OpenShift Container Platform 4.12

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: 2024-07-04

OpenShift Container Platform 4.12 ストレージ

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

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 OPENSHIFT CONTAINER PLATFORM ストレージの概要

OpenShift Container Platform は、オンプレミスおよびクラウドプロバイダーの両方で、複数のタイプのストレージをサポートします。OpenShift Container Platform クラスターで、永続データおよび非永続データ用のコンテナーストレージを管理できます。

1.1. OPENSHIFT CONTAINER PLATFORM ストレージの共通用語集

この用語集では、ストレージコンテンツで使用される一般的な用語を定義します。

アクセスモード

ボリュームアクセスモードは、ボリューム機能を表します。アクセスモードを使用して、永続ボリューム要求 (PVC) と永続ボリューム (PV) を一致させることができます。次に、アクセスモードの例を示します。

- ReadWriteOnce (RWO)
- ReadOnlyMany (ROX)
- ReadWriteMany (RWX)
- ReadWriteOncePod (RWOP)

Cinder

すべてのボリュームの管理、セキュリティー、およびスケジューリングを管理する Red Hat OpenStack Platform (RHOSP) 用の Block Storage サービス。

設定マップ

config map は、設定データを Pod に注入する方法を提供します。タイプ **ConfigMap** のボリューム内の config map に格納されたデータを参照できます。Pod で実行しているアプリケーションは、このデータを使用できます。

Container Storage Interface (CSI)

異なるコンテナーオーケストレーション (CO) システム間でコンテナーストレージを管理するための API 仕様。

動的プロビジョニング

このフレームワークを使用すると、ストレージボリュームをオンデマンドで作成できるため、クラスター管理者が永続ストレージを事前にプロビジョニングする必要がなくなります。

一時ストレージ

Pod とコンテナーは、その操作のために一時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。

ファイバーチャネル

データセンター、コンピューターサーバー、スイッチ、およびストレージ間でデータを転送するために使用されるネットワークテクノロジー。

FlexVolume

FlexVolume は、EXEC ベースのモデルを使用してストレージドライバーとインターフェイス接続するツリー外プラグインインターフェイスです。FlexVolume ドライバーバイナリーは、各ノード、場合によってはコントロールプレーンノードの事前定義されたボリュームプラグインパスにインストールする必要があります。

fsGroup

fsGroup は、Pod のファイルシステムグループ ID を定義します。

iSCSI

iSCSI (Internet Small Computer Systems Interface) は、データストレージ施設をリンクするための Internet Protocol ベースのストレージネットワーク標準です。iSCSI ボリュームを使用すると、既存の iSCSI (SCSI over IP) ボリュームを Pod にマウントできます。

hostPath

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。

KMS キー

Key Management Service (KMS) は、さまざまなサービスで必要なレベルのデータ暗号化を実現するのに役立ちます。KMS キーを使用して、データの暗号化、復号化、および再暗号化を行うことができます。

ローカルボリューム

ローカルボリュームは、ディスク、パーティション、ディレクトリーなどのマウントされたローカルストレージデバイスを表します。

NFS

ネットワークファイルシステム (NFS) を利用すると、リモートのホストがネットワーク経由でファイルシステムをマウントし、そのファイルシステムを、ローカルにマウントしているファイルシステムと同じように操作できるようになります。また、システム管理者は、リソースをネットワーク上の中央サーバーに統合することができるようになります。

OpenShift Data Foundation

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非依存永続ストレージのプロバイダーです。

永続ストレージ

Pod とコンテナーは、その操作のために永続的なストレージを必要とする場合があります。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、基盤となるストレージインフラストラクチャーに関する特定の知識がなくても、PVC を使用して PV リソースを要求できます。

永続ボリューム (PV)

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、基盤となるストレージインフラストラクチャーに関する特定の知識がなくても、PVC を使用して PV リソースを要求できます。

永続ボリューム要求 (PVC)

PVC を使用して、PersistentVolume を Pod にマウントできます。クラウド環境の詳細を知らなくてもストレージにアクセスできます。

Pod

OpenShift Container Platform クラスターで実行されている、ボリュームや IP アドレスなどの共有リソースを持つ 1 つ以上のコンテナー。Pod は、定義、デプロイ、および管理される最小のコンピュート単位です。

回収ポリシー

解放後のボリュームの処理方法をクラスターに指示するポリシー。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。

ロールベースアクセス制御 (RBAC)

ロールベースのアクセス制御 (RBAC) は、組織内の個々のユーザーのロールに基づいて、コンピューターまたはネットワークリソースへのアクセスを規制する方法です。

ステートレスアプリケーション

ステートレスアプリケーションは、あるセッションで生成されたクライアントデータを、そのクライアントとの次のセッションで使用するために保存しないアプリケーションプログラムです。

ステートフルアプリケーション

ステートフルアプリケーションは、データを永続ディスクストレージに保存するアプリケーションプログラムです。サーバー、クライアント、およびアプリケーションは、永続ディスクストレージを使用できます。OpenShift Container Platform で **Statefulset** オブジェクトを使用して一連の Pod のデプロイメントとスケーリングを管理し、これらの Pod の順序と一意性を保証できます。

静的プロビジョニング

クラスター管理者は、多数の PV を作成します。PV にはストレージの詳細が含まれます。PV は Kubernetes API に存在し、消費することができます。

ストレージ

OpenShift Container Platform は、オンプレミスおよびクラウドプロバイダーの両方で、多くのタイプのストレージをサポートします。OpenShift Container Platform クラスターで、永続データおよび非永続データ用のコンテナーストレージを管理できます。

ストレージクラス

ストレージクラスは、管理者が提供するストレージのクラスを説明する方法を提供します。さまざまなクラスが、サービスレベルの品質、バックアップポリシー、クラスター管理者によって決定された任意のポリシーにマップされる場合があります。

VMware vSphere の仮想マシンディスク (VMDK) ボリューム

仮想マシンディスク (VMDK) は、仮想マシンで使用される仮想ハードディスクドライブのコンテナーを記述するファイル形式です。

1.2. ストレージタイプ

OpenShift Container Platform ストレージは、一時ストレージおよび永続ストレージという 2 つのカテゴリーに大別されます。

1.2.1. 一時ストレージ

Pod およびコンテナーは性質上、一時的または遷移的であり、ステートレスアプリケーション用に設計されています。一時ストレージを使用すると、管理者および開発者は一部の操作についてローカルストレージをより適切に管理できるようになります。一時ストレージの概要、タイプ、および管理の詳細は、一時ストレージについて を参照してください。

1.2.2. 永続ストレージ

コンテナーにデプロイされるステートフルアプリケーションには永続ストレージが必要です。OpenShift Container Platform は、永続ボリューム (PV) と呼ばれる事前にプロビジョニングされたストレージフレームワークを使用して、クラスター管理者が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。これらのボリューム内のデータは、個々の Pod のライフサイクルを超えて存在することができます。開発者は永続ボリューム要求 (PVC) を使用してストレージ要件を要求できます。永続ストレージの概要、設定、およびライフサイクルの詳細は、永続ストレージについて を参照してください。

1.3. CONTAINER STORAGE INTERFACE (CSI)

CSI は、異なるコンテナーオーケストレーション (CO) システム間でコンテナーストレージを管理するための API 仕様です。基礎となるストレージインフラストラクチャーに関する特定の知識がなくても、コンテナーネイティブ環境でストレージボリュームを管理できます。CSI により、使用しているストレージベンダーに関係なく、ストレージは異なるコンテナーオーケストレーションシステム間で均一に機能します。CSI の詳細は、Container Storage Interface (CSI) の使用 を参照してください。

1.4. 動的プロビジョニング

動的プロビジョニングにより、ストレージボリュームをオンデマンドで作成し、クラスター管理者がストレージを事前にプロビジョニングする必要をなくすことができます。動的プロビジョニングの詳細は、動的プロビジョニング を参照してください。

第2章 一時ストレージについて

2.1. 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、スクラッチ領域、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 が利用可能なローカルストレージの量を検出できない。
- 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が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一杯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 は許可されない。

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を使用すると、管理者および開発者はローカルストレージをより適切に管理できますが、I/O スループットやレイテンシーに直接影響はありません。

2.2. 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と、**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。

2.3. 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。

要求と制限を指定することで、ローカルの一時ストレージを管理できます。Pod 内の各コンテナーは、以下を指定できます。

- **spec.containers[].resources.limits.ephemeral-storage**
- **spec.containers[].resources.requests.ephemeral-storage**

一時ストレージの制限と要求は、バイト単位で測定されます。ストレージは、E、P、T、G、M、k のいずれかの接尾辞を使用して、単純な整数または固定小数点数として表すことができます。Ei、Pi、Ti、Gi、Mi、Ki の 2 のべき乗も使用できます。たとえば、128974848、129e6、129M、および 123Mi はすべてほぼ同じ値を表します。接尾辞の大文字と小文字は区別する必要があります。400m の一時ストレージを指定すると、おそらく意図されたものであろう 400 メビバイト (400Mi) または 400 メガバイト (400M) ではなく、0.4 バイトが要求されます。

次の例は、2 つのコンテナーを持つ Pod を示しています。各コンテナーは、2GiB のローカル一時ストレージを要求します。各コンテナーには、4GiB のローカル一時ストレージの制限があります。したがって、Pod には 4GiB のローカル一時ストレージの要求と、8GiB のローカル一時ストレージの制限があります。

```
apiVersion: v1
kind: Pod
metadata:
  name: frontend
spec:
  containers:
  - name: app
    image: images.my-company.example/app:v4
    resources:
      requests:
        ephemeral-storage: "2Gi" 1
      limits:
        ephemeral-storage: "4Gi" 2
    volumeMounts:
    - name: ephemeral
      mountPath: "/tmp"
  - name: log-aggregator
    image: images.my-company.example/log-aggregator:v6
    resources:
      requests:
        ephemeral-storage: "2Gi" 3
    volumeMounts:
    - name: ephemeral
      mountPath: "/tmp"
  volumes:
    - name: ephemeral
      emptyDir: {}
```

1 3 ローカル一時ストレージの要求。

2 ローカル一時ストレージの制限。

Pod 仕様のこの設定は、スケジューラーが Pod のスケジューリングを決定する方法と、kubelet が Pod を削除する方法に影響します。まず、スケジューラーは、スケジュールされたコンテナーのリソース要求の合計がノードの容量よりも少ないことを確認します。この場合は、使用可能な一時ストレージ (割り当て可能なリソース) が 4GiB を超える場合にのみ、Pod をノードに割り当てることができます。

次に、コンテナーレベルでは、最初のコンテナーがリソース制限を設定するため、kubelet エビクションマネージャーはこのコンテナーのディスク使用量を測定し、このコンテナーのストレージ使用量がその制限 (4GiB) を超える場合は、Pod をエビクトします。Pod レベルでは、kubelet は、その Pod 内のすべてのコンテナーの制限を合計することで、Pod 全体のストレージ制限を計算します。この場合、Pod レベルでの合計ストレージ使用量は、すべてのコンテナーからのディスク使用量と Pod の **emptyDir** ボリュームの合計になります。この合計使用量が全体的な Pod ストレージの制限 (4GiB) を超えた場合、kubelet は Pod にエビクションのマークも付けます。

プロジェクトのクォータの定義については、プロジェクトごとのクォータ設定 を参照してください。

2.4. 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監視できます。**/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3章 永続ストレージについて

3.1. 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、**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。

3.2. 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。

3.2.1. 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。

3.2.2. 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。

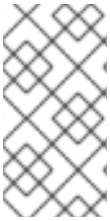
すべての PV のサイズが PVC サイズを超える可能性があります。これは、手動でプロビジョニングされる PV にとくに当てはまります。超過を最小限にするために、OpenShift Container Platform は他のすべての条件に一致する最小の PV にバインドします。

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。

3.2.3. 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は、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は、バインドされた PV を必要な期間保持できます。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。



注記

ファイル数が多い永続ボリュームを Pod に割り当てる場合、それらの Pod は失敗するか、起動に時間がかかる場合があります。詳細は、When using Persistent Volumes with high file counts in OpenShift, why do pods fail to start or take an excessive amount of time to achieve "Ready" state? を参照してください。

3.2.4. 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。

3.2.5. 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ため、ポリシーに基づいて処理される必要があります。

3.2.6. 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を指示します。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。

3.2.7. 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 <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。

3.2.8. 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リスト表示します。

```
$ 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** を削除しても、ボリュームは自動的に削除されません。

3.3. 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。

3.3.1. 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。

- AliCloud Disk
- AWS Elastic Block Store (EBS)
- AWS Elastic File Store (EFS)
- Azure Disk
- Azure File
- Cinder
- ファイバーチャネル
- GCP Persistent Disk
- GCP Filestore
- IBM VPC Block
- HostPath
- iSCSI
- ローカルボリューム
- NFS
- OpenStack Manila
- Red Hat OpenShift Data Foundation
- VMware vSphere

3.3.2. 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。

3.3.3. 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。

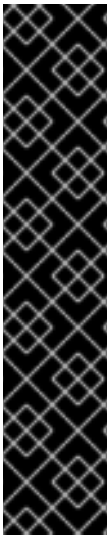
直接的なマッチングが常に最初に試行されます。ボリュームのモードは、要求モードと一致するか、要求した内容以上のものを含む必要があります。サイズは予想されるものより多いか、これと同等である必要があります。2 つのタイプのボリューム (NFS および iSCSI など) のどちらにも同じセットのアクセスモードがある場合、それらのいずれかがそれらのモードを持つ要求に一致する可能性があります。ボリュームのタイプ間で順序付けすることはできず、タイプを選択することはできません。

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3.1 アクセスモード

アクセスモード	CLI の省略形	説明
ReadWriteOnce	**RWO**	ボリュームはシングルノードで読み取り/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3.2 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
AliCloud Disk	▮	-	-
AWS EBS [2]	▮	-	-
AWS EFS	▮	▮	▮
Azure File	▮	▮	▮
Azure Disk	▮	-	-
Cinder	▮	-	-
ファイバーチャネル	▮	▮	▮ [3]
GCP Persistent Disk	▮	-	-
GCP Filestore	▮	▮	▮
HostPath	▮	-	-
IBM VPC Disk	▮	-	-

ボリュームプラグイン	ReadWriteOnce [1]	ReadOnlyMany	ReadWriteMany
iSCSI	▮	▮	▮ [3]
ローカルボリューム	▮	-	-
LVM Storage	▮	-	-
NFS	▮	▮	▮
OpenStack Manila	-	-	▮
Red Hat OpenShift Data Foundation	▮	-	▮
VMware vSphere	▮	-	▮ [4]

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. Amazon 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。

3. RAW ブロックボリュームのみが、ファイバーチャネルおよび iSCSI の ReadWriteMany (RWX) アクセスモードをサポートします。詳細は、ブロックボリュームのサポートを参照してください。

4. 基盤となる vSphere 環境が vSAN ファイルサービスをサポートしている場合、OpenShift Container Platform によってインストールされた vSphere Container Storage Interface (CSI) Driver Operator は ReadWriteMany (RWX) ボリュームのプロビジョニングをサポートします。vSAN ファイルサービスが設定されていない場合に RWX を要求すると、ボリュームの作成に失敗し、エラーがログに記録されます。詳細については、"Using Container Storage Interface" → "VMware vSphere CSI Driver Operator" を参照してください。

3.3.4. 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3.3 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。

フェーズ	説明
Bound	ボリュームが要求にバインドされています。
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>
```

3.3.4.1. マウントオプション

属性 **mountOptions** 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。

以下に例を示します。

マウントオプションの例

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv0001
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  mountOptions: 1
    - nfsvers=4.1
  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 Data Foundation (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。

関連情報

- ReadWriteMany vSphere ボリュームのサポート

3.4. 永続ボリューム要求

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

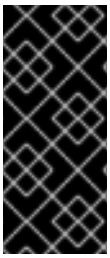
```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: myclaim 1
spec:
  accessModes:
    - ReadWriteOnce 2
  resources:
    requests:
      storage: 8Gi 3
  storageClassName: gold 4
status:
  ...
```

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

3.4.1. 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

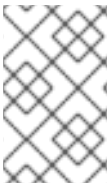
storageClassName を持つ PV のみが PVC にバインドされます。クラスター管理者は 1 つ以上のストレージクラスを提供するように動的プロビジョナーを設定できます。クラスター管理者は、PVC の仕様に一致する PV をオンデマンドで作成できます。



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。

3.4.2. 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。

3.4.3. 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。

3.4.4. 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。コンテナーのルート (/) や、ホストとコンテナーで同じパスにはマウントしないでください。これは、コンテナーに十分な特権が付与されている場合に、ホストシステムを破壊する可能性があります (例: ホストの **/dev/pts** ファイル)。ホストをマウントするには、**/host** を使用するのが安全です。

3 使用する同じ namespace にある PVC の名前

3.5. 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3.4 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	フルサポート
AliCloud Disk	▮	▮	▮
AWS EBS	▮	▮	▮
AWS EFS			
Azure Disk	▮	▮	▮
Azure File			
Cinder	▮	▮	▮
ファイバーチャネル	▮		▮
GCP	▮	▮	▮
HostPath			

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	フルサポート
IBM VPC Disk	▮	▮	▮
iSCSI	▮		▮
ローカルボリューム	▮		▮
LVM Storage	▮	▮	▮
NFS			
Red Hat OpenShift Data Foundation	▮	▮	▮
VMware vSphere	▮	▮	▮

重要

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の使用は、テクノロジープレビュー機能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 製品サポート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場合があります。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。

3.5.1. 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3.5 volumeMode の許容値

値	デフォルト
Filesystem	はい
Block	いいえ

表3.6 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
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。

3.6. FSGROUP を使用した POD タイムアウトの削減

ストレージボリュームに多数のファイル (~1,000,000 以上) が含まれる場合には、Pod のタイムアウトが生じる可能性があります。

デフォルトでは、OpenShift Container Platform は、ボリュームがマウントされる際に Pod の **securityContext** で指定される **fsGroup** に一致するよう、各ボリュームのコンテンツの所有者とパーミッションを再帰的に変更するため、これが発生する可能性があります。大規模なボリュームでは、所有者とパーミッションの確認と変更には時間がかかり、Pod の起動が遅くなる場合があります。**securityContext** 内で **fsGroupChangePolicy** フィールドを使用して、OpenShift Container Platform がボリュームの所有者およびパーミッションを確認し管理する方法を制御できます。

fsGroupChangePolicy は、Pod 内で公開される前にボリュームの所有者およびパーミッションを変更する動作を定義します。このフィールドは、**fsGroup**で制御される所有者およびパーミッションをサポートするボリュームタイプにのみ適用されます。このフィールドには、以下の 2 つの値を指定できま

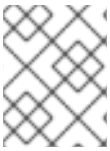
す。

- **OnRootMismatch**: ルートディレクトリーのパーミッションと所有者が、ボリュームの予想されるパーミッションと一致しない場合にのみ、パーミッションと所有者を変更します。これにより、ボリュームの所有者とパーミッションを変更するのに必要な時間を短縮でき、Pod のタイムアウトを減らすことができます。
- **Always**: ボリュームのマウント時に、常にボリュームのパーミッションと所有者を変更します。

fsGroupChangePolicy の例

```
securityContext:
  runAsUser: 1000
  runAsGroup: 3000
  fsGroup: 2000
  fsGroupChangePolicy: "OnRootMismatch" 1
  ...
```

1 **OnRootMismatch** は、再帰的なパーミッション変更をスキップさせるため、Pod のタイムアウトの問題を回避するのに役立ちます。



注記

fsGroupChangePolicyfield は、secret、configMap、および emptydir などの一時ボリュームタイプには影響を及ぼしません。

第4章 CONFIGURING PERSISTENT STORAGE

4.1. 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。

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 EBS ボリュームを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、OpenShift Container Platform クラスター全体で共有できます。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。KMS キーを定義して、AWS のコンテナー永続ボリュームを暗号化できます。

重要

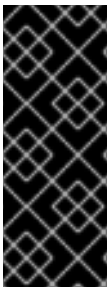
OpenShift Container Platform はデフォルトで、ツリー内または非 Container Storage Interface (CSI) プラグインを使用して AWS EBS ストレージをプロビジョニングします。今後の OpenShift Container Platform バージョンでは、既存の in-tree プラグインを使用してプロビジョニングされるボリュームは、同等の CSI ドライバーに移行される予定です。

CSI 自動移行はシームレスに行ってください。移行をしても、永続ボリューム、永続ボリューム要求、ストレージクラスなどの既存の API オブジェクトを使用する方法は変更されません。移行の詳細は、CSI の自動移行 を参照してください。

完全な移行後、in-tree プラグインは最終的に OpenShift Container Platform の今後のバージョンで削除されます。

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。



重要

OpenShift Container Platform 4.12 以降では、AWS Block インツリーボリュームプラグインと同等の CSI ドライバーに自動的に移行できます。

CSI 自動移行はシームレスに行ってください。移行をしても、永続ボリューム、永続ボリューム要求、ストレージクラスなどの既存の API オブジェクトを使用する方法は変更されません。移行の詳細は、CSI の自動移行 を参照してください。

4.1.1. 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。

4.1.2. 永続ボリューム要求の作成

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し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。この選択により、ストレージクレームの読み取りおよび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を作成し、永続ボリュームを生成します。

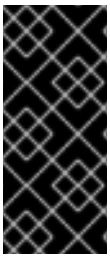
4.1.3. ボリュームのフォーマット

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。

この確認により、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、フォーマットされていない AWS ボリューム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。

4.1.4. 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るため、各タイプの EBS ボリュームを最大 39 個使用できます。

in-tree ボリュームプラグインでは不可能な追加のストレージオプション (ボリュームスナップショットなど) へのアクセスに関する詳細は、AWS Elastic Block Store CSI Driver Operator を参照してください。

4.1.5. KMS キーを使用した AWS 上のコンテナー永続ボリュームの暗号化

AWS でコンテナー永続ボリュームを暗号化するための KMS キーを定義すると、AWS へのデプロイ時に明示的なコンプライアンスおよびセキュリティーのガイドラインがある場合に役立ちます。

前提条件

- 基盤となるインフラストラクチャーには、ストレージが含まれている必要があります。
- AWS で顧客 KMS キーを作成する必要があります。

手順

1. ストレージクラスを作成します。

```
$ cat << EOF | oc create -f -
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: <storage-class-name> 1
parameters:
  fsType: ext4 2
  encrypted: "true"
  kmsKeyId: keyvalue 3
provisioner: ebs.csi.aws.com
reclaimPolicy: Delete
volumeBindingMode: WaitForFirstConsumer
EOF
```

1. ストレージクラスの名前を指定します。
2. プロビジョニングされたボリューム上に作成されるファイルシステム。
3. コンテナー永続ボリュームを暗号化するときに使用するキーの完全な Amazon リソースネーム (ARN) を指定します。キーを指定せず、**encrypted** フィールドが **true** に設定されていると、デフォルトの KMS キーが使用されます。AWS ドキュメントの Finding the key ID and key ARN on AWS の検索を参照してください。

2. KMS キーを指定するストレージクラスで永続ボリューム要求 (PVC) を作成します。

```
$ cat << EOF | oc create -f -
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mypvc
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Filesystem
  storageClassName: <storage-class-name>
  resources:
    requests:
      storage: 1Gi
EOF
```

3. PVC を使用するワークロードコンテナーを作成します。

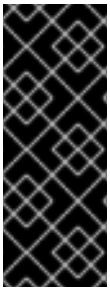
```
$ cat << EOF | oc create -f -
kind: Pod
metadata:
  name: mypod
spec:
  containers:
    - name: httpd
      image: quay.io/centos7/httpd-24-centos7
      ports:
        - containerPort: 80
      volumeMounts:
        - mountPath: /mnt/storage
          name: data
  volumes:
    - name: data
      persistentVolumeClaim:
        claimName: mypvc
EOF
```

4.1.6. 関連情報

- in-tree 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 Driver Operator を参照してください。

4.2. AZURE を使用した永続ストレージ

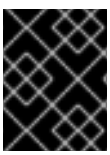
OpenShift Container Platform では、Microsoft Azure Disk ボリュームがサポートされます。Azure を使用して、OpenShift Container Platform クラスターに永続ストレージをプロビジョニングできます。これには、Kubernetes と Azure についてのある程度の理解があることが前提となります。Kubernetes 永続ボリュームフレームワークは、管理者がクラスターのプロビジョニングを永続ストレージを使用して実行できるようにし、ユーザーが基礎となるインフラストラクチャーの知識がなくてもこれらのリソースを要求できるように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

OpenShift Container Platform 4.11 以降では、Azure Disk インツリーボリュームプラグインを同等の CSI ドライバーに自動的に移行します。

CSI 自動移行はシームレスに行ってください。移行をしても、永続ボリューム、永続ボリューム要求、ストレージクラスなどの既存の API オブジェクトを使用する方法は変更されません。移行の詳細は、CSI の自動移行 を参照してください。



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

4.2.1. 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 はマネージド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マネージド外ディスクの両方の使用を許可しないため、**Shared** または **Dedicated** で作成されたマネージド外ディスクを 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

4.2.2. 永続ボリューム要求の作成

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し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。この選択により、ストレージクレームの読み取りおよび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を作成し、永続ボリュームを生成します。

4.2.3. 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。

4.2.4. PVC を使用して Ultra ディスクと共にマシンをデプロイするマシンセット

Ultra ディスクと共にマシンをデプロイする Azure で実行されるマシンセットを作成できます。Ultra ディスクは、最も要求の厳しいデータワークロードでの使用を目的とした高性能ストレージです。

in-tree プラグインおよび CSI ドライバーの両方が、Ultra ディスクを有効にするための PVC の使用をサポートします。PVC を作成せずに、データディスクとしての Ultra デイスクと共にマシンをデプロイすることもできます。

関連情報

- Microsoft Azure Ultra ディスクのドキュメント
- CSI PVC を使用して Ultra ディスクにマシンをデプロイするマシンセット
- データディスクとしての Ultra ディスク上にマシンをデプロイするマシンセット

4.2.4.1. マシンセットを使用した Ultra ディスクを持つマシンの作成

マシンセットの YAML ファイルを編集することで、Azure 上に Ultra ディスクと共にマシンをデプロイできます。

前提条件

- 既存の Microsoft Azure クラスターがある。

手順

1. 既存の Azure **MachineSet** カスタムリソース (CR) をコピーし、次のコマンドを実行して編集します。

```
$ oc edit machineset <machine-set-name>
```

ここで、**<machine-set-name>** は、Ultra ディスクと共にマシンをプロビジョニングするマシンセットです。

2. 示された位置に次の行を追加します。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
spec:
  template:
    spec:
      metadata:
        labels:
          disk: ultrassd (1)
      providerSpec:
        value:
          ultraSSDCapability: Enabled (2)
```

(1) このマシンセットによって作成されるノードを選択するために使用するラベルを指定します。この手順では、この値に **disk.ultrassd** を使用します。

(2) これらの行により、Ultra ディスクの使用が可能になります。

3. 次のコマンドを実行して、更新された設定を使用してマシンセットを作成します。

```
$ oc create -f <machine-set-name>.yaml
```

4. 以下の YAML 定義が含まれるストレージクラスを作成します。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ultra-disk-sc (1)
parameters:
  cachingMode: None
  diskIopsReadWrite: "2000" (2)
  diskMbpsReadWrite: "320" (3)
  kind: managed
  skuname: UltraSSD_LRS
provisioner: disk.csi.azure.com (4)
reclaimPolicy: Delete
volumeBindingMode: WaitForFirstConsumer (5)
```

(1) ストレージクラスの名前を指定します。この手順では、この値に **ultra-disk-sc** を使用しています。

(2) ストレージクラスの IOPS の数値を指定します。

3 ストレージクラスのスループットを MBps 単位で指定します。

4 Azure Kubernetes Service (AKS) バージョン 1.21 以降の場合は、**disk.csi.azure.com** を使用します。以前のバージョンの AKS の場合は、**kubernetes.io/azure-disk** を使用します。

5 オプション: ディスクを使用する Pod の作成を待機するには、このパラメーターを指定します。

5. 以下の YAML 定義が含まれる、**ultra-disk-sc** ストレージクラスを参照する永続ボリューム要求 (PVC) を作成します。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: ultra-disk 1
spec:
  accessModes:
  - ReadWriteOnce
  storageClassName: ultra-disk-sc 2
  resources:
    requests:
      storage: 4Gi 3
```

1 PVC の名前を指定します。この手順では、この値に **ultra-disk** を使用しています。

2 この PVC は **ultra-disk-sc** ストレージクラスを参照します。

3 ストレージクラスのサイズを指定します。最小値は **4Gi** です。

6. 以下の YAML 定義が含まれる Pod を作成します。

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-ultra
spec:
  nodeSelector:
    disk: ultrassd 1
  containers:
  - name: nginx-ultra
    image: alpine:latest
    command:
      - "sleep"
      - "infinity"
    volumeMounts:
    - mountPath: "/mnt/azure"
      name: volume
  volumes:
    - name: volume
      persistentVolumeClaim:
        claimName: ultra-disk 2
```

1. Ultra ディスクの使用を有効にするマシンセットのラベルを指定します。この手順では、この値に **disk.ultrassd** を使用します。
2. この Pod は **ultra-disk** PVC を参照します。

検証

1. 次のコマンドを実行して、マシンが作成されていることを確認します。

```
$ oc get machines
```

マシンは **Running** 状態になっているはずです。

2. 実行中でノードが接続されているマシンの場合、次のコマンドを実行してパーティションを検証します。

```
$ oc debug node/<node-name> -- chroot /host lsblk
```

このコマンドでは、**oc debug node/<node-name>** がノード **<node-name>** でデバッグシェルを開始し、**--** を付けてコマンドを渡します。渡されたコマンド **chroot /host** は、基盤となるホスト OS バイナリーへのアクセスを提供し、**lsblk** は、ホスト OS マシンに接続されているブロックデバイスを表示します。

次のステップ

- Pod 内から Ultra ディスクを使用するには、マウントポイントを使用するワークロードを作成します。次の例のような YAML ファイルを作成します。

```
apiVersion: v1
kind: Pod
metadata:
  name: ssd-benchmark1
spec:
  containers:
  - name: ssd-benchmark1
    image: nginx
    ports:
      - containerPort: 80
        name: "http-server"
    volumeMounts:
    - name: lun0p1
      mountPath: "/tmp"
  volumes:
    - name: lun0p1
      hostPath:
        path: /var/lib/lun0p1
        type: DirectoryOrCreate
  nodeSelector:
    disktype: ultrassd
```

4.2.4.2. Ultra ディスクを有効にするマシンセットのリソースに関するトラブルシューティング

このセクションの情報を使用して、発生する可能性のある問題を理解し、回復してください。

4.2.4.2.1. Ultra ディスクがサポートする永続ボリューム要求 (PVC) をマウントできない

Ultra ディスクでサポートされる永続ボリューム要求 (PVC) のマウントに問題がある場合、Pod は **ContainerCreating** 状態のままになり、アラートがトリガーされます。

たとえば、**additionalCapabilities.ultraSSDEnabled** パラメーターが Pod をホストするノードをサポートするマシンで設定されていない場合、以下のエラーメッセージが表示されます。

```
StorageAccountType UltraSSD_LRS can be used only when additionalCapabilities.ultraSSDEnabled is set.
```

- この問題を解決するには、以下のコマンドを実行して Pod を記述します。

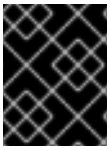
```
$ oc -n <stuck_pod_namespace> describe pod <stuck_pod_name>
```

4.3. 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 ボリュームを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、OpenShift Container Platform クラスター全体で共有できます。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。

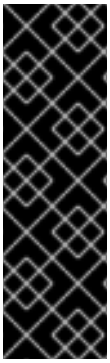


重要

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。

重要

Azure File ボリュームは Server Message Block を使用します。



重要

今後の OpenShift Container Platform バージョンでは、既存の in-tree プラグインを使用してプロビジョニングされるボリュームは、同等の CSI ドライバーに移行される予定です。CSI 自動移行はシームレスに行ってください。移行をしても、永続ボリューム、永続ボリューム要求、ストレージクラスなどの既存の API オブジェクトを使用する方法は変更されません。移行の詳細は、CSI の自動移行 を参照してください。

完全な移行後、in-tree プラグインは最終的に OpenShift Container Platform の今後のバージョンで削除されます。

関連情報

- Azure Files

4.3.1. 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。

4.3.2. 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。コンテナーのルート (/) や、ホストとコンテナーで同じパスにはマウントしないでください。これは、コンテナーに十分な特権が付与されている場合に、ホストシステムを破壊する可能性があります (例: ホストの **/dev/pts**

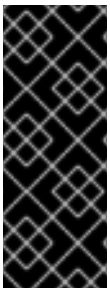
ファイル)。ホストをマウントするには、/**host** を使用するのが安全です。

3 以前に作成された **PersistentVolumeClaim** オブジェクトの名前。

4.4. 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、OpenShift Container Platform クラスター全体で共有できます。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

OpenShift Container Platform 4.11 以降では、Cinder インツリーボリュームプラグインと同等の CSI ドライバーに自動的に移行できます。

CSI 自動移行はシームレスに行ってください。移行をしても、永続ボリューム、永続ボリューム要求、ストレージクラスなどの既存の API オブジェクトを使用する方法は変更されません。移行の詳細は、CSI の自動移行 を参照してください。

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。

4.4.1. 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

4.4.1.1. 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

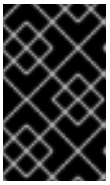
1. オブジェクト定義をファイルに保存します。

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
```

4.4.1.2. 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。

4.4.1.3. 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

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。

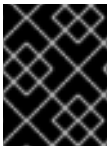
(5) パラメーター拡張後の名前の最大長さは 63 文字です。

(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。

4.5. 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。



重要

ファイバーチャネルを使用する永続ストレージは、ARM アーキテクチャーベースのインフラストラクチャーではサポートされません。

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、OpenShift Container Platform クラスター全体で共有できます。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。

関連情報

- ファイバーチャネルデバイスの使用

4.5.1. 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:
    wwids: [scsi-3600508b400105e210000900000490000] 1
```

```
targetWWNs: ['500a0981891b8dc5', '500a0981991b8dc5'] 2
lun: 2 3
fsType: ext4
```

1 World wide identifier (WWID)**FCwids** または FC **targetWWNs** および **lun** の組み合わせは設定する必要がありますが、両方を同時に設定することはできません。WWN ターゲットよりも FC WWID 識別子が推奨されます。FC WWID 識別子は、各ストレージデバイスに固有のものであり、デバイスのアクセスに使用されるパスに依存しないためです。この識別子は、SCSI Inquiry を発行して Device Identification Vital Product Data (**page 0x83**) または Unit Serial Number (**page 0x80**) を取得することにより獲得できます。FC WWID は、デバイスへのパスが変更したり、別のシステムからデバイスにアクセスする場合でも、ディスク上のデータ参照に **/dev/disk/by-id/** と識別されます。

2 3 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。

4.5.1.1. 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。

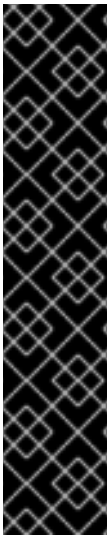
この方法でクォータを実施すると、エンドユーザーは永続ストレージを具体的な量 (10Gi など) で要求することができ、これを同等またはそれ以上の容量の対応するボリュームに一致させることができます。

4.5.1.2. 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。

4.6. FLEXVOLUME を使用した永続ストレージ



重要

FlexVolume は非推奨の機能です。非推奨の機能は依然として OpenShift Container Platform に含まれており、引き続きサポートされますが、本製品の今後のリリースで削除されるため、新規デプロイメントでの使用は推奨されません。

OpenShift Container Platform でボリュームドライバーを作成するには、out-of-tree Container Storage Interface (CSI) ドライバーが推奨されます。FlexVolume ドライバーのメンテナーは、CSI ドライバーを実装し、FlexVolume のユーザーを CSI に移行する必要があります。FlexVolume のユーザーは、ワークロードを CSI ドライバーに移行する必要があります。

OpenShift Container Platform で非推奨となったか、削除された主な機能の最新の一覧については、OpenShift Container Platform リリースノートの **非推奨および削除された機能**セクションを参照してください。

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。

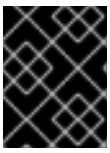
Pod は、**flexvolume** の in-tree 形式のプラグインを使用して FlexVolume ドライバーと対話します。

関連情報

- 永続ボリュームの拡張

4.6.1. 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。

4.6.2. 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。

4.6.3. 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、呼び出す操作のリスト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。

4.6.4. 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、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。

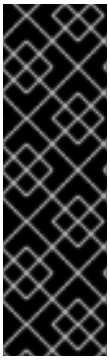
4.7. GCE PERSISTENT DISK を使用した永続ストレージ

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、OpenShift Container Platform クラスター全体で共有できます。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。

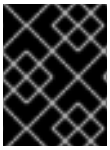


重要

OpenShift Container Platform 4.12 以降では、GCE Persist Disk in-tree ボリュームプラグインと同等の CSI ドライバーに自動的に移行できます。

CSI 自動移行はシームレスに行ってください。移行をしても、永続ボリューム、永続ボリューム要求、ストレージクラスなどの既存の API オブジェクトを使用する方法は変更されません。

移行の詳細は、CSI の自動移行 を参照してください。



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

4.7.1. 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。

4.7.2. 永続ボリューム要求の作成

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し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。この選択により、ストレージクレームの読み取りおよび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を作成し、永続ボリュームを生成します。

4.7.3. 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。

この確認により、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、フォーマットされていない GCE ボリューム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。

4.8. 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

iscsi-initiator-utils パッケージをインストールし、**/etc/iscsi/initiatorname.iscsi** でイニシエーター名を設定して、iSCSI イニシエーターがすべての OpenShift Container Platform ノードですでに設定されていることを確認しておく。**iscsi-initiator-utils** パッケージは、Red Hat Enterprise Linux CoreOS (RHCOS) を使用するデプロイメントにすでにインストールされている。

詳細は、ストレージデバイスの管理 を参照してください。

4.8.1. 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'
```

4.8.2. 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。

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、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。

4.8.3. 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。

4.8.3.1. 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 Container Platform 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。

4.8.4. 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。

4.8.5. 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。

4.9. 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

- NFS 共有のマウント

4.9.1. 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 サーバーのリスト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)
  volumeName: pv0001
  storageClassName: ""
```

(1) アクセスモード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
```

出力例

```
NAME         STATUS   VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   AGE
nfs-claim1   Bound    pv0001   5Gi        RWO                           2m
```

4.9.2. 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。

4.9.3. 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、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用して、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

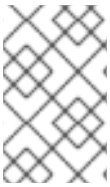
```
drwxrws---. nfsnobody 5555 unconfined_u:object_r:usr_t:s0   /opt/nfs
```

```
$ 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。

4.9.3.1. 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。以下に例を示します。

```
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。

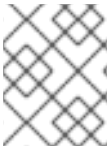
その結果、上記の Pod は受付をパスして起動します。しかし、グループ ID の範囲をチェックすることが望ましい場合は、カスタム SCC の使用が推奨されます。カスタム SCC は、最小および最大のグループ ID が定義され、グループ ID の範囲チェックが実施され、グループ ID の **5555** が許可されるように作成できます。

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。

4.9.3.2. 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。

4.9.3.3. 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
```

4.9.3.4. 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、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。

4.9.4. 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用して作成されます。

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 サーバー情報を使用して新規 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。

4.9.5. その他の設定とトラブルシューティング

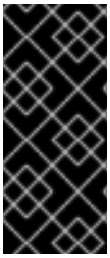
適切なエクスポートとセキュリティーマッピングを行うため、使用している NFS のバージョンおよびその設定方法に応じて追加の設定が必要になることがあります。以下は例になります。

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`

4.10. RED HAT OPENSHIFT DATA FOUNDATION

Red Hat OpenShift Data Foundation 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 Data Foundation は、デプロイメント、管理およびモニタ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 Data Foundation は、独自のドキュメントライブラリーを提供します。Red Hat OpenShift Data Foundation ドキュメントの完全なセットは、https://access.redhat.com/documentation/ja-jp/red_hat_openshift_data_foundation から利用できます。



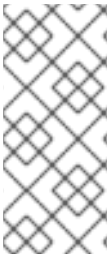
重要

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 Data Foundation は、サポート対象の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 Data Foundation Supportability and Interoperability Guide を参照してください。

4.11. 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。

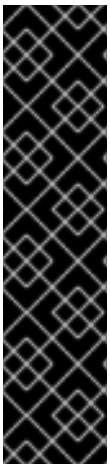


注記

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、OpenShift Container Platform クラスター全体で共有できます。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

OpenShift Container Platform は、vSphere 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 または CSI 以外のドライバーの使用に設定されます。

今後の OpenShift Container Platform バージョンでは、既存の in-tree プラグインを使用してプロビジョニングされるボリュームは、同等の CSI ドライバーに移行される予定です。CSI 自動移行はシームレスに行ってください。移行をしても、永続ボリューム、永続ボリューム要求、ストレージクラスなどの既存の API オブジェクトを使用する方法は変更されません。移行の詳細は、CSI の自動移行 を参照してください。

完全な移行後、in-tree プラグインは最終的に OpenShift Container Platform の今後のバージョンで削除されます。

関連情報

- VMware vSphere

4.11.1. 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。

4.11.2. 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 にクラスターをインストールする を参照してください。

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。

4.11.2.1. 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を作成し、永続ボリュームを生成します。

4.11.2.2. 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. 次のコマンドを入力して、ファイルから **PersistentVolumeClaim** オブジェクトを作成します。

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

4.11.3. 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。

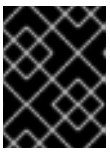
2. このボリュームに割り当てられるストレージの量。

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、ボリュームはシングル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
```

4.11.3.1. 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。

4.12. ローカルストレージを使用した永続ストレージ

4.12.1. 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、標準の永続ボリューム要求 (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。

4.12.1.1. 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. **openshift-local-storage** プロジェクトを作成します。

```
$ oc adm new-project openshift-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 namespace openshift-local-storage openshift.io/node-selector=''
```

3. オプション: 単一ノードデプロイメントの CPU の管理プールでローカルストレージを実行できるようにします。
シングルノードデプロイメントで Local Storage Operator を使用し、**literal** プールに属する CPU の使用を許可します。この手順は、管理ワークロードパーティショニングを使用する単一ノードインストールで実行します。

   Local Storage Operator が管理 CPU プールで実行できるようにするには、次のコマンドを実行します。

```
$ oc annotate namespace openshift-local-storage
workload.openshift.io/allowed='management'
```

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 **openshift-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** セクションにリスト表示されます。

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. 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: **openshift-local-storage.yaml**) を作成します。

      例: openshift-local-storage.yaml

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: local-operator-group
  namespace: openshift-local-storage
spec:
  targetNamespaces:
    - openshift-local-storage
---
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
```

```
metadata:
  name: local-storage-operator
  namespace: openshift-local-storage
spec:
  channel: stable
  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認可ポリシー。

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 openshift-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 openshift-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 **openshift-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 openshift-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。

4.12.1.2. 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。
- 以下の条件を満たすローカルディスクがある。
  - ノードに接続されている。
  - マウントされていない。
  - パーティションが含まれていない。

手順

1. ローカルボリュームリソースを作成します。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。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: "openshift-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" (3)
      volumeMode: Filesystem (4)
      fsType: xfs (5)
      devicePaths: (6)
        - /path/to/device (7)
```

(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) 永続ボリュームオブジェクトの作成時に使用するストレージクラスの名前。ローカルストレージ Operator は、ストレージクラスが存在しない場合にこれを自動的に作成します。

このローカルボリュームのセットを一意に識別するストレージクラスを使用するようにしてください。

4 ローカルボリュームのタイプを定義するボリュームモード (**Filesystem** または **Block**)。

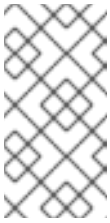
注記

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。

5 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。

6 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。

7 この値を、**LocalVolume** リソース**by-id**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 (例: **/dev/disk/by-id/wwn**)。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

RHEL KVM を使用して OpenShift Container Platform を実行している場合は、VM ディスクにシリアル番号を割り当てる必要があります。そうしないと、再起動後に VM ディスクを識別できません。**virsh edit <VM>** コマンドを使用して、**<serial>mydisk</serial>** 定義を追加できます。

例: ブロック

```
apiVersion: "local.storage.openshift.io/v1"
kind: "LocalVolume"
metadata:
  name: "local-disks"
  namespace: "openshift-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: "local-sc" 3
      volumeMode: Block 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

3 永続ボリュームオブジェクトの作成時に使用するストレージクラスの名前。

4 ローカルボリュームのタイプを定義するボリュームモード (**Filesystem** または **Block**)。

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、**LocalVolume** リソース**by-id**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 (例: **dev/disk/by-id/wwn**)。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

RHEL KVM を使用して OpenShift Container Platform を実行している場合は、VM ディスクにシリアル番号を割り当てる必要があります。そうしないと、再起動後に VM ディスクを識別できません。**virsh edit <VM>** コマンドを使用して、**<serial>mydisk</serial>** 定義を追加できます。

2. 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。作成したばかりのファイルを指定します。

```
$ 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 openshift-local-storage
```

出力例

```
NAME                                      READY   STATUS    RESTARTS   AGE
pod/diskmaker-manager-9wzms               1/1     Running   0          5m43s
pod/diskmaker-manager-jgvjp               1/1     Running   0          5m43s
pod/diskmaker-manager-tbdsj               1/1     Running   0          5m43s
pod/local-storage-operator-7db4bd9f79-t6k87   1/1     Running   0          14m

NAME                                     TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)
AGE
service/local-storage-operator-metrics   ClusterIP   172.30.135.36   <none>
8383/TCP,8686/TCP   14m

NAME                                  DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE
NODE SELECTOR   AGE
daemonset.apps/diskmaker-manager   3         3         3       3            3           <none>
5m43s

NAME                                     READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/local-storage-operator   1/1     1            1           14m

NAME                                                DESIRED   CURRENT   READY   AGE
replicaset.apps/local-storage-operator-7db4bd9f79   1         1         1       14m
```

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

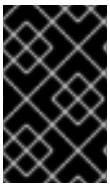
```
NAME                CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM
STORAGECLASS   REASON   AGE
local-pv-1cec77cf   100Gi      RWO            Delete           Available           local-sc                88m
local-pv-2ef7cd2a   100Gi      RWO            Delete           Available    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

4.12.1.3. 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

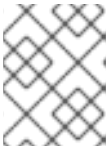
PV の手動プロビジョニングには、PVC の削除時に PV 全体でデータ漏洩が発生するリスクが含まれます。ローカルストレージ Operator は、ローカル PV のプロビジョニング時にデバイスのライフサイクルを自動化するために使用することが推奨されます。

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

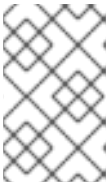
```
apiVersion: v1
kind: PersistentVolume
```

```
metadata:
  name: example-pv-filesystem
spec:
  capacity:
    storage: 100Gi
  volumeMode: Filesystem 1
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Delete
  storageClassName: local-sc 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リスト、またはディレクトリーが含まれるパスです。**Filesystem volumeMode** のディレクトリーのみを指定できます。



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-sc 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-sc
3m47s
example-pv1         1Gi        RWO            Delete           Bound       local-storage/pvc1   local-
sc              12h
example-pv2         1Gi        RWO            Delete           Bound       local-storage/pvc2   local-
sc              12h
example-pv3         1Gi        RWO            Delete           Bound       local-storage/pvc3   local-
sc              12h
```

4.12.1.4. 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
```

4.12.1.5. ローカル要求を割り当てます。

ローカルボリュームが永続ボリューム要求 (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

- 永続ボリューム要求 (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 内で永続ボリューム要求 (PVC) を宣言します。

```
apiVersion: v1
kind: Pod
spec:
# ...
  containers:
    volumeMounts:
    - name: local-disks (1)
      mountPath: /data (2)
  volumes:
  - name: local-disks
    persistentVolumeClaim:
      claimName: local-pvc-name (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。コンテナーのルート (/) や、ホストとコンテナーで同じパスにはマウントしないでください。これは、コンテナーに十分な特権が付与

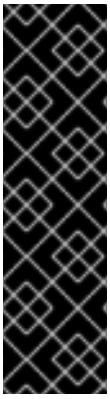
3 使用する既存の永続ボリューム要求 (PVC) の名前。

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
```

4.12.1.6. 詳細は、ローカルストレージデバイスの自動検出およびプロビジョニングを参照してください。

ローカルストレージ Operator はローカルストレージ検出およびプロビジョニングを自動化します。この機能を使用すると、ベアメタル、VMware、または割り当てられたデバイスを持つ AWS ストアインスタンスなど、デプロイメント時に動的プロビジョニングが利用できない場合にインストールを単純化できます。



重要

自動検出およびプロビジョニングはテクノロジープレビュー機能としてのみご利用いただけます。テクノロジープレビュー機能は、Red Hat 製品サポート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場合があります。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。

重要

Red Hat OpenShift Data Foundation をオンプレミスでデプロイするために使用する場合、またはプラットフォームに依存しないデプロイメントで使用する場合、自動検出とプロビジョニングは完全にサポートされます。

ローカルデバイスを自動的に検出し、選択したデバイスのローカルボリュームを自動的にプロビジョニングするには、以下の手順を使用します。

警告

LocalVolumeSet オブジェクトの使用には注意が必要です。ローカルディスクから永続ボリューム (PV) を自動的にプロビジョニングする場合、ローカル PV は一致するすべてのデバイスを要求する可能性があります。**LocalVolumeSet** オブジェクトを使用している場合、ローカルストレージ Operator がノードでローカルデバイスを管理する唯一のエンティティーであることを確認します。ノードを複数回ターゲットにする **Local VolumeSet** のインスタンスを複数作成することはサポートされていません。

前提条件

- クラスター管理者パーミッションがある。
- 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。
- OpenShift Container Platform Web コンソールまたは **oc** コマンドラインインターフェイス (CLI) へのアクセスがあること。

手順

1. Web コンソールからローカルデバイスの自動検出を有効にするには、以下を行います。
   a. **Operators → Installed Operators** をクリックします。
   b. **openshift-local-storage** namespace で **Local Storage** をクリックします。
   c. **Local Volume Discovery** タブをクリックします。
   d. **Create Local Volume Discovery**をクリックし、**Form view** または **YAML view** のいずれかを選択します。
   e. **LocalVolumeDiscovery** オブジェクトパラメーターを設定します。
   f. **Create** をクリックします。
   Local Storage Operator は、**auto-discover-devices** という名前のローカルボリューム検出インスタンスを作成します。
2. ノードで利用可能なデバイスの連続リストを表示するには、以下を実行します。
   a. OpenShift Container Platform Web コンソールにログインします。
   b. **Compute → Nodes** に移動します。
   c. 開くノードの名前をクリックします。「Node Details」ページが表示されます。
   d. **Disks** タブを選択して、選択したデバイスのリストを表示します。
   ローカルディスクを追加または削除しても、デバイスリストの更新が継続的に行われます。名前、ステータス、タイプ、モデル、容量、およびモードでデバイスをフィルターできます。
3. Web コンソールから検出されたデバイスのローカルボリュームを自動的にプロビジョニングするには、以下を実行します。
   a. **Operators → Installed Operators** に移動し、Operator のリストから **Local Storage** を選択します。
   b. **Local Volume Set → Create Local Volume Set**を選択します。
   c. ボリュームセット名とストレージクラス名を入力します。
   d. **All nodes** または **Select nodes** を選択し、適宜フィルターを適用します。

注記

All nodes または **Select nodes** を使用してフィルターするかどうかにかかわらず、ワーカーノードのみが利用可能になります。

e. ローカルボリュームセットに適用するディスクタイプ、モード、サイズ、および制限を選択し、**Create** をクリックします。
メッセージが数分後に表示され、「Operator reconciled successfully」という Operator の調整が正常に行われたことが示唆されます。

4. または、CLI から検出されたデバイスのローカルボリュームをプロビジョニングするには、以下を実行します。

a. 以下の例に示されるように、オブジェクト YAML ファイルを作成し、**local-volume-set.yaml** などのローカルボリュームセットを定義します。

```
apiVersion: local.storage.openshift.io/v1alpha1
kind: LocalVolumeSet
metadata:
  name: example-autodetect
spec:
  nodeSelector:
    nodeSelectorTerms:
      - matchExpressions:
          - key: kubernetes.io/hostname
            operator: In
            values:
              - worker-0
              - worker-1
  storageClassName: local-sc (1)
  volumeMode: Filesystem
  fsType: ext4
  maxDeviceCount: 10
  deviceInclusionSpec:
    deviceTypes: (2)
      - disk
      - part
    deviceMechanicalProperties:
      - NonRotational
    minSize: 10G
    maxSize: 100G
    models:
      - SAMSUNG
      - Crucial_CT525MX3
    vendors:
      - ATA
      - ST2000LM
```

(1) 検出されたデバイスからプロビジョニングされる永続ボリューム用に作成されるストレージクラスを判別します。ローカルストレージ Operator は、ストレージクラスが存在しない場合にこれを自動的に作成します。このローカルボリュームのセットを一意に識別するストレージクラスを使用するようにしてください。

(2) ローカルボリュームセット機能を使用する場合、ローカルストレージ Operator は論理ボリューム管理 (LVM) デバイスの使用をサポートしません。

b. ローカルボリュームセットオブジェクトを作成します。

```
$ oc apply -f local-volume-set.yaml
```

c. ローカル永続ボリュームがストレージクラスに基づいて動的にプロビジョニング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
```

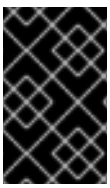
注記

結果は、ノードから削除された後に削除されます。シンボリックリンクは手動で削除する必要があります。

4.12.1.7. 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: "openshift-local-storage"
spec:
  tolerations:
    - key: localstorage (1)
      operator: Equal (2)
      value: "localstorage" (3)
  storageClassDevices:
      - storageClassName: "local-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。

2. オプション: テイントのマークが付けられたノードでのみローカル永続ボリュームを作成するには、以下の例のように YAML ファイルを変更し、**LocalVolume** 仕様を追加します。

```
spec:
  tolerations:
    - key: node-role.kubernetes.io/master
      operator: Exists
```

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。

4.12.1.8. ローカルストレージ Operator メトリクス

OpenShift Container Platform は、ローカルストレージ Operator の以下のメトリクスを提供します。

- **lso_discovery_disk_count**: 各ノードで検出されたデバイスの合計数
- **lso_lvset_provisioned_PV_count**: **LocalVolumeSet** オブジェクトによって作成される PV の合計数

- **lso_lvset_unmatched_disk_count**: 条件の不一致により、ローカルストレージ Operator がプロビジョニング用に選択しなかったディスクの合計数
- **lso_lvset_orphaned_symlink_count**: **LocalVolumeSet** オブジェクト基準に一致しなくなった PV のあるデバイスの数
- **lso_lv_orphaned_symlink_count**: **LocalVolume** オブジェクト基準に一致しなくなった PV のあるデバイスの数
- **lso_lv_provisioned_PV_count**: **LocalVolume** のプロビジョニングされた PV の合計数

これらのメトリクスを使用するには、以下の点を確認してください。

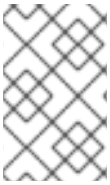
- ローカルストレージ Operator のインストール時に、モニタリングのサポートを有効にする。
- OpenShift Container Platform 4.9 以降にアップグレードする場合は、namespace に **operator-metering=true** ラベルを追加してメトリクスサポートを手動で有効にしてください。

メトリクスの詳細は、メトリクスの管理 を参照してください。

4.12.1.9. ローカルストレージ Operator のリソースの削除

4.12.1.9.1. ローカルボリュームまたはローカルボリュームセットの削除

ローカルボリュームおよびローカルボリュームセットを削除する必要がある場合があります。リソースのエントリーを削除し、永続ボリューム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。



注記

以下の手順では、ローカルボリュームを削除する例の概要を説明します。同じ手順を使用して、ローカルボリュームセットのカスタムリソースのシンボリックリンクを削除することもできます。

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 openshift-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. ノード上のディレクトリーと含まれるシンボリックリンクを削除します。

> **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する必要があります。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。

```
$ oc debug node/<node-name> -- chroot /host rm -rf /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。

4.12.1.9.2. 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 **openshift-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できる。

手順

1. プロジェクトにインストールされているローカルボリュームリソースを削除します (**localvolume**、**localvolumeset**、**localvolumediscovery**等)。

```
$ oc delete localvolume --all --all-namespaces
$ oc delete localvolumeset --all --all-namespaces
$ oc delete localvolumediscovery --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. **openshift-local-storage** プロジェクトを削除します。

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

4.12.2. 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。

4.12.2.1. 概要

OpenShift Container Platform はシングル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。

重要

コンテナーのルート (/) や、ホストとコンテナーで同じパスにはマウントしないでください。これは、コンテナーに十分な特権が付与されている場合、ホストシステムを破壊する可能性があります。ホストをマウントするには、**/host** を使用するのが安全です。以下の例では、ホストの / ディレクトリーが **/host** でコンテナーにマウントされています。

```
apiVersion: v1
kind: Pod
metadata:
  name: test-host-mount
spec:
  containers:
  - image: registry.access.redhat.com/ubi8/ubi
    name: test-container
    command: ['sh', '-c', 'sleep 3600']
    volumeMounts:
    - mountPath: /host
      name: host-slash
  volumes:
   - name: host-slash
     hostPath:
       path: /
       type: ''
```

4.12.2.2. hostPath ボリュームの静的なプロビジョニング

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 ボリュームはシングルノードで **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。コンテナーのルート (/) や、ホストとコンテナーで同じパスにはマウントしないでください。これにより、ホストシステムを破壊する可能性があります。ホストをマウントするには、**/host** を使用するのが安全です。

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
```

4.12.2.3. 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 内にホストパス共有をマウントするパス。コンテナーのルート (/) や、ホストとコンテナーで同じパスにはマウントしないでください。これは、コンテナーに十分な特権が付与されている場合に、ホストシステムを破壊する可能性があります (例: ホストの **/dev/pts** ファイル)。ホストをマウントするには、**/host** を使用するのが安全です。
4. 以前に作成された **PersistentVolumeClaim** オブジェクトの名前。

4.12.3. Logical Volume Manager Storage を使用した永続ストレージ

論理ボリュームマネージャーストレージ (LVM ストレージ) は、TopoLVM CSI ドライバーを使用して、シングルノード OpenShift クラスターでローカルストレージを動的にプロビジョニングします。

LVM ストレージは、論理ボリュームマネージャーを使用してシンプロビジョニングボリュームを作成し、限られたリソースのシングルノード OpenShift クラスターでブロックストレージの動的プロビジョニングを提供します。

LVM Storage を使用すると、ボリュームグループ、永続ボリューム要求 (PVC)、ボリュームスナップショット、およびボリュームクローンを作成できます。

4.12.3.1. Logical Volume Manager Storage のインストール

単一ノードの OpenShift クラスターに論理ボリュームマネージャー(LVM)ストレージをインストールし、ワークロードのストレージを動的にプロビジョニングするように設定できます。

OpenShift Container Platform CLI (**oc**)、OpenShift Container Platform Web コンソール、または Red Hat Advanced Cluster Management (RHACM)を使用して、シングルノード OpenShift クラスターに LVM ストレージをデプロイできます。

4.12.3.1.1. LVM Storage をインストールするための前提条件

LVM Storage をインストールするための前提条件は次のとおりです。

- 最低でも 10 ミリの CPU と 100 MiB の RAM があることを確認してください。
- すべてのマネージドクラスターに、ストレージのプロビジョニングに使用される専用のディスクがあることを確認してください。LVM Storage は、ファイルシステム署名が含まれていない空のディスクのみを使用します。確実にディスクが空で、ファイルシステム署名が含まれていないようにするには、使用する前にディスクを消去します。

- 以前の LVM Storage のインストールで設定したストレージデバイスを再利用できるプライベート CI 環境に LVM Storage をインストールする前に、使用されていないディスクが消去されていることを確認してください。LVM Storage をインストールする前にディスクをワイプしないと、ディスクを再利用するのに手動による介入が必要になります。

注記

使用中のディスクは消去できません。

- Red Hat Advanced Cluster Management (RHACM) を使用して LVM Storage をインストールする場合は、RHACM が OpenShift Container Platform クラスターにインストールされていることを確認してください。**RHACM を使用した LVM Storage のインストール**セクションを参照してください。

関連情報

- Red Hat Advanced Cluster Management for Kubernetes: オンライン接続時のインストール

4.12.3.1.2. OpenShift Container Platform Web コンソールを使用した LVM ストレージのインストール

Red Hat OpenShift Container Platform OperatorHub を使用して LVM ストレージをインストールできます。

前提条件

- シングルノード OpenShift クラスターにアクセスできます。
- **cluster-admin** および Operator のインストール権限を持つアカウントを使用しています。

手順

1. OpenShift Container Platform Web コンソールにログインします。
2. **Operators → OperatorHub** をクリックします。
3. スクロールするか、**LVM Storage** を **Filter by keyword** ボックスに入力して、LVM Storage を見つけます。
4. **Install** をクリックします。
5. **Install Operator** ページで、以下のオプションを設定します。
   a. **stable-4.12** としての **Update Channel**。
   b. **クラスター上の特定の namespace** としての **Installation Mode**。
   c. **Installed Namespace** を **Operator recommended namespace openshift-storage** に設定します。**openshift-storage** namespace が存在しない場合は、Operator のインストール中に作成されます。
   d. **Approval Strategy** を **Automatic** または **Manual** に設定します。
   Automatic (自動) 更新を選択すると、Operator Lifecycle Manager (OLM) は介入なしに、Operator の実行中のインスタンスを自動的にアップグレードします。

   Manual 更新を選択すると、OLM は更新要求を作成します。クラスター管理者は

Manual 更新を選択すると、OLM は更新要求を作成します。クラスター管理者は、Operator を新しいバージョンに更新できるように更新要求を手動で承認する必要があります。

6. **Install** をクリックします。

検証手順

- インストールが成功したことを示す緑色のチェックマークが LVM ストレージに表示されていることを確認します。

4.12.3.1.3. OpenShift Web コンソールを使用してインストールされた LVM ストレージのアンインストール

Red Hat OpenShift Container Platform Web コンソールを使用して、LVM ストレージをアンインストールできます。

前提条件

- LVM Storage によってプロビジョニングされたストレージを使用しているクラスター上のすべてのアプリケーションを削除しました。
- LVM Storage を使用してプロビジョニングされた永続ボリューム要求 (PVC) と永続ボリューム (PV) を削除しました。
- LVM Storage によってプロビジョニングされたすべてのボリュームスナップショットを削除しました。
- **oc get logicalvolume** コマンドを使用して、論理ボリュームリソースが存在しないことを確認しました。
- **cluster-admin** 権限を持つアカウントを使用して、シングルノード OpenShift クラスターにアクセスできます。

手順

1. **Operators → Installed Operators** ページから、**LVM Storage** にスクロールするか、**LVM Storage** を **Filter by name** に入力して検索し、クリックします。
2. **LVMCluster** タブをクリックします。
3. **LVMCluster** ページの右側で、**Actions** ドロップダウンメニューから **Delete LVMCluster** を選択します。
4. **Details** タブをクリックします。
5. **Operator Details** 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. **Remove** を選択します。LVM ストレージは実行を停止し、完全に削除されます。

4.12.3.1.4. RHACM を使用した LVM ストレージのインストール

LVM ストレージは、Red Hat Advanced Cluster Management (RHACM) を使用してシングルノード OpenShift クラスターにデプロイされます。RHACM に **Policy** オブジェクトを作成します。これは、**PlacementRule** リソースで指定されたセレクターに一致するマネージドクラスターに適用される

際に Operator をデプロイおよび設定します。このポリシーは、後でインポートされ、配置ルールを満たすクラスターにも適用されます。

前提条件

- **cluster-admin** および Operator インストール権限を持つアカウントを使用して、RHACM クラスターにアクセスします。
- LVM ストレージによって使用される各シングルノード OpenShift クラスターの専用ディスク。
- シングルノード OpenShift クラスターは、インポートまたは作成された RHACM によって管理される必要があります。

手順

1. OpenShift Container Platform の認証情報を使用して RHACM CLI にログインします。
2. ポリシーを作成する namespace を作成します。

```
# oc create ns lvms-policy-ns
```

3. ポリシーを作成するには、次の YAML を **policy-lvms-operator.yaml** などの名前でファイルに保存します。

```
apiVersion: apps.open-cluster-management.io/v1
kind: PlacementRule
metadata:
  name: placement-install-lvms
spec:
  clusterConditions:
  - status: "True"
    type: ManagedClusterConditionAvailable
  clusterSelector: (1)
    matchExpressions:
    - key: mykey
      operator: In
      values:
      - myvalue
---
apiVersion: policy.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
  name: binding-install-lvms
placementRef:
  apiGroup: apps.open-cluster-management.io
  kind: PlacementRule
  name: placement-install-lvms
subjects:
- apiGroup: policy.open-cluster-management.io
  kind: Policy
  name: install-lvms
---
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
```

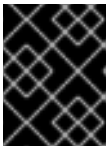
```
  annotations:
    policy.open-cluster-management.io/categories: CM Configuration Management
    policy.open-cluster-management.io/controls: CM-2 Baseline Configuration
    policy.open-cluster-management.io/standards: NIST SP 800-53
  name: install-lvms
spec:
  disabled: false
  remediationAction: enforce
  policy-templates:
  - objectDefinition:
      apiVersion: policy.open-cluster-management.io/v1
      kind: ConfigurationPolicy
      metadata:
        name: install-lvms
      spec:
        object-templates:
        - complianceType: musthave
          objectDefinition:
            apiVersion: v1
            kind: Namespace
            metadata:
              labels:
                openshift.io/cluster-monitoring: "true"
                pod-security.kubernetes.io/enforce: privileged
                pod-security.kubernetes.io/audit: privileged
                pod-security.kubernetes.io/warn: privileged
              name: openshift-storage
        - complianceType: musthave
          objectDefinition:
            apiVersion: operators.coreos.com/v1
            kind: OperatorGroup
            metadata:
              name: openshift-storage-operatorgroup
              namespace: openshift-storage
            spec:
              targetNamespaces:
              - openshift-storage
        - complianceType: musthave
          objectDefinition:
            apiVersion: operators.coreos.com/v1alpha1
            kind: Subscription
            metadata:
              name: lvms
              namespace: openshift-storage
            spec:
              installPlanApproval: Automatic
              name: lvms-operator
              source: redhat-operators
              sourceNamespace: openshift-marketplace
        remediationAction: enforce
        severity: low
  - objectDefinition:
      apiVersion: policy.open-cluster-management.io/v1
      kind: ConfigurationPolicy
      metadata:
        name: lvms
```

```
spec:
  object-templates:
    - complianceType: musthave
      objectDefinition:
        apiVersion: lvm.topolvm.io/v1alpha1
        kind: LVMCluster
        metadata:
          name: my-lvmcluster
          namespace: openshift-storage
        spec:
          storage:
            deviceClasses:
            - name: vg1
              deviceSelector: 2
                paths:
                - /dev/disk/by-path/pci-0000:87:00.0-nvme-1
                - /dev/disk/by-path/pci-0000:88:00.0-nvme-1
              thinPoolConfig:
                name: thin-pool-1
                sizePercent: 90
                overprovisionRatio: 10
              nodeSelector: 3
                nodeSelectorTerms:
                - matchExpressions:
                    - key: app
                      operator: In
                      values:
                      - test1
  remediationAction: enforce
  severity: low
```

1. LVM ストレージをインストールするシングルノード OpenShift クラスターに設定されたラベルと一致するように、**PlacementRule.spec.clusterSelector** のキーと値を置き換えます。

2. ボリュームグループを優先ディスクに制御または制限するには、**LVMCluster** YAML の **deviceSelector** セクションでディスクのローカルパスを手動で指定します。

3. 追加のワーカーノードのサブセットであるノードフィルターを追加するには、**nodeSelector** セクションに必要なフィルターを指定します。LVM Storage は、新しいノードが表示されると、追加のワーカーノードを検出して使用します。



重要

この **nodeSelector** ノードフィルターの一致は、Pod ラベルの一致と同じではありません。

4. 次のコマンドを実行して、namespace にポリシーを作成します。

```
# oc create -f policy-lvms-operator.yaml -n lvms-policy-ns 1
```

1. **policy-lvms-operator.yaml** は、ポリシーが保存されるファイルの名前です。

これにより、**lvms-policy-ns** namespace に **Policy**、**PlacementRule**、および **PlacementBinding** オブジェクトが作成されます。このポリシーは、配置ルールに一致するクラスター上に **Namespace**、**OperatorGroup**、**Subscription**、および **LVMCluster** リソースを作成します。これにより、選択基準に一致するシングルノード OpenShift クラスターに Operator がデプロイされ、ストレージをプロビジョニングするために必要なリソースをセットアップするように設定されます。Operator は **LVMCluster** CR で指定されたすべてのディスクを使用します。ディスクが指定されていない場合、Operator はシングルノード OpenShift ノード上の未使用のディスクをすべて使用します。

重要

LVMCluster に追加されたデバイスは削除できません。

関連情報

- Red Hat Advanced Cluster Management for Kubernetes: オンライン接続時のインストール
- LVM ストレージ参照 YAML ファイル

4.12.3.1.5. RHACM を使用してインストールされた LVM ストレージのアンインストール

RHACM を使用してインストールした LVM Storage をアンインストールするには、Operator のデプロイと設定のために作成した RHACM ポリシーを削除する必要があります。

RHACM ポリシーを削除しても、ポリシーが作成したリソースは削除されません。リソースを削除する追加のポリシーを作成する必要があります。

ポリシーを削除しても、作成されたリソースは削除されないため、次の手順を実行する必要があります。

1. LVM Storage によってプロビジョニングされたすべての永続ボリューム要求 (PVC) とボリュームスナップショットを削除します。
2. **LVMCluster** リソースを削除して、ディスク上に作成された論理ボリュームマネージャーリソースをクリーンアップします。
3. Operator をアンインストールする追加のポリシーを作成します。

前提条件

- ポリシーを削除する前に、以下が削除されていることを確認してください。
  - LVM Storage によってプロビジョニングされたストレージを使用しているマネージドクラスター上のすべてのアプリケーション。
  - LVM Storage を使用してプロビジョニングされた PVC および永続ボリューム (PV)。
  - LVM Storage によってプロビジョニングされたすべてのボリュームスナップショット。
- **cluster-admin** ロールを持つアカウントを使用して RHACM クラスターにアクセスできることを確認します。

手順

1. OpenShift CLI (**oc**) で、次のコマンドを使用して、ハブクラスターに LVM Storage をデプロイおよび設定するために作成した RHACM ポリシーを削除します。

```
# oc delete -f policy-lvms-operator.yaml -n lvms-policy-ns (1)
```

(1) **policy-lvms-operator.yaml** は、ポリシーが保存されたファイルの名前です。

2. **LVMCluster** リソースを削除するためのポリシーを作成するには、次の YAML を **lvms-remove-policy.yaml** などの名前でファイルに保存します。これにより、Operator はクラスター上で作成したすべての論理ボリュームマネージャーリソースをクリーンアップできます。

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name: policy-lvmcluster-delete
  annotations:
    policy.open-cluster-management.io/standards: NIST SP 800-53
    policy.open-cluster-management.io/categories: CM Configuration Management
    policy.open-cluster-management.io/controls: CM-2 Baseline Configuration
spec:
  remediationAction: enforce
  disabled: false
  policy-templates:
    - objectDefinition:
        apiVersion: policy.open-cluster-management.io/v1
        kind: ConfigurationPolicy
        metadata:
          name: policy-lvmcluster-removal
        spec:
          remediationAction: enforce (1)
          severity: low
          object-templates:
            - complianceType: mustnothave
              objectDefinition:
                kind: LVMCluster
                apiVersion: lvm.topolvm.io/v1alpha1
                metadata:
                  name: my-lvmcluster
                  namespace: openshift-storage (2)
---
apiVersion: policy.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
  name: binding-policy-lvmcluster-delete
placementRef:
  apiGroup: apps.open-cluster-management.io
  kind: PlacementRule
  name: placement-policy-lvmcluster-delete
subjects:
  - apiGroup: policy.open-cluster-management.io
    kind: Policy
    name: policy-lvmcluster-delete
---
apiVersion: apps.open-cluster-management.io/v1
kind: PlacementRule
metadata:
  name: placement-policy-lvmcluster-delete
```

```
spec:
  clusterConditions:
    - status: "True"
      type: ManagedClusterConditionAvailable
  clusterSelector:
    matchExpressions:
      - key: mykey
        operator: In
        values:
          - myvalue
```

1. **policy-template** の **spec.remediationAction** は、**spec.remediationAction** の前のパラメーター値によってオーバーライドされます。
2. この **namespace** フィールドには **openshift-storage** 値が必要です。

3. **PlacementRule.spec.clusterSelector** フィールドの値を設定して、LVM Storage をアンインストールするクラスターを選択します。

4. 次のコマンドを実行してポリシーを作成します。

```
# oc create -f lvms-remove-policy.yaml -n lvms-policy-ns
```

5. **LVMCluster** CR が削除されたかどうかを確認するポリシーを作成するには、次の YAML を **check-lvms-remove-policy.yaml** などの名前でファイルに保存します。

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name: policy-lvmcluster-inform
  annotations:
    policy.open-cluster-management.io/standards: NIST SP 800-53
    policy.open-cluster-management.io/categories: CM Configuration Management
    policy.open-cluster-management.io/controls: CM-2 Baseline Configuration
spec:
  remediationAction: inform
  disabled: false
  policy-templates:
    - objectDefinition:
        apiVersion: policy.open-cluster-management.io/v1
        kind: ConfigurationPolicy
        metadata:
          name: policy-lvmcluster-removal-inform
        spec:
          remediationAction: inform (1)
          severity: low
          object-templates:
            - complianceType: mustnothave
              objectDefinition:
                kind: LVMCluster
                apiVersion: lvm.topolvm.io/v1alpha1
                metadata:
                  name: my-lvmcluster
                  namespace: openshift-storage (2)
```

```
---
apiVersion: policy.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
  name: binding-policy-lvmcluster-check
placementRef:
  apiGroup: apps.open-cluster-management.io
  kind: PlacementRule
  name: placement-policy-lvmcluster-check
subjects:
  - apiGroup: policy.open-cluster-management.io
    kind: Policy
    name: policy-lvmcluster-inform
---
apiVersion: apps.open-cluster-management.io/v1
kind: PlacementRule
metadata:
  name: placement-policy-lvmcluster-check
spec:
  clusterConditions:
    - status: "True"
      type: ManagedClusterConditionAvailable
  clusterSelector:
    matchExpressions:
      - key: mykey
        operator: In
        values:
          - myvalue
```

1. **policy-template** の **spec.remediationAction** は、**spec.remediationAction** の前のパラメーター値によってオーバーライドされます。
2. **namespace** フィールドには **openshift-storage** 値が必要です。

6. 次のコマンドを実行してポリシーを作成します。

```
# oc create -f check-lvms-remove-policy.yaml -n lvms-policy-ns
```

7. 次のコマンドを実行して、ポリシーのステータスを確認します。

```
# oc get policy -n lvms-policy-ns
```

出力例

```
NAME                        REMEDIATION ACTION   COMPLIANCE STATE   AGE
policy-lvmcluster-delete    enforce              Compliant          15m
policy-lvmcluster-inform    inform               Compliant          15m
```

8. 両方のポリシーに準拠したら、次の YAML を **lvms-uninstall-policy.yaml** などの名前のファイルに保存して、LVM Storage をアンインストールするポリシーを作成します。

```
apiVersion: apps.open-cluster-management.io/v1
kind: PlacementRule
metadata:
```

```
  name: placement-uninstall-lvms
spec:
  clusterConditions:
  - status: "True"
    type: ManagedClusterConditionAvailable
  clusterSelector:
    matchExpressions:
    - key: mykey
      operator: In
      values:
      - myvalue
---
apiVersion: policy.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
  name: binding-uninstall-lvms
placementRef:
  apiGroup: apps.open-cluster-management.io
  kind: PlacementRule
  name: placement-uninstall-lvms
subjects:
- apiGroup: policy.open-cluster-management.io
  kind: Policy
  name: uninstall-lvms
---
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  annotations:
    policy.open-cluster-management.io/categories: CM Configuration Management
    policy.open-cluster-management.io/controls: CM-2 Baseline Configuration
    policy.open-cluster-management.io/standards: NIST SP 800-53
  name: uninstall-lvms
spec:
  disabled: false
  policy-templates:
  - objectDefinition:
      apiVersion: policy.open-cluster-management.io/v1
      kind: ConfigurationPolicy
      metadata:
        name: uninstall-lvms
      spec:
        object-templates:
        - complianceType: mustnothave
          objectDefinition:
            apiVersion: v1
            kind: Namespace
            metadata:
              name: openshift-storage
        - complianceType: mustnothave
          objectDefinition:
            apiVersion: operators.coreos.com/v1
            kind: OperatorGroup
            metadata:
              name: openshift-storage-operatorgroup
              namespace: openshift-storage
```

```
          spec:
            targetNamespaces:
            - openshift-storage
        - complianceType: mustnothave
          objectDefinition:
            apiVersion: operators.coreos.com/v1alpha1
            kind: Subscription
            metadata:
              name: lvms-operator
              namespace: openshift-storage
        remediationAction: enforce
        severity: low
    - objectDefinition:
        apiVersion: policy.open-cluster-management.io/v1
        kind: ConfigurationPolicy
        metadata:
          name: policy-remove-lvms-crds
        spec:
          object-templates:
          - complianceType: mustnothave
            objectDefinition:
              apiVersion: apiextensions.k8s.io/v1
              kind: CustomResourceDefinition
              metadata:
                name: logicalvolumes.topolvm.io
          - complianceType: mustnothave
            objectDefinition:
              apiVersion: apiextensions.k8s.io/v1
              kind: CustomResourceDefinition
              metadata:
                name: lvmclusters.lvm.topolvm.io
          - complianceType: mustnothave
            objectDefinition:
              apiVersion: apiextensions.k8s.io/v1
              kind: CustomResourceDefinition
              metadata:
                name: lvmvolumegroupnodestatuses.lvm.topolvm.io
          - complianceType: mustnothave
            objectDefinition:
              apiVersion: apiextensions.k8s.io/v1
              kind: CustomResourceDefinition
              metadata:
                name: lvmvolumegroups.lvm.topolvm.io
          remediationAction: enforce
          severity: high
```

9. 次のコマンドを実行してポリシーを作成します。

```
# oc create -f lvms-uninstall-policy.yaml -ns lvms-policy-ns
```

関連情報

- LVM ストレージ参照 YAML ファイル

4.12.3.2. LVM Storage で使用するデバイスのサイズを設定する際の制限事項

LVM Storage を使用したストレージのプロビジョニングで使用できるデバイスのサイズを設定する際の制限は、次のとおりです。

- プロビジョニングできる合計ストレージサイズは、基礎となる論理ボリュームマネージャー (LVM) シンプールのサイズとオーバープロビジョニング係数によって制限されます。
- 論理ボリュームのサイズは、物理エクステント (PE) のサイズと論理エクステント (LE) のサイズによって異なります。
  - PE および LE のサイズは、物理デバイスおよび論理デバイスの作成時に定義できます。
  - デフォルトの PE および LE サイズは 4 MB です。
  - PE のサイズを大きくした場合、LVM の最大サイズは、カーネルの制限とディスク領域によって決定されます。

表4.1 デフォルトの PE および LE サイズを使用した各アーキテクチャーのサイズ制限

アーキテクチャー	RHEL 6	RHEL 7	RHEL 8	RHEL 9
32 ビット	16 TB	-	-	-
64 ビット	8 EB [1] 100 TB [2]	8 EB [1] 500 TB [2]	8 EB	8 EB

1. 理論的サイズ。
2. テスト済みサイズ。

4.12.3.3. 論理ボリュームマネージャークラスターの作成

LVM ストレージをインストールした後、論理ボリュームマネージャークラスターを作成できます。

OpenShift Container Platform は、ユーザーがプロビジョニングしたベアメタルインフラストラクチャー上のシングルノード OpenShift クラスターの追加のワーカーノードをサポートします。LVM ストレージは、新しいノードが表示されると、追加のワーカーノードを検出して使用します。追加のワーカーノードにノードフィルターを設定する必要がある場合は、クラスターの作成中に YAML ビューを使用できます。

OpenShift Container Platform クラスターでは、**LVMCluster** カスタムリソース (CR) のインスタンスを 1 つだけ作成できます。

重要

このノードフィルターの一致は、Pod ラベルの一致と同じではありません。

前提条件

- OperatorHub から LVM Storage をインストールしました。

手順

1. OpenShift Container Platform Web コンソールで、**Operators → Installed Operators** をクリックして、インストールされているすべての Operator を表示します。
   選択された **Project** が **openshift-storage** であることを確認します。

2. **LVM Storage** をクリックし、**LVMCluster** の下の **Create LVMCluster** をクリックします。

3. **Create LVMCluster** ページで、**Form view** または **YAML view** のいずれかを選択します。

4. クラスターの名前を入力します。

5. **Create** をクリックします。

6. オプション: ノードフィルターを追加するには、**YAML view** をクリックし、**nodeSelector** セクションでフィルターを指定します。

```
apiVersion: lvm.topolvm.io/v1alpha1
kind: LVMCluster
metadata:
  name: my-lvmcluster
spec:
  storage:
    deviceClasses:
      - name: vg1
        thinPoolConfig:
          name: thin-pool-1
          sizePercent: 90
          overprovisionRatio: 10
        nodeSelector:
          nodeSelectorTerms:
            - matchExpressions:
              - key: app
                operator: In
                values:
                - test1
```

7. オプション: ディスクのローカルデバイスパスを編集するには、**YAML view** をクリックし、**deviceSelector** セクションでデバイスパスを指定します。

```
spec:
  storage:
    deviceClasses:
      - name: vg1
        deviceSelector:
          paths:
          - /dev/disk/by-path/pci-0000:87:00.0-nvme-1
          - /dev/disk/by-path/pci-0000:88:00.0-nvme-1
        thinPoolConfig:
          name: thin-pool-1
          sizePercent: 90
          overprovisionRatio: 10
```

検証手順

1. OpenShift Container Platform Web コンソールの左ペインから **Storage → Storage Classes** をクリックします。

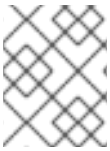
2. **LVMCluster** の作成で **lvms-<device-class-name>** ストレージクラスが作成されていることを確認します。デフォルトでは、**vg1** は **device-class-name** です。

関連情報

- シングルノード OpenShift クラスターへのワーカーノードの追加
- LVM ストレージ参照 YAML ファイル

4.12.3.4. LVM ストレージを使用したストレージのプロビジョニング

Operator のインストール中に作成されたストレージクラスを使用して、永続ボリューム要求 (PVC) をプロビジョニングできます。ブロックおよびファイル PVC をプロビジョニングできますが、ストレージは、PVC を使用する Pod が作成された場合にのみ割り当てられます。



注記

LVM Storage は、PVC を 1 GiB 単位でプロビジョニングします。要求されたストレージは、最も近い GiB に切り上げられます。

手順

1. LVM Storage のデプロイ時に作成される **StorageClass** を特定します。**StorageClass** 名の形式は **lvms-<device-class-name>** です。**device-class-name** は、**Policy** YAML の **LVMCluster** で指定したデバイスクラスの名前です。たとえば、**deviceClass** の名前が **vg1** の場合、**storageClass** の名前は **lvms-vg1** です。

   ストレージクラスの **volumeBindingMode** は **WaitForFirstConsumer** に設定されます。

2. アプリケーションがストレージを必要とする PVC を作成するには、次の YAML を **pvc.yaml** などの名前でファイルに保存します。

   PVC を作成する YAML の例

```
# block pvc
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: lvm-block-1
  namespace: default
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Block
  resources:
    requests:
      storage: 10Gi
  storageClassName: lvms-vg1
---
# file pvc
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: lvm-file-1
  namespace: default
```

```
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Filesystem
  resources:
    requests:
      storage: 10Gi
  storageClassName: lvms-vg1
```

3. 以下のコマンドを実行して PVC を作成します。

```
# oc create -f pvc.yaml -ns <application_namespace>
```

作成された PVC は、それらを使用する Pod をデプロイするまで **pending** 状態のままになります。

4.12.3.5. シングルノード OpenShift クラスターのストレージのスケーリング

OpenShift Container Platform は、ユーザーがプロビジョニングしたベアメタルインフラストラクチャー上のシングルノード OpenShift クラスターの追加のワーカーノードをサポートします。LVM ストレージは、ノードが表示されると、新しい追加のワーカーノードを検出して使用します。

関連情報

- シングルノード OpenShift クラスターへのワーカーノードの追加

4.12.3.5.1. シングルノード OpenShift クラスターに容量を追加してストレージをスケールアップする

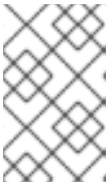
シングルノード OpenShift クラスターで設定済みのワーカーノードのストレージ容量をスケーリングするには、ディスクを追加して容量を増やすことができます。

前提条件

- シングルノード OpenShift クラスターごとに、LVM ストレージで使用される追加の未使用ディスクがあります。

手順

1. シングルノード OpenShift クラスターの OpenShift Container Platform コンソールにログインします。
2. **Operators → Installed Operators** ページで、**openshift-storage** namespace の **LVM Storage Operator** をクリックします。
3. **LVMCluster** タブをクリックして、クラスターで作成された **LVMCluster** CR を一覧表示します。
4. **Actions** ドロップダウンメニューから **Edit LVMCluster** を選択します。
5. **YAML** タブをクリックします。
6. **LVMCluster** CR YAML を編集して、**deviceSelector** セクションに新しいデバイスパスを追加します。



注記

LVMCluster の作成中に **deviceSelector** フィールドが含まれていない場合、**deviceSelector** セクションを CR に追加することはできません。**LVMCluster** を削除してから、新しい CR を作成する必要があります。

```
apiVersion: lvm.topolvm.io/v1alpha1
kind: LVMCluster
metadata:
  name: my-lvmcluster
spec:
  storage:
    deviceClasses:
    - name: vg1
      deviceSelector:
        paths:
        - /dev/disk/by-path/pci-0000:87:00.0-nvme-1 (1)
        - /dev/disk/by-path/pci-0000:88:00.0-nvme-1
        - /dev/disk/by-path/pci-0000:89:00.0-nvme-1 (2)
      thinPoolConfig:
        name: thin-pool-1
        sizePercent: 90
        overprovisionRatio: 10
```

1. パスは、名前 (/**dev**/**sdb**) またはパスで追加できます。
2. 新しいディスクが追加されます。

関連情報

- LVM ストレージ参照 YAML ファイル

4.12.3.5.2. RHACM を使用してシングルノード OpenShift クラスターに容量を追加してストレージをスケールアップする

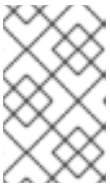
RHACM を使用して、シングルノード OpenShift クラスターで設定済みのワーカーノードのストレージ容量をスケーリングできます。

前提条件

- **cluster-admin** 権限を持つアカウントを使用して RHACM クラスターにアクセスできます。
- シングルノード OpenShift クラスターごとに、LVM ストレージで使用される追加の未使用ディスクがあります。

手順

1. OpenShift Container Platform の認証情報を使用して RHACM CLI にログインします。
2. 追加するディスクを見つけます。追加するディスクは、既存のディスクのデバイス名およびパスと一致するようにする必要があります。
3. シングルノード OpenShift クラスターに容量を追加するには、既存のポリシー YAML の **deviceSelector** セクション (**policy-lvms-operator.yaml** など) を編集します。



注記

LVMCluster の作成中に **deviceSelector** フィールドが含まれていない場合、**deviceSelector** セクションを CR に追加することはできません。**LVMCluster** を削除してから、新しい CR から再作成する必要があります。

```
apiVersion: apps.open-cluster-management.io/v1
kind: PlacementRule
metadata:
  name: placement-install-lvms
spec:
  clusterConditions:
  - status: "True"
    type: ManagedClusterConditionAvailable
  clusterSelector:
    matchExpressions:
    - key: mykey
      operator: In
      values:
      - myvalue
---
apiVersion: policy.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
  name: binding-install-lvms
placementRef:
  apiGroup: apps.open-cluster-management.io
  kind: PlacementRule
  name: placement-install-lvms
subjects:
- apiGroup: policy.open-cluster-management.io
  kind: Policy
  name: install-lvms
---
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  annotations:
    policy.open-cluster-management.io/categories: CM Configuration Management
    policy.open-cluster-management.io/controls: CM-2 Baseline Configuration
    policy.open-cluster-management.io/standards: NIST SP 800-53
  name: install-lvms
spec:
  disabled: false
  remediationAction: enforce
  policy-templates:
  - objectDefinition:
      apiVersion: policy.open-cluster-management.io/v1
      kind: ConfigurationPolicy
      metadata:
        name: install-lvms
      spec:
        object-templates:
        - complianceType: musthave
          objectDefinition:
```

```
          apiVersion: v1
          kind: Namespace
          metadata:
            labels:
              openshift.io/cluster-monitoring: "true"
              pod-security.kubernetes.io/enforce: privileged
              pod-security.kubernetes.io/audit: privileged
              pod-security.kubernetes.io/warn: privileged
            name: openshift-storage
        - complianceType: musthave
          objectDefinition:
            apiVersion: operators.coreos.com/v1
            kind: OperatorGroup
            metadata:
              name: openshift-storage-operatorgroup
              namespace: openshift-storage
            spec:
              targetNamespaces:
              - openshift-storage
        - complianceType: musthave
          objectDefinition:
            apiVersion: operators.coreos.com/v1alpha1
            kind: Subscription
            metadata:
              name: lvms
              namespace: openshift-storage
            spec:
              installPlanApproval: Automatic
              name: lvms-operator
              source: redhat-operators
              sourceNamespace: openshift-marketplace
        remediationAction: enforce
        severity: low
    - objectDefinition:
        apiVersion: policy.open-cluster-management.io/v1
        kind: ConfigurationPolicy
        metadata:
          name: lvms
        spec:
          object-templates:
            - complianceType: musthave
              objectDefinition:
                apiVersion: lvm.topolvm.io/v1alpha1
                kind: LVMCluster
                metadata:
                  name: my-lvmcluster
                  namespace: openshift-storage
                spec:
                  storage:
                    deviceClasses:
                    - name: vg1
                      deviceSelector:
                        paths:
                        - /dev/disk/by-path/pci-0000:87:00.0-nvme-1
                        - /dev/disk/by-path/pci-0000:88:00.0-nvme-1
                        - /dev/disk/by-path/pci-0000:89:00.0-nvme-1 # new disk is added
```

```
            thinPoolConfig:
              name: thin-pool-1
              sizePercent: 90
              overprovisionRatio: 10
            nodeSelector:
              nodeSelectorTerms:
              - matchExpressions:
                  - key: app
                    operator: In
                    values:
                    - test1
      remediationAction: enforce
      severity: low
```

4. 以下のコマンドを実行してポリシーを編集します。

```
# oc edit -f policy-lvms-operator.yaml -ns lvms-policy-ns 1
```

1 **policy-lvms-operator.yaml** は既存のポリシーの名前です。

これは、**LVMCluster** CR で指定された新しいディスクを使用してストレージをプロビジョニングします。

関連情報

- Red Hat Advanced Cluster Management for Kubernetes: オンライン接続時のインストール
- LVM ストレージ参照 YAML ファイル

4.12.3.5.3. PVC の拡張

容量を追加した後、新しいストレージを活用するには、LVM Storage で既存の永続ボリューム要求 (PVC) を拡張できます。

前提条件

- 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

1. 次のコマンドを実行して、目的の PVC リソースの **.spec.resources.requests.storage** フィールドを新しいサイズに変更します。

```
oc patch <pvc_name> -n <application_namespace> -p '{ "spec": { "resources": { "requests": { "storage": "<desired_size>" }}}}'
```

2. PVC の **status.conditions** フィールドを監視し、サイズ変更が完了したかどうかを確認します。OpenShift Container Platform は、拡張中に **Resizing** 条件を PVC に追加します。これは、拡張の完了後、削除されます。

関連情報

- シングルノード OpenShift クラスターに容量を追加してストレージをスケールアップする
- RHACM を使用してシングルノード OpenShift クラスターに容量を追加してストレージをスケールアップする
- ボリューム拡張サポートの有効化

4.12.3.6. シングルノード OpenShift クラスターでの LVM ストレージのアップグレード

現在、シングルノード OpenShift クラスターで、OpenShift Data Foundation Logical Volume Manager Operator 4.11 から LVM Storage 4.12 にアップグレードすることはできません。

重要

このプロセス中、データは保持されません。

手順

1. 永続ボリューム要求 (PVC) で保持するデータをバックアップします。
2. OpenShift Data Foundation Logical Volume Manager Operator とその Pod によってプロビジョニングされたすべての PVC を削除します。
3. OpenShift Container Platform 4.12 に LVM ストレージを再インストールします。
4. ワークロードを再作成します。
5. 4.12 へのアップグレード後に作成された PVC にバックアップデータをコピーします。

4.12.3.7. シングルノード OpenShift のボリュームスナップショット

LVM ストレージによってプロビジョニングされた永続ボリューム (PV) のボリュームスナップショットを取得できます。クローン作成されたボリュームのボリュームスナップショットを作成することもできます。ボリュームスナップショットは、次のことを行うのに役立ちます。

- アプリケーションデータをバックアップします。

重要

ボリュームスナップショットは元のデータと同じデバイスにあります。ボリュームスナップショットをバックアップとして使用するには、スナップショットをセキュアな場所に移動する必要があります。OpenShift API をデータ保護のバックアップおよび復元ソリューションに使用できます。

- ボリュームスナップショットが作成された状態に戻します。

関連情報

- OADP の機能

4.12.3.7.1. シングルノード OpenShift でのボリュームスナップショットの作成

シンプールの使用可能な容量とオーバープロビジョニングの制限に基づいて、ボリュームスナップショットを作成できます。LVM Storage は、**lvms-<deviceclass-name>** という名前で **VolumeSnapshotClass** を作成します。

前提条件

- 永続ボリューム要求 (PVC) が **Bound** 状態であることを確認しました。これは、一貫性のあるスナップショットに必要です。
- スナップショットを作成する前に、PVC へのすべての I/O を停止しました。

手順

1. **oc** コマンドを実行する必要があるシングルノード OpenShift にログインします。
2. 次の YAML を **lvms-vol-snapshot.yaml** などの名前でファイルに保存します。

   ボリュームスナップショットを作成する YAML の例

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: lvm-block-1-snap
spec:
  volumeSnapshotClassName: lvms-vg1
  source:
    persistentVolumeClaimName: lvm-block-1
```

3. PVC と同じ namespace で次のコマンドを実行して、スナップショットを作成します。

```
# oc create -f lvms-vol-snapshot.yaml
```

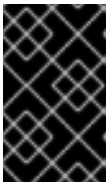
PVC の読み取り専用コピーがボリュームスナップショットとして作成されます。

4.12.3.7.2. シングルノード OpenShift でのボリュームスナップショットの復元

ボリュームスナップショットを復元すると、新しい永続ボリューム要求 (PVC) が作成されます。復元される PVC はボリュームスナップショットおよびソース PVC とは切り離されています。

前提条件

- ストレージクラスは、ソース PVC のストレージクラスと同じである必要がある。
- 要求された PVC のサイズは、スナップショットのソースボリュームのサイズと同じである必要がある。



重要

スナップショットは、スナップショットのソースボリュームと同じサイズの PVC に復元される必要があります。より大きな PVC が必要な場合は、スナップショットが正常に復元された後に PVC のサイズを変更できます。

手順

1. ソース PVC のストレージクラス名とボリュームスナップショット名を特定します。

2. 次の YAML を **lvms-vol-restore.yaml** などの名前でファイルに保存して、スナップショットを復元します。

   PVC を復元する YAML の例。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: lvm-block-1-restore
spec:
  accessModes:
  - ReadWriteOnce
  volumeMode: Block
  Resources:
    Requests:
      storage: 2Gi
  storageClassName: lvms-vg1
  dataSource:
    name: lvm-block-1-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

3. スナップショットと同じ namespace で次のコマンドを実行して、ポリシーを作成します。

```
# oc create -f lvms-vol-restore.yaml
```

4.12.3.7.3. シングルノード OpenShift でのボリュームスナップショットの削除

ボリュームスナップショットリソースと永続ボリューム要求 (PVC) を削除できます。

手順

1. 次のコマンドを実行して、ボリュームスナップショットリソースを削除します。

```
# oc delete volumesnapshot <volume_snapshot_name> -n <namespace>
```

注記

永続ボリューム要求 (PVC) を削除しても、PVC のスナップショットは削除されません。

2. 復元されたボリュームスナップショットを削除するには、次のコマンドを実行して、ボリュームスナップショットを復元するために作成された PVC を削除します。

```
# oc delete pvc <pvc_name> -n <namespace>
```

4.12.3.8. シングルノード OpenShift のボリュームのクローン作成

クローンは、既存のストレージボリュームの複製であり、他の標準ボリュームと同じように使用できます。

4.12.3.8.1. シングルノード OpenShift でのボリュームクローンの作成

ボリュームのクローンを作成して、データのポイントインタイムコピーを作成します。永続ボリューム要求は別のサイズでクローンできません。

重要

クローン作成された PVC には書き込みアクセス権限があります。

前提条件

- PVC が **Bound** 状態であることを確認しました。これは、一貫性のあるスナップショットに必要です。
- **StorageClass** がソース PVC のものと同じであることを確認しました。

手順

1. ソース PVC のストレージクラスを特定します。
2. ボリュームクローンを作成するには、次の YAML を **lvms-vol-clone.yaml** などの名前でファイルに保存します。

   ボリュームをクローン作成する YAML の例

```
apiVersion: v1
kind: PersistentVolumeClaim
Metadata:
  name: lvm-block-1-clone
Spec:
  storageClassName: lvms-vg1
  dataSource:
    name: lvm-block-1
    kind: PersistentVolumeClaim
  accessModes:
   - ReadWriteOnce
  volumeMode: Block
  Resources:
    Requests:
      storage: 2Gi
```

3. 次のコマンドを実行して、ソース PVC と同じ namespace にポリシーを作成します。

```
# oc create -f lvms-vol-clone.yaml
```

4.12.3.8.2. シングルノード OpenShift でのクローンボリュームの削除

クローン作成されたボリュームを削除できます。

手順

- クローン作成されたボリュームを削除するには、次のコマンドを実行して、クローン作成された PVC を削除します。

```
# oc delete pvc <clone_pvc_name> -n <namespace>
```

4.12.3.9. LVM Storage の監視

クラスターモニタリングを有効にするには、LVM Storage をインストールした namespace に次のラベルを追加する必要があります。

```
openshift.io/cluster-monitoring=true
```

重要

RHACM でクラスターモニタリングを有効化する方法の詳細は、可観測性 と カスタムメトリクスの追加 を参照してください。

4.12.3.9.1. メトリクス

メトリクスを表示することで、LVM Storage を監視できます。

次の表は、**topolvm** メトリクスについて説明しています。

表4.2 topolvm メトリクス

アラート	説明
topolvm_thinpool_data_percent	LVM シンプールで使用されているデータ領域の割合を示します。
topolvm_thinpool_metadata_percent	LVM シンプールで使用されているメタデータ領域の割合を示します。
topolvm_thinpool_size_bytes	LVM シンプールのサイズをバイト単位で示します。
topolvm_volumegroup_available_bytes	LVM ボリュームグループ内の利用可能な領域をバイト単位で示します。
topolvm_volumegroup_size_bytes	LVM ボリュームグループのサイズをバイト単位で示します。
topolvm_thinpool_overprovisioned_available	LVM シンプールの利用可能なオーバープロビジョニングサイズをバイト単位で示します。

注記

メトリクスは 10 分ごとに、または変更 (シンプールに新しい論理ボリュームが作成されるなど) があったときに更新されます。

4.12.3.9.2. アラート

シンプールとボリュームグループが最大ストレージ容量に達すると、それ以降の操作は失敗します。これにより、データ損失が発生する可能性があります。

LVM Storage は、シンプールとボリュームグループの使用量が特定の値を超えると、次のアラートを送信します。

表4.3 LVM Storage アラート

アラート	説明
VolumeGroupUsageAtThresholdNearFull	このアラートは、ボリュームグループとシンプールの両方の使用量がノード上で 75% を超えるとトリガーされます。データの削除またはボリュームグループの拡張が必要です。
VolumeGroupUsageAtThresholdCritical	このアラートは、ボリュームグループとシンプールの両方の使用量がノード上で 85% を超えるとトリガーされます。この場合、ボリュームグループは、かなりいっぱいになっています。データの削除またはボリュームグループの拡張が必要です。
ThinPoolDataUsageAtThresholdNearFull	このアラートは、ボリュームグループ内のシンプールのデータ使用量がノード上で 75% を超えるとトリガーされます。データの削除またはシンプールの拡張が必要です。
ThinPoolDataUsageAtThresholdCritical	このアラートは、ボリュームグループ内のシンプールのデータ使用量がノード上で 85% を超えるとトリガーされます。データの削除またはシンプールの拡張が必要です。
ThinPoolMetaDataUsageAtThresholdNearFull	このアラートは、ボリュームグループ内のシンプールのメタデータ使用量がノード上で 75% を超えるとトリガーされます。データの削除またはシンプールの拡張が必要です。
ThinPoolMetaDataUsageAtThresholdCritical	このアラートは、ボリュームグループ内のシンプールのメタデータ使用量がノード上で 85% を超えるとトリガーされます。データの削除またはシンプールの拡張が必要です。

4.12.3.10. must-gather を使用したログファイルおよび診断情報のダウンロード

LVM Storage が問題を自動的に解決できない場合、must-gather ツールを使用してログファイルと診断情報を収集し、ユーザーまたは Red Hat サポートが問題を確認して解決策を決定できるようにします。

手順

- LVM Storage クラスターに接続されているクライアントから **must-gather** コマンドを実行します。

```
$ oc adm must-gather --image=registry.redhat.io/lvms4/lvms-must-gather-rhel9:v4.12 --dest-dir=<directory_name>
```

関連情報

- must-gather ツールについて

4.12.3.11. LVM ストレージ参照 YAML ファイル

サンプルの **LVMCluster** カスタムリソース (CR) では、YAML ファイルのすべてのフィールドについて説明しています。

LVMCluster CR の例

```
apiVersion: lvm.topolvm.io/v1alpha1
kind: LVMCluster
metadata:
  name: my-lvmcluster
spec:
  tolerations:
  - effect: NoSchedule
    key: xyz
    operator: Equal
    value: "true"
  storage:
    deviceClasses: 1
    - name: vg1 2
      nodeSelector: 3
        nodeSelectorTerms: 4
        - matchExpressions:
          - key: mykey
            operator: In
            values:
            - ssd
      deviceSelector: 5
        paths:
        - /dev/disk/by-path/pci-0000:87:00.0-nvme-1
        - /dev/disk/by-path/pci-0000:88:00.0-nvme-1
        - /dev/disk/by-path/pci-0000:89:00.0-nvme-1
      thinPoolConfig: 6
        name: thin-pool-1 7
        sizePercent: 90 8
        overprovisionRatio: 10 9
status:
    deviceClassStatuses: 10
    - name: vg1
      nodeStatus: 11
      - devices: 12
        - /dev/nvme0n1
        - /dev/nvme1n1
        - /dev/nvme2n1
        node: my-node.example.com 13
        status: Ready 14
    ready: true 15
    state: Ready 16
```

1 クラスター上に作成される LVM ボリュームグループ。**deviceClass** を 1 つだけ作成できます。

2 ノード上に作成される LVM ボリュームグループの名前。

3 LVM ボリュームグループを作成するノード。フィールドが空の場合、すべてのノードが考慮されます。

4 ノードセレクター要件のリスト。

5 LVM ボリュームグループの作成に使用されるデバイスパスのリスト。このフィールドが空の場合、ノード上のすべての未使用ディスクが使用されます。デバイスが LVM ボリュームグループに追加された後は、デバイスを削除できません。

6 LVM シンプールの設定。

7 LVM ボリュームグループに作成されるシンプールの名前。

8 シンプールの作成に使用する LVM ボリュームグループの残りの領域の割合。

9 シンプールで使用可能なストレージと比較して、追加のストレージをプロビジョニングできる係数。

10 **deviceClass** のステータス。

11 各ノードの LVM ボリュームグループのステータス。

12 LVM ボリュームグループの作成に使用されるデバイスのリスト。

13 **deviceClass** が作成されたノード。

14 ノード上の LVM ボリュームグループのステータス。

15 このフィールドは非推奨です。

16 **LVMCluster** のステータス。

4.12.4. LVMS を使用したローカル永続ストレージのトラブルシューティング

OpenShift Container Platform は永続ボリューム (PV) の範囲を単一のプロジェクトに限定しないため、クラスター全体で共有し、永続ボリューム要求 (PVC) を使用して任意のプロジェクトで要求することができます。これにより、トラブルシューティングが必要な多くの問題が発生する可能性があります。

4.12.4.1. 保留状態でスタックしている PVC を調査する

Persistent Volume Claim (PVC) は、さまざまな理由で **Pending** 状態になることがあります。以下に例を示します。

- コンピューティングリソースが不十分
- ネットワークの問題
- ストレージクラスまたはノードセレクターが一致していない
- 利用可能なボリュームがない
- 永続ボリューム (PV) を持つノードは **Not Ready** 状態です

oc description コマンドを使用してスタック PVC の詳細を確認し、原因を特定します。

手順

1. 次のコマンドを実行して、PVC のリストを取得します。

```
$ oc get pvc
```

出力例

```
NAME       STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   AGE
lvms-test  Pending                                      lvms-vg1       11s
```

2. 次のコマンドを実行して、**Pending** 状態のままになっている PVC に関連するイベントを検査します。

```
$ oc describe pvc <pvc_name> 1
```

1 **<pvc_name>** を PVC の名前に置き換えます。たとえば、**lvms-vg1** です。

出力例

```
Type     Reason              Age              From                         Message
----     ------              ----             ----                         -------
Warning  ProvisioningFailed  4s (x2 over 17s)  persistentvolume-controller
storageclass.storage.k8s.io "lvms-vg1" not found
```

4.12.4.2. 不足している LVMS または Operator コンポーネントからの回復

ストレージクラスが「見つからない」エラーが発生した場合は、**LVMCluster** リソースをチェックし、すべての論理ボリュームマネージャーストレージ (LVMS) Pod が実行していることを確認してください。**LVMCluster** リソースが存在しない場合は作成できます。

手順

1. 次のコマンドを実行して、LVMCluster リソースの存在を確認します。

```
$ oc get lvmcluster -n openshift-storage
```

出力例

```
NAME           AGE
my-lvmcluster  65m
```

2. クラスターに **LVMCluster** リソースがない場合は、次のコマンドを実行して LVMCluster リソースを作成します。

```
$ oc create -n openshift-storage -f <custom_resource> 1
```

1 **<custom_resource>** を、要件に合わせたカスタムリソース URL またはファイルに置き換えます。

カスタムリソースの例

```
apiVersion: lvm.topolvm.io/v1alpha1
kind: LVMCluster
metadata:
  name: my-lvmcluster
spec:
  storage:
    deviceClasses:
    - name: vg1
      default: true
      thinPoolConfig:
        name: thin-pool-1
        sizePercent: 90
        overprovisionRatio: 10
```

3. 次のコマンドを実行して、LVMS のすべての Pod が **openshift-storage** namespace で **Running** 状態であることを確認します。

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

出力例

```
NAME                                READY   STATUS    RESTARTS      AGE
lvms-operator-7b9fb858cb-6nsml      3/3     Running   0             70m
topolvm-controller-5dd9cf78b5-7wwr2   5/5     Running   0           66m
topolvm-node-dr26h                  4/4     Running   0             66m
vg-manager-r6zdv                    1/1     Running   0             66m
```

予期される出力は、**lvms-operator** および **vg-manager** の 1 つの実行インスタンスです。各ノードには、**topolvm-controller** および **topolvm-node** のインスタンスが 1 つ必要です。

topolvm-node が **Init** 状態でスタックしている場合は、LVMS が使用できるディスクを見つけることができません。トラブルシューティングに必要な情報を取得するには、次のコマンドを実行して **vg-manager** Pod のログを確認します。

```
$ oc logs -l app.kubernetes.io/component=vg-manager -n openshift-storage
```

4.12.4.3. ノード障害からの回復

クラスター内の特定のノードに障害が発生したために、永続ボリューム要求 (PVC) が **Pending** 状態のままになることがあります。障害が発生したノードを特定するには、**topolvm-node** Pod の再起動回数を調べることができます。再起動回数の増加は、基礎となるノードに潜在的な問題があることを示しており、さらなる調査とトラブルシューティングが必要になる場合があります。

手順

- 次のコマンドを実行して、**topolvm-node** Pod インスタンスの再起動回数を調べます。

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

出力例

```
NAME                                READY   STATUS    RESTARTS      AGE
lvms-operator-7b9fb858cb-6nsml      3/3     Running   0             70m
topolvm-controller-5dd9cf78b5-7wwr2 5/5     Running   0             66m
topolvm-node-dr26h                  4/4     Running   0             66m
topolvm-node-54as8                  4/4     Running   0             66m
topolvm-node-78fft                  4/4     Running   17 (8s ago)   66m
vg-manager-r6zdv                    1/1     Running   0             66m
vg-manager-990ut                    1/1     Running   0             66m
vg-manager-an118                    1/1     Running   0             66m
```

ノードの問題を解決した後、PVC がまだ **Pending** 状態のままである場合は、強制クリーンアップ手順を実行する必要がある場合があります。

関連情報

- 強制クリーンアップの実行

4.12.4.4. ディスク障害からの回復

Persistent Volume Claim (PVC) に関連するイベントの検査中にエラーメッセージが表示された場合は、基になるボリュームまたはディスクに問題がある可能性があります。ディスクとボリュームのプロビジョニングの問題は、多くの場合、最初に **Failed to provision volume with StorageClass <storage_class_name>** などの一般的なエラーが発生します。通常、2 番目のより具体的なエラーメッセージが続きます。

手順

1. 次のコマンドを実行して、PVC に関連付けられたイベントを検査します。

```
$ oc describe pvc <pvc_name> 1
```

1 **<pvc_name>** を PVC の名前に置き換えます。ディスクまたはボリューム障害のエラーメッセージとその原因の例をいくつか示します。

   - **ボリュームの存在を確認できない:** ボリュームがすでに存在するかどうかの確認で問題が発生したことを示します。ボリューム検証の失敗は、ネットワーク接続の問題やその他の障害によって発生する可能性があります。
   - **バインドボリュームへの失敗:** 利用可能な永続ボリューム (PV) が PVC の要件と一致しない場合、ボリュームのバインドに失敗する可能性があります。
   - **FailedMount または FailedUnMount:** このエラーは、ボリュームをノードにマウントしようとしたとき、またはノードからボリュームをアンマウントしようとしたときの問題を示します。ディスクに障害が発生した場合、Pod が PVC を使用しようとしたときにこのエラーが表示されることがあります。
   - **ボリュームはすでに 1 つのノードに排他的に接続されており、別のノードには接続できない:** このエラーは、**ReadWriteMany** アクセスモードをサポートしていないストレージソリューションで発生する可能性があります。

2. 問題が発生しているホストへの直接接続を確立します。
3. ディスクの問題を解決します。

ディスクの問題を解決した後、エラーメッセージが続くか再発する場合は、強制クリーンアップ手順の実行が必要になる場合があります。

関連情報

- 強制クリーンアップの実行

4.12.4.5. 強制クリーンアップの実行

トラブルシューティング手順を完了した後もディスクまたはノード関連の問題が解決しない場合は、強制クリーンアップ手順の実行が必要になる場合があります。強制クリーンアップは、永続的な問題に包括的に対処し、LVMS が適切に機能することを保証するために使用されます。

前提条件

1. 論理ボリュームマネージャーストレージ (LVMS) ドライバーを使用して作成された永続ボリュームクレーム (PVC) はすべて削除されました。
2. これらの PVC を使用する Pod は停止されました。

手順

1. 次のコマンドを実行して、**openshift-storage** namespace に切り替えます。

```
$ oc project openshift-storage
```

2. 次のコマンドを実行して、**Logical Volume** のカスタムリソース (CR) が残っていないことを確認します。

```
$ oc get logicalvolume
```

出力例

```
No resources found
```

   a. **LogicalVolume** CR が残っている場合は、次のコマンドを実行してファイナライザーを削除します。

```
$ oc patch logicalvolume <name> -p '{"metadata":{"finalizers":[]}}' --type=merge (1)
```

   (1) **<name>** を CR の名前に置き換えます。

   b. ファイナライザーを削除した後、次のコマンドを実行して CR を削除します。

```
$ oc delete logicalvolume <name> (1)
```

   (1) **<name>** を CR の名前に置き換えます。

3. 次のコマンドを実行して、**LVMVolumeGroup** CR が残っていないことを確認します。

```
$ oc get lvmvolumegroup
```

出力例

```
No resources found
```

a. **LVMVolumeGroup** CR が残っている場合は、次のコマンドを実行してファイナライザーを削除します。

```
$ oc patch lvmvolumegroup <name> -p '{"metadata":{"finalizers":[]}}' --type=merge (1)
```

(1) **<name>** を CR の名前に置き換えます。

b. ファイナライザーを削除した後、次のコマンドを実行して CR を削除します。

```
$ oc delete lvmvolumegroup <name> (1)
```

(1) **<name>** を CR の名前に置き換えます。

4. 次のコマンドを実行して、**LVMVolumeGroupNodeStatus** CR を削除します。

```
$ oc delete lvmvolumegroupnodestatus --all
```

5. 次のコマンドを実行して、**LVMCluster** CR を削除します。

```
$ oc delete lvmcluster --all
```

第5章 CONTAINER STORAGE INTERFACE (CSI) の使用

5.1. 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。

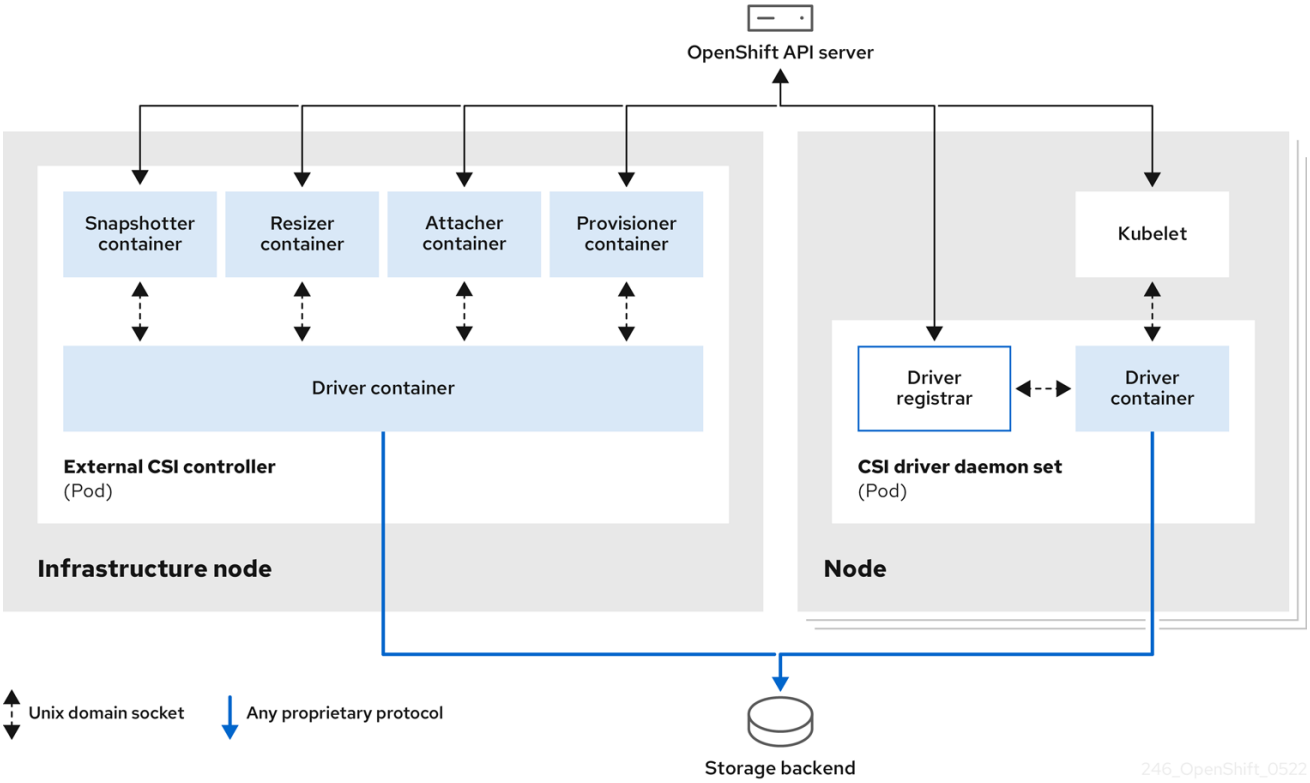
注記

OpenShift Container Platform 4.12 は、CSI 仕様 のバージョン 1.6.0 をサポートします。

5.1.1. 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。

5.1.1.1. 外部の CSI コントローラー

外部の CSI コントローラーは、5 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。

- スナップショットコンテナーは、**VolumeSnapshot** オブジェクトおよび **VolumeSnapshotContent** オブジェクトを監視し、**VolumeSnapshotContent** オブジェクトの作成および削除を担当します。
- リサイザーコンテナーは、**PersistentVolumeClaim** オブジェクトでより多くのストレージを要求した場合に、**PersistentVolumeClaim** の更新を監視し、CSI エンドポイントに対して **ControllerExpandVolume** 操作をトリガーするサイドカーコンテナーです。
- 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。

5.1.1.2. 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。

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

OpenShift Container Platform はデフォルトで特定の CSI ドライバーをインストールし、In-tree(インツリー) ボリュームプラグインでは不可能なユーザーストレージオプションを提供します。

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、OpenShift Container Platform は必要な CSI Driver Operator、CSI ドライバー、および必要なストレージクラスをインストールします。Operator およびドライバーのデフォルト namespace 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 と共にインストールされる 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5.1 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
AliCloud Disk	▮	-	▮
AWS EBS	▮	-	▮
AWS EFS	-	-	-
Google Compute Platform (GCP) persistent disk (PD)	▮	▮	▮
GCP Filestore	▮	-	▮
IBM VPC Block	▮[3]	-	▮[3]
LVM Storage	▮	▮	▮
Microsoft Azure Disk	▮	▮	▮
Microsoft Azure Stack Hub	▮	▮	▮
Microsoft Azure File	-	-	▮
OpenStack Cinder	▮	▮	▮

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
OpenShift Data Foundation	▮	▮	▮
OpenStack Manila	▮	-	-
Red Hat Virtualization (oVirt)	-	-	▮
VMware vSphere	▮[1]	-	▮[2]

1.

- vCenter Server と ESXi の両方に、vSphere バージョン 7.0 Update 3 以降が必要です。
- ファイル共有ボリュームはサポートされません。

2.

- オフラインボリューム拡張: 必要な vSphere の最低バージョンは 6.7 Update 3 P06 です。
- オンラインボリューム拡張: 必要な vSphere の最低バージョンは 7.0 Update 2 です。

3.

- オフラインスナップショットまたはサイズ変更はサポートされていません。ボリュームは、実行中の Pod にアタッチする必要があります。

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。

5.1.3. 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。

5.1.4. 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
```

5.1.5. ボリュームポピュレーター

ボリュームポピュレーターは、永続ボリュームクレーム (PVC) 仕様の **datasource** フィールドを使用して、事前に入力されたボリュームを作成します。

ボリュームの作成は現在有効になっており、テクノロジープレビュー機能としてサポートされています。ただし、OpenShift Container Platform にはボリュームポピュレーターは同梱されていません。

重要

ボリュームポピュレーターはテクノロジープレビュー機能のみです。テクノロジープレビュー機能は、Red Hat 製品サポート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場合があります。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。

ボリュームポピュレーターの詳細は、Kubernetes ボリュームポピュレーター を参照してください。

5.2. 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。

5.2.1. 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。

5.2.1.1. 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。
- Shared Resource CSI Driver は、テクノロジープレビュー機能としてインライン一時ボリュームをサポートします。

- コミュニティーまたはストレージベンダーは、これらのボリュームをサポートする他の CSI ドライバーを提供します。CSI ドライバーのプロバイダーが提供す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。

重要

Shared Resource CSI Driver は、テクノロジープレビュー機能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 製品サポート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場合があります。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。

5.2.2. 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
```

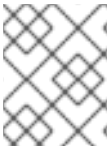
5.3. SHARED RESOURCE CSI DRIVER OPERATOR

クラスター管理者は、OpenShift Container Platform で Shared Resource CSI Driver を使用して、**Secret** または **ConfigMap** オブジェクトの内容が含まれるインライン一時ボリュームをプロビジョニングできます。これにより、ボリュームマウントを公開する Pod および他の Kubernetes タイプ、ならびに OpenShift Container Platform Builds は、クラスター内の namespace 全体でそれらのオブジェクトの内容を安全に使用できます。そのために、現在、**SharedSecret** カスタムリソース (**Secret** オブジェクト用) および **SharedConfigMap** カスタムリソース **ConfigMap** オブジェクト用) という 2 種類の共有リソースがあります。

重要

Shared Resource CSI Driver は、テクノロジープレビュー機能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 製品サポート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場合があります。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。



注記

Shared Resource CSI Driver を有効にするには、フィーチャーゲートを使用して機能を有効化 する必要があります。

5.3.1. 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。

5.3.2. namespace 間でのシークレットの共有

クラスター内の namespace 間でシークレットを共有するには、共有する **Secret** オブジェクトの **SharedSecret** カスタムリソース (CR) インスタンスを作成します。

前提条件

次のアクションを実行するためのパーミッションがある。

- cluster スコープレベルで **sharedsecrets.sharedresource.openshift.io** カスタムリソース定義 (CRD) のインスタンスを作成する。
- クラスター内の namespace 全体でロールおよびロールバインディングを管理し、これらのインスタンスを取得、リスト表示、監視できるユーザーを制御する。

- ロールおよびロールバインディングを管理し、Pod で指定されたサービスアカウントが、使用する **SharedSecret** CR インスタンスを参照する Container Storage Interface (CSI) ボリュームをマウントできるかどうかを制御する。
- 共有する Secret が含まれる namespace にアクセスする。

手順

- クラスター内の namespace 間で共有する **Secret** オブジェクトの **SharedSecret** CR インスタンスを作成します。

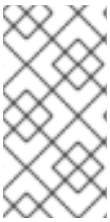
```
$ oc apply -f - <<EOF
apiVersion: sharedresource.openshift.io/v1alpha1
kind: SharedSecret
metadata:
  name: my-share
spec:
  secretRef:
    name: <name of secret>
    namespace: <namespace of secret>
EOF
```

5.3.3. Pod での SharedSecret インスタンスの使用

Pod から **SharedSecret** カスタムリソース (CR) インスタンスにアクセスするには、その **SharedSecret** CR インスタンスを使用するための、特定のサービスアカウント RBAC パーミッションを付与します。

前提条件

- クラスター内の namespace 間で共有する Secret の **SharedSecret** CR インスタンスを作成している。
- 次のアクションを実行するためのパーミッションがある。
  - ビルド設定を作成し、ビルドを開始します。
  - **oc get sharedsecrets**コマンドを入力し、空でないリストを取得して、使用可能な**SharedSecret** CR インスタンスを見つけます。
  - namespace で利用可能な **builder** サービスアカウントが指定の **SharedSecret** CR インスタンスを使用できるかどうかを判別します。つまり、**oc adm policy who-can use <identifier of specific SharedSecret>** 実行して、namespace の **builder** サービスアカウントが一覧に表示されるかどうかを確認できます。



注記

このリストの最後の 2 つの前提条件のいずれも満たされない場合は、**SharedSecret** CR インスタンスを検出し、サービスアカウントを有効にして **SharedSecret** CR インスタンスを使用できるように、必要なロールベースアクセス制御 (RBAC) を作成するか、誰かに依頼して作成します。

手順

1. YAML コンテンツと共に **oc apply** を使用して、その Pod で **SharedSecret** CR インスタンスを使用するための、特定のサービスアカウント RBAC パーミッションを付与します。

注記

現在、**kubectl** と **oc** には、**use** 動詞を Pod セキュリティーを中心としたロールに制限する特別な場合のロジックがハードコーディングされています。したがって、**oc create role …** を使用して、**SharedSecret** CR インスタンスの使用に必要なロールを作成することはできません。

```
$ oc apply -f - <<EOF
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: shared-resource-my-share
  namespace: my-namespace
rules:
  - apiGroups:
      - sharedresource.openshift.io
    resources:
      - sharedsecrets
    resourceNames:
      - my-share
    verbs:
      - use
EOF
```

2. **oc** コマンドを使用して、ロールに関連付けられた **RoleBinding** を作成します。

```
$ oc create rolebinding shared-resource-my-share --role=shared-resource-my-share --
serviceaccount=my-namespace:builder
```

3. Pod から **SharedSecret** CR インスタンスにアクセスします。

```
$ oc apply -f - <<EOF
kind: Pod
apiVersion: v1
metadata:
  name: my-app
  namespace: my-namespace
spec:
  serviceAccountName: default

# containers omitted …. Follow standard use of ‘volumeMounts’ for referencing your shared
resource volume

    volumes:
    - name: my-csi-volume
      csi:
        readOnly: true
        driver: csi.sharedresource.openshift.io
        volumeAttributes:
```

```
        sharedSecret: my-share

EOF
```

5.3.4. namespace 間での設定マップの共有

クラスター内の namespace 間で設定マップを共有するには、その設定マップの **SharedConfigMap** カスタムリソース (CR) インスタンスを作成します。

前提条件

次のアクションを実行するためのパーミッションがある。

- cluster スコープレベルで **sharedconfigmaps.sharedresource.openshift.io** カスタムリソース定義 (CRD) のインスタンスを作成する。
- クラスター内の namespace 全体でロールおよびロールバインディングを管理し、これらのインスタンスを取得、リスト表示、監視できるユーザーを制御する。
- クラスター内の namespace 全体でロールおよびロールバインディングを管理し、Container Storage Interface (CSI) ボリュームをマウントする Pod のどのサービスアカウントが、それらのインスタンスを使用できるかを制御する。
- 共有する Secret が含まれる namespace にアクセスする。

手順

1. クラスター内の namespace 間で共有する設定マップの **SharedConfigMap** CR インスタンスを作成します。

```
$ oc apply -f - <<EOF
apiVersion: sharedresource.openshift.io/v1alpha1
kind: SharedConfigMap
metadata:
  name: my-share
spec:
  configMapRef:
    name: <name of configmap>
    namespace: <namespace of configmap>
EOF
```

5.3.5. Pod での SharedConfigMap インスタンスの使用

次のステップ

Pod から **SharedConfigMap** カスタムリソース (CR) インスタンスにアクセスするには、その **SharedConfigMap** CR インスタンスを使用するための、特定のサービスアカウント RBAC パーミッションを付与します。

前提条件

- クラスター内の namespace 間で共有する設定マップの **SharedConfigMap** CR インスタンスを作成している。
- 次のアクションを実行するためのパーミッションがある。

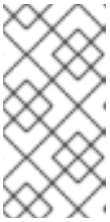
- ビルド設定を作成し、ビルドを開始します。
- **oc get sharedconfigmaps**コマンドを入力し、空でないリストを取得して、使用可能な**SharedConfigMap** CR インスタンスを検出します。
- namespace で利用可能な **builder** サービスアカウントが指定の **SharedSecret** CR インスタンスを使用できるかどうかを判別します。つまり、**oc adm policy who-can use <identifier of specific SharedSecret>** 実行して、namespace の **builder** サービスアカウントが一覧に表示されるかどうかを確認できます。

注記

このリストの最後の 2 つの前提条件のいずれも満たされない場合は、**SharedConfigMap** CR インスタンスを検出し、サービスアカウントを有効にして **SharedConfigMap** CR インスタンスを使用できるように、必要なロールベースアクセス制御 (RBAC) を作成するか、誰かに依頼して作成します。

手順

1. YAML コンテンツと共に **oc apply** を使用して、その Pod で **SharedConfigMap** CR インスタンスを使用するための、特定のサービスアカウント RBAC パーミッションを付与します。



注記

現在、**kubectl** と **oc** には、**use** 動詞を Pod セキュリティーを中心としたロールに制限する特別な場合のロジックがハードコーディングされています。したがって、**oc create role ...** を使用して、**SharedConfigMap** CR インスタンスの使用に必要なロールを作成することはできません。

```
$ oc apply -f - <<EOF
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: shared-resource-my-share
  namespace: my-namespace
rules:
  - apiGroups:
      - sharedresource.openshift.io
    resources:
      - sharedconfigmaps
    resourceNames:
      - my-share
    verbs:
      - use
EOF
```

2. **oc** コマンドを使用して、ロールに関連付けられた **RoleBinding** を作成します。

```
oc create rolebinding shared-resource-my-share --role=shared-resource-my-share --
serviceaccount=my-namespace:builder
```

3. Pod から **SharedConfigMap** CR インスタンスにアクセスします。

```
$ oc apply -f - <<EOF
```

```
kind: Pod
apiVersion: v1
metadata:
  name: my-app
  namespace: my-namespace
spec:
  serviceAccountName: default

# containers omitted …. Follow standard use of ‘volumeMounts’ for referencing your shared
resource volume

    volumes:
    - name: my-csi-volume
      csi:
        readOnly: true
        driver: csi.sharedresource.openshift.io
        volumeAttributes:
          sharedConfigMap: my-share

EOF
```

5.3.6. Shared Resource CSI Driver に対するその他のサポート制限

Shared Resource CSI Driver には、以下の重要な制限があります。

- ドライバーは、Container Storage Interface (CSI) のインライン一時ボリュームの制限の対象となります。
- **readOnly** フィールドの値は **true** である必要があります。そうでない場合には、Pod 起動時のボリュームのプロビジョニング時に、ドライバーはエラーを kubelet に返します。この制限は、関連するボリュームへの SELinux ラベルの適用に関して、アップストリームの Kubernetes CSI Driver で提案されたベストプラクティスに対応するためのものです。
- ドライバーは、**tmpfs** ボリュームしかサポートしないため、**FSType** フィールドを無視します。
- ドライバーは **NodePublishSecretRef** フィールドを無視します。代わりに、ドライバーは**use**動詞で **SubjectAccessReviews** を使用して、Pod が **SharedSecret** または **SharedConfigMap** カスタムリソース (CR) インスタンスが含まれるボリュームを取得できるかどうかを評価します。

5.3.7. 共有リソース Pod ボリュームの VolumeAttributes に関する追加情報

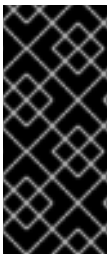
以下の属性は、さまざまな形で共有リソース Pod ボリュームに影響を及ぼします。

- **volumeAttributes** プロパティーの **refreshResource** 属性。
- Shared Resource CSI Driver 設定の **refreshResources** 属性。
- **volumeAttributes** プロパティーの **sharedSecret** および **sharedConfigMap** 属性。

5.3.7.1. refreshResource 属性

Shared Resource CSI Driver は、ボリュームの **volumeAttributes** プロパティーの **refreshResource** 属性を尊重します。この属性は、Pod 起動の一環としてボリュームが初回にプロビジョニングされた**後**

に、基礎となる **Secret** または **ConfigMap** オブジェクトの内容に対する更新がボリュームにコピーされるかどうかを制御します。**refreshResource** のデフォルト値は **true** です。これは、コンテンツが更新されることを意味します。



重要

Shared Resource CSI Driver 設定により、共有 **SharedSecret** および **SharedConfigMap** カスタムリソース (CR) インスタンス両方の更新が無効にされている場合、**volumeAttribute** プロパティーの **refreshResource** 属性は影響を及ぼしません。この属性の目的は、更新が一般的に許可されている場合に、特定のボリュームマウントの更新を無効にすることです。

5.3.7.2. refreshResources 属性

グローバルスイッチを使用して、共有リソースの更新を有効または無効にできます。このスイッチは Shared Resource CSI Driver の **csi-driver-shared-resource-config** 設定マップの **refreshResources** 属性で、**openshift-cluster-csi-drivers** namespace で確認できます。この **refreshResources** 属性を **false** に設定する場合、ボリュームに保存されている **Secret** または **ConfigMap** オブジェクト関連のコンテンツは、ボリュームの初回プロビジョニング後に更新されません。

重要

この Shared Resource CSI Driver 設定を使用して更新を無効にすると、ボリュームの **volumeAttributes** プロパティーの **refreshResource** 属性に関係なく、Shared Resource CSI Driver を使用するすべてのクラスターのボリュームマウントが影響を受けます。

5.3.7.3. Pod の共有リソースボリュームをプロビジョニングする前の volumeAttributes の検証

1 つのボリュームの **volumeAttributes** で、**sharedSecret** または **sharedConfigMap** 属性いずれかの値を、**SharedSecret** または **SharedConfigMap** CS インスタンスの値に設定する必要があります。設定しないと、Pod の起動時にボリュームがプロビジョニングされる際に、検証でそのボリュームの **volumeAttributes** がチェックされ、以下の条件下では kubelet にエラーが返されます。

- **sharedSecret** および **sharedConfigMap** 属性の両方に値が指定されている。
- **sharedSecret** および **sharedConfigMap** 属性の両方に値が指定されていない。
- **sharedSecret** または **sharedConfigMap** 属性の値が、クラスター内の **SharedSecret** または **SharedConfigMap** CR インスタンスの名前に対応していない。

5.3.8. 共有リソース、Insights Operator、および OpenShift Container Platform Builds 間のインテグレーション

共有リソース、Insights Operator、および OpenShift Container Platform Builds 間のインテグレーションにより、Red Hat サブスクリプション (RHEL エンタイトルメント) を OpenShift Container Platform Builds で簡単に使用できます。

従来、OpenShift Container Platform 4.9.x 以前では、認証情報を手動でインポートし、ビルドを実行している各プロジェクトまたは namespace にコピーしていました。

OpenShift Container Platform 4.10 以降では、OpenShift Container Platform Builds では、共有リソースおよび Insights Operator によって提供される単純なコンテンツアクセス機能を参照して、Red Hat サブスクリプション (RHEL エンタイトルメント) を使用できるようになりました。

- シンプルなコンテンツアクセス機能は、サブスクリプションの認証情報を、既知の **Secret** オブジェクトにインポートします。以下の「関連情報」セクションのリンクを参照してください。
- クラスター管理者は、その **Secret** オブジェクトに関する **SharedSecret** カスタムリソース (CR) インスタンスを作成し、特定のプロジェクトまたは namespace にパーミッションを付与します。特に、クラスター管理者は、その **SharedSecret** CR インスタンスを使用するための、**builder** サービスアカウントパーミッションを付与します。
- それらのプロジェクトまたは namespace 内で実行されるビルドは、**SharedSecret** CR インスタンスおよびそのエンタイトルメントが適用された RHEL コンテンツを参照する CSI Volume をマウントできます。

関連情報

- Insights Operator を使用した単純なコンテンツアクセス証明書のインポート
- ビルドシークレットとしてのサブスクリプションエンタイトルメントの追加

5.4. 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。

5.4.1. 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 Container Storage Interface (CSI) ボリュームス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 ドライバーのプロバイダーが提供す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。

5.4.2. 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。

5.4.2.1. 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。

5.4.2.2. 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。

5.4.3. 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。

5.4.3.1. 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/v1** 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** オブジェクト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。

5.4.4. 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。

5.4.4.1. 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。

5.4.4.2. 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。

5.4.5. 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。

警告

Pod によって使用されている PVC のボリュームスナップショットを作成すると、書き込まれていないデータやキャッシュされたデータがスナップショットから除外される可能性があります。すべてのデータがディスクに確実に書き込まれるようにするには、PVC を使用している Pod を削除してから、スナップショットを作成してください。

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** オブジェクトを使用してファイルを作成します。

   volumesnapshotclass.yaml

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: csi-hostpath-snap
driver: hostpath.csi.k8s.io (1)
deletionPolicy: Delete
```

   (1) この **VolumeSnapshotClass** オブジェクトのスナップショットを作成するために使用される CSI ドライバーの名前。名前は、スナップショットが作成される PVC に対応するストレージクラスの **Provisioner** フィールドと同じである必要があります。

注記

永続ストレージの設定に使用したドライバーによっては、追加のパラメーターが必要になる場合があります。既存の **VolumeSnapshotClass** オブジェクトを使用することもできます。

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/v1
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** 設定がなく、デフォルトのボリュームスナップショットクラスがある場合、スナップショットはデフォルトのボリュームスナップショットクラス名で作成されます。ただし、フィールドがなく、デフォルトのボリュームスナップショットクラスが存在しない場合には、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ビジョニングに必要です。

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/v1
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/v1
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。

5.4.6. 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。

   volumesnapshotclass.yaml

   ```
   apiVersion: snapshot.storage.k8s.io/v1
   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。

5.4.7. 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。* ボリュームスナップショットが作成され、使用できる状態である。

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
```

myclaim-restore などの新規 PVC が表示されます。

5.5. 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。

5.5.1. 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。

5.5.1.1. 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。
- クローン作成は、別のストレージクラスでサポートされています。
  - 宛先ボリュームは、ソースと異なるストレージクラスでも同じにすることができます。
  - デフォルトのストレージクラスを使用し、**spec** で **storageClassName** を省略できます。
- 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。
- 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。

5.5.2. 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** オブジェクトを使用し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。

5.6. CSI の自動移行

従来 OpenShift Container Platform に同梱されていた in-tree のストレージドライバーは非推奨となり、同等の Container Storage Interface (CSI) ドライバーに置き換えられます。OpenShift Container Platform では、サポートされている特定の in-tree ボリュームプラグインが同等の CSI ドライバーに自動移行されます。

5.6.1. 概要

in-tree(インツリー) ストレージプラグインを使用してプロビジョニングされ、この機能でサポートされるボリュームは、対応する Container Storage Interface (CSI) ドライバーに移行されます。このプロセスはデータ移行を実行しません。OpenShift Container Platform は、メモリー内の永続ボリュームオブジェクトしか変換しません。その結果、変換された永続ボリュームオブジェクトはディスクに保存されず、その内容も変更されません。

以下の in-tree(インツリー) から CSI ドライバーへの移行がサポートされます。

表5.2 in-tree/CSI ドライバーでサポートされる CSI 自動移行機能

in-tree/CSI ドライバー	サポートレベル	CSI の自動移行が自動的に有効になるか？
• Azure Disk • OpenStack Cinder • Amazon Web Services (AWS) Elastic Block Storage (EBS) • Google Compute Engine Persistent Disk (GCP PD)	Generally available (GA)	Yes. For more information, see **Automatic migration of in-tree volumes to CSI**.

in-tree/CSI ドライバー	サポートレベル	CSI の自動移行が自動的に有効になるか？
• Azure File • VMware vSphere	Technology Preview (TP)	無効。有効にするには、**CSI 自動移行の手動による有効化**を参照してください。 また、vSphere の場合は、以下の情報を参照してください。 • **vSphere in-tree PV を使用した OpenShift Container Platform 4.12 から 4.13 への更新** • **vSphere in-tree PV を使用した OpenShift Container Platform 4.12 から 4.14 への更新**

CSI 自動移行はシームレスに行ってください。この機能により、既存のすべての API オブジェクト (例: **PersistentVolume**、**PersistentVolumeClaim**、および **StorageClass**) を使用する方法が変更されることはありません。

in-tree の永続ボリューム (PV) または永続ボリューム要求 (PVC) の CSI 自動移行を有効にしても、元の in-tree ストレージプラグインがサポートしていない場合には、スナップショットや拡張などの新規の CSI ドライバー機能は有効にされません。

関連情報

- in-tree ボリュームから CSI への自動移行
- CSI 自動移行の手動による有効化

5.6.2. in-tree ボリュームから CSI への自動移行

OpenShift Container Platform は、以下の in-tree ボリュームタイプの対応する Container Storage Interface (CSI) ドライバーへの自動のシームレスな移行をサポートします。

- Azure Disk
- OpenStack Cinder
- Amazon Web Services (AWS) Elastic Block Storage (EBS)
- Google Compute Engine Persistent Disk (GCP PD)

これらのボリュームタイプの CSI 移行は一般提供 (GA) であるとみなされ、手動の介入は必要ありません。

新規の OpenShift Container Platform 4.11 以降のインストールでは、デフォルトのストレージクラスは CSI ストレージクラスになります。このストレージクラスを使用してプロビジョニングされるすべてのボリュームは CSI 永続ボリューム (PV) です。

4.10 以前から 4.11 にアップグレードされたクラスターの場合、CSI ストレージクラスが作成され、アップグレード前にデフォルトのストレージクラスが設定されていない場合にはそれがデフォルトに設定さ

れます。ごく稀なケースとして、同じ名前のストレージクラスがある場合、既存のストレージクラスは変更されません。既存の in-tree(インツリー) ストレージクラスはそのまま残り、既存の in-tree PV で機能するボリューム拡張などの特定の機能で必要になる場合があります。in-tree(インツリー) ストレージプラグインを参照するストレージクラスは機能し続けますが、デフォルトのストレージクラスを CSI ストレージクラスに切り替えることが推奨されます。

5.6.3. CSI 自動移行の手動による有効化

開発またはステージング環境の OpenShift Container Platform クラスターで Container Storage Interface (CSI) の移行をテストする必要がある場合、以下の in-tree ボリュームタイプについて、In-tree から CSI への移行を手動で有効にする必要があります。

- VMware vSphere Disk
- Azure File

重要

前述の in-tree ボリュームプラグインと CSI ドライバーのペアの CSI 自動移行は、テクノロジープレビュー機能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。

移行後、デフォルトのストレージクラスは in-tree(インツリー) ストレージクラスのままになります。

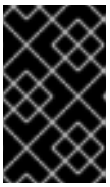
今後の OpenShift Container Platform リリースでは、すべてのストレージ in-tree プラグインについて CSI 自動移行はデフォルトで有効になる予定です。そのため、今この機能をテストして問題を報告することが強く推奨されます。

注記

CSI 自動移行のドレイン (解放) を有効にしてから、クラスター内のすべてのノードを順番に再起動します。これには少し時間がかかる場合があります。

手順

- 機能ゲートを有効にします (**ノード → クラスターの操作 → 機能ゲートを使用した機能の有効化** を参照してください)。



重要

機能ゲートを使用してテクノロジープレビュー機能をオンにした後にそれらをオフにすることはできません。その結果、クラスターのアップグレードはできなくなります。

以下の設定例により、現在テクノロジープレビュー (TP) ステータスにある、この機能でサポートされるすべての CSI ドライバーについて、CSI 自動移行が有効になります。

```
apiVersion: config.openshift.io/v1
```

```
kind: FeatureGate
metadata:
  name: cluster
spec:
  featureSet: TechPreviewNoUpgrade (1)
...
```

(1) Azure File と VMware vSphere の自動移行を有効にします。

CustomNoUpgrade featureSet と **featuregates** を以下のいずれかに設定して選択した CSI ドライバーの CSI 自動移行を指定できます。

- CSIMigrationAzureFile
- CSIMigrationvSphere

以下の設定例では、vSphere CSI ドライバーへの自動移行のみを有効にします。

```
apiVersion: config.openshift.io/v1
kind: FeatureGate
metadata:
  name: cluster
spec:
  featureSet: CustomNoUpgrade
  customNoUpgrade:
    enabled:
      - CSIMigrationvSphere (1)
    ...
```

(1) vSphere のみの自動移行を有効にします。

関連情報

- フィーチャーゲートを使用した機能の有効化

5.6.4. vSphere in-tree PV を使用した OpenShift Container Platform 4.12 から 4.13 への更新

vSphere in-tree 永続ボリューム (PV) を使用していて、OpenShift Container Platform 4.12 から 4.14 に更新する場合は、vSphere vCenter および ESXI ホストを 7.0 Update 3L または 8.0 Update 2 に更新します。更新しないと、OpenShift Container Platform の更新がブロックされます。vSphere の更新後、OpenShift Container Platform の更新が発生し、選択した場合にのみ vSphere の自動 Container Storage Interface (CSI)の移行が発生する可能性があります。

または、vSphere を更新したくない場合は、以下のコマンドを実行して管理者承認を実行することにより、OpenShift Container Platform の更新に進むことができます。

```
oc -n openshift-config patch cm admin-acks --patch '{"data":{"ack-4.12-kube-126-vsphere-migration-in-4.14":"true"}}' --type=merge
```

4.12 から 4.13 にアップグレードされたクラスターに対して CSI 移行がまだ有効になっていないため、通常は、OpenShift Container Platform 4.12 から 4.13 への更新に対して要求された管理者承認を提供しても安全です。ただし、Red Hat は、すべての in-tree ボリュームを CSI ドライバーによってシームレ

スに管理できるように、4.14 への今後の更新のための vSphere 環境の更新の計画を開始することを推奨します。

重要

OpenShift Container Platform 4.13.10 以降に更新せ **ず**、vSphere を更新せ **ず** に移行を選択した場合(**Additional resources**での **CSI 自動移行の手動有効化** を参照)、既知の問題が発生する可能性があります。移行を選択する前に、このナレッジベースの記事 を慎重にお読みください。

関連情報

- CSI 自動移行の手動による有効化

5.6.5. vSphere in-tree PV を使用した OpenShift Container Platform 4.12 から 4.14 への更新

vSphere in-tree 永続ボリューム (PV) を使用していて、OpenShift Container Platform 4.12 から 4.14 に更新する場合は、vSphere vCenter および ESXI ホストを 7.0 Update 3L または 8.0 Update 2 に更新します。更新しないと、OpenShift Container Platform の更新がブロックされます。vSphere の更新後、OpenShift Container Platform の更新が発生し、vSphere の自動 Container Storage Interface (CSI)の移行がデフォルトで自動的に実行されます。

vSphere を更新しない場合は、次のコマンドを **両方** 実行して管理者承認を実行して、OpenShift Container Platform の更新を続行できます。

```
oc -n openshift-config patch cm admin-acks --patch '{"data":{"ack-4.12-kube-126-vsphere-migration-in-4.14":"true"}}' --type=merge
```

```
oc -n openshift-config patch cm admin-acks --patch '{"data":{"ack-4.13-kube-127-vsphere-migration-in-4.14":"true"}}' --type=merge
```

重要

vSphere を更新せずに OpenShift Container Platform 4.14 に更新すると、OpenShift Container Platform 4.14 で CSI 移行がデフォルトで有効にされているために既知の問題が発生する可能性があります。更新する前に、このナレッジベースの記事 を慎重にお読みください。

OpenShift Container Platform 4.12 から 4.14 への更新には、Extended Update Support (EUS) から EUS への更新が伴います。このタイプの更新による影響とその実行方法については、以下の **関連情報** セクションで **EUS 間の更新** リンクを参照してください。

関連情報

- EUS から EUS への更新

5.7. ALICLOUD DISK CSI DRIVER OPERATOR

5.7.1. 概要

OpenShift Container Platform は、Alibaba AliCloud Disk Storage 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。

CSI Operator およびドライバーを使用する場合は、永続ストレージ および CSI ボリュームの設定 について理解しておくことが推奨されます。

AliCloud Disk 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 PV を作成するには、OpenShift Container Platform はデフォルトで AliCloud Disk CSI Driver Operator および AliCloud Disk CSI ドライバーを **openshift-cluster-csi-drivers** namespace にインストールします。

- **AliCloud Disk CSI Driver Operator** は、永続ボリューム要求 (PVC) の作成に使用できるストレージクラス (**alicloud-disk**) を提供します。AliCloud Disk CSI Driver Operator は、ストレージボリュームをオンデマンドで作成できるようにし、クラスター管理者がストレージを事前にプロビジョニングする必要がなくすことで、動的ボリュームのプロビジョニングをサポートします。
- **AliCloud Disk CSI ドライバー** を使用すると、AliCloud Disk PV を作成し、マウントできます。

5.7.2. 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。

関連情報

- CSI ボリュームの設定

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

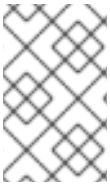
5.8.1. 概要

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。

Container Storage Interface (CSI) Operator およびドライバーを使用する場合は、永続ストレージ および CSI ボリュームの設定 を理解しておくことが推奨されます。

AWS EBS 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 PV を作成するには、OpenShift Container Platform はデフォルトで AWS EBS CSI ドライバー Operator および AWS EBS CSI ドライバーを **openshift-cluster-csi-drivers** namespace にインストールします。

- **AWS EBS CSI ドライバー Operator** は、PVC を作成するために使用できる StorageClass をデフォルトで提供します。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、AWS EBS StorageClass を作成するオプションもあります。
- **AWS EBS CSI ドライバー** を使用すると、AWS EBS PV を作成し、マウントできます。



注記

AWS EBS CSI Operator およびドライバーを OpenShift Container Platform 4.5 クラスターにインストールしている場合、OpenShift Container Platform 4.12 に更新する前に 4.5 Operator およびドライバーをアンインストールする必要があります。

5.8.2. 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。

今後の OpenShift Container Platform バージョンでは、既存の in-tree プラグインを使用してプロビジョニングされるボリュームは、同等の CSI ドライバーに移行される予定です。CSI 自動移行はシームレスに行ってください。移行をしても、永続ボリューム、永続ボリューム要求、ストレージクラスなどの既存の API オブジェクトを使用する方法は変更されません。移行の詳細は、CSI の自動移行 を参照してください。

完全な移行後、in-tree プラグインは最終的に OpenShift Container Platform の今後のバージョンで削除されます。

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。

関連情報

- 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

5.9. AWS ELASTIC FILE SERVICE CSI DRIVER OPERATOR

5.9.1. 概要

OpenShift Container Platform は、AWS Elastic File Service (EFS) 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。

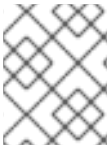
CSI Operator およびドライバーを使用する場合は、永続ストレージ および CSI ボリュームの設定 について理解しておくことが推奨されます。

AWS EFS CSI Driver Operator をインストールすると、OpenShift Container Platform はデフォルトで AWS EFS CSI Operator と AWS EFS CSI ドライバーを **openshift-cluster-csi-drivers** namespace にインストールします。これにより、AWS EFS CSI Driver Operator は、AWS EFS アセットにマウントする CSI がプロビジョニングする PV を作成することができます。

- **AWS EFS CSI Driver Operator**をインストールしても、永続ボリューム要求 (PVC) の作成に使

用するストレージクラスがデフォルトで作成されません。ただし、AWS EFS **StorageClass** を手動で作成することは可能です。AWS EFS CSI Driver Operator は、ストレージボリュームをオンデマンドで作成できるようにし、クラスター管理者がストレージを事前にプロビジョニングする必要がなくすことで、動的ボリュームのプロビジョニングをサポートします。

- **AWS EFS CSI ドライバー** を使用すると、AWS EFS PV を作成し、マウントできます。



注記

AWS EFS はリージョナルボリュームのみをサポートしており、ゾーンボリュームはサポートしていません。

5.9.2. 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。

5.9.3. AWS EFS CSI Driver Operator の設定

1. AWS EFS CSI ドライバードライバーをインストールします。
2. AWS EFS CSI ドライバーをインストールします。

5.9.3.1. AWS EFS CSI Driver Operator のインストール

AWS EFS 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、クラスター内で AWS EFS CSI ドライバー Operator をインストールおよび設定します。

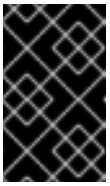
前提条件

- OpenShift Container Platform Web コンソールにアクセスできる。

手順

Web コンソールから AWS EFS CSI Driver 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. AWS EFS CSI Operator をインストールします。
   a. **Operators → OperatorHub** をクリックします。
   b. フィルターボックスに **AWS EFS CSI** と入力して、AWS EFS CSI Operator を探します。
   c. **AWS EFS CSI Driver Operator** ボタンをクリックします。



重要

AWS EFS Operator ではなく **AWS EFS CSI Driver Operator** を必ず選択してください。**AWS EFS Operator** はコミュニティー Operator であり、Red Hat ではサポートしていません。

d. **AWS EFS CSI Driver Operator** ページで **Install** をクリックします。

e. **Install Operator** のページで、以下のことを確認してください。

- **All namespaces on the cluster (default)**が選択されている。
- **Installed Namespace** が **openshift-cluster-csi-drivers** に設定されている。

f. **Install** をクリックします。
インストールが終了すると、AWS EFS CSI Operator が Web コンソールの **Installed Operators** に表示されます。

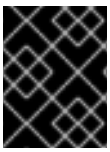
次のステップ

- AWS Secure Token Service (STS) と共に AWS EFS を使用している場合は、AWS EFS CSI ドライバーを STS で設定する必要があります。詳細は、AWS EFS CSI ドライバーと STS の設定 を参照してください。

5.9.3.2. Security Token Service を使用した AWS EFS CSI ドライバー Operator の設定

この手順では、AWS Security Token Service (STS) で OpenShift Container Platform と共に AWS EFS CSI ドライバー Operator を設定する方法を説明します。

この手順は、AWS EFS CSI Operator をインストールする前に実行しますが、**AWS EFS CSI ドライバー Operator のインストール手順** の一部として AWS EFS CSI ドライバーをまだインストールしていない場合に実行します。



重要

ドライバーのインストールおよびボリュームの作成後にこの手順を実行すると、ボリュームの Pod へのマウントに失敗します。

前提条件

- cluster-admin ロールを持つユーザーとしてクラスターにアクセスできる。
- AWS アカウントの認証情報。
- AWS EFS CSI Operator がインストールされている。

手順

AWS EFS CSI ドライバー Operator と STS を設定するには、以下を実行します。

1. STS とクラスターのインストールに使用した OpenShift Container Platform リリースイメージから CCO ユーティリティー (**ccoctl**) バイナリーをデプロイメントします。詳細は、Cloud Credential Operator ユーティリティーの設定を参照してください。
2. 以下の例に示されているように EFS **CredentialsRequest** YAML ファイルを作成および保存してから、それを **credrequests** ディレクトリーに配置します。

例

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: openshift-aws-efs-csi-driver
  namespace: openshift-cloud-credential-operator
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
    - action:
      - elasticfilesystem:*
      effect: Allow
      resource: '*'
  secretRef:
    name: aws-efs-cloud-credentials
    namespace: openshift-cluster-csi-drivers
  serviceAccountNames:
  - aws-efs-csi-driver-operator
  - aws-efs-csi-driver-controller-sa
```

3. **ccoctl** ツールを実行して AWS に新規の IAM ロールを生成し、その YAML ファイルをローカルファイルシステムに作成します (**<path_to_ccoctl_output_dir>/manifests/openshift-cluster-csi-drivers-aws-efs-cloud-credentials-credentials.yaml**)。

```
$ ccoctl aws create-iam-roles --name=<name> --region=<aws_region> --credentials-requests-dir=<path_to_directory_with_list_of_credentials_requests>/credrequests --identity-provider-arn=arn:aws:iam::<aws_account_id>:oidc-provider/<name>-oidc.s3.<aws_region>.amazonaws.com
```

- **name=<name>** は、追跡用に作成されたクラウドリソースにタグを付けるために使用される名前です。
- **region=<aws_region>** は、クラウドリソースが作成される AWS リージョンです。
- **dir=<path_to_directory_with_list_of_credentials_requests>/credrequests** は、前のステップの EFS CredentialsRequest ファイルが含まれるディレクトリーです。
- **<aws_account_id>** は AWS アカウント ID です。

  例

  ```
  $ ccoctl aws create-iam-roles --name my-aws-efs --credentials-requests-dir credrequests --identity-provider-arn arn:aws:iam::123456789012:oidc-provider/my-aws-efs-oidc.s3.us-east-2.amazonaws.com
  ```

  出力例

  ```
  2022/03/21 06:24:44 Role arn:aws:iam::123456789012:role/my-aws-efs -openshift-cluster-csi-drivers-aws-efs-cloud- created
  2022/03/21 06:24:44 Saved credentials configuration to: /manifests/openshift-cluster-csi-
  ```

```
drivers-aws-efs-cloud-credentials-credentials.yaml
2022/03/21 06:24:45 Updated Role policy for Role my-aws-efs-openshift-cluster-csi-drivers-aws-efs-cloud-
```

4. AWS EFS クラウド認証情報およびシークレットを作成します。

```
$ oc create -f <path_to_ccoctl_output_dir>/manifests/openshift-cluster-csi-drivers-aws-efs-cloud-credentials-credentials.yaml
```

例

```
$ oc create -f /manifests/openshift-cluster-csi-drivers-aws-efs-cloud-credentials-credentials.yaml
```

出力例

```
secret/aws-efs-cloud-credentials created
```

関連情報

- AWS EFS CSI Driver Operator のインストール
- Cloud Credential Operator ユーティリティーの設定
- AWS EFS CSI ドライバーのインストール

5.9.3.3. AWS EFS CSI ドライバーのインストール

前提条件

- OpenShift Container Platform Web コンソールにアクセスできる。

手順

1. **Administration** → **CustomResourceDefinitions** → **ClusterCSIDriver** をクリックします。
2. **Instances** タブで **Create ClusterCSIDriver** をクリックします。
3. 以下の YAML ファイルを使用します。

```
apiVersion: operator.openshift.io/v1
kind: ClusterCSIDriver
metadata:
    name: efs.csi.aws.com
spec:
  managementState: Managed
```

4. **Create** をクリックします。
5. 以下の条件が "true" に変わるのを待ちます。
   - AWSEFSDriverNodeServiceControllerAvailable

- AWSEFSDriverControllerServiceControllerAvailable

5.9.4. AWS EFS 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。

AWS EFS CSI ドライバー Operatorをインストールしても、ストレージクラスがデフォルトで作成されません。ただし、AWS EFS ストレージクラスを手動で作成することは可能です。

5.9.4.1. コンソールを使用した AWS EFS ストレージクラスの作成

手順

1. OpenShift Container Platform コンソールで、**Storage → StorageClasses** をクリックします。
2. **StorageClasses** ページで、**Create StorageClass** をクリックします。
3. **StorageClass** ページで、次の手順を実行します。
   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: 説明を入力します。
   c. 回収ポリシーを選択します。
   d. **Provisioner** ドロップダウンリストから **efs.csi.aws.com** を選択します。
   e. オプション: 選択したプロビジョナーの設定パラメーターを設定します。
4. **Create** をクリックします。

5.9.4.2. CLI を使用した AWS EFS ストレージクラスの作成

手順

- **StorageClass** オブジェクトを作成します。

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: efs-sc
provisioner: efs.csi.aws.com
parameters:
  provisioningMode: efs-ap 1
  fileSystemId: fs-a5324911 2
  directoryPerms: "700" 3
  gidRangeStart: "1000" 4
  gidRangeEnd: "2000" 5
  basePath: "/dynamic_provisioning" 6
```

1 動的プロビジョニングを有効にするには、**provisioningMode** に **efs-ap** を指定する必要があります。

2 **fileSystemId** は、手動で作成した EFS ボリュームの ID にする必要があります。

3 **directoryPerms** は、ボリュームのルートディレクトリーのデフォルトパーミッションです。この場合、ボリュームには所有者のみがアクセスできます。

4 5 **gidRangeStart** と **gidRangeEnd** は、AWS アクセスポイントの GID を設定する際に使用する POSIX グループ ID (GID) の範囲を設定します。指定しないと、デフォルトの範囲は 50000 - 7000000 になります。プロビジョニングされた各ボリューム、つまり AWS のアクセスポイントには、この範囲からの固有 GID が割り当てられます。

6 **basePath** は、動的にプロビジョニングされたボリュームを作成する際に使用される EFS ボリューム上のディレクトリーです。この場合は、EFS ボリューム上に /dynamic_provisioning/<random uuid> として PV がプロビジョニングされます。PV を使用する Pod には、そのサブディレクトリーのみがマウントされます。

注記

クラスター管理者は、それぞれが異なる EFS ボリュームを使用する複数の **StorageClass** オブジェクトを作成することができます。

5.9.5. AWS における EFS ボリュームへのアクセスの作成と設定

この手順では、OpenShift Container Platform で使用できるように、AWS で EFS ボリュームを作成および設定する方法を説明します。

前提条件

- AWS アカウントの認証情報。

手順

AWS で EFS ボリュームへのアクセスを作成および設定するには、以下の手順を実施します。

1. AWS のコンソールで、https://console.aws.amazon.com/efs を開きます。
2. **Create file system** をクリックします。
   - ファイルシステムの名前を入力します。
   - **Virtual Private Cloud(VPC)**には、OpenShift Container Platform の仮想プライベートクラウド (VPC) を選択します。
   - その他の選択項目については、デフォルト設定を受け入れます。
3. ボリュームとマウントターゲットが完全に作成され終わるのを待ちます。
   a. https://console.aws.amazon.com/efs#/file-systems にアクセスしてください。
   b. ボリュームをクリックし、**Network** タブで、すべてのマウントターゲットが利用可能になるまで待ちます (最長で 1 - 2 分間)。
4. **Network** タブでセキュリティーグループ ID をコピーします (次のステップで必要になります)。
5. https://console.aws.amazon.com/ec2/v2/home#SecurityGroups にアクセスし、EFS ボリュームで使用されているセキュリティーグループを探します。

6. **Inbound rules**タブで**Edit inbound rules** をクリックし、OpenShift Container Platform ノードが EFS ボリュームにアクセスできるようにするために、次の設定で新しいルールを追加します。

   - **Type**: NFS
   - **Protocol**: TCP
   - **Port range**: 2049
   - **Source**: ノードのカスタム/IP アドレス範囲 (例:"10.0.0.0/16")
     このステップで、OpenShift Container Platform がクラスターから NFS ポートを使用できるようになります。

7. ルールを保存します。

5.9.6. AWS EFS の動的プロビジョニング

AWS EFS CSI ドライバーは、他の CSI ドライバーとは異なる形態の動的プロビジョニングをサポートしています。既存の EFS ボリュームのサブディレクトリーとして新しい PV をプロビジョニングします。PV はお互いに独立しています。しかし、これらはすべて同じ EFS ボリュームを共有しています。ボリュームが削除されると、そのボリュームからプロビジョニングされたすべての PV も削除されます。EFS CSI ドライバーは、そのようなサブディレクトリーごとに AWS アクセスポイントを作成します。AWS AccessPoint の制限により、単一の **StorageClass**/EFS ボリュームから動的にプロビジョニングできるのは 1000 PV のみです。

重要

なお、**PVC.spec.resources** は EFS では強制されません。

以下の例では、5 GiB の容量を要求しています。しかし、作成された PV は無限であり、どんな量のデータ (ペタバイトのような) も保存することができます。ボリュームに大量のデータを保存してしまうと、壊れたアプリケーション、あるいは不正なアプリケーションにより、多額の費用が発生します。

AWS の EFS ボリュームサイズのモニタリングを使用することを強く推奨します。

前提条件

- AWS EFS ボリュームを作成している。
- AWS EFS ストレージクラスを作成している。

手順

動的プロビジョニングを有効にするには、以下の手順を実施します。

- 以前に作成した **StorageClass** を参照して、通常どおり PVC (または StatefulSet や Template) を作成します。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: test
spec:
  storageClassName: efs-sc
```

```
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 5Gi
```

動的プロビジョニングのセットアップに問題がある場合は、AWS EFS のトラブルシューティング を参照してください。

関連情報

- Creating and configuring access to AWS EFS volume(s)
- AWS EFS ストレージクラスの作成

5.9.7. AWS EFS による静的 PV の作成

動的プロビジョニングを行わずに、単一の PV として AWS EFS ボリュームを使用することが可能です。ボリューム全体が Pod にマウントされます。

前提条件

- AWS EFS ボリュームを作成している。

手順

- 以下の YAML ファイルで PV を作成します。

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: efs-pv
spec:
  capacity: 1
    storage: 5Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteMany
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  csi:
    driver: efs.csi.aws.com
    volumeHandle: fs-ae66151a 2
    volumeAttributes:
      encryptInTransit: "false" 3
```

1. **spec.capacity** には意味がなく、CSI ドライバーでは無視されます。PVC へのバインディング時にのみ使用されます。アプリケーションは、ボリュームに任意の量のデータを保存することができます。

2. **volumeHandle** は、AWS で作成した EFS ボリュームと同じ ID である必要があります。独自のアクセスポイントを提供する場合、**volumeHandle** は **<EFS volume ID>::<access point ID>** とします。たとえば、**fs-6e633ada::fsap-081a1d293f0004630** です。

3 必要に応じて、転送中の暗号化を無効にすることができます。デフォルトでは、暗号化が有効になっています。

静的 PV の設定に問題がある場合は、AWS EFS のトラブルシューティング を参照してください。

5.9.8. AWS EFS のセキュリティー

以下の情報は、AWS EFS のセキュリティーにとって重要です。

前述の動的プロビジョニングなどでアクセスポイントを使用する場合、Amazon はファイルの GID をアクセスポイントの GID に自動的に置き換えます。また、EFS では、ファイルシステムの権限を評価する際に、アクセスポイントのユーザー ID、グループ ID、セカンダリーグループ ID を考慮します。EFS は、NFS クライアントの ID を無視します。アクセスポイントの詳細については、https://docs.aws.amazon.com/efs/latest/ug/efs-access-points.html を参照してください。

その結果、EFS ボリュームは FSGroup を静かに無視します。OpenShift Container Platform は、ボリューム上のファイルの GID を FSGroup で置き換えることができません。マウントされた EFS アクセスポイントにアクセスできる Pod は、そこにあるすべてのファイルにアクセスできます。

これとは関係ありませんが、転送中の暗号化はデフォルトで有効になっています。詳しくは、https://docs.aws.amazon.com/efs/latest/ug/encryption-in-transit.html を参照してください。

5.9.9. AWS EFS のトラブルシューティング

以下の情報は、AWS EFS の問題をトラブルシューティングするためのガイダンスです。

- AWS EFS Operator と CSI ドライバーは、namespace **openshift-cluster-csi-drivers** で実行されます。
- AWS EFS Operator と CSI ドライバーのログ収集を開始するには、以下のコマンドを実行します。

```
$ oc adm must-gather
[must-gather      ] OUT Using must-gather plugin-in image: quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:125f183d13601537ff15b3239df95d47f0a604da2847b561151fedd699f5e3a5
[must-gather      ] OUT namespace/openshift-must-gather-xm4wq created
[must-gather      ] OUT clusterrolebinding.rbac.authorization.k8s.io/must-gather-2bd8x created
[must-gather      ] OUT pod for plug-in image quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:125f183d13601537ff15b3239df95d47f0a604da2847b561151fedd699f5e3a5 created
```

- AWS EFS Operator のエラーを表示するには、**ClusterCSIDriver** のステータスを表示します。

```
$ oc get clustercsidriver efs.csi.aws.com -o yaml
```

- Pod にボリュームをマウントできない場合 (以下のコマンドの出力に示す):

```
$ oc describe pod
...
  Type     Reason       Age    From               Message
  ----     ------       ----   ----               -------
  Normal   Scheduled    2m13s  default-scheduler  Successfully assigned default/efs-app to
```

```
ip-10-0-135-94.ec2.internal
  Warning  FailedMount  13s    kubelet            MountVolume.SetUp failed for volume "pvc-
d7c097e6-67ec-4fae-b968-7e7056796449" : rpc error: code = DeadlineExceeded desc =
context deadline exceeded (1)
  Warning  FailedMount  10s    kubelet            Unable to attach or mount volumes: unmounted
volumes=[persistent-storage], unattached volumes=[persistent-storage kube-api-access-
9j477]: timed out waiting for the condition
```

1 ボリュームがマウントされていないことを示す警告メッセージ。

このエラーは、OpenShift Container Platform ノードと AWS EFS 間のパケットを AWS がドロップすることで頻繁に発生します。

以下が正しいことを確認します。

- AWS のファイアウォールとセキュリティーグループ
- ネットワーク: ポート番号と IP アドレス

5.9.10. AWS EFS CSI ドライバー Operator のアンインストール

AWS EFS CSI ドライバー Operator をアンインストールすると、すべての EFS PV にアクセスできなくなる。

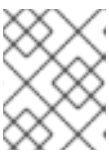
前提条件

- OpenShift Container Platform Web コンソールにアクセスできる。

手順

Web コンソールから AWS EFS CSI Driver 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. AWS EFS PV を使用するすべてのアプリケーションを停止します。
3. すべての AWS EFS PV を削除します。
   a. **Storage → PersistentVolumeClaims** をクリックします。
   b. AWS EFS CSI ドライバー Operator が使用している各 PVC を選択し、PVC の右端にあるドロップダウンメニューをクリックして、**Delete PersistentVolumeClaims**をクリックします。
4. AWS EFS CSI ドライバーをアンインストールします。



注記

Operator をアンインストールする前に、まず CSI ドライバーを削除する必要があります。

   a. **Administration → CustomResourceDefinitions → ClusterCSIDriver** をクリックします。
   b. **Instances** タブの **efs.csi.aws.com** の左端にあるドロップダウンメニューをクリックし、**Delete ClusterCSIDriver** をクリックします。

c. プロンプトが表示されたら、**Delete** をクリックします。

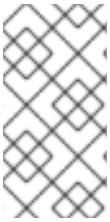
5. AWS EFS CSI Operator をアンインストールします。

a. **Operators → Installed Operators** をクリックします。

b. **Installed Operators**ページで、スクロールするか、**Search by name** ボックスに AWS EFS CSI と入力してオペレーターを見つけ、クリックします。

c. **Installed Operators > Operator details**ページの右上にある **Actions → Uninstall Operator** をクリックします。

d. **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。
アンインストールすると、AWS EFS CSI Driver Operator が Web コンソールの **Installed Operators** セクションにリスト表示されなくなります。



注記

クラスターを破棄 (**openshift-install destroy cluster**) する前に、AWS の EFS ボリュームを削除する必要があります。クラスターの VPC を使用する EFS ボリュームがある場合、OpenShift Container Platform クラスターを破棄することはできません。Amazon はこのような VPC の削除を許可していません。

5.9.11. 関連情報

- CSI ボリュームの設定

5.10. AZURE DISK CSI DRIVER OPERATOR

5.10.1. 概要

OpenShift Container Platform は、Microsoft Azure Disk Storage 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。

CSI Operator およびドライバーを使用する場合は、永続ストレージ および CSI ボリュームの設定 について理解しておくことが推奨されます。

Azure Disk 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 (PV) を作成するには、OpenShift Container Platform は、デフォルトで Azure Disk CSI Driver Operator および Azure Disk CSI ドライバーを **openshift-cluster-csi-drivers** namespace にインストールします。

- **Azure Disk CSI Driver Operator**: 永続ボリューム要求 (PVC) の作成に使用できる **managed-csi** というストレージクラスを提供します。Azure Disk CSI ドライバー Operator は、ストレージボリュームをオンデマンドで作成できるようにし、クラスター管理者がストレージを事前にプロビジョニングする必要がなくすことで、動的ボリュームのプロビジョニングをサポートします。

- **Azure Disk CSI ドライバー** を使用すると、Azure Disk PV を作成し、マウントできます。

5.10.2. 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。

注記

OpenShift Container Platform は、Azure Disk インツリーボリュームプラグインを同等の CSI ドライバーに自動的に移行します。詳細は、CSI 自動移行 を参照してください。

5.10.3. ストレージアカウントタイプを使用した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で、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。

ストレージクラスを作成するときに、ストレージアカウントの種類を指定できます。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、**Standard_LRS**、**Premium_LRS**、**StandardSSD_LRS**、**UltraSSD_LRS**、**Premium_ZRS**、および **StandardSSD_ZRS** です。Azure SKU レベルを見つける方法については、SKU Types を参照してください。

ZRS には、リージョン制限があります。これらの制限については、ZRS の制限 を参照してください。

前提条件

- 管理者権限を持つ OpenShift Container Platform クラスターへのアクセス

手順

次の手順を使用して、ストレージアカウントの種類でストレージクラスを作成します。

1. 次のような YAML ファイルを使用して、ストレージアカウントの種類を指定するストレージクラスを作成します。

```
$ oc create -f - << EOF
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: <storage-class> 1
provisioner: disk.csi.azure.com
parameters:
  skuName: <storage-class-account-type> 2
reclaimPolicy: Delete
volumeBindingMode: WaitForFirstConsumer
allowVolumeExpansion: true
EOF
```

1 ストレージクラス名。

2 ストレージアカウントの種類。これは、Azure ストレージアカウントの SKU レベル Standard_LRS、**Premium_LRS**、**StandardSSD_LRS**、**UltraSSD_LRS**、**Premium_ZRS**、**StandardSSD_ZRS** に対応しています。

2. ストレージクラスを一覧表示して、ストレージクラスが作成されたことを確認します。

```
$ oc get storageclass
```

出力例

```
$ oc get storageclass
NAME                 PROVISIONER          RECLAIMPOLICY   VOLUMEBINDINGMODE
ALLOWVOLUMEEXPANSION   AGE
azurefile-csi        file.csi.azure.com   Delete          Immediate              true                   68m
managed-csi (default)   disk.csi.azure.com   Delete          WaitForFirstConsumer   true
68m
sc-prem-zrs          disk.csi.azure.com   Delete          WaitForFirstConsumer   true
4m25s (1)
```

(1) ストレージアカウントタイプを使用する新しいストレージクラス。

5.10.4. PVC を使用して Ultra ディスクと共にマシンをデプロイするマシンセット

Ultra ディスクと共にマシンをデプロイする Azure で実行されるマシンセットを作成できます。Ultra ディスクは、最も要求の厳しいデータワークロードでの使用を目的とした高性能ストレージです。

in-tree プラグインおよび CSI ドライバーの両方が、Ultra ディスクを有効にするための PVC の使用をサポートします。PVC を作成せずに、データディスクとしての Ultra デイスクと共にマシンをデプロイすることもできます。

関連情報

- Microsoft Azure Ultra ディスクのドキュメント
- in-tree(インツリー)PVC を使用して Ultra ディスクにマシンをデプロイするマシンセット
- データディスクとしての Ultra ディスク上にマシンをデプロイするマシンセット

5.10.4.1. マシンセットを使用した Ultra ディスクを持つマシンの作成

マシンセットの YAML ファイルを編集することで、Azure 上に Ultra ディスクと共にマシンをデプロイできます。

前提条件

- 既存の Microsoft Azure クラスターがある。

手順

1. 既存の Azure **MachineSet** カスタムリソース (CR) をコピーし、次のコマンドを実行して編集します。

```
$ oc edit machineset <machine-set-name>
```

ここで、**<machine-set-name>** は、Ultra ディスクと共にマシンをプロビジョニングするマシンセットです。

2. 示された位置に次の行を追加します。

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
spec:
  template:
    spec:
      metadata:
        labels:
          disk: ultrassd (1)
      providerSpec:
        value:
          ultraSSDCapability: Enabled (2)
```

(1) このマシンセットによって作成されるノードを選択するために使用するラベルを指定します。この手順では、この値に **disk.ultrassd** を使用します。

(2) これらの行により、Ultra ディスクの使用が可能になります。

3. 次のコマンドを実行して、更新された設定を使用してマシンセットを作成します。

```
$ oc create -f <machine-set-name>.yaml
```

4. 以下の YAML 定義が含まれるストレージクラスを作成します。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ultra-disk-sc (1)
parameters:
  cachingMode: None
  diskIopsReadWrite: "2000" (2)
  diskMbpsReadWrite: "320" (3)
  kind: managed
  skuname: UltraSSD_LRS
provisioner: disk.csi.azure.com (4)
reclaimPolicy: Delete
volumeBindingMode: WaitForFirstConsumer (5)
```

(1) ストレージクラスの名前を指定します。この手順では、この値に **ultra-disk-sc** を使用しています。

(2) ストレージクラスの IOPS の数値を指定します。

(3) ストレージクラスのスループットを MBps 単位で指定します。

(4) Azure Kubernetes Service (AKS) バージョン 1.21 以降の場合は、**disk.csi.azure.com** を使用します。以前のバージョンの AKS の場合は、**kubernetes.io/azure-disk** を使用します。

(5) オプション: ディスクを使用する Pod の作成を待機するには、このパラメーターを指定します。

5. 以下の YAML 定義が含まれる、**ultra-disk-sc** ストレージクラスを参照する永続ボリューム要求 (PVC) を作成します。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: ultra-disk 1
spec:
  accessModes:
  - ReadWriteOnce
  storageClassName: ultra-disk-sc 2
  resources:
    requests:
      storage: 4Gi 3
```

1. PVC の名前を指定します。この手順では、この値に **ultra-disk** を使用しています。
2. この PVC は **ultra-disk-sc** ストレージクラスを参照します。
3. ストレージクラスのサイズを指定します。最小値は **4Gi** です。

6. 以下の YAML 定義が含まれる Pod を作成します。

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-ultra
spec:
  nodeSelector:
    disk: ultrassd 1
  containers:
  - name: nginx-ultra
    image: alpine:latest
    command:
      - "sleep"
      - "infinity"
    volumeMounts:
    - mountPath: "/mnt/azure"
      name: volume
  volumes:
    - name: volume
      persistentVolumeClaim:
        claimName: ultra-disk 2
```

1. Ultra ディスクの使用を有効にするマシンセットのラベルを指定します。この手順では、この値に **disk.ultrassd** を使用します。
2. この Pod は **ultra-disk** PVC を参照します。

検証

1. 次のコマンドを実行して、マシンが作成されていることを確認します。

```
$ oc get machines
```

マシンは **Running** 状態になっているはずです。

2. 実行中でノードが接続されているマシンの場合、次のコマンドを実行してパーティションを検証します。

```
$ oc debug node/<node-name> -- chroot /host lsblk
```

このコマンドでは、**oc debug node/<node-name>** がノード **<node-name>** でデバッグシェルを開始し、**--** を付けてコマンドを渡します。渡されたコマンド **chroot /host** は、基盤となるホスト OS バイナリーへのアクセスを提供し、**lsblk** は、ホスト OS マシンに接続されているブロックデバイスを表示します。

次のステップ

- Pod 内から Ultra ディスクを使用するには、マウントポイントを使用するワークロードを作成します。次の例のような YAML ファイルを作成します。

```
apiVersion: v1
kind: Pod
metadata:
  name: ssd-benchmark1
spec:
  containers:
  - name: ssd-benchmark1
    image: nginx
    ports:
      - containerPort: 80
        name: "http-server"
    volumeMounts:
    - name: lun0p1
      mountPath: "/tmp"
  volumes:
    - name: lun0p1
      hostPath:
        path: /var/lib/lun0p1
        type: DirectoryOrCreate
  nodeSelector:
    disktype: ultrassd
```

5.10.4.2. Ultra ディスクを有効にするマシンセットのリソースに関するトラブルシューティング

このセクションの情報を使用して、発生する可能性のある問題を理解し、回復してください。

5.10.4.2.1. Ultra ディスクがサポートする永続ボリューム要求 (PVC) をマウントできない

Ultra ディスクでサポートされる永続ボリューム要求 (PVC) のマウントに問題がある場合、Pod は **ContainerCreating** 状態のままになり、アラートがトリガーされます。

たとえば、**additionalCapabilities.ultraSSDEnabled** パラメーターが Pod をホストするノードをサポートするマシンで設定されていない場合、以下のエラーメッセージが表示されます。

> StorageAccountType UltraSSD_LRS can be used only when additionalCapabilities.ultraSSDEnabled is set.

- この問題を解決するには、以下のコマンドを実行して Pod を記述します。

```
$ oc -n <stuck_pod_namespace> describe pod <stuck_pod_name>
```

5.10.5. 関連情報

- Azure Disk を使用した永続ストレージ
- CSI ボリュームの設定

5.11. AZURE FILE CSI DRIVER OPERATOR

5.11.1. 概要

OpenShift Container Platform は、Microsoft Azure File Storage 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。

CSI Operator およびドライバーを使用する場合は、永続ストレージ および CSI ボリュームの設定 について理解しておくことが推奨されます。

Azure File 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 (PV) を作成するには、OpenShift Container Platform は、デフォルトで Azure File CSI Driver Operator および Azure File CSI ドライバーを **openshift-cluster-csi-drivers** namespace にインストールします。

- **Azure File CSI Driver Operator**: 永続ボリューム要求 (PVC) の作成に使用できる **azurefile-csi** というストレージクラスを提供します。
- **Azure File CSI ドライバー** を使用すると、Azure File PV を作成し、マウントできます。Azure File CSI ドライバーは、ストレージボリュームをオンデマンドで作成できるようにし、クラスター管理者がストレージを事前にプロビジョニングする必要がなくすことで、動的ボリュームのプロビジョニングをサポートします。

Azure File CSI Driver Operator は以下を **サポートしません**。

- 仮想ハードディスク (VHD)
- ネットワークファイルシステム (NFS): OpenShift Container Platform は NFS がサポートするストレージクラスをデプロイしません。
- FIPS モードが有効なノードでの実行

サポートされる機能の詳細は、サポートされる CSI ドライバーおよび機能 を参照してください。

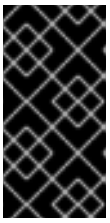
5.11.2. NFS のサポート

OpenShift Container Platform は、Network File System (NFS) を備えた Azure File Container Storage Interface (CSI) Driver Operator がサポートされていますが、次の制限があります。

- コントロールプレーンノードにスケジュールされている Azure File NFS ボリュームを含む Pod を作成すると、マウントが拒否されます。

この問題を回避するには、コントロールプレーンノードがスケジュール可能で、Pod がワーカーノードで実行できる場合は、**nodeSelector** または Affinity を使用してワーカーノードで Pod をスケジュールします。

- FS グループポリシーの動作:



重要

NFS を使用した Azure File CSI は、Pod によって要求された **fsGroupChangePolicy** を受け入れません。NFS を使用した Azure File CSI は、Pod によって要求されたポリシーに関係なく、デフォルトの **OnRootMismatch** FS グループポリシーを適用します。

- Azure File CSI Operator は、NFS のストレージクラスを自動的に作成しません。手動で作成する必要があります。次のようなファイルを使用します。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: <storage-class-name> 1
provisioner: file.csi.azure.com 2
parameters:
  protocol: nfs 3
  skuName: Premium_LRS  # available values: Premium_LRS, Premium_ZRS
mountOptions:
  - nconnect=4
```

1. ストレージクラス名。
2. Azure File CSI プロバイダーを指定します。
3. ストレージバックエンドプロトコルとして NFS を指定します。

5.11.3. 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。

関連情報

- Azure File を使用した永続ストレージ
- CSI ボリュームの設定

5.12. AZURE STACK HUB CSI DRIVER OPERATOR

5.12.1. 概要

OpenShift Container Platform は、Azure Stack Hub Storage 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。Azure Stack ポートフォリオの一部である Azure Stack Hub を使用すると、オンプレミス環境でアプリケーションを実行し、データセンターで Azure サービスを配信できます。

CSI Operator およびドライバーを使用する場合は、永続ストレージ および CSI ボリュームの設定 について理解しておくことが推奨されます。

Azure Stack Hub 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 (PV) を作成するには、OpenShift Container Platform は、デフォルトで Azure Stack Hub CSI Driver Operator および Azure Stack Hub CSI ドライバーを **openshift-cluster-csi-drivers** namespace にインストールします。

- **Azure Stack Hub CSI Driver Operator**は、デフォルトのストレージアカウントタイプとして Standard_LRS が設定されたストレージクラス **(managed-csi)** を提供し、永続的ボリューム要求 (PVC) の作成に使用することができます。Azure Stack Hub CSI Driver Operator は、ストレージボリュームをオンデマンドで作成できるようにし、クラスター管理者がストレージを事前にプロビジョニングする必要がなくすことで、動的ボリュームのプロビジョニングをサポートします。

- **Azure Stack Hub CSI ドライバー**を使用すると、Azure Stack Hub PV を作成し、マウントできます。

5.12.2. 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。

5.12.3. 関連情報

- CSI ボリュームの設定

5.13. GCP PD CSI DRIVER OPERATOR

5.13.1. 概要

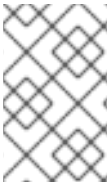
OpenShift Container Platform は、Google Cloud Platform (GCP) 永続ディスク (PD) ストレージの Container Storage Interface (CSI) ドライバーを使用して永続ボリューム (PV) をプロビジョニングできます。

Container Storage Interface (CSI) Operator およびドライバーを使用する場合、永続ストレージ および CSI ボリュームの設定 について理解しておくことが推奨されます。

GCP PD 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 (PV) を作成するには、OpenShift Container Platform はデフォルトで GCP PD CSI Driver Operator および GCP PD CSI ドライバーを **openshift-cluster-csi-drivers** namespace にインストールします。

- **GCP PD CSI Driver Operator**: デフォルトで、 Operator は PVC の作成に使用できるストレージクラスを提供します。GCE Persistent Disk を使用した永続ストレージ で説明されているように、GCP PD ストレージを作成するオプションもあります。

- **GCP PD ドライバー**: このドライバーを使用すると、GCP PD PV を作成し、マウントできます。



注記

OpenShift Container Platform では、GCE Persistent Disk in-tree ボリュームプラグインと同等の CSI ドライバーに自動的に移行できます。詳細は、CSI 自動移行 を参照してください。

5.13.2. 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。

5.13.3. GCP PD CSI ドライバーストレージクラスパラメーター

Google Cloud Platform (GCP) 永続ディスク (PD) の Container Storage Interface (CSI) ドライバーは CSI の **external-provisioner** サイドカーをコントローラーとして使用します。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、**CreateVolume** 操作をトリガーして永続ボリューム (PV) を管理します。

GCP PD CSI ドライバーは、**csi.storage.k8s.io/fstype** パラメーターキーを使用して動的プロビジョニングをサポートします。以下の表は、OpenShift Container Platform がサポートするすべての GCP PD CSI ストレージクラスパラメーターについて説明しています。

表5.3 CreateVolume パラメーター

パラメーター	値	デフォルト	説明
type	**pd-ssd** または **pd-standard**	**pd-standard**	標準の PV または solid-state-drive (SSD) PV を選択できます。
replication-type	**none** または **region-pd**	**none**	zonal またはリージョン PV を選択できます。
disk-encryption-kms-key	新規ディスクの暗号化に使用するキーの完全修飾リソース識別子。	空の文字列	顧客管理の暗号鍵 (CMEK) を使用して新規ディスクを暗号化します。

5.13.4. カスタムで暗号化された永続ボリュームの作成

PersistentVolumeClaim オブジェクトの作成時に、OpenShift Container Platform は新規永続ボリューム (PV) をプロビジョニングし、**PersistentVolume** オブジェクトを作成します。新規に作成された PV を暗号化することで、Google Cloud Platform (GCP) にカスタム暗号化キーを追加し、クラスター内の PV を保護することができます。

暗号化の場合、作成した新たに割り当てられる PV は、新規または既存の Google Cloud Key Management Service (KMS) キーを使用してクラスターで顧客管理の暗号鍵 (CMEK) を使用します。

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。
- Cloud KMS キーリングとキーのバージョンを作成している。

CMEK および Cloud KMS リソースの詳細は、顧客管理の暗号鍵 (CMEK) の使用 を参照してください。

手順

カスタムで暗号化された PV を作成するには、以下の手順を実行します。

1. Cloud KMS キーを使用してストレージクラスを作成します。以下の例では、暗号化されたボリュームの動的プロビジョニングを有効にします。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-gce-pd-cmek
provisioner: pd.csi.storage.gke.io
volumeBindingMode: "WaitForFirstConsumer"
allowVolumeExpansion: true
parameters:
  type: pd-standard
  disk-encryption-kms-key: projects/<key-project-id>/locations/<location>/keyRings/<key-ring>/cryptoKeys/<key> (1)
```

(1) このフィールドは、新規ディスクの暗号化に使用されるキーのリソース識別子である必要があります。値では、大文字と小文字が区別されます。キー ID の値を指定する方法は、Retrieving a resource's ID および Getting a Cloud KMS resource ID を参照してください。

注記

disk-encryption-kms-key パラメーターは既存のストレージクラスに追加することはできません。ただし、ストレージクラスを削除し、同じ名前および異なるパラメーターセットでこれを再作成できます。これを実行する場合、既存クラスのプロビジョナーは **pd.csi.storage.gke.io** である必要があります。

2. **oc** コマンドを使用して、ストレージクラスを OpenShift Container Platform クラスターにデプロイします。

```
$ oc describe storageclass csi-gce-pd-cmek
```

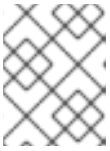
出力例

```
Name:                  csi-gce-pd-cmek
IsDefaultClass:        No
Annotations:           None
Provisioner:           pd.csi.storage.gke.io
Parameters:            disk-encryption-kms-key=projects/key-project-id/locations/location/keyRings/ring-name/cryptoKeys/key-name,type=pd-standard
AllowVolumeExpansion:  true
MountOptions:          none
```

```
ReclaimPolicy:          Delete
VolumeBindingMode:      WaitForFirstConsumer
Events:                 none
```

3. 直前の手順で作成したストレージクラスオブジェクトの名前に一致する **pvc.yaml** という名前のファイルを作成します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: podpvc
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: csi-gce-pd-cmek
  resources:
    requests:
      storage: 6Gi
```



注記

新規ストレージクラスをデフォルトとしてマークした場合は、**storageClassName** フィールドを省略できます。

4. PVC をクラスターに適用します。

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

5. PVC のステータスを取得し、これが作成され、新規にプロビジョニングされた PV にバインドされていることを確認します。

```
$ oc get pvc
```

出力例

```
NAME      STATUS    VOLUME                                     CAPACITY   ACCESS MODES
STORAGECLASS     AGE
podpvc    Bound     pvc-e36abf50-84f3-11e8-8538-42010a800002   10Gi       RWO           csi-
gce-pd-cmek  9s
```

注記

ストレージクラスで **volumeBindingMode** フィールドが **WaitForFirstConsumer** に設定されている場合、これを検証する前に PVC を使用するために Pod を作成する必要があります。

CMEK で保護される PV が OpenShift Container Platform クラスターで使用できるようになります。

関連情報

- GCE Persistent Disk を使用した永続ストレージ

- CSI ボリュームの設定

5.14. GOOGLE COMPUTE PLATFORM FILESTORE CSI ドライバーオペレーター

5.14.1. 概要

OpenShift Container Platform は、Google Compute Platform (GCP) Filestore Storage 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。

重要

GCP Filestore CSI Driver Operator はテクノロジープレビュー機能のみです。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。

CSI Operator およびドライバーを使用する場合は、永続ストレージ および CSI ボリュームの設定 について理解しておくことが推奨されます。

GCP Filestore Storage アセットにマウントする CSI プロビジョニング PV を作成するには、GCP Filestore CSI Driver Operator と GCP Filestore CSI ドライバーを **openshift-cluster-csi-drivers** namespace にインストールします。

- **GCP Filestore CSI Driver Operator** は、デフォルトではストレージクラスを提供しませんが、必要に応じて作成 できます。GCP Filestore CSI Driver Operator は、ストレージボリュームをオンデマンドで作成できるようにすることで動的なボリュームプロビジョニングをサポートし、クラスター管理者がストレージを事前にプロビジョニングする必要がなくなります。
- **GCP Filestore CSI ドライバー** を使用すると、GCP Filestore PV を作成してマウントできます。

5.14.2. 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。

5.14.3. GCP Filestore CSI Driver Operator のインストール

デフォルトでは、Google Compute Platform (GCP) Filestore Container Storage Interface (CSI) Driver Operator は OpenShift Container Platform にインストールされません。次の手順を使用して、GCP Filestore CSI Driver Operator をクラスターにインストールします。

前提条件

- OpenShift Container Platform Web コンソールにアクセスできる。

手順

ウェブコンソールから GCP Filestore CSI Driver Operator をインストールするには:

1. Web コンソールにログインします。
2. 次のコマンドを実行して、GCE プロジェクトで Filestore API を有効にします。

```
$ gcloud services enable file.googleapis.com  --project <my_gce_project>
```
(1)

(1) **<my_gce_project>** を Google Cloud プロジェクトに置き換えます。

これは、Google Cloud Web コンソールを使用して行うこともできます。

3. GCP Filestore CSI Operator をインストールします。
   a. **Operators → OperatorHub** をクリックします。
   b. フィルターボックスに **GCP Filestore** と入力して、GCP Filestore CSI Operator を見つけます。
   c. **GCP Filestore CSI Driver Operator** ボタンをクリックします。
   d. **GCP Filestore CSI Driver Operator** ページで、**Install** をクリックします。
   e. **Install Operator** のページで、以下のことを確認してください。
      - **All namespaces on the cluster (default)**が選択されている。
      - **Installed Namespace** が **openshift-cluster-csi-drivers** に設定されている。
   f. **Install** をクリックします。
   インストールが終了すると、GCP Filestore CSI Operator が Web コンソールの**Installed Operators**に表示されます。
4. GCP Filestore CSI ドライバーをインストールします。
   a. **administration → CustomResourceDefinitions → ClusterCSIDriver**をクリックします。
   b. **Instances** タブで **Create ClusterCSIDriver** をクリックします。
   以下の YAML ファイルを使用します。

```
apiVersion: operator.openshift.io/v1
kind: ClusterCSIDriver
metadata:
    name: filestore.csi.storage.gke.io
spec:
  managementState: Managed
```

   c. **Create** をクリックします。
   d. 以下の条件が "true" に変わるのを待ちます。
      - GCPFilestoreDriverCredentialsRequestControllerAvailable

- GCPFilestoreDriverNodeServiceControllerAvailable
- GCPFilestoreDriverControllerServiceControllerAvailable

関連情報

- Google Cloud で API を有効にします。
- Enabling an API using the Google Cloud web console 。

5.14.4. GCP Filestore Storage のストレージクラスの作成

Operator をインストールしたら、Google Compute Platform (GCP) Filestore ボリュームの動的プロビジョニング用のストレージクラスを作成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。

手順

ストレージクラスを作成するには、以下を行います。

1. 次のサンプル YAML ファイルを使用してストレージクラスを作成します。

   サンプル YAML ファイル

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: filestore-csi
provisioner: filestore.csi.storage.gke.io
parameters:
  network: network-name (1)
allowVolumeExpansion: true
volumeBindingMode: WaitForFirstConsumer
```

   (1) Filestore インスタンスを作成する GCP Virtual Private Cloud (VPC) ネットワークの名前を指定します。

2. Filestore インスタンスを作成する VPC ネットワークの名前を指定します。
   Filestore インスタンスを作成する VPC ネットワークを指定することを推奨します。VPC ネットワークが指定されていないと、Container Storage Interface (CSI) ドライバーは、プロジェクトのデフォルト VPC ネットワークにインスタンスを作成しようとします。IPI インストールでは、VPC ネットワーク名は通常、クラスター名に接尾辞 -network を付けたものです。ただし、UPI インストールでは、VPC ネットワーク名はユーザーが選択した任意の値にすることができます。

   次のコマンドを使用して **MachineSets** オブジェクトを調べると、VPC ネットワーク名を確認できます。

```
$ oc -n openshift-machine-api get machinesets -o yaml | grep "network:"
          - network: gcp-filestore-network
(...)
```

この例では、このクラスターの VPC ネットワーク名は gcp-filestore-network です。

5.14.5. クラスターと GCP Filestore の破棄

通常、クラスターを破棄すると、OpenShift Container Platform インストーラーはそのクラスターに属するすべてのクラウドリソースを削除します。ただし、クラスターが破棄されても、Google Compute Platform (GCP) Filestore インスタンスは自動的に削除されないため、クラスターを破棄する前に、Filestore ストレージクラスを使用するすべての永続ボリュームクレーム (PVC) を手動で削除する必要があります。

手順

すべての GCP Filestore PVC を削除するには:

1. ストレージクラス **filestore-csi** を使用して作成されたすべての PVC を一覧表示します。

```
$ oc get pvc -o json -A | jq -r '.items[] | select(.spec.storageClassName == "filestore-csi")
```

2. 前のコマンドでリストされたすべての PVC を削除します。

```
$ oc delete <pvc-name> 1
```

1 <pvc-name> を、削除する必要がある PVC の名前に置き換えます。

5.14.6. 関連情報

- CSI ボリュームの設定

5.15. IBM VPC BLOCK CSI DRIVER OPERATOR

5.15.1. 概要

OpenShift Container Platform は、IBM Virtual Private Cloud (VPC) Block Storage 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。

CSI Operator およびドライバーを使用する場合は、永続ストレージ および CSI ボリュームの設定 について理解しておくことが推奨されます。

IBM VPC Block 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 (PV) を作成するには、OpenShift Container Platform は、デフォルトで IBM VPC Block CSI Driver Operator および IBM VPC Block CSI ドライバーを **openshift-cluster-csi-drivers** namespace にインストールします。

- **IBM VPC Block CSI Driver Operator**は、永続ボリューム要求 (PVC) の作成に使用できる異なるレイヤー用に、**ibmc-vpc-block-10iops-tier** (デフォルト)、**ibmc-vpc-block-5iops-tier**、および **ibmc-vpc-block-custom** という 3 つのストレージクラスを提供します。IBM VPC Block CSI Driver Operator は、ストレージボリュームをオンデマンドで作成できるようにすることで動的なボリュームプロビジョニングをサポートするので、クラスター管理者はストレージを事前にプロビジョニングする必要がありません。
- **IBM VPC Block CSI ドライバー**を使用すると、IBM VPC Block PV を作成し、マウントできます。

5.15.2. 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。

関連情報

- CSI ボリュームの設定

5.16. OPENSTACK CINDER CSI DRIVER OPERATOR

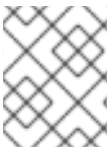
5.16.1. 概要

OpenShift Container Platform は、OpenStack Cinder 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。

Container Storage Interface (CSI) Operator およびドライバーを使用する場合、永続ストレージ および CSI ボリュームの設定 について理解しておくことが推奨されます。

OpenStack Cinder 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る PV を作成するには、OpenShift Container Platform は **openshift-cluster-csi-drivers** namespace に OpenStack Cinder CSI Driver Operator および OpenStack Cinder CSI ドライバーをインストールします。

- **OpenStack Cinder CSI Driver Operator** は、PVC の作成に使用できる CSI ストレージクラスを提供します。
- **OpenStack Cinder CSI ドライバー** を使用すると、OpenStack Cinder PV を作成し、マウントすることができます。



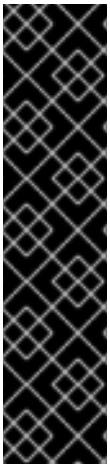
注記

OpenShift Container Platform では、Cinder インツリーボリュームプラグインと同等の CSI ドライバーに自動的に移行できます。詳細は、CSI 自動移行 を参照してください。

5.16.2. 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、Cider 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 または CSI 以外のドライバーの使用に設定されます。

今後の OpenShift Container Platform バージョンでは、既存の in-tree プラグインを使用してプロビジョニングされるボリュームは、同等の CSI ドライバーに移行される予定です。CSI 自動移行はシームレスに行ってください。移行をしても、永続ボリューム、永続ボリューム要求、ストレージクラスなどの既存の API オブジェクトを使用する方法は変更されません。移行の詳細は、CSI の自動移行 を参照してください。

完全な移行後、in-tree プラグインは最終的に OpenShift Container Platform の今後のバージョンで削除されます。

5.16.3. OpenStack Cinder CSI をデフォルトのストレージクラスに設定する

OpenStack Cinder CSI ドライバーは、**cinder.csi.openstack.org** パラメーターキーを使用して動的プロビジョニングをサポートします。

OpenShift Container Platform で OpenStack Cinder CSI プロビジョニングを有効にするには、デフォルトの in-tree(インツリー) ストレージクラスを **standard-csi** で上書きすることが推奨されます。または、永続ボリューム要求 (PVC) を作成し、ストレージクラスを standard-csi として指定できます。

OpenShift Container Platform では、デフォルトのストレージクラスは in-tree(インツリー)Cinder ドライバーを参照します。ただし、CSI の自動移行が有効な場合に、デフォルトのストレージクラスを使用して作成されたボリュームは実際には CSI ドライバーを使用します。

手順

以下の手順に従ってデフォルトの in-tree(インツリー) ストレージクラスを上書きし、**standard-csi** ストレージクラスを適用します。

1. ストレージクラスをリスト表示します。

```
$ oc get storageclass
```

出力例

```
NAME                 PROVISIONER                RECLAIMPOLICY   VOLUMEBINDINGMODE
ALLOWVOLUMEEXPANSION   AGE
standard(default)    cinder.csi.openstack.org   Delete          WaitForFirstConsumer   true
46h
standard-csi         kubernetes.io/cinder       Delete          WaitForFirstConsumer   true
46h
```

2. 以下の例に示されるように、デフォルトストレージクラスについてアノテーション **storageclass.kubernetes.io/is-default-class** の値を **false** に変更します。

```
$ oc patch storageclass standard -p '{"metadata": {"annotations":
{"storageclass.kubernetes.io/is-default-class": "false"}}}'
```

3. アノテーションを追加するか、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-csi -p '{"metadata": {"annotations":
{"storageclass.kubernetes.io/is-default-class": "true"}}}'
```

4. デフォルトで PVC が CSI ストレージクラスを参照していることを確認します。

```
$ oc get storageclass
```

出力例

```
NAME                 PROVISIONER             RECLAIMPOLICY   VOLUMEBINDINGMODE
ALLOWVOLUMEEXPANSION   AGE
standard             kubernetes.io/cinder    Delete          WaitForFirstConsumer   true
46h
standard-csi(default)  cinder.csi.openstack.org   Delete          WaitForFirstConsumer   true
46h
```

5. オプション: ストレージクラスを指定することなく新規 PVC を定義できます。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: cinder-claim
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

特定のストレージクラスを指定しない PVC は、デフォルトのストレージクラスを使用して自動的にプロビジョニングされます。

6. オプション: 新規ファイルを設定した後に、クラスター内にこのファイルを作成します。

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

関連情報

- CSI ボリュームの設定

5.17. OPENSTACK MANILA CSI DRIVER OPERATOR

5.17.1. 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。

Container Storage Interface (CSI) Operator およびドライバーを使用する場合、永続ストレージ および CSI ボリュームの設定 について理解しておくことが推奨されます。

Manila ストレージアセットにマウントされる CSI でプロビジョニングされる PV を作成するには、OpenShift Container Platform は Manila CSI Driver Operator および Manila CSI ドライバーを Manila サービスが有効にされている OpenStack クラスターにデフォルトでインストールします。

- **Manila CSI Driver Operator** は、利用可能なすべての Manila 共有タイプの PVC の作成に必要なストレージクラスを作成します。Operator は **openshift-cluster-csi-drivers** namespace にインストールされます。
- **Manila CSI ドライバー** を使用すると、Manila PV を作成し、マウントできます。ドライバーは **openshift-manila-csi-driver** namespace にインストールされます。

5.17.2. 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。

5.17.3. Manila CSI Driver Operator の制限事項

次の制限は、Manila Container Storage Interface (CSI) Driver Operator に適用されます。

NFS のみがサポートされています

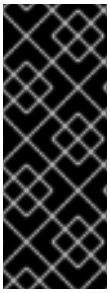
OpenStack Manila は、NFS、CIFS、CEPHFS など、多くのネットワーク接続ストレージプロトコルをサポートしており、これらは OpenStack クラウドで選択的に有効にすることができます。OpenShift Container Platform の Manila CSI Driver Operator は、NFS プロトコルの使用のみをサポートします。基盤となる OpenStack クラウドで NFS が利用可能でなく、有効化されていない場合は、Manila CSI Driver Operator を使用して OpenShift Container Platform のストレージをプロビジョニングすることはできません。

バックエンドが CephFS-NFS の場合、スナップショットはサポートされません

永続ボリューム (PV) のスナップショットを作成し、ボリュームをスナップショットに戻すには、使用している Manila 共有タイプがこれらの機能をサポートしていることを確認する必要があります。Red Hat OpenStack 管理者は、使用するストレージクラスに関連付けられた共有タイプで、スナップショットのサポート (**share type extra-spec snapshot_support**) およびスナップショットからの共有の作成 (**share type extra-spec create_share_from_snapshot_support**) を有効にする必要があります。

FSGroup はサポートされていません

Manila CSI は、複数のリーダーおよび複数のライターによるアクセス用の共有ファイルシステムを提供するため、FSGroup の使用をサポートしていません。これは、ReadWriteOnce アクセスモードで作成された永続ボリュームにも当てはまります。したがって、Manila CSI Driver で使用するために手動で作成するストレージクラスでは、**fsType** 属性を指定しないことが重要です。



重要

Red Hat OpenStack Platform 16.x および 17.x では、NFS を介した CephFS を使用する Shared File Systems サービス (Manila) は、Manila CSI を介した OpenShift Container Platform への共有の提供を完全にサポートします。ただし、このソリューションは大規模なスケールを意図したものではありません。CephFS NFS Manila-CSI Workload Recommendations for Red Hat OpenStack Platform の重要な推奨事項を確認してください。

5.17.4. 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 共有タイプ別にストレージクラス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。

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。

注記

Manila サービスはオプションです。サービスが Red Hat OpenStack Platform (RHOSP) で有効にされていない場合には、Manila CSI ドライバーがインストールされず、Manila のストレージクラスが作成されません。

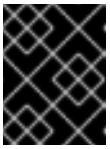
前提条件

- 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。

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** オブジェクトを使用し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。

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 ストレージクラスは 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。

関連情報

- CSI ボリュームの設定

5.18. RED HAT VIRTUALIZATION CSI DRIVER OPERATOR

5.18.1. 概要

OpenShift Container Platform は、Red Hat Virtualization (RHV) 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。

Container Storage Interface (CSI) Operator およびドライバーを使用する場合は、永続ストレージ および CSI ボリュームの設定 を理解しておくことが推奨されます。

RHV 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る PV を作成するには、OpenShift Container Platform は **openshift-cluster-csi-drivers** namespace にデフォルトで oVirt CSI ドライバーおよび oVirt CSI ドライバーをインストールします。

- **oVirt CSI Driver Operator** は、永続ボリューム要求 (PVC) の作成に使用できるデフォルトの **StorageClass** オブジェクトを提供します。

- **oVirt CSI ドライバー** を使用すると、oVirt PV を作成し、マウントできます。

5.18.2. 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。

注記

oVirt CSI ドライバーは、スナップショットをサポートしていません。

5.18.3. Red Hat Virtualization (RHV) CSI ドライバーストレージクラス

OpenShift Container Platform は、動的にプロビジョニングされる永続ボリュームを作成するために使用される **ovirt-csi-sc** という名前のタイプが **StorageClass** のデフォルトオブジェクトを作成します。

異なる設定の追加ストレージクラスを作成するには、以下のサンプル YAML で記述される **StorageClass** オブジェクトを使用してファイルを作成し、保存します。

ovirt-storageclass.yaml

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: <storage_class_name> 1
  annotations:
    storageclass.kubernetes.io/is-default-class: "<boolean>" 2
provisioner: csi.ovirt.org
allowVolumeExpansion: <boolean> 3
reclaimPolicy: Delete 4
volumeBindingMode: Immediate 5
parameters:
  storageDomainName: <rhv-storage-domain-name> 6
  thinProvisioning: "<boolean>" 7
  csi.storage.k8s.io/fstype: <file_system_type> 8
```

1. ストレージクラス名
2. ストレージクラスがクラスターのデフォルトストレージクラスの場合に **false** に設定されます。**true** に設定される場合、既存のデフォルトストレージクラスを編集し、**false** に設定する必要があります。
3. **true** は動的ボリューム拡張を有効にし、**false** はこれを防ぎます。**true** が推奨されます。
4. このストレージクラスの動的にプロビジョニングされる永続ボリュームは、この回収ポリシーで作成されます。このデフォルトポリシーは **Delete** です。
5. **PersistentVolumeClaims** をプロビジョニングし、バインドする方法を示します。設定されていない場合は、**VolumeBindingImmediate** が使用されます。このフィールドは、**VolumeScheduling** 機能を有効にするサーバーによってのみ適用されます。

6 使用する RHV ストレージドメイン名。

7 **true** の場合、ディスクはシンプロビジョニングされます。**false** の場合、ディスクは事前割り当てされます。シンプロビジョニングが推奨されています。

8 オプション: 作成するファイルシステムタイプ。使用できる値は **ext4**(デフォルト) または **xfs** です。

5.18.4. RHV での永続ボリュームの作成

PersistentVolumeClaim (PVC) オブジェクトの作成時に、OpenShift Container Platform は新規の永続ボリューム (PV) をプロビジョニングし、**PersistentVolume** オブジェクトを作成します。

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。
- **ovirt-credentials** シークレットに正しい RHV 認証情報を指定している。
- oVirt CSI ドライバーをインストールしている。
- 1 つ以上のストレージクラスが定義されている。

手順

- Web コンソールを使用して RHV で永続ボリュームを動的に作成する場合は、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。
  4. 適切な **StorageClass** オブジェクト (デフォルトは **ovirt-csi-sc**) を選択します。
  5. ストレージ要求の一意の名前を入力します。
  6. アクセスモードを選択します。現時点で、RWO (ReadWriteOnce) は唯一のサポートされているアクセスモードです。
  7. ストレージ要求のサイズを定義します。
  8. ボリュームモードを選択します。
     Filesystem: Pod にディレクトリーとしてマウントされます。このモードはデフォルトです。

     Block: ファイルシステムのないブロックデバイスです。
  9. **Create** をクリックして **PersistentVolumeClaim** オブジェクトを作成し、**PersistentVolume** オブジェクトを生成します。
- コマンドラインインターフェイス (CLI) を使用して RHV 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. 以下のサンプル YAML によって記述される **PersistentVolumeClaim** オブジェクトを使用

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

pvc-ovirt.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-ovirt
spec:
  storageClassName: ovirt-csi-sc 1
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: <volume size> 2
  volumeMode: <volume mode> 3
```

1. 必要なストレージクラスの名前。
2. ボリュームのサイズ (GiB)。
3. サポートされているオプション:
   - **Filesystem**: Pod にディレクトリーとしてマウントされます。このモードはデフォルトです。
   - **Block**: ファイルシステムのないブロックデバイスです。

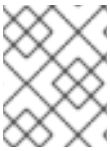
2. 以下のコマンドを実行して、直前の手順で保存されたオブジェクトを作成します。

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

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-ovirt
```

pvc-manila は、これが Bound であることを示します。



注記

Operator 認証情報を更新する必要がある場合は、How to modify the RHV credentials in OCP 4 の手順を参照してください。

関連情報

- CSI ボリュームの設定
- 動的プロビジョニング

5.19. VMWARE VSPHERE CSI ドライバー OPERATOR

5.19.1. 概要

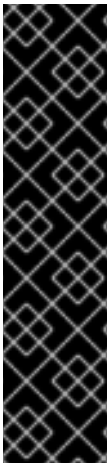
OpenShift Container Platform は、Virtual Machine Disk (VMDK) ボリュームの永続ディスク (PD) ストレージの Container Storage Interface (CSI) VMware vSphere ドライバーを使用して永続ボリューム (PV) をプロビジョニングできます。

CSI Operator およびドライバーを使用する場合は、永続ストレージ および CSI ボリュームの設定 について理解しておくことが推奨されます。

vSphere 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 (PV) を作成するには、OpenShift Container Platform は、デフォルトで vSphere CSI Driver Operator および vSphere CSI ドライバーを **openshift-cluster-csi-drivers** namespace にインストールします。

- **vSphere CSI Driver Operator**: Operator は、永続ボリューム要求 (PVC) の作成に使用できる **thin-csi** というストレージクラスを提供します。vSphere CSI ドライバー Operator は、ストレージボリュームをオンデマンドで作成できるようにし、クラスター管理者がストレージを事前にプロビジョニングする必要がなくすことで、動的ボリュームのプロビジョニングをサポートします。

- **vSphere CSI ドライバー**: このドライバーを使用すると、vSphere PV を作成し、マウントできます。OpenShift Container Platform 4.12.21 以降では、ドライバーのバージョンは 2.7.1 です。4.12.21 より前の OpenShift Container Platform 4.12 バージョンでは、バージョンは 2.6.1 です。vSphere CSI ドライバーは、XFS や Ext4 など、基盤となる Red Hat Core OS リリースでサポートされるすべてのファイルシステムをサポートします。サポートされているファイルシステムの詳細は、利用可能なファイルシステムの概要 を参照してください。



重要

OpenShift Container Platform は、vSphere 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 または CSI 以外のドライバーの使用に設定されます。

今後の OpenShift Container Platform バージョンでは、既存の in-tree プラグインを使用してプロビジョニングされるボリュームは、同等の CSI ドライバーに移行される予定です。CSI 自動移行はシームレスに行ってください。移行をしても、永続ボリューム、永続ボリューム要求、ストレージクラスなどの既存の API オブジェクトを使用する方法は変更されません。移行の詳細は、CSI の自動移行 を参照してください。

完全な移行後、in-tree プラグインは最終的に OpenShift Container Platform の今後のバージョンで削除されます。

5.19.2. 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。

5.19.3. vSphere CSI の制限

次の制限は、Manila Container Storage Interface (CSI) Driver Operator に適用されます。

- vSphere CSI Driver は、動的および静的なプロビジョニングをサポートします。PV 仕様で静的プロビジョニングを使用する場合、**csi.volumeAttributes** でキー **storage.kubernetes.io/csiProvisionerIdentity** を使用しないでください。このキーは動的にプ

ロビジョニングされた PV を示すためです。

- vSphere クライアントインターフェイスを使用したデータストア間での永続的なコンテナーボリュームの移行は、OpenShift Container Platform ではサポートされていません。

5.19.4. vSphere ストレージポリシー

vSphere CSI Driver Operator ストレージクラスは、vSphere のストレージポリシーを使用します。OpenShift Container Platform は、クラウド設定で設定されるデータストアをターゲットにするストレージポリシーを自動的に作成します。

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: thin-csi
provisioner: csi.vsphere.vmware.com
parameters:
  StoragePolicyName: "$openshift-storage-policy-xxxx"
volumeBindingMode: WaitForFirstConsumer
allowVolumeExpansion: false
reclaimPolicy: Delete
```

5.19.5. ReadWriteMany vSphere ボリュームのサポート

基盤となる vSphere 環境が vSAN ファイルサービスをサポートしている場合、OpenShift Container Platform によってインストールされた vSphere Container Storage Interface (CSI) Driver Operator は ReadWriteMany (RWX) ボリュームのプロビジョニングをサポートします。vSAN ファイルサービスが設定されていない場合、使用可能なアクセスモードは ReadWriteOnce (RWO) のみです。vSAN ファイルサービスが設定されていない場合に RWX を要求すると、ボリュームの作成に失敗し、エラーがログに記録されます。

ご使用の環境で vSAN ファイルサービスを設定する方法について、詳しくは vSAN File Service を参照してください。

次の永続ボリューム要求 (PVC) を行うことで、RWX ボリュームを要求できます。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: myclaim
spec:
  resources:
    requests:
      storage: 1Gi
  accessModes:
    - ReadWriteMany
  storageClassName: thin-csi
```

RWX ボリュームタイプの PVC を要求すると、vSAN ファイルサービスによってサポートされる永続ボリューム (PV) がプロビジョニングされます。

5.19.6. VMware vSphere CSI Driver Operator の要件

vSphere CSI Driver Operator をインストールするには、次の要件を満たす必要があります。

- VMware vSphere バージョン: 7.0 Update 2 以降、8.0 Update 1 以降
- vCenter バージョン: 7.0 Update 2 以降、8.0 Update 1 以降
- ハードウェアバージョン 15 以降の仮想マシン
- クラスターにサードパーティーの vSphere CSI ドライバーがインストールされていない

サードパーティーの vSphere CSI ドライバーがクラスターに存在する場合、OpenShift Container Platform はそれを上書きしません。サードパーティーの vSphere CSI ドライバーが存在すると、OpenShift Container Platform を OpenShift Container Platform 4.13 以降にアップグレードできなくなります。

注記

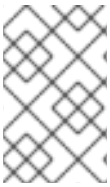
VMware vSphere CSI Driver Operator は、インストールマニフェストの **platform: vsphere** でデプロイされたクラスターでのみサポートされます。

サードパーティーの CSI ドライバーを削除するには、サードパーティーの vSphere CSI ドライバーの削除 を参照してください。

5.19.7. サードパーティー vSphere CSI Driver Operator の削除

OpenShift Container Platform 4.10 以降には、Red Hat がサポートする vSphere Container Storage Interface (CSI) Operator ドライバーの組み込みバージョンが含まれます。コミュニティーまたは別のベンダーが提供する vSphere CSI ドライバーをインストールした場合、OpenShift Container Platform の次のメジャーバージョン (4.13 以降など) への更新がクラスターで無効になる可能性があります。

OpenShift Container Platform 4.12 以降では、クラスターは引き続き完全にサポートされており、4.12.z などの 4.12 の z ストリームリリースの更新はブロックされませんが、OpenShift Container Platform の次のメジャーバージョンに更新する前に、サードパーティーの vSphere CSI Driver ドライバーを削除してこの状態を修正する必要があります。サードパーティーの vSphere CSI ドライバーの削除には、関連する永続ボリューム (PV) オブジェクトの削除が必要ないため、データ喪失は発生しません。



注記

以下の手順は完全ではない可能性があるため、ベンダーまたはコミュニティープロバイダーのアンインストールガイドを参照して、ドライバーおよびコンポーネントを完全に削除してください。

サードパーティーの vSphere CSI Driver をアンインストールするには、以下を実行します。

1. サードパーティーの vSphere CSI Driver(VMware vSphere Container Storage プラグイン) の Deployment および Daemonset オブジェクトを削除します。
2. サードパーティーの vSphere CSI Driver で以前にインストールされた configmap およびシークレットオブジェクトを削除します。
3. サードパーティーの vSphere CSI ドライバー **CSIDriver** オブジェクトを削除します。

```
~ $ oc delete CSIDriver csi.vsphere.vmware.com
```

```
csidriver.storage.k8s.io "csi.vsphere.vmware.com" deleted
```

OpenShift Container Platform クラスターからサードパーティーの vSphere CSI Driver を削除した後に、Red Hat の vSphere CSI Driver Operatorr のインストールが自動的に再開され、OpenShift Container Platform 4.11 以降へのアップグレードをブロックする可能性のある条件は自動的に削除されます。既存の vSphere CSI PV オブジェクトがある場合、それらのライフサイクルは Red Hat の vSphere CSI Driver Operator で管理されるようになります。

5.19.8. vSphere CSI トポロジーの設定

OpenShift Container Platform は、異なるゾーンおよびリージョンに OpenShift Container Platform for vSphere をデプロイする機能を提供します。これにより、複数のコンピューティングクラスターにデプロイできるため、単一障害点を回避するのに役立ちます。

注記

OpenShift Container Platform on vSphere は、複数のデータセンターをサポートしていません。

これは、vCenter でゾーンとリージョンのカテゴリーを定義し、これらのゾーンとリージョンのカテゴリーのタグを作成して、コンピューティングクラスターなどのさまざまな障害ドメインにこれらのカテゴリーを割り当てることによって実現されます。適切なカテゴリーを作成し、vCenter オブジェクトにタグを割り当てたら、それらの障害ドメインで Pod のスケジュールを担当する仮想マシン (VM) を作成する追加のマシンセットを作成できます。

手順

1. VMware vCenter vSphere クライアント GUI で、適切なゾーンとリージョンのカテゴリーとタグを定義します。
   vSphere では任意の名前でカテゴリーを作成できますが、OpenShift Container Platform では、トポロジーを定義するために **openshift-region** および **openshift-zone** 名を使用することを強く推奨します。

   次の例では、1 つのリージョンと 2 つのゾーンを持つ 2 つの障害ドメインを定義しています。

   表5.4 1つのリージョンと 2 つのゾーンを持つ vSphere トポロジー

計算クラスター	障害ドメイン	説明
コンピューティングクラスター: ocp1、データセンター: アトランタ	openshift-region: us-east-1 (タグ)、openshift-zone: us-east-1a (タグ)	これにより、リージョン us-east-1 にゾーン us-east-1a を持つ障害ドメインが定義されます。
コンピュータークラスター: ocp2、データセンター: アトランタ	openshift-region: us-east-1 (タグ)、openshift-zone: us-east-1b (タグ)	これにより、us-east-1b と呼ばれる同じリージョン内の別の障害ドメインが定義されます。

   vSphere のカテゴリーとタグの詳細については、VMware vSphere のドキュメントを参照してください。

2. Container Storage Interface (CSI) ドライバーがこのトポロジーを検出できるようにするには、**clusterCSIDriver** オブジェクトの YAML ファイルの **driverConfig** セクションを編集します。

- 以前に作成した **openshift-zone** および **openshift-region** カテゴリーを指定します。
- **driverType** を **vSphere** に設定します。

```
~ $ oc edit clustercsidriver csi.vsphere.vmware.com -o yaml
```

出力例

```
apiVersion: operator.openshift.io/v1
kind: ClusterCSIDriver
metadata:
 name: csi.vsphere.vmware.com
spec:
 logLevel: Normal
 managementState: Managed
 observedConfig: null
 operatorLogLevel: Normal
 unsupportedConfigOverrides: null
 driverConfig:
  driverType: vSphere (1)
    vSphere:
      topologyCategories: (2)
      - openshift-zone
      - openshift-region
```

(1) **driverType** が **vSphere** に設定されていることを確認します。

(2) vCenter で以前に作成された **openshift-zone** および **openshift-region** カテゴリー。

3. 次のコマンドを実行して、**CSINode** オブジェクトにトポロジーキーがあることを確認します。

```
~ $ oc get csinode
```

出力例

```
NAME                    DRIVERS   AGE
co8-4s88d-infra-2m5vd     1         27m
co8-4s88d-master-0        1         70m
co8-4s88d-master-1        1         70m
co8-4s88d-master-2        1         70m
co8-4s88d-worker-j2hmg    1          47m
co8-4s88d-worker-mbb46    1          47m
co8-4s88d-worker-zlk7d    1         47m
```

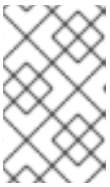
```
~ $ oc get csinode co8-4s88d-worker-j2hmg -o yaml
```

出力例

```
...
spec:
   drivers:
   - allocatable:
```

```
    count: 59
name: csi-vsphere.vmware.com
nodeID: co8-4s88d-worker-j2hmg
topologyKeys: 1
- topology.csi.vmware.com/openshift-zone
- topology.csi.vmware.com/openshift-region
```

1 vSphere **openshift-zone** および **openshift-region** カテゴリーからのトポロジーキー。



注記

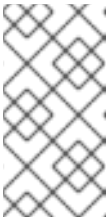
CSINode オブジェクトは、更新されたトポロジー情報を受信するのに時間がかかる場合があります。ドライバーが更新された後、**CSINode** オブジェクトにはトポロジーキーが含まれている必要があります。

4. 障害ドメイン全体のデータストアに割り当てるタグを作成します。
OpenShift Container Platform が複数の障害ドメインにまたがる場合、データストアがそれらの障害ドメイン間で共有されない可能性があります。この場合、永続ボリューム (PV) のトポロジーを意識したプロビジョニングが役立ちます。

   a. vCenter で、データストアにタグを付けるためのカテゴリーを作成します。例: **openshift-zonal-datastore-cat**。カテゴリーが OpenShift Container Platform クラスターに参加するデータストアのタグ付けに一意に使用される場合、他のカテゴリー名を使用できます。また、作成したカテゴリーの関連付け可能なエンティティーとして **StoragePod**、**Datastore**、および **Folder** が選択されていることを確認します。

   b. vCenter で、以前に作成したカテゴリーを使用するタグを作成します。この例では、タグ名 **openshift-zonal-datastore** を使用しています。

   c. 以前に作成したタグ (この例では **openshift-zonal-datastore**) を、動的プロビジョニングと見なされる障害ドメイン内の各データストアに割り当てます。



注記

カテゴリーとタグには、好きな名前を使用できます。この例で使用されている名前は、推奨事項として提供されています。定義するタグとカテゴリーが、OpenShift Container Platform クラスター内のすべてのホストと共有されるデータストアのみを一意に識別するようにします。

5. 各障害ドメインのタグベースのデータストアを対象とするストレージポリシーを作成します。

   a. vCenter で、メインメニューから **Policies and Profiles** をクリックします。

   b. **Policies and Profiles** ページのナビゲーションペインで、**VM Storage Policies** をクリックします。

   c. **CREATE** をクリックします。

   d. ストレージポリシーの名前を入力します。

   e. ルールについては、Tag Placement rules を選択し、目的のデータストアを対象とするタグとカテゴリーを選択します (この例では、**openshift-zonal-datastore** タグ)。
   データストアは、ストレージ互換性テーブルにリストされています。

6. 新しいゾーンストレージポリシーを使用する新しいストレージクラスを作成します。

   a. **Storage** > **StorageClasses** をクリックします。

   b. **StorageClasses** ページで、**Create StorageClass** をクリックします。

   c. **Name** に新しいストレージクラスの名前を入力します。

   d. **Provisioner** で、**csi.vsphere.vmware.com** を選択します。

   e. **Additional parameters** で、StoragePolicyName パラメーターの **Value** を、前に作成した新しいゾーンストレージポリシーの名前に設定します。

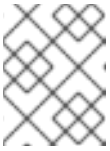
   f. **Create** をクリックします。

   出力例

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: zoned-sc (1)
provisioner: csi.vsphere.vmware.com
parameters:
  StoragePolicyName: zoned-storage-policy (2)
reclaimPolicy: Delete
allowVolumeExpansion: true
volumeBindingMode: WaitForFirstConsumer
```

(1) 新しいトポロジー対応ストレージクラス名。

(2) ゾーンストレージポリシーを指定します。



注記

前述の YAML ファイルを編集し、コマンド **oc create -f $FILE** を実行して、ストレージクラスを作成することもできます。

結果

トポロジー対応ストレージクラスからの永続ボリュームクレーム (PVC) と PV の作成は完全にゾーンであり、Pod のスケジュール方法に応じて、それぞれのゾーンでデータストアを使用する必要があります。

```
~ $ oc get pv <pv-name> -o yaml
```

出力例

```
...
nodeAffinity:
  required:
    nodeSelectorTerms:
    - matchExpressions:
      - key: topology.csi.vmware.com/openshift-zone (1)
```

```
        operator: In
        values:
        - <openshift-zone>
      -key: topology.csi.vmware.com/openshift-region (2)
        operator: In
        values:
        - <openshift-region>
...
peristentVolumeclaimPolicy: Delete
storageClassName: <zoned-storage-class-name> (3)
volumeMode: Filesystem
...
```

(1) (2) PV にはゾーン化されたキーがあります。

(3) PV はゾーンストレージクラスを使用しています。

関連情報

- VMware vSphere タグのドキュメント

5.19.9. 関連情報

- CSI ボリュームの設定

第6章 汎用的な一時ボリューム

6.1. 概要

汎用一時ボリュームは、永続ボリュームおよび動的プロビジョニングをサポートするすべてのストレージドライバーが提供できる一時ボリュームの一種です。汎用の一時ボリュームは、スクラッチデータ用に Pod ごとのディレクトリー (通常、プロビジョニング後は空) を提供する点で **emptyDir** ボリュームと類似しています。

汎用の一時ボリュームは Pod 仕様に準拠して指定され、Pod のライフサイクルに従います。これらは Pod と共に作成され、削除されます。

汎用の一時ボリュームには、以下の特徴があります。

- ストレージは、ローカルまたはネットワーク接続タイプとすることができます。
- ボリュームには、Pod が超過できない固定サイズを指定できます。
- ドライバーおよびパラメーターによっては、ボリュームに特定の初期データが含まれる場合があります。
- ドライバーがサポートしていれば、スナップショットの作成、クローンの作成、サイズ変更、ストレージ容量の追跡など、ボリュームに対する一般的な操作がサポートされます。

注記

汎用の一時ボリュームは、オフラインのスナップショットやサイズ変更をサポートしません。

この制約により、以下の Container Storage Interface (CSI) ドライバーは、以下の汎用一時ボリューム機能をサポートしません。

- Azure Disk CSI ドライバーは、サイズ変更をサポートしません。
- Cinder CSI ドライバーは、スナップショットをサポートしません。

6.2. ライフサイクルおよび永続ボリューム要求

ボリューム要求のパラメーターは Pod のボリュームソース内で許可されます。ラベル、アノテーション、および永続ボリューム要求 (PVC) のフィールドの全セットがサポートされます。このような Pod が作成されると、一時ボリュームコントローラーは (**汎用一時ボリュームの作成** の手順に示すテンプレートから) Pod と同じ namespace に実際の PVC オブジェクトを作成し、Pod が削除されると PVC が削除されるようにします。これがトリガーとなり、以下の 2 つの方法のいずれかでボリュームのバインディングおよびプロビジョニングが行われます。

- 直ちに (ストレージクラスが即時ボリュームバインディングを使用する場合は)
  即時バインディングの場合、スケジューラーはボリュームが利用可能になった後にボリュームにアクセスできるノードを強制的に選択させられます。
- Pod が一時的にノードにスケジュールされる場合 (**WaitForFirstConsumervolume** バインディングモード)
  スケジューラーは Pod に適したノードを選択できるため、このボリュームバインディングオプションは、汎用一時ボリュームに推奨されます。

リソースの所有権に関しては、汎用一時ストレージを持つ Pod は、その一時ストレージを提供する

PVC の所有者となります。Pod が削除されると、Kubernetes ガベージコレクターによって PVC が削除され、ストレージクラスのデフォルトの再利用ポリシーがボリュームを削除することになっているため、通常はボリュームの削除がトリガーされます。回収ポリシーが保持のストレージクラスを使用して、準一時ローカルストレージを作成できます。Pod が削除されてもストレージは存続するため、この場合は別途ボリュームのクリーンアップが行われるようにする必要があります。これらの PVC は存在しますが、それらは他の PVC と同様に使用できます。特に、それらはボリュームのクローン作成またはスナップショット作成時にデータソースとして参照できます。PVC オブジェクトは、ボリュームの現在のステータスも保持します。

関連情報

- 汎用一時ボリュームの作成

6.3. セキュリティー

汎用一時ボリューム機能を有効にすると、Pod を作成できるユーザーが永続ボリューム要求 (PVC) も間接的に作成できるようになります。この機能は、これらのユーザーが PVC を直接作成するパーミッションを持たない場合でも機能します。クラスター管理者はこれを認識している必要があります。これがセキュリティーモデルに適さない場合は、汎用的な一時ボリュームを持つ Pod などのオブジェクトを拒否する容認 Webhook を使用します。

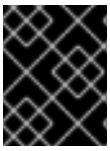
PVC に対する通常の namespace クォータがそのまま適用されるため、この新しいメカニズムを使用できる場合でも新しいメカニズムを使用して他のポリシーを回避できません。

6.4. 永続ボリューム要求の命名

自動的に作成される永続ボリューム要求 (PVC) には、Pod 名とボリューム名を組み合わせ、間にハイフン (-) を挿入した名前が付けられます。この命名規則では、異なる Pod 間および Pod と手動で作成された PVC 間で競合が生じる可能性があります。

たとえば、**pod-a** とボリューム **scratch** の組み合わせと、**pod** とボリューム **a-scratch** の組み合わせは、どちらも同じ PVC 名 **pod-a-scratch** になります。

このような競合は検出され、Pod 用に作成された場合にのみ、PVC は一時ボリュームに使用されます。このチェックは所有者の関係に基づいています。既存の PVC は上書きまたは変更されませんが、競合は解決されません。適切な PVC がないと、Pod は起動できません。



重要

同じ namespace 内で Pod とボリュームに名前を付ける際には、命名の競合が発生しないように注意してください。

6.5. 汎用一時ボリュームの作成

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. ファイルに汎用一時ボリュームの情報を追加します。

my-example-pod-with-generic-vols.yaml

```
kind: Pod
```

```
apiVersion: v1
metadata:
  name: my-app
spec:
  containers:
    - name: my-frontend
      image: busybox:1.28
      volumeMounts:
      - mountPath: "/mnt/storage"
        name: data
      command: [ "sleep", "1000000" ]
  volumes:
    - name: data (1)
      ephemeral:
        volumeClaimTemplate:
          metadata:
            labels:
              type: my-app-ephvol
          spec:
            accessModes: [ "ReadWriteOnce" ]
            storageClassName: "gp2-csi"
            resources:
              requests:
                storage: 1Gi
```

(1) 汎用一時ボリューム要求

第7章 永続ボリュームの拡張

7.1. 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、以下のコマンドを実行して **allowVolumeExpansion** 属性を追加します。

```
$ oc edit storageclass <storage_class_name> 1
```

1 ストレージクラスの名前を指定します。

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。

7.2. 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。

CSI ボリューム拡張は、以下をサポートしません。

- ボリューム拡張時の障害からの復旧
- 縮小

前提条件

- 基礎となる CSI ドライバーがサイズ変更をサポートする。
- 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。詳細は、ボリューム拡張サポートの有効化を参照してください。

手順

1. 永続ボリューム要求 (PVC) の場合は、**.spec.resources.requests.storage** を必要な新しいサイズに設定します。

2. PVC の **status.conditions** フィールドを監視し、サイズ変更が完了したかどうかを確認します。OpenShift Container Platform は、拡張時に **Resizing** 条件を PVC に追加します。これは、拡張の完了後に削除されます。

7.3. 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

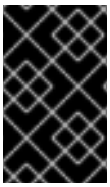
- FlexVolume プラグインのサイズ変更を使用するには、以下の方法で **ExpandableVolumePlugin** インターフェイスを実装する必要があります。

  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 はコントロールプレーンノードへの 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。

7.4. ローカルボリュームの拡張

ローカルストレージ Operator (LSO) を使用して作成された永続ボリューム (PV) および永続ボリューム要求 (PVC) を手動で拡張できます。

手順

1. 基礎となるデバイスを拡張します。これらのデバイスで適切な容量が利用できるようにします。
2. PV の **.spec.capacity** フィールドを編集して、新しいデバイスサイズに一致するように対応する PV オブジェクトを更新します。
3. PVC を PVet にバインドするためのストレージクラスに **allowVolumeExpansion:true** を設定します。
4. PVC に新しいサイズに一致するように **.spec.resources.requests.storage** を設定します。

Kubelet は、ボリューム上の基礎となるファイルシステムを自動的に拡張するはずです。必要に応じて、新しいサイズを反映するように PVC の status フィールドを更新します。

7.5. 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、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。まず、クラウドプロバイダーのボリュームオブジェクトを拡張します。次に、ノードの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る。

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できます。Pod が実行している場合は、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。

7.6. 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。

3. PV を手動で編集し、PV 仕様から **claimRef** エントリーを削除して、新しく作成された PVC を **Retain** とマークされた PV にバインドできるようにします。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。

関連情報

- 制御する側の **StorageClass** オブジェクトには、**allowVolumeExpansion** が **true** に設定されています (ボリューム拡張サポートの有効化 を参照)。

第8章 動的プロビジョニング

8.1. 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。

8.2. 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。

8.3. 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。

8.3.1. 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: <storage-class-name> 3
  annotations: 4
    storageclass.kubernetes.io/is-default-class: 'true'
    ...
provisioner: kubernetes.io/aws-ebs 5
parameters: 6
  type: gp3
...
```

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。

8.3.2. 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。ただし、クラスターには複数のストレージクラスを設定できますが、それらのうちの 1 つのみをデフォルトのストレージクラスにすることができます。

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
...
```

8.3.3. RHOSP Cinder オブジェクトの定義

cinder-storageclass.yaml

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: <storage-class-name> 1
provisioner: kubernetes.io/cinder
parameters:
  type: fast 2
  availability: nova 3
  fsType: ext4 4
```

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. Cinder で作成されるボリュームタイプ。デフォルトは空です。
3. 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。
4. 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。

8.3.4. 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。

8.3.5. 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: <storage-class-name> 1
provisioner: kubernetes.io/aws-ebs
parameters:
  type: io1 2
```

```
  iopsPerGB: "10" 3
  encrypted: "true" 4
  kmsKeyId: keyvalue 5
  fsType: ext4 6
```

1. (必須) ストレージクラスの名前。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. (必須) **io1**、**gp3**、**sc1**、**st1** から選択します。デフォルトは **gp3** です。有効な Amazon Resource Name (ARN) 値は、AWS のドキュメント を参照してください。
3. (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。詳細は、AWS のドキュメント を参照してください。
4. (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。
5. (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 値は、AWS のドキュメント を参照してください。
6. (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。

8.3.6. Azure Disk オブジェクト定義

azure-advanced-disk-storageclass.yaml

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: <storage-class-name> 1
provisioner: kubernetes.io/azure-disk
volumeBindingMode: WaitForFirstConsumer 2
allowVolumeExpansion: true
parameters:
  kind: Managed 3
  storageaccounttype: Premium_LRS 4
reclaimPolicy: Delete
```

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. **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。
3. 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 はマネージド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マネージド外ディスクの両方の使用を許可しないため、**Shared** または **Dedicated** で作成されたマネージド外ディスクを OpenShift Container Platform ノードに割り当てることはできません。

4 Azure ストレージアカウントの SKU 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 **Standard_LRS** ディスクのみを、マネージド VM はマネージドディスクのみを、アンマネージド VM はアンマネージドディスクのみを割り当てることができます。

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。

8.3.7. 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>
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 **Standard_LRS** などの Azure ストレージアカウントの SKU 層。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。

8.3.7.1. 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。

8.3.8. 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: <storage-class-name> 1
provisioner: kubernetes.io/gce-pd
parameters:
  type: pd-standard 2
  replication-type: none
volumeBindingMode: WaitForFirstConsumer
allowVolumeExpansion: true
reclaimPolicy: Delete
```

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. **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。

8.3.9. VMWare vSphere オブジェクトの定義

vsphere-storageclass.yaml

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: <storage-class-name> 1
provisioner: kubernetes.io/vsphere-volume 2
parameters:
  diskformat: thin 3
```

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 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。

3 **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。

8.4. デフォルトストレージクラスの変更

この手順を使用して、デフォルトのストレージクラスを変更します。たとえば、**gp3** と **standard** の 2 つのストレージクラスがあり、デフォルトのストレージクラスを **gp3** から **standard** に変更する必要がある場合などです。

手順

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
gp3 (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 gp3 -p '{"metadata": {"annotations": {"storageclass.kubernetes.io/is-default-class": "false"}}}'
```

3. **storageclass.kubernetes.io/is-default-class** アノテーションを **true** に設定して、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
gp3                  kubernetes.io/aws-ebs
standard (default)   kubernetes.io/aws-ebs
```