Red Hat Process Automation Manager 7.0

Deploying a Red Hat Process Automation Manager immutable server environment on Red Hat OpenShift Container Platform

Last Updated: 2018-10-01

Red Hat Process Automation Manager 7.0 Deploying a Red Hat Process Automation Manager immutable server environment on Red Hat OpenShift Container Platform

Red Hat Customer Content Services
brms-docs@redhat.com

Legal Notice

Abstract

This document describes how to deploy a Red Hat Process Automation Manager 7.0 immutable server environment on Red Hat OpenShift Container Platform.

Table of Contents

PREFACE

As a system engineer, you can deploy a Red Hat Process Automation Manager immutable server environment on Red Hat OpenShift Container Platform to provide an infrastructure to execute processes and other business assets. You can use standard integration tools to manage the immutable Process Server image. You can create new server images to add and update processes.

Prerequisites

- At least four gigabytes of memory are available in the OpenShift environment.
- The OpenShift project for the deployment is created.
- You have logged in to the project using the OpenShift web console and using the **oc** command.
- If you intend to scale any of the Business Central or Business Central Monitoring pods, your OpenShift environment supports persistent volumes with ReadWriteMany mode.

IMPORTANT

ReadWriteMany mode is not supported on OpenShift Online and OpenShift Dedicated.

CHAPTER 1. OVERVIEW OF RED HAT PROCESS AUTOMATION MANAGER ON RED HAT OPENSHIFT CONTAINER PLATFORM

If you have an OpenShift environment, you can deploy Red Hat Process Automation Manager into this environment.

In this solution, components of Red Hat Process Automation Manager are deployed as separate OpenShift pods. You can scale each of the pods up and down individually, providing as few or as many containers as necessary for a particular component. You can use standard OpenShift methods to manage the pods and balance the load.

The following key components are available as pods running in OpenShift:

- Process Server, also known as *Execution Server* or *KIE Server*, is the infrastructure element that runs a process or a group of processes. All process logic runs on execution servers.
  A database server is normally required for Process Server. You can provide a database server in another OpenShift pod or configure an execution server on OpenShift to use any other database server. Alternatively, Process Server can use an H2 database; in this case, the pod cannot be scaled.

  You can freely scale up a Process Server pod, providing as many copies, running on the same host or different hosts, as necessary. As you scale a pod up or down, all its copies run the same processes and use the same database server. OpenShift provides load balancing and a request can be handled by any of the pods.

  To run a different set of processes, deploy a separate Process Server pod, which can also be scaled up or down. You can have as many separate replicated Process Server pods as necessary.

- Business Central is a web-based interactive environment for authoring processes. It also provides a management and monitoring console. You can use Business Central to develop processes, deploy processes to Process Servers, and monitor the execution.
  Business Central is a centralized application. However, you can configure it for high availability, where multiple pods run and share the same data. (In the current version, the high-availability functionality is a technology preview).

  Business Central includes a Git repository that holds the source for the processes that you develop on it. It also includes a built-in Maven repository. Depending on configuration, Business Central can place the compiled processes (KJAR files) into the built-in Maven repository or (if configured) into an external Maven repository.

- Business Central Monitoring is a web-based management and monitoring console. It can manage deployment of processes to Process Servers and provide monitoring information, but does not include authoring capabilities. You can use this component to manage staging and production environments.

- Smart Router is an optional layer between Process Servers and other components that interact with them. It is required if you want Business Central or Business Central Monitoring to interact with several different Process Servers. Also, when your environment includes many processes running on different Process Servers, Smart Router provides a single endpoint to all client applications. A client application can make a REST API call requiring any process. Smart Router automatically determines which Process Server must be called for any particular request.

You can arrange these and other components into various environment configurations within OpenShift. You can use the templates provided with Red Hat Process Automation Manager to deploy the most common combinations.

The following environment types are typical:

- *Authoring*: An environment for creating and modifying processes using Business Central. It consists of pods that provide Business Central for the authoring work and a Process Server for test execution of the processes. For instructions about deploying this environment, see *Deploying a Red Hat Process Automation Manager authoring environment on Red Hat OpenShift Container Platform*.

- *Managed deployment*: An environment for running existing processes for staging and production purposes. This environment includes several groups of Process Server pods; you can deploy and undeploy processes on every such group and also scale the group up or down as necessary. Use Business Central Monitoring to deploy, run, and stop the processes and to monitor their execution. For instructions about deploying this environment, see *Deploying a Red Hat Process Automation Manager managed server environment on Red Hat OpenShift Container Platform*.

- *Deployment with immutable servers*: An alternate environment for running existing processes for staging and production purposes. In this environment, when you deploy a Process Server pod, it builds an image that loads and starts a process or group of processes. You cannot stop any process on the pod or add any new process to the pod. If you want to use another version of a process or modify the configuration in any other way, you deploy a new server image and displace the old one. In this system, you can use typical container-based integration workflows and do not need to use any other tools to manage the pods. Optionally, you can use Business Central Monitoring to monitor the performance of the environment and to stop and restart some of the process instances, but not to deploy additional processes to any Process Server or undeploy any existing ones (you can not add or remove containers). For instructions about deploying this environment, see *Deploying a Red Hat Process Automation Manager immutable server environment on Red Hat OpenShift Container Platform*.

To deploy a Red Hat Process Automation Manager environment on OpenShift, you can use the templates that are provided with Red Hat Process Automation Manager. You can modify the templates to ensure that your environment suits your needs.

CHAPTER 2. PREPARING TO DEPLOY RED HAT PROCESS AUTOMATION MANAGER IN YOUR OPENSHIFT ENVIRONMENT

Before deploying Red Hat Process Automation Manager in your OpenShift environment, you need to complete several preparatory tasks. You do not need to repeat these tasks if you want to deploy additional images, for example, for new versions of processes or for other processes.

2.1. ENSURING THE AVAILABILITY OF IMAGE STREAMS

You must ensure that the image streams that are required for the deployment are available in your OpenShift environment. Some versions of the OpenShift environment include the necessary image streams. You must check if they are available. If they are not available, you must install the **`rhpam70-image-streams.yaml`** file.

Procedure

1. Run the following commands:

```
$ oc get imagestreamtag -n openshift | grep rhpam70-businesscentral
$ oc get imagestreamtag -n openshift | grep rhpam70-kieserver
```

   If the outputs of both commands are not empty, the required image streams are available and no further action is required.

2. If the output of one or both of the commands is empty, download the **`rhpam-7.0.0-openshift-templates.zip`** product deliverable file from the Software Downloads page for Red Hat Process Automation Manager 7.0. Extract the **`rhpam70-image-streams.yaml`** file from it. Complete one of the following actions:

   - Run the following command:

```
$ oc create -f rhpam70-image-streams.yaml
```

   - Using the OpenShift Web UI, select **Add to Project** → **Import YAML / JSON**, then choose the file or paste its contents.

2.2. CREATING THE SECRETS FOR PROCESS SERVER

OpenShift uses objects called **`Secrets`** to hold sensitive information, such as passwords or keystores. See the Secrets chapter in the OpenShift documentation for more information.

You must create an SSL certificate for Process Server and provide it to your OpenShift environment as a secret.

Procedure

1. Generate an SSL keystore with a private and public key for SSL encryption for Process Server. In a production environment, generate a valid signed certificate that matches the expected URL of the Process Server. Save the keystore in a file named **`keystore.jks`**. Record the name of the certificate and the password of the keystore file.

See Generate a SSL Encryption Key and Certificate for more information on how to create a keystore with self-signed or purchased SSL certificates.

2. Use the **oc** command to generate a secret named **kieserver-app-secret** from the new keystore file:

```
$ oc create secret generic kieserver-app-secret --from-
file=keystore.jks
```

2.3. CREATING THE SECRETS FOR BUSINESS CENTRAL

If you are planning to deploy Business Central or Business Central Monitoring in your OpenShift environment, you must create an SSL certificate for Business Central and provide it to your OpenShift environment as a secret. Do not use the same certificate and keystore for Business Central and for Process Server.

Procedure

1. Generate an SSL keystore with a private and public key for SSL encryption for Business Central. In a production environment, generate a valid signed certificate that matches the expected URL of the Business Central. Save the keystore in a file named **keystore.jks**. Record the name of the certificate and the password of the keystore file.
See Generate a SSL Encryption Key and Certificate for more information on how to create a keystore with self-signed or purchased SSL certificates.

2. Use the **oc** command to generate a secret named **businesscentral-app-secret** from the new keystore file:

```
$ oc create secret generic businesscentral-app-secret --from-
file=keystore.jks
```

2.4. CHANGING GLUSTERFS CONFIGURATION

Check whether your OpenShift environment uses GlusterFS to provide permanent storage volumes. If it uses GlusterFS, to ensure optimal performance, tune your GlusterFS storage by changing the storage class configuration.

Procedure

1. To check whether your environment uses GlusterFS, run the following command:

```
oc get storageclass
```

In the results, check whether the **(default)** marker is on the storage class that lists **glusterfs**. For example, in the following output the default storage class is **gluster-container**, which does list **glusterfs**:

```
NAME              PROVISIONER                     AGE
gluster-block     gluster.org/glusterblock        8d
gluster-container (default) kubernetes.io/glusterfs 8d
```

If the result has a default storage class that does not list **glusterfs** or if the result is empty, you do not need to make any changes. In this case, skip the rest of this procedure.

2. To save the configuration of the default storage class into a YAML file, run the following command:

```
oc get storageclass <class_name> -o yaml >storage_config.yaml
```

Where **class-name** is the name of the default storage class. For example:

```
oc get storageclass gluster-container -o yaml >storage_config.yaml
```

3. Edit the **storage_config.yaml** file:

   a. Remove the lines with the following keys:

      - **creationTimestamp**
      - **resourceVersion**
      - **selfLink**
      - **uid**

   b. On the line with the **volumeoptions** key, add the following two options: **features.cache-invalidation on, performance.nl-cache on**. For example:

```
volumeoptions: client.ssl off, server.ssl off, features.cache-
invalidation on, performance.nl-cache on
```

4. To remove the existing default storage class, run the following command:

```
oc delete storageclass <class_name>
```

Where **class-name** is the name of the default storage class. For example:

```
oc delete storageclass gluster-container
```

5. To re-create the storage class using the new configuration, run the following command:

```
oc create -f storage_config.yaml
```

CHAPTER 3. ENVIRONMENT WITH IMMUTABLE SERVERS

You can deploy an environment that includes several different pods running Process Server with preloaded processes. The database servers are, by default, also run in pods. Each Process Server pod can be separately scaled as necessary.

In this case, any processes (KJAR files) must be loaded onto a Process Server at the time the image is created. You cannot load or unload processes on a running Process Server. The advantage of this approach is that the Process Server with the processes in it runs like any other containerized service and does not require specialized management. You can use existing approaches to cloud integration to deploy and manage the servers.

Optionally, you can also deploy a pod with Business Central Monitoring and a pod with Smart Router. You can use Business Central Monitoring to start and stop (but not deploy) processes on your Process Servers and to view monitoring data.

Smart Router is a single endpoint that can receive calls from client applications to any of your processes and route each call automatically to the server that actually runs the process.

When you create a Process Server image, you must build your processes using S2I (Source to Image). Provide a Git repository with the source of your processes and other business assets; if you develop the processes or assets in Business Central, copy the source into a separate repository for the S2I build. OpenShift automatically builds the source, installs the process into the Process Server image, and starts the process. No further management of the image is required. If you want to use a new version of the process, you can build a new image.

If you are using Business Central for authoring processes, you can extract the source for your process and place it into a separate Git repository (such as GitHub or an on-premise installation of GitLab) for use in the S2I build.

If you want to use Business Central Monitoring, you must install the Monitoring and Smart Router template *before* creating any Process Server images. You must also provide a Maven repository. Your integration process must ensure that all the versions of KJAR files built into any Process Server image are also available in the Maven repository.

3.1. DEPLOYING MONITORING AND SMART ROUTER FOR AN ENVIRONMENT WITH IMMUTABLE SERVERS

If you want to use Business Central Monitoring and Smart Router for an environment with immutable servers, you must deploy them before deploying any Process Servers. If you do not want to use these components, skip this procedure.

To deploy Business Central Monitoring and Smart Router for an environment with immutable servers, use the **`rhpam70-prod-immutable-monitor.yaml`** template file. You can extract this file from the **`rhpam-7.0.0-openshift-templates.zip`** product deliverable file. You can download the file from the Software Downloads page for Red Hat Process Automation Manager 7.0.

Procedure

1. Use one of the following methods to deploy the template:
    - In the OpenShift Web UI, select **Add to Project** → **Import YAML / JSON** and then select or paste the **`rhpam70-prod-immutable-monitor.yaml`** file. In the **Add Template** window, ensure **Process the template** is selected and click **Continue**.

- To use the OpenShift command line console, prepare the following command line:

```
oc new-app -f <template-path>/rhpam70-prod-immutable-monitor.yaml
-p BUSINESS_CENTRAL_HTTPS_SECRET=businesscentral-app-secret
```

  In this command line:

  - Replace **<template-path>** with the path to the downloaded template file.
  - Use as many **-p PARAMETER=value** pairs as needed to set the required parameters. You can view the template file to see descriptions for all parameters.

2. Set the following parameters as necessary:

- **Business Central Server Keystore Secret Name** (**BUSINESS_CENTRAL_HTTPS_SECRET**): The name of the secret for Business Central, as created in Section 2.3, “Creating the secrets for Business Central”.
- **Application Name** (**APPLICATION_NAME**): The name of the OpenShift application. It is used in the default URLs for Business Central and Smart Router. OpenShift also uses the application name to create a separate set of deployment configurations, services, routes, labels, and artifacts. You can deploy several applications using the same template into the same project, as long as you use different application names.
- **Maven repository URL** (**MAVEN_REPO_URL**): A URL for a Maven repository. You must upload all the processes (KJAR files) that are to be deployed in your environment into this repository.
- **Maven repository username** (**MAVEN_REPO_USERNAME**): The username for the Maven repository.
- **Maven repository password** (**MAVEN_REPO_PASSWORD**): The username for the Maven repository.
- **Business Central Server Certificate Name** (**BUSINESS_CENTRAL_HTTPS_NAME**): The name of the certificate in the keystore that you created in Section 2.3, “Creating the secrets for Business Central”.
- **Business Central Server Keystore Password** (**BUSINESS_CENTRAL_HTTPS_PASSWORD**): The password for the keystore that you created in Section 2.3, “Creating the secrets for Business Central”.
- **ImageStream Namespace** (**IMAGE_STREAM_NAMESPACE**): The namespace where the image streams are available. If the image streams were already available in your OpenShift environment (see Section 2.1, “Ensuring the availability of image streams”), the namespace is **openshift**. If you have installed the image streams file, the namespace is the name of the OpenShift project.
  You can also set other parameters as necessary.

3. Complete the creation of the environment, depending on the method that you are using:

- In the OpenShift Web UI, click **Create**.
- Complete and run the command line.

4. Record the suggested command line that is displayed. This command line includes the parameters that you must set when deploying the immutable servers.

3.2. EXTRACTING THE SOURCE CODE FROM BUSINESS CENTRAL FOR USE IN AN S2I BUILD

If you are using Business Central for authoring processes, you can extract the source code for your process and place it into a separate Git repository (such as GitHub or an on-premise installation of GitLab) for use in the S2I build.

Procedure

1. Use the following command to extract the source code:

```
git clone ssh://adminUser@bcHost:8001/MySpace/MyProject
```

   Replace:

   - **`adminUser`** with the administrative user for Business Central
   - **`bcHost`** with the host on which Business Central is running
   - **`MySpace`** with the name of the Business Central space in which the project is located
   - **`MyProject`** with the name of the project

2. Upload the source code to another Git repository for the S2I build.

3.3. DEPLOYING AN IMMUTABLE PROCESS SERVER

To deploy an immutable Process Server, use the **`rhpam70-prod-immutable-kieserver.yaml`** template file. You can extract this file from the **`rhpam-7.0.0-openshift-templates.zip`** product deliverable file. You can download the file from the Software Downloads page for Red Hat Process Automation Manager 7.0.

If you want to modify the environment defined by the template file, see Section 3.4, “Modifying the server template for an immutable server environment”.

This procedure also retrieves the source code for any processes that must run on this server and builds the processes (KJAR files).

You can configure the Process Server to connect to Smart Router and to Business Central Monitoring. If you use the server with Business Central Monitoring, you must ensure that the same versions of KJAR files are uploaded to the Maven repository that the Business Central Monitoring instance uses.

Procedure

1. Use one of the following methods to deploy the template:

   - In the OpenShift Web UI, select **Add to Project** → **Import YAML / JSON** and then select or paste the **`rhpam70-prod-immutable-kieserver.yaml`** file. In the **Add Template** window, ensure **Process the template** is selected and click **Continue**.
   - To use the OpenShift command line console, prepare the following command line:

```
oc new-app -f <template-path>/rhpam70-prod-immutable-
kieserver.yaml -p KIE_SERVER_HTTPS_SECRET=kieserver-app-secret
```

In this command line:

- Replace **`<template-path>`** with the path to the downloaded template file.
- Use as many **`-p PARAMETER=value`** pairs as needed to set the required parameters. You can view the template file to see descriptions for all parameters.
  Alternatively, if you have deployed the monitoring infrastructure (see Section 3.1, "Deploying monitoring and Smart Router for an environment with immutable servers"), use the command line that was displayed when you deployed the infrastructure and add the **`-p KIE_SERVER_HTTPS_SECRET=kieserver-app-secret`** parameter.

2. Set the following parameters as necessary:

- **KIE Server Keystore Secret Name** (**`KIE_SERVER_HTTPS_SECRET`**): The name of the secret for Process Server, as created in Section 2.2, "Creating the secrets for Process Server".
- **Application Name** (**`APPLICATION_NAME`**): The name of the OpenShift application. It is used in the default URL for Process Server. OpenShift also uses the application name to create a separate set of deployment configurations, services, routes, labels, and artifacts. You can deploy several applications using the same template into the same project, as long as you use different application names.
- **KIE Server Certificate Name** (**`KIE_SERVER_HTTPS_NAME`**): The name of the certificate in the keystore that you created in Section 2.2, "Creating the secrets for Process Server".
- **KIE Server Keystore Password** (**`KIE_SERVER_HTTPS_PASSWORD`**): The password for the keystore that you created in Section 2.2, "Creating the secrets for Process Server".
- **KIE Server ID** (**`KIE_SERVER_ID`**): The name of the server template on the Business Central that the Process Server is to join. Set this parameter to any value. If you do not set it and then the Process Server pod is restarted or scaled, each new instance of the server might join a new template.
- **KIE Server Container Deployment** (**`KIE_SERVER_CONTAINER_DEPLOYMENT`**): The identifying information of the decision service (KJAR file) that is built from your source. The format is: **`<containerId>=<groupId>:<artifactId>:<version>`**. You can provide two or more KJAR files using the **`|`** separator, for example: **`containerId=groupId:artifactId:version|c2=g2:a2:v2`**. The Maven build process must produce all these files from the source in the Git repository.
- **Git Repository URL** (**`SOURCE_REPOSITORY_URL`**): The URL for the Git repository that contains the source for your decision service.
- **Git Reference** (**`SOURCE_REPOSITORY_REF`**): The branch in the Git repository
- **Context Directory** (**`CONTEXT_DIR`**): The path to the source within the project downloaded from the Git repository
- **Artifact Directory** (**`ARTIFACT_DIR`**): The path within the project that contains the required binary files (KJAR files and any other necessary files) after a successful Maven build. Normally this directory is the target directory of the build. However, you can provide prebuilt binaries in this directory in the Git repository

- **ImageStream Namespace** (**`IMAGE_STREAM_NAMESPACE`**): The namespace where the image streams are available. If the image streams were already available in your OpenShift environment (see Section 2.1, “Ensuring the availability of image streams”), the namespace is **`openshift`**. If you have installed the image streams file, the namespace is the name of the OpenShift project.

3. If your build includes dependencies that are not available on the public Maven tree and require a separate repository, set the parameters to provide this repository:
   - **Maven repository URL** (**`MAVEN_REPO_URL`**): The URL for the Maven repository.
   - **Maven repository username** (**`MAVEN_REPO_USERNAME`**): The username for the Maven repository.
   - **Maven repository password** (**`MAVEN_REPO_PASSWORD`**): The username for the Maven repository.
4. If you want to use Business Central Monitoring or Smart Router, set the parameters that were displayed in the sample command line after you deployed the monitoring infrastructure (see Section 3.1, “Deploying monitoring and Smart Router for an environment with immutable servers”).
5. If you modified the template to use an external database server for the Process Server, set the following parameters:
   - KIE Server External Database Driver (**`KIE_SERVER_EXTERNALDB_DRIVER`**): The driver for the server, depending on the server type:
     - mysql
     - postgresql
     - mariadb
     - mssql
     - db2
     - oracle
     - sybase
   - KIE Server External Database User (**`KIE_SERVER_EXTERNALDB_USER`**) and KIE Server External Database Password (**`KIE_SERVER_EXTERNALDB_PWD`**): The user name and password for the external database server.
   - KIE Server External Database URL (**`KIE_SERVER_EXTERNALDB_URL`**): The JDBC URL for the external database server.
   - KIE Server External Database Dialect (**`KIE_SERVER_EXTERNALDB_DIALECT`**): The Hibernate dialect for the server, depending on the server type:
     - **`org.hibernate.dialect.MySQL5Dialect`** (used for MySQL and MariaDB)
     - **`org.hibernate.dialect.PostgreSQLDialect`**
     - **`org.hibernate.dialect.SQLServer2012Dialect`** (used for MS SQL)

- **org.hibernate.dialect.DB2Dialect**
- **org.hibernate.dialect.Oracle12cDialect**
- **org.hibernate.dialect.SybaseASE15Dialect**

- KIE Server External Database Host (**KIE_SERVER_EXTERNALDB_HOST**): The host name of the external database server.
- KIE Server External Database name (**KIE_SERVER_EXTERNALDB_DB**): The database name to use on the external database server.

6. If you created a custom image for using an external database server other than MySQL or PostgreSQL, as described in Section 3.5, "Building a custom Process Server image for an external database", set the KIE Server Image Stream Name (**KIE_SERVER_IMAGE_STREAM_NAME**) parameter to the following value:
   - For Microsoft SQL Server, **rhpam70-kieserver-mssql-openshift**
   - For MariaDB, **rhpam70-kieserver-mariadb-openshift**
   - For IBM DB2, **rhpam70-kieserver-db2-openshift**
   - For Oracle Database, **rhpam70-kieserver-oracle-openshift**
   - For Sybase, **rhpam70-kieserver-sybase-openshift**
7. Complete the creation of the environment, depending on the method that you are using:
   - In the OpenShift Web UI, click **Create**.
   - Complete and run the command line.

3.4. MODIFYING THE SERVER TEMPLATE FOR AN IMMUTABLE SERVER ENVIRONMENT

By default, the immutable server template creates a separate PostgreSQL pod to provide the database server for each replicable Process Server. If you prefer to use MySQL or to use an external server (outside the OpenShift project), you need to modify the **rhpam70-prod-immutable-kieserver.yaml** template before deploying the server.

An OpenShift template defines a set of objects that can be created by OpenShift. To change an environment configuration, you need to modify, add, or delete these objects. To simplify this task, comments are provided in the Red Hat Process Automation Manager templates.

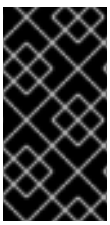
Some comments mark blocks within the template, staring with **BEGIN** and ending with **END**. For example, the following block is named **Sample block**:

```
## Sample block BEGIN
sample line 1
sample line 2
sample line 3
## Sample block END
```

For some changes, you might need to replace a block in one template file with a block from another template file provided with Red Hat Process Automation Manager. In this case, delete the block, then paste the new block in its exact location.

Procedure

- If you want to use MySQL instead of PostgreSQL, replace several blocks of the file, marked with comments from **`BEGIN`** to **`END`**, with blocks from the **`rhpam70-kieserver-mysql.yaml`** file:
  1. Replace the block named **`PostgreSQL database parameters`** with the block named **`MySQL database parameters`**. (Take this block and all subsequent replacement blocks from the **`rhpam70-kieserver-postgresql.yaml`** file.)
  2. Replace the block named **`PostgreSQL service`** with the block named **`MySQL service`**.
  3. Replace the block named **`PostgreSQL driver settings`** with the block named **`MySQL driver settings`**.
  4. Replace the block named **`PostgreSQL deployment config`** with the block named **`MySQL deployment config`**.
  5. Replace the block named **`PostgreSQL persistent volume claim`** with the block named **`MySQL persistent volume claim`**.
- If you want to use an external database server, replace several blocks of the file, marked with comments from **`BEGIN`** to **`END`**, with blocks from the **`rhpam70-kieserver-externaldb.yaml`** file, and also remove some blocks:
  1. Replace the block named **`PostgreSQL database parameters`** with the block named **`External database parameters`**. (Take this block and all subsequent replacement blocks from the **`rhpam70-kieserver-externaldb.yaml`** file.)
  2. Replace the block named **`PostgreSQL driver settings`** with the block named **`External database driver settings`**.
  3. Remove the following blocks of the file, marked with comments from **`BEGIN`** to **`END`**:
     - **`PostgreSQL service`**
     - **`PostgreSQL deployment config`**
     - **`PostgreSQL persistent volume claim`**



IMPORTANT

The standard Process Server image includes drivers for MySQL and PostgreSQL external database servers. If you want to use another database server, you must build a custom Process Server image. For instructions, see Section 3.5, “Building a custom Process Server image for an external database”.

3.5. BUILDING A CUSTOM PROCESS SERVER IMAGE FOR AN EXTERNAL DATABASE

If you want to use an external database server for a Process Server and this server is neither MySQL nor PostgreSQL, you must build a custom Process Server image with drivers for this server before deploying your environment.

You can use this build procedure to provide drivers for the following database servers:

- Microsoft SQL Server
- MariaDB
- IBM DB2
- Oracle Database
- Sybase

For the tested versions of the database servers, see Red Hat Process Automation Manager 7 Supported Configurations.

The build procedure creates a custom image that extends the existing Process Server image. It pushes this custom image into a new **`ImageStream`** in the **`openshift`** namespace with the same version tag as the original image.

Prerequisites

- You have logged on to your project in the OpenShift environment using the **`oc`** command as a user with the **`cluster-admin`** role.
- For IBM DB2, Oracle Database, or Sybase, you have downloaded the JDBC driver from the database server vendor.

Procedure

1. For IBM DB2, Oracle Database, or Sybase, provide the JDBC driver JAR in a local directory or on an HTTP server. Within the local directory or HTTP server, the following paths are expected:
   - For IBM DB2, **`<local_path_or_url>/com/ibm/db2/jcc/db2jcc4/10.5/db2jcc4-10.5.jar`**
   - For Oracle Database, **`<local_path_or_url>/com/oracle/ojdbc7/12.1.0.1/ojdbc7-12.1.0.1.jar`**
   - For Sybase, **`<local_path_or_url>/com/sysbase/jconn4/16.0_PL05/jconn4-16.0_PL05.jar`**
     Where **`<local_path_or_url>`** is the path to the local directory or the URL for the HTTP server where the driver is provided.
2. To install the source code for the custom build, download the **`rhpam-7.0.0-openshift-templates.zip`** product deliverable file from the Software Downloads page for Red Hat Process Automation Manager 7.0. Unzip the file and, using the command line, change to the **`templates/contrib/jdbc`** directory of the unzipped file.
3. Change to the following subdirectory:
   - For Microsoft SQL Server, **`mssql-driver-image`**

- For MariaDB, **`mariadb-driver-image`**
- For IBM DB2, **`db2-driver-image`**
- For Oracle Database, **`oracle-driver-image`**
- For Sybase, **`sybase-driver-image`**

4. Run the following command:

   - For Microsoft SQL Server or MariaDB:

```
../build.sh
```

   - For IBM DB2, Oracle Database, or Sybase:

```
../build.sh --artifact-repo=<local_path_or_url>
```

   Where **`<local_path_or_url>`** is the path to the local directory or the URL for the HTTP server where the driver is provided. For example:

```
../build.sh --artifact-repo=/home/builder/drivers
../build.sh --artifact-
repo=http://nexus.example.com/nexus/content/groups/public
```

   If you want to configure your OpenShift docker registry address in the process, add also the **`--registry=<registry_name.domain_name:port>`** parameter to your build command.

   Examples:

```
../build.sh --registry=docker-registry.custom-domain:80

../build.sh --artifact-repo=/home/builder/drivers --registry=docker-
registry.custom-domain:80
```

APPENDIX A. VERSIONING INFORMATION

Documentation last updated on: Monday, October 1, 2018.