



# **TIBCO® Product and Service Catalog**

## Cloud Deployment

*Version 5.1.0  
December 2022*



# Contents

---

<b>Contents</b>	<b>2</b>
<b>TIBCO PSC on Container Platforms</b>	<b>3</b>
TIBCO PSC All-in-one Container	4
Building and Running Docker Image for the TIBCO PSC All-in-one Container	5
TIBCO PSC Cluster	6
Build Docker Image for TIBCO PSC Cluster Container	8
TIBCO PSC Cluster Container Components YAML Files	8
Configuring Kubernetes Containers for TIBCO PSC	10
ConfigMap Parameters	11
Configuring Memory of Docker Container	13
Configure Memory and CPU for Kubernetes Pod	14
Configuration for Persistent Volume Types	14
Setting up Helm	17
TIBCO PSC Cluster on the Cloud	18
Tag and Push the Docker Image for Cloud Platforms	26
Deploying TIBCO PSC Cluster on Kubernetes by Using YAML Files	28
Deploying Kubernetes Dashboard (Optional)	29
Deploying TIBCO PSC Cluster by Using Helm	29
Access TIBCO PSC Cluster UI	31
<b>TIBCO Documentation and Support Services</b>	<b>32</b>
<b>Legal and Third-Party Notices</b>	<b>34</b>

# TIBCO PSC on Container Platforms

---

You can containerize TIBCO PSC and run it in a Docker or Kubernetes environment. To containerize TIBCO PSC, you must build and run the Docker images using the Dockerfiles bundled in `TIB_cim-ac_5.1.0_container.zip`.

The Dockerfiles are delivered as a ZIP file on the [TIBCO eDelivery](#) website. Download the `TIB_cim-ac_5.1.0_container.zip` file and extract its content to a separate directory named `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 PSC All-in-one Container](#) and [Build Docker Image for TIBCO PSC 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 PSC containers are available in the following modes:

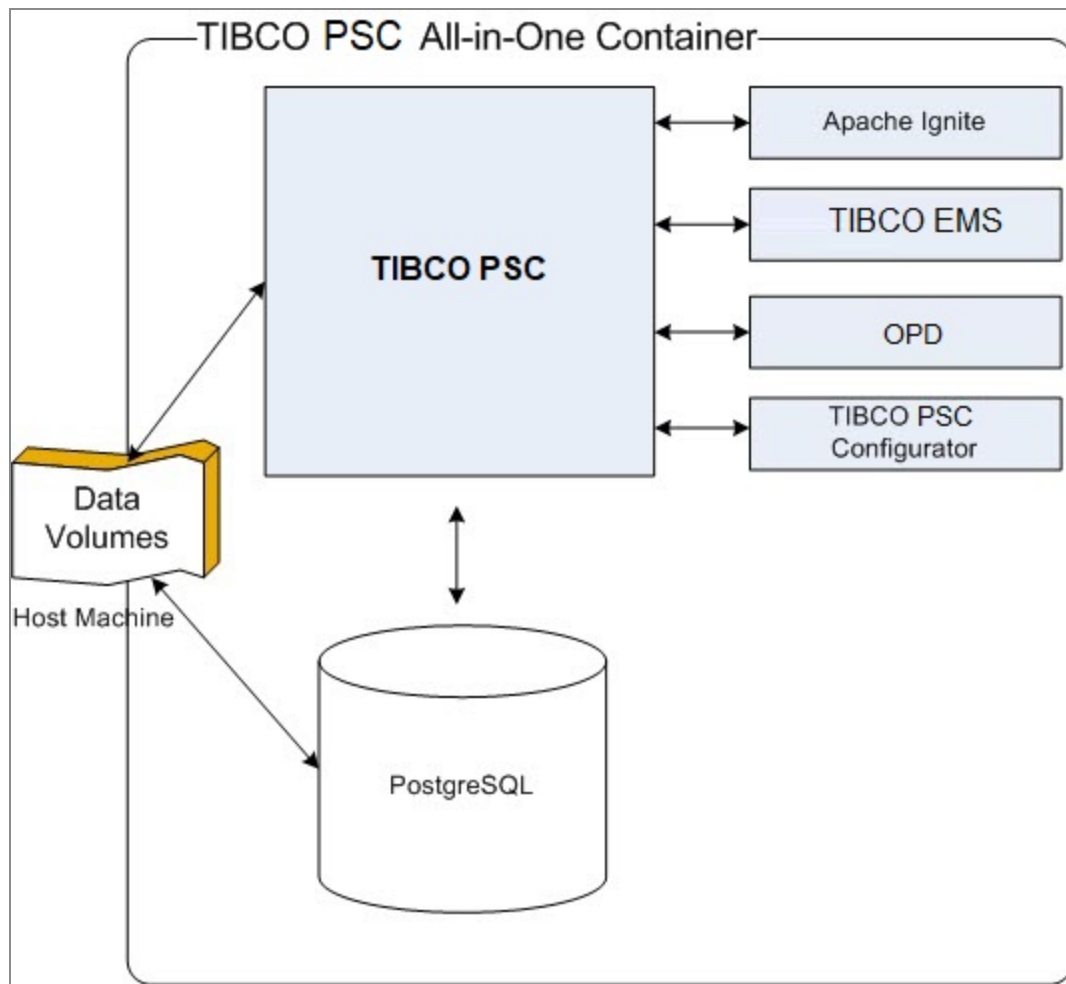
- [TIBCO PSC All-in-one Container](#)
- [TIBCO PSC Cluster](#)

**i Note:** Before you build and run the Docker image of TIBCO PSC all-in-one container, TIBCO PSC Cluster container, and the components included in the TIBCO PSC 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](#).

## TIBCO PSC All-in-one Container

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

You can run the all-in-one container quickly by using only Docker, without complex configurations. You can use the TIBCO PSC 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 PSC 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 PSC all-in-one container supports only the PostgreSQL database.

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

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

## Before you begin

- Ensure that you have Dockerfile for TIBCO PSC all-in-one Container. ReadMe.txt and Dockerfile are available in the build/PSC\_ALLINONE directory.
- Enable squash as an **Experimental** feature through Docker configuration. For more information, see [Docker documentation](#).

## Procedure

1. Build the MDMAllInOne image. (Follow the instructions given in the *TIBCO MDM Cloud Deployment Guide*)
2. Copy the TIBCO PSC installer file (TIB\_cim-ac\_5.1.0\_linux\_x86\_64.zip) to the directory where the Dockerfile is located at build/PSC\_ALLINONE.
3. On the command line, enter the following command:

```
$> docker build -t psc-all-in-one:5.1.0.latest --squash --rm=true
```

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

```
docker volume create --name mdmcommon_psc-all-in-one
docker volume create --name mdmconfig_psc-all-in-one
docker volume create --name mdmdynservices_psc-all-in-one
docker volume create --name mdmdbdata_psc-all-in-one
docker volume create --name postgresdata_psc-all-in-one
```

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

For example:

```
docker run -p 8080:8080 -p 6080:6080 -p 8070:8070 -e MDMPORT=8080 -e PROTOCOL=http \
-v mdmcommon_psc-all-in-one:/home/tibco/mdm/9.3/common \
-v mdmconfig_psc-all-in-one:/home/tibco/mdm/9.3/config \
-v mdmdynservices_psc-all-in-one:/home/tibco/mdm/9.3/dynservices \
```

```
-v postgresdata_psc-all-in-one:/home/tibco/mdm/9.3/bin/pgsql/data \
-v mdmdbdata_psc-all-in-one:/home/tibco/mdm/9.3/bin/pgsql/tablespaces \
psc-all-in-one:5.1.0.latest
```



**Note:** You can specify the 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 PSC apps in the All-in-One container, use the following URLs:

- PSC URL: <http://<hostname>:8080/eml/Login>
- Configurator URL: <http://<hostname>:6080/config/#/login>
- OPD URL: <http://<hostname>:8070/#/login>

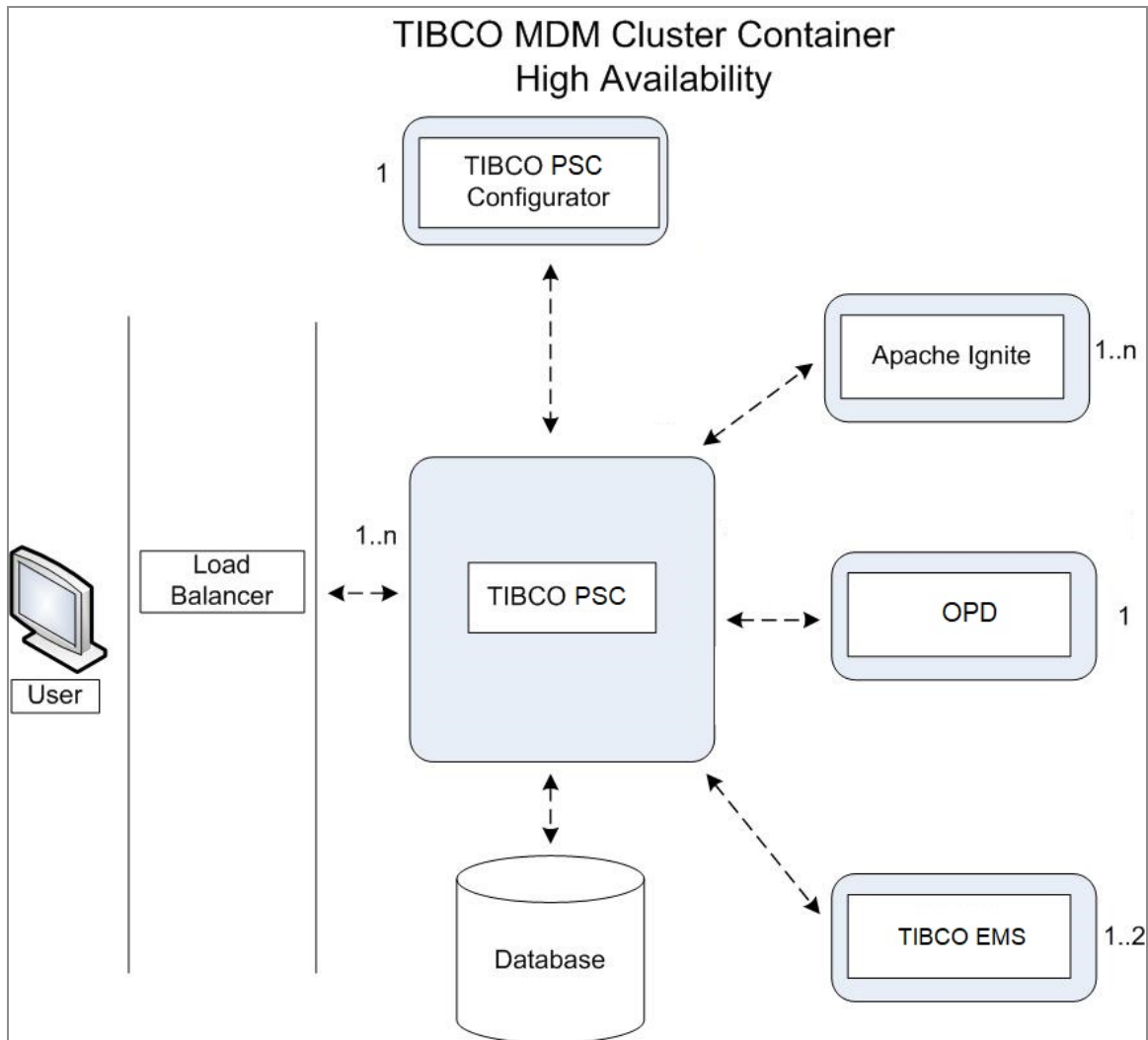
For more information, see `build/PSC_ALLINONE/ReadMe.txt` file.

## TIBCO PSC Cluster

TIBCO PSC cluster consists of the following containers: TIBCO PSC, TIBCO PSC\_CONFIG, Apache Ignite, and PSC\_OPD. For the supported versions of these containers, see the *Readme.txt* file of TIBCO PSC.

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

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



The TIBCO PSC cluster supports PostgreSQL and Oracle databases.

Kubernetes is required to run the TIBCO PSC 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 PSC Cluster Container

Build the Docker images for the components included in the TIBCO PSC cluster. The steps to build Docker images for TIBCO PSC cluster components are documented in the `ReadMe.txt` file available in each of the component directories. 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 PSC database schema. See the `ReadMe.txt` file located at `docker/build/PSC` and *TIBCO PSC Installation and Configuration*.
- Ensure that you have the `mdmc/mdm:9.3.1.HF1` Docker image built. To build the `mdmc/mdm:9.3.1.HF1` Docker image, follow the instructions given in the *TIBCO MDM Cloud Deployment Guide*.
- 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 PSC	<code>docker/build/PSC</code>
TIBCO OPD	<code>docker/build/PSC_OPD</code>
Apache Ignite	<code>docker/build/Ignite</code>
TIBCO PSC Configurator	<code>docker/build/PSC_CONFIG</code>

## TIBCO PSC 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 the configuration. Use YAML files to configure Kubernetes resources such as pods, services, and deployments.

Kubernetes Objects	Description
ConfigMap (Configuration.yaml)	<p>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.</p> <p>For the parameters available in the ConfigMap file, see <a href="#">ConfigMap Parameters</a>.</p>
Secrets ( Secrets.yaml)	<p>Secrets are objects, which stores sensitive information about your clusters such as database and EMS user names and passwords in the encrypted format.</p>
Deployments	<p>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.</p> <ul style="list-style-type: none"> <li>• TIBCO PSC: PSC.yaml</li> <li>• TIBCO PSC Configurator: PscConfigurator.yaml</li> <li>• Apache Ignite: IgniteCache.yaml</li> </ul> <p>In the deployment files, you can update the values of replicas, image names, and memory.</p> <div> <p><b>Warning:</b> Do not change any other parameters.</p> </div> <p>If you are deploying through Helm chart, see the YAML files located in each of the charts directory (docker/k8s_helm/pscheim/charts/component).</p>
Volumes	<p>Docker volume files are used for persisting data.</p> <ul style="list-style-type: none"> <li>• TIBCO PSC: PSCVolumes.yaml</li> </ul>
Services	<p>Service defines a logical set of Pods and a policy to access these pods.</p>

Kubernetes Objects	Description
	<p>You must create services for TIBCO PSC UI, TIBCO PSC Configurator UI, Apache Ignite, and PSC OPD UI.</p> <p>You must create headless services for the following components:</p> <ul style="list-style-type: none"><li>• TIBCO PSC: <code>PSC.yaml</code></li><li>• TIBCO PSC Configurator: <code>PscConfigurator.yaml</code></li><li>• Apache Ignite: <code>IgniteCache.yaml</code></li><li>• PSC_OPD: <code>opd.yaml</code></li></ul>

## Configuring Kubernetes Containers for TIBCO PSC

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 PSC cluster components, see [TIBCO PSC Cluster Container Components YAML Files](#).

### Procedure

1. Create the following Kubernetes objects that are required for deploying the TIBCO PSC cluster. These objects include required Kubernetes objects and services for the cluster:
  - Namespace
  - Rolebinding
  - Config Maps
  - Secrets
  - Kubernetes objects for TIBCO PSC, TIBCO PSC Configurator, Apache Ignite, and PSC OPD.
  - Services for TIBCO PSC UI, TIBCO PSC Configurator UI, PSC OPD UI, and

headless services for TIBCO PSC, Apache Ignite.

2. On the command line, using the `kubectl create` command, deploy the TIBCO PSC Cluster components using YAML files. For details, see the `DeploymentInstructions.txt` file in `docker/k8s_deployment`.

## Result

Kubernetes deployments and services 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/pschelm`.

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  Example: POSTGRES and ORACLE
MQ_MDM_DB_HOST	Database server host name  Example: pgsqldb-eks.cx4wjme9qqns.us-west-2.rds.amazonaws.com
MQ_MDM_DB_PORT	Database port  Example: <ul style="list-style-type: none"><li>• 5432 for POSTGRESQL</li></ul>

Parameter Name	Definition
	<ul style="list-style-type: none"> <li>1521 for ORACLE</li> </ul>
MQ_MDM_DB_NAME	<p>Database name</p> <p>Example: velodb</p>
MQ_MDM_DB_USETABLESPACES	<p>Set this value to <code>true</code>, to set your own tablespace location. For example, Microsoft Azure SQL and on-premise database setup.</p> <p>Set this value to <code>false</code> for cloud platforms where you do not know the tablespace location. For example, AWS and GCP.</p>
MQ_MDM_DB_MIN_CONN_COUNT MQ_MDM_DB_MAX_CONN_COUNT	Minimum and maximum number of database connections for TIBCO PSC database.
MQ_MDM_DB_FLUSH_STRATEGY	Specifies how the pool must be flushed if an error occurs. By default, the value is set to <code>FailingConnectionOnly</code> , which forces destroying connections with error.
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 <code>300000</code> , which 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 4 minutes.

**Note:** MQ\_PSC\_DB\_FLUSH\_STRATEGY, MQ\_PSC\_DB\_BLOCKING\_TIMEOUT, and MQ\_PSC\_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 PSC support:

MQ_MDM_JMS_CLUSTER_TYPE	Set the cluster type. For example, <code>TIBCOcluster</code> . (for TIBCO EMS, use <code>TIBCOcluster</code> )
-------------------------	--

Parameter Name	Definition
MQ_EMS_CLUSTER_URL	Set the cluster URL. For example, <code>tcp://localhost:7222</code> .
MQ_MDM_FAST_CACHE_ENABLED	To enable golden record cache, set this value to <code>true</code> . Else, set it to <code>false</code> .
MQ_MDM_HTTP_SESSION_REPLICATION_ENABLED	By default, the value is set to <b>true</b> to use multiple nodes and replicate the session.
FC_SERVICE_HOST	Example: PSC_SERVICE IP address (cluster/loadbalancer)
FC_SERVICE_PORT	Example: PSC_SERVICE Port exposed (cluster/loadbalancer)
FC_SERVICE_FLAG	While using Ingress make the flag as <code>true</code>
WEB_URL_HOST	<code>http://&lt;hostname&gt;:8080</code>
OPD_APP_DATA_VOLUME	Depending on the OPD deployment take select lite or heavy configuration. (see OPD document for more details)
MQ_IGNITE_COMPONENT_IDS	Example: MQ_IGNITE_COMPONENT_IDS: <code>ignite.namespace.svc.cluster.local</code>
MQ_MDM_COMPONENT_IDS	Example: MQ_PSC_COMPONENT_IDS: <code>psc.namespace.svc.cluster.local</code>

## 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 psc Docker container, perform the following steps:

1. Navigate to `docker/build/Psc/` 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.	<code>docker/k8s_deployment</code>
If you have used a helm chart to deploy TIBCO PSC, modify the <code>cpu</code> and <code>memory</code> values in the YAML file.	<code>docker/k8s_helm/psc/values.yaml</code> For other components, the <code>values.yaml</code> file is located under each subchart of the <code>psc</code> 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:

- `psc/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 `pvc.yaml` and `pvc.yaml` files.

- Set `dynamicProvisioning:false` in the `values.yaml` file.

## 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)	Local	None
Mounting happens with the host machine. For example,		

Provisioning Types	Storage Class	Cloud Provider
<pre>### NFS Mount ### nfs: path: "/var/nfs/general" server: "IPaddress"</pre>		

## 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
	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



**Provisioning  
types****Changes**

**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 a packaging format called charts. By using helm charts, you can deploy TIBCO PSC 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/psc` 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

psc directory.

2. Update the database and EMS 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 PSC Cluster on the Cloud

Navigate to the extracted `TIB_cim-ac_5.1.0_container` directory and locate the files and relevant directories to deploy the TIBCO PSC cluster on Microsoft Azure and Google Cloud Platform. You can run the dockerized TIBCO PSC application in a Kubernetes cluster on the cloud platform of your choice.

### Supported Versions

Before you begin, see TIBCO PSC 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:

- [Microsoft Azure](#) and [Azure Kubernetes Service \(AKS\)](#)
- [Google Cloud Platform \(GCP\)](#)

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 PSC on Microsoft Azure Based Kubernetes Cluster

By using Azure Kubernetes Service (AKS), you can easily deploy TIBCO PSC 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 PSC application, see [Build Docker Image for TIBCO PSC Cluster Container](#).
- You must have a Microsoft Azure account with an 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 Microsoft Azure. See [Setting Up a Kubernetes Cluster on AKS](#).
4. Create SQL Server on Azure virtual machine. See [Prerequisites for Deploying TIBCO PSC 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 an 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 PSC application that you want to deploy to the Kubernetes cluster. See [Build Docker Image for TIBCO PSC 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 pscresourcegroup --name pscregistry --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 pscregistry
```

The command returns a Login Succeeded message after completion.

### 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 the Kubernetes cluster. To enable a Kubernetes cluster to interact with other Azure resources, 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

**i 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 pscregistry

```
az aks update -n psccluster -g pscresourcegroup --attach-acr pscregistry
```

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 pscresourcegroup --name psccluster
```

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 pscresourcegroup --name psccluster
```

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 PSC Cluster on Kubernetes by Using YAML Files](#)

## Prerequisites for Deploying TIBCO PSC 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 PSC Cluster by Using Helm](#)

## Running TIBCO PSC on Google Cloud Platform

By using Google Cloud Platform (GCP) with Google Kubernetes Engine (GKE), you can easily deploy a TIBCO PSC 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 PSC Cluster by Using Helm](#)
3. [Access TIBCO PSC 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](#).

**i 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**.

**i Note:** You can select different types of clusters. For details, see [GCP documentation](#).

### What to do next

See [Deploying TIBCO PSC Cluster by Using Helm](#)

## Tag and Push the Docker Image for Cloud Platforms

Cloud Platform	Steps
AWS EKS	<ol style="list-style-type: none"> <li>1. Retrieve an authentication token and authenticate your Docker client to your registry.  For example:               <pre>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/pssc/psc</pre> </li> <li>2. Tag your images to push to ECR Registry               <pre>docker tag pssc/psc:latest AWS_ACCOUNT_ID.dkr.ecr.us-west-2.amazonaws.com/pssc/psc:latest</pre> </li> </ol>

Cloud Platform	Steps									
	<div>3. Run the following command to push this image to your newly created AWS repository:</div> <div><pre>docker push 061574984669.dkr.ecr.us-west-2.amazonaws.com/pscc/psc:latest</pre></div>									
Microsoft Azure	<div>1. To use the TIBCO PSC application container images with Azure Container Registry, tag the image with the login server address of your registry.</div> <div>a. View the list of your local image by using the <code>docker images</code> command.</div> <div><pre>\$ docker images</pre><table><tr><th>REPOSITORY</th><th>TAG</th><th>IMAGE ID</th></tr><tr><td>pscc/ignite</td><td>version.latest</td><td>18763f0d1403</td></tr><tr><td>3 weeks ago</td><td>790MB</td><td></td></tr></table></div> <div>b. Get the login server address for the Azure Container Registry by using the <code>az acr list</code> command.</div> <div><pre>az acr list --resource-group resource_group_name --query "[].{acrLoginServer:loginServer}" --output table</pre></div> <div>c. Tag your application image with the 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.</div> <div><pre>docker tag pscc/ignite:version.latest psctestregistry.azurecr.io/pscc/ignite:version.latest</pre></div> <div>d. Push the application image to your container registry by using the <code>docker push</code> command.</div> <div><pre>docker push</pre></div>	REPOSITORY	TAG	IMAGE ID	pscc/ignite	version.latest	18763f0d1403	3 weeks ago	790MB	
REPOSITORY	TAG	IMAGE ID								
pscc/ignite	version.latest	18763f0d1403								
3 weeks ago	790MB									

Cloud Platform	Steps
	<pre>psctestregistry.azurecr.io/pscc/ignite:version.latest</pre> <p>e. Push for all other TIBCO PSC cluster images by using the docker push command.</p>

## Deploying TIBCO PSC Cluster on Kubernetes by Using YAML Files

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

### Before you begin

- See [ConfigMap Parameters](#)
- For Microsoft Azure, See [Prerequisites for Deploying TIBCO PSC 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 the TIBCO PSC cluster. For information, see [Configuring Kubernetes Containers for TIBCO PSC](#).

### What to do next

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

## Deploying Kubernetes Dashboard (Optional)

You can deploy the 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 `psc-cluster-admin` service account.
    - i. Retrieve an authentication token for the `psc-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 psc-admin-service-account.yaml
```

3. On the Kubernetes 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 PSC 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	<p>Update the following values (Base64 encoded format) in the <code>values.yaml</code> file located at <code>docker/k8s_helm/psc</code>.</p> <ul style="list-style-type: none"> <li>• <code>azurestorageaccountname</code></li> <li>• <code>azurestorageaccountkey</code></li> </ul>
For AWS EKS	<p>a. Update <code>cidr</code> value of VPC in the <code>LoadBalancerSourceRanges</code> section of the <code>values.xml</code> file.</p> <p>b. Update all relevant field values in the <code>values.yaml</code> file.</p>
For GCP GKE	<p>On the command line, type the following command to connect to the gcloud GKE cluster:</p> <pre>gcloud auth configure-docker gcloud container clusters get-credentials GKE Cluster Name --region Region -- project Project Name</pre>

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

```
helm install --name Name_Of_Release Path_Of_PSC_Helm_Chart_Directory --create-namespace --namespace=Kubernetes_Namespace
```

Example:

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

## Result

The TIBCO PSC application has been successfully deployed.

## What to do next

[Access TIBCO PSC 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 PSC 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 PSC	<code>kubectl get services psc-ui</code>	<ul style="list-style-type: none"><li>• <b>For Kubernetes:</b><ul style="list-style-type: none"><li>◦ For TIBCO PSC (psc-ui): <code>http://ExternalIP:host_port/eml/Login</code></li></ul></li></ul>
TIBCO PSC Configurator	<code>kubectl get services psc-pscconfig</code>	For TIBCO PSC Configurator (psc-pscconfig): <code>http://ExternalIP:host_port/config/index.html#/login</code>

# 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 [TIBCO Product Documentation](#) website, mainly in HTML and PDF formats.

The [TIBCO Product Documentation](#) website is updated frequently and is more current than any other documentation included with the product.

## Product-Specific Documentation

The following documentation for TIBCO® Product and Service Catalog is available on the [TIBCO® Product and Service Catalog Product Documentation](#) page.

- *TIBCO® Product and Service Catalog Release Notes*
- *TIBCO® Product and Service Catalog Installation and Configuration*
- *TIBCO® Product and Service Catalog Product Catalog Guide*
- *TIBCO® Product and Service Catalog User Guide*
- *TIBCO® Product and Service Catalog Web Services*
- *TIBCO® Product and Service Catalog Offer and Price Designer User Guide*
- *TIBCO® Product and Service Catalog Cloud Deployment*
- *TIBCO® Product and Service Catalog Security Guidelines*

## How to Contact TIBCO Support

Get an overview of [TIBCO Support](#). You can contact TIBCO Support in the following ways:

- For accessing the Support Knowledge Base and getting personalized content about products you are interested in, visit the [TIBCO Support](#) website.



- For creating a Support case, you must have a valid maintenance or support contract with TIBCO. You also need a user name and password to log in to [TIBCO Support](#) website. If you do not have a user name, 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 TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

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

This document is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of Cloud Software Group, Inc.

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

Java and all Java based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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.

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.

This 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 this software version 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 SOFTWARE GROUP, INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.

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 "READ ME" FILES.

This and other products of Cloud Software Group, Inc. may be covered by registered patents. Please refer to TIBCO's Virtual Patent Marking document (<https://www.tibco.com/patents>) for details.

Copyright © 2010-2022. Cloud Software Group, Inc. All Rights Reserved.