Red Hat

Red Hat Process Automation Manager 7.3

Deploying a Red Hat Process Automation Manager environment on Red Hat OpenShift Container Platform using Automation Broker

Last Updated: 2021-03-29

Red Hat Process Automation Manager 7.3 Deploying a Red Hat Process Automation Manager environment on Red Hat OpenShift Container Platform using Automation Broker

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

Legal Notice

Abstract

This document describes how to deploy a Red Hat Process Automation Manager 7.3 environment on Red Hat OpenShift Container Platform using the Automation Broker with the Ansible Playbook.

Table of Contents

PREFACE

As a system engineer, you can deploy a Red Hat Process Automation Manager environment on Red Hat OpenShift Container Platform to provide an infrastructure to develop or execute processes and other business assets. You can use the Automation Broker to deploy the environment in an interactive process, setting all parameters as necessary. A Red Hat Process Automation Manager Ansible Playbook for the Automation Broker is available.

Prerequisites

- At least four gigabytes of memory must be available in the OpenShift environment.
- The OpenShift Ansible Broker must be installed and started in the OpenShift environment.
- The OpenShift project for the deployment must be created.
- You must be logged in to the project using the OpenShift web console and using the **oc** command.
  - Administrator access is required if Red Hat Process Automation Manager 7.3 image streams are not already available in the OpenShift enfironment.
- Dynamic persistent volume (PV) provisioning must be enabled. Alternatively, if dynamic PV provisioning is not enabled, enough persistent volumes must be available. By default, the following sizes are required:
  - Each deployed replicated set of Process Server pods, by default, requires one 1Gi PV for the database. You can change the database PV size in the playbook parameters. You can deploy multiple immutable servers; each requires a separate database PV. This requirement does not apply if you use an external database server.
  - By default, Business Central requires one 1Gi PV. You can change the PV size for Business Central persistent storage in the playbook parameters.
  - Business Central Monitoring requires one 64Mi PV.
  - Smart Router requires one 64Mi PV.
- 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

You can deploy Red Hat Process Automation Manager into a Red Hat OpenShift Container Platform 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 of Red Hat Process Automation Manager are available on OpenShift:

- Process Server, also known as *Execution Server* or *KIE Server*, is the infrastructure element that runs decision services, process applications, and other deployable assets (collectively referred to as *services*) . All logic of the services 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 as necessary, running on the same host or different hosts. As you scale a pod up or down, all its copies use the same database server and run the same services. OpenShift provides load balancing and a request can be handled by any of the pods.

  You can deploy a separate Process Server pod to run a different group of services. That pod 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 services. It also provides a management and monitoring console. You can use Business Central to develop services and deploy them to Process Servers. You can also use Business Central to monitor the execution of processes.
  Business Central is a centralized application. However, you can configure it for high availability, where multiple pods run and share the same data.

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

  IMPORTANT

  In the current version, high-availability Business Central functionality is for Technology Preview only. For more information on Red Hat Technology Preview features, see Technology Preview Features Scope.

- Business Central Monitoring is a web-based management and monitoring console. It can manage deployment of services 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 services 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 service. 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.

The following environment types are typical:

- *Authoring*: An environment for creating and modifying services using Business Central. It consists of pods that provide Business Central for the authoring work and a Process Server for test execution of the services.

- *Managed deployment*: An environment for running existing services for staging and production purposes. This environment includes several groups of Process Server pods; you can deploy and undeploy services on every such group and also scale the group up or down as necessary. Use Business Central Monitoring to deploy, run, and stop the services and to monitor their execution.

- *Deployment with immutable servers*: An alternate environment for running existing services for staging and production purposes. In this environment, when you deploy a Process Server pod, it builds an image that loads and starts a service or group of services. You cannot stop any service on the pod or add any new service to the pod. If you want to use another version of a service or modify the configuration in any other way, you deploy a new server image and displace the old one. In this system, the Process Server runs like any other pod on the OpenShift environment; you can use any 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 service instances, but not to deploy additional services to any Process Server or undeploy any existing ones (you can not add or remove containers).

You can also deploy a *trial* or evaluation environment. This environment includes Business Central and a Process Server. You can set it up quickly and use it to evaluate or demonstrate developing and running assets. However, the environment does not use any persistent storage, and any work you do in the environment is not saved.

You can use the Automation Broker with the Red Hat Process Automation Manager Ansible Playbook to deploy a Red Hat Process Automation Manager environment on OpenShift in interactive mode. You can set all possible configuration values during this procedure. During the installation, the Automation Broker can generate all the required secrets automatically. However, for production environments, you need to create correct secrets before the installation.

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 AND THE IMAGE REGISTRY

To deploy Red Hat Process Automation Manager components on Red Hat OpenShift Container Platform, you must ensure that OpenShift can download the correct images from the Red Hat registry. To download the images, OpenShift requires the information about their location (known as *image streams*). OpenShift also must be configured to authenticate with the Red Hat registry using your service account user name and password.

Some versions of the OpenShift environment include the required image streams. You must check if they are available. If image streams are available in OpenShift by default, you can use them if the OpenShift infrastructure is configured for registry authentication server. The administrator must complete the registry authentication configuration when installing the OpenShift environment.

Otherwise, you must configure registry authentication and install the image streams in the **openshift** namespace. You must have administrator access to your OpenShift environment to make these changes.

Procedure

1. Determine whether Red Hat OpenShift Container Platform is configured with the user name and password for Red Hat registry access. For details about the required configuration, see Configuring a Registry Location. If you are using an OpenShift Online subscription, it is configured for Red Hat registry access.

2. If Red Hat OpenShift Container Platform is configured with the user name and password for Red Hat registry access, run the following commands:

   ```
   $ oc get imagestreamtag -n openshift | grep rhpam73-businesscentral
   $ oc get imagestreamtag -n openshift | grep rhpam73-kieserver
   ```

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

3. If the output of one or both of the commands is empty or if OpenShift is not configured with the user name and password for Red Hat registry access, complete the following steps:

   a. Log in to OpenShift with the **oc** command as a user with administrator permissions.

   b. Complete the steps documented in Registry Service Accounts for Shared Environments. You must log in to Red Hat Customer Portal to access the document and to complete the steps to create a registry service account.

   c. Select the **OpenShift Secret** tab and click the link under **Download secret** to download the YAML secret file.

d. View the downloaded file and note the name that is listed in the **name:** entry.

e. Run the following commands:

```
oc create -f <file_name>.yaml -n openshift
oc secrets link default <secret_name> --for=pull -n openshift
oc secrets link builder <secret_name> --for=pull -n openshift
```

Where **<file_name>** is the name of the downloaded file and <secret_name> is the name that is listed in the **name:** entry of the file.

f. Download the **rhpam-7.3.0-openshift-templates.zip** product deliverable file from the Software Downloads page and extract the **rhpam73-image-streams.yaml** file.

g. Complete one of the following actions:

- Run the following command:

  ```
  $ oc create -f rhpam73-image-streams.yaml -n openshift
  ```

- Using the OpenShift Web UI, select **Add to Project → Import YAML / JSON**and 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. For more information about OpenShift secrets, see the Secrets chapter in the OpenShift documentation.

Process Server uses an SSL certificate to provide HTTPS access. The deployment can create a sample secret automatically. However, in production environments 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.
   For more information on how to create a keystore with self-signed or purchased SSL certificates, see Generate a SSL Encryption Key and Certificate .

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, note that this component uses an SSL certificate to provide HTTPS access. The deployment can create a sample secret automatically. However, in production environments 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.
   For more information on how to create a keystore with self-signed or purchased SSL certificates, see Generate a SSL Encryption Key and Certificate .

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. CREATING THE SECRETS FOR SMART ROUTER

If you are planning to deploy Smart Router in your OpenShift environment, note that this component uses an SSL certificate to provide HTTPS access. The deployment can create a sample secret automatically. However, in production environments you must create an SSL certificate for Smart Router and provide it to your OpenShift environment as a secret. Do not use the same certificate and keystore for Smart Router as the ones used for Process Server or Business Central.

Procedure

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

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

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

2.5. 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-7.3.0-openshift-templates.zip** product deliverable file from the Software Downloads page. 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
```

2.6. 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
   ```

   Replace **<class-name>** with 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>
```

Replace **<class-name>** with 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. DEPLOYING A RED HAT PROCESS AUTOMATION MANAGER ENVIRONMENT USING THE AUTOMATION BROKER

To deploy a Red Hat Process Automation Manager environment using the Automation Broker, you must find the Ansible Playbook in the OpenShift catalog, run it, and set the parameters as required.

Procedure

1. In the OpenShift Web UI, select **Add to Project → Browse Catalog**
2. In the search field, type **Red Hat Process Automation Manager**.
3. Select the **Red Hat Process Automation Manager 7.3 APB** catalog entry.
4. Click **Next**.
5. Select the required architecture elements, as described on the screen, and click the **Next** button.

   IMPORTANT

   If you want to deploy an environment with immutable servers and a monitoring infrastructure, you must first install the **Immutable Server - Monitor** option and then the **Immutable Server - KIE Process Server** option.

6. Enter the parameters as described on the screen. In most cases, the default values lead to a working deployment; you can modify them as necessary. You must enter the following settings:
   - For the **Immutable Server - Monitor** option:
     - The **Maven repository URL** field. You must provide a Maven repository with the same versions of all the artifacts that are deployed on any monitored immutable servers.
     - The **Admin password** field. You must record the administrative user name and password to configure monitored servers to connect to Business Central Monitoring.
   - For the **Immutable Server - KIE Process Server** option:
     - The **KIE Server Container Deployment**, **Git Repository URL**, and **Git Repository Reference** fields. These settings determine the source code that the deployment process builds and deploys on the Process Server.
     - If you deployed the **Immutable Server - Monitor** option and want to connect the server to the monitoring infrastructure:
       - Under **Router integration**, the service name of the **rhpam-immutable-mon-smartrouter** service.
       - Under **Controller integration**, the service name of the **rhpam-immutable-mon-rhpamcentrmon** service and the admin username and password that you set in the **Immutable Server - KIE Process Server** option.

IMPORTANT

Avoid each of the following combinations of settings. These combinations produce an invalid environment.

- **Process server>Database type H2** and **Process server>Number of replicas** exceeding **1**.
- **Process server>Database type External** and **Process server>Sets of Process Servers** exceeding **1**.
- **Red Hat - Single Sign-On** configured and **Process server>Sets of Process Servers** exceeding **1**.

7. If you are using the **External** database type for the Process Server in the **Authoring**, **Immutable Server - Process Server**, or **Managed Environment** option, set the parameters under the **External Database** heading. Set the host, port, database name, and database JDBC URL to the correct values for your database server. Use the following values for the other fields:
   - **Driver name**: The driver for the server, depending on the server type:
     - mysql
     - postgresql
     - mariadb
     - mssql
     - db2
     - oracle
     - sybase
   - **Dialect class** (**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**
8. If you created a custom image for using an external database server other than MySQL or PostgreSQL, as described in Section 2.5, “Building a custom Process Server image for an external database”, you must also set the **Process Server Image Stream Name** parameter (under the **Process Server** heading) to the following value:
   - For Microsoft SQL Server, **rhpam73-kieserver-mssql-openshift**
   - For MariaDB, **rhpam73-kieserver-mariadb-openshift**

- For IBM DB2, **rhpam73-kieserver-db2-openshift**
- For Oracle Database, **rhpam73-kieserver-oracle-openshift**
- For Sybase, **rhpam73-kieserver-sybase-openshift**

9. Click **Next** to commence deployment.
   After deploying the environment, you can access it using the HTTPS routes displayed in the OpenShift Web console. HTTP requests are redirected to HTTPS.

IMPORTANT

After deploying the service, you can scale the Process Server pods up and down as necessary. Do not scale the database pods.

CHAPTER 4. DEPROVISIONING A RED HAT PROCESS AUTOMATION MANAGER ENVIRONMENT USING THE AUTOMATION BROKER

If you deployed a Red Hat Process Automation Manager environment using the Automation Broker, you can also use the Broker to deprovision the environment.

Procedure

1. In the OpenShift Web UI, view the **Overview** page for your project.
2. Locate the Service Instance for the Red Hat Process Automation Manager environment that you want to remove.
3. Click the menu button at the instance and select **Delete**. The Automation Broker runs the deprovisioning playbook.

APPENDIX A. VERSIONING INFORMATION

Documentation last updated on Monday, March 01, 2021.