Red Hat Data Grid 8.0

Running Data Grid on OpenShift

Data Grid Documentation

Last Updated: 2020-05-29

Red Hat Data Grid 8.0 Running Data Grid on OpenShift

Data Grid Documentation

Legal Notice

Abstract

Get high performance in-memory data storage with Data Grid clusters on OpenShift.

Table of Contents

CHAPTER 1. RED HAT DATA GRID 4
1.1. DATA GRID DOCUMENTATION 4
1.2. DATA GRID DOWNLOADS 4

CHAPTER 2. GETTING STARTED WITH DATA GRID OPERATOR 5
2.1. INFINISPAN CUSTOM RESOURCE (CR) 5
2.2. CREATING DATA GRID CLUSTERS 5
2.3. VERIFYING DATA GRID CLUSTERS 6

CHAPTER 3. CREATING DATA GRID SERVICES 8
3.1. CACHE SERVICE 8
3.1.1. Cache Configuration 8
3.2. DATA GRID SERVICE 9
3.3. CREATING DATA GRID SERVICES 9
3.3.1. Cache Service Resources 9
3.3.2. Data Grid Service Resources 10

CHAPTER 4. STOPPING AND STARTING DATA GRID SERVICES 12
4.1. GRACEFULLY SHUTTING DOWN DATA GRID CLUSTERS 12
4.2. RESTARTING DATA GRID CLUSTERS 12

CHAPTER 5. ADJUSTING CONTAINER SPECIFICATIONS 13
5.1. JVM, CPU, AND MEMORY RESOURCES 13
5.2. STORAGE RESOURCES 13
5.2.1. Persistent Volume Claims 14

CHAPTER 6. CREATING NETWORK SERVICES 15
6.1. GETTING THE SERVICE FOR INTERNAL CONNECTIONS 15
6.2. EXPOSING DATA GRID TO EXTERNAL CLIENTS 15

CHAPTER 7. CONFIGURING AUTHENTICATION 17
7.1. DEFAULT CREDENTIALS 17
7.2. RETRIEVING CREDENTIALS 17
7.3. ADDING CUSTOM CREDENTIALS 17

CHAPTER 8. SECURING DATA GRID CONNECTIONS 19
8.1. USING RED HAT OPENSHIFT SERVICE CERTIFICATES 19
8.1.1. Red Hat OpenShift Service Certificates 19
8.1.2. Retrieving TLS Certificates 20
8.2. USING CUSTOM TLS CERTIFICATES 20
8.2.1. Certificate Secrets 20
8.2.2. Keystore Secrets 21

CHAPTER 9. MONITORING DATA GRID WITH PROMETHEUS 22
9.1. SETTING UP PROMETHEUS 22

CHAPTER 10. CONNECTING TO DATA GRID CLUSTERS 24
10.1. INVOKING THE DATA GRID REST API 24
10.2. CONFIGURING HOT ROD CLIENTS 25

CHAPTER 11. MONITORING DATA GRID LOGS 27
11.1. CONFIGURING DATA GRID LOGGING 27
11.2. LOG LEVELS 27

CHAPTER 12. CONFIGURING CROSS-SITE REPLICATION 29
12.1. DATA GRID CLUSTER AND PROJECT NAMING 29
12.2. CREATING SERVICE ACCOUNT TOKENS 29
12.3. ADDING BACKUP LOCATIONS TO DATA GRID CLUSTERS 30
12.3.1. Cross-Site Replication Resources 31

CHAPTER 13. REFERENCE 32
13.1. NETWORK SERVICES 32
13.2. DATA GRID OPERATOR UPGRADES 32
13.3. TECHNOLOGY PREVIEW 33

CHAPTER 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.

CHAPTER 2. GETTING STARTED WITH DATA GRID OPERATOR

Data Grid Operator lets you create, configure, and manage Data Grid clusters.

Prerequisites

- Create a Data Grid Operator subscription.
- Have an **oc** client.

2.1. INFINISPAN CUSTOM RESOURCE (CR)

Data Grid Operator adds a new Custom Resource (CR) of type **Infinispan** that lets you handle Data Grid clusters as complex units on OpenShift.

You configure Data Grid clusters running on OpenShift by modifying the **Infinispan** CR.

The minimal **Infinispan** CR for Data Grid clusters is as follows:

```
apiVersion: infinispan.org/v1
kind: Infinispan
metadata:
  name: 1
spec:
  replicas: 2
```

1 Names the Data Grid cluster.

2 Sets the number of nodes in the Data Grid cluster.

Reference

Managing resources from Custom Resource Definitions

2.2. CREATING DATA GRID CLUSTERS

Use Data Grid Operator to create clusters of two or more Data Grid nodes.

Procedure

1. Specify the number of Data Grid nodes in the cluster with **spec.replicas** in your **Infinispan** CR. For example, create a **cr_minimal.yaml** file as follows:

```
$ cat > cr_minimal.yaml<<EOF
apiVersion: infinispan.org/v1
kind: Infinispan
metadata:
  name: example-rhdatagrid
spec:
  replicas: 2
EOF
```

2. Apply your **Infinispan** CR.

```
$ oc apply -f cr_minimal.yaml
```

3. Watch Data Grid Operator create the Data Grid nodes.

```
$ oc get pods -w

NAME                    READY  STATUS            RESTARTS   AGE
example-rhdatagrid-1      0/1    ContainerCreating  0          4s
example-rhdatagrid-2      0/1    ContainerCreating  0          4s
example-rhdatagrid-3      0/1    ContainerCreating  0          5s
infinispan-operator-0     1/1    Running            0          3m
example-rhdatagrid-3      1/1    Running            0          8s
example-rhdatagrid-2      1/1    Running            0          8s
example-rhdatagrid-1      1/1    Running            0          8s
```

Next Steps

Try changing the value of **replicas:** and watching Data Grid Operator scale the cluster up or down.

2.3. VERIFYING DATA GRID CLUSTERS

Review log messages to ensure that Data Grid nodes receive clustered views.

Procedure

- Do either of the following:
  - Retrieve the cluster view from logs.

```
$ oc logs example-rhdatagrid-0 | grep ISPN000094

INFO  [org.infinispan.CLUSTER] (MSC service thread 1-2) \
ISPN000094: Received new cluster view for channel infinispan: \
[example-rhdatagrid-0|0] (1) [example-rhdatagrid-0]

INFO  [org.infinispan.CLUSTER] (jgroups-3,{example_crd_name-0) \
ISPN000094: Received new cluster view for channel infinispan: \
[example-rhdatagrid-0|1] (2) [example-rhdatagrid-0, example-rhdatagrid-1]
```

  - Retrieve the **Infinispan** CR for Data Grid Operator.

```
$ oc get infinispan -o yaml
```

The response indicates that Data Grid pods have received clustered views:

```
conditions:
  - message: 'View: [example-rhdatagrid-0, example-rhdatagrid-1]'
    status: "True"
    type: wellFormed
```

TIP

Use **oc wait** with the **wellFormed** condition for automated scripts.

```
$ oc wait --for condition=wellFormed --timeout=240s infinispan/example-rhdatagrid
```

CHAPTER 3. CREATING DATA GRID SERVICES

Data Grid services are stateful applications that provide flexible and robust in-memory data storage.

3.1. CACHE SERVICE

Cache service provides a volatile, low-latency data store that dramatically increases application response rates.

Cache service nodes:

- Synchronously distribute data across the cluster to ensure consistency.
- Maintain single copies of cache entries to reduce size.
- Store cache entries off-heap and use eviction for JVM efficiency.
- Ensure data consistency with a default partition handling configuration.

IMPORTANT

You can create multiple cache definitions with Cache service but only as copies of the default configuration.

If you update Cache service nodes with the **Infinispan** CR or update the version, you lose all data in the cache.

3.1.1. Cache Configuration

Cache service nodes use the following cache configuration:

```
<distributed-cache name="default" 1
                   mode="SYNC" 2
                   owners="1"> 3
  <memory>
   <off-heap eviction="MEMORY" 4
             strategy="REMOVE"/> 5
  </memory>
  <partition-handling when-split="ALLOW_READ_WRITES" 6
                      merge-policy="REMOVE_ALL"/> 7
</distributed-cache>
```

1. Names the cache instance as "default".
2. Uses synchronous distribution for storing data across the cluster.
3. Configures one replica for each cache entry across the cluster.
4. Stores cache entries as bytes in native memory (off-heap).
5. Removes old entries to make space when adding new entries.
6. Specifies a conflict resolution strategy that allows read and write operations for cache entries even if segment owners are in different partitions.

7 Specifies a merge policy that removes entries from the cache when Data Grid detects conflicts.

3.2. DATA GRID SERVICE

Data Grid service provides a configurable Data Grid server distribution for OpenShift.

- Use with advanced capabilities like cross-site replication as well as indexing and querying.
- Remotely access Data Grid service clusters from Hot Rod or REST clients and dynamically create caches using any Data Grid cache mode and configuration.

NOTE

Data Grid does not provide default caches for Data Grid service nodes. However, you can use cache configuration templates to get started.

3.3. CREATING DATA GRID SERVICES

Define the **.spec.service.type** resource to create Cache service and Data Grid service nodes with Data Grid Operator.

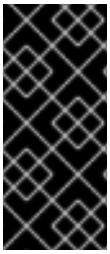
By default, Data Grid Operator creates Data Grid clusters configured as a Cache service.

Procedure

- Specify the service type for Data Grid clusters with **spec.service.type** in your **Infinispan** CR and then apply the changes.

For example, create Data Grid service clusters as follows:

```
spec:
  ...
  service:
    type: DataGrid
```



IMPORTANT

You cannot change **.spec.service.type** after you create Data Grid clusters.

For example, if you create a cluster of Cache service nodes, you cannot change the service type to Data Grid service. In this case you must create a new cluster with Data Grid service nodes in a different OpenShift namespace.

Reference

- Cache Service Resources
- Data Grid Service Resources

3.3.1. Cache Service Resources

```
apiVersion: infinispan.org/v1
```

```
kind: Infinispan
metadata:
  name: example-rhdatagrid 1
spec:
  replicas: 4 2
  service:
    type: Cache 3
  security:
    endpointSecretName: endpoint-identities 4
    endpointEncryption: 5
        type: secret
        certSecretName: tls-secret
  container: 6
    extraJvmOpts: "-XX:NativeMemoryTracking=summary"
    cpu: "2000m"
    memory: 1Gi
  logging: 7
    categories:
      org.infinispan: trace
      org.jgroups: trace
  expose: 8
    type: LoadBalancer
```

1. Names the Data Grid cluster.
2. Specifies the number of nodes in the cluster.
3. Creates Cache service clusters.
4. Adds an authentication secret with user credentials.
5. Adds a custom encryption secret for secure connections.
6. Allocates resources to nodes.
7. Configures logging.
8. Configures services for external traffic.

3.3.2. Data Grid Service Resources

```
apiVersion: infinispan.org/v1
kind: Infinispan
metadata:
  name: example-rhdatagrid 1
spec:
  replicas: 6 2
  service:
    type: DataGrid 3
    container:
      storage: 2Gi 4
    sites: 5
      local:
```

```
      expose:
        type: LoadBalancer
    locations:
    - name: azure
      url: openshift://api.azure.host:6443
      secretName: azure-identities
    - name: aws
      url: openshift://api.aws.host:6443
      secretName: aws-identities
security:
  endpointSecretName: endpoint-identities 6
  endpointEncryption: 7
      type: secret
      certSecretName: tls-secret
container: 8
  extraJvmOpts: "-XX:NativeMemoryTracking=summary"
  cpu: "1000m"
  memory: 1Gi
logging: 9
  categories:
    org.infinispan: debug
    org.jgroups: debug
expose: 10
  type: LoadBalancer
```

1. Names the Data Grid cluster.
2. Specifies the number of nodes in the cluster.
3. Creates Data Grid service clusters.
4. Configures size of the persistent volume.
5. Provides connection information for backup locations.
6. Adds an authentication secret with user credentials.
7. Adds a custom encryption secret for secure connections.
8. Allocates resources to nodes.
9. Configures logging.
10. Configures services for external traffic.

CHAPTER 4. STOPPING AND STARTING DATA GRID SERVICES

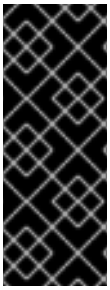
Gracefully shut down Data Grid clusters to avoid data loss.

Cache configuration

Both Cache service and Data Grid service store permanent cache definitions in persistent volumes so they are still available after cluster restarts.

Data

Data Grid service nodes can write all cache entries to persistent storage during cluster shutdown if you add cache stores.



IMPORTANT

You should configure the storage size for Data Grid service nodes to ensure that the persistent volume can hold all your data.

If the available container storage is less than the amount of memory available to Data Grid service nodes, Data Grid writes an exception to logs and data loss occurs during shutdown.

4.1. GRACEFULLY SHUTTING DOWN DATA GRID CLUSTERS

- Set the value of **replicas** to **0** and apply the changes.

```
spec:
  replicas: 0
```

4.2. RESTARTING DATA GRID CLUSTERS

- Set the value of **spec.replicas** to the same number of nodes that were in the cluster before you shut it down.

For example, you shut down a cluster of 6 nodes. When you restart the cluster, you must set:

```
spec:
  replicas: 6
```

This allows Data Grid to restore the distribution of data across the cluster. When all nodes in the cluster are running, you can then add or remove nodes.

CHAPTER 5. ADJUSTING CONTAINER SPECIFICATIONS

You can allocate CPU and memory resources, specify JVM options, and configure storage for Data Grid nodes.

5.1. JVM, CPU, AND MEMORY RESOURCES

```
spec:
  ...
  container:
    extraJvmOpts: "-XX:NativeMemoryTracking=summary" 1
    cpu: "1000m" 2
    memory: 1Gi 3
```

1. Specifies JVM options.
2. Allocates host CPU resources to node, measured in CPU units.
3. Allocates host memory resources to nodes, measured in bytes.

When Data Grid Operator creates Data Grid clusters, it uses **spec.container.cpu** and **spec.container.memory** to:

- Ensure that OpenShift has sufficient capacity to run the Data Grid node. By default Data Grid Operator requests **512Mi** of **memory** and **0.5** **cpu** from the OpenShift scheduler.
- Constrain node resource usage. Data Grid Operator sets the values of **cpu** and **memory** as resource limits.

Garbage collection logging

By default, Data Grid Operator does not log garbage collection (GC) messages. You can optionally add the following JVM options to direct GC messages to stdout:

```
extraJvmOpts: "-Xlog:gc*:stdout:time,level,tags"
```

5.2. STORAGE RESOURCES

```
spec:
  ...
  service:
    type: DataGrid
    container:
      storage: 2Gi 1
```

1. Configures the storage size for Data Grid service nodes.

By default, Data Grid Operator allocates **1Gi** for storage for both Cache service and Data Grid service nodes. You can configure storage size only for Data Grid service nodes.

Persistence

Data Grid service lets you configure Single File cache stores for data persistence:

```
<persistence>
   <file-store />
</persistence>
```

Reference

- Shutting Down Data Grid Clusters
- Persistent Volume Claims

5.2.1. Persistent Volume Claims

Data Grid Operator mounts persistent volumes at:
/opt/datagrid/server/data

NOTE

Persistent volume claims use the **ReadWriteOnce (RWO)** access mode.

Reference

- Shutting Down Data Grid Clusters
- Storage Resources

CHAPTER 6. CREATING NETWORK SERVICES

Network services provide access to Data Grid clusters for client connections.

6.1. GETTING THE SERVICE FOR INTERNAL CONNECTIONS

By default, Data Grid Operator creates a service that provides access to Data Grid clusters from clients running in OpenShift.

This internal service has the same name as your Data Grid cluster, for example:

```
metadata:
  name: example-rhdatagrid
```

Procedure

- Check that the internal service is available as follows:

```
$ oc get services

NAME                TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)
example-rhdatagrid  ClusterIP   192.0.2.0       <none>        11222/TCP
```

Reference

- Network Services

6.2. EXPOSING DATA GRID TO EXTERNAL CLIENTS

Expose Data Grid clusters to clients running outside OpenShift with external services.

Procedure

- Specify an external service type with **spec.expose.type** in your **Infinispan** CR and then apply the changes.

```
spec:
  ...
  expose: 1
    type: LoadBalancer 2
    nodePort: 30000 3
```

1. Exposes an external service.
2. Specifies either a **LoadBalancer** or **NodePort** service resource type.
3. Defines the port where the external service is exposed. If you do not define a port, and the service type is **NodePort**, the platform selects a port to use. If the service type is **LoadBalancer**, the exposed port is **11222** by default.

LoadBalancer

Use for OpenShift clusters where a load balancer service is available to handle external network traffic. You can then use the URL for the load balancer service for client connections.
To access Data Grid with unencrypted Hot Rod client connections you must use a load balancer service.

NodePort

Use for local OpenShift clusters.

Verification

- Check that the **-external** service is available.

```
$ oc get services | grep external

NAME                          TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)
example-rhdatagrid-external   LoadBalancer   192.0.2.24      <none>        11222/TCP
```

Reference

- Network Services

CHAPTER 7. CONFIGURING AUTHENTICATION

Application users must authenticate with Data Grid clusters. Data Grid Operator generates default credentials or you can add your own.

7.1. DEFAULT CREDENTIALS

Data Grid Operator generates base64-encoded default credentials stored in an authentication secret named **example-rhdatagrid-generated-secret**

Username	Description
developer	Default application user.
operator	Internal user that interacts with Data Grid clusters.

7.2. RETRIEVING CREDENTIALS

Get credentials from authentication secrets to access Data Grid clusters.

Procedure

- Retrieve credentials from authentication secrets, as in the following example:

```
$ oc get secret example-rhdatagrid-generated-secret
```

Base64-decode credentials.

```
$ oc get secret example-rhdatagrid-generated-secret \
-o jsonpath="{.data.identities\.yaml}" | base64 --decode

credentials:
- username: developer
  password: dIRs5cAAsHIeeRIL
- username: operator
  password: uMBo9CmEdEduYk24
```

7.3. ADDING CUSTOM CREDENTIALS

Add custom credentials to an authentication secret.

Procedure

1. Create an **identities.yaml** file that contains credentials for application users and the **operator** user for Data Grid Operator, for example:

```
credentials:
- username: testuser
  password: testpassword
```

```
- username: operator
  password: supersecretoperatorpassword
```

2. Create an authentication secret with **identities.yaml** as follows:

```
$ oc create secret generic --from-file=identities.yaml connect-secret
```

3. Specify the authentication secret with **spec.security.endpointSecretName** in your **Infinispan** CR and then apply the changes.

```
spec:
  ...
  security:
    endpointSecretName: connect-secret (1)
```

(1) specifies the authentication secret.

CHAPTER 8. SECURING DATA GRID CONNECTIONS

Encrypt connections between clients and Data Grid nodes with Red Hat OpenShift service certificates or custom TLS certificates.

8.1. USING RED HAT OPENSHIFT SERVICE CERTIFICATES

Data Grid Operator automatically generates TLS certificates signed by the Red Hat OpenShift service CA. You can use these certificates to encrypt remote client connections.

Procedure

- Set the following **spec.security.endpointEncryption** configuration in your **Infinispan** CR and then apply the changes.

```
spec:
  ...
  security:
    endpointEncryption:
      type: service
      certServiceName: service.beta.openshift.io 1
      certSecretName: example-rhdatagrid-cert-secret 2
```

1 Specifies the Red Hat OpenShift Service.

2 Specifies the name of the secret where Data Grid Operator stores service certificates and keys.

Data Grid Operator stores in a secret named **-cert-secret** that is prefixed with the Data Grid cluster name, for example:

```
metadata:
  name: example-rhdatagrid
```

The preceding cluster name results in a secret named **example-rhdatagrid-cert-secret**.

8.1.1. Red Hat OpenShift Service Certificates

If the Red Hat OpenShift service CA is available, Data Grid Operator automatically generates a certificate, **tls.crt**, and key, **tls.key**, in PEM format.

NOTE

Service certificates use the internal DNS name of the Data Grid cluster as the common name (CN), for example:

Subject: CN = example-infinispan.mynamespace.svc

For this reason, service certificates can be fully trusted only inside OpenShift. If you want to encrypt connections with clients running outside OpenShift, you should use custom TLS certificates.

Certificates are valid for one year and are automatically replaced before they expire.

8.1.2. Retrieving TLS Certificates

Get TLS certificates from encryption secrets to create client trust stores.

- Retrieve **tls.crt** from encryption secrets as follows:

```
$ oc get secret example-rhdatagrid-cert-secret \
-o jsonpath='{.data.tls\.crt}' | base64 -d > tls.crt
```

8.2. USING CUSTOM TLS CERTIFICATES

Use custom PKCS12 keystore or TLS certificate/key pairs to encrypt connections between clients and Data Grid clusters.

Prerequisites

Create either a keystore or certificate secret. See:

- Certificate Secrets
- Keystore Secrets

Procedure

1. Add the encryption secret to your OpenShift namespace, for example:

```
$ oc apply -f tls_secret.yaml
```

2. Specify the encryption secret with **spec.security.endpointEncryption** in your **Infinispan** CR and then apply the changes.

```
spec:
  ...
  security:
    endpointEncryption: 1
          type: secret 2
          certSecretName: tls-secret 3
```

1. encrypts traffic to and from Data Grid endpoints.
2. configures Data Grid to use secrets that contain encryption certificates.
3. names the encryption secret.

8.2.1. Certificate Secrets

```
apiVersion: v1
kind: Secret
metadata:
  name: tls-secret
type: Opaque
```

```
data:
  tls.key:  "LS0tLS1CRUdJTiBQUk ..." (1)
  tls.crt: "LS0tLS1CRUdJTiBDRVl ..." (2)
```

1. Adds a base64 encoded TLS key.
2. Adds a base64 encoded TLS certificate.

8.2.2. Keystore Secrets

```
apiVersion: v1
kind: Secret
metadata:
  name: tls-secret
type: Opaque
stringData:
  alias: server (1)
  password: password (2)
data:
  keystore.p12:  "MIIKDgIBAzCCCdQGCSqGSIb3DQEHA..." (3)
```

1. Specifies an alias for the keystore.
2. Specifies a password for the keystore.
3. Adds a base64 encoded keystore.

CHAPTER 9. MONITORING DATA GRID WITH PROMETHEUS

Data Grid exposes a metrics endpoint that provides statistics and events to Prometheus.

9.1. SETTING UP PROMETHEUS

Set up Prometheus so it can authenticate with and monitor Data Grid clusters.

Prerequisites

- Install the Prometheus Operator.
- Create a running Prometheus instance.

Procedure

1. Add an authentication secret to your Prometheus namespace.
   This secret allows Prometheus to authenticate with your Data Grid cluster. You can find Data Grid credentials in the authentication secret in your Data Grid Operator namespace.

```
apiVersion: v1
stringData:
  username: developer (1)
  password: dIRs5cAAsHIeeRIL (2)
kind: Secret
metadata:
  name: basic-auth
type: Opaque
```

(1) specifies an application user. **developer** is the default application user.

(2) specifies a corresponding password.

2. Create a service monitor instance that configures Prometheus to monitor your Data Grid cluster.

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  labels:
    k8s-app: prometheus
  name: datagrid-monitoring (1)
  namespace: infinispan-monitoring (2)
spec:
  endpoints:
    - basicAuth:
        username:
          key: username
          name: basic-auth (3)
        password:
          key: password
          name: basic-auth (4)
      interval: 30s
```

```
    scheme: https 5
    tlsConfig:
      insecureSkipVerify: true
      serverName: certificate-CN 6
namespaceSelector:
  matchNames:
    - infinispan 7
selector:
  matchLabels:
    app: infinispan-service
    clusterName: cluster-name 8
```

1. names the service monitor.
2. specifies your Prometheus namespace.
3. specifies the name of the authentication secret that has Data Grid credentials.
4. specifies the name of the authentication secret that has Data Grid credentials.
5. specifies that Data Grid endpoints use encryption. If you do not use TLS, remove **spec.endpoints.scheme**.
6. specifies the Common Name (CN) of the TLS certificate for Data Grid encryption. If you use an OpenShift service certificate, the CN matches the **metadata.name** resource for your Data Grid cluster. If you do not use TLS, remove **spec.endpoints.tlsConfig**.
7. specifies the Data Grid Operator namespace.
8. specifies the name of the Data Grid cluster.

CHAPTER 10. CONNECTING TO DATA GRID CLUSTERS

Connect to Data Grid via the REST or Hot Rod endpoints. You can then remotely create and modify cache definitions and store data across Data Grid clusters.

The examples in this section use **$SERVICE_HOSTNAME** to denote the service that provides access to your Data Grid cluster.

Clients running in OpenShift can specify the name of the internal service that Data Grid Operator creates.

Clients running outside OpenShift should specify hostnames according to the type of external service and provider. For example, if using a load balancer service on AWS, the service hostname could be:

.status.loadBalancer.ingress[0].hostname

On GCP or Azure, hostnames might be as follows:

.status.loadBalancer.ingress[0].ip

10.1. INVOKING THE DATA GRID REST API

You can invoke the Data Grid REST API with any appropriate HTTP client.

For convenience, the following examples show how to invoke the REST API with **curl** using unencrypted connections. It is beyond the scope of this document to describe how to configure HTTP clients to use encryption.

Procedure

1. Open a remote shell to a Data Grid node, for example:

```
$ oc rsh example-rhdatagrid
```

2. Cache service provides a default cache instance, but Data Grid service does not. Before you can store data with Data Grid service clusters, you must create a cache as in the following example:

```
$ curl -X POST \
  -u developer:$PASSWORD \
  $SERVICE_HOSTNAME:11222/rest/v2/caches/default
  ...
  < HTTP/1.1 200 OK
  ...
```

3. Put an entry in the cache.

```
$ curl -X POST \
  -u developer:$PASSWORD \
  -H 'Content-type: text/plain' -d 'world' \
  $SERVICE_HOSTNAME:11222/rest/v2/caches/default/hello
  ...
  < HTTP/1.1 204 No Content
```

4. Verify the entry.

```
$ curl -X GET \
  -u developer:$PASSWORD \
  $SERVICE_HOSTNAME:11222/rest/v2/caches/default/hello/
  ...
  < HTTP/1.1 200 OK
  ...
  world
```

10.2. CONFIGURING HOT ROD CLIENTS

Configure Hot Rod Java clients to connect to Data Grid clusters.

Hot Rod client ConfigurationBuilder

```
import org.infinispan.client.hotrod.configuration.ConfigurationBuilder;

ConfigurationBuilder builder = new ConfigurationBuilder();
builder.addServer()
      //Connection
      .host("$SERVICE_HOSTNAME").port(11222)
      //Client intelligence
      //External clients can use `BASIC` intelligence only.
      .clientIntelligence(ClientIntelligence.BASIC)
      .security()
        //Authentication
        .authentication().enable()
          //Application user credentials.
          //The default username is developer.
          .username("developer")
          .password("$PASSWORD")
          .serverName("$CLUSTER_NAME")
          .saslQop(SaslQop.AUTH)
          .saslMechanism("DIGEST-MD5")
        //Encryption
        .ssl()
          .sniHostName("$SERVICE_HOSTNAME")
          //Path to the TLS certificate.
          //Clients automatically generate trust stores from certificates.
          .trustStorePath(tls.crt);
```

Hot Rod client properties

```
# Connection
infinispan.client.hotrod.server_list=$SERVICE_HOSTNAME:11222

# Client intelligence
# External clients can use `BASIC` intelligence only.
infinispan.client.hotrod.client_intelligence=BASIC

# Authentication
infinispan.client.hotrod.use_auth=true
# Application user credentials.
# The default username is developer.
infinispan.client.hotrod.auth_username=developer
```

```
infinispan.client.hotrod.auth_password=$PASSWORD
infinispan.client.hotrod.auth_server_name=$CLUSTER_NAME
infinispan.client.hotrod.sasl_properties.javax.security.sasl.qop=auth
infinispan.client.hotrod.sasl_mechanism=DIGEST-MD5

# Encryption
infinispan.client.hotrod.sni_host_name=$SERVICE_HOSTNAME
# Path to the TLS certificate.
# Clients automatically generate trust stores from certificates.
infinispan.client.hotrod.trust_store_path=tls.crt
```

CHAPTER 11. MONITORING DATA GRID LOGS

Set logging categories to different message levels to monitor, debug, and troubleshoot Data Grid clusters.

11.1. CONFIGURING DATA GRID LOGGING

Procedure

1. Specify logging configuration with **spec.logging** in your **Infinispan** CR and then apply the changes.

```
spec:
  ...
  logging: 1
    categories: 2
      org.infinispan: debug 3
      org.jgroups: debug
```

1 configures Data Grid logging.

2 adds logging categories.

3 names logging categories and levels.

NOTE

The root logging category is **org.infinispan** and is **INFO** by default.

2. Retrieve logs from Data Grid nodes as required.

```
$ oc logs -f $POD_NAME
```

11.2. LOG LEVELS

Log levels indicate the nature and severity of messages.

Log level	Description
trace	Provides detailed information about running state of applications. This is the most verbose log level.
debug	Indicates the progress of individual requests or activities.
info	Indicates overall progress of applications, including lifecycle events.

Log level	Description
warn	Indicates circumstances that can lead to error or degrade performance.
error	Indicates error conditions that might prevent operations or activities from being successful but do not prevent applications from running.

CHAPTER 12. CONFIGURING CROSS-SITE REPLICATION

Set up cross-site replication to back up data between Data Grid clusters running in different locations.

For example, you use Data Grid Operator to manage a Data Grid cluster at a data center in London, **LON**. At another data center in New York City, **NYC**, you also use Data Grid Operator to manage a Data Grid cluster. In this case, you can add **LON** and **NYC** as backup locations for each other.

IMPORTANT

Cross-site replication functionality is currently Technology Preview. Contact Red Hat support for more information.

Prerequisites

- Ensure that a load balancer service is available for OpenShift. This service allows external access to OpenShift Container Platform clusters. See Configuring ingress cluster traffic using a load balancer.

12.1. DATA GRID CLUSTER AND PROJECT NAMING

Data Grid Operator expects Data Grid clusters in each site to have the same cluster names and be running in matching namespaces.

For example, in the **LON** site you create a Data Grid cluster with **metadata.name: mydatagrid** in a OpenShift project named "my-xsite". In this case you must create Data Grid clusters in other backup locations, such as **NYC**, with identical names in matching namespaces.

In effect, you must create Data Grid cluster names and OpenShift namespaces at each backup location that mirror one another.

12.2. CREATING SERVICE ACCOUNT TOKENS

Traffic between independent OpenShift installations occurs through a Kubernetes API. OpenShift Container Platform clusters use tokens to authenticate with and access the API.

To enable cross-site replication between Data Grid clusters you must add tokens to the namespace on each site. For example, **LON** needs a secret with the token for **NYC**. **NYC** also needs a secret with the token for **LON**.

Procedure

1. Create service accounts on each OpenShift instance.
   For example, create a service account on **LON** as follows:

```
$ oc create sa lon
serviceaccount/lon created
```

2. Add the view role to service accounts.
   For example, if your Data Grid cluster runs in the "my-xsite" namespace, add the view role to the service account on **LON** as follows:

```
$ oc policy add-role-to-user view system:serviceaccount:my-xsite:lon
```

3. Retrieve tokens from each service account.
   The following example shows the service account token for **LON**:

```
$ oc sa get-token lon

eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9...
```

4. Create secrets that contain service account tokens for the backup locations.
   a. Log in to OpenShift Container Platform at **NYC**.
   b. Add the service account token to a **lon-token** secret.

```
oc create secret generic lon-token --from-
literal=token=eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9...
```

   c. Repeat the preceding steps to create a **nyc-token** secret on **LON**.

After you add service account tokens to each backup location, the OpenShift instances can authenticate with each other so that Data Grid clusters can form cross-site views.

Reference

Using service accounts in applications

12.3. ADDING BACKUP LOCATIONS TO DATA GRID CLUSTERS

Configure Data Grid clusters as backup locations so that they can communicate over a dedicated JGroups transport channel for replicating data.

Procedure

1. Configure Data Grid clusters at each site with the **Infinispan** CR as necessary.
   For example, create **lon.yaml** to configure **LON** and **nyc.yaml** to configure **NYC**. Both configurations must include the following:
   - **.spec.service.sites.local** names the local site for Data Grid clusters.
   - **.spec.service.sites.locations** provides the location of all site masters. Data Grid nodes use this information to connect with each other and form cross-site views.
2. Instantiate Data Grid clusters at each site, for example:
   a. Apply the **Infinispan** CR for **LON**.

```
$ oc apply -f lon.yaml
```

   b. Log in to OpenShift Container Platform at **NYC**.
   c. Apply the **Infinispan** CR for **NYC**.

```
$ oc apply -f nyc.yaml
```

3. Verify that Data Grid clusters form a cross-site view.
   For example, do the following on **LON**:

```
$ oc logs example-rhdatagrid-0 | grep x-site

INFO  [org.infinispan.XSITE] (jgroups-5,example-rhdatagrid-0-<id>) ISPN000439: Received
new x-site view: [NYC]
INFO  [org.infinispan.XSITE] (jgroups-7,example-rhdatagrid-0-<id>) ISPN000439: Received
new x-site view: [NYC, LON]
```

Reference

Cross-Site Replication Resources

12.3.1. Cross-Site Replication Resources

```
spec:
  ...
  service:
    type: DataGrid (1)
    sites:
      local:
        name: LON (2)
        expose:
          type: LoadBalancer (3)
      locations: (4)
      - name: LON (5)
        url: openshift://api.site-a.devcluster.openshift.com:6443 (6)
        secretName: lon-token (7)
      - name: NYC
        url: openshift://api.site-b.devcluster.openshift.com:6443
        secretName: nyc-token
```

1. Specifies Data Grid service. Data Grid supports cross-site replication with Data Grid service clusters only.
2. Names the local site for a Data Grid cluster.
3. Specifies **LoadBalancer** as the service that handles communication between backup locations.
4. Provides connection information for all backup locations.
5. Specifies a backup location that matches **.spec.service.sites.local.name**.
6. Specifies the URL of the Kubernetes API for the backup location.
7. Specifies the secret that contains the service account token for the backup site.

CHAPTER 13. REFERENCE

Find useful information for Data Grid clusters that you create with Data Grid Operator.

13.1. NETWORK SERVICES

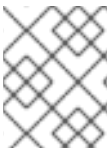
Internal service

- Allow Data Grid nodes to discover each other and form clusters.
- Provide access to Data Grid endpoints from clients in the same OpenShift namespace.

Service	Port	Protocol	Description
$ClusterName	**11222**	TCP	Internal access to Data Grid endpoints
$ClusterName-ping	**8888**	TCP	Cluster discovery

External service

Provides access to Data Grid endpoints from clients outside OpenShift or in different namespaces.



NOTE

You must create the external service with Data Grid Operator. It is not available by default.

Service	Port	Protocol	Description
$ClusterName-external	**11222**	TCP	External access to Data Grid endpoints.

Cross-site service

Allows Data Grid to back up data between clusters in different locations.

Service	Port	Protocol	Description
$NAME-sites	**7900**	TCP	JGroups RELAY2 channel for cross-site communication.

Reference

Creating Network Services

13.2. DATA GRID OPERATOR UPGRADES

Data Grid Operator upgrades Data Grid when new versions become available.

To upgrade Data Grid clusters, Data Grid Operator checks the version of the image for Data Grid nodes. If Data Grid Operator determines that a new version of the image is available, it gracefully shuts down all nodes, applies the new image, and then restarts the nodes.

On Red Hat OpenShift, the Operator Lifecycle Manager (OLM) enables upgrades for Data Grid Operator. When you install Data Grid Operator, you select either **Automatic** or **Manual** updates with the **Approval Strategy**. This determines how Data Grid Operator upgrades clusters. See the OpenShift documentation for more information.

Reference

- Understanding the Operator Lifecycle Manager
- Approval Strategy

13.3. TECHNOLOGY PREVIEW

Technology Preview features or capabilities are not supported with Red Hat production service-level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them for production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information, see Red Hat Technology Preview Features Support Scope .