Red Hat JBoss Data Virtualization 6.4

Red Hat JBoss Data Virtualization for OpenShift

Learn how to use Red Hat JBoss Data Virtualization with OpenShift.

Last Updated: 2019-06-24

Red Hat JBoss Data Virtualization 6.4 Red Hat JBoss Data Virtualization for OpenShift

Learn how to use Red Hat JBoss Data Virtualization with OpenShift.

Documentation Team

Legal Notice

Abstract

Table of Contents

CHAPTER 1. BEFORE YOU BEGIN 3
1.1. WHAT IS RED HAT JBOSS DATA VIRTUALIZATION FOR OPENSHIFT? 3
1.2. VERSION COMPATIBILITY AND SUPPORT 3

CHAPTER 2. YOUR FIRST STEPS 4
2.1. OBTAIN THE LATEST JDV FOR OPENSHIFT IMAGE 4
2.2. PREPARING JDV PROJECT ARTIFACTS 4
2.2.1. How to add your configuration artifacts to the OpenShift image 4
2.2.2. How to add your runtime artifacts to the OpenShift image 4
2.2.2.1. Data sources artifacts 4
2.2.2.2. Resource adapter artifacts 5
2.3. HOW TO CONFIGURE SECRETS 6
2.4. USING DATA GRID FOR OPENSHIFT WITH JDV FOR OPENSHIFT 7
2.4.1. JDG for OpenShift authentication environment variables 8
2.4.2. JDG for OpenShift resource adapter properties 8

CHAPTER 3. TUTORIALS 11
3.1. PREREQUISITE 11
3.2. DEPLOY YOUR PROJECT USING THE RED HAT JBOSS DATA VIRTUALIZATION FOR OPENSHIFT IMAGE 11
3.3. HOW TO USE A CACHE AS A MATERIALIZATION TARGET 12

CHAPTER 1. BEFORE YOU BEGIN

1.1. WHAT IS RED HAT JBOSS DATA VIRTUALIZATION FOR OPENSHIFT?

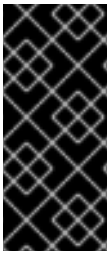
Red Hat JBoss Data Virtualization for OpenShift is a Red Hat JBoss Data Virtualization (JDV) image that runs in an OpenShift container. It allows you to quickly and easily create, deploy and scale Red Hat JBoss Data Virtualization sessions in a secure environment.

1.2. VERSION COMPATIBILITY AND SUPPORT

To see which versions of OpenShift and JDV are compatible, click here: OpenShift and Atomic Platform Tested Integrations.

The Red Hat JBoss Data Virtualization for OpenShift image is based on Red Hat JBoss Data Virtualization 6.4. There are some differences in functionality between the JDV for OpenShift image and the standalone version of Red Hat JBoss Data Virtualization:

- Cached results are automatically replicated to every member of the JDV for OpenShift cluster.
- The JDV for OpenShift image is built on the Red Hat JBoss Enterprise Application Platform for OpenShift image and has inherited its differences. To read about them, click here: Comparison: EAP and EAP for OpenShift Image.



IMPORTANT

The Red Hat Data Grid (RHDG) Hot Rod client is built into the JDV image. Hot Rod clients are compatible with specific Data Grid versions.

You must install the version of the JDV image that corresponds to the version of your RHDG instances.

JDV version	JDV image tag	Bundled Hot Rod client version	RHDG version
JDV 6.4.6	1.5	infinispan-client-hotrod-8.5.0	RHDG 7.2
JDV 6.4.7 and later	1.6	infinispan-client-hotrod-9.4.13	RHDG 7.3

CHAPTER 2. YOUR FIRST STEPS

2.1. OBTAIN THE LATEST JDV FOR OPENSHIFT IMAGE

To download the JDV for OpenShift image and application templates, click here: Red Hat Registry .

2.2. PREPARING JDV PROJECT ARTIFACTS

2.2.1. How to add your configuration artifacts to the OpenShift image

A simple way to add your artifacts, such as the virtual databases files, modules, drivers, translators, and additional generic deployments, to the image is to include them in the application source deployment directory. The artifacts are downloaded during the build process and injected into the image. This configuration is built into the image when these artifacts are uploaded so that only the data sources and associated resource adapters need to be added at runtime.

To deploy a virtual database, create an empty marker file in the same directory and with the same name as the VDB but with the additional extension **.dodeploy**. For example, if the VDB file is called **database.vdb**, the marker file must be called **database.vdb.dodeploy**.

2.2.2. How to add your runtime artifacts to the OpenShift image

Runtime artifacts from environment files are provided through the **OpenShift Secret** mechanism. They are referenced in environment files that are, in turn, referenced in your JDV template.

JDV application template	Description
datavirt64-basic	This is an application template for JBoss Data Virtualization 6.4 services built using S2I.
datavirt64-secure	This template allows you to configure certificates for serving secure content.
datavirt64-extensions-support	This template allows you to install extensions (such as third-party database drivers) and configure certificates to serve secure content.

2.2.2.1. Data sources artifacts

There are three types of data sources:

1. Default internal data sources. These are PostgreSQL, MySQL, and MongoDB databases. These data sources are available on OpenShift by default through the **Red Hat Registry** so you do not need to configure any additional environment files. Set the environment variable to the name of the OpenShift service for the database to be discovered and used as a data source. For more information, click here: DB_SERVICE_PREFIX_MAPPING

2. Other internal data sources. These are not available by default through the **Red Hat Registry** but do run on OpenShift. To add these data sources, you must supply environment files to **OpenShift Secrets**.

3. External data sources that are not run on OpenShift. To add these data sources you must supply environment files to **OpenShift Secrets**.

Here is an example data source environment file:

```
# derby datasource
ACCOUNTS_DERBY_DATABASE=accounts
ACCOUNTS_DERBY_JNDI=java:/accounts-ds
ACCOUNTS_DERBY_DRIVER=derby
ACCOUNTS_DERBY_USERNAME=derby
ACCOUNTS_DERBY_PASSWORD=derby
ACCOUNTS_DERBY_TX_ISOLATION=TRANSACTION_READ_UNCOMMITTED
ACCOUNTS_DERBY_JTA=true

# Connection info for an xa datasource
ACCOUNTS_DERBY_XA_CONNECTION_PROPERTY_DatabaseName=/opt/eap/standalone/data/da
tabases/derby/accounts

# _HOST and _PORT are required, but not used
ACCOUNTS_DERBY_SERVICE_HOST=dummy
ACCOUNTS_DERBY_SERVICE_PORT=1527
```

The **DATASOURCES** property is a comma-separated list of data source property prefixes. These prefixes are appended to every property belonging that data source. Multiple data sources can then be included in a single environment file. (Alternatively, each data source can be provided in a separate environment file.)

Data sources contain two property types: connection pool-specific properties and data driver-specific properties. In the above example, **ACCOUNTS** is the data source prefix, **XA_CONNECTION_PROPERTY** is the generic driver property, and **DatabaseName** is the property specific to the driver.

After you add the environment files to **OpenShift Secrets**, they are called within the JDV template using the **ENV_FILES** environment property, the value of which is a comma-separated list of fully qualified environment files:

```
{
   “Name”: “ENV_FILES”,
   “Value”: “/etc/jdv-extensions/datasources1.env,/etc/jdv-extensions/datasources2.env”
}
```

2.2.2.2. Resource adapter artifacts

To add a resource adapter, you must supply an environment file to **OpenShift Secrets**:

```
#RESOURCE_ADAPTER
RESOURCE_ADAPTERS=QSFILE

QSFILE_ID=fileQS
QSFILE_MODULE_SLOT=main
QSFILE_MODULE_ID=org.jboss.teiid.resource-adapter.file
QSFILE_CONNECTION_CLASS=org.teiid.resource.adapter.file.FileManagedConnectionFactory
QSFILE_CONNECTION_JNDI=java:/marketdata-file
QSFILE_PROPERTY_ParentDirectory=/home/jboss/source/injected/injected-files/data
QSFILE_PROPERTY_AllowParentPaths=true
```

The **RESOURCE_ADAPTERS** property is a comma-separated list of resource adapter property prefixes. These prefixes are appended to all properties for that resource adapter. Multiple resource adapter can then be included in a single environment file.

The resource adapter environment files are added to the **OpenShift Secret** for the project namespace. These environment files are then called within the JDV template using the **ENV_FILES** environment property, the value of which is a comma-separated list of fully-qualified environment files:

```
{
   “Name”: “ENV_FILES”,
   “Value”: “/etc/jdv-extensions/resourceadapter1.env,/etc/jdv-extensions/resourceadapter2.env”
}
```

2.3. HOW TO CONFIGURE SECRETS

Before you begin, you must have configured two keystores:

- A secure socket layer (SSL) keystore to provide private and public keys for https traffic encryption.
- A JGroups keystore to provide private and public keys for network traffic encryption between nodes in the cluster.

WARNING

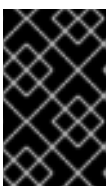
Self-signed certificates do not provide secure communication and are intended for internal testing purposes. For production environments Red Hat recommends that you use your own SSL certificate purchased from a verified Certificate Authority (CA) for SSL-encrypted connections (HTTPS).

1. Create a **secret** to hold the two keystores that provide authorization to applications in the project:

```
$ oc secret new <jdv-secret-name> <ssl.jks> <jgroups.jceks>
```

2. Send your runtime artifacts to the JDV for OpenShift image using the **OpenShift Secrets** mechanism. (These files need to be present locally so that the secrets can be created for them.)

```
$ oc secrets new <datavirt-app-config> <datasource.env> <resourceadapter.env>
<additional/data/files/>
```



IMPORTANT

If the project does not require any runtime artifacts, the secret must still be present in the OpenShift project or the deployment will fail. You can create an empty secret:

```
$ touch <empty.env>
$ oc secrets new <datavirt-app-config> <empty.env>
```

3. Create a service account:

```
$ oc create serviceaccount <service-account-name>
```

4. Add the **view** role to the service account:

```
$ oc policy add-role-to-user view system:serviceaccount:<project-name>:<service-account-name> -n <project-name>
```

5. Add the project's secrets to the account:

```
$ oc secret link <service-account-name> <jdv-secret-name> <jdv-datasource-secret> <jdv-resourceadapter-secret> <jdv-datafiles-secret>
```

6. To use Red Hat Single Sign-On (SSO) for authentication, use the **datavirt64-secure-s2i** application template. For more information on configuring SSO, click here: Automatic and Manual SSO Client Registration Methods

2.4. USING DATA GRID FOR OPENSHIFT WITH JDV FOR OPENSHIFT

IMPORTANT

Before you start spinning up clusters, make sure your JDV and RHDG instance versions are aligned for Hot Rod client compatibility. See Version Compatibility with Red Hat Data Grid.

There are two use cases for integration:

- You want to use JDG as a data source for JDV.
- You want to use JDG as an external materialization target for JDV. When deployed as a materialization target, JDG for OpenShift uses in-memory caching to store results from common queries to other remote databases, increasing performance.

In each of these use cases, both the JDG for OpenShift and JDV for OpenShift deployments need to be configured. The environment variable to specify these cache names is different depending on whether JDG for OpenShift is to be used as a data source or a materialization:

Using JDG for OpenShift as a data source

CACHE_NAMES

Comma-separated list of the cache names to be used for the JDG for OpenShift data source.

Using JDG for OpenShift as a materialization target

DATAVIRT_CACHE_NAMES

This is a comma-separated list of the cache names to be used for the JDG for OpenShift materialization target. When the image is built, two caches will be created per cache name provided: {cachename} and ST_{cachename}. The required cache of teiid-alias-naming-cache, will also be

created. These three caches enable JBoss Data Grid to simultaneously maintain and refresh materialization caches.

IMPORTANT

Red Hat JBoss Data Grid, and the JDG for OpenShift image, support multiple protocols; however, when deployed with JDV for OpenShift, only the Hot Rod protocol is supported. This protocol is not encrypted.

For more information on the Hot Rod protocol, click here: Remote Querying chapter.

2.4.1. JDG for OpenShift authentication environment variables

To use JDG for OpenShift as an authenticated data source, additional environment variables must be provided in the JDG for OpenShift application template. These environment variables provide authentication details for the Hot Rod protocol and authorization details for the caches in the JDG for OpenShift deployment:

Environment Variable	Description	Example value
USERNAME	This is the username for the JDG user.	jdg-user
PASSWORD	This is the password for the JDG user.	JBoss.123
HOTROD_AUTHENTICATION	This enablea Hot Rod authentication.	true
CONTAINER_SECURITY_ROLE_MAPPER	This is the role mapper for the Hot Rod protocol.	identity-role-mapper
CONTAINER_SECURITY_ROLES	This provides security roles and permissions for the role mapper.	admin=ALL

These resource adapter properties can also be configured:

Resource Adapter Property	Description	Example value
<cache-name>_CACHE_SECURITY_AUTHORIZATION_ENABLED	This enables authorization checks for the cache.	true
<cache-name>_CACHE_SECURITY_AUTHORIZATION_ROLES	This sets the valid roles required to access the cache.	admin

2.4.2. JDG for OpenShift resource adapter properties

To use JDG for OpenShift with JDV for OpenShift, properties specific to JDG are required within a

resource adapter. As with all resource adapters, these can be included as a separate resource adapter environment file or along with other resource adapters in a larger environment file and supplied to the build as an OpenShift secret.

Here are the standard properties required by JDV for OpenShift to configure a resource adapter:

```
RESOURCE_ADAPTERS={RA_NAME1},{RA_NAME2},..
{RA_NAME1}_ID
{RA_NAME1}_MODULE_SLOT
{RA_NAME1}_MODULE_ID
{RA_NAME1}_CONNECTION_CLASS
{RA_NAME1}_CONNECTION_JNDI
```

RA_NAME1 is the user-defined name of the resource adapter, which will be used as the prefix to defining the properties associated with that resource adapter.

Additional properties for the JDG resource adapter

Resource adapter property	Description	Required
RemoteServerList	Server List (host:port[;host:port...]) with which to connect.	Yes.

Additional resource adapter for using JDG for OpenShift as a data source

Resource adapter property	Description	Required
UserName	SASL mechanisms defined for the JDG Hot Rod connector.	This is true if you are using JDG as an authenticated data source.
Password	SASL mechanisms defined for the JDG Hot Rod connector.	This is true if you are using JDG as an authenticated data source.
AuthenticationRealm	Security realm defined for the Hot Rod connector.	This is true if you are using JDG as an authenticated data source.
AuthenticationServerName	SASL server name defined for the Hot Rod connector.	This is true if you are using JDG as an authenticated data source.
SASLMechanism	SASL mechanisms defined for the JDG Hot Rod connector.	This is true if you are using JDG as an authenticated data source.

Here is a resource adapter you can use to integrate JDG with OpenShift:

```
RESOURCE_ADAPTERS=MAT_CACHE
MAT_CACHE_ID=infinispanDS
MAT_CACHE_MODULE_SLOT=main
MAT_CACHE_MODULE_ID=org.jboss.teiid.resource-adapter.infinispan.hotrod
MAT_CACHE_CONNECTION_CLASS=org.teiid.resource.adapter.infinispan.hotrod.InfinispanManaged
```

```
ConnectionFactory MAT_CACHE_CONNECTION_JNDI=java:/infinispanRemoteDSL
MAT_CACHE_PROPERTY_RemoteServerList=${DATAGRID_APP_HOTROD_SERVICE_HOST}:${
DATAGRID_APP_HOTROD_SERVICE_PORT}
```

Here is a resource adapter you can use to make JDG for OpenShift a data source:

```
RESOURCE_ADAPTERS=MAT_CACHE
MAT_CACHE_ID=infinispanDS
MAT_CACHE_MODULE_SLOT=main
MAT_CACHE_MODULE_ID=org.jboss.teiid.resource-adapter.infinispan.hotrod
MAT_CACHE_CONNECTION_CLASS=org.teiid.resource.adapter.infinispan.hotrod.InfinispanManaged
ConnectionFactory MAT_CACHE_CONNECTION_JNDI=java:/infinispanRemoteDSL
MAT_CACHE_PROPERTY_RemoteServerList=${DATAGRID_APP_HOTROD_SERVICE_HOST}:${
DATAGRID_APP_HOTROD_SERVICE_PORT}

MAT_CACHE_PROPERTY_UserName=jdg
MAT_CACHE_PROPERTY_Password=JBoss.123
MAT_CACHE_PROPERTY_AuthenticationRealm=ApplicationRealm
MAT_CACHE_PROPERTY_AuthenticationServerName=jdg-server
MAT_CACHE_PROPERTY_SaslMechanism=DIGEST-MD5
```

Line breaks separate the standard JDV for OpenShift resource adapter configuration, the additional properties required for JDG for OpenShift, and the authentication properties for the JDG data source. The **PROPERTY_UserName** and its associated password correspond to the values provided in the JDG for OpenShift application template.

CHAPTER 3. TUTORIALS

3.1. PREREQUISITE

Before undertaking these tutorials, you must configure your environment correctly.

See the OpenShift Primer for instructions.

3.2. DEPLOY YOUR PROJECT USING THE RED HAT JBOSS DATA VIRTUALIZATION FOR OPENSHIFT IMAGE

Use this workflow to create and deploy a project. As an example, the tutorial uses the **dynamicvdb-datafederation** quickstart which combines data from a relational source (H2) and a Microsoft Excel file. After deploying it, you will learn how to switch data sources.

1. Create a new project:

```
$ oc new-project jdv-app-demo
```

2. Create a service account to be used for the Red Hat JBoss Data Virtualization for OpenShift deployment:

```
$ oc create serviceaccount datavirt-service-account
```

3. Add the view role to the service account:

```
$ oc policy add-role-to-user view system:serviceaccount:jdv-app-demo:datavirt-service-account
```

4. The Red Hat JBoss Data Virtualization for OpenShift template requires SSL and JGroups keystores.
(These keystores are required even if the application will not use https.)
The following commands will prompt you for passwords:

   a. Generate a secure key for the SSL keystore:

```
$ keytool -genkeypair -alias https -storetype JKS -keystore keystore.jks
```

   b. Generate a secure key for the JGroups keystore:

```
$ keytool -genseckey -alias jgroups -storetype JCEKS -keystore jgroups.jceks
```

5. Use the SSL and JGroup keystore files to create the keystore secret for the project:

```
$ oc secret new datavirt-app-secret keystore.jks jgroups.jceks
```

6. Create a secret with the **datasources.env** file:

```
git clone https://github.com/jboss-openshift/openshift-quickstarts/blob/master/datavirt/dynamicvdb-datafederation/datasources.env
```

```
$ oc secrets new datavirt-app-config datasources.env
```

7. Link the keystore and environment secrets to the service account:

```
$ oc secrets link datavirt-service-account datavirt-app-secret datavirt-app-config
```

8. Log in to the OpenShift Web Console: https://127.0.0.1:8443
9. Click **jdv-app-demo**.
10. Click **Add to Project**.
11. Click **Browse Catalog**.
12. Enter **datavirt** in the **Filter by keyword** search bar.
13. Click **basic-s2i**.
14. Enter these parameters:
Git Repository URL: https://github.com/jboss-openshift/openshift-quickstarts
Git Reference: master
CONTEXT_DIR: datavirt64/dynamicvdb-datafederation/app
15. Click **Deploy**.
16. Switch to using a MySQL data source instead of H2:

```
$ oc env dc/datavirt-app QS_DB_TYPE=mysql5
```

3.3. HOW TO USE A CACHE AS A MATERIALIZATION TARGET

Having deployed the **dynamicvdb-datafederation** quickstart, you can now deploy a Red Hat JBoss Data Grid instance. This allows you to quickly query the cache for data, without having to go back to the original sources.

You can use any of the **Red Hat JBoss Data Grid for OpenShift**templates, but it is better to use non-persistent templates as these provide you with clustering and high availability functionality. To obtain them, click here: https://github.com/jboss-container-images/jboss-datavirt-6-openshift-image/tree/datavirt64/resources/openshift/templates

1. Enter the **jdv-app-demo** project:

```
$ oc project jdv-app-demo
```

2. Create a service account for JDG for OpenShift:

```
$ oc create serviceaccount datagrid-service-account
```

3. Add **view role** permissions to the service account:

```
$ oc policy add-role-to-user view system:serviceaccount:jdv-app-demo:datagrid-service-account
```

4. Create the keystore secret for the project:

```
$ oc secret new datagrid-app-secret jgroups.jceks
```

5. Link the keystore secret to the service account:

```
$ oc secrets link datagrid-service-account datagrid-app-secret
```

6. Log in to the **OpenShift Web Console**.
7. Click the **jdv-app-demo** project space.
8. Click **Add to Project**.
9. Enter **datagrid** in the **Filter by keyword** search bar.
10. Click the **datagrid71-https** template.
11. In the **DATAVIRT_CACHE_NAMES** environment variable field, enter **stockCache**.
12. In the **CACHE_TYPE_DEFAULT** environment variable field, enter **replicated**.
13. Enter these parameters:
    USERNAME: jdg
    PASSWORD: JBoss.123
    JDG User Roles/Groups (ADMIN_GROUP): admin,___schema_manager
    HOTROD_AUTHENTICATION: true
    CONTAINERSECURITYROLE_MAPPER: identity-role-mapper
    CONTAINER_SECURITY_ROLES:"admin=ALL,jdg=ALL"
14. Click **Deploy**.
15. Set JDG as the materialization target:

```
$ oc env bc/datavirt-app DATAGRID_MATERIALIZATION=true
```