OpenShift Dedicated 4

ストレージ

OpenShift Dedicated 4 でのストレージの設定および管理

Last Updated: 2020-05-15

OpenShift Dedicated 4 ストレージ

OpenShift Dedicated 4 でのストレージの設定および管理

法律上の通知

概要

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

目次

第1章 永続ストレージについて

1.1. 永続ストレージの概要

ストレージの管理は、コンピュートリソースの管理とは異なります。OpenShift Dedicated 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、Persistent Volume Claim (永続ボリューム要求、PVC) を使用すると、基礎となるストレージインフラストラクチャーについての特定の知識がなくても PV リソースを要求することができます。

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

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

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

重要

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

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

1.2. 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。

1.2.1. 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。

1.2.2. 要求のバインド

PVC の作成時に、ストレージの特定容量の要求、必要なアクセスモードの指定のほか、ストレージクラスを作成してストレージの記述や分類を行います。マスターのコントロールループは新規 PVC の有無を監視し、新規 PVC を適切な PV にバインドします。適切な PV がない場合には、ストレージクラスのプロビジョナーが PV を作成します。

すべての PV のサイズが PVC サイズを超える可能性があります。これは、手動でプロビジョニングされる PV にとくに当てはまります。超過を最小限にするために、OpenShift Dedicated 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。

1.2.3. Pod および要求した PV の使用

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

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

1.2.4. ボリュームの解放

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

1.2.5. ボリュームの回収

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

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

重要

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

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

注記

動的にプロビジョニングされたボリュームは常に削除されます。

1.3. 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。

PV オブジェクト定義例

```
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。

1.3.1. PV の種類

OpenShift Dedicated は以下の **PersistentVolume** プラグインをサポートします。

- AWS Elastic Block Store (EBS)

1.3.2. 容量

通常、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。

1.3.3. アクセスモード

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

要求は、同様のアクセスモードのボリュームに一致します。一致する条件はアクセスモードとサイズの 2 つの条件のみです。要求のアクセスモードは要求 (request) を表します。そのため、より多くのアクセスを付与することはできますが、アクセスを少なくすることはできません。たとえば、要求により

RWO が要求されるものの、利用できる唯一のボリュームが NFS PV (RWO+ROX+RWX) の場合に、要求は RWO をサポートする NFS に一致します。

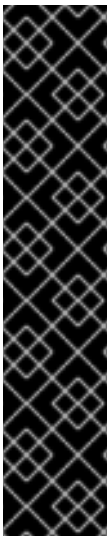
直接的なマッチングが常に最初に試行されます。ボリュームのモードは、要求モードと一致するか、要求した内容以上のものを含む必要があります。サイズは予想されるものより多いか、またはこれと同等である必要があります。2 つのタイプのボリューム (NFS および iSCSI など) のどちらにも同じセットのアクセスモードがある場合、それらのいずれかがそれらのモードを持つ要求に一致する可能性があります。ボリュームのタイプ間で順序付けすることはできず、タイプを選択することはできません。

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

以下の表では、アクセスモードをまとめています。

表1.1 アクセスモード

アクセスモード	CLI の省略形	説明
ReadWriteOnce	**RWO**	ボリュームは単一ノードで読み取り/書き込みとしてマウントできます。



重要

ボリュームの **AccessModes** は、ボリュームの機能の記述子です。それらは施行されている制約ではありません。ストレージプロバイダーはリソースの無効な使用から生じるランタイムエラーに対応します。

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

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

表1.2 サポート対象の PV 向けアクセスモード

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

注記

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。

1.3.4. 制限

OpenShift Dedicated で PV を使用する場合には以下の制限が適用されます。

- PV は、クラスターがプロビジョニングされる場所に応じて EBS ボリューム (AWS) または GCP ストレージ (GCP) のいずれかでプロビジョニングされます。
- EBS ボリュームおよび GCE 永続ディスクは複数のノードにマウントできないので、RWFS アクセスモードのみを使用できます。
- **emptyDir** には Pod と同じライフサイクルがあります。
  - **emptyDir** ボリュームはコンテナーのクラッシュ/再起動後も永続します。
  - **emptyDir** ボリュームは Pod の削除時に削除されます。

1.3.5. 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1.3 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	要求が検出されていますが、リソースがまだクラスターにより回収されていません。
Failed	ボリュームが自動回収に失敗しています。

以下を実行して PV にバインドされている PVC の名前を表示できます。

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

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

各 Persistent Volume Claim (永続ボリューム要求、PVC) には、要求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。 以下に例を示します。

PVC オブジェクト定義例

```
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

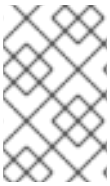
1.4.1. ストレージクラス

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

ClusterStorageOperator は、使用されるプラットフォームによってデフォルトの StorageClass をインストールする可能性があります。この StorageClass 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 StorageClass 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 StorageClass 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 StorageClass のみをデフォルトとして設定する必要があります。

1.4.2. 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。

1.4.3. 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、これ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。

1.4.4. ボリュームとしての要求

Pod は要求をボリュームとして使用することでストレージにアクセスします。この要求を使用して、Pod と同じ namespace 内に要求を共存させる必要があります。クラスターは Pod の namespace で要

求を見つけ、これを使用して要求をサポートする **PersistentVolume** を取得します。以下のように、ボリュームはホストにマウントされ、Pod に組み込まれます。

ホストおよび Pod のサンプルへのボリュームのマウント

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

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

第2章 永続ボリュームの拡張

2.1. OPENSHIFT DEDICATED の PERSISTENT VOLUME CLAIM（永続ボリューム要求、PVC) の拡張

ファイルサイズのサイズ変更を必要とするボリュームタイプ(AWS EBS など) に基づいて PVC を拡張するには 2 つの手順からなるプロセスが必要です。このプロセスでは、クラウドプロバイダーでボリュームオブジェクトを拡張してから実際のノードでファイルシステムを拡張します。これらのステップは、PVC オブジェクトの編集後に自動的に実行され、Pod の再起動が必要になる場合があります。

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 では、**allowVolumeExpansion** が **true** に設定されている必要があります。これは OpenShift Dedicated のデフォルト設定です。

重要

Amazon Elastic Block Store (EBS) ボリュームのサイズを縮小することはサポートされません。ただし、小規模なボリュームを作成し、**oc rsync**などのツールを使用してデータをそのボリュームに移行することができます。ボリュームの変更後、同じボリュームに追加の変更を行うには少なくとも 6 時間待機する必要があります。

手順

1. **spec.resources.requests** 値を編集して PVC を編集し、新規サイズを要求します。以下の **oc patch** コマンドは PVC のサイズを変更します。

```
$ oc patch pvc <pvc_name> -p '{"spec": {"resources": {"requests": {"storage": "8Gi"}}}}'
```

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** に設定される可能性があります。以下のコマンドは、状態を確認するために使用されます。

```
$ oc describe pvc mysql

Name:          mysql
Namespace:     my-project
StorageClass:  gp2
Status:        Bound
Volume:        pvc-5fa3feb4-7115-4735-8652-8ebcfec91bb9
Labels:        app=cakephp-mysql-persistent
               template=cakephp-mysql-persistent
               template.openshift.io/template-instance-owner=6c7f7c56-1037-4105-8c08-
55a6261c39ca
Annotations:   pv.kubernetes.io/bind-completed: yes
               pv.kubernetes.io/bound-by-controller: yes
               volume.beta.kubernetes.io/storage-provisioner: kubernetes.io/aws-ebs
               volume.kubernetes.io/selected-node: ip-10-0-128-221.us-east-2.compute.internal
               volume.kubernetes.io/storage-resizer: kubernetes.io/aws-ebs
```

```
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:      1Gi (1)
Access Modes:  RWO
VolumeMode:    Filesystem
Conditions: (2)
  Type                 Status  LastProbeTime     LastTransitionTime  Reason  Message
  ----                 ------  -----------------  ------------------  ------  -------
  FileSystemResizePending  True    <Timestamp>       <Timestamp>               Waiting for
user to (re-)start a Pod to
                                                                               finish file system resize of volume on
node.
Events:
  Type      Reason              Age   From                         Message
  ----      ------              ----  ----                         -------
  Normal    WaitForFirstConsumer   36m   persistentvolume-controller  waiting for first
consumer to be created before binding
  Normal    ProvisioningSucceeded  36m   persistentvolume-controller  Successfully
provisioned volume
                                                                     pvc-5fa3feb4-7115-4735-8652-8ebcfec91bb9
using
                                                                     kubernetes.io/aws-ebs
Mounted By:  mysql-1-q4nz7 (3)
```

(1) PVC の現在の容量。

(2) 関連する状態がここに表示されます。

(3) このボリュームを現在マウントしている Pod

3. 直前のコマンドの出力に Pod を再起動するためのメッセージが含まれる場合、その出力で指定されたマウントしている Pod を削除します。

```
$ oc delete pod mysql-1-q4nz7
```

4. Pod が実行されている場合、新たに要求されたサイズが利用可能になり、**FileSystemResizePending** 状態は PVC から削除されます。