# TIBCO® MDM

## Cloud Deployment Guide

*Version 9.3.2*
*May 2025*

Document Updated: September 2025

# Contents

# TIBCO MDM on Container Platforms

You can containerize TIBCO MDM and run it in a Docker or Kubernetes environment. To containerize TIBCO MDM, you must build and run the Docker images using the Dockerfiles bundled in TIB_mdm_9.3.2_container.zip.

The Dockerfiles are delivered as a ZIP file on the TIBCO eDelivery website. Download the TIB_mdm_9.3.2_container.zip file and extract its content to a separate directory named as docker. In the docker directory, locate the ready-to-use Dockerfile and other scripts required to build the images. You can build images using the Dockerfiles, and then run them as containers. For information about building images, see Building and Running Docker Image for the TIBCO MDM All-in-one Container and Build Docker Image for TIBCO MDM Cluster Container.

To run an application, you require the application, all its dependencies, and configuration files. A container provides an OS environment to hold together all supporting components and tools that are required to run the application. This provides a consistent environment without the overhead of OS dependencies and other infrastructural requirements. For information about Docker concepts, such as, Dockerfile, Docker Image, and Container, see Docker documentation.

To save (persist) data and also to share data between containers, Docker volumes are used. Docker volumes are file systems mounted on Docker containers to persist or retain data generated by the running container.

The TIBCO MDM containers are available in the following modes:

- TIBCO MDM All-in-one Container
- TIBCO MDM Cluster

**Note:** Before you build and run the Docker image of TIBCO MDM all-in-one container, TIBCO MDM Cluster container, and the components included in the TIBCO MDM Cluster container, install Docker on the machine and perform the initial setup based on your operating system. For complete details on Docker installation, see Docker documentation.

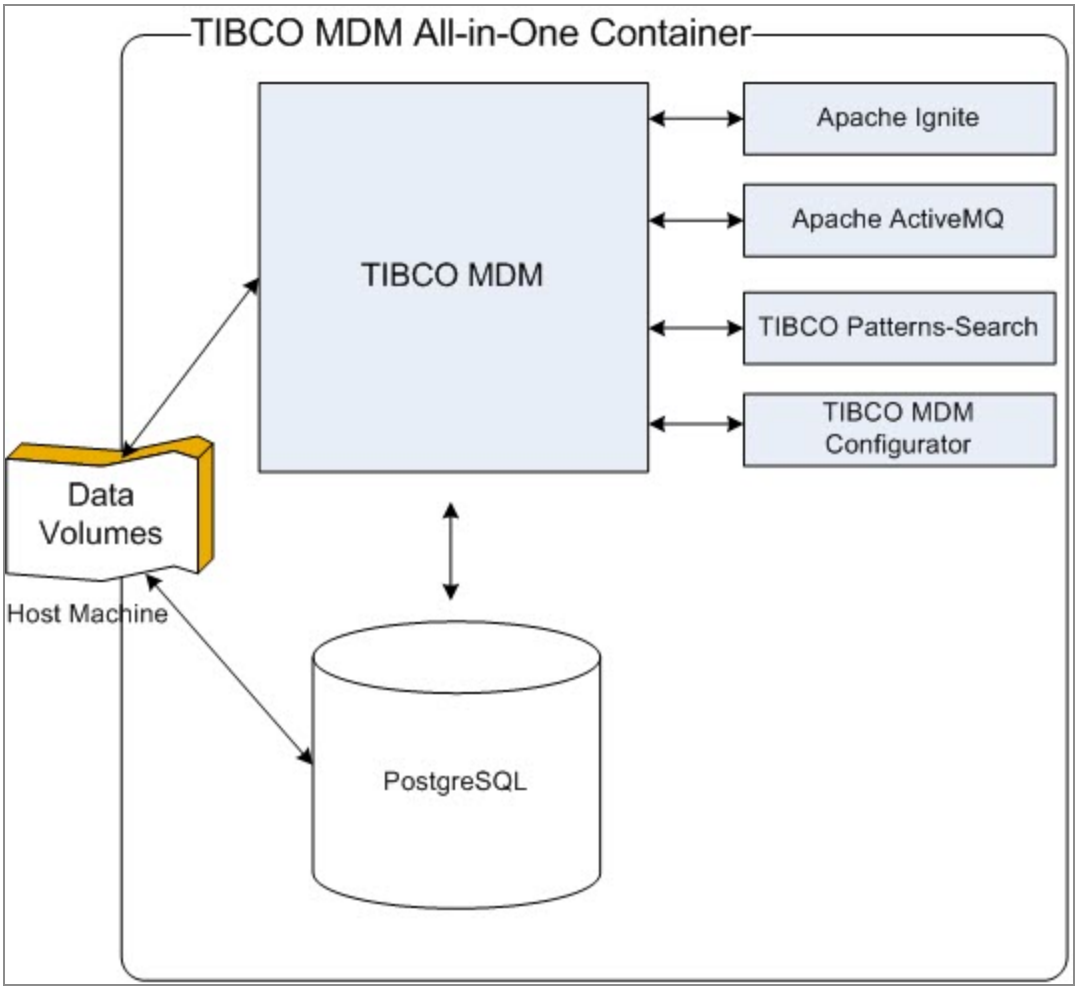
Additionally, TIBCO MDM supports a high-level CRI-O container runtime that offers a lightweight alternative to Docker. CRI-O is an open-source implementation of Kubernetes' container runtime interface (CRI). With the support of Open Container

Initiative (OCI), you can now select different container runtimes (for example, CRI-O) instead of Docker, which conform to the specification.

# TIBCO MDM All-in-one Container

TIBCO MDM all-in-one container bundles components (TIBCO MDM, TIBCO MDM Configurator, JBoss Wildfly, PostgreSQL, Apache ActiveMQ, Apache Ignite, and TIBCO Patterns - Search) as a single container. For the supported versions of these components, see the *Readme.txt* file of TIBCO MDM

You can run the all-in-one container quickly by using only Docker, without complex configurations. You can use the TIBCO MDM all-in-one container in development and QA environments for testing and demos.



The container is configured and ready to be used. To ensure data persistence, you must mount data volumes in the TIBCO MDM all-in-one container. Docker volumes persist even if the container itself is stopped or deleted. You can re-initialize the container by dropping and recreating the volumes, without building the image again. TIBCO MDM all-in-one container supports only the PostgreSQL database.

# Building and Running Docker Image for the TIBCO MDM All-in-one Container

Before you run TIBCO MDM all-in-one container, you must build a Docker image of it.

### Before you begin

- Ensure that you have Dockerfile for TIBCO MDM all-in-one Container. Dockerfile is available in the docker/build/MDMAllInOne directory.
- Enable squash as an **Experimental** feature through Docker configuration. For more information, see Docker documentation.

### Procedure

1. Copy the TIBCO MDM installer file (TIB_mdm-JBOSS_9.3.1_linux_x86_64.zip) to the directory where the Dockerfile is located.
2. On the command line, enter the following command:

   ```
   $> docker build -t mdm:versionnumber.GA --squash --rm=true .
   ```

3. Create the required Docker volumes by using the following commands:

   ```
   docker volume create --name mdmcommon
   docker volume create --name mdmconfig
   docker volume create --name mdmdynservices
   docker volume create --name postgresdata
   docker volume create --name mdmdbdata
   ```

4. On the command line, enter the following command to run the Docker container:

   For example:

   ```
   docker run
   -p 8080:8080
   ```

```
-p 6080:6080
-e SWAGGER_MDM_HOST=host_IP_address
-e SWAGGER_MDM_PORT=8080
-e MDMPORT=8080
-e PROTOCOL=http
-v mdmcommon:/home/mdmuser/tibco/mdm/versionnumber/common
-v mdmconfig:/home/mdmuser/tibco/mdm/versionnumber/config
-v mdmdynservices:/home/mdmuser/tibco/mdm/versionnumber/dynservices
-v postgresdata:/home/mdmuser/tibco/mdm/versionnumber/bin/pgsql/data
-v mdmdbdata:/home/mdmuser/tibco/mdm/versionnumber/bin/pgsql/tablespaces
mdm:versionnumber.latest
```

**Note:** You can specify minimum and maximum memory required by using JAVA_OPTS based on your need. For example,

```
--memory=container memory
-Xms=jvm_minimum_memory
-Xmx=jvm_maximum_memory
```

```
--memory=4096m
-Xms=512m
-Xmx=2048m
```

### What to do next

To access TIBCO MDM, use the following URL: http://*hostname*:8080/eml/Login. For example: http://localhost:8080/eml/Login.

# TIBCO MDM Cluster

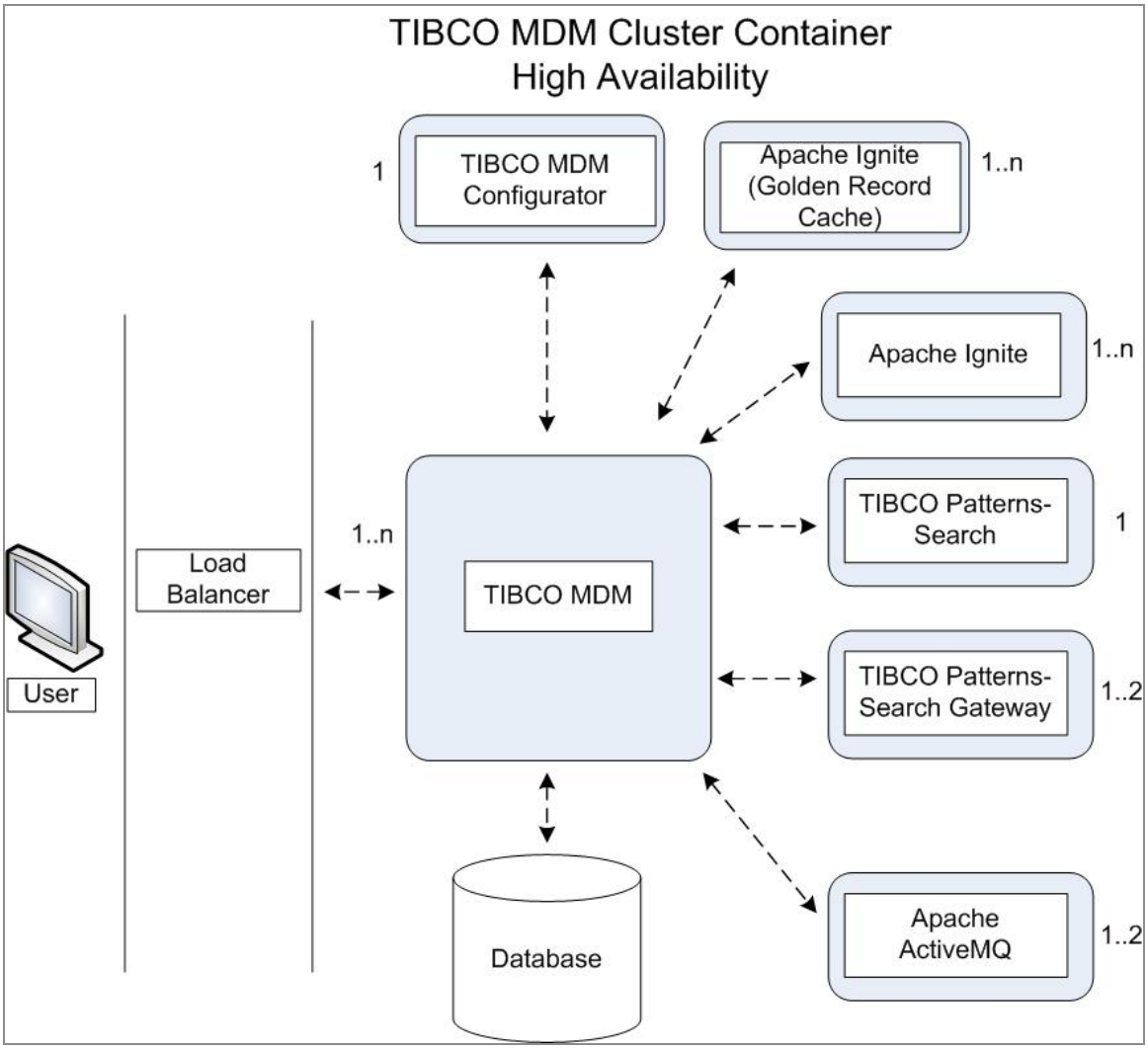
**Tip:** If you prefer a video on the *Clustering Setup* within container platforms, visit https://youtu.be/wUCx5mHPM8M.

TIBCO MDM cluster consists of the following containers: TIBCO MDM, TIBCO MDM Configurator, Apache ActiveMQ, Apache Ignite, Apache Ignite FC (Fast Cache or Golden Record Cache), and TIBCO Patterns - Search. For the supported versions of these containers, see the *Readme.txt* file of TIBCO MDM. For Apache Ignite FC (Fast

Cache or Golden Record Cache), you must set the MQ_MDM_FAST_CACHE_ENABLED property value to true in the Configuration.yaml file located at docker/k8s_deployment.

You can scale up or down TIBCO MDM Server without user request being intercepted. Therefore, TIBCO MDM cluster container can be configured for high availability environment. You can use the TIBCO MDM cluster in production environment for complex testing and demos.

**Note:** You can scale Apache Ignite based on your requirements and must not scale it down.




The TIBCO MDM cluster supports PostgreSQL, Oracle, and Microsoft SQL Server databases. TIBCO Patterns - Search and Apache ActiveMQ servers can also be scaled as primary and secondary backup servers. You can deploy TIBCO Patterns - Search

single server and TIBCO Patterns - Search with Gateway server. See k8s_deployment/DeploymentInstructions.txt or k8s_helm/Readme.txt (applicable if deploying using Helm chart).

Kubernetes is required to run the TIBCO MDM cluster. Kubernetes is an orchestration engine for managing containerized applications across multiple hosts providing basic mechanisms for deployment, maintenance, and scaling of applications. For more information about Kubernetes, see Kubernetes Documentation.

# Build Docker Image for TIBCO MDM Cluster Container

Build the Docker images for the components included in the TIBCO MDM cluster. The steps to build Docker images for TIBCO MDM cluster components are documented in the ReadMe.txt file available in each of the component directory. Before you build the Docker image, consider the following points:

- Ensure that you have a Dockerfile for each component for which you are creating the Docker image. See Dockerfile Locations for Cluster Components.
- Create TIBCO MDM database schema. See the ReadMe.txt file located at docker/build/Mdm and *TIBCO MDM Installation and Configuration*.
- Ensure that you have the mdmbase:9.3.2.latest image built. To build the mdmbase:9.3.2.latest Docker image, see the ReadMe.txt file located at docker/build/MdmBase.
  **Important:** You must first build the mdmbase:9.3.2.latest Docker image before you create the Apache Ignite, TIBCO Patterns - Search, Apache ActiveMQ , TIBCO MDM and TIBCO MDM Configurator Docker images.
- For faster performance, configure memory of Docker container. See Configuring Memory of Docker Container.

# Dockerfile Locations for Cluster Components

*See the following table for the Dockerfile and readme location for each component:*

| Component Name | Dockerfile and Readme Location |
|---|---|
| TIBCO MDM | docker/build/Mdm |
| TIBCO MDM Base | docker/build/MdmBase |
| Apache Ignite | docker/build/Ignite |
| Apache ActiveMQ | docker/build/ActiveMQ |
| TIBCO MDM Configurator | docker/build/MdmConfig |
| TIBCO Patterns - Search | docker/build/Patterns |
| TIBCO Patterns - Search Gateway | docker/build/PatternsGateway |

# TIBCO MDM Cluster Container Components YAML Files

All YAML files are located at docker/k8s_deployment.

The YAML files define the Kubernetes objects that are required for deployment. You can update YAML files and deploy objects to the cluster to change configuration. Use YAML files to configure Kubernetes resources such as pods, services, and deployments.

| Kubernetes Objects | Description |
|---|---|
| ConfigMap (Configuration.yaml) | A config map stores configuration data for containers. Config map separates out configurations from your Pods and components. It is easier to change and manage config maps, without hardcoding configuration data to Pod specifications.<br><br>For the parameters available in the ConfigMap file, see ConfigMap |

| Kubernetes Objects | Description |
|---|---|
| | Parameters. |
| Secrets ( Secrets.yaml) | Secrets are objects, which stores sensitive information about your clusters such as database user name and password in the encrypted format. |
| Deployments | Deployment object consists of specification for Pods and defines ReplicaSets. If one of the instances of your application fails, it is replaced by another replica without user request being affected.<br>• TIBCO MDM: Mdm.yaml<br>• TIBCO MDM Configurator: MdmConfigurator.yaml<br>• Apache Ignite: IgniteCache.yaml<br>• Apache ActiveMQ: ActiveMQ.yaml<br>In the deployment files, you can update values of replicas, image name, and memory.<br>**Warning:** Do not change any other parameters.<br>If you are deploying through Helm chart, see the YAML files located in each of the charts directory (docker/k8s_ helm/mdm/charts/*component*/templates). |
| Volumes | Docker volume files are used for persisting data.<br>• TIBCO MDM: MdmVolumes.yaml<br>• TIBCO Patterns - Search: PatternsVolumes.yaml |
| StatefulSets | • TIBCO Patterns - Search: Patterns.yaml.<br>◦ For OpenShift: patterns_Storage_minishft.yaml<br>The storage required for the checkpoint or restore feature of TIBCO Patterns - Search.<br>• TIBCO Patterns - Search Gateway: |

| Kubernetes Objects | Description |
| --- | --- |
| | PatternsGatewayPrimaryServers.yaml and PatternsGatewayBackupServers.yaml.<br>Perform the steps listed in the DeploymentInstructions.txt file. |
| Services | Service defines a logical set of Pods and a policy to access these pods.<br><br>You must create services for TIBCO MDM UI, TIBCO MDM Configurator UI, Apache ActiveMQ UI, Apache Ignite, TIBCO Patterns-Search, and Apache ActiveMQ.<br><br>You must create headless services for the following components:<br>• TIBCO MDM: Mdm.yaml<br>• TIBCO MDM Configurator: MdmConfigurator.yaml<br>• Apache Ignite: IgniteCache.yaml<br>• Apache Ignite for gloden record cache: IgniteFastCache.yaml<br>• TIBCO Patterns - Search: Patterns.yaml<br>◦ Primary node (PatternsPrimaryServers.yaml)<br>◦ Backup node (PatternsBackupServers.yaml)<br>• TIBCO Patterns - Search Gateway<br>◦ Primary gateway node (PatternsGatewayPrimaryServers.yaml)<br>◦ Backup gateway node (PatternsGatewayBackupServers.yaml)<br>• Apache ActiveMQ: ActiveMQ.yaml |

# Configuring Kubernetes Containers for TIBCO MDM

The YAML configuration files contain the configuration details for deployment.
For more information about the Kubernetes concepts, see Kubernetes Documentation.

### Before you begin

For information about the YAML file configurations of TIBCO MDM cluster components, see TIBCO MDM Cluster Container Components YAML Files.

### Procedure

1. Create the following Kubernetes objects that are required for deploying TIBCO MDM cluster. These objects include required Kubernetes objects and services for the cluster:
    - Namespace
    - Rolebinding
    - Config Map
    - Secret
    - Kubernetes objects for TIBCO MDM, TIBCO MDM Configurator, Apache Ignite, Apache ActiveMQ, and statefulset objects for TIBCO Patterns - Search.
    - Services for TIBCO MDM UI, TIBCO MDM Configurator UI, Apache ActiveMQ UI, and headless services for TIBCO MDM, Apache Ignite, TIBCO Pattern - Search, and Apache ActiveMQ.
2. On the command line, using the kubectl create command, deploy TIBCO MDM Cluster components using YAML files. For details, see the DeploymentInstructions.txt file in docker/k8s_deployment.

### Result

Kubernetes containers are successfully created.

# ConfigMap Parameters

You can update the ConfigMap parameters for YAML files and Helm chart based on the configuration that you are using.

- For the YAML files, update the parameters in the Configuration.yaml file located at docker/k8s_deployment.
- For the Helm chart, update the parameters in the values.yaml file located at docker/k8s_helm/mdm.

The following table lists the parameters available in the ConfigMap file, their definitions and example values:

| Parameter Name | Definition |
| --- | --- |
| MQ_MDM_DB_TYPE | Database type<br>Example: POSTGRESQL, ORACLE, and SQLSERVER |
| MQ_MDM_DB_HOST | Database server host name<br>Example: pgsqldb-eks.cx4wjme9qqns.us-west-2.rds.amazonaws.com |
| MQ_MDM_DB_PORT | Database port<br>Example:<br>• 5432 for POSTGRESQL<br>• 1521 for ORACLE<br>• 1433 for SQLSERVER |
| MQ_MDM_DB_NAME | Database name<br>Example: velodb |
| MQ_MDM_DB_USETABLESPACES | Set this value to true, to set your own tablespace location. For example, Microsoft Azure SQL and on-premise database setup.<br>Set this value to false for cloud platforms where you do not know the tablespace location. For example, AWS and GCP. |
| MQ_MDM_DB_MIN_CONN_COUNT<br>MQ_MDM_DB_MAX_CONN_COUNT | Minimum and maximum number of database connections for TIBCO MDM database. |
| MQ_MDM_DB_FLUSH_STRATEGY | Specifies how the pool must be flushed if an error occurs. By default, the value is set to FailingConnectionOnly, which forces destroying connections with error. |

| Parameter Name | Definition |
|---|---|
| MQ_MDM_DB_BLOCKING_TIMEOUT | Specifies the length of time required to wait for a connection that is available when all the connections are checked out. By default, the value is set to 300000, that is 30 seconds. |
| MQ_MDM_DB_IDLE_TIMEOUT | Indicates the number of minutes after which unused connections are closed. By default, the value is set to 4minutes. |

**Note:** MQ_MDM_DB_FLUSH_STRATEGY, MQ_MDM_DB_BLOCKING_TIMEOUT, and MQ_MDM_DB_IDLE_TIMEOUT are required only for the JBoss WildFly application server. For the supported valid values, see WildFly documentation.

| | |
|---|---|
| Add the following properties for TIBCO Enterprise Message Service™ support: | |
| MQ_MDM_JMS_CLUSER_TYPE | Set the cluster type. For example, GenericCluster or TIBCOCluster. |
| MQ_EMS_CLUSTER_URL | Set the cluster URL. For example, tcp://localhost:7222. |
| MQ_MDM_FAST_CACHE_ENABLED | To enable golden record cache, set this value to true. Else, set it to false. |
| MQ_MDM_HTTP_SESSION_REPLICATION_ENABLED | By default, the value is set to **true** to use multiple nodes and replicate the session. |
| MQ_ACTIVEMQ_COMPONENT_IDS | The name by which TIBCO MDM server discovers other services<br>Example: MQ_ACTIVEMQ_COMPONENT_IDS: activemq.*namespace*.svc.cluster.local |
| MQ_PATTERNS_COMPONENT_IDS | Example: MQ_PATTERNS_COMPONENT_IDS: patterns.*namespace*.svc.cluster.local<br>For TIBCO Patterns - Search gateway: patternsgateway.namespace.svc.cluster.local |

| Parameter Name | Definition |
|---|---|
| MQ_IGNITE_COMPONENT_IDS | Example: MQ_IGNITE_COMPONENT_IDS: ignite.*namespace*.svc.cluster.local |
| MQ_MDM_COMPONENT_IDS | Example: MQ_MDM_COMPONENT_IDS: mdm.*namespace*.svc.cluster.local |
| MQ_TZ | Set the time zone before you deploy containers in Kubernetes. All containers including the database must be in the same time zone. By default, the time zone is set to Asia/Kolkata. |
| MQ_PATTERNS_PRIMARYE<br>MQ_PATTERNS_BACKUP | To deploy a TIBCO Patterns - Search server with primary and backup nodes. Examples:<br>• patternsprimary.development.svc.cluster.local<br>• patternsbackup.development.svc.cluster.local |
| MQ_PATTERNS_PRIMARY_ GATEWAY<br>MQ_PATTERNS_BACKUP_GATEWAY | To deploy TIBCO Patterns - Search Gatway. Examples:<br>• patternsprimarygateway.development.svc.cluster.local<br>• patternsbackupgateway.development.svc.cluster.local |
| **For TIBCO MDM REST API through Swagger UI** | |
| SWAGGER_MDM_HOST | An IP address of the TIBCO MDM server on which you want to try out TIBCO MDM REST APIs. |
| SWAGGER_MDM_PORT | Port of the TIBCO MDM server on which you want to try out TIBCO MDM REST APIs. |

# Configuring Memory of Docker Container

You can increase the memory of a Docker container to get faster performance. Before building the Docker image, you must change the valid values in the entry-point.sh file in each component directory located at docker/build/.

To change mdm Docker container, perform the following steps:

1. Navigate to docker/build/Mdm/ directory and open the entry-point.sh file.

2. Add or modify any valid arguments for the JAVA_OPTS parameter, for example, The **Xms** default value is **512m**.

```
JAVA_OPTS="-Xms512m -Xmx${mem_in_mb}m -XX:MetaspaceSize=96M -XX:MaxMetaspaceSize=512m -
Dfile.encoding=UTF-8 -Djdk.tls.client.protocols=TLSv1.2 -Djava.net.preferIPv4Stack=true -
Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true -Djboss.as.management.blocking.timeout=3600
-XX:ErrorFile=${ERROR_FILENAME}-java-$$ -XX:HeapDumpPath=${HEAP_DUMP_PATH}" $@
```

# Configure Memory and CPU for Kubernetes Pod

You can set limits for CPU and RAM use for Kubernetes Pod.

| Description | File Location |
|---|---|
| For Kubernetes Pod, modify the values in the deployment file. | docker/k8s_deployment |
| If you have used a helm chart to deploy TIBCO MDM, modify the cpu and memory values in the YAML file. | docker/k8s_helm/mdm/values.yaml<br>For other components, the values.yaml file is located under each subchart of the mdm directory. |

# Configuration for Persistent Volume Types

You can configure the PersistentVolume (PV) storage in the cluster. To use a different storage option for deployment with Helm, update the storage name and volumeType in the following files:

- mdm/values.yaml
- mdm/charts/patterns/values.yaml
- mdm/charts/patternsgateway/values.yaml

## PersistentVolume for Static Provisioning Type

Create PersistentVolume for each PersistentVolumeClaim. To use static provisioning, see the following files:

- DeploymentInstructions.txt (located at docker/k8s_deployment): For information about the volumes in StatefulSets, see mdm_pv.yaml and mdm_pvc.yaml files.
- ReadMe.txt (for Helm chart) (located at docker/k8s_helm/Samples): Set dynamicProvisioning:false in the values.yaml file.

**Note:** TIBCO MDM is supported with NoFS=false. Therefore, you must create the PersistentVolume for the static provisioning type with the ReadWriteMany access mode.

## PersistentVolume for Dynamic Provisioning Type

For the dynamic provisioning type, Persistent Volumes are created automatically. For Helm deployment, by default dynamic provisioning is set for Azure file, Azure disk, and AWS EBS.

**Note:** For the awsefs volume type, you must deploy aws-efs-csi-driver by using the following command:

```
- helm install aws-efs-csi-driver https://github.com/kubernetes-sigs/aws-efs-csi-driver/releases/download/v0.3.0/helm-chart.tgz
```

## Supported Provisioning Types

The following table lists the supported provisioning types and their storage class and cloud provider:

| Provisioning Types | Storage Class | Cloud Provider |
|---|---|---|
| Static/Dynamic | Local | None |
| | AWS EFS | AWS |
| | AWS EBS | AWS |
| | Azure File | AZURE |
| | Azure Disk | AZURE |
| | GCP PD | GCP |
| Network File System (NFS)<br>Mounting happens with the host machine. For example,<br>`### NFS Mount ###`<br>`nfs:`<br>`path: "/var/nfs/general"`<br>`server: "`*IPaddress*`"` | Local | None |

## Supported VolumeTypes and Provisioner Values

The following table lists the supported VolumeType and Provisioner values based on the storage and cloud provider:

| Provisioning types | Changes | | |
|---|---|---|---|
| Static/Dynamic | **Volume type** | **Provisioner** | **Attribute Changes** |
| | host | kubernetes.io/no-provisioner | hostProvisioner |
| | awsefs | efs.csi.aws.com | efsProvisioner |

| Provisioning types | Changes |
|---|---|

| Volume type | Provisioner | Attribute Changes |
|---|---|---|
| awsebs | kubernetes.io/aws-ebs | ebsProvisioner |
| azureFile | kubernetes.io/azure-file | azureFileProvisioner |
| azureDisk | kubernetes.io/azure-disk | azureDiskProvisioner |
| gke | kubernetes.io/gce-pd | gkeProvisioner |
| nfs | nfsProvisioner: path: "/mqvol" server: *IPAddress* | nfsprovisioner |

**Note:** Based on your volume type, configure parameters related to the specified attribute. For example, if you select the awsefs volume type, then you can add **efsFileSystemId** in the efsProvisioner attribute.

# Setting up Helm

Helm is the application package manager that you can run on top of Kubernetes. Helm uses packaging format called charts. By using helm charts, you can deploy TIBCO MDM to run in Kubernetes.

### Before you begin

1. Install Helm

   **Note:** When you obtain third-party software or services, it is your responsibility to ensure you understand the license terms associated with such third-party software or services and comply with such terms.

2. Add the helm path in the system *PATH* variable

3. Ensure that Docker and Kubernetes cluster are running

### Procedure

1. Navigate to the docker/k8s_helm/mdm directory and update each values.yaml file of chart and subcharts.
   For example, replace the Docker image registry with the URL of container registry. Additionally, update the tag name, service type, cloud provider, and so on. For information, see Configuration for Persistent Volume Types.

   > **Warning:** You must have a container registry created with the deployed images. If you have not created it earlier, see Create Container Registry.

   For other components, the values.yaml file is located under each subchart of the mdm directory.
2. Update the database credentials (Base64 encoded format) in the values.yaml file.
3. Update all other database values in the values.yaml file. For information, see ConfigMap Parameters.

# TIBCO MDM Cluster on the Cloud

Navigate to the extracted TIB_mdm_9.3.2_container directory and locate the files and relevant directories to deploy TIBCO MDM cluster on Microsoft Azure, Amazon EKS, Google Cloud Platform, and Red Hat OpenShift Container Platform. You can run the dockerized TIBCO MDM application in a Kubernetes cluster on the cloud platform of your choice.

## Supported Versions

Before you begin, see TIBCO MDM Readme for supported versions of Docker and cloud platforms.

## Concepts

Before you begin, you must be familiar with the Administration knowledge of the cloud platform and the service that you want to use:

- Amazon Web Services (AWS) and Amazon Elastic Kubernetes Service (EKS)

- Microsoft Azure and Azure Kubernetes Service (AKS)
- Google Cloud Platform (GCP)
- OpenShift Container Platform

To run the application in a Kubernetes cluster on a cloud platform, ensure that you have an active account on that cloud platform.

# Running TIBCO MDM on Microsoft Azure Based Kubernetes Cluster

By using Azure Kubernetes Service (AKS), you can easily deploy TIBCO MDM to a Kubernetes cluster managed by Microsoft Azure.
For more details about the AKS, see Azure Kubernetes Service documentation.

### Before you begin

- Build docker image of the TIBCO MDM application, see Build Docker Image for TIBCO MDM Cluster Container.
- You must have a Microsoft Azure account with active subscription. If required, create an Azure account.

### Procedure

1. Set up Microsoft command-line interface (Azure CLI) environment. See Setting Up the Azure CLI Environment.
2. Create an Azure Container Registry (ACR) and push the Docker images of the application to it. See Setting Up an Azure Container Registry.
3. Create a Kubernetes cluster and deploy it to the Microsoft Azure. See Setting Up a Kubernetes Cluster on AKS.
4. Create SQL Server on Azure virtual machine. See Prerequisites for Deploying TIBCO MDM on Microsoft Azure.
5. Based on your application architecture, deploy the application on the Kubernetes cluster.

# Setting Up the Azure CLI Environment

You can use either the Microsoft Azure portal or Azure CLI to run Microsoft Azure commands. In the following sections, the procedures are provided for the Azure CLI.

### Before you begin

You must have a Microsoft Azure account with active subscription. If required, create an Azure account.

### Procedure

1. Install the Azure CLI. For installation instructions, see Microsoft Azure CLI documentation
2. In the CLI, sign in to Microsoft Azure by using the login command.

   ```
   az login
   ```

   The CLI opens a browser and loads the sign-in page.
3. Sign in with your account credentials in the browser.

   For details, see Get started with Azure CLI

### What to do next

Create an Azure Container Registry (ACR) and push the Docker image of the application to it, see Setting Up an Azure Container Registry.

# Setting Up an Azure Container Registry

Microsoft Azure uses the Azure Container Registry for securely storing docker images of your application. To create an Azure Container Registry, you must create an Azure Resource group. An Azure resource group is a logical grouping in which Azure resources are deployed and managed.
For more information about commands used in the following procedure, see Microsoft Azure CLI documentation.

### Before you begin

- Set up the Azure CLI environment. See Setting Up the Azure CLI Environment.
- Ensure that you have built Docker images of the TIBCO MDM application that you want to deploy to the Kubernetes cluster. See Build Docker Image for TIBCO MDM Cluster Container.

### Procedure

1. Create a resource group.
   - **Using the Azure Portal:** See Create resource groups.
   - **Using the Azure CLI:** Run the following command:

     ```
     az group create --name ResourceGroupName --location RegionName
     ```

     For example, create a new resource group in the East US region.

     ```
     az group create --name mdmresourcegroup --location eastus
     ```

2. Create virtual network by using the Azure portal and ensure that all resources are created under the same virtual network.
3. Create a new managed Kubernetes cluster.
   - **Using the Azure Portal:** See Create an AKS cluster.
   - **Using the Azure CLI:** Run the following command:

     ```
     az aks create -g ResourceGroupName -n resourcegroupclustername --location
     RegionName --node-vm-size size --node-count nodecount --generate-ssh-keys
     ```

     For example, create a kubernetes cluster with the default vm size (Standard) and default node pool.

     ```
     az aks create -g mdmresourcegroup -n mdmcluster --location eastus --node-vm-size
     Standard_DS3_v2 --node-count 3 --generate-ssh-keys
     ```

4. Create an Azure Container Registry instance in your resource group by using the `az acr create` command.
   - **Using the Azure Portal:** See Create a container registry.
   - **Using the Azure CLI:** Run the following command:

```
az acr create --resource-group ResourceGroupName --name RegistryName --sku Basic
```

For example, create a managed container registry with the Basic SKU.

```
az acr create --resource-group mdmresourcegroup --name mdmregistry --sku Basic
```

5. Log in to the container registry created earlier.
   - **Using the Azure Portal:** See Log in to registry.
   - **Using the Azure CLI:** Run the following command:

     ```
     az acr login --name acrName
     ```

     For example, log in to the specified Azure Container Registry.

     ```
     az acr login --name mdmregistry
     ```

     The command returns a Login Succeeded message after completed.

### What to do next

1. Push the application image to the registry. See Tag and Push the Docker Image for Cloud Platforms.
2. Deploy the Kubernetes cluster on Microsoft Azure. See Setting Up a Kubernetes Cluster on AKS

## Setting Up a Kubernetes Cluster on AKS

Azure Kubernetes Services (AKS) manages the Kubernetes environment and provides options to quickly deploy Kubernetes cluster. To enable a Kubernetes cluster to interact with other Azure resource, perform the steps mentioned in this section.

### Before you begin

Set up the Azure Container Registry and push the application Docker images to it, see Setting Up an Azure Container Registry.

### Procedure

1. Create AKS Cluster through portal or CLI

> **Note:** Make a note of the Resource Group and Cluster Name parameters.

2. Run the login command.

```
az login
```

3. To set trust relationship between AKS cluster and ACR, run the following command:

```
az aks update -n myAKSCluster -g MyResourceGroup --attach-acr MyACRRegistry
```

For example, attach AKS cluster to ACR by name mdmregistry

```
az aks update -n mdmcluster -g mdmresourcegroup --attach-acr mdmregistry
```

4. Configure kubectl to connect your Kubernetes cluster by using the az aks get-credentials command.

```
az aks kubernetes get-credentials --resource-group <resource_group_name> --name=<cluster_name>
```

For example, get the access credentials for your cluster.

```
az aks get-credentials --resource-group mdmresourcegroup --name  mdmcluster
```

5. View the dashboard for a Kubernetes cluster in a web browser.

```
az aks browse --resource-group myResourceGroup --name myAKSCluster
```

For example,

```
az aks browse --resource-group mdmresourcegroup --name mdmcluster
```

6. Create binding for kubectl (Kubernetes command-line tool ) cluster role

```
kubectl create clusterrolebinding kubernetes-dashboard --clusterrole=cluster-admin --
```

```
serviceaccount=kube-system:kubernetes-dashboard
```

7. Verify the connection to your Kubernetes cluster by using the kubectl get nodes command.

```
kubectl get nodes
```

### What to do next

Deploying TIBCO MDM Cluster on Kubernetes by Using YAML Files

## Prerequisites for Deploying TIBCO MDM on Microsoft Azure

### Procedure

1. Create a SQL Server virtual machine.
   - **Using the GUI:** See Create a SQL Server VM in the Azure portal.
   - **Using the CLI:** See Create a SQL virtual machine.
2. In the networking section of a virtual machine, open inbound ports for client IP to connect to virtual machine by using SSH or the remote desktop connection. For example, for the SQL Server database, open 1433.

### What to do next

See Deploying TIBCO MDM Cluster by Using Helm

## Running TIBCO MDM on Amazon EKS Based Kubernetes Cluster

By using the Amazon Elastic Kubernetes Service (EKS), you can easily deploy a TIBCO MDM application in the Kubernetes cluster managed by Amazon.
For more details about the EKS, see Amazon Elastic Kubernetes Service documentation.

### Before you begin

- Install and configure kubectl (Kubernetes command-line tool)
- Install the latest AWS Command Line Interface (AWS CLI)

### Procedure

1. Create your Amazon EKS cluster IAM role
2. Create a Virtual Private Cloud (VPC) for your Amazon EKS cluster
3. Create Amazon EKS Cluster
4. Create a kubeconfig file for Amazon EKS with the AWS CLI
5. Create your Amazon EKS worker node role using AWS Cloud Formation
6. Launch your managed node group

### What to do next

See Deploying TIBCO MDM Cluster by Using Helm

# Running TIBCO MDM on Google Cloud Platform

By using Google Cloud Platform (GCP) with Google Kubernetes Engine (GKE), you can easily deploy a TIBCO MDM application in the Kubernetes cluster.

For more details about GCP, see Google Cloud documentation and Google Kubernetes Engine documentation.

### Procedure

1. Setting Up GCP on GKE
2. Deploying TIBCO MDM Cluster by Using Helm
3. Access TIBCO MDM Cluster UI

# Setting Up GCP on GKE

This section provides detailed steps to set up Google Cloud Platform (GCP) with Google Kubernetes Engine (GKE).

### Before you begin

1. In the Cloud Console, on the project selector page, select or create a Cloud project.
2. Ensure that billing is enabled for your Google Cloud project.
3. Install and configure GCP Cloud SDK.
4. Install Helm.

> **Note:** When you obtain third-party software or services, it is your responsibility to ensure you understand the license terms associated with such third-party software or services and comply with such terms.

5. Enable the following APIs on the Cloud Console:
   - Container Registry API
   - GKE API

### Procedure

1. Create a custom mode network by using console or gcloud.
2. Create a Virtual Private Cloud (VPC) network.
   Use the VPC network for all objects to be created for Google Cloud Platform.
3. Create Cloud SQL for PostgreSQL. After entering a password for the PostgreSQL user, perform the following steps:
   a. Select region and database version.
   b. In the Configuration options > Connectivity section,
      i. Select the **Private IP** check box.
      ii. In the Associated Networking drop-down list, select the previously created vpc network to create a private connection in vpc network.
      iii. Select the **Public IP** check box.
      iv. In the Authorized networks section, click **Add Network** and enter your client IP. The connection is required to connect from your network for database schema installation.
   c. Click **Create**.

4. Create Container Registry.
5. Create a regional cluster. After entering the boot disk size, perform the following steps:
    a. In the Node Security section, select the **Allow full access to all Cloud APIs** option for smooth communication.
    b. In the Networking section, select the **Public cluster** option.
    c. From the Network list, select the previously created cluster.
    d. Click **Create**.

**Note:** You can select different types of clusters. For details, see GCP documentation.

### What to do next

See Deploying TIBCO MDM Cluster by Using Helm

## Running TIBCO MDM on OpenShift Based Kubernetes Cluster

By using Red Hat OpenShift Container Platform, you can deploy a TIBCO MDM application in the Kubernetes cluster. As OpenShift Container Platform is built on top of Kubernetes cluster, you are not required to install Kubernetes separately. For details about OpenShift Container Platform, see OpenShift Container Platform documentation.

### Before you begin

Build docker image of the TIBCO MDM application, see Build Docker Image for TIBCO MDM Cluster Container.

### Procedure

1. Set up PostgreSQL. See Setting up PostgreSQL in OpenShift.
2. Based on your application architecture, deploy the application on the Kubernetes cluster. See Deploying TIBCO MDM on OpenShift
3. Create routes. See Creating Routes.

# Setting up PostgreSQL in OpenShift

You can connect to an external PostgreSQL database.

1. Create a service, for example:

```
kind: "Service"
    apiVersion: "v1"
    metadata:
    name: "external-postgres-service"
    spec:
    ports:
    -
    name: "external-postgres-service"
    protocol: "TCP"
    port: 5432
    targetPort: 5432
    nodePort: 0
    selector: {}
```

2. Create endpoint, for example:

```
kind: "Endpoints"
    apiVersion: "v1"
    metadata:
    name: "external-postgres-service"
    subsets:
    -
    addresses:
    -
    ip: "10.97.90.56"
    ports:
    -
    port: 5432
    name: "external-postgres-service"
```

3. In the mdmconfigmap, change the value of the MQ_MDM_DB_HOST property to **external-postgres-service** or add the IP address of external PostgreSQL database.

# Deploying TIBCO MDM on OpenShift

### Before you begin

Check the CPUs and the memory before importing the .yaml files.

### Procedure

1. Log in to the OKD console.
2. Create a new project. For example, Development.
3. In the Development project, expand **Add to Project** and select the **Import YAML/JSON** option.
4. In the **Import YAML/JSON** window, upload the YAML files by dragging and dropping or by clicking **Browse** and selecting the files. For a list of YAML files, see TIBCO MDM Cluster Container Components YAML Files.

### What to do next

1. Verify the configuration at the following locations:

| Files | Location |
|---|---|
| mdmconfigMap | **Resources** > **Config Maps** |
| mdmsecret | **Resources** > **Secrets** |
| Apache Ignite, Apache ActiveMQ, TIBCO MDM, and TIBCO MDM Configurator | **Applications** > **Deployments** |
| Apache Ignite, Apache ActiveMQ, TIBCO MDM, and TIBCO MDM Configurator | **Applications** > **Services** |

   For troubleshooting, go to **Applications** > **Logs** and check the logs for the TIBCO MDM pod and ensure that the application is working.
2. Create routes. See Creating Routes.

## Creating Routes

To access TIBCO MDM and the Configurator, you must create routes. Routing makes your application publicly visible.

1. In the OKD console, go to **Applications** > **Services** > **MDM** or **Applications** > **Services** > **MDM Config**.

2. Click **Create route**.
3. In the Create Route window, you can enter values or keep it blank. If you leave the fields empty, routes are created by using the default values.

What to do next

1. Go to **Applications** > **Routes**, verify the newly created routes.
2. To access TIBCO MDM and the Configurator, click the URL in the **Hostname** field. For more information, see Access TIBCO MDM Cluster UI.

# Tag and Push the Docker Image for Cloud Platforms

| Cloud Platform | Steps |
|---|---|
| AWS EKS | 1. Retrieve an authentication token and authenticate your Docker client to your registry.<br>For example:<br>`aws ecr get-login-password --region us-west-2 \| docker login --username AWS --password-stdin AWS_ACCOUNT_ID.dkr.ecr.us-west-2.amazonaws.com/mdmc/mdm`<br>2. Tag your images to push to ECR Registry<br>`docker tag mdmc/mdm:latest AWS_ACCOUNT_ID.dkr.ecr.us-west-2.amazonaws.com/mdmc/mdm:latest`<br>3. Run the following command to push this image to your newly created AWS repository:<br>`docker push 061574984669.dkr.ecr.us-west-2.amazonaws.com/mdmc/mdm:latest` |
| Microsoft | 1. To use the TIBCO MDM application container images with Azure |

| Cloud Platform | Steps |
|---|---|
| Azure | Container Registry, tag the image with the login server address of your registry.<br>a. View the list of your local image by using the docker images command.<br>`$ docker images`<br>`REPOSITORY TAG IMAGE ID CREATED SIZE`<br>`mdmc/ignite *version*.latest 18763f0d1403 3 weeks ago 790MB`<br>b. Get the login server address for the Azure Container Registry by using the az acr list command.<br>`az acr list --resource-group *resource_group_name* --query "[]. {acrLoginServer:loginServer}" --output table`<br>c. Tag your application image with login server address of your registry from the earlier step. This creates an alias of the application image with a fully qualified path to your registry.<br>`docker tag mdmc/ignite:*version*.latest mdmtestregistry.azurecr.io/mdmc/ignite:*version*.latest`<br>d. Push the application image to your container registry by using the docker push command.<br>`docker push mdmtestregistry.azurecr.io/mdmc/ignite:*version*.latest`<br>e. Push for all other TIBCO MDM cluster images by using the docker push command. |

# Deploying TIBCO MDM Cluster on Kubernetes by Using YAML Files

To work with TIBCO MDM on the containerization platform, deploy TIBCO MDM cluster on Kubernetes by using the YAML files.

### Before you begin

- See ConfigMap Parameters
- For Microsoft Azure, See Prerequisites for Deploying TIBCO MDM on Microsoft Azure

### Procedure

1. Navigate to the docker/k8s_deployment directory.
2. Update the Secrets.yaml file with the database credentials (Base64 encoded format).
3. Update Docker image registry names in each deployment and statefulset.yaml file.
4. Create the Kubernetes objects and update the YAML files that are required for deploying TIBCO MDM cluster. For information, see Configuring Kubernetes Containers for TIBCO MDM.

### What to do next

Use the IP obtained to connect to TIBCO MDM from your browser or open TIBCO MDM in a web browser by using the external endpoint URLs in the Azure portal. For information, see Access TIBCO MDM Cluster UI.

# Deploying Kubernetes Dashboard (Optional)

You can deploy Kubernetes cluster and access the Kubernetes dashboard.

### Procedure

1. Deploy the Kubernetes Dashboard.
2. Consider the following example for AWS EKS, perform the following actions:
    a. Connect to the Kubernetes Dashboard with the mdm-cluster-admin service account.
        i. Retrieve an authentication token for the mdm-cluster-admin service account.

        Copy the *authentication_token* value from the output. You can use this token to connect to the dashboard.

```
kubectl apply -f mdmeks-admin-service-account.yaml
```

3. On the Kuberenetes Dashboard window, perform the following actions:
    a. Select the **Token** option.
    b. In the **Enter Token** field, copy the *authentication_token* output.
    c. Click **SIGN IN**.

# Deploying TIBCO MDM Cluster by Using Helm

## Before you begin

See Setting up Helm

## Procedure

1. Check the helm version.

```
helm version
```

2. Perform the following tasks for each cloud platform:

| Cloud Platform | Steps |
|---|---|
| Azure | Update the following values (Base64 encoded format) in the values.yaml file located at docker/k8s_helm/mdm.<br>• azurestorageaccountname<br>• azurestorageaccountkey |
| For AWS EKS | a. Update cidr value of VPC in the LoadBalancerSourceRanges section of the values.xml file.<br>b. Update all relevant field values in the values.yaml file. |
| For GCP | On the command line, type the following command to connect to the |

| Cloud Platform | Steps |
|---|---|
| GKE | gcloud GKE cluster:<br>`gcloud auth configure-docker gcloud container clusters get-credentials GKE Cluster Name --region Region --project Project Name` |

3. Run the following command to deploy TIBCO MDM on Kubernetes using helm chart:

```
helm install --name Name_Of_Release Path_Of_MDM_Helm_Chart_Directory --create-namespace --namespace=Kubernets_Namespace
```

Example:

```
helm install mdm ./mdm --create-namespace --namespace=development
```

### Result

The TIBCO MDM application has been successfully deployed.

### What to do next

Access TIBCO MDM Cluster UI

For more information about deploying the Helm chart on Kubernetes cluster, see the ReadMe.txt file located at docker/k8s_helm.

# Access TIBCO MDM Cluster UI

Run the following command to check services with an external IP and the displayed port. By using the get services command, you can get the IP address and port of the exposed services of cluster:

| Services | Command | URL |
|---|---|---|
| TIBCO MDM | kubectl get services | • **For Kubernetes:** |

<table>
<tr><th>Services</th><th>Command</th><th>URL</th></tr>
<tr><td></td><td>mdm-ui</td><td rowspan="3"><ul><li>For TIBCO MDM (mdm-ui): http://ExternalIP:host_port/eml/Login<br><br>For TIBCO MDM Configurator (mdm-mdmconfig): http://ExternalIP:host_port/config/index.html#/login</li><li>For TIBCO MDM ActiveMQ UI (mdm-activemq-ui): http://ExternalIP:host_port/index.html</li></ul><ul><li>For OpenShift: For example:<ul><li>For TIBCO MDM: http://mdm-development.IPAddress/eml/Login</li><li>For TIBCO MDM Configurator: http://mdmconfig-development.IPAdress/config</li></ul></li></ul></td></tr>
<tr><td>TIBCO MDM Configurator</td><td>kubectl get services mdm-mdmconfig</td></tr>
<tr><td>Apache ActiveMQ</td><td>kubectl get services mdm-activemq-ui</td></tr>
</table>

# TIBCO Documentation and Support Services

For information about this product, you can read the documentation, contact TIBCO Support, and join TIBCO Community.

## How to Access TIBCO Documentation

Documentation for TIBCO products is available on the Product Documentation website, mainly in HTML and PDF formats.

The Product Documentation website is updated frequently and is more current than any other documentation included with the product.

## Product-Specific Documentation

Documentation for TIBCO® MDM is available on the TIBCO® MDM Product Documentation page.

## How to Contact Support for TIBCO Products

You can contact the Support team in the following ways:

- To access the Support Knowledge Base and getting personalized content about products you are interested in, visit our product Support website.
- To create a Support case, you must have a valid maintenance or support contract with a Cloud Software Group entity. You also need a username and password to log in to the product Support website. If you do not have a username, you can request one by clicking **Register** on the website.

## How to Join TIBCO Community

TIBCO Community is the official channel for TIBCO customers, partners, and employee subject matter experts to share and access their collective experience. TIBCO Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable

customers to gain full value from TIBCO products. In addition, users can submit and vote on feature requests from within the TIBCO Ideas Portal. For a free registration, go to TIBCO Community.

# Legal and Third-Party Notices

SOME CLOUD SOFTWARE GROUP, INC. (“CLOUD SG”) SOFTWARE AND CLOUD SERVICES EMBED, BUNDLE, OR OTHERWISE INCLUDE OTHER SOFTWARE, INCLUDING OTHER CLOUD SG SOFTWARE (COLLECTIVELY, “INCLUDED SOFTWARE”). USE OF INCLUDED SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED CLOUD SG SOFTWARE AND/OR CLOUD SERVICES. THE INCLUDED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER CLOUD SG SOFTWARE AND/OR CLOUD SERVICES OR FOR ANY OTHER PURPOSE.

USE OF CLOUD SG SOFTWARE AND CLOUD SERVICES IS SUBJECT TO THE TERMS AND CONDITIONS OF AN AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER AGREEMENT WHICH IS DISPLAYED WHEN ACCESSING, DOWNLOADING, OR INSTALLING THE SOFTWARE OR CLOUD SERVICES (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH LICENSE AGREEMENT OR CLICKWRAP END USER AGREEMENT, THE LICENSE(S) LOCATED IN THE “LICENSE” FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE SAME TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.



TIBCO, the TIBCO logo, the TIBCO O logo, FTL, eFTL, and Rendezvous are either registered trademarks or trademarks of Cloud Software Group, Inc. in the United States and/or other countries.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only. You acknowledge that all rights to these third party marks are the exclusive property of their respective owners. Please refer to Cloud SG’s Third Party Trademark Notices (https://www.cloud.com/legal) for more information.

This document includes fonts that are licensed under the SIL Open Font License, Version 1.1, which is available at: https://scripts.sil.org/OFL

Copyright (c) Paul D. Hunt, with Reserved Font Name Source Sans Pro and Source Code Pro.

Cloud SG software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. See the “readme” file for the availability of a specific version of Cloud SG software on a specific operating system platform.

THIS DOCUMENT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. CLOUD SG MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S), THE PROGRAM(S), AND/OR THE SERVICES DESCRIBED IN THIS DOCUMENT AT ANY TIME WITHOUT NOTICE.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND "README" FILES.

This and other products of Cloud SG may be covered by registered patents. For details, please refer to the Virtual Patent Marking document located at https://www.cloud.com/legal.