Red Hat Data Grid 8.4

Data Grid Operator ガイド

OpenShift での Data Grid クラスターの作成

Last Updated: 2024-06-05

Red Hat Data Grid 8.4 Data Grid Operator ガイド

OpenShift での Data Grid クラスターの作成

法律上の通知

概要

Data Grid Operator は、運用インテリジェンスを提供し、OpenShift に Data Grid をデプロイするための管理の複雑さを軽減します。

目次

RED HAT DATA GRID

Data Grid は、高性能の分散型インメモリーデータストアです。

スキーマレスデータ構造

さまざまなオブジェクトをキーと値のペアとして格納する柔軟性があります。

グリッドベースのデータストレージ

クラスター間でデータを分散および複製するように設計されています。

エラスティックスケーリング

サービスを中断することなく、ノードの数を動的に調整して要件を満たします。

データの相互運用性

さまざまなエンドポイントからグリッド内のデータを保存、取得、およびクエリーします。

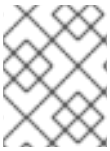
DATA GRID のドキュメント

Data Grid のドキュメントは、Red Hat カスタマーポータルで入手できます。

- Data Grid 8.4 ドキュメント
- Data Grid 8.4 コンポーネントの詳細
- Data Grid 8.4 でサポートされる設定
- Data Grid 8 機能のサポート
- Data Grid で非推奨の機能

DATA GRID のダウンロード

Red Hat カスタマーポータルで Data Grid Software Downloads にアクセスします。



注記

Data Grid ソフトウェアにアクセスしてダウンロードするには、Red Hat アカウントが必要です。

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、用語の置き換えは、今後の複数のリリースにわたって段階的に実施されます。詳細は、Red Hat CTO である Chris Wright のメッセージ をご覧ください。

第1章 DATA GRID OPERATOR

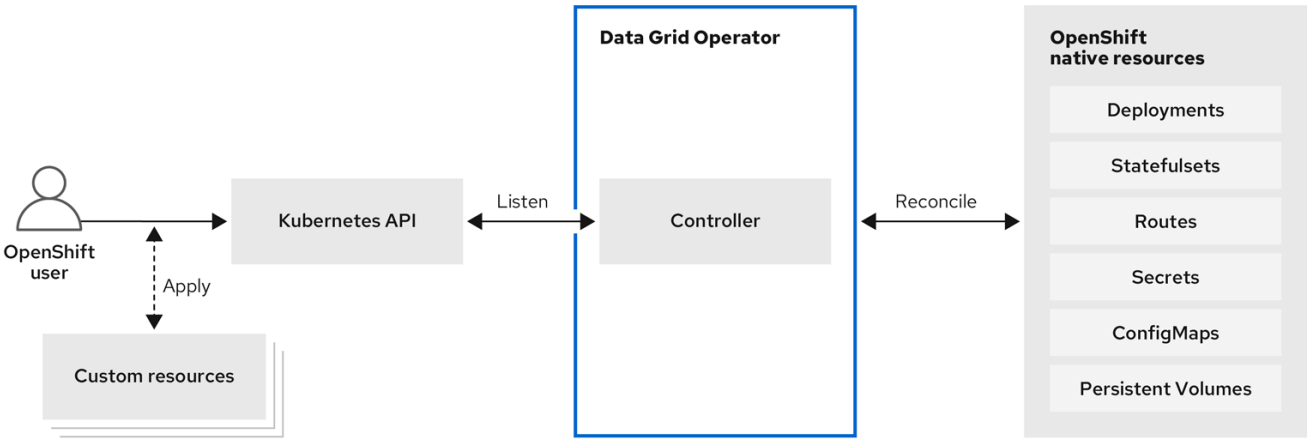
Data Grid Operator は、Kubernetes および Red Hat OpenShift に Data Grid をデプロイする際に、運用インテリジェンスを提供し、管理の複雑さを軽減します。

1.1. DATA GRID OPERATOR のデプロイメント

Data Grid Operator のインストール時に、Red Hat OpenShift で Data Grid クラスターをデプロイし、管理するためのカスタムリソース定義 (CRD) で Kubernetes API を拡張します。

Data Grid Operator を操作するには、OpenShift ユーザーは OpenShift Web コンソールまたは **oc** クライアントを使用してカスタムリソース (CR) を適用します。Data Grid Operator は **Infinispan** CR をリッスンし、Data Grid のデプロイメントに必要な StatefulSets や Secrets などのネイティブリソースを自動的にプロビジョニングします。Data Grid Operator は、クラスター用の Pod 数やクロスサイトレプリケーション用のバックアップの場所など、**Infinispan** CR の仕様に応じて Data Grid サービスも設定します。

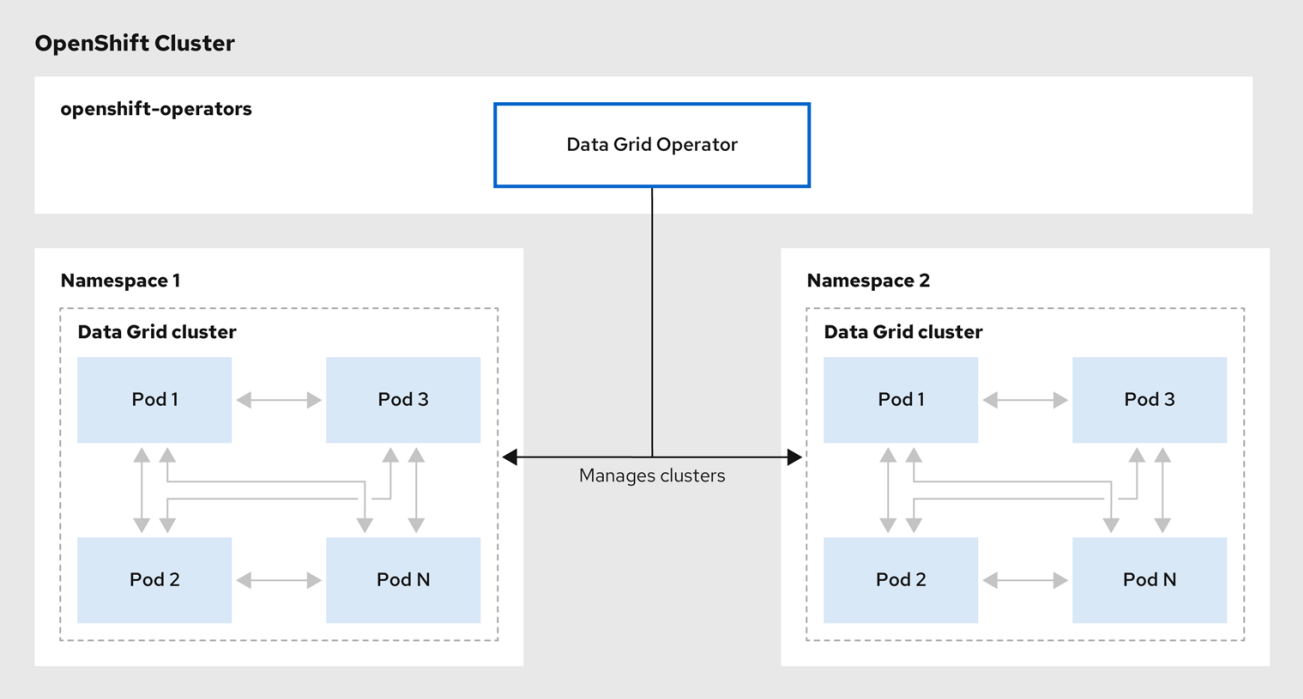
図1.1 カスタムリソース



1.2. クラスター管理

Data Grid Operator を 1 つインストールして、別々の namespace で Data Grid バージョンが異なる複数のクラスターを管理できます。ユーザーが CR を適用してデプロイメントを変更するたびに、Data Grid Operator は変更を全 Data Grid クラスターにグローバルに適用します。

図1.2 Operator 管理のクラスター



1.3. リソースの調整

Data Grid Operator は、**Cache** CR などのカスタムリソースと Data Grid クラスター上のリソースを照合します。

双方向の調整は、Data Grid Console、コマンドラインインターフェイス (CLI)、またはその他のクライアントアプリケーションを介して Data Grid リソースに加えた変更と CR を同期し、その逆も同様です。たとえば、Data Grid Console を介してキャッシュを作成する場合、Data Grid Operator は宣言型の Kubernetes 表現を追加します。

調整を実行するために、Data Grid Operator は、**Infinispan** リソースの変更を検出する各 Data Grid クラスターの **listener** Pod を作成します。

調整に関する注記

- Data Grid Console、CLI、またはその他のクライアントアプリケーションを介してキャッシュを作成すると、Data Grid Operator は、Kubernetes 命名ポリシーに準拠した一意の名前で対応する **Cache** CR を作成します。
- Data Grid Operator が **listener** Pod で作成する Data Grid リソースの宣言型 Kubernetes 表現は、**Infinispan** CR にリンクされています。
  Infinispan CR を削除すると、関連するリソース宣言がすべて削除されます。

第2章 クライアントプラグインとしてのネイティブ DATA GRID CLI のインストール

Data Grid には、ネイティブ実行可能ファイルにコンパイルされたコマンドラインインターフェイス (CLI) が含まれており、**oc** クライアントのプラグインとしてインストールできます。インストール後には、**oc** クライアントを使用して以下を行うことができます。

- Data Grid Operator サブスクリプションを作成し、Data Grid Operator のインストールを削除する。
- Data Grid クラスターを設定し、サービスを設定する。
- リモートシェルで Data Grid リソースを操作する。

2.1. ネイティブ DATA GRID CLI プラグインのインストール

oc クライアントのプラグインとして、ネイティブ Data Grid コマンドラインインターフェイス (CLI) をインストールします。

前提条件

- **oc** クライアントがある。
- Data Grid ソフトウェアダウンロード から、ネイティブ Data Grid CLI ディストリビューションをダウンロードします。

手順

1. ネイティブ Data Grid CLI ディストリビューションの **.zip** アーカイブをデプロイメントします。
2. ネイティブ実行ファイルをコピーするか、kubectl-infinispan という名前のファイルにハードリンクを作成します。以下に例を示します。

```
cp redhat-datagrid-cli kubectl-infinispan
```

3. **kubectl-infinispan** を **PATH** に追加します。
4. CLI がインストールされていることを確認します。

```
oc plugin list

The following compatible plugins are available:
/path/to/kubectl-infinispan
```

5. **infinispan --help** コマンドを使用して、利用可能なコマンドを表示します。

```
oc infinispan --help
```

関連情報

- プラグインによる OpenShift CLI の拡張

2.2. KUBECTL-INFINISPAN コマンドリファレンス

このトピックでは、クライアントの **kubectl-infinispan** プラグインを詳しく説明します。

ヒント

--help 引数を使用して、利用可能なオプションと各コマンドの説明を表示します。

たとえば、**oc infinispan create cluster --help** は、Data Grid クラスター作成に関連するすべてのコマンドオプションを出力します。

コマンド	説明
oc infinispan install	Data Grid Operator サブスクリプションを作成し、デフォルトでグローバル名前空間にインストールします。
oc infinispan create cluster	Data Grid クラスターを作成します。
oc infinispan get clusters	実行中の Data Grid クラスターを表示します。
oc infinispan shell	Data Grid クラスターでインタラクティブなリモートシェルセッションを開始します。
oc infinispan delete cluster	Data Grid クラスターを削除します。
oc infinispan uninstall	Data Grid Operator のインストールおよびすべての管理リソースを削除します。

第3章 DATA GRID OPERATOR のインストール

Data Grid Operator を OpenShift namespace にインストールして、Data Grid クラスターを作成して管理します。

3.1. RED HAT OPENSHIFT への DATA GRID OPERATOR のインストール

OpenShift 上の Data Grid Operator へのサブスクリプションを作成し、さまざまな Data Grid バージョンをインストールし、、自動更新を受信できるようにします。

自動更新はまず Data Grid Operator に適用され、その後各 Data Grid ノードに適用されます。Data Grid Operator は、クラスターを一度に 1 つのノードで更新し、各ノードを正常にシャットダウンしてから、更新されたバージョンでオンラインに戻してから、次のノードに進みます。

前提条件

- OpenShift で実行している **OperatorHub** へのアクセスがある。OpenShift Container Platform などの一部の OpenShift 環境では、管理者の認証情報が必要になる場合があります。
- 特定の namespace にインストールする予定がある場合は、Data Grid Operator の OpenShift プロジェクトがある。

手順

1. OpenShift Web コンソールにログインします。
2. **OperatorHub** に移動します。
3. Data Grid Operator を見つけ、これを選択します。
4. **Install** を選択し、**Create Operator Subscription** に進みます。
5. サブスクリプションのオプションを指定します。

   インストールモード

   Data Grid Operator は、**特定の** namespace または **すべての** namespace にインストールできます。

   更新チャネル

   Data Grid Operator 8.4.x の更新を取得します。

   承認ストラテジー

   8.4.x チャネルから更新を自動的にインストールするか、またはインストール前に承認が必要です。
6. **Subscribe** を選択して Data Grid Operator をインストールします。
7. **Installed Operators** に移動し、Data Grid Operator のインストールを確認します。

3.2. ネイティブ CLI プラグインを使用した DATA GRID OPERATOR のインストール

ネイティブ Data Grid CLI プラグイン **kubectl-infinispan** を使用して Data Grid Operator をインストールします。

前提条件

- **PATH** に **kubectl-infinispan** があること。

手順

1. 以下のように、**oc infinispan install** コマンドを実行して Data Grid Operator サブスクリプションを作成します。

```
oc infinispan install --channel=8.4.x
                      --source=redhat-operators
                      --source-namespace=openshift-marketplace
```

2. インストールを確認します。

```
oc get pods -n openshift-operators | grep infinispan-operator
NAME                              READY   STATUS
infinispan-operator-<id>            1/1     Running
```

ヒント

コマンドオプションおよび説明には、**oc infinispan install --help** を使用します。

3.3. OPENSHIFT クライアントを使用した DATA GRID OPERATOR のインストール

OperatorHub またはネイティブ Data Grid CLI を使用してインストールする代わりに、**oc** クライアントを使用して、Data Grid Operator サブスクリプションを作成できます。

前提条件

- **oc** クライアントがある。

手順

1. プロジェクトを設定します。

   a. Data Grid Operator のプロジェクトを作成します。

   b. Data Grid Operator が特定の Data Grid クラスターのみを制御する必要がある場合は、そのクラスターのプロジェクトを作成します。

```
oc new-project ${INSTALL_NAMESPACE} 1
oc new-project ${WATCH_NAMESPACE} 2
```

   1 Data Grid Operator をインストールするプロジェクトを作成します。

   2 Data Grid Operator がすべてのプロジェクトを監視する必要がない場合は、オプションとして特定の Data Grid クラスターのプロジェクトを作成します。

2. **OperatorGroup** リソースを作成します。

すべての Data Grid クラスターの制御

```
oc apply -f - << EOF
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
 name: datagrid
 namespace: ${INSTALL_NAMESPACE}
EOF
```

特定の Data Grid クラスターの制御

```
oc apply -f - << EOF
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
 name: datagrid
 namespace: ${INSTALL_NAMESPACE}
spec:
 targetNamespaces:
 - ${WATCH_NAMESPACE}
EOF
```

3. Data Grid Operator のサブスクリプションを作成します。

```
oc apply -f - << EOF
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
 name: datagrid-operator
 namespace: ${INSTALL_NAMESPACE}
spec:
 channel: 8.4.x
 installPlanApproval: Automatic
 name: datagrid
 source: redhat-operators
 sourceNamespace: openshift-marketplace
EOF
```

注記

8.4.x チャネルから更新を手動で承認する場合は、**spec.installPlanApproval** フィールドの値を **Manual** に変更します。

4. インストールを確認します。

```
oc get pods -n ${INSTALL_NAMESPACE}
NAME                              READY   STATUS
infinispan-operator-<id>             1/1     Running
```

第4章 DATA GRID クラスターの作成

Infinispan CR または **oc** クライアントのネイティブ Data Grid CLI プラグインを使用して、OpenShift で実行している Data Grid クラスターを作成します。

4.1. INFINISPAN カスタムリソース (CR)

Data Grid Operator は、OpenShift で Data Grid クラスターを複雑なユニットとして処理できるようにするタイプ **Infinispan** の新しいカスタムリソース (CR) を追加します。

Data Grid Operator は、Data Grid クラスターのインスタンス化および設定に使用する **Infinispan** カスタムリソース (CR) をリッスンし、StatefulSet や Services などの OpenShift リソースを管理します。

Infinispan CR

```
apiVersion: infinispan.org/v1
kind: Infinispan
metadata:
  name: infinispan
spec:
  replicas: 2
  version: <Data Grid_version>
  service:
    type: DataGrid
```

フィールド	説明
apiVersion	**Infinispan** API のバージョンを宣言します。
kind	**Infinispan** CR を宣言します。
metadata.name	Data Grid クラスターの名前を指定します。
spec.replicas	Data Grid クラスターの Pod 数を指定します。
spec.service.type	作成する Data Grid サービスのタイプを指定します。
spec.version	クラスターの Data Grid サーバーのバージョンを指定します。

4.2. DATA GRID クラスターの作成

ネイティブ CLI プラグイン **kubectl-infinispan** を使用して Data Grid クラスターを作成します。

前提条件

- Data Grid Operator をインストールしている。
- **PATH** に **kubectl-infinispan** があること。

手順

1. **infinispan create cluster** コマンドを実行します。
たとえば、以下のように 2 つの Pod を含めて Data Grid クラスターを作成します。

```
oc infinispan create cluster --replicas=3 -Pservice.type=DataGrid infinispan
```

ヒント

--version 引数を追加して、クラスターの Data Grid バージョンを制御します。たとえば、**--version=8.4.6-1** とします。バージョンを指定しない場合、Data Grid Operator は、サポートされている最新の Data Grid バージョンでクラスターを作成します。

2. Data Grid Operator が Data Grid Pod を作成するのを監視します。

```
oc get pods -w
```

次のステップ

Data Grid クラスターを作成した後に、**oc** を使用して **Infinispan** CR への変更を適用し、Data Grid サービスを設定します。

kubectl-infinispan で Data Grid クラスターを削除し、必要に応じて再作成することもできます。

```
oc infinispan delete cluster infinispan
```

関連情報

- kubectl-infinispan コマンドリファレンス

4.3. DATA GRID クラスタービューの検証

Data Grid Pod がクラスターを正常に形成していることを確認します。

前提条件

- Data Grid クラスターを 1 つ以上作成します。

手順

- Data Grid Operator の **Infinispan** CR を取得します。

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

応答は、以下の例のように、Data Grid Pod がクラスター化されたビューを受け取ったことを示します。

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

ヒント

自動スクリプトで以下を実行します。

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

ログからクラスタービューを取得する
次のように、Data Grid ログからクラスタービューを取得することもできます。

```
oc logs infinispan-0 | grep ISPN000094
```

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

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

4.4. DATA GRID クラスターの変更

Data Grid Operator にカスタム **Infinispan** CR を指定して、Data Grid クラスターを設定します。

前提条件

- Data Grid Operator をインストールしている。
- Data Grid クラスターを 1 つ以上作成します。
- **oc** クライアントがある。

手順

1. **Infinispan** CR を定義する YAML ファイルを作成します。
   たとえば、Data Grid Pod の数を 2 に変更する **my_infinispan.yaml** ファイルを作成します。

```
cat > cr_minimal.yaml<<EOF
apiVersion: infinispan.org/v1
kind: Infinispan
metadata:
  name: infinispan
spec:
  replicas: 2
  version: <Data Grid_version>
  service:
    type: DataGrid
EOF
```

2. **Infinispan** CR を適用します。

```
oc apply -f my_infinispan.yaml
```

3. Data Grid Operator で Data Grid Pod がスケーリングされることを確認します。

```
oc get pods -w
```

4.5. DATA GRID クラスターの停止および起動

クラスターの状態を正しく維持するために、適切な順序で Data Grid Pod を停止して起動します。

Data Grid サービス Pod のクラスターは、シャットダウン前に存在していた Pod 数と同じ数で再起動する必要があります。これにより、Data Grid はクラスター全体でのデータの分散を復元できます。Data Grid Operator がクラスターを完全に再起動した後、Pod を安全に追加および削除できます。

手順

1. **spec.replicas** フィールドを **0** に変更し、Data Grid クラスターを停止します。

```
spec:
  replicas: 0
```

2. クラスターを再起動する前に、Pod の数が正しいことを確認してください。

```
oc get infinispan infinispan -o=jsonpath='{.status.replicasWantedAtRestart}'
```

3. **spec.replicas** フィールドを同じ Pod 数に変更して、Data Grid クラスターを再起動します。

```
spec:
  replicas: 6
```

第5章 DATA GRID クラスターの設定

Data Grid Operator が管理するクラスターにカスタム Data Grid 設定を適用します。

5.1. DATA GRID クラスターへのカスタム設定の適用

Data Grid の設定を **ConfigMap** に追加して、Data Grid Operator で利用できるようにします。これにより、Data Grid Operator は、カスタム設定を Data Grid クラスターに適用できます。

重要

Data Grid Operator は、カスタム設定に加えて、デフォルト設定を適用し、Data Grid クラスターの管理を継続できるようにします。

cache-container 要素またはフィールド外でカスタム設定を適用する場合は注意してください。カスタム設定は、エンドポイント、セキュリティーレルム、クラスタートランスポートなどの基礎となる Data Grid Server メカニズムに適用できます。この設定を変更すると、エラーが発生して Data Grid のデプロイメントでダウンタイムが生じる場合があります。

ヒント

Data Grid Helm チャートを使用して、OpenShift に完全に設定可能な Data Grid Server インスタンスのクラスターをデプロイします。

前提条件

- XML、YAML、または JSON 形式の有効な Data Grid 設定がある。

手順

1. **ConfigMap** の **data** フィールドの **infinispan-config.[xml|yaml|json]** キーに Data Grid 設定を追加します。

XML

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-config
  namespace: rhdg-namespace
data:
  infinispan-config.xml: >
    <infinispan>
      <!-- Custom configuration goes here. -->
    </infinispan>
```

YAML

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-config
```

```
    namespace: rhdg-namespace
data:
  infinispan-config.yaml: >
    infinispan:
      # Custom configuration goes here.
```

JSON

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-config
  namespace: rhdg-namespace
data:
  infinispan-config.json: >
    {
      "infinispan": {
      }
    }
```

2. YAML ファイルから **ConfigMap** を作成します。

```
oc apply -f cluster-config.yaml
```

3. **Infinispan** CR の **spec.configMapName** フィールドで **ConfigMap** の名前を指定し、変更を適用します。

```
spec:
  configMapName: "cluster-config"
```

次のステップ

クラスターがすでに Data Grid Operator を実行している場合は、これを再起動して設定を適用します。**ConfigMap** で Data Grid 設定を変更するたびに、Data Grid Operator は更新を検出し、クラスターを再起動して変更を適用します。

関連情報

- Data Grid Helm チャート

5.2. カスタム DATA GRID 設定

XML、YAML、または JSON 形式の **ConfigMap** に Data Grid の設定を追加できます。

5.2.1. キャッシュテンプレート

XML

```
<infinispan>
  <cache-container>
    <distributed-cache-configuration name="base-template">
      <expiration lifespan="5000"/>
    </distributed-cache-configuration>
```

```
    <distributed-cache-configuration name="extended-template"
                                     configuration="base-template">
      <encoding media-type="application/x-protostream"/>
      <expiration lifespan="10000"
                  max-idle="1000"/>
    </distributed-cache-configuration>
  </cache-container>
</infinispan>
```

YAML

```
infinispan:
  cacheContainer:
    caches:
      base-template:
        distributedCacheConfiguration:
          expiration:
            lifespan: "5000"
      extended-template:
        distributedCacheConfiguration:
          configuration: "base-template"
          encoding:
            mediaType: "application/x-protostream"
          expiration:
            lifespan: "10000"
            maxIdle: "1000"
```

JSON

```
{
  "infinispan" : {
    "cache-container" : {
      "caches" : {
        "base-template" : {
          "distributed-cache-configuration" : {
            "expiration" : {
              "lifespan" : "5000"
            }
          }
        },
        "extended-template" : {
          "distributed-cache-configuration" : {
            "configuration" : "base-template",
            "encoding": {
              "media-type": "application/x-protostream"
              },
            "expiration" : {
              "lifespan" : "10000",
              "max-idle" : "1000"
            }
          }
        }
      }
```

```
    }
  }
}
```

5.2.2. ロギング設定

ConfigMap の一部として Apache Log4j 設定を XML 形式で含めることもできます。

注記

Infinispan CR の **spec.logging.categories** フィールドを使用して、Data Grid クラスターのログレベルを調整します。高度なファイルベースのログ機能が必要な場合にのみ、Apache Log4j 設定を追加してください。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: logging-config
  namespace: rhdg-namespace
data:
  infinispan-config.xml: >
    <infinispan>
      <!-- Add custom Data Grid configuration if required. -->
      <!-- You can provide either Data Grid configuration, logging configuration, or both. -->
    </infinispan>

  log4j.xml: >
    <?xml version="1.0" encoding="UTF-8"?>
    <Configuration name="ServerConfig" monitorInterval="60" shutdownHook="disable">
        <Appenders>
            <!-- Colored output on the console -->
            <Console name="STDOUT">
                <PatternLayout pattern="%d{HH:mm:ss,SSS} %-5p (%t) [%c] %m%throwable%n"/>
            </Console>
        </Appenders>

        <Loggers>
            <Root level="INFO">
                <AppenderRef ref="STDOUT" level="TRACE"/>
            </Root>
            <Logger name="org.infinispan" level="TRACE"/>
        </Loggers>
    </Configuration>
```

関連情報

- カスタム Data Grid 設定

5.3. DATA GRID のカスタム設定の保護

Data Grid Server のカスタム設定をセキュアに定義および保存します。パスワードなどの機密性の高いテキスト文字列を保護するには、Data Grid Server 設定に直接ではなく、認証情報ストアにエントリーを追加します。

前提条件

- XML、YAML、または JSON 形式の有効な Data Grid 設定がある。

手順

1. **CredentialStore Secret** ファイルを作成します。
2. **data** フィールドを使用して、認証情報とそのエイリアスを指定します。

   user-secret.yaml

   ```
   apiVersion: v1
   kind: Secret
   metadata:
     name: user-secret
   type: Opaque
   data:
     postgres_cred: sensitive-value
     mysql_cred: sensitive-value2
   ```

3. Secret ファイルを適用します。

   ```
   oc apply -f user-secret.yaml
   ```

4. **Infinispan** CR を開いて編集します。
5. **spec.security.credentialStoreSecretName** フィールドに、認証情報ストアシークレットの名前を指定します。

   Infinispan CR

   ```
   spec:
     security:
       credentialStoreSecretName: user-secret
   ```

6. 変更を適用します。
7. Data Grid Server 設定を開いて編集します。
8. 設定に **credential-reference** を追加します。
   a. **credentials** を **store** の名前として指定します。
   b. **alias** 属性を認証情報シークレットに定義されたキーの 1 つとして指定します。

      Data Grid.xml

      ```
      <credential-store>
          <credential-reference store="credentials"  alias="postgres_cred"/>
      </credential-store>
      ```

関連情報

- 認証情報ストアの参照

第6章 DATA GRID クラスターのアップグレード

Data Grid Operator を使用すると、ダウンタイムやデータ損失なしで、Data Grid クラスターをあるバージョンから別のバージョンにアップグレードできます。

重要

Hot Rod ローリングアップグレードは、テクノロジープレビュー機能として利用できます。

6.1. テクノロジープレビュー機能

テクノロジープレビュー機能は、Red Hat の実稼働環境でのサービスレベルアグリーメント (SLA) ではサポートされておらず、機能的に完全でない可能性があります。

Red Hat は、テクノロジープレビュー機能の実稼働環境での使用を推奨していません。これらの機能により、近日発表予定の製品機能をリリースに先駆けてご提供でき、お客様は開発プロセス時に機能をテストして、フィードバックをお寄せいただくことができます。

詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

6.2. DATA GRID クラスターのアップグレード

spec.upgrades.type フィールドは、新しいバージョンが利用可能になったときに Data Grid Operator が Data Grid クラスターをアップグレードする方法を制御します。クラスターのアップグレードには、次の 2 つのタイプがあります。

Shutdown

Data Grid クラスターをサービスのダウンタイムでアップグレードします。これはデフォルトのアップグレードタイプです。

HotRodRolling

サービスのダウンタイムなしで Data Grid クラスターをアップグレードします。

シャットダウンのアップグレード

シャットダウンのアップグレードを実行するために、Data Grid Operator は次のことを行います。

1. 既存のクラスターを正常にシャットダウンします。
2. 既存のクラスターを削除します。
3. ターゲットバージョンで新しいクラスターを作成します。

Hot Rod ローリングのアップグレード

Hot Rod ローリングのアップグレードを実行するために、Data Grid Operator は次のことを行います。

1. 既存のクラスターと並行して実行されるターゲットバージョンを使用して、新しい Data Grid クラスターを作成します。
2. 既存のクラスターから新しいクラスターにデータを転送するためのリモートキャッシュストアを作成します。
3. すべてのクライアントを新しいクラスターにリダイレクトします。

4. すべてのデータおよびクライアント接続が新しいクラスターに転送されるときに、既存のクラスターを削除します。

重要

永続的なキャッシュストアでパッシベーションを有効にするキャッシュを使用して、Hot Rod ローリングアップグレードを実行しないでください。アップグレードが正常に完了しない場合、Data Grid Operator がターゲットクラスターをロールバックするときに、パッシベーションによってデータが失われる可能性があります。

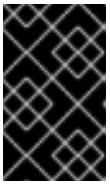
キャッシュ設定でパッシベーションが有効になっている場合は、シャットダウンのアップグレードを実行する必要があります。

6.3. ダウンタイムを伴う DATA GRID クラスターのアップグレード

ダウンタイムを伴う Data Grid クラスターをアップグレードすると、サービスが中断しますが、追加の容量は必要ありません。

前提条件

- インストールした Data Grid Operator バージョンが Data Grid ターゲットバージョンをサポートしている。
- 必要に応じて、アップグレード中にデータを保持するように永続キャッシュストアを設定します。



重要

アップグレードプロセスの開始時に、Data Grid Operator は既存のクラスターをシャットダウンします。これにより、永続キャッシュストアを設定しない場合にデータが失われます。

手順

1. **spec.version** フィールドに Data Grid のバージョン番号を指定します。
2. デフォルトである **spec.upgrades.type** フィールドの値として **Shutdown** が設定されていることを確認します。

```
spec:
  version: 8.4.6-1
  upgrades:
    type: Shutdown
```

3. 必要に応じて、変更を適用します。

新しい Data Grid バージョンが利用可能になったら、**spec.version** フィールドの値を手動で変更して、アップグレードをトリガーする必要があります。

6.4. DATA GRID クラスターの HOT ROD ローリングアップグレードの実行

Hot Rod ローリングアップグレードを実行すると、サービスを中断することなく、新しい Data Grid バージョンに移行できます。ただし、このアップグレードタイプでは追加の容量が必要であり、一時的に異なるバージョンの 2 つの Data Grid クラスターが同時に実行されます。

前提条件

- インストールした Data Grid Operator バージョンが Data Grid ターゲットバージョンをサポートしている。

手順

1. **spec.version** フィールドに Data Grid のバージョン番号を指定します。
2. **spec.upgrades.type** フィールドの値として **HotRodRolling** を指定します。

```
spec:
  version: 8.4.6-1
  upgrades:
    type: HotRodRolling
```

3. 変更を適用します。

新しい Data Grid バージョンが利用可能になったら、**spec.version** フィールドの値を手動で変更して、アップグレードをトリガーする必要があります。

6.4.1. Hot Rod ローリングアップグレードから失敗した場合の回復

元のクラスターがまだ存在する場合は、失敗した Hot Rod ローリングアップグレードを以前のバージョンにロールバックできます。

前提条件

- Hot Rod ローリングアップグレードが進行中で、最初の Data Grid クラスターが存在する。

手順

1. Hot Rod ローリングアップグレードが進行中であることを確認します。

```
oc get infinispan <cr_name> -o yaml
```

status.hotRodRollingUpgradeStatus フィールドが存在する必要があります。

2. **Infinispan CR** の **spec.version** フィールドを、**status.hotRodRollingUpgradeStatus** で定義されている元のクラスターバージョンに更新します。
Data Grid Operator は、新しく作成されたクラスターを削除します。

第7章 DATA GRID サービスの設定

Data Grid Operator を使用して、キャッシュサービスまたは Data Grid サービス Pod のクラスターを作成します。

7.1. サービスの種別

サービスは、Data Grid Server イメージをベースにしたステートフルなアプリケーションであり、柔軟性と堅牢なインメモリーデータストレージを提供します。Data Grid クラスターを作成する場合、**spec.service.type** フィールドでサービスタイプとして **DataGrid** または **Cache** のいずれかを指定します。

DataGrid サービスタイプ

全設定および機能を備えた Data Grid クラスターをデプロイする。

Cache サービスタイプ

最小設定で Data Grid クラスターをデプロイする。

Red Hat では、**DataGrid** サービスタイプは、以下を行うことができるため、クラスターには このサービスタイプを推奨します。

- クロスサイトレプリケーションを使用して、グローバルクラスター全体のデータをバックアップします。
- 有効な設定でキャッシュを作成します。
- ファイルベースのキャッシュストアを追加して、データを永続ボリュームに保存します。
- Data Grid Query API を使用したキャッシュ間で値をクエリーします。
- 高度な Data Grid 機能を使用します。

重要

Cache サービスタイプは、最小設定でレイテンシーを低く抑えたデータストアを便利に作成できるように設計されています。**Infinispan** CRD の開発を進めていくと、**DataGrid** CR のほうが、最終的にユーザーの選択肢が増え、デプロイメントのオーバーヘッドを抑えることができるので、この目的を達成するのにより適したアプローチであることが分かりました。このため、**Cache** サービスタイプは **Infinispan** CRD の次のバージョンで削除される予定で、現在開発は積極的には行われていません。

DataGrid サービスタイプは、クラスターのアップグレードやデータ移行など、複雑な操作を自動化する新機能や拡張ツールから、これからも恩恵を受けます。

7.2. DATA GRID サービス POD の作成

クロスサイトのレプリケーションなどの Data Grid 機能とカスタムキャッシュ定義を使用するには、Data Grid サービス Pod のクラスターを作成します。

手順

1. **spec.service.type: DataGrid** を設定し、他の Data Grid サービスリソースを設定する **Infinispan** CR を作成します。

```
apiVersion: infinispan.org/v1
kind: Infinispan
metadata:
  name: infinispan
spec:
  replicas: 2
  version: <Data Grid_version>
  service:
    type: DataGrid
```

重要

Pod の作成後に **spec.service.type** フィールドを変更することはできません。サービスタイプを変更するには、既存の Pod を削除してから新規の Pod を作成する必要があります。

2. **Infinispan** CR を適用して、クラスターを作成します。

7.2.1. Data Grid サービス CR

このトピックでは、Data Grid サービス Pod の **Infinispan** CR について説明します。

```
apiVersion: infinispan.org/v1
kind: Infinispan
metadata:
  name: infinispan
  annotations:
    infinispan.org/monitoring: 'true'
spec:
  replicas: 6
  version: 8.4.6-1
  upgrades:
    type: Shutdown
  service:
    type: DataGrid
    container:
      storage: 2Gi
      # The ephemeralStorage and storageClassName fields are mutually exclusive.
      ephemeralStorage: false
      storageClassName: my-storage-class
    sites:
      local:
      name: azure
      expose:
        type: LoadBalancer
      locations:
      - name: azure
        url: openshift://api.azure.host:6443
        secretName: azure-token
      - name: aws
        clusterName: infinispan
        namespace: rhdg-namespace
        url: openshift://api.aws.host:6443
        secretName: aws-token
```

```
  security:
    endpointSecretName: endpoint-identities
    endpointEncryption:
        type: Secret
        certSecretName: tls-secret
  container:
    extraJvmOpts: "-XX:NativeMemoryTracking=summary"
    cpu: "2000m:1000m"
    memory: "2Gi:1Gi"
  logging:
    categories:
      org.infinispan: debug
      org.jgroups: debug
      org.jgroups.protocols.TCP: error
      org.jgroups.protocols.relay.RELAY2: error
  expose:
    type: LoadBalancer
  configMapName: "my-cluster-config"
  configListener:
    enabled: true
  affinity:
    podAntiAffinity:
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 100
        podAffinityTerm:
          labelSelector:
            matchLabels:
              app: infinispan-pod
              clusterName: infinispan
              infinispan_cr: infinispan
          topologyKey: "kubernetes.io/hostname"
```

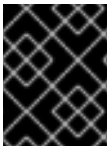
フィールド	説明
metadata.name	Data Grid クラスターに名前を付けます。
metadata.annotations.infinispan.org/monitoring	クラスターの **ServiceMonitor** を自動的に作成します。
spec.replicas	クラスター内の Pod の数を指定します。
spec.version	クラスターの Data Grid サーバーのバージョンを指定します。
spec.upgrades.type	新しいバージョンが利用可能になったときに、Data Grid Operator が Data Grid クラスターをアップグレードする方法を制御します。
spec.service.type	タイプ Data Grid サービスを設定します。**DataGrid** の値は、Data Grid サービス Pod でクラスターを作成します。

フィールド	説明
spec.service.container	Data Grid サービス Pod のストレージリソースを設定します。
spec.service.sites	クロスサイトのレプリケーションを設定します。
spec.security.endpointSecretName	Data Grid のユーザー認証情報が含まれる認証シークレットを指定します。
spec.security.endpointEncryption	TLS 証明書およびキーストアを指定して、クライアント接続を暗号化します。
spec.container	Data Grid Pod の JVM、CPU、およびメモリーリソースを指定します。
spec.logging	Data Grid のロギングカテゴリーを設定します。
spec.expose	ネットワーク上で Data Grid エンドポイントを公開する方法を制御します。
spec.configMapName	Data Grid の設定が含まれる **ConfigMap** を指定します。
spec.configListener.enabled	各 Data Grid クラスターに **listener** Pod を作成します。これにより、Data Grid Operator はサーバー側の変更を **Cache** CR などの Data Grid リソースと調整できます。 **listener** Pod は最小限のリソースを消費し、デフォルトで有効になっています。**false** の値を設定すると、**listener** Pod が削除され、双方向の調整が無効になります。これは、Data Grid Console、CLI、またはクライアントアプリケーションを介して作成された Data Grid リソースの宣言型 Kubernetes 表現が必要ない場合にのみ行う必要があります。
spec.configListener.logging.level	**ConfigListener** デプロイメントのロギングレベルを設定します。デフォルトのレベルは **info** です。**debug** または **error** に変更できます。
spec.affinity	Data Grid の可用性を保証する anti-affinity ストラテジーを設定します。

7.3. ストレージリソースの割り当て

Data Grid サービス Pod にストレージを割り当てることはできますが、キャッシュサービス Pod には割り当てることはできません。

デフォルトで、Data Grid Operator は永続ボリューム要求に **1Gi** を割り当てます。ただし、シャットダウン時に Data Grid がクラスターの状態を保持できるように、Data Grid サービス Pod で利用可能なストレージの容量を調整する必要があります。



重要

利用可能なコンテナーストレージが、利用可能なメモリー量を下回ると、データ損失が発生する可能性があります。

手順

1. ストレージリソースを **spec.service.container.storage** フィールドで割り当てます。
2. 必要に応じて、**ephemeralStorage** フィールドまたは storageClassName **フィールド** を設定します。

注記

これらのフィールドは同時に使用でいません。どちらか 1 つだけを **Infinispan** CR に追加します。

3. 変更を適用します。

一時ストレージ

```
spec:
  service:
    type: DataGrid
    container:
      storage: 2Gi
      ephemeralStorage: true
```

StorageClass オブジェクトの名前

```
spec:
  service:
    type: DataGrid
    container:
      storage: 2Gi
      storageClassName: my-storage-class
```

フィールド	説明
spec.service.container.storage	Data Grid サービス Pod のストレージの量を指定します。

フィールド	説明
spec.service.container.ephemeralStorage	ストレージが一時的または永続的であるかどうかを定義します。一時ストレージを使用するには、値を **true** に設定します。これは、クラスターのシャットダウンまたは再起動時に、ストレージ内のすべてのデータが削除されることを意味します。デフォルト値は **false** です。これは、ストレージが永続的であることを意味します。
spec.service.container.storageClassName	永続ボリューム要求 (PVC) に使用する **StorageClass** オブジェクトの名前を指定します。このフィールドを含める場合は、既存のストレージクラスを値として指定する必要があります。このフィールドを含めない場合、永続ボリューム要求は **storageclass.kubernetes.io/is-default-class** アノテーションが **true** に設定されたストレージクラスを使用します。

7.3.1. Persistent Volume Claim (永続ボリューム要求)

Data Grid Operator は永続ボリューム要求 (PVC) を作成し、コンテナーストレージを **/opt/infinispan/server/data** にマウントします。

キャッシュ

キャッシュを作成すると、Data Grid はクラスターの再起動後にキャッシュを使用できるように、設定を永続的に保存します。これは、キャッシュサービス Pod と Data Grid サービス Pod の両方に適用されます。

データ

キャッシュサービス Pod のクラスターでは、データは常に揮発性です。クラスターをシャットダウンすると、データが完全に失われます。

クラスターのシャットダウン中に Data Grid サービス Pod でデータを永続化させる場合は、**<file-store/>** 要素を Data Grid キャッシュ設定に追加して、ファイルベースのキャッシュストアを使用します。

7.4. CPU およびメモリーの割り当て

Infinispan CR を使用して CPU およびメモリーリソースを Data Grid Pod に割り当てます。

注記

Data Grid Operator は、Data Grid Pod の作成時に OpenShift スケジューラーから **1Gi** のメモリーを要求します。CPU 要求はデフォルトでバインドされません。

手順

1. **spec.container.cpu** フィールドで CPU ユニットの数を割り当てます。

2. メモリーの量 (バイト単位) を **spec.container.memory** フィールドで割り当てます。
   cpu および **memory** フィールドでは **<limit>:<requests>** 形式で値を指定します。たとえば、**cpu: "2000m:1000m"** は Pod を最大 **2000m** の CPU に制限し、起動時に各 Pod に対して **1000m** の CPU を要求します。値を 1 つ指定すると、制限と要求の両方が設定されます。

3. **Infinispan** CR を適用します。
   クラスターが実行されている場合には、Data Grid Operator は Data Grid Pod を再起動して変更を有効にします。

```
spec:
  container:
    cpu: "2000m:1000m"
    memory: "2Gi:1Gi"
```

7.5. JVM オプションの設定

起動時に追加の JVM オプションを Data Grid Pod に渡します。

手順

1. **Infinispan** CR にファイルされた **spec.container** を使用して JVM オプションを設定します。

2. **Infinispan** CR を適用します。
   クラスターが実行されている場合には、Data Grid Operator は Data Grid Pod を再起動して変更を有効にします。

JVM オプション

```
spec:
  container:
    extraJvmOpts: "-<option>=<value>"
    routerExtraJvmOpts: "-<option>=<value>"
    cliExtraJvmOpts: "-<option>=<value>"
```

フィールド	説明
spec.container.extraJvmOpts	Data Grid サーバーの追加の JVM オプションを指定します。
spec.container.routerExtraJvmOpts	Gossip ルーターの追加の JVM オプションを指定します。
spec.container.cliExtraJvmOpts	Data Grid CLI の追加の JVM オプションを指定します。

7.6. POD プローブの設定

必要に応じて、Data Grid Pod で使用される Liveness、Readiness、および Startup プローブの値を設定します。

Data Grid Operator は、プローブ値を適切なデフォルト値に自動的に設定します。デフォルト値が要件に一致しないと判断した場合にのみ、独自の値を指定することを推奨します。

手順

1. **spec.service.container.*Probe** フィールドを使用してプローブ値を設定します。

```
spec:
  service:
    container:
      readinessProbe:
        failureThreshold: 1
        initialDelaySeconds: 1
        periodSeconds: 1
        successThreshold: 1
        timeoutSeconds: 1
      livenessProbe:
        failureThreshold: 1
        initialDelaySeconds: 1
        periodSeconds: 1
        successThreshold: 1
        timeoutSeconds: 1
      startupProbe:
        failureThreshold: 1
        initialDelaySeconds: 1
        periodSeconds: 1
        successThreshold: 1
        timeoutSeconds: 1
```

重要

特定のプローブ値に値が指定されていない場合、Data Grid Operator のデフォルトが使用されます。

2. **Infinispan** CR を適用します。
クラスターが実行中の場合、変更を有効にするために、Data Grid Operator が Data Grid Pod を再起動します。

7.7. POD の優先順位の設定

1 つ以上の優先クラスを作成して、他の Pod と比較した Pod の重要性を示します。優先度の高い Pod は、優先度の低い Pod よりも先にスケジュールされ、特にリソースが制約されている場合に、重要なワークロードを実行する Pod の優先順位が確保されます。

前提条件

- OpenShift への **cluster-admin** アクセス権がある。

手順

1. 名前と値を指定して、**PriorityClass** オブジェクトを定義します。

high-priority.yaml

```
apiVersion: scheduling.k8s.io/v1
kind: PriorityClass
metadata:
  name: high-priority
value: 1000000
globalDefault: false
description: "Use this priority class for high priority service pods only."
```

2. 優先クラスを作成します。

```
oc create -f high-priority.yaml
```

3. Pod 設定で優先クラス名を参照します。

Infinispan CR

```
kind: Infinispan
...
spec:
  scheduling:
    affinity:
      ...
    priorityClassName: "high-priority"
    ...
```

既存の優先クラス名を参照する必要があります。参照しない場合は、Pod が拒否されます。

4. 変更を適用します。

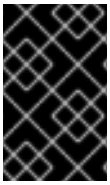
関連情報

- Pod スケジューリングの決定に Pod の優先順位を含める

7.8. INFINISPAN CR の FIPS モード

Red Hat OpenShift Container Platform は、特定の連邦情報処理標準 (FIPS) コンポーネントを使用して、OpenShift クラスターが FIPS コンプライアンス監査の要件を確実に満たすようにすることができます。

OpenShift クラスターで FIPS モードを有効にした場合に、Data Grid Operator は **Infinispan** カスタムリソース (CR) の FIPS モードを自動的に有効にします。



重要

クライアント証明書認証は現在、FIPS モードではサポートされていません。**spec.security.endpointEncryption.clientCert** を **None** 以外の値に設定して **Infinispan** CR を作成しようとすると失敗します。

関連情報

- FIPS 暗号化のサポート Red Hat OpenShift Container Platform

7.9. ログレベルの調整

問題をデバッグする必要がある場合は、さまざまな Data Grid のロギングカテゴリーのレベルを変更します。ログレベルを調整して特定のカテゴリーのメッセージ数を減らし、コンテナーリソースの使用を最小限に抑えることもできます。

手順

1. **Infinispan** CR の **spec.logging.categories** フィールドで、Data Grid ロギングを設定します。

```
spec:
  logging:
    categories:
      org.infinispan: debug
      org.jgroups: debug
```

2. 変更を適用します。
3. 必要に応じて、Data Grid Pod からログを取得します。

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

7.9.1. ロギングの参照

ログカテゴリーとレベルに関する情報を検索します。

表7.1 ログカテゴリー

root カテゴリー	説明	デフォルトレベル
org.infinispan	Data Grid メッセージ	**info**
org.jgroups	クラスタートランスポートメッセージ	**info**

表7.2 ログレベル

ログレベル	説明
trace	アプリケーションの実行状態に関する詳細情報を提供します。これは最も詳細なログレベルです。
debug	個々の要求またはアクティビティーの進捗を示します。
info	ライフサイクルイベントを含むアプリケーションの全体的な進捗状況を示します。
warn	エラーが発生したり、パフォーマンスが低下する可能性のある状況を示します。

ログレベル	説明
error	操作またはアクティビティーの正常な実行を妨げる可能性がありますが、アプリケーションの実行は妨げないエラー状態を示します。

ガベージコレクション (GC) メッセージ

Data Grid Operator はデフォルトで GC メッセージをログに記録しません。以下の JVM オプションを使用して、GC メッセージを **stdout** に転送することができます。

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

7.10. キャッシュサービス POD の作成

最小限の設定で、揮発性があり、レイテンシーの低いデータストアのキャッシュサービス Pod を使用して、Data Grid クラスターを作成します。

重要

キャッシュサービス Pod は揮発性ストレージのみを提供します。つまり、**Infinispan** CR を変更したり、Data Grid クラスターのバージョンを更新すると、すべてのデータが失われます。

手順

1. **spec.service.type: Cache** を設定し、他のキャッシュサービスリソースを設定する **Infinispan** CR を作成します。

```
apiVersion: infinispan.org/v1
kind: Infinispan
metadata:
  name: infinispan
spec:
  replicas: 2
  version: <Data Grid_version>
  service:
    type: Cache
```

2. **Infinispan** CR を適用して、クラスターを作成します。

7.10.1. キャッシュサービス CR

このトピックでは、キャッシュサービス Pod の **Infinispan** CR について説明します。

```
apiVersion: infinispan.org/v1
kind: Infinispan
metadata:
  name: infinispan
```

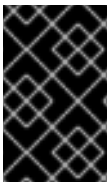
```
  annotations:
    infinispan.org/monitoring: 'true'
spec:
  replicas: 2
  version: 8.4.6-1
  upgrades:
    type: Shutdown
  service:
    type: Cache
    replicationFactor: 2
  autoscale:
    maxMemUsagePercent: 70
    maxReplicas: 5
    minMemUsagePercent: 30
    minReplicas: 2
  security:
    endpointSecretName: endpoint-identities
    endpointEncryption:
        type: Secret
        certSecretName: tls-secret
  container:
    extraJvmOpts: "-XX:NativeMemoryTracking=summary"
    cpu: "2000m:1000m"
    memory: "2Gi:1Gi"
  logging:
    categories:
      org.infinispan: trace
      org.jgroups: trace
  expose:
    type: LoadBalancer
  affinity:
    podAntiAffinity:
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 100
        podAffinityTerm:
          labelSelector:
            matchLabels:
              app: infinispan-pod
              clusterName: infinispan
              infinispan_cr: infinispan
          topologyKey: "kubernetes.io/hostname"
```

フィールド	説明
metadata.name	Data Grid クラスターに名前を付けます。
metadata.annotations.infinispan.org/monitoring	クラスターの **ServiceMonitor** を自動的に作成します。
spec.replicas	クラスター内の Pod の数を指定します。自動スケーリング機能を有効にする場合、このフィールドは Pod の初期数を指定します。

フィールド	説明
spec.version	クラスターの Data Grid サーバーのバージョンを指定します。
spec.upgrades.type	新しいバージョンが利用可能になったときに、Data Grid Operator が Data Grid クラスターをアップグレードする方法を制御します。
spec.service.type	タイプ Data Grid サービスを設定します。**Cache** の値は、キャッシュサービス Pod を使用してクラスターを作成します。
spec.service.replicationFactor	クラスター全体で各エントリーのコピー数を設定します。キャッシュサービス Pod のデフォルトは 2 で、データの損失を回避するために各キャッシュエントリーを複製します。
spec.autoscale	自動スケーリングを有効にし、設定します。
spec.security.endpointSecretName	Data Grid のユーザー認証情報が含まれる認証シークレットを指定します。
spec.security.endpointEncryption	TLS 証明書およびキーストアを指定して、クライアント接続を暗号化します。
spec.container	Data Grid Pod の JVM、CPU、およびメモリーリソースを指定します。
spec.logging	Data Grid のロギングカテゴリーを設定します。
spec.expose	ネットワーク上で Data Grid エンドポイントを公開する方法を制御します。
spec.affinity	Data Grid の可用性を保証する anti-affinity ストラテジーを設定します。

7.11. 自動スケーリング

Data Grid Operator は、メモリー使用量に基づいて Pod を作成または削除することにより、キャッシュサービス Pod でデフォルトのキャッシュを監視して、クラスターを自動的にスケールアップまたはスケールダウンできます。



重要

自動スケーリングは、キャッシュサービス Pod のクラスターでのみ利用できます。Data Grid Operator は、Data Grid サービス Pod のクラスターの自動スケーリングを実行しません。

自動スケーリングを有効にすると、Data Grid Operator が Pod を作成または削除する必要があるタイミングを決定できるメモリー使用しきい値を定義します。Data Grid Operator は、デフォルトキャッシュの統計を監視し、メモリー使用量が設定されたしきい値に達すると、クラスターをスケールアップまたはスケールダウンします。

最大しきい値

このしきい値は、スケールアップまたはエビクションを実行する前に、クラスター内の Pod が使用できるメモリー量の上限を設定します。Data Grid Operator は、設定するメモリーの最大容量にノードが達することを検出すると、可能な場合は新規のノードを作成します。Data Grid Operator が新規のノードを作成できない場合、メモリー使用量が 100 パーセントに達すると、エビクションを実行します。

最小しきい値

このしきい値は、Data Grid クラスター全体のメモリー使用量の下限を設定します。Data Grid Operator は、メモリー使用量が最小値を下回ったことを検出すると、Pod をシャットダウンします。

デフォルトのキャッシュのみ

自動スケーリング機能は、デフォルトのキャッシュでのみ動作します。他のキャッシュをクラスターに追加する予定の場合、**Infinispan** CR に **autoscale** フィールドを含めないでください。この場合、エビクションを使用して、各ノードのデータコンテナーのサイズを制御する必要があります。

7.11.1. 自動スケーリングの設定

キャッシュサービス Pod でクラスターを作成する場合、Data Grid Operator をクラスターを自動的にスケーリングするように設定できます。

手順

1. **spec.autoscale** リソースを **Infinispan** CR に追加し、自動スケーリングを有効にします。

注記

自動スケーリングを無効にするには、**autoscale.disabled** フィールドに **true** の値を設定します。

2. 以下のフィールドを使用して、自動スケーリングのしきい値を設定します。

フィールド	説明
spec.autoscale.maxMemUsagePercent	各ノードのメモリー使用量の最大しきい値をパーセンテージで指定します。
spec.autoscale.maxReplicas	クラスターのキャッシュサービス Pod の最大数を指定します。
spec.autoscale.minMemUsagePercent	クラスターのメモリー使用量の最小しきい値をパーセンテージで指定します。
spec.autoscale.minReplicas	クラスターのキャッシュサービス Pod の最小数を指定します。

たとえば、以下を **Infinispan** CR に追加します。

```
spec:
  service:
    type: Cache
  autoscale:
    disabled: false
    maxMemUsagePercent: 70
    maxReplicas: 5
    minMemUsagePercent: 30
    minReplicas: 2
```

3. 変更を適用します。

7.12. DATA GRID リソースへのラベルとアノテーションの追加

Data Grid Operator が作成および管理する Pod とサービスに、キー/値のラベルとアノテーションを添付します。ラベルは、オブジェクト間の関係を識別して、Data Grid リソースをより適切に整理および監視するのに役立ちます。アノテーションは、クライアントアプリケーションまたはデプロイメントおよび管理ツールの任意の非識別メタデータです。

注記

Red Hat サブスクリプションラベルは、Data Grid リソースに自動的に適用されます。

手順

1. **Infinispan** CR を開いて編集します。
2. ラベルとアノテーションを **metadata.annotations** セクションの Data Grid リソースに添付します。
   - アノテーションの値を **metadata.annotations** セクションで直接定義します。
   - **metadata.labels** フィールドを使用してラベルの値を定義します。
3. **Infinispan** CR を適用します。

カスタムアノテーション

```
apiVersion: infinispan.org/v1
kind: Infinispan
metadata:
  annotations:
    infinispan.org/targetAnnotations: service-annotation1, service-annotation2
    infinispan.org/podTargetAnnotations: pod-annotation1, pod-annotation2
    infinispan.org/routerAnnotations: router-annotation1, router-annotation2

    service-annotation1: value
    service-annotation2: value
    pod-annotation1: value
    pod-annotation2: value
    router-annotation1: value
    router-annotation2: value
```

カスタムラベル

```
apiVersion: infinispan.org/v1
kind: Infinispan
metadata:
  annotations:
    infinispan.org/targetLabels: service-label1, service-label2
    infinispan.org/podTargetLabels: pod-label1, pod-label2
  labels:
    service-label1: value
    service-label2: value
    pod-label1: value
    pod-label2: value
    # The operator does not attach these labels to resources.
    my-label: my-value
    environment: development
```

7.13. 環境変数を使用したラベルとアノテーションの追加

Data Grid Operator の環境変数を設定して、すべての Data Grid Pod とサービスに自動的に伝播するラベルとアノテーションを追加します。

手順

spec.config.env フィールドを使用して、以下のいずれかの方法で、Data Grid Operator サブスクリプションにラベルとアノテーションを追加します。

- **oc edit subscription** コマンドを使用します。

```
oc edit subscription datagrid -n openshift-operators
```

- Red Hat OpenShift Console を使用します。
  1. **Operators** > **Installed Operators** > **Data Grid Operator** に移動します。
  2. **Actions** メニューから **Edit Subscription** を選択します。

環境変数を使用したラベルとアノテーション

```
spec:
  config:
    env:
      - name: INFINISPAN_OPERATOR_TARGET_LABELS
        value: |
         {"service-label1":"value",
         service-label1":"value"}
      - name: INFINISPAN_OPERATOR_POD_TARGET_LABELS
        value: |
         {"pod-label1":"value",
         "pod-label2":"value"}
      - name: INFINISPAN_OPERATOR_TARGET_ANNOTATIONS
        value: |
         {"service-annotation1":"value",
         "service-annotation2":"value"}
      - name: INFINISPAN_OPERATOR_POD_TARGET_ANNOTATIONS
```

```
value: |
 {"pod-annotation1":"value",
 "pod-annotation2":"value"}
```

7.14. DATA GRID OPERATOR サブスクリプションの環境変数の定義

Data Grid Operator サブスクリプションの作成時または編集時に、サブスクリプションの環境変数を定義できます。

注記

Red Hat OpenShift Console を使用している場合は、まず Data Grid Operator をインストールしてから、既存のサブスクリプションを編集する必要があります。

spec.config.env フィールド

環境変数を定義するための **name** フィールドと **value** フィールドが含まれます。

ADDITIONAL_VARS 変数

環境変数の名前が JSON 配列の形式で含まれます。**ADDITIONAL_VARS** 変数の **value** 内の環境変数は、関連付けられた Operator によって管理される各 Data Grid Server Pod に自動的に伝播します。

前提条件

- Operator Lifecycle Manager (OLM) がインストールされている。
- **oc** クライアントがある。

手順

1. Data Grid Operator のサブスクリプション定義 YAML を作成します。
   a. **spec.config.env** フィールドを使用して環境変数を定義します。
   b. **ADDITIONAL_VARS** 変数内に、JSON 配列の環境変数名を追加します。

   subscription-datagrid.yaml

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: datagrid
  namespace: openshift-operators
spec:
  channel: 8.4.x
  installPlanApproval: Automatic
  name: datagrid
  source: redhat-operators
  sourceNamespace: openshift-marketplace
  config:
    env:
    - name: ADDITIONAL_VARS
      value: "[\"VAR_NAME\", \"ANOTHER_VAR\"]"
    - name: VAR_NAME
```

```
      value: $(VAR_NAME_VALUE)
    - name: ANOTHER_VAR
      value: $(ANOTHER_VAR_VALUE)
```

たとえば、環境変数を使用してローカルタイムゾーンを設定します。

subscription-datagrid.yaml

```
kind: Subscription
spec:
  ...
  config:
    env:
    - name: ADDITIONAL_VARS
      value: "[\"TZ\"]"
    - name: TZ
      value: "JST-9"
```

2. Data Grid Operator のサブスクリプションを作成します。

```
oc apply -f subscription-datagrid.yaml
```

検証

- **subscription-datagrid.yaml** から環境変数を取得します。

```
oc get subscription datagrid -n openshift-operators -o jsonpath='{.spec.config.env[*].name}'
```

次のステップ

1. **oc edit subscription** コマンドを使用して、環境変数を変更します。

```
oc edit subscription datagrid -n openshift-operators
```

2. 変更を Data Grid クラスターに確実に反映するには、既存のクラスターを再作成する必要があります。既存の **Infinispan** CR に関連する **StatefulSet** を削除して、Pod を終了します。

- Red Hat OpenShift コンソールで、**Operators** > **Installed Operators** > **Data Grid Operator** に移動します。**Actions** メニューから **Edit Subscription** を選択します。

第8章 認証の設定

アプリケーションユーザーには、Data Grid クラスターにアクセスするために認証情報が必要です。デフォルトの生成された認証情報を使用するか、独自の認証情報を追加することができます。

8.1. デフォルトの認証情報

Data Grid Operator は、以下のユーザーの base64 でエンコードされた認証情報を生成します。

ユーザー	Secret 名	説明
developer	**infinispan-generated-secret**	デフォルトのアプリケーションユーザーの認証情報。
operator	**infinispan-generated-operator-secret**	Data Grid Operator が Data Grid リソースとの対話に使用する認証情報。

8.2. 認証情報の取得

Data Grid クラスターにアクセスするために、認証シークレットから認証情報を取得します。

手順

- 認証シークレットから認証情報を取得します。

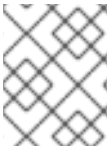
```
oc get secret infinispan-generated-secret
```

Base64 でデコードされた認証情報。

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

8.3. カスタムユーザーの認証情報の追加

カスタム認証情報を使用して Data Grid クラスターエンドポイントへのアクセスを設定します。



注記

spec.security.endpointSecretName を変更すると、クラスターの再起動がトリガーされます。

手順

1. 追加する認証情報を使用して **identities.yaml** ファイルを作成します。

```
credentials:
- username: myfirstusername
  password: changeme-one
```

```
- username: myseconduser name
  password: changeme-two
```

2. **identities.yaml** から認証シークレットを作成します。

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

3. **Infinispan** CR の **spec.security.endpointSecretName** で認証シークレットを指定し、変更を適用します。

```
spec:
  security:
    endpointSecretName: connect-secret
```

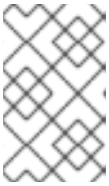
8.4. OPERATOR パスワードの変更

自動生成されたパスワードを使用しない場合は、**operator** ユーザーのパスワードを変更できます。

手順

- 以下のように **infinispan-generated-operator-secret** シークレットの **password** キーを更新します。

```
oc patch secret infinispan-generated-operator-secret -p='{"stringData":{"password":
"supersecretoperatorpassword"}}'
```

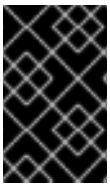


注記

generated-operator-secret シークレットの **password** キーのみを更新する必要があります。パスワードを更新すると、Data Grid Operator はそのシークレットの他のキーを自動的に更新します。

8.5. ユーザー認証の無効化

ユーザーが Data Grid クラスターにアクセスでき、認証情報を提供せずにデータを操作できるようにします。



重要

エンドポイントが **spec.expose.type** を介して OpenShift クラスターの外部からアクセスできる場合には、認証を無効にしないでください。開発環境の認証のみを無効にする必要があります。

手順

1. **false** を **Infinispan** CR の **spec.security.endpointAuthentication** フィールドの値として設定します。

```
spec:
  security:
    endpointAuthentication: false
```

2. 変更を適用します。

第9章 クライアント証明書認証の設定

プロジェクトにクライアントトラストストアを追加し、有効な証明書を提示するクライアントからのみの接続を許可するように Data Grid を設定します。これにより、クライアントがパブリック認証局 (CA) によって信頼されることが確認され、デプロイメントのセキュリティーが向上します。

9.1. クライアント証明書認証

クライアント証明書認証は、クライアントが提示する証明書に基づいて、インバウンド接続を制限します。

以下のいずれかのストラテジーのトラストストアを使用するように Data Grid を設定できます。

検証

クライアント証明書を検証するには、署名認証局 (通常は root CA 証明書) の証明書チェーンの一部が含まれるトラストストアが Data Grid に必要となります。CA によって署名された証明書を提示するクライアントは、Data Grid に接続できます。

クライアント証明書を検証するために **Validate** ストラテジーを使用する場合、認証を有効にする場合は有効な Data Grid 認証情報を提供するようにクライアントも設定する必要があります。

認証

root CA 証明書に加えて、すべてのパブリッククライアント証明書が含まれるトラストストアが必要です。署名済み証明書を提示するクライアントのみが Data Grid に接続できます。

クライアント証明書を検証するために **Authenticate** ストラテジーを使用する場合、識別名 (DN) の一部として、証明書に有効な Data Grid 認証情報が含まれていることを確認する必要があります。

9.2. クライアント証明書認証の有効化

クライアント証明書認証を有効にするには、**Validate** または **Authenticate** ストラテジーのいずれかでトラストストアを使用するように Data Grid を設定します。

手順

1. **Infinispan** CR の **spec.security.endpointEncryption.clientCert** フィールドの値として、**Validate** または **Authenticate** を設定します。

注記

デフォルト値は **None** です。

2. クライアントトラストストアを含むシークレットを **spec.security.endpointEncryption.clientCertSecretName** フィールドで指定します。デフォルトでは、Data Grid Operator は **<cluster-name>-client-cert-secret** という名前のトラストストアシークレットを想定します。

注記

シークレットは OpenShift クラスターの各 **Infinispan** CR インスタンスに固有のものである必要があります。**Infinispan** CR を削除すると、OpenShift は関連付けられたシークレットも自動的に削除します。

```
spec:
  security:
    endpointEncryption:
        type: Secret
        certSecretName: tls-secret
        clientCert: Validate
        clientCertSecretName: infinispan-client-cert-secret
```

3. 変更を適用します。

次のステップ

すべてのクライアント証明書が含まれるトラストストアに Data Grid Operator を提供します。または、PEM 形式で証明書を提供し、Data Grid にクライアントトラストストアを生成させることもできます。

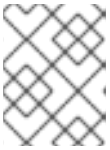
9.3. クライアントトラストストアの提供

必要な証明書が含まれるトラストストアがある場合は、Data Grid Operator で利用可能にすることができます。

Data Grid は、**PKCS12** 形式のトラストストアのみをサポートします。

手順

1. クライアントトラストストアを含むシークレットの名前を **metadata.name** フィールドの値として指定します。



注記

この名前は **spec.security.endpointEncryption.clientCertSecretName** フィールドの値に一致する必要があります。

2. トラストストアのパスワードを **stringData.truststore-password** フィールドで指定します。
3. **data.truststore.p12** フィールドでトラストストアを指定します。

```
apiVersion: v1
kind: Secret
metadata:
  name: infinispan-client-cert-secret
type: Opaque
stringData:
    truststore-password: changme
data:
    truststore.p12:  "<base64_encoded_PKCS12_trust_store>"
```

4. 変更を適用します。

9.4. クライアント証明書の提供

Data Grid Operator は、PEM 形式の証明書からトラストストアを生成できます。

手順

1. クライアントトラストストアを含むシークレットの名前を **metadata.name** フィールドの値として指定します。

注記

この名前は **spec.security.endpointEncryption.clientCertSecretName** フィールドの値に一致する必要があります。

2. 署名証明書または CA 証明書バンドルを **data.trust.ca** フィールドの値として指定します。

3. **Authenticate** ストラテジーを使用してクライアント ID を検証する場合は、**data.trust.cert.<name>** フィールドで Data Grid エンドポイントに接続可能な各クライアントの証明書を追加します。

注記

Data Grid Operator は、**<name>** の値をトラストストアの生成時に証明書のエイリアスとして使用します。

4. オプションで、**stringData.truststore-password** フィールドでトラストストアのパスワードを指定します。
指定しない場合、Data Grid Operator は password をトラストストアのパスワードとして設定します。

```
apiVersion: v1
kind: Secret
metadata:
  name: infinispan-client-cert-secret
type: Opaque
stringData:
    truststore-password: changme
data:
    trust.ca: "<base64_encoded_CA_certificate>"
    trust.cert.client1: "<base64_encoded_client_certificate>"
    trust.cert.client2: "<base64_encoded_client_certificate>"
```

5. 変更を適用します。

第10章 暗号化の設定

Red Hat OpenShift サービス証明書またはカスタム TLS 証明書で、クライアントと Data Grid Pod との間の接続を暗号化します。

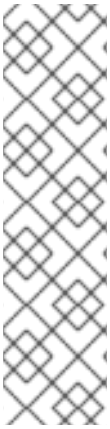
10.1. RED HAT OPENSHIFT サービス証明書を使用した暗号化

Data Grid Operator は、Red Hat OpenShift サービス CA によって署名された TLS 証明書を自動的に生成します。次に、Data Grid Operator は証明書およびキーをシークレットに格納し、それらを取得してリモートクライアントで使用できるようにします。

Red Hat OpenShift サービス CA が利用可能な場合、Data Grid Operator は以下の **spec.security.endpointEncryption** 設定を **Infinispan** CR に追加します。

```
spec:
  security:
    endpointEncryption:
      type: Service
      certServiceName: service.beta.openshift.io
      certSecretName: infinispan-cert-secret
```

フィールド	説明
spec.security.endpointEncryption.certServiceName	TLS 証明書を提供するサービスを指定します。
spec.security.endpointEncryption.certSecretName	サービス証明書およびキーで PEM 形式のシークレットを指定します。デフォルトは **<cluster_name>-cert-secret** です。



注記

サービス証明書は、Data Grid クラスターの内部 DNS 名を共通名 (CN) として使用します。以下に例を示します。

Subject: CN = example-infinispan.mynamespace.svc

このため、サービス証明書は OpenShift 内でのみ全面的に信頼することができます。OpenShift の外部で実行しているクライアントとの接続を暗号化する場合は、カスタム TLS 証明書を使用する必要があります。

サービス証明書は 1 年間有効で、有効期限が切れる前に自動的に置き換えられます。

10.2. TLS 証明書の取得

暗号化シークレットから TLS 証明書を取得して、クライアントトラストストアを作成します。

手順

- 以下のように、暗号化シークレットから **tls.crt** を取得します。

```
oc get secret infinispan-cert-secret -o jsonpath='{.data.tls\.crt}' | base64 --decode > tls.crt
```

10.3. 暗号化の無効化

クライアントが Data Grid との接続を確立するために TLS 証明書を必要としないように、暗号化を無効化することができます。

重要

エンドポイントが **spec.expose.type** を介して OpenShift クラスターの外部からアクセスできる場合には、暗号化を無効化にしないでください。開発環境の暗号化のみを無効にする必要があります。

手順

1. **None** を **Infinispan** CR の **spec.security.endpointEncryption.type** フィールドの値として設定します。

```
spec:
  security:
    endpointEncryption:
      type: None
```

2. 変更を適用します。

10.4. カスタム TLS 証明書の使用

カスタムの PKCS12 キーストアまたは TLS 証明書/キーのペアを使用して、クライアントと Data Grid クラスターとの間の接続を暗号化します。

前提条件

- キーストアまたは証明書シークレットを作成します。

注記

シークレットは OpenShift クラスターの各 **Infinispan** CR インスタンスに固有のものである必要があります。**Infinispan** CR を削除すると、OpenShift は関連付けられたシークレットも自動的に削除します。

手順

1. 暗号化シークレットを OpenShift namespace に追加します。以下に例を示します。

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

2. **Infinispan** CR の **spec.security.endpointEncryption.certSecretName** フィールドで暗号化シークレットを指定します。

```
spec:
  security:
    endpointEncryption:
```

```
    type: Secret
    certSecretName: tls-secret
```

3. 変更を適用します。

10.4.1. カスタム暗号化シークレット

キーストアまたは証明書/キーペアを追加して Data Grid の接続をセキュアに保つカスタムの暗号化シークレットには、特定のフィールドを含める必要があります。

キーストアシークレット

```
apiVersion: v1
kind: Secret
metadata:
  name: tls-secret
type: Opaque
stringData:
  alias: server
  password: changeme
data:
  keystore.p12:  "MIIKDgIBAzCCCdQGCSqGSIb3DQEHA..."
```

フィールド	説明
stringData.alias	キーストアのエイリアスを指定します。
stringData.password	キーストアのパスワードを指定します。
data.keystore.p12	base64 でエンコードされたキーストアを追加します。

証明書シークレット

```
apiVersion: v1
kind: Secret
metadata:
  name: tls-secret
type: Opaque
data:
  tls.key:  "LS0tLS1CRUdJTiBQUk ..."
  tls.crt: "LS0tLS1CRUdJTiBDRVI ..."
```

フィールド	説明
data.tls.key	base64 でエンコードされた TLS キーを追加します。
data.tls.crt	base64 でエンコードされた TLS 証明書を追加します。

第11章 ユーザーロールとパーミッションの設定

ユーザーのロールベースアクセス制御 (RBAC) を設定して、Data Grid サービスへのアクセスの安全性を確保します。これには、キャッシュおよび Data Grid リソースにアクセスするパーミッションがあるように、ロールをユーザーに割り当てる必要があります。

11.1. セキュリティー承認の有効化

デフォルトでは、**Infinispan** CR インスタンスとの後方互換性を確保するために、承認は無効になっています。以下の手順を実行して承認を有効にし、Data Grid ユーザーのロールベースアクセス制御 (RBAC) を使用します。

手順

1. **true** を **Infinispan** CR の **spec.security.authorization.enabled** フィールドの値として設定します。

```
spec:
  security:
    authorization:
      enabled: true
```

2. 変更を適用します。

11.2. ユーザーロールとパーミッション

Data Grid Operator は、さまざまなパーミッションに関連付けられたデフォルトロールのセットを提供します。

表11.1 デフォルトのロールおよびパーミッション

Role	権限	説明
admin	ALL	Cache Manager ライフサイクルの制御など、すべてのパーミッションを持つスーパーユーザー。
deployer	ALL_READ、ALL_WRITE、LISTEN、EXEC、MONITOR、CREATE	**application** パーミッションに加えて、Data Grid リソースを作成および削除できます。
application	ALL_READ、ALL_WRITE、LISTEN、EXEC、MONITOR	**observer** パーミッションに加え、Data Grid リソースへの読み取りおよび書き込みアクセスがあります。また、イベントをリッスンし、サーバータスクおよびスクリプトを実行することもできます。
observer	ALL_READ、MONITOR	**monitor** パーミッションに加え、Data Grid リソースへの読み取りアクセスがあります。

Role	権限	説明
monitor	MONITOR	Data Grid クラスターの統計を表示できます。

Data Grid Operator の認証情報

Data Grid Operator は、Data Grid クラスターでの認証に使用する認証情報を生成し、内部操作を実行します。デフォルトでは、セキュリティー承認を有効にすると、Data Grid Operator の認証情報に **admin** ロールが自動的に割り当てられます。

関連情報

- セキュリティー承認の仕組み (**Data Grid セキュリティーガイド**)

11.3. ロールおよびパーミッションのユーザーへの割り当て

ユーザーが Data Grid クラスターリソースにアクセスできるかどうかを制御するロールをユーザーに割り当てます。ロールには、読み取り専用のアクセスから無制限のアクセスまで、さまざまなパーミッションレベルを設定できます。

注記

ユーザーは暗黙的に認証を取得します。たとえば、admin は **admin** パーミッションを自動的に取得します。deployer という名前のユーザーには、自動的に **deployer** ロールがあります。

手順

1. ロールをユーザーに割り当てる **identities.yaml** ファイルを作成します。

```
credentials:
  - username: admin
    password: changeme
  - username: my-user-1
    password: changeme
    roles:
      - admin
  - username: my-user-2
    password: changeme
    roles:
      - monitor
```

2. **identities.yaml** から認証シークレットを作成します。
必要に応じて、既存のシークレットを最初に削除します。

```
oc delete secret connect-secret --ignore-not-found
oc create secret generic --from-file=identities.yaml connect-secret
```

3. **Infinispan** CR の **spec.security.endpointSecretName** で認証シークレットを指定し、変更を適用します。

```
spec:
  security:
    endpointSecretName: connect-secret
```

11.4. カスタムロールおよびパーミッションの追加

カスタムロールは、パーミッションのさまざまな組み合わせで定義できます。

手順

1. **Infinispan** CR を開いて編集します。
2. カスタムロールおよびそれらに関連付けられたパーミッションを **spec.security.authorization.roles** フィールドで指定します。

```
spec:
  security:
    authorization:
      enabled: true
      roles:
        - name: my-role-1
          permissions:
            - ALL
        - name: my-role-2
          permissions:
            - READ
            - WRITE
```

3. 変更を適用します。

第12章 DATA GRID へのネットワークアクセスの設定

Data Grid Console、Data Grid コマンドラインインターフェイス (CLI)、REST API、および Hot Rod エンドポイントにアクセスできるように Data Grid クラスターを公開します。

12.1. 内部接続向けのサービスの取得

デフォルトでは、Data Grid Operator は、OpenShift 上で実行されているクライアントから Data Grid クラスターへのアクセスを提供するサービスを作成します。

この内部サービスの名前は、Data Grid クラスターと同じになります。以下に例を示します。

```
metadata:
  name: infinispan
```

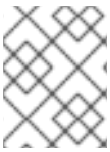
手順

- 内部サービスが以下のように利用できることを確認します。

```
oc get services
```

12.2. LOADBALANCER サービスを使用した DATA GRID の公開

LoadBalancer サービスを使用して、OpenShift 外部で実行されているクライアントで Data Grid クラスターを利用できるようにします。



注記

暗号化されていない Hot Rod クライアント接続で Data Grid にアクセスするには、LoadBalancer サービスを使用する **必要** があります。

手順

1. **Infinispan** CR に **spec.expose** を追加します。
2. **LoadBalancer** を **spec.expose.type** フィールドでサービスタイプとして指定します。
3. オプションで、サービスが **spec.expose.port** フィールドで公開されるネットワークポートを指定します。

```
spec:
  expose:
    type: LoadBalancer
    port: 65535
```

4. 変更を適用します。
5. **-external** サービスが使用できることを確認します。

```
oc get services | grep external
```

12.3. NODEPORT サービスを使用した DATA GRID の公開

NodePort サービスを使用して、ネットワークで Data Grid クラスターを公開します。

手順

1. **Infinispan** CR に **spec.expose** を追加します。
2. **NodePort** を **spec.expose.type** フィールドでサービスタイプとして指定します。
3. Data Grid が **spec.expose.nodePort** フィールドで公開されるポートを定義します。

```
spec:
  expose:
    type: NodePort
    nodePort: 30000
```

4. 変更を適用します。
5. **-external** サービスが使用できることを確認します。

```
oc get services | grep external
```

12.4. ルートを使用した DATA GRID の公開

passthrough 暗号化のある OpenShift **Route** を使用して、Data Grid クラスターをネットワークで利用可能にします。

注記

Hot Rod クライアントを使用して Data Grid にアクセスするには、SNI を使用して TLS を設定する必要があります。

手順

1. **Infinispan** CR に **spec.expose** を追加します。
2. **Route** を **spec.expose.type** フィールドでサービスタイプとして指定します。
3. オプションで、**spec.expose.host** フィールドでホスト名を追加します。

```
spec:
  expose:
    type: Route
    host: www.example.org
```

4. 変更を適用します。
5. ルートが利用可能であることを確認します。

```
oc get routes
```

ルート ポート

ルート を作成すると、ネットワーク接続を受け入れ、ポート **11222** でリッスンする Data Grid サービスにトラフィックをリダイレクトするネットワーク上のポートを公開します。

ルート が利用できるポートは、暗号化を使用するかどうかによって異なります。

ポート	説明
80	暗号化は無効になっています。
443	暗号化が有効になっています。

12.5. ネットワークサービス

Data Grid Operator が作成し、管理するネットワークサービスの参照情報。

Service	ポート	プロトコル	説明
<cluster_name>	**11222**	TCP	OpenShift クラスター内または OpenShift **Route** からの Data Grid エンドポイントへのアクセス。
<cluster_name>-admin	**11223**	TCP	Data Grid Operator が内部で使用する、OpenShift クラスター内の Data Grid エンドポイントへのアクセス。このポートは、ポート 11222 とは異なるセキュリティーレルムを使用します。このポートには、ユーザーアプリケーションからアクセスしないでください。
<cluster_name>-ping	**8888**	TCP	Data Grid Pod のクラスター検出。
<cluster_name>-external	**11222**	TCP	**LoadBalancer** または **NodePort** サービスからの Data Grid エンドポイントへのアクセス。
<cluster_name>-site	**7900**	TCP	クロスサイト通信向けの JGroups RELAY2 チャネル。

注記

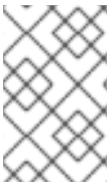
Data Grid コンソールへのアクセスは、OpenShift サービス経由またはポート 11222 を公開する OpenShift **Route** 経由でのみ行ってください。

第13章 クロスサイトレプリケーションの設定

地理的に分散したクラスターを統合サービスとして設定し、Data Grid Operator で可用性を確保します。

以下を使用して、クラスターがクロスサイトレプリケーションを実行するように設定できます。

- Data Grid Operator が管理する接続。
- 設定および管理する接続。



注記

同じ **Infinispan** CR で、Data Grid クラスターの管理接続と手動接続の両方を使用できます。Data Grid クラスターが各サイトで同じ方法を使用して接続を確立できるようにする必要があります。

13.1. クロスサイトレプリケーションの公開タイプ

NodePort サービス、**LoadBalancer** サービス、または OpenShift **Route** を使用して、Data Grid クラスター間のバックアップ操作のネットワークトラフィックを処理できます。クロスサイトレプリケーションの設定を開始する前に、Red Hat OpenShift クラスターで利用可能な公開タイプを判断する必要があります。場合によっては、公開タイプを設定する前に管理者がサービスをプロビジョニングしなければならない場合があります。

NodePort

NodePort は、OpenShift クラスター外で利用可能な IP アドレスの **30000** から **32767** の範囲の静的ポートでネットワークトラフィックを受け入れるサービスです。

NodePort をクロスサイトレプリケーションの公開タイプとして使用するには、管理者は各 OpenShift ノードの外部 IP アドレスをプロビジョニングする必要があります。ほとんどの場合、管理者は OpenShift ノードの外部 IP アドレスの DNS ルーティングを設定する必要もあります。

LoadBalancer

LoadBalancer は、ネットワークトラフィックを OpenShift クラスターの正しいノードに転送するサービスです。

LoadBalancer をクロスサイトレプリケーションの公開タイプとして使用できるかどうかは、ホストプラットフォームによって異なります。AWS はネットワークロードバランサー (NLB) をサポートしますが、他のクラウドプラットフォームには対応しません。**LoadBalancer** サービスを使用するには、管理者はまず NLB がサポートする Ingress コントローラーを作成する必要があります。

ルート

OpenShift **Route** を使用すると、Data Grid クラスターはパブリックセキュア URL を介して相互に接続できます。

Data Grid は、TLS と SNI ヘッダーを使用して、OpenShift **Route** を介してクラスター間でバックアップリクエストを送信します。これを行うには、Data Grid がクロスサイトレプリケーションのためにネットワークトラフィックを暗号化できるように、TLS 証明書を使用してキーストアを追加する必要があります。

クロスサイトレプリケーションの公開タイプとして **Route** を指定すると、Data Grid Operator は、管理する Data Grid クラスターごとに TLS パススルー暗号化を使用してルートを作成します。**Route** のホスト名を指定することはできますが、作成済みの **Route** を指定することはできません。

関連情報

- ingress クラストラフィックの設定の概要

13.2. マネージドのクロスサイトレプリケーション

Data Grid Operator は、異なるデータセンターで実行している Data Grid クラスターを検出し、グローバルクラスターを形成できます。

マネージドのクロスサイト接続を設定する場合には、Data Grid Operator は各 Data Grid クラスターにルーター Pod を作成します。Data Grid Pod は **<cluster_name>-site** サービスを使用してこれらのルーター Pod に接続し、バックアップ要求を送信します。

ルーター Pod はすべての Pod IP アドレスのレコードを保持し、RELAY メッセージヘッダーを解析し、バックアップ要求を正しい Data Grid クラスターに転送します。ルーター Pod がクラッシュすると、OpenShift が復元するまで、Data Grid Pod はいずれも、他に利用可能なルーター Pod を使用し出します。

重要

クロスサイト接続を管理するために、Data Grid Operator は Kubernetes API を使用します。各 OpenShift クラスターに、リモート Kubernetes API へのネットワークアクセスと、各バックアップクラスター用のサービスアカウントトークンが必要です。

注記

Data Grid クラスターは、Data Grid Operator が設定するすバックアップの場所が検出されるまで実行は開始しません。

13.2.1. マネージドクロスサイト接続用のサービスアカウントトークンの作成

Data Grid Operator が Data Grid クラスターを自動的に検出し、クロスサイト接続を管理できるように OpenShift クラスターでサービスアカウントトークンを生成します。

前提条件

- すべての OpenShift クラスターが Kubernetes API にアクセスできることを確認します。Data Grid Operator はこの API を使用してクロスサイト接続を管理します。

注記

Data Grid Operator はリモート Data Grid クラスターを変更しません。サービスアカウントトークンは、Kubernetes API 経由で読み取り専用のアクセスを提供します。

手順

1. OpenShift クラスターにログインします。
2. サービスアカウントを作成します。

たとえば、**LON** でサービスアカウントを作成します。

```
oc create sa -n <namespace> lon
```

3. 以下のコマンドを使用して、ビューロールをサービスアカウントに追加します。

```
oc policy add-role-to-user view -n <namespace> -z lon
```

4. **NodePort** サービスを使用してネットワーク上で Data Grid クラスターを公開する場合、**cluster-reader** ロールをサービスアカウントに追加する必要もあります。

```
oc adm policy add-cluster-role-to-user cluster-reader -z lon -n <namespace>
```

5. 他の OpenShift クラスターで直前の手順を繰り返します。
6. 各 OpenShift クラスターでサービスアカウントトークンを交換します。

13.2.2. サービスアカウントトークンの交換

OpenShift クラスターでサービスアカウントトークンを生成し、各バックアップ場所のシークレットに追加します。この手順で生成するトークンには有効期限がありません。バインドされたサービスアカウントトークンについては、バインドされたサービスアカウントトークンの交換 を参照してください。

前提条件

- サービスアカウントを作成している。

手順

1. OpenShift クラスターにログインします。
2. 次のようにサービスアカウントトークンのシークレットファイルを作成します。

sa-token.yaml

```
apiVersion: v1
kind: Secret
metadata:
  name: ispn-xsite-sa-token (1)
  annotations:
    kubernetes.io/service-account.name: "<service-account>" (2)
type: kubernetes.io/service-account-token
```

(1) シークレットの名前を指定します。

(2) サービスアカウント名を指定します。

3. OpenShift クラスターにシークレットを作成します。

```
oc -n <namespace> create -f sa-token.yaml
```

4. サービスアカウントトークンを取得します。

```
oc -n <namespace> get secrets ispn-xsite-sa-token -o jsonpath="{.data.token}" | base64 -d
```

このコマンドにより、ターミナルにトークンが出力されます。

5. バックアップ OpenShift クラスターにデプロイメント用のトークンをコピーします。
6. バックアップ OpenShift クラスターにログインします。
7. バックアップの場所用のサービスアカウントトークンを追加します。

```
oc -n <namespace> create secret generic <token-secret> --from-literal=token=<token>
```

<token-secret> は、**Infinispan** CR で設定したシークレットの名前です。

次のステップ

- 他の OpenShift クラスターで直前の手順を繰り返します。

関連情報

- サービスアカウントトークンシークレットの作成

13.2.3. バインドされたサービスアカウントトークンの交換

有効期間が制限されたサービスアカウントトークンを作成し、各バックアップ場所のシークレットに追加します。Data Grid Operator がリモート OpenShift クラスターにアクセスできなくなるのを防ぐために、トークンを定期的に更新する必要があります。有効期限のないトークンについては、サービスアカウントトークンの交換 を参照してください。

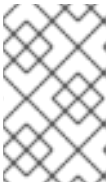
前提条件

- サービスアカウントを作成している。

手順

1. OpenShift クラスターにログインします。
2. サービスアカウント用のバインドされたトークンを作成します。

```
oc -n <namespace> create token <service-account>
```



注記

デフォルトでは、サービスアカウントトークンの有効期間は 1 時間です。有効期間を秒単位で指定するには、コマンドオプション **--duration** を使用してください。

このコマンドにより、ターミナルにトークンが出力されます。

3. バックアップ OpenShift クラスターにデプロイメント用のトークンをコピーします。
4. バックアップ OpenShift クラスターにログインします。

5. バックアップの場所用のサービスアカウントトークンを追加します。

```
oc -n <namespace> create secret generic <token-secret> --from-literal=token=<token>
```

<token-secret> は、**Infinispan** CR で設定したシークレットの名前です。

6. 他の OpenShift クラスターでもこの手順を繰り返します。

期限切れのトークンの削除
トークンの有効期限が切れた場合は、期限切れのトークンシークレットを削除し、新しいトークンシークレットを生成して交換する手順を再度実行します。

1. バックアップ OpenShift クラスターにログインします。
2. 期限切れのシークレット **<token-secret>** を削除します。

```
oc -n <namespace> delete secrets <token-secret>
```

3. 手順を再度実行して新しいトークンを作成し、新しい **<token-secret>** を生成します。

関連情報

- バインドされたサービスアカウントトークンの作成

13.2.4. マネージドのクロスサイト接続の設定

Data Grid クラスターでクロスサイトビューを確立するように Data Grid Operator を設定します。

前提条件

- Determine a suitable expose type for cross-site replication.
  OpenShift **Route** を使用する場合は、TLS 証明書と安全なクロスサイト接続を備えたキーストアを追加する必要があります。
- 各 Data Grid クラスターに Red Hat OpenShift サービスアカウントトークンを作成して交換します。

手順

1. 各 Data Grid クラスターに **Infinispan** CR を作成します。
2. ローカルサイトの名前を **spec.service.sites.local.name** で指定します。
3. クロスサイトレプリケーションの公開タイプを設定します。
   a. **spec.service.sites.local.expose.type** フィールドの値を次のいずれかに設定します。
      - **NodePort**
      - **LoadBalancer**
      - **ルート**
   b. オプションで、次のフィールドを使用してポートまたはカスタムホスト名を指定します。
      - **NodePort** サービスを使用する場合の **spec.service.sites.local.expose.nodePort**。

- **LoadBalancer** サービスを使用する場合の **spec.service.sites.local.expose.port**。
- OpenShift **Route** を使用する場合は、**spec.service.sites.local.expose.routeHostName** を使用します。

4. RELAY メッセージを **service.sites.local.maxRelayNodes** フィールドで送信できる Pod の数を指定します。

ヒント

パフォーマンス向上のため **RELAY** メッセージを送信するようにクラスター内のすべての Pod を設定します。すべての Pod がバックアップ要求を直接送信する場合には、バックアップ要求を転送する必要はありません。

5. **spec.service.sites.locations** でバックアップの場所として動作する各 Data Grid クラスターの名前、URL、およびシークレットを指定します。
6. リモートサイトの Data Grid クラスター名または namespace がローカルサイトに一致しない場合は、それらの値を **clusterName** および **namespace** フィールドで指定します。
以下は、**LON** および **NYC** の **Infinispan** CR 定義の例になります。

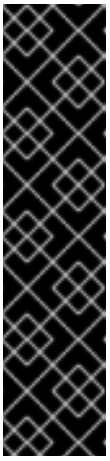
- LON

```
apiVersion: infinispan.org/v1
kind: Infinispan
metadata:
  name: infinispan
spec:
  replicas: 3
  version: <Data Grid_version>
  service:
    type: DataGrid
    sites:
      local:
        name: LON
        expose:
          type: LoadBalancer
          port: 65535
        maxRelayNodes: 1
      locations:
        - name: NYC
          clusterName: <nyc_cluster_name>
          namespace: <nyc_cluster_namespace>
          url: openshift://api.rhdg-nyc.openshift-aws.myhost.com:6443
          secretName: nyc-token
  logging:
    categories:
      org.jgroups.protocols.TCP: error
      org.jgroups.protocols.relay.RELAY2: error
```

- NYC

```
apiVersion: infinispan.org/v1
kind: Infinispan
metadata:
```

```
  name: nyc-cluster
spec:
  replicas: 2
  version: <Data Grid_version>
  service:
    type: DataGrid
    sites:
      local:
        name: NYC
        expose:
          type: LoadBalancer
          port: 65535
        maxRelayNodes: 1
      locations:
        - name: LON
          clusterName: infinispan
          namespace: rhdg-namespace
          url: openshift://api.rhdg-lon.openshift-aws.myhost.com:6443
          secretName: lon-token
  logging:
    categories:
      org.jgroups.protocols.TCP: error
      org.jgroups.protocols.relay.RELAY2: error
```



重要

JGroups TCP および RELAY2 プロトコルのログレベルを下げるために、**Infinispan** CR のロギングカテゴリーを調整してください。これにより、多数のログファイルがコンテナーストレージを使用することを防ぎます。

```
spec:
  logging:
    categories:
      org.jgroups.protocols.TCP: error
      org.jgroups.protocols.relay.RELAY2: error
```

7. 他の Data Grid サービスリソースで **Infinispan** CR を設定してから、変更を適用します。
8. Data Grid クラスターがクロスサイトビューを形成することを確認します。
   a. **Infinispan** CR を取得します。

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

   b. **type: CrossSiteViewFormed** 条件を確認します。

次のステップ

クラスターがクロスサイトビューを形成している場合は、バックアップの場所をキャッシュに追加し始めることができます。

関連情報

- クロスサイトレプリケーションの Data Grid ガイド

13.3. クロスサイト接続の手動設定

静的ネットワーク接続の詳細を指定して、OpenShift の外部で実行される Data Grid クラスターでクロスサイトレプリケーションを実行できます。Data Grid が実行される OpenShift クラスターの外部で Kubernetes API にアクセスできないというシナリオでは、手動でのクロスサイト接続が必要です。

前提条件

- Determine a suitable expose type for cross-site replication.
  OpenShift **Route** を使用する場合は、TLS 証明書と安全なクロスサイト接続を備えたキーストアを追加する必要があります。
- 各 Data Grid クラスターおよび各 **<cluster-name>-site** サービスの正しいホスト名とポートがあることを確認します。
  Data Grid クラスターを手動で接続してクロスサイトビューを形成するには、Data Grid サービスで予測可能なネットワークの場所が必要です。つまり、作成前にネットワークの場所を把握しておく必要があります。

手順

1. 各 Data Grid クラスターに **Infinispan** CR を作成します。
2. ローカルサイトの名前を **spec.service.sites.local.name** で指定します。
3. クロスサイトレプリケーションの公開タイプを設定します。
   a. **spec.service.sites.local.expose.type** フィールドの値を次のいずれかに設定します。
      - **NodePort**
      - **LoadBalancer**
      - **ルート**
   b. オプションで、次のフィールドを使用してポートまたはカスタムホスト名を指定します。
      - **NodePort** サービスを使用する場合の **spec.service.sites.local.expose.nodePort**。
      - **LoadBalancer** サービスを使用する場合の **spec.service.sites.local.expose.port**。
      - OpenShift **Route** を使用する場合は、**spec.service.sites.local.expose.routeHostName** を使用します。
4. **spec.service.sites.locations** でバックアップの場所として動作する各 Data Grid クラスターの名前と静的 URL を指定します。以下に例を示します。
   - LON

```
apiVersion: infinispan.org/v1
kind: Infinispan
metadata:
  name: infinispan
spec:
  replicas: 3
```

```
  version: <Data Grid_version>
  service:
    type: DataGrid
    sites:
      local:
        name: LON
        expose:
          type: LoadBalancer
          port: 65535
        maxRelayNodes: 1
      locations:
        - name: NYC
          url: infinispan+xsite://infinispan-nyc.myhost.com:7900
  logging:
    categories:
      org.jgroups.protocols.TCP: error
      org.jgroups.protocols.relay.RELAY2: error
```

- NYC

```
apiVersion: infinispan.org/v1
kind: Infinispan
metadata:
  name: infinispan
spec:
  replicas: 2
  version: <Data Grid_version>
  service:
    type: DataGrid
    sites:
      local:
        name: NYC
        expose:
          type: LoadBalancer
          port: 65535
        maxRelayNodes: 1
      locations:
        - name: LON
          url: infinispan+xsite://infinispan-lon.myhost.com
  logging:
    categories:
      org.jgroups.protocols.TCP: error
      org.jgroups.protocols.relay.RELAY2: error
```

重要

JGroups TCP および RELAY2 プロトコルのログレベルを下げるために、**Infinispan** CR のロギングカテゴリーを調整してください。これにより、多数のログファイルがコンテナーストレージを使用することを防ぎます。

```
spec:
  logging:
    categories:
      org.jgroups.protocols.TCP: error
      org.jgroups.protocols.relay.RELAY2: error
```

5. 他の Data Grid サービスリソースで **Infinispan** CR を設定してから、変更を適用します。
6. Data Grid クラスターがクロスサイトビューを形成することを確認します。
   a. **Infinispan** CR を取得します。

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

   b. **type: CrossSiteViewFormed** 条件を確認します。

次のステップ

クラスターがクロスサイトビューを形成している場合は、バックアップの場所をキャッシュに追加し始めることができます。

関連情報

- クロスサイトレプリケーションの Data Grid ガイド

13.4. GOSSIP ルーター POD への CPU とメモリーの割り当て

CPU とメモリーリソースを Data Grid Gossip ルーターに割り当てます。

前提条件

- Gossip ルーターが有効である。**service.sites.local.discovery.launchGossipRouter** プロパティーは、デフォルト値である **true** に設定する必要があります。

手順

1. **service.sites.local.discovery.cpu** フィールドを使用して CPU ユニットの数を割り当てます。
2. **service.sites.local.discovery.memory** フィールドを使用して、メモリー量をバイト単位で割り当てます。
   cpu および **memory** フィールドでは **<limit>:<requests>** 形式で値を指定します。たとえば、**cpu: "2000m:1000m"** は Pod を最大 **2000m** の CPU に制限し、起動時に各 Pod に対して **1000m** の CPU を要求します。値を 1 つ指定すると、制限と要求の両方が設定されます。
3. **Infinispan** CR を適用します。

```
spec:
```

```
service:
  type: DataGrid
  sites:
    local:
      name: LON
      discovery:
        launchGossipRouter: true
        memory: "2Gi:1Gi"
        cpu: "2000m:1000m"
```

13.5. ローカル GOSSIP ルーターとサービスの無効化

Data Grid Operator は各サイトで Gossip ルーターを開始しますが、Data Grid クラスターメンバー間のトラフィックを管理するために必要な Gossip ルーターは 1 つだけです。リソースを節約するために、追加の Gossip ルーターを無効にすることができます。

たとえば、**LON** サイトと **NYC** サイトに Data Grid クラスターがあるとします。次の手順は、**LON** サイトで Gossip ルーターを無効にし、Gossip ルーターが有効になっている **NYC** に接続する方法を示しています。

手順

1. 各 Data Grid クラスターに **Infinispan** CR を作成します。
2. **spec.service.sites.local.name** フィールドでローカルサイトの名前を指定します。
3. **LON** クラスターの場合、**spec.service.sites.local.discovery.launchGossipRouter** フィールドの値として **false** を設定します。
4. **LON** クラスターの場合は、**spec.service.sites.locations.url** で **URL** を指定して **NYC** に接続します。
5. **NYC** 設定では、**spec.service.sites.locations.url** を指定しないでください。

LON

```
apiVersion: infinispan.org/v1
kind: Infinispan
metadata:
  name: infinispan
spec:
  replicas: 3
  service:
    type: DataGrid
    sites:
      local:
        name: LON
        discovery:
          launchGossipRouter: false
      locations:
        - name: NYC
          url: infinispan+xsite://infinispan-nyc.myhost.com:7900
```

NYC

```
apiVersion: infinispan.org/v1
kind: Infinispan
metadata:
  name: infinispan
spec:
  replicas: 3
  service:
    type: DataGrid
    sites:
      local:
        name: NYC
      locations:
        - name: LON
```

重要

3 つ以上のサイトがある場合、Data Grid では、すべてのリモートサイトで Gossip ルーターを有効にしておくことを推奨します。複数の Gossip ルーターがあり、そのうちの 1 つが使用できなくなった場合、残りのルーターは引き続きメッセージを交換します。単一の Gossip ルーターが定義されていて、それが使用できなくなると、リモートサイト間の接続が切断されます。

次のステップ

クラスターがクロスサイトビューを形成している場合は、バックアップの場所をキャッシュに追加し始めることができます。

関連情報

- Data Grid クロスサイトレプリケーション

13.6. クロスサイトレプリケーションを設定するためのリソース

以下の表は、クロスサイトリソースのフィールドと説明を示しています。

表13.1 service.type

フィールド	説明
service.type: DataGrid	Data Grid は、Data Grid サービスクラスターのみでクロスサイトレプリケーションをサポートします。

表13.2 service.sites.local

フィールド	説明
service.sites.local.name	Data Grid クラスターが実行されるローカルサイトに名前を付けます。
service.sites.local.maxRelayNodes	クロスサイトレプリケーション用に RELAY メッセージを送信できる Pod の最大数を指定します。デフォルト値は **1** です。

フィールド	説明
service.sites.local.discovery.launchGossipRouter	**false** の場合、クロスサイトサービスと Gossip ルーター Pod はローカルサイトに作成されません。デフォルト値は **true** です。
service.sites.local.discovery.memory	メモリー量をバイト単位で割り当てます。**<limit>:<requests>** という形式を使用します (例 **"2Gi:1Gi"**)。
service.sites.local.discovery.cpu	CPU ユニットの数を割り当てます。**<limit>:<requests>** という形式を使用します (例 **"2000m:1000m"**)。
service.sites.local.expose.type	クロスサイトレプリケーションのネットワークサービスを指定します。Data Grid クラスターは、このサービスを使用して通信し、バックアップ操作を実行します。値は、**NodePort**、**LoadBalancer**、または **Route** に設定できます。
service.sites.local.expose.nodePort	**NodePort** サービス経由で Data Grid を公開する場合、デフォルトの範囲内の **30000** から **32767** の静的ポートを指定します。ポートを指定しないと、プラットフォームは利用可能なポートを選択します。
service.sites.local.expose.port	**LoadBalancer** サービス経由で Data Grid を公開する場合は、サービスのネットワークポートを指定します。デフォルトのポートは **7900** です。
service.sites.local.expose.routeHostName	OpenShift **Route** を介して Data Grid を公開する場合は、カスタムホスト名を指定します。値を設定しない場合、OpenShift はホスト名を生成します。

表13.3 service.sites.locations

フィールド	説明
service.sites.locations	すべてのバックアップの場所の接続情報を提供します。
service.sites.locations.name	**.spec.service.sites.local.name** に一致するバックアップの場所を指定します。

フィールド	説明
service.sites.locations.url	マネージド接続の Kubernetes API の URL または手動接続の静的 URL を指定します。 **openshift://** を使用して、OpenShift クラスターの Kubernetes API の URL を指定します。 **openshift://** URL は有効な CA 署名の証明書を指定する必要があることに注意してください。自己署名証明書は使用できません。 静的ホスト名とポートに **infinispan+xsite://<hostname>:<port>** 形式を使用します。デフォルトのポートは **7900** です。
service.sites.locations.secretName	バックアップサイトのサービスアカウントトークンが含まれるシークレットを指定します。
service.sites.locations.clusterName	ローカルサイトのクラスター名と異なる場合は、バックアップの場所でクラスター名を指定します。
service.sites.locations.namespace	ローカルサイトの namespace に一致しない場合は、バックアップの場所にある Data Grid クラスターの namespace を指定します。

マネージドのクロスサイト接続

```
spec:
  service:
    type: DataGrid
    sites:
      local:
        name: LON
        expose:
          type: LoadBalancer
        maxRelayNodes: 1
      locations:
      - name: NYC
        clusterName: <nyc_cluster_name>
        namespace: <nyc_cluster_namespace>
        url: openshift://api.site-b.devcluster.openshift.com:6443
        secretName: nyc-token
```

手動によるクロスサイト接続

```
spec:
  service:
    type: DataGrid
    sites:
```

```
  local:
    name: LON
    expose:
      type: LoadBalancer
      port: 65535
    maxRelayNodes: 1
  locations:
  - name: NYC
    url: infinispan+xsite://infinispan-nyc.myhost.com:7900
```

13.7. クロスサイト接続のセキュリティー保護

キーストアとトラストストアを追加して、Data Grid クラスターがクロスサイトレプリケーショントラフィックを保護できるようにします。

クロスサイトレプリケーションの公開タイプとして OpenShift **Route** を使用するには、キーストアを追加する必要があります。公開タイプとして **NodePort** または **LoadBalancer** を使用する場合、クロスサイト接続の保護はオプションです。

注記

クロスサイトレプリケーションは、OpenShift CA サービスをサポートしていません。独自の証明書を指定する必要があります。

前提条件

- Data Grid が使用して RELAY メッセージの暗号化と復号化ができる PKCS12 キーストアを用意する。
  クロスサイト接続のセキュリティーを保護するには、Pod およびルーター Pod をリレーするためのキーストアを提供する必要があります。
  リレー Pod とルーター Pod で同じキーストアにすることも、それぞれに個別のキーストアを提供することもできます。
  各 Data Grid クラスターに同じキーストアを使用するか、各クラスターに一意のキーストアを使用することもできます。

- Data Grid リレー Pod とルーター Pod の公開証明書を検証する証明書チェーンまたはルート CA 証明書の一部を含む PKCS12 トラストストアがある。

手順

1. クロスサイト暗号化シークレットを作成します。

   a. キーストアシークレットを作成します。

   b. トラストストアシークレットを作成します。

2. 各 Data Grid クラスターの **Infinispan** CR を変更し、**encryption.transportKeyStore.secretName** および **encryption.routerKeyStore.secretName** フィールドのシークレット名を指定します。

3. 必要に応じて RELAY メッセージを暗号化するように他のフィールドを設定して変更を適用します。

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

```
metadata:
  name: infinispan
spec:
  replicas: 2
  version: <Data Grid_version>
  expose:
    type: LoadBalancer
  service:
    type: DataGrid
    sites:
      local:
        name: SiteA
        # ...
        encryption:
          protocol: TLSv1.3
          transportKeyStore:
            secretName: transport-tls-secret
            alias: transport
            filename: keystore.p12
          routerKeyStore:
            secretName: router-tls-secret
            alias: router
            filename: keystore.p12
          trustStore:
            secretName: truststore-tls-secret
            filename: truststore.p12
      locations:
        # ...
```

13.7.1. サイト間の暗号化を設定するためのリソース

以下の表は、クロスサイト接続を暗号化するフィールドおよび説明を示しています。

表13.4 service.type.sites.local.encryption

フィールド	説明
service.type.sites.local.encryption.protocol	クロスサイト接続に使用する TLS プロトコルを指定します。デフォルト値は **TLSv1.2** ですが、必要に応じて **TLSv1.3** を設定できます。
service.type.sites.local.encryption.transportKeyStore	リレー Pod のキーストアシークレットを設定します。
service.type.sites.local.encryption.routerKeyStore	ルーター Pod のキーストアシークレットを設定します。
service.type.sites.local.encryption.trustStore	リレー Pod とルーター Pod のトラストストアシークレットを設定します。

表13.5 service.type.sites.local.encryption.transportKeyStore

フィールド	説明
secretName	リレー Pod が RELAY メッセージの暗号化および復号化に使用できるキーストアを含むシークレットを指定します。このフィールドは必須です。
alias	オプションで、キーストアの証明書のエイリアスを指定します。デフォルト値は **transport** です。
filename	オプションでキーストアのファイル名を指定します。デフォルト値は **keystore.p12** です。

表13.6 service.type.sites.local.encryption.routerKeyStore

フィールド	説明
secretName	ルーター Pod が RELAY メッセージの暗号化および復号化に使用できるキーストアを含むシークレットを指定します。このフィールドは必須です。
alias	オプションで、キーストアの証明書のエイリアスを指定します。デフォルト値は **router** です。
filename	オプションでキーストアのファイル名を指定します。デフォルト値は **keystore.p12** です。

表13.7 service.type.sites.local.encryption.trustStore

フィールド	説明
secretName	リレー Pod とルーター Pod の公開証明書を検証するためのトラストストアを含むシークレットを指定します。このフィールドは必須です。
filename	オプションでトラストストアのファイル名を指定します。デフォルト値は **truststore.p12** です。

13.7.2. クロスサイト暗号化シークレット

クロスサイトレプリケーション暗号化シークレットは、クロスサイト接続を保護するためのキーストアとトラストストアを追加します。

クロスサイト暗号化シークレット

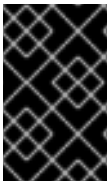
```
apiVersion: v1
kind: Secret
metadata:
  name: tls-secret
type: Opaque
```

```
stringData:
  password: changeme
  type: pkcs12
data:
  <file-name>: "MIIKDgIBAzCCCdQGCSqGSIb3DQEHA..."
```

フィールド	説明
stringData.password	キーストアまたはトラストストアのパスワードを指定します。
stringData.type	オプションでキーストアまたはトラストストアタイプを指定します。デフォルト値は **pkcs12** です。
data.<file-name>	base64 でエンコードされたキーストアまたはトラストストアを追加します。

13.8. 同じ OPENSHIFT クラスターでのサイトの設定

評価およびデモの目的で、同じ OpenShift クラスター内の Pod 間でバックアップするように Data Grid を設定できます。



重要

ClusterIP をクロスサイトレプリケーションの公開タイプとしての使用は、デモのみを目的としています。この公開タイプのみを使用して、ラップトップまたはその性質のある一時的な概念実証のデプロイメントを実行することが推奨されます。

手順

1. 各 Data Grid クラスターに **Infinispan** CR を作成します。
2. ローカルサイトの名前を **spec.service.sites.local.name** で指定します。
3. **ClusterIP** を **spec.service.sites.local.expose.type** フィールドの値として設定します。
4. **spec.service.sites.locations.clusterName** でバックアップの場所として動作する Data Grid クラスターの名前を指定します。
5. 両方の Data Grid クラスターの名前が同じである場合は、**spec.service.sites.locations.namespace** でバックアップの場所の namespace を指定します。

```
apiVersion: infinispan.org/v1
kind: Infinispan
metadata:
  name: example-clustera
spec:
  replicas: 1
  expose:
    type: LoadBalancer
  service:
```

```
    type: DataGrid
    sites:
      local:
        name: SiteA
        expose:
          type: ClusterIP
        maxRelayNodes: 1
      locations:
        - name: SiteB
          clusterName: example-clusterb
          namespace: cluster-namespace
```

6. 他の Data Grid サービスリソースで **Infinispan** CR を設定してから、変更を適用します。
7. Data Grid クラスターがクロスサイトビューを形成することを確認します。
   a. **Infinispan** CR を取得します。

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

   b. **type: CrossSiteViewFormed** 条件を確認します。

第14章 DATA GRID サービスの監視

Data Grid は、クラスターの状態を監視および視覚化するために Prometheus および Grafana が使用できるメトリックを公開します。

注記

本書では、OpenShift Container Platform でモニタリングを設定する方法について説明します。コミュニティー Prometheus デプロイメントを使用している場合は、これらの手順は一般的なガイドとして役に立ちます。ただし、インストールおよび使用方法については、Prometheus のドキュメントを参照してください。

Prometheus Operator のドキュメントを参照してください。

14.1. PROMETHEUS サービスモニターの作成

Data Grid Operator は、Data Grid クラスターからメトリックをスクレープする Prometheus **ServiceMonitor** を自動的に作成します。

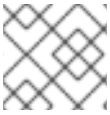
手順

OpenShift Container Platform で、ユーザー定義プロジェクトのモニタリングを有効にします。

Operator がモニタリングアノテーションが **true** に設定されている **Infinispan** CR を検出すると、Data Grid Operator は以下を行います。

- **<cluster_name>-monitor** という名前の **ServiceMonitor** を作成します。
- 値がまだ明示的に設定されていない場合は、**infinispan.org/monitoring: 'true'** アノテーションを **Infinispan** CR メタデータに追加します。

```
apiVersion: infinispan.org/v1
kind: Infinispan
metadata:
  name: infinispan
  annotations:
    infinispan.org/monitoring: 'true'
```



注記

Data Grid で認証するために、Prometheus は **Operator** の認証情報を使用します。

検証

Prometheus が Data Grid メトリックを以下のようにスクレープすることを確認できます。

1. OpenShift Web コンソールで、**</> Developer** パースペクティブを選択してから、**Monitoring** を選択します。
2. Data Grid クラスターが実行される namespace の **Dashboard** タブを開きます。
3. **Metrics** タブを開き、以下のような Data Grid メトリクスをクエリーできることを確認します。

```
vendor_cache_manager_default_cluster_size
```

関連情報

- Enabling monitoring for user-defined projects

14.1.1. Prometheus サービスモニターの無効化

Prometheus が Data Grid クラスターのメトリックをスクレープしない場合は、**ServiceMonitor** を無効にできます。

手順

1. **'false'** を **Infinispan** CR の **infinispan.org/monitoring** アノテーションの値として設定します。

```
apiVersion: infinispan.org/v1
kind: Infinispan
metadata:
  name: infinispan
  annotations:
    infinispan.org/monitoring: 'false'
```

2. 変更を適用します。

14.1.2. サービスモニターターゲットラベルの設定

ServiceMonitor の **spec.targetLabels** フィールドを使用して、基礎となるメトリクスにサービスラベルを伝播するように、生成された **ServiceMonitor** を設定できます。サービスラベルを使用して、監視対象のエンドポイントから収集されたメトリクスをフィルタリングおよび集計します。

手順

1. **Infinispan** CR で **infinispan.org/targetLabels** アノテーションを設定して、サービスに適用するラベルを定義します。
2. **Infinispan** CR の **infinispan.org/serviceMonitorTargetLabels** アノテーションを使用して、メトリクスに必要なラベルのコンマ区切りのリストを指定します。

```
apiVersion: infinispan.org/v1
kind: Infinispan
metadata:
  name: infinispan
  annotations:
    infinispan.org/targetLabels: "label1,label2,label3"
    infinispan.org/serviceMonitorTargetLabels: "label1,label2"
```

3. 変更を適用します。

14.2. GRAFANA OPERATOR のインストール

各種ニーズに対応するために、Data Grid Operator はコミュニティーバージョンの Grafana Operator と統合し、Data Grid サービスのダッシュボードを作成します。

Grafana が OpenShift ユーザーワークロードのモニタリングと統合されるまでの唯一のオプションは、コミュニティーバージョンに依存することです。**OperatorHub** から OpenShift に Grafana Operator をインストールすることができます。また、**alpha** チャネルのサブスクリプションを作成する必要があり

ます。

ただし、すべてのコミュニティー Operator のポリシーと同様に、Red Hat は Grafana Operator を認定しておらず、Data Grid との組み合わせに対するサポートを提供していません。Grafana Operator をインストールすると、続行する前にコミュニティーバージョンに関する警告を確認するように求められます。

14.3. GRAFANA データソースの作成

Grafana ダッシュボードで Data Grid メトリックを視覚化できるように **GrafanaDatasource** CR を作成します。

前提条件

- **oc** クライアントがある。
- OpenShift Container Platform への **cluster-admin** アクセスがあること。
- OpenShift Container Platform で、ユーザー定義プロジェクトのモニタリングを有効にします。
- **alpha** チャネルから Grafana Operator をインストールし、**Grafana** CR を作成します。

手順

1. Grafana が Prometheus から Data Grid メトリックを読み取りできるようにする **ServiceAccount** を作成します。

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: infinispan-monitoring
```

   a. **ServiceAccount** を適用します。

```
oc apply -f service-account.yaml
```

   b. **cluster-monitoring-view** パーミッションを **ServiceAccount** に付与します。

```
oc adm policy add-cluster-role-to-user cluster-monitoring-view -z infinispan-monitoring
```

2. Grafana データソースを作成します。

   a. **ServiceAccount** のトークンを取得します。

```
oc serviceaccounts get-token infinispan-monitoring
```

   b. 以下の例のように、**spec.datasources.secureJsonData.httpHeaderValue1** フィールドにトークンが含まれる **GrafanaDataSource** を定義します。

```
apiVersion: integreatly.org/v1alpha1
kind: GrafanaDataSource
metadata:
  name: grafanadatasource
```

```
spec:
  name: datasource.yaml
  datasources:
    - access: proxy
      editable: true
      isDefault: true
      jsonData:
        httpHeaderName1: Authorization
        timeInterval: 5s
        tlsSkipVerify: true
      name: Prometheus
      secureJsonData:
        httpHeaderValue1: >-
          Bearer
          eyJhbGciOiJSUzI1NiIsImtpZCI6Imc4O...
      type: prometheus
      url: 'https://thanos-querier.openshift-monitoring.svc.cluster.local:9091'
```

3. **GrafanaDataSource** を適用します。

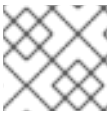
```
oc apply -f grafana-datasource.yaml
```

次のステップ

Grafana ダッシュボードを Data Grid Operator 設定プロパティーで有効にします。

14.4. DATA GRID ダッシュボードの設定

Data Grid Operator は、Data Grid クラスターの Grafana ダッシュボードを設定できるようにするグローバル設定プロパティーを提供します。



注記

Data Grid Operator の実行中にグローバル設定プロパティーを変更できます。

前提条件

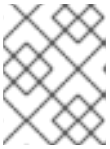
- Data Grid Operator は、Grafana Operator が実行されている namespace を監視する必要があります。

手順

1. Data Grid Operator namespace に **infinispan-operator-config** という名前の **ConfigMap** を作成します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: infinispan-operator-config
data:
  grafana.dashboard.namespace: infinispan
  grafana.dashboard.name: infinispan
  grafana.dashboard.monitoring.key: middleware
```

2. Data Grid クラスターの namespace を **data.grafana.dashboard.namespace** プロパティーで指定します。



注記

このプロパティーの値を削除すると、ダッシュボードが削除されます。値を変更すると、Dashboard はその namespace に移動します。

3. **data.grafana.dashboard.name** プロパティーでダッシュボードの名前を指定します。

4. 必要な場合は、モニタリングキーを **data.grafana.dashboard.monitoring.key** プロパティーで指定します。

5. **infinispan-operator-config** を作成するか、設定を更新します。

```
oc apply -f infinispan-operator-config.yaml
```

6. 以下で入手可能な Grafana UI を開きます。

```
oc get routes grafana-route -o jsonpath=https://"{.spec.host}"
```

14.5. DATA GRID クラスターに対する JMX リモートポートの有効化

JMX リモートポートを有効にして、Data Grid MBean を公開し、Data Grid を Cryostat などの外部監視システムと統合します。

Data Grid クラスターに対して JMX を有効にすると、以下が発生します。

1. 各 Data Grid Server Pod が、Operator ユーザー認証情報を含む "admin" セキュリティーレルムを利用して、認証された JMX エンドポイントをポート **9999** で公開します。

2. **<cluster-name>-admin** サービスがポート **9999** を公開します。

注記

JMX を有効または無効にできるのは、**Infinispan** CR の作成中のみです。CR インスタンスが作成されると、JMX 設定を変更することはできません。

手順

1. **Infinispan** CR で JMX を有効にします。

```
apiVersion: infinispan.org/v1
kind: Infinispan
metadata:
  name: infinispan
spec:
  jmx:
    enabled: true
```

2. Operator ユーザー認証情報を取得して、クライアント JMX 接続を認証します。

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

関連情報

- JMX 統計の有効化

14.6. CRYOSTAT を使用した JFR レコーディングのセットアップ

OpenShift 上で実行される Data Grid クラスターの JDK Flight Recorder (JFR) モニタリングを有効にします。

Cryostat を使用した JFR レコーディング

JFR は、JVM のパフォーマンスのさまざまな側面に関する洞察を提供し、クラスターの検査とデバッグを容易にします。要件に応じて、Cryostat が提供する統合ツールを使用してレコーディングを保存および分析したり、レコーディングを外部監視アプリケーションにエクスポートしたりできます。

前提条件

- Cryostat Operator をインストールしている。Operator Lifecycle Manager (OLM) を使用して、OpenShift プロジェクトに Cryostat Operator をインストールできます。
- Data Grid クラスターで JMX を有効にしている。クラスターをデプロイした後は JMX 設定を変更できないため、デプロイする前に JMX を有効にする必要があります。

手順

1. **Infinispan** CR と同じ名前空間に Cryostat CR を作成します。

```
apiVersion: operator.cryostat.io/v1beta1
kind: Cryostat
metadata:
  name: cryostat-sample
spec:
  minimal: false
  enableCertManager: true
```

注記

Cryostat Operator では、トラフィック暗号化のために cert-manager が必要です。cert-manager が有効になっていてもインストールされていない場合、デプロイメントは失敗します。詳細は、Cryostat のインストール を参照してください。

2. **Cryostat** CR の準備が完了するまで待機します。

```
oc wait -n <namespace> --for=condition=MainDeploymentAvailable cryostat/cryostat-sample
```

3. Cryostat **status.applicationUrl** を開きます。

```
oc -n <namespace> get cryostat cryostat-sample
```

4. Operator ユーザー認証情報を取得して、Cryostat UI でクライアント JMX 接続を認証します。

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

5. Cryostat UI で、**Security** メニューに移動します。
6. **Store Credentials** ウィンドウで、**Add** ボタンをクリックします。**Store Credentials** ウィンドウが開きます。
7. **Match Expression** フィールドに、一致式の詳細を次の形式で入力します。

```
target.labels['infinispan_cr'] == '<cluster_name>'
```

関連情報

- Cryostat のインストール
- Cryostat 認証情報の設定
- Data Grid クラスターに対する JMX リモートポートの有効化

第15章 ANTI-AFFINITY による可用性の保証

Kubernetes には、単一障害点からワークロードを保護する anti-affinity 機能が含まれます。

15.1. ANTI-AFFINITY ストラテジー

クラスターの各 Data Grid ノードは、クラスターの OpenShift ノードで実行される Pod で実行されます。各 Red Hat OpenShift ノードは、物理ホストシステムで実行されます。anti-affinity は、OpenShift ノード全体に Data Grid ノードを分散することで機能し、ハードウェア障害が発生した場合でも、Data Grid クラスターを引き続き使用できるようにします。

Data Grid Operator は、2 つの anti-affinity ストラテジーを提供します。

kubernetes.io/hostname

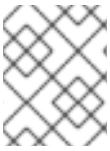
Data Grid レプリカ Pod は、さまざまな OpenShift ノードでスケジュールされます。

topology.kubernetes.io/zone

Data Grid レプリカ Pod は、複数のゾーンにまたがってスケジュールされます。

フォールトトレランス

anti-affinity ストラテジーは、さまざまな方法でクラスターの可用性を保証します。



注記

以下のセクションの式は、OpenShift ノードまたはゾーンの数が Data Grid ノードの数よりも大きい場合にのみ適用されます。

さまざまな OpenShift ノードでの Pod のスケジュール

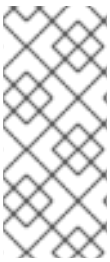
以下のタイプのキャッシュに対して、**x** ノードの障害に対する耐性を提供します。

- Replicated: **x = spec.replicas - 1**
- Distributed: **x = num_owners - 1**

複数ゾーンにまたがる Pod のスケジューリング

以下のタイプのキャッシュに対して **x** ゾーンが存在する場合、**x** ゾーンの障害に対する耐性を提供します。

- Replicated: **x = spec.replicas - 1**
- Distributed: **x = num_owners - 1**



注記

spec.replicas

各 Data Grid クラスターの Pod 数を定義します。

num_owners

キャッシュ内の各エントリーのレプリカの数を定義するキャッシュ設定属性です。

15.2. ANTI-AFFINITY の設定

OpenShift が、Data Grid クラスターの Pod をスケジュールする場所を指定し、可用性を確保します。

手順

1. **spec.affinity** ブロックを **Infinispan** CR に追加します。
2. 必要に応じて anti-affinity ストラテジーを設定します。
3. **Infinispan** CR を適用します。

15.2.1. anti-affinity ストラテジーの設定

Infinispan CR で anti-affinity ストラテジーを設定し、OpenShift が Data Grid レプリカ Pod をスケジュールする場所を制御します。

トポロジーキー	説明
topologyKey: "topology.kubernetes.io/zone"	Data Grid のレプリカ Pod を複数のゾーンにまたがってスケジュールします。
topologyKey: "kubernetes.io/hostname"	さまざまな OpenShift ノードで Data Grid レプリカ Pod をスケジュールします。

さまざまな OpenShift ノードでの Pod のスケジュール
以下は、**Infinispan** CR に **spec.affinity** フィールドを設定しない場合に、Data Grid Operator が使用する anti-affinity ストラテジーです。

```
spec:
  affinity:
    podAntiAffinity:
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 100
        podAffinityTerm:
          labelSelector:
            matchLabels:
              app: infinispan-pod
              clusterName: <cluster_name>
              infinispan_cr: <cluster_name>
          topologyKey: "kubernetes.io/hostname"
```

さまざまなノードが必要
以下の例では、さまざまなノードを利用できない場合、OpenShift は Data Grid Pod をスケジュールしません。

```
spec:
  affinity:
    podAntiAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
      - labelSelector:
          matchLabels:
            app: infinispan-pod
            clusterName: <cluster_name>
            infinispan_cr: <cluster_name>
        topologyKey: "topology.kubernetes.io/hostname"
```

注記

さまざまな OpenShift ノードで Data Grid レプリカ Pod をスケジュールできるようにするには、利用可能な OpenShift ノードの数は **spec.replicas** の値よりも大きくなければなりません。

複数の OpenShift ゾーンにまたがった Pod のスケジュール
以下の例では、Pod のスケジュール時に複数のゾーンを優先しますが、ゾーンをまたいでスケジュールできない場合は、さまざまな OpenShift ノードで Data Grid レプリカ Pod をスケジュールします。

```
spec:
  affinity:
    podAntiAffinity:
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 100
        podAffinityTerm:
          labelSelector:
            matchLabels:
              app: infinispan-pod
              clusterName: <cluster_name>
              infinispan_cr: <cluster_name>
          topologyKey: "topology.kubernetes.io/zone"
      - weight: 90
        podAffinityTerm:
          labelSelector:
            matchLabels:
              app: infinispan-pod
              clusterName: <cluster_name>
              infinispan_cr: <cluster_name>
          topologyKey: "kubernetes.io/hostname"
```

複数のゾーンが必要
以下の例では、Data Grid レプリカ Pod をスケジュールする場合にのみ、ゾーンストラテジーを使用します。

```
spec:
  affinity:
    podAntiAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
      - labelSelector:
          matchLabels:
            app: infinispan-pod
            clusterName: <cluster_name>
            infinispan_cr: <cluster_name>
        topologyKey: "topology.kubernetes.io/zone"
```

第16章 DATA GRID OPERATOR を使用したキャッシュの作成

Cache CR を使用して、Data Grid Operator でキャッシュ設定を追加し、Data Grid がデータを保存する方法を制御します。

16.1. DATA GRID キャッシュ

キャッシュ設定はデータストアの特性および機能を定義し、Data Grid スキーマで有効である必要があります。Data Grid は、キャッシュ設定を定義する XML または JSON 形式のスタンドアロンファイルを作成することを推奨します。検証を容易にし、Java などのクライアント言語で XML スニペットを維持する必要がある状況を回避するために、Data Grid 設定をアプリケーションコードから分離する必要があります。

OpenShift で実行されている Data Grid クラスターでキャッシュを作成するには、以下を行う必要があります。

- **Cache** CR を OpenShift フロントエンドでキャッシュを作成するためのメカニズムとして使用します。
- **Batch** CR を使用して、スタンドアロン設定ファイルから一度に複数のキャッシュを作成します。
- Data Grid コンソール にアクセスし、XML または JSON 形式でキャッシュを作成します。

Hot Rod または HTTP クライアントを使用できますが、特定のユースケースでプログラムによるリモートキャッシュの作成が必要でない限り、Data Grid は **Cache** CR または **Batch** CR を推奨します。

キャッシュ CR

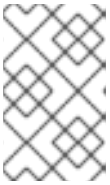
- **Cache** CR は Data Grid サービス Pod にのみ適用されます。
- 各 **Cache** CR は、Data Grid クラスター上の単一のキャッシュに対応します。

16.2. CACHE CR を使用したキャッシュの作成

XML または YAML 形式の有効な設定を使用して、Data Grid サービスクラスターにキャッシュを作成するには、次の手順を実行します。

手順

1. **metadata.name** フィールドに一意の値を使用して **Cache** CR を作成します。
2. ターゲット Data Grid クラスターを **spec.clusterName** フィールドで指定します。
3. **spec.name** フィールドで、キャッシュに名前を付けます。



注記

キャッシュ設定の **name** 属性は有効になりません。**spec.name** フィールドで名前を指定しない場合、キャッシュは **metadata.name** フィールドの値を使用します。

4. **spec.template** フィールドを使用してキャッシュ設定を追加します。

5. **Cache** CR を適用します。以下に例を示します。

```
oc apply -f mycache.yaml
cache.infinispan.org/mycachedefinition created
```

CR の例をキャッシュする

XML

```
apiVersion: infinispan.org/v2alpha1
kind: Cache
metadata:
  name: mycachedefinition
spec:
  clusterName: infinispan
  name: myXMLcache
  template: <distributed-cache mode="SYNC" statistics="true"><encoding media-type="application/x-protostream"/><persistence><file-store/></persistence></distributed-cache>
```

YAML

```
apiVersion: infinispan.org/v2alpha1
kind: Cache
metadata:
  name: mycachedefinition
spec:
  clusterName: infinispan
  name: myYAMLcache
  template: |-
    distributedCache:
      mode: "SYNC"
      owners: "2"
      statistics: "true"
      encoding:
        mediaType: "application/x-protostream"
      persistence:
        fileStore: ~
```

16.3. CACHE CR を使用したキャッシュの更新

Data Grid Operator が **Cache** CR のキャッシュ設定の変更を処理する方法を制御できます。

Data Grid Operator は、実行時に Data Grid Server のキャッシュ設定を更新しようとします。更新に失敗すると、Data Grid Operator は以下のストラテジーのいずれかを使用します。

保持ストラテジー

Operator は **Cache** CR のステータスを **Ready=False** に更新します。**Cache** CR を手動で削除し、新しいキャッシュ設定を作成できます。これはデフォルトのストラテジーです。

再作成ストラテジー

Operator は Data Grid クラスターからキャッシュを削除し、**Cache** CR から最新の **spec.template** 値で新しいキャッシュを作成します。

重要

デプロイメントがデータ損失を許容できる場合にのみ **recreate** ストラテジーを設定します。

前提条件

- 有効な **Cache** CR があること。

手順

1. **spec.updates.strategy** フィールドを使用して **Cache** CR ストラテジーを設定します。

   mycache.yaml

   ```
   spec:
     updates:
       strategy: recreate
   ```

2. 変更を **Cache** CR に適用します。以下に例を示します。

   ```
   oc apply -f mycache.yaml
   ```

16.4. 永続キャッシュストアの追加

永続キャッシュストアを Data Grid サービス Pod に追加して、データを永続ボリュームに保存できます。

Data Grid は、**/opt/infinispan/server/data** ディレクトリーに単一ファイルキャッシュストア (**.dat** ファイル) を作成します。

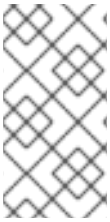
手順

- 以下の例のように、**<file-store/>** 要素を Data Grid キャッシュの **persistence** 設定に追加します。

  ```
  <distributed-cache name="persistent-cache" mode="SYNC">
    <encoding media-type="application/x-protostream"/>
    <persistence>
      <file-store/>
    </persistence>
  </distributed-cache>
  ```

16.5. キャッシュサービス POD へのキャッシュの追加

キャッシュサービス Pod には、推奨設定のデフォルトのキャッシュ設定が含まれます。このデフォルトのキャッシュを使用すると、キャッシュを作成せずに Data Grid の使用を開始できます。



注記

デフォルトのキャッシュは推奨される設定を提供するため、キャッシュをデフォルトのコピーとしてのみ作成する必要があります。複数のカスタムキャッシュが必要な場合は、キャッシュサービス Pod の代わりに Data Grid サービス Pod を作成する必要があります。

手順

- Data Grid コンソールにアクセスし、XML または JSON 形式でデフォルト設定のコピーを提供します。
- Data Grid CLI を使用して、以下のようにデフォルトキャッシュからコピーを作成します。

```
[//containers/default]> create cache --template=default mycache
```

16.5.1. デフォルトのキャッシュ設定

このトピックでは、キャッシュサービス Pod のデフォルトのキャッシュ設定について説明します。

```
<distributed-cache name="default"
                   mode="SYNC"
                   owners="2">
  <memory storage="OFF_HEAP"
          max-size="<maximum_size_in_bytes>"
          when-full="REMOVE" />
  <partition-handling when-split="ALLOW_READ_WRITES"
                      merge-policy="REMOVE_ALL"/>
</distributed-cache>
```

デフォルトのキャッシュ:

- 同期分散を使用して、クラスター全体でデータを保存します。
- クラスターの各エントリーの 2 つのレプリカを作成します。
- キャッシュエントリーをネイティブメモリー (off-heap) にバイトとして保存します。
- データコンテナーの最大サイズをバイト単位で定義します。Data Grid Operator は、Pod の作成時に最大サイズを計算します。
- キャッシュエントリーをエビクトして、データコンテナーのサイズを制御します。自動スケーリングを有効にして、エントリーを削除する代わりにメモリー使用量が増加したときに Data Grid Operator が Pod を追加するようにすることができます。
- セグメントの所有者が異なるパーティションにある場合でも、キャッシュエントリーの読み取りおよび書き込み操作を可能にする競合解決ストラテジーを使用します。
- Data Grid が競合を検出すると、キャッシュからエントリーを削除するマージポリシーを指定します。

第17章 バッチ操作の実行

Data Grid Operator は、Data Grid リソースを一括で作成できる **Batch** CR を提供します。**Batch** CR は、Data Grid コマンドラインインターフェイス (CLI) をバッチモードで使用して、操作のシーケンスを実行します。

注記

Batch CR インスタンスを変更しても効果はありません。バッチ操作は、Data Grid リソースを変更する "one-time" イベントです。CR の **.spec** フィールドを更新するには、またはバッチ操作が失敗した場合には、**Batch** CR の新規インスタンスを作成する必要があります。

17.1. インラインバッチ操作の実行

別の設定アーティファクトを必要としない場合は、バッチ操作を **Batch** CR に直接追加します。

手順

1. **Batch** CR を作成します。
   a. バッチ操作を **spec.cluster** フィールドの値として実行する Data Grid クラスターの名前を指定します。
   b. 各 CLI コマンドを追加して、**spec.config** フィールドの行で実行します。

```
apiVersion: infinispan.org/v2alpha1
kind: Batch
metadata:
  name: mybatch
spec:
  cluster: infinispan
  config: |
    create cache --template=org.infinispan.DIST_SYNC mycache
    put --cache=mycache hello world
    put --cache=mycache hola mundo
```

2. **Batch** CR を適用します。

```
oc apply -f mybatch.yaml
```

3. **Batch** CR が成功するまで待ちます。

```
oc wait --for=jsonpath='{.status.phase}'=Succeeded Batch/mybatch
```

17.2. バッチ操作の CONFIGMAP の作成

Data Grid キャッシュ設定などの追加のファイルがバッチ操作で使用できるように **ConfigMap** を作成します。

前提条件

デモンストレーションの目的で、手順を開始する前に、ホストファイルシステムにいくつかの設定アーティファクトを追加する必要があります。

- 一部のファイルを追加できる **/tmp/mybatch** ディレクトリーを作成します。

```
mkdir -p /tmp/mybatch
```

- Data Grid キャッシュ設定を作成します。

```
cat > /tmp/mybatch/mycache.xml<<EOF
<distributed-cache name="mycache" mode="SYNC">
  <encoding media-type="application/x-protostream"/>
  <memory max-count="1000000" when-full="REMOVE"/>
</distributed-cache>
EOF
```

手順

1. 実行するすべてのコマンドが含まれる **batch** ファイルを作成します。
たとえば、以下の **batch** ファイルは、mycache という名前のキャッシュを作成し、2 つのエントリーをこれに追加します。

```
create cache mycache --file=/etc/batch/mycache.xml
put --cache=mycache hello world
put --cache=mycache hola mundo
```

重要

ConfigMap は、**/etc/batch** の Data Grid Pod にマウントされます。バッチ操作のすべての **--file=** ディレクティブの前に、そのパスを付ける必要があります。

2. バッチ操作が必要とするすべての設定アーティファクトが、**batch** ファイルと同じディレクトリーにあることを確認します。

```
ls /tmp/mybatch

batch
mycache.xml
```

3. ディレクトリーから **ConfigMap** を作成します。

```
oc create configmap mybatch-config-map --from-file=/tmp/mybatch
```

17.3. CONFIGMAP を使用したバッチ操作の実行

設定アーティファクトを含むバッチ操作を実行します。

前提条件

- バッチ操作が必要とするファイルを含む **ConfigMap** を作成している。

手順

1. Data Grid クラスターの名前を **spec.cluster** フィールドの値として指定する **Batch** CR を作成します。
2. **spec.configMap** フィールドで **batch** ファイルおよび設定アーティファクトを含む **ConfigMap** の名前を設定します。

```
cat > mybatch.yaml<<EOF
apiVersion: infinispan.org/v2alpha1
kind: Batch
metadata:
  name: mybatch
spec:
  cluster: infinispan
  configMap: mybatch-config-map
EOF
```

3. **Batch** CR を適用します。

```
oc apply -f mybatch.yaml
```

4. **Batch** CR が成功するまで待ちます。

```
oc wait --for=jsonpath='{.status.phase}'=Succeeded Batch/mybatch
```

17.4. バッチステータスメッセージ

Batch CR の **status.Phase** フィールドで、バッチ操作を確認し、トラブルシューティングします。

フェーズ	説明
Succeeded	すべてのバッチ操作が正常に完了しました。
Initializing	バッチ操作はキューに入れられ、リソースは初期化されています。
Initialized	バッチ操作を開始する準備ができました。
Running	バッチ操作が進行中です。
Failed	1 つ以上のバッチ操作に失敗しました。

失敗した操作

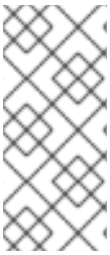
バッチ操作はアトミックではありません。batch スクリプトのコマンドが失敗しても、他の操作に影響を与えたり、ロールバックしたりすることはありません。

注記

バッチ操作にサーバーまたは構文エラーがある場合は、**status.Reason** フィールドの **Batch** CR でログメッセージを表示できます。

17.5. バッチ操作の例

これらのバッチ操作の例を、**Batch** CR で Data Grid リソースを作成および変更する際の開始点として使用します。



注記

設定ファイルは、**ConfigMap** を介してのみ、Data Grid Operator に渡すことができます。

ConfigMap は、**/etc/batch** の Data Grid Pod にマウントされるため、すべての **--file=** ディレクティブの前にそのパスを追加する必要があります。

17.5.1. キャッシュ

- 設定ファイルから複数のキャッシュを作成します。

```
echo "creating caches..."
create cache sessions --file=/etc/batch/infinispan-prod-sessions.xml
create cache tokens --file=/etc/batch/infinispan-prod-tokens.xml
create cache people --file=/etc/batch/infinispan-prod-people.xml
create cache books --file=/etc/batch/infinispan-prod-books.xml
create cache authors --file=/etc/batch/infinispan-prod-authors.xml
echo "list caches in the cluster"
ls caches
```

- ファイルからテンプレートを作成してから、テンプレートからキャッシュを作成します。

```
echo "creating caches..."
create cache mytemplate --file=/etc/batch/mycache.xml
create cache sessions --template=mytemplate
create cache tokens --template=mytemplate
echo "list caches in the cluster"
ls caches
```

17.5.2. カウンター

Batch CR を使用して、オブジェクトの数を記録するためにインクリメントおよびデクリメントできる複数のカウンターを作成します。

カウンターを使用して、識別子を生成したり、レートリミッターとして機能したり、リソースがアクセスされた回数を追跡したりできます。

```
echo "creating counters..."
create counter --concurrency-level=1 --initial-value=5 --storage=PERSISTENT --type=weak
mycounter1
create counter --initial-value=3 --storage=PERSISTENT --type=strong mycounter2
create counter --initial-value=13 --storage=PERSISTENT --type=strong --upper-bound=10
mycounter3
echo "list counters in the cluster"
ls counters
```

17.5.3. Protobuf スキーマ

Protobuf スキーマを登録して、キャッシュ内の値をクエリーします。Protobuf スキーマ (**.proto** files) は、カスタムエンティティーに関するメタデータを提供し、フィールドのインデックス作成を制御します。

```
echo "creating schema..."
schema --upload=person.proto person.proto
schema --upload=book.proto book.proto
schema --upload=author.proto book.proto
echo "list Protobuf schema"
ls schemas
```

17.5.4. タスク

org.infinispan.tasks.ServerTask を実装するタスク、または **javax.script** スクリプト API と互換性のあるスクリプトをアップロードします。

```
echo "creating tasks..."
task upload --file=/etc/batch/myfirstscript.js myfirstscript
task upload --file=/etc/batch/mysecondscript.js mysecondscript
task upload --file=/etc/batch/mythirdscript.js mythirdscript
echo "list tasks"
ls tasks
```

関連情報

- Data Grid CLI Guide

第18章 DATA GRID クラスターのバックアップおよび復元

Data Grid Operator を使用すると、障害復旧およびクラスター間で Data Grid リソースを移行するために、Data Grid クラスターの状態をバックアップおよび復元できます。

18.1. BACKUP CR および RESTORE CR

Backup CR および **Restore** CR は、ランタイム時にインメモリーデータを保存し、Data Grid クラスターを簡単に再作成できるようにします。

Backup CR または **Restore** CR を適用すると、Data Grid クラスターにゼロ容量メンバーとして参加する新しい Pod を作成します。つまり、参加するためにクラスターのリバランスまたは状態遷移は必要ありません。

バックアップ操作の場合、Pod はキャッシュエントリーおよびその他のリソースを繰り返し処理し、永続ボリューム (PV) の **/opt/infinispan/backups** ディレクトリーにアーカイブ (**.zip**) を作成します。

注記

Data Grid クラスターの他の Pod は、キャッシュエントリーを繰り返し処理するときにバックアップ Pod に応答するだけでよいため、バックアップの実行がパフォーマンスに大きな影響を与えることはありません。

復元操作の場合、Pod は PV のアーカイブから Data Grid リソースを取得し、それらを Data Grid クラスターに適用します。

バックアップまたは復元操作が完了すると、Pod はクラスターを離れ、終了します。

調整

Data Grid Operator は**Backup** CR および **Restore** CR を調整しません。これは、バックアップと復元操作は "one-time" イベントであることを意味します。

既存の **Backup** CR または **Restore** CR インスタンスを変更しても、操作は実行されず、効果もありません。**.spec** フィールドを更新する場合は、**Backup** CR または **Restore** CR の新規インスタンスを作成する必要があります。

18.2. DATA GRID クラスターのバックアップ

Data Grid クラスターの状態を永続ボリュームに保存するバックアップファイルを作成します。

前提条件

- **spec.service.type: DataGrid** で **Infinispan** CR を作成します。
- Data Grid クラスターへのアクティブなクライアント接続がないことを確認します。
  Data Grid のバックアップは、スナップショットの分離を提供しません。また、キャッシュがバックアップされた後、データの変更はアーカイブに書き込まれません。
  クラスターの正確な状態をアーカイブするには、バックアップする前に、クライアントを常に切断する必要があります。

手順

1. **metadata.name** フィールドで **Backup** CR に名前を付けます。

2. **spec.cluster** フィールドでバックアップする Data Grid クラスターを指定します。

3. **spec.volume.storage** および **spec.volume.storage.storageClassName** フィールドで、バックアップアーカイブを永続ボリューム (PV) に追加する Persistent Volume Claim(永続ボリューム要求、PVC) を設定します。

```
apiVersion: infinispan.org/v2alpha1
kind: Backup
metadata:
  name: my-backup
spec:
  cluster: source-cluster
  volume:
    storage: 1Gi
    storageClassName: my-storage-class
```

4. 任意で **spec.resources** フィールドを含めて、バックアップを作成する Data Grid リソースを指定します。
spec.resources フィールドを含めない場合、**Backup** CR はすべての Data Grid リソースが含まれるアーカイブを作成します。**spec.resources** フィールドを指定した場合、**Backup** CR はそれらのリソースのみが含まれるアーカイブを作成します。

```
spec:
  ...
  resources:
    templates:
      - distributed-sync-prod
      - distributed-sync-dev
    caches:
      - cache-one
      - cache-two
    counters:
      - counter-name
    protoSchemas:
      - authors.proto
      - books.proto
    tasks:
      - wordStream.js
```

以下の例のように * ワイルドカード文字を使用することもできます。

```
spec:
  ...
  resources:
    caches:
      - "*"
    protoSchemas:
      - "*"
```

5. **Backup** CR を適用します。

```
oc apply -f my-backup.yaml
```

検証

1. **status.phase** フィールドに **Backup** CR の **Succeeded** のステータスがあり、Data Grid ログに以下のメッセージがあることを確認します。

```
ISPN005044: Backup file created 'my-backup.zip'
```

2. 以下のコマンドを実行して、バックアップが正常に作成されていることを確認します。

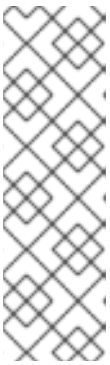
```
oc describe Backup my-backup
```

18.3. DATA GRID クラスターの復元

バックアップアーカイブから Data Grid クラスターの状態を復元します。

前提条件

- ソースクラスターに **Backup** CR を作成します。
- Data Grid サービス Pod のターゲット Data Grid クラスターを作成します。



注記

既存のキャッシュを復元する場合、操作はキャッシュ内のデータを上書きしますが、キャッシュ設定は上書きしません。

たとえば、ソースクラスターに **mycache** という名前の分散キャッシュをバックアップします。次に、すでに複製されたキャッシュとして存在するターゲットクラスターで **mycache** を復元します。この場合、ソースクラスターからのデータは復元され、**mycache** はターゲットクラスターで複製された設定を維持します。

- 復元するターゲットの Data Grid クラスターにアクティブなクライアント接続がないことを確認します。
バックアップから復元したキャッシュエントリーは、最近のキャッシュエントリーを上書きする可能性があります。
たとえば、クライアントが **cache.put(k=2)** 操作を実行してから、**k=1** を含むバックアップを復元します。

手順

1. **metadata.name** フィールドで **Restore** CR に名前を付けます。
2. **spec.backup** フィールドで使用する **Backup** CR を指定します。
3. **spec.cluster** フィールドで復元する Data Grid クラスターを指定します。

```
apiVersion: infinispan.org/v2alpha1
kind: Restore
metadata:
  name: my-restore
spec:
  backup: my-backup
  cluster: target-cluster
```

4. 任意で **spec.resources** フィールドを追加して、特定のリソースのみを復元します。

```
spec:
  ...
  resources:
    templates:
      - distributed-sync-prod
      - distributed-sync-dev
    caches:
      - cache-one
      - cache-two
    counters:
      - counter-name
    protoSchemas:
      - authors.proto
      - books.proto
    tasks:
      - wordStream.js
```

5. **Restore** CR を適用します。

```
oc apply -f my-restore.yaml
```

検証

- **status.phase** フィールドに **Restore** CR の **Succeeded** のステータスがあり、Data Grid ログに以下のメッセージがあることを確認します。

```
ISPN005045: Restore 'my-backup' complete
```

その後、Data Grid コンソールを開くか、CLI 接続を確立して、データおよび Data Grid リソースが予想通りに復元されていることを確認します。

18.4. バックアップおよび復元のステータス

Backup および **Restore** CR には、操作の各フェーズのステータスを提供する **status.phase** フィールドが含まれます。

ステータス	説明
Initializing	システムは要求を受け入れ、コントローラーは Pod を作成する基礎となるリソースを準備しています。
Initialized	コントローラーは、すべての基礎となるリソースを正常に準備しました。
Running	Pod が作成され、Data Grid クラスターで操作が進行中です。
Succeeded	Data Grid クラスターで操作が正常に完了し、Pod が終了しています。

ステータス	説明
Failed	操作が正常に完了せず、Pod は終了しました。
Unknown	コントローラーは Pod のステータスを取得したり、操作の状態を判別したりすることはできません。この条件は通常、Pod との一時的な通信エラーを示しています。

18.4.1. 失敗したバックアップおよび復元操作の処理

Backup CR または **Restore** CR の **status.phase** フィールドが **Failed** の場合、再度操作を試行する前に Pod ログを確認して根本原因を判別する必要があります。

手順

1. 失敗した操作を実行した Pod のログを確認します。
   Pod は終了しますが、**Backup** CR または **Restore** CR を削除するまでそのまま利用できます。

   ```
   oc logs <backup|restore_pod_name>
   ```

2. Pod ログによって示されるエラー状態またはその他の障害の原因を解決します。
3. **Backup** CR または **Restore** CR の新規インスタンスを作成し、操作を再度試みます。

第19章 DATA GRID へのカスタムコードのデプロイ

スクリプトやイベントリスナーなどのカスタムコードを Data Grid クラスターに追加します。

カスタムコードを Data Grid クラスターにデプロイする前に、これを利用可能にする必要があります。これを行うには、永続ボリューム (PV) からアーティファクトをコピーするか、HTTP または FTP サーバーからアーティファクトをダウンロードします。あるいは、両方の方法を使用することができます。

19.1. DATA GRID クラスターへのコードアーティファクトのコピー

アーティファクトを永続ボリューム (PV) に追加してから、これらを Data Grid Pod にコピーします。

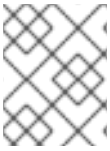
この手順では、以下を実行する永続ボリューム要求 (PVC) をマウントする一時的な Pod を使用する方法について説明します。

- コードのアーティファクトを PV に追加できます (書き込み操作を実行します)。
- Data Grid Pod が PV からコードアーティファクトをロードできるようにします (読み取り操作を実行します)。

これらの読み取りおよび書き込み操作を実行するには、特定の PV アクセスモードが必要です。ただし、さまざまな PVC アクセスモードのサポートはプラットフォームに依存します。

さまざまなプラットフォームで PVC を作成する方法については、本書では扱いません。分かりやすくするため、以下の手順では **ReadWriteMany** アクセスモードの PVC を示しています。

場合によっては、**ReadOnlyMany** または **ReadWriteOnce** アクセスモードのみを使用できます。同じ **spec.volumeName** の PVC を回収し、再利用することで、これらのアクセスモードの組み合わせを使用できます。



注記

ReadWriteOnce アクセスモードを使用すると、クラスター内のすべての Data Grid Pod が同じ OpenShift ノードにスケジュールされます。

手順

1. Data Grid クラスターの namespace に変更します。

```
oc project rhdg-namespace
```

2. 以下のように、カスタムコードアーティファクトの PVC を作成します。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: datagrid-libs
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 100Mi
```

3. PVC を適用します。

```
oc apply -f datagrid-libs.yaml
```

4. 以下のように、PVC をマウントする Pod を作成します。

```
apiVersion: v1
kind: Pod
metadata:
  name: datagrid-libs-pod
spec:
  securityContext:
    fsGroup: 2000
  volumes:
    - name: lib-pv-storage
      persistentVolumeClaim:
        claimName: datagrid-libs
  containers:
    - name: lib-pv-container
      image: registry.redhat.io/datagrid/datagrid-8-rhel8:8.4
      volumeMounts:
        - mountPath: /tmp/libs
          name: lib-pv-storage
```

5. Pod を Data Grid namespace に追加し、準備ができるまで待機します。

```
oc apply -f datagrid-libs-pod.yaml
oc wait --for=condition=ready --timeout=2m pod/datagrid-libs-pod
```

6. コードのアーティファクトを Pod にコピーし、それらが PVC に読み込まれるようにします。たとえば、ローカルの **locallibs** ディレクトリーからコードアーティファクトをコピーするには、以下を実行します。

```
oc cp --no-preserve=true libs datagrid-libs-pod:/tmp/
```

7. Pod を削除します。

```
oc delete pod datagrid-libs-pod
```

永続ボリュームを **Infinispan** CR の **spec.dependencies.volumeClaimName** で指定してから、変更を適用します。

```
apiVersion: infinispan.org/v1
kind: Infinispan
metadata:
  name: infinispan
spec:
  replicas: 2
  dependencies:
    volumeClaimName: datagrid-libs
  service:
    type: DataGrid
```

注記

永続ボリュームでカスタムコードを更新する場合は、変更を読み込むことができるように、Data Grid クラスターを再起動する必要があります。

関連情報

- Configuring persistent storage
- Persistent Volumes
- Access Modes
- How to manually reclaim and reuse OpenShift Persistent volumes that are "Released" (Red Hat ナレッジベース)

19.2. コードアーティファクトのダウンロード

アーティファクトを HTTP または FTP サーバーに追加し、Data Grid Operator が各 Data Grid ノードの **{lib_path}** ディレクトリーにアーティファクトをダウンロードできるようにします。

ファイルのダウンロード時に、Data Grid Operator はファイルタイプを自動的に検出できます。Data Grid Operator は、ダウンロード完了後に **zip** または **tgz** などのアーカイブファイルもファイルシステムにデプロイメントします。

org.postgresql:postgresql:42.3.1 などの **groupId:artifactId:version** 形式を使用して Maven アーティファクトをダウンロードすることもできます。

注記

Data Grid Operator が Data Grid ノードを作成するたびに、アーティファクトをノードにダウンロードします。

前提条件

- コードアーティファクトを HTTP または FTP サーバーでホストするか、maven リポジトリーに公開しておく。

手順

1. **spec.dependencies.artifacts** フィールドを **Infinispan** CR に追加します。
2. 次のいずれかを行います。
   - **HTTP** または **FTP** 経由でダウンロードするファイルの場所を **spec.dependencies.artifacts.url** フィールドの値として指定します。
   - ダウンロードする Maven アーティファクトを、**spec.dependencies.artifacts.maven** フィールドの値として **groupId:artifactId:version** 形式で提供します。
3. オプションで、**spec.dependencies.artifacts.hash** フィールドでダウンロードの整合性を検証するチェックサムを指定します。
   hash フィールドでは、値が **<algorithm>:<checksum>** の形式で指定する必要があります。ここで、**<algorithm>** は **sha1|sha224|sha256|sha384|sha512|md5** になります。

```
apiVersion: infinispan.org/v1
kind: Infinispan
metadata:
  name: infinispan
spec:
  replicas: 2
  dependencies:
    artifacts:
      - url: http://example.com:8080/path
        hash:
sha256:596408848b56b5a23096baa110cd8b633c9a9aef2edd6b38943ade5b4edcd686
  service:
    type: DataGrid
```

4. 変更を適用します。

第20章 DATA GRID クラスターからのクラウドイベントの送信

CloudEvents を Apache Kafka トピックに送信し、Data Grid を Knative ソースとして設定します。

重要

Red Hat OpenShift Serverless を使用したクラウドイベントの送信は、テクノロジープレビュー機能としてご利用いただけます。

20.1. テクノロジープレビュー機能

テクノロジープレビュー機能は、Red Hat の実稼働環境でのサービスレベルアグリーメント (SLA) ではサポートされておらず、機能的に完全でない可能性があります。

Red Hat は、テクノロジープレビュー機能の実稼働環境での使用を推奨していません。これらの機能により、近日発表予定の製品機能をリリースに先駆けてご提供でき、お客様は開発プロセス時に機能をテストして、フィードバックをお寄せいただくことができます。

詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

重要

この機能は非推奨であり、RHDG 8.4.x 以降の Data Grid リリースでは効果がなくなる予定です。

20.2. クラウドイベント

キャッシュ内のエントリーの作成、更新、削除、または期限切れの場合に、Data Grid クラスターから **CloudEvents** を送信できます。

Data Grid は、以下の例のように構造化イベントを JSON 形式で Kafka に送信します。

```
{
   "specversion": "1.0",
   "source": "/infinispan/<cluster_name>/<cache_name>",
   "type": "org.infinispan.entry.created",
   "time": "<timestamp>",
   "subject": "<key-name>",
   "id": "key-name:CommandInvocation:node-name:0",
   "data": {
      "property": "value"
   }
}
```

フィールド	説明
type	Data Grid キャッシュエントリーのイベントの前に **org.infinispan.entry** を付けます。
data	エントリー値。

フィールド	説明
subject	文字列に変換されたエントリーキー。
id	イベント用に生成された識別子。

20.3. クラウドイベントの有効化

CloudEvents を送信するように Data Grid を設定します。

前提条件

- Data Grid トピックをリッスンする Kafka クラスターを設定します。

手順

1. **spec.cloudEvents** を **Infinispan** CR に追加します。
   a. **spec.cloudEvents.acks** フィールドで確認応答の数を設定します。値は 0、1、または all です。
   b. Data Grid が **spec.cloudEvents.bootstrapServers** フィールドでイベントを送信する Kafka サーバーをリスト表示します。
   c. Data Grid イベントの Kafka トピックを **spec.cloudEvents.cacheEntriesTopic** フィールドで指定します。

```
spec:
  cloudEvents:
    acks: "1"
    bootstrapServers: my-cluster-kafka-bootstrap_1.<namespace_1>.svc:9092,my-cluster-kafka-bootstrap_2.<namespace_2>.svc:9092
    cacheEntriesTopic: target-topic
```

2. 変更を適用します。

第21章 リモートクライアント接続の確立

Data Grid Console、コマンドラインインターフェイス (CLI)、およびリモートクライアントから Data Grid クラスターに接続します。

21.1. クライアント接続の詳細

Data Grid へのクライアント接続には、次の情報が必要です。

- Hostname
- ポート
- 認証用の認証情報 (必要な場合)
- 暗号化を使用する場合の TLS 証明書

ホスト名

使用するホスト名は、クライアントが Data Grid と同じ OpenShift クラスターで実行されているかどうかによって異なります。

同じ OpenShift クラスターで実行されているクライアントアプリケーションは、Data Grid クラスターの内部サービス名を使用します。

```
metadata:
  name: infinispan
```

別の OpenShift または OpenShift の外部で実行されているクライアントアプリケーションは、Data Grid がネットワーク上でどのように公開されているかに依存するホスト名を使用します。

LoadBalancer サービスは、ロードバランサーの URL を使用します。**NodePort** サービスは、ノードのホスト名を使用します。Red Hat OpenShift **Route** は、定義したカスタムホスト名またはシステムが生成したホスト名のいずれかを使用します。

ポート

OpenShift および **LoadBalancer** サービス経由のクライアント接続はポート **11222** を使用します。

NodePort サービスは、 **30000** から **60000** の範囲のポートを使用します。ルートは、ポート **80** (暗号化なし) または **443** (暗号化あり) を使用します。

関連情報

- Configuring Network Access to Data Grid
- Retrieving Credentials
- Retrieving TLS Certificates

21.2. リモートシェルを使用した DATA GRID クラスターへの接続

Data Grid クラスターへのリモートシェルセッションを開始し、コマンドラインインターフェイス (CLI) を使用して Data Grid リソースと連携して管理操作を実行します。

前提条件

- **PATH** に **kubectl-infinispan** があること。
- 有効な Data Grid 認証情報があること。

手順

1. **infinispan shell** コマンドを実行して、Data Grid クラスターに接続します。

```
oc infinispan shell <cluster_name>
```

注記

認証シークレットにアクセスでき、Data Grid ユーザーが 1 つしかない場合には、**kubectl-infinispan** プラグインは認証情報を自動的に検出して Data Grid に対して認証します。デプロイメントに複数の Data Grid 認証情報がある場合は、**--username** 引数を使用してユーザーを指定し、プロンプトが表示されたら、対応するパスワードを入力します。

2. 必要に応じて CLI 操作を実行します。

ヒント

Tab キーを押すか、**--help** 引数を使用して、利用可能なオプションとヘルプテキストを表示します。

3. **quit** コマンドを使用して、リモートシェルセッションを終了します。

関連情報

- Data Grid コマンドラインインターフェイスの使用

21.3. DATA GRID コンソールへのアクセス

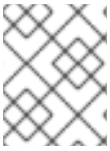
コンソールにアクセスして、キャッシュの作成、管理操作の実行、および Data Grid クラスターの監視を行います。

前提条件

- ブラウザーからコンソールにアクセスできるように、ネットワーク上で Data Grid を公開している。
  たとえば、**LoadBalancer** サービスを設定するか、**Route** を作成します。

手順

- **$HOSTNAME:$PORT** にある任意のブラウザーからコンソールにアクセスします。
  $HOSTNAME:$PORT を、Data Grid が使用可能なネットワークの場所に置き換えます。



注記

Data Grid コンソールへのアクセスは、OpenShift サービス経由またはポート 11222 を公開する OpenShift **Route** 経由でのみ行ってください。

21.4. HOT ROD クライアント

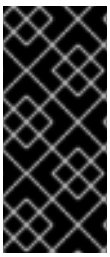
Hot Rod は、Data Grid がリモートクライアントで高性能データ転送機能を提供するバイナリー TCP プロトコルです。

クライアントのインテリジェンス
Hot Rod プロトコルには、クライアントにキャッシュトポロジーの最新のビューを提供するメカニズムが含まれています。クライアントインテリジェンスは、読み取りおよび書き込み操作のネットワークホップ数を減らすことにより、パフォーマンスを向上させます。

同じ OpenShift クラスターで実行されているクライアントは、Data Grid Pod の内部 IP アドレスにアクセスできるため、任意のクライアントインテリジェンスを使用できます。

HASH_DISTRIBUTION_AWARE は、デフォルトのインテリジェンスメカニズムであり、クライアントがリクエストをプライマリーオーナーにルーティングできるようにします。これにより、Hot Rod クライアントに最高のパフォーマンスが提供されます。

別の OpenShift または OpenShift の外部で実行されているクライアントは、**LoadBalancer**、**NodePort**、または OpenShift **Route** を使用して Data Grid にアクセスできます。



重要

OpenShift **Route** を介した Hot Rod クライアント接続には暗号化が必要です。SNI を使用して TLS を設定する必要があります。そうしないと、Hot Rod 接続に失敗します。

暗号化されていない Hot Rod クライアント接続の場合は、**LoadBalancer** サービスまたは **NodePort** サービスを使用する必要があります。

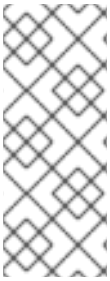
Hot Rod クライアントは、次の状況で **BASIC** インテリジェンスを使用する必要があります。

- **LoadBalancer** サービス、**NodePort** サービス、または OpenShift **Route** 経由で Data Grid に接続します。
- クロスサイトレプリケーションの使用時に別の OpenShift クラスターにフェイルオーバーする。

OpenShift クラスター管理者は、Data Grid へのトラフィックを制限するネットワークポリシーを定義できます。場合によっては、ネットワーク分離ポリシーにより、クライアントが同じ OpenShift クラスターで実行されているが namespace が異なる場合でも、**BASIC** インテリジェンスを使用する必要があります。

21.4.1. Hot Rod クライアント設定 API

ConfigurationBuilder インターフェイスを使用して、Hot Rod クライアント接続をプログラムで設定できます。



注記

次の例の **$SERVICE_HOSTNAME** を、Data Grid クラスターの内部サービス名に置き換えます。

```
metadata:
  name: infinispan
```

OpenShift の場合

ConfigurationBuilder

```
import org.infinispan.client.hotrod.configuration.ConfigurationBuilder;
import org.infinispan.client.hotrod.configuration.SaslQop;
import org.infinispan.client.hotrod.impl.ConfigurationProperties;
...

ConfigurationBuilder builder = new ConfigurationBuilder();
    builder.addServer()
            .host("$HOSTNAME")
            .port(ConfigurationProperties.DEFAULT_HOTROD_PORT)
           .security().authentication()
            .username("username")
            .password("changeme")
            .realm("default")
            .saslQop(SaslQop.AUTH)
            .saslMechanism("SCRAM-SHA-512")
           .ssl()
            .sniHostName("$SERVICE_HOSTNAME")
            .trustStoreFileName("/var/run/secrets/kubernetes.io/serviceaccount/service-ca.crt")
            .trustStoreType("pem");
```

hotrod-client.properties

```
# Connection
infinispan.client.hotrod.server_list=$HOSTNAME:$PORT

# Authentication
infinispan.client.hotrod.use_auth=true
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=SCRAM-SHA-512

# Encryption
infinispan.client.hotrod.sni_host_name=$SERVICE_HOSTNAME
infinispan.client.hotrod.trust_store_file_name=/var/run/secrets/kubernetes.io/serviceaccount/service-ca.crt
infinispan.client.hotrod.trust_store_type=pem
```

OpenShift の外部

ConfigurationBuilder

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

ConfigurationBuilder builder = new ConfigurationBuilder();
    builder.addServer()
            .host("$HOSTNAME")
            .port("$PORT")
           .security().authentication()
            .username("username")
            .password("changeme")
            .realm("default")
            .saslQop(SaslQop.AUTH)
            .saslMechanism("SCRAM-SHA-512")
           .ssl()
            .sniHostName("$SERVICE_HOSTNAME")
            //Create a client trust store with tls.crt from your project.
            .trustStoreFileName("/path/to/truststore.pkcs12")
            .trustStorePassword("trust_store_password")
            .trustStoreType("PCKS12");
    builder.clientIntelligence(ClientIntelligence.BASIC);
```

hotrod-client.properties

```
# Connection
infinispan.client.hotrod.server_list=$HOSTNAME:$PORT

# Client intelligence
infinispan.client.hotrod.client_intelligence=BASIC

# Authentication
infinispan.client.hotrod.use_auth=true
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=SCRAM-SHA-512

# Encryption
infinispan.client.hotrod.sni_host_name=$SERVICE_HOSTNAME
# Create a client trust store with tls.crt from your project.
infinispan.client.hotrod.trust_store_file_name=/path/to/truststore.pkcs12
infinispan.client.hotrod.trust_store_password=trust_store_password
infinispan.client.hotrod.trust_store_type=PCKS12
```

21.4.2. 証明書認証用の Hot Rod クライアントの設定

クライアント証明書認証を有効にすると、Data Grid との接続をネゴシエートする際に、クライアントは有効な証明書を提示する必要があります。

検証ストラテジー

Validate ストラテジーを使用する場合、署名済み証明書を提示できるように、キーストアでクライアントを設定する必要があります。また、Data Grid の認証情報と適切な認証メカニズムを使用してクライアントを設定する必要もあります。

認証ストラテジー

Authenticate ストラテジーを使用する場合、識別名 (DN) の一部として署名済み証明書および有効な Data Grid 認証情報が含まれるキーストアでクライアントを設定する必要があります。Hot Rod クライアントは、**EXTERNAL** 認証メカニズムも使用する必要があります。

注記

セキュリティー承認を有効にする場合、クライアント証明書から Common Name(CN) に適切なパーミッションを持つロールに割り当てる必要があります。

以下の例は、**Authenticate** ストラテジーを使用したクライアント証明書認証用の Hot Rod クライアント設定を示しています。

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

ConfigurationBuilder builder = new ConfigurationBuilder();
    builder.security()
          .authentication()
           .saslMechanism("EXTERNAL")
          .ssl()
           .keyStoreFileName("/path/to/keystore")
           .keyStorePassword("keystorepassword".toCharArray())
           .keyStoreType("PCKS12");
```

21.4.3. Hot Rod クライアントからのキャッシュの作成

Hot Rod クライアントを使用して、OpenShift で実行される Data Grid クラスターでキャッシュをリモートで作成できます。ただし、Data Grid は、Hot Rod クライアントではなく、Data Grid コンソール、CLI、または **Cache** CR を使用してキャッシュを作成することを推奨します。

プログラムでのキャッシュの作成
以下の例は、キャッシュ設定を **ConfigurationBuilder** に追加してから、**RemoteCacheManager** を使用して作成する方法を示しています。

```
import org.infinispan.client.hotrod.DefaultTemplate;
import org.infinispan.client.hotrod.RemoteCache;
import org.infinispan.client.hotrod.RemoteCacheManager;
...

    builder.remoteCache("my-cache")
          .templateName(DefaultTemplate.DIST_SYNC);
    builder.remoteCache("another-cache")
          .configuration("<infinispan><cache-container><distributed-cache name=\"another-cache\">
<encoding media-type=\"application/x-protostream\"/></distributed-cache></cache-container>
</infinispan>");
    try (RemoteCacheManager cacheManager = new RemoteCacheManager(builder.build())) {
    // Get a remote cache that does not exist.
    // Rather than return null, create the cache from a template.
    RemoteCache<String, String> cache = cacheManager.getCache("my-cache");
```

```
        // Store a value.
        cache.put("hello", "world");
        // Retrieve the value and print it.
        System.out.printf("key = %s\n", cache.get("hello"));
```

この例は、**XMLStringConfiguration()** メソッドを使用して、CacheWithXMLConfiguration という名前のキャッシュを作成し、キャッシュ設定を XML として渡す方法を示しています。

```
import org.infinispan.client.hotrod.RemoteCacheManager;
import org.infinispan.commons.configuration.XMLStringConfiguration;
...

private void createCacheWithXMLConfiguration() {
    String cacheName = "CacheWithXMLConfiguration";
    String xml = String.format("<distributed-cache name=\"%s\">" +
                               "<encoding media-type=\"application/x-protostream\"/>" +
                               "<locking isolation=\"READ_COMMITTED\"/>" +
                               "<transaction mode=\"NON_XA\"/>" +
                               "<expiration lifespan=\"60000\" interval=\"20000\"/>" +
                              "</distributed-cache>"
                              , cacheName);
    manager.administration().getOrCreateCache(cacheName, new XMLStringConfiguration(xml));
    System.out.println("Cache with configuration exists or is created.");
}
```

Hot Rod クライアントプロパティーの使用
存在しない名前付きキャッシュの **cacheManager.getCache()** 呼び出しを呼び出すと、Data Grid は null を返す代わりに Hot Rod クライアントプロパティーからそれらを作成します。

以下の例のように、キャッシュ設定を **hotrod-client.properties** に追加します。

```
# Add cache configuration
infinispan.client.hotrod.cache.my-cache.template_name=org.infinispan.DIST_SYNC
infinispan.client.hotrod.cache.another-cache.configuration=<infinispan><cache-container>
<distributed-cache name=\"another-cache\"/></cache-container></infinispan>
infinispan.client.hotrod.cache.my-other-cache.configuration_uri=file:/path/to/configuration.xml
```

21.5. REST API へのアクセス

Data Grid は、HTTP クライアントを使用して対話できる RESTful インターフェイスを提供します。

前提条件

- REST API にアクセスできるように、ネットワークで Data Grid を公開します。
  たとえば、**LoadBalancer** サービスを設定するか、**Route** を作成します。

手順

- **$HOSTNAME:$PORT/rest/v2** にある任意の HTTP クライアントで REST API にアクセスします。
  $HOSTNAME:$PORT を、Data Grid がクライアント接続をリッスンするネットワークの場所に置き換えます。

関連情報

- Data Grid REST API