Red Hat OpenShift Service on AWS 4

ストレージ

Red Hat OpenShift Service on AWS クラスターのストレージの設定

Last Updated: 2025-12-06

Red Hat OpenShift Service on AWS 4 ストレージ

Red Hat OpenShift Service on AWS クラスターのストレージの設定

Legal Notice

Abstract

このドキュメントでは、Red Hat OpenShift Service on AWS (ROSA) クラスターへのストレージの設定を説明します。

Table of Contents

第1章 RED HAT OPENSHIFT SERVICE ON AWS ストレージの概要

Red Hat OpenShift Service on AWS は、Amazon Elastic Block Store (Amazon EBS) および Amazon Elastic File System (Amazon EFS) ストレージをサポートします。Red Hat OpenShift Service on AWS クラスターでは、永続データと非永続データのコンテナーストレージを管理できます。

1.1. RED HAT OPENSHIFT SERVICE ON AWS 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、ボリュームの機能を表すものです。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)

config map

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。

fsGroup

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

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。

ネストされたマウントポイント

ネストされたマウントポイントは、前のボリュームで作成されたマウントポイントを使用しようとするマウントポイントです。

ネストされたマウントポイントを使用した Pod 定義の例

```
kind: Pod
apiVersion: v1
metadata:
  name: webapp
  labels:
    name: webapp
spec:
  containers:
    - name: webapp
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
      - mountPath: /mnt/web
        name: web
      - mountPath: /mnt/web/redis (1)
        name: redis
  volumes:
    - name: redis
      persistentVolumeClaim:
       claimName: "redis"
    - name: web
      persistentVolumeClaim:
        claimName: "web"
```

(1) ネストされたマウントポイント

Red Hat OpenShift Service on AWS ではマウントポイントの作成順序が不確定であるため、ネストされたマウントポイントを **使用しないでください**。使用した場合、競合状態や未定義の動作が発生する可能性があります。

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。Red Hat OpenShift Service on AWS 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)

Red Hat OpenShift Service on AWS 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

Red Hat OpenShift Service on AWS 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。Red Hat OpenShift Service on AWS 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

Red Hat OpenShift Service on AWS は、オンプレミスプロバイダーと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。Red Hat OpenShift Service on AWS クラスターで永続データと非永続データのコンテナーストレージを管理できます。

Storage class

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。

1.2. ストレージタイプ

Red Hat OpenShift Service on AWS ストレージは、一時ストレージと永続ストレージという 2 つのカテゴリーに大きく分類されます。

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。Red Hat OpenShift Service on AWS 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

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。

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退避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。

永続ボリュームとは異なり、エフェメラルストレージは構造化されておらず、スペースは、システム、コンテナーランタイム、および Red Hat OpenShift Service on AWS による他の使用に加えて、ノードで実行されているすべての Pod 間で共有されます。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、Red Hat OpenShift Service on AWS が必要に応じて Pod をスケジュールし、ローカルストレージの過剰な使用からノードを保護することもできます。

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。

2.2.1. 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。

2.2.2. 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。Red Hat OpenShift Service on AWS は、このパーティションへの分離とともに、共有アクセスを識別して提供しようとします。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**

2.3.1. 一時ストレージの制限と要求の単位

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。

重要

各バイト量の接尾辞では大文字と小文字が区別されます。必ず大文字と小文字を正しく使い分けてください。要求を 400 メガバイトに設定する場合は、"400M" のように、大文字の "M" を使用します。400 メビバイトを要求するには、大文字の "400Mi" を使用します。"400m" の一時ストレージを指定すると、ストレージが 0.4 バイトしか要求されません。

2.3.2. 一時ストレージの要求と制限の例

次のサンプル設定ファイルは、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。
- 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。
  - 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。

クォータと制限を含む一時ストレージ設定の例

```
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"
        limits:
          ephemeral-storage: "4Gi"
      volumeMounts:
      - name: ephemeral
        mountPath: "/tmp"
    volumes:
      - name: ephemeral
        emptyDir: {}
```

1. ローカル一時ストレージに対するコンテナーの要求。
2. ローカル一時ストレージに対するコンテナーの制限。

2.3.3. Pod のスケジューリングと退避に影響する一時ストレージ設定

Pod 仕様の設定は、スケジューラーが Pod のスケジュールを決定する方法と、kubelet が Pod を退避するタイミングの両方に影響します。

- 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、ノードの利用可能な一時ストレージ (割り当て可能なリソース) が 4 GiB を超える場合に限り、Pod をノードに割り当てることができます。
- 次に、最初のコンテナーによってリソース制限が設定されるため、kubelet 退避マネージャーがこのコンテナーのディスク使用量をコンテナーレベルで測定し、コンテナーのストレージ使用量がその制限 (4 GiB) を超えた場合に Pod を退避します。kubelet 退避マネージャーは、合計使用量が全体の Pod ストレージ制限 (8 GiB) を超えた場合にも、Pod に退避のマークを付けます。

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。

手順

- **/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/disk/by-partuuid/4cd1448a-01    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

コンテナーにデプロイされた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。{microshift-short} 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、ノード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を使用してストレージ要件を要求できます。

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。Red Hat OpenShift Service on AWS 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関する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 Red Hat OpenShift Service on AWS ノード全体で共有でき、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。

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、Red Hat OpenShift Service on AWS 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. 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。

3.2.5. 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 Red Hat OpenShift Service on AWS 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、Red Hat OpenShift Service on AWS の **PersistentVolume** オブジェクトと、Amazon Elastic Block Store (Amazon 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。

3.2.6. 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. 次のコマンドを実行して 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。

3.2.7. 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できます。

手順

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。

以下のコマンドを実行して、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_name> -o jsonpath='{.spec.claimRef.name}'
```

3.3.1. PV の種類

Red Hat OpenShift Service on AWS は、永続ボリューム用のストレージオプションとして、以下のものをサポートしています。

- AWS Elastic Block Store (EBS)。これはデフォルトでインストールされます。
- AWS Elastic File Store (EFS)

Red Hat OpenShift Service on AWS は、他のストレージベンダーの Kubernetes Container Storage Interface (CSI) 互換ボリュームプロビジョナーと連携して機能します。Red Hat OpenShift Service on AWS の CSI ドライバーの詳細は、**関連情報** セクションの **CSI ボリュームの設定** を参照してください。

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 の機能を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。

重要

ボリュームアクセスモードは、ボリュームの機能を表すものです。これらは強制される制約ではありません。リソースの無効な使用によって発生する実行時エラーについては、ストレージプロバイダーが責任を負います。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。

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。
ReadWriteOncePod	**RWOP**	ボリュームは、1 つのノード上の 1 つの Pod によって読み取り/書き込みとしてマウントできます。

表3.2 永続ボリュームでサポートされるアクセスモード

ボリュームプラグイン	ReadWriteOnce [1]	ReadWriteOncePod	ReadOnlyMany	ReadWriteMany
AWS EBS [2]	▮	▮		
AWS EFS	▮	▮	▮	▮
LVM Storage	▮	▮		

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

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

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. GCP hyperdisk-balanced ディスクの場合:
   - サポートされているアクセスモードは次のとおりです。
     - **ReadWriteOnce**
     - **ReadWriteMany**

- **ReadWriteMany** アクセスモードが有効になっているディスクでは、クローン作成とスナップショット作成は無効になっています。
- **ReadWriteMany** 内の単一の hyperdisk-balanced ディスクボリュームを最大 8 つのインスタンスにアタッチできます。
- すべてのインスタンスからディスクをデタッチした場合にのみ、**ReadWriteMany** でディスクのサイズを変更できます。
- その他の制限

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。
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。

3.3.4.1. 最後のフェーズの移行時間

LastPhaseTransitionTime フィールドには、永続ボリューム (PV) が別のフェーズ (**pv.Status.Phase**) に移行するたびに更新されるタイムスタンプが含まれます。PV の最後のフェーズ移行の時間を確認するには、次のコマンドを実行します。

```
$ oc get pv <pv_name> -o json | jq '.status.lastPhaseTransitionTime' 1
```

1 最後のフェーズの移行を確認する PV の名前を指定します。

3.3.4.2. 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)
- AWS Elastic File Storage (EFS)

関連情報

- CSI ボリュームの設定

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、デフォルトのストレージクラスをインストールします。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

3.4.5. PVC の使用状況に関する統計情報の表示

永続ボリューム要求 (PVC) 使用状況の統計情報を表示できます。

重要

PVC 使用状況の統計情報コマンドは、テクノロジープレビュー機能です。テクノロジープレビュー機能は、Red Hat 製品のサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、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。

3.4.5.1. PVC 使用状況の統計情報を表示するに、ユーザーパーミッションが必要

PVC 使用状況の統計情報を表示するには、適切な特権が必要です。

必要な特権でログオンする場合:

- 管理者特権がある場合は、管理者としてログオンしてください。
- 管理者特権が **ない** 場合:
  1. 次のコマンドを実行して、クラスターロールを作成し、ユーザーに追加します。

```
$ oc create clusterrole routes-view --verb=get,list --resource=routes
$ oc adm policy add-cluster-role-to-user routes-view <user-name> (1)
$ oc adm policy add-cluster-role-to-user cluster-monitoring-view <user-name> (2)
```

1 2 ユーザーの名前。

3.4.5.2. PVC の使用状況に関する統計情報の表示

- クラスター全体の統計情報を表示するには、次のコマンドを実行します。

```
$ oc adm top pvc -A
```

コマンド出力例

```
NAMESPACE     NAME         USAGE(%)
namespace-1   data-etcd-1  3.82%
namespace-1   data-etcd-0  3.81%
namespace-1   data-etcd-2  3.81%
namespace-2   mypvc-fs-gp3 0.00%
default       mypvc-fs     98.36%
```

- 指定された namespace の PVC 使用状況の統計情報を表示するには、次のコマンドを実行します。

```
$ oc adm top pvc -n <namespace-name> 1
```

1 **<namespace-name>** は、指定された namespace の名前に置き換えます。

コマンド出力例

```
NAMESPACE     NAME        USAGE(%)
namespace-1   data-etcd-2 3.81% 1
namespace-1   data-etcd-0 3.81%
namespace-1   data-etcd-1 3.82%
```

1 この例では、指定された namespace は **namespace-1** です。

- 指定された PVC および指定された namespace の使用状況の統計情報を表示するには、次のコマンドを実行します。

```
$ oc adm top pvc <pvc-name> -n <namespace-name> 1
```

1 **<pvc-name>** は指定された PVC の名前に、**<namespace-name>** は指定された namespace の名前に置き換えます。

コマンド出力例

```
NAMESPACE   NAME        USAGE(%)
namespace-1 data-etcd-0 3.81% 1
```

1 この例では、指定された namespace は **namespace-1** で、指定された PVC は **data-etcd-0** です。

3.5. ブロックボリュームのサポート

Red Hat OpenShift Service on AWS 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
Amazon Elastic Block Store (Amazon EBS)	▮	▮	▮
Amazon Elastic File Storage (Amazon EFS)			
LVM Storage	▮	▮	▮

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 を使用して POD のタイムアウトを減らす

ストレージボリュームに多数のファイル (たとえば、100 万以上) が含まれている場合、Pod のタイムアウトが発生する可能性があります。

これは、デフォルトで、Red Hat OpenShift Service on AWS が各ボリュームのコンテンツの所有権と権限を再帰的に変更して、そのボリュームがマウントされたときに Pod の **securityContext** で指定された **fsGroup** に一致させるために発生する可能性があります。多くのファイルが含まれるボリュームの場合、所有権と権限の確認と変更に時間がかかり、Pod の起動が遅くなる可能性があります。**securityContext** 内の **fsGroupChangePolicy** フィールドを使用して、Red Hat OpenShift Service on AWS がボリュームの所有権と権限をチェックおよび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** によって制御される所有権と権限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**: (デフォルト) 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。

注記

fsGroupChangePolicy は、secret、configMap、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。

fsGroupChangePolicy は、namespace レベルまたは Pod レベルで設定できます。

3.6.1. namespace レベルで fsGroup を変更する

任意の **fsGroupChangePolicy** 設定を namespace レベルで適用すると、その後その namespace に作成されるすべての Pod に、その設定が継承されます。ただし、個々の Pod に継承された **fsGroupChangePolicy** 設定は、必要に応じてオーバーライドできます。Pod レベルで **fsGroupChangePolicy** を設定すると、その Pod に継承された namespace レベルの設定がオーバーライドされます。

前提条件

- 管理者特権で Red Hat OpenShift Service on AWS クラスターにログインした。
- Red Hat OpenShift Service on AWS にアクセスできる。

手順

namespace ごとに **fsGroupChangePolicy** を設定するには、以下を実行します。

1. 任意の namespace を選択します。
   a. **Administration** > **Namespaces** をクリックします。
   b. **Namespaces** ページで、任意の namespace をクリックします。 **Namespace details** ページが表示されます。
2. **fsGroupChangePolicy** ラベルを namespace に追加します。
   a. **Namespace details** ページで、**Labels** の横にある **Edit** をクリックします。
   b. **Edit labels** ダイアログで、**storage.openshift.io/fsgroup-change-policy** ラベルを追加し、次のいずれかと同じ設定にします。
      - **OnRootMismatch**: ルートディレクトリーの権限と所有権が、ボリュームの予期される権限と一致しない場合にのみ、権限と所有権を変更することを指定します。これは、Pod のタイムアウト問題を回避するために役立ちます。
      - **Always**: (デフォルト) ボリュームがマウントされるたびに、ボリュームの権限と所有権を必ず変更することを指定します。
   c. **Save** をクリックします。

検証

以前に編集した namespace で Pod を起動し、namespace に設定した値が **spec.securityContext.fsGroupChangePolicy** パラメーターに含まれていることを確認します。

fsGroupChangePolicy 設定を示す Pod YAML ファイルの例

```
securityContext:
```

```
seLinuxOptions:
  level: 's0:c27,c24'
runAsNonRoot: true
fsGroup: 1000750000
fsGroupChangePolicy: OnRootMismatch 1
...
```

1 この値は namespace から継承されます。

3.6.2. Pod レベルで fsGroup を変更する

新規または既存のデプロイメントで **fsGroupChangePolicy** パラメーターを設定すると、管理する Pod にこのパラメーター値が適用されます。同様に、Statefulset でもこれを行うことができます。既存の Pod を編集して **fsGroupChangePolicy** を設定することはできませんが、新しい Pod を作成するときにこのパラメーターを設定することはできます。

この手順では、既存のデプロイメントで **fsGroupChangePolicy** パラメーターを設定する方法について説明します。

前提条件

- Red Hat OpenShift Service on AWS にアクセスできる。

手順

既存のデプロイメントで **fsGroupChangePolicy** パラメーターを設定するには、以下を実行します。

1. **Workloads** > **Deployments** をクリックします。
2. **Deployment** ページで、任意のデプロイメントをクリックします。
3. **Deployment details** ページで、**YAML** タブをクリックします。
4. 次のサンプルファイルを使用して、**spec.template.spec.securityContext** の下にあるデプロイメントの YAML ファイルを編集します。

   fsGroupChangePolicy を設定するデプロイメント YAML ファイルの例

```
...
spec:
replicas: 3
selector:
matchLabels:
app: my-app
template:
metadata:
creationTimestamp: null
labels:
app: my-app
spec:
containers:
- name: container
image: 'image-registry.openshift-image-registry.svc:5000/openshift/httpd:latest'
ports:
```

```
    - containerPort: 8080
      protocol: TCP
    resources: {}
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    imagePullPolicy: Always
  restartPolicy: Always
  terminationGracePeriodSeconds: 30
  dnsPolicy: ClusterFirst
  securityContext:
    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。デフォルト値は **Always** です。ボリュームがマウントされるたびに必ずボリュームの権限と所有権が変更されます。

5. **Save** をクリックします。

3.7. SELINUXCHANGEPOLICY を使用して POD のタイムアウトを減らす

SELinux (Security-Enhanced Linux) は、システム上のすべてのオブジェクト (ファイル、プロセス、ネットワークポートなど) にセキュリティーラベル (コンテキスト) を割り当てるセキュリティーメカニズムです。これらのラベルにより、プロセスが何にアクセスできるか決まります。Red Hat OpenShift Service on AWS において、SELinux はコンテナーのエスケープや、ホストシステムや他のコンテナーへのアクセスを防ぐ役割を果たします。

Pod が起動すると、コンテナーランタイムは、Pod の SELinux コンテキストに合わせて、ボリューム上のすべてのファイルに対して再帰的にラベルの再設定を行います。多くのファイルを含むボリュームの場合、これにより Pod の起動時間が大幅に長くなる可能性があります。

マウントオプションは、-o context マウントオプションを使用して正しい SELinux ラベルでボリュームを直接マウントしようとすることで、すべてのファイルの再帰的なラベル付けを回避するように指定します。これにより、Pod のタイムアウト問題を回避することができます。

RWOP と SELinux マウントオプション

ReadWriteOncePod (RWOP) 永続ボリュームは、デフォルトで SELinux マウント機能を使用します。

マウントオプション機能はドライバーに依存しており、AWS EBS および Red Hat OpenShift Data Foundation ではデフォルトで有効になっています。サードパーティーのドライバーは、ストレージベンダーにお問い合わせください。

RWO、RWX、SELinux マウントオプション

ReadWriteOnce (RWO) および ReadWriteMany (RWX) ボリュームは、デフォルトでラベルの再帰的な再設定を使用します。

重要

今後の Red Hat OpenShift Service on AWS バージョンでは、RWO および RWX ボリュームは **デフォルトでマウントオプション** を使用します。

今後のマウントオプションのデフォルトへの移行を支援するために、Red Hat OpenShift Service on AWS 4.20 では、潜在的な競合を知らせるために Pod の作成時および Pod の実行時に SELinux 関連の

競合を報告し、解決できるようにします。このレポートの詳細は、Kowledge Base の記事 を参照してください。

SELinux 関連の競合を解決できない場合は、選択した Pod または namespace のデフォルトとして、今後のマウントオプションへの移行を事前にオプトアウトできます。オプトアウトする場合は、**SELinux マウントオプションのデフォルトをオプトアウトする** を参照してください。

3.7.1. RWO、RWX、SELinux マウントオプション機能のテスト

Red Hat OpenShift Service on AWS 4.20 では、RWO および RWX ボリュームのマウントオプション機能をテクノロジープレビュー機能として評価できます。

重要

RWO/RWX SELinux マウントはテクノロジープレビュー機能です。テクノロジープレビュー機能は、Red Hat 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。

マウントオプション機能を評価するには、以下を実行します。

- フィーチャーゲートを有効にします。フィーチャーゲートを有効にする方法については、**フィーチャーゲートを使用した機能の有効化** セクションを参照してください。
  RWO および RWX ボリュームには、デフォルトの動作としてマウントオプションが追加されました。

アプリケーションを慎重にテストし、ストレージがどのように使用されているかを観察します。問題が発生する場合は、この ナレッジベースの記事 を参照して、マウントオプションをオプトアウトすることを検討してください。**SELinux マウントオプションのデフォルトをオプトアウトする** セクションを参照してください。

3.7.2. SELinux マウントオプションのデフォルトをオプトアウトする

マウントオプションの今後の移行をデフォルトでオプトアウトする場合は、個々の Pod または namespace レベルで **seLinuxChangePolicy** パラメーターを **Recursive** に設定します。

3.7.2.1. namespace レベルでの seLinuxChangePolicy の変更

任意の **seLinuxChangePolicy** 設定を namespace レベルで適用すると、その後その namespace に作成されるすべての Pod に、その設定が継承されます。ただし、個々の Pod に継承された **seLinuxChangePolicy** 設定は、必要に応じてオーバーライドできます。Pod レベルで **seLinuxChangePolicy** を設定すると、その Pod に継承された namespace レベルの設定がオーバーライドされます。

前提条件

- 管理者特権で Red Hat OpenShift Service on AWS クラスターにログインした。
- Red Hat OpenShift Service on AWS にアクセスできる。

手順

namespace ごとに **SELinuxChangePolicy** を設定するには、以下を実行します。

1. 任意の namespace を選択します。
   a. **Administration** > **Namespaces** をクリックします。
   b. **Namespaces** ページで、任意の namespace をクリックします。**Namespace details** ページが表示されます。
2. **seLinuxChangePolicy** ラベルを namespace に追加します。
   a. **Namespace details** ページで、**Labels** の横にある **Edit** をクリックします。
   b. **Edit labels** ダイアログで、**storage.openshift.io/selinux-change-policy=Recursive** ラベルを追加します。
      これは、Pod ボリューム上のすべてのファイルに対して、適切な SELinux コンテキストへの再帰的なラベルの再設定を行うように指定します。
   c. **Save** をクリックします。

検証

編集した namespace で Pod を起動し、**spec.securityContext.seLinuxChangePolicy** パラメーターが **Recursive** に設定されていることを確認します。

seLinuxChangePolicy 設定を示す pod YAML ファイルの例

```
securityContext:
  seLinuxOptions:
    level: 's0:c27,c19'
  runAsNonRoot: true
  fsGroup: 1000740000
  seccompProfile:
    type: RuntimeDefault
  seLinuxChangePolicy: Recursive (1)
...
```

1. この値は namespace から継承されます。

3.7.2.2. Pod レベルでの seLinuxChangePolicy の変更

新規または既存のデプロイメントで **seLinuxChangePolicy** パラメーターを設定すると、管理する Pod にこのパラメーター値が適用されます。同様に、StatefulSet でもこれを行うことができます。既存の Pod を編集して **seLinuxChangePolicy** を設定することはできませんが、新しい Pod を作成するときにこのパラメーターを設定することはできます。

この手順では、既存のデプロイメントで **seLinuxChangePolicy** パラメーターを設定する方法について説明します。

前提条件

- Red Hat OpenShift Service on AWS にアクセスできる。

手順

既存のデプロイメントで **seLinuxChangePolicy** パラメーターを設定するには、以下を実行します。

1. **Workloads** > **Deployments** をクリックします。
2. **Deployment** ページで、任意のデプロイメントをクリックします。
3. **Deployment details** ページで、**YAML** タブをクリックします。
4. 次のサンプルファイルに従って、**spec.template.spec.securityContext** の下にあるデプロイメントの YAML ファイルを編集します。

seLinuxChangePolicy を設定するデプロイメント YAML ファイルの例

```
...
securityContext:
  seLinuxChangePolicy: Recursive (1)
  ...
```

(1) すべての Pod ボリューム上のすべてのファイルに対して、適切な SELinux コンテキストへの再帰的なラベルの再設定を行うように指定します。

5. **Save** をクリックします。

第4章 永続ストレージの設定

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

Red Hat OpenShift Service on AWS クラスターは、Amazon Elastic Block Store (Amazon EBS) ボリュームを使用する 2 つのストレージクラスで事前に構築されています。これらのストレージクラスはすぐに使用でき、Kubernetes と AWS にある程度精通していることを前提としています。

事前に構築された 2 つのストレージクラスは次のとおりです。

名前	プロビジョナー
gp2-csi	ebs.csi.aws.com
gp3-csi (デフォルト)	ebs.csi.aws.com

gp3-csi ストレージクラスがデフォルトとして設定されています。ただし、いずれかのストレージクラスをデフォルトのストレージクラスとして選択できます。

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。Amazon 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 にバインドされていません。Red Hat OpenShift Service on AWS クラスター全体で共有できます。永続ボリューム要求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。デフォルトでは、Red Hat OpenShift Service on AWS バージョン 4.10 以降を使用して新しく作成されたクラスターは、gp3 ストレージと AWS EBS CSI ドライバー を使用します。

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。

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、Red Hat OpenShift Service on AWS でボリュームとしてマウントする前に、基盤となるインフラストラクチャーに存在する必要があります。

手順

1. Red Hat OpenShift Service on AWS Web 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

Red Hat OpenShift Service on AWS 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含まれてい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。

この検証により、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できるようになります。これは、Red Hat OpenShift Service on AWS が最初に使用する前にフォーマットするためです。

4.1.4. ノード上の EBS ボリュームの最大数

デフォルトでは、Red Hat OpenShift Service on AWS は、1 つのノードにアタッチされた最大 39 個の EBS ボリュームをサポートします。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 ボリュームに対し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の詳細は、AWS Elastic Block Store CSI Driver Operator を参照してください。

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、Red Hat OpenShift Service on AWS 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。

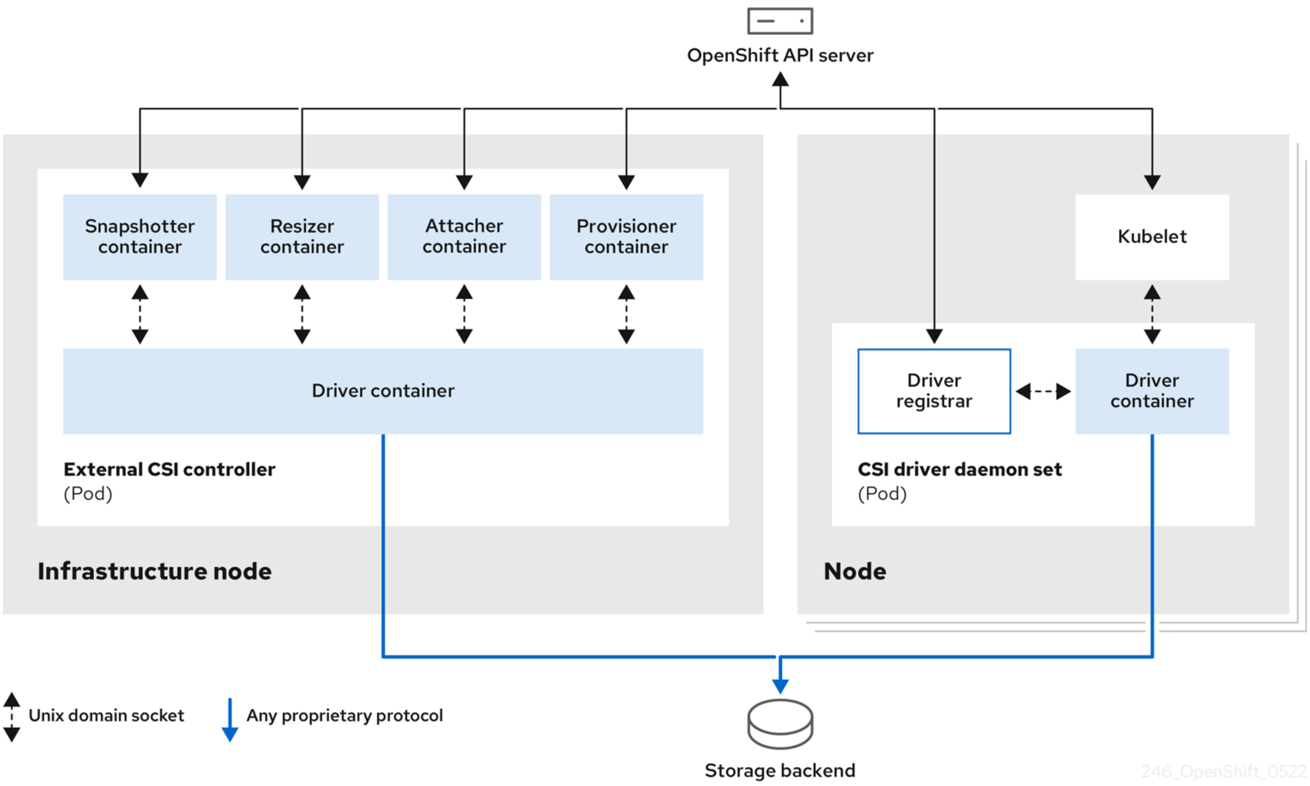
注記

Red Hat OpenShift Service on AWS 4 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、それらが実行される Red Hat OpenShift Service on AWS を認識しません。Red Hat OpenShift Service on AWS で 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、Red Hat OpenShift Service on AWS とストレージドライバーの間のブリッジとして機能する複数のコンポーネントをデプロイする必要があります。

次の図は、Red Hat OpenShift Service on AWS クラスターの Pod で実行されるコンポーネントの概要を示しています。



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。
- 外部の CSI アタッチャーコンテナーは、Red Hat OpenShift Service on AWS からの **attach** および **detach** 呼び出しを、CSI ドライバーへのそれぞれの **ControllerPublish** および **ControllerUnpublish** 呼び出しに変換します。
- Red Hat OpenShift Service on AWS からの **provision** および **delete** 呼び出しを、CSI ドライバーへのそれぞれの **CreateVolume** および **DeleteVolume** 呼び出しに変換する外部 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 Red Hat OpenShift Service on AWS 接続 API を実装するために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 ドライバーデーモンセットは、Red Hat OpenShift Service on AWS 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、それをユーザーワークロード (Pod) で永続ボリューム (PV) として使用できるようにするすべてのノードで 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。Red Hat OpenShift Service on AWS は、これらの呼び出しが実装されている場合、**NodePublish**/**NodeUnpublish** および **NodeStage**/**NodeUnstage** 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. Red Hat OpenShift Service on AWS でサポートされている CSI ドライバー

Red Hat OpenShift Service on AWS 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 ボリュームプラグインでは不可能なストレージオプションをユーザーに提供します。

これらのサポートされているストレージアセットにマウントする CSI プロビジョニングされた永続ボリュームを作成するために、Red Hat OpenShift Service on AWS は、必要な CSI ドライバー Operator、CSI ドライバー、および必要なストレージクラスをデフォルトで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 の詳細は、特定の CSI Driver Operator のドキュメントを参照してください。

重要

AWS EFS ドライバーはデフォルトではインストールされないため、手動でインストールする必要があります。AWS EFS CSI ドライバーのインストール手順については、**関連情報** セクションの「AWS Elastic File Service CSI Driver Operator」を参照してください。

次の表は、Red Hat OpenShift Service on AWS とともにインストールされ、Red Hat OpenShift Service on AWS でサポートされている CSI ドライバーと、ドライバーがサポートしている CSI 機能 (ボリュームのスナップショットやサイズ変更など) について説明しています。

次の表に記載されているドライバーに加えて、Red Hat OpenShift Service on AWS はサードパーティーのストレージベンダーの CSI ドライバーでも機能します。Red Hat はサードパーティーのプロビジョナーや接続されている CSI ドライバーを監視しておらず、ソースコード、デプロイメント、操作、Kubernetes の互換性は、ベンダーがすべて制御しています。これらのボリュームプロビジョナーはお客様が管理するものとし、各ベンダーにはそのサポートを提供する責任があります。詳細は、**関連情報** セクションの **Red Hat OpenShift Service on AWS の責任共有**を参照してください。

表5.1 Red Hat OpenShift Service on AWS でサポートされている CSI ドライバーと機能

CSI ドライバー	CSI ボリュームスナップショット	CSI ボリュームグループスナップショット [1]	CSI のクローン作成	CSI のサイズ変更	インラインの一時ボリューム
AWS EBS	▮			▮	
AWS EFS					
LVM Storage	▮		▮	▮	

関連情報

- Red Hat エコシステムポータル
- サードパーティーサポートポリシー

関連情報

- AWS Elastic File Service CSI Driver Operator
- Red Hat OpenShift Service on AWS の責任共有

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、Red Hat OpenShift Service on AWS でストレージクラスを作成する方法と、設定に使用できるパラメーターを文書化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:
  csi.storage.k8s.io/fstype: xfs 3
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。
3. vSphere CSI ドライバーは、XFS や Ext4 など、基盤となる Red Hat Core オペレーティングシステムリリースでサポートされているすべてのファイルシステムをサポートします。

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
mysql       Bound       kubernetes-dynamic-pv-3271ffcb4e1811e8   1Gi

ACCESS MODES   STORAGECLASS   AGE
RWO            gp3-csi        3s
```

5.2. デフォルトストレージクラスの管理

5.2.1. 概要

デフォルトのストレージクラスを管理すると、複数の異なる目的を達成できます。

- 動的プロビジョニングを無効にして静的プロビジョニングを強制します。
- 他の優先ストレージクラスがある場合に、storage operator による最初のデフォルトストレージクラスの作成を阻止します。
- デフォルトのストレージクラスに名前を付け直すか変更します。

これらの目的を達成するには、**ClusterCSIDriver** オブジェクトの **spec.storageClassState** フィールドの設定を変更します。このフィールドで可能な設定は以下のとおりです。

- **Managed**: (デフォルト) コンテナーストレージインターフェイス (CSI) オペレーターがデフォルトのストレージクラスをアクティブに管理しているため、クラスター管理者によってデフォルトのストレージクラスに対して行われた手動変更のほとんどは削除され、デフォルトのストレージクラスは継続的に再作成されます。手動で削除しようとしました。
- **Unmanaged**: デフォルトのストレージクラスを変更できます。CSI Operator はストレージクラスをアクティブに管理しないため、自動的に作成するデフォルトのストレージクラスを調整します。
- **Removed**: CSI Operator はデフォルトのストレージクラスを削除します。

5.2.2. Web コンソールを使用したデフォルトのストレージクラスの管理

前提条件

- Red Hat OpenShift Service on AWS Web コンソールにアクセスできる。
- クラスター管理者権限でクラスターにアクセスできる。

手順

Web コンソールを使用してデフォルトのストレージクラスを管理するには、次の手順を実行します。

1. Web コンソールにログインします。
2. **Administration** > **CustomResourceDefinitions** をクリックします。
3. **CustomResourceDefinitions** ページで、**clustercsidriver** と入力して、**ClusterCSIDriver** オブジェクトを見つけます。
4. **ClusterCSIDriver** をクリックし、**Instances** タブをクリックします。
5. 目的のインスタンスの名前をクリックし、**YAML** タブをクリックします。
6. **spec.storageClassState** フィールドを追加し、値を **Managed**、**Unmanaged**、または **Removed** に設定します。

例

```
...
spec:
  driverConfig:
    driverType: ''
  logLevel: Normal
  managementState: Managed
  observedConfig: null
  operatorLogLevel: Normal
  storageClassState: Unmanaged 1
...
```

1 **spec.storageClassState** フィールドが "Unmanaged" に設定されている

7. **Save** をクリックします。

5.2.3. CLI を使用したデフォルトのストレージクラスの管理

前提条件

- クラスター管理者権限でクラスターにアクセスできる。

手順

CLI を使用してストレージクラスを管理するには、次のコマンドを実行します。

```
$ oc patch clustercsidriver $DRIVERNAME --type=merge -p "{\"spec\":{\"storageClassState\":\"${STATE}\"}}" 1
```

1 ここで、**${STATE}** は "削除"、"管理"、または "非管理" です。

$DRIVERNAME はプロビジョナー名です。コマンド **oc get sc** を実行すると、プロビジョナー名を見つけることができます。

5.2.4. デフォルトのストレージクラスが存在しないか、複数のデフォルトストレージクラスがある

5.2.4.1. 複数のデフォルトのストレージクラス

デフォルト以外のストレージクラスをデフォルトとしてマークし、既存のデフォルトストレージクラスの設定を解除しない場合、またはデフォルトのストレージクラスがすでに存在するときにデフォルトのストレージクラスを作成した場合は、複数のデフォルトストレージクラスが発生する可能性があります。複数のデフォルトストレージクラスが存在する場合、デフォルトストレージクラス (**pvc.spec.storageClassName** =nil) を要求するすべての永続ボリューム要求 (PVC) は、そのストレージクラスのデフォルトステータスと管理者に関係なく、最後に作成されたデフォルトストレージクラスを取得します。アラートダッシュボードで、複数のデフォルトストレージクラス **MultipleDefaultStorageClasses** があるというアラートを受け取ります。

5.2.4.2. デフォルトのストレージクラスなし

PVC が存在しないデフォルトのストレージクラスの使用を試みる可能性があるシナリオは 2 つあります。

- 管理者がデフォルトのストレージクラスを削除するか、デフォルト以外としてマークした後、ユーザーがデフォルトのストレージクラスを要求する PVC を作成します。
- インストール時に、インストーラーは、まだ作成されていないデフォルトのストレージクラスを要求する PVC を作成します。

前述のシナリオでは、PVC は無期限に保留状態のままになります。この状況を解決するには、デフォルトのストレージクラスを作成するか、既存のストレージクラスの 1 つをデフォルトとして宣言します。デフォルトのストレージクラスが作成または宣言されるとすぐに、PVC は新しいデフォルトのストレージクラスを取得します。可能であれば、最終的に PVC は通常どおり静的または動的にプロビジョニングされた PV にバインドされ、保留状態から抜け出します。

5.2.5. デフォルトストレージクラスの変更

次の手順を使用して、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。

前提条件

- クラスター管理者権限でクラスターにアクセスできる。

手順

デフォルトのストレージクラスを変更するには、以下を実行します。

1. ストレージクラスを一覧表示します。

```
$ oc get storageclass
```

出力例

```
NAME            TYPE
gp3 (default)    ebs.csi.aws.com (1)
standard        ebs.csi.aws.com
```

(1) **(default)** はデフォルトのストレージクラスを示します。

2. 目的のストレージクラスをデフォルトにします。
目的のストレージクラスに、次のコマンドを実行して **storageclass.kubernetes.io/is-default-class** アノテーションを **true** に設定します。

```
$ oc patch storageclass standard -p '{"metadata": {"annotations":
{"storageclass.kubernetes.io/is-default-class": "true"}}}'
```

注記

短期間であれば、複数のデフォルトのストレージクラスを使用できます。ただし、最終的には 1 つのデフォルトのストレージクラスのみが存在することを確認する必要があります。

複数のデフォルトストレージクラスが存在する場合、デフォルトストレージクラス (**pvc.spec.storageClassName** =nil) を要求するすべての永続ボリューム要求 (PVC) は、そのストレージクラスのデフォルトステータスと管理者に関係なく、最後に作成されたデフォルトストレージクラスを取得します。アラートダッシュボードで、複数のデフォルトストレージクラス **MultipleDefaultStorageClasses** があるというアラートを受け取ります。

3. 古いデフォルトストレージクラスからデフォルトのストレージクラス設定を削除します。
古いデフォルトのストレージクラスの場合は、次のコマンドを実行して **storageclass.kubernetes.io/is-default-class** アノテーションの値を **false** に変更します。

```
$ oc patch storageclass gp3 -p '{"metadata": {"annotations": {"storageclass.kubernetes.io/is-default-class": "false"}}}'
```

4. 変更内容を確認します。

```
$ oc get storageclass
```

出力例

```
NAME               TYPE
gp3                ebs.csi.aws.com
standard (default)   ebs.csi.aws.com
```

5.3. AWS ELASTIC BLOCK STORE CSI DRIVER OPERATOR

5.3.1. 概要

Red Hat OpenShift Service on AWS は AWS EBS 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 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 プロビジョニングされた PV を作成するために、Red Hat OpenShift Service on AWS は AWS EBS CSI Driver Operator (Red Hat Operator) と AWS EBS CSI ドライバーをデフォルトで **openshift-cluster-csi-drivers** namespace にインストールします。

- **AWS EBS CSI Driver 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。必要に応じて、このデフォルトのストレージクラスを無効にできます (デフォルトストレージクラスの管理 を参照)。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。

5.3.2. 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 Red Hat OpenShift Service on AWS ユーザーに付与します。

> **重要**
>
> Red Hat OpenShift Service on AWS は、デフォルトで CSI プラグインを使用して Amazon Elastic Block Store (Amazon EBS) ストレージをプロビジョニングします。

Red Hat OpenShift Service on AWS で AWS EBS 永続ボリュームを動的にプロビジョニングする方法については、Amazon 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

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

5.4. AWS ELASTIC FILE SERVICE CSI DRIVER OPERATOR

> **重要**
>
> この手順は、AWS EFS CSI Driver Operator (Red Hat Operator) に固有のものであり、Red Hat OpenShift Service on AWS 4.10 にのみ適用されます。

5.4.1. 概要

Red Hat OpenShift Service on AWS 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 を理解しておくことが推奨されます。

AWS EFS CSI Driver Operator をインストールした後、Red Hat OpenShift Service on AWS 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、デフォルトでは、永続ボリューム要求 (PVC) の作成に使用されるストレージクラスは作成されません。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。

5.4.2. 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 Red Hat OpenShift Service on AWS ユーザーに付与します。

5.4.3. AWS EFS CSI Driver Operator の設定

1. AWS EFS と AWS Secure Token Service (STS) を使用している場合は、STS のロール Amazon Resource Name (ARN) を取得します。これは、AWS EFS CSI Driver Operator をインストールするために必要です。

2. AWS EFS CSI Driver Operator をインストールします。

3. AWS EFS CSI ドライバーをインストールします。

5.4.3.1. Security Token Service のロール Amazon Resource Name の取得

この手順では、AWS Security Token Service (STS) 環境の Red Hat OpenShift Service on AWS で AWS EFS CSI Driver Operator を設定するために、ロールの Amazon Resource Name (ARN) を取得する方法を説明します。

重要

AWS EFS CSI Driver Operator をインストールする前に、この手順を実行してください (**AWS EFS CSI Driver Operator のインストール** に記載された手順を参照)。

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。

手順

1. 以下の内容を含む IAM ポリシー JSON ファイルを作成します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticfilesystem:DescribeAccessPoints",
        "elasticfilesystem:DescribeFileSystems",
        "elasticfilesystem:DescribeMountTargets",
```

```
        "ec2:DescribeAvailabilityZones",
        "elasticfilesystem:TagResource"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "elasticfilesystem:CreateAccessPoint"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "aws:RequestTag/efs.csi.aws.com/cluster": "true"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "elasticfilesystem:DeleteAccessPoint",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/efs.csi.aws.com/cluster": "true"
        }
      }
    }
  ]
}
```

2. 以下の内容で IAM 信頼 JSON ファイルを作成します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::<your_aws_account_ID>:oidc-
provider/<openshift_oidc_provider>"  (1)
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "<openshift_oidc_provider>:sub": [  (2)
            "system:serviceaccount:openshift-cluster-csi-drivers:aws-efs-csi-driver-operator",
            "system:serviceaccount:openshift-cluster-csi-drivers:aws-efs-csi-driver-controller-sa"
          ]
        }
      }
    }
  ]
}
```

1. AWS アカウント ID および OpenShift OIDC プロバイダーエンドポイントを指定します。

次のコマンドを実行して AWS アカウント ID を取得します。

```
$ aws sts get-caller-identity --query Account --output text
```

2 OpenShift OIDC エンドポイントを再度指定します。

3. IAM ロールを作成します。

```
ROLE_ARN=$(aws iam create-role \
  --role-name "<your_cluster_name>-aws-efs-csi-operator" \
  --assume-role-policy-document file://<your_trust_file_name>.json \
  --query "Role.Arn" --output text); echo $ROLE_ARN
```

ロール ARN をコピーします。AWS EFS CSI Driver Operator をインストールするときに必要になります。

4. IAM ポリシーを作成します。

```
POLICY_ARN=$(aws iam create-policy \
  --policy-name "<your_cluster_name>-aws-efs-csi" \
  --policy-document file://<your_policy_file_name>.json \
  --query 'Policy.Arn' --output text); echo $POLICY_ARN
```

5. IAM ポリシーを IAM ロールに割り当てます。

```
$ aws iam attach-role-policy \
  --role-name "<your_cluster_name>-aws-efs-csi-operator" \
  --policy-arn $POLICY_ARN
```

次のステップ

AWS EFS CSI Driver Operator をインストール します。

関連情報

- AWS EFS CSI Driver Operator のインストール
- AWS EFS CSI ドライバーのインストール

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

AWS EFS CSI Driver Operator (Red Hat Operator) は、デフォルトでは Red Hat OpenShift Service on AWS にインストールされていません。以下の手順を使用して、クラスター内で AWS EFS CSI Driver Operator をインストールおよび設定します。

前提条件

- Red Hat OpenShift Service on AWS 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. **Ecosystem** → **Software Catalog** 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。

   a. **AWS EFS CSI Driver Operator** ページで **Install** をクリックします。
   b. **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。
   c. **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 EFS CSI ドライバーをインストール します。

5.4.3.3. AWS EFS CSI ドライバーのインストール

AWS EFS CSI Driver Operator (Red Hat Operator) をインストールした後、AWS EFS CSI ドライバーをインストールします。

前提条件

- Red Hat OpenShift Service on AWS 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。

```yaml
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

5.4.4. 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 Driver Operator (Red Hat Operator) は、インストール後、デフォルトではストレージクラスを作成しません。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。

5.4.4.1. 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. Red Hat OpenShift Service on AWS Web 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。

5.4.4.2. 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。

5.4.5. AWS EFS CSI クロスアカウントのサポート

クロスアカウントのサポートにより、1 つの AWS アカウントに Red Hat OpenShift Service on AWS クラスターを配置し、AWS Elastic File System (EFS) Container Storage Interface (CSI) ドライバーを使用して別の AWS アカウントにファイルシステムをマウントできます。

前提条件

- 管理者権限による Red Hat OpenShift Service on AWS クラスターへのアクセス
- 2 つの有効な AWS アカウントがある。
- EFS CSI Operator がインストールされている。EFS CSI Operator のインストール方法は、**AWS EFS CSI Driver Operator のインストール** セクションを参照してください。
- Red Hat OpenShift Service on AWS クラスターと EFS ファイルシステムがどちらも、同じ AWS リージョンに配置されている。
- 次の手順で使用する 2 つの仮想プライベートクラウド (VPC) が異なるネットワーク Classless Inter-Domain Routing (CIDR) 範囲を使用していることを確認する。
- Red Hat OpenShift Service on AWS CLI (**oc**) へのアクセス。
- AWS CLI へアクセスできる。
- **jq** コマンドライン JSON プロセッサーへアクセスできる。

手順

次の手順では、以下のアカウントを設定する方法を説明します。

- Red Hat OpenShift Service on AWS の AWS アカウント A: VPC 内にデプロイされた Red Hat OpenShift Service on AWS クラスター (v4.16 以降) が含まれています。
- AWS アカウント B: VPC (サブネット、ルートテーブル、ネットワーク接続を含む) が含まれています。この VPC に EFS ファイルシステムを作成します。

アカウント間で AWS EFS を使用するには:

1. 環境をセットアップします。
   a. 次のコマンドを実行して環境変数を設定します。

```
export CLUSTER_NAME="<CLUSTER_NAME>" 1
export AWS_REGION="<AWS_REGION>" 2
export AWS_ACCOUNT_A_ID="<ACCOUNT_A_ID>" 3
export AWS_ACCOUNT_B_ID="<ACCOUNT_B_ID>" 4
export AWS_ACCOUNT_A_VPC_CIDR="<VPC_A_CIDR>" 5
export AWS_ACCOUNT_B_VPC_CIDR="<VPC_B_CIDR>" 6
export AWS_ACCOUNT_A_VPC_ID="<VPC_A_ID>" 7
export AWS_ACCOUNT_B_VPC_ID="<VPC_B_ID>" 8
export SCRATCH_DIR="<WORKING_DIRECTORY>" 9
export CSI_DRIVER_NAMESPACE="openshift-cluster-csi-drivers" 10
export AWS_PAGER="" 11
```

   1 任意のクラスター名。

   2 任意の AWS リージョン。

   3 AWS アカウント A の ID。

   4 AWS アカウント B の ID。

   5 アカウント A の VPC の CIDR 範囲。

   6 アカウント B の VPC の CIDR 範囲。

   7 アカウント A (クラスター) の VPC ID

   8 アカウント B の VPC ID (EFS クロスアカウント)

   9 一時ファイルを保存するために使用する任意の書き込み可能なディレクトリー。

   10 ドライバーがデフォルト以外の namespace にインストールされている場合は、この値を変更します。

   11 AWS CLI のすべての出力を stdout に直接出力させます。

   b. 次のコマンドを実行して作業ディレクトリーを作成します。

```
mkdir -p $SCRATCH_DIR
```

c. Red Hat OpenShift Service on AWS CLI で次のコマンドを実行して、クラスターの接続を確認します。

```
$ oc whoami
```

d. Red Hat OpenShift Service on AWS クラスターを決定し、ノードセレクターを設定します。
EFS クロスアカウント機能を使用するには、EFS CSI コントローラー Pod が稼働しているノードに AWS IAM ポリシーを割り当てる必要があります。ただし、これは Red Hat OpenShift Service on AWS タイプで一貫しているわけではありません。

- クラスターが Hosted Control Plane (HyperShift) としてデプロイされている場合は、次のコマンドを実行して、ワーカーノードのラベルを格納するように **NODE_SELECTOR** 環境変数を設定します。

```
export NODE_SELECTOR=node-role.kubernetes.io/worker
```

- Red Hat OpenShift Service on AWS タイプでは、次のコマンドを実行して、マスターノードのラベルを保持するように **NODE_SELECTOR** 環境変数を設定します。

```
export NODE_SELECTOR=node-role.kubernetes.io/master
```

e. 次のコマンドを実行して、アカウント切り替え用の環境変数として AWS CLI プロファイルを設定します。

```
export AWS_ACCOUNT_A="<ACCOUNT_A_NAME>"
export AWS_ACCOUNT_B="<ACCOUNT_B_NAME>"
```

f. 次のコマンドを実行して、両方のアカウントで AWS CLI のデフォルト出力形式が JSON に設定されていることを確認します。

```
export AWS_DEFAULT_PROFILE=${AWS_ACCOUNT_A}
aws configure get output
export AWS_DEFAULT_PROFILE=${AWS_ACCOUNT_B}
aws configure get output
```

上記のコマンドが次の結果を返す場合:

- **値なし**: デフォルト出力形式がすでに JSON に設定されており、変更は必要ありません。
- **何らかの値**: JSON 形式を使用するように AWS CLI を再設定します。出力形式の変更方法は、AWS ドキュメントの **Setting the output format in the AWS CLI**を参照してください。

g. 次のコマンドを実行して、**AWS_DEFAULT_PROFILE** との競合を防ぐために、シェルで **AWS_PROFILE** を設定解除します。

```
unset AWS_PROFILE
```

2. AWS アカウント B の IAM ロールとポリシーを設定します。

a. 次のコマンドを実行して、アカウント B のプロファイルに切り替えます。

```
export AWS_DEFAULT_PROFILE=${AWS_ACCOUNT_B}
```

b. 次のコマンドを実行して、EFS CSI Driver Operator の IAM ロール名を定義します。

```
export ACCOUNT_B_ROLE_NAME=${CLUSTER_NAME}-cross-account-aws-efs-csi-operator
```

c. 次のコマンドを実行して、IAM 信頼ポリシーファイルを作成します。

```
cat <<EOF > $SCRATCH_DIR/AssumeRolePolicyInAccountB.json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::${AWS_ACCOUNT_A_ID}:root"
            },
            "Action": "sts:AssumeRole",
            "Condition": {}
        }
    ]
}
EOF
```

d. 次のコマンドを実行して、EFS CSI Driver Operator の IAM ロールを作成します。

```
ACCOUNT_B_ROLE_ARN=$(aws iam create-role \
  --role-name "${ACCOUNT_B_ROLE_NAME}" \
  --assume-role-policy-document
file://$SCRATCH_DIR/AssumeRolePolicyInAccountB.json \
  --query "Role.Arn" --output text) \
&& echo $ACCOUNT_B_ROLE_ARN
```

e. 次のコマンドを実行して、IAM ポリシーファイルを作成します。

```
cat << EOF > $SCRATCH_DIR/EfsPolicyInAccountB.json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeSubnets"
            ],
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "elasticfilesystem:DescribeMountTargets",
```

```
                "elasticfilesystem:DeleteAccessPoint",
                "elasticfilesystem:ClientMount",
                "elasticfilesystem:DescribeAccessPoints",
                "elasticfilesystem:ClientWrite",
                "elasticfilesystem:ClientRootAccess",
                "elasticfilesystem:DescribeFileSystems",
                "elasticfilesystem:CreateAccessPoint",
                "elasticfilesystem:TagResource"
            ],
            "Resource": "*"
        }
    ]
}
EOF
```

f. 次のコマンドを実行して、IAM ポリシーを作成します。

```
ACCOUNT_B_POLICY_ARN=$(aws iam create-policy --policy-name
"${CLUSTER_NAME}-efs-csi-policy" \
   --policy-document file://$SCRATCH_DIR/EfsPolicyInAccountB.json \
   --query 'Policy.Arn' --output text) \
&& echo ${ACCOUNT_B_POLICY_ARN}
```

g. 次のコマンドを実行して、ポリシーをロールにアタッチします。

```
aws iam attach-role-policy \
   --role-name "${ACCOUNT_B_ROLE_NAME}" \
   --policy-arn "${ACCOUNT_B_POLICY_ARN}"
```

3. AWS アカウント A の IAM ロールとポリシーを設定します。

a. 次のコマンドを実行して、アカウント A のプロファイルに切り替えます。

```
export AWS_DEFAULT_PROFILE=${AWS_ACCOUNT_A}
```

b. 次のコマンドを実行して、IAM ポリシードキュメントを作成します。

```
cat << EOF > $SCRATCH_DIR/AssumeRoleInlinePolicyPolicyInAccountA.json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "${ACCOUNT_B_ROLE_ARN}"
    }
  ]
}
EOF
```

c. AWS アカウント A で、次のコマンドを実行して、AWS 管理ポリシー "AmazonElasticFileSystemClientFullAccess" を Red Hat OpenShift Service on AWS クラスターのマスターロールにアタッチします。

```
EFS_CLIENT_FULL_ACCESS_BUILTIN_POLICY_ARN=arn:aws:iam::aws:policy/Amazon
ElasticFileSystemClientFullAccess
declare -A ROLE_SEEN
for NODE in $(oc get nodes --selector="${NODE_SELECTOR}" -o
jsonpath='{.items[*].metadata.name}'); do
  INSTANCE_PROFILE=$(aws ec2 describe-instances \
    --filters "Name=private-dns-name,Values=${NODE}" \
    --query 'Reservations[].Instances[].IamInstanceProfile.Arn' \
    --output text | awk -F'/' '{print $NF}' | xargs)
  MASTER_ROLE_ARN=$(aws iam get-instance-profile \
    --instance-profile-name "${INSTANCE_PROFILE}" \
    --query 'InstanceProfile.Roles[0].Arn' \
    --output text | xargs)
  MASTER_ROLE_NAME=$(echo "${MASTER_ROLE_ARN}" | awk -F'/' '{print $NF}' |
xargs)
  echo "Checking role: '${MASTER_ROLE_NAME}'"
  if [[ -n "${ROLE_SEEN[$MASTER_ROLE_NAME]:-}" ]]; then
    echo "Already processed role: '${MASTER_ROLE_NAME}', skipping."
    continue
  fi
  ROLE_SEEN["$MASTER_ROLE_NAME"]=1
  echo "Assigning policy ${EFS_CLIENT_FULL_ACCESS_BUILTIN_POLICY_ARN} to
role ${MASTER_ROLE_NAME}"
  aws iam attach-role-policy --role-name "${MASTER_ROLE_NAME}" --policy-arn
"${EEFS_CLIENT_FULL_ACCESS_BUILTIN_POLICY_ARN}"
done
```

4. ロールの引き受けを許可するために、IAM エンティティーにポリシーをアタッチします。
この手順はクラスターの設定によって異なります。次のどちらの場合も、EFS CSI Driver Operator はエンティティーを使用して AWS に対して認証します。このエンティティーには、アカウント B でロールを引き受けるための権限を付与する必要があります。

   クラスターで:

   - **STS が有効でない場合**: EFS CSI Driver Operator は、AWS 認証に IAM ユーザーエンティティーを使用します。「ロールの引き受けを許可するために IAM ユーザーにポリシーをアタッチする」ステップに進んでください。
   - **STS が有効な場合**: EFS CSI Driver Operator は、AWS 認証に IAM ロールエンティティーを使用します。「ロールの引き受けを許可するために IAM ロールにポリシーをアタッチする」ステップに進んでください。

5. ロールの引き受けを許可するために IAM ユーザーにポリシーをアタッチします。

   a. 次のコマンドを実行して、EFS CSI Driver Operator によって使用されている IAM ユーザーを特定します。

```
EFS_CSI_DRIVER_OPERATOR_USER=$(oc -n openshift-cloud-credential-operator get
credentialsrequest/openshift-aws-efs-csi-driver -o json | jq -r '.status.providerStatus.user')
```

   b. 次のコマンドを実行して、その IAM ユーザーにポリシーをアタッチします。

```
aws iam put-user-policy \
  --user-name "${EFS_CSI_DRIVER_OPERATOR_USER}"  \
  --policy-name efs-cross-account-inline-policy \
  --policy-document
```

```
file://$SCRATCH_DIR/AssumeRoleInlinePolicyPolicyInAccountA.json
```

6. ロールの引き受けを許可するために IAM ロールにポリシーをアタッチします。

   a. 次のコマンドを実行して、EFS CSI Driver Operator によって現在使用されている IAM ロール名を特定します。

   ```
   EFS_CSI_DRIVER_OPERATOR_ROLE=$(oc -n ${CSI_DRIVER_NAMESPACE} get
   secret/aws-efs-cloud-credentials -o jsonpath='{.data.credentials}' | base64 -d | grep
   role_arn | cut -d'/' -f2) && echo ${EFS_CSI_DRIVER_OPERATOR_ROLE}
   ```

   b. 次のコマンドを実行して、EFS CSI Driver Operator によって使用されている IAM ロールにポリシーをアタッチします。

   ```
    aws iam put-role-policy \
      --role-name "${EFS_CSI_DRIVER_OPERATOR_ROLE}"  \
      --policy-name efs-cross-account-inline-policy \
      --policy-document
   file://$SCRATCH_DIR/AssumeRoleInlinePolicyPolicyInAccountA.json
   ```

7. VPC ピアリングを設定します。

   a. 次のコマンドを実行して、アカウント A からアカウント B へのピアリングリクエストを開始します。

   ```
   export AWS_DEFAULT_PROFILE=${AWS_ACCOUNT_A}
   PEER_REQUEST_ID=$(aws ec2 create-vpc-peering-connection --vpc-id
   "${AWS_ACCOUNT_A_VPC_ID}" --peer-vpc-id "${AWS_ACCOUNT_B_VPC_ID}" --
   peer-owner-id "${AWS_ACCOUNT_B_ID}" --query
   VpcPeeringConnection.VpcPeeringConnectionId --output text)
   ```

   b. 次のコマンドを実行して、アカウント B でピアリングリクエストを承認します。

   ```
   export AWS_DEFAULT_PROFILE=${AWS_ACCOUNT_B}
   aws ec2 accept-vpc-peering-connection --vpc-peering-connection-id
   "${PEER_REQUEST_ID}"
   ```

   c. 次のコマンドを実行して、アカウント A のルートテーブル ID を取得し、アカウント B の VPC にルートを追加します。

   ```
   export AWS_DEFAULT_PROFILE=${AWS_ACCOUNT_A}
   for NODE in $(oc get nodes --selector=node-role.kubernetes.io/worker | tail -n +2 | awk
   '{print $1}')
   do
     SUBNET=$(aws ec2 describe-instances --filters "Name=private-dns-
   name,Values=$NODE" --query
   'Reservations[*].Instances[*].NetworkInterfaces[*].SubnetId' | jq -r '.[0][0][0]')
     echo SUBNET is ${SUBNET}
     ROUTE_TABLE_ID=$(aws ec2 describe-route-tables --filters
   "Name=association.subnet-id,Values=${SUBNET}" --query
   'RouteTables[*].RouteTableId' | jq -r '.[0]')
     echo Route table ID is $ROUTE_TABLE_ID
     aws ec2 create-route --route-table-id ${ROUTE_TABLE_ID} --destination-cidr-block
   ${AWS_ACCOUNT_B_VPC_CIDR} --vpc-peering-connection-id
   ${PEER_REQUEST_ID}
   ```

```
done
```

d. 次のコマンドを実行して、アカウント B のルートテーブル ID を取得し、アカウント A の VPC にルートを追加します。

```
export AWS_DEFAULT_PROFILE=${AWS_ACCOUNT_B}
for ROUTE_TABLE_ID in $(aws ec2 describe-route-tables   --filters "Name=vpc-
id,Values=${AWS_ACCOUNT_B_VPC_ID}"   --query "RouteTables[].RouteTableId" | jq -
r '.[]')
do
    echo Route table ID is $ROUTE_TABLE_ID
    aws ec2 create-route --route-table-id ${ROUTE_TABLE_ID} --destination-cidr-block
${AWS_ACCOUNT_A_VPC_CIDR} --vpc-peering-connection-id
${PEER_REQUEST_ID}
done
```

8. アカウント A から EFS への NFS トラフィックを許可するように、アカウント B のセキュリティーグループを設定します。

a. 次のコマンドを実行して、アカウント B のプロファイルに切り替えます。

```
export AWS_DEFAULT_PROFILE=${AWS_ACCOUNT_B}
```

b. 次のコマンドを実行して、EFS アクセス用の VPC セキュリティーグループを設定します。

```
SECURITY_GROUP_ID=$(aws ec2 describe-security-groups --filters Name=vpc-
id,Values="${AWS_ACCOUNT_B_VPC_ID}" | jq -r '.SecurityGroups[].GroupId')
aws ec2 authorize-security-group-ingress \
 --group-id "${SECURITY_GROUP_ID}" \
 --protocol tcp \
 --port 2049 \
 --cidr "${AWS_ACCOUNT_A_VPC_CIDR}" | jq .
```

9. アカウント B にリージョン全体の EFS ファイルシステムを作成します。

a. 次のコマンドを実行して、アカウント B のプロファイルに切り替えます。

```
export AWS_DEFAULT_PROFILE=${AWS_ACCOUNT_B}
```

b. 次のコマンドを実行して、リージョン全体の EFS ファイルシステムを作成します。

```
CROSS_ACCOUNT_FS_ID=$(aws efs create-file-system --creation-token efs-token-1 \
--region ${AWS_REGION} \
--encrypted | jq -r '.FileSystemId') \
&& echo $CROSS_ACCOUNT_FS_ID
```

c. 次のコマンドを実行して、EFS のリージョン全体のマウントターゲットを設定します。

```
for SUBNET in $(aws ec2 describe-subnets \
  --filters "Name=vpc-id,Values=${AWS_ACCOUNT_B_VPC_ID}" \
  --region ${AWS_REGION} \
  | jq -r '.Subnets.[].SubnetId'); do \
    MOUNT_TARGET=$(aws efs create-mount-target --file-system-id
${CROSS_ACCOUNT_FS_ID} \
```

```
    --subnet-id ${SUBNET} \
    --region ${AWS_REGION} \
    | jq -r '.MountTargetId'); \
    echo ${MOUNT_TARGET}; \
done
```

これにより、VPC の各サブネットにマウントポイントが作成されます。

10. クロスアカウントアクセス用に EFS Operator を設定します。

    a. 次のコマンドを実行して、後のステップで作成するシークレットとストレージクラスのカスタム名を定義します。

    ```
    export SECRET_NAME=my-efs-cross-account
    export STORAGE_CLASS_NAME=efs-sc-cross
    ```

    b. Red Hat OpenShift Service on AWS で次のコマンドを実行して、アカウント B のロール ARN を参照するシークレットを作成します。

    ```
    oc create secret generic ${SECRET_NAME} -n ${CSI_DRIVER_NAMESPACE} --from-literal=awsRoleArn="${ACCOUNT_B_ROLE_ARN}"
    ```

    c. Red Hat OpenShift Service on AWS で次のコマンドを実行して、CSI ドライバーコントローラーに新しく作成されたシークレットへのアクセス権を付与します。

    ```
    oc -n ${CSI_DRIVER_NAMESPACE} create role access-secrets --verb=get,list,watch --resource=secrets
    oc -n ${CSI_DRIVER_NAMESPACE} create rolebinding --role=access-secrets default-to-secrets --serviceaccount=${CSI_DRIVER_NAMESPACE}:aws-efs-csi-driver-controller-sa
    ```

    d. Red Hat OpenShift Service on AWS で次のコマンドを実行して、アカウント B の EFS ID と以前に作成したシークレットを参照する新しいストレージクラスを作成します。

    ```
    cat << EOF | oc apply -f -
    kind: StorageClass
    apiVersion: storage.k8s.io/v1
    metadata:
      name: ${STORAGE_CLASS_NAME}
    provisioner: efs.csi.aws.com
    parameters:
      provisioningMode: efs-ap
      fileSystemId: ${CROSS_ACCOUNT_FS_ID}
      directoryPerms: "700"
      gidRangeStart: "1000"
      gidRangeEnd: "2000"
      basePath: "/dynamic_provisioning"
      csi.storage.k8s.io/provisioner-secret-name: ${SECRET_NAME}
      csi.storage.k8s.io/provisioner-secret-namespace: ${CSI_DRIVER_NAMESPACE}
    EOF
    ```

関連情報

- Setting the output format in the AWS CLI

5.4.6. One Zone ファイルシステム

5.4.6.1. One Zone ファイルシステムの概要

Red Hat OpenShift Service on AWS は、単一のアベイラビリティーゾーン (AZ) 内にデータを冗長的に格納する EFS ストレージオプションである AWS Elastic File System (EFS) One Zone ファイルシステムをサポートしています。これは、リージョン内の複数の AZ をまたいでデータを冗長的に保存するデフォルトの EFS ストレージオプションとは対照的です。

Red Hat OpenShift Service on AWS 4.19 からアップグレードされたクラスターは、リージョン EFS ボリュームと互換性があります。

> **注記**
>
> One Zone ボリュームの動的プロビジョニングは、シングルゾーンクラスターでのみサポートされます。クラスター内のすべてのノードは、動的プロビジョニングに使用される EFS ボリュームと同じ AZ に存在する必要があります。
>
> 永続ボリューム (PV) に、ボリュームが存在するゾーンを示す正しい **spec.nodeAffinity** があることを前提として、リージョンクラスターで手動でプロビジョニングされた One Zone ボリュームがサポートされます。

Cloud Credential Operator (CCO) Mint モードまたは Passthrough の場合、追加の設定は必要ありません。ただし、Security Token Service (STS) の場合は、**STS を使用した One Zone ファイルシステムのセットアップ** セクションの手順を使用します。

5.4.6.2. STS を使用した One Zone ファイルシステムのセットアップ

次の手順では、Security Token Service (STS) を使用して AWS One Zone ファイルシステムを設定する方法について説明します。

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。

手順

STS を使用して One Zone ファイルシステムを設定するには、以下を実行します。

1. **Security Token Service のロール Amazon Resource Name の取得**セクションの手順に従って、**credrequests** ディレクトリーに 2 つの **CredentialsRequests** を作成します。
   - **コントローラー** の **CredentialsRequest** の場合は、変更せずに手順に従います。
   - **ドライバーノード** の **CredentialsRequest** の場合は、次のサンプルファイルを使用します。

   ドライバーノードの CredentialsRequest YAML ファイルの例

   ```
   apiVersion: cloudcredential.openshift.io/v1
   kind: CredentialsRequest
   metadata:
     annotations:
   ```

```
    credentials.openshift.io/role-arns-vars: NODE_ROLEARN (1)
  name: openshift-aws-efs-csi-driver-node
  namespace: openshift-cloud-credential-operator
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
    - action:
      - elasticfilesystem:DescribeMountTargets
      - ec2:DescribeAvailabilityZones
      effect: Allow
      resource: '*'
  secretRef:
    name: node-aws-efs-cloud-credentials
    namespace: openshift-cluster-csi-drivers
  serviceAccountNames:
  - aws-efs-csi-driver-node-sa
```

1. **metadata.annotations.credentials.openshift.io/role-arns-vars** を **NODE_ROLEARN** に設定します。

ccoctl の出力例

```
2025/08/26 14:05:24 Role arn:aws:iam::269733383066:role/my-arn-1-blll6-openshift-cluster-csi-drivers-aws-efs-cloud-cre created (1)
2025/08/26 14:05:24 Saved credentials configuration to: /home/my-arn/project/go/src/github.com/openshift/myinst/aws-sts-compact-1/manifests/openshift-cluster-csi-drivers-aws-efs-cloud-credentials-credentials.yaml
2025/08/26 14:05:24 Updated Role policy for Role my-arn-1-blll6-openshift-cluster-csi-drivers-aws-efs-cloud-cre
2025/08/26 14:05:24 Role arn:aws:iam::269733383066:role/my-arn-1-blll6-openshift-cluster-csi-drivers-node-aws-efs-clou created (2)
2025/08/26 14:05:24 Saved credentials configuration to: manifests/openshift-cluster-csi-drivers-node-aws-efs-cloud-credentials-credentials.yaml
2025/08/26 14:05:24 Updated Role policy for Role my-arn-1-blll6-openshift-cluster-csi-drivers-node-aws-efs-clou
```

1. コントローラーの Amazon Resource Name (ARN)
2. ドライバーノードの ARN

2. この手順の前半で作成したコントローラー ARN を使用して、AWS EFS CSI ドライバーをインストールします。

3. 次のようなコマンドを実行して、Operator のサブスクリプションを編集し、ドライバーノードの ARN と **NODE_ROLEARN** を追加します。

```
$ oc -n openshift-cluster-csi-drivers edit subscription aws-efs-csi-driver-operator
...
 config:
  env:
  - name: ROLEARN
```

```
    value: arn:aws:iam::269733383066:role/my-arn-1-blll6-openshift-cluster-csi-drivers-aws-efs-cloud-cre (1)
  - name: NODE_ROLEARN
    value: arn:aws:iam::269733383066:role/my-arn-1-blll6-openshift-cluster-csi-drivers-node-aws-efs-clou (2)
...
```

1. コントローラーの ARN。すでに存在します。
2. ドライバーノードの ARN

5.4.7. Amazon Elastic File Storage 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式の動的プロビジョニングをサポートします。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

- Amazon Elastic File Storage (Amazon EFS) ボリュームが作成されている。
- 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

- AWS EFS ストレージクラスの作成

5.4.8. Amazon Elastic File Storage を使用した静的 PV の作成

動的プロビジョニングを行わずに、Amazon Elastic File Storage (Amazon EFS) ボリュームを単一の PV として使用できます。ボリューム全体が Pod にマウントされます。

前提条件

- Amazon EFS ボリュームが作成されている。

手順

- 以下の YAML ファイルで PV を作成します。

```yaml
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。

5.4.9. Amazon Elastic File Storage のセキュリティー

次の情報は、Amazon Elastic File Storage (Amazon EFS) のセキュリティーに重要です。

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は、https://docs.aws.amazon.com/efs/latest/ug/efs-access-points.html を参照してください。

結果として、EFS ボリュームは暗黙のうちに FSGroup を無視します。Red Hat OpenShift Service on AWS は、ボリューム上のファイルの GID を FSGroup に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。詳細は、https://docs.aws.amazon.com/efs/latest/ug/encryption-in-transit.html を参照してください。

5.4.10. AWS EFS ストレージ CSI 使用状況メトリクス

5.4.10.1. 使用状況メトリクスの概要

Amazon Web Services (AWS) Elastic File Service (EFS) Container Storage Interface (CSI) 使用状況メトリクスを使用すると、動的または静的にプロビジョニングされた EFS ボリュームによって使用されているスペースの量を監視できます。

重要

メトリクスを有効にするとパフォーマンスが低下する可能性があるため、この機能はデフォルトで無効になっています。

AWS EFS 使用状況メトリクス機能は、ボリューム内のファイルを再帰的にウォークスルーし、AWS EFS CSI ドライバーのボリュームメトリクスを収集します。この作業によりパフォーマンスが低下する可能性があるため、管理者はこの機能を明示的に有効にする必要があります。

5.4.10.2. Web コンソールを使用した使用状況メトリクスの有効化

Web コンソールを使用して Amazon Web Services (AWS) Elastic File Service (EFS) Container Storage Interface (CSI) の使用状況メトリクスを有効にするには、次の手順を実行します。

1. **Administration** > **CustomResourceDefinitions** をクリックします。
2. **CustomResourceDefinitions** ページの **Name** ドロップダウンボックスの横に、**clustercsidriver** と入力します。
3. **CRD ClusterCSIDriver** をクリックします。
4. **YAML** タブをクリックします。
5. **spec.aws.efsVolumeMetrics.state** の下で、値を **RecursiveWalk** に設定します。
   RecursiveWalk は、AWS EFS CSI ドライバーのボリュームメトリクス収集が、ボリューム内のファイルを再帰的にウォークスルーすることによって実行されることを示します。

   ClusterCSIDriver efs.csi.aws.com YAML ファイルの例

```
spec:
  driverConfig:
    driverType: AWS
    aws:
      efsVolumeMetrics:
        state: RecursiveWalk
        recursiveWalk:
          refreshPeriodMinutes: 100
          fsRateLimit: 10
```

6. オプション: 再帰ウォークの動作を定義するために、次のフィールドを設定することもできます。
    - **refreshPeriodMinutes**: ボリュームメトリクスの更新頻度を分単位で指定します。このフィールドを空白のままにすると、適切なデフォルトが選択されますが、これは時間の経過とともに変更される可能性があります。現在のデフォルトは 240 分です。有効な範囲は 1 - 43,200 分です。
    - **fsRateLimit**: ファイルシステムごとに、goroutine でボリュームメトリクスを処理するためのレート制限を定義します。このフィールドを空白のままにすると、適切なデフォルトが選択されますが、これは時間の経過とともに変更される可能性があります。現在のデフォルトは 5 つの goroutine です。有効な範囲は 1 - 100 の goroutine です。
7. **Save** をクリックします。

注記

AWS EFS CSI 使用状況メトリクスを **無効化** するには、前述の手順を使用しますが、**spec.aws.efsVolumeMetrics.state** の値を **RecursiveWalk** から **Disabled** に変更します。

5.4.10.3. CLI を使用した使用状況メトリクスの有効化

CLI を使用して Amazon Web Services (AWS) Elastic File Service (EFS) Container Storage Interface (CSI) 使用状況メトリクスを有効にするには、次の手順を実行します。

1. 次のコマンドを実行して ClusterCSIDriver を編集します。

```
$ oc edit clustercsidriver efs.csi.aws.com
```

2. **spec.aws.efsVolumeMetrics.state** の下で、値を **RecursiveWalk** に設定します。
RecursiveWalk は、AWS EFS CSI ドライバーのボリュームメトリクス収集が、ボリューム内のファイルを再帰的にウォークスルーすることによって実行されることを示します。

ClusterCSIDriver efs.csi.aws.com YAML ファイルの例

```
spec:
  driverConfig:
    driverType: AWS
    aws:
      efsVolumeMetrics:
        state: RecursiveWalk
```

```
          recursiveWalk:
            refreshPeriodMinutes: 100
            fsRateLimit: 10
```

3. オプション: 再帰ウォークの動作を定義するために、次のフィールドを設定することもできます。

   - **refreshPeriodMinutes**: ボリュームメトリクスの更新頻度を分単位で指定します。このフィールドを空白のままにすると、適切なデフォルトが選択されますが、これは時間の経過とともに変更される可能性があります。現在のデフォルトは 240 分です。有効な範囲は 1 - 43,200 分です。

   - **fsRateLimit**: ファイルシステムごとに、goroutine でボリュームメトリクスを処理するためのレート制限を定義します。このフィールドを空白のままにすると、適切なデフォルトが選択されますが、これは時間の経過とともに変更される可能性があります。現在のデフォルトは 5 つの goroutine です。有効な範囲は 1 - 100 の goroutine です。

4. **efs.csi.aws.com** オブジェクトへの変更を保存します。

> **注記**
>
> AWS EFS CSI 使用状況メトリクスを **無効化** するには、前述の手順を使用しますが、**spec.aws.efsVolumeMetrics.state** の値を **RecursiveWalk** から **Disabled** に変更します。

5.4.11. Amazon Elastic File Storage のトラブルシューティング

次の情報は、Amazon Elastic File Storage (Amazon EFS) に関する問題のトラブルシューティング方法に関するガイダンスです。

- 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、AWS が Red Hat OpenShift Service on AWS ノードと Amazon EFS の間でパケットを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

5.4.12. AWS EFS CSI Driver Operator のアンインストール

AWS EFS CSI Driver Operator (Red Hat 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ります。

前提条件

- Red Hat OpenShift Service on AWS 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 Driver 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. **Ecosystem** → **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 Operator を見つけてクリックします。

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、Red Hat OpenShift Service on AWS クラスターを破棄できません。Amazon はこのような VPC の削除を許可していません。

5.4.13. 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。

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。この PVC が存在する

間は、他の PVC と同じように使用できます。特に、これ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:
```

```yaml
  - 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。

永続ボリューム (PV) の縮小は **サポートされていません。**

前提条件

- 基盤となる CSI ドライバーがサイズ変更をサポートしている。「関連情報」セクションの「Red Hat OpenShift Service on AWS でサポートされている CSI ドライバー」を参照してください。
- 動的プロビジョニングを使用している。
- 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。Red Hat OpenShift Service on AWS は拡張中に PVC に **Resizing** 条件を追加します。この条件は拡張が完了すると削除されます。

7.3. 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。

7.4. ファイルシステムを使用した永続ボリューム要求 (PVC) の拡張

ファイルシステムのサイズ変更が必要なボリュームタイプ (AWS Elastic Block Store (EBS) など) に基づく PVC の拡張は、2 段階のプロセスです。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。

7.5. ボリューム拡張時の障害からの復旧

サイズ変更リクエストが失敗または保留状態のままになる場合は、永続ボリューム要求 (PVC) の **.spec.resources.requests.storage** に別のサイズ変更値を入力して再試行できます。新しい値は、元のボリュームサイズよりも大きくする必要があります。

まだ問題が解決しない場合は、次の手順に従って回復してください。

手順

PVC の **.spec.resources.requests.storage** に別のさらに小さいサイズ変更値を入力しても機能しない場合は、次の手順を実行します。

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。**persistentVolumeReclaimPolicy** を **Retain** に変更します。

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。

7.6. 関連情報

- ボリューム拡張サポートの有効化
- Red Hat OpenShift Service on AWS でサポートされている CSI ドライバー

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。

Red Hat OpenShift Service on AWS 永続ボリュームフレームワークは、この機能を有効にし、管理者が永続ストレージを使用してクラスター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。

Red Hat OpenShift Service on AWS では、多くのストレージタイプが永続ボリュームとして使用できます。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

Red Hat OpenShift Service on AWS は、以下のプロビジョナープラグインを提供します。これらのプラグインには、クラスターの設定済みプロバイダーの API を使用して新しいストレージリソースを作成する動的プロビジョニングの汎用実装があります。

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
Amazon Elastic Block Store (Amazon EBS)	**ebs.csi.aws.com**	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は、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。

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、デフォルトのストレージクラスをインストールします。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を説明します。

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: ebs.csi.aws.com (5)
parameters: (6)
  type: gp3-csi
...
```

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. 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: ebs.csi.aws.com
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

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。

8.4. デフォルトストレージクラスの変更

次の手順を使用して、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。

前提条件

- クラスター管理者権限でクラスターにアクセスできる。

手順

デフォルトのストレージクラスを変更するには、以下を実行します。

1. ストレージクラスを一覧表示します。

```
$ oc get storageclass
```

出力例

```
NAME            TYPE
gp3 (default)    ebs.csi.aws.com (1)
standard        ebs.csi.aws.com
```

1. **(default)** はデフォルトのストレージクラスを示します。

2. 目的のストレージクラスをデフォルトにします。
目的のストレージクラスに、次のコマンドを実行して **storageclass.kubernetes.io/is-default-class** アノテーションを **true** に設定します。

```
$ oc patch storageclass standard -p '{"metadata": {"annotations":
{"storageclass.kubernetes.io/is-default-class": "true"}}}'
```

> **注記**
>
> 短期間であれば、複数のデフォルトのストレージクラスを使用できます。ただし、最終的には1つのデフォルトのストレージクラスのみが存在することを確認する必要があります。
>
> 複数のデフォルトストレージクラスが存在する場合、デフォルトストレージクラス (**pvc.spec.storageClassName** =nil) を要求するすべての永続ボリューム要求 (PVC) は、そのストレージクラスのデフォルトステータスと管理者に関係なく、最後に作成されたデフォルトストレージクラスを取得します。アラートダッシュボードで、複数のデフォルトストレージクラス **MultipleDefaultStorageClasses** があるというアラートを受け取ります。

3. 古いデフォルトストレージクラスからデフォルトのストレージクラス設定を削除します。
   古いデフォルトのストレージクラスの場合は、次のコマンドを実行して **storageclass.kubernetes.io/is-default-class** アノテーションの値を **false** に変更します。

   ```
   $ oc patch storageclass gp3 -p '{"metadata": {"annotations": {"storageclass.kubernetes.io/is-default-class": "false"}}}'
   ```

4. 変更内容を確認します。

   ```
   $ oc get storageclass
   ```

   出力例

   ```
   NAME               TYPE
   gp3                ebs.csi.aws.com
   standard (default)   ebs.csi.aws.com
   ```