Red Hat Data Grid 8.0

Data Grid Library Mode

Data Grid Documentation
Abstract

Use Data Grid as an embedded library in custom applications.
# Table of Contents

1. **RED HAT DATA GRID** ................................................................. 2  
   1.1. Data Grid Documentation ................................................. 2  
   1.2. Data Grid Downloads .................................................. 2  

2. **CONFIGURING THE DATA GRID MAVEN REPOSITORY** ......................... 2  
   2.1. Downloading the Data Grid Maven Repository ......................... 2  
   2.2. Adding the Red Hat GA Maven Repository ............................... 3  
   2.3. Configuring Your Data Grid POM ..................................... 3  

3. **INSTALLING DATA GRID IN LIBRARY MODE** ........................................ 4  

4. **RUNNING DATA GRID AS AN EMBEDDED LIBRARY** ................................. 4
1. RED HAT DATA GRID

Data Grid is a high-performance, distributed in-memory data store.

**Schemaless data structure**
- Flexibility to store different objects as key-value pairs.

**Grid-based data storage**
- Designed to distribute and replicate data across clusters.

**Elastic scaling**
- Dynamically adjust the number of nodes to meet demand without service disruption.

**Data interoperability**
- Store, retrieve, and query data in the grid from different endpoints.

1.1. Data Grid Documentation

Documentation for Data Grid is available on the Red Hat customer portal.

- Data Grid 8.0 Documentation
- Data Grid 8.0 Component Details
- Supported Configurations for Data Grid 8.0

1.2. Data Grid Downloads

Access the Data Grid Software Downloads on the Red Hat customer portal.

NOTE

You must have a Red Hat account to access and download Data Grid software.

2. CONFIGURING THE DATA GRID MAVEN REPOSITORY

Data Grid Java distributions are available from Maven.

You can download the Data Grid Maven repository from the customer portal or pull Data Grid dependencies from the public Red Hat Enterprise Maven repository.

2.1. Downloading the Data Grid Maven Repository

Download and install the Data Grid Maven repository to a local file system, Apache HTTP server, or Maven repository manager if you do not want to use the public Red Hat Enterprise Maven repository.

Procedure

1. Log in to the Red Hat customer portal.
2. Navigate to the Software Downloads for Data Grid.

3. Download the Red Hat Data Grid 8.0 Maven Repository.

4. Extract the archived Maven repository to your local file system.

5. Open the README.md file and follow the appropriate installation instructions.

2.2. Adding the Red Hat GA Maven Repository

Configure your Maven settings file, typically ~/.m2/settings.xml, to include the Red Hat GA repository. Alternatively, include the repository directly in your project pom.xml file.

The following configuration uses the public Red Hat Enterprise Maven repository. To use the Data Grid Maven repository that you downloaded from the Red Hat customer portal, change the value of url elements to the correct location.

```xml
<repositories>
  <repository>
    <id>redhat-ga</id>
    <name>Red Hat GA Repository</name>
    <url>https://maven.repository.redhat.com/ga/</url>
  </repository>
</repositories>

<pluginRepositories>
  <pluginRepository>
    <id>redhat-ga</id>
    <name>Red Hat GA Repository</name>
    <url>https://maven.repository.redhat.com/ga/</url>
  </pluginRepository>
</pluginRepositories>
```

Reference

- Red Hat Enterprise Maven Repository

2.3. Configuring Your Data Grid POM

Maven uses configuration files called Project Object Model (POM) files to define projects and manage builds. POM files are in XML format and describe the module and component dependencies, build order, and targets for the resulting project packaging and output.

Procedure

1. Open your project pom.xml for editing.

2. Define the version.infinispan property with the correct Data Grid version.

3. Include the infinispan-bom in a dependencyManagement section.
   The Bill Of Materials (BOM) controls dependency versions, which avoids version conflicts and means you do not need to set the version for each Data Grid artifact you add as a dependency to your project.

4. Save and close pom.xml.
The following example shows the Data Grid version and BOM:

```xml
<properties>
  <version.infinispan>10.1.8.Final-redhat-00001</version.infinispan>
</properties>

<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.infinispan</groupId>
      <artifactId>infinispan-bom</artifactId>
      <version>${version.infinispan}</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

Next Steps
Add Data Grid artifacts as dependencies to your `pom.xml` as required.

3. INSTALLING DATA GRID IN LIBRARY MODE
Add Data Grid as an embedded library in your project.

**Procedure**
- Add the `infinispan-core` artifact as a dependency in your `pom.xml` as follows:

```xml
<dependency>
  <groupId>org.infinispan</groupId>
  <artifactId>infinispan-core</artifactId>
  <version>${version.infinispan}</version>
</dependency>
```

4. RUNNING DATA GRID AS AN EMBEDDED LIBRARY
Learn how to run Data Grid as an embedded data store in your project.

**Procedure**
- Initialize the default Cache Manager and add a cache definition as follows:

```java
GlobalConfigurationBuilder global = GlobalConfigurationBuilder.defaultClusteredBuilder();
DefaultCacheManager cacheManager = new DefaultCacheManager(global.build());
ConfigurationBuilder builder = new ConfigurationBuilder();
builder.clustering().cacheMode(CacheMode.DIST_SYNC);
cacheManager.administration().withFlags(CacheContainerAdmin.AdminFlag.VOLATILE).getOrCreateCache("myCache", builder.build());
```
The preceding code initializes a default, clustered Cache Manager. Cache Managers contain your cache definitions and control cache lifecycles.

Data Grid does not provide default cache definitions so after initializing the default Cache Manager, you need to add at least one cache instance. This example uses the `ConfigurationBuilder` class to create a cache definition that uses the distributed, synchronous cache mode. You then call the `getOrCreateCache()` method that either creates a cache named "myCache" on all nodes in the cluster or returns it if it already exists.

**Next steps**

Now that you have a running Cache Manager with a cache created, you can add some more cache definitions, put some data into the cache, or configure Data Grid as needed.

**Reference**

- Configuring Data Grid Programmatically
- Setting Up Cluster Transport
- `org.infinispan.Cache`
- `org.infinispan.commons.api.CacheContainerAdmin`
- `org.infinispan.configuration.cache.CacheMode`
- `org.infinispan.configuration.cache.Configuration`
- `org.infinispan.configuration.cache.ConfigurationBuilder`
- `org.infinispan.configuration.global.GlobalConfigurationBuilder`
- `org.infinispan.manager.DefaultCacheManager`
- `org.infinispan.manager.EmbeddedCacheManager`