Managed Kubernetes - AppViewX Stack Install and Upgrade on EKS

Version: 2022.1.0

Copyright AppViewX, Inc.

Trademarks

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).

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
v1.0	Managed Kubernetes - AppViewX Stack Install and Upgrade on EKS	December 2022

About this Guide

The guide provides information about the AppViewX stack; procedure for installing and accessing the application.

Audience

The guide is for the users who want to install the AppViewX stack on managed EKS, specifically

- Platform engineers
- Implementation specialist
- Kubernetes administrators

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: EKS Components

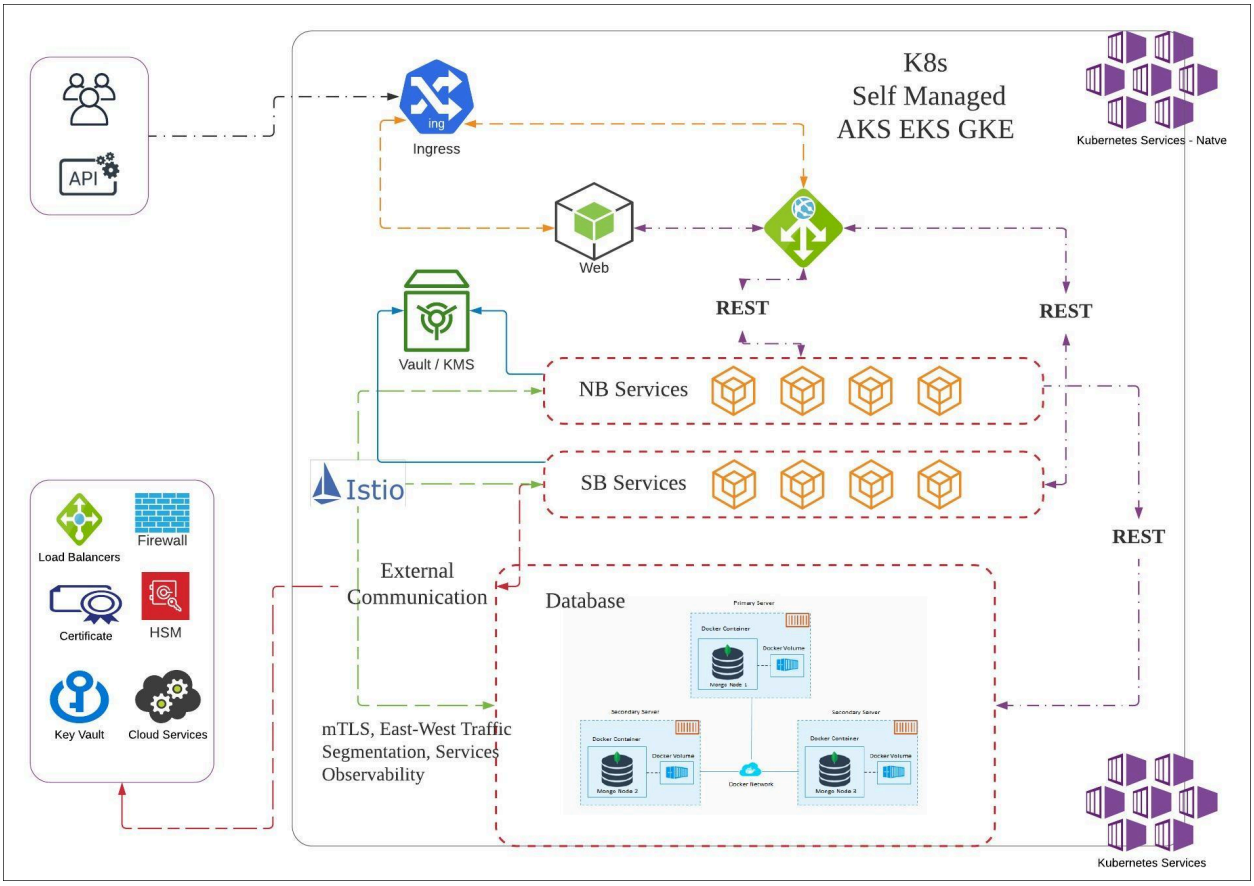
The following EKS components are utilized by AppViewX:

- Storage bucket for storing mongodb and vault backups
- Amazon Kubernetes engine

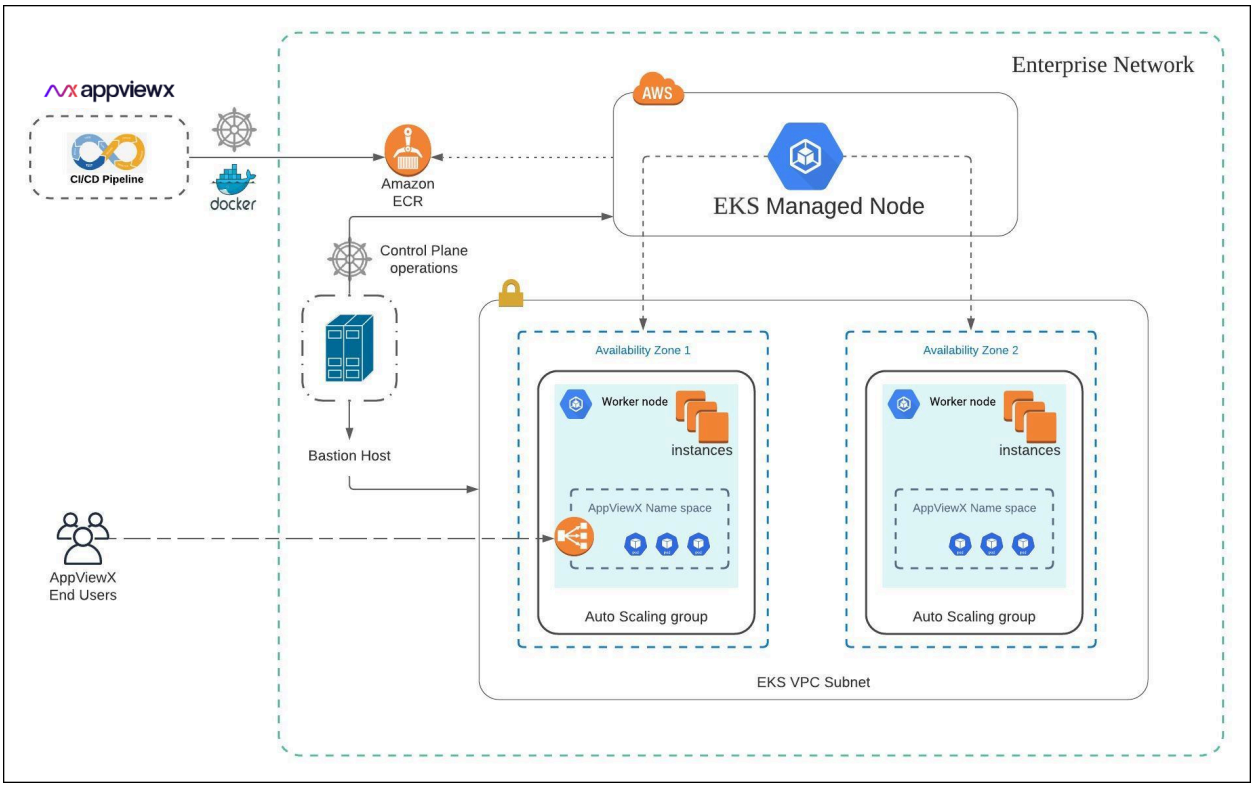
Chapter 2: Deployment Architecture

- Kubernetes Architecture
- EKS Deployment Model
- Cloud Connector

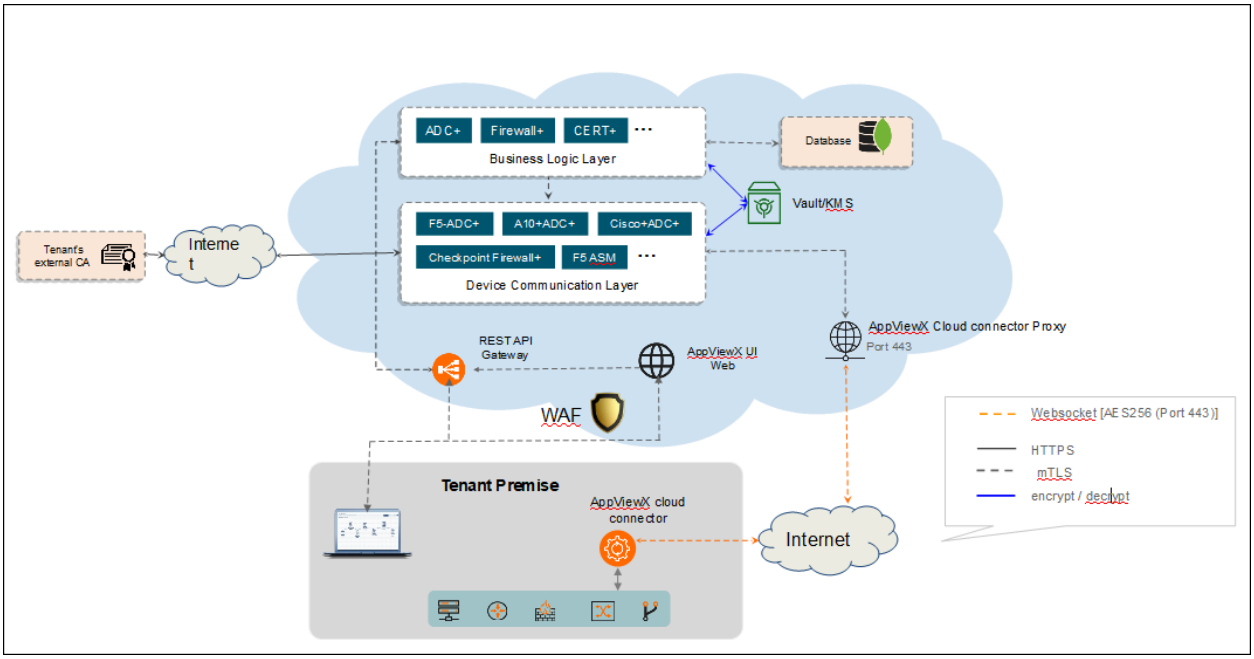
Kubernetes Architecture



EKS Deployment Model



Cloud Connector



For more details on cloud connectors refer to AppViewX Cloud Connector User Guide

Chapter 3: Prerequisites

- Bastion Host Setup
- EKS Cluster
- AWS S3 Bucket
- Configuring CSI

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: 1.22
- 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
- MongoDB backup
- Vault backup

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 k8s 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

Note: Provide the EKS Cluster Name and the Region name where EKS cluster is being executed.

```
# Provide Inputs
clusterName=<eksClusterName>
region=<awsRegionName>


bucketName="$clusterName-avx-jar-backup"
roleName=$bucketName-"role"
csiRoleName=$clusterName-"csi-role"
csiPolicyName=$clusterName-"csi-policy"
policyName=$bucketName-"policy"
aws s3api create-bucket --bucket "${bucketName}" --region "${region}" --create-bucket-configuration LocationConstraint="${region}"
```

```
oidcUrl=$(aws eks describe-cluster --name $clusterName --region $region | jq -r '.cluster.identity.oidc.issuer' | sed 's/https:\/\///g')

# Create policy for accessing s3 bucket

policyDocument='{

   "Version": "2012-10-17",

   "Statement": [

      {

         "Sid": "VisualEditor0",

         "Effect": "Allow",

         "Action": [

            "s3:PutObject",

            "s3:GetObject"

         ],

         "Resource": "arn:aws:s3:::'$bucketName'/*"

      }

   ]

}'


policyArn=$(aws iam create-policy --policy-name $policyName --policy-document "${policyDocument}" | jq -r '.Policy.Arn')


assumePolicyDocument='{

   "Version": "2012-10-17",

   "Statement": [

      {

         "Effect": "Allow",

         "Principal": {

            "Federated": "arn:aws:iam::291657924554:oidc-provider/'$oidcUrl'"

         },

         "Action": "sts:AssumeRoleWithWebIdentity",

         "Condition": {

            "StringEquals": {

               "'$oidcUrl':aud": "sts.amazonaws.com"

            }

         }

      }

   ]

}'
```

```
roleArn=$(aws iam create-role --role-name $roleName --assume-role-policy-document "${assumePolicyDocument}" | jq -r '.Role.Arn')

aws iam attach-role-policy --policy-arn $policyArn --role-name $roleName

# Create the IAM role.
awsEbsCsiDriverTrustPolicy='{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam:291657924554::oidc-provider/$oidcUrl"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "$oidcUrl:aud": "sts.amazonaws.com",
          "$oidcUrl:sub": "system:serviceaccount:kube-system:ebs-csi-controller-sa"
        }
      }
    }
  ]
}'

aws iam create-role --role-name $csiRoleName --assume-role-policy-document "$awsEbsCsiDriverTrustPolicy"

# Attach the required AWS managed policy to the role
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy \
  --role-name $csiRoleName

# Annotate the ebs-csi-controller-sa Kubernetes service account with the ARN of the IAM role.
kubectl annotate serviceaccount ebs-csi-controller-sa -n kube-system \
    eks.amazonaws.com/role-arn=arn:aws:iam:291657924554::role/$csiRoleName

# Restart the ebs-csi-controller deployment for the annotation to take effect.
kubectl rollout restart deployment ebs-csi-controller -n kube-system
```

```
kubectl apply -f - <<EOF
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gp3
provisioner: ebs.csi.aws.com
parameters:
  type: gp3
reclaimPolicy: Retain
volumeBindingMode: WaitForFirstConsumer
EOF


echo ""
echo "Annotation": $roleArn
```

After the script is executed,

- Capture the output **Annotation** which is required in the global utility config.
- 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

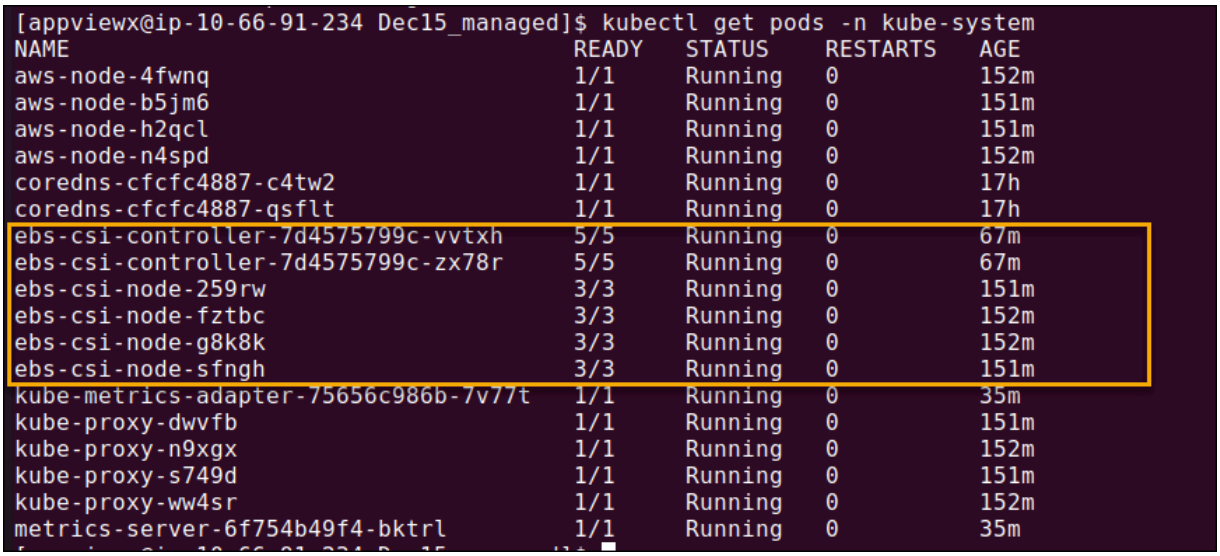
```
helm repo update
```

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
```



```
[appviewx@ip-10-66-91-234 Dec15_managed]$ kubectl get pods -n kube-system
NAME                                  READY   STATUS    RESTARTS   AGE
aws-node-4fwnq                        1/1     Running   0          152m
aws-node-b5jm6                        1/1     Running   0          151m
aws-node-h2qcl                        1/1     Running   0          151m
aws-node-n4spd                        1/1     Running   0          152m
coredns-cfcfc4887-c4tw2               1/1     Running   0          17h
coredns-cfcfc4887-qsflt               1/1     Running   0          17h
ebs-csi-controller-7d4575799c-vvtxh   5/5     Running   0          67m
ebs-csi-controller-7d4575799c-zx78r   5/5     Running   0          67m
ebs-csi-node-259rw                    3/3     Running   0          151m
ebs-csi-node-fztbc                    3/3     Running   0          152m
ebs-csi-node-g8k8k                    3/3     Running   0          152m
ebs-csi-node-sfngh                    3/3     Running   0          151m
kube-metrics-adapter-75656c986b-7v77t 1/1     Running   0          35m
kube-proxy-dwvfb                      1/1     Running   0          151m
kube-proxy-n9xgx                      1/1     Running   0          152m
kube-proxy-s749d                      1/1     Running   0          151m
kube-proxy-ww4sr                      1/1     Running   0          152m
metrics-server-6f754b49f4-bktrl       1/1     Running   0          35m
```

5. Refer to the Amazon EKS User Guide - Create CSI IAM Role, and click the AWS CLI tab within the content to create your Amazon EBS CSI plugin IAM role with the AWS CLI.

> **Attention:** Ignore Step 4 from the content in the AWS CLI tab.

Chapter 4: Install AppViewX in Managed Kubernetes

Migration Strategy

To migrate from AppViewX legacy versions (2022.1.0, 2021.1.0, and 2020.3.0) to Managed Kubernetes, take a backup of the mongo and vault in the respective legacy versions.

> **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 (link to be shared post release).
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:

Table 1.

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 to whether third party monitoring components such as ELK needs to be installed. (True/False)
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*: 2022.1.0_FP_750-alpine

Parameters	Description of Values
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: ""
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*

Parameters	Description of Values

	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 usecases 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

Parameters	Description of Values
	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

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. • **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

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: Let the installation proceed to completion until 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 <path to the mongo backup tar file>

./vault_restore.sh -p <path to the vault backup file>
```

Note: 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 follows:

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. Run the command **./appviewxctl installtpt --config .appviewxctl.yaml**.

Post-Installation Steps

INGRESS_HOSTS and INGRESS_PORT variables should be added in **appviewx.properties** which is embedded in **avx-common-config** config map for getting it exposed inside the containers.

To add these properties manually in the config map, follow the steps below:

1. Take a backup of current config map using the command

```
kubectl get cm avx-common-config -n avx -o yaml > avx-common-config.yaml
```

2. Execute the command:

```
kubectl get svc -n istio-system istio-ingressgateway
```

   Make a note of the **EXTERNAL-IP** of the Load-balancer. A sample output is displayed below.

```
NAME                   TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)                      AGE
istio-ingressgateway   LoadBalancer   10.76.14.241   35.200.143.48   80:31540/TCP,443:32056/TCP   7h
```

   In the sample output shared the EXTERNALIP is "35.200.143.48"

3. Edit the config map and add the parameters at EOF. Use the command below.

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

   Add the above parameters after the DATACENTER_LIST variable like below

```
\nDATACENTER_LIST=absecon\nINGRESS_HOSTS=<LB_URL>\nINGRESS_PORT=443
```

Note: Replace <LB_URL> with the EXTERNAL-IP fetched in step 2 above.

4. After saving the config map, restart all the pods which are prefixed with **avx**. Use the command.

```
kubectl delete pod -n avx --force $(kubectl get pods -n avx | grep avx| '{print $1}')
```

Chapter 5: Upgrade AppViewX in Managed Kubernetes

To upgrade AppViewX with a new image version, follow the steps below:

1. Ensure to take a backup of the DB and Vault for rollback in case something goes wrong during upgrade.

   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.

2. Navigate to the installer directory.
3. Edit the **.appviewxctl.yaml** file's upgrade section for the parameters mentioned below.

Table 2.

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*: 2022.1.0_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

4. After editing the file, execute the command

```
./appviewxctl upgrade --config .appviewxctl.yaml
```

Rollback Steps

1. Restore the DB using the restore scripts (step 10 in the Installation Steps section) for self-managed DB or in atlas using snapshot restore in the dashboard.
2. Update the **.appviewxctl.yaml** upgrade section's values to the previous image tag and re-run the upgrade command.

Chapter 6: 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 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
```

Chapter 7: More Information

For the latest, most complete information about known and fixed issues with the AppViewX modules, see the latest revision of the release notes.

To access Software Release Notifications for AppViewX Releases, visit our Help center at https://help.appviewx.com/home. You need to log in to your AppViewX account. From the Help center, search by the specific release number or navigate to Release Portal and choose the release, for example, v20.3.0.

Documentation Feedback

We request you to provide feedback, comments, and suggestions so that we can improve the documentation. You can send your comments to *tech-documentation@appviewx.com*

If you are preferred to send feedback through e-mail, be sure to include the following information with your comments:

- Document or topic name
- URL or page number
- Software release version (if applicable).

Requesting Technical Support

Technical product support is available through AppViewX help support center, request to send an email to *help@appviewx.com*

Self-Help Online Tools and Resources

For quick and easy problem resolution, AppViewX is designed an online self-service portal called the help support center that provides you with the following features:

- Find help support center: https://help.appviewx.com/home
- Find product technical documentation: https://help.appviewx.com/documentation
- Find solutions and answer questions using our Knowledge Base: https://internalkb.appviewx.com/knowledge-base
- Download the latest versions of software: https://release.appviewx.com