TIBCO Software, A
Business Unit of Cloud
Software Group
Santa Clara, CA
www.tibco.com

How to Configure TIBCO Messaging Quasar Powered - by Apache Pulsar in a Kubernetes Environment

This document describes how to configure Apache Pulsar in a Kubernetes environment.

Version 1.3 July 2023

Updated for TIBCO Messaging Quasar - Powered by Apache Pulsar 2.11.1-1, generic Kubernetes, and Helm support

For more information, please contact:

TIBCO Software, A Business Unit of Cloud Software Group
Santa Clara, CA
USA

TIBCO®

Table of Contents

TIBCO®

Table of Figures

1 Overview

Running TIBCO Messaging Quasar - Powered by Apache Pulsar (APD) on different Kubernetes Container Platforms involves:

- Configuring a Kubernetes cluster for TIBCO Messaging Quasar - Powered by Apache Pulsar. The Kubernetes Container can be configured on Azure Kubernetes Service (AKS), Amazon Elastic Kubernetes Service (EKS), Google Kubernetes Engine (GK), or a generic Kubernetes environment.
- Creating a Docker® image of the Apache Pulsar Core, where the container will be hosted in a Docker Registry
- Creating persisted volumes for Zookeeper, Bookkeeper, and the Broker
- Configuring and creating Kubernetes containers based on the Docker® image for the individual components
- Configuring Load Balancer(s) in Kubernetes to access APD
- Ensuring APD is instrumented (JMX) for TIBCO Message Monitor with the APD extension.
- Optionally, Helm Charts may be used to deploy APD.

1.1 Supported Versions

The steps described in this document are supported for the following versions of the products and components involved:

- TIBCO Messaging - Powered by Apache Pulsar 2.11.0-1 or *later* (TIBCO Distribution)

  TIBCO Messaging - Powered by Apache Pulsar can be downloaded from *edelivery.tibco.com*
- Docker Community/Enterprise Edition should be most recent version.
- Kubernetes 1.2x or Red Hat OpenShift Container Platform 4.x. Recommend latest versions of the container platform
- HELM 3.9.2 or *later*
- TIBCO Msgmon 1.0 or *newer* (optional)

1.2 Prerequisites

The reader of this document must be familiar with:

- Kubernetes concepts, installation, and administration
- Kubernetes CLI, *kubectl*
- TIBCO Messaging Quasar – Powered by Apache Pulsar configuration
- Helm Chart configuration (optional)
- TIBCO Msgmon (optional)

1.3 Prepare Local Environment

The following infrastructure should already be in place:

- A *Linux* or *MacOS* machine equipped for building Docker images
- The following software must already be downloaded to the *Linux* or *macOS* machine equipped for building Docker images.

  Note: All software must be for Linux!

- TIBCO Messaging - Powered by Apache Pulsar installation package from TIBCO (APD). Download the EE from *edelivery.tibco.com*
- The *tibapd__kubernetes_files_2.11.zip.* The zip file contains the necessary *Docker* and *Kubernetes, and helm* build files. Download from *https://community.tibco.com/wiki/tibcor-messaging-article-links-quick-access*
- Create a directory, such as *tibapd_kubernetes_files_2.11.*
- *Unzip* tibapd_kubernetes_files_2.11.zip into the new directory
- *Unzip* the TIBCO Messaging Quasar – Powered by Apache Pulsar installation package. Copy the *TIB_msg-apd*.rpm* file(s) to the *tibapd_kubernetes_files_2.11/docker/docker_files/bin* directory. **Note:** The tibapd Docker image will be ~3.0GB. By default, it will contain all features of Pulsar, including *apis, connectors,* and *offloaders.* If not all Pulsar features are required, and to get a smaller base Pulsar container (1.4GB), only use the *TIB_msg-apd_2.x.x_linux_x86_64-core.rpm*.
- Install Docker on the workstation to build the Apache Pulsar image.
- Install the kubectl command-line tool to manage and deploy applications to Kubernetes in Kubernetes from a workstation. *OC* may also be used in OpenShift environments.
- Install Helm on the workstation, if desired. (optional)

1.4 Pulsar Architecture

This document will outline the creation of the APD architecture in Kubernetes. Using this guide, along with the accompanying software, will create/use:

- Kubernetes cluster that spans multiple availability zones (if so created).
- Container Repository for the TIBCO Messaging Quasar – Powered by Apache Pulsar image
- Load Balancers (Classic - Kubernetes) for external access to the broker
- Three (3) Apache Zookeeper instances
- Three (3) Apache Bookkeeper (bookie) instances
- Three (3) Apache Pulsar Broker instances
- Three (3) Apache Pulsar Proxy instances
- The Brokers, Bookies, and Zookeepers are all configured to use *persisted volumes* (PV) for persisted storage. The persisted volume type is dependent on the container platform Kafka is deployed on.

2 Building the APD Docker image

2.1 Creating the Base Docker Image

The content of the container that will run on Kubernetes derives from a Docker image that first needs to be created and then hosted in a Docker registry.

To create an Pulsar Docker image, use the **`tibapdcreateimage`** script on a machine equipped for building Docker images.

Note: Redhat/Ubi8 is used for the base OS. This can be changed, but other modifications (not documented) may be required.

Note: The **`tibapdcreateimage`** script will define the REST ports for the Msgmon APD extension. *8001* will be used for Zookeeper, *8002* will be used for the Bookies, *8080* is used for the Broker, and *8081* for the Proxies. These can be changed in the script, Docker *run.* or in the Kubernetes yaml files.

Use the following steps to prepare the environment:

- Change directory to the *tibapd_kubernetes_files_2.11/docker* directory.
- The **`tibapdcreateimage`** script will be located in the *docker* directory, and does not require any modifications for the creating the Docker image.
- The **`tibapdcreateimage`** has options controlling what feature to build, output directory, and the tag.

For example:

```
./tibapdcreateimage
Syntax: ./tibapdcreateimage <APD files location> [-f <APD feature>]* [-d
<Docker image output directory>] [-t <tag name>]

     where the required arguments are:
         <Location of the APD files - bin/conf/scripts>

     and the [optional arguments] are:
         -f <APD feature: zookeeper, bookie, broker, apdconfig, or all>.
             Default is all.
         -d <Docker image output directory>. Create a Docker image file
there.
         Default is no Docker image file is created.
         -t <tag name> used to tag the image.

         *: Multiple -f entries allowed.
```

Figure 1 - tibapdcreateimage options

The default is build the image with *all* features, the image is not created in an output directory, and the tag is *latest.*

The **tibapdcreateimage** only requires the directory where the APD docker files and APD installation files are located. The default will be *docker_files,* as shown in the following example.

```
./tibapdcreateimage docker_files
```

Figure 2 - Running the tibapdcreteimage script

Once complete, the new ***tibapd:latest*** Docker image will be created.

The **`tibapdcreateimage`** script can be modified to meet your specific needs, if required.

2.2 Hosting the Image

Tag the images to suit your Docker registry location and push the images there.
Note: If the images are to be hosted on AKS/EKS/GKE, there may be additional steps required to login into the registry. See the specific cloud provider's documentation for details on uploading the Docker images to the respective registry.

Note: with AKS, it is now possible to attach the Azure Container Registry (ACR) directly to the Azure Kubernetes Cluster, making this a simple step, which does not require a *secret.*

Note: With Red Hat OpenShift, a Kubernetes *secret* is often required, and must be defined to pull the image from the registry. See the Red Hat OpenShift documentation for details on creating the secret and modifying the Kubernetes yaml files to use them.

For example:

```
> docker tag tibapd:latest docker.company.com/path/tibapd:latest
> docker push docker.company.com/path/tibapd:latest
```

Figure 3 - Tag and Push the tibapd Docker Image

3 Kubernetes Setup

As previously mentioned, APD can run on virtually all Kubernetes Container Platforms. These include generic on premise Kubernetes, on premise or cloud versions of Red Hat OpenShift, Azure Kubernetes Service (AKS), Amazon Elastic Kubernetes Service (EKS), and Google Kubernetes Engine (GKE). Though not tested, APD should also work with SuSE Rancher and Tanzu.
The Pulsar deployment on any of these container platforms is similar with differences mainly with the persisted storage.
This section will provide details of configuring APD in Kubernetes, highlighting the differences between container platforms.

3.1 Create the Kubernetes Cluster

A new or existing Kubernetes cluster can be used to deploy the TIBCO Messaging – Powered Apache Pulsar components. In general, a minimum of three (3) nodes are required. Each node requires a minimum of 8 cores, and 32+ GB of RAM for the APD components, depending on requirements.

Note: If using an existing Kubernetes cluster, ensure there is sufficient resources for all pods!

Note: If more than three (3) bookies or brokers are used, additional nodes are required, due to the pod distribution policy set in Kubernetes.

The system resource requirements for Pulsar in Kubernetes will be similar to a Pulsar environment running on bare metal.

Note: The general size and number of nodes required, is based on a small production environment for TIBCO Messaging – Powered by Apache Pulsar. If just a standalone APD cluster is being built, the number of *replicas,* along with the *vCPUs, RAM,* and *persisted storage* can all be adjusted in the *yaml* files used to deploy Pulsar in Kubernetes. See section 4.1 for details.

3.2 Configuring Kubectl to the Kubernetes Service

The Kubernetes command line tool, *kubectl,* is used to configure the Kubernetes cluster for Apache Zookeeper/Kafka. For *OpenShift* container platforms, the *OC* cli can also be used.
After the Kubernetes cluster has been built, *kubectl* must be configured to connect to the Kubernetes cluster. Follow the documentation for the Kubernetes platform being used to connect

Use *kubectl get nodes* as shown in the following example to verify connecting to the cluster.

```
NAME                              STATUS   ROLES    AGE   VERSION
ip-192-168-25-83.ec2.internal     Ready    <none>   10m   v1.22.17-eks-a59e1f0
ip-192-168-34-80.ec2.internal     Ready    <none>   10m   v1.22.17-eks-a59e1f0
ip-192-168-95-155.ec2.internal    Ready    <none>   10m   v1.22.17-eks-a59e1f0
```

Figure 4 - Verify connecting to the Kubernetes Cluster

4 Configuring APD in Kubernetes

After the *tibapd* Docker image is pushed to the container registry, Kubernetes can be configured to run the tibapd container. **Note:** All APD components are derived from the same Docker image.

4.1 Configuring APD for Kubernetes

There are five (5) templates used for the APD configuration in Kubernetes. These ***must*** be applied in a specific order to ensure the cluster is properly initialized. All yaml templates will be found under *tibapd_kubernetes_files_2.11/kubernetes.*

4.1.1 Storage Class

A Kubernetes *storageclass* must defined for the persistent volume. The storage class definition is dependent on which container platform is used, and whether it is on premise or on a public cloud. On premise storage classes can vary greatly depending on what is used for persisted storage. This document does not provide guidance on the storage class setup for on premise environments. The **`storageClassName:apd-storage`** is used in all other *yaml* files for *Zookeeper, Bookkeeper,* and the *Broker*, so what is used for on premise cluster should reference this storage class name.

AKS Storage Class

The *aks-storage.yaml* file will create a *managed Premium_LRS.* storageclass by default. In Azure, other types of disks are offered, such as standard (HDD). See the Azure documentation for details. No changes are required to the file, unless wanting to change the disk type offered by Azure. Below is an example of *aks-storage.yaml.* This can be used with any of the persisted storage types.

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: apd-storage
provisioner: kubernetes.io/azure-disk
reclaimPolicy: Retain
volumeBindingMode: WaitforFirstConsumer
parameters:
  storageaccounttype: Premium_LRS
  kind: Managed
```

To deploy in AKS, use:

```
> kubectl apply -f aks/aks-storage.yaml
```

EKS Storage Class

In previous releases of EKS, special drivers for persisted volumes (PV) were not required. Current releases, (*1.23 and newer*) of EKS/Kubernetes do require an EBS/CSI driver. This section will describe how to add the Addon driver.

- Before creating addon aws-ebs-csi-driver, the following two links are required for creating IAM role for service accounts. First, complete Steps 1 through 4. Once completed, there should be a role with ${cluster_1}-AmazonEKS_EBS_CSI_DriverRole.

  https://docs.aws.amazon.com/eks/latest/userguide/csi-iam-role.html
  https://docs.aws.amazon.com/eks/latest/userguide/enable-iam-roles-for-service-accounts.html

- Step 5 is required for creating StorageClass, PV and PVC since Kubernetes 1.23. https://docs.aws.amazon.com/eks/latest/userguide/kubernetes-versions.html
- Below are steps for creating the Add-on for the EKS (*region_1/cluster_1*). Remember to change XXXX to your AWS Account, YYYY to your cluster name, and ZZZZ for the AWS region. The script, *APD_Kubernetes/2.11/kubernetes/eks/ebs-csi-setup.sh,* is also provided to do the setup.

- *# Step 1 ~ 4*
- *cluster_1="YYYY"*
- *region_1="ZZZZ"*
- *$ oidc_id=$(aws eks describe-cluster --region ${region_1} --name ${cluster_1} --query "cluster.identity.oidc.issuer" --output text | cut -d '/' -f 5)*
- *$ aws iam list-open-id-connect-providers --region ${region_1} | grep $oidc_id | cut -d "/" -f4*
  ***If output is returned, then you already have an IAM OIDC provider for your cluster and you can skip the next step. If no output is returned, then you must create an IAM OIDC provider for your cluster.*
- *$ eksctl utils associate-iam-oidc-provider --region ${region_1} --cluster ${cluster_1} --approve*
- *$ eksctl create iamserviceaccount --name ebs-csi-controller-sa --namespace kube-system --region ${region_1} --cluster ${cluster_1} --attach-policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy --approve --role-only --role-name ${cluster_1}-AmazonEKS_EBS_CSI_DriverRole*
- *# Step 5, XXXX is the AWS Account*
  $ aws eks create-addon --region ${region_1} --cluster-name ${cluster_1} --addon-name aws-ebs-csi-driver --addon-version v1.16.0-eksbuild1 --service-account-role-arn arn:aws:iam::XXXX:role/${cluster_1}-AmazonEKS_EBS_CSI_DriverRole

- Once the above steps are completed, The *storage class* can be created. The *eks-storage.yaml* file will create the persisted storage with using AWS's *GP2* storage class. There are other options available on AWS, such as *GP3,io1, and io2*. See the Amazon documentation for details. No changes are required to the file, unless wanting to change the disk type offered by AWS. Below is an example of *eks-storage.yaml.* This can be used with any of the persisted storage types.

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: apd-storage
  labels:
    k8s-addon: storage-aws.addons.k8s.io
provisioner: kubernetes.io/aws-ebs
reclaimPolicy: Retain
parameters:
  type: gp2
  fsType: ext4
```

To deploy in EKS, use:

```
> kubectl apply -f eks/eks-storage.yaml
```

GKE Storage Class

The *gke-storage.yaml* file will create the persisted storage with *pd-ssd* (Zonal/Regional) storage. Other storage options are available. See the Google Cloud documentation at https://cloud.google.com/compute/docs/disks for more details. The default persisted storage will encrypt the data *at rest,* and provide replication within the region. No changes are required to the file, unless wanting to change the disk type offered by Google.

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: apd-storage
provisioner: kubernetes.io/gce-pd
reclaimPolicy: Retain
parameters:
  type: pd-ssd
```

To deploy in GKE, use:

```
> kubectl apply -f gke/gke-storage.yaml
```

4.1.2 Zookeeper Configuration File

The *apd_files_2.11/kubernetes/tibapd-zookeeper.yaml* is used to configure the Zookeeper statefulset/pods, and the services for access. There are some necessary changes required to this file. This section will outline these modifications.

- The container registry for the *tibapd image*. The statefulset defined in *tibapd-zookeeper.yaml* requires the *image* be updated to reference the container registry defined in section 2.2. Ensure the proper permissions are set. The image maybe something different than **latest,** depending on how it was *tagged* in Docker.

```
image: <your container registry>/tibapd:latest
```

- The storage size *requests* for the zookeeper persisted storage. The default storage value for the nodes is set to *10 Gi.* While this should be sufficient for most environments, it may be too large/small for all environments. Under the *volumeClaimTemplates*, modify the storage resource request to a smaller/larger value if required.

```
storageClassName: apd-storage
 resources:
    requests:
       storage: 10Gi
```

- **Note:** There are other optional changes which can be made. These include the *port numbers (Zookeeper* and *REST), resource allocations (RAM/vCPU), zookeeper DNS name,* and the *number of replicas.* However, changing any of these values will have ramifications on the other Pulsar processes and the K8 cluster. It is ***not*** recommended that any other changes be made with a deep knowledge of Pulsar and Kubernetes!

4.1.3 APD Config Configuration File

The *apd_files_2.11/kubernetes/tibapd-config.yaml* is used to configure the apd config job that will initialize the metadata for the Pulsar cluster in Zookeeper. It is imperative that Zookeeper is running! There are some necessary changes required to this file. This section will outline these modifications.

- The container registry for the *tibapd image*. The job defined in *tibapd-config.yaml* requires the *image* be updated to reference the container registry defined in section 2.2. Ensure the proper permissions are set. The image maybe something different than **latest,** depending on how it was *tagged* in Docker.

```
image: <your container registry>/tibapd:latest
```

- The *environmental* variables can be set. There is a variable for *Zookeeper Server names and ports, Configuration Store Servers and ports,* and the name of the *Pulsar Cluster.* It is ***not*** recommended to change the Zookeeper or Configuration Store Serves values, but the **cluster name** can be changed. By default, it is *mycluster.* **Note:** The cluster name must also be changed on the *broker* configuration.

```
-name: _BROKER_CLUSTER_NAME
 value: “mycluster”
```

4.1.4 Bookkeeper Configuration File

The *apd_files_2.11/kubernetes/tibapd-bookie.yaml* is used to configure the Bookkeeper statefulset/pods, and the services for access. There are some necessary changes required to this file. This section will outline these modifications.

- The container registry for the *tibapd image*. The statefulset defined in *tibapd-bookie.yaml* requires the *image* be updated to reference the container registry defined in section 2.2. Ensure the proper permissions are set. The image maybe something different than **latest,** depending on how it was *tagged* in Docker.

```
image: <your container registry>/tibapd:latest
```

- The storage size *requests* for the bookie persisted storage. There are several for the bookie ledgers and journals. The default values differ for the different storage. While the defaults should be sufficient for most environments, the ledger and journal allocations may be too large/small for all environments. Under the *volumeClaimTemplates*, modify the storage resource request to a smaller/larger value if required.

```
storageClassName: apd-storage
 resources:
    requests:
       storage: 25Gi
```

- **Note:** There are other optional changes which can be made. These include the *port number (Bookie* and *REST), resource allocations (RAM/vCPU), bookie DNS name, zookeeper servers* and *ports* variable, and the *number of replicas.* However, changing any of these values will have ramifications on the other Pulsar processes and the AKS cluster. It is ***not*** recommended that any other changes be made with a deep knowledge of Pulsar and Kubernetes!

4.1.5 Broker Configuration File

The *apd_files_2.11/kubernetes/yaml_files/tibapd-broker.yaml* is used to configure the Pulsar Broker statefulset/pods, and the services for access. There are some necessary changes required to this file. This section will outline these modifications.

- The container registry for the *tibapd image*. The statefulset defined in *tibapd-broker.yaml* requires the *image* be updated to reference the container registry defined in section 2.2. Ensure the proper permissions are set. The image maybe something different than **latest,** depending on how it was *tagged* in Docker.

```
image: <your container registry>/tibapd:latest
```

- The *environmental* variables can be set. There is a variable for *Zookeeper Server names and ports, Configuration Store Servers and ports, Pulsar memory usage,* the name of the *Pulsar Cluster,* and etc. It is ***not*** recommended to change the environmental variables, except the

cluster name can be changed. By default, it is *mycluster.* **Note:** The cluster name **must match** what is defined in the *config* configuration.

```
-name: _BROKER_CLUSTER_NAME
 value: “mycluster”
```

- In the *service* section, under the *LoadBalancer,* a change is required for the *trusted IP range.* The *trusted IP range* will determine what IP addresses can connect to the load balancer, and it is recommended **not** to be configured to *0.0.0.0/0.* This will need to be changed for the *admin* and *broker* loadbalancers. The example below shows the change required to *tibapd-broker.yaml.*

```
 sessionAffinity: None
externalTrafficPolicy: Cluster
loadBalancerSourceRanges:
  - <your trusted IP range in the form of 0.0.0.0/0>
```

4.1.6 Apply the configuration

Once all of the yaml files have been updated, they can be applied using *kubectl* to the Kubernetes cluster.

Note: There is a specific order they must be applied in, to ensure the Pulsar cluster is initialized properly. The order **must be** *storage, zookeeper, config, bookie,* and finally, the *broker.*

- Use *kubectl apply –f tibapd-zookeeper.yaml* to apply the zookeeper configuration.
- Wait until the three zookeeper pods are *running!*
- Use *kubectl apply –f tibadp-config.yaml*
- Wait until the config pod has *completed!*
- Use *kubectl logs tibapdconfig-job-xxxx* and verify the end of the log looks similar to the following:

```
21:43:18.618 [Curator-Framework-0] INFO
org.apache.curator.framework.imps.CuratorFrameworkImpl - backgroundOperationsLoop
exiting
21:43:18.727 [main-EventThread] INFO  org.apache.zookeeper.ClientCnxn -
EventThread shut down for session: 0x10000575d910000
21:43:18.727 [main] INFO  org.apache.zookeeper.ZooKeeper - Session:
0x10000575d910000 closed
21:43:18.941 [main] INFO  org.apache.pulsar.PulsarClusterMetadataSetup - Cluster
metadata for 'mycluster' setup correctly
```

- Use *kubectl apply –f tibadp-bookie.yaml*
- Wait until the three bookie-pods are *running!*
- Use *kubectl apply –f tibadp-broker.yaml*

Use *kubectl get svc,pods* to verify the services, and the pods are available. Do not continue until the load balancer has been assigned an External IP address as shown in the following example.

```
NAME                  TYPE           CLUSTER-IP       EXTERNAL-IP
PORT(S)                                       AGE
```



```
service/admin-lb     LoadBalancer   10.100.169.112
a92949125b8484968af28637c369cf05-596924812.us-east-1.elb.amazonaws.com
30881:30881/TCP                                  4m8s
service/bookies      ClusterIP      None             <none>
3181/TCP,8002/TCP                                5m16s
service/broker-lb    LoadBalancer   10.100.21.172
a04c4b24b909a449aa5e2148aaa3b802-312595671.us-east-1.elb.amazonaws.com
30652:30652/TCP                                  4m9s
service/brokers      ClusterIP      None             <none>
6650/TCP,6651/TCP,8080/TCP,8443/TCP,8081/TCP     4m9s
service/kubernetes   ClusterIP      10.100.0.1       <none>
443/TCP                                          111m
service/zookeepers   ClusterIP      None             <none>
2181/TCP,2888/TCP,3888/TCP,8001/TCP              25m

NAME                          READY   STATUS      RESTARTS   AGE
pod/bookie-0                  1/1     Running     0          5m16s
pod/bookie-1                  1/1     Running     0          5m16s
pod/bookie-2                  1/1     Running     0          5m16s
pod/broker-0                  1/1     Running     0          4m9s
pod/broker-1                  1/1     Running     0          4m9s
pod/broker-2                  1/1     Running     0          4m9s
pod/tibapdconfig-job-5lj66    0/1     Completed   0          6m43s
pod/zookeeper-0               1/1     Running     0          17m
pod/zookeeper-1               1/1     Running     0          18m
pod/zookeeper-2               1/1     Running     0          20m
```

Figure 5 – Running APD environment

4.2 Stopping or Deleting the APD processes

To stop the TIBCO Messaging Quasar – Powered by Apache Pulsar processes running in Kubernetes without deleting them, use the **`kubectl scale`** operation to set its number of replicas to 0. All configuration and data information will be retained.

For example:

```
> kubectl scale --replicas=0 statefulset broker
> kubectl scale --replicas=0 statefulset bookie
> kubectl scale --replicas=0 statefulset zookeeper
```

To start the processes again, set its number of replicas back to three (3).

```
> kubectl scale --replicas=3 statefulset zookeeper
> kubectl scale --replicas=3 statefulset bookie
> kubectl scale --replicas=3 statefulset broker
```

Figure 6 - To Stop and Start the APD Statefulsets

To delete the statefulsets, storage, and services entirely, use the *kubectl delete* operation:

```
> kubectl delete -f tibapd-broker.yaml,tibapd-bookie.yaml,tibapd-
config.yaml,tibapd-zookeeper.yaml,tibapd-storage.yaml
```

The corresponding pod, statefulsets, storage class, configuration, and services will be deleted. The PVC and PV **will not** be deleted, nor will the corresponding data. To delete the data, PV, and PVC, use the following:

```
> kubectl delete pvc,pv -all
```

4.3 Connecting to the APD Pods

The *kubectl exec* command using the name of the *Zookeeper, Bookie,* or *Broker pods* name to access the pod. This can be useful for running *pulsar-admin* commands, *pulsar-client* commands, viewing the logs, modifying the configuration file, etc. The example below shows access the *broker-0* pod.

```
> kubectl exec -it broker-0 -- /bin/bash
```

Figure 7 - Pod Access Example

5 Accessing and Testing the Pulsar Environment in Kubernetes

Pulsar can be accessed multiple ways when running in Kubernetes. APD can be accessed by:

- Connecting to any one of the Kubernetes pods
- Connecting to the Broker through clients running in another Kubernetes pod running in the Kubernetes cluster
- Connecting to the External Kubernetes Load balancer

5.1 Internal Access to APD

The simplest approach to connect to Pulsar is just be connecting to any one of the Kubernetes pods running a Pulsar Broker component. From there, any Pulsar Admin or Client application can be ran.

```
$ kubectl exec -it broker-0 -- /bin/bash
[root@broker-0 volume]# cd /opt/tibco/apd/core
[root@broker-0 core]# bin/pulsar-admin tenants create tibco
[root@broker-0 core]# bin/pulsar-admin namespaces create tibco/messaging
[root@broker-0 core]# bin/pulsar-admin topics create tibco/messaging/mytopic
[root@broker-0 core]# bin/pulsar-admin topics list tibco/messaging
"persistent://tibco/messaging/mytopic"
```

Figure 8 - Internal Connection to Broker to create a topic

5.2 External Access to Pulsar

When *tibapd-broker.yaml* was applied to the K8 cluster, two K8 Services were created, as shown below:

```
NAME         TYPE           CLUSTER-IP    EXTERNAL-IP     PORT(S)           AGE
admin-lb     LoadBalancer   10.0.109.31   40.76.129.210   30881:30881/TCP   11m
broker-lb    LoadBalancer   10.0.40.241   40.88.251.49    30652:30652/TCP   11m
```

These Load balancers provide external access to the TIBCO Messaging Quasar – Powered by Apache Pulsar environment running in Kubernetes.

The *LoadBalancer* K8 service, *admin-lb,* will provide external admin access to Pulsar on port *30881*, while the *broker-lb* will allow client access to Pulsar on port *30652*. In the following example, the broker-lb *external-IP* and *port* will be used to produce messages to the newly created topic, *mytopic,* created in the previous example. The pulsar-client produces an abundant amount of output to produce the messages. Most of it has been omitted from this example. However, the connection to the server and success is shown.

Note: an external machine with APD installed is required for this test. Two terminal sessions are required.

In the first terminal session, change directory to where the APD *core* is installed, and start a client consumer process as shown in the following example.

```
$bin/pulsar-client --url pulsar://40.88.251.49:30652 consume tibco/messaging/mytopic -n
5 -s rwf
```

Figure 9 - Start an external client consumer process

In the second terminal session, start and run the producer.

```
$bin/pulsar-client --url pulsar://40.88.251.49:30652 produce tibco/messaging/mytopic -n
5 -m "hi pulsar"
16:26:43.314 [pulsar-client-io-1-1] INFO  org.apache.pulsar.client.impl.ConnectionPool -
[[id: 0xfa0f7828, L:/192.168.0.214:58054 - R:/40.88.251.49:30652]] Connected to server
16:26:43.802 [pulsar-client-io-1-1] INFO
org.apache.pulsar.client.impl.ProducerStatsRecorderImpl - Starting Pulsar producer perf
with config: {
  "topicName" : "tibco/messaging/mytopic", …
…16:26:47.381 [main] INFO  org.apache.pulsar.client.impl.PulsarClientImpl - Client
closing. URL: pulsar://40.88.251.49:30652
16:26:47.455 [pulsar-client-io-1-1] INFO  org.apache.pulsar.client.impl.ProducerImpl -
[tibco/messaging/mytopic] [mycluster-1-0] Closed Producer
16:26:47.458 [pulsar-client-io-1-1] INFO  org.apache.pulsar.client.impl.ClientCnx - [id:
0xfa0f7828, L:/192.168.0.214:58054 ! R:/40.88.251.49:30652] Disconnected
16:26:47.461 [pulsar-client-io-1-1] INFO  org.apache.pulsar.client.impl.ClientCnx - [id:
0x0c34997f, L:/192.168.0.214:58055 ! R:/40.88.251.49:30652] Disconnected
16:26:47.464 [main] INFO  org.apache.pulsar.client.cli.PulsarClientTool - 5 messages
successfully produced
```

Figure 10 – External Access to broker to produce messages

The output from the should be similar to the example below, verifying data can be sent to/from the broker from an external source.

```
15:44:00.550 [pulsar-client-io-1-1] INFO  org.apache.pulsar.client.impl.ConnectionPool -
[[id: 0xa490681d, L:/192.168.23.23:63480 - R:/ 40.88.251.49:30652]] Connected to server
15:44:00.551 [pulsar-client-io-1-1] INFO  org.apache.pulsar.client.impl.ClientCnx - [id:
0xa490681d, L:/192.168.23.23:63480 - R:/ 40.88.251.49:30652] Connected through proxy to
target broker at broker-0.brokers.default.svc.cluster.local:6650
15:44:00.640 [pulsar-client-io-1-1] INFO  org.apache.pulsar.client.impl.ConsumerImpl -
[tibco/messaging/mytopic][rwf] Subscribing to topic on cnx [id: 0xa490681d,
L:/192.168.23.23:63480 - R:/ 40.88.251.49:30652], consumerId 0
15:44:00.711 [pulsar-client-io-1-1] INFO  org.apache.pulsar.client.impl.ConsumerImpl -
[tibco/messaging/mytopic][rwf] Subscribed to topic on /40.88.251.49:30652 -- consumer: 0
----- got message -----
key:[null], properties:[], content:hi pulsar
----- got message -----
key:[null], properties:[], content:hi pulsar
----- got message -----
key:[null], properties:[], content:hi pulsar
----- got message -----
key:[null], properties:[], content:hi pulsar
----- got message -----
key:[null], properties:[], content:hi pulsar
15:44:12.282 [main] INFO  org.apache.pulsar.client.impl.PulsarClientImpl - Client
closing. URL: pulsar:// 40.88.251.49:30652
15:44:12.351 [pulsar-client-io-1-1] INFO  org.apache.pulsar.client.impl.ConsumerImpl -
[tibco/messaging/mytopic] [rwf] Closed consumer
15:44:12.354 [pulsar-client-io-1-1] INFO  org.apache.pulsar.client.impl.ClientCnx - [id:
0xa490681d, L:/ 40.88.251.49:63480 ! R:/20.81.16.53:30652] Disconnected
```

Figure 11 - client output

6 Using Helm to deploy APD

TIBCO Messaging Quasar - Powered by Apache Pulsar (APD) can now be configured to use Helm Charts to deploy APD on Kubernetes. The section can be used to deploy APD using Helm Charts. All Helm charts are part of the *tibapd_kubernetes_files_2.11.zip,* and will be in the *helm* directory.

6.1 Preparing the Environment to Use Helm

Helm is an alternate way to deploy containers in Kubernetes. The Kubernetes environment must exist and be sized appropriately for the deployment.

Follow sections 2 through 4.1.1 before continuing. The *tibapd* image must exist, and have been *tagged/pushed* to the appropriate registry.

All Kubernetes platforms, be it private on-premise, or in a public cloud, have a different setup for the persistent storage used by the *storageclass.* Use section 4.1.1 to configure the storageclass.
Note: On-premise persistent storage can vary. Please work with the Kubernetes administrator for your environment to determine what pv and pvc should be configured and used.

6.2 Using Helm to Deploy APD

Once the Kubernetes nodes are ready and the storageclass exists, APD can be deployed with Helm.

The similar *yaml* files used for deployment in Kubernetes, are used for Helm, with the exception, almost all values have been modified to be a variable that helm can change at deployment. See section 4.1 for details on the different yaml files used.

Note: an additional change was the *load balancer* services creation was separated to its own yaml file. If external load balancer is not required, remove the *helm/charts/broker/templates/tibapd-broker-lb.yaml* file.

6.2.1 Helm Values

The *yaml* files used to deploy APD have variables for most of the commonly set values. These include variables for the Docker image registry, port values, and service names, etc. The following example show all variables available for modification.

```
# Copyright (c) 2023 Cloud Software Group, Inc. All Rights Reserved.
Confidential and Proprietary.

#pulsar:
# Generic
  logLevel: "info"
  imageName: "tibapd:latest"
  lastRelease: "first"
  storageclass: "apd-storage"
```

```
  cluster_name: "mycluster"
  lbsourcerange: ""
# Broker specific
  broker_port: "6650"
  tlsbroker_port: "6651"
  broker_http_port: "8080"
  httptls_port: "8443"
  brokerproxy_port: "6652"
  httpproxy_port: "8081"
  broker_lb_port: "30652"
  admin_lb_port: "30881"
  broker_replicas: "3"
  broker_lb_name: "broker-lb"
  admin_lb_name: "admin-lb"
  broker_service_name: "brokers"
  broker_name: "broker"
  broker_volumeclaim: "broker-conf"
  broker_conf_size: "100Mi"
# Zookeeper specific
  zookeeper_port: "2181"
  zookeeper_peer: "2888"
  zookeeper_leader: "3888"
  zookeeper_rest: "8001"
  zookeeper_service_name: "zookeepers"
  zookeeper_name: "zookeeper"
  zookeeper_data_size: "10Gi"
  zookeeper_data_volumeclaim: "zookeeper-data"
  zookeeper_conf_size: "100Mi"
  zookeeper_conf_volumeclaim: "zookeeper-conf"
# Bookie specific
  bookie_port: "3181"
  bookie_rest: "8002"
  bookie_service_name: "bookies"
  bookie_name: "bookie"
  bookie_replicas: "3"
  bookie_conf_volumeclaim: "bookie-conf"
  bookie_conf_size: "100Mi"
  bookie_journal_volumeclaim: "bookie-journal"
  bookie_journal_size: "25Gi"
  bookie_ledger_volumeclaim: "bookie-ledger"
  bookie_ledger_size: "25Gi"
  zookeeper_ledger_volumeclaim: "zookeeper-ledger"
  zookeeper_ledger_size: "100Mi"
# Config specfic
  config_volume: "adpconfig-conf"
```

Figure 12 - Default Values

All values listed are the default values used by Helm in the different charts. The *values.yaml* is used by all of the charts. All values can be changed directly in the yaml file or by setting the variable value when installing the chart.

6.2.2 Helm Charts

APD uses three charts to start APD; *zookeeper, bookie,* and *broker.* The charts **must** be started in a specific order. The Zookeeper chart is first, which also configures the metadata for the Pulsar cluster, then the bookies, and finally the brokers.
To make the install simple, a bash script, *install-pulsar.sh,* is provided. The following example shows the script.

```
#
# Copyright (c) 2023 Cloud Software Group, Inc. All Rights Reserved.
Confidential and Proprietary.
# Script to uninstall the Pulsar components
#
TIBAPD_IMAGE_NAME="tibco/tibapd:latest" (1)
NAMESPACE=tibco (2)
settings="imageName=registry.azurecr.io/$TIBAPD_IMAGE_NAME,lbsourcerange=20
3.119.81.22/32,cluster_name=testcluster,broker_replicas=3" (3)
#
# Install zookeeper and wait for the tib-config job to complete
echo " Installing zookeeper and configuring the cluster..."
helm upgrade --install --namespace $NAMESPACE zookeeper Charts/zookeeper -f
values.yaml --set $settings --wait --wait-for-jobs
#
# Install the bookies
echo " "
echo " Installing the bookies..."
helm upgrade --install --namespace $NAMESPACE bookie Charts/bookie -f
values.yaml --set $settings --wait
#
sleep 10
#
# Install the brokers
echo " "
echo " Installing the brokers..."
helm upgrade --install --namespace $NAMESPACE broker Charts/broker -f
values --set $settings --wait
echo " "
echo " done."
```

Figure 13 - install-quasar.sh script

1) The name of the APD Docker image
2) The Kubernetes *namespace* to use
3) All variables to be set from the defaults. In this example the *registry* for the Docker image, the *lbsoucerange* for external access, the Plusar *cluster_name,* and the number of broker *replicas.* This line can contain any of the variables shown in the previous figure.

The script is not required, and all Helm Chart installs can be done with any of the supported Helm commands.

A second script is also provided, *uninstall-quasar.sh.* This script will uninstall all Helm Charts in the reverse order. No changes are required to this script, except for the *namespace* name.

6.2.3 Installing the Helm Charts

To install the Helm Charts, run the install-quasar script. It will take a few minutes to run, as the Helm will wait for the configuration job to finish before continuing.

```
Installing zookeeper and configuring the cluster...
Release "zookeeper" does not exist. Installing it now.
NAME: zookeeper
LAST DEPLOYED: Fri May  5 14:40:42 2023
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None

 Installing the bookies...
Release "bookie" does not exist. Installing it now.
NAME: bookie
LAST DEPLOYED: Fri May  5 14:41:21 2023
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None

 Installing the brokers...
Release "broker" does not exist. Installing it now.
NAME: broker
LAST DEPLOYED: Fri May  5 14:41:58 2023
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None

 done.
```

Figure 14 - Successful Helm Chart installation

Use" *kubectl get pods"* to verify all pods are running and ready.

```
NAME                      READY   STATUS      RESTARTS   AGE
bookie-0                  1/1     Running     0          5m9s
bookie-1                  1/1     Running     0          5m9s
bookie-2                  1/1     Running     0          5m9s
broker-0                  1/1     Running     0          4m32s
broker-1                  1/1     Running     0          4m32s
broker-2                  1/1     Running     0          4m31s
broker-3                  1/1     Running     0          4m31s
tibapdconfig-job-fmlf8    0/1     Completed   0          5m48s
zookeeper-0               1/1     Running     0          5m48s
zookeeper-1               1/1     Running     0          5m48s
zookeeper-2               1/1     Running     0          5m48s
```

To access and test the new APD cluster, follow Section 5.

The uninstall-pulsar.sh script can be used to uninstall the Helm Charts.

```
 Uninstalling the brokers...
release "broker" uninstalled

 Uninstalling the bookies...
release "bookie" uninstalled

 Uninstallng zookeeper...
release "zookeeper" uninstalled

 done.
```

Figure 15 - Uninstalling the Helm Charts

Note: The *storageclass* and *persistent volumes* need to be uninstalled manually.