Red Hat Data Grid 8.4

Data Grid Code Tutorials

Learn how to use Data Grid capabilities

Last Updated: 2024-04-19

Red Hat Data Grid 8.4 Data Grid Code Tutorials

Learn how to use Data Grid capabilities

Legal Notice

Abstract

Run code tutorials for remote caches and embedded caches that demonstrate various Data Grid capabilities and usage patterns.

Table of Contents

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.

DATA GRID DOCUMENTATION

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

- Data Grid 8.4 Documentation
- Data Grid 8.4 Component Details
- Supported Configurations for Data Grid 8.4
- Data Grid 8 Feature Support
- Data Grid Deprecated Features and Functionality

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.

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message.

CHAPTER 1. REMOTE CACHES

Deploy multiple Data Grid Server instances to create remote cache clusters that give you a fault-tolerant and scalable data tier with high-speed access from Hot Rod and REST clients.

1.1. REMOTE CACHE TUTORIALS

To run these tutorials you need at least one locally running instance of Data Grid Server. Each tutorial will try to connect to a running server in **localhost:11222** with **admin**/**password** credentials. However, if a Docker instance is found, and the server is not running, tutorials will spin up a local server with **Testcontainers**.

You can download the distribution and run the following commands:

```
$ ./bin/cli.sh user create admin -p "password"
$ ./bin/server.sh
```

NOTE

Data Grid Server enables authentication and authorization by default. Creating a user named **admin** gives you administrative access to Data Grid Server.

Building and running remote cache tutorials

You can build and run remote cache tutorials directly in your IDE or from the command line as follows:

```
$ mvn -s /path/to/maven-settings.xml clean package exec:exec
```

1.2. HOT ROD JAVA CLIENT TUTORIALS

- Data Grid requires Java 11 at a minimum. However, Hot Rod Java clients running in applications that require Java 8 can continue using older versions of client libraries.

Tutorial link	Description
Remote cache use example	The simplest code example that demonstrates how a remote distributed cache works.
Per cache configuration	Demonstrates how to configure caches dynamically when we connect to the Data Grid Server.
Near caches	Demonstrates how configure near caching to improve the read performance in remote caches.
Cache Admin API	Demonstrates how to use the Administration API to create caches and cache templates dynamically.
Encoding	Demonstrates how encoding of caches work.

Tutorial link	Description
Client listeners	Detect when data changes in a remote cache with Client Listeners.
Query	Demonstrates how to query remote cache values.
Continuous query	Demonstrates how to use Continuous Query and remote caches.
Transactions	Demonstrates how remote transactions work.
Secured caches	Demonstrates how to configure caches that have authorization enabled.
TLS authorization	Demonstrates how to connect to Data Grid Server with TLS authorization.
Counters	Demonstrates how remote counters work.
Multimap	Demonstrates how remote multimap works.
Task execution	Demonstrates how to register server tasks and how to execute them from the Hot Rod client.
JUnit 5 and Testcontainers	Demonstrates how to use the Data Grid and JUnit 5 extension.
Persistence	Demonstrates how to use the Data Grid and persistent caches.
Redis Client	Demonstrates how to use the Data Grid and Redis client to read and write using the Resp protocol.
Reactive API	Demonstrates how to use the Data Grid with the reactive API based on Mutiny.

Data Grid documentation

You can find more resources for Hot Rod Java clients in our documentation at:

- Hot Rod Java client guide
- Marshalling and Encoding Data Guide
- Querying Data Grid caches
- REST API
- Resp Protocol

- Smallrye Mutiny

CHAPTER 2. EMBEDDED CACHES

Add Data Grid as a dependency to your Java project and use embedded caches that increase application performance and give you capabilities to handle complex use cases.

2.1. EMBEDDED CACHE TUTORIALS

You can run embedded cache tutorials directly in your IDE or from the command line as follows:

```
$ mvn -s /path/to/maven-settings.xml clean package exec:exec
```

Tutorial link	Description
Distributed caches	Demonstrates how Distributed Caches work.
Replicated caches	Demonstrates how Replicated Caches work.
Invalidated caches	Demonstrates how Invalidated Caches work.
Transactions	Demonstrates how transactions work.
Streams	Demonstrates how Distributed Streams work.
JCache integration	Demonstrates how JCache works.
Functional Maps	Demonstrates how Functional Map API works.
Map API	Demonstrates how the Map API works with Data Grid caches.
Multimap	Demonstrates how to use Multimap.
Queries	Uses Data Grid Query to perform full-text queries on cache values.
Clustered Listeners	Detects when data changes in an embedded cache with Clustered Listeners.
Counters	Demonstrates how to use an embedded Clustered Counter.
Clustered Locks	Demonstrates how to use an embedded Clustered Lock.
Clustered execution	Demonstrates how to use an embedded Clustered Counter.

Data Grid documentation

You can find more resources about embedded caches in our documentation at:

- Embedding Data Grid Caches
- Querying Data Grid caches

2.2. KUBERNETES AND OPENSHIFT TUTORIAL

This tutorial contains instructions on how to run Infinispan library mode (as a microservice) in Kubernetes/OpenShift.

Prerequisites: Maven and Docker daemon running in the background.

Prerequisites

- A running Openshift or Kubernetes cluster

Building the tutorial

This tutorial is built using the maven command:

```
mvn package
```

Note that **target/** directory contains additional directories like **docker** (with generated Dockerfile) and **classes/META-INF/jkube** with Kubernetes and OpenShift deployment templates.

TIP

If the Docker Daemon is down, the build will omit processing Dockerfiles. Use **docker** profile to turn it on manually.

Deploying the tutorial to Kubernetes

This is handle by the JKube maven plugin, just invoke:

```
mvn k8s:build k8s:push k8s:resource k8s:apply -Doptions.image=<IMAGE_NAME> (1)
```

(1) **IMAGE_NAME** must be replaced with the FQN of the container to deploy to Kubernetes. This container must be created in a repository that you have permissions to push to and is accessible from within your Kubernetes cluster.

Viewing and scaling up

Everything should be up and running at this point. Now login into the OpenShift or Kubernetes cluster and scale the application

```
kubectl scale --replicas=3  deployment/$(kubectl get rs --namespace=myproject | grep infinispan |
awk '{print $1}') --namespace=myproject
```

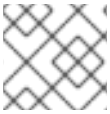
Undeploying the tutorial

This is handled by the JKube maven plugin, just invoke:

```
mvn k8s:undeploy
```

CHAPTER 3. SPRING AND SPRING BOOT

3.1. SPRING AND SPRING BOOT TUTORIALS



NOTE

These code tutorials use Data Grid Server and require at least one running instance.

Run Spring examples

Two simple tutorials can be run with Spring without Spring Boot:

- Test caching

```
$ {package_exec}@spring-caching
```

- Test annotations

```
$ {package_exec}@spring-annotations
```

Run Spring Boot examples

```
$ mvn -s /path/to/maven-settings.xml spring-boot:run
```

Displaying actuator statistics

Navigate to **http://localhost:8080/actuator/metrics** in your browser to display a list of available metrics. Cache metrics are prefixed with **"cache."** Display each metric for each cache using tags. For example for the 'puts' stats in the basque-names cache:

http://localhost:8080/actuator/metrics/cache.puts?tag=name:basque-names

Collecting statistics with Prometheus

The **prometheus.yml** file in this project contains a *host.docker.internal* binding that allows Prometheus to scrap metrics that the Spring actuator exposes.

Change the **YOUR_PATH** value in the following command to the directory where Prometheus is running and then run:

Podman

```
$ podman run -d --name=prometheus -p 9090:9090 -v YOUR_PATH/integrations/spring-boot/prometheus.yml:/etc/prometheus/prometheus.yml prom/prometheus --config.file=/etc/prometheus/prometheus.yml
```

Tutorial link	Description
Spring Boot and Spring Cache remote mode	Demonstrates how to use Spring Caches with Spring Boot and the Data Grid Server.

Tutorial link	Description
Spring Boot and Spring Session remote mode	Demonstrates how to use Spring Session with Spring Boot and the Data Grid Server.
Spring Boot and Spring Cache embedded mode	Demonstrates how to use Spring Caches with Spring Boot and Data Grid Embedded.
Spring Boot and Spring Session embedded mode	Demonstrates how to use Spring Session with Spring Boot and Data Grid Embedded.
Spring cache embedded without Spring Boot	Demonstrates how to use Spring Cache and Data Grid Embedded without Spring Boot.

Data Grid documentation

You can find more resources in our documentation at:

- Using Data Grid with Spring
- Data Grid Spring Boot Starter