform component version number in the AppViewX Cloud Connector inventory details 21.1.0.0 . A completed download/upgrade is indicated by the symbol in the same location 21.1.0.1 .

Note: Based on the internet bandwidth and the number of cloud connectors being installed, the downloading of the cloud connector may vary between 5 to 15 minutes.

13. To approve the cloud connector installation:

    a. Go to (**Menu**) > **Platform** > **Connectivity** > **Cloud Connector**.
    The **Settings :: Cloud Connector** inventory page is displayed.

    b. For the cloud connector instance installed, from the **Actions** column, click **Approve**.

Troubleshooting: If the AppViewX Cloud Connector instance has been approved but is not moved to the **Running** state, you can check the pod status and/or restart the pod(s), as required.

Managing ADC Devices

The following ADC vendors can be managed within the AppViewX cloud environment:

- F5
- AVI
- NginxPlus
- A10
- Citrix
- HAProxy
- BigIQ
- Array
- F5XC

To manage F5 BIG-IP and A10 devices and to handle certificates through the AppViewX Cloud Connector, ensure the following prerequisites are met:

- Prerequisites for F5 BIG-IP Devices
- Prerequisites for A10 Devices

Prerequisites for F5 BIG-IP Devices

To manage certificates on F5 BIG-IP devices, follow the steps below:

1. Ensure you have a Licensed version of the iControl jar for F5 BIG-IP devices (Refer: https://devcentral.f5.com/s/articles/iControl-Library-For-Java-With-Source)
2. Download the axis.jar from the axis library.
3. In the AppViewX Cloud Connector installation package, copy the **iControl jar** and the **axis jar** to the folder **/deps/external_libs**.

Note: If the pre-requisite libraries are placed manually to the installed/upgraded Cloud Connector, then restart the pods by executing the commands below.

```
cd <cc_installation_directory>/deps/tools
```

```
./k3s kubectl delete pods -A --all --force
```

Note: Check with the Customer Support team that the iControl jar has been uploaded to the AWS instance. Without these two upload operations, F5 functionalities will fail with the following error

message: **The pre-requisite library required for managing the F5 vendor is not configured. Please contact the system admin for more details.**

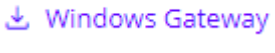
Prerequisites for A10 Devices

For certificate management and device backup on A10 devices, navigate to the **Authentication Settings** and configure the **Node Password** to match the password of the node where the AppViewX Cloud Connector instance is deployed.

For detailed instructions on how to configure the Authentication Settings, click here.

Installing the AppViewX Windows Gateway

To integrate Microsoft IIS servers and Microsoft CAs with the AppViewX SaaS, you will need to install the AppViewX Windows Gateway.

- To download the AppViewX Windows Gateway, from the AppViewX CERT+ landing page, under **Get started with CERT+** > **Connectivity**, click 

.
- To install and manage the AppViewX Windows Gateway, refer to the **AppViewX Windows Gateway Setup Guide**.

Troubleshooting the AppViewX Cloud Connector

- Managing Certificates on F5 BIG-IP Devices
- AppViewX Cloud Connector Health
- Connectivity Checks
- Installation Errors
- Log Analysis
- Steps to check pod status
- Steps to restart pods

Managing Certificates on F5 BIG-IP Devices

- In the event of not being able to manage certificates on an F5 BIG-IP device, ensure that the iControl jar is copied and restart the necessary services using the command given below:

```
./deps/tools/k3s kubectl rollout restart deployment avx-mid-server-platform -n cc
```

AppViewX Cloud Connector Health

Amber/red health indicator

- Check if the AppViewX Cloud Connector is up and running.
  - If no, check if there is a connectivity issue due to:
    - Firewall policies
    - Network configuration changes at the tenant's and/or AppViewX's end
  - If the AppViewX Cloud Connector is up and running, check the health indicators to determine if the traffic to the AppViewX Cloud Connector is configured correctly.
    - If the health indicator is amber/red:
      - Check if the AppViewX Cloud Connector is up and running.
      - If yes, validate the connectivity from the AppViewX Cloud Connector node to the AppViewX SaaS.

Connectivity Checks

- Scenario 1: At the time of installation
  - Check if the AppViewX Cloud Connector is able to reach the AppViewX cloud.
- Scenario 2: After the package has been successfully installed
  - Check the AppViewX Cloud Connector's health indicator.
  - If the health indicator is amber/red:
    - Check if the AppViewX Cloud Connector is up and running in the network .
    - If yes, validate the connectivity from the AppViewX Cloud Connector node to the AppViewX SaaS.

Installation Errors

Prerequisite check failure

At the time of the package installation, check for the following prerequisites:

- Hardware
  - Check if the current hardware configuration is according to the prerequisites.
- Connectivity
  - Check the firewall policies, proxy settings, and network configuration settings. Refer to the firewall and network-related prerequisites.
- OS Version
  - Check if the current system configuration is according to the prerequisites.
- Docker installation
  - Check if the current configuration is according to the prerequisites.

> **Note:** Since RHEL8+ excludes Docker support, Docker prerequisites are not applicable when the AppViewX Cloud Connector is being installed on a RHEL 8+ node.

SHA256 checksum failure

Cross check the SHA256 checksum in the AppViewX Cloud Connector inventory with the SHA256 checksum in the installer package.

To view the SHA256 checksum in the installer package, execute the command given below:

```
sha256sum <absolute path of the installer package file>
```

Installation Errors

Full list of installation error codes and their resolutions

Error Code	Error Message	Resolution
CC_CONF_001	Improper docker version (Docker version currently installed:	Ensure that the installed version of the Docker is 20.10.5 or above.

Full list of installation error codes and their resolutions (continued)

Error Code	Error Message	Resolution
	<current version number>)	
CC_CONF_002	Incompatible system architecture	Ensure the operating system on the node complies with the following prerequisites: • x86 64 bit
CC_CONF_003	Failed to establish connection to AppViewX cloud	Check the firewall policies and proxy settings in the tenant premises.
CC_CONF_004	Docker is not installed.	• Install Docker with non sudo access. • Required configuration: version 20.10.5 or higher • For instructions for installing the Docker Engine, click here. • For post-installation steps for Linux:, click here. • In the event of a VM reboot, the Docker needs to be restarted. To configure the Docker to restart on boot, follow the instructions given here.
CC_CONF_005	Insufficient memory (Free memory: <available memory>)	Required RAM specification: 8GB
CC_CONF_006	Disk space available is low: <available disk space in MB>	Minimum available disk space required: 16GB
CC_CONF_007	Docker not running or not accessible for non sudoers	1. To check the Docker status, execute **one** of the following commands: • `service docker status` • `systemctl status docker` 2. To start the Docker, execute **one** of the following commands:

Full list of installation error codes and their resolutions (continued)

Error Code	Error Message	Resolution
		• service docker start • systemctl start docker 3. Ensure that the Docker is accessible to non sudoers. • For post-installation steps for Linux: https://docs.docker.com/engine/install/linux-postinstall/ • In the event of a VM reboot, the Docker needs to be restarted. To configure the Docker to restart on boot, follow the instructions given here.
CC_CONF_008	Cluster already exists.	1. Uninstall the AppViewX Cloud Connector. 2. Reinstall the AppViewX Cloud Connector in the same/different node.

CC_CONF_009	firewalld is running	• Execute the following script to open the port in firewalld that requires sudo access:
		`./deps/utils/open-ips-ports-firewalld.sh`
		OR
		• Execute the following commands:
		`sudo firewall-cmd --permanent --add-port=22/tcp`
		`sudo firewall-cmd --permanent --add-source=10.42.0.0/16`
		`sudo firewall-cmd --permanent --add-source=10.43.0.0/16`
		`sudo firewall-cmd --direct --permanent --add-rule ipv4 filter FORWARD 1 -m state --state NEW,RELATED,ESTABLISHED -j ACCEPT`
		`sudo firewall-cmd --permanent --add-forward-port=port=30020:proto=tcp:toport=30020:toaddr=`
		`sudo firewall-cmd --permanent --add-forward-port=port=30021:proto=tcp:toport=30021:toaddr=`

Full list of installation error codes and their resolutions (continued)

Error Code	Error Message	Resolution
		sudo firewall-cmd --permanent --add-forward-port=port=30022:proto=tcp:toport=30022:toaddr= sudo firewall-cmd --reload
CC_CONF_010	Not met cpu requirement:- No of available processors(vCPU): <number>	The required number of processors (vCPU) is 4.
CC_CONF_011	Docker running with Incompatible storage Driver	Update the storage driver to **overlay2**. For setup instructions, click here.
CC_CONF_012	A default route is not available in the tenant premises or the tenant is not connected to a network.	• Add a default route with an IP address. • Ensure that the network connection is up.
CC_CONF_013	Cluster already exists	Uninstall the AppViewX Cloud Connector.
CC_CONF_015	Low available disk space	Minimum available (free) disk space required for **/var/lib**: **5 GB** In case of restrictions in meeting this requirement, it is recommended to change the data root directory from **/var/lib** to another dedicated directory. For instructions on changing the data root directory, click here.
CC_CONF_016	Low available disk space	Minimum available (free) disk space required for **/run**: **3 GB** **Note:** Currently, in the event that this requirement is not met, a workaround is not available.

Full list of installation error codes and their resolutions (continued)

Error Code	Error Message	Resolution
CC_CONF_017	`systemctl` command not found	Ensure that your operating system has `systemctl` and `systemd` installed.
CC_CONF_021	Local IP is configured as Nameserver	1. On the command line terminal, execute the following command: `cat /etc/resolv.conf` Contents of the resolv.conf file are displayed, which include the nameserver IP addresses. 2. Check the nameserver IP address(es) in the resolv.conf file. **Scenario 1**: If the nameserver IP address is other than 127.0.0.*: a. From the resolv.conf file, copy one (or more) IP address(es). b. On the command line terminal of the node where the cloud connector is installed, navigate to the location: **(Cloud_Connector_Installed_Folder)/deps/tools**. c. To edit the config map of the coredns pod, execute the following command: `./k3s kubectl edit cm coredns -n kube-system` d. Replace the forward IP address in the coredns pod config map with the IP address(es) copied from the resolv.conf file. e. Restart the coredns pod: ./k3s kubectl delete pod <COREDNS_POD_NAME> -n kube-system. **Scenario 2**: Nameserver IP address in the resolv.conf file is 127.0.0.*: a. Execute the following command: `/run/systemd/resolve/resolv.conf`. b. Now, check the nameserver IP address(es).

Full list of installation error codes and their resolutions (continued)

Error Code	Error Message	Resolution
		Scenario 2.1: If the nameserver IP address is other than 127.0.0.*: i. Copy the IP address(es). ii. On the command line terminal of the node where the cloud connector is installed, navigate to the location: **(Cloud_Connector_Installed_Folder)/deps/tools**. iii. To edit the config map of the coredns pod, execute the following command: `./k3s kubectl edit cm coredns -n kube-system` iv. Replace the forward IP address in the coredns pod config map with the IP address(es) copied from the resolv.conf file. v. Restart the coredns pod: `./k3s kubectl delete pod <COREDNS_POD_NAME> -n kube-system`. **Scenario 2.2**: If the /run/systemd/resolve directory is not created: i. On the command line terminal of the node where the cloud connector is installed, navigate to the location: **(Cloud_Connector_Installed_Folder)/deps/tools**. ii. To edit the config map of the coredns pod, execute the following command: `./k3s kubectl edit cm coredns -n kube-system` iii. Replace the forward IP address in the coredns pod config map with the location: /etc/resolv.conf. iv. Restart the coredns pod: `./k3s kubectl delete pod <COREDNS_POD_NAME> -n kube-system`.

Full list of installation error codes and their resolutions (continued)

Error Code	Error Message	Resolution
CC_INSTALL_001	Moving images and k3s binary failed	• Ensure that the user has **write** permission to the **/var/lib** and the **/usr/local/bin** directories. • Ensure that the sha256sum of the downloaded installer package matches with that of the corresponding cloud connector in the AppViewX Cloud Connector inventory. In case of a mismatch, redownload the package.
CC_INSTALL_002	Triggering k3s install failed	• Ensure that the sha256sum of the downloaded installer package matches with that of the corresponding cloud connector in the AppViewX Cloud Connector inventory. In case of a mismatch, redownload the package
CC_INSTALL_003	Setting Kube config failed	Uninstall and reinstall the AppViewX Cloud Connector.
CC_INSTALL_004	Importing CC base image failed	• Uninstall and reinstall the AppViewX Cloud Connector. • Ensure that the sha256sum of the downloaded installer package matches with that of the corresponding cloud connector in the AppViewX Cloud Connector inventory. In case of a mismatch, redownload the package
CC_PLATFORM_001	Failed to untar the upgrade dependencies	For SRE:

Full list of installation error codes and their resolutions (continued)

Error Code	Error Message	Resolution
		1. Retry the upgrade operation once. 2. If the upgrade operation fails, reinstall the Cloud Connector.
CC_PLATFORM_002	Upgrade operation failed	For SRE: Check user logs for cause of operation failure (insufficient disk and/or memory). • If the cause of failure is insufficient disk and/or memory space, direct the tenant to free disk and/or memory. • If not, reinstall the Cloud Connector.

Troubleshooting: If your error remains unresolved even after executing the above troubleshooting steps, email the AppViewX Technical Support team at help@appviewx.com or call them at +1 (212) 390 1644.

Log Analysis

For troubleshooting common error scenarios, the AppViewX Cloud Connector's logs can be analyzed to identify the cause and the solution required, therefore.

Within the installation directory, the AppViewX Cloud Connector logs can be accessed at **./deps/logs/cloud-connector.log**.

AppViewX comes with a set of commands required for troubleshooting based on log analysis.

To troubleshoot based on log analysis, you can use the `kubectl` commands as usual. You can also use the k3s available in the **./deps/tools folder**. Using k3s, you can fire `kubectl` commands in the form k3s kubectl.

Syntax:

```
<kubectl-command-parameters>
```

- To list all the pods and their status, use the following syntax:

  `./deps/tools/k3s kubectl get pods -A`

- To delete a pod and then restart it, use the following syntax:

  `./deps/tools/k3s kubectl delete pods <pod-id> -n <name-space>`

- To describe the pods, use the following syntax:

  `./deps/tools/k3s kubectl describe pods <pod-id> -n <name-space>`

- To check the logs via `kubectl` command for k3s related pods, use the following syntax:

  `./deps/tools/k3s kubectl logs <pod-id> -n kube-system`

Note: The k3s cluster created by the AppViewX Cloud Connector is called cc.

Troubleshooting: If your error remains unresolved even after executing the above troubleshooting steps, email the AppViewX Technical Support team at help@appviewx.com or call them at +1 (212) 390 1644.

Steps to check pod status

- Navigate to `deps/tools/`
- Execute the command `./k3s kubectl get pods -n cc`

```
-bash-4.2$ cd deps/tools/
-bash-4.2$ pwd
/home/appviewx/Dec8/deps/tools
-bash-4.2$ ./k3s kubectl get pods -n cc
NAME                                    READY   STATUS    RESTARTS   AGE
avx-mid-server-starter-5b8c7d4c49-bhhsq   1/1     Running   0          2d
avx-mid-server-platform-5754947f99-hx7q6   1/1     Running   0          2d
-bash-4.2$
```

Expected result:

There should be 2 pods `avx-mid-server-starter` and `avx-mid-server-platform` and both should be in running state.

Pod Description:

`avx-mid-server-starter`: Responsible for startup of the CC, this downloads the required artifacts from AppViewX servers to CC nodes during startup. Post successful startup, this pod does not play any

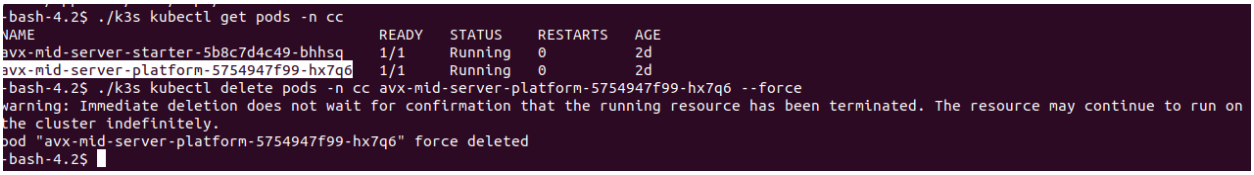
significance. Restarting the pod would download the artifacts once again and upgrade to the latest cloud connector changes from the server.

`avx-mid-server-platform`: Responsible for all device communication, it checks with AppViewX SaaS servers, and if there are any actions that need to be performed and if it finds that actions have to be performed (for example **Discovery**, **Device Addition**, **Certificate push**, and so on), then the commands will be executed on the end device from this pod and response will be sent back to AppViewX servers.

Steps to restart pods

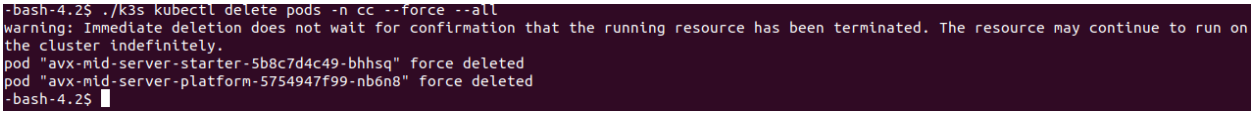
Restart specific pod:

1. Navigate to `deps/tools/`
2. Execute the command `./k3s kubectl delete pods -n cc <podname> --force`



Restart both starter and platform pods:

1. Navigate to `deps/tools/`
2. Execute the command `./k3s kubectl delete pods -n cc --force --all`



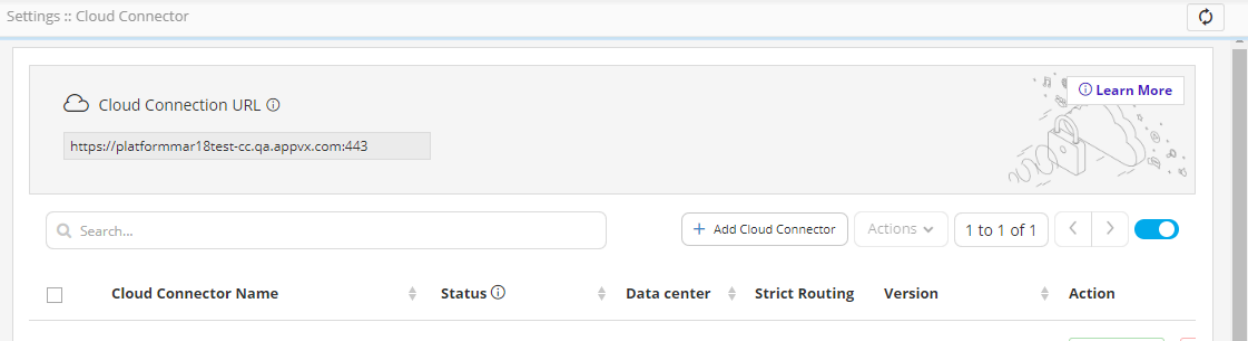
Managing the AppViewX Cloud Connector

To help you work with and manage the AppViewX Cloud Connector, this section introduces you to the Cloud Connector inventory and outlines the steps for performing the various AppViewX Cloud Connector actions, uninstalling the AppViewX Cloud Connector, as well as monitoring its health.

- Understanding the AppViewX Cloud Connector Inventory
- AppViewX Cloud Connector Actions
- Monitoring the Health of the AppViewX Cloud Connector

Understanding the AppViewX Cloud Connector Inventory

The AppViewX Cloud Connector inventory details page displays important information pertaining to the AppViewX Cloud Connector. The page provides easy access to functions that let you add a new AppViewX Cloud Connector and perform configuration actions like starting, pausing, and deleting the AppViewX Cloud Connector instance and so on.



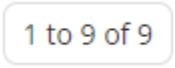
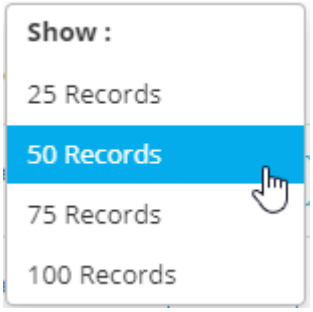
Elements and Fields in the AppViewX Cloud Connector Inventory

Element/Field	Description
Cloud Connection URL	AppViewX cloud URL of the server (internal) that hosts the AppViewX instance of the AppViewX Cloud Connector
Learn More (in the banner)	Displays a quick introduction to the AppViewX Cloud Connector using a graphical representation of how the infrastructure is deployed and works. The diagram is followed by a brief description of the key terms related to the cloud connector setup.
Search	 To search for a AppViewX Cloud Connector entry: 1. In the **Search** field, enter the value you want to filter the records for. 2. Press **Enter**. The **Settings :: Cloud Connector** page is updated to show details of only those records that match the search criteria.

Elements and Fields in the AppViewX Cloud Connector Inventory (continued)

Element/Field	Description
Add Cloud Connector	To add a new AppViewX Cloud Connector, click + Add Cloud Connector. **Note:** For steps on adding a cloud connector: • Via the native OS, click here • Via a virtual image, click here
Actions	**Note:** This button is enabled only when one or multiple AppViewX Cloud Connectors are selected. AppViewX lets you perform the following actions on a AppViewX Cloud Connector: • Start • Pause • Upgrade • Update config • Delete To perform these actions, click Actions. **Note:** For detailed steps to perform each of the above listed actions, refer to the AppViewX Cloud Connector Actions.

Elements and Fields in the AppViewX Cloud Connector Inventory (continued)

Element/Field	Description
	For easier viewing of records, AppViewX lets you set the record count preference, which is the number of records that will be displayed on one page. To set the record count preference: 1. Click  . 2. From the **Show** menu displayed, select your record count preference (for example, 50 records).  The **Settings :: Cloud Connector** page is updated according to the record count preference selected. The message, **Record count preference saved successfully**, is displayed. The UI control is also updated to display the current selection.
	If the cloud connector entries span more than one page, use this control to navigate between the pages in the cloud connector inventory.
	If enabled, the **Auto Refresh** feature automatically refreshes the AppViewX Cloud Connector inventory details every 5 seconds. To enable this feature, use the  key.

Elements and Fields in the AppViewX Cloud Connector Inventory (continued)

<table>
<tr><th>Element/Field</th><th>Description</th></tr>
<tr><td></td><td>Tip: Enabling Auto Refresh gives you a real-time status update of the AppViewX Cloud Connector's health, therefore facilitating for timely troubleshooting in the event that it is required.</td></tr>
<tr><td>Cloud Connector Name</td><td>This field displays the following two details:
• Name assigned to the AppViewX Cloud Connector when it is added
• Health status of the AppViewX Cloud Connector</td></tr>
<tr><td>Status</td><td>This field has the following values:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Waiting for response</td><td>After the AppViewX Cloud Connector is registered, the package must be downloaded and installed.
This status indicates that this installation is pending.</td></tr>
<tr><td>Waiting for approval</td><td>After the AppViewX Cloud Connector is installed, the admin must approve/reject the installation by clicking the Approve / Reject buttons from the Action field.
This status indicates that the admin's response to the installation is pending.</td></tr>
<tr><td>Running</td><td>The AppViewX Cloud Connector has been approved by the admin and is running.</td></tr>
</table></td></tr>
</table>

Elements and Fields in the AppViewX Cloud Connector Inventory (continued)

<table>
<tr><th>Element/Field</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Paused</td><td>The AppViewX Cloud Connector has been approved by the admin but is paused.

Note: The AppViewX Cloud Connector is paused when it has to undergo maintenance and/or troubleshooting.</td></tr>
</table>
</td></tr>
<tr><td>Data Center</td><td>Physical location where the AppViewX Cloud Connector system is hosted</td></tr>
<tr><td>Strict Routing</td><td>To enable strict data center-based routing, turn on the toggle under Strict Routing.</td></tr>
<tr><td>Version</td><td>Version of the AppViewX Cloud Connector platform component

Note: If a new version of the AppViewX Cloud Connector platform component is available, the Upgrade button is displayed for that AppViewX Cloud Connector.</td></tr>
<tr><td>View Log</td><td>To view the activity log for a AppViewX Cloud Connector, click View for that AppViewX Cloud Connector.</td></tr>
<tr><td>Action</td><td>This field lets you perform the following actions for a AppViewX Cloud Connector

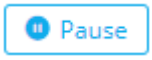
• Pause a running AppViewX Cloud Connector

• Start a paused AppViewX Cloud Connector

• Approve a AppViewX Cloud Connector

• Reject a AppViewX Cloud Connector</td></tr>
</table>

Elements and Fields in the AppViewX Cloud Connector Inventory (continued)

Element/Field	Description
	This field displays an action that can be performed for the AppViewX Cloud Connector, depending on the current status of the AppViewX Cloud Connector. For example, if the connector is running, this field shows the  button. **Remember:** Only the admin user, with the modify permission, can approve/reject a AppViewX Cloud Connector.
TLS Certificate	If a custom TLS certificate has been uploaded at the time of adding the AppViewX Cloud Connector, this field displays the common name and other details (for example, validity) of the custom TLS certificate.
Proxy	Details of the proxy server, if it has been used for the deployment
Last Heartbeat	Timestamp of the latest health analysis of the AppViewX Cloud Connector
Registered On	Timestamp of the AppViewX Cloud Connector installation
SHA256 Checksum	Details of the SHA256 token It ensures that the downloaded AppViewX Cloud Connector package is the same as the checksum displayed in the AppViewX Cloud Connector inventory page.

AppViewX Cloud Connector Actions

Key actions that can be performed include:

- Starting the AppViewX Cloud Connector
- Pausing the AppViewX Cloud Connector
- Upgrading the AppViewX Cloud Connector Version

Starting the AppViewX Cloud Connector

After the admin user has approved its installation, you need to 'start' the AppViewX Cloud Connector—you need to enable the AppViewX Cloud Connector to route traffic between the internal network and the AppViewX cloud.

To start a AppViewX Cloud Connector after it has been approved or paused:

1. Navigate to the AppViewX Cloud Connector inventory.
2. Select the checkbox for the AppViewX Cloud Connector you want to start.
3. Click 

.
4. From the menu displayed, select **Start**.
   The AppViewX Cloud Connector Status is set to **Running**.

Pausing the AppViewX Cloud Connector

The AppViewX Cloud Connector can be paused for regular maintenance or troubleshooting. Pausing the AppViewX Cloud Connector will pause all activities that have to be performed in the network premises—for example, discovering and scanning certificates, accessing endpoints within the network, and so on.

To pause the AppViewX Cloud Connector:

1. Navigate to the AppViewX Cloud Connector inventory.
2. Select the checkbox for the AppViewX Cloud Connector you want to pause.
3. Click 

.
4. From the menu displayed, select **Pause**.
   The AppViewX Cloud Connector Status is set to **Paused**.

Upgrading the AppViewX Cloud Connector Version

AppViewX provides a seamless CI/CD pipeline to capture the AppViewX Cloud Connector versioning and upgrades on the release portal. If a new version of the AppViewX Cloud Connector component is

available, you will receive a notification with the name of the cloud connector that is ready for upgrade and the Upgrade button will be displayed for that AppViewX Cloud Connector.

1. Navigate to the AppViewX Cloud Connector inventory.
2. Select the checkbox for the AppViewX Cloud Connector you want to upgrade.
3. Click Actions.
4. From the menu displayed, select **Upgrade**.
   The **Confirmation message** dialog box is displayed. It shows the current version as well as the upgraded version number, and the checksum of the binaries that will be uploaded.
5. (Optional) To view the checksum of the binaries that will be uploaded as part of the cloud connector's version upgrade, click **Checksum of the binaries uploaded**.
6. (Optional) Validate the binaries.
7. Click**Upgrade**.
   - When the upgrade starts, the **Upgrade in progress** status is displayed and a notification informing that the upgrade is in progress is received.
   - If the AppViewX Cloud Connector version is upgraded successfully:
     - The symbol is prefixed to the version number.
     - You will receive an upgrade successful notification.
   - If the upgrade fails:
     - You will receive an upgrade failure notification.
     - The **Upgrade** button is displayed again.
       - When an upgrade fails, internally, AppViewX attempts to rollback to the previous successful version of the AppViewX Cloud Connector. When the rollback is successful, the **Upgrade** button is displayed again so that you can retry the version upgrade.

     OR

     - The status is updated to **Upgrade failed**.
       - The upgrade is marked as failed if the rollback is unsuccessful. In this case, please reach out to saashelp@appviewx.com for further assistance.

> **Note:** Based on the internet bandwidth and the number of cloud connectors being installed, the downloading of the cloud connector may vary between 5 to 15 minutes.

Updating the Certificate Configuration

If a new certificate has been pushed to the AppViewX Cloud Connector, you scan the selected AppViewX Cloud Connectors to display the updated details in the AppViewX Cloud Connector inventory. To update the certificate configuration for a AppViewX Cloud Connector:

1. Select the checkbox for the required AppViewX Cloud Connector.
2. Click Actions.
3. From the menu displayed, select **Update Config**.
   On successful update of the certificate configuration, the message **Update config triggered successfully** is displayed.

Deleting an AppViewX Cloud Connector Instance

The AppViewX Cloud Connector instance may have to be deleted in events like if there is a fault with the system on which the instance is installed or if it is a faulty installation.

> **Warning:** Deleting a AppViewX Cloud Connector instance without having a backup node will result in traffic blockage.

1. Select the checkbox for the AppViewX Cloud Connector you want to delete.
2. Click Actions.
3. From the menu displayed, select **Delete**.
4. In the **Confirmation message** dialog box, click **Delete**.
   The selected AppViewX Cloud Connector is deleted.

> **Attention:** As mentioned in the image above, deleting the AppViewX Cloud Connector will only delete the data from AppViewX. To remove it from the host machine, you will have to uninstall the AppViewX Cloud Connector.

Uninstalling an AppViewX Cloud Connector Instance

To uninstall a AppViewX Cloud Connector instance:

1. On the node where the AppViewX Cloud Connector agent is installed, run the **uninstall.sh** script (included in the AppViewX Cloud Connector agent's download package).
2. For uninstallation on RHEL8+, when prompted, enter the sudo password.
   The AppViewX Cloud Connector instance is uninstalled.

Monitoring the Health of the AppViewX Cloud Connector

As a precautionary measure, to ensure in-time troubleshooting in the event of a failure, AppViewX enables runtime health analysis of the AppViewX Cloud Connector Connectivity Service. Accordingly, a color-coded health indicator is displayed for each AppViewX Cloud Connector.

Descriptions for the color-coded health indicators

Color of the health indicator	Description
🟢	The AppViewX Cloud Connector is working as expected.
🟠	Although there are no current problems with routing traffic to and from the AppViewX Cloud Connector, the AppViewX Cloud Connector's health needs to be checked. To resolve, refer to the Troubleshooting section.
🔴	The AppViewX Cloud Connector is not receiving traffic. The AppViewX Cloud Connector's health is analyzed for 3 to 5 minutes before it is declared to be down. To resolve, refer to the Troubleshooting section.

For details on how the health indicators are displayed, refer to the Understanding the AppViewX Cloud Connector Inventory page.

Frequently Asked Questions

- Disabling firewalld
- Docker Prerequisites
- Monitoring the Health of the AppViewX Cloud Connector
- Steps to check pod status

Disabling firewalld

- Disable the firewalld in the tenant's node (**Ubuntu**) where the AppViewX Cloud Connector is to be installed.

  To check the current status of firewalld, execute the command given below: `sudo ufw status`

  To permanently disable firewalld, execute the command given below: `sudo ufw disable`

- Disable the firewalld in the tenant's node (**CentOS** and **RedHat**) where the AppViewX Cloud Connector is to be installed.

  To check the current status of firewalld, execute the command given below: `sudo systemctl status firewalld --now`

  To permanently disable the firewalld, execute the command given below: `sudo systemctl disable firewalld --now`

  To restrict other devices from enabling the firewalld, execute the command given below: `sudo systemctl mask firewalld --now`

Docker Prerequisites

Note: Since RHEL8+ does not include Docker support, Docker prerequisites are not applicable when the AppViewX Cloud Connector is being installed on a RHEL8+ node.

- Docker version 20.10.5 or above installed with non-sudo access with basic read and write permissions

  Note: Support for rootless Docker is excluded.

  For Docker installation instructions, refer to the links below:

- For installing the Docker Engine: https://docs.docker.com/engine/install/
- For post-installation steps for Linux: https://docs.docker.com/engine/install/linux-postinstall/

Important: In the event of a VM reboot, the Docker needs to be restarted. To configure the Docker to restart on boot, follow the instructions given here.

Note: If **/var/lib** is going to be a separate mount, ensure that it has minimum 5 GB of free space.

In case of restrictions in meeting this requirement, it is recommended to change the data root directory from **/var/lib** to another dedicated directory. For instructions on changing the data root directory, click here.

- Bash shell support in the node for the installation of the AppViewX Cloud Connector Connectivity Service
- Changing the Data Root Directory

Monitoring the Health of the AppViewX Cloud Connector

As a precautionary measure, to ensure in-time troubleshooting in the event of a failure, AppViewX enables runtime health analysis of the AppViewX Cloud Connector Connectivity Service. Accordingly, a color-coded health indicator is displayed for each AppViewX Cloud Connector.

Descriptions for the color-coded health indicators

Color of the health indicator	Description
🟢	The AppViewX Cloud Connector is working as expected.
🟠	Although there are no current problems with routing traffic to and from the AppViewX Cloud Connector, the AppViewX Cloud Connector's health needs to be checked. To resolve, refer to the Troubleshooting section.
🔴	The AppViewX Cloud Connector is not receiving traffic.

Descriptions for the color-coded health indicators (continued)

Color of the health indicator	Description
	The AppViewX Cloud Connector's health is analyzed for 3 to 5 minutes before it is declared to be down. To resolve, refer to the Troubleshooting section.

For details on how the health indicators are displayed, refer to the Understanding the AppViewX Cloud Connector Inventory page.

Steps to check pod status

- Navigate to `deps/tools/`
- Execute the command `./k3s kubectl get pods -n cc`

```
-bash-4.2$ cd deps/tools/
-bash-4.2$ pwd
/home/appviewx/Dec8/deps/tools
-bash-4.2$ ./k3s kubectl get pods -n cc
NAME                                    READY   STATUS    RESTARTS   AGE
avx-mid-server-starter-5b8c7d4c49-bhhsq   1/1     Running   0          2d
avx-mid-server-platform-5754947f99-hx7q6   1/1     Running   0          2d
-bash-4.2$
```

Expected result:

There should be 2 pods `avx-mid-server-starter` and `avx-mid-server-platform` and both should be in running state.

Pod Description:

`avx-mid-server-starter`: Responsible for startup of the CC, this downloads the required artifacts from AppViewX servers to CC nodes during startup. Post successful startup, this pod does not play any significance. Restarting the pod would download the artifacts once again and upgrade to the latest cloud connector changes from the server.

`avx-mid-server-platform`: Responsible for all device communication, it checks with AppViewX SaaS servers, and if there are any actions that need to be performed and if it finds that actions have to be performed (for example **Discovery**, **Device Addition**, **Certificate push**, and so on), then the commands will be executed on the end device from this pod and response will be sent back to AppViewX servers.

Steps to restart pods

Restart specific pod:

1. Navigate to `deps/tools/`
2. Execute the command `./k3s kubectl delete pods -n cc <podname> --force`

```
-bash-4.2$ ./k3s kubectl get pods -n cc
NAME                                      READY   STATUS    RESTARTS   AGE
avx-mid-server-starter-5b8c7d4c49-bhhsq   1/1     Running   0          2d
avx-mid-server-platform-5754947f99-hx7q6  1/1     Running   0          2d
-bash-4.2$ ./k3s kubectl delete pods -n cc avx-mid-server-platform-5754947f99-hx7q6 --force
warning: Immediate deletion does not wait for confirmation that the running resource has been terminated. The resource may continue to run on
the cluster indefinitely.
pod "avx-mid-server-platform-5754947f99-hx7q6" force deleted
-bash-4.2$
```

Restart both starter and platform pods:

1. Navigate to `deps/tools/`
2. Execute the command `./k3s kubectl delete pods -n cc --force --all`

```
-bash-4.2$ ./k3s kubectl delete pods -n cc --force --all
warning: Immediate deletion does not wait for confirmation that the running resource has been terminated. The resource may continue to run on
the cluster indefinitely.
pod "avx-mid-server-starter-5b8c7d4c49-bhhsq" force deleted
pod "avx-mid-server-platform-5754947f99-nb6n8" force deleted
-bash-4.2$
```

Deploying the AppViewX OVA

The AppViewX Virtual Image is an OVA that is bundled with the software, network, and Docker prerequisites for installing the AppViewX Cloud Connector without altering the OS configuration on their systems. (The .ova file that can be downloaded from here).

Detailed instructions for the OVA deployment are documented are here.

Detailed instructions for updating the AppViewX virtual image from the AppViewX repository are documented here.

Updating the AppViewX Virtual Image from the AppViewX Repository

The AppViewX virtual image, which is used to install the AppViewX Cloud Connector, can be updated with the latest OS-level updates and security patches from the AppViewX repository.

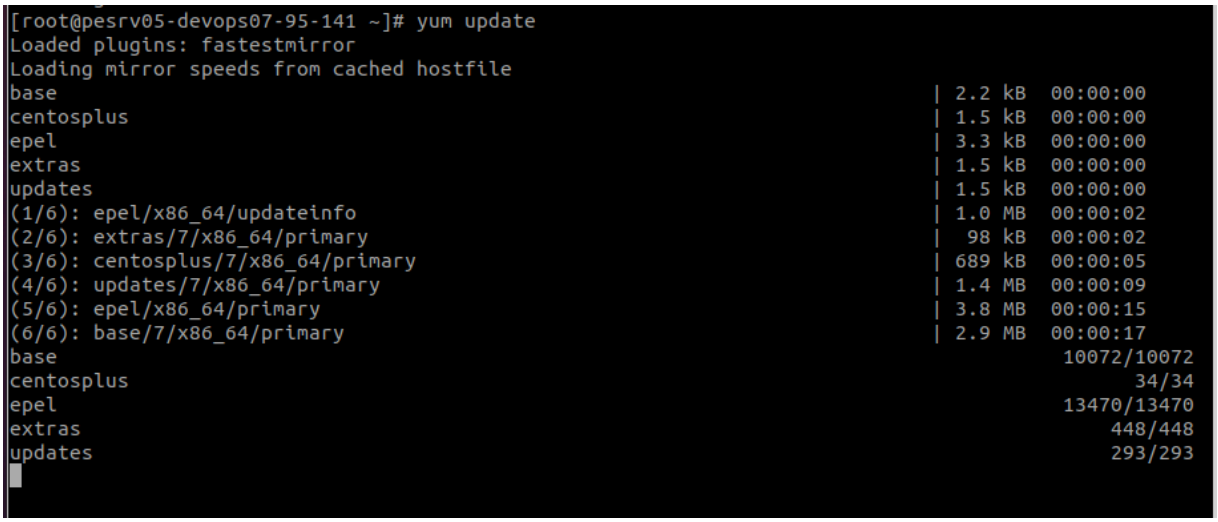
- Prerequisites for Updating the Virtual Image from the Repository
- Updating the Virtual Image from the Repository

Prerequisites for Updating the Virtual Image from the Repository

- For the AppViewX nodes, access to the following URL: https://repos.appviewx.com
- Root/sudo access to configure `yum`.

Updating the Virtual Image from the Repository

1. From the terminal, login as the root user to the AppViewX Cloud Connector OVA.
2. To get the latest updates from the AppViewX repository, execute the `yum update` command, as shown in the image below:



Synchronizing the Node Clock with the Network Time

- On the node on which the AppViewX Cloud Connector is installed, ensure that the node's clock is synchronized with the network time using NTP/PTP.

  For the **ntpd** package, execute the following sequence of commands:

```
yum install -y ntp

systemctl enable ntpd

systemctl start ntpd
```

  For the **chronyd** package, execute the following sequence of commands:

- ```
  yum install -y chrony
  systemctl enable chronyd
  systemctl start chronyd
  ```

OR

- ```
  dnf install -y chrony
  systemctl enable chronyd
  systemctl start chronyd
  ```

Validating the SHA256 Checksum

Cross check the SHA256 checksum in the AppViewX Cloud Connector inventory with the SHA256 checksum in the installer package.

To view the SHA256 checksum in the installer package, execute the command given below:

```
sha256sum <absolute path of the installer package file>
```

Appendix A: Network Scan Recommendations

This section lists the AppViewX-recommended best practices for ensuring that network scans for cloud connectors yield more accurate results, thus facilitating a more secure and compliant network infrastructure and appropriate measures for mitigating potential risks.

Selecting Ports for Scanning

- Commonly, the ports **443** and **8443** are used for scanning. Additional ports identified at the time of your onboarding can be added to the list.
- If a list of specific ports cannot be identified, a standard port scan is the next best recommendation.

> **Note:** Research suggests that 99% of open ports can be identified by scanning the top 3328 ports. For more information, click here. For the complete list of standard ports that are scanned by AppViewX, click here.

- While the all ports scan can also be used, it can be time consuming and add a significant load to the infrastructure. AppViewX recommends a lesser number of ports, so that the scan time is less and the process is more optimally completed.
- When performing larger subnet scans, for example for a /16 subnet, for throttle scanning, split larger subnets into smaller batches. For example, split the /16 subnet into its /24 equivalent.

Configuring the AppViewX Cloud Connectors' Infrastructure

- For production environments, it is recommended to have two cloud connectors per datacenter. This enables high availability within a datacenter as well as across all datacenters in the environment.
- Enable strict routing for the cloud connectors within the same datacenter so traffic can be optimally routed between the cloud connectors.
- For scanning more than a 100 subnets within a span of 24 hours, allocate additional computing resources by provisioning one cloud connector for every 100 subnets (so there'll be a total of 1000 IPs across the subnet).

Setting Batch Limits for Network Discovery

For network discovery, 10K is the maximum recommended batch limit.

Setting the Scanning Intensity

During a network scan, the AppViewX Network Plugin sends the number of packets to the IP address configured on the network scan. The load on the target network can be controlled by selecting a scanning intensity from the range 1 to 12.

Scanning intensity 1 to 4

Scanning intensities between 1 and 4 are designed to scan **less than 250 ports** or, for larger networks, common SSL ports like **443**, **8443**.

For a larger network, these intensities can take up to several days to complete scanning, especially if a **all ports** scan is triggered.

Scanning intensity 5 to 12

Scanning intensities between 5 and 12 are designed for scanning **standard ports** and **all ports**. For these higher intensities, the number of network connections increases, which then decreases the time required for scanning.

Tip: A scanning intensity in the range **4 to 6** is known to be appropriately reliable and accurate, and consumes very less bandwidth.

Warning: A scan intensity in the range **8 to 12** is known to establish high packet transmission and a higher number of connections per second. For example, **setting intensity = 12** will establish **16K connections/second**. Setting the scanning intensity to a value in this range is not recommended unless your network team confirms that your network infrastructure can handle the number of connections established.

Optimizing the Load Factor

For handling increased loads, it is preferable to adopt horizontal scaling by adding more cloud connectors. The requests are then handled using the round robin allocation method across all the available cloud connectors.

Example: For a load of 140K subnets, it is recommended to add one cloud connector for scanning a set of 17.5K subnets. Since a standard ports scan and a all ports scan yield nearly identical results, and a standard port scan is 7x faster, it is proposed to run the scan in two phases: **phase 1** will cover the standard ports and **phase 2** will cover all ports.

Cloud connector distribution across datacenters will be decided based on your IP distribution across those datacenters.

To reduce network latency, more datacenters (and cloud connectors) should be added based on your subnet topology.

Appendix B: Automated Installation without Internet

After you have entered the tenant ID, if the host machine is not connected to the internet, the following error message is displayed: **The AppViewX Cloud URL is unreachable.** followed by the prompt: **Do you want to configure the proxy? (y/n).**

To proceed with the installation:

1. When prompted **Do you want to configure the proxy? (y/n)**:, enter **y**.
2. Enter the proxy details as prompted.

Note: If you enter **n**, the automated installation of the cloud connector is skipped. You can choose to install the cloud connector using the user interface.

Appendix C: CIS Benchmarking for AppViewX Cloud Connector

This report is specific to the v1.27-v1.29 release line of K3s and the v1.8 release of the CIS Kubernetes Benchmark.

Note:

- **PASS** denotes that the controls conform to the standards.
- **WARN/FAIL** denotes that the specific control does not conform to the standard.
- **NA** is not applicable for k3s or cloud connector.

Overall Compliance	65.91%

Summary

Policy	PASS/NA	WARN/FAIL
Count	87	45

Controls	Status	Comment
Overview		
Testing controls methodology		
1.1 Control Plane Node Configuration Files		
1.1.1 Ensure that the API server pod specification file permissions are set to 600 or more restrictive (Automated)	PASS	
1.1.2 Ensure that the API server pod specification file ownership is set to root (Automated)	NA	Not applicable for k3s.
1.1.3 Ensure that the controller manager pod specification file permissions are set to 600 or more restrictive (Automated)	NA	Not applicable for k3s.
1.1.4 Ensure that the controller manager pod specification file ownership is set to root (Automated)	NA	Not applicable for k3s.

Controls	Status	Comment
1.1.5 Ensure that the scheduler pod specification file permissions are set to 600 or more restrictive (Automated)	NA	Not applicable for k3s.
1.1.6 Ensure that the scheduler pod specification file ownership is set to root (Automated)	NA	Not applicable for k3s.
1.1.7 Ensure that the etcd pod specification file permissions are set to 600 or more restrictive (Automated)	NA	Not applicable for k3s.
1.1.8 Ensure that the etcd pod specification file ownership is set to root (Automated)	NA	Not applicable for k3s.
1.1.9 Ensure that the Container Network Interface file permissions are set to 600 or more restrictive (Automated)	NA	Not applicable for k3s.
1.1.10 Ensure that the Container Network Interface file ownership is set to root (Manual)	NA	Not applicable for k3s.
1.1.11 Ensure that the etcd data directory permissions are set to 700 or more restrictive (Automated)	PASS	
1.1.12 Ensure that the etcd data directory ownership is set to etcd (Automated)	NA	Not applicable for k3s.
1.1.13 Ensure that the admin.conf file permissions are set to 600 or more restrictive (Automated)	NA	Not applicable for k3s.
1.1.14 Ensure that the admin.conf file ownership is set to root (Automated)	PASS	
1.1.15 Ensure that the scheduler.conf file permissions are set to 600 or more restrictive (Automated)	PASS	
1.1.16 Ensure that the scheduler.conf file ownership is set to root (Automated)	PASS	
1.1.17 Ensure that the controller-manager.conf file permissions are set to 600 or more restrictive (Automated)	PASS	
1.1.18 Ensure that the controller-manager.conf file ownership is set to root (Automated)	PASS	
1.1.19 Ensure that the Kubernetes PKI directory and file ownership is set to root (Automated)	PASS	

Controls	Status	Comment
1.1.20 Ensure that the Kubernetes PKI certificate file permissions are set to 600 or more restrictive (Manual)	FAIL	Currently permissions are set to 644.
1.1.21 Ensure that the Kubernetes PKI key file permissions are set to 600 (Manual)	PASS	
1.2 API Server	PASS	
1.2.1 Ensure that the --anonymous-auth argument is set to false (Manual)	PASS	
1.2.2 Ensure that the --token-auth-file parameter is not set (Automated)	PASS	
1.2.3 Ensure that the --DenyServiceExternalIPs is not set (Automated)	PASS	
1.2.4 Ensure that the --kubelet-client-certificate and --kubelet-client-key arguments are set as appropriate (Automated)	PASS	
1.2.5 Ensure that the --kubelet-certificate-authority argument is set as appropriate (Automated)	NA	Not applicable for k3s.
1.2.6 Ensure that the --authorization-mode argument is not set to AlwaysAllow (Automated)	PASS	
1.2.7 Ensure that the --authorization-mode argument includes Node (Automated)	PASS	
1.2.8 Ensure that the --authorization-mode argument includes RBAC (Automated)	PASS	
1.2.9 Ensure that the admission control plugin EventRateLimit is set (Manual)	FAIL	Not supported.
1.2.10 Ensure that the admission control plugin AlwaysAdmit is not set (Automated)	PASS	
1.2.11 Ensure that the admission control plugin AlwaysPullImages is set (Manual)	FAIL	The cloud connector does not directly import images from external registries; instead, it imports all images into the local

Controls	Status	Comment
		registry in the node before consumption.
1.2.12 Ensure that the admission control plugin SecurityContextDeny is set if PodSecurityPolicy is not used (Manual)	NA	Not applicable for k3s.
1.2.13 Ensure that the admission control plugin ServiceAccount is set (Automated)	PASS	
1.2.14 Ensure that the admission control plugin NamespaceLifecycle is set (Automated)	PASS	
1.2.15 Ensure that the admission control plugin NodeRestriction is set (Automated)	PASS	
1.2.16 Ensure that the --profiling argument is set to false (Automated)	PASS	
1.2.17 Ensure that the --audit-log-path argument is set (Automated)	NA	Not applicable for k3s.
1.2.18 Ensure that the --audit-log-maxage argument is set to 30 or as appropriate (Automated)	NA	Not applicable for k3s.
1.2.19 Ensure that the --audit-log-maxbackup argument is set to 10 or as appropriate (Automated)	NA	Not applicable for k3s.
1.2.20 Ensure that the --audit-log-maxsize argument is set to 100 or as appropriate (Automated)	NA	Not applicable for .
1.2.21 Ensure that the --request-timeout argument is set as appropriate (Manual)	NA	Not applicable for k3s.
1.2.22 Ensure that the --service-account-lookup argument is set to true (Automated)	PASS	
1.2.23 Ensure that the --service-account-key-file argument is set as appropriate (Automated)	NA	Not applicable for k3s.
1.2.24 Ensure that the --etcd-certfile and --etcd-keyfile arguments are set as appropriate (Automated)	FAIL	The etcd is not used, instead SQLite is used for cloud connector deployment. Not applicable for cloud connector deployment.

Controls	Status	Comment
1.2.25 Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate (Automated)	PASS	
1.2.26 Ensure that the --client-ca-file argument is set as appropriate (Automated)	PASS	
1.2.27 Ensure that the --etcd-cafile argument is set as appropriate (Automated)	FAIL	The etcd is not used, instead SQLite is used for cloud connector deployment. Not applicable for cloud connector deployment.
1.2.28 Ensure that the --encryption-provider-config argument is set as appropriate (Manual)	NA	Not applicable for k3s.
1.2.29 Ensure that encryption providers are appropriately configured (Manual)	NA	Not applicable for k3s.
1.2.30 Ensure that the API Server only makes use of Strong Cryptographic Ciphers (Manual)	PASS	
1.3 Controller Manager		
1.3.1 Ensure that the --terminated-pod-gc-threshold argument is set as appropriate (Manual)	FAIL	Not supported.
1.3.2 Ensure that the --profiling argument is set to false (Automated)	PASS	
1.3.3 Ensure that the --use-service-account-credentials argument is set to true (Automated)	PASS	
1.3.4 Ensure that the --service-account-private-key-file argument is set as appropriate (Automated)	PASS	
1.3.5 Ensure that the --root-ca-file argument is set as appropriate (Automated)	PASS	
1.3.6 Ensure that the RotateKubeletServerCertificate argument is set to true (Automated)	NA	Not applicable for k3s.
1.3.7 Ensure that the --bind-address argument is set to 127.0.0.1 (Automated)	PASS	

Controls	Status	Comment
1.4 Scheduler	PASS	
1.4.1 Ensure that the --profiling argument is set to false (Automated)	PASS	
1.4.2 Ensure that the --bind-address argument is set to 127.0.0.1 (Automated)	PASS	
2 Etcd Node Configuration		
2.1 Ensure that the --cert-file and --key-file arguments are set as appropriate (Automated)	FAIL	The etcd is not used, instead sqllite is used for cloud connector deployment. Not applicable for cloud connector deployment.
2.2 Ensure that the --client-cert-auth argument is set to true (Automated)	FAIL	The etcd is not used, instead sqllite is used for cloud connector deployment. Not applicable for cloud connector deployment.
2.3 Ensure that the --auto-tls argument is not set to true (Automated)	FAIL	The etcd is not used, instead sqllite is used for cloud connector deployment. Not applicable for cloud connector deployment.
2.4 Ensure that the --peer-cert-file and --peer-key-file arguments are set as appropriate (Automated)	FAIL	The etcd is not used, instead sqllite is used for cloud connector deployment. Not applicable for cloud connector deployment.
2.5 Ensure that the --peer-client-cert-auth argument is set to true (Automated)	FAIL	The etcd is not used, instead sqllite is used for cloud connector deployment. Not

Controls	Status	Comment
		applicable for cloud connector deployment.
2.6 Ensure that the --peer-auto-tls argument is not set to true (Automated)	FAIL	The etcd is not used, instead sqllite is used for cloud connector deployment. Not applicable for cloud connector deployment.
2.7 Ensure that a unique Certificate Authority is used for etcd (Automated)	FAIL	The etcd is not used, instead sqllite is used for cloud connector deployment. Not applicable for cloud connector deployment.
3 Control Plane Configuration		
3.1 Authentication and Authorization		
3.1.1 Client certificate authentication should not be used for users (Manual)	FAIL	Since the cloud connector is a self-installed agent and upgrades are managed over the air, no external user access is required.
3.1.2 Service account token authentication should not be used for users (Manual)	FAIL	Since the cloud connector is a self-installed agent and upgrades are managed over the air, no external user access is required.
3.1.3 Bootstrap token authentication should not be used for users (Manual)	PASS	
3.2 Logging		

Controls	Status	Comment
3.2.1 Ensure that a minimal audit policy is created (Manual)	PASS	Audit policy is enabled (warn).
3.2.2 Ensure that the audit policy covers key security concerns (Manual)	FAIL	Not supported.
4.1 Worker Node Configuration Files		
4.1.1 Ensure that the kubelet service file permissions are set to 600 or more restrictive (Automated)	NA	Not applicable for k3s.
4.1.2 Ensure that the kubelet service file ownership is set to root (Automated)	NA	Not applicable for k3s.
4.1.3 If proxy kubeconfig file exists ensure permissions are set to 600 or more restrictive (Manual)	PASS	
4.1.4 If proxy kubeconfig file exists ensure ownership is set to root (Manual)	PASS	
4.1.5 Ensure that the --kubeconfig kubelet.conf file permissions are set to 600 or more restrictive (Automated)	PASS	
4.1.6 Ensure that the --kubeconfig kubelet.conf file ownership is set to root (Automated)	PASS	
4.1.7 Ensure that the certificate authorities file permissions are set to 600 or more restrictive (Manual)	PASS	
4.1.8 Ensure that the client certificate authorities file ownership is set to root (Manual)	PASS	
4.1.9 Ensure that the kubelet --config configuration file has permissions set to 600 or more restrictive (Automated)	PASS	
4.1.10 Ensure that the kubelet --config configuration file ownership is set to root (Automated)	PASS	
4.2 Kubelet		
4.2.1 Ensure that the --anonymous-auth argument is set to false (Automated)	PASS	
4.2.2 Ensure that the --authorization-mode argument is not set to AlwaysAllow (Automated)	PASS	

Controls	Status	Comment
4.2.3 Ensure that the --client-ca-file argument is set as appropriate (Automated)	PASS	
4.2.4 Verify that the --read-only-port argument is set to 0 (Manual)	PASS	
4.2.5 Ensure that the --streaming-connection-idle-timeout argument is not set to 0 (Manual)	PASS	
4.2.6 Ensure that the --make-iptables-util-chains argument is set to true (Automated)	PASS	
4.2.7 Ensure that the --hostname-override argument is not set (Manual)	NA	Not applicable for k3s.
4.2.8 Ensure that the eventRecordQPS argument is set to a level which ensures appropriate event capture (Manual)	PASS	
4.2.9 Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate (Manual)	NA	Not applicable for k3s.
4.2.10 Ensure that the --rotate-certificates argument is not set to false (Manual)	PASS	
4.2.11 Verify that the RotateKubeletServerCertificate argument is set to true (Manual)	FAIL	Does not affect the operation of the cloud connector, as external kube API access is required for any host level access for the cluster alone.
4.2.12 Ensure that the Kubelet only makes use of Strong Cryptographic Ciphers (Manual)	FAIL	The cloud connector runs on a single host machine.
4.2.13 Ensure that a limit is set on pod PIDs (Manual)	FAIL	Will be implemented in the upcoming release.
5.1 RBAC and Service Accounts		
5.1.1 Ensure that the cluster-admin role is only used where required (Manual)	FAIL	Since the cloud connector is a self-installed agent and upgrades are

Controls	Status	Comment
		managed over the air, these permissions must be maintained for installation and upgrade functionality.
5.1.2 Minimize access to secrets (Manual)	FAIL	Since the cloud connector is a self-installed agent and upgrades are managed over the air, these permissions must be maintained for installation and upgrade functionality. Additionally CC doesn't have multiple users.
5.1.3 Minimize wildcard use in Roles and ClusterRoles (Manual)	PASS	
5.1.4 Minimize access to create pods (Manual)	FAIL	Since the cloud connector is a self-installed agent and upgrades are managed over the air, these permissions must be maintained for installation and upgrade functionality. Additionally CC doesn't have multiple users.
5.1.5 Ensure that default service accounts are not actively used. (Manual)	PASS	Separate service accounts are created for the cc namespace.

Controls	Status	Comment
5.1.6 Ensure that Service Account Tokens are only mounted where necessary (Manual)	FAIL	Not supported.
5.1.7 Avoid use of system group (Manual)	FAIL	Since its a default feature, its not removed as of now
5.1.8 Limit use of the Bind, Impersonate and Escalate permissions in the Kubernetes cluster (Manual)	FAIL	The cluster access to be restricted by creating a less permissive user account.
5.1.9 Minimize access to create persistent volumes (Manual)	FAIL	Cloud connector uses persistent volumes for storing logs and maintaining other external libraries such iControl jar. The new pods should have the capability to support upgrade from GUI.
5.1.10 Minimize access to the proxy sub-resource of nodes (Manual)	FAIL	Not supported.
5.1.11 Minimize access to the approval sub-resource of certificatesigningrequests objects (Manual)	FAIL	This is not relevant since the cloud connector operates exclusively on Linux-based systems.
5.1.12 Minimize access to webhook configuration objects (Manual)	FAIL	Not supported.
5.1.13 Minimize access to the service account token creation (Manual)	FAIL	Not supported.
5.2 Pod Security Standards		
5.2.1 Ensure that the cluster has at least one active policy control mechanism in place (Manual)	PASS	Addressed as part of FP3.1 release*
5.2.2 Minimize the admission of privileged containers (Manual)	FAIL	Baseline policy will be enforced in FP3.1

Controls	Status	Comment
		(Tentative), restricted policy not supported*
5.2.3 Minimize the admission of containers wishing to share the host process ID namespace (Automated)	FAIL	Baseline policy will be enforced in FP3.1 (Tentative), restricted policy not supported*
5.2.4 Minimize the admission of containers wishing to share the host IPC namespace (Automated)	FAIL	Baseline policy will be enforced in FP3.1 (Tentative), restricted policy not supported*
5.2.5 Minimize the admission of containers wishing to share the host network namespace (Automated)	FAIL	Cloud connector uses host network for communicating outside with the devices.
5.2.6 Minimize the admission of containers with allowPrivilegeEscalation (Automated)	FAIL	avx-mid-server-platform pod will have allowPriviegeEscalation as true due to nmap command execution during network discovery use cases.
5.2.7 Minimize the admission of root containers (Automated)	FAIL	All the business pods in the cc namespace will be running as non-root. The pods are running using the host user id who installed the CC (avxctl refresh all to be fired). The log clean up cronjob will be running as root to support backward compatibility.

Controls	Status	Comment
5.2.8 Minimize the admission of containers with the NET_RAW capability (Automated)	FAIL	Not supported.
5.2.9 Minimize the admission of containers with added capabilities (Automated)	FAIL	Not supported.
5.2.10 Minimize the admission of containers with capabilities assigned (Manual)	FAIL	Not supported.
5.2.11 Minimize the admission of Windows HostProcess containers (Manual)	FAIL	Not relevant since the cloud connector operates exclusively on Linux-based systems.
5.2.12 Minimize the admission of HostPath volumes (Manual)	FAIL	The cloud connector requires mounting files from the host machine, so hostPath volumes must be allowed.
5.2.13 Minimize the admission of containers which use HostPorts (Manual)	FAIL	The cloud connector uses host ports to expose services like IoT. Host ports admission is required.
5.3 Network Policies and CNI		
5.3.1 Ensure that the CNI in use supports NetworkPolicies (Manual)	PASS	Network policies has been added for cc namespace.
5.3.2 Ensure that all Namespaces have NetworkPolicies defined (Manual)	FAIL	Network policies will be available as part of FP3.1 release for the cc namespace*
5.4 Secrets Management		
5.4.1 Prefer using Secrets as files over Secrets as environment variables (Manual)	PASS	No business specific secrets for the

Controls	Status	Comment
		application are created.
5.4.2 Consider external secret storage (Manual)	FAIL	The cloud connector is a light weight agent installed on a single host. So it uses the default secret management system provided by k3s.
5.5 Extensible Admission Control		
5.5.1 Configure Image Provenance using ImagePolicyWebhook admission controller (Manual)	FAIL	No supported.
5.7 General Policies		
5.7.1 Create administrative boundaries between resources using namespaces (Manual)	PASS	Separated namespace has been created
5.7.2 Ensure that the seccomp profile is set to docker/default in your Pod definitions (Manual)	PASS	Addressed as port FP3.1.
5.7.3 Apply SecurityContext to your Pods and Containers (Manual)	PASS	
5.7.4 The default namespace should not be used (Manual)	PASS	

The report is prepared base on the k3s self assement guide v1.8.

* policies will require the cloud connector reinstallation to be effective

Chapter 3: Managed Kubernetes

These Managed Kubernetes - Install and Upgrade Guides provides the prerequisites and the procedure for installing and accessing AppViewX on AKS, EKS, and GKE.

- AppViewX Install and Upgrade for AKS
- AppViewX Install and Upgrade for EKS
- AppViewX Install and Upgrade for GKE

AppViewX Install and Upgrade for AKS

This guide provides the prerequisites and the procedure for installing, upgrading, and accessing the AppViewX application.

- AppViewX Architecture
- Architecture Overview
- AppViewX Deployment Architecture
- Managed Kubernetes Architecture
- AKS Components
- Prerequisites
- Install AppViewX in Managed Kubernetes
- Upgrade AppViewX in Managed Kubernetes
- Downloading Images from AppViewX Repository
- Kubernetes Version Upgrade in AKS
- Uninstall and Cleanup
- Troubleshooting

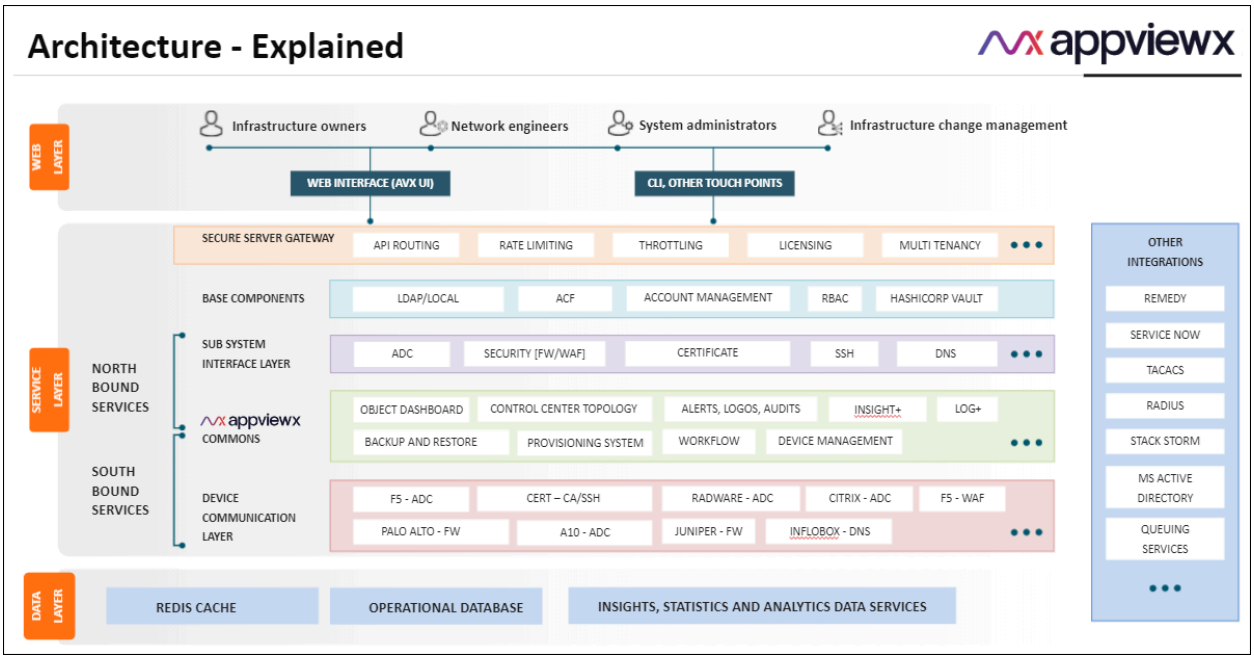
AppViewX Architecture

Architecture Explained

AppViewX is designed based on the microservice architecture and is deployed on Kubernetes—an open-source platform for deploying and managing containers.

The microservice architecture of AppViewX makes it easier to move to containerized workloads and the containers being orchestrated using Kuberneters.

Kubernetes provides container runtime, orchestration, self-healing mechanisms, service discovery and load balancing and it is used for the deployment, scaling, management, and composition of application containers across clusters.



Benefits of AppViewX Architecture

In order to optimally utilize the resources, AppViewX has adopted Kubernetes to achieve higher security by adopting a zero trust network model. The features of AppViewX coupled with Kubernetes are given below.

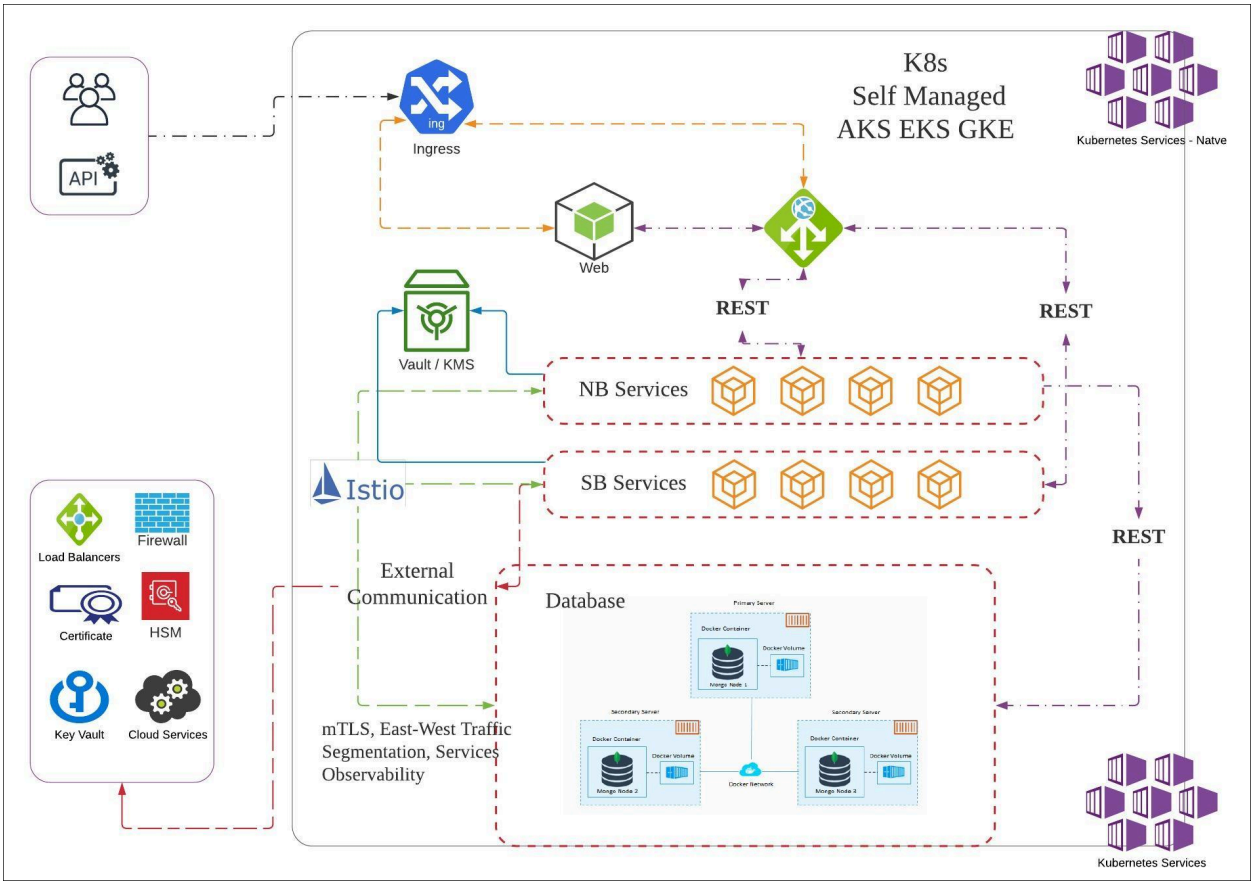
- **Auto scaling** - AppViewX services can have a custom throttling capability based on pre-configured memory configuration per API. This will enable AppViewX services to utilize (scale up) resources optimally as the demand surges and scale down when not in use. This will help to horizontally scale the vendor components on demand and optimize the resource usage.
- **Resiliency** - There is no guarantee that AppViewX services may run without any interruptions and they are bound to fail. Kubernetes keeps deployments healthy by restarting containers that have failed, by killing and replacing unresponsive containers based on health checks. This helps to mitigate the common pain point of the application's upkeep process.
- **Security** - AppViewX architecture is designed around the concept of zero trust network model to enforce tighter security within the Kubernetes cluster. This means no one is trusted by default and requires verification to gain access to the services.

Architecture Overview

AppViewX Kubernetes Architecture

AppViewX workloads are containerized workloads running as microservices and these containers are orchestrated by managed Kubernetes services. Users can prefer the managed k8s platform of their choice.

AppViewX supports deployment on all the three public clouds AWS, Azure and GCP (Google Cloud Platform) using their managed kubernetes engine / services EKS, AKS and GKE specifically.



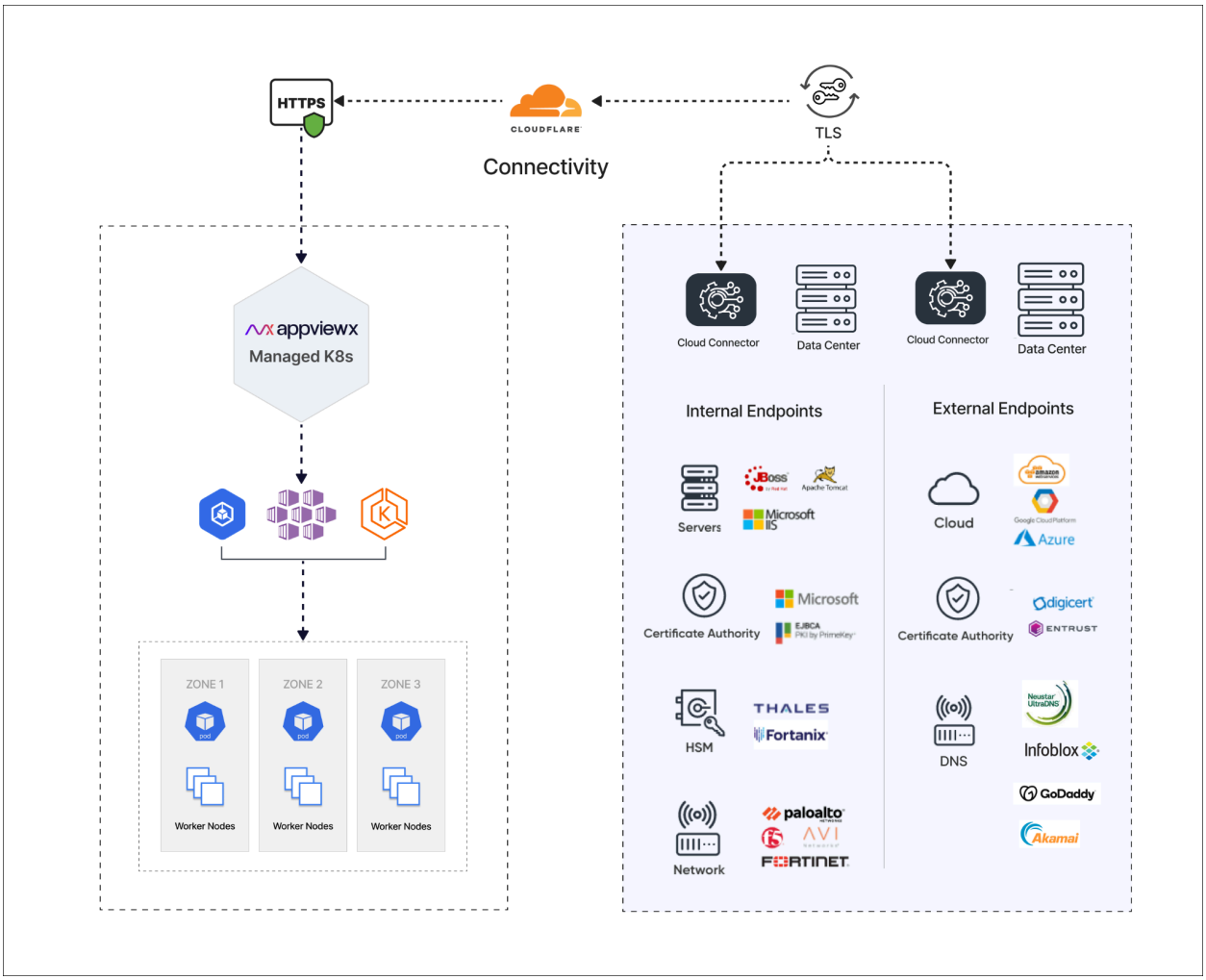
Benefits of AppViewX Architecture

In order to optimally utilize the resources, AppViewX has adopted Kubernetes to achieve higher security by adopting a zero trust network model. The features of AppViewX coupled with Kubernetes are given below.

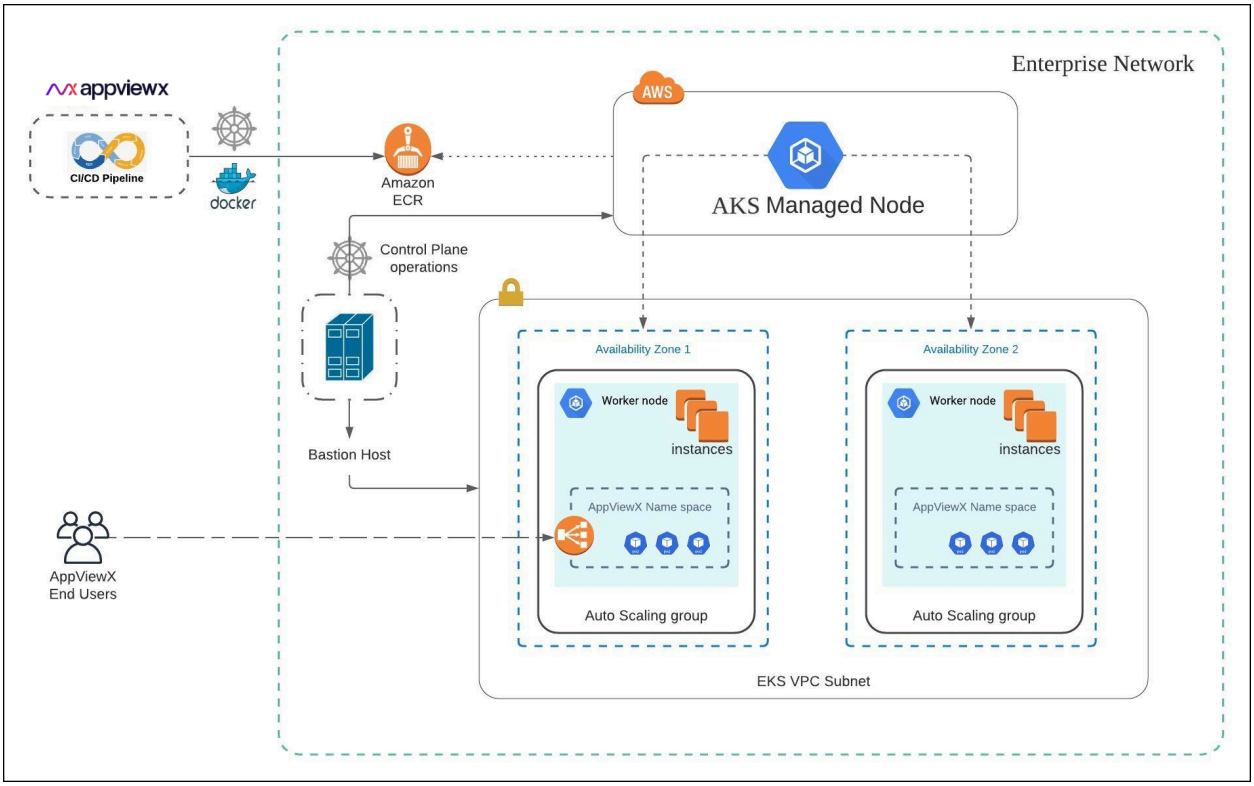
- **Auto scaling** - AppViewX services can have a custom throttling capability based on pre-configured memory configuration per API. This will enable AppViewX services to utilize (scale up) resources optimally as the demand surges and scale down when not in use. This will help to horizontally scale the vendor components on demand and optimize the resource usage.
- **Resiliency** - There is no guarantee that AppViewX services may run without any interruptions and they are bound to fail. Kubernetes keeps deployments healthy by restarting containers that have failed, by killing and replacing unresponsive containers based on health checks. This helps to mitigate the common pain point of the application's upkeep process.
- **Security** - AppViewX architecture is designed around the concept of zero trust network model to enforce tighter security within the Kubernetes cluster. This means no one is trusted by default and requires verification to gain access to the services.

AppViewX Deployment Architecture

The figure below shows a standard AppViewX deployment architecture model via managed Kubernetes service for AKS.



AKS Deployment Model

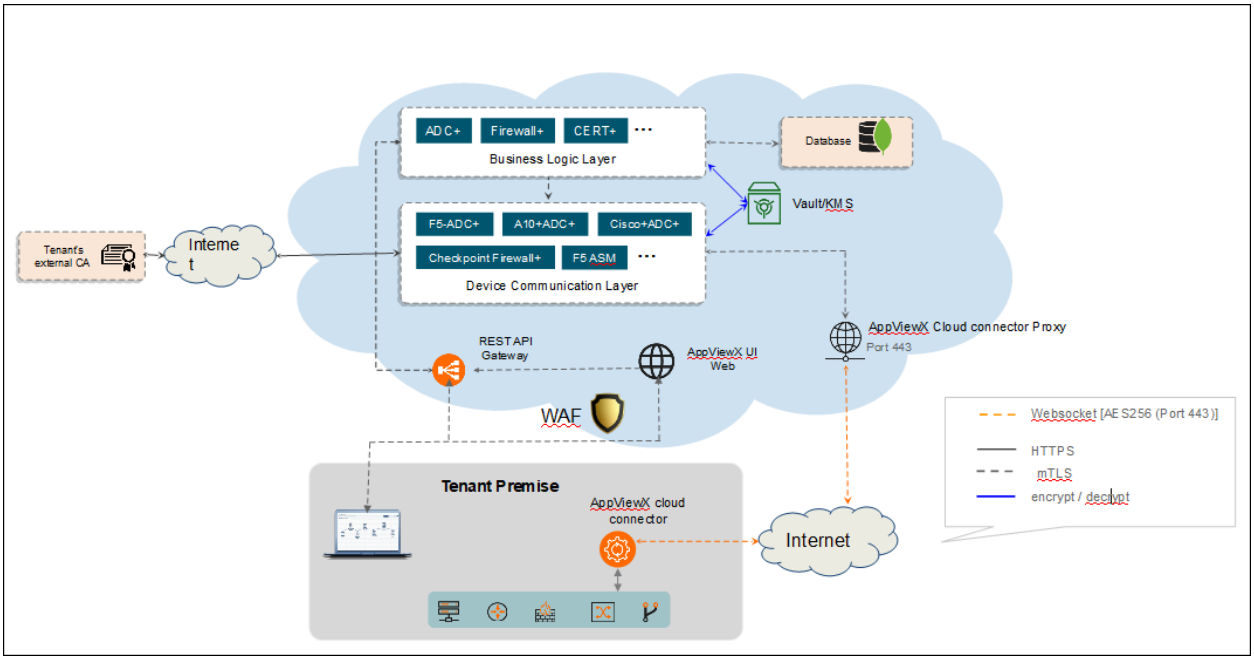


Cloud Connector

AppViewX Cloud Connector is a lightweight plug-in that establishes connectivity between AppViewX Cloud and the Enterprise Network. The cloud connector serves as a secure channel for communication between AppViewX and your enterprise network without requiring any complex network or infrastructure configuration

Key features of the AppViewX Cloud Connector:

- A self-serviceable, Linux-based lightweight setup
- Secure communication between the AppViewX and the AppViewX Cloud Connector using TLS and AES encryption
- Connectivity from the AppViewX to the enterprises' network endpoints
- No complex network setup (Inbound Firewall Whitelisting, VPN setup, and so on)



For more details on cloud connectors refer to AppViewX Cloud Connector User Guide.

Note: The below steps have to be performed in all the cloud connector host machines after the 2022.1.0FP2 to FP3 patch upgrade and before the FP3 cloud connector upgrade.

1. Navigate to the installation path in the cloud connector host machine.
2. Execute the following command:

```
./deps/tools/k3s kubectl get deploy avx-mid-server-starter -n cc -o yaml > starter.yaml && sed -i "s/-Xmx2560m/-Xmx4g/g" starter.yaml

&& ./deps/tools/k3s kubectl replace -f starter.yaml
```

Managed Kubernetes Architecture

Managed Kubernetes clusters are composed of the following main components — a control plane and worker nodes. Each cluster runs in its own, fully managed Virtual Private Cloud (VPC).

- The **control plane** is composed of three master nodes, each running in a different Availability Zone to ensure high availability. Incoming traffic directed to the Kubernetes API passes through the respective cloud service load balancer.
- The **worker nodes** run on virtual instances located in a VPC. Managed kubernetes service engine provides managed node groups with automated lifecycle management. This lets users automatically create, update, or shut down nodes with one operation.

Managed Kubernetes service scales the Kubernetes control plane across multiple Availability Zones of the public cloud to ensure high availability and it automatically scales control plane instances based on load, detects and replaces unhealthy control plane instances, and automatically patches the control plane.

Managed Kubernetes workload instances are deployed in multiple availability zones within the region. Each instance has replicas of the services and nodes which exist across all the virtual instances.

Each zone or instance has an active pod listening to other instances. In case of a failure in any instance, the active pod ensures seamless functioning of the application by activating the nodes from any other working cluster.

AKS Components

The following AKS components are utilized by AppViewX:

- Azure Kubernetes service
- Storage account for storing mongodb and vault backups
- Service principal for accessing the ACR registry

Prerequisites

The following prerequisites must be met before the installation process.

- Managed Kubernetes Version Support Matrix
- Disks Used for AppViewX Installation
- AppViewX Docker Images
- AppViewX Helm Charts
- Bastion Host Setup
- AKS Cluster
- Azure Storage
- Azure Service Principal

Managed Kubernetes Version Support Matrix

Public Cloud	
Mode of Deployment	**Azure**
Release, Vendor, & Product Support **AppViewX v2023.1.0 FP3**	
Managed K8s Deployment (AKS)	
K8s version 1.29	Yes

Disks Used for AppViewX Installation

Discs Used

Volume	Size	Quantity
logs volume	50Gi	1
avx-kafka	20Gi	3
zookeeper	20Gi	3
consul-server	10Gi	3
mongo-configdb	10Gi	3
mongo-shareddb	256Gi	3
redis	5Gi	3

If a third party is installed, the values are as follows:

Discs Used (Third Party)

Volume	Size	Quantity
Elasticsearch-ELK	10Gi	1
Elasticsearch-Insight	10Gi	1

AppViewX Docker Images

AppViewX Docker images are hosted in a private registry https://images.appviewx.com. These images can be pulled using an authentication token (contact AppViewX Support, help@appviewx.com for the authentication token) and can be hosted in the private or public repository at the customer end.

The list of docker images are

- <registry link>/appviewx/pilot:1.19.0
- <registry link>/appviewx/proxyv2:1.19.0
- <registry link>/appviewx/istio-operator:1.19.0
- <registry link>/appviewx/vault:1.13.7
- <registry link>/appviewx/redis:7.2.0
- <registry link>/appviewx/mongo-init:<tag>
- <registry link>/appviewx/avx-cloud-gateway:<tag>
- <registry link>/appviewx/avx-cloud-web:<tag>
- <registry link>/appviewx/avx-cloud-mongoseed:<tag>
- <registry link>/appviewx/avx-cloud-managedservice-mks:<tag>
- <registry link>/appviewx/avx-platform-report-generator:<tag>
- <registry link>/appviewx/avx-python-sandbox:<tag>
- <registry link>appviewx/avx-mid-server-base:<tag>
- <registry link>/appviewx/consul:1.16.1
- <registry link>/appviewx/kafka:0.32.0-kafka-3.3.1
- <registry link>/appviewx/operator:0.32.0
- <registry link>/appviewx/alpine:3.13.6
- <registry link>/appviewx/kube-metrics-adapter:v0.2.1
- <registry link>/appviewx/kube-state-metrics:v1.9.8
- <registry link>/appviewx/backup-utility-image:v3.0
- <registry link>/appviewx/prometheus:v2.45.0
- <registry link>/appviewx/metrics-server:v0.6.4
- <registry link>/appviewx/elasticsearch:8.9.1
- <registry link>/appviewx/elasticsearch-insight:8.9.1
- <registry link>/appviewx/filebeat:8.9.1
- <registry link>/appviewx/grafana:10.1.1
- <registry link>/appviewx/kibana:8.9.1
- <registry link>/appviewx/logstash:8.9.1
- <registry link>/appviewx/logstash-syslog:8.9.1
- <registry link>/appviewx/alertmanager:v0.26.0

- <registry link>/appviewx/node-exporter:v1.6.1
- <registry link>/appviewx/redis_exporter:v1.53.0

The steps to download the images from AppViewX repository are as follows:

1. Get the source image repository credentials from AppViewX Support team.
2. Configure the docker using the command

```
docker login -u ${USERNAME} -p ${PASSWORD} ${DOCKER_REPOSITORY}
```

3. Configure the respective cloud provider CLI (Google cloud) and ensure you have access to push docker images to GCR.
4. To push the docker images, use the helper script provided by AppViewX. Follow the steps below.

   a. Download the artifact Managed-Kubernetes_helper_scripts.tar.gz to the bastion host and extract using the command:

   ```
   tar -xf Managed-Kubernetes_helper_scripts.tar.gz
   ```

   b. Navigate to the extracted directory **mk8s_helper_scripts**.

   ```
   cd mk8s_helper_scripts
   ```

   c. Execute the script **avx_image_pull_push.sh** using the command

   ```
   ./avx_image_pull_push.sh <Image tag> <customer registry url>
   ```

   Note: Replace <Image tag> and <customer registry url> with the actual values.

AppViewX Helm Charts

The helm charts used by AppViewX for installation are released as a part of the installer. The installer consists of helm charts and an AppViewX utility which helps orchestrate the deployment, patch, upgrade and maintenance of AppViewX across managed kubernetes deployment.

Bastion Host Setup

The following packages must be installed on the bastion host or the host/tool from where the installation is triggered

Azure CLI

To set up the Azure CLI refer to Install the Azure CLI on Linux on the Microsoft documentation website.

Execute the following command:

```
curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash
```

Kubectl

To set up Kubectl refer to Install and Set Up kubectl on Linux on the Microsoft documentation website.

Execute the following commands

- sudo curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
- sudo chmod +x kubectl
- sudo mv ./kubectl /usr/bin/#

Verify installation by executing the command

```
kubectl version
```

Helm

Helm is required only if the deployment is triggered from any other machine instead of the DevOps pipeline. To set up Helm refer to Installing Helm on the Helm documentation website.

Execute the following command:

- curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
- chmod 700 get_helm.sh
- ./get_helm.sh#

Verify installation by executing the command

```
helm version
```

AKS Cluster

To create an AKS cluster refer to Microsoft's online manual - Azure Kubernetes Service (AKS). Although Microsoft manuals are always up-to-date, the recommended choice to make before creating the cluster is as follows:

- Kubernetes version: 1.29
- The network model:
  - Azure Kubenet (supported by AppViewX).
  - Azure CNI (Recommended for optimal performance).
- Managed identity: System assigned managed identity.
- Enable Kubernetes RBAC.
- **Agent nodepool**: Three nodes of Machine type **D2sv4** with Auto Scaling disabled. Add taint to agent nodepool as **CriticalAddonsOnly=true:NoSchedule** to disable scheduling of application pods to the agent nodepool.

> **Note:** The taint **CriticalAddonsOnly=true:NoSchedule** prevents the application pods from being scheduled on system node pools.

- **User nodepool**:
  - **appnodepool**: Three nodes of type **Da8sv4** with Auto Scaling disabled
  - **mongonodepool**: Three nodes of type **Da8sv4** with Auto Scaling disabled. Add label **mongo=true** and taint **designatedMongo=true:NoSchedule** to the nodepool (to be performed while creating the cluster).

> **Note:** A minimum of 3 availability zone are needed during cluster creation to support the single AZ failover.

- Select multi zones for the Agent nodepool and the User Nodepool.

> **Note:** The number of nodes mentioned here are applicable for managing up-to 25K certs. This number will vary if there are more certificates to manage.

Azure Storage

Azure Storage Account

A Storage account is required to store

- iControlJar
- MongoDB backup
- Vault backup
- Axisjar

Always create a storage account with a valid name to indicate the storage account for a specific AKS cluster. A typical naming convention could be **<clusternamestorage>**.

> **Note:** Storage account access by the AKS pods is granted by the storage account connection string.

For more information on the Azure storage account, refer to Microsoft's online manual - Create a storage account.

Azure Storage Container

The following containers must be created in the storage accounts that have already been setup.

1. **icontroljar**: The iControlJar needs to be placed here before installing AppViewX plugins
2. **mongo-backup**: Backup job stores mongodb backup into the container.
3. **vault-backup**: Backup job stores vault backup into the container
4. **axisjar**: Container name should be **axisjar**.

For more information on the Azure storage containers, refer to Microsoft's online manual - Quickstart: Upload, download, and list blobs with the Azure portal.

Azure Service Principal

Azure Service Principal is used to create the image pull secrets and download images from ACR. When creating the Service Principal

- Set the expiration to never
- Assign the ACR pull access role

For information on commands to create the Azure Service Principal refer to Microsoft's online manual - Azure Container Registry authentication with service principals.

A summary of steps are as follows:

- To derive the service principal id and the password execute the helper script provided by AppViewX. To use this script follow the steps below.

  1. Download the artifact Managed-Kubernetes_helper_scripts.tar.gz to the bastion host and extract using the command:

     ```
     tar -xf Managed-Kubernetes_helper_scripts.tar.gz
     ```

  2. Navigate to the extracted directory **mk8s_helper_scripts**.

     ```
     cd mk8s_helper_scripts
     ```

  3. Edit the file **acr_reg_config.sh** and replace ACR_NAME with the actual value.

  4. Execute the **acr_reg_config.sh** file.

     ```
     bash acr_reg_config.sh
     ```

Note: After the script execution, capture the outputs as they are required in the global utility config.

Install AppViewX in Managed Kubernetes

Migration Strategy

Attention: If you are performing a fresh install, then refer the next sub-topic **Installation Steps**.

To migrate from AppViewX on-prem versions (2022.1.0, 2021.1.0, and 2020.3.0) to Managed Kubernetes, it is important to take a backup of the mongodb and vault in the respective on-prem versions. Before you take the backup, execute the script below.

```
db.profile.update({'_id' : 'installationType'}, {$set : {"value" : "Managed_K8s"}})
```

Note: Refer to the specific version of the release documents from the release portal and perform the backups or contact the AppViewX support team.

After performing the backup, follow the installation steps detailed in the section below. At step 11 of the installation process, ensure to restore the data at this stage.

Installation Steps

This section describes the steps to for installing the AppViewX Stack on AKS.

1. Download the installer from the release portal.
2. Create a directory Managedk8s-installer in the bastion host and extract the installer file **tar -xf installer.tar.gz** in the same directory.
3. Verify that the extracted installer must have the following files
   - appviewxctl (binary)
   - helm_charts (directory of helm charts)
4. Generate the configuration files based on the cloud provider. If the cloud provider is **Azure**, execute the command below.

```
./appviewxctl config generate --provider azure
```

5. Verify that the execution of the above command creates the configuration files named **.appviewxctl.yaml** in the same location.
6. The file .appviewxctl will be populated with the fields necessary for installation, in particular cloud provider that was provided in previous command (**-- provider**).
7. Edit the **appviewxctl.yaml** file and populate the values as described below:

appviewxctl.yaml file - Parameters and Description

Parameters	Description of Values
chartPath	The path to the helm_charts which is to be installed. It points to the helm_charts directory extracted in step 3.
configFile	The path to the kube config file to be used by helm and kubectl. If the bastion host is already configured and kube config is under $HOME/.kube directory, then keep this field empty.
install.enableAppBackupCron	Boolean value to enable/disable the backup cronjobs. (True/False).

Parameters	Description of Values
	This value is needed for self-managed mongodb only. For atlas backup this has to be scheduled in the atlas dashboard.
install.enablePrivateImagePullSecret	Boolean value to enable image pull secret. Set values as **false** if the cluster already has access to the container registry. Otherwise set it to **true** and fill all the details of the access keys described in below sections.
install.enableThirdPartyInstall	Boolean value (True/False) to determine whether third party monitoring components such as ELK, Monitoring, and Insight needs to be installed.
install.thirdPartyApp.elk	Boolean value to add Elk component. Set to True if it needs to be installed.
install.thirdPartyApp.monitoring	Boolean value to add Monitoring component. Set to True if it needs to be installed.
install.thirdPartyApp.insight	Boolean value to add Insight component. Set to True if it needs to be installed.
install.imageRegistry	The URL of the container registry where the images are to be pulled from by the pods. *Example*: appviewx.azureacr.io
install.imageTag	The tag of the image that will be used for installation. *Example*: 2022.1.0_FP_750-alpine
install.isSaasEnabled	Boolean value for SaaS enablement. This value should be set to **true** for Managed K8s.
install.kafkaCloudConnector	It is a combination of three values.

Parameters	Description of Values
	• enable • password • user Set **enable** to ***true*** and keep the user, password fields empty for Managed K8s. *Example* kafkaCloudConnector: enable: true password: "" user: ""
install.mongo	It is a combination of fields specific to the type of mongodb used.
dbIsolation	Boolean value to indicate whether the database isolation is to be enabled. In order for database isolation to work, the following prerequisite must be taken care of while creating the cluster node group. • Add label **mongo=true** and taint **designatedMongo=true:NoSchedule** to the nodepool to be used for mongodb.
mongoAtlas	The fields specific to mongo atlas are as folows: • **enable**: Boolean value to decide if mongo atlas to be used. If set to *false*, a self managed mongo cluster will be created. If set to *true* mongo atlas will be used and details of which are to be provided in below mentioned fields. • **host**: URL of the mongodb atlas cluster. • **password**: password of the mongodb atlas cluster. • **user**: username in the mongodb atlas cluster. *Example:*

Parameters	Description of Values
	mongo: dbIsolation: false mongoAtlas: enable: true host: "managed-k8s.test.mongodb.net" password: "samplepassword" user: "user1"
install.useDockerPrivateRegistry	Set this to **true** if the dockerhub private repository is to be used for pulling the necessary images needed. Otherwise set the value **false** and the container registry ACR, ECR, and GCR will be used based on the cloud provider. If this value is set to *true*, populate the below values, otherwise keep it empty. • **dockerhub.pass**: password to be used for authenticating in the dockerhub private repository. • **dockerhub.username**: username configured in the dockerhub private repository. *Example:* useDockerPrivateRegistry: true dockerhub: pass: "testpassword" username: "appviewx"
install.size	The size of the installation. Based on the use cases and number of certs to be managed there different sizes (contact AppViewX for sizing recommendations). The supported size values are (case sensitive values) • xsmall • small • medium • large

Parameters	Description of Values
	• xlarge • custom *Example:* `size: small` **Note:** The size provided must be taken into cluster creation and nodegroup sizes must be defined accordingly. Follow the same document link above for nodegroup sizes.

install.plugins	The list of plugins that will be installed. Each plugin will have three fields
	• enable
	• imageTag
	• name
	
	Set enable to **true** if the plugin is to be installed. If the same image tag is to be used as defined in the global ImageTag keep it **latest** otherwise override with some other tag of your choice.
	
	Example:
	`- enable: true`
	`imageTag: latest`
	`name: avx-config-server`
	
	To enable Cloud DC support in Managed Kubernetes, set plugins as follows:
	`- enable: true`
	`imageTag: latest`
	`name: avx-mid-server-platform`
	
	Note: Ensure that **install.isSaasEnabled** and

Parameters	Description of Values
	install.kafkaCloudConnector are set to **true**.
internalLoadBalancer	If set to **true**, all the Loadbalancers will be private and can only be accessed within the VPC else it will be public.
install.enableSftpStorage	Change to true to use SFTP server for mongodb, vault, and icontrol.jar storage. Boolean (Default: false)
install.sftpServerDetails. dbBackupPath	Provide the location of mongodb backup storage directory. String (Default: “”)
install.sftpServerDetails. vaultBackupPath	Provide the location of vault backup storage directory. String (Default: “”)
install.sftpServerDetails. sftpServerUserName	Provide the username of SFTP server. String (Default: “”)
install.sftpServerDetails. sftpServerIp	Provide the sftp server IP. String (Default: “”)
cloudConnectorEnabled	A boolean value (true/false) to denote the cloud connector usage for southbound communications. If a cloud connector is used set the value to **true**.

The next fields are to be filled with values that must be collected during the cluster creation and setup process and filled as mentioned below.

appviewxctl.yaml file - Parameters and Description (for cluster creation)

Parameters	Description of Values
install.privateImagePullSecret	In this section populate the details of the access keys needed to authenticate and pull the image from the registry. They are not needed if the Dockerhub is used as described above. • **registry**: The ACR registry URL • **servicePrincipalPassword**: The service Principal Password for accessing the registry. • **servicePrincipalUsername**: The service Principal Username for accessing the registry.

Parameters	Description of Values
	Example: registry: "appviewxsample.azurecr.io" servicePrincipalPassword: "qLPUSA4R1ALkA~GH6m4v70iAC_jajEo9T" servicePrincipalUsername: "20892076-ct8a-4700-a7c0-178u066q9a9c"
install.storageAccess	The storage bucket details to be used for setting up backup capability. • **bucketObject**: The storage bucket access string. • **serviceAccountAnnotation**: "none" *Example:* bucketObject: "DefaultEndpointsProtocol=https;AccountName=sampleappviewx;AccountKey=Qy0SKtry2MR4Ik0OIG+po3p0Kgla7u4KEjlYo10jHYdVlZXP2/v4lMomkZK6s58YLsLbzkutkyjHJINuCo2Y7w==;EndpointSuffix=core.windows.net" serviceAccountAnnotation: "none"

The following fields must be added to integrate the kubernetes cluster to the external vault.

appviewxctl.yaml file - Parameters and Description (for external vault)

Parameters	Description
install.externalVault.enable	A boolean value (true/false) to denote if the external vault is to be used in the setup. True is to enable the external vault.
install.externalVault.externalVaultAddr	Contains the vault URL and listening port *Example*: https://pm-lxs-node01.lab.appviewx.net:8200
install.externalVault.externalVaultAuthRole	Name of the role created against the access kubernetes auth path

Parameters	Description
install.externalVault.externalVaultCACertSecret	Name of the secret where **vault-ca.crt** file is mounted.
install.externalVault.externalVaultDBRole	Static role created to access the database cred.
install.externalVault.externalVaultEnginePath	Enter the value “/database”
install.externalVault.externalVaultKubeAuthPath	The Kuberenetes access path created with cluster information for service account authentication.
install.externalVault.externalVaultSAName	The Service account used to create externalVaultAuthRole.
install.externalVault.mongoPasswordVaultEngine	Enter the value DATABASE

8. Once the values are filled in *.appviewxctl* as described in the step above, proceed with the installation. Before doing so, check if the the preconditions are met by executing the command

```
./appviewxctl preflight --config .appviewxctl.yaml
```

   This will prompt if the necessary prerequisites are met.

9. The metrics server in the Azure clusters comes pre-installed with the cluster, hence they must be disabled from the **avx_pre_req** chart.

   a. Navigate to helm_charts/avx_pre_req.

   b. Edit the **values.yaml** file by setting the following parameters.

```
avx-metrics-server:

enable: false
```

   The metrics server installation is disabled.

10. To proceed with installation, execute the command

```
./appviewxctl install --config .appviewxctl.yaml
```

> **Note:** The installation will take several minutes to complete. Upon completion you see the following message:
>
> [Install] Successfully installed Appviewx infra stack

This would imply the completion of infra component setup.

11. This step involves restoring the existing data from the previous AppViewX version's cluster in case there is a need to migrate from the older versions to the Managed K8s version. **Ignore this step if it's a fresh setup with no migration necessary.**

    To restore mongodb and vault fetch the backup files and place them in the bastion in a directory such as /home/user/backup execute the mongo_restore and vault_restore scripts as follows:

    ```
    ./mongo_restore.sh <mongo backup tar filepath>
    ```

    ```
    ./vault_restore.sh -p <vault backup filepath> --removedek
    ```

    Attention: If the data is being restored from an older version (2020.3.0 - 2022.1) then use the command

    ```
    ./vault_restore.sh -p <vault backup filepath> --removedek
    ```

    Note:
    - The backup files must have extension as **.tar.gz**
    - The above commands work for a self-managed mongodb setup. Setting up the mongodb atlas requires the installation of mongodb tools in the bastion host as described below.

    For an rpm based OS:

    ```
    echo -e "[mongodb-org-4.2] \nname=MongoDB
     Repository\nbaseurl=https://repo.mongodb.org/yum/redhat/\$releasever/mongodb-org/4.2/x86_64/\ngpgcheck=1\nenabled=1\ngpgkey=https://
    www.mongodb.org/static/pgp/server-4.2.asc" > /etc/yum.repos.d/mongodb-org-4.2.repo
    yum install mongodb-org-shell-4.2.0
    yum install mongodb-org-tools-4.2.0
    ```

    For a debian based OS:

    ```
    wget -qO - https://www.mongodb.org/static/pgp/server-6.0.asc | sudo apt-key add -
    sudo apt-get install gnupg
    wget -qO - https://www.mongodb.org/static/pgp/server-6.0.asc | sudo apt-key add -
    echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/6.0 multiverse" | sudo
     tee /etc/apt/sources.list.d/mongodb-org-6.0.list
    sudo apt-get update
    sudo apt-get install -y mongodb-mongosh
    ```

```
sudo apt-get install -y mongodb-org-tools
```

Verify if the mongo restore commands have executed successfully using the command

```
mongorestore -- version
```

12. To proceed with the AppViewX application installation, execute the command:

```
./appviewxctl installapp --config .appviewxctl.yaml
```

Once installation is complete the following messages are displayed:

```
[Install] Appviewx infrastructure chart [avx-app] installed successfully
[Install] Successfully installed Appviewx application stack
[Install] Fetching login URL for app
[Install] Waiting for Public IP allotment for istio service
[Install] AppViewX Web URL: https://34.100.197.159/appviewx/
[Install] AppViewX Gateway URL: https://34.100.197.159/avxmgr/
[Install] Grafana URL: https://34.100.197.159/grafana/
[Install] Kibana URL: https://34.100.197.159/kibana/login
[Install] Run below commands to get mongo user credentials
 export MONGO_USER=$(kubectl get secret -n avx mongo-key -o=jsonpath='{.data.mongo-init-user}' | base64 -d)
 export MONGO_PASS=$(kubectl get secret -n avx mongo-key -o=jsonpath='{.data.mongo-init-pass}' | base64 -d)
[Install] Run below commands to get Elasticsearch and Kibana credentials
 export ES_PASS=$(kubectl get secret -n avx elasticsearch-pw-elasticsearch -o=jsonpath='{.data.password}' | base64 -d)
 export KIBANA_PASS=$(kubectl get secret -n avx elasticsearch-pw-kibana -o=jsonpath='{.data.password}' | base64 -d
[Install] Application Installation completed successfully
```

Note: Follow the URLs and commands given in the output message to get the credentials and access the application.

13. If installation of the third party monitoring components was not enabled during the entire process, they can be installed later by the following steps:

    a. While installing the third party components (helm_charts/avx_third_party/values.yaml), the only that values are set to 'true' by default are - *prometheus, nodeexporter, kube-state metrics*. The other components are set as 'false' by default and must be to set to true if they are to be enabled, they are - *elk-elasticsearch, elk-filebeat, elk-kibana, elk-logstash, grafana, elasticsearch-insight, logstash-syslog.*

    b. Edit the **.appviewxctl.yaml** file and set **install.enableThirdPartyInstall** to ‘true’

c. Configure the following **thirdPartyApp** parameters as true as per the requirements:
   - **install.thirdPartyApp.elk**
   - **install.thirdPartyApp.monitoring**
   - **install.thirdPartyApp.insight**

d. Now, edit the file **values.yaml** present at location helm_charts/appviewx_monitoring/prometheus/chart/values.yaml and append the below values at the end of the file (only if that are not present).

```
limits:
  cpu_limit: 80
  memory_limit: 80
  disk_limit: 80
  timelimit_cpu_memory: 5
  timelimit_disk: 1
  timelimit_pod: 1
  timelimit_node: 1
```

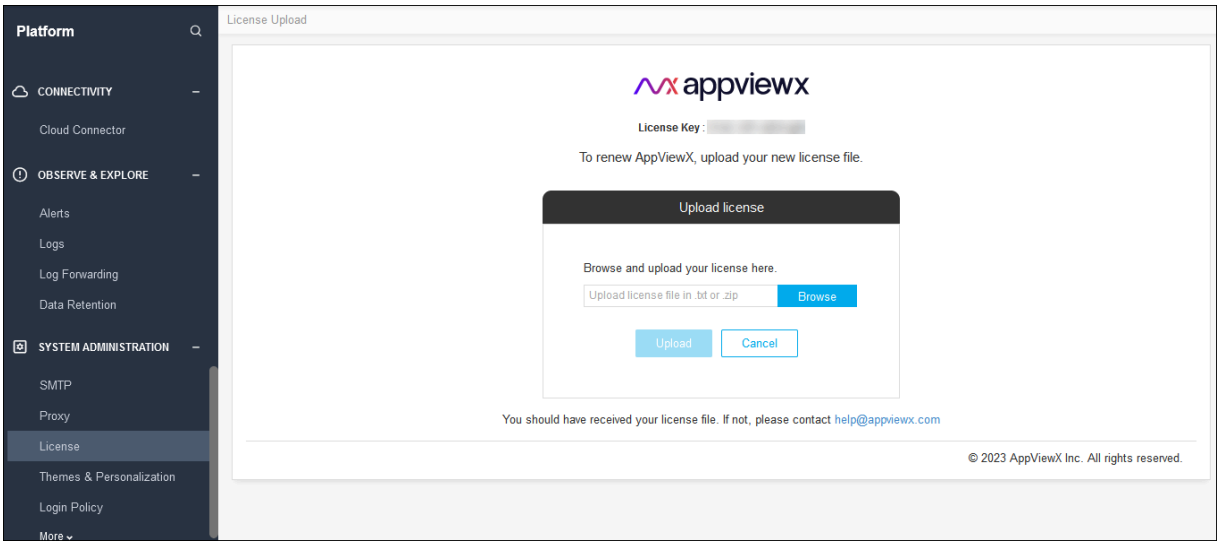
e. Run the command below

```
./appviewxctl installtpt --config .appviewxctl.yaml
```

Customers migrating from AppViewX version 2020.3.0 to Managed Kubernetes FP3, it is mandatory to upgrade the license.

To upgrade the license

1. Login to the AppViewX with valid credentials.
2. Navigate to Platform >> System Administration >> License page.
3. Click **Upgrade License**.



4. Click **Browse** to find the latest license key file.
5. Click **Upload**.

> **Note:** For the licenses contact AppViewX Support at help@appviewx.com or customerlicences@appviewx.com.

Upgrade AppViewX in Managed Kubernetes

> **Attention:**
>
> - If you are using the self managed private docker registry instead of AppViewX's docker registry, then before proceeding with the upgrade, ensure you have copied the latest images to your registry. The list of images can be found in the Prerequisite section - AppViewX Docker Images.
> - If you are currently using AppViewX v2022.1.0 FP3 (i.e. after applying the infra hotfix for FP3) and already in Kube 1.26, then you must follow these prerequisite steps before upgrading to Hudson or the next infra upgrade:
>
> 1. Execute the command
>
>  ```
>  kubectl get secrets -n avx sh.helm.release.v1.vault.v2 -o json | jq .data.release -r | base64 --decode | base64 --decode | gunzip
>  ```
>
> This creates the file **manifest.json**.
> 2. Open the **manifest.json** using VIM or any other editor.
> 3. Search for parameter **PodDisruptionBudget**, find its API version and change it from **v1beta1** to **v1**. Save the changes.
> 4. Execute the command.

```
DATA=`cat manifest.json | gzip -c | base64 | base64 | tr -d '\n\r'`
```

```
kubectl patch secret -n avx sh.helm.release.v1.vault.v2 --type='json' -p="[{\"op\":\"replace\",\"path\":\"/data/release\",\"value\":\"$DATA\"}]"
```

To upgrade AppViewX with a new image version, follow the steps below:

1. Ensure to take a backup of the MongoDB and Vault for rollback in case something goes wrong during upgrade. Before you take the backup, execute the script below.

   ```
   db.profile.update({'_id' : 'installationType'}, {$set : {"value" : "Managed_K8s"}})
   ```

2. To take the backups, execute the commands below.
   For self-managed mongodb:

   ```
   kubectl create job --from=cronjob/mongo-backup -n avx mongo-backup-<unique-identifier>
   ```

   ```
   kubectl create job --from=cronjob/vault-backup -n avx vault-backup-<unique-identifier>
   ```

   Replace <unique-identifier> in above commands with some random string and run. Monitor the pods until completion and verify the backups are placed in the storage bucket.

   Note: Atlas backup must be taken in the atlas dashboard. Refer to the atlas snapshots section in the page Backup and Restore.

3. Navigate to the installer directory.
4. Edit the **appviewxctl.yaml** file's upgrade section for the parameters mentioned below.

appviewxctl.yaml file - Parameters and Description

Parameters	Description of Values
upgrade.imageRegistry	The URL of the container registry where the images are to be pulled from by the pods. *Example*: appviewx.azureacr.io
upgrade.imageTag	The tag of the image that will be used for installation. *Example*: 2023.1.0_FP_750-alpine
upgrade.isSaasEnabled	Boolean value for SaaS enablement. This value should be set to **true** for Managed K8s.

Parameters	Description of Values

upgrade.plugins	The list of plugins that will be installed. Each plugin will have three fields
	• enable
	• imageTag
	• name
	
	Set enable to **true** if the plugin is to be upgraded. If the same image tag is to be used as defined in the global ImageTag keep it **latest** otherwise override with some other tag of your choice.
	
	Note: The list of plugins to be enabled should match the ones in the install section.
	
	Example:
	`- enable: true`
	`imageTag: latest`
	`name: avx-config-server`

5. Add the following component parameters in the **appviewxctl.yaml** file.

Parameters	Description of Values
install.thirdPartyApp.elk	Boolean value to add Elk component. Set to True for upgrade.
install.thirdPartyApp.monitoring	Boolean value to add Monitoring component. Set to True for upgrade.
install.thirdPartyApp.insight	Boolean value to add Insight component. Set to True for upgrade.

6. Update the following install parameters in the **appviewxctl.yaml** file required to integrate the kubernetes cluster to the external vault.

appviewxctl.yaml file - Parameters and Description

Parameters	Description of Values
cloudConnectorEnabled	A boolean value (true/false) to denote the cloud connector usage for southbound communications. If a cloud connector is used set the value to **true**.
install.externalVault.enable	A boolean value (true/false) to denote if the external vault is to be used in the setup. True is to enable the external vault.
install.externalVault.externalVaultAddr	Contains the vault URL and listening port *Example*: https://pm-lxs-node01.lab.appviewx.net:8200
install.externalVault.externalVaultAuthRole	Name of the role created against the access kubernetes auth path
install.externalVault.externalVaultCACertSecret	Name of the secret where **vault-ca.crt** file is mounted.
install.externalVault.externalVaultDBRole	Static role created to access the database cred.
install.externalVault.externalVaultEnginePath	Enter the value “/database”
install.externalVault.externalVaultKubeAuthPath	The Kuberenetes access path created with cluster information for service account authentication.
install.externalVault.externalVaultSAName	The Service account used to create externalVaultAuthRole.
install.externalVault.mongoPasswordVaultEngine	Enter the value DATABASE

Note: Path parameters should have a leading forward slash '/'.

7. Before performing the Infra Upgrade, update the following parameters.

a. Add the following additional plugins in the install and upgrade section of the appviewxctl.yaml file before proceeding with the upgrade:
   - avx-subsystem-codesigning
   - avx-python-sandbox-sync
   - avx-python-sandbox
   - avx-platform-aep-gateway

   Sample with default values:

```
- enable: true
  imageTag: latest
  name: avx-platform-aep-gateway
- enable: false
  imageTag: latest
  name: avx-subsystem-codesigning
- enable: true
  imageTag: latest
  name: avx-python-sandbox-sync
- enable: true
  imageTag: latest
  name: avx-python-sandbox
```

b. **appviewxctl.yaml file - Parameters and Description**

Parameters	Description of Values
upgrade.upgradeInfra	Boolean value to upgrade infra component. Set to True for upgrade.
upgrade.upgradeThirdParty	Boolean value to upgrade the monitoring (ELK, insight, and monitoring) components. Set to True for upgrade.

8. Download the upgrade tar file (**upgrade.tar.gz**) from the release portal and extract it to a suitable location. (The extracted files contain the binary and helm charts tar.)
9. Navigate to the folder where the upgrade tar is extracted.
10. Copy the appviewxctl binary from the current folder (extracted folder location) to the installer location.

```
cp appviewxctl <absolute path of the installer directory>
```

11. To upgrade AppViewX infra, execute the command

> **Note:** If you plan on enabling additional 3pt monitoring components as part of the infra upgrade do the following:
>
> a. Navigate to <installer>/helm_charts/avx_thrid_party/.
>
> b. Edit the **values.yaml** file.
>
> c. Set "enable" to true for the components you wish to enable as part of the upgrade.

```
./appviewxctl infraUpgrade --config .appviewxctl.yaml
```

This will prompt the following message

```
Please provide the path of updated helm charts tar. :
```

Enter the absolute path (extracted file path) of the new helm charts artifact.

12. After the infra upgrade is complete, execute the command

```
./appviewxctl upgrade --config .appviewxctl.yaml
```

> **Note:** It is mandatory to carry out the Infra upgrade before the plugin upgrade.

Rollback Steps

a. Restore the DB using the restore scripts (step 11 in the Installation Steps section) for self-managed DB or in atlas using snapshot restore in the dashboard.

b. Update the **appviewxctl.yaml** upgrade section's values to the previous image tag and re-run the upgrade command.

Downloading Images from AppViewX Repository

Prerequisites

1. Get the source image repository credentials from AppViewX.
2. Configure the docker using the command

```
docker login -u ${USERNAME} -p ${PASSWORD} ${DOCKER_REPOSITORY}
```

3. Configure the respective cloud provider CLI (Azure) and ensure you have access to push docker images to ACR.

The script for image push and pull is as follows:

```
appVersion=$1 # App image version. E.g: 2022.1.0_FP_750-alpine

targetImageRegistry=$2 # Image resgistry name


# Validate required inputs

if [ -z "$appVersion" ] || [ -z "$targetImageRegistry" ];then

{

    echo "Please provide script parametes as ./script.sh <appVersion> <targetImageRegistry>"

    exit

}

fi


# Set the registry login

if echo $targetImageRegistry | grep -iq "amazonaws";then

{

    registryProvider="ecr"

    region=$(echo $targetImageRegistry | cut -d "." -f4)

    aws ecr get-login-password --region $region | docker login --username AWS --password-stdin $targetImageRegistry

}

elif echo $targetImageRegistry | grep -iq "azurecr";then

{

    registryProvider="acr"

    az acr login -n $targetImageRegistry

}

elif echo $targetImageRegistry | grep -iq "gcr";then

{

    registryProvider="gcr"

    gcloud auth print-access-token | docker login -u oauth2accesstoken \

    --password-stdin $(echo $targetImageRegistry | cut -d '/' -f2)

}

else

{

    echo "Unknown regrsity provider"

    exit 2

}

fi


# Image tag mappings
```

```
imageTags='[
  {
    "imageName": "avx-cloud-managedservice",
    "tagVersion": "appVersion",
    "upload": true
  },
  {
    "imageName": "avx-cloud-web",
    "tagVersion": "appVersion",
    "upload": true
  },
  {
    "imageName": "avx-cloud-gateway",
    "tagVersion": "appVersion",
    "upload": true
  },
  {
    "imageName": "avx-platform-report-generator",
    "tagVersion": "appVersion",
    "upload": true
  },
  {
    "imageName": "mongo-init",
    "tagVersion": "appVersion",
    "upload": true
  },
  {
    "imageName": "avx-cloud-mongoseed",
    "tagVersion": "appVersion",
    "upload": true
  },
  {
    "imageName": "alpine",
    "tagVersion": "3.17.2",
    "upload": true
  },
  {
```

```
    "imageName": "pilot",
    "tagVersion": "1.16.2",
    "upload": true
},
{
    "imageName": "proxyv2",
    "tagVersion": "1.16.2",
    "upload": true
},
{
    "imageName": "istio-operator",
    "tagVersion": "1.16.2",
    "upload": true
},
{
    "imageName": "consul",
    "tagVersion": "1.10.3",
    "upload": true
},
{
    "imageName": "vault",
    "tagVersion": "1.8.4",
    "upload": true
},
{
    "imageName": "redis",
    "tagVersion": "6.2.3",
    "upload": true
},
{
    "imageName": "kafka",
    "tagVersion": "1.1.0-kafka-2.6.0",
    "upload": true
},
{
    "imageName": "kafka",
    "tagVersion": "1.1.0-kafka-2.7.0",
```



```
    "upload": true
},
{
    "imageName": "kafka",
    "tagVersion": "1.1.0-kafka-2.8.0",
    "upload": true
},
{
    "imageName": "operator",
    "tagVersion": "1.1.0",
    "upload": true
},
{
    "imageName": "kube-metrics-adapter",
    "tagVersion": "v0.1.16",
    "upload": true
},
{
    "imageName": "kibana",
    "tagVersion": "7.15.1",
    "upload": true
},
{
    "imageName": "grafana",
    "tagVersion": "8.5.0",
    "upload": true
},
{
    "imageName": "filebeat",
    "tagVersion": "7.15.1",
    "upload": true
},
{
    "imageName": "logstash",
    "tagVersion": "7.15.1",
    "upload": true
},
```



```
  {
    "imageName": "logstash-syslog",
    "tagVersion": "7.6.0",
    "upload": true
  },
  {
    "imageName": "elasticsearch",
    "tagVersion": "7.15.1",
    "upload": true
  },
  {
    "imageName": "elasticsearch-insight",
    "tagVersion": "7.16.3",
    "upload": true
  },
  {
    "imageName": "prometheus",
    "tagVersion": "v2.35.0",
    "upload": true
  }
]'

for row in $(echo "${imageTags}" | jq -r '.[] | @base64'); do
  _jq() {
   echo ${row} | base64 --decode | jq -r ${1}
  }
  imageUpload=$(_jq '.upload')
  tagVersion=$(_jq '.tagVersion')
  if [ $imageUpload == "true" ];then
  {
    if [ "${tagVersion}" == "appVersion" ];then
    {
      docker pull docker.io/appviewx/$(_jq '.imageName'):$appVersion
      docker tag docker.io/appviewx/$(_jq '.imageName'):$appVersion $targetImageRegistry/appviewx/$(_jq '.imageName'):$appVersion
      docker push $targetImageRegistry/appviewx/$(_jq '.imageName'):$appVersion
    }
    else
```

```
    {
        docker pull docker.io/appviewx/$(_jq '.imageName'):$(_jq '.tagVersion')
        docker tag docker.io/appviewx/$(_jq '.imageName'):$(_jq '.tagVersion') $targetImageRegistry/appviewx/$(_jq '.imageName'):$(_jq '.tagVersion')
        docker push $targetImageRegistry/appviewx/$(_jq '.imageName'):$(_jq '.tagVersion')
    }
    fi
  }
  fi
done
```

Execute the Image Push-Pull Script

To execute the above image push-pull script, run the command

```
./avx_image_pull_push.sh <image-tag> <targetImageRegistry>
```

Kubernetes Version Upgrade in AKS

When upgrading a supported AKS cluster, Kubernetes minor versions can't be skipped. All upgrades must be performed sequentially by major version number. For example, upgrades between *1.14.x -> 1.15.x* or *1.15.x -> 1.16.x* are allowed, however *1.14.x -> 1.16.x* is not allowed. The upgrades must be performed sequentially through the available minor versions.

For example, if AKS cluster is currently using Kubernetes version 1.22.11, but needs to be upgraded to version 1.24.6. then you must perform the sequential upgrade of versions from 1.22.11 to 1.23.13, and then from version 1.23.13 to 1.24.6.

- Steps to Upgrade the Kubernetes Version in AKS

Prerequisites

- To retrieve information about the current Kubernetes context execute the command:

```
kubectl config get-contexts
```

  This command is part of the kubectl utility, which interacts with Kubernetes clusters.

- To find the current context, you can simply execute the following command:

```
kubectl config current-context
```

Steps to Upgrade the Kubernetes Version in AKS

Step 1 - Upgrade Validation

To check which Kubernetes releases are available for your cluster, use the az aks get-upgrades command. The following example checks for available upgrades to myAKSCluster in myResourceGroup:

Syntax

```
az aks get-upgrades --resource-group myResourceGroup --name myAKSCluster --output table
```

Example

```
az aks get-upgrades --resource-group appviewx_kt_IU_RG --name appviewx_kt_IU --output table
```

Step 2 - Verify Nodepool List

List node pools in the managed Kubernetes cluster. To get a list of nodes in the cluster run `kubectl get nodes` command.

Syntax

```
az aks nodepool list --resource-group MyResourceGroup --cluster-name MyManagedCluster -o table
```

Example

```
az aks nodepool list --resource-group appviewx_kt_IU_RG --cluster-name appviewx_kt_IU -o table
```

Step 3 - Verify and Set Max Surge Value

By default, AKS configures upgrades to surge with one extra node. A default value of one for the max surge settings will enable AKS to minimize workload disruption by creating an extra node before the cordon/drain of existing applications to replace an older versioned node. The max surge value may be

customized per node pool to enable a trade-off between upgrade speed and upgrade disruption. By increasing the max surge value, the upgrade process completes faster, but setting a large value for max surge may cause disruptions during the upgrade process.

For production node pools, we recommend a max_surge setting of 33%.

Verify the Max Surge Value

To verify the current max surge value in the nodepool run the commands below.

Syntax

```
az aks nodepool list --cluster-name MyManagedCluster --resource-group MyResourceGroup
```

Example

```
az aks nodepool list --cluster-name appviewx_kt_IU --resource-group appviewx_kt_IU_RG
```

or

```
az aks nodepool list --cluster-name appviewx_kt_IU --resource-group appviewx_kt_IU_RG | grep -i maxsurge
```

Update the Max surge Value

To update max surge value for an existing node pool

Syntax

```
az aks nodepool update -n mynodepool -g MyResourceGroup --cluster-name MyManagedCluster --max-surge 33%
```

Example

Since the cluster has three nodepools, the commands for each nodepool are as follows:

- `az aks nodepool update -n workernodeiu -g appviewx_kt_IU_RG --cluster-name appviewx_kt_IU --max-surge 33%`
- `az aks nodepool update -n agentpool -g appviewx_kt_IU_RG --cluster-name appviewx_kt_IU --max-surge 33%`
- `az aks nodepool update -n dbpool -g appviewx_kt_IU_RG --cluster-name appviewx_kt_IU --max-surge 33%`

Step 4 - Verify PDB

Decide how many instances can be down at the same time for a short period due to a voluntary disruption. You can specify only one of `maxUnavailable` and `minAvailable` in a single

`PodDisruptionBudget.maxUnavailable` can only be used to control the eviction of pods that have an associated controller managing them.

To verify the pods disruption budgets set for all the pods, run the command below.

```
kubectl get poddisruptionbudgets -A
```

Step 5 - Kube Upgrade

Before performing the kube upgrade, scale down all the AppViewX replicas manually to 0 manually by executing the commands below.

- `kubectl patch hpa -n avx --patch '{"spec":{"minReplicas":1}}' $(kubectl get hpa -A  | grep avx | awk '{print $2}')`
- `kubectl patch hpa -n avx --patch '{"spec":{"maxReplicas":1}}' $(kubectl get hpa -A  | grep avx | awk '{print $2}')`
- `kubectl scale --replicas=0 -n avx $(kubectl get deploy -n avx | awk '{print "deploy/"$1}'| tail -n +2)`
- `kubectl delete pods -n avx $(kubectl get pods -n avx | grep avx | awk '{print $1}') --force`

With a list of available versions for your AKS cluster, use the `az aks upgrade` command to upgrade. During the upgrade process,

- AKS will add a new buffer node (or as many nodes as configured in max surge) to the cluster that runs the specified Kubernetes version.
- It cordons and drains one of the old nodes to minimize disruption to running applications. If you're using max surge, it will cordon and drain as many nodes at the same time as the number of buffer nodes specified.
- When the old node is fully drained, it will be reimaged to receive the new version, and it will become the buffer node for the following node to be upgraded.
- This process repeats until all nodes in the cluster have been upgraded.
- At the end of the process, the last buffer node will be deleted, maintaining the existing agent node count and zone balance.

Syntax

```
az aks upgrade --resource-group myResourceGroup --name myAKSCluster  --kubernetes-version KUBERNETES_VERSION
```

Example

- First, upgrade to version 1.25.5

```
az aks upgrade --resource-group appviewx_kt_IU_RG --name appviewx_kt_IU --kubernetes-version 1.25.5
```

- Then, upgrade to version 1.26.3

```
az aks upgrade --resource-group appviewx_kt_IU_RG --name appviewx_kt_IU --kubernetes-version 1.26.3
```

It takes a few minutes to upgrade the cluster, depending on the number of nodes present. After the upgrade is completed, check the kube version by executing the command

```
kubectl get no
```

Note: Do not scale up the pods after cluster upgrade as it is handled by the infra upgrade and plugins upgrade followed by.

Uninstall and Cleanup

The process of uninstalling requires one to navigate to the installer directory and execute the following command

```
./appviewxctl uninstall --config .appviewxctl.yaml
```

The following messages are displayed after the uninstall command is executed successfully.

```
./appviewxctl uninstall --config .appviewxctl.yaml

[Init] Using log file at [/avx/appviewxctl-3196327299.log] to dump logs
[Init] Initialise persistent flag config
[Init] Using config file
[Uninstall] Uninstalling appviewx application
[Uninstall] Uninstalling Appviewx application helm chart
[Uninstall] Uninstalling application backup helm chart
[Uninstall] Uninstalling Infra application helm chart
[Uninstall] Uninstalling Third party application helm chart
[Uninstall] Uninstalling IstioOperator from the cluster
[Uninstall] Uninstalling PVCs from the avx namespace
[Uninstall] Uninstalling Pre-requisite helm chart
[Uninstall] Uninstalling Appviewx installed namespaces
[Uninstall] Successfully uninstalled appviewx application and all the related
```

Note: In the Managed K8s environments removal of PVCs do not occur at times as it may require patching PVCs first before deletion. This may cause certain error messages to display, indicating that PVC has changed. In case such an error occurs, re-run the above command to solve the issue and uninstall the application.

Sometimes the namespaces take a longer time to be removed. Hence, post installation, check if namespaces are in the terminating state (use the command: **kubectl get namespace**). If any namespace is in the terminating state, manually remove the namespaces by executing the commands below:

```
kubectl get namespace "istio-operator" -o json | tr -d "\n" | sed "s/\"finalizers\": \[[^]]\+\]/\"finalizers\": []/" | kubectl replace
 --raw /api/v1/namespaces/istio-operator/finalize -f - 2>/dev/null
```

```
kubectl get namespace "istio-system" -o json | tr -d "\n" | sed "s/\"finalizers\": \[[^]]\+\]/\"finalizers\": []/" | kubectl replace
 --raw /api/v1/namespaces/istio-system/finalize -f - 2>/dev/null
```

```
kubectl get namespace "avx" -o json | tr -d "\n" | sed "s/\"finalizers\": \[[^]]\+\]/\"finalizers\": []/" | kubectl replace --raw /api/v1/namespaces/avx/finalize -f -
 2>/dev/null
```

```
kubectl delete ns istio-operator --force 2>/dev/null
```

```
kubectl delete ns istio-system --force 2>/dev/null
```

```
kubectl delete ns avx --force 2>/dev/null
```

Troubleshooting

Error	Resolution
403: Access Forbidden - License not available. This error occurs when only the managed kubernetes cluster is restarted every morning.	In case of managed kubernetes, when the cluster is restarted it is recommend to restart all pods.

AppViewX Install and Upgrade for EKS

This guide provides the prerequisites and the procedure for installing, upgrading, and accessing the AppViewX application..

- AppViewX Architecture
- Architecture Overview

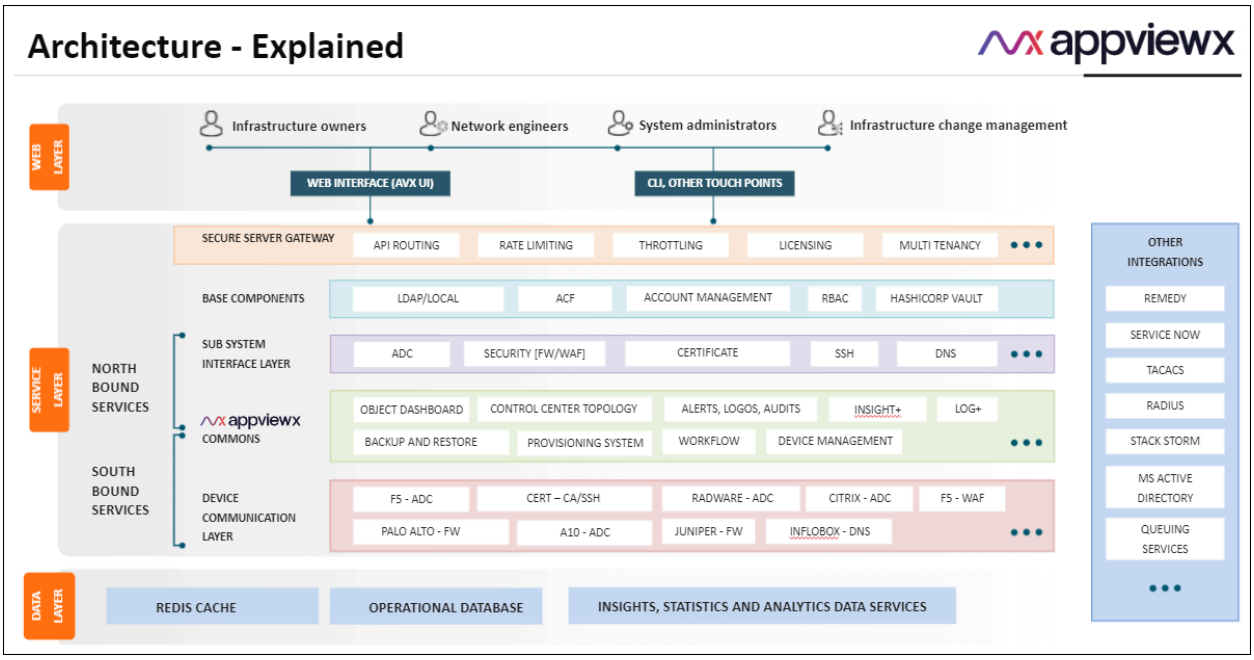
AppViewX Architecture

Architecture Explained

AppViewX is designed based on the microservice architecture and is deployed on Kubernetes—an open-source platform for deploying and managing containers.

The microservice architecture of AppViewX makes it easier to move to containerized workloads and the containers being orchestrated using Kuberneters.

Kubernetes provides container runtime, orchestration, self-healing mechanisms, service discovery and load balancing and it is used for the deployment, scaling, management, and composition of application containers across clusters.



Benefits of AppViewX Architecture

In order to optimally utilize the resources, AppViewX has adopted Kubernetes to achieve higher security by adopting a zero trust network model. The features of AppViewX coupled with Kubernetes are given below.

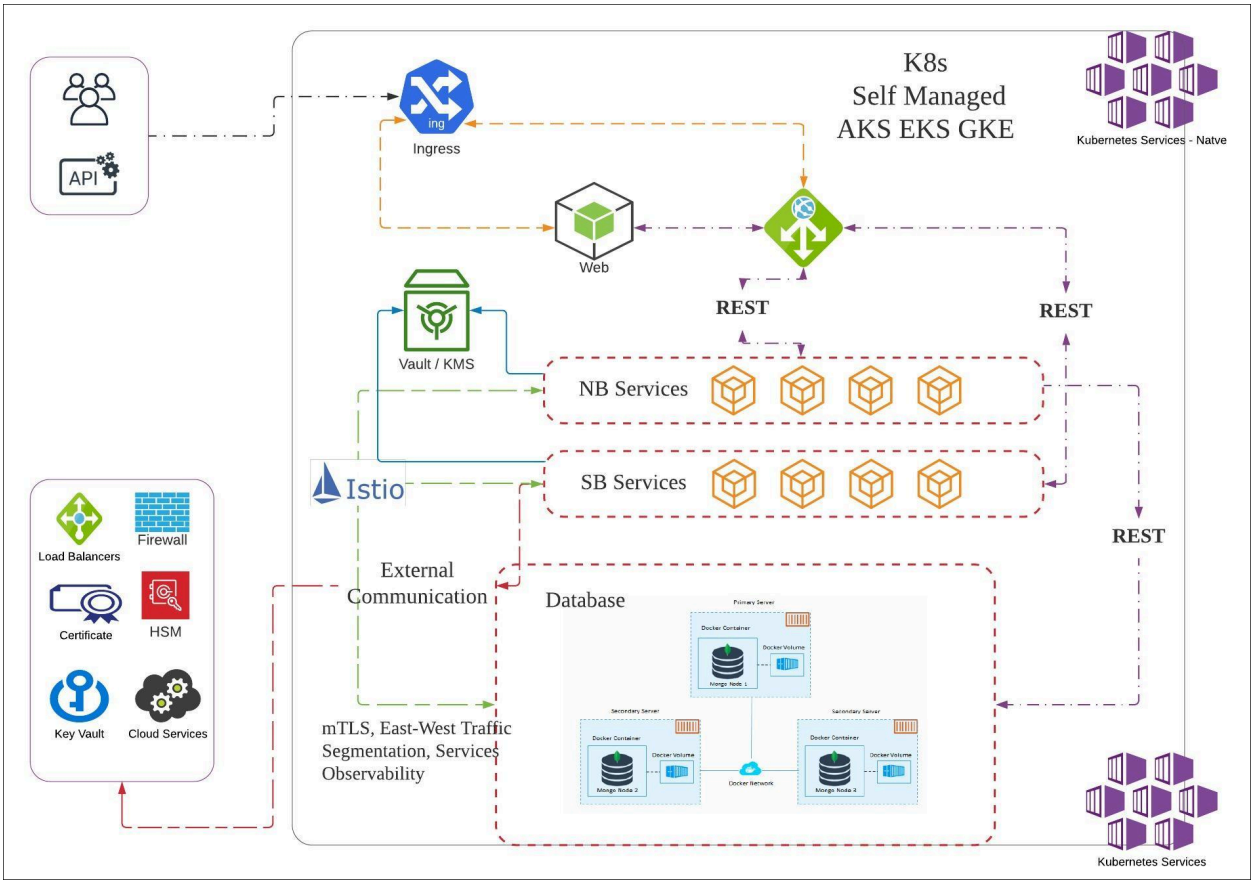
- **Auto scaling** - AppViewX services can have a custom throttling capability based on pre-configured memory configuration per API. This will enable AppViewX services to utilize (scale up) resources optimally as the demand surges and scale down when not in use. This will help to horizontally scale the vendor components on demand and optimize the resource usage.
- **Resiliency** - There is no guarantee that AppViewX services may run without any interruptions and they are bound to fail. Kubernetes keeps deployments healthy by restarting containers that have failed, by killing and replacing unresponsive containers based on health checks. This helps to mitigate the common pain point of the application's upkeep process.
- **Security** - AppViewX architecture is designed around the concept of zero trust network model to enforce tighter security within the Kubernetes cluster. This means no one is trusted by default and requires verification to gain access to the services.

Architecture Overview

AppViewX Kubernetes Architecture

AppViewX workloads are containerized workloads running as microservices and these containers are orchestrated by managed Kubernetes services. Users can prefer the managed k8s platform of their choice.

AppViewX supports deployment on all the three public clouds AWS, Azure and GCP (Google Cloud Platform) using their managed kubernetes engine / services EKS, AKS and GKE specifically.



Benefits of AppViewX Architecture

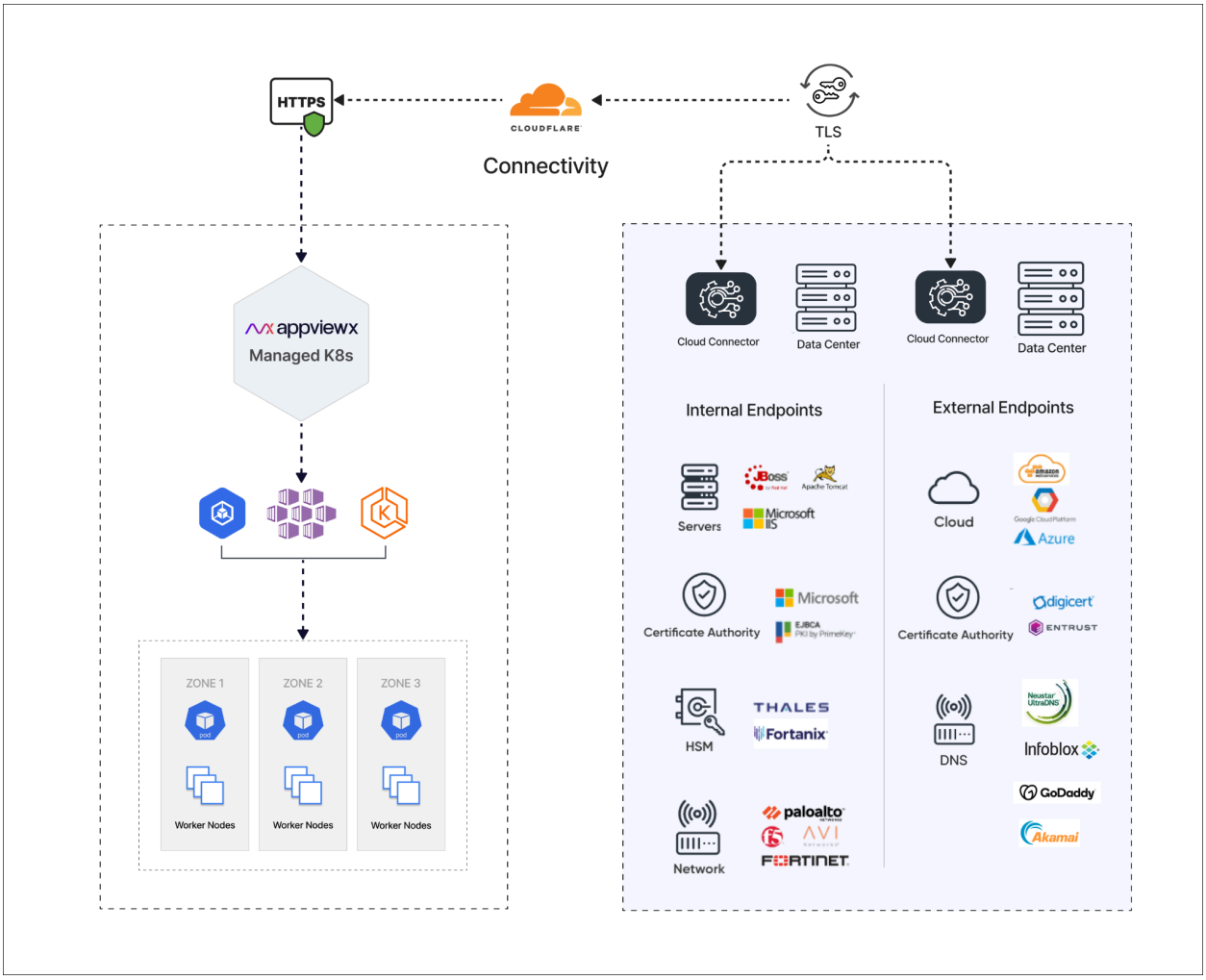
In order to optimally utilize the resources, AppViewX has adopted Kubernetes to achieve higher security by adopting a zero trust network model. The features of AppViewX coupled with Kubernetes are given below.

- **Auto scaling** - AppViewX services can have a custom throttling capability based on pre-configured memory configuration per API. This will enable AppViewX services to utilize (scale up) resources optimally as the demand surges and scale down when not in use. This will help to horizontally scale the vendor components on demand and optimize the resource usage.

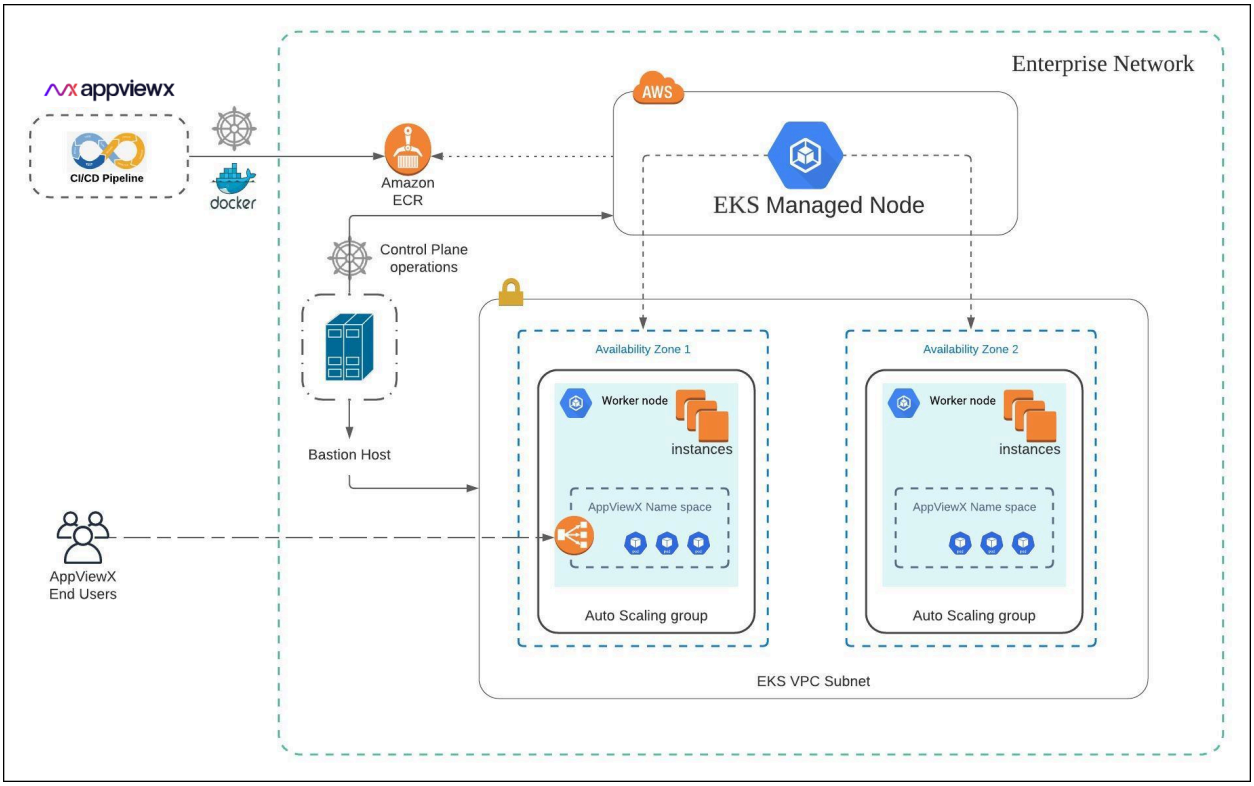
- **Resiliency** - There is no guarantee that AppViewX services may run without any interruptions and they are bound to fail. Kubernetes keeps deployments healthy by restarting containers that have failed, by killing and replacing unresponsive containers based on health checks. This helps to mitigate the common pain point of the application's upkeep process.
- **Security** - AppViewX architecture is designed around the concept of zero trust network model to enforce tighter security within the Kubernetes cluster. This means no one is trusted by default and requires verification to gain access to the services.

AppViewX Deployment Architecture

The figure below shows a standard AppViewX deployment architecture model via managed Kubernetes service for AKS.



EKS Deployment Model

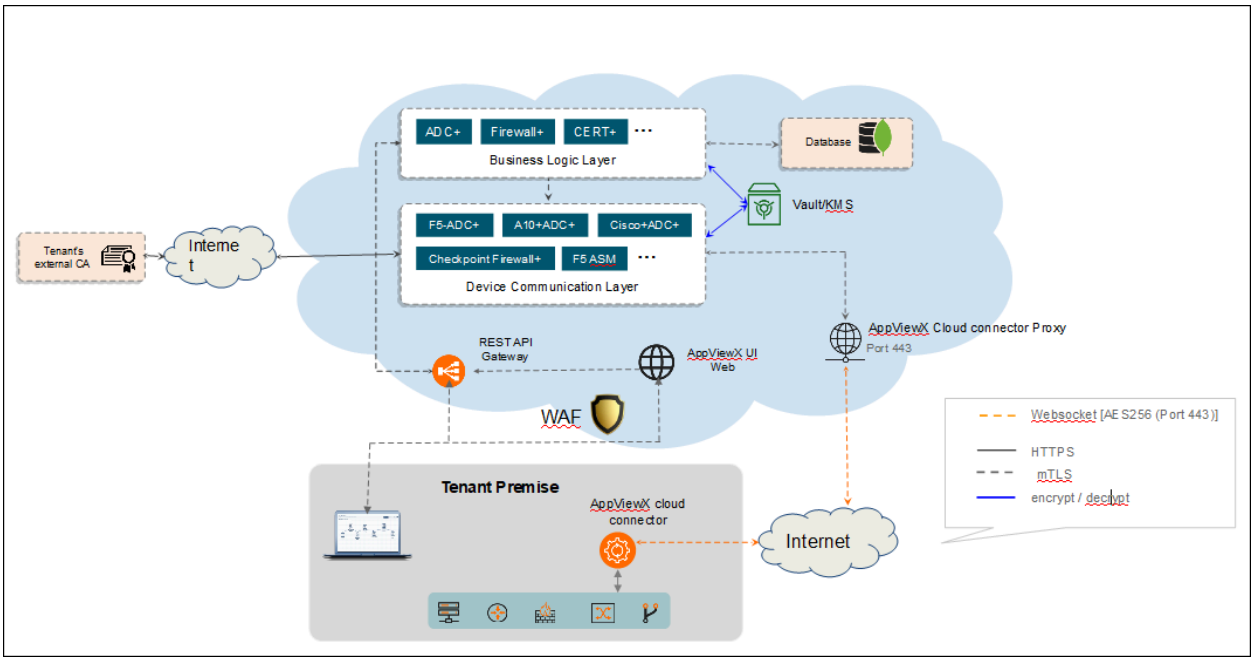


Cloud Connector

AppViewX Cloud Connector is a lightweight plug-in that establishes connectivity between AppViewX Cloud and the Enterprise Network. The cloud connector serves as a secure channel for communication between AppViewX and your enterprise network without requiring any complex network or infrastructure configuration

Key features of the AppViewX Cloud Connector:

- A self-serviceable, Linux-based lightweight setup
- Secure communication between the AppViewX and the AppViewX Cloud Connector using TLS and AES encryption
- Connectivity from the AppViewX to the enterprises' network endpoints
- No complex network setup (Inbound Firewall Whitelisting, VPN setup, and so on)



For more details on cloud connectors refer to AppViewX Cloud Connector User Guide.

Note: The below steps have to be performed in all the cloud connector host machines after the 2022.1.0FP2 to FP3 patch upgrade and before the FP3 cloud connector upgrade.

1. Navigate to the installation path in the cloud connector host machine.
2. Execute the following command:

```
./deps/tools/k3s kubectl get deploy avx-mid-server-starter -n cc -o yaml > starter.yaml && sed -i "s/-Xmx2560m/-Xmx4g/g" starter.yaml

&& ./deps/tools/k3s kubectl replace -f starter.yaml
```

Managed Kubernetes Architecture

Managed Kubernetes clusters are composed of the following main components — a control plane and worker nodes. Each cluster runs in its own, fully managed Virtual Private Cloud (VPC).

- The **control plane** is composed of three master nodes, each running in a different Availability Zone to ensure high availability. Incoming traffic directed to the Kubernetes API passes through the respective cloud service load balancer.
- The **worker nodes** run on virtual instances located in a VPC. Managed kubernetes service engine provides managed node groups with automated lifecycle management. This lets users automatically create, update, or shut down nodes with one operation.

Managed Kubernetes service scales the Kubernetes control plane across multiple Availability Zones of the public cloud to ensure high availability and it automatically scales control plane instances based on load, detects and replaces unhealthy control plane instances, and automatically patches the control plane.

Managed Kubernetes workload instances are deployed in multiple availability zones within the region. Each instance has replicas of the services and nodes which exist across all the virtual instances.

Each zone or instance has an active pod listening to other instances. In case of a failure in any instance, the active pod ensures seamless functioning of the application by activating the nodes from any other working cluster.

EKS Components

The following EKS components are utilized by AppViewX:

- Storage bucket for storing mongodb and vault backups
- Amazon Kubernetes engine

Prerequisites

The following prerequisites must be met before the installation process.

- Managed Kubernetes Version Support Matrix
- Disks Used for AppViewX Installation
- AppViewX Docker Images
- AppViewX Helm Charts
- Bastion Host Setup
- EKS Cluster
- AWS S3 Bucket
- Configuring CSI

Managed Kubernetes Version Support Matrix

Public Cloud	
Mode of Deployment	**Amazon**
Release, Vendor, & Product Support **AppViewX v2023.1.0 FP3**	
Managed K8s Deployment (EKS)	
K8s version 1.29	Yes

Disks Used for AppViewX Installation

Discs Used

Volume	Size	Quantity
logs volume	50Gi	1
avx-kafka	20Gi	3
zookeeper	20Gi	3
consul-server	10Gi	3
mongo-configdb	10Gi	3
mongo-shareddb	256Gi	3
redis	5Gi	3

If a third party is installed, the values are as follows:

Discs Used (Third Party)

Volume	Size	Quantity
Elasticsearch-ELK	10Gi	1
Elasticsearch-Insight	10Gi	1

AppViewX Docker Images

AppViewX Docker images are hosted in a private registry https://images.appviewx.com. These images can be pulled using an authentication token (contact AppViewX Support, help@appviewx.com for the authentication token) and can be hosted in the private or public repository at the customer end.

The list of docker images are

- <registry link>/appviewx/pilot:1.19.0
- <registry link>/appviewx/proxyv2:1.19.0
- <registry link>/appviewx/istio-operator:1.19.0
- <registry link>/appviewx/vault:1.13.7
- <registry link>/appviewx/redis:7.2.0
- <registry link>/appviewx/mongo-init:<tag>
- <registry link>/appviewx/avx-cloud-gateway:<tag>
- <registry link>/appviewx/avx-cloud-web:<tag>
- <registry link>/appviewx/avx-cloud-mongoseed:<tag>
- <registry link>/appviewx/avx-cloud-managedservice-mks:<tag>
- <registry link>/appviewx/avx-platform-report-generator:<tag>
- <registry link>/appviewx/avx-python-sandbox:<tag>
- <registry link>appviewx/avx-mid-server-base:<tag>
- <registry link>/appviewx/consul:1.16.1
- <registry link>/appviewx/kafka:0.32.0-kafka-3.3.1
- <registry link>/appviewx/operator:0.32.0
- <registry link>/appviewx/alpine:3.13.6
- <registry link>/appviewx/kube-metrics-adapter:v0.2.1
- <registry link>/appviewx/kube-state-metrics:v1.9.8
- <registry link>/appviewx/backup-utility-image:v3.0
- <registry link>/appviewx/prometheus:v2.45.0
- <registry link>/appviewx/metrics-server:v0.6.4
- <registry link>/appviewx/elasticsearch:8.9.1
- <registry link>/appviewx/elasticsearch-insight:8.9.1
- <registry link>/appviewx/filebeat:8.9.1
- <registry link>/appviewx/grafana:10.1.1
- <registry link>/appviewx/kibana:8.9.1
- <registry link>/appviewx/logstash:8.9.1
- <registry link>/appviewx/logstash-syslog:8.9.1
- <registry link>/appviewx/alertmanager:v0.26.0

- <registry link>/appviewx/node-exporter:v1.6.1
- <registry link>/appviewx/redis_exporter:v1.53.0

The steps to download the images from AppViewX repository are as follows:

1. Get the source image repository credentials from AppViewX Support team.
2. Configure the docker using the command

```
docker login -u ${USERNAME} -p ${PASSWORD} ${DOCKER_REPOSITORY}
```

3. Configure the respective cloud provider CLI (Google cloud) and ensure you have access to push docker images to GCR.
4. To push the docker images, use the helper script provided by AppViewX. Follow the steps below.

   a. Download the artifact Managed-Kubernetes_helper_scripts.tar.gz to the bastion host and extract using the command:

   ```
   tar -xf Managed-Kubernetes_helper_scripts.tar.gz
   ```

   b. Navigate to the extracted directory **mk8s_helper_scripts**.

   ```
   cd mk8s_helper_scripts
   ```

   c. Execute the script **avx_image_pull_push.sh** using the command

   ```
   ./avx_image_pull_push.sh <Image tag> <customer registry url>
   ```

   Note: Replace <Image tag> and <customer registry url> with the actual values.

AppViewX Helm Charts

The helm charts used by AppViewX for installation are released as a part of the installer. The installer consists of helm charts and an AppViewX utility which helps orchestrate the deployment, patch, upgrade and maintenance of AppViewX across managed kubernetes deployment.

Bastion Host Setup

The following packages must be installed on the bastion host or the host/tool (AWS DevOps) from where the installation is triggered

AWS CLI

To set up the AWS CLI refer to Installing or updating the latest version of the AWS CLI on the AWS documentation website.

Kubectl

To set up Kubectl refer to Install and Set Up kubectl on Linux on the Kubernetes documentation website.

Execute the following commands:

- `sudo curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"`
- `sudo chmod +x kubectlmv`
- `sudo mv ./kubectl /usr/bin/#`

Verify installation by executing the command

```
kubectl version
```

Helm

Helm is required only if the deployment is triggered from any other machine instead of the DevOps pipeline. To set up Helm refer to Installing Helm on the Helm documentation website.

Execute the following command:

- `curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3`
- `chmod 700 get_helm.sh`
- `./get_helm.sh#`

Verify installation by executing the command

```
helm version
```

jq

To set up the jq refer to Download jq on the github.

EKS Cluster

To create an EKS cluster refer to AWS documentation website - Creating an Amazon EKS cluster. Although Microsoft manuals are always up-to-date, the recommended choice to make before creating the cluster is as follows:

- Kubernetes version: 1.29
- User nodepool:
  - **appnodepool**: Three nodes of type **t3.2xlarge** with Auto Scaling disabled
  - **mongonodepool**: Three nodes of type **t3.2xlarge** with Auto Scaling disabled. Add label **mongo=true** and taint **designatedMongo=true:NoSchedule** to the nodepool (to be performed while creating the cluster).

Note: A minimum of 3 availability zones are needed during cluster creation to support the single AZ failover.

- Select multi zones for both Agent nodepool and User nodepool.

Get EKS Cluster kubeconfig

To get the EKS cluster kubeconfig, execute the script below:

```
eksClusterName="<EKS_CLUSTER_NAME>"

aws eks update-kubeconfig --name $eksClusterName --region ap-south-1
```

AWS S3 Bucket

A S3 bucket is required to store

- iControlJar: Directory name is **icontroljar** and the jar has to be placed here.
- MongoDB backup: Directory name should be **mongo-backup**.
- Vault backup: Directory name should be **vault-backup**.
- Axisjar: Directory name should be **axisjar**.

Lets understand the different approaches to create a S3 bucket and configure S3 buckets that are accessible by EKS nodes.

Approach 1

In this approach,

1. Create a bucket.
2. Create an IAM policy.
3. Attach this policy to the node groups with read/write access to the bucket.

Approach 2 (Recommended)

A standard and secure way of attaching permissions to pods in kubernetes are the AWS IRSA (IAM role for service account). Users can create a role and policy and then add an annotation to the pod service account. Follow the AWS official documentation website - IAM roles for service accounts.

The steps to create a S3 bucket and configure the IAM roles for IRSA are as follows:

- This step can also be performed using a helper script provided by AppViewX. To use this script follow the steps below.

  1. Download the artifact Managed-Kubernetes_helper_scripts.tar.gz to the bastion host and extract using the command:

     ```
     tar -xf Managed-Kubernetes_helper_scripts.tar.gz
     ```

  2. Navigate to the extracted directory **mk8s_helper_scripts**.

     ```
     cd mk8s_helper_scripts
     ```

  3. Edit the file **eks_config.sh** and replace <actualBucketname>, <actualAccountNumber>, <eksClusterName>, and <awsRegionName> with the actual values.

  4. Execute the **eks_config.sh** file.

     ```
     bash eks_config.sh
     ```

> **Attention:** Please enter the actual values in the script below before executing it.

After the script is executed,

- Capture the output **Annotation** which is required in the global utility config. (This value must be added to the sub-field **serviceAccountAnnotation** of the parameter **storageAccess**.)
- Configure the **Authentication to AWS ECR** (AWS Image registry) to pull images from ECR.
- Get the **Image registry** name (images are stored here) and the **AccessKey/secretKey** which are required in the global utility config.

Configuring CSI

To configure CSI

1. Execute the command below

```
helm repo add aws-ebs-csi-driver https://kubernetes-sigs.github.io/aws-ebs-csi-driver
```

2. Execute the command below

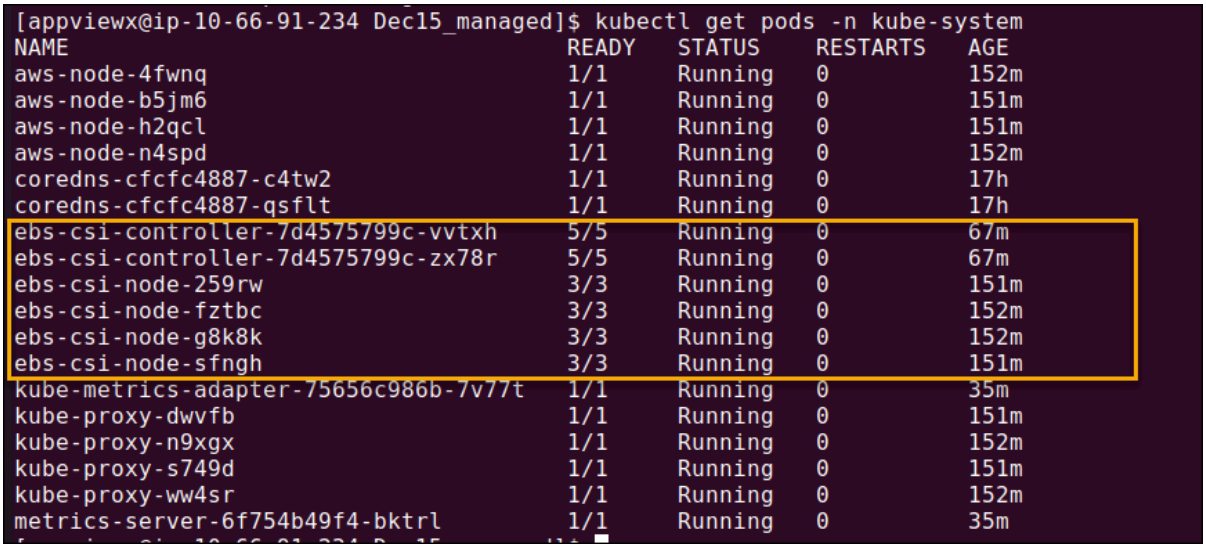
```
helm repo update
```

3. Execute the command below

```
helm upgrade --install aws-ebs-csi-driver --namespace kube-system aws-ebs-csi-driver/aws-ebs-csi-driver
```

4. Verify the status of the pods (CSI) by executing the command:

```
kubectl get pods -n kube-system
```



5. The creation of Amazon EBS CSI plugin IAM role with the AWS CLI is handled in the script contained in the **Approach 2 (Recommended)** section of the topic AWS S3 Bucket. If you consider **Approach 1** in the previous section, refer the following guide Amazon EKS User Guide - Create CSI IAM Role to perform the role creations.

> **Attention:** Ignore Step 4 from the content in the AWS CLI tab if you are not using encrypted volume.

Install AppViewX in Managed Kubernetes

Migration Strategy

Attention: If you are performing a fresh install, then refer the next sub-topic **Installation Steps**.

To migrate from AppViewX on-prem versions (2022.1.0, 2021.1.0, and 2020.3.0) to Managed Kubernetes, it is important to take a backup of the mongodb and vault in the respective on-prem versions. Before you take the backup, execute the script below.

```
db.profile.update({'_id' : 'installationType'}, {$set : {"value" : "Managed_K8s"}})
```

Note: Refer to the specific version of the release documents from the release portal and perform the backups or contact the AppViewX support team.

After performing the backup, follow the installation steps detailed in the section below. At step 10 of the installation process, ensure to restore the data at this stage.

Installation Steps

This section describes the steps to for installing the AppViewX Stack on EKS.

1. Download the installer from the release portal.
2. Create a directory Managedk8s-installer in the bastion host and extract the installer file**tar -xf installer.tar.gz** in the same directory.
3. Verify that the extracted installer must have the following files
   - appviewxctl (binary)
   - helm_charts (directory of helm charts)
4. Generate the configuration files based on the cloud provider. If the cloud provider is **Amazon**, execute the command below.

   ```
   ./appviewxctl config generate --provider aws
   ```

5. Verify that the execution of the above command creates the configuration files named **.appviewxctl.yaml** in the same location.
6. The file .appviewxctl will be populated with the fields necessary for installation, in particular cloud provider that was provided in the previous command (**-- provider**).
7. Edit the .appviewxctl.yaml file and populate the values as described below:

appviewxctl.yaml file - Parameters and Description

Parameters	Description of Values
chartPath	The path to the helm_charts which is to be installed. It points to the helm_charts directory extracted in step 3.
configFile	The path to the kube config file to be used by helm and kubectl. If the bastion host is already configured and kube config is under $HOME/.kube directory, then keep this field empty.
install.enableAppBackupCron	Boolean value to enable/disable the backup cronjobs. (True/False). This value is needed for self-managed mongodb only. For atlas backup this has to be scheduled in the atlas dashboard.
install.enablePrivateImagePullSecret	Boolean value to enable image pull secret. Set values as **false** if the cluster already has access to the container registry. Otherwise set it to **true** and fill all the details of the access keys described in below sections.
install.enableThirdPartyInstall	Boolean value (True/False) to determine whether third party monitoring components such as ELK, Monitoring, and Insight needs to be installed.
install.thirdPartyApp.elk	Boolean value to add Elk component. Set to True if it needs to be installed.
install.thirdPartyApp.monitoring	Boolean value to add Monitoring component. Set to True if it needs to be installed.
install.thirdPartyApp.insight	Boolean value to add Insight component. Set to True if it needs to be installed.

Parameters	Description of Values
install.imageRegistry	The URL of the container registry where the images are to be pulled from by the pods.
install.imageTag	The tag of the image that will be used for installation. *Example*: 2023.1.0_FP_750-alpine
install.isSaasEnabled	Boolean value to enable SaaS. This value should be set to **true** for Managed K8s.
install.kafkaCloudConnector	It is a combination of three values. • enable • password • user Set **enable** to ***true*** and keep the user, password fields empty for Managed K8s. *Example* kafkaCloudConnector: enable: true password: "" user: ""
install.mongo	It is a combination of fields specific to the type of mongodb used.
dbIsolation	Boolean value to indicate whether the database isolation is to be enabled. In order for database isolation to work, the following prerequisite must be taken care of while creating the cluster node group. • Add label **mongo=true** and taint **designatedMongo=true:NoSchedule** to the nodepool to be used for mongodb.
mongoAtlas	The fields specific to mongodb atlas are as follows:

Parameters	Description of Values

	• **enable**: Boolean value to decide if mongodb atlas to be used. If set to *false*, a self managed mongodb cluster will be created. If set to *true* mongodb atlas will be used and details of which are to be provided in below mentioned fields.
	• **host**: URL of the mongodb atlas cluster.
	• **password**: password of the mongodb atlas cluster.
	• **user**: username in the mongodb atlas cluster.
	
	Example:
	
	`mongo:`
	`  dbIsolation: false`
	`  mongoAtlas:`
	`  enable: true`
	`  host: "managed-k8s.test.mongodb.net"`
	`  password: "samplepassword"`
	`  user: "user1"`

install.useDockerPrivateRegistry	Set this to **true** if the dockerhub private repository is to be used for pulling the necessary images needed. Otherwise set the value **false** and the container registry ACR, ECR, and GCR will be used based on the cloud provider.
	
	If this value is set to *true*, populate the below values, otherwise keep it empty.
	• **dockerhub.pass**: password to be used for authenticating in the dockerhub private repository.
	• **dockerhub.username**: username configured in the dockerhub private repository.
	
	Example:
	
	`useDockerPrivateRegistry: true`
	`dockerhub:`

Parameters	Description of Values
	pass: "testpassword" username: "appviewx"
install.size	The size of the installation. Based on the use cases and number of certs to be managed there different sizes (contact AppViewX for sizing recommendations). The sizes supported are (case sensitive values) • xsmall • small • medium • large • xlarge • custom *Example:* size: small **Note:** The size provided must be taken into cluster creation and nodegroup sizes must be defined accordingly. Follow the same document link above for nodegroup sizes.
install.plugins	The list of plugins that will be installed. Each plugin will have three fields • enable • imageTag • name Set enable to **true** if the plugin is to be installed. If the same image tag is to be used as defined in the global ImageTag keep it **latest** otherwise override with some other tag of your choice. *Example:*

Parameters	Description of Values

	<pre>- enable: true
	imageTag: latest
	name: avx-config-server</pre>
	To enable Cloud DC support in Managed Kubernetes, set plugins as follows:
	<pre>- enable: true
	imageTag: latest
	name: avx-mid-server-platform</pre>
	Note: Ensure that **install.isSaasEnabled** and **install.kafkaCloudConnector** are set to **true**.

storageAccess	This parameter contains two sub fields bucketObject and serviceAccountAnnotation as described below. • **bucketObject**: name of the S3 bucket created in the topic AWS S3 Bucket. • **serviceAccountAnnotation**: the Annotation value captured in the **Approach 2 (Recommended)** section of the topic AWS S3 Bucket.
internalLoadBalancer	If set to **true**, all the Loadbalancers will be private and can only be accessed within the VPC else it will be public.
install.enableSftpStorage	Change to true to use SFTP server for mongodb, vault, and icontrol.jar storage. Boolean (Default: false)
install.sftpServerDetails. dbBackupPath	Provide the location of mongodb backup storage directory. String (Default: “”)
install.sftpServerDetails. vaultBackupPath	Provide the location of vault backup storage directory. String (Default: “”)

Parameters	Description of Values
install.sftpServerDetails. sftpServerUserName	Provide the username of SFTP server. String (Default: “”)
install.sftpServerDetails. sftpServerIp	Provide the sftp server IP. String (Default: “”)
cloudConnectorEnabled	A boolean value (true/false) to denote the cloud connector usage for southbound communications. If a cloud connector is used set the value to **true**.

The next fields are to be filled with values that must be collected during the cluster creation and setup process and filled as mentioned below.

appviewxctl.yaml file - Parameters and Description (during cluster creation)

Parameters	Description of Values
install.privateImagePullSecret	In this section populate the details of the access keys needed to authenticate and pull the image from the registry. They are not needed if the Dockerhub is used as described above. • **accessKeyId**: The access key ID of the ECR. • **secretAccessKey**: The secret access key of the ECR. • **registry**: The ECR registry URL *Example:* accessKeyId: AKIAIOSFODNN7EXAMPLE secretAccessKey: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY registry: 467889948468.dkr.ecr.ap-south-1.amazonaws.com

The following fields must be added to integrate the kubernetes cluster to the external vault.

appviewxctl.yaml file - Parameters and Description (for external vault)

Parameters	Description
install.externalVault.enable	A boolean value (true/false) to denote if the external vault is to be used in the setup. True is to enable the external vault.
install.externalVault.externalVaultAddr	Contains the vault URL and listening port

Parameters	Description
	Example: https://pm-lxs-node01.lab.appviewx.net:8200
install.externalVault.externalVaultAuthRole	Name of the role created against the access kubernetes auth path
install.externalVault.externalVaultCACertSecret	Name of the secret where **vault-ca.crt** file is mounted.
install.externalVault.externalVaultDBRole	Static role created to access the database cred.
install.externalVault.externalVaultEnginePath	Enter the value “/database”
install.externalVault.externalVaultKubeAuthPath	The Kuberenetes access path created with cluster information for service account authentication.
install.externalVault.externalVaultSAName	The Service account used to create externalVaultAuthRole.
install.externalVault.mongoPasswordVaultEngine	Enter the value DATABASE

8. Once the values are filled in *.appviewxctl* as described in the step above, proceed with the installation. Before doing so, check if the the preconditions are met by executing the command

```
./appviewxctl preflight --config .appviewxctl.yaml
```

This will prompt if the necessary prerequisites are met.

9. To proceed with installation, execute the command

```
./appviewxctl install --config .appviewxctl.yaml
```

Note: The installation will take several minutes to complete. Upon completion you see the following message:

[Install] Successfully installed Appviewx infra stack

This would imply the completion of infra component setup.

10. This step involves restoring the existing data from the previous AppViewX version's cluster in case there is a need to migrate from the older versions to the Managed K8s version. **Ignore this step if it's a fresh setup with no migration necessary.**

    To restore mongodb and vault fetch the backup files and place them in the bastion in a directory such as /home/user/backup execute the mongo_restore and vault_restore scripts as follows:

    ```
    ./mongo_restore.sh <mongo backup tar filepath>
    ```

    ```
    ./vault_restore.sh -p <vault backup filepath>
    ```

    Attention: If the data is being restored from an older version (2020.3.0 - 2022.1) then use the command

    ```
    ./vault_restore.sh -p <vault backup filepath> --removedek
    ```

    Note:

    - The backup files must have extension as **.tar.gz**
    - The above commands work for a self-managed mongodb setup. Setting up the mongodb atlas requires the installation of mongodb tools in the bastion host as described below.

    For an rpm based OS:

    ```
    echo -e "[mongodb-org-4.2] \nname=MongoDB
     Repository\nbaseurl=https://repo.mongodb.org/yum/redhat/\$releasever/mongodb-org/4.2/x86_64/\ngpgcheck=1\nenabled=1\ngpgkey=https://
    www.mongodb.org/static/pgp/server-4.2.asc" > /etc/yum.repos.d/mongodb-org-4.2.repo
    yum install mongodb-org-shell-4.2.0
    yum install mongodb-org-tools-4.2.0
    ```

    For a debian based OS:

    ```
    wget -qO - https://www.mongodb.org/static/pgp/server-6.0.asc | sudo apt-key add -
    sudo apt-get install gnupg
    wget -qO - https://www.mongodb.org/static/pgp/server-6.0.asc | sudo apt-key add -
    echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/6.0 multiverse" | sudo
     tee /etc/apt/sources.list.d/mongodb-org-6.0.list
    sudo apt-get update
    sudo apt-get install -y mongodb-mongosh
    sudo apt-get install -y mongodb-org-tools
    ```

    Verify if the mongodb restore commands have executed successfully using the command

```
mongorestore -- version
```

11. To proceed with the AppViewX application installation, execute the command:

```
./appviewxctl installapp --config .appviewxctl.yaml
```

Once installation is complete the following messages are displayed:

```
[Install] Appviewx infrastructure chart [avx-app] installed successfully
[Install] Successfully installed Appviewx application stack
[Install] Fetching login URL for app
[Install] Waiting for Public IP allotment for istio service
[Install] AppViewX Web URL: https://34.100.197.159/appviewx/
[Install] AppViewX Gateway URL: https://34.100.197.159/avxmgr/
[Install] Grafana URL: https://34.100.197.159/grafana/
[Install] Kibana URL: https://34.100.197.159/kibana/login
[Install] Run below commands to get mongo user credentials
 export MONGO_USER=$(kubectl get secret -n avx mongo-key -o=jsonpath='{.data.mongo-init-user}' | base64 -d)
 export MONGO_PASS=$(kubectl get secret -n avx mongo-key -o=jsonpath='{.data.mongo-init-pass}' | base64 -d)
[Install] Run below commands to get Elasticsearch and Kibana credentials
 export ES_PASS=$(kubectl get secret -n avx elasticsearch-pw-elasticsearch -o=jsonpath='{.data.password}' | base64 -d)
 export KIBANA_PASS=$(kubectl get secret -n avx elasticsearch-pw-kibana -o=jsonpath='{.data.password}' | base64 -d
[Install] Application Installation completed successfully
```

Note: Follow the URLs and commands given in the output message to get the credentials and access the application.

12. If installation of the third party monitoring components was not enabled during the entire process, they can be installed later by the following steps:

    a. While installing the third party components (helm_charts/avx_third_party/values.yaml), the only that values are set to 'true' by default are - *prometheus, nodeexporter, kube-state metrics*. The other components are set as 'false' by default and must be to set to true if they are to be enabled, they are - *elk-elasticsearch, elk-filebeat, elk-kibana, elk-logstash, grafana, elasticsearch-insight, logstash-syslog.*

    b. Edit the **.appviewxctl.yaml** file and set **install.enableThirdPartyInstall** to ‘true’

    c. Configure the following **thirdPartyApp** parameters as true as per the requirements:

- **install.thirdPartyApp.elk**
- **install.thirdPartyApp.monitoring**
- **install.thirdPartyApp.insight**

d. Now, edit the file **values.yaml** present at location helm_charts/appviewx_monitoring/prometheus/chart/values.yaml and append the below values at the end of the file (only if that are not present).

```
limits:
  cpu_limit: 80
  memory_limit: 80
  disk_limit: 80
  timelimit_cpu_memory: 5
  timelimit_disk: 1
  timelimit_pod: 1
  timelimit_node: 1
```

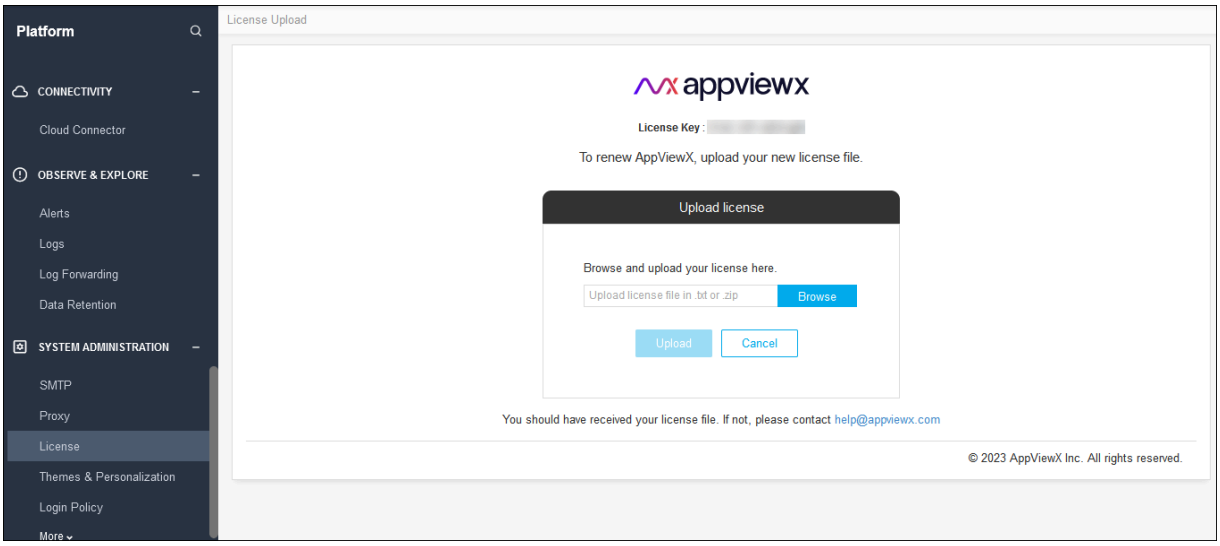
e. Run the command below

```
./appviewxctl installtpt --config .appviewxctl.yaml
```

Customers migrating from AppViewX version 2020.3.0 to Managed Kubernetes FP3, it is mandatory to upgrade the license.

To upgrade the license,

1. Login to the AppViewX with valid credentials.
2. Navigate to Platform >> System Administration >> License page.
3. Click **Upgrade License**.



4. Click **Browse** to find the latest license key file.
5. Click **Upload**.

Note: For the licenses contact AppViewX Support at help@appviewx.com or customerlicences@appviewx.com.

Upgrade AppViewX in Managed Kubernetes

Attention:

- If you are using the self managed private docker registry instead of AppViewX's docker registry, then before proceeding with the upgrade, ensure you have copied the latest images to your registry. The list of images can be found in the Prerequisite section - AppViewX Docker Images.
- If you are currently using AppViewX v2022.1.0 FP3 (i.e. after applying the infra hotfix for FP3) and already in Kube 1.26, then you must follow these prerequisite steps before upgrading to Hudson or the next infra upgrade:
  1. Execute the command

     ```
     kubectl get secrets -n avx sh.helm.release.v1.vault.v2 -o json | jq .data.release -r | base64 --decode | base64 --decode | gunzip
     ```

     This creates the file **manifest.json**.
  2. Open the **manifest.json** using VIM or any other editor.
  3. Search for parameter **PodDisruptionBudget**, find its API version and change it from **v1beta1** to **v1**. Save the changes.
  4. Execute the command.

```
DATA=`cat manifest.json | gzip -c | base64 | base64 | tr -d '\n\r'`
```

```
kubectl patch secret -n avx sh.helm.release.v1.vault.v2 --type='json' -p="[{\"op\":\"replace\",\"path\":\"/data/release\",\"value\":\"$DATA\"}]"
```

To upgrade AppViewX with a new image version, follow the steps below:

1. Ensure to take a backup of the MongoDB and Vault for rollback in case something goes wrong during upgrade. Before you take the backup, execute the script below.

   ```
   db.profile.update({'_id' : 'installationType'}, {$set : {"value" : "Managed_K8s"}})
   ```

2. To take the backups, execute the commands below.
   For self-managed mongodb:

   ```
   kubectl create job --from=cronjob/mongo-backup -n avx mongo-backup-<unique-identifier>
   ```

   ```
   kubectl create job --from=cronjob/vault-backup -n avx vault-backup-<unique-identifier>
   ```

   Replace <unique-identifier> in above commands with some random string and run. Monitor the pods until completion and verify the backups are placed in the storage bucket.

   Note: Atlas backup must be taken in the atlas dashboard. Refer to the atlas snapshots section in the page Backup and Restore. .

3. Navigate to the installer directory.
4. Edit the **appviewxctl.yaml** file's upgrade section for the parameters mentioned below.

appviewxctl.yaml file - Parameters and Description

Parameters	Description of Values
upgrade.imageRegistry	The URL of the container registry where the images are to be pulled from by the pods.
upgrade.imageTag	The tag of the image that will be used for installation. *Example*: 2023.1.0_FP_750-alpine
upgrade.isSaasEnabled	Boolean value for SaaS enablement. This value should be set to **true** for Managed K8s.
upgrade.plugins	The list of plugins that will be installed. Each plugin will have three fields

Parameters	Description of Values
	• enable • imageTag • name Set enable to **true** if the plugin is to be upgraded. If the same image tag is to be used as defined in the global ImageTag keep it **latest** otherwise override with some other tag of your choice. **Note:** The list of plugins to be enabled should match the ones in the install section. *Example:* - enable: true imageTag: latest name: avx-config-server

5. Add the following component parameters in the **appviewxctl.yaml** file.

appviewxctl.yaml file - Parameters and Description

Parameters	Description of Values
install.thirdPartyApp.elk	Boolean value to add Elk component. Set to True for upgrade.
install.thirdPartyApp.monitoring	Boolean value to add Monitoring component. Set to True for upgrade.
install.thirdPartyApp.insight	Boolean value to add Insight component. Set to True for upgrade.

6. Update the following install parameters in the **appviewxctl.yaml** file required to integrate the kubernetes cluster to the external vault.

appviewxctl.yaml file - Parameters and Description

Parameters	Description of Values
cloudConnectorEnabled	A boolean value (true/false) to denote the cloud connector usage for southbound communications. If a cloud connector is used set the value to **true**.
install.externalVault.enable	A boolean value (true/false) to denote if the external vault is to be used in the setup. True is to enable the external vault.
install.externalVault.externalVaultAddr	Contains the vault URL and listening port *Example*: https://pm-lxs-node01.lab.appviewx.net:8200
install.externalVault.externalVaultAuthRole	Name of the role created against the access kubernetes auth path
install.externalVault.externalVaultCACertSecret	Name of the secret where **vault-ca.crt** file is mounted.
install.externalVault.externalVaultDBRole	Static role created to access the database cred.
install.externalVault.externalVaultEnginePath	Enter the value “/database”
install.externalVault.externalVaultKubeAuthPath	The Kuberenetes access path created with cluster information for service account authentication.
install.externalVault.externalVaultSAName	The Service account used to create externalVaultAuthRole.
install.externalVault.mongoPasswordVaultEngine	Enter the value DATABASE

Note: Path parameters should have a leading forward slash '/'.

7. Before performing the Infra Upgrade, update the following parameters.

a. Add the following additional plugins in the install and upgrade section of the appviewxctl.yaml file before proceeding with the upgrade:
   - avx-subsystem-codesigning
   - avx-python-sandbox-sync
   - avx-python-sandbox
   - avx-platform-aep-gateway

   Sample with default values:

```
- enable: true
  imageTag: latest
  name: avx-platform-aep-gateway
- enable: false
  imageTag: latest
  name: avx-subsystem-codesigning
- enable: true
  imageTag: latest
  name: avx-python-sandbox-sync
- enable: true
  imageTag: latest
  name: avx-python-sandbox
```

b. **appviewxctl.yaml file - Parameters and Description**

Parameters	Description of Values
upgrade.upgradeInfra	Boolean value to upgrade infra component. Set to True for upgrade.
upgrade.upgradeThirdParty	Boolean value to upgrade the monitoring (ELK, insight, and monitoring) components. Set to True for upgrade.

8. Download the upgrade tar file (**upgrade.tar.gz**) from the release portal and extract it to a suitable location. (The extracted files contain the binary and helm charts tar.)
9. Navigate to the folder where the upgrade tar is extracted.
10. Copy the appviewxctl binary from the current folder (extracted folder location) to the installer location.

```
cp appviewxctl <absolute path of the installer directory>
```

11. To upgrade AppViewX infra, execute the command

> **Note:** If you plan on enabling additional 3pt monitoring components as part of the infra upgrade do the following:
>
> a. Navigate to <installer>/helm_charts/avx_thrid_party/.
>
> b. Edit the **values.yaml** file.
>
> c. Set "enable" to true for the components you wish to enable as part of the upgrade.

```
./appviewxctl infraUpgrade --config .appviewxctl.yaml
```

This will prompt the following message

```
Please provide the path of updated helm charts tar. :
```

Enter the absolute path (extracted file path) of the new helm charts artifact.

12. After the infra upgrade is complete, execute the command

```
./appviewxctl upgrade --config .appviewxctl.yaml
```

> **Note:** It is mandatory to carry out the Infra upgrade before the plugin upgrade.

Rollback Steps

a. Restore the DB using the restore scripts (step 11 in the Installation Steps section) for self-managed DB or in atlas using snapshot restore in the dashboard.

b. Update the **appviewxctl.yaml** upgrade section's values to the previous image tag and re-run the upgrade command.

Downloading Images from AppViewX Repository

Prerequisites

1. Get the source image repository credentials from AppViewX.
2. Configure the docker using the command

```
docker login -u ${USERNAME} -p ${PASSWORD} ${DOCKER_REPOSITORY}
```

3. Configure the respective cloud provider CLI (AWS) and ensure you have access to push docker images to ECR.

The script for image push and pull is as follows:

```
appVersion=$1 # App image version. E.g: 2022.1.0_FP_750-alpine

targetImageRegistry=$2 # Image resgistry name


# Validate required inputs

if [ -z "$appVersion" ] || [ -z "$targetImageRegistry" ];then

{

    echo "Please provide script parametes as ./script.sh <appVersion> <targetImageRegistry>"

    exit

}

fi


# Set the registry login

if echo $targetImageRegistry | grep -iq "amazonaws";then

{

    registryProvider="ecr"

    region=$(echo $targetImageRegistry | cut -d "." -f4)

    aws ecr get-login-password --region $region | docker login --username AWS --password-stdin $targetImageRegistry

}

elif echo $targetImageRegistry | grep -iq "azurecr";then

{

    registryProvider="acr"

    az acr login -n $targetImageRegistry

}

elif echo $targetImageRegistry | grep -iq "gcr";then

{

    registryProvider="gcr"

    gcloud auth print-access-token | docker login -u oauth2accesstoken \

    --password-stdin $(echo $targetImageRegistry | cut -d '/' -f2)

}

else

{

    echo "Unknown regrsity provider"

    exit 2

}

fi


# Image tag mappings
```

```
imageTags='[
  {
    "imageName": "avx-cloud-managedservice",
    "tagVersion": "appVersion",
    "upload": true
  },
  {
    "imageName": "avx-cloud-web",
    "tagVersion": "appVersion",
    "upload": true
  },
  {
    "imageName": "avx-cloud-gateway",
    "tagVersion": "appVersion",
    "upload": true
  },
  {
    "imageName": "avx-platform-report-generator",
    "tagVersion": "appVersion",
    "upload": true
  },
  {
    "imageName": "mongo-init",
    "tagVersion": "appVersion",
    "upload": true
  },
  {
    "imageName": "avx-cloud-mongoseed",
    "tagVersion": "appVersion",
    "upload": true
  },
  {
    "imageName": "alpine",
    "tagVersion": "3.17.2",
    "upload": true
  },
  {
```

```
    "imageName": "pilot",
    "tagVersion": "1.16.2",
    "upload": true
},
{
    "imageName": "proxyv2",
    "tagVersion": "1.16.2",
    "upload": true
},
{
    "imageName": "istio-operator",
    "tagVersion": "1.16.2",
    "upload": true
},
{
    "imageName": "consul",
    "tagVersion": "1.10.3",
    "upload": true
},
{
    "imageName": "vault",
    "tagVersion": "1.8.4",
    "upload": true
},
{
    "imageName": "redis",
    "tagVersion": "6.2.3",
    "upload": true
},
{
    "imageName": "kafka",
    "tagVersion": "1.1.0-kafka-2.6.0",
    "upload": true
},
{
    "imageName": "kafka",
    "tagVersion": "1.1.0-kafka-2.7.0",
```

```
    "upload": true
},
{
    "imageName": "kafka",
    "tagVersion": "1.1.0-kafka-2.8.0",
    "upload": true
},
{
    "imageName": "operator",
    "tagVersion": "1.1.0",
    "upload": true
},
{
    "imageName": "kube-metrics-adapter",
    "tagVersion": "v0.1.16",
    "upload": true
},
{
    "imageName": "kibana",
    "tagVersion": "7.15.1",
    "upload": true
},
{
    "imageName": "grafana",
    "tagVersion": "8.5.0",
    "upload": true
},
{
    "imageName": "filebeat",
    "tagVersion": "7.15.1",
    "upload": true
},
{
    "imageName": "logstash",
    "tagVersion": "7.15.1",
    "upload": true
},
```



```
  {
    "imageName": "logstash-syslog",
    "tagVersion": "7.6.0",
    "upload": true
  },
  {
    "imageName": "elasticsearch",
    "tagVersion": "7.15.1",
    "upload": true
  },
  {
    "imageName": "elasticsearch-insight",
    "tagVersion": "7.16.3",
    "upload": true
  },
  {
    "imageName": "prometheus",
    "tagVersion": "v2.35.0",
    "upload": true
  }
]'

for row in $(echo "${imageTags}" | jq -r '.[] | @base64'); do
  _jq() {
   echo ${row} | base64 --decode | jq -r ${1}
  }
  imageUpload=$(_jq '.upload')
  tagVersion=$(_jq '.tagVersion')
  if [ $imageUpload == "true" ];then
  {
    if [ "${tagVersion}" == "appVersion" ];then
    {
      docker pull docker.io/appviewx/$(_jq '.imageName'):$appVersion
      docker tag docker.io/appviewx/$(_jq '.imageName'):$appVersion $targetImageRegistry/appviewx/$(_jq '.imageName'):$appVersion
      docker push $targetImageRegistry/appviewx/$(_jq '.imageName'):$appVersion
    }
    else
```

```
    {
        docker pull docker.io/appviewx/$(_jq '.imageName'):$(_jq '.tagVersion')
        docker tag docker.io/appviewx/$(_jq '.imageName'):$(_jq '.tagVersion') $targetImageRegistry/appviewx/$(_jq '.imageName'):$(_jq '.tagVersion')
        docker push $targetImageRegistry/appviewx/$(_jq '.imageName'):$(_jq '.tagVersion')
    }
    fi
  }
  fi
done
```

Execute the Image Push-Pull Script

To execute the above image push-pull script, run the command

```
./avx_image_pull_push.sh <image-tag> <targetImageRegistry>
```

Uninstall and Cleanup

The process of uninstalling requires one to navigate to the installer directory and execute the following command

```
./appviewxctl uninstall --config .appviewxctl.yaml
```

The following messages are displayed after the uninstall command is executed successfully.

```
./appviewxctl uninstall --config .appviewxctl.yaml

[Init] Using log file at [/avx/appviewxctl-3196327299.log] to dump logs
[Init] Initialise persistent flag config
[Init] Using config file
[Uninstall] Uninstalling appviewx application
[Uninstall] Uninstalling Appviewx application helm chart
[Uninstall] Uninstalling application backup helm chart
[Uninstall] Uninstalling Infra application helm chart
[Uninstall] Uninstalling Third party application helm chart
[Uninstall] Uninstalling IstioOperator from the cluster
[Uninstall] Uninstalling PVCs from the avx namespace
[Uninstall] Uninstalling Pre-requisite helm chart
[Uninstall] Uninstalling Appviewx installed namespaces
[Uninstall] Successfully uninstalled appviewx application and all the related resources
```

Note: In the Managed K8s environments removal of PVCs do not occur at times as it may require patching PVCs first before deletion. This may cause certain error messages to display, indicating that PVC has changed. In case such an error occurs, re-run the above command to solve the issue and uninstall the application.

Sometimes the namespaces take a longer time to be removed. Hence, post installation, check if namespaces are in the terminating state (use the command: **kubectl get namespace**). If any namespace is in the terminating state, manually remove the namespaces by executing the commands below:

```
kubectl get namespace "istio-operator" -o json | tr -d "\n" | sed "s/\"finalizers\": \[[^]]\+\]/\"finalizers\": []/" | kubectl replace
 --raw /api/v1/namespaces/istio-operator/finalize -f - 2>/dev/null
```

```
kubectl get namespace "istio-system" -o json | tr -d "\n" | sed "s/\"finalizers\": \[[^]]\+\]/\"finalizers\": []/" | kubectl replace
 --raw /api/v1/namespaces/istio-system/finalize -f - 2>/dev/null
```

```
kubectl get namespace "avx" -o json | tr -d "\n" | sed "s/\"finalizers\": \[[^]]\+\]/\"finalizers\": []/" | kubectl replace --raw /api/v1/namespaces/avx/finalize -f -
 2>/dev/null
```

```
kubectl get svc "istio-ingressgateway-proxy" -n istio-system -o json | tr -d "\n" | sed "s/\"finalizers\": \[[^]]\+\]/\"finalizers\": []/" | kubectl $kubeconfig_string replace
 --raw /api/v1/namespaces/istio-system/services/istio-ingressgateway-proxy -f - 2>/dev/null
```

```
kubectl get svc "istio-ingressgateway" -n istio-system -o json | tr -d "\n" | sed "s/\"finalizers\": \[[^]]\+\]/\"finalizers\": []/" | kubectl $kubeconfig_string replace
 --raw /api/v1/namespaces/istio-system/services/istio-ingressgateway -f - 2>/dev/null
```

```
kubectl delete ns istio-operator --force 2>/dev/null
```

```
kubectl delete ns istio-system --force 2>/dev/null
```

```
kubectl delete ns avx --force 2>/dev/null
```

Troubleshooting

Error	Resolution
403: Access Forbidden - License not available. This error occurs when only the managed kubernetes cluster is restarted every morning.	In case of managed kubernetes, when the cluster is restarted it is recommend to restart all pods.

AppViewX Install and Upgrade for GKE

This guide provides the prerequisites and the procedure for installing, upgrading, and accessing the AppViewX application.

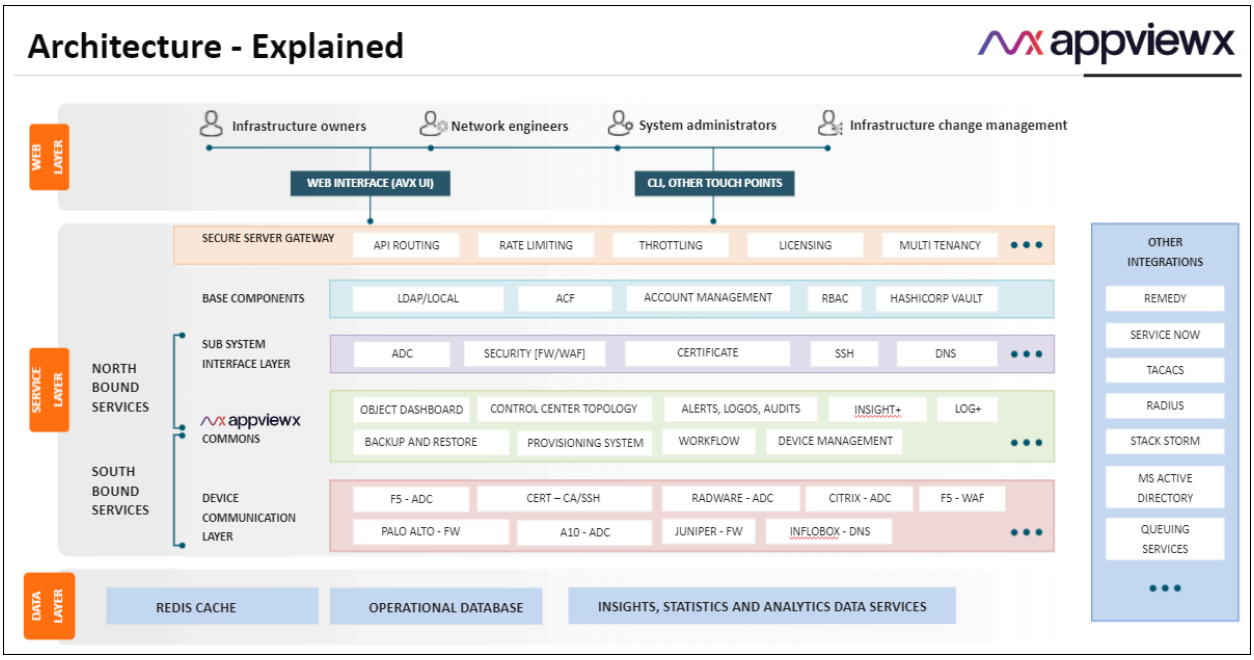
AppViewX Architecture

Architecture Explained

AppViewX is designed based on the microservice architecture and is deployed on Kubernetes—an open-source platform for deploying and managing containers.

The microservice architecture of AppViewX makes it easier to move to containerized workloads and the containers being orchestrated using Kuberneters.

Kubernetes provides container runtime, orchestration, self-healing mechanisms, service discovery and load balancing and it is used for the deployment, scaling, management, and composition of application containers across clusters.



Benefits of AppViewX Architecture

In order to optimally utilize the resources, AppViewX has adopted Kubernetes to achieve higher security by adopting a zero trust network model. The features of AppViewX coupled with Kubernetes are given below.

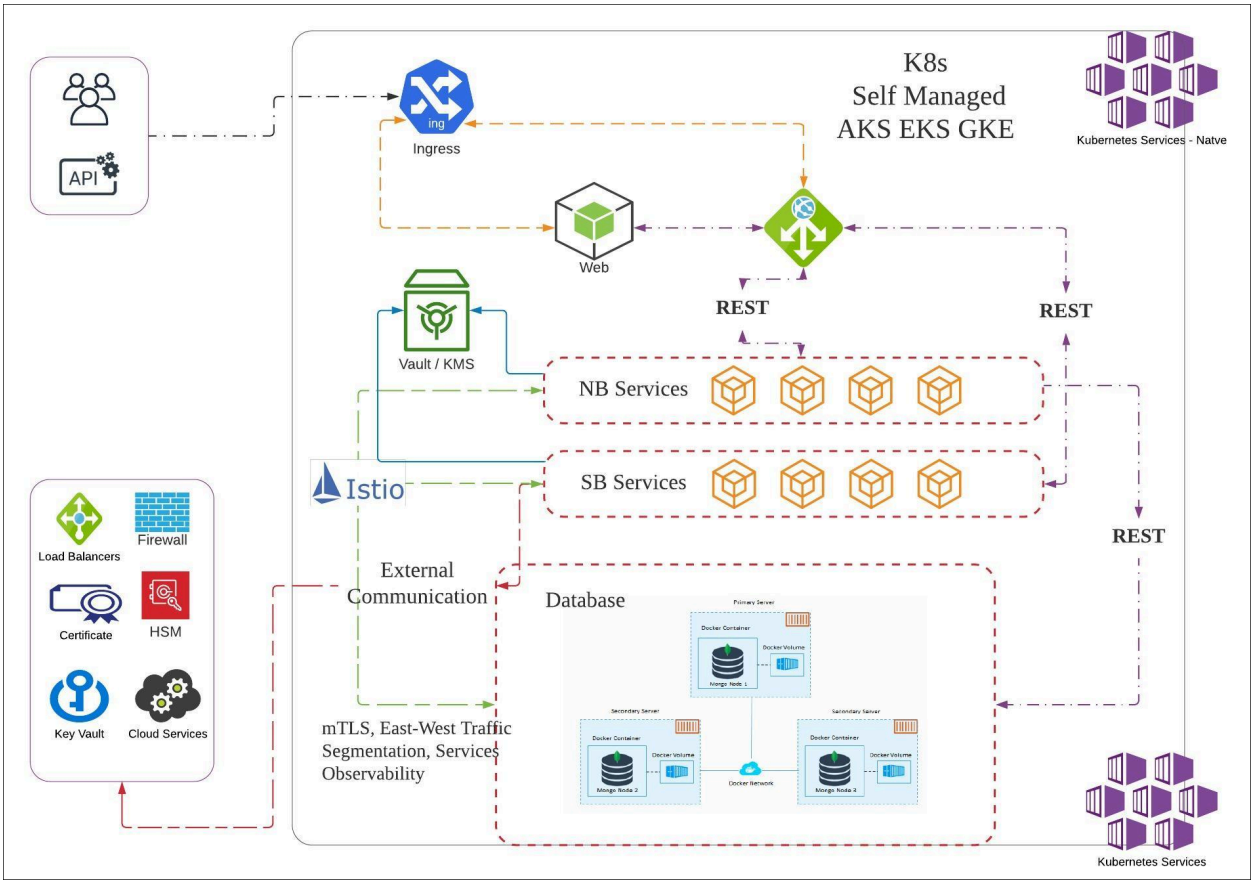
- **Auto scaling** - AppViewX services can have a custom throttling capability based on pre-configured memory configuration per API. This will enable AppViewX services to utilize (scale up) resources optimally as the demand surges and scale down when not in use. This will help to horizontally scale the vendor components on demand and optimize the resource usage.
- **Resiliency** - There is no guarantee that AppViewX services may run without any interruptions and they are bound to fail. Kubernetes keeps deployments healthy by restarting containers that have failed, by killing and replacing unresponsive containers based on health checks. This helps to mitigate the common pain point of the application's upkeep process.
- **Security** - AppViewX architecture is designed around the concept of zero trust network model to enforce tighter security within the Kubernetes cluster. This means no one is trusted by default and requires verification to gain access to the services.

Architecture Overview

AppViewX Kubernetes Architecture

AppViewX workloads are containerized workloads running as microservices and these containers are orchestrated by managed Kubernetes services. Users can prefer the managed k8s platform of their choice.

AppViewX supports deployment on all the three public clouds AWS, Azure and GCP (Google Cloud Platform) using their managed kubernetes engine / services EKS, AKS and GKE specifically.



Benefits of AppViewX Architecture

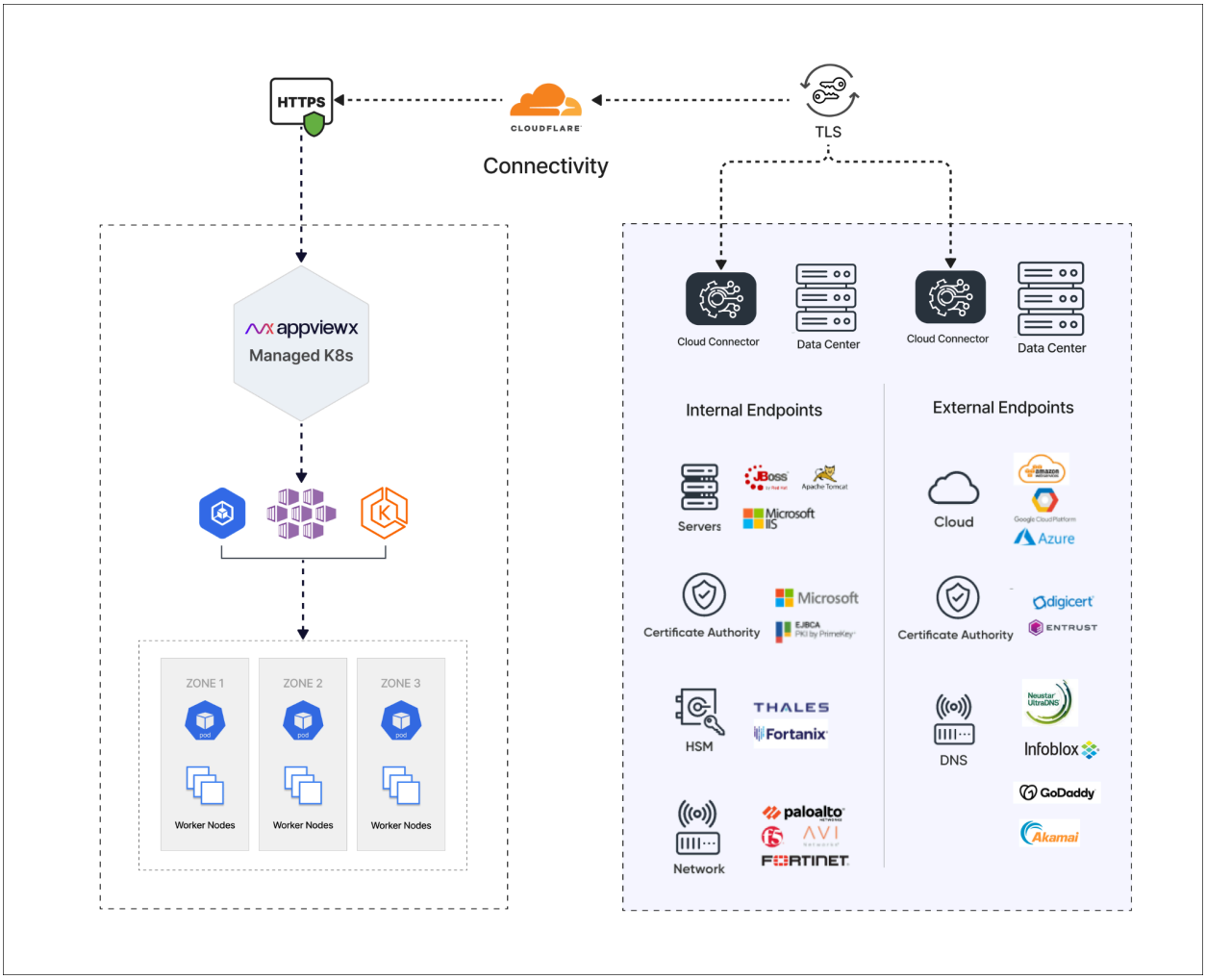
In order to optimally utilize the resources, AppViewX has adopted Kubernetes to achieve higher security by adopting a zero trust network model. The features of AppViewX coupled with Kubernetes are given below.

- **Auto scaling** - AppViewX services can have a custom throttling capability based on pre-configured memory configuration per API. This will enable AppViewX services to utilize (scale up) resources optimally as the demand surges and scale down when not in use. This will help to horizontally scale the vendor components on demand and optimize the resource usage.

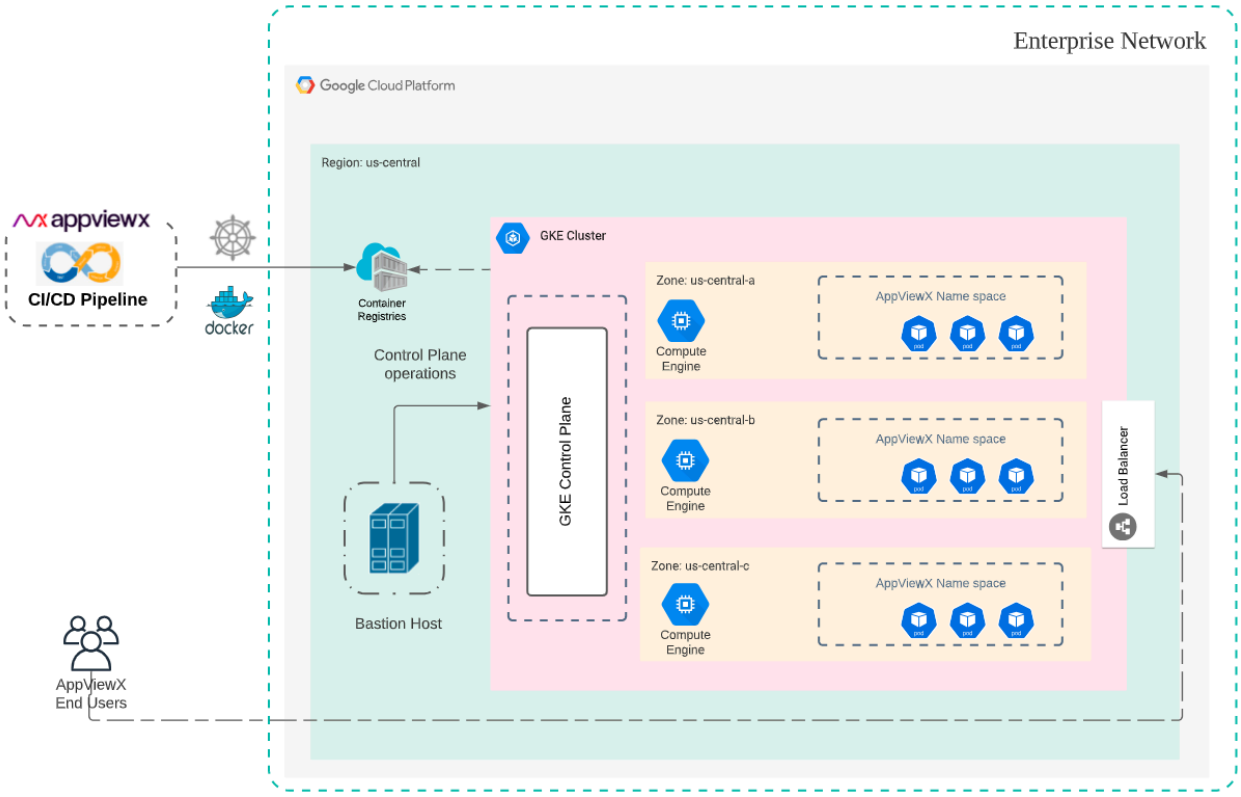
- **Resiliency** - There is no guarantee that AppViewX services may run without any interruptions and they are bound to fail. Kubernetes keeps deployments healthy by restarting containers that have failed, by killing and replacing unresponsive containers based on health checks. This helps to mitigate the common pain point of the application's upkeep process.
- **Security** - AppViewX architecture is designed around the concept of zero trust network model to enforce tighter security within the Kubernetes cluster. This means no one is trusted by default and requires verification to gain access to the services.

AppViewX Deployment Architecture

The figure below shows a standard AppViewX deployment architecture model via managed Kubernetes service for GKE.



GKE Deployment Model

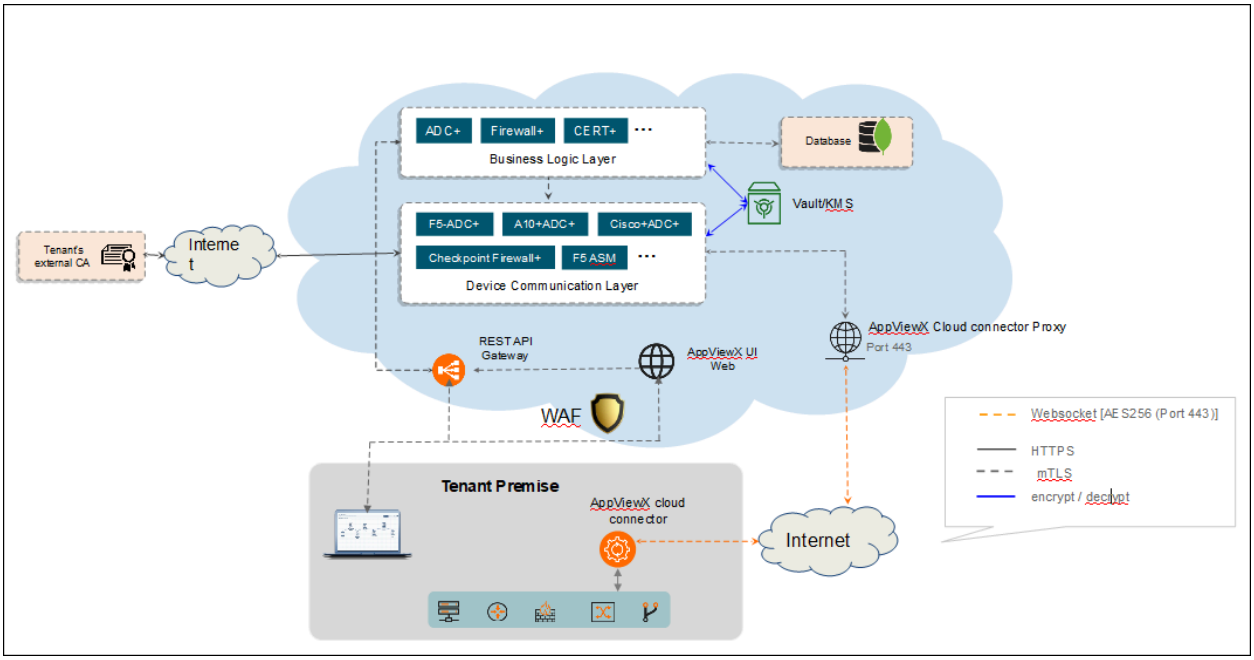


Cloud Connector

AppViewX Cloud Connector is a lightweight plug-in that establishes connectivity between AppViewX Cloud and the Enterprise Network. The cloud connector serves as a secure channel for communication between AppViewX and your enterprise network without requiring any complex network or infrastructure configuration

Key features of the AppViewX Cloud Connector:

- A self-serviceable, Linux-based lightweight setup
- Secure communication between the AppViewX and the AppViewX Cloud Connector using TLS and AES encryption
- Connectivity from the AppViewX to the enterprises' network endpoints
- No complex network setup (Inbound Firewall Whitelisting, VPN setup, and so on)



For more details on cloud connectors refer to AppViewX Cloud Connector User Guide.

Note: The below steps have to be performed in all the cloud connector host machines after the 2022.1.0FP2 to FP3 patch upgrade and before the FP3 cloud connector upgrade.

1. Navigate to the installation path in the cloud connector host machine.
2. Execute the following command:

```
./deps/tools/k3s kubectl get deploy avx-mid-server-starter -n cc -o yaml > starter.yaml && sed -i "s/-Xmx2560m/-Xmx4g/g" starter.yaml

&& ./deps/tools/k3s kubectl replace -f starter.yaml
```

Managed Kubernetes Architecture

Managed Kubernetes clusters are composed of the following main components — a control plane and worker nodes. Each cluster runs in its own, fully managed Virtual Private Cloud (VPC).

- The **control plane** is composed of three master nodes, each running in a different Availability Zone to ensure high availability. Incoming traffic directed to the Kubernetes API passes through the respective cloud service load balancer.
- The **worker nodes** run on virtual instances located in a VPC. Managed kubernetes service engine provides managed node groups with automated lifecycle management. This lets users automatically create, update, or shut down nodes with one operation.

Managed Kubernetes service scales the Kubernetes control plane across multiple Availability Zones of the public cloud to ensure high availability and it automatically scales control plane instances based on load, detects and replaces unhealthy control plane instances, and automatically patches the control plane.

Managed Kubernetes workload instances are deployed in multiple availability zones within the region. Each instance has replicas of the services and nodes which exist across all the virtual instances.

Each zone or instance has an active pod listening to other instances. In case of a failure in any instance, the active pod ensures seamless functioning of the application by activating the nodes from any other working cluster.

GCP Components

The following GCP components are utilized by AppViewX:

- Google Kubernetes engine (refer to version support metrics in the next section).
- Storage Bucket for storing MongoDB and Vault backups.
- Service account for accessing storage bucket and GCR registry.

Prerequisites

The following prerequisites must be met before the installation process.

- Managed Kubernetes Version Support Matrix
- Disks Used for AppViewX Installation
- AppViewX Docker Images
- AppViewX Helm Charts
- Bastion Host Setup
- GKE Cluster
- GCP Storage Bucket

Managed Kubernetes Version Support Matrix

Public Cloud	
Mode of Deployment	**Google**
Release, Vendor, & Product Support **AppViewX v2023.1.0 FP3**	
Managed K8s Deployment (GKE)	
K8s version 1.29	Yes

Disks Used for AppViewX Installation

Discs Used

Volume	Size	Quantity
logs volume	50Gi	1
avx-kafka	20Gi	3
zookeeper	20Gi	3
consul-server	10Gi	3
mongo-configdb	10Gi	3
mongo-shareddb	256Gi	3
redis	5Gi	3

If a third party is installed, the values are as follows:

Discs Used (Third Party)

Volume	Size	Quantity
Elasticsearch-ELK	10Gi	1
Elasticsearch-Insight	10Gi	1

AppViewX Docker Images

AppViewX Docker images are hosted in a private registry https://images.appviewx.com. These images can be pulled using an authentication token (contact AppViewX Support, help@appviewx.com for the authentication token) and can be hosted in the private or public repository at the customer end.

The list of docker images are

- <registry link>/appviewx/pilot:1.19.0
- <registry link>/appviewx/proxyv2:1.19.0
- <registry link>/appviewx/istio-operator:1.19.0
- <registry link>/appviewx/vault:1.13.7
- <registry link>/appviewx/redis:7.2.0
- <registry link>/appviewx/mongo-init:<tag>
- <registry link>/appviewx/avx-cloud-gateway:<tag>
- <registry link>/appviewx/avx-cloud-web:<tag>
- <registry link>/appviewx/avx-cloud-mongoseed:<tag>
- <registry link>/appviewx/avx-cloud-managedservice-mks:<tag>
- <registry link>/appviewx/avx-platform-report-generator:<tag>
- <registry link>/appviewx/avx-python-sandbox:<tag>
- <registry link>appviewx/avx-mid-server-base:<tag>
- <registry link>/appviewx/consul:1.16.1
- <registry link>/appviewx/kafka:0.32.0-kafka-3.3.1
- <registry link>/appviewx/operator:0.32.0
- <registry link>/appviewx/alpine:3.13.6
- <registry link>/appviewx/kube-metrics-adapter:v0.2.1
- <registry link>/appviewx/kube-state-metrics:v1.9.8
- <registry link>/appviewx/backup-utility-image:v3.0
- <registry link>/appviewx/prometheus:v2.45.0
- <registry link>/appviewx/metrics-server:v0.6.4
- <registry link>/appviewx/elasticsearch:8.9.1
- <registry link>/appviewx/elasticsearch-insight:8.9.1
- <registry link>/appviewx/filebeat:8.9.1
- <registry link>/appviewx/grafana:10.1.1
- <registry link>/appviewx/kibana:8.9.1
- <registry link>/appviewx/logstash:8.9.1
- <registry link>/appviewx/logstash-syslog:8.9.1
- <registry link>/appviewx/alertmanager:v0.26.0

- <registry link>/appviewx/node-exporter:v1.6.1
- <registry link>/appviewx/redis_exporter:v1.53.0

The steps to download the images from AppViewX repository are as follows:

1. Get the source image repository credentials from AppViewX Support team.
2. Configure the docker using the command

```
docker login -u ${USERNAME} -p ${PASSWORD} ${DOCKER_REPOSITORY}
```

3. Configure the respective cloud provider CLI (Google cloud) and ensure you have access to push docker images to GCR.
4. To push the docker images, use the helper script provided by AppViewX. Follow the steps below.

   a. Download the artifact Managed-Kubernetes_helper_scripts.tar.gz to the bastion host and extract using the command:

   ```
   tar -xf Managed-Kubernetes_helper_scripts.tar.gz
   ```

   b. Navigate to the extracted directory **mk8s_helper_scripts**.

   ```
   cd mk8s_helper_scripts
   ```

   c. Execute the script **avx_image_pull_push.sh** using the command

   ```
   ./avx_image_pull_push.sh <Image tag> <customer registry url>
   ```

   Note: Replace <Image tag> and <customer registry url> with the actual values.

AppViewX Helm Charts

The helm charts used by AppViewX for installation are released as a part of the installer. The installer consists of helm charts and an AppViewX utility which helps orchestrate the deployment, patch, upgrade and maintenance of AppViewX across managed kubernetes deployment.

Bastion Host Setup

The following packages must be installed on the bastion host or the host/tool (Azure DevOps) from where the installation is triggered

GCP CLI

To set up the GCP CLI refer to Install the gcloud CLI on the Google documentation website.

Kubectl

To set up Kubectl refer to Install and Set Up kubectl on Linux on the Kubernetes documentation website.

Execute the following commands

- `sudo curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"`
- `sudo chmod +x kubectl`
- `sudo mv ./kubectl /usr/bin/#`

Verify installation by executing the command

```
kubectl version
```

Helm

Helm is required only if the deployment is triggered from any other machine instead of the DevOps pipeline. To set up Helm refer to Installing Helm on the Helm documentation website.

Execute the following command:

- `curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3`
- `chmod 700 get_helm.sh`
- `./get_helm.sh#`

Verify installation by executing the command

```
helm version
```

GKE Cluster

To create an GKE cluster refer to Google cloud documentation website - Creating a regional cluster. Although Google cloud manuals are always up-to-date, the recommended choice to make before creating the cluster is as follows:

- Kubernetes version: 1.29
- User nodepool:
  - **appnodepool**: Three nodes of type **n2-standard-8** with Auto Scaling disabled
  - **mongonodepool**: Three nodes of type **n2-standard-8** with Auto Scaling disabled. Add label **mongo=true** and taint **designatedMongo=true:NoSchedule** to the nodepool (to be performed while creating the cluster).
- Select multi zones for the Nodepools

GCP Storage Bucket

A storage bucket is required to store

- iControlJar: Container name is **icontroljar** and the jar has to be placed here.
- MongoDB backup: Container name should be **mongo-backup**.
- Vault backup: Container name should be **vault-backup**.
- Axisjar: Jar needs to be added here - **axisjar**.

A summary of steps for creating the storage bucket is as follows:

Note: After the script is executed, capture the output **Annotation** which is required in the global utility config.

1. Create a storage account with a valid name to indicate the storage account for a specific GKE cluster.
2. Configure Storage buckets and Image Registry access for GKE nodes.
3. The first workload identity should be enabled cluster wide. This operation may be performed from the portal after the cluster creation or at the time of cluster creation. Refer google document Using Workload Identity
4. The above steps can also be performed using a helper script provided by AppViewX. To use this script follow the steps below.

   a. Download the artifact Managed-Kubernetes_helper_scripts.tar.gz to the bastion host and extract using the command:

   ```
   tar -xf Managed-Kubernetes_helper_scripts.tar.gz
   ```

   b. Navigate to the extracted directory **mk8s_helper_scripts**.

   ```
   cd mk8s_helper_scripts
   ```

c. Edit the file **gcp_sc_config.sh** and replace <PROJECT_ID>, <CLUSTER_NAME>, <NODE_POOL_1,NODE_POOL_2>, <REGION_NAME> with the actual values.

d. Execute the **gcp_sc_config.sh** file.

```
bash gcp_sc_config.sh
```

5. Store output of the script in step 4d and pass the annotation in the global utility config **serviceAccountAnnotation** (refer the second table in Installation Step 7)

Install AppViewX in Managed Kubernetes

Migration Strategy

Attention: If you are performing a fresh install, then refer the next sub-topic **Installation Steps**.

To migrate from AppViewX on-prem versions (2022.1.0, 2021.1.0, and 2020.3.0) to Managed Kubernetes, it is important to take a backup of the mongodb and vault in the respective on-prem versions. Before you take the backup, execute the script below.

```
db.profile.update({'_id' : 'installationType'}, {$set : {"value" : "Managed_K8s"}})
```

Note: Refer to the specific version of the release documents from the release portal and perform the backups or contact the AppViewX support team.

After performing the backup, follow the installation steps detailed in the section below. At step 11 of the installation process, ensure to restore the data at this stage.

Installation Steps

This section describes the steps to for installing the AppViewX Stack on AKS.

1. Download the installer from the release portal.
2. Create a directory Managedk8s-installer in the bastion host and extract the installer file **tar -xf installer.tar.gz** in the same directory.
3. Verify that the extracted installer must have the following files
   - appviewxctl (binary)
   - helm_charts (directory of helm charts)

4. Generate the configuration files based on the cloud provider. If the cloud provider is **Google**, execute the command below.

```
./appviewxctl config generate --provider gcp
```

5. Verify that the execution of the above command creates the configuration files named **.appviewxctl.yaml** in the same location.
6. The file .appviewxctl will be populated with the fields necessary for installation, in particular cloud provider that was provided in the previous command (**-- provider**).
7. Edit the .appviewxctl.yaml file and populate the values as described below:

appviewxctl.yaml file - Parameters and Description

Parameters	Description of Values
chartPath	The path to the helm_charts which is to be installed. It points to the helm_charts directory extracted in step 3.
configFile	The path to the kube config file to be used by helm and kubectl. If the bastion host is already configured and kube config is under $HOME/.kube directory, then keep this field empty.
install.enableAppBackupCron	Boolean value to enable/disable the backup cronjobs. (True/False). This value is needed for self-managed mongo only. For atlas backup this has to be scheduled in the atlas dashboard.
install.enablePrivateImagePullSecret	Boolean value to enable image pull secret. Set values as **false** if the cluster already has access to the container registry. Otherwise set it to **true** and fill all the details of the access keys described in below sections.

Parameters	Description of Values
install.enableThirdPartyInstall	Boolean value (True/False) to determine whether third party monitoring components such as ELK, Monitoring, and Insight needs to be installed.
install.thirdPartyApp.elk	Boolean value to add Elk component. Set to True if it needs to be installed.
install.thirdPartyApp.monitoring	Boolean value to add Monitoring component. Set to True if it needs to be installed.
install.thirdPartyApp.insight	Boolean value to add Insight component. Set to True if it needs to be installed.
install.imageRegistry	The URL of the container registry where the images are to be pulled from by the pods. *Example*: gcr.io/pe-qa-358108
install.imageTag	The tag of the image that will be used for installation. *Example*: 2023.1.0_FP_750-alpine
install.isSaasEnabled	Boolean value for SaaS enablement. This value should be set to **true** for Managed K8s.

install.kafkaCloudConnector	It is a combination of three values.
	• enable
	• password
	• user
	Set **enable** to ***true*** and keep the user, password fields empty for Managed K8s.
	
	Example
	`kafkaCloudConnector:`
	`  enable: true`
	`  password: ""`
	`  user: ""`

Parameters	Description of Values
install.mongo	It is a combination of fields specific to the type of mongodb used.
dbIsolation	Boolean value to indicate whether the database isolation is to be enabled. In order for database isolation to work, the following prerequisite must be taken care of while creating the cluster node group. • Add label **mongo=true** and taint **designatedMongo=true:NoSchedule** to the nodepool to be used for mongodb.

mongoAtlas	The fields specific to mongodb atlas are as folows:
	• **enable**: Boolean value to decide if mongodb atlas to be used. If set to *false*, a self managed mongodb cluster will be created. If set to *true* mongodb atlas will be used and details of which are to be provided in below mentioned fields.
	• **host**: URL of the mongodb atlas cluster.
	• **password**: password of the mongodb atlas cluster.
	• **user**: username in the mongodb atlas cluster.
	
	Example:
	
	`mongo:`
	`  dbIsolation: false`
	`  mongoAtlas:`
	`  enable: true`
	`  host: "managed-k8s.test.mongodb.net"`
	`  password: "samplepassword"`
	`  user: "user1"`

install.useDockerPrivateRegistry	Set this to **true** if the dockerhub private repository is to be used for pulling the necessary images needed. Otherwise set the value **false** and the

Parameters	Description of Values

	container registry ACR, ECR, and GCR will be used based on the cloud provider.
	
	If this value is set to *true*, populate the below values, otherwise keep it empty.
	• **dockerhub.pass**: password to be used for authenticating in the dockerhub private repository.
	• **dockerhub.username**: username configured in the dockerhub private repository.
	
	Example:
	`useDockerPrivateRegistry: true`
	`dockerhub:`
	`pass: "testpassword"`
	`username: "appviewx"`

install.size	The size of the installation. Based on the use cases and number of certs to be managed there different sizes (contact AppViewX for sizing recommendations). The sizes supported are (case sensitive values) • xsmall • small • medium • large • xlarge • custom *Example:* `size: small` **Note:** The size provided must be taken into cluster creation and nodegroup sizes must be defined accordingly.

Parameters	Description of Values

install.plugins	The list of plugins that will be installed. Each plugin will have three fields
	• enable
	• imageTag
	• name
	
	Set enable to **true** if the plugin is to be installed. If the same image tag is to be used as defined in the global ImageTag keep it **latest** otherwise override with some other tag of your choice.
	
	Example:
	
	`- enable: true`
	`imageTag: latest`
	`name: avx-config-server`
	
	To enable Cloud DC support in Managed Kubernetes, set plugins as follows:
	
	`- enable: true`
	`imageTag: latest`
	`name: avx-mid-server-platform`
	
	Note: Ensure that **install.isSaasEnabled** and **install.kafkaCloudConnector** are set to **true**.

internalLoadBalancer	If set to **true**, all the Loadbalancers will be private and can only be accessed within the VPC else it will be public.
install.enableSftpStorage	Change to true to use SFTP server for mongodb, vault, and icontrol.jar storage. Boolean (Default: false)
install.sftpServerDetails. dbBackupPath	Provide the location of mongodb backup storage directory. String (Default: “”)

Parameters	Description of Values
install.sftpServerDetails. vaultBackupPath	Provide the location of vault backup storage directory. String (Default: “”)
install.sftpServerDetails. sftpServerUserName	Provide the username of SFTP server. String (Default: “”)
install.sftpServerDetails. sftpServerIp	Provide the sftp server IP. String (Default: “”)
cloudConnectorEnabled	A boolean value (true/false) to denote the cloud connector usage for southbound communications. If a cloud connector is used set the value to **true**.

The next fields are to be filled with values that must be collected during the cluster creation and setup process and filled as mentioned below.

Parameters	Description of Values

install.privateImagePullSecret	In this section populate the details of the access keys needed to authenticate and pull the image from the registry. They are not needed if the Dockerhub is used as described above.
	• **registry**: The registry whose token must be provided below and used to pull images.
	• **token**: The login token for the registry used. Token can be generated from CLI if authenticated in the CLI from the respective google cloud account. A sample command to generate token of gcr.io registry
	`gcloud auth print-access-token |docker login -u oauth2accesstoken --password-stdin https://gcr.io`
	Example:
	`registry: "gcr.io/pe-qa-358108"`
	`token: "sample token"`

install.storageAccess	The storage bucket details to be used for setting up backup capability. • **bucketObject**: The name of the bucket object. • **serviceAccountAnnotation**: Annotation of service account that provides access to the storage bucket

Parameters	Description of Values
	Example: bucketObject: "appviewx-samplebucket" serviceAccountAnnotation: "avx-storage-bucket-access-gsa@sampleproject.iam.gserviceaccount.com"

The following fields must be added to integrate the kubernetes cluster to the external vault.

appviewxctl.yaml file - Parameters and Description (for external vault)

Parameters	Description
install.externalVault.enable	A boolean value (true/false) to denote if the external vault is to be used in the setup. True is to enable the external vault.
install.externalVault.externalVaultAddr	Contains the vault URL and listening port *Example*: https://pm-lxs-node01.lab.appviewx.net:8200
install.externalVault.externalVaultAuthRole	Name of the role created against the access kubernetes auth path
install.externalVault.externalVaultCACertSecret	Name of the secret where **vault-ca.crt** file is mounted.
install.externalVault.externalVaultDBRole	Static role created to access the database cred.
install.externalVault.externalVaultEnginePath	Enter the value “/database”
install.externalVault.externalVaultKubeAuthPath	The Kuberenetes access path created with cluster information for service account authentication.
install.externalVault.externalVaultSAName	The Service account used to create externalVaultAuthRole.
install.externalVault.mongoPasswordVaultEngine	Enter the value DATABASE

8. Once the values are filled in *.appviewxctl* as described in the step above, proceed with the installation. Before doing so, check if the the preconditions are met by executing the command

```
./appviewxctl preflight --config .appviewxctl.yaml
```

This will prompt if the necessary prerequisites are met.

9. The metrics server in the GCP clusters comes pre-installed with the cluster, hence they must be disabled from the **avx_pre_req** chart.
   a. Navigate to helm_charts/avx_pre_req.
   b. Edit the **values.yaml** file by setting the following parameters.

```
avx-metrics-server:

enable: false
```

   The metrics server installation is disabled.

10. To proceed with installation, execute the command

```
./appviewxctl install --config .appviewxctl.yaml
```

> **Note:** The installation will take several minutes to complete. Upon completion you see the following message:
>
> [Install] Successfully installed Appviewx infra stack
>
> This would imply the completion of infra component setup.

11. This step involves restoring the existing data from the previous AppViewX version's cluster in case there is a need to migrate from the older versions to the Managed K8s version. **Ignore this step if it's a fresh setup with no migration necessary.**

    To restore mongodb and vault fetch the backup files and place them in the bastion in a directory such as /home/user/backup execute the mongo_restore and vault_restore scripts as follows:

```
./mongo_restore.sh <mongo backup tar filepath>
```

```
./vault_restore.sh -p <vault backup filepath> --removedek
```

> **Attention:** If the data is being restored from an older version (2020.3.0 - 2022.1) then use the command
>
> ```
> ./vault_restore.sh -p <vault backup filepath> --removedek
> ```

> **Note:**

- The backup files must have extension as **.tar.gz**
- The above commands work for a self-managed mongodb setup. Setting up the mongodb atlas requires the installation of mongodb tools in the bastion host as described below.

For an rpm based OS:

```
echo -e "[mongodb-org-4.2] \nname=MongoDB
 Repository\nbaseurl=https://repo.mongodb.org/yum/redhat/\$releasever/mongodb-org/4.2/x86_64/\ngpgcheck=1\nenabled=1\ngpgkey=https://
www.mongodb.org/static/pgp/server-4.2.asc" > /etc/yum.repos.d/mongodb-org-4.2.repo
yum install mongodb-org-shell-4.2.0
yum install mongodb-org-tools-4.2.0
```

For a debian based OS:

```
wget -qO - https://www.mongodb.org/static/pgp/server-6.0.asc | sudo apt-key add -
sudo apt-get install gnupg
wget -qO - https://www.mongodb.org/static/pgp/server-6.0.asc | sudo apt-key add -
echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/6.0 multiverse" | sudo
 tee /etc/apt/sources.list.d/mongodb-org-6.0.list
sudo apt-get update
sudo apt-get install -y mongodb-mongosh
sudo apt-get install -y mongodb-org-tools
```

Verify if the mongodb restore commands have executed successfully using the command

```
mongorestore -- version
```

12. To proceed with the AppViewX application installation, execute the command:

```
./appviewxctl installapp --config .appviewxctl.yaml
```

Once installation is complete the following messages are displayed:

```
[Install] Appviewx infrastructure chart [avx-app] installed successfully
[Install] Successfully installed Appviewx application stack
[Install] Fetching login URL for app
[Install] Waiting for Public IP allotment for istio service
[Install] AppViewX Web URL: https://34.100.197.159/appviewx/
[Install] AppViewX Gateway URL: https://34.100.197.159/avxmgr/
[Install] Grafana URL: https://34.100.197.159/grafana/
[Install] Kibana URL: https://34.100.197.159/kibana/login
[Install] Run below commands to get mongo user credentials
```

```
export MONGO_USER=$(kubectl get secret -n avx mongo-key -o=jsonpath='{.data.mongo-init-user}' | base64 -d)
export MONGO_PASS=$(kubectl get secret -n avx mongo-key -o=jsonpath='{.data.mongo-init-pass}' | base64 -d)
[Install] Run below commands to get Elasticsearch and Kibana credentials
export ES_PASS=$(kubectl get secret -n avx elasticsearch-pw-elasticsearch -o=jsonpath='{.data.password}' | base64 -d)
export KIBANA_PASS=$(kubectl get secret -n avx elasticsearch-pw-kibana -o=jsonpath='{.data.password}' | base64 -d
[Install] Application Installation completed successfully
```

Note: Follow the URLs and commands given in the output message to get the credentials and access the application.

13. If installation of the third party monitoring components was not enabled during the entire process, they can be installed later by the following steps:

    a. While installing the third party components (helm_charts/avx_third_party/values.yaml), the only that values are set to 'true' by default are - *prometheus, nodeexporter, kube-state metrics*. The other components are set as 'false' by default and must be to set to true if they are to be enabled, they are - *elk-elasticsearch, elk-filebeat, elk-kibana, elk-logstash, grafana, elasticsearch-insight, logstash-syslog.*

    b. Edit the **.appviewxctl.yaml** file and set **install.enableThirdPartyInstall** to ‘true’

    c. Configure the following **thirdPartyApp** parameters as true as per the requirements:
    - **install.thirdPartyApp.elk**
    - **install.thirdPartyApp.monitoring**
    - **install.thirdPartyApp.insight**

    d. Now, edit the file **values.yaml** present at location helm_charts/appviewx_monitoring/prometheus/chart/values.yaml and append the below values at the end of the file (only if that are not present).

```
limits:
  cpu_limit: 80
  memory_limit: 80
  disk_limit: 80
  timelimit_cpu_memory: 5
  timelimit_disk: 1
  timelimit_pod: 1
  timelimit_node: 1
```

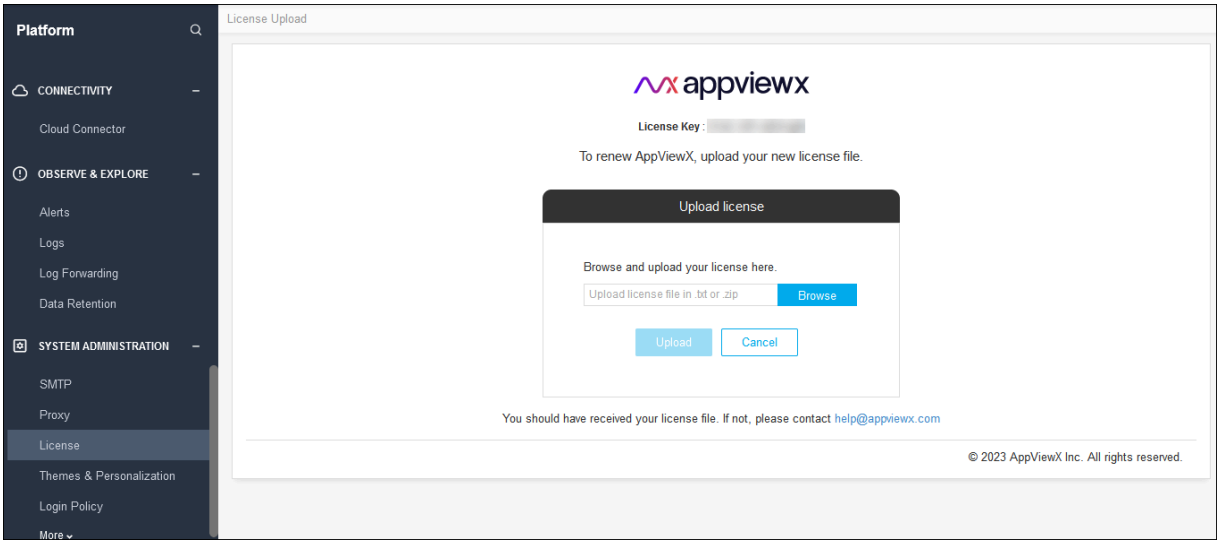
e. Run the command below

```
./appviewxctl installtpt --config .appviewxctl.yaml
```

Customers migrating from AppViewX version 2020.3.0 to Managed Kubernetes FP3, it is mandatory to upgrade the license.

To upgrade the license,

1. Login to the AppViewX with valid credentials.
2. Navigate to Platform >> System Administration >> License page.
3. Click **Upgrade License**.



4. Click **Browse** to find the latest license key file.
5. Click **Upload**.

> **Note:** For the licenses contact AppViewX Support at help@appviewx.com or customerlicences@appviewx.com.

Upgrade AppViewX in Managed Kubernetes

> **Attention:**

- If you are using the self managed private docker registry instead of AppViewX's docker registry, then before proceeding with the upgrade, ensure you have copied the latest images to your registry. The list of images can be found in the Prerequisite section - AppViewX Docker Images.
- If you are currently using AppViewX v2022.1.0 FP3 (i.e. after applying the infra hotfix for FP3) and already in Kube 1.26, then you must follow these prerequisite steps before upgrading to Hudson or the next infra upgrade:
  1. Execute the command

     ```
     kubectl get secrets -n avx sh.helm.release.v1.vault.v2 -o json | jq .data.release -r | base64 --decode | base64 --decode | gunzip
     ```

     This creates the file **manifest.json**.
  2. Open the **manifest.json** using VIM or any other editor.
  3. Search for parameter **PodDisruptionBudget**, find its API version and change it from **v1beta1** to **v1**. Save the changes.
  4. Execute the command.

     ```
     DATA=`cat manifest.json | gzip -c | base64 | base64 | tr -d '\n\r'`
     ```

     ```
     kubectl patch secret -n avx sh.helm.release.v1.vault.v2 --type='json' -p="[{\"op\":\"replace\",\"path\":\"/data/release\",\"value\":\"$DATA\"}]"
     ```

To upgrade AppViewX with a new image version, follow the steps below:

1. Ensure to take a backup of the MongoDB and Vault for rollback in case something goes wrong during upgrade. Before you take the backup, execute the script below.

   ```
   db.profile.update({'_id' : 'installationType'}, {$set : {"value" : "Managed_K8s"}})
   ```

2. To take the backups, execute the commands below.
   For self-managed mongodb:

   ```
   kubectl create job --from=cronjob/mongo-backup -n avx mongo-backup-<unique-identifier>
   ```

   ```
   kubectl create job --from=cronjob/vault-backup -n avx vault-backup-<unique-identifier>
   ```

   Replace <unique-identifier> in above commands with some random string and run. Monitor the pods until completion and verify the backups are placed in the storage bucket.

   Note: Atlas backup must be taken in the atlas dashboard. Refer to the atlas snapshots section in the page Backup and Restore.

3. Navigate to the installer directory.

4. Edit the **appviewxctl.yaml** file's upgrade section for the parameters mentioned below.

Parameters	Description of Values
upgrade.imageRegistry	The URL of the container registry where the images are to be pulled from by the pods. *Example*: gcr.io/pe-qa-358108
upgrade.imageTag	The tag of the image that will be used for installation. *Example*: 2023.1.0_FP_750-alpine
upgrade.isSaasEnabled	Boolean value for SaaS enablement. This value should be set to **true** for Managed K8s.
upgrade.plugins	The list of plugins that will be installed. Each plugin will have three fields • enable • imageTag • name Set enable to **true** if the plugin is to be upgraded. If the same image tag is to be used as defined in the global ImageTag keep it **latest** otherwise override with some other tag of your choice. **Note:** The list of plugins to be enabled should match the ones in the install section. *Example:* - enable: true imageTag: latest name: avx-config-server

5. Add the following component parameters in the **appviewxctl.yaml** file.

Parameters	Description of Values
install.thirdPartyApp.elk	Boolean value to add Elk component. Set to True for upgrade.
install.thirdPartyApp.monitoring	Boolean value to add Monitoring component. Set to True for upgrade.
install.thirdPartyApp.insight	Boolean value to add Insight component. Set to True for upgrade.

6. Update the following install parameters in the **appviewxctl.yaml** file required to integrate the kubernetes cluster to the external vault.

appviewxctl.yaml file - Parameters and Description

Parameters	Description of Values
cloudConnectorEnabled	A boolean value (true/false) to denote the cloud connector usage for southbound communications. If a cloud connector is used set the value to **true**.
install.externalVault.enable	A boolean value (true/false) to denote if the external vault is to be used in the setup. True is to enable the external vault.
install.externalVault.externalVaultAddr	Contains the vault URL and listening port *Example*: https://pm-lxs-node01.lab.appviewx.net:8200
install.externalVault.externalVaultAuthRole	Name of the role created against the access kubernetes auth path
install.externalVault.externalVaultCACertSecret	Name of the secret where **vault-ca.crt** file is mounted.
install.externalVault.externalVaultDBRole	Static role created to access the database cred.
install.externalVault.externalVaultEnginePath	Enter the value “/database”

Parameters	Description of Values
install.externalVault.externalVaultKubeAuthPath	The Kuberenetes access path created with cluster information for service account authentication.
install.externalVault.externalVaultSAName	The Service account used to create externalVaultAuthRole.
install.externalVault.mongoPasswordVaultEngine	Enter the value DATABASE

Note: Path parameters should have a leading forward slash '/'.

7. Before performing the Infra Upgrade, update the following parameters.
   a. Add the following additional plugins in the install and upgrade section of the appviewxctl.yaml file before proceeding with the upgrade:
      - avx-subsystem-codesigning
      - avx-python-sandbox-sync
      - avx-python-sandbox
      - avx-platform-aep-gateway

      Sample with default values:

```
- enable: true
  imageTag: latest
  name: avx-platform-aep-gateway
- enable: false
  imageTag: latest
  name: avx-subsystem-codesigning
- enable: true
  imageTag: latest
  name: avx-python-sandbox-sync
- enable: true
  imageTag: latest
  name: avx-python-sandbox
```

b. **appviewxctl.yaml file - Parameters and Description**

Parameters	Description of Values
upgrade.upgradeInfra	Boolean value to upgrade infra component. Set to True for upgrade.
upgrade.upgradeThirdParty	Boolean value to upgrade the monitoring (ELK, insight, and monitoring) components. Set to True for upgrade.

8. Download the upgrade tar file (**upgrade.tar.gz**) from the release portal and extract it to a suitable location. (The extracted files contain the binary and helm charts tar.)
9. Navigate to the folder where the upgrade tar is extracted.
10. Copy the appviewxctl binary from the current folder (extracted folder location) to the installer location.

```
cp appviewxctl <absolute path of the installer directory>
```

11. To upgrade AppViewX infra, execute the command

> **Note:** If you plan on enabling additional 3pt monitoring components as part of the infra upgrade do the following:
>
> a. Navigate to <installer>/helm_charts/avx_thrid_party/.
>
> b. Edit the **values.yaml** file.
>
> c. Set "enable" to true for the components you wish to enable as part of the upgrade.

```
./appviewxctl infraUpgrade --config .appviewxctl.yaml
```

This will prompt the following message

```
Please provide the path of updated helm charts tar. :
```

Enter the absolute path (extracted file path) of the new helm charts artifact.

12. After the infra upgrade is complete, execute the command

```
./appviewxctl upgrade --config .appviewxctl.yaml
```

Rollback Steps

a. Restore the DB using the restore scripts (step 11 in the Installation Steps section) for self-managed DB or in atlas using snapshot restore in the dashboard.

b. Update the **appviewxctl.yaml** upgrade section's values to the previous image tag and re-run the upgrade command.

Cloud Connector (CC) Upgrade

To pave the way for smooth CC upgrade, run the following command in all the cloud connector machines, ***after 2022.1.0FP2 to FP3 patch upgrade*** and ***before FP3 CC upgrade***.

- Navigate to the installation path of Cloud Connector machine.
- Execute the command

```
./deps/tools/k3s kubectl get deploy avx-mid-server-starter -n cc -o yaml > starter.yaml && sed -i "s/-Xmx2560m/-Xmx4g/g" starter.yaml && ./deps/tools/k3s kubectl replace -f starter.yaml
```

Downloading Images from AppViewX Repository

Prerequisites

1. Get the source image repository credentials from AppViewX.
2. Configure the docker using the command

```
docker login -u ${USERNAME} -p ${PASSWORD} ${DOCKER_REPOSITORY}
```

3. Configure the respective cloud provider CLI (Google cloud) and ensure you have access to push docker images to GCR.

The script for image push and pull is as follows:

```
appVersion=$1 # App image version. E.g: 2022.1.0_FP_750-alpine
targetImageRegistry=$2 # Image resgistry name

# Validate required inputs
if [ -z "$appVersion" ] || [ -z "$targetImageRegistry" ];then
{
    echo "Please provide script parametes as ./script.sh <appVersion> <targetImageRegistry>"
    exit
}
```

```
fi


# Set the registry login
if echo $targetImageRegistry | grep -iq "amazonaws";then
{
    registryProvider="ecr"
    region=$(echo $targetImageRegistry | cut -d "." -f4)
    aws ecr get-login-password --region $region | docker login --username AWS --password-stdin $targetImageRegistry
}
elif echo $targetImageRegistry | grep -iq "azurecr";then
{
    registryProvider="acr"
    az acr login -n $targetImageRegistry
}
elif echo $targetImageRegistry | grep -iq "gcr";then
{
    registryProvider="gcr"
    gcloud auth print-access-token | docker login -u oauth2accesstoken \
    --password-stdin $(echo $targetImageRegistry | cut -d '/' -f2)
}
else
{
    echo "Unknown regrsity provider"
    exit 2
}
fi


# Image tag mappings
imageTags='[
   {
      "imageName": "avx-cloud-managedservice",
      "tagVersion": "appVersion",
      "upload": true
   },
   {
      "imageName": "avx-cloud-web",
      "tagVersion": "appVersion",
```

```
    "upload": true
},
{
    "imageName": "avx-cloud-gateway",
    "tagVersion": "appVersion",
    "upload": true
},
{
    "imageName": "avx-platform-report-generator",
    "tagVersion": "appVersion",
    "upload": true
},
{
    "imageName": "mongo-init",
    "tagVersion": "appVersion",
    "upload": true
},
{
    "imageName": "avx-cloud-mongoseed",
    "tagVersion": "appVersion",
    "upload": true
},
{
    "imageName": "alpine",
    "tagVersion": "3.17.2",
    "upload": true
},
{
    "imageName": "pilot",
    "tagVersion": "1.16.2",
    "upload": true
},
{
    "imageName": "proxyv2",
    "tagVersion": "1.16.2",
    "upload": true
},
```



```
{
    "imageName": "istio-operator",
    "tagVersion": "1.16.2",
    "upload": true
},
{
    "imageName": "consul",
    "tagVersion": "1.10.3",
    "upload": true
},
{
    "imageName": "vault",
    "tagVersion": "1.8.4",
    "upload": true
},
{
    "imageName": "redis",
    "tagVersion": "6.2.3",
    "upload": true
},
{
    "imageName": "kafka",
    "tagVersion": "1.1.0-kafka-2.6.0",
    "upload": true
},
{
    "imageName": "kafka",
    "tagVersion": "1.1.0-kafka-2.7.0",
    "upload": true
},
{
    "imageName": "kafka",
    "tagVersion": "1.1.0-kafka-2.8.0",
    "upload": true
},
{
    "imageName": "operator",
```

```
    "tagVersion": "1.1.0",
    "upload": true
},
{
    "imageName": "kube-metrics-adapter",
    "tagVersion": "v0.1.16",
    "upload": true
},
{
    "imageName": "kibana",
    "tagVersion": "7.15.1",
    "upload": true
},
{
    "imageName": "grafana",
    "tagVersion": "8.5.0",
    "upload": true
},
{
    "imageName": "filebeat",
    "tagVersion": "7.15.1",
    "upload": true
},
{
    "imageName": "logstash",
    "tagVersion": "7.15.1",
    "upload": true
},
{
    "imageName": "logstash-syslog",
    "tagVersion": "7.6.0",
    "upload": true
},
{
    "imageName": "elasticsearch",
    "tagVersion": "7.15.1",
    "upload": true
```



```
    },
    {
        "imageName": "elasticsearch-insight",
        "tagVersion": "7.16.3",
        "upload": true
    },
    {
        "imageName": "prometheus",
        "tagVersion": "v2.35.0",
        "upload": true
    }
]'

for row in $(echo "${imageTags}" | jq -r '.[] | @base64'); do
    _jq() {
     echo ${row} | base64 --decode | jq -r ${1}
    }
    imageUpload=$(_jq '.upload')
    tagVersion=$(_jq '.tagVersion')
    if [ $imageUpload == "true" ];then
    {
        if [ "${tagVersion}" == "appVersion" ];then
        {
            docker pull docker.io/appviewx/$(_jq '.imageName'):$appVersion
            docker tag docker.io/appviewx/$(_jq '.imageName'):$appVersion $targetImageRegistry/appviewx/$(_jq '.imageName'):$appVersion
            docker push $targetImageRegistry/appviewx/$(_jq '.imageName'):$appVersion
        }
        else
        {
            docker pull docker.io/appviewx/$(_jq '.imageName'):$(_jq '.tagVersion')
            docker tag docker.io/appviewx/$(_jq '.imageName'):$(_jq '.tagVersion') $targetImageRegistry/appviewx/$(_jq '.imageName'):$(_jq '.tagVersion')
            docker push $targetImageRegistry/appviewx/$(_jq '.imageName'):$(_jq '.tagVersion')
        }
        fi
    }
    fi
done
```

Execute the Image Push-Pull Script

To execute the above image push-pull script, run the command

```
./avx_image_pull_push.sh <image-tag> <targetImageRegistry>
```

Uninstall and Cleanup

The process of uninstalling requires one to navigate to the installer directory and execute the following command

```
./appviewxctl uninstall --config .appviewxctl.yaml
```

The following messages are displayed after the uninstall command is executed successfully.

```
./appviewxctl uninstall --config .appviewxctl.yaml

[Init] Using log file at [/avx/appviewxctl-3196327299.log] to dump logs
[Init] Initialise persistent flag config
[Init] Using config file
[Uninstall] Uninstalling appviewx application
[Uninstall] Uninstalling Appviewx application helm chart
[Uninstall] Uninstalling application backup helm chart
[Uninstall] Uninstalling Infra application helm chart
[Uninstall] Uninstalling Third party application helm chart
[Uninstall] Uninstalling IstioOperator from the cluster
[Uninstall] Uninstalling PVCs from the avx namespace
[Uninstall] Uninstalling Pre-requisite helm chart
[Uninstall] Uninstalling Appviewx installed namespaces
[Uninstall] Successfully uninstalled appviewx application and all the related resources
```

Note: In the Managed K8s environments removal of PVCs do not occur at times as it may require patching PVCs first before deletion. This may cause certain error messages to display, indicating that PVC has changed. In case of such an error occurs re-run the above command to solve the issue and uninstall the application.

Sometimes the namespaces take a longer time to be removed. Hence, post installation, check if namespaces are in the terminating state (use the command: **kubectl get namespace**). If any namespace is in the terminating state, manually remove the namespaces by executing the commands below:

```
kubectl get namespace "istio-operator" -o json | tr -d "\n" | sed "s/\"finalizers\": \[[^]]\+\]/\"finalizers\": []/" | kubectl replace

 --raw /api/v1/namespaces/istio-operator/finalize -f - 2>/dev/null
```

```
kubectl get namespace "istio-system" -o json | tr -d "\n" | sed "s/\"finalizers\": \[[^]]\+\]/\"finalizers\": []/" | kubectl replace --raw /api/v1/namespaces/istio-system/finalize -f - 2>/dev/null
```

```
kubectl get namespace "avx" -o json | tr -d "\n" | sed "s/\"finalizers\": \[[^]]\+\]/\"finalizers\": []/" | kubectl replace --raw /api/v1/namespaces/avx/finalize -f - 2>/dev/null
```

```
kubectl delete ns istio-operator --force 2>/dev/null
```

```
kubectl delete ns istio-system --force 2>/dev/null
```

```
kubectl delete ns avx --force 2>/dev/null
```

Troubleshooting

Error	Resolution
403: Access Forbidden - License not available. This error occurs when only the managed kubernetes cluster is restarted every morning.	In case of managed kubernetes, when the cluster is restarted it is recommend to restart all pods.

Chapter 4: MSP Portal User Guide

AppViewX is a leader in automated certificate lifecycle management (CLM) and PKI, providing turnkey solutions for Machine Identity Management and PKI across hybrid infrastructures. The AVX One platform offers a comprehensive multi-tenancy SaaS solution tailored for fast-growing MSPs.

Designed to address complex cybersecurity challenges, the AppViewX MSP portal is a central manager console that features an intuitive interface, providing a unified single-pane-of-glass view and access across customer environments. Built for MSPs serving diverse clients, the console offers end-to-end provisioning, governance, flexible licensing models, tenant insights, reporting capabilities supported by efficient monitoring of multiple customers across various AppViewX cloud environments.

Benefits of AppViewX for MSSPs

- **Comprehensive Machine Identity Management**

  AppViewX provides MSSPs with a comprehensive Machine Identity Management platform, enabling them to manage, secure, and automate the lifecycle of digital certificates, keys, and other machine identities across diverse environments.

- **Enhanced Security Posture**

  By centralizing visibility and control over machine identities, AppViewX empowers MSSPs to proactively identify and mitigate security risks, ensuring continuous compliance with industry regulations.

- **Streamlined Operations and Automation**

  AppViewX's intuitive interface and automation capabilities streamline operations for MSSPs, minimizing manual intervention and reducing the risk of human error.

- **Scalable Solutions for Growth**

  AppViewX offers scalable solutions that cater to the evolving needs of MSSPs and their clients, supporting growth and expansion while maintaining high levels of performance and reliability.

What is the MSP Portal?

Built for MSPs serving diverse clients, the MSP portal is a central manager that offers end-to-end provisioning, governance, flexible licensing models, tenant insights, reporting capabilities supported by efficient monitoring of multiple customers across various AppViewX cloud environments.

The MSP portal is enabled to work with two user persona:

- **SRE User**

  The SRE user will:

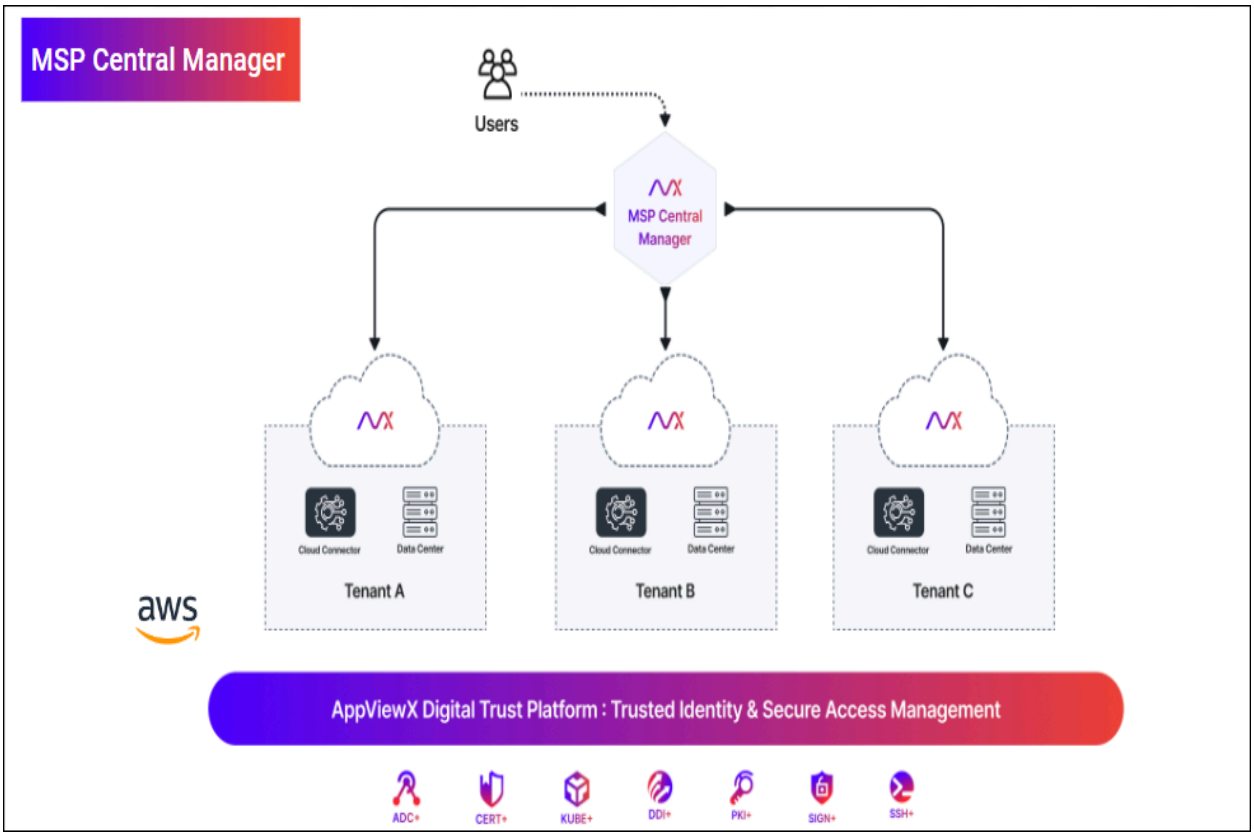
  - Manage license plans for MSPs.
  - Deploy the provisioning cluster for the MSPs.
  - Onboard MSPs in the provisioning cluster.
  - Manage the MSP license in the MSP portal.
  - Import workflows for infrastructure provisioning in the MSP portal.
  - White label the MSP portal.
  - Provision the infrastructure and the application in the MSP portal.
  - Aggregate usage and license data from the MSP portal.
- **MSP User**

  The MSP user will:

  - Create and manage tenants.
  - Upgrade tenant licenses.
  - Create and manage custom plans for tenants.
  - Modify plans associated with clusters.
  - Move tenants across clusters.
  - Whitelabel MSP portal and the tenant(s).
  - Gain holistic insights into tenant application and license metrics.

Key Benefits of the MSP Portal for Partners

The MSP Central Manager connects to all the AppViewX Cloud environments managed by an MSP and offers an unified view of the tenant data in a single view. This provides MSPs with a centralized portal across all their AppViewX Cloud instances.



Key benefits of this implementation include:

- **Governance**

  Consolidated view of tenants, licensing and usage

- **Operations**

  Centralized search of accounts and tenant insights, and application reports with easy access to tenant(s)

- **Provisioning**

  Centralized self-service portal for account onboarding, whitelabeling and provisioning of tenants

- **Mitigation**

  Quick triaging and remediation across the customers environment

Accessing the MSP Portal

After AppViewX's Site Reliability Engineering (SRE) team has set up the MSP Portal for your requirements, you will receive an email with your MSP portal URL, credentials for logging in to the portal, and your license details.

To access the MSP portal:

1. Login to AppViewX with your valid credentials.
2. Go to (**Menu**) > **MSP Portal**.
   The MSP portal dashboard is displayed.

Viewing the MSP Portal Dashboards

1. Go to (**Menu**) > **MSP Portal** > **Governance**.
2. From the following options, select the dashboard you want to view:
   - **Usage and Adoption**
   - **Tenants & License**

   By default, on login, the **Tenants & License** dashboard is displayed.

Usage and Adoption

The **Usage and Adoption** dashboard offers the MSPs holistic insights into application usage and license information across all tenants, as well as into the up-sell and cross-sell opportunities based on usage. This dashboard presents the following tenant-related data across product lines for which the license is enabled:

- Tenant summary
- Tenants by product line
- Platform adoption
- CERT-SSL/TLS adoption
- CERT lifecycle management operations
- ADC app and device adoption

- ADC app actions and device operations
- DDI adoption
- KUBE adoption

Tenants Insights

The Tenant Insights dashboard offers MSPs a single pane of glass view into the tenants by product lines and a seamless way to access individual tenants to perform remedial actions. Data under tenant insights is categorized as:

- **Product Insights**
  - This dashboard lists insights for all product lines for which tenants have been onboarded.
  - To view the insights for an individual product line, under **Dashboard Name**, click the insight name.
  - For the selected product line, this insights dashboard shows certificate category-wise reports for CAs and certificate status.
  - You can use the **Filter by tenant** dropdown list to filter the reports for individual tenants and login to the selected tenant's environment. For instructions on how to login/impersonate a tenant, click here.
- **License Insights**
  - This dashboard shows the license usage for each product line for which tenants have been onboarded.
  - To view the license usage details for a product line, click the accordion for the product line to expand (and collapse) the details view.
  - You can use the **Filter by tenant** dropdown list to filter the license usage details for individual tenants.

Managing Clusters in the MSP Portal

Accessing the Cluster Management Inventory

Go to (**Menu**) > **MSP Portal** > **Management** > **Clusters**.

The **Cluster Management** inventory is displayed.

Understanding the Cluster Management Inventory

Fields in the Cluster Management inventory

Field	Description
Filter	Lets you filter the cluster inventory entries for the following parameters: • Cluster type • Region • Tag name • Version • Status
Cluster Name	Displays the name of the cluster
Cluster Type	Displays the type of the cluster. The valid types are: • Compute • Database
Version	Displays the version of the software. Note: It also displays if an upgrade is available for the cluster.
Status	Displays if the cluster is active.
Region	Displays the region in which the cluster is created.
Tag Name	Displays the name of the associated tag.
Tenants	Displays the number of tenants associated with the respective cluster.
Tenants Threshold	Displays the threshold for the tenants that can be associated with the respective cluster.
Updated on	Displays the date of the last cluster update.

Fields in the Cluster Management inventory (continued)

Field	Description
Action	Displays the possible actions that can be taken on the cluster. The possible actions are:  - Delete Cluster  - Modify Cluster

Modifying Clusters

Important: MSPs can only modify the plans associated with a cluster.

To modify a cluster:

1. Go to (**Menu**) > **MSP Portal** > **Management** > **Clusters**.
   The **Cluster Management** inventory is displayed.
2. From the **Cluster Management** inventory, select the checkbox for the cluster you want to modify.
3. For the selected cluster, from the **Actions** column, click
   The **Modify Cluster** section is displayed.
4. Update the cluster plan details as required.
5. Click **Modify**.
   The cluster metadata is updated in the **Cluster Management** inventory.

Managing Tags

There is a common tag associated with one Database and Compute cluster pair in each region. For example, a compute and database cluster in the XYZ region will have a common tag. There is one-to-one mapping between the cluster pair and the tag. Defining tags is crucial for upgrading tenants to newer versions and ensures that both compute and database clusters are upgraded together.

You can add your own tag and associate it with a compute and database cluster pair.

Note: A cluster pair cannot be associated with multiple tags.

To manage tags for clusters:

1. Go to (**Menu**) > **MSP Portal** > **Management** > **Clusters**.
   The **Cluster Management** inventory is displayed.
2. From the **Cluster Management** inventory, click **Manage Tags**.
   The **Manage Tags** window is displayed.
3. Enter/Select the tag details.

Field descriptions for the tag details

Field	Description
***Tag Name**	Enter a valid tag name. **Note:** If a version is associated with the tag, you will not be able to update the tag.
***Region**	Select the region to be associated with the tenant cluster.
***Compute Cluster**	From the dropdown list, select the compute cluster with which this tag will be associated. **Note:** The compute cluster values will be loaded based on the region selected and the ones that are not already associated with a tag.
***Cluster Name in Cloud**	This is a read-only field that is auto-populated based on the compute cluster selection.
***Database Cluster**	From the dropdown list, select the database cluster with which this tag will be associated.

Field	Description
	Note: The values for database clusters loaded will correspond to the same plan as the compute cluster.
**: Mandatory fields*	

4. Click **Add Tag**.
   The new tag will be associated with the cluster pair and the cluster inventory will be updated accordingly.

Managing Plans in the MSP Portal

The **Plan Management** inventory lists the predefined plans as well as the custom plans (created when the predefined plans don't align with the tenants' requirements) for each product type offered by AppViewX. This inventory is the source for the **Plan & Product Details** section that is displayed at the time of creating and upgrading a tenant.

- Accessing the Plan Management Inventory
- Understanding the Plan Management Inventory
- Creating Custom Plans

Accessing the Plan Management Inventory

Go to (**Menu**) > **MSP Portal** > **Management** > **Plans**.
The **Plan Management** inventory is displayed.

Understanding the Plan Management Inventory

Fields in the Plan Management inventory

Field	Description
Search	Enter a keyword/list of keywords to search for records that match the search value.
Plan Name	AppViewX-assigned name of the plan

Fields in the Plan Management inventory (continued)

Field	Description
Product Type	Product type that is part of the plan
Plan Category	For CERT+, plans belong to one of the following categories: • Server certificate • Device certificate For PKI+, plans belong to one of the following categories: • Certificate authority • Issuance certificate For DDI+, plans belong to the following categories: • DDI Objects For KUBE+, plans belong to the following categories: • Kube Cluster For SIGN+, plans belong to the following categories: • Signatures
Plan Type	Specifies if the plan is a licensed one or for a free trial
Source Type	Tenants can be onboarded from the AppViewX website or from the AWS marketplace. This field will have one of the following two values: • Internal (plan defined by AppViewX's PM team) • AWS Market Place (plan created in accordance with AWS)
Plan Description	Additional details related to the plan

Fields in the Plan Management inventory (continued)

Field	Description
Action	: View plan details. : Delete a plan **Note:** Only custom plans can be deleted. To delete a plan: 1. From the **Plan Management** inventory, for the required plan, from the Action column, click . The **Delete Plan** confirmation dialog box is displayed. 2. Click **Yes**. The selected plan is deleted.

Creating Custom Plans

Custom plans can be created when any of the predefined product plans do not meet the tenants' requirements. Your tenant success team or the sales team will raise a request for a custom plan, as required by a tenant. The AppViewX SRE team then creates a custom plan based on the requirement. These custom plans can then be used for all tenants.

Note:

- Custom plans can be created only for licensed versions and not for free trials.
- If yours is a pool-based license, the MSP's license will be validated for the product limits before a custom plan is created.

To create a custom plan:

1. Go to (**Menu**) > **MSP Portal** > **MSP Management** > **Plans**.
   The **Plan Management** inventory is displayed.
2. From the **Plan Management** inventory, click **Add Plan**.
   The **Add Plan** dialog box is displayed.
3. Enter/Select the custom plan details.

Field descriptions for the custom plan details

Field	Description
***Plan Source Type**	Tenants can be onboarded from the AppViewX website or from the AWS marketplace. Plans are, therefore, created in accordance with the source of onboarding. • If this new plan has been created by AppViewX's PM team, from the dropdown list, select **Internal**.
***Product Type**	From the following options, select the product type for which this new plan will be applicable: • DDI+ • CERT+ • SIGN+ • KUBE+ • PKI+
***Plan Category**	This dropdown list is populated based on the **Product Type** selected. For CERT+, select a plan category from the following options: • **Device Certificate (Default)** • **Server Certificate** For PKI+, select a plan category from the following options: • **Issuance Certificate (Default)** • **Certificate Authority** For DDI+, select a plan category from the following options: • **DDI Objects** For KUBE+, select a plan category from the following options: • **Kube cluster** For SIGN+, select a plan category from the following options:

Field	Description
	• **Signatures**
***Number of Managed Certificates**	**Note:** This field is displayed only **Product Type** = **CERT+**. Enter the number of device/server certificates that will be managed by this plan.
***Number of Issuance Certificates**	**Note:** This field is displayed only **Product Type** = **PKI+** and **Plan Category** = **Issuance Certificate**. Enter the number of issuance certificates that will be managed by this plan.
***Number of CA**	**Note:** This field is displayed only **Product Type** = **PKI** and **Plan Category** = **Certificate Authority**. Enter the number of certificate authorities that will be managed by this plan.
***Number of DDI Objects**	**Note:** This field is displayed only **Product Type** = **DDI+**. Enter the number of DDI objects that will be managed by this plan.
***Number of Managed Clusters**	**Note:** This field is displayed only **Product Type** = **KUBE+**. Enter the number of KUBE clusters that will be managed by this plan.
***Number of Managed Signatures**	**Note:** This field is displayed only **Product Type** = **SIGN+**. Enter the number of signatures that will be managed by this plan.
***Plan Name**	By default, based on the above selections, a plan name is auto-populated in this field, in adherence to AppViewX's recommended naming convention (which is used for the predefined plans). If required, you can edit the plan name.

Field	Description
***Plan Description**	Enter any additional details related to the plan.

4. Click **Create**.

   The new plan is listed in the **Plan Management** inventory.

> **Note:** If a plan has to be modified, the recommended course of action is to delete that plan and create a new one.

Managing Tenants in the MSP Portal

Tenant management in the MSP Portal includes the provision to create, delete, and move tenants across clusters, extend the free trial of a tenant as well as move tenants from a free trial to a licensed account, and so on.

Accessing the Tenant Management Inventory

Go to (**Menu**) > **MSP Portal** > **Management** > **Tenants**.

The **Tenant Management** inventory is displayed.

Understanding the Tenant Management Inventory

Fields in the Tenant Management Inventory

Field	Description
Filter	Allows you to apply filters to the tenant inventory to search for specific tenant records. The available filters are: • Plan • Tenant type • Request type • Cluster • Region • Product type • Status
Add Tenant	Allows you to onboard a tenant manually
Company Name	Displays the name of the tenant company
Tenant Email	Displays the tenant's registered email ID
Tenant	Displays the name of the tenant.
Tenant Domain	Displays the domain used by the tenant to access the product.
Expiration Date	Hover over **View Expiry** to see the expiry date of each product the tenant has a trial/ licensed version of.
Product Type	Displays the AppViewX product name(s) that the tenant has a trial/licensed version of
Plan Name	Displays the name of the AppViewX plan that the user has subscribed to. The plan name gives the details of the product configuration/capability made available to the tenant.
Plan Type	Displays whether the tenant is using a trial plan or is a licensed tenant. It also displays the status of the tenant if they have opted for the trial plan.
Status	Displays the status of the tenant as:

Fields in the Tenant Management Inventory (continued)

Field	Description
	• Active • Inprogress • Expired • Failed
tenant Type	Displays the tenant type
PKIaaS	Displays whether PKIaaS has been enabled or not
Region	Displays the region the tenant is associated with
Cluster Name	Displays the name of the cluster associated with the tenant
DB Snapshot ID	Displays the snapshot ID of the database to which the tenant has been onboarded in the compute cluster
Request Info	Displays the information of the workflow request processed for tenant related actions such as onboarding, License upgrade and so on. **Note:** To view the request, click the request ID. A stage-wise view of the workflow request is displayed.
Request Status	Displays the status of the tenant addition request
Source	Displays the source from where the tenant was onboarded. For example, tenant inventory, AWS, SaaS web page, and so on.
Action	Displays the icons for actions that can be performed for the tenants: **Note:** These action icons are also displayed when you hover over any tenant row. Only the actions valid for that particular tenant will be displayed.

Fields in the Tenant Management Inventory (continued)

Field	Description
	• (**Tenant Login**) • (**Delete Tenant**) • (**Extend Trial**) • (**Tenant Cluster Movement**) • (**Activate License**) • (**Repropagate tenant details**)

Creating a New Tenant

Note: All tenant creation and management activities will be validated against the plans in the MSP license. MSPs can create and manage tenants only within the limits specified in the plans and for the specified products. Otherwise, the application will throw an error.

1. Go to (**Menu**) > **MSP Portal** > **Management** > **Tenants**.
   The **Tenant Management** inventory is displayed.
2. From the **Tenant Management** inventory, click **Add Tenant**.
   The **Add Tenant** page is displayed.
3. Enter/Select the **General** tenant details.

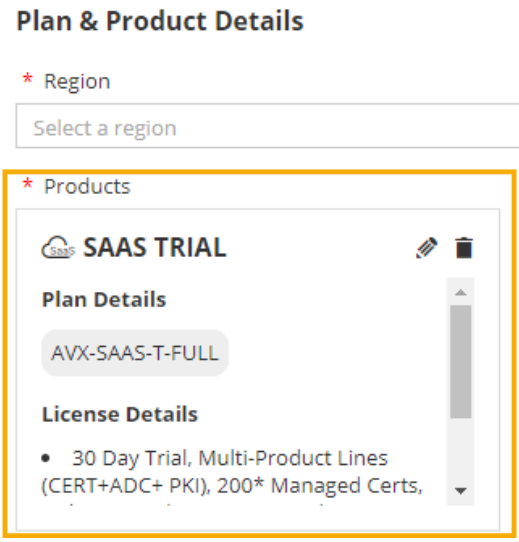
Field descriptions for the General tenant details

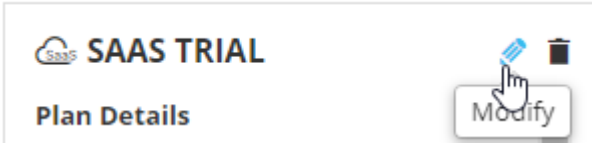
Field	Description
***Company Name**	Enter the name of the tenant's company. For example: newco.
***Email**	Enter a valid email address of the account owner. For example: xxx@newco.com
***Customer Type**	From the dropdown list, select **Customer**.
**: Mandatory fields*	

4. In the **Tenant Details** section, in the **Domain** field (mandatory), enter a valid domain name.
5. Select the **Plan & Product Details** for this tenant.

Field	Description
***Region**	Select the region in which the tenant's SaaS account will be set up.
***Products**	a. Click **Add Product**. The **Modify Product** window is displayed. b. From the **Product Type** dropdown list, select the product for which the tenant is evaluating the trial version/has purchased the license. c. This field is displayed only if **Product Type** = **CERT+**. From the following options, select the **License Subscription Model** for CERT+ : • **Certificate instance based count** (existing): Certificate count is derived from the number of connectors associated with a certificate. • **Certificate based count** (introduced in v2023.1.0 FP3): Certificate count for the license is based on the number of certificates in the inventory. For more details on the CERT+ licensing models, refer to the CERT+ User Guide. d. From the **Plan Name** dropdown list, select the plan(s) chosen by the tenant. This list is populated based on the selected product type. **License Details of the selected plan(s)** are displayed in the **Modify Product** window. This banner displays all the details of the license plan selected, such as the number of certificates managed under each plan, issuing CAs, trial duration, and so on. e. In the **End Date** field, click  to specify the expiration date according to the selected product and plan. **Note:** This field is not applicable for a trial tenant. f. Click **Add**.

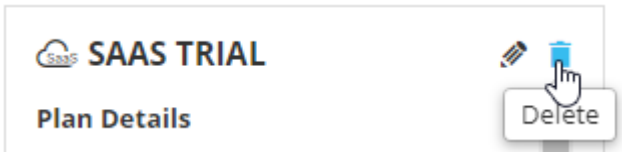
The product details for the selected product(s) are displayed as shown in the image below:



g. To modify the plan details, click 

.

h. To delete the plan details, click



.

i. If the tenant is using a licensed version of multiple product types and wants to have a common license end date for all product types and plans selected, enable the **Enable Common License End Date** toggle key.

j. In the **End Date** field, click to select a common license end date for all product types. Refer to the tenant's purchase order for the latest license end date for this tenant.

> **Note:** If you need to modify this end date after a tenant has been created, you will be required to click **Activate License/Update License** from the **Action** column for the required tenant and update the **Plan date**.

6. Select the **Cluster & Other Details**.

Field descriptions for the Cluster & Other Details

Field	Description
***Tag Name**	Select the applicable tag from the dropdown menu. **Note**: Tags are loaded on the basis of the region and plan associated.
***Compute Cluster**	The name of the compute cluster attached to the specified region is displayed.
***Database Cluster**	The name of the database cluster attached with the specified region is displayed.
***Request Type**	From the dropdown list, select a tenant request type from the following options: • Sales • Channel • tenant Success • Marketing • Others
**: Mandatory fields*	

7. Click **Add**.
   The tenant is added to the inventory.

   The tenant's URL, username, and temporary password are automatically emailed on the email address specified in the **General** tenant details.

   The MSP header, logo, and favicon, as configured in the MSP Portal, is also applied to the end tenant's environment.

Extending Free Trial

You have the provision to extend the duration of a tenant's free trial. This is subject to approvals from the relevant stakeholders based on discussion with the prospect requesting trial extension. As a standard practice, a maximum extension of 60 days will be allowed, based on approval.

Process flow for trial extension

1. Prospect requests for **X** days of trial extension.
2. MSP gets notified on the account and trial extension duration.
3. MSP extends the free trial period.

To extend the free trial duration of a prospect/tenant:

1. Go to (**Menu**) > **MSP Portal** > **Management** > **Tenants**.
   The **Tenant Management** inventory is displayed.
2. From the **Tenant Management** inventory, for the required trial tenant, from the **Action** column, click .

   Tip: You can also hover your mouse over the selected tenant to view the icon.

   The **Tenant Trial Extension** page is displayed.

   The tenant details are auto-populated and non-editable.
3. To extend a tenant's trial period, in the **Number of days to extend** field, enter the number of days for which the trial is to be extended.
   Once the default 30-day trial duration expires, AppViewX lets you extend the trial duration for an additional 60 days (which makes the total trial duration to be 90 days). The trial period can be extended by a minimum of 15 days and a maximum of 30 days at one time. So, you can extend the trial:
   • twice for 30 days each time
   • four times for 15 days each time
   • twice by 15 days and once by 30 days

   Note:
   If the value entered in this field is more than the **Remaining Trial Extension Period**, an error message is displayed.
4. Click **Extend Trial**.
   The trial period is extended and all the columns in the inventory are replaced with the current values for the tenant.

Activating a Tenant License

Note: All tenant creation and management activities will be validated against the plans in the MSP license. MSPs can create and manage tenants only within the limits specified in the plans and for the specified products. Otherwise, the application will throw an error.

1. Go to (**Menu**) > **MSP Portal** > **Management** > **Tenants**.
   The **Tenant Management** inventory is displayed.
2. From the **Tenant Management** inventory, from the **Action** column, click .

   Tip: You can also hover your mouse over the selected tenant to display the icon.

   The **Upgrade Tenant** page is displayed.

   Note: Fields in the **Tenant Details** and the **Current Plan & Product Details** sections are auto-populated and cannot be edited.
3. In the **Upgrade Plan & Product Details** section:
   a. From the **License Action** dropdown list, select the reason for the license upgrade.
   b. From **Products**, click **Add Products** to modify the product and plan details for the tenant.
   For descriptions of the fields in this section, refer to the Creating a New Tenant section.

   Important: The **Upgrade Plan & Product Details** section includes the following warning: **Only the below selected product license will be pushed to tenant. Make sure to select the necessary products and plans.** This means that all upgrades done to the license here will overwrite the existing license details for the tenant. Users are, therefore, advised to ensure that the upgraded license includes all products and plans that a tenant requires, and not just the modifications.

   Note: In the event that the upgrade includes additional products added to the tenant's license, AppViewX includes the **Enable Common License End Date** to set a common license end date for all of a tenant's new and existing products, so that all products can then be renewed on the same date.

4. To update a tenant's cluster, edit the fields in the **Cluster & Other Details** section.
   For descriptions of the fields in this section, refer to the Creating a New Tenant section.

   If the license activation requires the tenant to be migrated from one cluster to another (depending on the plan selected), the **Move Cluster & Upgrade Tenant** confirmation dialog box is displayed.
5. Read through the warning in the **Move Cluster & Upgrade Tenant** confirmation dialog box and, to proceed with cluster migration, click **Yes**.
   If you do not wish to migrate the cluster, click **Cancel**. In this case, the license activation process cannot proceed. In the **Upgrade Tenant** window, either update the plan accordingly or click **Cancel**.
6. Click **Upgrade**.
   The tenant license is upgraded as specified.

> **Note:** You can check the workflow execution of this request by clicking the Request ID in the **Request Info** column in the tenant inventory.

Deleting Trial Tenants

All trial accounts will be available for 30 days with further data retention for 60 more days.

MSPs have the provision to delete trial tenants at the end of the data retention period. As the trial period comes to and end, the tenant can opt for the following:

- Extension of the trial period: The trial period can be extended for up to a maximum of 60 days. For more information on extending the trial period, click here.
- Upgrading to a licensed version: At the end of the trial period, the tenant can be upgraded to a licensed version. For more information on this, click here.
- Not upgrade: If a trial customer does not wish to upgrade to a licensed account, they can be deleted manually through the MSP Portal.

> **Note:** Data for trial customers can be retained for 60 days.

To delete a trial customer on-demand:

1. Go to (**Menu**) > **MSP Portal** > **Management** > **Tenants**.
   The **Tenant Management** inventory is displayed.
2. Under the **Plan Type** column, confirm that the tenant is on a **Trial** plan.

3. Under the **Status** column, confirm if it is an **Active** tenant.
4. To delete the tenant, under the **Action** column, click  .

> **Tip:** You can also hover your mouse over the selected tenant to display the  icon.

The **Delete Tenant** pop-up window is displayed.

5. Enter the **Reason for Tenant Deletion**.
6. Click **Yes**.

The offboarding workflow will be triggered and the tenant will be deleted from the inventory. In the tenant **Inventory**, the tenant **Status** is set to **Deleted**.

If the tenant is a trial customer, the database snapshot is also deleted. For a licensed tenant, however, the database snapshot is retained.

Moving Tenants Across Clusters

MSPs have the provision to move tenants from one cluster to another. A licensed tenant can request for a dedicated cluster in the workload cluster as well. The tenant will have a dedicated schema in the Database cluster.

To move a tenant to another cluster:

1. From the **Tenant Management** inventory, for the required tenant, from the **Action** column, click  .

> **Tip:** You can also hover your mouse over the selected tenant to display the  icon.

The **Tenant Cluster movement** page is displayed.

Only the following fields can be modified for moving tenants across clusters:

- **Choose a type of cluster to move**
- **Tag Name**

The remaining details are auto-populated.

To move a tenant across clusters, the **Tenant Cluster movement** page captures the following details:

Field	Description
***Tenant ID**	Displays the Tenant ID. **Note:** This is a read-only field and is auto-populated based on the tenant selected.
***Region**	Displays the region with which the tenant cluster is associated. **Note:** This is a read-only field and is auto-populated based on the tenant selected.
***Product Type**	Displays the product type(s) the tenant has a trial/licensed version for **Note:** This is a read-only field and is auto-populated based on the tenant selected.
***Current Plan**	Displays the current plan of the tenant. **Note:** This is a read-only field and is auto-populated based on the tenant selected.
***Current Tag Name**	Displays the current Tag Name of the tenant. **Note:** This is a read-only field and is auto-populated based on the tenant selected.
***Current Compute Cluster**	Displays the name of the current compute cluster associated with the tenant. **Note:** This is a read-only field and is auto-populated based on the tenant selected.
***Current Database Cluster**	Displays the name of the current database cluster associated with the tenant.

Field	Description
	Note: This is a read-only field and is auto-populated based on the tenant selected.
***Tag Name**	Select the **Tag Name** from the options available in the dropdown.
***Compute Cluster**	Displays the name of the Computer Cluster mapped to the tag. **Note:** This is a read-only field and is auto-populated based on the tag selected.
***Database Cluster**	Displays the name of the Database Cluster mapped to the tag. **Note:** This is a read-only field and is auto-populated based on the tag selected.

2. Modify the tenant cluster details, as required.

Field descriptions for the editable cluster details

Field	Description
***Choose a type of cluster to move**	Select the required cluster type to move the tenant to.
***Tag Name**	Select the **Tag Name** from the options available in the dropdown.
**: Mandatory fields*	

Field descriptions for the non-editable cluster details

Field	Description
***Tenant ID**	Displays the Tenant ID. **Note:** This is a read-only field and is auto-populated based on the tenant selected.
***Region**	Displays the region with which the tenant cluster is associated.

Field	Description
	Note: This is a read-only field and is auto-populated based on the tenant selected.
***Product Type**	Displays the product type(s) the tenant has a trial/licensed version for **Note:** This is a read-only field and is auto-populated based on the tenant selected.
***Current Plan**	Displays the current plan of the tenant. **Note:** This is a read-only field and is auto-populated based on the tenant selected.
***Current Tag Name**	Displays the current Tag Name of the tenant. **Note:** This is a read-only field and is auto-populated based on the tenant selected.
***Current Compute Cluster**	Displays the name of the current compute cluster associated with the tenant. **Note:** This is a read-only field and is auto-populated based on the tenant selected.
***Current Database Cluster**	Displays the name of the current database cluster associated with the tenant. **Note:** This is a read-only field and is auto-populated based on the tenant selected.
***Compute Cluster**	Displays the name of the Computer Cluster mapped to the tag.

Field	Description
	Note: This is a read-only field and is auto-populated based on the tag selected.
***Database Cluster**	Displays the name of the Database Cluster mapped to the tag. **Note:** This is a read-only field and is auto-populated based on the tag selected.

3. Click **Trigger Cluster Movement**.
   The tenant will be moved to the specified cluster.

Upgrading License Details for a Tenant

1. Go to (**Menu**) > **MSP Portal** > **Management** > **Tenants**.
   The **Tenant Management** inventory is displayed.
2. From the **Tenant Management** inventory, from the **Action** column, click .

   Tip: You can also hover your mouse over the selected tenant to display the icon.

   The **Upgrade Tenant** page is displayed.

   Note: Fields in the **Tenant Details** and the **Current Plan & Product Details** sections are auto-populated and cannot be edited.

3. In the **Upgrade Plan & Product Details** section:

   a. From the **License Action** dropdown list, select the reason for the license upgrade.

   b. From **Products**, click **Add Products** to modify the product and plan details for the tenant.
   For descriptions of the fields in this section, refer to the Creating a New Tenant section.

Important: The **Upgrade Plan & Product Details** section includes the following warning: **Only the below selected product license will be pushed to tenant. Make sure to select the necessary products and plans.** This means that all upgrades done to the license here will overwrite the existing license details for the tenant. Users are, therefore, advised to ensure that the upgraded license includes all products and plans that a tenant requires, and not just the modifications.

Note: In the event that the upgrade includes additional products added to the tenant's license, AppViewX includes the **Enable Common License End Date** to set a common license end date for all of a tenant's new and existing products, so that all products can then be renewed on the same date.

4. To update a tenant's cluster, edit the fields in the **Cluster & Other Details** section.
   For descriptions of the fields in this section, refer to the Creating a New Tenant section.

   If the license activation requires the tenant to be migrated from one cluster to another (depending on the plan selected), the **Move Cluster & Upgrade Tenant** confirmation dialog box is displayed.
5. Read through the warning in the **Move Cluster & Upgrade Tenant** confirmation dialog box and, to proceed with cluster migration, click **Yes**.
   If you do not wish to migrate the cluster, click **Cancel**. In this case, the license activation process cannot proceed. In the **Upgrade Tenant** window, either update the plan accordingly or click **Cancel**.
6. Click **Upgrade**.
   The tenant license is upgraded as specified.

Note: You can check the workflow execution of this request by clicking the Request ID in the **Request Info** column in the tenant inventory.

Repropagating Tenant Details

To repropagate tenant details:

1. Go to (**Menu**) > **MSP Portal** > **Management** > **Tenants**.
   The **Tenant Management** inventory is displayed.
2. From the **Tenant Management Inventory**,for the required tenant, from the **Action** column, click .

Tip: You can also hover your mouse over the selected tenant to view display the  icon.

3. In the **Re-propagate Tenant Details** confirmation dialog box, click **Yes**.

Viewing MSP License Usage Details

Go to (**Menu**) > **Platform** > **System Administration** > **License**.
The **Settings :: License** page is displayed.

Multiple metrics to track your license usage are displayed on this page.

For a pool based license, the metrics will show the permitted and used count of resources.

For a pay as you go license, the metrics will show the number of resources used; there is no maximum limit for a pay as you go license.

Impersonating Tenants

Tenant impersonation allows MSP admins to act on behalf of their clients with the help of MSP credentials. This feature enables MSP admins to provide support, manage resources, and perform administrative tasks directly within the clients' environments.

Important: By default, the MSP admin has impersonation access for all tenants.

Important: MSP admins can impersonate a tenant only if the tenant, at the time of their first login, has selected the **Allow MSP User impersonation** in the EULA/terms and conditions page.

To enable tenant impersonation for the MSP admins:

1. Go to (**Menu**) > **Platform** > **Identity** > **Resource**.
2. From the list of resources displayed, select **MSP access**.
   The **Resource > Modify :: <MSP access>** page is displayed.
3. Under the **Access Control** tab, from the **List**, select **Tenants**.

4. Select checkboxes corresponding to the tenants the MSP admin needs to impersonate.
5. Click **Save**.

• Logging in to a Tenant's Environment

Logging in to a Tenant's Environment

Chapter 5: AppViewX Windows Gateway Setup

This guide outlines the steps for installing the AppViewX Windows Gateway for enabling communication between AppViewX and Windows. It also includes the steps for installing and using the AppViewX validator to validate the accessibility of the target machine on which the AppViewX Windows Gateway will be installed.

Overview

AppViewX Windows Gateway

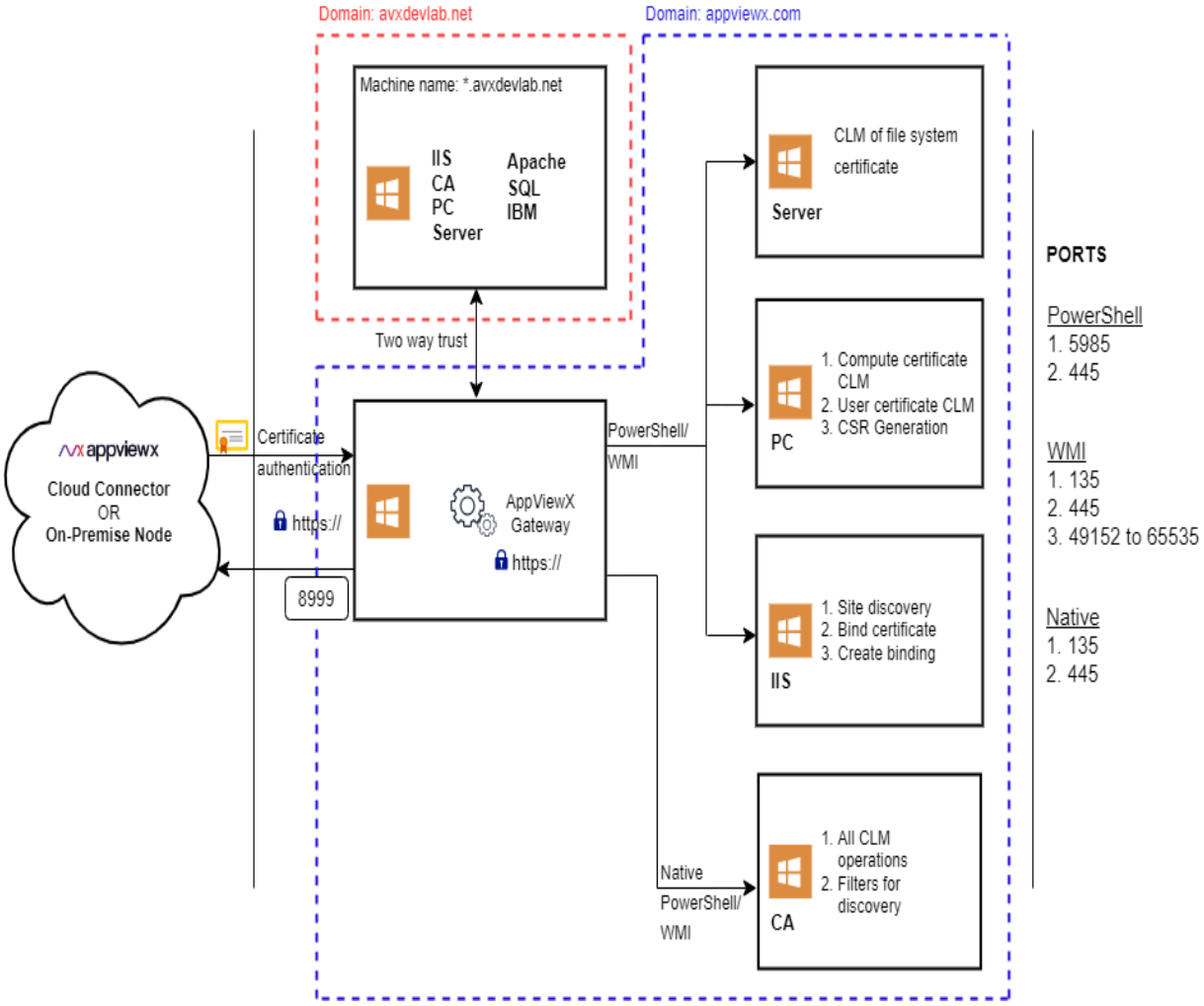
The AppViewX Windows Gateway is packaged with two components:

- AppViewX Windows Gateway Service
- AppViewX Windows Gateway Troubleshooting tool

AppViewX Windows Gateway service is a Windows Communication Foundation service that enables secure communication between AppViewX and Windows server infrastructure. Following are the key features of the that are supported by AppViewX for Windows Server Infrastructure:

- Certificate Life Cycle Management (CLM) on Windows servers (version 2012 R2 and above), Microsoft CA Servers, IBM Websphere, and Weblogic.
- Binding of certificates to IIS (Version 7.5 and above)
- Discovering certificates from the file system
- Executing custom scripts on PowerShell

AppViewX Windows Gateway Troubleshooting tool facilitates the trouble shooting of any issues in the communication between AppViewX Windows Gateway service and the Windows server infrastructure in your premises.

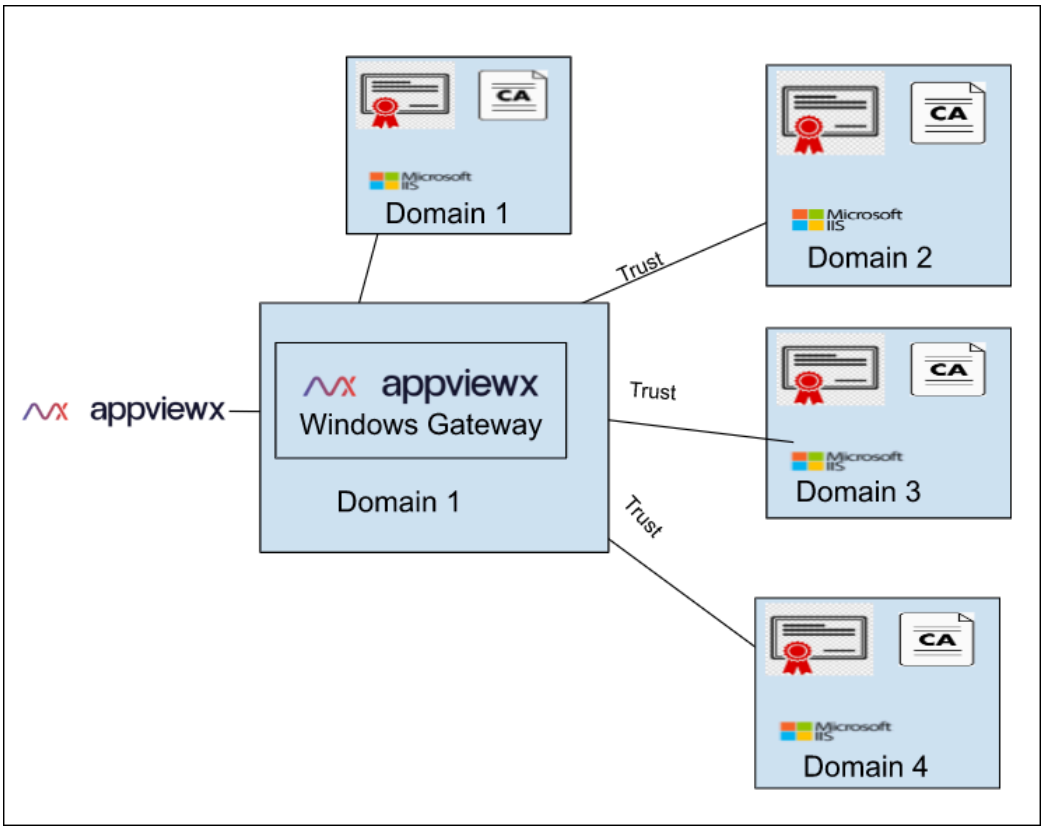


Deployment Modes

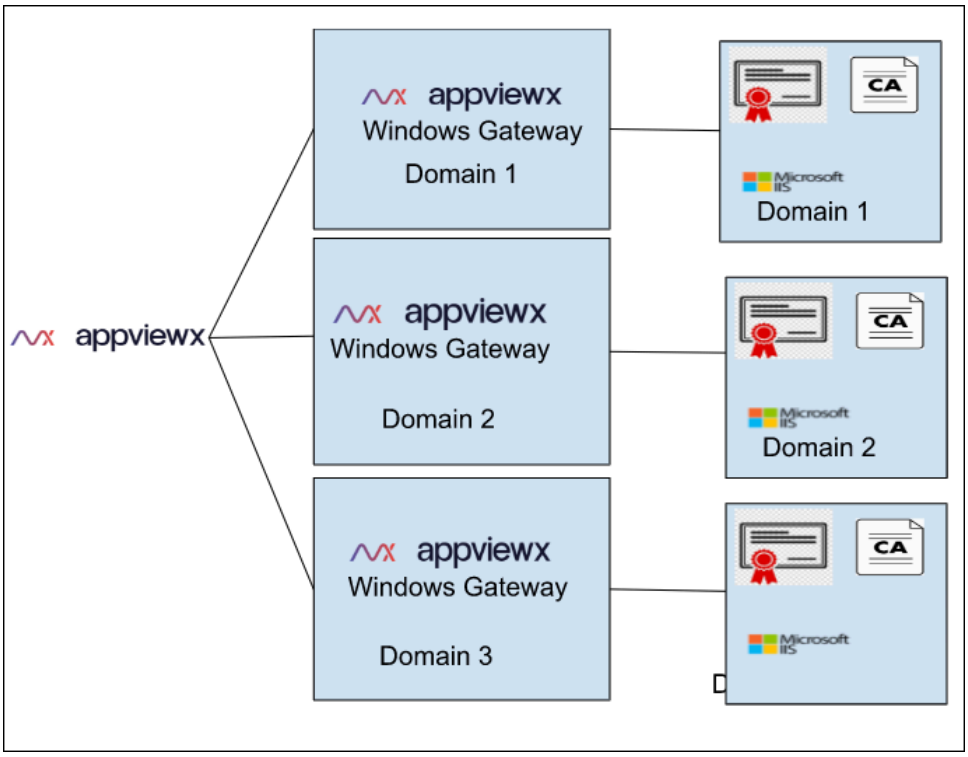
AppViewX WG installation is different for trusted and untrusted domains.

Trusted Domains

If your organization has multiple domains and each of these domains are trusted, then as depicted in the following figure, one installation of the AWG would be sufficient to manage the Windows server infrastructure of all the domains.



Alternatively, if the domains are independent, then at least one installation of the AWG is needed for each such untrusted domain, as shown in the figure below.



Setting up the AppViewX Windows Gateway

Step 1: Checking Prerequisites

Software prerequisites

Name	Description
Operating System	AppViewX Windows Gateway is supported Windows Server 2012, Windows Server 2012 R2,

Software prerequisites (continued)

Name	Description
	Windows Server 2016, Windows Server 2019, and Windows Server 2022.
.NET framework	.NET framework version 4.5.2 is required.
Admin access	Administrator privilege is needed to install AppViewX Windows Gateway.
PowerShell	PowerShell version 4.0 is needed

Hardware prerequisites

Hardware	Capability
RAM	8 GB
HDD	10 GB
CPU	Intel or AMD processor with 64-bit support, 1.8 GHz or faster processor

Firewall prerequisites

Component	Port
Default Port communication from AppViewX to a AppViewX Windows Gateway	8999

Note:

- The firewall must not block the following port and the respective port must open on the Agent.
- During the installation of AppViewX Windows Gateway, the default port can be reconfigured. For more details refer Step 3 of the installation process.

Step 2: Downloading the AppViewX Windows Gateway Installer

Download and unarchive the **AppViewX.CertPlus.Installer.zip** file from the release portal. The download package consists of the following files:

File Name	Description
AppViewX.CertPlus.Installer.exe	Installer executable
ClientCertificateGateway.pfx	Default client certificate
ServerCertificateGateway.pfx	Default server certificate
config.xml	Application configuration settings that will override the default settings after the AppViewX Windows Gateway is installed.
Readme.txt	Help file with details of the AppViewX Windows Gateway.
InstallationLog.txt	Logs the success and error messages from the installation process.

Step 3: Installing the AppviewX Windows Gateway

Before you begin: By default, the AppViewX Windows Gateway securely communicates with AppViewX using the server/client certificates that are shipped along with the AppViewX Windows Gateway installer. If you choose to use a different server and client certificate for authentication, then follow the steps below:

1. From Windows explorer, browse to the location where you have unarchived the AppViewX Windows Gateway installer package.
2. Rename the default server certificate **ServerCertificateGateway.pfx** to **ServerCertificateGateway-Backup.pfx** and the client certificate file **ClientCertificateGateway.pfx** to **ClientCertificateGateway-Backup.pfx**.
3. Copy the server and client certificates that you intend to use in this directory.
4. Rename the server certificate file to **ServerCertificateGateway.pfx** and the client certificate file to **ClientCertificateGateway.pfx**, and then replace the default certificates in the installation folder.

Note: While installing the AppViewX Windows Gateway, you will be prompted to provide the server and client passwords.

CAUTION: If the certificate is replaced, ensure that the respective password has been provided to add the certificate to the store. The incorrect password during the installation of AppViewX Windows Gateway will cause the Windows Agent installation to fail.

1. Execute the **AppViewX.CertPlus.Installer.exe** file.
   The welcome screen for the setup wizard is displayed.
2. Click **Next**.
   The **License Agreement** is displayed.
3. Select **I accept the terms in the license agreement**.
4. Click **Next**.
   The **Destination Folder** screen is displayed.
5. To install the AppViewX Windows Gateway at the default location, click **Next**.
   OR

   To change the default destination folder:

   a. Click **Change**.

   b. On the **Change Current Destination Folder** screen, use the **Look in** dropdown list/  (up one level) icon/  (create new folder) icon to navigate to/create the required destination folder.

   c. On the **Change Current Destination Folder** screen, use the **Look in** dropdown list/  (up one level) icon/  (create new folder) icon to navigate to/create the required destination folder.

   d. Click **OK**.

   e. On the **Destination Folder** screen, click **Next**.
      The **Optionally you can modify the below details** screen is displayed.

   f. Enter the details as required.

Field descriptions for the details

Field	Description
Please select default certificate store	Select the certificate store from which the certificates will be discovered and pushed to by AppViewX from the following options: • Current User Store This type of certificate store is local to a user account on a computer. It is located in the registry under the HKEY_CURRENT_USER root.

Field	Description
	• Local Machine Store (default) This type of certificate store is local to a computer and global to all the user accounts on the computer. It is located in the registry under the HKEY_LOCAL_MACHINE root. This configures the gateway for communicating with the appropriate certificate store.
Port	Port for accessing the service. Default value: 8999 (can be modified if required)
Server certificate thumbprint	If you are using a custom certificate, enter the corresponding server certificate thumbprint value.
Client certificate password	Password for accessing the client certificate For custom client certificates, enter the certificate password.
Server certificate password	Password for accessing the server certificate For custom server certificates, enter the certificate password.

Note: Refer to the **Before you Begin** section to use custom server and client certificates.

6. Click **Next**.
   The **Ready to Install the Program** screen is displayed.
7. Click **Install**.
   This will:
   • Install the AppViewX Windows Gateway Troubleshooter tool
   • AppViewX Windows Gateway service

• Navigating through the Installation

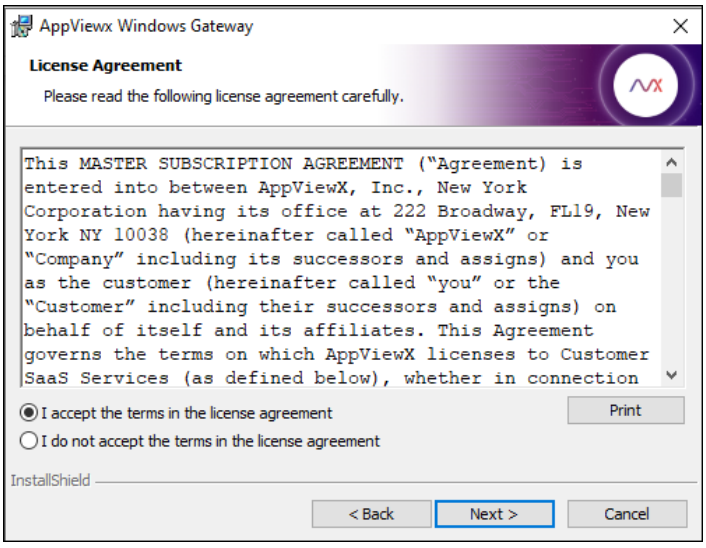
Navigating through the Installation

1. Execute the **AppViewX.CertPlus.Installer.exe** file.

   The following welcome screen for the setup wizard is displayed.

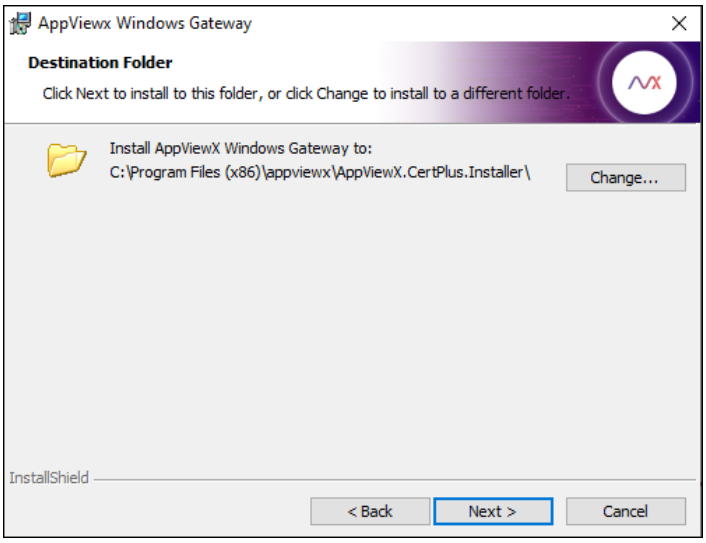
2. Click **Next**.

   The **License Agreement** is displayed.



3. Select **I accept the terms in the license agreement**.
4. Click **Next**.

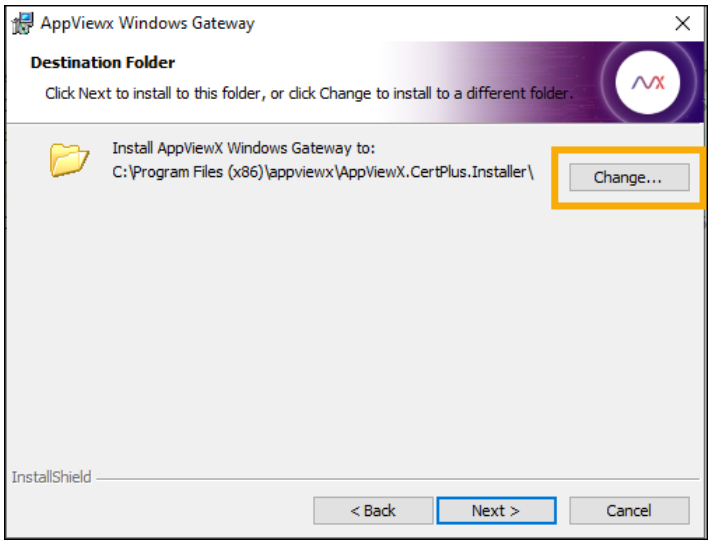
   The **Destination Folder** screen is displayed.



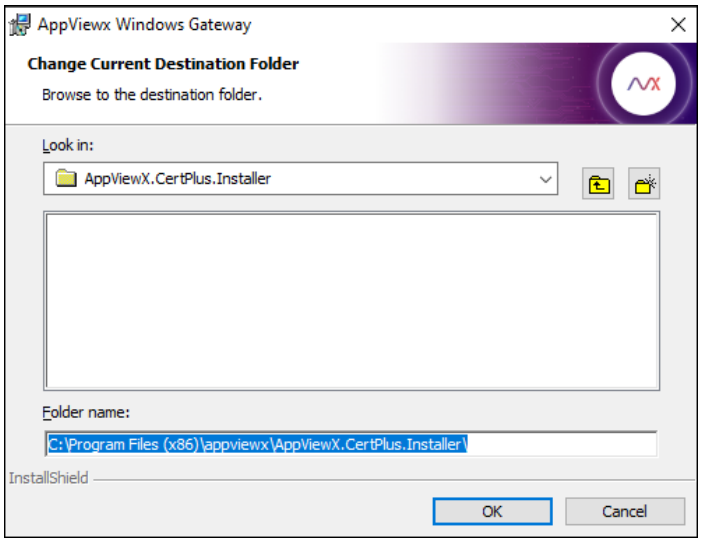
5. To install the AppViewX Windows Gateway at the default location, click **Next**.

   To change the default destination folder:

   a. Click **Change**.

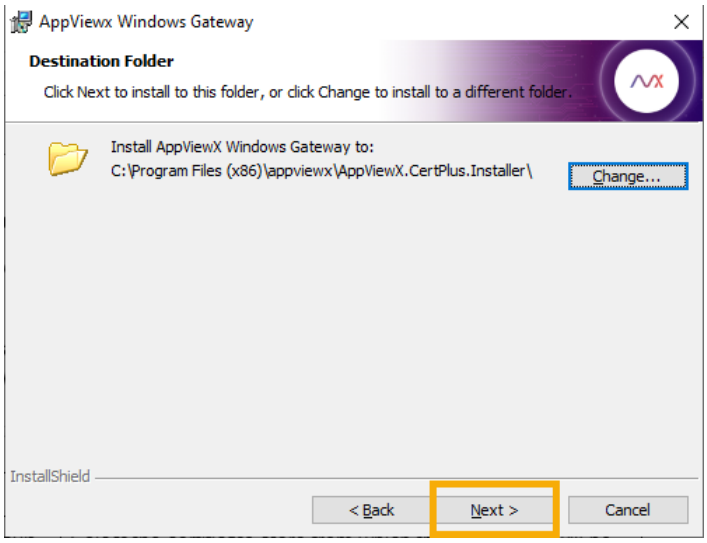


   b. On the **Change Current Destination Folder** screen, use the **Look in** dropdown list/  (up one level) icon/  (create new folder) icon to navigate to/create the required destination folder.

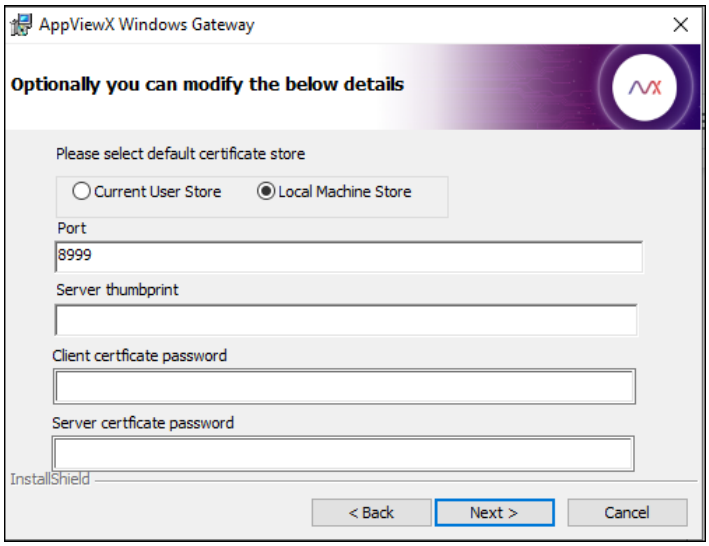


c. Click **OK**.

d. On the **Destination Folder** screen, click **Next**.



The **Optionally you can modify the below details** screen is displayed.



Enter the following details (optional):

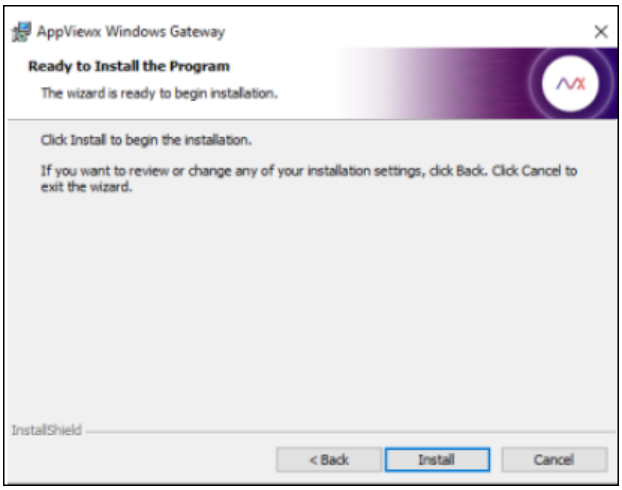
Field	Description
Please select default certificate store	Select the certificate store from which the certificates will be discovered and pushed to by AppViewX from the following options: • Current User Store This type of certificate store is local to a user account on a computer. It is located in the registry under the HKEY_CURRENT_USER root. • Local Machine Store (default) This type of certificate store is local to a computer and global to all the user accounts on the computer. It is located in the registry under the HKEY_LOCAL_MACHINE root. This configures the gateway for communicating with the appropriate certificate store.
Port	Port for accessing the service.

Field	Description
	Default value: 8999 (can be modified if required)
Server certificate thumbprint	If you are using a custom certificate, enter the corresponding server certificate thumbprint value.
Client certificate password	Password for accessing the client certificate For custom client certificates, enter the certificate password.
Server certificate password	Password for accessing the server certificate For custom server certificates, enter the certificate password.

Note: Read the **Before you Begin** section to use custom server and client certificates.

6. Click **Next**.

   The Ready to Install the Program screen is displayed.



7. Click **Install**.

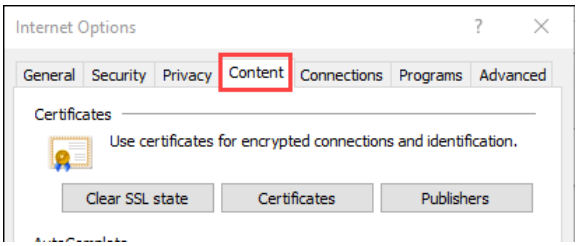
   This will:

   • Install the AppViewX Windows Gateway Troubleshooter tool

   • AppViewX Windows Gateway service.

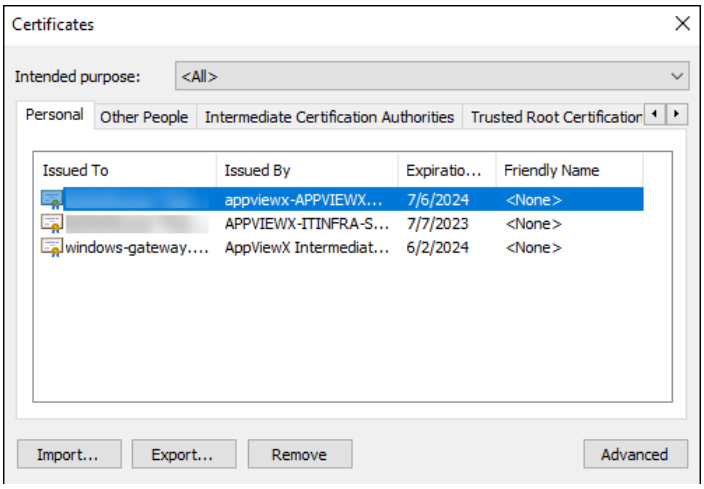
Step 4: Verifying the AppviewX Windows Gateway Installation

1. To verify the Windows AppViewX Gateway installation on Internet Explorer, import the client authentication certificate **ClientCertificateGateway.pfx**, from the download package (password: **appviewx**).
2. Navigate to Internet Explorer's **Settings** > **Internet Options**, and then click the **Content** tab.

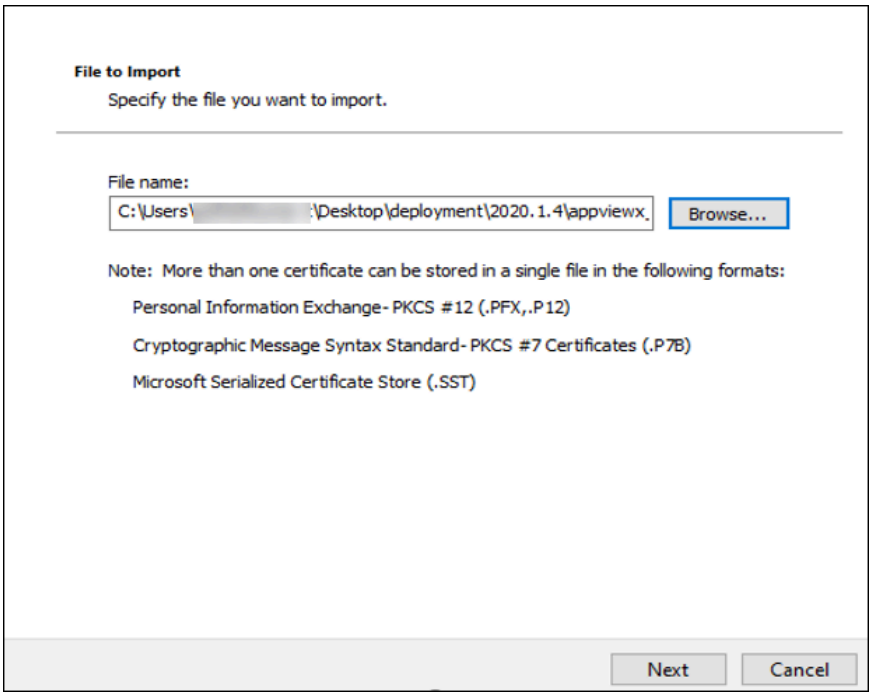


3. Click the **Certificates** button.

   The **Certificates** popup window opens.



4. Click the **Import** button on the **Certificates** page.



5. Go to the URL in the format: **https://hostname:portnumber/appviewx/rest/help**. For example: https://10.10.10.10:8999/appviewx/rest/help

   The page displayed confirms the accessibility and installation of the service.

Operations at https://localhost:8999/appviewx/rest

This page describes the service operations at this endpoint.

Uri	Method	Description
BindCertificateToGateway	POST	Service at https://localhost:8999/appviewx/rest/BindCertificateToGateway
BindCertificateToSite	POST	Service at https://localhost:8999/appviewx/rest/BindCertificateToSite
BindSQLServerCertificate	POST	Service at https://localhost:8999/appviewx/rest/BindSQLServerCertificate
BootPropertiesReader	POST	Service at https://localhost:8999/appviewx/rest/BootPropertiesReader
CertDeviceInfo	POST	Service at https://localhost:8999/appviewx/rest/CertDeviceInfo
CheckConnection	POST	Service at https://localhost:8999/appviewx/rest/CheckConnection
CreateAndSubmitRequest	POST	Service at https://localhost:8999/appviewx/rest/CreateAndSubmitRequest
CreateCSR	POST	Service at https://localhost:8999/appviewx/rest/CreateCSR
CreateCSRKey	POST	Service at https://localhost:8999/appviewx/rest/CreateCSRKey
DeleteFile	POST	Service at https://localhost:8999/appviewx/rest/DeleteFile
DeleteKeys	POST	Service at https://localhost:8999/appviewx/rest/DeleteKeys
DeviceInfo	POST	Service at https://localhost:8999/appviewx/rest/DeviceInfo
DiscoverCertificates	POST	Service at https://localhost:8999/appviewx/rest/DiscoverCertificates
DiscoverCertStoreCertificates	POST	Service at https://localhost:8999/appviewx/rest/DiscoverCertStoreCertificates
DiscoverFileCertificates	POST	Service at https://localhost:8999/appviewx/rest/DiscoverFileCertificates
DiscoverIBM	POST	Service at https://localhost:8999/appviewx/rest/DiscoverIBM
DiscoverKeys	POST	Service at https://localhost:8999/appviewx/rest/DiscoverKeys
ExecuteScriptInPowershell	POST	Service at https://localhost:8999/appviewx/rest/ExecuteScriptInPowershell
ExecuteWLSTScript	POST	Service at https://localhost:8999/appviewx/rest/ExecuteWLSTScript
ExtractCertificate	POST	Service at https://localhost:8999/appviewx/rest/ExtractCertificate
GetCertStores	POST	Service at https://localhost:8999/appviewx/rest/GetCertStores
LatestLog	POST	Service at https://localhost:8999/appviewx/rest/LatestLog
MicrosoftCAs	POST	Service at https://localhost:8999/appviewx/rest/MicrosoftCAs
MqConnector	POST	Service at https://localhost:8999/appviewx/rest/MqConnector
Ping	GET	Service at https://localhost:8999/appviewx/rest/Ping
PushAndBindCertificate	POST	Service at https://localhost:8999/appviewx/rest/PushAndBindCertificate
PushCertificate	POST	Service at https://localhost:8999/appviewx/rest/PushCertificate
PushDiscoveredCertificates	POST	Service at https://localhost:8999/appviewx/rest/PushDiscoveredCertificates
ReadFile	POST	Service at https://localhost:8999/appviewx/rest/ReadFile
ReadMultipleFiles	POST	Service at https://localhost:8999/appviewx/rest/ReadMultipleFiles
RemoveCertificateFromStore	POST	Service at https://localhost:8999/appviewx/rest/RemoveCertificateFromStore
RemoveSiteBinding	POST	Service at https://localhost:8999/appviewx/rest/RemoveSiteBinding
RevokeCertificate	POST	Service at https://localhost:8999/appviewx/rest/RevokeCertificate
SaveKeys	POST	Service at https://localhost:8999/appviewx/rest/SaveKeys

Note: In the event that a custom client authentication certificate is used, ensure that the CRL mentioned in the certificate is reachable from the AppViewX Windows Gateway hosting server.

Note: The steps to import the client certificate will differ depending on the web browser.

6. To register the AppViewX Windows Gateway with AppViewX, navigate to the AppViewX Cert+ (on the SaaS deployment) admin UI/UX, and then **Settings** > **Certificate**.

Note: To add the AppViewX Windows Gateway for

- Microsoft Enterprise CA integration, see ***Microsoft Enterprise CA*** section under chapter **CERT+ Setup** > **Configuring CA Settings** in Cert Admin guide.
- Microsoft Standalone CA integration, see ***Microsoft Standalone CA*** section under chapter **CERT+ Setup** > **Configuring CA Settings** in Cert Admin guide.
- Microsoft Device integrations, see ***Microsoft Devices Integration*** section under chapter **CERT+ Setup** in Cert Admin guide.

7. Register the gateway using the following URL format: **https://hostname:portnumber/appviewx**. For example: https://10.10.10.10:8999/appviewx

Note:

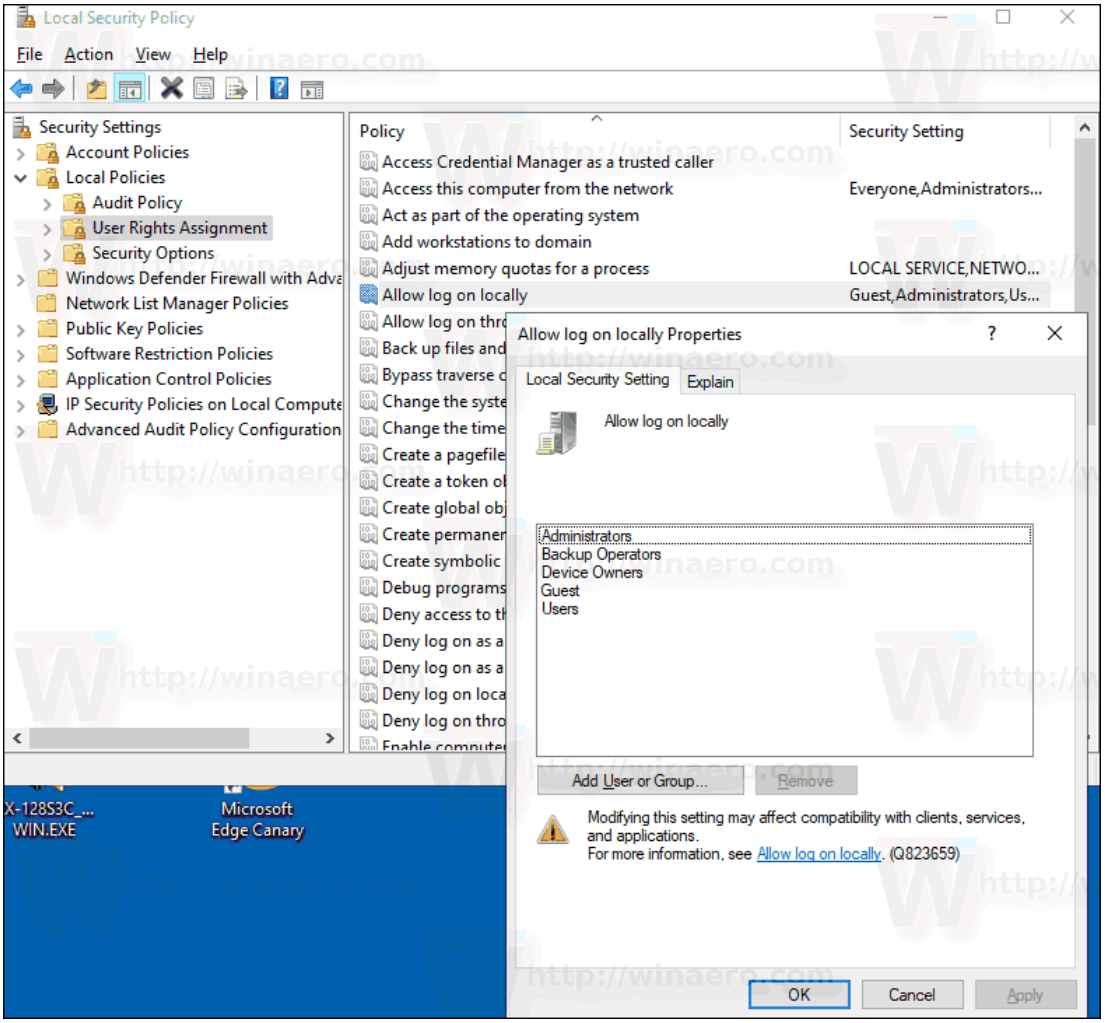
- The AppViewX's custom client authentication uses CRL and OCSP as proposed by Microsoft. If you choose to use Microsoft's client authentication then comment the config file as below:

```
<!--<serviceCredentials>
<clientCertificate>
<authentication certificateValidationMode="Custom"
customCertificateValidatorType="AppViewX.CertPlus.Service.CustomValidator,  AppViewX.CertPlus.Service" />
</clientCertificate>
</serviceCredentials>-->
```

- AppViewX recommends customers not to change this default authentication configuration provided by AppViewX.
- Refer Appendix A for the prerequisites for managing the Windows Server infrastructure and Appendix B for troubleshooting the target machine.

Step 5: Managing a Target Server

To manage a target server with different credentials, the user account can be configured using the AppViewX user interface. Enable the **Allow log on locally** user rights assignment security policy for the account.

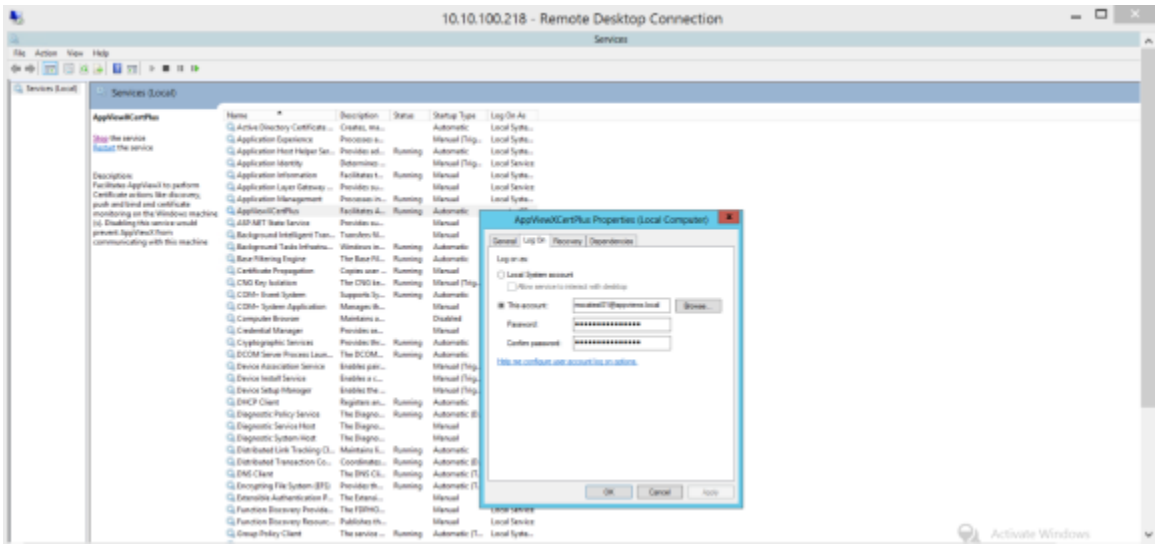


Non-Admin Service Account

- The AppViewX Windows Gateway can be installed using a service account that is part of the local administrator group account.
- In this case, the following command has to be executed from the PowerShell:

```
netsh http add urlacl url=https://+:8999/appviewx/user=Username@domainname
```

- In the above command, the value for `user = <domainserviceaccount>` and the `URL` must be changed respectively.
- On the Regedit path, "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\EventLog\Application\AppViewXCertPlus" add the service account and give Full Control permission.
- On the Installation path of the application, the user needs permission to read and write.
- If the network has a policy that the service account cannot be part of the administrator group or that the service account is only a part of the user group, then:
  - The AppViewX Windows Gateway is installed using an admin account.
  - It is then associated with the service account in **services.msc**, by adding the account in the properties of the AppViewXCertPlus service. Refer to the following image.



- Once this is done, stop and start the `AppViewXCertPlus Service` in `services.msc`.

Step 6: Disabling Current Operating System Information

On the https header, modify the registry.

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\HTTP\Parameters] "DisableServerHeader"=dword:00000002
```

Uninstalling the AppViewX Windows Gateway

Uninstalling AppViewX Windows Gateway involves the following steps:

1. Go to Windows control panel, select **Add or Remove program**.
2. Select **AppViewX.CertPlus.Installer**, and then click on **Uninstall** button.

Updating AppViewX Windows Gateway

Before updating the AppViewX Windows Gateway to a newer version, the old version of the AppViewX Windows Gateway should be uninstalled. Follow the instructions in Chapter 5 to uninstall AppViewX Windows Gateway.

After uninstalling the older version of AppViewX Windows Gateway, proceed with the installation of the new AppViewX Windows Gateway. Refer Chapter 2 for instructions on Installing the AppViewX Windows Gateway.

Appendix A

General Prerequisites

If a device that has the AppViewX Microsoft Gateway installed on it has to be managed in AppViewX, communication mode reset to WMI always.

Additional prerequisites that can be validated manually or by the AppViewX Windows Gateway Troubleshooting tool provided with the AppViewX Windows Gateway

Component	Description	Scripts
.Net Framework 4.5 and above	Download dotnet-framework-runtime from Microsoft software download center.	
POWERSHELL 4+	Download PowerShell from Microsoft software download center.	Powershell $PSVersionTable.PSVersion
Certadm.dll (Applicable ONLY if CA servers to be managed)	Check if dll is available in the C: WindowsSystem32 folder or install the Microsoft Remote Server Administration Tools (RSAT) for the respective	cd C:WindowsSystem32 and then dir certadm.dll

Additional prerequisites that can be validated manually or by the AppViewX Windows Gateway Troubleshooting tool provided with the AppViewX Windows Gateway (continued)

Component	Description	Scripts
	OS from Microsoft software download center.	
CertUtil	File will be available at the System32 folder.	Run certutil in the command prompt.
NetSH	Copy to the System32 folder if it is not available.	Run netsh in the command prompt.
RPC	Start the Remote procedure call in the services	net start RpcSs
WMI	Start the Windows Management Instrumentation in the services.	net start Winmgmt
WinRM	Start the Windows Remote Management.	net start WinRM
User Permission	When the users are added in the Group and the machine is not restarted a permission error will occur. Ensure that the machine is restarted when the user is added to a group.	Gwmi win32_groupuser -computer ptpll594|? {$_.groupcomponent -like '*""Administrators""'} \|select PartComponentnet localgroup administratorsCheck if user can access C$/windows/temp or admin$/Temp Local admin addition needs to restart.
	When the users are added in the Group and the machine is not restarted a permission error will occur. Ensure that the machine is restarted when the user is added to a group.	Gwmi win32_groupuser -computer ptpll594|? {$_.groupcomponent -like '*""Administrators""'} \|select PartComponent net localgroup administratorsCheck if user can access C$/windows/temp or admin$/Temp Local admin addition needs to restart.

Additional prerequisites that can be validated manually or by the AppViewX Windows Gateway Troubleshooting tool provided with the AppViewX Windows Gateway (continued)

Component	Description	Scripts
File Operations		Check if the user can access C $/windows/temp or admin$/temp. If you do not have c-drive then change the configuration to the available drive.
Port	Check if the port is already in use.	netstat -an |find ""8999" Check the Firewall outbound rules for the port Ping test from AppViewX Antivirus block for the port Turn off the local firewall Check the server, client, root, and intermediate certificates Check if the C: Logs folder exists and the permissions If you check in the Internet Explorer then the enhanced security must be disabled in the server role local server.
Powershell Remoting		Enter-PSSession -ComputerName <computername> -Credential <username>

Firewall Requirements

The firewall must not block the following ports:

Component	Port
Powershell	5985
WMI	135 + Dynamic port: 49152-65534
SMB	445
Native	135

Minimum Permissions Required for Communication

The AppViewX Windows Gateway agent communicates with the CAs via the following three communication modes:

- WMI
- Native API
- PowerShell

WMI

The WMI infrastructure is a Microsoft Windows operating system component known as the WMI service (winmgmt). The ability to obtain management data from remote computers is what makes WMI useful. Remote WMI connections are made through DCOM.

Recommended usage:

- WMI is enabled by default on many Windows servers.
- DCOM remains integrated into the Windows OS and is used by the Windows services to communicate, such as Microsoft Management Console certificate store.
- Organizations that prefer not to use PowerShell remoting or WinRM can use DCOM (WMI) as a communication method.
- It is commonly used in older Windows servers, such as Windows Server 2012 or 2008 R2.

Standard remote WMI queries use RPC to connect and RPC in turn uses a mess of ports. Initially, the Collector connects to the remote system via TCP port 135. The remote system then selects a high port and instructs the Collector to use this new port for subsequent communications. The high port depends on the OS but the current Windows OS uses ports 49152 to 65535.

PORTS USED: 445, 135 + dynamic port: 49152-65534

To use a static DCOM port for WMI in Windows, instead of numerous high ports, please follow the instructions on the Microsoft site for allocating a static port for WMI communication. Refer to Setting Up a Fixed Port for WMI | Microsoft Learn.

Important: Log on locally is needed to impersonate the user, to prevent execution of arbitrary scripts or commands on a remote Windows machine without proper authentication or authorization.

For the following use cases, this section lists the minimum permissions required for the AppViewX Windows Gateway to communicate with the CAs via WMI:

- Discovery
- Create CSR
- Create Certificate
- Create Certificate-Upload CSR
- Renew Certificate
- Revoke Certificate
- Certificate Push
- Certificate Bind

Discovery

Microsoft CA

Requirement	AppViewX Windows Gateway	Microsoft CA
User account type	Service account	Service account
User permission	NA	• Full control permission to **C:\Windows\Temp** • Read permission at CA level for the service account or the service account group

Microsoft CA (continued)

Requirement	AppViewX Windows Gateway	Microsoft CA
Services	WMI Service, certutil.exe command availability	WMI Service, certutil.exe command availability

IIS

Requirement	AppViewX Windows Gateway	IIS
User account type	Admin account	Admin account **Note:** The IIS administration tool can be handled only by local administrators, as local admin rights are typically required to register and run a Windows service that listens on HTTPS-based APIs for Certificate Management in the Windows Certificate Store or filesystem.
User permission		Full control permission to **C:\Windows\Temp**
Services	WMI Service	WMI Service

Microsoft PC

Requirement	AppViewX Windows Gateway	Microsoft PC
User account type	Admin account	Admin account **Note:** The Microsoft PC administration tool can be handled only

Microsoft PC (continued)

Requirement	AppViewX Windows Gateway	Microsoft PC
		by local administrators, as local admin rights are typically required to register and run a Windows service that listens on HTTPS-based APIs for Certificate Management in the Windows Certificate Store or filesystem.
User permission		Full control permission to **C:\Windows\Temp**
Services	WMI Service	WMI Service

Microsoft Server

Requirement	AppViewX Windows Gateway	Microsoft Server
User account type	Service account	Service account
User permission	NA	• Full control permission to **C:\Windows\Temp** • Read permission at CA level for the service account or the service account group
Services	WMI Service, certutil.exe command availability	WMI Service, certutil.exe command availability

Create CSR

IIS

Requirement	AppViewX Windows Gateway	IIS
User account type	Admin account	Admin account **Note:** The IIS administration tool can be handled only by local administrators, as local admin rights are typically required to register and run a Windows service that listens on HTTPS-based APIs for Certificate Management in the Windows Certificate Store or filesystem.
Services	WMI Service	WMI Service

Microsoft PC

Requirement	AppViewX Windows Gateway	Microsoft PC
User account type	Admin account	Admin account **Note:** The Microsoft PC administration tool can be handled only by local administrators, as local admin rights are typically required to register and run a Windows service that listens on HTTPS-based APIs for Certificate

Microsoft PC (continued)

Requirement	AppViewX Windows Gateway	Microsoft PC
		Management in the Windows Certificate Store or filesystem.
Services	WMI Service	WMI Service

Create Certificate

Microsoft CA

Requirement	AppViewX Windows Gateway	Microsoft CA
User account type	Service account	Service account
User permission	NA	• Request certificates permission at the CA level for the service account or the service account group or the authenticated users • Enroll permission at the certificate template level for the service account or the service account group or the authenticated users
Services	WMI Service, certutil.exe command availability	WMI Service, certutil.exe command availability

Create Certificate-Upload CSR

Microsoft CA

Requirement	AppViewX Windows Gateway	Microsoft CA
User account type	Service account	Service account

Microsoft CA (continued)

Requirement	AppViewX Windows Gateway	Microsoft CA
User permission	NA	• Request certificates permission at the CA level for the service account or the service account group or the authenticated users • Enroll permission at the certificate template level for the service account or the service account group or the authenticated users
Services	WMI Service, certutil.exe command availability	WMI Service, certutil.exe command availability

Renew Certificate

Microsoft CA

Requirement	AppViewX Windows Gateway	Microsoft CA
User account type	Service account	Service account
User permission	NA	• Request certificates permission at the CA level for the service account or the service account group or the authenticated users • Enroll permission at the certificate template level for the service account or the service account group or the authenticated users

Microsoft CA (continued)

Requirement	AppViewX Windows Gateway	Microsoft CA
Services	WMI Service, certutil.exe command availability	WMI Service, certutil.exe command availability

Revoke Certificate

Microsoft CA

Requirement	AppViewX Windows Gateway	Microsoft CA
User account type	Service account	Service account
User permission	NA	• Request certificates permission at the CA level for the service account or the service account group or the authenticated users • Enroll permission at the certificate template level for the service account or the service account group or the authenticated users
Services	WMI Service, certutil.exe command availability	WMI Service, certutil.exe command availability

Certificate Push

IIS

Requirement	AppViewX Windows Gateway	IIS
User account type	Admin account	Admin account **Note:** The IIS administration tool can be handled only by

IIS (continued)

Requirement	AppViewX Windows Gateway	IIS
		local administrators, as local admin rights are typically required to register and run a Windows service that listens on HTTPS-based APIs for Certificate Management in the Windows Certificate Store or filesystem.
Services	WMI Service	WMI Service

Microsoft PC

Requirement	AppViewX Windows Gateway	Microsoft PC
User account type	Admin account	Admin account **Note:** The Microsoft PC administration tool can be handled only by local administrators, as local admin rights are typically required to register and run a Windows service that listens on HTTPS-based APIs for Certificate Management in the Windows Certificate Store or filesystem.
Services	WMI Service	WMI Service

Microsoft Server

Requirement	AppViewX Windows Gateway	Microsoft Server
User account type	Service account	Service account
User permission	NA	• Full control permission to **C:\Windows\Temp** • Read permission at CA level for the service account or the service account group
Services	WMI Service	WMI Service

Certificate Bind

IIS

Requirement	AppViewX Windows Gateway	IIS
User account type	Admin account	Admin account **Note:** The IIS administration tool can be handled only by local administrators, as local admin rights are typically required to register and run a Windows service that listens on HTTPS-based APIs for Certificate Management in the Windows Certificate Store or filesystem.
Services	WMI Service	WMI Service

Native API

The Native API mode is only used at Microsoft CA communication. It uses the RPC based protocol for communication and sends a DCOM message.

Recommended usage:

- The customer retains control of the credentials since they enter them directly during local logon. Additionally, they have the option to use a managed service account with this approach.
- Because the libraries are built by Microsoft, communication occurs more quickly in the native mode.

PORTS USED: 135, 145

For the following use cases, this section lists the minimum permissions required for the AppViewX Windows Gateway to communicate with the CAs via Native API:

- All Operations
- Discovery
- Create Certificate
- Create Certificate-Upload CSR
- Renew Certificate
- Revoke Certificate

All Operations

Microsoft CA

Requirement	AppViewX Windows Gateway	Microsoft CA
User account type	Service account	Service account
User permission	NA	• Read, request certificates, and issue and manage certificates permission at the CA level for the service account/service account group /authenticated users • Enroll permission at the certificate template level for

Microsoft CA (continued)

Requirement	AppViewX Windows Gateway	Microsoft CA
		the service account/service account group/ authenticated users
Services	RPC Service	RPC Service, certutil.exe command availability

Discovery

Microsoft CA

Requirement	AppViewX Windows Gateway	Microsoft CA
User account type	Service account	Service account
User permission	NA	• Read permission at the CA level for the service account or the service account group
Services	RPC Service	RPC Service, certutil.exe command availability

Create Certificate

Microsoft CA

Requirement	AppViewX Windows Gateway	Microsoft CA
User account type	Service account	Service account
User permission	NA	• Request certificates permission at the CA level for the service account or the service account group or the authenticated users

Microsoft CA (continued)

Requirement	AppViewX Windows Gateway	Microsoft CA
		• Enroll permission at the certificate template level for the service account or the service account group or the authenticated users
Services	RPC Service	RPC Service, certutil.exe command availability

Create Certificate-Upload CSR

Microsoft CA

Requirement	AppViewX Windows Gateway	Microsoft CA
User account type	Service account	Service account
User permission	NA	• Request certificates permission at the CA level for the service account or the service account group or the authenticated users • Enroll permission at the certificate template level for the service account or the service account group or the authenticated users
Services	RPC Service	RPC Service

Renew Certificate

Microsoft CA

Requirement	AppViewX Windows Gateway	Microsoft CA
User account type	Service account	Service account
User permission	NA	• Request certificates permission at the CA level for the service account or the service account group or the authenticated users • Enroll permission at the certificate template level for the service account or the service account group or the authenticated users
Services	RPC Service	RPC Service, certutil.exe command availability

Revoke Certificate

Microsoft CA

Requirement	AppViewX Windows Gateway	Microsoft CA
User account type	Service account	Service account
User permission	NA	Issue and manage certificates permission at CA level for the service account or the service account group or the authenticated users
Services	RPC Service	RPC Service,certutil.exe command availability

PowerShell

Overview

PowerShell Remoting is a built-in feature in Windows hosts that allows administrators to connect to remote hosts to execute scripts and PowerShell commands. Administrators need to enable PowerShell Remoting on the target machine for smooth communication. It is a powerful tool for efficiently and securely managing remote systems.

Recommended Usage:

- WinRM is a more modern protocol used by many organizations.
- Ports 5985 are used in WinRM and WinRM v2.
- All WinRM data is encrypted using "Integrated Windows Authentication," preferably set to Kerberos authentication on the host machine.
- Essentially, WinRM is an HTTP-based API, and the data returned is in XML format rather than objects, as WinRM is XML-based.

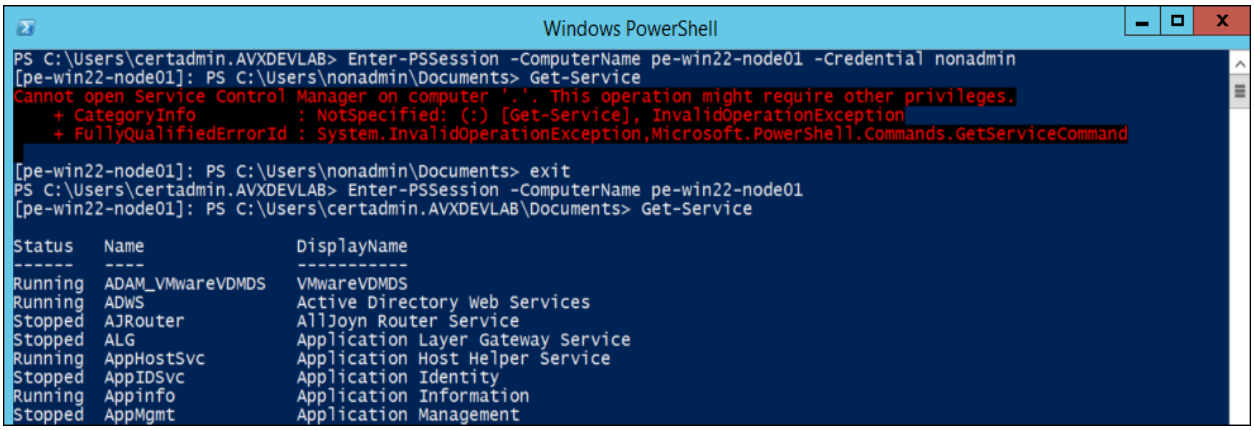
PORTS USED: 5985

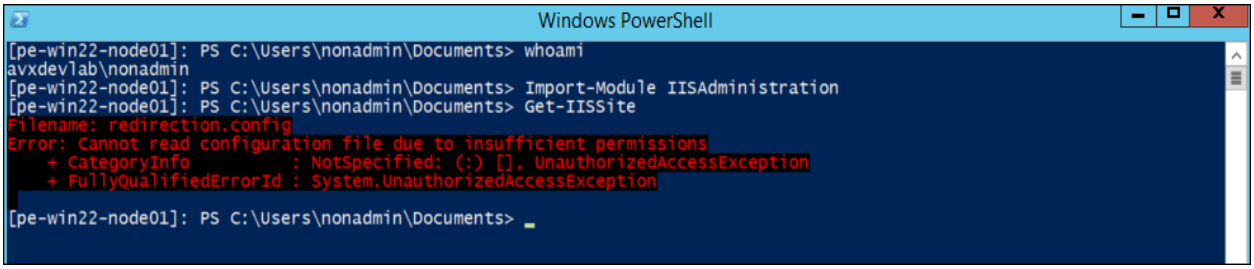
Note: SMB port number **445** will be used for any file transfer from the gateway machine to the remote machine, including for push certficate.

Justifying Admin Access

To create remote sessions and run remote commands, the current user must, by default, be a member of the Administrators group on the remote computer or provide administrator credentials. Otherwise, the command will fail.

PowerShell remoting can be enabled for a standard user, but they need administrative privileges to manage IIS and the Local Computer Store. In the screenshot, a non-admin user faces privilege restrictions when listing system services, whereas an admin user can view the services without issue.





Certificate Store Access

- **Admin Privileges for Certificate Management:** Managing certificates in system-wide locations, such as the Local Machine certificate store, requires admin privileges. This includes tasks like installing, renewing, revoking, and deleting certificates.
- **Service Configuration:** Administrative access is necessary for configuring services to use specific system certificates for secure communication. This includes modifying service configurations, such as those for SQL Server.
- **Private Key Management:** Admin privileges are needed to manage private keys associated with certificates. This includes importing/exporting certificates with their private keys, configuring key access permissions, and configuring key archival and recovery policies.
- **MS Certificate Authority (CA) Operations:** Tasks related to managing a Certificate Authority (CA), such as configuring CA settings, issuing and revoking certificates, and managing certificate templates, often require admin privileges.
- **System Integrity:** Certificate management is crucial for system integrity and security. Admin privileges ensure that only authorized users can manage certificates, reducing the risk of unauthorized access, tampering, or misuse.
- **Security:** PowerShell remoting allows users to execute commands and scripts on remote computers. Admin access ensures that only users with sufficient privileges can perform potentially sensitive or impactful actions on remote systems.

- **System Management:** Many administrative tasks, such as IIS administration and accessing certain registry keys, require administrative privileges. PowerShell remoting enables administrators to perform these tasks remotely, but admin access is necessary to execute the required commands successfully.
- **Resource Control:** Admin access ensures that users have the necessary permissions to access and modify system resources, such as files, directories, and registry keys, on remote machines. This level of access is often required for effective system management and troubleshooting.
- **Configuration Management:** PowerShell remoting is commonly used in configuration management and automation scenarios. Administrators need admin access to deploy configurations, install updates, and perform other management tasks remotely.

To learn about the system and configuration requirements for running remote commands in PowerShell, refer to the "about Remote Requirements" section on the Microsoft documentation website.

For the following use cases, this section lists the minimum permissions required for the AppViewX Windows Gateway to communicate with the CAs via PowerShell:

> **Important:** Log on locally is needed to impersonate the user, to prevent execution of arbitrary scripts or commands on a remote Windows machine without proper authentication or authorization.

- Discovery
- Create CSR
- Create Certificate
- Create Certificate-Upload CSR
- Renew Certificate
- Revoke Certificate
- Certificate Push
- Certificate Bind

Discovery

Microsoft CA

Requirement	AppViewX Windows Gateway	Microsoft CA
User account type	Service account	Service account

Microsoft CA (continued)

Requirement	AppViewX Windows Gateway	Microsoft CA
User permission	NA	• Full control permission to **C:\Windows\Temp** • Read permission at CA level for the service account or the service account group
Services	RPC Service,WinRM Service,WinRM Configuration, Powershell remoting,certutil.exe command availability	RPC Service,WinRM Service,WinRM Configuration, Powershell remoting,certutil.exe command availability

IIS

Requirement	AppViewX Windows Gateway	IIS
User account type	Admin account	Admin account
Services	RPC Service,WinRM Service,WinRM Configuration, Powershell remoting	RPC Service,WinRM Service,WinRM Configuration, Powershell remoting

Microsoft PC

Requirement	AppViewX Windows Gateway	Microsoft PC
User account type	Admin account	Admin account
Services	RPC Service,WinRM Service,WinRM Configuration, Powershell remoting`	RPC Service,WinRM Service,WinRM Configuration, Powershell remoting

Microsoft Server

Requirement	AppViewX Windows Gateway	Microsoft Server
User account type	Admin account	Admin account  **Note:** The Microsoft Server administration

Microsoft Server (continued)

Requirement	AppViewX Windows Gateway	Microsoft Server
		tool can be handled only by local administrators, as local admin rights are typically required to register and run a Windows service that listens on HTTPS-based APIs for Certificate Management in the Windows Certificate Store or filesystem.
User permission	NA	Read permission for the folder to be discovered
Services	RPC Service,WinRM Service,WinRM Configuration, Powershell remoting	RPC Service,WinRM Service,WinRM Configuration, Powershell remoting

Create CSR

IIS

Requirement	AppViewX Windows Gateway	IIS
User account type	Admin account	Admin account **Note:** The IIS administration tool can be handled only by local administrators, as local admin rights are typically required to register and run a Windows service that listens on HTTPS-based

IIS (continued)

Requirement	AppViewX Windows Gateway	IIS
		APIs for Certificate Management in the Windows Certificate Store or filesystem.
Services	RPC Service,WinRM Service,WinRM Configuration, Powershell remoting	RPC Service,WinRM Service,WinRM Configuration, Powershell remoting

Microsoft PC

Requirement	AppViewX Windows Gateway	Microsoft PC
User account type	Admin account	Admin account **Note:** The Microsoft PC administration tool can be handled only by local administrators, as local admin rights are typically required to register and run a Windows service that listens on HTTPS-based APIs for Certificate Management in the Windows Certificate Store or filesystem.
Services	RPC Service,WinRM Service,WinRM Configuration, Powershell remoting`	RPC Service,WinRM Service,WinRM Configuration, Powershell remoting

Create Certificate

Microsoft CA

Requirement	AppViewX Windows Gateway	Microsoft CA
User account type	Service account	Service account
User permission	NA	• Request certificates permission at CA level for the service account/service account group/ authenticated users • Enroll permission at the certificate template level for the service account/service account group/ authenticated users
Services	RPC Service,WinRM Service,WinRM Configuration, Powershell remoting,certutil.exe command availability	RPC Service,WinRM Service,WinRM Configuration, Powershell remoting,certutil.exe command availability

Create Certificate-Upload CSR

Microsoft CA

Requirement	AppViewX Windows Gateway	Microsoft CA
User account type	Service account	Service account
User permission	NA	• Request certificates permission at CA level for the service account/service account group/ authenticated users • Enroll permission at the certificate template level for

Microsoft CA (continued)

Requirement	AppViewX Windows Gateway	Microsoft CA
		the service account/service account group/ authenticated users
Services	RPC Service,WinRM Service,WinRM Configuration, Powershell remoting,certutil.exe command availability	RPC Service,WinRM Service,WinRM Configuration, Powershell remoting,certutil.exe command availability

Renew Certificate

Microsoft CA

Requirement	AppViewX Windows Gateway	Microsoft CA
User account type	Service account	Service account
User permission	NA	• Request certificates permission at CA level for the service account/service account group/ authenticated users • Enroll permission at the certificate template level for the service account/service account group/ authenticated users
Services	RPC Service,WinRM Service,WinRM Configuration, Powershell remoting,certutil.exe command availability	RPC Service,WinRM Service,WinRM Configuration, Powershell remoting,certutil.exe command availability

Revoke Certificate

Microsoft CA

Requirement	AppViewX Windows Gateway	Microsoft CA
User account type	Service account	Service account
User permission	NA	Issue and manage certificates permission at CA level for the service account or the service account group or the authenticated users
Services	RPC Service,WinRM Service,WinRM Configuration, PowerShell remoting,certutil.exe command availability	RPC Service,WinRM Service,WinRM Configuration, PowerShell remoting,certutil.exe command availability

Certificate Push

IIS

Requirement	AppViewX Windows Gateway	IIS
User account type	Admin account	Admin account **Note:** The IIS administration tool can be handled only by local administrators, as local admin rights are typically required to register and run a Windows service that listens on HTTPS-based APIs for Certificate Management in the Windows Certificate Store or filesystem.

IIS (continued)

Requirement	AppViewX Windows Gateway	IIS
User permission	NA	Full control permission to **C:\Windows\Temp**
Services	RPC Service,WinRM Service,WinRM Configuration, Powershell remoting	RPC Service,WinRM Service,WinRM Configuration, Powershell remoting

Microsoft PC

Requirement	AppViewX Windows Gateway	Microsoft PC
User account type	Admin account	Admin account **Note:** The Microsoft PC administration tool can be handled only by local administrators, as local admin rights are typically required to register and run a Windows service that listens on HTTPS-based APIs for Certificate Management in the Windows Certificate Store or filesystem.
Services	RPC Service,WinRM Service,WinRM Configuration, Powershell remoting`	RPC Service,WinRM Service,WinRM Configuration, Powershell remoting

Microsoft Server

Requirement	AppViewX Windows Gateway	Microsoft Server
User account type	Admin account	Admin account

Microsoft Server (continued)

Requirement	AppViewX Windows Gateway	Microsoft Server
		Note: The Microsoft Server administration tool can be handled only by local administrators, as local admin rights are typically required to register and run a Windows service that listens on HTTPS-based APIs for Certificate Management in the Windows Certificate Store or filesystem.
User permission	NA	Write permission for the folder to be discovered
Services	RPC Service,WinRM Service,WinRM Configuration, Powershell remoting`	RPC Service,WinRM Service,WinRM Configuration, Powershell remoting

Certificate Bind

IIS

Requirement	AppViewX Windows Gateway	IIS
User account type	Admin account	Admin account **Note:** The IIS administration tool can be handled only by local administrators, as local admin rights are typically required

IIS (continued)

Requirement	AppViewX Windows Gateway	IIS
		to register and run a Windows service that listens on HTTPS-based APIs for Certificate Management in the Windows Certificate Store or filesystem.
Services	RPC Service,WinRM Service,WinRM Configuration, Powershell remoting	RPC Service,WinRM Service,WinRM Configuration, Powershell remoting

Appendix B

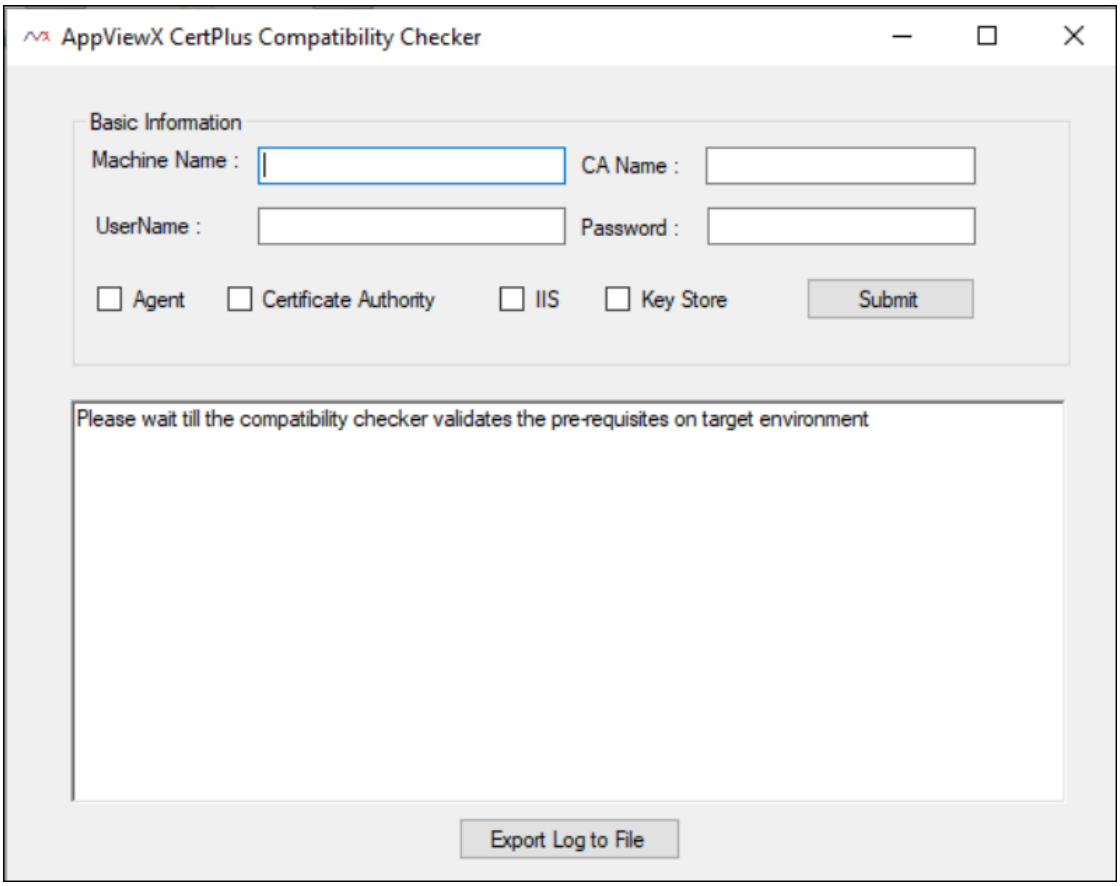
AppViewX Windows Gateway Troubleshooting Tool

The AppViewX Troubleshooting tool is used to analyze the accessibility of the target machine, to which the AppViewX communicates.

Accessing the Validator

To launch the validator:

From the Windows **Start** menu, execute the **AppViewX.CertPlus.Validator.exe** file.
The **AppViewX CertPlus Compatibility Checker** screen is displayed.



Validating the Target Machine

1. On the **AppViewX CertPlus Compatibility Checker** screen:

   a. Enter the **Basic Information** required.

Field descriptions for the Basic Information section

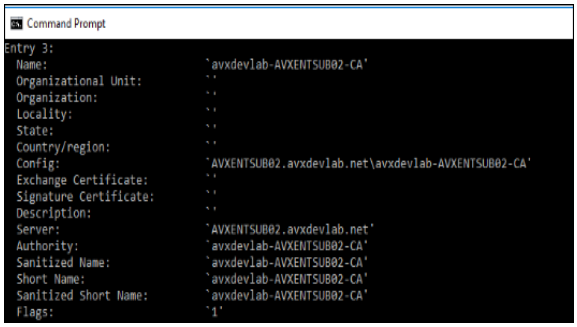
Name	Description	Condition
Machine Name	Enter the hostname of the target machine for validation.	Mandatory field.
CA Name	Enter the name of the Certificate Authority from the CA Config.	Mandatory only when the **Certificate Authority** option (explained in the next step) is selected .

Name	Description	Condition
User Name	Enter the username for accessing the target machine.	Mandatory field Format: username@domainname
Password	Enter the password for accessing the target machine.	Mandatory field

b. From the following choices, select one or more options as required:

Descriptions of the options

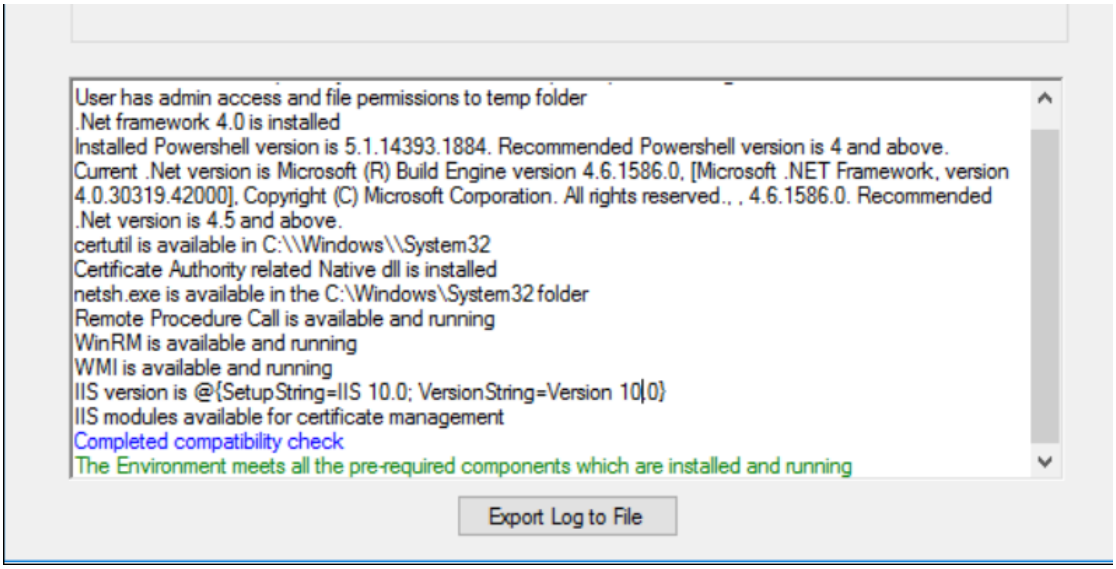
Option	Description
Agent	To install the AppViewX Windows Gateway in the target machine, select this option. This will validate the prerequisites required for the installation. The machine name will be entered in the **Machine Name** field.
Certificate Authority	To validate the Certificate Authority-related functionality, select this option. The CA Name is mandatory only in this case. Use the `certutil -dump` command in a cmd window to get the CA Name. In the output, the value for **Server** is the Machine Name and the value for **Name** is the CA Name. In the sample screenshot shown below, the machine name is **AVXENTSUB02.avxdevlab.net** and the CA Name is **avxdevlab-AVXENTSUB02-CA**.

Option	Description
	
IIS	To validate the IIS-sites related functionality, select this option.
Key Store	To validate only the Microsoft Certificate-store related functionality, select this option.

2. Click **Submit**.

Note: Mandatory fields (from **Machine Name**, **CA Name**, **UserName**, and **Password**) that have been missed will be highlighted in red after you click **Submit**.

The validation summary is displayed in the text box below the **Basic Information** section, as shown in the image below:



Color coding followed in the validation summary

Color	Indicates
Black	Success information and output
Red	An error or warning
Blue	Completion of the validation process
Green	Successful completion of the process

3. To export the validation summary as a log file, click **Export Log to File**.

   Following are the validations performed by the AppViewX Windows Troubleshooting tool:

Validations performed by the AppViewX Windows Troubleshooting tool

Validate	Description	Agent	CA	IIS	Keystore
User	The validator will connect to the target machine with the username and password specified, and check if the target machine has admin access.	Yes	Yes	Yes	Yes
.Net framework	The validator will check if .Net framework version 4.5.2+ is installed. It will also display the current version installed.	Yes	Yes	Yes	Yes
PowerShell	The validator will check if PowerShell is installed. It will also display the current version of PowerShell installed.	Yes	Yes	Yes	Yes
CertUtil	The validator will check if the certutil	Yes	Yes	No	No

Validate	Description	Agent	CA	IIS	Keystore
	component is available. The certutil component is used to retrieve the CA name and the corresponding templates.				
Certadm.dll	The validator will check if this component, a native component to access the CA, is available in the **C:\Windows\System32** folder. Sometimes, while trying to access this component during verification, it will return an error. Therefore, a manual check must be performed.	Yes	Yes	No	No
netsh.exe	This is used to bind the certificate to the installed agent port (8999).	Yes	No	No	No
RPC	The validator will check if the Remote Procedure Call (RPC) service is installed and running on the target machine. This service should be running to perform all remote operations.	Yes	Yes	Yes	Yes

Validate	Description	Agent	CA	IIS	Keystore
WinRM	The validator will check if the Windows Remote Management service is installed and running on the target machine. This service is required for the PowerShell execution.	Yes	Yes	Yes	Yes
WMI	The validator will check if the Windows Management Instrumentation service is installed and running on the target machine. This service is required for the WMI and PowerShell execution.	Yes	Yes	Yes	Yes
IIS	The validator will check if the IIS server is installed and, if yes, the current IIS version.	No	No	Yes	No

Chapter 6: Support

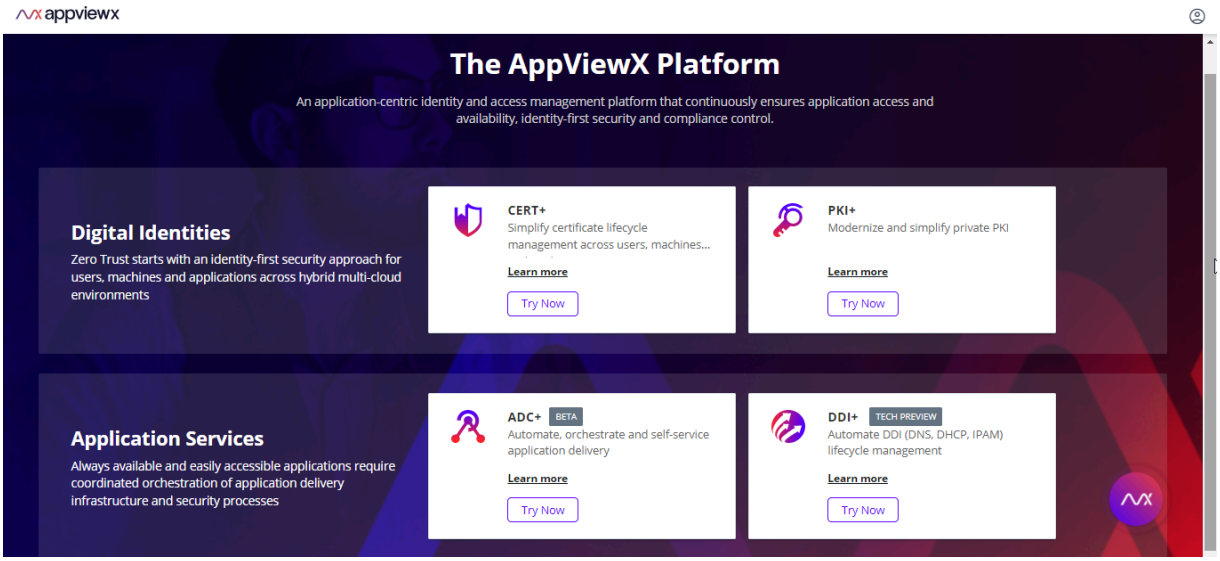
AppViewX's Customer Success team is dedicated to help you with the workings of AppViewX's SaaS-based product line. We have introduced the AppViewX Chatbot, an in-product support interface to help you make your queries specific and, therefore, enable AppViewX's support teams to facilitate expedited solutions. You can use the chatbot to request a demo, a trial extension, a subscription upgrade, or for a query resolution.

Using the AppViewX Chatbot

To access the chatbot:

1. Log in to your SaaS account.

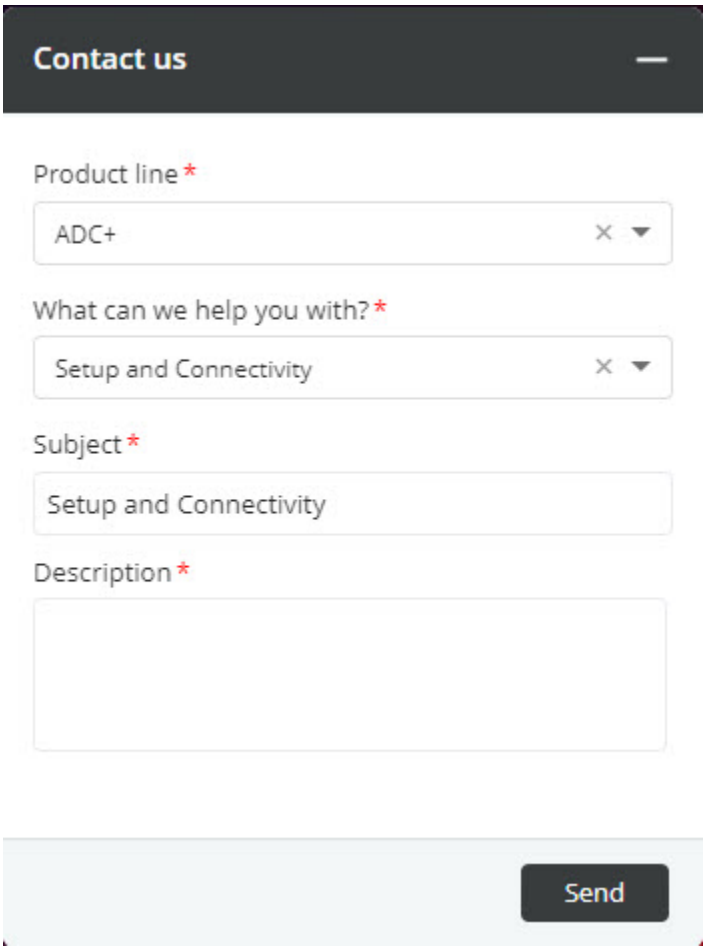
   The AppViewX Platform landing page is displayed.



2. To access the AppViewX chatbot, click  from the bottom-right corner of the screen.

   Note: This chatbot icon is available on all product screens, enabling you to send a request at any point during a process.

   The **Contact Us** pop-up window is displayed.



3. In accordance to your query, enter the following details:

Field	Description
Product line*	From the dropdown list, select one from the following product line options: • CERT+ • ADC+ • PKI+
What can we help you with?*	From the dropdown list, select a category closest to your requirement. The categories in this list include:

Field	Description
	• Setup and Connectivity • Download/Installation • Artifacts/Solution Guides • System Impaired • Request for upgrade • Request for trial extension • Critical • Others
Subject*	This field is automatically updated with the category you selected in the **What can we help you with?** Field. This field is editable, so you can change the subject line if it helps to better explain your query.
Description*	In this field, enter the details of your requirement. **Note:** This field has a character limit of 255 characters.

4. Click **Send**.
   Depending on the category selected, in the **What can we help you with?** field, the relevant AppViewX support team will get in touch with you.

Note: You can also reach out to our teams using the following details:
- salesops@appviewx.com
- saashelp@appviewx.com.
- Phone

- +1 212 390 1644
- +1 206 207 7541

Chapter 7: Glossary

An explanation of the terms used in this guide:

Term	Description
SaaS	Software as a Service
EKS	Elastic Kubernetes Service
TLS	Transport Layer Security
AES	Advanced Encryption Standard
AZ	Availability Zone
VPN	Virtual Private Network
VPC	Virtual Private Cloud
EC2	Elastic compute
AWS	Amazon Web services
HA	High Availability
DR	Disaster Recovery
mTLS	Mutual TLS Authentication