Managed Kubernetes - AppViewX Stack Install and Upgrade on GKE

Version: 2022.1.0

Copyright AppViewX, Inc.

Copyright © 2022 AppViewX, Inc. All Rights Reserved.

This document may not be copied, disclosed, transferred, or modified without the prior written consent of AppViewX, Inc. While all content is believed to be correct at the time of publication, it is provided as general-purpose information. The content is subject to change without notice and is provided “as is” and with no expressed or implied warranties whatsoever, including, but not limited to, a warranty for accuracy made by AppViewX. The software described in this document is provided under written license only, contains valuable trade secrets and proprietary information, and is protected by the copyright laws of the United States and other countries. Unauthorized use of software or its documentation can result in civil damages and criminal prosecution.

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

Copyright © 2022 AppViewX, Inc. All Rights Reserved.

Preface

Revision History

Revision	Description	Date
v1.0	Managed Kubernetes - AppViewX Stack Install and Upgrade on GKE	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 GKE, 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.

Copyright © 2022 AppViewX, Inc. All Rights Reserved.

Chapter 1: GCP Components

The following GCP components are utilized by AppViewX:

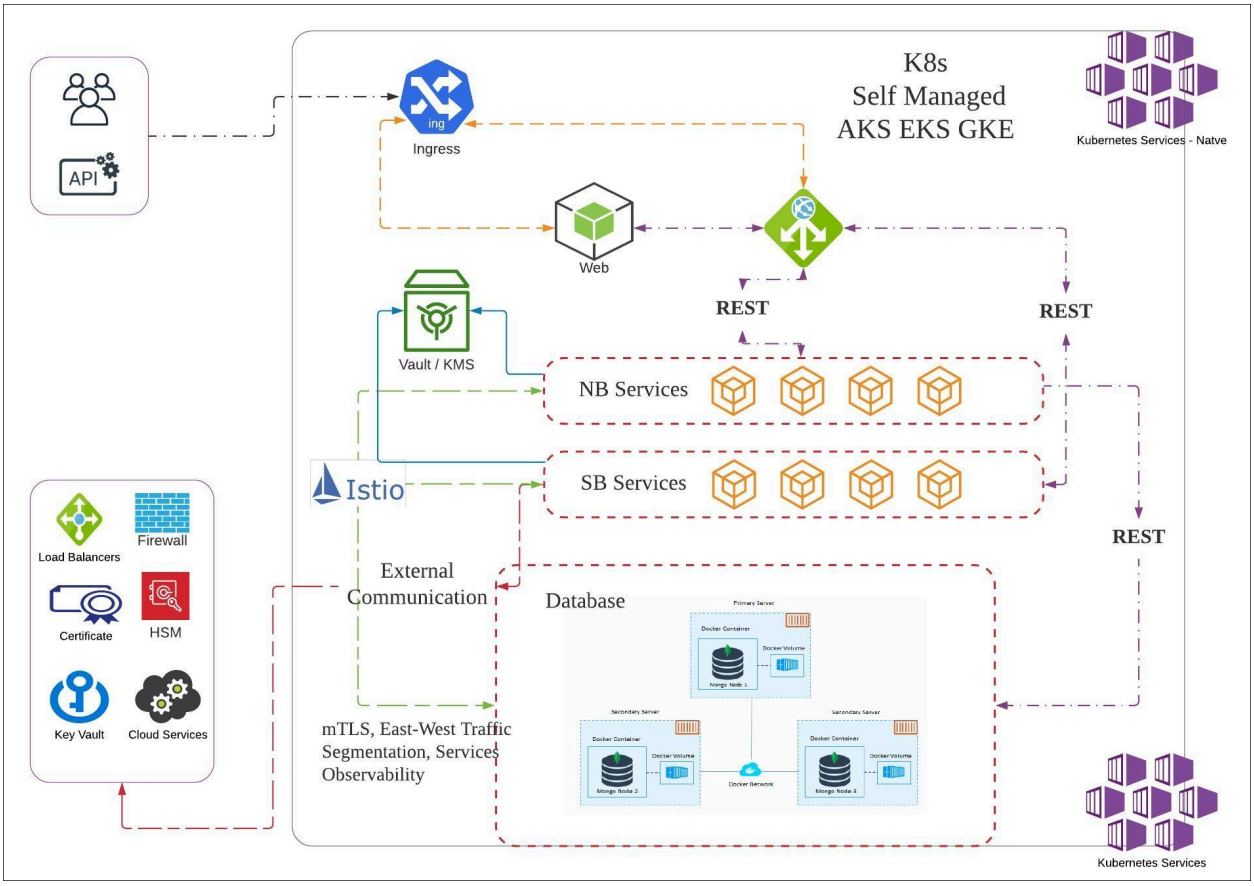
- Google Kubernetes engine
- Storage Bucket for storing MongoDB and Vault backups
- Service account for accessing storage bucket and GCR registry

Copyright © 2022 AppViewX, Inc. All Rights Reserved.

Chapter 2: Deployment Architecture

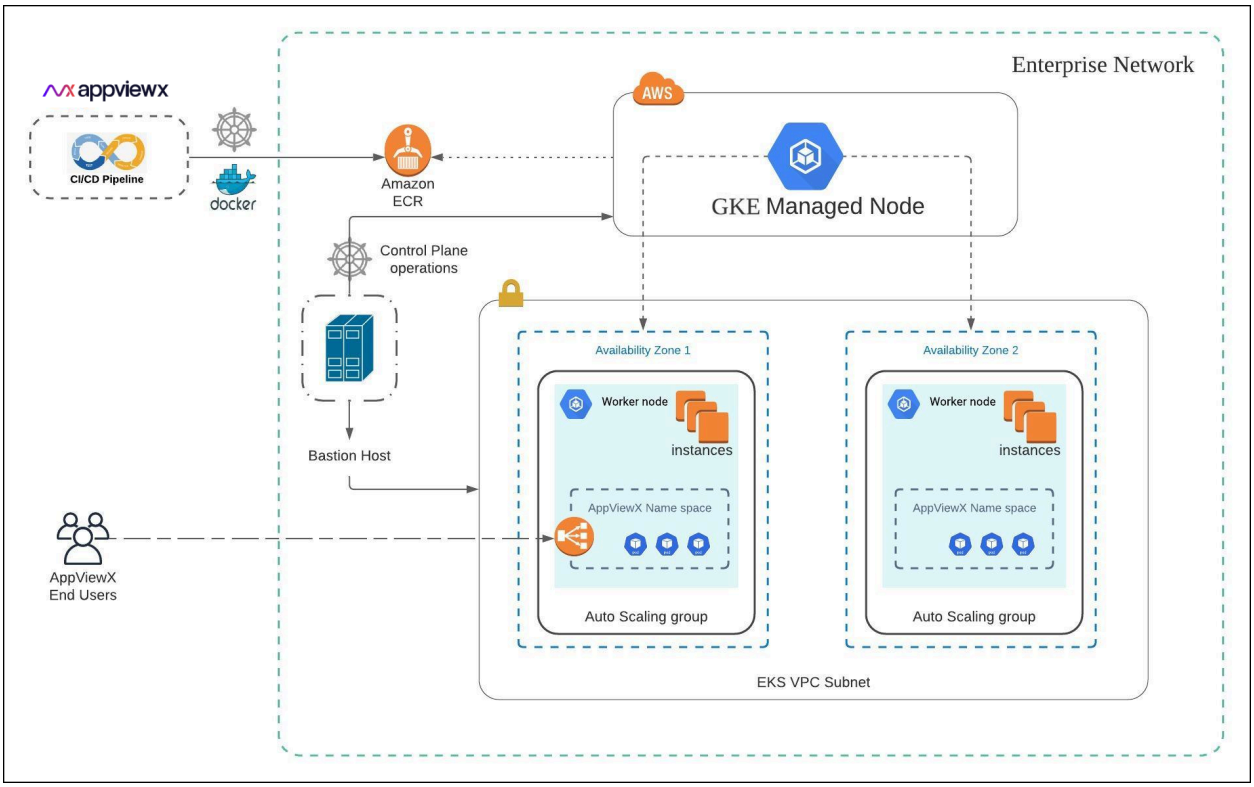
- Kubernetes Architecture
- GKE Deployment Model
- Cloud Connector

Kubernetes Architecture

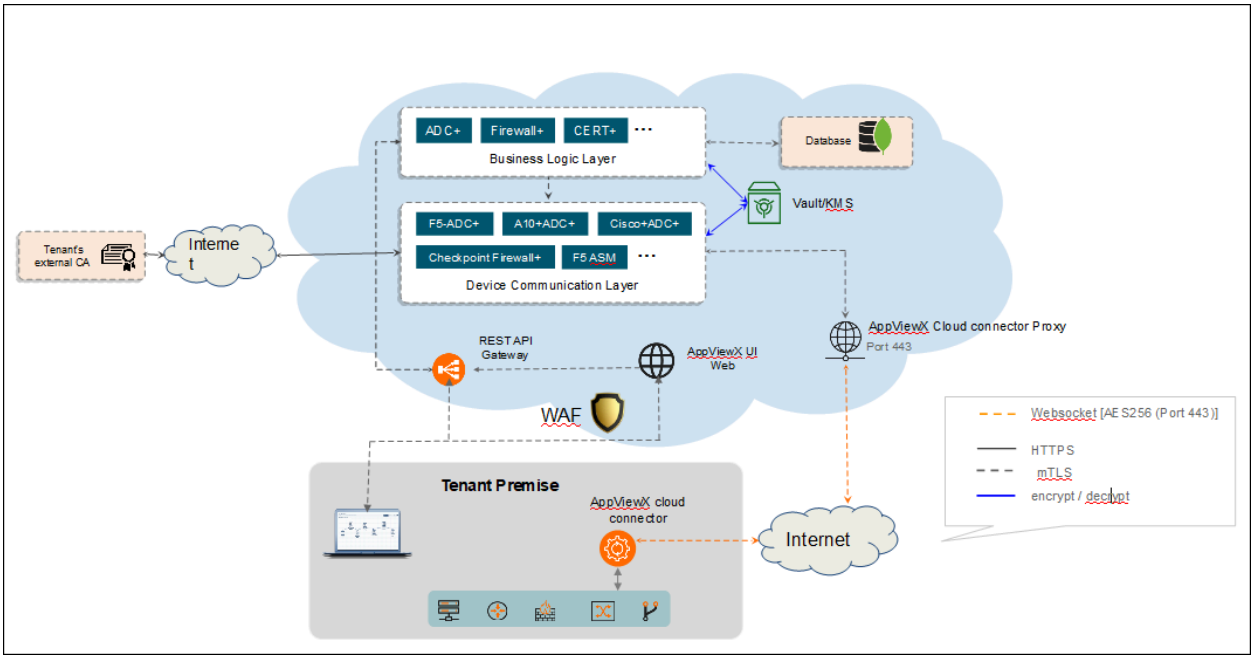


Copyright © 2022 AppViewX, Inc. All Rights Reserved.

GKE Deployment Model



Cloud Connector



Copyright © 2022 AppViewX, Inc. All Rights Reserved.

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

Copyright © 2022 AppViewX, Inc. All Rights Reserved.

Chapter 3: Prerequisites

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:

Copyright © 2022 AppViewX, Inc. All Rights Reserved.

- `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: 1.22
- 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
- MongoDB backup
- Vault backup

A summary of steps for creating the storage bucket is as follows: .

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

Copyright © 2022 AppViewX, Inc. All Rights Reserved.

4. Provide the GKE cluster name, bucket name for storing the MongoDB/Vault backup, and downloading the iControlJar.
5. Save the code block in a file.
6. Grant execute permission (chmod +x <filename>).
7. Run the script (bash filename).
8. Store output of the script and pass as annotation in the global utility config.

The above steps are listed in the script below:

```
#########################################################################
# Provide inputs
# Peoject ID where the EKS cluster and bucket should be created
project=<PROJECT_ID>

# ClusteName is the GKE cluster name where the stack will be installed
clusterName=<CLUSTER_NAME>

# Comma separated nodepool name, e.g: "appnodepool,mongonodepool"
nodePools="<NODE_POOL_1,NODE_POOL_2>"

# Region name, e.g: asia-south1
region=<REGION_NAME>
#########################################################################
# Default values
k8sServiceAccount="avx-storage-bucket-access-ksa"
gcpServiceAccount="avx-storage-bucket-access-gsa"
k8sServiceAccountImagePull="avx-gcr-registry-access-ksa"

# First workload identity should be enabled cluster wide. Can be done from the portal after cluster creation or at the time of cluster creation
gcloud container clusters update $clusterName \
    --region=asia-south1 \
    --workload-pool=$project.svc.id.goog

# Enable meta server on the node pools
for i in $(echo $nodePools | tr ',' '\n');do
  gcloud container node-pools update $i --region=$region \
    --cluster=$clusterName \
    --workload-metadata=GKE_METADATA
```

Copyright © 2022 AppViewX, Inc. All Rights Reserved.

```
done


# Create gcp IAM service account
gcloud iam service-accounts describe $gcpServiceAccount@$project.iam.gserviceaccount.com


if [ $? -ne 0 ]; then
    gcloud iam service-accounts create $gcpServiceAccount --project=$project
fi


# Bind gcp IAM service account with Kubernetes Serice account for bucket access
gcloud iam service-accounts add-iam-policy-binding \
  --role roles/iam.workloadIdentityUser \
  --member "serviceAccount:$project.svc.id.goog[avx/$k8sServiceAccount]" \
  $gcpServiceAccount@$project.iam.gserviceaccount.com


# Bind gcp IAM service account with Kubernetes Serice account for image pull access
gcloud iam service-accounts add-iam-policy-binding \
  --role roles/iam.workloadIdentityUser \
  --member "serviceAccount:$project.svc.id.goog[avx/$k8sServiceAccountImagePull]" \
  $gcpServiceAccount@$project.iam.gserviceaccount.com


# Bind IAM role to the IAM service account
gcloud projects add-iam-policy-binding $project \
--member=serviceAccount:$gcpServiceAccount@$project.iam.gserviceaccount.com \
--role=roles/storage.objectAdmin


echo ""
echo "Annotation:  $gcpServiceAccount@$project.iam.gserviceaccount.com"
```

Note: After the script is executed, capture the output **Annotation** which is required in the global utility config.

Copyright © 2022 AppViewX, Inc. All Rights Reserved.

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 mongodb 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 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 (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 **Google**, execute the command below.

   ```
   ./appviewxctl config generate --provider google
   ```

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**).

Copyright © 2022 AppViewX, Inc. All Rights Reserved.

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 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.
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. *Example*: gcr.io/pe-qa-358108
install.imageTag	The tag of the image that will be used for installation.

Copyright © 2022 AppViewX, Inc. All Rights Reserved.

Parameters	Description of Values
	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. • 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:

Copyright © 2022 AppViewX, Inc. All Rights Reserved.

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.

Copyright © 2022 AppViewX, Inc. All Rights Reserved.

Parameters	Description of Values

	Example:
	`useDockerPrivateRegistry: true`
	`dockerhub:`
	`  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. For details of each size refer to document - Managed Kubernetes Sizing. 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

Copyright © 2022 AppViewX, Inc. All Rights Reserved.

Parameters	Description of Values

	• 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`

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. To procure these values refer to the page AVX Stack Install and Upgrade to GKE.

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:*

Copyright © 2022 AppViewX, Inc. All Rights Reserved.

Parameters	Description of Values
	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 *Example:* bucketObject: "appviewx-samplebucket" serviceAccountAnnotation: "avx-storage-bucket-access-gsa@sampleproject.iam.gserviceaccount.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. 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:** Let the installation proceed to completion until you see the following message:
>
> [Install] Successfully installed Appviewx infra stack

Copyright © 2022 AppViewX, Inc. All Rights Reserved.

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 <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
    ```

12. To proceed with the AppViewX application installation, execute the command:

    ```
    ./appviewxctl installapp --config .appviewxctl.yaml
    ```

Copyright © 2022 AppViewX, Inc. All Rights Reserved.

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. 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:

Copyright © 2022 AppViewX, Inc. All Rights Reserved.

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}')
```

Copyright © 2022 AppViewX, Inc. All Rights Reserved.

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. *Example*: gcr.io/pe-qa-358108
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

Copyright © 2022 AppViewX, Inc. All Rights Reserved.

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 11 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.

Copyright © 2022 AppViewX, Inc. All Rights Reserved.

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 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
```

Copyright © 2022 AppViewX, Inc. All Rights Reserved.

```
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
```

Copyright © 2022 AppViewX, Inc. All Rights Reserved.

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

Copyright © 2022 AppViewX, Inc. All Rights Reserved.