Red Hat Data Grid 8.1

OpenShift での Data Grid の実行

OpenShift での Data Grid サービスの設定および実行

Last Updated: 2020-10-12

Red Hat Data Grid 8.1 OpenShift での Data Grid の実行

OpenShift での Data Grid サービスの設定および実行

法律上の通知

概要

Data Grid Operator は運用上の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

第1章 RED HAT DATA GRID

Data Grid は、高パフォーマンスで分散型の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
異なるオブジェクトをキーと値のペアとして保存する柔軟性。

グリッドベースのデータストレージ
クラスター全体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。

柔軟スケーリング
サービスの中断なしに要求を満たすようにノードの数を動的に調整します。

データの相互運用性
異なるエンドポイントからグリッドにデータを保存、取得、およびクエリーします。

1.1. 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から入手できます。

- Data Grid 8.1 ドキュメント
- Data Grid 8.1 コンポーネントの詳細
- Data Grid 8.1 のサポート対象の設定

1.2. 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 ソフトウェアへアクセスおよびダウンロードするには、Red Hat アカウントが必要です。

第2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、Data Grid クラスターを作成および管理します。

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

Data Grid Operator は、OpenShift 上の Operator Lifecycle Manager (OLM) 用にパッケージ化され、Data Grid バージョンの選択や自動更新の受信を可能にします。

前提条件

- OpenShift で実行される **OperatorHub** へのアクセス。OpenShift Container Platform などの一部の OpenShift 環境では、管理者のクレデンシャルが必要な場合があります。
- Data Grid Operator の OpenShift プロジェクト。

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、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

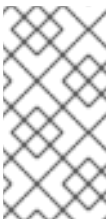
   Data Grid Operator を特定の OpenShift namespace にインストールします。Data Grid Operator を複数の namespace にインストールすることはできません。

   更新チャネル

   Data Grid Operator バージョンの更新にサブスクライブします。

   承認ストラテジー

   新しい Data Grid バージョンが利用できるようになると、手動による更新のインストールや、Data Grid Operator による自動インストールが可能になります。

   注記

   自動更新は最初に Data Grid Operator に適用され、次に各 Data Grid ノードに適用されます。Data Grid Operator は一度に 1 つのノードを更新し、各ノードを正常にシャットダウンしてから次のノードに移動する前に、更新されたバージョンでこれをオンラインに戻します。

6. **Subscribe** を選択して Data Grid Operator をインストールします。

第3章 DATA GRID OPERATOR を使い始める

Data Grid Operator を使用すると、Data Grid クラスターを作成、設定 および管理できます。

前提条件

- Data Grid Operator をインストールします。
- **oc** クライアントが存在します。

3.1. INFINISPAN カスタムリソース (CR)

Data Grid Operator は、タイプ **Infinispan** の新しいカスタムリソース (CR) を追加します。これにより、Data Grid クラスターを OpenShift で複雑なユニットとして処理できます。

Data Grid Operator は、Data Grid クラスターのインスタンス化および設定に使用する **Infinispan** カスタムリソース (CR) を監視し、StatefulSets や Services などの OpenShift リソースを管理します。よって、**Infinispan** CR は OpenShift 上の Data Grid への主要なインターフェースです。

最小の **Infinispan** CR は以下のとおりです。

```
apiVersion: infinispan.org/v1 1
kind: Infinispan 2
metadata:
  name: example-infinispan 3
spec:
  replicas: 4
```

1. Infinispan API バージョンを宣言します。
2. Infinispan CR を宣言します。
3. Data Grid クラスターの名前を指定します。
4. Data Grid クラスターのノード数を指定します。

3.2. 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 ノードのクラスターを作成します。

手順

1. クラスター内の Data Grid ノードの数を **Infinispan** CR の **spec.replicas** で指定します。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 による Data Grid ノードの作成を監視します。

```
$ 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: の値を変更し、Data Grid Operator がクラスターをスケールアップまたはダウンするのを監視します。

3.3. DATA GRID クラスターの検証

ログメッセージを確認し、Data Grid ノードがクラスター化されたビューを受信することを確認します。

手順

- 次のいずれかを行います。
  - ログからクラスタービューを取得します。

```
$ 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]
```

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

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

応答は、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

oc wait に自動スクリプトの **wellFormed** 条件を使用します。

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

第4章 DATA GRID サービスの設定

Data Grid Operator を使用して Cache Service または Data Grid Service ノードを作成します。

4.1. サービスのタイプ

サービスは Data Grid サーバーイメージに基づいたステートフルなアプリケーションで、柔軟で堅牢なインメモリーデータストレージを提供します。

Cache Service と Data Grid Service の 2 つのサービスタイプがあります。

4.1.1. Cache Service

最小設定が必要で、揮発性で低レイテンシーなデータストアを Data Grid が提供する場合は、Cache Service を使用します。

デフォルトでは、Cache Service ノードは以下を行います。

- クラスター全体でデータを同期的に分散し、一貫性を確保します。
- キャッシュエントリーの単一コピーを維持してサイズを減らします。
- キャッシュエントリーをオフヒープに格納し、JVM 効率化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でデータの一貫性を確保します。

4.1.2. Data Grid Service

以下を行う場合は Data Grid Service を使用します。

- 分散設定またはレプリケートされた設定を使用して、ランタイム時に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、永続ボリュームにデータを保存します。
- クロスサイトレプリケーション
- 検索などの高度なデータグリッド機能を利用します。

4.2. サービスタイプの設定

デフォルトでは、Data Grid Operator は Cache Service ノードを使用する Data Grid クラスターを作成します。キャッシュサービスを使用する場合は、サービスタイプを設定する必要はありません。Data Grid Service ノードを作成する必要がある場合は、**.spec.service.type** リソースを定義します。

手順

- **DataGrid** を **Infinispan** CR の **spec.service.type** の値として指定します。

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

注記

Data Grid クラスターの作成後に **.spec.service.type** を変更することはできません。

たとえば、Cache Service ノードのクラスターを作成した後に Data Grid Service ノードに変更したい場合を考えてみましょう。この場合、Cache Service ノードの既存クラスターを削除し、Data Grid Service ノードで新規クラスターを作成する必要があります。

リファレンス

- キャッシュサービスリソース
- Data Grid サービスリソース

4.3. CACHE SERVICE ノードの設定

Cache Service ノードに特定のプロパティーを設定できます。

4.3.1. デフォルトのキャッシュ設定

Cache Service ノードは以下のキャッシュ設定を使用します。

```
<distributed-cache name="default" 1
                   mode="SYNC" 2
                   owners="2"> 3
  <memory storage="OFF_HEAP" 4
          max-size="<maximum_size_in_bytes>" 5
          when-full="REMOVE"> 6
  </memory>
  <partition-handling when-split="ALLOW_READ_WRITES" 7
                      merge-policy="REMOVE_ALL"/> 8
</distributed-cache>
```

1. キャッシュインスタンスを「default」として指定します。
2. クラスター全体にデータを保存するには、同期分散を使用します。
3. クラスター全体のキャッシュエントリーごとにレプリカを 1 つ設定します。
4. キャッシュエントリーをバイト単位でネイティブメモリー (off-heap) に格納します。
5. データコンテナーの最大サイズをバイト単位で指定します。Data Grid Operator はノードの作成時に最大サイズを計算します。
6. 新規エントリーの追加時に古いエントリーを削除し、スペースを作ります。
7. 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指定します。
8. Data Grid が競合を検出するときに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。

4.3.2. 自動スケーリングの設定

Cache Service ノードを使用してクラスターを作成すると、Data Grid Operator はメモリー使用量に基づいて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。

Data Grid Operator は、Data Grid クラスターの各ノードのメモリー使用状況を監視します。Cache Service クラスターに追加容量が必要であることを検出すると、Data Grid Operator はエントリーをエビクトする代わりに新規ノードを作成します。同様に、メモリー使用量が特定のしきい値を下回ることを検知すると、Data Grid Operator はノードをシャットダウンします。

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。
2. 以下の例のように、クラスターのメモリー使用のしきい値およびノード数を設定します。

```
spec:
  ...
  autoscale: 1
    maxMemUsagePercent: 70 2
    maxReplicas: 5 3
    minMemUsagePercent: 30 4
    minReplicas: 2 5
```

1. 自動スケーリングを有効にします。
2. 各ノードのメモリー使用量に対して最大しきい値をパーセンテージで設定します。Data Grid Operator は、クラスター内のすべてのノードがしきい値に達したことを検出すると、可能な場合は新しいノードを作成します。Data Grid Operator が新規ノードを作成できない場合、メモリー使用量が 100% に達するとエビクションが実行されます。
3. クラスターのノードの最大数を定義します。
4. クラスター全体のメモリー使用量に対して、最小しきい値をパーセンテージとして設定します。Data Grid Operator がメモリー使用量が最小値を下回ることを検知すると、ノードがシャットダウンします。
5. クラスターのノードの最小数を定義します。

3. 変更を適用します。

4.3.3. 所有者数の設定

所有者の数は、Data Grid クラスター全体にまたがるキャッシュエントリーのコピー数を制御します。

手順

1. 以下のように、**Infinispan** CR の **spec.replicationFactor** リソースで所有者の数を指定します。

```
spec:
  ...
  replicationFactor: 3 1
```

1. 各キャッシュエントリーに対して 3 つのレプリカを設定します。

2. 変更を適用します。

第5章 DATA GRID クラスターの停止および起動

Data Grid Operator を使用して Data Grid クラスターを停止し、起動します。

キャッシュ定義

キャッシュサービスと Data Grid Service は永続ボリュームに永続的なキャッシュ定義を保存するため、クラスターの再起動後も利用できます。

データ

ファイルベースのキャッシュストアを追加すると、クラスターのシャットダウン時にすべてのキャッシュエントリーを永続ストレージに書き込むことができます。

5.1. DATA GRID クラスターのシャットダウン

Cache サービスノードをシャットダウンすると、キャッシュのすべてのデータが削除されます。Data Grid Service ノードの場合は、永続ボリュームにすべてのデータを保持できるように Data Grid Service ノードのストレージサイズを設定する必要があります。

利用可能なコンテナーストレージが Data Grid Service ノードで利用可能なメモリー量よりも小さい場合、Data Grid は以下の例外をログに書き込み、シャットダウン中にデータの損失が発生します。

```
WARNING: persistent volume size is less than memory size. Graceful shutdown may not work.
```

手順

- **replicas** の値を **0** に設定し、変更を適用します。

```
spec:
  replicas: 0
```

5.2. DATA GRID クラスターの再起動

シャットダウン後に Data Grid クラスターを再起動するには、以下の手順を実行します。

前提条件

Data Grid Service ノードの場合は、シャットダウンする前と同じ数のノードでクラスターを再起動する必要があります。たとえば、6 つのノードのクラスターをシャットダウンします。このクラスターを再起動する際に、**spec.replicas** の値として 6 を指定する必要があります。

これにより、Data Grid はクラスター全体のデータの分散を復元できます。クラスターのすべてのノードが実行中である場合、ノードを追加または削除できます。

Data Grid クラスターの適切なノードの数は以下のとおりです。

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

手順

- **spec.replicas** の値を、クラスターの適切なノード数に設定します。以下に例を示します。

```
spec:
  replicas: 6
```

第6章 コンテナー仕様の調整

CPU およびメモリーリソースの割り当て、JVM オプションの指定、および Data Grid ノードのストレージを設定できます。

6.1. JVM、CPU、およびメモリーリソース

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

1 JVM オプションを指定します。

2 CPU ユニットで測定されるノードにホスト CPU リソースを割り当てます。

3 ホストのメモリーリソースをバイト単位で測定されたノードに割り当てます。

Data Grid Operator が Data Grid クラスターを作成する場合、**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、Data Grid Operator は OpenShift スケジューラーから **512Mi** の **memory** および **0.5** の **cpu** を要求します。
- ノードのリソースの使用を制限します。Data Grid Operator は、リソース制限として **cpu** および **memory** の値を設定します。

ガベッジコレクションのロギング

デフォルトでは、Data Grid Operator はガベージコレクション (GC) メッセージをログに記録しません。必要に応じて以下の JVM オプションを追加して、GC メッセージを標準出力(stdout) に転送できます。

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

6.2. ストレージリソース

デフォルトでは、Data Grid Operator は Cache Service ノードと Data Grid Service ノードの両方のストレージに **1Gi** を割り当てます。ただし、Data Grid Service ノードのストレージサイズを設定できます。

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

1 Data Grid Service ノードのストレージサイズを設定します。

リファレンス

- Persistent Volume Claim (永続ボリューム要求、PVC)

第7章 DATA GRID へのネットワークアクセスの設定

ネットワークで Data Grid クラスターをクライアント接続が使用できるようにします。

7.1. 内部接続のサービスの取得

デフォルトでは、Data Grid Operator は、OpenShift で稼働す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。

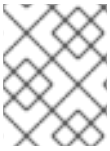
```
$ oc get services

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

リファレンス

- ネットワークサービス

7.2. ロードバランサーによる DATA GRID の公開

ロードバランサーサービスを使用して、OpenShift の外部で実行されているクライアントが Data Grid クラスターを利用できるようにします。



注記

暗号化されていない Hot Rod クライアント接続で Data Grid にアクセスするには、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** が含まれるようにします。
2. **spec.expose.type** で **LoadBalancer** をサービスタイプとして指定します。

```
spec:
  ...
  expose:
    type: LoadBalancer (1)
    nodePort: 30000 (2)
```

(1) ポート **11222** のロードバランサーサービスを介して、ネットワーク上の Data Grid を公開します。

2 必要に応じて、ロードバランサーサービスがトラフィックを転送するノードポートを定義します。

3. 変更を適用します。
4. **-external** サービスが利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      <none>        11222/TCP
```

7.3. ノードポートによる DATA GRID の公開

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

手順

1. **Infinispan** CR に **spec.expose** が含まれるようにします。
2. **spec.expose.type** で **NodePort** をサービスタイプとして指定します。

```
spec:
  ...
  expose:
    type: NodePort 1
    nodePort: 30000 2
```

1 ノードポートサービスを介して、ネットワーク上で Data Grid を公開します。

2 Data Grid が公開されるポートを定義します。ポートを定義しない場合は、プラットフォームがポートを選択します。

3. 変更を適用します。
4. **-external** サービスが利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
```

7.4. ROUTE による DATA GRID の公開

パススルー暗号化で OpenShift Route を使用して、Data Grid クラスターをネットワーク上で利用可能にします。

手順

1. **Infinispan** CR に **spec.expose** が含まれるようにします。
2. **spec.expose.type** で **Route** をサービスタイプとして指定します。

3. **spec.expose.host** でホスト名を指定します。

```
spec:
  ...
  expose:
    type: Route (1)
    host: www.example.org (2)
```

(1) OpenShift Route を介して、ネットワーク上で Data Grid を公開します。

(2) Data Grid が公開されるホスト名を指定します。

4. 変更を適用します。

5. 使用できることを確認します。

```
$ oc get routes

NAME                CLASS    HOSTS   ADDRESS   PORTS   AGE
example-infinispan  <none>   *                 80      73s
```

第8章 認証の設定

アプリケーションユーザーを、Data Grid クラスターで認証する必要があります。Data Grid Operator によってデフォルトのクレデンシャルが生成されるか、独自のクレデンシャルを追加できます。

8.1. デフォルトのクレデンシャル

Data Grid Operator は、**example-infinispan-generated-secret** という名前の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クレデンシャルをを生成します。

ユーザー名	説明
developer	デフォルトのアプリケーションユーザー。
operator	Data Grid クラスターと対話する内部ユーザー。

8.2. クレデンシャルの取得

認証シークレットからクレデンシャルを取得して Data Grid クラスターにアクセスします。

手順

- 以下の例のように、認証シークレットからクレデンシャルを取得します。

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

base64 でクレデンシャルをデコードします。

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

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

8.3. IDENTITIES ファイル

Data Grid アプリケーションユーザーと Operator ユーザーのクレデンシャルを **identities.yaml** ファイルに定義できます。**identities.yaml** に追加したすべてのユーザーが Data Grid クラスターおよびデータにアクセスできます。

重要

identities.yaml に Operator ユーザーが含まれる必要があります。

identities ファイルの例

```
credentials:
```

```
- username: testuser
  password: testpassword
- username: operator
  password: supersecretoperatorpassword
```

8.4. カスタムクレデンシャルの追加

カスタムクレデンシャルで、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。

手順

1. **identities.yaml** ファイルにカスタムクレデンシャルを定義します。
2. 以下のように **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. 認証シークレットを **Infinispan** CR の **spec.security.endpointSecretName** で指定し、変更を適用します。

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

1 クレデンシャルが含まれる認証シークレットの名前を指定します。

検証

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。Data Grid Operator が変更を適用するときに Data Grid クラスターを監視できます。

```
$ oc get pods -w
```

Data Grid クラスターの実行中に、以下のようにカスタムクレデンシャルを検証します。

1. Data Grid ノードに対してリモートシェルを開きます。以下に例を示します。

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

2. クレデンシャルが **users.properties** ファイルに存在することを確認します。

```
$ cat /opt/datagrid/server/conf/users.properties
```

3. リモートシェルを終了します。

```
$ exit
```

第9章 DATA GRID 接続の保護

Red Hat OpenShift サービス証明書またはカスタム TLS 証明書を使用して、クライアントと Data Grid ノード間の接続を暗号化します。

9.1. RED HAT OPENSHIFT SERVICE 証明書による暗号化

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。その後、Data Grid Operator は証明書および鍵をシークレットに保存し、それらを取得し、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 1
      certSecretName: example-infinispan-cert-secret 2
```

1 Red Hat OpenShift Service を指定します。

2 サービス証明書 **tls.crt** およびキー **tls.key** が含まれるシークレットを PEM 形式で指定します。名前を指定しない場合、Data Grid Operator は **<cluster_name>-cert-secret** を使用します。

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 内部でのみ完全に信頼されます。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 年間有効であり、期限が切れる前に自動的に置き換えられます。

9.1.1. 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。

- 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
```

9.2. カスタム TLS 証明書の使用

カスタム PKCS12 キーストアまたは TLS 証明書/キーペアを使用して、クライアントと Data Grid クラスター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のシークレットを作成します。以下を参照してください。

- 証明書のシークレット
- キーストアのシークレット

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** で暗号化シークレットを指定し、変更を適用します。

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

1. Data Grid エンドポイントを発信元および発信先とするトラフィックを暗号化します。
2. 暗号化証明書が含まれるシークレットを使用するように Data Grid を設定します。
3. 暗号化シークレットの名前を指定します。

9.2.1. 証明書のシークレット

```
apiVersion: v1
kind: Secret
metadata:
  name: tls-secret
type: Opaque
data:
    tls.key:  "LS0tLS1CRUdJTiBQUk ..." 1
    tls.crt: "LS0tLS1CRUdJTiBDRVI ..." 2
```

1. base64 でエンコードされた TLS キーを追加します。
2. base64 でエンコードされた TLS 証明書を追加します。

9.2.2. キーストアのシークレット

```
apiVersion: v1
kind: Secret
```

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

1. キーストアのエイリアスを指定します。
2. キーストアのパスワードを指定します。
3. base64 でエンコードされたキーストアを追加します。

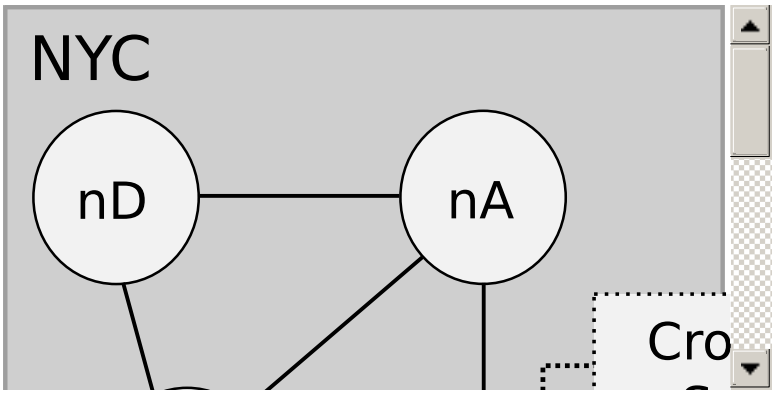
第10章 クロスサイトレプリケーションの設定

グローバル Data Grid クラスターを設定して、複数のサイトにまたがってデータをバックアップします。

10.1. DATA GRID OPERATOR でのクロスサイトレプリケーション

Data Grid クラスターが別の場所で稼働している場合、Data Grid Operator を使用してそれらを接続し、複数のサイト間でデータをバックアップできるようにします。

たとえば、以下の図では、Data Grid Operator はニューヨーク市 **NYC** のデータセンターで Data Grid クラスターを管理します。Data Grid Operator は、ロンドン **LON** の別のデータセンターでも Data Grid クラスターを管理します。



Data Grid Operator は Kubernetes API を使用して、**NYC** および **LON** の OpenShift Container Platform クラスター間でセキュアな接続を確立します。次に、Data Grid Operator は複数のサイト間のレプリケーションサービスを作成します。これにより、Data Grid クラスターは複数の場所にデータをバックアップできます。

各 Data Grid クラスターには、すべてのバックアップリクエストを調整するサイトマスターノードが 1 つあります。Data Grid Operator はサイトマスターノードを識別し、クロスサイトレプリケーションサービス経由のすべてのトラフィックがサイトマスターに送信されるようにします。

現在のサイトマスターノードがオフラインになると、新しい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そこに転送します。

10.1.1. バックアップの場所の命名要件

Data Grid Operator は、各バックアップの場所にある Data Grid クラスター名と namespace が同じであることを想定します。

たとえば、**LON** サイトで、「my-xsite」という OpenShift プロジェクトにある **metadata.name: mydatagrid** で Data Grid クラスターを作成します。**NYC** サイトの Data Grid クラスターを **LON** のバックアップとして使用するには、同一の namespace の同一の名前でそのクラスターを作成する必要があります。

10.2. 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間でセキュアな接続を確立するには、サービスアカウントトークンを生成し、交換する必要があります。これらのトークンにより、OpenShift Container Platform クラスターを認証できるため、Data Grid オペレーター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. 各 OpenShift クラスターでサービスアカウントを作成します。
   たとえば、以下のように **LON** でサービスアカウントを作成します。

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

2. view ロールをサービスアカウントに追加します。
   たとえば、Data Grid クラスターが「my-xsite」名前空間で実行される場合、以下のように view ロールを **LON** のサービスアカウントに追加します。

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

3. 各サービスアカウントからトークンを取得します。
   以下の例は、**LON** のサービスアカウントトークンを示しています。

```
$ oc sa get-token lon

eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9...
```

4. バックアップの場所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作成します。
   a. **NYC** で OpenShift Container Platform にログインします。
   b. サービスアカウントトークンを **lon-token** シークレットに追加します。

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

   c. 上記の手順を繰り返して、**LON** で **nyc-token** シークレットを作成します。

サービスアカウントトークンを各バックアップの場所に追加した後、Data Grid クラスターは複数のサイトビューを形成することができます。

リファレンス

アプリケーションでのサービスアカウントの使用

10.3. バックアップの場所の DATA GRID クラスターへの追加

Data Grid クラスターをバックアップの場所として設定し、データの複製用に専用の JGroups トランスポートチャネル上で通信できるようにします。

手順

1. 必要に応じて、各サイトの Data Grid クラスターを **Infinispan** CR で設定します。
   たとえば、**lon.yaml** を作成して **LON** を設定し、**nyc.yaml** を作成して **NYC** を設定します。両方の設定には以下が含まれている必要があります。

   - **.spec.service.sites.local** は Data Grid クラスターのローカルサイトの名前を指定します。
   - **.spec.service.sites.locations** はすべてのサイトマスターの場所を提供します。Data Grid ノードはこの情報を使用して相互に接続し、クロスサイトビューを形成します。

2. 各サイトの Data Grid クラスターをインスタンス化します。以下に例を示します。

   a. **LON** の **Infinispan** CR を適用します。

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

   b. **NYC** で OpenShift Container Platform にログインします。

   c. **NYC** の **Infinispan** CR を適用します。

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

3. Data Grid クラスターがクロスサイトビューを形成していることを確認します。
   たとえば、**LON** で以下を行います。

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

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

リファレンス

クロスサイトレプリケーションリソース

10.3.1. クロスサイトレプリケーションリソース

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

1. Data Grid Service を指定します。Data Grid は、Data Grid Service クラスターでのクロスサイトレプリケーションのみをサポートします。
2. Data Grid クラスターのローカルサイトの名前を指定します。
3. **LoadBalancer** をバックアップの場所間の通信を処理するサービスとして指定します。
4. すべてのバックアップの場所の接続情報を提供します。

5 **.spec.service.sites.local.name** と一致するバックアップの場所を指定します。

6 バックアップの場所について Kubernetes API の URL を指定します。

7 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。

第11章 DATA GRID クラスターへの接続

REST または Hot Rod エンドポイントを使用して Data Grid に接続します。これにより、リモートでキャッシュ定義を作成して変更し、Data Grid クラスター全体のデータを保存できます。

本セクションの例では、**$SERVICE_HOSTNAME** を使用して Data Grid クラスターへのアクセスを提供するサービスを示します。

OpenShift で実行されているクライアントは、Data Grid Operator が作成する内部サービスの名前を指定できます。

OpenShift の外部で実行しているクライアントは、外部サービスおよびプロバイダーのタイプに応じてホスト名を指定する必要があります。たとえば、AWS でロードバランサーサービスを使用する場合、サービスのホスト名は以下のようになります。

.status.loadBalancer.ingress[0].hostname

GCP または Azure では、ホスト名は以下のようになります。

.status.loadBalancer.ingress[0].ip

11.1. DATA GRID REST API の呼び出し

適切な HTTP クライアントを使用して Data Grid REST API を呼び出すことができます。

以下の例は、暗号化されていない接続を使用して **curl** で REST API を呼び出す方法を示しています。暗号化を使用するように HTTP クライアントを設定する方法については、本書の対象範囲外となります。

手順

1. 必要な場合は、Data Grid ノードに対してリモートシェルを開きます。以下に例を示します。

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

2. キャッシュサービスはデフォルトのキャッシュインスタンスを提供しますが、Data Grid サービスは提供されません。Data Grid Service クラスターを使用してデータを保存する前に、以下の例のようにキャッシュを作成する必要があります。

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

3. キャッシュにエントリーを追加します。

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

4. エントリーを確認します。

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

11.2. HOT ROD クライアントの設定

Hot Rod Java クライアントを Data Grid クラスターに接続するように設定します。

Hot Rod クライアント ConfigurationBuilder

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

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

Hot Rod クライアントプロパティー

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

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

# Authentication
infinispan.client.hotrod.use_auth=true
# Application user credentials.
```

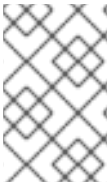
```
# The default username is developer.
infinispan.client.hotrod.auth_username=developer
infinispan.client.hotrod.auth_password=$PASSWORD
infinispan.client.hotrod.auth_server_name=$CLUSTER_NAME
infinispan.client.hotrod.sasl_properties.javax.security.sasl.qop=auth
infinispan.client.hotrod.sasl_mechanism=DIGEST-MD5

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

第12章 DATA GRID キャッシュの作成

Data Grid がデータを保存する方法を設定するキャッシュ定義を追加します。Data Grid Operator または Hot Rod または REST クライアントのいずれかでキャッシュを作成できます。

12.1. DATA GRID サービスクラスターのキャッシュの作成

infinispan.org 設定テンプレートを使用するか、有効な XML キャッシュ定義から Data Grid Service クラスターでキャッシュを作成できます。



注記

テンプレートまたは XML 定義のいずれかを指定する **Cache** CR を使用すると一度に 1 つのキャッシュのみを作成できます。**Cache** CR にテンプレートと XML 定義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。

12.1.1. キャッシュ作成用のクレデンシャルシークレットの追加

Data Grid Operator は、キャッシュを作成するために Data Grid Service クラスターで認証する必要があります。クレデンシャルをシークレットに追加し、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します。

以下の手順では、クレデンシャルを新規シークレットに追加する方法を説明します。クレデンシャルが含まれるカスタムシークレットがある場合、新規のシークレットを作成する代わりに使用することができます。

手順

1. **StringData** マップの Data Grid Service クラスターにアクセスするための有効なユーザークレデンシャルを提供する **Secret** オブジェクトタイプを定義します。
   たとえば、以下のように **developer** ユーザーのクレデンシャルを提供する **auth_secret.yaml** ファイルを作成します。

```
apiVersion: v1
stringData:
  username: developer (1)
  password: G8ZdJvSaY3lOOwfM (2)
kind: Secret
metadata:
  name: basic-auth (3)
type: Opaque
```

(1) キャッシュを作成できるユーザーの名前を指定します。

(2) ユーザーに対応するパスワードを指定します。

(3) シークレットの名前を指定します。

2. 以下の例のように、ファイルからシークレットを作成します。

```
$ oc create secret generic --from-file=auth_secret.yaml basic-auth
```

12.1.1.1. カスタムクレデンシャルシークレットの使用

Data Grid Operator では、クレデンシャルがシークレットの **username** および **password** キーの値として存在している必要があります。Data Grid のクレデンシャルが含まれ、異なるキー名を使用するカスタムシークレットがある場合は、**Cache** CR でこれらの名前をオーバーライドできます。

たとえば、以下のような Data Grid ユーザーとそのパスワードの一覧を保持する「my-credentials」というシークレットがあります。

```
stringData:
  app_user1: spock
  app_user1_pw: G8ZdJvSaY3lOOwfM
  app_user2: jim
  app_user2_pw: zTzz2gVyyF4JsYsH
```

手順

- **Cache** CR で、以下のように **username** および **password** でカスタムキー名を上書きします。

```
spec:
  adminAuth:
    username:
      key: app_user1 (1)
      name: my-credentials (2)
    password:
      key: app_user1_pw (3)
      name: my-credentials
```

(1) **username** でキー名 **app_user1** を上書きします。

(2) カスタムクレデンシャルシークレットの名前を指定します。

(3) **password** でキー名 **app_user1_pw** を上書きします。

12.1.2. テンプレートからの Data Grid キャッシュの作成

infinispan.org キャッシュ設定テンプレートを使用して Data Grid Service クラスターでキャッシュを作成するには、以下の手順を実行します。

前提条件

- Data Grid クラスターにアクセスするのに有効なユーザークレデンシャルが含まれるシークレットを作成します。
- キャッシュに使用するキャッシュ設定テンプレートを特定します。以下の REST 呼び出しを使用して、利用可能なすべての設定テンプレートを取得できます。

```
GET /rest/v2/cache-managers/default/cache-configs/templates
```

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。

たとえば、以下の CR は **org.infinispan.DIST_SYNC** キャッシュ設定テンプレートを使用する "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 (1)
spec:
  adminAuth: (2)
    secretName: basic-auth
  clusterName: example-infinispan (3)
  name: mycache (4)
  templateName: org.infinispan.DIST_SYNC (5)
```

1. **Cache** CR に名前を付けます。
2. **username** および **password** キーでクレデンシャルを提供するシークレットまたはカスタムクレデンシャルのシークレットのオーバーライドを指定します。
3. Data Grid Operator がキャッシュを作成するターゲット Data Grid クラスターの名前を指定します。
4. Data Grid キャッシュインスタンスに名前を付けます。
5. キャッシュを作成する **infinispan.org** キャッシュ設定テンプレートを指定します。

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
```

12.1.3. XML からのリモート Data Grid キャッシュの作成

有効な **infinispan.xml** キャッシュ定義を使用して Data Grid Service クラスターにキャッシュを作成するには、以下の手順を実行します。

前提条件

- Data Grid クラスターにアクセスするのに有効なユーザークレデンシャルが含まれるシークレットを作成します。

手順

1. 作成する XML キャッシュ定義が含まれる **Cache** CR を作成します。

```
apiVersion: infinispan.org/v2alpha1
kind: Cache
metadata:
  name: mycachedefinition (1)
spec:
  adminAuth: (2)
    secretName: basic-auth
  clusterName: example-infinispan (3)
```

```
  name: mycache 4
  template: <infinispan><cache-container><distributed-cache/></cache-container>
</infinispan> 5
```

1. **Cache** CR に名前を付けます。
2. **username** および **password** キーでクレデンシャルを提供するシークレットまたはカスタムクレデンシャルのシークレットのオーバーライドを指定します。
3. Data Grid Operator がキャッシュを作成するターゲット Data Grid クラスターの名前を指定します。
4. Data Grid キャッシュインスタンスに名前を付けます。
5. キャッシュを作成する XML キャッシュ定義を指定します。**name** 属性は無視されます。作成されたキャッシュにのみに **spec.name** が適用されます。

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
```

12.1.4. 永続キャッシュストアの設定

単一ファイルキャッシュストアを Data Grid サービスノードに追加して、データを永続ボリュームに保存することができます。

以下のように **persistence** 要素を使用して、キャッシュストアを Data Grid キャッシュ定義の一部として設定します。

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

次に、Data Grid は **/opt/datagrid/server/data** ディレクトリーに単一ファイルのキャッシュストアである **.dat** ファイルを作成します。

手順

- 以下の例のように、キャッシュストアで XML キャッシュ定義が含まれる **Cache** CR を作成します。

```
apiVersion: infinispan.org/v2alpha1
kind: Cache
metadata:
  name: mycachestore
spec:
  adminAuth:
    secretName: basic-auth
  clusterName: example-infinispan 1
  name: mycache 2
  template: <infinispan><cache-container><distributed-cache><persistence><file-store/>
</persistence></distributed-cache></cache-container></infinispan>
```

- 以下の例のように、**XMLStringConfiguration** クラスを使用して Hot Rod エンドポイント経由でキャッシュストアのある XML キャッシュ定義を提供します。

```
private void createCacheWithXMLConfiguration() {
      String cacheName = "mycachestore";
      String xml = String.format("<infinispan>" +
                                 "<cache-container>" +
                                 "<distributed-cache name=\"%s\" mode=\"SYNC\">" +
                                  "<persistence>" +
                                    "<file-store/>" +
                                  "</persistence>" +
                                 "</distributed-cache>" +
                                 "</cache-container>" +
                                "</infinispan>"
                                , cacheName);
      manager.administration().getOrCreateCache(cacheName, new
XMLStringConfiguration(xml));
      System.out.println("Cache created or already exists.");
   }
```

リファレンス

- Persistent Volume Claim (永続ボリューム要求、PVC)

12.2. CACHE SERVICE クラスターのキャッシュの作成

Cache Service クラスターはデフォルトのキャッシュ設定を提供します。複数のキャッシュを作成できますが、デフォルト設定のコピーとしてのみ作成できます。

手順

- 以下の REST 呼び出しを実行します。

```
POST /rest/v2/caches/$CACHENAME?template=default
```

テスト環境または開発環境では、以下のように行うことができます。

1. Data Grid ノードに対してリモートシェルを開きます。以下に例を示します。

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

2. 「mycache」という名前のキャッシュを以下のように作成します。

```
$ curl -X POST -u $USERNAME:$PASSWORD $ClusterIP:11222/rest/v2/caches/mycache?
template=default
```

3. リモートシェルを終了します。

```
$ exit
```

12.3. DATA GRID キャッシュの検証

Data Grid キャッシュを作成したら、以下の手順を実行してそれらが存在することを確認します。

手順

- 以下の REST 呼び出しを実行し、利用可能なキャッシュの一覧を表示します。

```
GET $SERVICE_HOSTNAME:11222/rest/v2/caches/
```

テスト環境または開発環境では、以下のように行うことができます。

1. Data Grid ノードに対してリモートシェルを開きます。以下に例を示します。

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

2. 以下のように利用可能なキャッシュの一覧を取得します。

```
$ curl -u $USERNAME:$PASSWORD $ClusterIP:11222/rest/v2/caches/
```

3. リモートシェルを終了します。

```
$ exit
```

第13章 PROMETHEUS での PROMETHEUS の監視

Data Grid は、統計およびイベントを Prometheus に提供するメトリクスエンドポイントを公開します。

13.1. PROMETHEUS の設定

Data Grid クラスターで認証し、監視できるように Prometheus を設定します。

前提条件

- Prometheus Operator をインストールします。
- 実行中の Prometheus インスタンスを作成します。

手順

1. 認証シークレットを Prometheus namespace に追加します。
   このシークレットにより、Prometheus は Data Grid クラスターで認証できます。Data Grid クレデンシャルは、Data Grid Operator namespace の認証シークレットで確認できます。

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

(1) アプリケーションユーザーを指定します。**developer** はデフォルトのアプリケーションユーザーです。

(2) 対応するパスワードを指定します。

2. Prometheus が Data Grid クラスターを監視するように設定するサービスモニターインスタンスを作成します。

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

```
      name: basic-auth 4
    interval: 30s
    scheme: https 5
    tlsConfig:
      insecureSkipVerify: true
      serverName: certificate-CN 6
namespaceSelector:
  matchNames:
    - infinispan 7
selector:
  matchLabels:
    app: infinispan-service
    clusterName: cluster-name 8
```

1. サービスモニターの名前を指定します。
2. Prometheus namespace を指定します。
3. Data Grid クレデンシャルを持つ認証シークレットの名前を指定します。
4. Data Grid クレデンシャルを持つ認証シークレットの名前を指定します。
5. Data Grid エンドポイントが暗号化を使用することを指定します。TLS を使用しない場合は、**spec.endpoints.scheme** を削除します。
6. Data Grid 暗号化の TLS 証明書の共通名 (CN) を指定します。OpenShift サービス証明書を使用する場合、CN は Data Grid クラスターの **metadata.name** リソースと一致します。TLS を使用しない場合は、**spec.endpoints.tlsConfig** を削除します。
7. Data Grid Operator namespace を指定します。
8. Data Grid クラスターの名前を指定します。

第14章 DATA GRID ログの監視

ロギングカテゴリーを異なるメッセージレベルに設定し、Data Grid クラスターの監視、デバッグ、およびトラブルシューティングを行います。

14.1. DATA GRID ロギングの設定

手順

1. **Infinispan** CR の **spec.logging** でロギング設定を指定し、変更を適用します。

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

1. Data Grid ロギングを設定します。
2. ロギングカテゴリーを追加します。
3. ロギングカテゴリーおよびレベルの名前を指定します。

注記

ルートロギングカテゴリーは **org.infinispan** で、デフォルトでは **INFO** です。

2. 必要に応じて Data Grid ノードからログを取得します。

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

14.2. ログレベル

ログレベルはメッセージの性質と重大度を示します。

ログレベル	説明
trace	アプリケーションの実行状態の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	個々のリクエスト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	エラーまたはパフォーマンスの低下につながる可能性のある状況を示します。

ログレベル	説明
error	操作やアクティビティーの達成を妨げる可能性があるが、アプリケーションの実行を妨げないエラー状態を示します。

第15章 リファレンス

Data Grid Operator で作成した Data Grid サービスおよびクラスターに関する情報を検索します。

15.1. キャッシュサービスリソース

```
apiVersion: infinispan.org/v1
kind: Infinispan
metadata:
  name: example-infinispan (1)
spec:
  replicas: 4 (2)
  service:
    type: Cache (3)
    replicationFactor: (4)
  autoscale: (5)
    maxMemUsagePercent: 70
    maxReplicas: 5
    minMemUsagePercent: 30
    minReplicas: 2
  security:
    endpointSecretName: endpoint-identities (6)
    endpointEncryption: (7)
        type: Secret
        certSecretName: tls-secret
  container: (8)
    extraJvmOpts: "-XX:NativeMemoryTracking=summary"
    cpu: "2000m"
    memory: 1Gi
  logging: (9)
    categories:
      org.infinispan: trace
      org.jgroups: trace
  expose: (10)
    type: LoadBalancer
```

(1) Data Grid クラスターの名前を指定します。

(2) クラスター内のノード数を指定します。

(3) Cache Service クラスターを作成します。

(4) クラスター全体の各キャッシュエントリーのレプリカ数を設定します。

(5) 自動クラススケーリングを有効化および設定します。

(6) ユーザークレデンシャルで認証シークレットを追加します。

(7) セキュアな接続用のカスタム暗号化シークレットを追加します。

(8) リソースをノードに割り当てます。

(9) ロギングを設定します。

10 外部トラフィックのサービスを設定します。

15.2. DATA GRID サービスリソース

```
apiVersion: infinispan.org/v1
kind: Infinispan
metadata:
  name: example-infinispan 1
spec:
  replicas: 6 2
  service:
    type: DataGrid 3
    container:
      storage: 2Gi 4
    sites: 5
      local:
        expose:
          type: LoadBalancer
      locations:
      - name: azure
        url: openshift://api.azure.host:6443
        secretName: azure-identities
      - name: aws
        url: openshift://api.aws.host:6443
        secretName: aws-identities
  security:
    endpointSecretName: endpoint-identities 6
    endpointEncryption: 7
        type: Secret
        certSecretName: tls-secret
  container: 8
    extraJvmOpts: "-XX:NativeMemoryTracking=summary"
    cpu: "1000m"
    memory: 1Gi
  logging: 9
    categories:
      org.infinispan: debug
      org.jgroups: debug
  expose: 10
    type: LoadBalancer
```

1 Data Grid クラスターの名前を指定します。

2 クラスター内のノード数を指定します。

3 Data Grid Service クラスターを作成します。

4 永続ボリュームのサイズを設定します。

5 バックアップの場所の接続情報を提供します。

6 ユーザークレデンシャルで認証シークレットを追加します。

7 セキュアな接続用のカスタム暗号化シークレットを追加します。

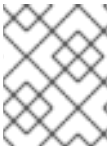
8 リソースをノードに割り当てます。

9 ロギングを設定します。

10 外部トラフィックのサービスを設定します。

15.3. PERSISTENT VOLUME CLAIM (永続ボリューム要求、PVC)

Data Grid Operator は永続ボリュームを以下にマウントします。
/opt/datagrid/server/data



注記

Persistent Volume Claim (永続ボリューム要求、PVC) は **ReadWriteOnce (RWO)** アクセスモードを使用します。

15.4. ネットワークサービス

内部サービス

- Data Grid ノードが相互を検出し、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	プロトコル	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	クラスター検出

外部サービス

OpenShift 外部のクライアントから、または異なる namespace で 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 で外部サービスを作成する必要があります。デフォルトでは使用できません。

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

クロスサイトサービス

Data Grid は、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	プロトコル	説明
<cluster_name>-site	**7900**	TCP	クロスサイト通信用の JGroups RELAY2 チャネル。

リファレンス

ネットワークサービスの作成

15.5. DATA GRID OPERATOR のアップグレード

Data Grid Operator は、新規バージョンが利用可能になると Data Grid をアップグレードします。

Data Grid クラスターをアップグレードするため、Data Grid Operator は Data Grid ノードのイメージのバージョンをチェックします。Data Grid Operator がイメージの新規バージョンを利用できると判断すると、すべてのノードを正常にシャットダウンし、新規イメージを適用した後にノードを再起動します。

Red Hat OpenShift では、Operator Lifecycle Manager (OLM) は Data Grid Operator のアップグレードを有効にします。Data Grid Operator のインストール時に、**Approval Strategy** で **Automatic** または **Manual** 更新のいずれかを選択します。これにより、Data Grid Operator がクラスターをアップグレードする方法を決定します。詳細は、OpenShift のドキュメントを参照してください。

リファレンス

- Understanding the Operator Lifecycle Manager
- Approval Strategy

15.6. 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いないため、Red Hat では実稼働環境での使用を推奨していません。Red Hat は実稼働環境でこれらを使用することを推奨していません。これらの機能は、近々発表予定の製品機能をリリースに先駆けてご提供することにより、お客様は機能性をテストし、開発プロセス中に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、「テクノロジプレビュー機能のサポート範囲」を参照してください。