Red Hat Data Grid 8.1

OpenShift での Data Grid の実行

OpenShift で Data Grid サービスを設定して実行する

Last Updated: 2023-12-15

Red Hat Data Grid 8.1 OpenShift での Data Grid の実行

OpenShift で Data Grid サービスを設定して実行する

法律上の通知

概要

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

目次

RED HAT DATA GRID

Data Grid は、高性能の分散型インメモリーデータストアです。

スキーマレスデータ構造

さまざまなオブジェクトをキーと値のペアとして格納する柔軟性があります。

グリッドベースのデータストレージ

クラスター間でデータを分散および複製するように設計されています。

エラスティックスケーリング

サービスを中断することなく、ノードの数を動的に調整して要件を満たします。

データの相互運用性

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

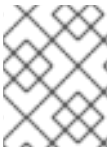
DATA GRID のドキュメント

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

- Data Grid 8.1 ドキュメント
- Data Grid 8.1 コンポーネントの詳細
- Data Grid 8.1 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、Red Hat CTO である Chris Wright のメッセージ をご覧ください。

第1章 DATA GRID OPERATOR 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 8.1.x の更新を取得します。

   承認ストラテジー

   8.1.x チャネルから更新を自動的にインストールするか、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: 8.1.x
installPlanApproval: Automatic (1)
name: datagrid
source: redhat-operators
sourceNamespace: openshift-marketplace
EOF
```

(1) 8.1.x チャネルからの更新を手動で承認する場合は、**Manual** を指定します。

4. インストールを確認します。

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

第2章 DATA GRID OPERATOR のスタートガイド

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 をインストールしている。
- **oc** クライアントがある。

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 1
kind: Infinispan 2
metadata:
  name: example-infinispan 3
spec:
  replicas: 2 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 クラスターに名前を付けます。
4. Data Grid クラスター内のノードの数を指定します。

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 ノードのクラスターを作成します。

手順

1. **Infinispan** CR の **spec.replicas** で、クラスター内の Data Grid ノードの数を指定します。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 ノード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

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

- 以下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

自動スクリプトには、**wellFormed** 条件で **oc wait** を使用します。

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

第3章 DATA GRID サービスのセットアップ

Data Grid Operator を使用して、Cache サービスまたは Data Grid サービスノードのクラスターを作成します。

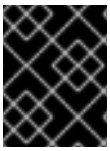
3.1. サービスの種別

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。

キャッシュサービス

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が必要な場合は、キャッシュサービスを使用します。キャッシュサービスノード:

- 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

キャッシュサービスノードは揮発性があるため、**Infinispan** CR でクラスターに変更を適用するか、Data Grid 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。

Data Grid サービス

必要に応じて、Data Grid サービスを使用します。

- 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 検索やその他の高度な機能を使用します。

3.2. キャッシュサービスノードの作成

デフォルトでは、Data Grid Operator はキャッシュサービスノードを含む Data Grid クラスターを作成します。

手順

1. **Infinispan** CR を作成します。

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

```
metadata:
  name: example-infinispan
spec:
  replicas: 2
  service:
    type: Cache 1
```

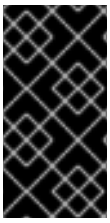
1 キャッシュサービスノードを作成します。これは **Infinispan** CR のデフォルトです。

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

3.2.1. 自動スケーリングの設定

キャッシュサービスノードでクラスターを作成する場合、Data Grid Operator はデフォルトキャッシュのメモリー使用量に基づいて、ノード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。

Data Grid Operator は、キャッシュサービスノード上のデフォルトキャッシュを監視します。キャッシュにデータを追加すると、メモリー使用量が増加します。クラスターに追加の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のではなく、新しい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っていることが検出された場合、Data Grid Operator はノードをシャットダウンします。



重要

自動スケーリングは、デフォルト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。

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. **autoscale** フィールドを使用して、クラスターのメモリー使用量のしきい値とノード数を設定します。

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

1 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設定します。Data Grid Operator は、クラスター内のいずれかのノードがしきい値に達したことを検出すると、可能であれば新しい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。

2 クラスターのノード数の最大数を定義します。

3 クラスター全体のメモリー使用量の最小しきい値をパーセンテージで設定します。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、ノードを

4 クラスターのノード数の最小数を定義します。

3. 変更を適用します。

3.2.2. 所有者数の設定

所有者の数は、Data Grid クラスター全体に複製される各キャッシュエントリーのコピーの数を制御します。キャッシュサービスノードのデフォルトは 2 で、データの損失を防ぐために各エントリーを複製します。

手順

1. 次のように、**Infinispan** CR の **spec.service.replicationFactor** リソースで所有者数を指定します。

```
spec:
  ...
  service:
    type: Cache
    replicationFactor: 3 (1)
```

1 各キャッシュエントリーに 3 つのレプリカを設定します。

2. 変更を適用します。

3.2.3. キャッシュサービスリソース

```
apiVersion: infinispan.org/v1
kind: Infinispan
metadata:
  # Names the cluster.
  name: example-infinispan
spec:
  # Specifies the number of nodes in the cluster.
  replicas: 4
  service:
    # Configures the service type as Cache.
    type: Cache
    # Sets the number of replicas for each entry across the cluster.
    replicationFactor: 2
  # Enables and configures automatic scaling.
  autoscale:
    maxMemUsagePercent: 70
    maxReplicas: 5
    minMemUsagePercent: 30
    minReplicas: 2
  # Configures authentication and encryption.
  security:
    # Defines a secret with custom credentials.
    endpointSecretName: endpoint-identities
```

```
  # Adds a custom TLS certificate to encrypt client connections.
  endpointEncryption:
      type: Secret
      certSecretName: tls-secret
# Sets container resources.
container:
  extraJvmOpts: "-XX:NativeMemoryTracking=summary"
  cpu: "2000m"
  memory: 1Gi
# Configures logging levels.
logging:
  categories:
    org.infinispan: trace
    org.jgroups: trace
# Configures how the cluster is exposed on the network.
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"
```

3.3. DATA GRID サービスノードの作成

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 サービスノードのクラスターを作成します。

手順

1. **Infinispan** CR の **spec.service.type** の値として **DataGrid** を指定します。

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

注記

ノードの作成後に **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、既存のノードを削除してから新規のノードを作成する必要があります。

2. 他の Data Grid サービスリソースを使用してノードを設定します。

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

3.3.1. Data Grid サービスリソース

```
apiVersion: infinispan.org/v1
kind: Infinispan
metadata:
  # Names the cluster.
  name: example-infinispan
spec:
  # Specifies the number of nodes in the cluster.
  replicas: 6
  service:
    # Configures the service type as Data Grid.
    type: DataGrid
    # Configures storage resources.
    container:
      storage: 2Gi
      storageClassName: my-storage-class
    # Configures cross-site replication.
    sites:
      local:
      name: azure
      expose:
        type: LoadBalancer
      locations:
      - name: azure
        url: openshift://api.azure.host:6443
        secretName: azure-token
      - name: aws
        url: openshift://api.aws.host:6443
        secretName: aws-token
  # Configures authentication and encryption.
  security:
    # Defines a secret with custom credentials.
    endpointSecretName: endpoint-identities
    # Adds a custom TLS certificate to encrypt client connections.
    endpointEncryption:
        type: Secret
        certSecretName: tls-secret
  # Sets container resources.
  container:
    extraJvmOpts: "-XX:NativeMemoryTracking=summary"
    cpu: "1000m"
    memory: 1Gi
  # Configures logging levels.
  logging:
    categories:
      org.infinispan: debug
      org.jgroups: debug
      org.jgroups.protocols.TCP: error
      org.jgroups.protocols.relay.RELAY2: fatal
  # Configures how the cluster is exposed on the network.
  expose:
    type: LoadBalancer
```

```
# Configures affinity and anti-affinity strategies.
affinity:
  podAntiAffinity:
    preferredDuringSchedulingIgnoredDuringExecution:
    - weight: 100
      podAffinityTerm:
        labelSelector:
          matchLabels:
            app: infinispan-pod
            clusterName: example-infinispan
            infinispan_cr: example-infinispan
        topologyKey: "kubernetes.io/hostname"
```

3.4. 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。

```
apiVersion: infinispan.org/v1
kind: Infinispan
metadata:
  annotations:
    # Add labels that you want to attach to services.
    infinispan.org/targetLabels: svc-label1, svc-label2
    # Add labels that you want to attach to pods.
    infinispan.org/podTargetLabels: pod-label1, pod-label2
  labels:
    # Add values for your 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 コンテナー仕様の調整

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

4.1. 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. ホスト CPU リソースをノードに割り当てます (CPU ユニットで測定)。
3. ホストメモリーリソースをノードに割り当てます (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。

ガベー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 stdout に転送することができます。

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

4.2. ストレージリソース

デフォルトでは、Data Grid Operator はキャッシュサービスと Data Grid サービスノードの両方にストレージに **1Gi** を割り当てます。Data Grid サービスノードのストレージリソースは設定できますが、キャッシュサービスノードのストレージリソースは設定できません。

```
spec:
  ...
  service:
    type: DataGrid
    container:
      storage: 2Gi 1
      storageClassName: my-storage-class 2
```

1. Data Grid サービスノードのストレージサイズを設定します。
2. 永続ボリューム要求に使用する 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。

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 **/opt/infinispan/server/data** にマウントします。

注記

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。

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 クラスターを停止および起動します。

キャッシュ定義

キャッシュサービスおよび Data Grid サービスの両方が永続的なキャッシュ定義を永続ボリュームに保存し、クラスターの再起動後に引き続き利用できるようにします。

Data

Data Grid サービスノードは、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中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

キャッシュサービスノードをシャットダウンすると、キャッシュ内のすべてのデータが削除されます。Data Grid サービスノードの場合は、永続ボリュームがすべてのデータを保持できるように、Data Grid サービス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 サービスノードで利用可能なメモリー量より少ない場合、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、次の手順を実行します。

前提条件

Data Grid サービスノードの場合は、シャットダウンする前と同じ数のノードを指定してクラスターを再起動する必要があります。たとえば、6 つのノードで構成されるクラスターをシャットダウンしたとします。そのクラスターを再起動するときは、**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 はクラスター全体でのデータの分散を復元できます。クラスターの全ノードが実行されている場合、ノードを追加または削除できます。

Data Grid クラスターの正しいノード数は次のようにして確認できます。

```
$ 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。次に例を示します。

```
spec:
  replicas: 6
```

第6章 CONFIGURING NETWORK ACCESS TO DATA GRID

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。

6.1. 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
```

関連情報

- Network Services

6.2. 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. **spec.expose.type** でサービスタイプとして **LoadBalancer** を指定します。

```
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通じて、ネットワーク上に 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
```

6.3. 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. **spec.expose.type** でサービスタイプとして **NodePort** を指定します。

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

1 ノードポートサービスを通じてネットワーク上に 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、プラットフォームによってポートが選択されます。

3. 変更を適用します。
4. **-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
```

6.4. 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. **spec.expose.type** でサービスタイプとして **Route** を指定します。

3. 必要に応じて、**spec.expose.host** を使用してホスト名を追加します。

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

(1) OpenShift Route を通じてネットワーク上に Data Grid を公開します。

(2) 必要に応じて、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. 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。

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

第7章 DATA GRID コネクションのセキュリティー確保

認証と暗号化を使用してクライアント接続をセキュリティー保護し、ネットワークへの侵入を防ぎ、データを保護します。

7.1. 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。

7.1.1. デフォルトの認証情報

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認証情報を生成します。

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。

7.1.2. Retrieving Credentials

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

- 以下の例のように、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
- username: operator
  password: uMBo9CmEdEduYk24
```

7.1.3. カスタム認証情報の追加

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。

手順

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

```
credentials:
```

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

重要

identities.yaml には operator ユーザーが含まれている必要があります。

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 (1)
```

(1) 認証情報を含む認証シークレットの名前を指定します。

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
```

7.2. 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 ノードとの間の接続を暗号化します。

7.2.1. Red Hat OpenShift サービス証明書を使用した暗号化

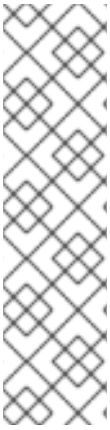
Data Grid Operator は、Red Hat OpenShift サービス CA によって署名された TLS 証明書を自動的に生成します。次に、Data Grid Operator は証明書およびキーをシークレットに格納し、それらを取得してリモートクライアントで使用できるようにします。

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 (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** とキー **tls.key** を含むシークレットに PEM 形式で名前を付けます。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。

7.2.2. Retrieving TLS Certificates

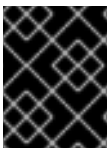
暗号化シークレットから TLS 証明書を取得して、クライアントトラストストアを作成します。

- 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
```

7.2.3. 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

Data Grid では、エンドポイントが **spec.expose.type** を介して OpenShift クラスターの外部に公開されている実稼働環境で暗号化を無効にすることは推奨しません。

手順

- **Infinispan** CR の **spec.security.endpointEncryption.type** フィールドの値として **None** を設定し、変更を適用します。

```
spec:
  ...
  security:
    endpointEncryption:
        type: None 1
```

1 Data Grid エンドポイントの暗号化を無効にします。

7.2.4. 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。参照:
- 証明書シークレット
- キーストアシークレット

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 エンドポイントとの間の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に名前を付けます。

7.2.4.1. 証明書シークレット

```
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。

7.2.4.2. キーストアシークレット

```
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。

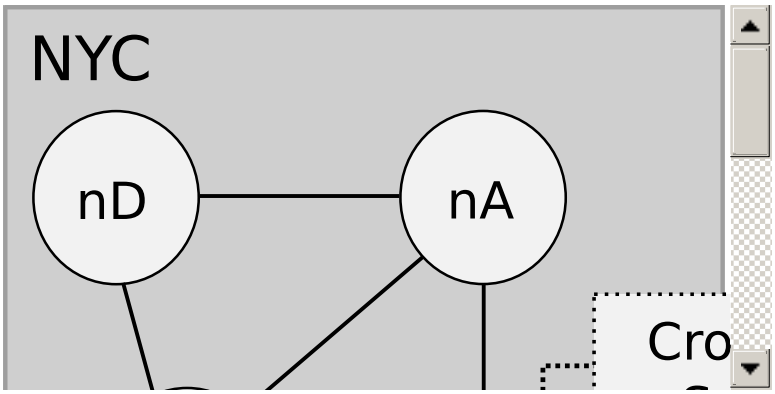
第8章 クロスサイトレプリケーションの設定

グローバル Data Grid クラスターをセットアップして、サイト間でデータをバックアップします。

8.1. DATA GRID OPERATOR を使用したクロスサイトレプリケーション

Data Grid クラスターを異なる場所で実行している場合は、Data Grid Operator を使用してクラスターを接続し、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 がニューヨーク市のデータセンター **NYC** で Data Grid クラスターを管理しています。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 クラスター間のセキュア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。

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。

8.2. 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. 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

8.3. 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。

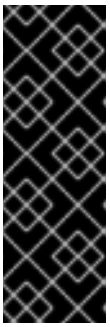
8.4. クロスサイトレプリケーションのための DATA GRID クラスターの設定

Data Grid クラスターをバックアップの場所として設定し、データを複製するために専用の JGroups トランスポートチャネルを介して通信できるようにします。

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ている。
- すべてのクラスターが Data Grid サービスノードである。

- OpenShift プロジェクト名が一致している。



重要

クロスサイトレプリケーションを実行する場合、Data Grid Operator は、Data Grid クラスターが同じ名前を持ち、同じ namespace で実行されることを必要とします。

たとえば、**LON** に **xsite-cluster** という名前のプロジェクトでクラスターを作成するとします。**NYC** のクラスターも、**xsite-cluster** という名前のプロジェクトで実行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. **metadata.name** を使用して、各 Data Grid クラスターに同じ名前を指定します。
3. ローカルサイトの名前を **spec.service.sites.local.name** で指定します。
4. **spec.service.sites.local.expose.type** を使用して、ローカルサイトの公開サービスタイプを設定します。
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。
   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
      locations:
        - name: LON
          url: openshift://api.rhdg-lon.openshift-aws.myhost.com:6443
          secretName: lon-token
        - name: NYC
          url: openshift://api.rhdg-nyc.openshift-aws.myhost.com:6443
          secretName: nyc-token
```

   - NYC

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

```
  name: example-infinispan
spec:
  replicas: 2
  service:
    type: DataGrid
    sites:
      local:
        name: NYC
        expose:
          type: LoadBalancer
      locations:
        - name: NYC
          url: openshift://api.rhdg-nyc.openshift-aws.myhost.com:6443
          secretName: nyc-token
        - name: LON
          url: openshift://api.rhdg-lon.openshift-aws.myhost.com:6443
          secretName: lon-token
```

6. クロスサイトレプリケーションのロギングレベルを次のように調整します。

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

前述の設定では、JGroups TCP および RELAY2 プロトコルのロギングを減らします。これにより、コンテナーストレージを使用するログファイルが大量に生成される可能性があるクラスターバックアップ操作に関する過剰なメッセージを減らします。

7. 他の Data Grid サービスリソースを使用してノードを設定します。

8. **Infinispan** CR を適用します。

9. ノードログをチェックして、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。次に例を示します。

```
$ 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]
```

次のステップ

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

関連情報

- クロスサイトレプリケーションリソース
- キャッシュへのバックアップの場所の追加
- Data Grid Guide to Cross-Site Replication

8.4.1. 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
  logging:
    categories:
      org.jgroups.protocols.TCP: error (8)
      org.jgroups.protocols.relay.RELAY2: fatal (9)
```

(1) Data Grid サービスを指定します。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。

(2) Data Grid クラスターのローカルサイトに名前を付けます。

(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** に一致するバックアップの場所を指定します。

(6) バックアップ場所の OpenShift 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。

(8) JGroups TCP プロトコルのエラーメッセージをログに記録します。

(9) JGroups RELAY2 プロトコルの致命的なメッセージをログに記録します。

第9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。

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、実稼働環境での使用は推奨していません。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。

Red Hat テクノロジープレビュー機能のサポート範囲

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 サービスノード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。

9.1. DATA GRID OPERATOR の認証情報の追加

Data Grid Operator は、キャッシュを作成するために Data Grid サービスクラスターで認証する必要があります。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、シークレットに認証情報を追加します。

次の手順では、新しいシークレットに認証情報を追加する方法について説明します。認証情報を含むカスタムシークレットがすでにある場合は、新しいシークレットを作成する代わりにそれを使用できます。

手順

1. **StringData** マップ内の Data Grid サービスクラスターにアクセスするための有効なユーザー認証情報を提供する **Secret** オブジェクトタイプを定義します。
   たとえば、次のように **developer** ユーザーの認証情報を提供する **basic-auth.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に名前を付けます。

(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. 次の例のように、ファイルからシークレットを作成します。

```
$ oc apply -f basic-auth.yaml
```

9.1.1. カスタム認証情報シークレットの使用

Data Grid Operator では、認証情報がシークレット内の **username** キーと **password** キーの値として存在する必要があります。Data Grid 認証情報を含むカスタムシークレットがあるが、別のキー名を使用している場合は、**Cache** CR でそれらの名前をオーバーライドできます。

たとえば、次のように Data Grid ユーザーとそのパスワードのリストを保持する "my-credentials" という名前のシークレットがあるとします。

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

手順

- **Cache** CR で、次のようにカスタムキー名を **username** と **password** でオーバーライドします。

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

(1) **app_user1** キー名を **username** でオーバーライドします。

(2) カスタム認証情報シークレットの名前を指定します。

(3) **app_user1_pw** キー名を **passwd** でオーバーライドします。

9.2. XML からの 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 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

- Data Grid クラスターにアクセスするための有効なユーザー認証情報を含むシークレットを作成している。

手順

1. 作成する XML キャッシュ定義を含む **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 name="mycache"
mode="SYNC"><persistence><file-store/></persistence></distributed-cache></cache-
container></infinispan> (5)
```

(1) **Cache** CR に名前を付けます。

(2) **username** と **password** のキーを認証情報に提供するシークレット、またはカスタム認証情報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 クラスター上のキャッシュに名前を付けます。

(5) キャッシュを作成するための XML キャッシュ定義を指定します。**name** 属性は無視されることに注意してください。生成されるキャッシュには **spec.name** のみが適用されます。

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
```

9.3. テンプレートからの DATA GRID キャッシュの作成

キャッシュ設定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、次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ための有効なユーザー認証情報を含むシークレットを作成している。
- 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ている。使用可能な設定テンプレートのリストは、Data Grid コンソールで確認できます。

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。
たとえば、次の 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** と **password** のキーを認証情報に提供するシークレット、またはカスタム認証情報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ための **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
```

9.4. 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. 各サイトに同じ名前のキャッシュ設定を作成します。
各サイトのキャッシュ設定では、異なるキャッシュモードとバックアップストラテジーを使用できます。Data Grid は、キャッシュ名に基づいてデータを複製します。
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

```
<infinispan>
  <cache-container>
    <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>
  </cache-container>
</infinispan>
```

- LON

```
<infinispan>
  <cache-container>
    <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>
  </cache-container>
</infinispan>
```

関連情報

- クロスサイトレプリケーションのためのクラスターの設定
- Data Grid Guide to Cross-Site Replication

9.4.1. 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、ノード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。

9.5. 永続キャッシュストアの追加

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できます。

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

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

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

- 次のようにキャッシュストアをキャッシュ設定に追加します。

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

関連情報

- ストレージリソース

第10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。

10.1. 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
- 認証情報
- 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。

Ports

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

10.2. DATA GRID キャッシュの作成

OpenShift で Data Grid を実行するときにキャッシュを作成するには、以下の方法を使用できます。

- **Cache** CR を使用します。
- **Cache** CR を使用しない場合は、Data Grid CLI を使用して一度に複数のキャッシュを作成します。

- **Cache** CR または Data Grid CLI の代わりに、Data Grid コンソールにアクセスして XML または JSON 形式でキャッシュを作成します。
- 必要な場合にのみ、Hot Rod クライアントを使用して、プログラムで、またはキャッシュごとのプロパティーを使用してキャッシュを作成します。

関連情報

Data Grid Operator を使用したキャッシュの作成

10.3. DATA GRID CLI を使用した接続

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。

CLI はサーバーディストリビューションの一部として利用でき、ローカルホスト上で実行して OpenShift 上の Data Grid クラスターへのリモート接続を確立できます。

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。

10.3.1. Data Grid CLI を使用したキャッシュの作成

CLI を使用してキャッシュを Data Grid クラスターに追加します。

前提条件

- CLI を実行できるように、サーバーのディストリビューションをダウンロードしている。
- 必要なクライアント接続の詳細を取得している。

手順

1. XML または JSON 形式でキャッシュ設定を含むファイルを作成します。次に例を示します。

```
cat > infinispan.xml<<EOF
<infinispan>
  <cache-container>
    <distributed-cache name="mycache">
      <encoding>
        <key media-type="application/x-protostream"/>
        <value media-type="application/x-protostream"/>
      </encoding>
    </distributed-cache>
  </cache-container>
</infinispan>
EOF
```

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

```
$ bin/cli.sh -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。

3. 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。

4. **create queue** コマンドと **--file** オプションを使用してキャッシュを追加します。

```
[//containers/default]> create cache --file=infinispan.xml mycache
```

5. **ls** コマンドを使用して、キャッシュが存在することを確認します。

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

6. 必要に応じて、**describe** コマンドを使用してキャッシュ設定を取得します。

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

関連情報

- サーバー分布のダウンロード
- Data Grid コマンドラインインターフェイスの使用

10.3.2. キャッシュをバッチで作成する

Data Grid CLI を使用したバッチ操作で複数のキャッシュを追加します。

前提条件

- CLI を実行できるように、サーバーのディストリビューションをダウンロードしている。
- 必要なクライアント接続の詳細を取得している。

手順

1. XML または JSON 形式のキャッシュ設定を含むファイルを少なくとも 1 つ作成します。
2. たとえば、次のようにバッチファイルを作成します。

```
cat > caches.batch<<EOF
echo "connecting"
connect --username=developer --password=dIRs5cAAsHIeeRIL
echo "creating caches..."
create cache firstcache --file=infinispan-one.xml
create cache secondcache --file=infinispan-two.xml
create cache thirdcache --file=infinispan-three.xml
create cache fourthcache --file=infinispan-four.xml
```

```
echo "verifying caches"
ls caches
EOF
```

3. CLI でキャッシュを作成します。

```
$ bin/cli.sh -c https://$SERVICE_HOSTNAME:$PORT --trustall -f /tmp/caches.batch
```

$SERVICE_HOSTNAME:$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

- サーバー分布のダウンロード
- Data Grid コマンドラインインターフェイスの使用

10.4. 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、ルートを作成します。

手順

1. **$SERVICE_HOSTNAME:$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。
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。

10.5. 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 ノード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 ノードの内部 IP アドレスにアクセスできるため、任意のクライアントの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。

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

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。

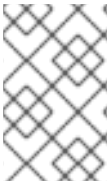
```
import org.infinispan.client.hotrod.configuration.ClientIntelligence;
import org.infinispan.client.hotrod.configuration.ConfigurationBuilder;
import org.infinispan.client.hotrod.configuration.SaslQop;
...

ConfigurationBuilder builder = new ConfigurationBuilder();
    builder.addServer()
            .host("$SERVICE_HOSTNAME")
            .port("$PORT")
           .security().authentication()
            .username("username")
            .password("password")
            .realm("default")
            .saslQop(SaslQop.AUTH)
            .saslMechanism("SCRAM-SHA-512")
           .ssl()
            .sniHostName("$SERVICE_HOSTNAME")
            .trustStorePath("/path/to/tls.crt");
    builder.clientIntelligence(ClientIntelligence.BASIC);
```

-

10.5.2. 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
# Path to the TLS certificate.
# Clients automatically generate trust stores from certificates.
infinispan.client.hotrod.trust_store_path=/var/run/secrets/kubernetes.io/serviceaccount/service-ca.crt
```

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

10.5.3. Hot Rod クライアントを使用したキャッシュの作成

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("<infinispan>" +
                              "<cache-container>" +
                              "<distributed-cache name=\"%s\" mode=\"SYNC\">" +
                                "<encoding media-type=\"application/x-protostream\"/>" +
                                "<locking isolation=\"READ_COMMITTED\"/>" +
                                "<transaction mode=\"NON_XA\"/>" +
                                "<expiration lifespan=\"60000\" interval=\"20000\"/>" +
                              "</distributed-cache>" +
                              "</cache-container>" +
                            "</infinispan>"
                            , 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。

次の例のように、キャッシュ設定を Hot Rod クライアントプロパティーに追加します。

```
# 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
```

10.6. 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、ルートを作成します。

手順

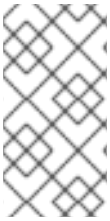
- **$SERVICE_HOSTNAME:$PORT/rest/v2** の任意の HTTP クライアントで REST API にアクセスします。
  $SERVICE_HOSTNAME:$PORT を、ネットワーク上で Data Grid を使用できるホスト名とポートに置き換えます。

関連情報

- Data Grid REST API

10.7. キャッシュサービスノードへのキャッシュの追加

キャッシュサービスノードには、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、Cache サービスノードの代わりに Data Grid サービスノード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
```

10.7.1. デフォルトのキャッシュ設定

キャッシュサービスノードのデフォルトのキャッシュは次のとおりです。

```
<infinispan>
  <cache-container>
    <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
      <partition-handling when-split="ALLOW_READ_WRITES" 7
                          merge-policy="REMOVE_ALL"/> 8
    </distributed-cache>
  </cache-container>
</infinispan>
```

1. キャッシュインスタンスに default という名前を付けます。
2. 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。
3. クラスターの各キャッシュエントリーのレプリカを 2 つ設定します。
4. 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。
5. 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、ノードの作成時に最大サイズを計算します。
6. 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 がノードを追加するようにすることができます。
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に名前を付けます。
8. Data Grid が競合を検出すると、キャッシュからエントリーを削除するマージポリシーを指定します。

第11章 PROMETHEUS を使用した DATA GRID の監視

Data Grid は、統計とイベントを Prometheus に提供するメトリックエンドポイントを公開します。

11.1. PROMETHEUS サービスモニターの作成

Data Grid クラスターを監視するように Prometheus を設定するサービスモニターインスタンスを定義します。

前提条件

- OpenShift クラスターに Prometheus スタックをセットアップします。

手順

1. Prometheus が Data Grid クラスターで認証できるように、Data Grid 認証情報を含む認証シークレットを作成します。

```
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** がデフォルトです。

(2) 対応するパスワードを指定します。

2. 認証シークレットを Prometheus namespace に追加します。

```
$ oc apply -f basic-auth.yaml
```

3. Data Grid クラスターを監視するように Prometheus を設定するサービスモニターを作成します。

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  labels:
    k8s-app: prometheus
  name: datagrid-monitoring (1)
  namespace: infinispan-monitoring (2)
spec:
  endpoints:
    - targetPort: 11222 (3)
      path: /metrics (4)
      honorLabels: true
      basicAuth:
        username:
```

```
      key: username
      name: basic-auth (5)
    password:
      key: password
      name: basic-auth
  interval: 30s
  scrapeTimeout: 10s
  scheme: https (6)
  tlsConfig:
    insecureSkipVerify: true
    serverName: example-infinispan (7)
namespaceSelector:
  matchNames:
  - infinispan (8)
selector:
  matchLabels:
    app: infinispan-service
    clusterName: example-infinispan (9)
```

(1) サービスモニターインスタンスに名前を付けます。

(2) Prometheus スタックの namespace を指定します。

(3) Data Grid メトリクスエンドポイントのポートを **11222** に設定します。

(4) Data Grid がメトリクスを公開するパスを設定します。

(5) Data Grid 認証情報を持つ認証シークレットを指定します。

(6) Data Grid クラスターがエンドポイント暗号化を使用することを指定します。

(7) 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。

(8) Data Grid クラスターの namespace を指定します。

(9) Data Grid クラスターの名前を指定します。

4. サービスモニターインスタンスを Prometheus namespace に追加します。

```
$ oc apply -f service-monitor.yaml
```

関連情報

- Prometheus Operator
- OpenShift クラスターモニタリングスタックを使用したサービスのモニタリング

第12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。

12.1. 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

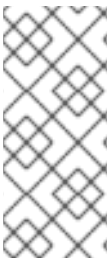
以下のタイプのキャッシュに対して、**x** ノードの障害に対する耐性を提供します。

- 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。

12.2. 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。

関連情報

- Anti-Affinity ストラテジーの設定

12.3. 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。

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 (1)
        podAffinityTerm:
          labelSelector:
            matchLabels:
              app: infinispan-pod
              clusterName: <cluster_name>
              infinispan_cr: <cluster_name>
          topologyKey: "kubernetes.io/hostname" (2)
```

(1) ホスト名ストラテジーを最も優先されるものとして設定します。

(2) 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。

さまざまなノードが必要

```
spec:
  affinity:
    podAntiAffinity:
      requiredDuringSchedulingIgnoredDuringExecution: (1)
      - labelSelector:
          matchLabels:
            app: infinispan-pod
            clusterName: <cluster_name>
            infinispan_cr: <cluster_name>
        topologyKey: "topology.kubernetes.io/hostname"
```

(1) 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。

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

次の例では、Pod をスケジュールするときに複数のゾーンを優先します。

```
spec:
  affinity:
    podAntiAffinity:
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 100 (1)
        podAffinityTerm:
          labelSelector:
            matchLabels:
              app: infinispan-pod
              clusterName: <cluster_name>
              infinispan_cr: <cluster_name>
          topologyKey: "topology.kubernetes.io/zone" (2)
      - weight: 90 (3)
        podAffinityTerm:
          labelSelector:
            matchLabels:
              app: infinispan-pod
              clusterName: <cluster_name>
              infinispan_cr: <cluster_name>
          topologyKey: "kubernetes.io/hostname" (4)
```

(1) ゾーンストラテジーを最も優先されるものとして設定します。

(2) 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。

(3) 次に優先されるものとしてホスト名ストラテジーを設定します。

(4) ゾーンをまたいでスケジュールできない場合は、Data Grid レプリカ Pod を異なる OpenShift ノード上でスケジュールします。

複数のゾーンが必要

```
spec:
  affinity:
    podAntiAffinity:
      requiredDuringSchedulingIgnoredDuringExecution: (1)
      - labelSelector:
          matchLabels:
            app: infinispan-pod
            clusterName: <cluster_name>
            infinispan_cr: <cluster_name>
        topologyKey: "topology.kubernetes.io/zone"
```

(1) 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。

第13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て、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。

13.1. DATA GRID ロギングの設定

手順

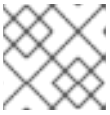
1. **Infinispan** CR の **spec.logging** でロギング設定を指定してから、変更を適用します。

```
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とレベルに名前を付けます。



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
```

13.2. 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。

第14章 参照

Data Grid Operator を使用して作成する Data Grid サービスとクラスターに関する情報を示します。

14.1. NETWORK SERVICES

内部サービス

- Data Grid ノードが相互に検出し、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	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	クラスターの検出

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	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 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	プロトコル	説明
<cluster_name>-site	**7900**	TCP	クロスサイト通信向けの JGroups RELAY2 チャネル。

関連情報

ネットワークサービスの作成