OpenShift Container Platform 4.5

ストレージ

OpenShift Container Platform でのストレージの設定および管理

Last Updated: 2022-11-30

OpenShift Container Platform 4.5 ストレージ

OpenShift Container Platform でのストレージの設定および管理

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

法律上の通知

概要

本書では、各種のストレージのバックエンドから永続ボリュームを設定し、Pod からの動的な割り当てを管理する方法について説明します。

目次

第1章 一時ストレージについて

1.1. 概要

永続ストレージに加え、Pod とコンテナーは、操作に一時または短期的なローカルストレージを必要とする場合があります。この一時ストレージは、個別の Pod の寿命より長くなることはなく、一時ストレージは Pod 間で共有することはできません。

Pod は、スクラッチスペース、キャッシュ、ログに一時ローカルストレージを使用します。ローカルストレージのアカウントや分離がないことに関連する問題には、以下が含まれます。

- Pod は利用可能なローカルストレージのサイズを認識しない。
- Pod がローカルストレージを要求しても確実に割り当てられない可能性がある。
- ローカルストレージはベストエフォートのリソースである。
- Pod は、他の Pod でローカルストレージがいっぱいになるとエビクトされる可能性があり、十分なストレージが回収されるまで、新しい Pod は入れない。

一時ストレージは、永続ボリュームとは異なり、体系化されておらず、システム、コンテナーランタイム、Openshift Container Platform での他の用途に加え、ノードで実行中のすべての Pod 間で領域を共有します。一時ストレージフレームワークにより、Pod は短期的なローカルストレージのニーズを指定できます。またこれにより、OpenShift Container Platform は該当する場合に Pod をスケジュールし、ローカルストレージの過剰な使用に対してノードを保護することができます。

一時ストレージフレームワークでは、管理者および開発者がこのローカルストレージの管理を改善できますが、I/O スループットやレイテンシーに関する確約はありません。

1.2. 一時ストレージのタイプ

一時ローカルストレージは常に、プライマリーパーティションで利用できるようになっています。プライマリーパーティションを作成する基本的な方法には、Root、ランタイム の 2 つがあります。

Root
このパーティションでは、kubelet の root ディレクトリー **/var/lib/kubelet/** (デフォルト) と **/var/log/** ディレクトリーを保持します。このパーティションは、ユーザーの Pod、OS、Kubernetes システムのデーモン間で共有できます。Pod は、**EmptyDir** ボリューム、コンテナーログ、イメージ階層、コンテナーの書き込み可能な階層を使用して、このパーティションを使用できます。Kubelet はこのパーティションの共有アクセスおよび分離を管理します。このパーティションは一時的なもので、アプリケーションは、このパーティションからディスク IOPS などのパフォーマンス SLA は期待できません。

ランタイム
これは、ランタイムがオーバーレイファイルシステムに使用可能なオプションのパーティションです。OpenShift Container Platform は、このパーティションの分離および共有アクセスを特定して提供します。コンテナーイメージ階層と書き込み可能な階層は、ここに保存されます。ランタイムパーティションが存在する場合は、**root** パーティションにはイメージ階層もその他の書き込み可能階層も含まれません。

1.3. 一時ストレージ管理

クラスター管理者は、非終了状態のすべての Pod の一時ストレージに対して制限範囲や一時ストレージの要求数を定義するクォータを設定することで、プロジェクト内で一時ストレージを管理できます。開発者は Pod およびコンテナーのレベルで、このコンピュートリソースの要求および制限を設定する

こともできます。

1.4. 一時ストレージのモニタリング

/bin/df をツールとして使用し、一時コンテナーデータが置かれているボリューム (**/var/lib/kubelet** および **/var/lib/containers**) の一時ストレージの使用をモニターすることができます。**/var/lib/kubelet** のみが使用できる領域は、クラスター管理者によって **/var/lib/containers** が別のディスクに置かれる場合に **df** コマンドを使用すると表示されます。

/var/lib での使用済みおよび利用可能な領域の人間が判読できる値を表示するには、以下のコマンドを実行します。

```
$ df -h /var/lib
```

この出力には、**/var/lib** での一時ストレージの使用状況が表示されます。

出力例

```
Filesystem  Size  Used Avail Use% Mounted on
/dev/sda1    69G   32G   34G  49% /
```

第2章 永続ストレージについて

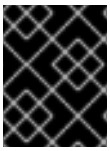
2.1. 永続ストレージの概要

ストレージの管理は、コンピュートリソースの管理とは異なります。OpenShift Container Platform は Kubernetes 永続ボリューム (PV) フレームワークを使用してクラスター管理者がクラスターの永続ストレージのプロビジョニングを実行できるようにします。開発者は、永続ボリューム要求 (PVC) を使用すると、基礎となるストレージインフラストラクチャーについての特定の知識がなくても PV リソースを要求することができます。

PVC はプロジェクトに固有のもので、開発者が PV を使用する手段として作成し、使用します。PV リソース自体のスコープはいずれの単一プロジェクトにも設定されず、それらは OpenShift Container Platform クラスター全体で共有でき、すべてのプロジェクトから要求できます。PV が PVC にバインドされた後は、その PV を追加の PVC にバインドすることはできません。これにはバインドされた PV を単一の namespace (バインディングプロジェクトの namespace) にスコープ設定する作用があります。

PV は、クラスター管理者によって静的にプロビジョニングされているか、または **StorageClass** オブジェクトを使用して動的にプロビジョニングされているクラスター内の既存ストレージの一部を表す、**PersistentVolume** API オブジェクトで定義されます。これは、ノードがクラスターリソースであるのと同様にクラスター内のリソースです。

PV は **Volumes** などのボリュームプラグインですが、PV を使用する個々の Pod から独立したライフサイクルを持ちます。PV オブジェクトは、NFS、iSCSI、またはクラウドプロバイダー固有のストレージシステムのいずれの場合でも、ストレージの実装の詳細をキャプチャーします。



重要

インフラストラクチャーにおけるストレージの高可用性は、基礎となるストレージのプロバイダーに委ねられています。

PVC は、開発者によるストレージの要求を表す **PersistentVolumeClaim** API オブジェクトによって定義されます。これは Pod がノードリソースを消費する点で Pod に似ており、PVC は PV リソースを消費します。たとえば、Pod は特定のレベルのリソース (CPU およびメモリーなど) を要求し、PVC は特定のストレージ容量およびアクセスモードを要求できます。たとえば、それらは読み取り/書き込みで 1 回、読み取り専用で複数回マウントできます。

2.2. ボリュームおよび要求のライフサイクル

PV はクラスターのリソースです。PVC はそれらのリソースの要求であり、リソースに対する要求チェックとして機能します。PV と PVC 間の相互作用には以下のライフサイクルが設定されます。

2.2.1. ストレージのプロビジョニング

PVC で定義される開発者からの要求に対応し、クラスター管理者はストレージおよび一致する PV をプロビジョニングする 1 つ以上の動的プロビジョナーを設定します。

または、クラスター管理者は、使用可能な実際のストレージの詳細を保持する多数の PV を前もって作成できます。PV は API に存在し、利用可能な状態になります。

2.2.2. 要求のバインド

PVC の作成時に、ストレージの特定容量の要求、必要なアクセスモードの指定のほか、ストレージクラスを作成してストレージの記述や分類を行います。マスターのコントロールループは新規 PVC の有無

を監視し、新規 PVC を適切な PV にバインドします。適切な PV がない場合には、ストレージクラスのプロビジョナーが PV を作成します。

すべての PV のサイズが PVC サイズを超える可能性があります。これは、手動でプロビジョニングされる PV にとくに当てはまります。超過を最小限にするために、OpenShift Container Platform は他のすべての条件に一致する最小の PV にバインドします。

要求は、一致するボリュームが存在しないか、ストレージクラスを提供するいずれの利用可能なプロビジョナーで作成されない場合には無期限でバインドされないままになります。要求は、一致するボリュームが利用可能になるとバインドされます。たとえば、多数の手動でプロビジョニングされた 50Gi ボリュームを持つクラスターは 100Gi を要求する PVC に一致しません。PVC は 100Gi PV がクラスターに追加されるとバインドされます。

2.2.3. Pod および要求した PV の使用

Pod は要求をボリュームとして使用します。クラスターは要求を検査して、バインドされたボリュームを検索し、Pod にそのボリュームをマウントします。複数のアクセスモードをサポートするボリュームの場合、要求を Pod のボリュームとして使用する際に適用するモードを指定する必要があります。

要求が存在し、その要求がバインドされている場合、バインドされた PV を必要な期間保持することができます。Pod のスケジュールおよび要求された PV のアクセスは、**persistentVolumeClaim** を Pod のボリュームブロックに組み込んで実行できます。

2.2.4. 使用中のストレージオブジェクトの保護

使用中のストレージオブジェクトの保護機能を使用すると、Pod または PVC にバインドされる PV によってアクティブに使用されている PVC がシステムから削除されないようにすることができます。これらが削除されると、データが失われる可能性があります。

使用中のストレージオブジェクトの保護はデフォルトで有効にされています。

注記

PVC は、PVC を使用する **Pod** オブジェクトが存在する場合に Pod によってアクティブに使用されます。

ユーザーが Pod によってアクティブに使用されている PVC を削除する場合でも、PVC はすぐに削除されません。PVC の削除は、PVC が Pod によってアクティブに使用されなくなるまで延期されます。また、クラスター管理者が PVC にバインドされる PV を削除しても、PV はすぐに削除されません。PV の削除は、PV が PVC にバインドされなくなるまで延期されます。

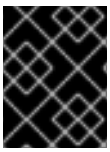
2.2.5. 永続ボリュームの解放

ボリュームの処理が終了したら、API から PVC オブジェクトを削除できます。これにより、リソースを回収できるようになります。ボリュームは要求の削除時に解放 (リリース) されたものとみなされますが、別の要求で利用できる状態にはなりません。以前の要求側に関連するデータはボリューム上に残るので、ポリシーに基づいて処理される必要があります。

2.2.6. 永続ボリュームの回収ポリシー

永続ボリュームの回収ポリシーは、クラスターに対してリリース後のボリュームの処理方法について指示します。ボリュームの回収ポリシーは、**Retain**、**Recycle** または **Delete** のいずれかにすることができます。

- **Retain** 回収ポリシーは、サポートするボリュームプラグインのリソースの手動による回収を許可します。
- **Recycle** 回収ポリシーは、ボリュームがその要求からリリースされると、バインドされていない永続ボリュームのプールにボリュームをリサイクルします。



重要

Recycle 回収ポリシーは OpenShift Container Platform 4 では非推奨となっています。動的プロビジョニングは、同等またはそれ以上の機能で推奨されます。

- **Delete** 回収ポリシーは、OpenShift Container Platform の **PersistentVolume** オブジェクトと、AWS EBS または VMware vSphere などの外部インフラストラクチャーの関連するストレージアセットの両方を削除します。

注記

動的にプロビジョニングされたボリュームは常に削除されます。

2.2.7. 永続ボリュームの手動回収

永続ボリューム要求 (PVC) が削除されても、永続ボリューム (PV) は依然として存在し、released (リリース済み) とみなされます。ただし、PV は、直前の要求側のデータがボリューム上に残るため、別の要求には利用できません。

手順

クラスター管理者として PV を手動で回収するには、以下を実行します。

1. PV を削除します。

```
$ oc delete <pv-name>
```

AWS EBS、GCE PD、Azure Disk、Cinder ボリュームなどの外部インフラストラクチャーの関連するストレージアセットは、PV の削除後も引き続き存在します。

2. 関連するストレージアセットのデータをクリーンアップします。
3. 関連するストレージアセットを削除します。または、同じストレージアセットを再利用するには、ストレージアセットの定義で新規 PV を作成します。

回収される PV が別の PVC で使用できるようになります。

2.2.8. 永続ボリュームの回収ポリシーの変更

永続ボリュームの回収ポリシーを変更するには、以下を実行します。

1. クラスターの永続ボリュームを一覧表示します。

```
$ oc get pv
```

出力例

```
NAME                        CAPACITY   ACCESSMODES   RECLAIMPOLICY   STATUS
```

```
CLAIM           STORAGECLASS     REASON    AGE
 pvc-b6efd8da-b7b5-11e6-9d58-0ed433a7dd94   4Gi        RWO           Delete          Bound
default/claim1    manual                    10s
 pvc-b95650f8-b7b5-11e6-9d58-0ed433a7dd94   4Gi        RWO           Delete          Bound
default/claim2    manual                    6s
 pvc-bb3ca71d-b7b5-11e6-9d58-0ed433a7dd94   4Gi        RWO           Delete          Bound
default/claim3    manual                    3s
```

2. 永続ボリュームの 1 つを選択し、その回収ポリシーを変更します。

```
$ oc patch pv <your-pv-name> -p '{"spec":{"persistentVolumeReclaimPolicy":"Retain"}}'
```

3. 選択した永続ボリュームに正しいポリシーがあることを確認します。

```
$ oc get pv
```

出力例

```
NAME                                       CAPACITY   ACCESSMODES   RECLAIMPOLICY   STATUS
CLAIM           STORAGECLASS     REASON    AGE
 pvc-b6efd8da-b7b5-11e6-9d58-0ed433a7dd94   4Gi        RWO           Delete          Bound
default/claim1    manual                    10s
 pvc-b95650f8-b7b5-11e6-9d58-0ed433a7dd94   4Gi        RWO           Delete          Bound
default/claim2    manual                    6s
 pvc-bb3ca71d-b7b5-11e6-9d58-0ed433a7dd94   4Gi        RWO           Retain          Bound
default/claim3    manual                    3s
```

上記の出力では、要求 **default/claim3** にバインドされたボリュームに **Retain** 回収ポリシーが含まれるようになりました。ユーザーが要求 **default/claim3** を削除した場合、ボリュームは自動的に削除されません。

2.3. 永続ボリューム

各 PV には、以下の例のように、ボリュームの仕様およびステータスである **spec** および **status** が含まれます。

PersistentVolume オブジェクト定義の例

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv0001 (1)
spec:
  capacity:
    storage: 5Gi (2)
  accessModes:
    - ReadWriteOnce (3)
  persistentVolumeReclaimPolicy: Retain (4)
  ...
status:
  ...
```

1. 永続ボリュームの名前。
2. ボリュームに利用できるストレージの量。
3. 読み取り書き込みおよびマウントパーミッションを定義するアクセスモード。
4. リソースのリリース後にそれらのリソースがどのように処理されるかを示す回収ポリシー。

2.3.1. PV の種類

OpenShift Container Platform は以下の永続ボリュームプラグインをサポートします。

- AWS Elastic Block Store (EBS)
- Azure Disk
- Azure File
- Cinder
- ファイバーチャネル
- GCE Persistent Disk
- HostPath
- iSCSI
- ローカルボリューム
- NFS
- OpenStack Manila
- Red Hat OpenShift Container Storage
- VMware vSphere

2.3.2. 容量

通常、永続ボリューム (PV) には特定のストレージ容量があります。これは PV の **capacity** 属性を使用して設定されます。

現時点で、ストレージ容量は設定または要求できる唯一のリソースです。今後は属性として IOPS、スループットなどが含まれる可能性があります。

2.3.3. アクセスモード

永続ボリュームは、リソースプロバイダーでサポートされるすべての方法でホストにマウントできます。プロバイダーには各種の機能があり、それぞれの PV のアクセスモードは特定のボリュームでサポートされる特定のモードに設定されます。たとえば、NFS は複数の読み取り/書き込みクライアントをサポートしますが、特定の NFS PV は読み取り専用としてサーバー上でエクスポートされる可能性があります。それぞれの PV は、その特定の PV の機能について記述するアクセスモードの独自のセットを取得します。

要求は、同様のアクセスモードのボリュームに一致します。一致する条件はアクセスモードとサイズの 2 つの条件のみです。要求のアクセスモードは要求 (request) を表します。そのため、より多くのアクセスを付与することはできますが、アクセスを少なくすることはできません。たとえば、要求により RWO が要求されるものの、利用できる唯一のボリュームが NFS PV (RWO+ROX+RWX) の場合に、要求は RWO をサポートする NFS に一致します。

直接的なマッチングが常に最初に試行されます。ボリュームのモードは、要求モードと一致するか、要求した内容以上のものを含む必要があります。サイズは予想されるものより多いか、またはこれと同等である必要があります。2 つのタイプのボリューム (NFS および iSCSI など) のどちらにも同じセットのアクセスモードがある場合、それらのいずれかがそれらのモードを持つ要求に一致する可能性があります。ボリュームのタイプ間で順序付けすることはできず、タイプを選択することはできません。

同じモードのボリュームはすべて分類され、サイズ別 (一番小さいものから一番大きいもの順) に分類されます。バインダーは一致するモードのグループを取得し、1 つのサイズが一致するまでそれぞれを (サイズの順序で) 繰り返し処理します。

以下の表では、アクセスモードをまとめています。

表2.1 アクセスモード

アクセスモード	CLI の省略形	説明
ReadWriteOnce	**RWO**	ボリュームは単一ノードで読み取り/書き込みとしてマウントできます。
ReadOnlyMany	**ROX**	ボリュームは数多くのノードで読み取り専用としてマウントできます。
ReadWriteMany	**RWX**	ボリュームは数多くのノードで読み取り/書き込みとしてマウントできます。

重要

ボリュームのアクセスモードは、ボリューム機能の記述子になります。それらは施行されている制約ではありません。ストレージプロバイダーはリソースの無効な使用から生じるランタイムエラーに対応します。

たとえば、NFS は **ReadWriteOnce** アクセスモードを提供します。ボリュームの ROX 機能を使用する必要がある場合は、要求に **read-only** のマークを付ける必要があります。プロバイダーのエラーは、マウントエラーとしてランタイム時に表示されます。

iSCSI およびファイバーチャネルボリュームには現在、フェンシングメカニズムがありません。ボリュームが一度に 1 つのノードでのみ使用されるようにする必要があります。ノードのドレイン (解放) などの特定の状況では、ボリュームは 2 つのノードで同時に使用できます。ノードをドレイン (解放) する前に、まずこれらのボリュームを使用する Pod が削除されていることを確認してください。

表2.2 サポート対象の PV 向けアクセスモード

ボリュームプラグイン	ReadWriteOnce [1]	ReadOnlyMany	ReadWriteMany
AWS EBS [2]	▮	-	-

ボリュームプラグイン	ReadWriteOnce [1]	ReadOnlyMany	ReadWriteMany
Azure File	▮	▮	▮
Azure Disk	▮	-	-
Cinder	▮	-	-
ファイバーチャネル	▮	▮	-
GCE Persistent Disk	▮	-	-
HostPath	▮	-	-
iSCSI	▮	▮	-
ローカルボリューム	▮	-	-
NFS	▮	▮	▮
OpenStack Manila	-	-	▮
Red Hat OpenShift Container Storage	▮	-	▮
VMware vSphere	▮	-	-

1. ReadWriteOnce (RWO) ボリュームは複数のノードにマウントできません。ノードに障害が発生すると、システムは、すでに障害が発生しているノードに割り当てられているため、割り当てられた RWO ボリュームを新規ノードにマウントすることはできません。複数割り当てのエラーメッセージが表示される場合には、シャットダウンまたはクラッシュしたノードで Pod を強制的に削除し、動的永続ボリュームの割り当て時などの重要なワークロードでのデータ損失を回避します。

2. AWS EBS に依存する Pod の再作成デプロイメントストラテジーを使用します。

2.3.4. フェーズ

ボリュームは以下のフェーズのいずれかにあります。

表2.3 ボリュームのフェーズ

フェーズ	説明
Available	まだ要求にバインドされていない空きリソースです。
Bound	ボリュームが要求にバインドされています。

フェーズ	説明
Released	要求が削除されていますが、リソースがまだクラスターにより回収されていません。
Failed	ボリュームが自動回収に失敗しています。

以下を実行して PV にバインドされている PVC の名前を表示できます。

```
$ oc get pv <pv-claim>
```

2.3.4.1. マウントオプション

アノテーション **volume.beta.kubernetes.io/mount-options** を使用して PV のマウント中にマウントオプションを指定できます。

以下は例になります。

マウントオプションの例

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv0001
  annotations:
    volume.beta.kubernetes.io/mount-options: rw,nfsvers=4,noexec 1
spec:
  capacity:
    storage: 1Gi
  accessModes:
  - ReadWriteOnce
  nfs:
    path: /tmp
    server: 172.17.0.2
  persistentVolumeReclaimPolicy: Retain
  claimRef:
    name: claim1
    namespace: default
```

1 指定のマウントオプションは、PV がディスクにマウントされている時に使用されます。

以下の PV タイプがマウントオプションをサポートします。

- AWS Elastic Block Store (EBS)
- Azure Disk
- Azure File
- Cinder
- GCE Persistent Disk

- iSCSI
- ローカルボリューム
- NFS
- Red Hat OpenShift Container Storage (Ceph RBD のみ)
- VMware vSphere

注記

ファイバーチャネルおよび HostPath PV はマウントオプションをサポートしません。

2.4. 永続ボリューム要求 (PVC)

各 **PersistentVolumeClaim** オブジェクトには、永続ボリューム要求 (PVC) の仕様およびステータスである **spec** および **status** が含まれます。 以下が例になります。

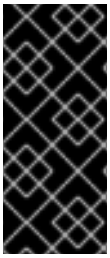
PersistentVolumeClaim オブジェクト定義の例

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: myclaim 1
spec:
  accessModes:
    - ReadWriteOnce 2
  resources:
    requests:
      storage: 8Gi 3
  storageClassName: gold 4
status:
  ...
```

1. PVC の名前
2. 読み取り書き込みおよびマウントパーミッションを定義するアクセスモード
3. PVC に利用できるストレージの量
4. 要求で必要になる **StorageClass** の名前

2.4.1. ストレージクラス

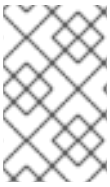
要求は、ストレージクラスの名前を **storageClassName** 属性に指定して特定のストレージクラスをオプションでリクエストできます。リクエストされたクラスの PV、つまり PVC と同じ **storageClassName** を持つ PV のみが PVC にバインドされます。クラスター管理者は 1 つ以上のストレージクラスを提供するように動的プロビジョナーを設定できます。クラスター管理者は、PVC の仕様に一致する PV をオンデマンドで作成できます。



重要

Cluster Storage Operator は、使用されるプラットフォームに応じてデフォルトのストレージクラスをインストールする可能性があります。このストレージクラスは Operator によって所有され、制御されます。アノテーションとラベルを定義するほかは、これを削除したり、変更したりすることはできません。異なる動作が必要な場合は、カスタムストレージクラスを定義する必要があります。

クラスター管理者は、すべての PVC にデフォルトストレージクラスを設定することもできます。デフォルトのストレージクラスが設定されると、PVC は "" に設定された **StorageClass** または **storageClassName** アノテーションがストレージクラスなしの PV にバインドされるように明示的に要求する必要があります。



注記

複数のストレージクラスがデフォルトとしてマークされている場合、PVC は **storageClassName** が明示的に指定されている場合にのみ作成できます。そのため、1 つのストレージクラスのみをデフォルトとして設定する必要があります。

2.4.2. アクセスモード

要求は、特定のアクセスモードのストレージを要求する際にボリュームと同じ規則を使用します。

2.4.3. リソース

要求は、Pod の場合のようにリソースの特定の数量を要求できます。今回の例では、ストレージに対する要求です。同じリソースモデルがボリュームと要求の両方に適用されます。

2.4.4. ボリュームとしての要求

Pod は要求をボリュームとして使用することでストレージにアクセスします。この要求を使用して、Pod と同じ namespace 内に要求を共存させる必要があります。クラスターは Pod の namespace で要求を見つけ、これを使用して要求をサポートする **PersistentVolume** を取得します。以下のように、ボリュームはホストにマウントされ、Pod に組み込まれます。

ホストおよび Pod のサンプルへのボリュームのマウント

```
kind: Pod
apiVersion: v1
metadata:
  name: mypod
spec:
  containers:
    - name: myfrontend
      image: dockerfile/nginx
      volumeMounts:
      - mountPath: "/var/www/html" 1
        name: mypd 2
  volumes:
    - name: mypd
      persistentVolumeClaim:
        claimName: myclaim 3
```

1. Pod 内にボリュームをマウントするためのパス
2. マウントするボリュームの名前
3. 使用する同じ namespace にある PVC の名前

2.5. ブロックボリュームのサポート

OpenShift Container Platform は、raw ブロックボリュームを静的にプロビジョニングできます。これらのボリュームにはファイルシステムがなく、ディスクに直接書き込むアプリケーションや、独自のストレージサービスを実装するアプリケーションにはパフォーマンス上の利点があります。

raw ブロックボリュームは、PV および PVC 仕様で **volumeMode: Block** を指定してプロビジョニングされます。

重要

raw ブロックボリュームを使用する Pod は、特権付きコンテナーを許可するように設定する必要があります。

以下の表は、ブロックボリュームをサポートするボリュームプラグインを表示しています。

表2.4 ブロックボリュームのサポート

ボリュームプラグイン	手動のプロビジョニング	動的なプロビジョニング	完全対応
AWS EBS	▮	▮	▮
Azure Disk	▮	▮	▮
Azure File			
Cinder	▮	▮	
ファイバーチャネル	▮		
GCP	▮	▮	▮
HostPath			
iSCSI	▮		▮
ローカルボリューム	▮		▮
NFS			
Red Hat OpenShift Container Storage	▮	▮	▮

ボリュームプラグイン	手動のプロビジョニング	動的なプロビジョニング	完全対応
VMware vSphere	▮	▮	▮

注記

手動でプロビジョニングできるものの、完全にサポートされていないブロックボリュームはいずれも、テクノロジープレビューとしてのみ提供されます。テクノロジープレビュー機能は、Red Hat 製品のサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。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。テクノロジープレビューの機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。Red Hat のテクノロジープレビュー機能のサポート範囲についての詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

2.5.1. ブロックボリュームの例

PV の例

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: block-pv
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  volumeMode: Block (1)
  persistentVolumeReclaimPolicy: Retain
  fc:
    targetWWNs: ["50060e801049cfd1"]
    lun: 0
    readOnly: false
```

(1) **volumeMode** を **Block** に設定して、この PV が raw ブロックボリュームであることを示します。

PVC の例

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: block-pvc
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Block (1)
  resources:
    requests:
      storage: 10Gi
```

1. **volumeMode** を **Block** に設定して、raw ブロック PVC が要求されていることを示します。

Pod 仕様の例

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-with-block-volume
spec:
  containers:
    - name: fc-container
      image: fedora:26
      command: ["/bin/sh", "-c"]
      args: [ "tail -f /dev/null" ]
      volumeDevices:  1
        - name: data
          devicePath: /dev/xvda 2
  volumes:
    - name: data
      persistentVolumeClaim:
        claimName: block-pvc 3
```

1. **volumeMounts** ではなく **volumeDevices** がブロックデバイスに使用されます。**PersistentVolumeClaim** ソースのみを raw ブロックボリュームと共に使用できます。

2. **mountPath** ではなく **devicePath** が raw ブロックがシステムにマップされる物理デバイスへのパスを表します。

3. ボリュームソースのタイプは **persistentVolumeClaim** であり、予想通りに PVC の名前に一致する必要があります。

表2.5 volumeMode の許容値

値	デフォルト
Filesystem	Yes
Block	No

表2.6 ブロックボリュームのバインディングシナリオ

PV volumeMode	PVC volumeMode	バインディングの結果
Filesystem	Filesystem	バインド
Unspecified	Unspecified	バインド
Filesystem	Unspecified	バインド

PV volumeMode	PVC volumeMode	バインディングの結果
Unspecified	Filesystem	バインド
Block	Block	バインド
Unspecified	Block	バインドなし
Block	Unspecified	バインドなし
Filesystem	Block	バインドなし
Block	Filesystem	バインドなし

重要

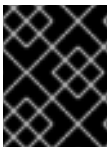
値を指定しないと、**Filesystem** のデフォルト値が指定されます。

第3章 永続ストレージの設定

3.1. AWS ELASTIC BLOCK STORE を使用した永続ストレージ

OpenShift Container Platform は AWS Elastic Block Store volumes (EBS) をサポートします。Amazon EC2 を使用して、OpenShift Container Platform クラスターに永続ストレージをプロビジョニングできます。これには、Kubernetes および AWS についてのある程度の理解があることが前提となります。

Kubernetes 永続ボリュームフレームワークは、管理者がクラスターのプロビジョニングを永続ストレージを使用して実行できるようにし、ユーザーが基礎となるインフラストラクチャーの知識がなくてもこれらのリソースを要求できるようにします。AWS Elastic Block Store ボリュームは動的にプロビジョニングできます。永続ボリュームは単一のプロジェクトまたは namespace にバインドされず、それらは OpenShift Container Platform クラスター間で共有できます。永続ボリューム要求 (PVC) はプロジェクトまたは namespace に固有のもので、ユーザーによって要求されます。



重要

インフラストラクチャーにおけるストレージの高可用性は、基礎となるストレージのプロバイダーに委ねられています。

3.1.1. 関連情報

- In-tree(インツリー) ボリュームプラグインでは不可能なボリュームスナップショットなどの追加のストレージオプションへのアクセスについての詳細は、AWS Elastic Block Store CSI ドライバー Operator を参照してください。

3.1.2. EBS ストレージクラスの作成

ストレージクラスを使用すると、ストレージのレベルや使用状況を区別し、記述することができます。ストレージクラスを定義することにより、ユーザーは動的にプロビジョニングされた永続ボリュームを取得できます。

手順

1. OpenShift Container Platform コンソールで、**Storage → Storage Classes** をクリックします。
2. ストレージクラスの概要では、**Create Storage Class**をクリックします。
3. 表示されるページで必要なオプションを定義します。
    a. ストレージクラスを参照するための名前を入力します。
    b. オプションの説明を入力します。
    c. 回収ポリシーを選択します。
    d. ドロップダウンリストから **kubernetes.io/aws-ebs** を選択します。
    e. 必要に応じてストレージクラスの追加パラメーターを入力します。
4. **Create** をクリックしてストレージクラスを作成します。

3.1.3. 永続ボリューム要求 (PVC) の作成

前提条件

ストレージは、ボリュームとして OpenShift Container Platform にマウントされる前に基礎となるインフラストラクチャーになければなりません。

手順

1. OpenShift Container Platform コンソールで、**Storage → Persistent Volume Claims**をクリックします。
2. 永続ボリューム要求 (PVC) の概要で、**Create Persistent Volume Claim**をクリックします。
3. 表示されるページで必要なオプションを定義します。
   a. ドロップダウンメニューから以前に作成されたストレージクラスを選択します。
   b. ストレージ要求の一意の名前を入力します。
   c. アクセスモードを選択します。これにより、作成されたストレージ要求の読み取り/書き込みアクセスが決定されます。
   d. ストレージ要求のサイズを定義します。
4. **Create** をクリックして永続ボリューム要求 (PVC) を作成し、永続ボリュームを生成します。

3.1.4. ボリュームのフォーマット

OpenShift Container Platform は、ボリュームをマウントしてコンテナーに渡す前に、永続ボリューム定義の **fsType** パラメーターで指定されたファイルシステムがボリュームにあるかどうか確認します。デバイスが指定されたファイルシステムでフォーマットされていない場合、デバイスのデータはすべて消去され、デバイスはそのファイルシステムで自動的にフォーマットされます。

これにより、OpenShift Container Platform がフォーマットされていない AWS ボリュームを初回の使用前にフォーマットするため、それらを永続ボリュームとして使用することが可能になります。

3.1.5. ノード上の EBS ボリュームの最大数

OpenShift Container Platform では、デフォルトで 1 つのノードに最大 39 の EBS ボリュームを割り当てることができます。この制限は、AWS ボリュームの制限 に合致します。ボリュームの制限は、インスタンスのタイプによって異なります。

重要

クラスター管理者は、In-tree または Container Storage Interface (CSI) ボリュームのいずれかと、それぞれのストレージクラスを使用する必要がありますが、ボリュームの両方のタイプを同時に使用することはできません。割り当てられている EBS ボリュームの最大数は、In-tree および CSI ボリュームについて別々にカウントされます。

3.2. AZURE を使用した永続ストレージ

OpenShift Container Platform では、Microsoft Azure Disk ボリュームがサポートされます。Azure を使用して、OpenShift Container Platform クラスターに永続ストレージをプロビジョニングできます。これには、Kubernetes と Azure についてのある程度の理解があることが前提となります。Kubernetes 永続ボリュームフレームワークは、管理者がクラスターのプロビジョニングを永続ストレージを使用して実行できるようにし、ユーザーが基礎となるインフラストラクチャーの知識がなくてもこれらのリソー

スを要求できるようにします。Azure Disk ボリュームは動的にプロビジョニングできます。永続ボリュームは単一のプロジェクトまたは namespace にバインドされず、それらは OpenShift Container Platform クラスター間で共有できます。Persistent volume claim (PVC) はプロジェクトまたは namespace に固有のもので、ユーザーによって要求されます。

重要

インフラストラクチャーにおけるストレージの高可用性は、基礎となるストレージのプロバイダーに委ねられています。

関連情報

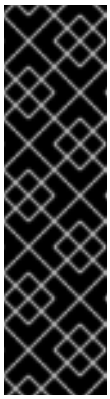
- Microsoft Azure Disk

3.2.1. Azure ストレージクラスの作成

ストレージクラスを使用すると、ストレージのレベルや使用状況を区別し、記述することができます。ストレージクラスを定義することにより、ユーザーは動的にプロビジョニングされた永続ボリュームを取得できます。

手順

1. OpenShift Container Platform コンソールで、**Storage → Storage Classes** をクリックします。
2. ストレージクラスの概要では、**Create Storage Class**をクリックします。
3. 表示されるページで必要なオプションを定義します。
   a. ストレージクラスを参照するための名前を入力します。
   b. オプションの説明を入力します。
   c. 回収ポリシーを選択します。
   d. ドロップダウンリストから **kubernetes.io/azure-disk** を選択します。
      i. ストレージアカウントのタイプを入力します。これは、Azure ストレージアカウントの SKU の層に対応します。有効なオプションは、**Premium_LRS**、**Standard_LRS**、**StandardSSD_LRS**、および **UltraSSD_LRS** です。
      ii. アカウントの種類を入力します。有効なオプションは **shared**、**dedicated** および **managed** です。



重要

Red Hat は、ストレージクラスでの **kind: Managed** の使用のみをサポートします。

Shared および **Dedicated** の場合、Azure は管理対象外のディスクを作成しますが、OpenShift Container Platform はマシンの OS (root) ディスクの管理ディスクを作成します。ただし、Azure Disk はノードで管理ディスクおよび管理対象外ディスクの両方の使用を許可しないため、**Shared** または **Dedicated** で作成された管理対象外ディスクを OpenShift Container Platform ノードに割り当てることはできません。

e. 必要に応じてストレージクラスの追加パラメーターを入力します。

4. **Create** をクリックしてストレージクラスを作成します。

関連情報

- Azure Disk Storage Class

3.2.2. 永続ボリューム要求 (PVC) の作成

前提条件

ストレージは、ボリュームとして OpenShift Container Platform にマウントされる前に基礎となるインフラストラクチャーになければなりません。

手順

1. OpenShift Container Platform コンソールで、**Storage → Persistent Volume Claims** をクリックします。
2. 永続ボリューム要求 (PVC) の概要で、**Create Persistent Volume Claim** をクリックします。
3. 表示されるページで必要なオプションを定義します。
   a. ドロップダウンメニューから以前に作成されたストレージクラスを選択します。
   b. ストレージ要求の一意の名前を入力します。
   c. アクセスモードを選択します。これにより、作成されたストレージ要求の読み取り/書き込みアクセスが決定されます。
   d. ストレージ要求のサイズを定義します。
4. **Create** をクリックして永続ボリューム要求 (PVC) を作成し、永続ボリュームを生成します。

3.2.3. ボリュームのフォーマット

OpenShift Container Platform は、ボリュームをマウントしてコンテナーに渡す前に、永続ボリューム定義の **fsType** パラメーターで指定されたファイルシステムがボリュームにあるかどうか確認します。デバイスが指定されたファイルシステムでフォーマットされていない場合、デバイスのデータはすべて消去され、デバイスはそのファイルシステムで自動的にフォーマットされます。

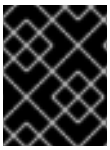
これにより、OpenShift Container Platform がフォーマットされていない Azure ボリュームを初回の使用前にフォーマットするため、それらを永続ボリュームとして使用することが可能になります。

3.3. AZURE FILE を使用した永続ストレージ

OpenShift Container Platform では、Microsoft Azure File ボリュームがサポートされます。Azure を使用して、OpenShift Container Platform クラスターに永続ストレージをプロビジョニングできます。これには、Kubernetes と Azure についてのある程度の理解があることが前提となります。

Kubernetes 永続ボリュームフレームワークは、管理者がクラスターのプロビジョニングを永続ストレージを使用して実行できるようにし、ユーザーが基礎となるインフラストラクチャーの知識がなくてもこれらのリソースを要求できるようにします。Azure File ボリュームは動的にプロビジョニングできます。

永続ボリュームは単一のプロジェクトまたは namespace にバインドされず、それらは OpenShift Container Platform クラスター間で共有できます。永続ボリューム要求 (PVC) はプロジェクトまたは namespace に固有のもので、アプリケーションで使用できるようにユーザーによって要求されます。



重要

インフラストラクチャーにおけるストレージの高可用性は、基礎となるストレージのプロバイダーに委ねられています。

追加リソース

- Azure Files

3.3.1. Azure File 共有永続ボリューム要求 (PVC) の作成

永続ボリューム要求 (PVC) を作成するには、最初に Azure アカウントおよびキーを含む **Secret** オブジェクトを定義する必要があります。このシークレットは **PersistentVolume** 定義に使用され、アプリケーションで使用できるように永続ボリューム要求 (PVC) によって参照されます。

前提条件

- Azure File 共有があること。
- この共有にアクセスするための認証情報 (とくにストレージアカウントおよびキー) が利用可能であること。

手順

1. Azure File の認証情報が含まれる **Secret** オブジェクトを作成します。

```
$ oc create secret generic <secret-name> --from-literal=azurestorageaccountname=
<storage-account> \ 1
  --from-literal=azurestorageaccountkey=<storage-account-key> 2
```

1 Azure File ストレージアカウントの名前。

2 Azure File ストレージアカウントキー。

2. 作成した **Secret** オブジェクトを参照する **PersistentVolume** を作成します。

```
apiVersion: "v1"
kind: "PersistentVolume"
metadata:
  name: "pv0001" 1
spec:
  capacity:
    storage: "5Gi" 2
  accessModes:
    - "ReadWriteOnce"
  storageClassName: azure-file-sc
  azureFile:
```

```
    secretName: <secret-name> 3
    shareName: share-1 4
    readOnly: false
```

1. 永続ボリュームの名前。
2. この永続ボリュームのサイズ。
3. Azure File 共有の認証情報を含むシークレットの名前。
4. Azure File 共有の名前。

3. 作成した永続ボリュームにマップする **PersistentVolumeClaim** オブジェクトを作成します。

```
apiVersion: "v1"
kind: "PersistentVolumeClaim"
metadata:
  name: "claim1" 1
spec:
  accessModes:
    - "ReadWriteOnce"
  resources:
    requests:
      storage: "5Gi" 2
  storageClassName: azure-file-sc 3
  volumeName: "pv0001" 4
```

1. 永続ボリューム要求 (PVC) の名前。
2. この永続ボリューム要求 (PVC) のサイズ。
3. 永続ボリュームのプロビジョニングに使用されるストレージクラスの名前。**PersistentVolume** 定義で使用されるストレージクラスを指定します。
4. Azure File 共有を参照する既存の **PersistentVolume** オブジェクトの名前。

3.3.2. Azure File 共有の Pod へのマウント

永続ボリューム要求 (PVC) の作成後に、これをアプリケーション内で使用できます。以下の例は、この共有を Pod 内にマウントする方法を示しています。

前提条件

- 基礎となる Azure File 共有にマップされる永続ボリューム要求 (PVC) があること。

手順

- 既存の永続ボリューム要求 (PVC) をマウントする Pod を作成します。

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-name 1
```

```
spec:
  containers:
    ...
    volumeMounts:
    - mountPath: "/data" 2
      name: azure-file-share
  volumes:
    - name: azure-file-share
      persistentVolumeClaim:
        claimName: claim1 3
```

1. Pod の名前。
2. Pod 内に Azure File 共有をマウントするパス。
3. 以前に作成された **PersistentVolumeClaim** オブジェクトの名前。

3.4. CINDER を使用した永続ストレージ

OpenShift Container Platform は OpenStack Cinder をサポートします。これには、Kubernetes と OpenStack についてある程度の理解があることが前提となります。

Cinder ボリュームは動的にプロビジョニングできます。永続ボリュームは単一のプロジェクトまたは namespace にバインドされず、それらは OpenShift Container Platform クラスター間で共有できます。永続ボリューム要求 (PVC) はプロジェクトまたは namespace に固有のもので、ユーザーによって要求されます。

関連情報

- OpenStack Block Storage が仮想ハードドライブの永続ブロックストレージ管理を提供する方法についての詳細は、OpenStack Cinder を参照してください。

3.4.1. Cinder を使用した手動プロビジョニング

ストレージは、ボリュームとして OpenShift Container Platform にマウントされる前に基礎となるインフラストラクチャーになければなりません。

前提条件

- Red Hat OpenStack Platform (RHOSP) 用に設定された OpenShift Container Platform
- Cinder ボリューム ID

3.4.1.1. 永続ボリュームの作成

OpenShift Container Platform に永続ボリューム (PV) を作成する前に、オブジェクト定義でこれを定義する必要があります。

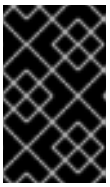
手順

1. オブジェクト定義をファイルに保存します。

   cinder-persistentvolume.yaml

```
apiVersion: "v1"
kind: "PersistentVolume"
metadata:
  name: "pv0001" (1)
spec:
  capacity:
    storage: "5Gi" (2)
  accessModes:
    - "ReadWriteOnce"
  cinder: (3)
    fsType: "ext3" (4)
    volumeID: "f37a03aa-6212-4c62-a805-9ce139fab180" (5)
```

1. 永続ボリューム要求 (PVC) または Pod によって使用されるボリュームの名前。
2. このボリュームに割り当てられるストレージの量。
3. Red Hat OpenStack Platform (RHOSP) Cinder ボリュームの **cinder** を示します。
4. ボリュームの初回マウント時に作成されるファイルシステム。
5. 使用する Cinder ボリューム



重要

ボリュームをフォーマットしてプロビジョニングした後には、**fstype** パラメーターの値は変更しないでください。この値を変更すると、データの損失や、Pod の障害につながる可能性があります。

2. 前のステップで保存したオブジェクト定義ファイルを作成します。

```
$ oc create -f cinder-persistentvolume.yaml
```

3.4.1.2. 永続ボリュームのフォーマット

OpenShift Container Platform は初回の使用前にフォーマットするため、フォーマットされていない Cinder ボリュームを PV として使用できます。

OpenShift Container Platform がボリュームをマウントし、これをコンテナーに渡す前に、システムは PV 定義の **fsType** パラメーターで指定されたファイルシステムがボリュームに含まれるかどうかをチェックします。デバイスが指定されたファイルシステムでフォーマットされていない場合、デバイスのデータはすべて消去され、デバイスはそのファイルシステムで自動的にフォーマットされます。

3.4.1.3. Cinder ボリュームのセキュリティー

お使いのアプリケーションで Cinder PV を使用する場合に、そのデプロイメント設定にセキュリティーを追加します。

前提条件

- 適切な **fsGroup** ストラテジーを使用する SCC が作成される必要があります。

手順

手順

1. サービスアカウントを作成して、そのアカウントを SCC に追加します。

```
$ oc create serviceaccount <service_account>
```

```
$ oc adm policy add-scc-to-user <new_scc> -z <service_account> -n <project>
```

2. アプリケーションのデプロイ設定で、サービスアカウント名と **securityContext** を指定します。

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: frontend-1
spec:
  replicas: 1 (1)
  selector: (2)
    name: frontend
  template: (3)
    metadata:
      labels: (4)
        name: frontend (5)
    spec:
      containers:
      - image: openshift/hello-openshift
        name: helloworld
        ports:
        - containerPort: 8080
          protocol: TCP
      restartPolicy: Always
      serviceAccountName: <service_account> (6)
      securityContext:
        fsGroup: 7777 (7)
```

(1) 実行する Pod のコピー数です。

(2) 実行する Pod のラベルセレクターです。

(3) コントローラーが作成する Pod のテンプレート。

(4) Pod のラベル。ラベルセレクターからのラベルを組み込む必要があります。

(5) パラメーター拡張後の名前の最大長さは 63 文字です。

(6) 作成したサービスアカウントを指定します。

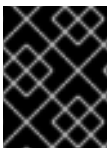
(7) Pod の **fsGroup** を指定します。

3.5. ファイバーチャネルを使用した永続ストレージ

OpenShift Container Platform ではファイバーチャネルがサポートされており、ファイバーチャネルボリュームを使用して OpenShift Container Platform クラスターに永続ストレージをプロビジョニングで

きます。これには、Kubernetes と Fibre Channel についてある程度の理解があることが前提となります。

Kubernetes 永続ボリュームフレームワークは、管理者がクラスターのプロビジョニングを永続ストレージを使用して実行できるようにし、ユーザーが基礎となるインフラストラクチャーの知識がなくてもこれらのリソースを要求できるようにします。永続ボリュームは単一のプロジェクトまたは namespace にバインドされず、それらは OpenShift Container Platform クラスター間で共有できます。Persistent volume claim (PVC) はプロジェクトまたは namespace に固有のもので、ユーザーによって要求されます。



重要

インフラストラクチャーにおけるストレージの高可用性は、基礎となるストレージのプロバイダーに委ねられています。

関連情報

- ファイバーチャネルデバイスの使用

3.5.1. プロビジョニング

PersistentVolume API を使用してファイバーチャネルボリュームをプロビジョニングするには、以下が利用可能でなければなりません。

- **targetWWN** (ファイバーチャネルターゲットのワールドワイド名の配列)。
- 有効な LUN 番号。
- ファイルシステムの種類。

永続ボリュームと LUN は 1 対 1 でマッピングされます。

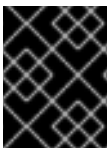
前提条件

- ファイバーチャネル LUN は基礎となるインフラストラクチャーに存在している必要があります。

PersistentVolume オブジェクト定義

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv0001
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  fc:
    targetWWNs: ['500a0981891b8dc5', '500a0981991b8dc5'] 1
    lun: 2
    fsType: ext4
```

1 ファイバーチャネル WWN は、**/dev/disk/by-path/pci-<IDENTIFIER>-fc-0x<WWN>-lun-<LUN#>** として識別されます。ただし、**WWN** までのパス (**0x** を含む) と WWN の後の文字 (**-** (ハイフン) を含む) を入力する必要はありません。



重要

ボリュームをフォーマットしてプロビジョニングした後に **fstype** パラメーターの値を変更すると、データ損失や Pod にエラーが発生する可能性があります。

3.5.1.1. ディスククォータの実施

LUN パーティションを使用してディスククォータとサイズ制限を実施します。各 LUN は単一の永続ボリュームにマップされ、固有の名前を永続ボリュームにに使用する必要があります。

この方法でクォータを実施すると、エンドユーザーは永続ストレージを具体的な量 (10Gi など) で要求することができ、これを同等またはそれ以上の容量の対応するボリュームに一致させることができます。

3.5.1.2. ファイバーチャネルボリュームのセキュリティー

ユーザーは永続ボリューム要求 (PVC) でストレージを要求します。この要求はユーザーの namespace にのみ存在し、同じ namespace 内の Pod からのみ参照できます。namespace をまたいで永続ボリュームにアクセスしようとすると、Pod にエラーが発生します。

それぞれのファイバーチャネル LUN は、クラスター内のすべてのノードからアクセスできる必要があります。

3.6. FLEXVOLUME を使用した永続ストレージ

OpenShift Container Platform は、ドライバーとのインターフェイスに実行可能なモデルを使用する out-of-tree 形式のプラグイン、FlexVolume をサポートします。

組み込みプラグインがないバックエンドのストレージを使用する場合は、FlexVolume ドライバーを使用して OpenShift Container Platform を拡張し、アプリケーションに永続ストレージを提供できます。

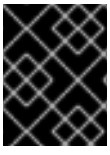
Pod は、**flexvolume** の in-tree 形式のプラグインを使用して FlexVolume ドライバーと対話します。

関連情報

- 永続ボリュームの拡張

3.6.1. FlexVolume ドライバーについて

FlexVolume ドライバーは、クラスター内のすべてのノードの明確に定義されたディレクトリーに格納されている実行可能ファイルです。OpenShift Container Platform は、**flexVolume** をソースとする **PersistentVolume** オブジェクトによって表されるボリュームのマウントまたはアンマウントが必要になるたびに、FlexVolume ドライバーを呼び出します。



重要

OpenShift Container Platform では、FlexVolume について割り当ておよび割り当て解除の操作はサポートされません。

3.6.2. FlexVolume ドライバーの例

FlexVolume ドライバーの最初のコマンドライン引数は常に操作名です。その他のパラメーターは操作ごとに異なります。ほとんどの操作は、JSON (JavaScript Object Notation) 文字列をパラメーターとして取ります。このパラメーターは完全な JSON 文字列であり、JSON データを含むファイルの名前ではありません。

FlexVolume ドライバーには以下が含まれます。

- すべての **flexVolume.options**。
- **kubernetes.io/** という接頭辞が付いた **flexVolume** のいくつかのオプション。たとえば、**fsType** や **readwrite** などです。
- **kubernetes.io/secret/** という接頭辞が付いた参照先シークレット (指定されている場合) の内容。

FlexVolume ドライバーの JSON 入力例

```
{
 "fooServer": "192.168.0.1:1234", 1
       "fooVolumeName": "bar",
 "kubernetes.io/fsType": "ext4", 2
 "kubernetes.io/readwrite": "ro", 3
 "kubernetes.io/secret/<key name>": "<key value>", 4
 "kubernetes.io/secret/<another key name>": "<another key value>",
}
```

1. **flexVolume.options** のすべてのオプション。
2. **flexVolume.fsType** の値。
3. **flexVolume.readOnly** に基づく **ro**/**rw**。
4. **flexVolume.secretRef** によって参照されるシークレットのすべてのキーと値。

OpenShift Container Platform は、ドライバーの標準出力に JSON データが含まれていると想定します。指定されていない場合、出力には操作の結果が示されます。

FlexVolume ドライバーのデフォルトの出力例

```
{
 "status": "<Success/Failure/Not supported>",
 "message": "<Reason for success/failure>"
}
```

ドライバーの終了コードは、成功の場合は **0**、エラーの場合は **1** です。

操作はべき等です。 すでに割り当てられているボリュームのマウント操作は成功します。

3.6.3. FlexVolume ドライバーのインストール

OpenShift Container Platform を拡張するために使用される FlexVolume ドライバーはノードでのみ実行されます。FlexVolume を実装するには、呼び出す操作の一覧とインストールパスのみが必要になります。

前提条件

- FlexVolume ドライバーは、以下の操作を実装する必要があります。

  init

  ドライバーを初期化します。すべてのノードの初期化中に呼び出されます。

  - 引数: なし
  - 実行場所: ノード
  - 予期される出力: デフォルトの JSON

  mount

  ボリュームをディレクトリーにマウントします。これには、デバイスの検出、その後のデバイスのマウントを含む、ボリュームのマウントに必要なあらゆる操作が含まれます。

  - 引数: **<mount-dir> <json>**
  - 実行場所: ノード
  - 予期される出力: デフォルトの JSON

  unmount

  ボリュームをディレクトリーからアンマウントします。これには、アンマウント後にボリュームをクリーンアップするために必要なあらゆる操作が含まれます。

  - 引数: **<mount-dir>**
  - 実行場所: ノード
  - 予期される出力: デフォルトの JSON

  mountdevice

  ボリュームのデバイスを、個々の Pod がマウントをバインドするディレクトリーにマウントします。

この呼び出しでは FlexVolume 仕様に指定されるシークレットを渡しません。ドライバーでシークレットが必要な場合には、この呼び出しを実装しないでください。

- 引数: **<mount-dir> <json>**
- 実行場所: ノード
- 予期される出力: デフォルトの JSON

  unmountdevice

  ボリュームのデバイスをディレクトリーからアンマウントします。

- 引数: **<mount-dir>**

- 実行場所: ノード
- 予期される出力: デフォルトの JSON
  - その他のすべての操作は、**{"status": "Not supported"}** と終了コード **1** を出して JSON を返します。

手順

FlexVolume ドライバーをインストールします。

1. この実行可能ファイルがクラスター内のすべてのノードに存在することを確認します。
2. この実行可能ファイルをボリュームプラグインのパス (**/etc/kubernetes/kubelet-plugins/volume/exec/<vendor>~<driver>/<driver>**) に配置します。

たとえば、ストレージ **foo** の FlexVolume ドライバーをインストールするには、実行可能ファイルを **/etc/kubernetes/kubelet-plugins/volume/exec/openshift.com~foo/foo** に配置します。

3.6.4. FlexVolume ドライバーを使用したストレージの使用

OpenShift Container Platform の各 **PersistentVolume** オブジェクトは、ストレージバックエンドの 1 つのストレージアセット (ボリュームなど) を表します。

手順

- インストールされているストレージを参照するには、**PersistentVolume** オブジェクトを使用します。

FlexVolume ドライバーを使用した永続ボリュームのオブジェクト定義例

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv0001 1
spec:
  capacity:
    storage: 1Gi 2
  accessModes:
    - ReadWriteOnce
  flexVolume:
    driver: openshift.com/foo 3
    fsType: "ext4" 4
    secretRef: foo-secret 5
    readOnly: true 6
    options: 7
      fooServer: 192.168.0.1:1234
      fooVolumeName: bar
```

1. ボリュームの名前。これは永続ボリューム要求 (PVC) を使用するか、または Pod からボリュームを識別するために使用されます。この名前は、バックエンドストレージのボリューム名とは異なるものにすることができます。
2. このボリュームに割り当てられるストレージの量。

3 ドライバーの名前。このフィールドは必須です。

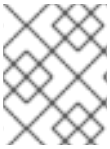
4 ボリュームに存在するオプションのファイルシステム。このフィールドはオプションです。

5 シークレットへの参照。このシークレットのキーと値は、起動時に FlexVolume ドライバーに渡されます。このフィールドはオプションです。

6 読み取り専用のフラグ。このフィールドはオプションです。

7 FlexVolume ドライバーの追加オプション。**options** フィールドでユーザーが指定するフラグに加え、以下のフラグも実行可能ファイルに渡されます。

```
"fsType":"<FS type>",
"readwrite":"<rw>",
"secret/key1":"<secret1>"
...
"secret/keyN":"<secretN>"
```



注記

シークレットは、呼び出しのマウント/マウント解除を目的とする場合にのみ渡されます。

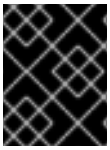
3.7. GCE PERSISTENT DISK を使用した永続ストレージ

OpenShift Container Platform では、GCE Persistent Disk ボリューム (gcePD) がサポートされます。GCE を使用して、OpenShift Container Platform クラスターに永続ストレージをプロビジョニングできます。これには、Kubernetes と GCE についてある程度の理解があることが前提となります。

Kubernetes 永続ボリュームフレームワークは、管理者がクラスターのプロビジョニングを永続ストレージを使用して実行できるようにし、ユーザーが基礎となるインフラストラクチャーの知識がなくてもこれらのリソースを要求できるようにします。

GCE Persistent Disk ボリュームは動的にプロビジョニングできます。

永続ボリュームは単一のプロジェクトまたは namespace にバインドされず、それらは OpenShift Container Platform クラスター間で共有できます。Persistent volume claim (PVC) はプロジェクトまたは namespace に固有のもので、ユーザーによって要求されます。



重要

インフラストラクチャーにおけるストレージの高可用性は、基礎となるストレージのプロバイダーに委ねられています。

関連情報

- GCE Persistent Disk

3.7.1. GCE ストレージクラスの作成

ストレージクラスを使用すると、ストレージのレベルや使用状況を区別し、記述することができます。ストレージクラスを定義することにより、ユーザーは動的にプロビジョニングされた永続ボリュームを取得できます。

手順

1. OpenShift Container Platform コンソールで、**Storage → Storage Classes** をクリックします。
2. ストレージクラスの概要では、**Create Storage Class** をクリックします。
3. 表示されるページで必要なオプションを定義します。
   a. ストレージクラスを参照するための名前を入力します。
   b. オプションの説明を入力します。
   c. 回収ポリシーを選択します。
   d. ドロップダウンリストから **kubernetes.io/gce-pd** を選択します。
   e. 必要に応じてストレージクラスの追加パラメーターを入力します。
4. **Create** をクリックしてストレージクラスを作成します。

3.7.2. 永続ボリューム要求 (PVC) の作成

前提条件

ストレージは、ボリュームとして OpenShift Container Platform にマウントされる前に基礎となるインフラストラクチャーになければなりません。

手順

1. OpenShift Container Platform コンソールで、**Storage → Persistent Volume Claims** をクリックします。
2. 永続ボリューム要求 (PVC) の概要で、**Create Persistent Volume Claim** をクリックします。
3. 表示されるページで必要なオプションを定義します。
   a. ドロップダウンメニューから以前に作成されたストレージクラスを選択します。
   b. ストレージ要求の一意の名前を入力します。
   c. アクセスモードを選択します。これにより、作成されたストレージ要求の読み取り/書き込みアクセスが決定されます。
   d. ストレージ要求のサイズを定義します。
4. **Create** をクリックして永続ボリューム要求 (PVC) を作成し、永続ボリュームを生成します。

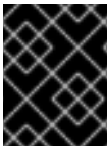
3.7.3. ボリュームのフォーマット

OpenShift Container Platform は、ボリュームをマウントしてコンテナーに渡す前に、永続ボリューム定義の **fsType** パラメーターで指定されたファイルシステムがボリュームにあるかどうか確認します。デバイスが指定されたファイルシステムでフォーマットされていない場合、デバイスのデータはすべて消去され、デバイスはそのファイルシステムで自動的にフォーマットされます。

これにより、OpenShift Container Platform がフォーマットされていない GCE ボリュームを初回の使用前にフォーマットするため、それらを永続ボリュームとして使用することが可能になります。

3.8. HOSTPATH を使用した永続ストレージ

OpenShift Container Platform クラスター内の hostPath ボリュームは、ファイルまたはディレクトリーをホストノードのファイルシステムから Pod にマウントします。ほとんどの Pod には hostPath ボリュームは必要ありませんが、アプリケーションが必要とする場合は、テスト用のクイックオプションが提供されます。



重要

クラスター管理者は、特権付き Pod として実行するように Pod を設定する必要があります。これにより、同じノードの Pod へのアクセスが付与されます。

3.8.1. 概要

OpenShift Container Platform は単一ノードクラスターでの開発およびテスト用の hostPath マウントをサポートします。

実稼働クラスターでは、hostPath を使用しません。代わりにクラスター管理者は、GCE Persistent Disk ボリューム、NFS 共有、Amazon EBS ボリュームなどのネットワークリソースをプロビジョニングします。ネットワークリソースは、ストレージクラスを使用した動的プロビジョニングの設定をサポートします。

hostPath ボリュームは静的にプロビジョニングする必要があります。

3.8.2. hostPath ボリュームの静的プロビジョニング

hostPath ボリュームを使用する Pod は、手動の (静的) プロビジョニングで参照される必要があります。

手順

1. 永続ボリューム (PV) を定義します。**PersistentVolume** オブジェクト定義を使用して **pv.yaml** ファイルを作成します。

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: task-pv-volume (1)
  labels:
    type: local
spec:
  storageClassName: manual (2)
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteOnce (3)
  persistentVolumeReclaimPolicy: Retain
  hostPath:
    path: "/mnt/data" (4)
```

(1) ボリュームの名前。この名前は永続ボリューム要求 (PVC) または Pod で識別されるものです。

(2) 永続ボリューム要求 (PVC) をこの永続ボリュームにバインドするために使用されます。

3 ボリュームは単一ノードで **read-write** としてマウントできます。

4 設定ファイルでは、ボリュームがクラスターのノードの **/mnt/data** にあるように指定します。

2. ファイルから PV を作成します。

```
$ oc create -f pv.yaml
```

3. 永続ボリューム要求 (PVC) を定義します。**PersistentVolumeClaim** オブジェクト定義を使用して、ファイル **pvc.yaml** を作成します。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: task-pvc-volume
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: manual
```

4. ファイルから PVC を作成します。

```
$ oc create -f pvc.yaml
```

3.8.3. 特権付き Pod での hostPath 共有のマウント

永続ボリューム要求 (PVC) の作成後に、これをアプリケーション内で使用できます。以下の例は、この共有を Pod 内にマウントする方法を示しています。

前提条件

- 基礎となる hostPath 共有にマップされる永続ボリューム要求 (PVC) があること。

手順

- 既存の永続ボリューム要求 (PVC) をマウントする特権付き Pod を作成します。

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-name 1
spec:
  containers:
    ...
    securityContext:
      privileged: true 2
    volumeMounts:
    - mountPath: /data 3
      name: hostpath-privileged
```

```
...
securityContext: {}
volumes:
  - name: hostpath-privileged
    persistentVolumeClaim:
      claimName: task-pvc-volume 4
```

1. Pod の名前。
2. Pod は、ノードのストレージにアクセスするために特権付き Pod として実行される必要があります。
3. 特権付き Pod 内に hostPath 共有をマウントするパス。
4. 以前に作成された **PersistentVolumeClaim** オブジェクトの名前。

3.9. ISCSI を使用した永続ストレージ

iSCSI を使用して、OpenShift Container Platform クラスターに永続ストレージをプロビジョニングできます。これには、Kubernetes と iSCSI についてある程度の理解があることが前提となります。

Kubernetes 永続ボリュームフレームワークは、管理者がクラスターのプロビジョニングを永続ストレージを使用して実行できるようにし、ユーザーが基礎となるインフラストラクチャーの知識がなくてもこれらのリソースを要求できるようにします。

重要

インフラストラクチャーにおけるストレージの高可用性は、基礎となるストレージのプロバイダーに委ねられています。

重要

Amazon Web Services で iSCSI を使用する場合、iSCSI ポートのノード間の TCP トラフィックを組み込むようにデフォルトのセキュリティーポリシーを更新する必要があります。デフォルトで、それらのポートは **860** および **3260** です。

重要

OpenShift では、クラスターのすべてのノードが iSCSI イニシエーターをすでに設定している、つまり、 **iscsi-initiator-utils** パッケージをインストールし、それらのイニシエーターの名前を **/etc/iscsi/initiatorname.iscsi** に設定していることを前提とします。上記にリンクしたストレージ管理ガイドを参照してください。

3.9.1. プロビジョニング

OpenShift Container Platform でストレージをボリュームとしてマウントする前に、基礎となるインフラストラクチャーにストレージが存在することを確認します。iSCSI に必要になるのは、iSCSI ターゲットポータル、有効な iSCSI 修飾名 (IQN)、有効な LUN 番号、ファイルシステムタイプ、および **PersistentVolume** API のみです。

PersistentVolume オブジェクト定義

```
apiVersion: v1
```

```
kind: PersistentVolume
metadata:
  name: iscsi-pv
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  iscsi:
      targetPortal: 10.16.154.81:3260
      iqn: iqn.2014-12.example.server:storage.target00
      lun: 0
      fsType: 'ext4'
```

3.9.2. ディスククォータの実施

LUN パーティションを使用してディスククォータとサイズ制限を実施します。それぞれの LUN には 1 つの永続ボリュームです。Kubernetes では、永続ボリュームに一意の名前を使用する必要があります。

上記の方法でクォータを実施すると、エンドユーザーは永続ストレージを具体的な量 (10Gi など) で要求することができ、これを同等またはそれ以上の容量の対応するボリュームに一致させることができます。

3.9.3. iSCSI ボリュームのセキュリティー

ユーザーは **PersistentVolumeClaim** オブジェクトでストレージを要求します。この要求はユーザーの namespace にのみ存在し、同じ namespace 内の Pod からのみ参照できます。namespace をまたいで永続ボリューム要求 (PVC) にアクセスしようとすると、Pod にエラーが発生します。

それぞれの iSCSI LUN は、クラスター内のすべてのノードからアクセスできる必要があります。

3.9.3.1. チャレンジハンドシェイク認証プロトコル (CHAP) 設定

オプションで、OpenShift は CHAP を使用して iSCSI ターゲットに対して自己認証を実行できます。

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: iscsi-pv
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  iscsi:
    targetPortal: 10.0.0.1:3260
    iqn: iqn.2016-04.test.com:storage.target00
    lun: 0
    fsType: ext4
    chapAuthDiscovery: true 1
    chapAuthSession: true 2
    secretRef:
      name: chap-secret 3
```

1. iSCSI 検出の CHAP 認証を有効にします。
2. iSCSI セッションの CHAP 認証を有効にします。
3. ユーザー名 +パスワードを使用してシークレットオブジェクトの名前を指定します。この **Secret** オブジェクトは、参照されるボリュームを使用できるすべての namespace で利用可能でなければなりません。

3.9.4. iSCSI のマルチパス化

iSCSI ベースのストレージの場合は、複数のターゲットポータルの IP アドレスに同じ IQN を使用することでマルチパスを設定できます。マルチパス化により、パス内の 1 つ以上のコンポーネントで障害が発生した場合でも、永続ボリュームにアクセスすることができます。

Pod 仕様でマルチパスを指定するには、**portals** フィールドを使用します。以下に例を示します。

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: iscsi-pv
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  iscsi:
    targetPortal: 10.0.0.1:3260
    portals: ['10.0.2.16:3260', '10.0.2.17:3260', '10.0.2.18:3260'] 1
    iqn: iqn.2016-04.test.com:storage.target00
    lun: 0
    fsType: ext4
    readOnly: false
```

1. **portals** フィールドを使用してターゲットポータルを追加します。

3.9.5. iSCSI のカスタムイニシエーター IQN

iSCSI ターゲットが特定に IQN に制限されている場合に、カスタムイニシエーターの iSCSI Qualified Name (IQN) を設定します。 ただし、iSCSI PV が割り当てられているノードが必ずこれらの IQN を使用する保証はありません。

カスタムのイニシエーター IQN を指定するには、**initiatorName** フィールドを使用します。

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: iscsi-pv
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  iscsi:
```

```
targetPortal: 10.0.0.1:3260
portals: ['10.0.2.16:3260', '10.0.2.17:3260', '10.0.2.18:3260']
iqn: iqn.2016-04.test.com:storage.target00
lun: 0
initiatorName: iqn.2016-04.test.com:custom.iqn (1)
fsType: ext4
readOnly: false
```

(1) イニシエーターの名前を指定します。

3.10. ローカルボリュームを使用した永続ストレージ

OpenShift Container Platform は、ローカルボリュームを使用する永続ストレージでプロビジョニングすることが可能です。ローカルの永続ボリュームを使用すると、標準の PVC インターフェイスを使用して、ディスクやパーティションなどのローカルのストレージデバイスにアクセスできます。

ローカルボリュームは、Pod をノードに手動でスケジュールせずに使用できます。ボリュームのノード制約がシステムによって認識されるためです。ただし、ローカルボリュームは、依然として基礎となるノードの可用性に依存しており、すべてのアプリケーションに適している訳ではありません。

注記

ローカルボリュームは、静的に作成された永続ボリュームとしてのみ使用できます。

3.10.1. ローカルストレージ Operator のインストール

ローカルストレージ Operator はデフォルトで OpenShift Container Platform にインストールされません。以下の手順を使用してこの Operator をインストールし、クラスター内でローカルボリュームを有効にできるように設定します。

前提条件

- OpenShift Container Platform Web コンソールまたはコマンドラインインターフェイス (CLI) へのアクセス。

手順

1. **local-storage** プロジェクトを作成します。

```
$ oc new-project local-storage
```

2. オプション: インフラストラクチャーノードでのローカルストレージの作成を許可します。ロギングやモニタリングなどのコンポーネントに対応するために、ローカルストレージ Operator を使用してインフラストラクチャーノードでボリュームを作成する必要がある場合があります。

   ローカルストレージ Operator にワーカーノードだけでなくインフラストラクチャーノードが含まれるように、デフォルトのノードセレクターを調整する必要があります。

   ローカルストレージ Operator がクラスター全体のデフォルトセレクターを継承しないようにするには、以下のコマンドを実行します。

```
$ oc annotate project local-storage openshift.io/node-selector=''
```

-

UI での操作

Web コンソールからローカルストレージ Operator をインストールするには、以下の手順を実行します。

1. OpenShift Container Platform Web コンソールにログインします。
2. **Operators → OperatorHub** に移動します。
3. **Local Storage** をフィルターボックスに入力して、ローカルストレージ Operator を見つけます。
4. **Install** をクリックします。
5. **Install Operator** ページで、**A specific namespace on the cluster**を選択します。ドロップメニューから **local-storage** を選択します。
6. **Update Channel** および **Approval Strategy** の値を必要な値に調整します。
7. **Install** をクリックします。

これが完了すると、ローカルストレージ Operator は Web コンソールの **Installed Operators** セクションに一覧表示されます。

CLI からの操作

1. CLI からローカルストレージ Operator をインストールします。
   a. 以下のコマンドを実行して OpenShift Container Platform のメジャーおよびマイナーバージョンを取得します。これは、次の手順の **channel** の値に必要です。

```
$ OC_VERSION=$(oc version -o yaml | grep openshiftVersion | \
    grep -o '[0-9]*[.][0-9]*' | head -1)
```

   b. ローカルストレージ Operator の Operator グループおよびサブスクリプションを定義するために、オブジェクト YAML ファイル (例: **local-storage.yaml**) を作成します。

local-storage の例

```
apiVersion: operators.coreos.com/v1alpha2
kind: OperatorGroup
metadata:
  name: local-operator-group
  namespace: local-storage
spec:
  targetNamespaces:
    - local-storage
---
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: local-storage-operator
  namespace: local-storage
spec:
  channel: "${OC_VERSION}"
```

```
installPlanApproval: Automatic (1)
name: local-storage-operator
source: redhat-operators
sourceNamespace: openshift-marketplace
```

(1) インストール計画のユーザー承認ポリシー。

2. 以下のコマンドを実行して、ローカルストレージ Operator オブジェクトを作成します。

```
$ oc apply -f local-storage.yaml
```

この時点で、Operator Lifecycle Manager (OLM) はローカルストレージ Operator を認識できるようになります。Operator の ClusterServiceVersion (CSV) はターゲット namespace に表示され、Operator で指定される API は作成用に利用可能になります。

3. すべての Pod およびローカルストレージ Operator が作成されていることを確認して、ローカルストレージのインストールを検証します。

a. 必要な Pod すべてが作成されていることを確認します。

```
$ oc -n local-storage get pods
```

出力例

```
NAME                                  READY   STATUS    RESTARTS   AGE
local-storage-operator-746bf599c9-vlt5t   1/1     Running   0          19m
```

b. ClusterServiceVersion (CSV) YAML マニフェストをチェックして、ローカルストレージ Operator が **local-storage** プロジェクトで利用できることを確認します。

```
$ oc get csvs -n local-storage
```

出力例

```
NAME                                         DISPLAY         VERSION               REPLACES   PHASE
local-storage-operator.4.2.26-202003230335   Local Storage   4.2.26-202003230335              Succeeded
```

すべてのチェックが渡されると、ローカルストレージ Operator が正常にインストールされます。

3.10.2. ローカルストレージ Operator を使用したローカルボリュームのプロビジョニング

ローカルボリュームは動的プロビジョニングで作成できません。代わりに、永続ボリュームがローカルストレージ Operator によって作成されることがあります。このローカルボリュームプロビジョナーは、定義されたリソースで指定されているパスでファイルシステムまたはブロックボリュームデバイスを検索します。

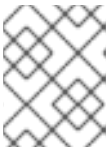
前提条件

- ローカルストレージ Operator がインストールされていること。

- ローカルディスクが OpenShift Container Platform ノードに割り当てられていること。

手順

1. ローカルボリュームリソースを作成します。これは、ノードおよびローカルボリュームへのパスを定義する必要があります。



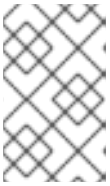
注記

同じデバイスに別のストレージクラス名を使用しないでください。これを行うと、複数の永続ボリューム (PV) が作成されます。

例: ファイルシステム

```
apiVersion: "local.storage.openshift.io/v1"
kind: "LocalVolume"
metadata:
  name: "local-disks"
  namespace: "local-storage" 1
spec:
  nodeSelector: 2
    nodeSelectorTerms:
    - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - ip-10-0-140-183
          - ip-10-0-158-139
          - ip-10-0-164-33
  storageClassDevices:
    - storageClassName: "local-sc"
      volumeMode: Filesystem 3
      fsType: xfs 4
      devicePaths: 5
        - /path/to/device 6
```

1. ローカルストレージ Operator がインストールされている namespace。
2. オプション: ローカルストレージボリュームが割り当てられているノードの一覧が含まれるノードセレクター。以下の例では、**oc get node** から取得したノードホスト名を使用します。値が定義されない場合、ローカルストレージ Operator は利用可能なすべてのノードで一致するディスクの検索を試行します。
3. ボリュームモード (**Filesystem** または **Block**) で、ローカルボリュームのタイプを定義します。
4. ローカルボリュームの初回マウント時に作成されるファイルシステム。
5. 選択するローカルストレージデバイスの一覧を含むパスです。
6. この値を、**/dev/xvdg** などの LocalVolume リソースへの実際のローカルディスクのファイルパスに置き換えます。プロビジョナーが正常にデプロイされると、これらのローカルディスク用に PV が作成されます。



注記

raw ブロックボリューム (**volumeMode: block**) はファイルシステムでフォーマットされません。このモードは、Pod で実行しているすべてのアプリケーションが raw ブロックデバイスを使用できる場合にのみ使用してください。

例: ブロック

```
apiVersion: "local.storage.openshift.io/v1"
kind: "LocalVolume"
metadata:
  name: "local-disks"
  namespace: "local-storage" 1
spec:
  nodeSelector: 2
    nodeSelectorTerms:
    - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - ip-10-0-136-143
          - ip-10-0-140-255
          - ip-10-0-144-180
  storageClassDevices:
    - storageClassName: "localblock-sc"
      volumeMode: Block 3
      devicePaths: 4
        - /path/to/device 5
```

1. ローカルストレージ Operator がインストールされている namespace。
2. オプション: ローカルストレージボリュームが割り当てられているノードの一覧が含まれるノードセレクター。以下の例では、**oc get node** から取得したノードホスト名を使用します。値が定義されない場合、ローカルストレージ Operator は利用可能なすべてのノードで一致するディスクの検索を試行します。
3. ボリュームモード (**Filesystem** または **Block**) で、ローカルボリュームのタイプを定義します。
4. 選択するローカルストレージデバイスの一覧を含むパスです。
5. この値を、**/dev/xvdg** などの LocalVolume リソースへの実際のローカルディスクのファイルパスに置き換えます。プロビジョナーが正常にデプロイされると、これらのローカルディスク用に PV が作成されます。

2. 先に作成したファイルを指定して、OpenShift Container Platform クラスターにローカルボリュームリソースを作成します。

```
$ oc create -f <local-volume>.yaml
```

3. プロビジョナーが作成され、対応するデーモンセットが作成されていることを確認します。

```
$ oc get all -n local-storage
```

出力例

```
NAME                                        READY   STATUS    RESTARTS   AGE
pod/local-disks-local-provisioner-h97hj     1/1     Running   0          46m
pod/local-disks-local-provisioner-j4mnn     1/1     Running   0          46m
pod/local-disks-local-provisioner-kbdnx     1/1     Running   0          46m
pod/local-disks-local-diskmaker-ldldw       1/1     Running   0          46m
pod/local-disks-local-diskmaker-lvrv4       1/1     Running   0          46m
pod/local-disks-local-diskmaker-phxdq       1/1     Running   0          46m
pod/local-storage-operator-54564d9988-vxvhx 1/1     Running   0          47m

NAME                                  TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)     AGE
service/local-storage-operator        ClusterIP   172.30.49.90     <none>        60000/TCP   47m

NAME                                        DESIRED   CURRENT   READY   UP-TO-DATE
AVAILABLE   NODE SELECTOR   AGE
daemonset.apps/local-disks-local-provisioner   3         3         3       3
3           <none>
46m
daemonset.apps/local-disks-local-diskmaker     3         3         3       3
3           <none>
46m

NAME                                   READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/local-storage-operator   1/1     1            1           47m

NAME                                              DESIRED   CURRENT   READY   AGE
replicaset.apps/local-storage-operator-54564d9988   1         1         1       47m
```

デーモンセットプロセスの必要な数と現在の数に注意してください。必要な数が **0** の場合、これはラベルセレクターが無効であることを示します。

4. 永続ボリュームが作成されていることを確認します。

```
$ oc get pv
```

出力例

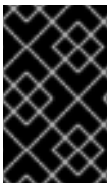
```
NAME                CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM
STORAGECLASS    REASON   AGE
local-pv-1cec77cf   100Gi      RWO            Delete           Available           local-sc                88m
local-pv-2ef7cd2a   100Gi      RWO            Delete           Available           local-sc
82m
local-pv-3fa1c73    100Gi      RWO            Delete           Available           local-sc                48m
```

重要

LocalVolume オブジェクトを編集しても、既存の永続ボリュームの **fsType** または **volumeMode** は変更されません。これが破壊的な操作になる可能性があるためです。

3.10.3. ローカルストレージ Operator のないローカルボリュームのプロビジョニング

ローカルボリュームは動的プロビジョニングで作成できません。代わりに、永続ボリュームは、永続ボリューム (PV) をオブジェクト定義に定義して作成できます。このローカルボリュームプロビジョナーは、定義されたリソースで指定されているパスでファイルシステムまたはブロックボリュームデバイスを検索します。



重要

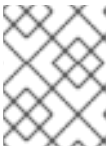
PV の手動プロビジョニングには、PVC の削除時に PV 全体でデータ漏洩が発生するリスクが含まれます。ローカルストレージ Operator は、ローカル PV のプロビジョニング時にデバイスのライフサイクルを自動化するために使用することが推奨されます。

前提条件

- ローカルディスクが OpenShift Container Platform ノードに割り当てられていること。

手順

1. PV を定義します。**PersistentVolume** オブジェクト定義を使用して、**example-pv-filesystem.yaml** または **example-pv-block.yaml** などのファイルを作成します。このリソースは、ノードおよびローカルボリュームへのパスを定義する必要があります。



注記

同じデバイスに別のストレージクラス名を使用しないでください。同じ名前を使用すると、複数の PV が作成されます。

example-pv-filesystem.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: example-pv-filesystem
spec:
  capacity:
    storage: 100Gi
  volumeMode: Filesystem (1)
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Delete
  storageClassName: local-storage (2)
  local:
    path: /dev/xvdf (3)
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - example-node
```

(1) PV のタイプを定義するボリュームモード (**Filesystem** または **Block**)。

(2) PV リソースの作成時に使用するストレージクラスの名前。この PV のセットを一意に特定するストレージクラスを使用にしてください。

(3) 選択するローカルストレージデバイスの一覧を含むパスです。

注記

raw ブロックボリューム (**volumeMode: block**) はファイルシステムでフォーマットされません。このモードは、Pod で実行しているすべてのアプリケーションが raw ブロックデバイスを使用できる場合にのみ使用します。

example-pv-block.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: example-pv-block
spec:
  capacity:
    storage: 100Gi
  volumeMode: Block (1)
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Delete
  storageClassName: local-storage (2)
  local:
    path: /dev/xvdf (3)
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - example-node
```

(1) PV のタイプを定義するボリュームモード (**Filesystem** または **Block**)。

(2) PV リソースの作成時に使用するストレージクラスの名前。この PV のセットを一意に特定するストレージクラスを使用するようにしてください。

(3) 選択するローカルストレージデバイスの一覧を含むパスです。

2. OpenShift Container Platform クラスターに PV リソースを作成します。作成したばかりのファイルを指定します。

```
$ oc create -f <example-pv>.yaml
```

3. ローカル PV が作成されていることを確認します。

```
$ oc get pv
```

出力例

```
NAME                    CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM
STORAGECLASS    REASON   AGE
example-pv-filesystem   100Gi      RWO            Delete           Available
local-storage            3m47s
```

```
example-pv1   1Gi   RWO   Delete   Bound   local-storage/pvc1   local-
storage   12h
example-pv2   1Gi   RWO   Delete   Bound   local-storage/pvc2   local-
storage   12h
example-pv3   1Gi   RWO   Delete   Bound   local-storage/pvc3   local-
storage   12h
```

3.10.4. ローカルボリュームの永続ボリューム要求 (PVC) の作成

ローカルボリュームは、Pod でアクセスされる永続ボリューム要求 (PVC) として静的に作成される必要があります。

前提条件

- 永続ボリュームがローカルボリュームプロビジョナーを使用して作成されていること。

手順

1. 対応するストレージクラスを使用して PVC を作成します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: local-pvc-name (1)
spec:
  accessModes:
  - ReadWriteOnce
  volumeMode: Filesystem (2)
  resources:
    requests:
      storage: 100Gi (3)
  storageClassName: local-sc (4)
```

(1) PVC の名前。

(2) PVC のタイプ。デフォルトは **Filesystem** です。

(3) PVC に利用できるストレージの量。

(4) 要求で必要になるストレージクラスの名前。

2. 作成したファイルを指定して、PVC を OpenShift Container Platform クラスターに作成します。

```
$ oc create -f <local-pvc>.yaml
```

3.10.5. ローカル要求を割り当てます。

ローカルボリュームが PersistentVolumeClaim (PVC) にマップされた後に、これをリソース内に指定できます。

前提条件

- PVC が同じ namespace に存在する。

手順

1. 定義された要求をリソースの仕様に追加します。以下の例では、Pod 内で PVC を宣言します。

```
apiVersion: v1
kind: Pod
spec:
  ...
  containers:
    volumeMounts:
    - name: localpvc (1)
      mountPath: "/data" (2)
  volumes:
  - name: localpvc
    persistentVolumeClaim:
      claimName: localpvc (3)
```

(1) マウントするボリュームの名前。

(2) ボリュームがマウントされる Pod 内のパス。

(3) 使用する既存 PVC の名前。

2. 作成したファイルを指定して、OpenShift Container Platform クラスターにリソースを作成します。

```
$ oc create -f <local-pod>.yaml
```

3.10.6. ローカルストレージ Operator Pod での容認の使用

テイントはノードに適用し、それらが一般的なワークロードを実行しないようにすることができます。ローカルストレージ Operator がテイントのマークが付けられたノードを使用できるようにするには、容認を **Pod** または **DaemonSet** 定義に追加する必要があります。これにより、作成されたリソースをこれらのテイントのマークが付けられたノードで実行できるようになります。

容認を **LocalVolume** リソースでローカルストレージ Operator Pod に適用し、テイントをノード仕様でノードに適用します。ノードのテイントはノードに対し、テイントを容認しないすべての Pod を拒否するよう指示します。他の Pod にはない特定のテイントを使用することで、ローカルストレージ Operator Pod がそのノードでも実行されるようにできます。

重要

テイントおよび容認は、key、value、および effect で設定されています。引数として、これは **key=value:effect** として表現されます。演算子により、これらの 3 つのパラメーターのいずれかを空のままにすることができます。

前提条件

- ローカルストレージ Operator がインストールされていること。

- ローカルディスクがテイントを持つ OpenShift Container Platform ノードに割り当てられている。
- テイントのマークが付けられたノードがローカルストレージのプロビジョニングを行うことが想定されます。

手順

テイントのマークが付けられたノードでスケジュールするようにローカルボリュームを設定するには、以下を実行します。

1. 以下の例に示されるように、**Pod** を定義する YAML ファイルを変更し、**LocalVolume** 仕様を追加します。

```
apiVersion: "local.storage.openshift.io/v1"
kind: "LocalVolume"
metadata:
  name: "local-disks"
  namespace: "local-storage"
spec:
  tolerations:
    - key: localstorage (1)
      operator: Equal (2)
      value: "localstorage" (3)
  storageClassDevices:
      - storageClassName: "localblock-sc"
        volumeMode: Block (4)
        devicePaths: (5)
          - /dev/xvdg
```

(1) ノードに追加したキーを指定します。

(2) **Equal** Operator を指定して、**key**/**value** パラメーターが一致するようにします。Operator が **Exists** の場合、システムはキーが存在することを確認し、値を無視します。Operator が **Equal** の場合、キーと値が一致する必要があります。

(3) テイントのマークが付けられたノードの値 **local** を指定します。

(4) ボリュームモード (**Filesystem** または **Block**) で、ローカルボリュームのタイプを定義します。

(5) 選択するローカルストレージデバイスの一覧を含むパスです。

定義された容認は結果として作成されるデーモンセットに渡されます。これにより、diskmaker およびプロビジョナー Pod を指定されたテイントが含まれるノード用に作成できます。

3.10.7. ローカルストレージ Operator のリソースの削除

3.10.7.1. ローカルボリュームの削除

ローカルボリュームを削除する必要がある場合があります。LocalVolume リソースのエントリーを削除し、PersistentVolume を削除することで通常は十分ですが、同じデバイスパスを再使用する場合や別のストレージクラスでこれを管理する必要がある場合には、追加の手順が必要になります。

警告

以下の手順では、root ユーザーとしてノードにアクセスします。この手順のステップ以外にノードの状態を変更すると、クラスターが不安定になる可能性があります。

前提条件

- 永続ボリュームの状態は **Released** または **Available** である必要があります。

警告

使用中の永続ボリュームを削除すると、データの損失や破損につながる可能性があります。

手順

1. 以前に作成したローカルボリュームを編集して、不要なディスクを削除します。
   a. クラスターリソースを編集します。

   ```
   $ oc edit localvolume <name> -n local-storage
   ```

   b. **devicePaths** の下の行に移動し、不要なディスクを表すものを削除します。
2. 作成した永続ボリュームを削除します。

   ```
   $ oc delete pv <pv-name>
   ```

3. ノードのシンボリックリンクを削除します。
   a. ノードにデバッグ Pod を作成します。

   ```
   $ oc debug node/<node-name>
   ```

   b. ルートディレクトリーをホストに切り替えます。

   ```
   $ chroot /host
   ```

   c. ローカルボリュームのシンボリックリンクを含むディレクトリーに移動します。

   ```
   $ cd /mnt/local-storage/<sc-name> 1
   ```

   1 ローカルボリュームの作成に使用されるストレージクラスの名前。

d. 削除したデバイスに属するシンボリックリンクを削除します。

```
$ rm <symlink>
```

3.10.7.2. ローカルストレージ Operator のアンインストール

ローカルストレージ Operator をアンインストールするには、Operator および **local-storage** プロジェクトの作成されたすべてのリソースを削除する必要があります。

警告

ローカルストレージ PV がまだ使用中の状態でローカルストレージ Operator をアンインストールすることは推奨されません。PV は Operator の削除後も残りますが、PV およびローカルストレージリソースを削除せずに Operator がアンインストールされ、再インストールされる場合に予測できない動作が生じる可能性があります。

前提条件

- OpenShift Container Platform Web コンソールにアクセスします。

手順

1. プロジェクトのローカルボリュームリソースを削除します。

```
$ oc delete localvolume --all --all-namespaces
```

2. Web コンソールからローカルストレージ Operator をアンインストールします。
   a. OpenShift Container Platform Web コンソールにログインします。
   b. **Operators → Installed Operators** に移動します。
   c. **Local Storage** をフィルターボックスに入力して、ローカルストレージ Operator を見つけます。
   d. ローカルストレージ Operator の末尾にある Options メニュー  をクリックします。
   e. **Uninstall Operator** をクリックします。
   f. 表示されるウィンドウで **Remove** をクリックします。
3. ローカルストレージ Operator で作成された PV は削除されるまでクラスターに残ります。これらのボリュームが使用されなくなったら、以下のコマンドを実行してこれらのボリュームを削除します。

```
$ oc delete pv <pv-name>
```

4. **local-storage** プロジェクトを削除します。

```
$ oc delete project local-storage
```

3.11. NFS を使用した永続ストレージ

OpenShift Container Platform クラスターは、NFS を使用する永続ストレージでプロビジョニングすることが可能です。永続ボリューム (PV) および永続ボリューム要求 (PVC) は、プロジェクト全体でボリュームを共有するための便利な方法を提供します。PV 定義に含まれる NFS に固有の情報は、**Pod** 定義で直接定義することも可能ですが、この方法の場合にはボリュームが一意のクラスターリソースとして作成されされないため、ボリュームが競合の影響を受けやすくなります。

関連情報

- ネットワークファイルシステム (NFS)

3.11.1. プロビジョニング

ストレージは、ボリュームとして OpenShift Container Platform にマウントされる前に基礎となるインフラストラクチャーになければなりません。NFS ボリュームをプロビジョニングするには、NFS サーバーの一覧とエクスポートパスのみが必要です。

手順

1. PV のオブジェクト定義を作成します。

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv0001 1
spec:
  capacity:
    storage: 5Gi 2
  accessModes:
  - ReadWriteOnce 3
  nfs: 4
    path: /tmp 5
    server: 172.17.0.2 6
  persistentVolumeReclaimPolicy: Retain 7
```

1. ボリュームの名前。これは、各種の **oc <command> pod** コマンドの PV アイデンティティーです。
2. このボリュームに割り当てられるストレージの量。
3. これはボリュームへのアクセスの制御に関連するように見えますが、実際はラベルの場合と同様に、PVC を PV に一致させるために使用されます。現時点では、**accessModes** に基づくアクセスルールは適用されていません。
4. 使用されているボリュームタイプ。 この場合は **nfs** プラグインです。
5. NFS サーバーがエクスポートしているパス。

6 NFS サーバーのホスト名または IP アドレス

7 PV の回収ポリシー。これはボリュームのリリース時に生じることを定義します。

注記

各 NFS ボリュームは、クラスター内のスケジュール可能なすべてのノードによってマウント可能でなければなりません。

2. PV が作成されたことを確認します。

```
$ oc get pv
```

出力例

```
NAME    LABELS    CAPACITY    ACCESSMODES   STATUS      CLAIM   REASON   AGE
pv0001  <none>    5Gi         RWO           Available           31s
```

3. 新規 PV にバインドされる永続ボリューム要求 (PVC) を作成します。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: nfs-claim1
spec:
  accessModes:
    - ReadWriteOnce 1
  resources:
    requests:
      storage: 5Gi 2
```

1 PV について前述されているように、**accessModes** はセキュリティーを実施するのではなく、PV を PVC と一致させるラベルとして機能します。

2 この要求は **5Gi** 以上の容量を提供する PV を検索します。

4. 永続ボリューム要求 (PVC) が作成されたことを確認します。

```
$ oc get pvc
NAME         STATUS   VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   AGE
nfs-claim1   Bound    pv0001   5Gi        RWO            gp2            2m
```

3.11.2. ディスククォータの実施

ディスクパーティションを使用して、ディスククォータとサイズ制限を実施することができます。それぞれのパーティションを独自のエクスポートとすることができ、それぞれのエクスポートは 1 つの PV になります。それぞれのエクスポートは 1 つの PV になります。OpenShift Container Platform は PV に固有の名前を適用しますが、NFS ボリュームのサーバーとパスの一意性については管理者に委ねられています。

この方法でクォータを実施すると、開発者は永続ストレージを具体的な量 (10Gi など) で要求することができ、同等かそれ以上の容量の対応するボリュームに一致させることができます。

3.11.3. NFS ボリュームのセキュリティー

このセクションでは、一致するパーミッションや SELinux の考慮点を含む、NFS ボリュームのセキュリティーについて説明します。ユーザーは、POSIX パーミッションやプロセス UID、補助グループおよび SELinux の基礎的な点を理解している必要があります。

開発者は、**Pod** 定義の **volumes** セクションで、PVC を名前で参照するか、または NFS ボリュームのプラグインを直接参照して NFS ストレージを要求します。

NFS サーバーの **/etc/exports** ファイルにはアクセス可能な NFS ディレクトリーが含まれています。ターゲットの NFS ディレクトリーには、POSIX の所有者とグループ ID があります。OpenShift Container Platform NFS プラグインは、同じ POSIX の所有者とエクスポートされる NFS ディレクトリーにあるパーミッションを使って、コンテナーの NFS ディレクトリーをマウントします。ただし、コンテナーは NFS マウントの所有者と同等の有効な UID では実行されません。 これは期待される動作です。

ターゲットの NFS ディレクトリーが NFS サーバーに表示される場合を例に取って見てみましょう。

```
$ ls -lZ /opt/nfs -d
```

出力例

```
drwxrws---. nfsnobody 5555 unconfined_u:object_r:usr_t:s0   /opt/nfs
```

```
$ id nfsnobody
```

出力例

```
uid=65534(nfsnobody) gid=65534(nfsnobody) groups=65534(nfsnobody)
```

次に、コンテナーは SELinux ラベルに一致し、ディレクトリーにアクセスするために UID の **65534**、**nfsnobody** 所有者、または補助グループの **5555** のいずれかで実行される必要があります。

注記

所有者 ID **65534** は一例として使用されています。NFS の **root_squash** が **root**、uid **0** を **nfsnobody**、uid **65534** にマップしても、NFS エクスポートは任意の所有者 ID を持つことができます。所有者 **65534** は NFS エクスポートには必要ありません。

3.11.3.1. グループ ID

NFS アクセスに対応する際の推奨される方法として、補助グループを使用することができます (NFS エクスポートのパーミッションを変更するオプションがないことを前提としています)。OpenShift Container Platform の補助グループは共有ストレージに使用されます (例: NFS)。これとは対照的に、iSCSI などのブロックストレージは、Pod の **securityContext** で **fsGroup** SCC ストラテジーと **fsGroup** の値を使用します。

注記

永続ストレージへのアクセスを取得するには、通常はユーザー ID ではなく、補助グループ ID を使用することが推奨されます。

ターゲット NFS ディレクトリーの例で使用したグループ ID は **5555** なので、Pod は、**supplementalGroups** を使用してグループ ID を Pod の **securityContext** 定義の下で定義することができます。以下は例になります。

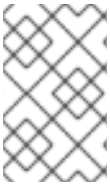
```
spec:
  containers:
    - name:
    ...
  securityContext: 1
    supplementalGroups: [5555] 2
```

1 **securityContext** は特定のコンテナーの下位ではなく、この Pod レベルで定義します。

2 Pod 向けに定義される GID の配列。この場合、配列には 1 つの要素があります。追加の GID はコンマで区切られます。

Pod の要件を満たすカスタム SCC が存在しない場合、Pod は **restricted** SCC に一致する可能性があります。この SCC では、**supplementalGroups** ストラテジーが **RunAsAny** に設定されています。 これは、指定されるグループ ID は範囲のチェックなしに受け入れられることを意味します。

その結果、上記の Pod は受付をパスして起動します。しかし、グループ ID の範囲をチェックすることが望ましい場合は、カスタム SCC の使用が推奨されます。カスタム SCC は、最小および最大のグループ ID が定義され、グループ ID の範囲チェックが実施され、グループ ID の **5555** が許可されるように作成できます。

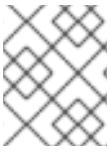


注記

カスタム SCC を使用するには、まずこれを適切なサービスアカウントに追加する必要があります。たとえば、**Pod** 仕様に指定がない場合には、指定されたプロジェクトで **default** サービスアカウントを使用します。

3.11.3.2. ユーザー ID

ユーザー ID は、コンテナーイメージまたは **Pod** 定義で定義することができます。



注記

永続ストレージへのアクセスを取得する場合、通常はユーザー ID ではなく、補助グループ ID を使用することが推奨されます。

上記のターゲット NFS ディレクトリーの例では、コンテナーは UID を **65534** (ここではグループ ID を省略します) に設定する必要があります。 したがって以下を **Pod** 定義に追加することができます。

```
spec:
  containers: 1
  - name:
```

```
...
  securityContext:
    runAsUser: 65534 2
```

1 Pod には、各コンテナーに固有の **securityContext** 定義と、その Pod で定義されたすべてのコンテナーに適用される Pod の **securityContext** が含まれます。

2 **65534** は **nfsnobody** ユーザーです。

プロジェクトが **default** で、SCC が **restricted** の場合、Pod で要求されるユーザー ID の **65534** は許可されません。したがって、Pod は以下の理由で失敗します。

- **65534** をそのユーザー ID として要求する。
- ユーザー ID **65534** を許可する SCC を確認するために Pod で利用できるすべての SCC が検査される。SCC のすべてのポリシーがチェックされますが、ここでのフォーカスはユーザー ID になります。
- 使用可能なすべての SCC が独自の **runAsUser** ストラテジーとして **MustRunAsRange** を使用しているため、UID の範囲チェックが要求される。
- **65534** は SCC またはプロジェクトのユーザー ID 範囲に含まれていない。

一般に、事前定義された SCC は変更しないことが勧められています。ただし、この状況を改善するには、カスタム SCC を作成することが推奨されます。 カスタム SCC は、最小および最大のユーザー ID が定義され、UID 範囲のチェックの実施が設定されており、UID **65534** が許可されるように作成できます。

注記

カスタム SCC を使用するには、まずこれを適切なサービスアカウントに追加する必要があります。たとえば、**Pod** 仕様に指定がない場合には、指定されたプロジェクトで **default** サービスアカウントを使用します。

3.11.3.3. SELinux

Red Hat Enterprise Linux (RHEL) および Red Hat Enterprise Linux CoreOS (RHCOS) システムは、デフォルトでリモートの NFS サーバーで SELinux を使用するように設定されます。

RHEL および RHCOS 以外のシステムの場合、SELinux は Pod からリモートの NFS サーバーへの書き込みを許可しません。NFS ボリュームは正常にマウントされますが、読み取り専用です。以下の手順で、正しい SELinux パーミッションを有効にする必要があります。

前提条件

- **container-selinux** パッケージがインストールされている必要があります。このパッケージは **virt_use_nfs** SELinux ブール値を提供します。

手順

- 以下のコマンドを使用して **virt_use_nfs** ブール値を有効にします。**-P** オプションを使用すると、再起動後もこのブール値を永続化できます。

```
# setsebool -P virt_use_nfs 1
```

3.11.3.4. エクスポート設定

任意のコンテナーユーザーにボリュームの読み取りと書き出しを許可するには、NFS サーバーにエクスポートされる各ボリュームは以下の条件を満たしている必要があります。

- すべてのエクスポートは、次の形式を使用してエクスポートする必要があります。

```
/<example_fs> *(rw,root_squash)
```

- ファイアウォールは、マウントポイントへのトラフィックを許可するように設定する必要があります。
  - NFSv4 の場合、デフォルトのポート **2049** (**nfs**) を設定します。

    NFSv4

```
# iptables -I INPUT 1 -p tcp --dport 2049 -j ACCEPT
```

  - NFSv3 の場合、以下の 3 つのポートを設定します。**2049** (**nfs**)、**20048** (**mountd**)、**111** (**portmapper**)。

    NFSv3

```
# iptables -I INPUT 1 -p tcp --dport 2049 -j ACCEPT
```

```
# iptables -I INPUT 1 -p tcp --dport 20048 -j ACCEPT
```

```
# iptables -I INPUT 1 -p tcp --dport 111 -j ACCEPT
```

- NFS エクスポートとディレクトリーは、ターゲット Pod からアクセスできるようにセットアップされる必要があります。この場合、エクスポートをコンテナーのプライマリー UID で所有されるように設定するか、または上記のグループ ID に示されるように **supplementalGroups** を使用して Pod にグループアクセスを付与します。

3.11.4. リソースの回収

NFS は OpenShift Container Platform の **Recyclable** プラグインインターフェイスを実装します。回収タスクは、それぞれの永続ボリュームに設定されるポリシーに基づいて自動プロセスによって処理されます。

デフォルトで、PV は **Retain** に設定されます。

PV への要求が削除され、PV がリリースされると、PV オブジェクトを再利用できません。代わりに、新規の PV が元のボリュームと同じ基本ボリュームの情報を使って作成されます。

たとえば、管理者は **nfs1** という名前の PV を作成するとします。

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfs1
spec:
  capacity:
```

```
    storage: 1Mi
  accessModes:
    - ReadWriteMany
  nfs:
    server: 192.168.1.1
    path: "/"
```

ユーザーは、**nfs1** にバインドされる **PVC1** を作成します。次にユーザーは **PVC1** を削除し、**nfs1** への要求を解除します。これにより、**nfs1** は **Released** になります。管理者が同じ NFS 共有を利用可能にする必要がある場合には、同じ NFS サーバー情報を使って新規 PV を作成する必要があります。 この場合、PV の名前は元の名前とは異なる名前にします。

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfs2
spec:
  capacity:
    storage: 1Mi
  accessModes:
    - ReadWriteMany
  nfs:
    server: 192.168.1.1
    path: "/"
```

元の PV を削除して、PV を同じ名前で再作成することは推奨されません。PV のステータスを **Released** から **Available** に手動で変更しようとすると、エラーが発生し、データが失われる可能性があります。

3.11.5. その他の設定とトラブルシューティング

適切なエクスポートとセキュリティーマッピングを行うため、使用している NFS のバージョンおよびその設定方法に応じて追加の設定が必要になることがあります。以下は例になります。

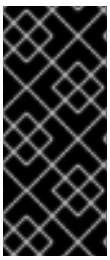
NFSv4 のマウントにすべてのファイルの所有者が **nobody:nobody** と誤って表示される。	• NFS の ID マッピング設定 (**/etc/idmapd.conf**) に原因がある可能性が高い。 • NFSv4 mount incorrectly shows all files with ownership as nobody:nobody を参照してください。
NFSv4 の ID マッピングが無効になっている	• NFS クライアントとサーバーの両方で以下を実行してください。 `# echo 'Y' > /sys/module/nfsd/parameters/nfs4_disable_idmapping`

3.12. RED HAT OPENSHIFT CONTAINER STORAGE

Red Hat OpenShift Container Storage は、インハウスまたはハイブリッドクラウドのいずれの場合でもファイル、ブロックおよびオブジェクトストレージをサポートし、OpenShift Container Platform の

すべてに対応する永続ストレージのプロバイダーです。Red Hat のストレージソリューションとして、Red Hat OpenShift Container Storage は、デプロイメント、管理およびモニターリングを行うために OpenShift Container Platform に完全に統合されています。

Red Hat OpenShift Container Storage は、独自のドキュメントライブラリーを提供します。以下の Red Hat OpenShift Container Storage ドキュメントすべては https://access.redhat.com/documentation/ja-jp/red_hat_openshift_container_storage/4.5/ から入手できます。



重要

OpenShift Container Platform でインストールされた仮想マシンをホストするハイパーコンバージドノードを使用する Red Hat Hyperconverged Infrastructure (RHHI) for Virtualization の上部にある OpenShift Container Storage は、サポートされる設定ではありません。サポートされるプラットフォームについての詳細は、Red Hat OpenShift Container Storage Supportability and Interoperability Guide を参照してください。

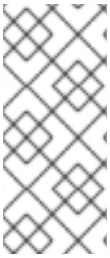
Red Hat OpenShift Container Storage についてのトピック	Red Hat OpenShift Container Storage ドキュメントの参照先
新機能、既知の問題、主なバグ修正およびテクノロジープレビュー	Red Hat OpenShift Container Storage 4.5 リリースノート
サポートされるワークロード、レイアウト、ハードウェアおよびソフトウェア要件、サイジング、スケーリングに関する推奨事項	Red Hat OpenShift Container Storage 4.5 デプロイメントのプランニング
環境がインターネットに直接接続していない場合のデプロイの準備	Preparing to deploy OpenShift Container Storage 4.5 in a disconnected environment
ローカルまたはクラウドストレージの Amazon Web Services を使用した OpenShift Container Storage のデプロイ	Amazon Web Services を使用した OpenShift Container Storage 4.5 のデプロイ
ベアメタルインフラストラクチャーでの OpenShift Container Storage のローカルストレージへのデプロイ	ベアメタルインフラストラクチャーを使用した OpenShift Container Storage 4.5 のデプロイ
OpenShift Container Platform VMWare vSphere クラスターへの OpenShift Container Storage のデプロイ	Deploying OpenShift Container Storage 4.5 using VMware
外部の Red Hat Ceph Storage クラスターを使用するように OpenShift Container Storage をデプロイする	Deploying OpenShift Container Storage 4.5 with external storage
ローカルまたはクラウドストレージの Google Cloud Platform を使用した OpenShift Container Storage のデプロイおよび管理	Google Cloud Platform を使用した OpenShift Container Storage 4.5 のデプロイおよび管理

Red Hat OpenShift Container Storage についてのトピック	Red Hat OpenShift Container Storage ドキュメントの参照先
既存の OpenShift Container Platform Azure クラスターへの OpenShift Container Storage のデプロイおよび管理	Microsoft Azure を使用した OpenShift Container Storage 4.5 のデプロイおよび管理
Red Hat OpenShift Container Storage 4.5 クラスターの管理	Red Hat OpenShift Container Storage 4.5 の管理
Red Hat OpenShift Container Storage 4.5 クラスターのモニターリング	Monitoring Red Hat OpenShift Container Storage 4.5
エラーおよび問題のトラブルシューティング	Red Hat OpenShift Container Storage 4.4 のトラブルシューティング
OpenShift Container Platform クラスターのバージョン 3 からバージョン 4 への移行	Red Hat OpenShift Container Platform 4.5 移行

3.13. VMWARE VSPHERE ボリュームを使用した永続ストレージ

OpenShift Container Platform では、VMWare vSphere の仮想マシンディスク (VMDK: Virtual Machine Disk) ボリュームの使用が可能となります。VMWare vSphere を使用して、OpenShift Container Platform クラスターに永続ストレージをプロビジョニングできます。これには、Kubernetes と VMWare vSphere についてのある程度の理解があることが前提となります。

VMware vSphere ボリュームは動的にプロビジョニングできます。OpenShift Container Platform は vSphere にディスクを作成し、このディスクを正しいイメージに割り当てます。



注記

OpenShift Container Platform は、自由にクラスターないのノードにあるボリュームをアタッチしたり、アタッチ解除できるように、個別の永続ディスクとして新規ボリュームをプロビジョニングします。そのため、スナップショットを使用するボリュームをバックアップしたり、スナップショットからボリュームを復元したりできません。詳細は、スナップショットの制限 を参照してください。

Kubernetes 永続ボリュームフレームワークは、管理者がクラスターのプロビジョニングを永続ストレージを使用して実行できるようにし、ユーザーが基礎となるインフラストラクチャーの知識がなくてもこれらのリソースを要求できるようにします。

永続ボリュームは単一のプロジェクトまたは namespace にバインドされず、それらは OpenShift Container Platform クラスター間で共有できます。永続ボリューム要求 (PVC) はプロジェクトまたは namespace に固有のもので、ユーザーによって要求されます。

関連情報

- VMware vSphere

3.13.1. VMware vSphere ボリュームの動的プロビジョニング

VMware vSphere ボリュームの動的プロビジョニングは推奨される方法です。

3.13.2. 前提条件

- 使用するコンポーネントの要件を満たす VMware vSphere バージョンにインストールされている OpenShift Container Platform クラスター。VSphere バージョンのサポートに関する詳細は、vSphere へのクラスターのインストール について参照してください。

以下のいずれかの手順を使用し、デフォルトのストレージクラスを使用してそれらのボリュームを動的にプロビジョニングできます。

3.13.2.1. UI を使用した VMware vSphere ボリュームの動的プロビジョニング

OpenShift Container Platform は、ボリュームをプロビジョニングするために **thin** ディスク形式を使用する **thin** という名前のデフォルトのストレージクラスをインストールします。

前提条件

- ストレージは、ボリュームとして OpenShift Container Platform にマウントされる前に基礎となるインフラストラクチャーになければなりません。

手順

1. OpenShift Container Platform コンソールで、**Storage → Persistent Volume Claims** をクリックします。
2. 永続ボリューム要求 (PVC) の概要で、**Create Persistent Volume Claim** をクリックします。
3. 結果のページで必要なオプションを定義します。
   a. **thin** ストレージクラスを選択します。
   b. ストレージ要求の一意の名前を入力します。
   c. アクセスモードを選択し、作成されるストレージ要求の読み取り/書き込みアクセスを決定します。
   d. ストレージ要求のサイズを定義します。
4. **Create** をクリックして永続ボリューム要求 (PVC) を作成し、永続ボリュームを生成します。

3.13.2.2. CLI を使用した VMware vSphere ボリュームの動的プロビジョニング

OpenShift Container Platform は、ボリュームをプロビジョニングするために **thin** ディスク形式を使用する **thin** という名前のデフォルトの StorageClass をインストールします。

前提条件

- ストレージは、ボリュームとして OpenShift Container Platform にマウントされる前に基礎となるインフラストラクチャーになければなりません。

手順 (CLI)

1. 以下の内容でファイル **pvc.yaml** を作成して VMware vSphere PersistentVolumeClaim を定義できます。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc (1)
spec:
  accessModes:
  - ReadWriteOnce (2)
  resources:
    requests:
      storage: 1Gi (3)
```

(1) 永続ボリューム要求 (PVC) を表す一意の名前。

(2) 永続ボリューム要求 (PVC) のアクセスモード。**ReadWriteOnce** では、ボリュームは単一ノードによって読み取り/書き込みパーミッションでマウントできます。

(3) 永続ボリューム要求 (PVC) のサイズ。

2. ファイルから **PersistentVolumeClaim** オブジェクトを作成します。

```
$ oc create -f pvc.yaml
```

3.13.3. VMware vSphere ボリュームの静的プロビジョニング

VMware vSphere ボリュームを静的にプロビジョニングするには、永続ボリュームフレームワークが参照する仮想マシンディスクを作成する必要があります。

前提条件

- ストレージは、ボリュームとして OpenShift Container Platform にマウントされる前に基礎となるインフラストラクチャーになければなりません。

手順

1. 仮想マシンディスクを作成します。VMware vSphere ボリュームを静的にプロビジョニングする前に、仮想マシンディスク (VMDK) を手動で作成する必要があります。以下の方法のいずれかを使用します。

   - **vmkfstools** を使用して作成します。セキュアシェル (SSH) を使用して ESX にアクセスし、以下のコマンドを使用して vmdk ボリュームを作成します。

     ```
     $ vmkfstools -c <size> /vmfs/volumes/<datastore-name>/volumes/<disk-name>.vmdk
     ```

   - **vmware-diskmanager** を使用して作成します。

     ```
     $ shell vmware-vdiskmanager -c -t 0 -s <size> -a lsilogic <disk-name>.vmdk
     ```

2. VMDK を参照する永続ボリュームを作成します。**PersistentVolume** オブジェクト定義を使用して **pv1.yaml** ファイルを作成します。

```
apiVersion: v1
kind: PersistentVolume
```

```
metadata:
  name: pv1 (1)
spec:
  capacity:
    storage: 1Gi (2)
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  vsphereVolume: (3)
    volumePath: "[datastore1] volumes/myDisk" (4)
    fsType: ext4 (5)
```

1. ボリュームの名前。この名前は永続ボリューム要求 (PVC) または Pod で識別されるものです。
2. このボリュームに割り当てられるストレージの量。
3. vSphere ボリュームの **vsphereVolume** で使用されるボリュームタイプ。ラベルは vSphere VMDK ボリュームを Pod にマウントするために使用されます。ボリュームの内容はアンマウントされても保持されます。このボリュームタイプは、VMFS データストアと VSAN データストアの両方がサポートされます。
4. 使用する既存の VMDK ボリューム。**vmkfstools** を使用した場合、前述のようにボリューム定義で、データストア名を角かっこ **[]** で囲む必要があります。
5. マウントするファイルシステムタイプです。ext4、xfs、または他のファイルシステムなどが例になります。

重要

ボリュームをフォーマットしてプロビジョニングした後に fsType パラメーターの値を変更すると、データ損失や Pod にエラーが発生する可能性があります。

3. ファイルから **PersistentVolume** オブジェクトを作成します。

```
$ oc create -f pv1.yaml
```

4. 直前の手順で作成した永続ボリュームにマップする永続ボリューム要求 (PVC) を作成します。**PersistentVolumeClaim** オブジェクト定義を使用して、ファイル **pvc1.yaml** を作成します。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc1 (1)
spec:
  accessModes:
    - ReadWriteOnce (2)
  resources:
    requests:
      storage: "1Gi" (3)
  volumeName: pv1 (4)
```

1. 永続ボリューム要求 (PVC) を表す一意の名前。
2. 永続ボリューム要求 (PVC) のアクセスモード。ReadWriteOnce では、ボリュームは単一ノードによって読み取り/書き込みパーミッションでマウントできます。
3. 永続ボリューム要求 (PVC) のサイズ。
4. 既存の永続ボリュームの名前。

5. ファイルから **PersistentVolumeClaim** オブジェクトを作成します。

```
$ oc create -f pvc1.yaml
```

3.13.3.1. VMware vSphere ボリュームのフォーマット

OpenShift Container Platform は、ボリュームをマウントしてコンテナーに渡す前に、**PersistentVolume** (PV) 定義の **fsType** パラメーター値で指定されたファイルシステムがボリュームに含まれることを確認します。デバイスが指定されたファイルシステムでフォーマットされていない場合、デバイスのデータはすべて消去され、デバイスはそのファイルシステムで自動的にフォーマットされます。

OpenShift Container Platform は初回の使用前にフォーマットするため、フォーマットされていない vSphere ボリュームを PV として使用できます。

第4章 CONTAINER STORAGE INTERFACE (CSI) の使用

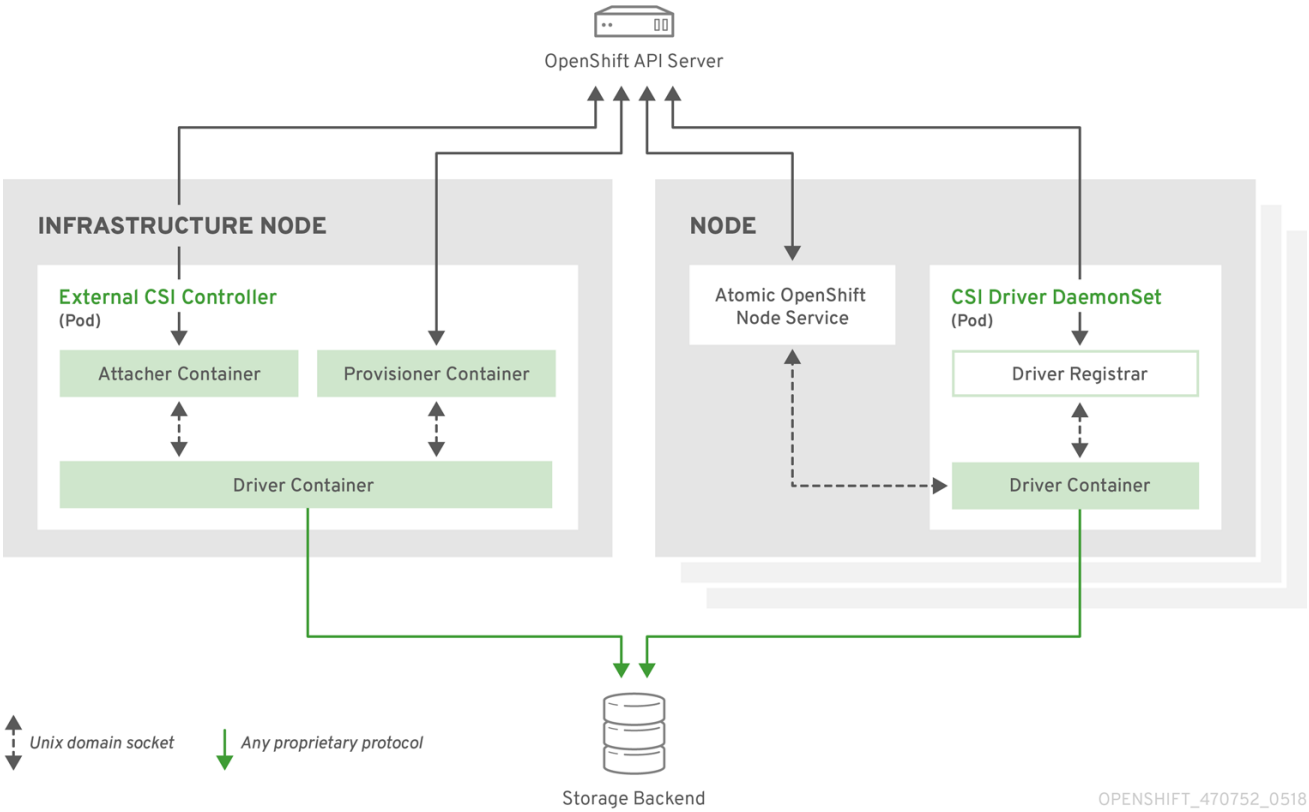
4.1. CSI ボリュームの設定

Container Storage Interface (CSI) により、OpenShift Container Platform は CSI インターフェイス を永続ストレージとして実装するストレージバックエンドからストレージを使用できます。

4.1.1. CSI アーキテクチャー

CSI ドライバーは通常、コンテナーイメージとして提供されます。これらのコンテナーは、実行先の OpenShift Container Platform を認識しません。OpenShift Container Platform でサポートされる CSI 互換のストレージバックエンドを使用するには、クラスター管理者は、OpenShift Container Platform とストレージドライバーの橋渡しとして機能するコンポーネントを複数デプロイする必要があります。

以下の図では、OpenShift Container Platform クラスターの Pod で実行されるコンポーネントの俯瞰図を示しています。



異なるストレージバックエンドに対して複数の CSI ドライバーを実行できます。各ドライバーには、独自の外部コントローラーのデプロイメントおよびドライバーと CSI レジストラーを含むデーモンセットが必要です。

4.1.1.1. 外部の CSI コントローラー

外部の CSI コントローラーは、3 つのコンテナーを含む 1 つまたは複数の Pod を配置するデプロイメントです。

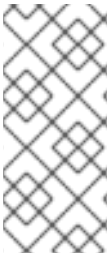
- OpenShift Container Platform からの **attach** および **detach** の呼び出しを適切な CSI ドライバーへの **ControllerPublish** および **ControllerUnpublish** 呼び出しに変換する外部の CSI アタッチャーコンテナー。

- OpenShift Container Platform からの **provision** および **delete** 呼び出しを適切な CSI ドライバーへの **CreateVolume** および **DeleteVolume** 呼び出しに変換する外部の CSI プロビジョナーコンテナー。
- CSI ドライバーコンテナー

CSI アタッチャーおよび CSI プロビジョナーコンテナーは、Unix Domain Socket を使用して、CSI ドライバーコンテナーと通信し、CSI の通信が Pod 外に出ないようにします。CSI ドライバーは Pod 外からはアクセスできません。

注記

通常、**attach**、**detach**、**provision** および **delete** 操作では、CSI ドライバーがストレージバックエンドに対する認証情報を使用する必要があります。CSI コントローラー Pod をインフラストラクチャーノードで実行し、コンピュートノードで致命的なセキュリティー違反が発生した場合でも認証情報がユーザープロセスに漏洩されないようにします。



注記

外部のアタッチャーは、サードパーティーの **attach** または **detach** 操作をサポートしない CSI ドライバーに対しても実行する必要があります。外部のアタッチャーは、CSI ドライバーに対して **ControllerPublish** または **ControllerUnpublish** 操作を実行しません。ただし、必要な OpenShift Container Platform 割り当て API を実装できるように依然として実行する必要があります。

4.1.1.2. CSI ドライバーのデーモンセット

CSI ドライバーのデーモンセットは、OpenShift Container Platform が CSI ドライバーによって提供されるストレージをノードにマウントして、永続ボリューム (PV) としてユーザーワークロード (Pod) で使用できるように、全ノードで Pod を実行します。CSI ドライバーがインストールされた Pod には、以下のコンテナーが含まれます。

- ノード上で実行中の **openshift-node** サービスに CSI ドライバーを登録する CSI ドライバーレジストラー。このノードで実行中の **openshift-node** プロセスは、ノードで利用可能な Unix Domain Socket を使用して CSI ドライバーに直接接続します。
- CSI ドライバー

ノードにデプロイされた CSI ドライバーには、ストレージバックエンドへの認証情報をできる限り少なく指定する必要があります。OpenShift Container Platform は、**NodePublish**/**NodeUnpublish** および **NodeStage**/**NodeUnstage** (実装されている場合) などの CSI 呼び出しのノードプラグインセットのみを使用します。

4.1.2. OpenShift Container Platform でサポートされる CSI ドライバー

OpenShift Container Platform は、in-tree (インツリー) ボリュームプラグインでは不可能なストレージオプションをユーザーに付与する特定の CSI ドライバーをサポートします。

これらのサポートされるストレージアセットにマウントする CSI でプロビジョニングされた永続ボリュームを作成するには、必要な CSI ドライバーおよびストレージクラスをインストールする CSI ドライバー Operator をインストールし、設定できます。Operator およびドライバーのインストールについての詳細は、特定の CSI ドライバー Operator のドキュメントを参照してください。

以下の表は、OpenShift Container Platform で利用可能な CSI ドライバーと、ボリュームスナップショット、クローン作成、およびサイズ変更などの対応する CSI 機能について説明しています。

表4.1 OpenShift Container Platform でサポートされる CSI ドライバーおよび機能

CSI ドライバー	CSI ボリュームスナップショット	CSI のクローン作成	CSI のサイズ変更
AWS EBS (テクノロジープレビュー)	▮	-	▮
OpenStack Manila	▮	▮	▮

重要

CSI ドライバーが上記の表に記載されていない場合は、CSI ストレージベンダーが提供するインストール手順に従って、サポートされている CSI 機能を使用する必要があります。

4.1.3. 動的プロビジョニング

永続ストレージの動的プロビジョニングは、CSI ドライバーおよび基礎となるストレージバックエンドの機能により異なります。CSI ドライバーのプロバイダーは、OpenShift Container Platform でのストレージクラスの作成方法および設定に利用でじるパラメーターについての文書を作成する必要があります。

作成されたストレージクラスは、動的プロビジョニングを有効にするために設定できます。

手順

- デフォルトのストレージクラスを作成します。 これにより、特殊なストレージクラスを必要としないすべての PVC がインストールされた CSI ドライバーでプロビジョニングされます。

```
# oc create -f - << EOF
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: <storage-class> 1
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: <provisioner-name> 2
parameters:
EOF
```

1. 作成されるストレージクラスの名前。
2. インストールされている CSI ドライバーの名前。

4.1.4. CSI ドライバーの使用例

以下の例では、テンプレートを変更せずにデフォルトの MySQL テンプレートをインストールします。

前提条件

- CSI ドライバーがデプロイされている。
- 動的プロビジョニング用にストレージクラスが作成されている。

手順

- MySQL テンプレートを作成します。

```
# oc new-app mysql-persistent
```

出力例

```
--> Deploying template "openshift/mysql-persistent" to project default
...
```

```
# oc get pvc
```

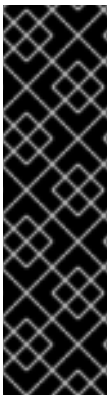
出力例

```
NAME             STATUS    VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
mysql            Bound     kubernetes-dynamic-pv-3271ffcb4e1811e8   1Gi
RWO            cinder        3s
```

4.2. CSI インラインの一時ボリューム

Container Storage Interface (CSI) のインライン一時ボリュームを使用すると、Pod のデプロイ時にインラインの一時ボリュームを作成し、Pod の破棄時にそれらを削除する **Pod** 仕様を定義できます。

この機能は、サポートされている Container Storage Interface (CSI) ドライバーでのみ利用できます。



重要

CSI インラインの一時ボリュームは、テクノロジープレビュー機能としてのみご利用可能です。テクノロジープレビュー機能は、Red Hat 製品のサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。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。テクノロジープレビューの機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

4.2.1. CSI インラインの一時ボリュームの概要

従来は、Container Storage Interface (CSI) ドライバーでサポートされるボリュームは **PersistentVolume** および **PersistentVolumeClaim** オブジェクトの組み合わせでのみ使用できます。

この機能により、**PersistentVolume** オブジェクトではなく、**Pod** 仕様に CSI ボリュームを直接指定できます。インラインボリュームは一時的なボリュームであり、Pod の再起動後は永続化されません。

4.2.1.1. サポートの制限

デフォルトで、OpenShift Container Platform は以下の制限下で CSI インラインの一時ボリュームのクローン作成をサポートします。

- サポートは CSI ドライバーでのみ利用可能です。in-tree (インツリー) および FlexVolumes はサポートされません。
- OpenShift Container Platform には CSI ドライバーが含まれません。コミュニティーまたはストレージベンダー が提供する CSI ドライバーを使用します。CSI ドライバーの提供されるインストール手順に従います。
- CSI ドライバーは、**Ephemeral** 機能を含む、インラインボリューム機能を実装していない可能性があります。詳細は、CSI ドライバーのドキュメントを参照してください。

4.2.2. Pod 仕様への CSI インライン一時ボリュームの埋め込み

CSI インラインの一時ボリュームを OpenShift Container Platform の **Pod** 仕様に埋め込むことができます。ランタイム時に、ネストされたインラインボリュームは、関連付けられた Pod の一時的なライフサイクルに従うため、CSI ドライバーは Pod の作成および破棄時にボリューム操作のすべてのフェーズをすべて処理できます。

手順

1. **Pod** オブジェクト定義を作成し、これをファイルに保存します。
2. CSI インラインの一時ボリュームをファイルに埋め込みます。

   my-csi-app.yaml

```
kind: Pod
apiVersion: v1
metadata:
  name: my-csi-app
spec:
  containers:
    - name: my-frontend
      image: busybox
      volumeMounts:
      - mountPath: "/data"
        name: my-csi-inline-vol
      command: [ "sleep", "1000000" ]
  volumes: (1)
    - name: my-csi-inline-vol
      csi:
        driver: inline.storage.kubernetes.io
        volumeAttributes:
          foo: bar
```

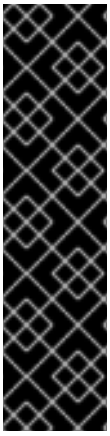
   (1) Pod で使用されるボリュームの名前。

3. 直前のステップで保存したオブジェクト定義ファイルを作成します。

```
$ oc create -f my-csi-app.yaml
```

4.3. CSI ボリュームスナップショット

本書では、サポートされる Container Storage Interface (CSI) ドライバーでボリュームスナップショットを使用して、OpenShift Container Platform でデータ損失から保護する方法について説明します。永続ボリューム についてある程度理解していることが推奨されます。



重要

CSI ボリュームスナップショットはテクノロジープレビュー機能です。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。テクノロジープレビューの機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲についての詳細は、https://access.redhat.com/ja/support/offerings/techpreview/ を参照してください。

4.3.1. CSI ボリュームスナップショットの概要

スナップショット は、特定の時点におけるクラスター内のストレージボリュームの状態を表します。ボリュームスナップショットは新規ボリュームのプロビジョニングに使用できます。

OpenShift Container Platform はデフォルトで CSI ボリューム酢ナップショットをサポートします。ただし、特定の CSI ドライバーが必要です。

CSI ボリュームのスナップショットを使用して、クラスター管理者は以下を行うことができます。

- スナップショットをサポートするサードパーティーの CSI ドライバーをデプロイします。
- 既存のボリュームスナップショットから永続ボリューム要求 (PVC) を新たに作成します。
- 既存の PVC のスナップショットを作成します。
- スナップショットを別の PVC として復元します。
- 既存のボリュームスナップショットを削除します。

CSI ボリュームスナップショットを使用すると、アプリケーション開発者は以下を行うことができます。

- ボリュームスナップショットは、アプリケーションレベルまたはクラスターレベルのストレージバックアップソリューションを開発するためのビルディングブロックとして使用します。
- 迅速に直前の開発バージョンにロールバックします。
- 毎回フルコピーを作成する必要がないため、ストレージをより効率的に使用できます。

ボリュームスナップショットを使用する場合は、以下の点に注意してください。

- サポートは CSI ドライバーでのみ利用可能です。in-tree (インツリー) および FlexVolumes はサポートされません。
- OpenShift Container Platform には一部の CSI ドライバーのみが同梱されます。OpenShift Container Platform ドライバー Operator によって提供されない CSI ドライバーについては、コ

ミュニティーまたはストレージベンダー が提供する CSI ドライバーを使用することが推奨されます。CSI ドライバーの提供されるインストール手順に従います。

- CSI ドライバーは、ボリュームのスナップショット機能を実装している場合もあれば、実装していない場合もあります。ボリュームスナップショットのサポートを提供している CSI ドライバーは、**csi-external-snapshotter** サイドカーコンテナーを使用する可能性があります。詳細は、CSI ドライバーで提供されるドキュメントを参照してください。

- OpenShift Container Platform 4.5 は、CSI 仕様 のバージョン 1.1.0 をサポートします。

4.3.2. CSI スナップショットコントローラーおよびサイドカー

OpenShift Container Platform は、コントロールプレーンにデプロイされるスナップショットコントローラーを提供します。さらに、CSI ドライバーベンダーは、CSI ドライバーのインストール時にインストールされるヘルパーコンテナーとして CSI スナップショットサイドカーコンテナーを提供します。

CSI スナップショットコントローラーおよびサイドカーは、OpenShift Container Platform API を使用してボリュームのスナップショットを提供します。これらの外部コンポーネントはクラスターで実行されます。

外部コントローラーは CSI スナップショットコントローラー Operator によってデプロイされます。

4.3.2.1. 外部コントローラー

CSI スナップショットコントローラーは **VolumeSnapshot** および **VolumeSnapshotContent** オブジェクトをバインドします。コントローラーは、**VolumeSnapshotContent** オブジェクトを作成し、削除して動的プロビジョニングを管理します。

4.3.2.2. 外部サイドカー

CSI ドライバーベンダーは、**csi-external-snapshotter** サイドカーを提供します。これは、CSI ドライバーでデプロイされる別のヘルパーコンテナーです。サイドカーは、**CreateSnapshot** および **DeleteSnapshot** 操作をトリガーしてスナップショットを管理します。ベンダーが提供するインストールの手順に従います。

4.3.3. CSI スナップショットコントローラー Operator について

CSI スナップショットコントローラー Operator は **openshift-cluster-storage-operator** namespace で実行されます。これは、デフォルトですべてのクラスターの Cluster Version Operator (CVO) によってインストールされます。

CSI スナップショットコントローラー Operator は、**openshift-cluster-storage-operator** namespace で実行される CSI スナップショットコントローラーをインストールします。

4.3.3.1. ボリュームスナップショット CRD

OpenShift Container Platform のインストール時に、CSI スナップショットコントローラー Operator は、**snapshot.storage.k8s.io/v1beta1** API グループに以下のスナップショットのカスタムリソース定義 (CRD) を作成します。

VolumeSnapshotContent

クラスター管理者がプロビジョニングしたクラスター内のボリュームのスナップショット。**PersistentVolume** オブジェクトと同様に、**VolumeSnapshotContent** CRD はストレージバックエンドの実際のスナップショットを参照するクラスターリソースです。

手動でプロビジョニングされたスナップショットの場合、クラスター管理者は多くの **VolumeSnapshotContent** CRD を作成します。これらには、ストレージシステム内の実際のボリュームスナップショットの詳細が含まれます。

VolumeSnapshotContent CRD には namespace が使用されず、これはクラスター管理者によって使用されるものです。

VolumeSnapshot

PersistentVolumeClaim オブジェクトと同様に、**VolumeSnapshot** CRD はスナップショットの開発者要求を定義します。CSI スナップショットコントローラー Operator は、適切な **VolumeSnapshotContent** CRD で **VolumeSnapshot** CRD のバインディングを処理する CSI スナップショットコントローラーを実行します。バインディングは 1 対 1 のマッピングです。
VolumeSnapshot CRD には namespace が使用されます。開発者は、CRD をスナップショットの個別の要求として使用します。

VolumeSnapshotClass

クラスター管理者は、**VolumeSnapshot** CRD に属する異なる属性を指定できます。これらの属性は、ストレージシステムの同じボリュームで作成されるスナップショット間で異なる場合があります。この場合、それらは永続ボリューム要求 (PVC) の同じストレージクラスを使用して表現できません。
VolumeSnapshotClass CRD は、スナップショットの作成時に使用する **csi-external-snapshotter** サイドカーのパラメーターを定義します。これにより、ストレージバックエンドは、複数のオプションがサポートされる場合に動的に作成するスナップショットの種類を認識できます。

動的にプロビジョニングされるスナップショットは **VolumeSnapshotClass** CRD を使用して、スナップショットの作成時に使用するストレージプロバイダー固有のパラメーターを指定します。

VolumeSnapshotContentClass CRD には namespace が使用されず、クラスター管理者がストレージバックエンドのグローバル設定オプションを有効にするために使用します。

4.3.4. ボリュームスナップショットのプロビジョニング

スナップショットをプロビジョニングする方法は、動的な方法と手動による方法の 2 種類があります。

4.3.4.1. 動的プロビジョニング

既存のスナップショットを使用する代わりに、スナップショットを永続ボリューム要求 (PVC) から動的に取得するように要求できます。パラメーターは **VolumeSnapshotClass** CRD を使用して指定されます。

4.3.4.2. 手動プロビジョニング

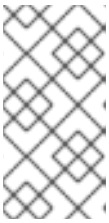
クラスター管理者は、多数の **VolumeSnapshotContent** オブジェクトを手動で事前にプロビジョニングできます。これらは、クラスターユーザーが利用できる実際のボリュームのスナップショットの詳細を保持します。

4.3.5. ボリュームスナップショットの作成

VolumeSnapshot オブジェクトを作成すると、OpenShift Container Platform はボリュームスナップショットを作成します。

前提条件

- 実行中の OpenShift Container Platform クラスターにログインしている。
- **VolumeSnapshot** オブジェクトをサポートする CSI ドライバーを使用して作成される PVC。
- ストレージバックエンドをプロビジョニングするストレージクラス。
- スナップショットの作成に使用する必要のある永続ボリューム要求 (PVC) を使用している Pod はありません。



注記

Pod が PVC を使用している場合は、PVC のボリュームスナップショットを作成しません。これを実行すると、PVC が一時停止 (停止) されないため、データが破損する可能性があります。まず実行中の Pod の終了処理を実行し、スナップショットの一貫性を維持します。

手順

ボリュームのスナップショットを動的に作成するには、以下を実行します。

1. 以下の YAML によって記述される **VolumeSnapshotClass** オブジェクトを使ってファイルを作成します。

   volumesnapshotclass.yaml

   ```
   apiVersion: snapshot.storage.k8s.io
   kind: VolumeSnapshotClass (1)
   metadata:
     name: csi-hostpath-snap
   driver: hostpath.csi.k8s.io
   deletionPolicy: Delete
   ```

   (1) ボリュームスナップショットに属する異なる属性を指定できます。

2. 以下のコマンドを実行して、直前の手順で保存されたオブジェクトを作成します。

   ```
   $ oc create -f volumesnapshotclass.yaml
   ```

3. **VolumeSnapshot** オブジェクトを作成します。

   volumesnapshot-dynamic.yaml

   ```
   apiVersion: snapshot.storage.k8s.io
   kind: VolumeSnapshot
   metadata:
     name: mysnap
   spec:
     volumeSnapshotClassName: csi-hostpath-snap (1)
     source:
       persistentVolumeClaimName: myclaim (2)
   ```

   (1) ボリュームスナップショットによる特定クラスの要求。**volumeSnapshotClassName** が空の場合、スナップショットは作成されません。

2 永続ボリュームにバインドされる **PersistentVolumeClaim** オブジェクトの名前。これは、スナップショットの作成に使用する内容を定義します。スナップショットの動的プロ

4. 以下のコマンドを実行して、直前の手順で保存されたオブジェクトを作成します。

```
$ oc create -f volumesnapshot-dynamic.yaml
```

スナップショットを手動でプロビジョニングするには、以下を実行します。

1. 上記のようにボリュームスナップショットクラスを定義するだけでなく、**volumeSnapshotContentName** パラメーターの値をスナップショットのソースとして指定します。

volumesnapshot-manual.yaml

```
apiVersion: snapshot.storage.k8s.io
kind: VolumeSnapshot
metadata:
  name: snapshot-demo
spec:
  source:
    volumeSnapshotContentName: mycontent 1
```

1 事前にプロビジョニングされたスナップショットには、**volumeSnapshotContentName** パラメーターが必要です。

2. 以下のコマンドを実行して、直前の手順で保存されたオブジェクトを作成します。

```
$ oc create -f volumesnapshot-manual.yaml
```

検証

スナップショットがクラスターで作成されると、スナップショットに関する追加情報が利用可能になります。

1. 作成したボリュームスナップショットの詳細を表示するには、以下のコマンドを実行します。

```
$ oc describe volumesnapshot mysnap
```

以下の例は、**mysnap** ボリュームスナップショットについての詳細を表示します。

volumesnapshot.yaml

```
apiVersion: snapshot.storage.k8s.io
kind: VolumeSnapshot
metadata:
  name: mysnap
spec:
  source:
    persistentVolumeClaimName: myclaim
  volumeSnapshotClassName: csi-hostpath-snap
status:
  boundVolumeSnapshotContentName: snapcontent-1af4989e-a365-4286-96f8-
```

```
d5dcd65d78d6 (1)
  creationTime: "2020-01-29T12:24:30Z" (2)
  readyToUse: true (3)
  restoreSize: 500Mi
```

(1) コントローラーによって作成された実際のストレージコンテンツへのポインター。

(2) スナップショットが作成された時間。スナップショットには、このタイミングで利用できるボリュームコンテンツが含まれます。

(3) 値が **true** に設定されている場合、スナップショットを使用して新規 PVC として復元できます。
値が **false** に設定されている場合、スナップショットが作成されています。ただし、ストレージバックエンドは、スナップショットを新規ボリュームとして復元できるようにするために、追加のタスクを実行してスナップショットを使用できる状態にする必要があります。たとえば、Amazon Elastic Block Store データを別の低コストの場所に移動する場合があり、これには数分の時間がかかる可能性があります。

2. ボリュームのスナップショットが作成されたことを確認するには、以下のコマンドを実行します。

```
$ oc get volumesnapshotcontent
```

実際のコンテンツへのポインターが表示されます。**boundVolumeSnapshotContentName** フィールドにデータが設定される場合、**VolumeSnapshotContent** オブジェクトが存在し、スナップショットが作成されています。

3. スナップショットの準備が完了していることを確認するには、**VolumeSnapshot** オブジェクトに **readyToUse: true** があることを確認します。

4.3.6. ボリュームスナップショットの削除

OpenShift Container Platform によるボリュームスナップショットの削除方法を設定できます。

手順

1. 以下の例のように、**VolumeSnapshotClass** オブジェクトで必要な削除ポリシーを指定します。

volumesnapshot.yaml

```
apiVersion: snapshot.storage.k8s.io
kind: VolumeSnapshotClass
metadata:
  name: csi-hostpath-snap
driver: hostpath.csi.k8s.io
deletionPolicy: Delete (1)
```

(1) ボリュームスナップショットの削除時に **Delete** 値を設定すると、**VolumeSnapshotContent** オブジェクトと共に基礎となるスナップショットが削除されます。**Retain** 値を設定すると、基礎となるスナップショットと **VolumeSnapshotContent** オブジェクトの両方が残ります。
Retain 値を設定し、対応する **VolumeSnapshotContent** オブジェクトを削除せずに

VolumeSnapshot オブジェクトを削除すると、コンテンツは残ります。スナップショット自体はストレージバックエンドにも保持されます。

2. 以下のコマンドを入力してボリュームスナップショットを削除します。

```
$ oc delete volumesnapshot <volumesnapshot_name>
```

出力例

```
volumesnapshot.snapshot.storage.k8s.io "mysnapshot" deleted
```

3. 削除ポリシーが **Retain** に設定されている場合は、以下のコマンドを入力してボリュームスナップショットのコンテンツを削除します。

```
$ oc delete volumesnapshotcontent <volumesnapshotcontent_name>
```

4. オプション: **VolumeSnapshot** オブジェクトが正常に削除されていない場合は、以下のコマンドを実行して残されているリソースのファイナライザーを削除し、削除操作を続行できるようにします。

重要

永続ボリューム要求 (PVC) またはボリュームスナップショットのコンテンツのいずれかから **VolumeSnapshot** オブジェクトへの既存の参照がない場合にのみファイナライザーを削除します。**--force** オプションを使用する場合でも、すべてのファイナライザーが削除されるまで削除操作でスナップショットオブジェクトは削除されません。

```
$ oc patch -n $PROJECT volumesnapshot/$NAME --type=merge -p '{"metadata":
{"finalizers":null}}'
```

出力例

```
volumesnapshotclass.snapshot.storage.k8s.io "csi-ocs-rbd-snapclass" deleted
```

ファイナライザーが削除され、ボリュームスナップショットが削除されます。

4.3.7. ボリュームスナップショットの復元

VolumeSnapshot CRD コンテンツは、既存のボリュームを以前の状態に復元するために使用されます。

VolumeSnapshot CRD がバインドされ、**readyToUse** 値が **true** に設定された後に、そのリソースを使用して、スナップショットからのデータが事前に設定されている新規ボリュームをプロビジョニングできます。

前提条件

- 実行中の OpenShift Container Platform クラスターにログインしている。

- ボリュームスナップショットをサポートする Container Storage Interface (CSI) ドライバーを使用して作成される永続ボリューム要求 (PVC)。
- ストレージバックエンドをプロビジョニングするストレージクラス。

手順

1. 以下のように PVC に **VolumeSnapshot** データソースを指定します。

   pvc-restore.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: myclaim-restore
spec:
  storageClassName: csi-hostpath-sc
  dataSource:
    name: mysnap 1
    kind: VolumeSnapshot 2
    apiGroup: snapshot.storage.k8s.io 3
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

   1. ソースとして使用するスナップショットを表す **VolumeSnapshot** オブジェクトの名前。
   2. **VolumeSnapshot** の値に設定する必要があります。
   3. **snapshot.storage.k8s.io** の値に設定する必要があります。

2. 以下のコマンドを実行して PVC を作成します。

```
$ oc create -f pvc-restore.yaml
```

3. 以下のコマンドを実行して、復元された PVC が作成されていることを確認します。

```
$ oc get pvc
```

   2 つの異なる PVC が表示されます。

4.4. CSI ボリュームのクローン作成

ボリュームのクローン作成により、既存の永続ボリュームが複製されます。これは OpenShift Container Platform におけるデータ損失からの保護に役立ちます。この機能は、サポートされている Container Storage Interface (CSI) ドライバーでのみ利用できます。CSI ボリュームのクローンをプロビジョニングする前に、永続ボリューム について理解しておく必要があります。

4.4.1. CSI ボリュームのクローン作成の概要

Container Storage Interface (CSI) ボリュームのクローンは、特定の時点における既存の永続ボリュームの複製です。

ボリュームのクローン作成はボリュームのスナップショットに似ていますが、より効率的な方法です。たとえば、クラスター管理者は、既存のクラスターボリュームの別のインスタンスを作成してクラスターボリュームを複製できます。

クローン作成により、バックエンドのデバイスでは、新規の空のボリュームが作成されるのではなく、指定したボリュームの複製が作成されます。動的プロビジョニングの後には、標準のボリュームを使用するのと同じように、ボリュームのクローンを使用できます。

クローン作成に必要な新しい API オブジェクトはありません。**PersistentVolumeClaim** オブジェクトの既存の **dataSource** フィールドは、同じ namespace の既存の PersistentVolumeClaim の名前を許可できるように拡張されます。

4.4.1.1. サポートの制限

デフォルトで、OpenShift Container Platform は以下の制限の下で CSI ボリュームのクローン作成をサポートします。

- 宛先永続ボリューム要求 (PVC) はソース PVC と同じ namespace に存在する必要があります。
- ソースストレージおよび宛先ストレージクラスは同じである必要があります。
- サポートは CSI ドライバーでのみ利用可能です。in-tree (インツリー) および FlexVolumes はサポートされません。
- OpenShift Container Platform には CSI ドライバーが含まれません。コミュニティーまたはストレージベンダー が提供する CSI ドライバーを使用します。CSI ドライバーの提供されるインストール手順に従います。
- CSI ドライバーは、ボリュームのクローン作成機能を実装していない可能性もあります。詳細は、CSI ドライバーのドキュメントを参照してください。
- OpenShift Container Platform 4.5 は、CSI 仕様 のバージョン 1.1.0 をサポートします。

4.4.2. CSI ボリュームクローンのプロビジョニング

CSI ボリュームクローンのプロビジョニングは、クローン作成された永続ボリューム要求 (PVC) API オブジェクトの作成によってトリガーされます。クローンは、他の永続ボリュームと同じルールに従って、別の PVC の内容を事前に設定します。例外として、同じ namespace の既存 PVC を参照する **dataSource** を追加する必要があります。

前提条件

- 実行中の OpenShift Container Platform クラスターにログインしている。
- PVC がボリュームのクローン作成をサポートする CSI ドライバーを使用して作成されている。
- ストレージバックエンドが動的プロビジョニング用に設定されている。静的プロビジョナーのクローン作成のサポートは利用できません。

手順

既存の PVC から PVC のクローンを作成するには、以下を実行します。

1. 以下の YAML によって記述される **PersistentVolumeClaim** オブジェクトを使ってファイルを作成し、保存します。

   pvc-clone.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-1-clone
  namespace: mynamespace
spec:
  storageClassName: csi-cloning 1
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
  dataSource:
    kind: PersistentVolumeClaim
    name: pvc-1
```

   1 ストレージのバックエンドをプロビジョニングするストレージクラスの名前。デフォルトのストレージクラスを使用でき、**storageClassName** は仕様で省略できます。

2. 以下のコマンドを実行して、直前の手順で保存されたオブジェクトを作成します。

```
$ oc create -f pvc-clone.yaml
```

   新規の PVC **pvc-1-clone** が作成されます。

3. 以下のコマンドを実行して、ボリュームのクローンが作成され、準備状態にあることを確認します。

```
$ oc get pvc pvc-1-clone
```

   pvc-1-clone は、これが **Bound** であることを示します。

   これで、新たにクローン作成された PVC を使用して Pod を設定する準備が整いました。

4. YAML によって記述される **Pod** オブジェクトと共にファイルを作成し、保存します。以下に例を示します。

```
kind: Pod
apiVersion: v1
metadata:
  name: mypod
spec:
  containers:
    - name: myfrontend
      image: dockerfile/nginx
      volumeMounts:
      - mountPath: "/var/www/html"
        name: mypd
  volumes:
```

```
- name: mypd
  persistentVolumeClaim:
    claimName: pvc-1-clone 1
```

1 CSI ボリュームのクローン作成の操作時に作成されるクローン作成された PVC。

作成された **Pod** オブジェクトは、元の **dataSource** PVC とは別に、クローンされた PVC の使用、クローン、スナップショット、または削除を実行できるようになりました。

4.5. AWS ELASTIC BLOCK STORE CSI ドライバー OPERATOR

4.5.1. 概要

OpenShift Container Platform は、AWS Elastic Block Store (EBS) の Container Storage Interface (CSI) ドライバーを使用して永続ボリューム (PV) をプロビジョニングできます。

PV、永続ボリューム要求 (PVC)、動的プロビジョニング および RBAC 認証 について理解しておくことを推奨します。

PVC を作成する前に、AWS EBS CSI ドライバー Operator をインストールする必要があります。Operator は、PVC の作成に使用できるデフォルトの StorageClass を提供します。AWS Elastic Block Store を使用した永続ストレージ で説明されているように、EBS StorageClass を作成するオプションもあります。

Operator のインストール後に、OpenShift Container Platform クラスターに必要な AWS EBS CSI カスタムリソース (CR) も作成する必要があります。

重要

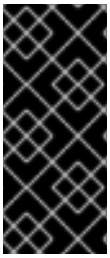
AWS EBS CSI ドライバー Operator はテクノロジープレビュー機能です。テクノロジープレビュー機能は、Red Hat 製品のサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。Red Hat は実稼働環境でこれらを使用することを推奨していません。テクノロジープレビューの機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲についての詳細は、https://access.redhat.com/ja/support/offerings/techpreview/ を参照してください。

4.5.2. CSI について

ストレージベンダーはこれまで Kubernetes の一部としてストレージドライバーを提供してきました。Container Storage Interface (CSI) の実装では、サードパーティーのプロバイダーは、コア Kubernetes コードを変更せずに標準のインターフェイスを使用してストレージプラグインを提供できます。

CSI Operator は、in-tree (インツリー) ボリュームプラグインでは不可能なボリュームスナップショットなどのストレージオプションを OpenShift Container Platform ユーザーに付与します。



重要

OpenShift Container Platform は、AWS EBS ストレージをプロビジョニングするためにデフォルトで in-tree (インツリー) または CSI 以外のドライバーの使用に設定されます。この in-tree (インツリー) ドライバーは、OpenShift Container Platform の後続の更新で削除されます。その時点で、既存の in-tree (インツリー) ドライバーを使用してプロビジョニングされるボリュームが CSI ドライバーへの移行用に予定されます。

OpenShift Container Platform での AWS EBS 永続ボリュームの動的プロビジョニングに関する詳細は、AWS Elastic Block Store を使用した永続ストレージ を参照してください。

4.5.3. AWS Elastic Block Store CSI ドライバー Operator のインストール

AWS Elastic Block Store (EBS) Container Storage Interface (CSI) ドライバー Operator により、既存の AWS EBS in-tree (インツリー) ストレージプラグインを置き換えることができます。

重要

AWS EBS CSI ドライバー Operator はテクノロジープレビュー機能です。テクノロジープレビュー機能は、Red Hat 製品のサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。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。

AWS EBS CSI ドライバー Operator をインストールすると、OpenShift Container Platform の **PersistentVolumeClaims**、**PersistentVolumes**、および **StorageClasses** API オブジェクトで CSI ボリュームを使用できるように CSI ドライバーが提供されます。また、永続ボリューム要求 (PVC) の作成に使用できる StorageClass もデプロイします。

AWS EBS CSI ドライバー Operator はデフォルトでは OpenShift Container Platform にインストールされません。以下の手順を使用してこの Operator をインストールし、クラスター内で AWS EBS CSI ドライバーを有効にできるように設定します。

前提条件

- OpenShift Container Platform Web コンソールにアクセスします。

手順

Web コンソールから AWS EBS CSI ドライバー Operator をインストールするには、以下を実行します。

1. Web コンソールにログインします。
2. **Operators → OperatorHub** に移動します。
3. AWS EBS CSI ドライバー Operator を見つけるには、**AWS EBS CSI** をフィルターボックスに入力します。
4. **Install** をクリックします。
5. **Install Operator** ページで、**All namespaces on the cluster(default)** が選択されていることを確認します。**Installed Namespace** ドロップダウンメニューから **openshift-aws-ebs-csi-driver-operator** を選択します。
6. **Update Channel** および **Approval Strategy** の値を必要な値に調整します。

7. **Install** をクリックします。

これが終了すると、AWS EBS CSI ドライバー Operator が Web コンソールの **Installed Operators** セクションに一覧表示されます。

4.5.4. AWS Elastic Block Store CSI ドライバーのインストール

AWS Elastic Block Store (EBS) Container Storage Interface (CSI) ドライバーは、AWS EBS 永続ボリュームを作成し、マウントできるカスタムリソース (CR) です。

このドライバーは、デフォルトでは OpenShift Container Platform にインストールされず、AWS EBS CSI ドライバー Operator のインストール後にインストールする必要があります。

前提条件

- AWS EBS CSI ドライバー Operator がインストールされている。
- OpenShift Container Platform Web コンソールへのアクセスが必要です。

手順

Web コンソールから AWS EBS CSI ドライバーをインストールするには、以下の手順を実行します。

1. OpenShift Container Platform Web コンソールにログインします。
2. **Operators → Installed Operators** に移動します。
3. 一覧から **AWS EBS CSI Driver Operator** を見つけ、Operator リンクをクリックします。
4. ドライバーを作成します。
   a. **Details** タブで、**Create Instance** をクリックします。
   b. オプション: **YAML view** を選択し、表記の追加などの変更をドライバーオブジェクトテンプレートに対して実行します。
   c. **Create** をクリックして確定します。

重要

クラスターの名前を変更し、特定の namespace を指定することはサポートされる機能ではありません。

4.5.5. AWS Elastic Block Store CSI ドライバー Operator のアンインストール

AWS EBS CSI ドライバー Operator をアンインストールする前に、Operator によって使用されるすべての永続ボリューム要求 (PVC) を削除する必要があります。

前提条件

- OpenShift Container Platform Web コンソールにアクセスします。

手順

Web コンソールから AWS EBS CSI ドライバー Operator をアンインストールするには、以下を実行します。

1. Web コンソールにログインします。
2. **Storage → Persistent Volume Claims**に移動します。
3. AWS EBS CSI ドライバー Operator によって使用される PVC を選択し、**Delete** をクリックします。
4. **Operators → Installed Operators** ページからスクロールするか、または **AWS EBS CSI** を **Filter by name** フィールドに入力して Operator を見つけます。次に、それをクリックします。
5. **Installed Operators** 詳細ページの右側で、**Actions** ドロップダウンメニューから **Uninstall Operator** を選択します。
6. **Uninstall Operator** ウィンドウでプロンプトが表示されたら、**Uninstall** ボタンをクリックして namespace から Operator を削除します。Operator によってクラスターにデプロイされたアプリケーションは手動でクリーンアップする必要があります。

これが終了すると、AWS EBS CSI ドライバー Operator が Web コンソールの **Installed Operators** セクションに一覧表示されなくなります。

追加リソース

- AWS Elastic Block Store を使用した永続ストレージ
- CSI ボリュームの設定

4.6. OPENSTACK MANILA CSI ドライバー OPERATOR

4.6.1. 概要

OpenShift Container Platform は、OpenStack Manila 共有ファイルシステムサービスの Container Storage Interface (CSI) ドライバーを使用して永続ボリューム (PV) をプロビジョニングできます。

PV、永続ボリューム要求 (PVC)、動的プロビジョニング および RBAC 認証 について理解しておくことを推奨します。

PVC を作成する前に、Manila CSI ドライバー Operator をインストールする必要があります。Operator は、動的プロビジョニングに必要なすべての利用可能な Manila 共有タイプに必要なストレージクラスを作成します。

Operator のインストール後に、OpenShift Container Platform クラスターに必要な ManilaDriver カスタムリソース (CR) も作成する必要があります。

4.6.2. Manila CSI ドライバー Operator のインストール

Manila Container Storage Interface (CSI) ドライバー Operator は、デフォルトでは OpenShift Container Platform にインストールされません。以下の手順を使用してこの Operator をインストールし、クラスター内で OpenStack Manila CSI ドライバーを有効にできるように設定します。

前提条件

- OpenShift Container Platform Web コンソールへのアクセスが必要です。
- 基礎となる Red Hat OpenStack Platform (RHOSP) インフラストラクチャークラウドは、Manila 提供の NFS 共有をデプロイします。

手順

Web コンソールから Manila CSI ドライバー Operator をインストールするには、以下の手順を実行します。

1. OpenShift Container Platform Web コンソールにログインします。
2. **Operators → OperatorHub** に移動します。
3. **Manila CSI Driver Operator** をフィルターボックスに入力して Operator を見つけます。
4. **Install** をクリックします。
5. **Install Operator** ページで、**Installed Namespace** ドロップダウンメニューから **openshift-manila-csi-driver-operator** を選択します。
6. **Update Channel** および **Approval Strategy** の値を必要な値に調整します。サポートされている唯一の **インストールモード** は **All namespaces on the cluster** です。
7. **Install** をクリックします。

これが終了すると、Manila CSI ドライバー Operator が Web コンソールの **Installed Operators** セクションに一覧表示されます。

4.6.3. OpenStack Manila CSI ドライバーのインストール

OpenStack Manila Container Storage Interface (CSI) ドライバーは、OpenStack Manila 共有を作成し、マウントできるカスタムリソース (CR) です。これは、スナップショットの作成およびスナップショットからの共有のリカバリーもサポートします。

このドライバーは、デフォルトでは OpenShift Container Platform にインストールされず、Manila CSI ドライバー Operator のインストール後にインストールする必要があります。

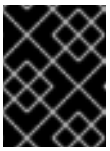
前提条件

- Manila CSI ドライバー Operator がインストールされている。
- OpenShift Container Platform Web コンソールまたはコマンドラインインターフェイス (CLI) へのアクセス。

UI の手順

Web コンソールから Manila CSI ドライバーをインストールするには、以下の手順を実行します。

1. OpenShift Container Platform Web コンソールにログインします。
2. **Operators → Installed Operators** に移動します。
3. 一覧から **Manila CSI Driver Operator** を見つけ、Operator リンクをクリックします。
4. ドライバーを作成します。
    a. **Details** タブで、**Create Instance** をクリックします。
    b. オプション: **YAML view** を選択し、表記の追加などの変更を ManilaDriver オブジェクトテンプレートに対して実行します。
    c. **Create** をクリックして確定します。



重要

クラスターの名前を変更し、特定の namespace を指定することはサポートされる機能ではありません。

CLI の手順

CLI から Manila CSI ドライバーをインストールするには、以下の手順を実行します。

1. **maniladriver.yaml** などのオブジェクト YAML ファイルを作成して、ManilaDriver を定義します。

   maniladriver の例

```
apiVersion: csi.openshift.io/v1alpha1
kind: ManilaDriver
metadata:
  name: cluster (1)
```

   (1) クラスターの名前を変更し、特定の namespace を指定することはサポートされる機能ではありません。

2. 直前の手順で作成したファイルを指定して、OpenShift Container Platform クラスターに ManilaDriver CR オブジェクトを作成します。

```
$ oc create -f maniladriver.yaml
```

Operator のインストールが終了すると、Red Hat OpenStack Platform (RHOSP) での RWX 永続ボリュームの動的プロビジョニングのために Manila CSI ドライバーが OpenShift Container Platform にデプロイされます。

検証

1. 以下のコマンドを入力して、ManilaDriver CR が正常に作成されていることを確認します。

```
$ oc get all -n openshift-manila-csi-driver
```

   出力例

```
NAME                                                  READY   STATUS    RESTARTS   AGE
pod/csi-nodeplugin-nfsplugin-lzvpm                    1/1     Running   0          18h
pod/csi-nodeplugin-nfsplugin-slvg2                    1/1     Running   0          18h
pod/csi-nodeplugin-nfsplugin-xmps9                    1/1     Running   0          18h
pod/openstack-manila-csi-controllerplugin-7d4f5d985b-mw4x5   3/3     Running   0
17h
pod/openstack-manila-csi-nodeplugin-6xchs             2/2     Running   0          18h
pod/openstack-manila-csi-nodeplugin-bkcmz             2/2     Running   0          18h
pod/openstack-manila-csi-nodeplugin-rlpps             2/2     Running   0          18h

NAME                                                  DESIRED CURRENT READY UP-TO-DATE
AVAILABLE NODE SELECTOR AGE
daemonset.apps/csi-nodeplugin-nfsplugin               3       3       3     3
3
18h
```

```
daemonset.apps/openstack-manila-csi-nodeplugin                3      3      3    3         3
18h

NAME                                                          READY UP-TO-DATE AVAILABLE AGE
deployment.apps/openstack-manila-csi-controllerplugin         1/1   1         1         18h

NAME                                                          DESIRED CURRENT READY AGE
replicaset.apps/openstack-manila-csi-controllerplugin-7d4f5d985b  1       1       1     17h
```

2. 以下のコマンドを入力して、ストレージクラスが正常に作成されていることを確認します。

```
$ oc get storageclasses | grep -E "NAME|csi-manila-"
```

出力例

```
NAME            PROVISIONER             RECLAIMPOLICY   VOLUMEBINDINGMODE
ALLOWVOLUMEEXPANSION  AGE
csi-manila-gold   manila.csi.openstack.org  Delete          Immediate           false                  18h
```

4.6.4. Manila CSI ボリュームの動的プロビジョニング

OpenShift Container Platform は利用可能な Manila 共有タイプ別に StorageClass をインストールします。

作成される YAML ファイルは Manila およびその Container Storage Interface (CSI) プラグインから完全に切り離されます。アプリケーション開発者は、ReadWriteMany (RWX) ストレージを動的にプロビジョニングし、YAML マニフェストを使用してストレージを安全に使用するアプリケーションと共に Pod をデプロイできます。ReadWriteOnce (RWO) などの他のアクセスモードをプロビジョニングすることもできます。

PVC 定義のストレージクラス参照を除き、AWS、GCP、Azure、および他のプラットフォームで OpenShift Container Platform で使用する同じ Pod および永続ボリューム要求 (PVC) 定義をオンプレミスで使用できます。

前提条件

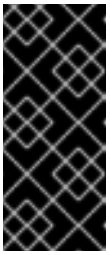
- Red Hat OpenStack Platform (RHOSP) は適切な Manila 共有インフラストラクチャーでデプロイされ、OpenShift Container Platform でボリュームを動的にプロビジョニングし、マウントするために使用できます。

手順 (UI)

Web コンソールを使用して Manila CSI ボリュームを動的に作成するには、以下を実行します。

1. OpenShift Container Platform コンソールで、**Storage → Persistent Volume Claims**をクリックします。
2. 永続ボリューム要求 (PVC) の概要で、**Create Persistent Volume Claim**をクリックします。
3. 結果のページで必要なオプションを定義します。
   a. 適切な StorageClass を選択します。
   b. ストレージ要求の一意の名前を入力します。

c. アクセスモードを選択し、作成する PVC の読み取りおよび書き込みアクセスを指定します。



重要

- この PVC を満たす永続ボリューム (PV) をクラスター内の複数ノードの複数 Pod にマウントする必要がある場合には、RWX を使用します。
- 追加の Pod が動的にプロビジョニングされないようにする必要がある場合には、RWO モードを使用します。

4. ストレージ要求のサイズを定義します。
5. **Create** をクリックして永続ボリューム要求 (PVC) を作成し、永続ボリュームを生成します。

手順 (CLI)

コマンドラインインターフェイス (CLI) を使用して Manila CSI ボリュームを動的に作成するには、以下を実行します。

1. 以下の YAML によって記述される **PersistentVolumeClaim** オブジェクトを使ってファイルを作成し、保存します。

pvc-manila.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-manila
spec:
  accessModes: 1
    - ReadWriteMany
  resources:
    requests:
   storage: 10Gi
  storageClassName: csi-manila-gold 2
```

1 この PVC を満たす永続ボリューム (PV) をクラスター内の複数ノードの複数 Pod にマウントする必要がある場合には、RWX を使用します。ReadWriteOnce (RWO) モードを使用して、追加の Pod が動的にプロビジョニングされないようにします。

2 ストレージのバックエンドをプロビジョニングするストレージクラスの名前。Manila StorageClasses は Operator によってプロビジョニングされ、これには **csi-manila-** 接頭辞があります。

2. 以下のコマンドを実行して、直前の手順で保存されたオブジェクトを作成します。

```
$ oc create -f pvc-manila.yaml
```

新規 PVC が作成されます。

3. ボリュームが作成され、準備状態にあることを確認するには、以下のコマンドを実行します。

```
$ oc get pvc pvc-manila
```

pvc-manila は、これが **Bound** であることを示します。

新規 PVC を使用して Pod を設定できるようになりました。

4.6.5. Manila CSI ドライバー Operator のアンインストール

Manila Container Storage Interface (CSI) ドライバー Operator をアンインストールする前に、Operator によって使用されるすべての永続ボリューム要求 (PVC) を削除する必要があります。

前提条件

- OpenShift Container Platform Web コンソールにアクセスします。

手順

Web コンソールから Manila CSI ドライバー Operator をアンインストールするには、以下を実行します。

1. Web コンソールにログインします。
2. **Storage → Persistent Volume Claims** に移動します。
3. Manila CSI Driver Operator によって使用される PVC を選択し、**Delete** をクリックします。
4. **Operators → Installed Operators** ページからスクロールするか、または **Manila CSI** を **Filter by name** フィールドに入力して Operator を見つけます。次に、それをクリックします。
5. **Installed Operators** 詳細ページの右側で、**Actions** ドロップダウンメニューから **Uninstall Operator** を選択します。
6. **Uninstall Operator** ウィンドウでプロンプトが表示されたら、**Uninstall** ボタンをクリックして namespace から Operator を削除します。Operator によってクラスターにデプロイされたアプリケーションは手動でクリーンアップする必要があります。

これが終了すると、Manila CSI ドライバー Operator が Web コンソールの **Installed Operators** セクションに一覧表示されなくなります。

追加リソース

- CSI ボリュームの設定

第5章 永続ボリュームの拡張

5.1. ボリューム拡張サポートの有効化

永続ボリュームを拡張する前に、**StorageClass** オブジェクトでは **allowVolumeExpansion** フィールドを **true** に設定している必要があります。

手順

- **StorageClass** オブジェクトを編集し、**allowVolumeExpansion** 属性を追加します。以下の例では、ストレージクラスの設定の下部にこの行を追加する方法を示しています。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
...
parameters:
  type: gp2
reclaimPolicy: Delete
allowVolumeExpansion: true (1)
```

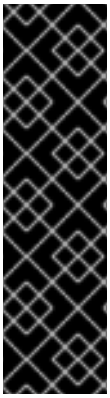
(1) この属性を **true** に設定すると、PVC を作成後に拡張することができます。

5.2. CSI ボリュームの拡張

Container Storage Interface (CSI) を使用して、作成後にストレージボリュームを拡張することができます。

OpenShift Container Platform はデフォルトで CSI ボリューム拡張をサポートします。ただし、特定の CSI ドライバーが必要です。

OpenShift Container Platform 4.5 は、CSI 仕様 のバージョン 1.1.0 をサポートします。



重要

CSI ボリュームの拡張は、テクノロジープレビュー機能としてのみ利用できます。テクノロジープレビュー機能は、Red Hat 製品のサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。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。テクノロジープレビューの機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

5.3. サポートされているドライバーでの FLEXVOLUME の拡張

FlexVolume を使用してバックエンドストレージシステムに接続する場合は、永続ストレージボリュームを作成後に拡張することができます。これは、OpenShift Container Platform で永続ボリューム要求 (PVC) を手動で更新して実行できます。

FlexVolume は、ドライバーが **RequiresFSResize** が **true** の状態で設定されている場合に拡張を許可します。FlexVolume は、Pod の再起動時に拡張できます。

他のボリュームタイプと同様に、FlexVolume ボリュームは Pod によって使用される場合にも拡張できます。

前提条件

- 基礎となるボリュームドライバーがサイズ変更をサポートする。
- ドライバーは **RequiresFSResize** 機能が **true** の状態で設定されている。
- 動的プロビジョニングが使用される。
- 制御する側の **StorageClass** オブジェクトには **allowVolumeExpansion** が **true** に設定されている。

手順

- FlexVolume プラグインのサイズ変更を使用するには、以下の方法で **ExpandableVolumePlugin** インターフェイスを実装する必要があります。

  RequiresFSResize

  true の場合、容量を直接更新します。**false** の場合、**ExpandFS** メソッドを呼び出し、ファイルシステムのサイズ変更を終了します。

  ExpandFS

  true の場合、**ExpandFS** を呼び出し、物理ボリュームの拡張の実行後にファイルシステムのサイズを変更します。ボリュームドライバーは、ファイルシステムのサイズ変更と共に物理ボリュームのサイズ変更も実行できます。

重要

OpenShift Container Platform はマスターノードへの FlexVolume プラグインのインストールをサポートしないため、FlexVolume のコントロールプレーンの拡張をサポートしません。

5.4. ファイルシステムを使用した永続ボリューム要求 (PVC) の拡張

ファイルシステムのサイズ変更を必要とするボリュームタイプ (GCE PD、EBS、および Cinder など) に基づいて PVC を拡張するには 2 つの手順からなるプロセスが必要です。このプロセスでは、クラウドプロバイダーでボリュームオブジェクトを拡張してから実際のノードでファイルシステムを拡張します。

ノードでのファイルシステムの拡張は、新規 Pod がボリュームと共に起動する場合にのみ実行されます。

前提条件

- 制御する側の **StorageClass** オブジェクトでは、**allowVolumeExpansion** が **true** に設定されている必要があります。

手順

1. **spec.resources.requests** を編集して PVC を編集し、新規サイズを要求します。たとえば、以下では **ebs** PVC を 8 Gi に拡張します。

```
kind: PersistentVolumeClaim
```

```
apiVersion: v1
metadata:
  name: ebs
spec:
  storageClass: "storageClassWithFlagSet"
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 8Gi (1)
```

1. **spec.resources.requests** をさらに大きな量を表す値に更新すると、PVC が拡張されます。

2. クラウドプロバイダーオブジェクトのサイズ変更が終了すると、PVC は **FileSystemResizePending** に設定されます。以下のコマンドを入力して状態を確認します。

```
$ oc describe pvc <pvc_name>
```

3. クラウドプロバイダーオブジェクトのサイズ変更が終了すると、**PersistentVolume** オブジェクトは **PersistentVolume.Spec.Capacity** に新規に要求されたサイズを反映します。この時点で、PVC から新規 Pod を作成または再作成してファイルシステムのサイズ変更を終了することができます。Pod が実行されている場合、新たに要求されたサイズが利用可能になり、**FileSystemResizePending** 状態が PVC から削除されます。

5.5. ボリューム拡張時の障害からの復旧

基礎となるストレージの拡張に失敗した場合に、OpenShift Container Platform の管理者は永続ボリューム要求 (PVC) の状態を手動で復旧し、サイズ変更要求を取り消します。そうでない場合には、サイズ変更要求が管理者の介入なしにコントローラーによって継続的に再試行されます。

手順

1. **Retain** 回収ポリシーで要求 (PVC) にバインドされている永続ボリューム (PV) にマークを付けます。これは、PV を編集し、**persistentVolumeReclaimPolicy** を **Retain** に変更して実行できます。

2. PVC を削除します。これは後ほど再作成されます。

3. 新規に作成された PVC が **Retain** というマークが付けられた PV にバインドされるには、PV を手動で編集し、PV 仕様から **claimRef** エントリーを削除します。これで、PV には **Available** というマークが付けられます。

4. より小さいサイズ、または基礎となるストレージプロバイダーによって割り当て可能なサイズで PVC を再作成します。

5. PVC の **volumeName** フィールドを PV の名前に設定します。これにより、PVC がプロビジョニングされた PV にのみバインドされます。

6. PV で回収ポリシーを復元します。

第6章 動的プロビジョニング

6.1. 動的プロビジョニングについて

StorageClass リソースオブジェクトは、要求可能なストレージを記述し、分類するほか、動的にプロビジョニングされるストレージのパラメーターを要求に応じて渡すための手段を提供します。**StorageClass** オブジェクトは、さまざまなレベルのストレージとストレージへのアクセスを制御するための管理メカニズムとしても機能します。クラスター管理者 (**cluster-admin**) またはストレージ管理者 (**storage-admin**) は、ユーザーが基礎となるストレージボリュームソースに関する詳しい知識がなくても要求できる **StorageClass** オブジェクトを定義し、作成します。

OpenShift Container Platform の永続ボリュームフレームワークはこの機能を有効にし、管理者がクラスターに永続ストレージをプロビジョニングできるようにします。フレームワークにより、ユーザーは基礎となるインフラストラクチャーの知識がなくてもこれらのリソースを要求できるようになります。

OpenShift Container Platform では、数多くのストレージタイプを永続ボリュームとして使用することができます。これらはすべて管理者によって静的にプロビジョニングされますが、一部のストレージタイプは組み込みプロバイダーとプラグイン API を使用して動的に作成できます。

6.2. 利用可能な動的プロビジョニングプラグイン

OpenShift Container Platform は、以下のプロビジョナープラグインを提供します。 これらには、クラスターの設定済みプロバイダーの API を使用して新規ストレージリソースを作成する動的プロビジョニング用の一般的な実装が含まれます。

ストレージタイプ	プロビジョナープラグインの名前	注記
Red Hat OpenStack Platform (RHOSP) Cinder	**kubernetes.io/cinder**	
RHOSP Manila Container Storage Interface (CSI)	**manila.csi.openstack.org**	インストールが完了すると、OpenStack Manila CSI Driver Operator および ManilaDriver は、動的プロビジョニングに必要なすべての利用可能な Manila 共有タイプに必要なストレージクラスを自動的に作成します。
AWS Elastic Block Store (EBS)	**kubernetes.io/aws-ebs**	複数クラスターを複数の異なるゾーンで使用する際の動的プロビジョニングの場合、各ノードに **Key=kubernetes.io/cluster/<cluster_name>,Value=<cluster_id>** のタグを付けます。ここで、**<cluster_name>** および **<cluster_id>** はクラスターごとに固有の値になります。
Azure Disk	**kubernetes.io/azure-disk**	

ストレージタイプ	プロビジョナープラグインの名前	注記
Azure File	**kubernetes.io/azure-file**	**persistent-volume-binder** サービスアカウントでは、Azure ストレージアカウントおよびキーを保存するためにシークレットを作成し、取得するためのパーミッションが必要です。
GCE Persistent Disk (gcePD)	**kubernetes.io/gce-pd**	マルチゾーン設定では、GCE プロジェクトごとに OpenShift Container Platform クラスターを実行し、現行クラスターのノードが存在しないゾーンで PV が作成されないようにすることが推奨されます。
VMware vSphere	**kubernetes.io/vsphere-volume**	

重要

選択したプロビジョナープラグインでは、関連するクラウド、ホスト、またはサードパーティープロバイダーを、関連するドキュメントに従って設定する必要もあります。

6.3. ストレージクラスの定義

現時点で、**StorageClass** オブジェクトはグローバルスコープオブジェクトであり、**cluster-admin** または **storage-admin** ユーザーによって作成される必要があります。

重要

Cluster Storage Operator は、使用されるプラットフォームに応じてデフォルトのストレージクラスをインストールする可能性があります。このストレージクラスは Operator によって所有され、制御されます。アノテーションとラベルを定義するほかは、これを削除したり、変更したりすることはできません。異なる動作が必要な場合は、カスタムストレージクラスを定義する必要があります。

以下のセクションでは、**StorageClass** オブジェクトの基本的な定義とサポートされている各プラグインタイプの具体的な例について説明します。

6.3.1. 基本 StorageClass オブジェクト定義

以下のリソースは、ストレージクラスを設定するために使用するパラメーターおよびデフォルト値を示しています。この例では、AWS ElasticBlockStore (EBS) オブジェクト定義を使用します。

StorageClass 定義の例

```
kind: StorageClass (1)
apiVersion: storage.k8s.io/v1 (2)
metadata:
```

```
  name: gp2 3
  annotations: 4
    storageclass.kubernetes.io/is-default-class: 'true'
    ...
provisioner: kubernetes.io/aws-ebs 5
parameters: 6
  type: gp2
...
```

1. (必須) API オブジェクトタイプ。
2. (必須) 現在の apiVersion。
3. (必須) ストレージクラスの名前。
4. (オプション) ストレージクラスのアノテーション。
5. (必須) このストレージクラスに関連付けられているプロビジョナーのタイプ。
6. (オプション) 特定のプロビジョナーに必要なパラメーター。これはプラグインによって異なります。

6.3.2. ストレージクラスのアノテーション

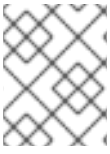
ストレージクラスをクラスター全体のデフォルトとして設定するには、以下のアノテーションをストレージクラスのメタデータに追加します。

```
storageclass.kubernetes.io/is-default-class: "true"
```

以下に例を示します。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
...
```

これにより、特定のストレージクラスを指定しない永続ボリューム要求 (PVC) がデフォルトのストレージクラスによって自動的にプロビジョニングされるようになります。



注記

ベータアノテーションの **storageclass.beta.kubernetes.io/is-default-class** は依然として使用可能ですが、今後のリリースで削除される予定です。

ストレージクラスの記述を設定するには、以下のアノテーションをストレークラスのメタデータに追加します。

```
kubernetes.io/description: My Storage Class Description
```

以下に例を示します。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  annotations:
    kubernetes.io/description: My Storage Class Description
...
```

6.3.3. RHOSP Cinder オブジェクトの定義

cinder-storageclass.yaml

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: gold
provisioner: kubernetes.io/cinder
parameters:
  type: fast (1)
  availability: nova (2)
  fsType: ext4 (3)
```

(1) Cinder で作成されるボリュームタイプ。デフォルトは空です。

(2) アベイラビリティーゾーン。指定しない場合、ボリュームは通常 OpenShift Container Platform クラスターのノードがあるすべてのアクティブゾーンでラウンドロビンされます。

(3) 動的にプロビジョニングされたボリュームで作成されるファイルシステム。この値は、動的にプロビジョニングされる永続ボリュームの **fsType** フィールドにコピーされ、ボリュームの初回マウント時にファイルシステムが作成されます。デフォルト値は **ext4** です。

6.3.4. RHOSP Manila Container Storage Interface (CSI) オブジェクト定義

インストールが完了すると、OpenStack Manila CSI Driver Operator および ManilaDriver は、動的プロビジョニングに必要なすべての利用可能な Manila 共有タイプに必要なストレージクラスを自動的に作成します。

6.3.5. AWS Elastic Block Store (EBS) オブジェクト定義

aws-ebs-storageclass.yaml

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: slow
provisioner: kubernetes.io/aws-ebs
parameters:
  type: io1 (1)
  iopsPerGB: "10" (2)
  encrypted: "true" (3)
  kmsKeyId: keyvalue (4)
  fsType: ext4 (5)
```

1. (必須) **io1**、**gp2**、**sc1**、**st1** から選択します。デフォルトは **gp2** です。有効な Amazon Resource Name (ARN) 値については、AWS のドキュメント を参照してください。

2. (オプション) **io1** ボリュームのみ。1 GiB あたり 1 秒あたりの I/O 処理数。AWS ボリュームプラグインは、この値と要求されたボリュームのサイズを乗算してボリュームの IOPS を算出します。値の上限は、AWS でサポートされる最大値である 20,000 IOPS です。詳細については、AWS のドキュメント を参照してください。

3. (オプション) EBS ボリュームを暗号化するかどうかを示します。有効な値は **true** または **false** です。

4. (オプション) ボリュームを暗号化する際に使用するキーの完全な ARN。値を指定しない場合でも **encypted** が **true** に設定されている場合は、AWS によってキーが生成されます。有効な ARN 値については、AWS のドキュメント を参照してください。

5. (オプション) 動的にプロビジョニングされたボリュームで作成されるファイルシステム。この値は、動的にプロビジョニングされる永続ボリュームの **fsType** フィールドにコピーされ、ボリュームの初回マウント時にファイルシステムが作成されます。デフォルト値は **ext4** です。

6.3.6. Azure Disk オブジェクト定義

azure-advanced-disk-storageclass.yaml

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: managed-premium
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: kubernetes.io/azure-disk
volumeBindingMode: WaitForFirstConsumer (1)
allowVolumeExpansion: true
parameters:
  kind: Managed (2)
  storageaccounttype: Premium_LRS (3)
reclaimPolicy: Delete
```

1. **WaitForFirstConsumer** を使用することが強く推奨されます。これにより、Pod を利用可能なゾーンから空きのあるワーカーノードにスケジュールするのに十分なストレージがボリュームプロビジョニングされます。

2. 許容値は、**Shared** (デフォルト)、**Managed**、および **Dedicated** です。

重要

Red Hat は、ストレージクラスでの **kind: Managed** の使用のみをサポートします。

Shared および **Dedicated** の場合、Azure は管理対象外のディスクを作成しますが、OpenShift Container Platform はマシンの OS (root) ディスクの管理ディスクを作成します。ただし、Azure Disk はノードで管理ディスクおよび管理対象外ディスクの両方の使用を許可しないため、**Shared** または **Dedicated** で作成された管理対象外ディスクを OpenShift Container Platform ノードに割り当てることはできません。

3 Azure ストレージアカウントの SKU の層。デフォルトは空です。プレミアム VM は **Standard_LRS** ディスクと **Premium_LRS** ディスクの両方を割り当て、標準 VM は

a. **kind** が **Shared** に設定されている場合は、Azure は、クラスターと同じリソースグループにあるいくつかの共有ストレージアカウントで、アンマネージドディスクをすべて作成します。

b. **kind** が **Managed** に設定されている場合は、Azure は新しいマネージドディスクを作成します。

c. **kind** が **Dedicated** に設定されており、**storageAccount** が指定されている場合には、Azure は、クラスターと同じリソースグループ内にある新規のアンマネージドディスク用に、指定のストレージアカウントを使用します。これを機能させるには、以下が前提となります。

- 指定のストレージアカウントが、同じリージョン内にあること。
- Azure Cloud Provider にストレージアカウントへの書き込み権限があること。

d. **kind** が **Dedicated** に設定されており、**storageAccount** が指定されていない場合には、Azure はクラスターと同じリソースグループ内の新規のアンマネージドディスク用に、新しい専用のストレージアカウントを作成します。

6.3.7. Azure File のオブジェクト定義

Azure File ストレージクラスはシークレットを使用して Azure ストレージアカウント名と Azure ファイル共有の作成に必要なストレージアカウントキーを保存します。これらのパーミッションは、以下の手順の一部として作成されます。

手順

1. シークレットの作成および表示を可能にする **ClusterRole** オブジェクトを定義します。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
#  name: system:azure-cloud-provider
  name: <persistent-volume-binder-role> 1
rules:
- apiGroups: ['']
  resources: ['secrets']
  verbs:     ['get','create']
```

1 シークレットを表示し、作成するためのクラスターロールの名前。

2. クラスターロールをサービスアカウントに追加します。

```
$ oc adm policy add-cluster-role-to-user <persistent-volume-binder-role>
```

出力例

```
system:serviceaccount:kube-system:persistent-volume-binder
```

3. Azure File **StorageClass** オブジェクトを作成します。

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: <azure-file> (1)
provisioner: kubernetes.io/azure-file
parameters:
  location: eastus (2)
  skuName: Standard_LRS (3)
  storageAccount: <storage-account> (4)
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

(1) ストレージクラス名永続ボリューム要求 (PVC) は、関連する永続ボリュームをプロビジョニングするためにこのストレージクラスを使用します。

(2) **eastus** などの Azure ストレージアカウントの場所。デフォルトは空であり、新規 Azure ストレージアカウントが OpenShift Container Platform クラスターの場所に作成されます。

(3) SKU は、**Standard_LRS** などの Azure ストレージアカウントの層になります。デフォルトは空です。つまり、新しい Azure ストレージアカウントは **Standard_LRS** SKU で作成されます。

(4) Azure ストレージアカウントの名前。ストレージアカウントが提供されると、**skuName** および **location** は無視されます。ストレージアカウントを指定しない場合、ストレージクラスは、定義された **skuName** および **location** に一致するアカウントのリソースグループに関連付けられたストレージアカウントを検索します。

6.3.7.1. Azure File を使用する場合の考慮事項

以下のファイルシステム機能は、デフォルトの Azure File ストレージクラスではサポートされません。

- シンボリックリンク
- ハードリンク
- 拡張属性
- スパースファイル
- 名前付きパイプ

また、Azure File がマウントされるディレクトリーの所有者 ID (UID) は、コンテナーのプロセス UID とは異なります。**uid** マウントオプションは **StorageClass** オブジェクトに指定して、マウントされたディレクトリーに使用する特定のユーザー ID を定義できます。

以下の **StorageClass** オブジェクトは、マウントされたディレクトリーのシンボリックリンクを有効にした状態で、ユーザーおよびグループ ID を変更する方法を示しています。

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
```

```
  name: azure-file
mountOptions:
  - uid=1500 1
  - gid=1500 2
  - mfsymlinks 3
provisioner: kubernetes.io/azure-file
parameters:
  location: eastus
  skuName: Standard_LRS
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

1 マウントされたディレクトリーに使用するユーザー ID を指定します。

2 マウントされたディレクトリーに使用するグループ ID を指定します。

3 シンボリックリンクを有効にします。

6.3.8. GCE PersistentDisk (gcePD) オブジェクトの定義

gce-pd-storageclass.yaml

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: standard
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: kubernetes.io/gce-pd
parameters:
  type: pd-standard 1
  replication-type: none
volumeBindingMode: WaitForFirstConsumer
allowVolumeExpansion: true
reclaimPolicy: Delete
```

1 **pd-standard** または **pd-ssd** のいずれかを選択します。デフォルトは **pd-standard** です。

6.3.9. VMWare vSphere オブジェクトの定義

vsphere-storageclass.yaml

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: slow
provisioner: kubernetes.io/vsphere-volume 1
parameters:
  diskformat: thin 2
```

1. OpenShift Container Platform で VMware vSphere を使用する方法の詳細については、VMware vSphere のドキュメント を参照してください。

2. **diskformat**: **thin**、**zeroedthick** および **eagerzeroedthick** はすべて有効なディスクフォーマットです。ディスクフォーマットの種類に関する詳細は、vSphere のドキュメントを参照してください。デフォルト値は **thin** です。

6.4. デフォルトストレージクラスの変更

AWS を使用している場合は、以下のプロセスを使用してデフォルトのストレージクラスを変更します。このプロセスでは、**gp2** と **standard** の 2 つのストレージクラスが定義されており、デフォルトのストレージクラスを **gp2** から **standard** に変更する必要がある場合を想定しています。

1. ストレージクラスを一覧表示します。

```
$ oc get storageclass
```

出力例

```
NAME                 TYPE
gp2 (default)        kubernetes.io/aws-ebs 1
standard             kubernetes.io/aws-ebs
```

1. **(default)** はデフォルトのストレージクラスを示します。

2. デフォルトのストレージクラスのアノテーション **storageclass.kubernetes.io/is-default-class** の値を **false** に変更します。

```
$ oc patch storageclass gp2 -p '{"metadata": {"annotations": {"storageclass.kubernetes.io/is-default-class": "false"}}}'
```

3. アノテーションを追加するか、またはアノテーションを **storageclass.kubernetes.io/is-default-class=true** として変更することで、別のストレージクラスをデフォルトにします。

```
$ oc patch storageclass standard -p '{"metadata": {"annotations": {"storageclass.kubernetes.io/is-default-class": "true"}}}'
```

4. 変更内容を確認します。

```
$ oc get storageclass
```

出力例

```
NAME                 TYPE
gp2                  kubernetes.io/aws-ebs
standard (default)   kubernetes.io/aws-ebs
```