Red Hat Data Grid 8.2

Data Grid Operator ガイド

OpenShift での Data Grid クラスターの作成

Last Updated: 2023-12-05

Red Hat Data Grid 8.2 Data Grid Operator ガイド

OpenShift での Data Grid クラスターの作成

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

Copyright © 2022 | You need to change the HOLDER entity in the en-US/Data_Grid_Operator_Guide.ent file |.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

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 5

DATA GRID のドキュメント 6

DATA GRID のダウンロード 7

多様性を受け入れるオープンソースの強化 8

第1章 DATA GRID OPERATOR のインストール 9
1.1. RED HAT OPENSHIFT への DATA GRID OPERATOR のインストール 9
1.2. コマンドラインからの DATA GRID OPERATOR のインストール 9
1.3. DATA GRID クラスターのアップグレード 11
アップグレードの通知 11

第2章 INFINISPAN CR を使い始める 12
2.1. INFINISPAN カスタムリソース (CR) 12
2.2. DATA GRID クラスターの作成 12
2.3. DATA GRID クラスターの検証 13
2.4. DATA GRID クラスターの停止および起動 14

第3章 DATA GRID サービスの設定 15
3.1. サービスの種別 15
3.1.1. Data Grid サービス 15
3.1.2. キャッシュサービス 15
3.2. DATA GRID サービス POD の作成 16
3.2.1. Data Grid サービス CR 16
3.3. キャッシュサービス POD の作成 18
3.3.1. キャッシュサービス CR 18
3.4. 自動スケーリング 20
3.4.1. 自動スケーリングの設定 21
3.5. ストレージリソースの割り当て 22
3.5.1. 永続ボリューム要求 23
3.6. JVM、CPU、およびメモリー 23
3.7. ログレベルの調整 23
3.7.1. ロギングの参照 24
3.8. DATA GRID リソースへのラベルの追加 25

第4章 認証の設定 27
4.1. デフォルトの認証情報 27
4.2. 認証情報の取得 27
4.3. カスタムユーザーの認証情報の追加 27
4.4. OPERATOR パスワードの変更 28
4.5. ユーザー認証の無効化 28

第5章 クライアント証明書認証の設定 30
5.1. クライアント証明書認証 30
5.2. クライアント証明書認証の有効化 30
5.3. クライアントトラストストアの提供 31
5.4. クライアント証明書の提供 31

第6章 暗号化の設定 33
6.1. RED HAT OPENSHIFT サービス証明書を使用した暗号化 33
6.2. TLS 証明書の取得 33
6.3. 暗号化の無効化 34

6.4. カスタム TLS 証明書の使用 34
6.4.1. カスタム暗号化シークレット 35

第7章 ユーザーロールとパーミッションの設定 37
7.1. セキュリティー承認の有効化 37
7.2. ユーザーロールとパーミッション 37
Data Grid Operator の認証情報 38
7.3. ロールおよびパーミッションのユーザーへの割り当て 38
7.4. カスタムロールおよびパーミッションの追加 39

第8章 DATA GRID へのネットワークアクセスの設定 40
8.1. 内部接続向けのサービスの取得 40
8.2. ロードバランサーを介した DATA GRID の公開 40
8.3. ノードポートを介した DATA GRID の公開 41
8.4. ルートを介した DATA GRID の公開 41
8.5. ネットワークサービス 42
8.5.1. 内部サービス 42
8.5.2. 外部サービス 42
8.5.3. クロスサイトサービス 43

第9章 DATA GRID サービスの監視 44
9.1. PROMETHEUS サービスモニターの作成 44
9.1.1. Prometheus サービスモニターの無効化 45
9.2. GRAFANA OPERATOR のインストール 45
9.3. GRAFANA データソースの作成 45
9.4. DATA GRID ダッシュボードの設定 47

第10章 クロスサイトレプリケーションの設定 48
10.1. DATA GRID OPERATOR を使用したクロスサイト接続の管理 48
10.1.1. サービスアカウントトークンの作成 48
10.1.2. サービスアカウントトークンの交換 49
10.1.3. クロスサイト接続を処理するための Data Grid Operator の設定 50
10.1.4. 管理対象クロスサイト接続のリソース 52
10.2. DATA GRID クラスターの手動接続 53
10.2.1. 手動のクロスサイト接続のリソース 55
10.3. 同じ OPENSHIFT クラスターでのサイトの設定 56

第11章 ANTI-AFFINITY による可用性の保証 58
11.1. ANTI-AFFINITY ストラテジー 58
フォールトトレランス 58
11.2. ANTI-AFFINITY の設定 58
11.2.1. anti-affinity ストラテジーの設定 59
さまざまな OpenShift ノードでの Pod のスケジュール 59
さまざまなノードが必要 59
複数の OpenShift ゾーンにまたがった Pod のスケジュール 60
複数のゾーンが必要 60

第12章 DATA GRID OPERATOR を使用したキャッシュの作成 61
12.1. テクノロジープレビュー 61
12.2. DATA GRID キャッシュ 61
12.3. キャッシュ CR 61
12.4. XML からキャッシュの作成 62
12.5. テンプレートからのキャッシュの作成 63
12.6. キャッシュへのバックアップの場所の追加 63
12.6.1. バックアップの場所をオフラインにする時のパフォーマンスに関する考慮事項 64

12.7. 永続キャッシュストアの追加 65

第13章 バッチ操作の実行 66
13.1. インラインバッチ操作の実行 66
13.2. バッチ操作の CONFIGMAP の作成 66
13.3. CONFIGMAP を使用したバッチ操作の実行 67
13.4. バッチステータスメッセージ 68
13.5. バッチ操作の例 68
13.5.1. キャッシュ 69
13.5.2. カウンター 69
13.5.3. Protobuf スキーマ 69
13.5.4. タスク 70

第14章 DATA GRID クラスターのバックアップおよび復元 71
14.1. BACKUP CR および RESTORE CR 71
14.2. DATA GRID クラスターのバックアップ 71
14.3. DATA GRID クラスターの復元 73
14.4. バックアップおよび復元のステータス 74
14.4.1. 失敗したバックアップおよび復元操作の処理 75

第15章 DATA GRID へのカスタムコードのデプロイ 76
15.1. DATA GRID クラスターへのコードアーティファクトのコピー 76
15.2. コードアーティファクトのダウンロード 78

第16章 DATA GRID クラスターからのクラウドイベントの送信 80
16.1. テクノロジープレビュー 80
16.2. クラウドイベント 80
16.3. クラウドイベントの有効化 80

第17章 リモートクライアント接続の確立 82
17.1. クライアント接続の詳細 82
17.2. DATA GRID キャッシュ 82
17.3. DATA GRID CLI の接続 83
17.4. DATA GRID コンソールへのアクセス 84
17.5. HOT ROD クライアント 84
17.5.1. Hot Rod クライアント設定 API 85
OpenShift の場合 85
OpenShift の外部 85
17.5.2. Hot Rod クライアントプロパティー 86
OpenShift の場合 86
OpenShift の外部 86
17.5.3. 証明書認証用の Hot Rod クライアントの設定 87
17.5.4. Hot Rod クライアントからのキャッシュの作成 87
プログラムでのキャッシュの作成 87
Hot Rod クライアントプロパティーの使用 88
17.6. REST API へのアクセス 88
17.7. キャッシュサービス POD へのキャッシュの追加 89
17.7.1. デフォルトのキャッシュ設定 89

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 8.2 ドキュメント
- Data Grid 8.2 コンポーネントの詳細
- Data Grid 8.2 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、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、弊社の CTO である Chris Wright のメッセージ を参照してください。

第1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。

1.1. 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 8.2.x の更新を取得します。

   承認ストラテジー

   8.2.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。

1.2. コマンドラインからの DATA GRID OPERATOR のインストール

OpenShift の **OperatorHub** を使用して Data Grid Operator をインストールする代わりに、**oc** クライアントを使用してサブスクリプションを作成します。

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: 8.2.x
installPlanApproval: Automatic
name: datagrid
source: redhat-operators
sourceNamespace: openshift-marketplace
EOF
```

注記

8.2.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
```

1.3. DATA GRID クラスターのアップグレード

Data Grid Operator は、新規のバージョンが利用可能になると、Data Grid クラスターを自動的にアップグレードできます。アップグレードのタイミングを制御したい場合は、手動でアップグレードを実行することもできます。

アップグレードの通知

Data Grid クラスターを手動でアップグレードし、Data Grid Operator サブスクリプションのチャネルを **8.1.x** から **8.2.x** にアップグレードした場合は、8.2.0 の問題に起因するデータ損失の可能性を防ぐために、速やかに最新の Data Grid 8.2.x バージョンへのアップグレードを適用する必要があります。詳細は、Data Grid Operator 8.2 リリースノート を参照してください。

第2章 INFINISPAN CR を使い始める

Data Grid Operator のインストール後に、OpenShift で Data Grid クラスターを作成する方法について説明します。

2.1. 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と設定、および StatefulSets や Services などの OpenShift リソースの管理に使用する **Infinispan** カスタムリソース (CR) を監視します。これにより、**Infinispan** CR は OpenShift 上の Data Grid へのプライマリーインターフェイスになります。

最小の **Infinispan** CR は以下のようになります。

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

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。

2.2. DATA GRID クラスターの作成

Data Grid Operator を使用して、2 つ以上の Data Grid Pod の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。
- **oc** クライアントがある。

手順

1. **Infinispan** CR の **spec.replicas** で、クラスター内の Data Grid Pod の数を指定します。
たとえば、以下のように **cr_minimal.yaml** ファイルを作成します。

```
$ cat > cr_minimal.yaml<<EOF
apiVersion: infinispan.org/v1
```

```
kind: Infinispan
metadata:
  name: example-infinispan
spec:
  replicas: 2
EOF
```

2. **Infinispan** CR を適用します。

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

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

```
$ oc get pods -w

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

次のステップ

replicas: の値の変更を試みてください。また、Data Grid Operator がクラスターをスケールアップまたはスケールダウンすることを監視してみてください。

2.3. DATA GRID クラスターの検証

Data Grid Pod が正常にクラスターを形成したことを確認します。

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: [example-infinispan-0, example-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/example-infinispan
```

または、以下のようにログからクラスタービューを取得できます。

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

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

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

2.4. 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。

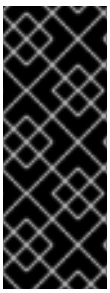
```
$ oc get infinispan example-infinispan -o=jsonpath='{.status.replicasWantedAtRestart}'
```

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3章 DATA GRID サービスの設定

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



重要

spec.service.type フィールドの値を指定しない場合、Data Grid Operator はデフォルトでキャッシサービス Pod を作成します。

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。

3.1. 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。

3.1.1. Data Grid サービス

必要に応じて、Data Grid サービス Pod のクラスターをデプロイします。

- 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。

3.1.2. キャッシュサービス

最小限の設定で、レイテンシーの低いデータストアが必要な場合は、キャッシサービス Pod のクラスターをデプロイします。

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。ただし、設定のオーバーヘッドなしに高パフォーマンスキャッシングを持つアプリケーションを迅速に提供する必要がある場合は、キャッシュサービス Pod を使用して以下を実行できます。

- データストレージの需要が増減したときに、容量に合わせて自動的にスケーリングします。
- 一貫性を確保するためにデータを同期的に分散します。
- クラスター全体でキャッシュの各エントリーを複製します。
- キャッシュエントリーを off-heap に保存し、JVM の効率化のためにエビクションを使用します。
- デフォルトのパーティション処理設定とデータの一貫性を確保します。

重要

Red Hat は、キャッシュサービス Pod の代わりに Data Grid サービス Pod をデプロイすることを推奨しています。

キャッシュサービスは、**Infinispan** CRD の次のバージョンで削除される予定です。Data Grid サービスは引き続き活発に開発されており、クラスターのアップグレードやデータの移行などの複雑な操作を自動化するための新機能と改善されたツールの恩恵を受け続けます。

3.2. 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、他のデータグリッド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: example-infinispan
spec:
  replicas: 2
  service:
    type: DataGrid
```

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

3.2.1. 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: example-infinispan
  annotations:
    infinispan.org/monitoring: 'true'
spec:
  replicas: 6
  service:
    type: DataGrid
    container:
      storage: 2Gi
      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: example-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: "1000m"
  memory: 1Gi
logging:
  categories:
    org.infinispan: debug
    org.jgroups: debug
    org.jgroups.protocols.TCP: error
    org.jgroups.protocols.relay.RELAY2: error
expose:
  type: LoadBalancer
affinity:
  podAntiAffinity:
    preferredDuringSchedulingIgnoredDuringExecution:
    - weight: 100
      podAffinityTerm:
        labelSelector:
          matchLabels:
            app: infinispan-pod
            clusterName: example-infinispan
            infinispan_cr: example-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.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。
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。

フィールド	説明
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.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。

3.3. 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。

手順

1. **spec.service.type: DataGrid** 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: example-infinispan
spec:
  replicas: 2
  service:
    type: Cache
```

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

3.3.1. 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: example-infinispan
```

```
  annotations:
    infinispan.org/monitoring: 'true'
spec:
  replicas: 2
  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"
    memory: 1Gi
  logging:
    categories:
      org.infinispan: trace
      org.jgroups: trace
  expose:
    type: LoadBalancer
  affinity:
    podAntiAffinity:
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 100
        podAffinityTerm:
          labelSelector:
            matchLabels:
              app: infinispan-pod
              clusterName: example-infinispan
              infinispan_cr: example-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。
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。

フィールド	説明
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。

3.4. 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。

3.4.1. 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。

3.5. 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. オプションで、必要に応じて **ephemeralStorage** および **storageClassName** フィールドを設定します。

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

3. 変更を適用します。

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。
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。

3.5.1. 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。

3.6. JVM、CPU、およびメモリー

Infinispan CR で JVM オプションを設定できるだけでなく、CPU とメモリーの割り当ても設定できます。

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

フィールド	説明
spec.container.extraJvmOpts	JVM オプションを指定します。
spec.container.cpu	ホスト CPU リソースを Data Grid Pod に割り当てます (CPU ユニットで測定)。
spec.container.memory	ホストメモリーを Data Grid Pod に割り当てます (バイト単位で測定)。

Data Grid Operator が Data Grid クラスターを作成すると、**spec.container.cpu** および **spec.container.memory** を使用して以下を行います。

- OpenShift に Data Grid ノードを実行するのに十分な容量があることを確認します。デフォルトでは、Data Grid Operator は OpenShift スケジューラーから **512Mi** の **memory**と、**0.5** の **CPU** を要求します。
- ノードリソースの使用を制限します。Data Grid Operator は、**cpu** および **memory** の値をリソース制限として設定します。

3.7. 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
```

3.7.1. 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3.1 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3.2 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。

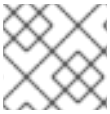
ガベージコレクション (GC) メッセージ

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"
```

3.8. DATA GRID リソースへのラベルの追加

キー/値のラベルを、Data Grid Operator が作成および管理する Pod およびサービスに割り当てます。これらのラベルは、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。



注記

Red Hat サブスクリプションラベルは、Data Grid Pod 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. Data Grid Operator が **metadata.annotations** を使用してリソースに割り当てるラベルを追加します。
3. **metadata.labels** を使用してラベルの値を追加します。
   a. **metadata.annotations.infinispan.org/targetLabels** フィールドでサービスに割り当てるラベルを指定します。
   b. **metadata.annotations.infinispan.org/podTargetLabels** フィールドで Pod に割り当てるラベルを指定します。
   c. **metadata.labels** フィールドでラベルの値を定義します。

```
apiVersion: infinispan.org/v1
kind: Infinispan
metadata:
  annotations:
    infinispan.org/targetLabels: svc-label1, svc-label2
    infinispan.org/podTargetLabels: pod-label1, pod-label2
  labels:
    svc-label1: svc-value1
    svc-label2: svc-value2
    pod-label1: pod-value1
    pod-label2: pod-value2
```

```
# The operator does not attach these labels to resources.
my-label: my-value
environment: development
```

4. **Infinispan** CR を適用します。

関連情報

- Labels and Selectors
- Labels:Kubernetes User Guide

第4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。

4.1. デフォルトの認証情報

Data Grid Operator は、以下のような名前の付いた認証シークレットに保存される base64 でエンコードされたデフォルトの認証情報を生成します。

Username	Secret 名	説明
developer	**example-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	**example-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。

4.2. 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 example-infinispan-generated-secret
```

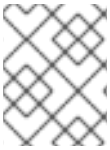
Base64 でデコードされた認証情報。

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

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

4.3. 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って **identities.yaml** ファイルを作成します。

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

4.4. 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、**example-infinispan-generated-operator-secret** シークレットの **password** キーを更新します。

```
oc patch secret example-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。

4.5. 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5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。

5.1. 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。

5.2. 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: Service
        certSecretName: tls-secret
        clientCert: Validate
        clientCertSecretName: example-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。

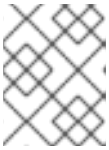
5.3. クライアントトラストストアの提供

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: example-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。

5.4. 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: example-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6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。

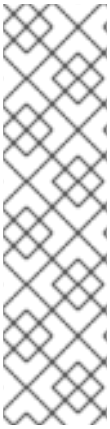
6.1. RED HAT OPENSHIFT サービス証明書を使用した暗号化

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: example-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。

6.2. 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。

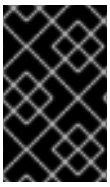
手順

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

6.3. 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
```

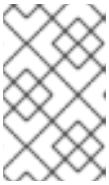
2. 変更を適用します。

6.4. 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。

6.4.1. カスタム暗号化シークレット

このトピックでは、カスタム暗号化シークレットのリソースについて説明します。

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: "LS0tLS1CRUdJTiBDRVl ..."
```

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。

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

第7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。

7.1. 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。

7.2. 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7.1 デフォルトのロールおよびパーミッション

ロール	パーミッション	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。

ロール	パーミッション	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

- How security authorization works (**Data Grid Security Guide**).

7.3. 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
```

7.4. 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8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。

8.1. 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: example-infinispan
```

手順

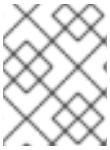
- 内部サービスが以下のように利用できることを確認します。

```
$ oc get services

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

8.2. ロードバランサーを介した DATA GRID の公開

ロードバランサーサービスを使用して、OpenShift 外部で実行しているクライアントが Data Grid クラスターを使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、ロードバランサー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。デフォルトのポートは **7900** です。

```
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

NAME                        TYPE           CLUSTER-IP    EXTERNAL-IP   PORT(S)
example-infinispan-external  LoadBalancer    192.0.2.24    hostname.com  11222/TCP
```

8.3. ノードポートを介した DATA GRID の公開

ノードポートサービスを使用して、ネットワークに 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

NAME                        TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)
example-infinispan-external  NodePort        192.0.2.24       <none>        11222:30000/TCP
```

8.4. ルートを介した DATA GRID の公開

パススルーの暗号化で OpenShift Route を使用して、ネットワーク上で Data Grid クラスターを利用できるようにします。

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

NAME                CLASS    HOSTS   ADDRESS   PORTS   AGE
example-infinispan   <none>   *               443     73s
```

ルートポート

ルートを作成すると、クライアント接続を受け入れるネットワーク上のポートが公開され、ポート **11222** でリッスンする Data Grid サービスにトラフィックがリダイレクトされます。

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。

Port	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。

8.5. 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。

8.5.1. 内部サービス

- Data Grid Pod が相互に検出し、クラスターを形成できるようにします。
- 同じ OpenShift namespace のクライアントから Data Grid エンドポイントへのアクセスを提供します。

サービス	ポート	Protocol	説明
<cluster_name>	**11222**	TCP	Data Grid エンドポイントへの内部アクセス
<cluster_name>-ping	**8888**	TCP	クラスターの検出

8.5.2. 外部サービス

OpenShift の外部のクライアントから、または異なる namespace のクライアントから Data Grid エンドポイントへのアクセスを提供します。

注記

Data Grid Operator を使用して外部サービスを作成する必要があります。これはデフォルトでは利用できません。

サービス	ポート	Protocol	説明
<cluster_name>-external	**11222**	TCP	Data Grid エンドポイントへの外部アクセス

8.5.3. クロスサイトサービス

Data Grid が、異なる場所にあるクラスター間でデータをバックアップできるようにします。

サービス	ポート	Protocol	説明
<cluster_name>-site	**7900**	TCP	クロスサイト通信向けの JGroups RELAY2 チャネル。

第9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クスを公開します。

注記

本書では、OpenShift Container Platform でモニター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。

9.1. 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クスをスクレープする 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ーリングを有効にします。

Operator がモニター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: example-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 メトリクス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

9.1.1. 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クス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: example-infinispan
  annotations:
    infinispan.org/monitoring: 'false'
```

2. 変更を適用します。

9.2. 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ー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。

9.3. GRAFANA データソースの作成

Grafana ダッシュボードで Data Grid メトリクス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ー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 メトリクス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

eyJhbGciOiJSUzI1NiIsImtpZCI6Imc4O...
```

   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。

9.4. 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: example-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、モニターリングキーを **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}"
```

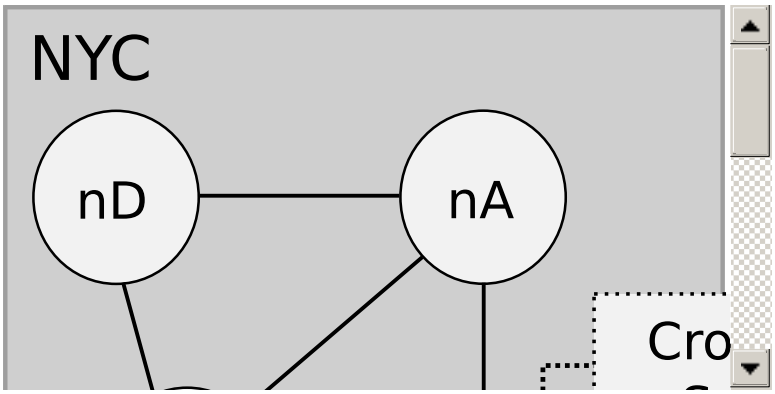
第10章 クロスサイトレプリケーションの設定

Data Grid クラスター間でデータをバックアップするようにクロスサイトレプリケーションを設定して、Data Grid Operator でのサービスの可用性を確保します。

10.1. DATA GRID OPERATOR を使用したクロスサイト接続の管理

1 つのデータセンターの Data Grid Operator は、Data Grid Operator が別のデータセンターで管理する Data Grid クラスターを検出できます。この検出により、Data Grid は自動的にクロスサイトビューを形成し、グローバルクラスターを作成できます。

以下の図は、Data Grid Operator がニューヨーク市 **NYC** のデータセンターで Data Grid クラスターを管理している例を示しています。Data Grid Operator は、ロンドンにある別のデータセンター **LON** でも、Data Grid クラスターを管理しています。



Data Grid Operator は、Kubernetes API を使用して、**NYC** と **LON** の OpenShift Container Platform クラスター間の安全な接続を確立します。その後、Data Grid Operator はクロスサイトレプリケーションサービスを作成し、Data Grid クラスターが複数の場所にまたがってデータをバックアップできるようにします。

重要

各 OpenShift クラスターの Data Grid Operator には、リモート Kubernetes API へのネットワークアクセスが必要です。

注記

自動接続を設定すると、Data Grid Operator が設定内のすべてのバックアップの場所を検出するまで、Data Grid クラスターの実行は開始されません。

各 Data Grid クラスターには、すべてのバックアップ要求を調整する 1 つのサイトマスターノードがあります。Data Grid Operator はサイトマスターノードを特定し、クロスサイトレプリケーションサービスを介したすべてのトラフィックがサイトマスターに送られるようにします。

現在のサイトマスターノードがオフラインになると、新規のノードがサイトマスターになります。Data Grid Operator は新しいサイトマスターノードを自動的に検出し、クロスサイトレプリケーションサービスを更新してバックアップ要求をノードに転送します。

10.1.1. サービスアカウントトークンの作成

バックアップの場所として動作する各 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。クラスターはこれらのトークンを使用して相互に認証するため、Data Grid Operator はクロスサイトレプリケーションサービスを作成できます。

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 lon
serviceaccount/lon created
```

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 system:serviceaccount:<namespace>:lon
```

4. ノードポート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があります。

```
$ oc adm policy add-cluster-role-to-user cluster-reader -z <service-account-name> -n
<namespace>
```

5. 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

- Using service accounts in applications

10.1.2. 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作成したら、それらを各バックアップの場所のシークレットに追加します。たとえば、**LON** で **NYC** のサービスアカウントトークンを追加します。**NYC** で **LON** のサービスアカウントトークンを追加します。

前提条件

- 各サービスアカウントからトークンを取得します。
  以下のコマンドを使用するか、OpenShift Web コンソールからトークンを取得します。

```
$ oc sa get-token lon

eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9...
```

手順

1. OpenShift クラスターにログインします。
2. 以下のコマンドを使用して、バックアップの場所のサービスアカウントトークンを追加します。

```
$ oc create secret generic <token-name> --from-literal=token=<token>
```

たとえば、**NYC** で OpenShift クラスターにログインし、以下のように **lon-token** シークレットを作成します。

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

3. 他の OpenShift クラスターで直前の手順を繰り返します。

10.1.3. クロスサイト接続を処理するための Data Grid Operator 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

- 各バックアップの場所のサービスアカウントトークンを含むシークレットを作成します。

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. **spec.service.sites.local.expose.type** フィールドの値を **NodePort** または **LoadBalancer** のいずれかに設定します。
4. 必要に応じて、以下のフィールドを使用してポートを設定します。
   - **spec.service.sites.local.expose.nodePort** (**NodePort** を使用する場合)
   - **spec.service.sites.local.expose.port** (**LoadBalancer** を使用する場合)
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: example-infinispan
spec:
  replicas: 3
  service:
    type: DataGrid
    sites:
      local:
        name: LON
        expose:
          type: LoadBalancer
          port: 65535
      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
  service:
    type: DataGrid
    sites:
      local:
        name: NYC
        expose:
          type: LoadBalancer
          port: 65535
      locations:
        - name: LON
          clusterName: example-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。

10.1.4. 管理対象クロスサイト接続のリソース

このトピックでは、Data Grid Operator が管理するクロスサイト接続のリソースについて説明します。

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

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。
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.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。この値は **NodePort** または **LoadBalancer** 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** 経由で Data Grid を公開する場合、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。

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。
service.sites.locations.url	バックアップの場所の Kubernetes API の URL を指定します。
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。

10.2. DATA GRID クラスターの手動接続

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。

同じ **Infinispan** CR の Data Grid クラスターには、自動接続と手動接続の両方を使用できます。ただし、Data Grid クラスターが、各サイトで同じ方法で接続を確立することを確認する必要があります。

前提条件

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 サービスの予測可能なネットワークの場所が必要です。

ネットワークの場所は、作成する前に知っておく必要があります。そのためには、以下が必要です。

- バックアップの場所として設定する予定の各 Data Grid クラスターのホスト名とポートがあります。
- OpenShift で実行されているリモート Data Grid クラスターの **<cluster-name>-site** サービスのホスト名があります。
  <cluster-name>-site サービスを使用して、Data Grid Operator が管理するクラスターと他のクラスター間でクロスサイトビューを形成する必要があります。

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. **spec.service.sites.local.expose.type** フィールドの値を **NodePort** または **LoadBalancer** のいずれかに設定します。

4. 必要に応じて、以下のフィールドを使用してポートを設定します。

   - **spec.service.sites.local.expose.nodePort** (**NodePort** を使用する場合)

   - **spec.service.sites.local.expose.port** (**LoadBalancer** を使用する場合)

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: example-infinispan
spec:
  replicas: 3
  service:
    type: DataGrid
    sites:
      local:
        name: LON
        expose:
          type: LoadBalancer
          port: 65535
      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: example-infinispan
spec:
  replicas: 2
  service:
    type: DataGrid
    sites:
      local:
        name: NYC
        expose:
          type: LoadBalancer
          port: 65535
      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
```

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。

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。

10.2.1. 手動のクロスサイト接続のリソース

このトピックでは、手動で管理するクロスサイト接続のリソースについて説明します。

```
spec:
  service:
    type: DataGrid
    sites:
      local:
        name: LON
        expose:
          type: LoadBalancer
          port: 65535
      locations:
      - name: NYC
        url: infinispan+xsite://infinispan-nyc.myhost.com:7900
```

フィールド	説明

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。
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.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。この値は **NodePort** または **LoadBalancer** 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** 経由で Data Grid を公開する場合、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.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。
service.sites.locations.url	**infinispan+xsite://<hostname>:<port>** の形式でバックアップの場所の静的 URL を指定します。デフォルトのポートは **7900** です。

10.3. 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 クラスターのノード間でバックアップするように Data Grid を設定できます。

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
      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11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。

11.1. 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。

11.2. 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。

11.2.1. 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。

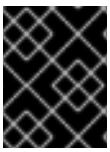
```
spec:
  affinity:
    podAntiAffinity:
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 100
        podAffinityTerm:
          labelSelector:
            matchLabels:
              app: infinispan-pod
              clusterName: <cluster_name>
              infinispan_cr: <cluster_name>
          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12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。



重要

Data Grid Operator を使用したキャッシュの作成は、テクノロジープレビューとして利用可能です。

12.1. テクノロジープレビュー

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。

12.2. 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。

12.3. キャッシュ CR

Cache CR で Data Grid キャッシュを設定する方法の詳細を確認してください。

Cache CR を使用する場合、以下のルールが適用されます。

- **Cache** CR は Data Grid サービス Pod にのみ適用されます。
- 各 **Cache** CR に 1 つのキャッシュを作成できます。

- **Cache** CR にテンプレートと XML 設定の両方が含まれる場合、Data Grid Operator はテンプレートを使用します。
- OpenShift Web コンソールのキャッシュを編集すると、変更はユーザーインターフェイスに反映されますが、Data Grid クラスターには反映されません。キャッシュの編集はできません。キャッシュ設定を変更するには、まずコンソールまたは CLI でキャッシュを削除してからキャッシュを再作成する必要があります。
- OpenShift Web コンソールで **Cache** CR を削除しても、Data Grid クラスターからキャッシュは削除されません。コンソールまたは CLI を使用してキャッシュを削除する必要があります。

注記

以前のバージョンでは、キャッシュの作成時に Data Grid Operator がクラスターにアクセスできるように、認証情報をシークレットに追加する必要がありました。

これは不要になりました。Data Grid Operator は、**operator** ユーザーおよび対応するパスワードを使用してキャッシュ操作を実行します。

12.4. XML からキャッシュの作成

有効な **infinispan.xml** 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、以下の手順を実施します。

手順

1. XML キャッシュ設定が含まれる **Cache** CR を作成します。
   a. **metadata.name** フィールドで **Cache** CR の名前を指定します。
   b. 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。
   c. **spec.name** フィールドで、キャッシュに名前を付けます。

      注記

      XML 設定の **name** 属性は無視されます。**spec.name** フィールドのみが生成されるキャッシュに適用されます。

   d. **spec.template** フィールドで XML キャッシュ設定を追加します。

```
apiVersion: infinispan.org/v2alpha1
kind: Cache
metadata:
  name: mycachedefinition
spec:
  clusterName: example-infinispan
  name: mycache
  template: <distributed-cache name="mycache" mode="SYNC"><persistence><file-store/></persistence></distributed-cache>
```

2. **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
```

12.5. テンプレートからのキャッシュの作成

キャッシュテンプレートを使用して Data Grid サービスクラスターにキャッシュを作成するには、以下の手順を実施します。

前提条件

- キャッシュに使用するキャッシュテンプレートを特定します。
  使用可能なテンプレートの一覧は、Data Grid コンソールにあります。

手順

1. 使用するテンプレートの名前を指定する **Cache** CR を作成します。
   a. **metadata.name** フィールドで **Cache** CR の名前を指定します。
   b. 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。
   c. **spec.name** フィールドで、キャッシュに名前を付けます。
   d. **spec.template** フィールドで、キャッシュテンプレートを指定します。
      以下の例では、**org.infinispan.DIST_SYNC** キャッシュテンプレートから mycache という名前のキャッシュを作成します。

```
apiVersion: infinispan.org/v2alpha1
kind: Cache
metadata:
  name: mycachedefinition
spec:
  clusterName: example-infinispan
  name: mycache
  templateName: org.infinispan.DIST_SYNC
```

2. **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
```

12.6. キャッシュへのバックアップの場所の追加

Data Grid クラスターをクロスサイトのレプリケーションを実行するように設定するには、バックアップの場所をキャッシュ設定に追加できます。

手順

1. リモートサイトをバックアップの場所として指定するキャッシュ設定を作成します。
   Data Grid は、キャッシュ名に基づいてデータを複製します。このため、キャッシュ設定のサイト名は、**Infinispan** CR のサイト名 **spec.service.sites.local.name** と一致する必要があります。

2. **take-offline** 要素を使用して、バックアップの場所を自動的にオフラインになるように設定します。

   a. **min-wait** 属性で、バックアップの場所がオフラインになるまでの時間をミリ秒単位で設定します。

3. その他の有効なキャッシュ設定を定義します。

4. バックアップの場所を、グローバルクラスターのすべてのサイトの名前付きキャッシュに追加します。
   たとえば、NYC のバックアップとして LON を追加する場合、LON のバックアップとして NYC を追加する必要があります。

以下の設定例は、キャッシュのバックアップの場所を示しています。

- NYC

```
<distributed-cache name="customers">
  <encoding media-type="application/x-protostream"/>
  <backups>
    <backup site="LON" strategy="SYNC">
      <take-offline min-wait="120000"/>
    </backup>
  </backups>
</distributed-cache>
```

- LON

```
<replicated-cache name="customers">
  <encoding media-type="application/x-protostream"/>
  <backups>
    <backup site="NYC" strategy="ASYNC" >
      <take-offline min-wait="120000"/>
    </backup>
  </backups>
</replicated-cache>
```

関連情報

- Data Grid Guide to Cross-Site Replication

12.6.1. バックアップの場所をオフラインにする時のパフォーマンスに関する考慮事項

リモートサイトが使用できなくなった場合、バックアップの場所は自動的にオフラインになります。これにより、Pod がオフラインのバックアップの場所にデータを複製しようとする (エラーが発生してクラスターのパフォーマンスに影響を及ぼす可能性がある) ことを防ぎます。

バックアップの場所がオフラインになるまでの待機時間を設定できます。経験則として、1、2 分が適しています。ただし、さまざまな待機期間をテストし、それらのパフォーマンスへの影響を評価して、デプロイメントに適した値を決定する必要があります。

たとえば、OpenShift がサイトマスター Pod を終了すると、Data Grid Operator が新規のサイトマスターを選択するまでの短時間、そのバックアップの場所は利用できなくなります。この場合、最小の待機時間が十分な長さではない場合、バックアップの場所はオフラインになります。そこで、これらの

バックアップの場所をオンライン状態にし、状態転送操作を実行して、データが同期していることを確認する必要があります。

同様に、最小の待機時間が長すぎる場合は、バックアップの試行が失敗することでノードの CPU 使用率が増大し、パフォーマンスが低下する可能性があります。

12.7. 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>
```

第13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。

13.1. 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: example-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 の **status.Phase** フィールドをチェックして、操作が正常に完了したことを確認します。

13.2. 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
```

13.3. 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: example-infinispan
  configMap: mybatch-config-map
EOF
```

3. **Batch** CR を適用します。

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

4. **Batch** CR の **status.Phase** フィールドをチェックして、操作が正常に完了したことを確認します。

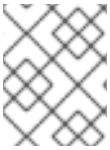
13.4. バッチステータスメッセージ

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。

13.5. 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。

13.5.1. 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
```

13.5.2. 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
```

13.5.3. 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
```

13.5.4. 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14章 DATA GRID クラスターのバックアップおよび復元

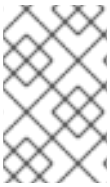
Data Grid Operator を使用すると、障害復旧およびクラスター間で Data Grid リソースを移行するために、Data Grid クラスターの状態をバックアップおよび復元できます。

14.1. 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。

14.2. 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

- **Infinispan** CR を **spec.service.type: DataGrid** を使用して作成します。
- 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
```

14.3. 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、または 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。

14.4. 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。

Status	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。
実行中	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。

Status	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。

14.4.1. 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15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。

15.1. 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-rhel8:8.2
      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: example-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)

15.2. 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展開します。

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。このダウンロードは、Data Grid Operator が Pod の終了後に Pod を再作成する際にも発生します。

前提条件

- HTTP または FTP サーバーでコードアーティファクトをホストします。

手順

1. **spec.dependencies.artifacts** フィールドを **Infinispan** CR に追加します。
   a. **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。
   b. 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。
   c. 必要に応じて **spec.dependencies.artifacts.type** フィールドでファイルタイプを設定します。
   ファイルタイプが URL に含まれていない場合や、ファイルタイプが URL のエクステンションと実際には異なる場合は、明示的にファイルタイプを設定する必要があります。

注記

type: file を設定する場合、Data Grid Operator は、ファイルをファイルシステムに抽出せずにそのままダウンロードします。

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

2. 変更を適用します。

第16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。

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としてご利用いただけます。

16.1. テクノロジープレビュー

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。

16.2. 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。
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。

16.3. 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 サーバーを一覧表示します。

   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17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。

17.1. クライアント接続の詳細

Data Grid に接続する前に、以下の情報を取得する必要があります。

- サービスホスト名
- 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

サービスのホスト名

サービスホスト名は、ネットワーク上で Data Grid を公開する方法や、クライアントが OpenShift 上で実行されているかによって異なります。

OpenShift で実行しているクライアントの場合、Data Grid Operator が作成する内部サービス名を使用できます。

OpenShift の外部で実行しているクライアントの場合、ロードバランサーを使用する場合はサービスホスト名はロケーション URL になります。ノードポートサービスの場合、サービスホスト名はノードホスト名になります。ルートの場合、サービスホスト名は、カスタムホスト名か、システム定義のホスト名のいずれかになります。

ポート

OpenShift 上のクライアント接続、およびロードバランサー経由のクライアント接続は、ポート **11222** を使用します。

ノードポートサービスは、**30000** から **60000** までの範囲で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

17.2. 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。

17.3. DATA GRID CLI の接続

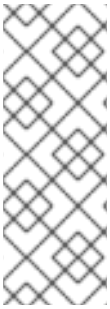
コマンドラインインターフェイス (CLI) を使用して Data Grid クラスターに接続し、管理操作を実行します。

前提条件

- OpenShift で Data Grid クラスターに接続できるように CLI ディストリビューションをダウンロードしている。

Data Grid CLI は、サーバーディストリビューションで、またはネイティブ実行可能ファイルとして使用できます。

サーバーディストリビューションの一部として CLI をダウンロードおよびインストールする方法の詳細は、**Getting Started with Data Grid Server**の手順に従います。ネイティブ CLI の場合は、ZIP ダウンロードに含まれる **README** ファイルのインストール手順に従う必要があります。



注記

Data Grid ノードへのリモートシェルを開き、CLI にアクセスすることができます。

```
$ oc rsh example-infinispan-0
```

ただし、この方法で CLI を使用すると、コンテナーに割り当てられたメモリーが使用されます。これにより、メモリー不足の例外が発生する可能性があります。

手順

1. Data Grid クラスターへの CLI コネクションを作成します。

   サーバーのディストリビューションの使用

   ```
   $ bin/cli.sh -c https://$SERVICE_HOSTNAME:$PORT --trustall
   ```

   ネイティブ CLI の使用

   ```
   $ ./redhat-datagrid-cli -c https://$SERVICE_HOSTNAME:$PORT --trustall
   ```

   $SERVICE_HOSTNAME:$PORT を、ネットワーク上で Data Grid を使用できるホスト名とポートに置き換えます。

2. プロンプトが表示されたら、Data Grid の認証情報を入力します。

3. 必要に応じて CLI 操作を実行します。以下に例を示します。

   a. **ls** コマンドを使用して、クラスターに設定されたキャッシュを一覧表示します。

   ```
   [//containers/default]> ls caches
   mycache
   ```

   b. **describe** コマンドでキャッシュ設定を表示します。

   ```
   [//containers/default]> describe caches/mycache
   ```

関連情報

- Getting Started with Data Grid Server
- Data Grid Software Downloads
- Using the Data Grid Command Line Interface

17.4. 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、ロードバランサーサービスを設定するか、またはルートを作成します。

手順

- **$SERVICE_HOSTNAME:$PORT** で任意のブラウザーからコンソールにアクセスします。
  $SERVICE_HOSTNAME:$PORT を、ネットワーク上で Data Grid を使用できるホスト名とポートに置き換えます。

17.5. 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

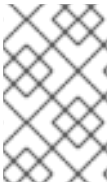
クライアントインテリジェンスとは、クライアントが Data Grid Pod にリクエストを見つけて送信できるように、Hot Rod プロトコルが提供するメカニズムを指します。

OpenShift 上で実行されている Hot Rod 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。デフォルトのインテリジェンスである **HASH_DISTRIBUTION_AWARE** が推奨されます。これにより、クライアントは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なり、パフォーマンスが向上します。

OpenShift の外部で実行される Hot Rod クライアントは、**BASIC** インテリジェンスを使用する必要があります。

17.5.1. 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

$SERVICE_HOSTNAME:$PORT は、Data Grid クラスターへのアクセスが許可されるホスト名およびポートを示します。これらの変数は、お使いの環境の実際のホスト名およびポートに置き換える必要があります。

OpenShift の場合
OpenShift で実行されている Hot Rod クライアントは、以下の設定を使用できます。

```
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("$SERVICE_HOSTNAME")
            .port(ConfigurationProperties.DEFAULT_HOTROD_PORT)
          .security().authentication()
            .username("username")
            .password("changeme")
            .realm("default")
            .saslQop(SaslQop.AUTH)
            .saslMechanism("SCRAM-SHA-512")
          .ssl()
            .sniHostName("$SERVICE_HOSTNAME")
            .trustStoreFileName("/var/run/secrets/kubernetes.io/serviceaccount/service-ca.crt")
            .trustStoreType("pem");
```

OpenShift の外部
OpenShift の外部で実行されている Hot Rod クライアントは、以下の設定を使用できます。

```
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("$SERVICE_HOSTNAME")
            .port("$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);
```

17.5.2. Hot Rod クライアントプロパティー

Hot Rod クライアント接続は、アプリケーションクラスパスの **hotrod-client.properties** ファイルで設定できます。

注記

$SERVICE_HOSTNAME:$PORT は、Data Grid クラスターへのアクセスが許可されるホスト名およびポートを示します。これらの変数は、お使いの環境の実際のホスト名およびポートに置き換える必要があります。

OpenShift の場合
OpenShift で実行されている Hot Rod クライアントは、以下のプロパティーを使用できます。

```
# Connection
infinispan.client.hotrod.server_list=$SERVICE_HOSTNAME:$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
OpenShift の外部で実行されている Hot Rod クライアントは、以下のプロパティーを使用できます。

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

17.5.3. 証明書認証用の Hot Rod クライアントの設定

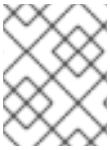
クライアント証明書認証を有効にすると、Data Grid との接続をネゴシエートする際に、クライアントは有効な証明書を提示する必要があります。

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");
```

17.5.4. 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
```

17.6. 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、ロードバランサーサービスを設定するか、またはルートを作成します。

手順

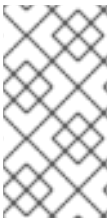
- **$SERVICE_HOSTNAME:$PORT/rest/v2** の任意の HTTP クライアントで REST API にアクセスします。
  $SERVICE_HOSTNAME:$PORT を、ネットワーク上で Data Grid を使用できるホスト名とポートに置き換えます。

関連情報

- Data Grid REST API

17.7. 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
```

17.7.1. 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。