Red Hat OpenShift Container Storage 4.8

Configuring OpenShift Container Storage for Metro-DR stretch cluster

クラスターおよびストレージ管理者の災害復旧タスク

Last Updated: 2023-02-21

Red Hat OpenShift Container Storage 4.8 Configuring OpenShift Container Storage for Metro-DR stretch cluster

クラスターおよびストレージ管理者の災害復旧タスク

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

法律上の通知

概要

このソリューションガイドの目的は、OpenShift Container Storage を Kubernetes ゾーントポロジーラベルと共にデプロイするのに必要な手順およびコマンドの詳細を提供し、可用性の高いストレージインフラストラクチャーを実現します。 This is a technology preview feature and is available only for deployment using local storage devices. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

目次

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、弊社の CTO である Chris Wright のメッセージ を参照してください。

RED HAT ドキュメントへのフィードバック (英語のみ)

弊社のドキュメントについてのご意見をお聞かせください。ドキュメントの改善点があれば、ぜひお知らせください。フィードバックをお寄せいただくには、以下をご確認ください。

- 特定の部分についての簡単なコメントをお寄せいただく場合は、以下をご確認ください。
  1. ドキュメントの表示が **Multi-page HTML** 形式になっていていることを確認してください。ドキュメントの右上隅に **Feedback** ボタンがあることを確認してください。
  2. マウスカーソルを使用して、コメントを追加するテキストの部分を強調表示します。
  3. 強調表示されたテキストの下に表示される **Add Feedback** ポップアップをクリックします。
  4. 表示される指示に従ってください。
- より詳細なフィードバックをお寄せいただく場合は、Bugzilla のチケットを作成してください。
  1. Bugzilla の Web サイトに移動します。
  2. **Component** セクションで、**documentation** を選択します。
  3. **Description** フィールドに、ドキュメントの改善に向けたご提案を記入してください。ドキュメントの該当部分へのリンクも追加してください。
  4. **Submit Bug** をクリックします。

第1章 ストレッチクラスターを使用したメトロ災害復旧の概要

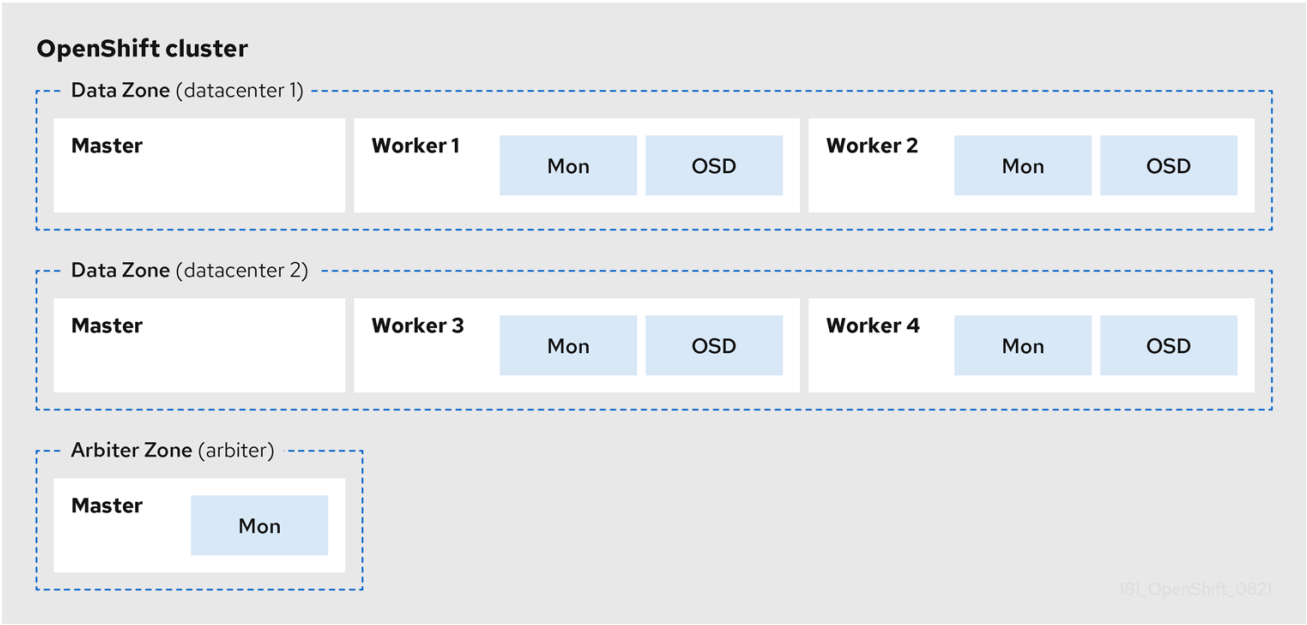
Red Hat OpenShift Container Storage デプロイメントは、災害復旧機能を備えたストレージインフラストラクチャーを提供するために、2 つの異なる地理的ロケーション間で展開できます。2 つの拠点のうちどちらかが部分的または完全に利用できないといった災害に直面した場合、OpenShift Container Platform のデプロイメント上にデプロイされた OpenShift Container Storage が存続できるようにしなければなりません。このソリューションは、インフラストラクチャーのサーバー間に特定のレイテンシー要件があるメトロポリタンに広がるデータセンターでのみ利用できます。

注記

現在、ストレッチクラスターを使用した Metro-DR ソリューションは、レイテンシーが、異なる場所にある OpenShift Container Platform ノード間で 4 ミリ秒のラウンドトリップタイム (RTT) を超えない場合にデプロイできます。より高いレイテンシーでデプロイする予定がある場合は、Red Hat カスタマーサポートにお問い合わせください。

以下の図は、Metro-DR ストレッチクラスターの最も簡単なデプロイメントを示しています。

OpenShift ノードおよび OpenShift Container Storage デーモン

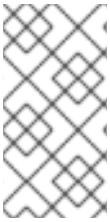


この図では、Arbiter ゾーンにデプロイされた OpenShift Container Storage モニターの Pod にはマスターノード向けの耐性が組み込まれています。マスターノードは、可用性の高い OpenShift Container Platform コントロールプレーンに必要です。また、1 つのゾーンの OpenShift Container Platform ノードに、他の 2 つのゾーン内の OpenShift Container Platform ノードとのネットワーク接続があることが重要です。

第2章 災害復旧を有効にしてストレージクラスターをデプロイする準備

2.1. METRO-DR を有効にする要件

- 3 つの異なるゾーンに 3 つ以上の OpenShift Container Platform マスターノードがあることを確認してください。3 つのゾーンのそれぞれに 1 つのマスターノード。
- また、4 つ以上の OpenShift Container Platform ワーカーノードが 2 つの Data Zone に均等に分散されていることを確認します。
- Bare Metal 上のストレッチクラスターの場合は、SSD ドライブを OpenShift Container Platform マスターノードのルートドライブとして使用する必要があります。
- 各ノードのゾーンラベルで事前にラベル付けされていることを確認します。詳細は、OpenShift Container Platform ノードへのトポロジーゾーンラベルの適用 セクションを参照してください。
- Metro-DR ソリューションは、レイテンシーがゾーン (RTT は最大 4ms) 間の 2 ミリ秒を超えない状態でデプロイするように設計されています。より高いレイテンシーでデプロイする予定がある場合は、Red Hat カスタマーサポート にお問い合わせください。



注記

スケーリングロジックが競合しているため、柔軟なスケーリングと arbiter の両方を有効にすることはできません。Flexible scaling を使用すると、1 度に 1 つのノードを OpenShift Container Storage クラスターに追加することができます。arbiter クラスターでは、2 つのデータゾーンごとに 1 つ以上のノードを追加する必要があります。

2.2. トポロジーゾーンラベルの OPENSHIFT CONTAINER PLATFORM ノードへの適用

サイトの停止時に、arbiter 関数を持つゾーンは arbiter ラベルを使用します。これらのラベルは任意となるため、3 つの場所で一意にする必要があります。

たとえば、以下のようにノードにラベルを付けることができます。

```
topology.kubernetes.io/zone=arbiter for Master0

topology.kubernetes.io/zone=datacenter1 for Master1, Worker1, Worker2

topology.kubernetes.io/zone=datacenter2 for Master2, Worker3, Worker4
```

- oc CLI を使用してラベルをノードに適用するには、以下を実行します。

```
$ oc label node <NODENAME> topology.kubernetes.io/zone=<LABEL>
```

- ラベルを検証するには、3 つゾーンのサンプルラベルを使用して以下のコマンドを実行します。

```
$ oc get nodes -l topology.kubernetes.io/zone=<LABEL> -o name
```

または、1つのコマンドを実行して、そのゾーンを持つすべてのノードを表示できます。

```
$ oc get nodes -L topology.kubernetes.io/zone
```

Metro DR ストレッチクラストポロジーのゾーンラベルが適切な OpenShift Container Platform ノードに適用され、3 つのロケーションが定義されました。

次のステップ

- OpenShift Container Platform OperatorHub からのストレージ Operator のインストール

2.3. ローカルストレージ OPERATOR のインストール

Local Storage Operator は、Red Hat OpenShift Container Platform Operator Hub を使用してインストールできます。

前提条件

- cluster-admin および Operator インストールのパーミッションを持つアカウントを使用して OpenShift Container Platform クラスターにアクセスできること。

手順

1. OpenShift Web コンソールにログインします。
2. **Operators → OperatorHub** をクリックします。
3. **Filter by keyword...** ボックスに **local storage** と入力して、オペレータのリストから **Local Storage** オペレーターを検索し、クリックします。
4. **Install** をクリックします。
5. **Install Operator** ページで、以下のオプションを設定します。
   a. Channel を **stable-4.8** として更新します。
   b. Installation Mode オプションに **A specific namespace on the cluster**を選択します。
   c. Installed Namespace に **Operator recommended namespace openshift-local-storage** を選択します。
   d. Approval Strategy に **Automatic** を選択します。
6. **Install** をクリックします。

検証手順

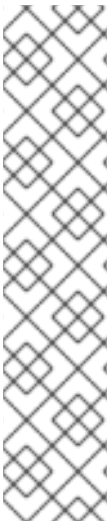
- ローカルストレージ Operator が **ステータス Succeeded** を表示していることを確認します。

2.4. RED HAT OPENSHIFT CONTAINER STORAGE OPERATOR のインストール

Red Hat OpenShift Container Storage は、Red Hat OpenShift Container Platform Operator Hub を使用してインストールできます。

前提条件

- cluster-admin および Operator インストールのパーミッションを持つアカウントを使用して OpenShift Container Platform クラスターにアクセスできること。
- Red Hat OpenShift Container Platform クラスター内の 2 つのデータセンターに均等に分散された 4 ワーカーノードがあります。
- その他のリソース要件については、デプロイメントのプランニング を参照してください。



注記

- OpenShift Container Storage のクラスター全体でのデフォルトノードセレクターを上書きする必要がある場合は、コマンドラインインターフェイスで以下のコマンドを使用し、**openshift-storage** namespace の空のノードセレクターを指定できます (この場合、openshift-storage namespace を作成します)。

```
$ oc annotate namespace openshift-storage openshift.io/node-selector=
```

- ノードに Red Hat OpenShift Container Storage リソースのみがスケジュールされるように、そのノードに **infra** のテイントを設定します。これにより、サブスクリプションコストを節約できます。詳細は、ストレージリソースの管理および割り当てガイドの Red Hat OpenShift Container Storage に専用のワーカーノードを使用する方法 の章を参照してください。

手順

1. Web コンソールで **Operators → OperatorHub** ページに移動し、クリックします。
2. スクロールするか、またはキーワードを Filter by keyword ボックスに入力し、OpenShift Container Storage Operator を検索します。
3. OpenShift Container Storage Operator ページで、**Install** をクリックします。
4. **Install Operator** ページで、以下の必須オプションがデフォルトで選択されます。
   a. Channel を **stable-4.8** として更新します。
   b. Installation Mode オプションに **A specific namespace on the cluster**を選択します。
   c. Installed Namespace に **Operator recommended namespace openshift-storage** を選択します。namespace **openshift-storage** が存在しない場合、これは Operator のインストール時に作成されます。
   d. **承認ストラテジー** を **Automatic** または **Manual** として選択します。
   e. **Install** をクリックします。
   Automatic (自動) 更新を選択している場合、Operator Lifecycle Manager (OLM) は介入なしに、Operator の実行中のインスタンスを自動的にアップグレードします。

   Manual (手動) 更新を選択している場合、OLM は更新要求を作成します。クラスター管理者は、Operator が新規バージョンに更新されるように更新要求を手動で承認する必要があります。

検証手順

OpenShift Container Storage Operator に、インストールが正常に実行されたことを示す緑色のチェックマークが表示されていることを確認します。

次のステップ

- OpenShift Container Storage クラスターを作成します。
  詳細は、OpenShift Container Storage クラスターの作成 を参照してください。

第3章 OPENSHIFT CONTAINER STORAGE クラスターの作成

以下の手順を使用して、OpenShift Container Storage オペレーターのインストール後に OpenShift Container Storage クラスターを作成します。

前提条件

- Preparing to deploy storage cluster with disaster recovery enabled セクションのすべての要件を満たしていることを確認してください。

手順

1. OpenShift Web コンソールにログインします。
2. **Operators → Installed Operators** をクリックし、インストールされた Operator をすべて表示します。
   選択された **Project** が **openshift-storage** であることを確認します。
3. Storage Cluster の **OpenShift Container Storage → Create Instance** リンクをクリックします。
4. **Internal-Attached devices** としてモードを選択します。
   インストールされていない場合に、ローカルストレージ Operator をインストールすることを求めるプロンプトが出されます。**Install** をクリックし、ローカルストレージ Operator のインストール で説明されているように手順に従います。

   ストレージボリュームのセットをフィルターすることにより、専用のストレージクラスを作成してストレージを消費できます。
5. ディスクの検出
   a. **Select nodes** オプションを使用して、data-center ゾーンから割り当てられたストレージデバイスを持つレベル付けされたノードを選択します。
      選択したノードにテイントのマークが付けられており、そのノードがウィザードで検出されない場合は、ローカルストレージ Operator リソースの容認を追加する回避策として Red Hat ナレッジベースソリューション に記載されている手順に従ってください。

      選択したノードが集約された 30 CPU および 72 GiB の RAM の OpenShift Container Storage クラスターの要件と一致しない場合は、最小クラスターがデプロイされます。
      ノードの最小要件については、プランニングガイドの リソース要件 セクションを参照してください。

   b. **Next** をクリックします。
6. ストレージクラスの作成
   a. **Local Volume Set Name** を入力します。

   b. **Storage Class Name** を入力します。デフォルトで、ボリュームセット名がストレージクラス名について表示されます。名前を変更することもできます。

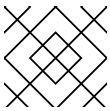
   c. 直前の手順でディスク検出で選択されたノードは **Filter Disks** セクションに表示されます。
      以下のいずれかを選択します。

      - **Disks on all nodes**: ディスクを検出したすべてのノードを選択します。

- **Disks on selected nodes** ディスクを検出したノードのサブセットを選択します。高可用性を確保するために、ワーカーノードは 3 つの異なる物理ノード、ラック、障害ドメインに分散します。

d. **SSD または NVMe** を選択して、サポートされる設定を構築します。サポート対象外のテストインストールに **HDD** を選択できます。

e. **Advanced** セクションを拡張し、以下のオプションを設定します。

ボリュームモード	デフォルトではブロックが選択されます。
デバイスタイプ	ディスクタイプを選択します。デフォルトでは、Disk および Part が選択されます。
ディスクサイズ	含める必要のあるデバイスの最小および最大の許容サイズ。  **注記** デバイスの最小サイズ 100GB を設定する必要があります。
最大ディスク制限	これは、ノードで作成できる PV の最大数を示します。このフィールドが空のままの場合、PV は一致するノードで利用可能なすべてのディスクに作成されます。

f. **Next** をクリックします。新規ストレージクラスの作成を確認するポップアップが表示されます。

g. **Yes** をクリックして続行します。

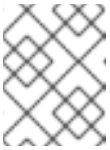
7. **Capacity and nodes** を設定します。

a. ストレッチクラスターを使用する必要がある場合は、**Enable arbiter** チェックボックスを選択します。

- 利用可能なドロップダウンリストから **arbiter zone** を選択します。

b. **Storage Class** を選択します。デフォルトで、直前の手順で作成された新規ストレージクラスが選択されます。

c. **選択したノード** には、直前の手順で選択したノードが表示されます。この一覧では、直前の手順で検出されたディスクを表示するのに数分かかる場合があります。



注記

選択したノードセクションに表示される各ノードのゾーンラベルを確認して、それらが正しくラベル付けされていることを確認します。

d. **Next** をクリックします。

8. (オプション) **Security and network** 設定を設定します。

a. **Enable encryption** チェックボックスを選択して、ブロックおよびファイルストレージを暗号化します。

b. 1つまたは両方の **Encryption level** を選択します。

- **クラスター全体の暗号化** クラスター全体 (ブロックおよびファイル) を暗号化します。
- **Storage class encryption**(ストレージクラスの暗号化): 暗号化対応のストレージクラスを使用して暗号化された永続ボリューム (ブロックのみ) を作成します。

c. **Connect to an external key management service**チェックボックスを選択します。これはクラスター全体の暗号化の場合はオプションになります。

i. **Key Management Service Provider**はデフォルトで **Vault** に設定されます。

ii. Vault **Service Name**、Vault サーバーのホスト **Address**('https://<hostname または ip>')、**Port number** および **Token** を入力します。

iii. **Advanced Settings** を展開して、Vault 設定に基づいて追加の設定および証明書の詳細を入力します。

A. OpenShift Container Storage 専用かつ特有のキー値のシークレットパスを **Backend Path** に入力します。

B. (オプション) **TLS Server Name** および **Vault Enterprise Namespace** を入力します。

C. それぞれの PEM でエンコードされた証明書ファイルをアップロードして、**CA Certificate**、**Client Certificate**、および **Client Private Key** を指定します。

D. **Save** をクリックします。

d. 単一のネットワークを使用する場合は **Default** (SDN) を選択し、複数のネットワークインターフェイスを使用する場合は **Custom** (Multus) ネットワークを選択します。

i. ドロップダウンメニューから **Public Network Interface** を選択します。

ii. ドロップダウンメニューから **Cluster Network Interface** を選択します。

e. **Next** をクリックします。

9. 設定の詳細を確認します。設定を変更するには、**Back** をクリックして以前の設定ページに戻ります。

10. **Create** をクリックします。

検証手順

1. インストールされたストレージクラスターの最後の **Status** が緑色のチェックマークと共に **Phase: Ready** と表示されていることを確認します。

a. **Operators → Installed Operators → Storage Cluster** のリンクをクリックして、ストレージクラスターのインストールのステータスを表示します。

b. または、Operator **Details** タブで、**Storage Cluster** タブをクリックすると、ステータスを表示できます。

2. **Storage Cluster** タブで **ocs-storagecluster** をクリックします。

a. YAML タブで、spec セクションで **arbiter** キーを検索し、以下を確認します。

- 'enable' が **true** に設定されている。
- 'arbiterLocation' が **arbiter** に設定されている。
- 'replica' が **4** に設定されている。
- 'failureDomain' が **zone** に設定されている。

```
spec:
    arbiter:
      enable: true
    [..]
    nodeTopologies:
      arbiterLocation: arbiter
        [..]
      replica: 4
status:
    conditions:
    [..]
    failureDomain: zone
    [..]
```

3. OpenShift Container Storage のすべてのコンポーネントが正常にインストールされていることを確認するには、OpenShift Container Storage インストールの確認 を参照してください。

第4章 OPENSHIFT CONTAINER STORAGE デプロイメントの検証

このセクションを使用して、OpenShift Container Storage が正常にデプロイされていることを確認します。

4.1. POD の状態の確認

OpenShift Containers Storage の Pod が実行状態にあることを確認するには、以下の手順に従います。

手順

1. OpenShift Web コンソールにログインします。
2. OpenShift Web コンソールの左側のペインから **Workloads → Pods** をクリックします。
3. **Project** ドロップダウンリストから **openshift-storage** を選択します。
   各コンポーネントについて予想される Pod 数や、これがノード数によってどのように異なるかについての詳細は、表4.1「OpenShift Container Storage クラスターに対応する Pod」 を参照してください。
4. **Running** タブおよび **Completed** タブをクリックして、Pod が実行中で完了状態になっていることを確認します。

表4.1 OpenShift Container Storage クラスターに対応する Pod

コンポーネント	対応する Pod
OpenShift Container Storage Operator	• **ocs-operator-*** (任意のワーカーノードに 1 Pod) • **ocs-metrics-exporter-***
Rook-ceph Operator	**rook-ceph-operator-*** (任意のワーカーノードに 1 Pod)
Multicloud Object Gateway	• **noobaa-operator-*** (任意のワーカーノードに 1 Pod) • **noobaa-core-*** (任意のストレージノードに 1 Pod) • **noobaa-db-*** (任意のストレージノードに 1 Pod) • **noobaa-endpoint-*** (任意のストレージノードに 1 Pod)

コンポーネント	対応する Pod
MON	**rook-ceph-mon-*** (5 Pod は data-center ゾーンごとに 2 つ、arbiter ゾーンに 1 つと、3 つのゾーンに分散されます)。
MGR	**rook-ceph-mgr-*** (任意のストレージノード上の 2 つの Pod)
MDS	**rook-ceph-mds-ocs-storagecluster-cephfilesystem-*** (2 つの Pod が 2 つのデータセンターゾーンに分散されている)
RGW	**rook-ceph-rgw-ocs-storagecluster-cephobjectstore-*** (2 つの Pod が 2 つのデータセンターゾーンに分散されている)

CSI	• **cephfs**
	  ◦ **csi-cephfsplugin-*** (各ワーカーノードに 1 Pod)
	  ◦ **csi-cephfsplugin-provisioner-*** (ワーカーノードに分散する 2 Pod)
	• **rbd**
	  ◦ **csi-rbdplugin-*** (各ワーカーノードに 1 Pod)
	  ◦ **csi-rbdplugin-provisioner-*** (ストレージノードに分散する 2 Pod)

rook-ceph-crashcollector	**rook-ceph-crashcollector-*** (各ストレージノードに 1 Pod、arbiter ゾーンの 1 Pod)
OSD	• **rook-ceph-osd-*** (各デバイス用に 1 Pod) • **rook-ceph-osd-prepare-ocs-deviceset-*** (各デバイス用に 1 Pod)

4.2. OPENSHIFT CONTAINER STORAGE クラスターが正常であることの確認

OpenShift Container Storage のクラスターが正常であることを確認するには、手順の手順に従います。

手順

1. **Storage → Overview** をクリックし、**Block and File** タブをクリックします。
2. **Status カードで**、**Storage Cluster** および **Data Resiliency** に緑色のチェックマークが表示されていることを確認します。
3. **Details カード** で、クラスター情報が表示されていることを確認します。

ブロックおよびファイルダッシュボードを使用した OpenShift Container Storage クラスターの正常性については、OpenShift Container Storage のモニターリング を参照してください。

4.3. MULTICLOUD OBJECT GATEWAY が正常であることの確認

OpenShift Container Storage Multicloud Object Gateway が正常であることを確認するには、手順のステップに従います。

手順

1. OpenShift Web コンソールから **Storage → Overview** をクリックし、**Object** タブをクリックします。
2. **Status card** で、**Object Service** と **Data Resiliency** の両方が **Ready** 状態 (緑のチェックマーク) にあることを確認します。
3. **Details カード** で、Multicloud Object Gateway 情報が表示されることを確認します。

オブジェクトサービスダッシュボードを使用した OpenShift Container Storage クラスターの正常性については、OpenShift Container Storage のモニターリング を参照してください。

4.4. OPENSHIFT CONTAINER STORAGE 固有のストレージクラスが存在することの確認

ストレージクラスがクラスターに存在することを確認するには、手順のステップに従います。

手順

1. OpenShift Web コンソールから **Storage → Storage Classes**をクリックします。
2. 以下のストレージクラスが OpenShift Container Storage クラスターの作成時に作成されることを確認します。
   - **ocs-storagecluster-ceph-rbd**
   - **ocs-storagecluster-cephfs**
   - **openshift-storage.noobaa.io**
   - **ocs-storagecluster-ceph-rgw**

第5章 ゾーン対応サンプルアプリケーションのインストール

このセクションを使用では、OpenShift Container Storage Metro Disaster 復旧設定を検証するためにゾーン対応のサンプルアプリケーションをデプロイします。

重要

データゾーン間のレイテンシーがあると、ノードやゾーン間のレイテンシーが低い (たとえば、すべてのノードが同じ場所にある) OpenShift クラスターと比較して、パフォーマンスの低下が予想されます。どの程度パフォーマンスが低下するかは、ゾーン間のレイテンシーや、ストレージを使用するアプリケーションの動作 (書き込みトラフィックが多いなど) によって異なります。必要なサービスレベルに対して十分なアプリケーションのパフォーマンスを確保するために、必ず Metro DR クラスター設定で重要なアプリケーションをテストしてください。

5.1. ゾーン認識サンプルアプリケーションのインストール

このセクションでは、**ocs-storagecluster-cephfs** ストレージクラスを使用して、同時に複数の Pod で使用できる Read-Write-Many (RWX) PVC を作成します。使用するアプリケーションは File Uploader と呼ばれます。

このアプリケーションは、ファイルを保存するために同じ RWX ボリュームを共有しているため、トポロジーゾーンをまたいでアプリケーションを展開し、サイトが停止した場合でもアプリケーションを利用できるようにすることができます。OpenShift Container Storage は、ゾーン認識および高可用性を備えた Metro DR ストレッチクラスターとして設定されているため、これは永続的なデータアクセスにも有効です。

1. 新しいプロジェクトを作成します。

```
oc new-project my-shared-storage
```

2. file-uploader という名前の PHP アプリケーションのサンプルをデプロイします。

```
oc new-app openshift/php:7.2-ubi8~https://github.com/christianh814/openshift-php-upload-demo --name=file-uploader
```

出力サンプル

```
Found image 4f2dcc0 (9 days old) in image stream "openshift/php" under tag "7.2-ubi8" for
"openshift/php:7.2-
ubi8"

Apache 2.4 with PHP 7.2
-----------------------
PHP 7.2 available as container is a base platform for building and running various PHP 7.2
applications and frameworks. PHP is an HTML-embedded scripting language. PHP attempts
to make it easy for developers to write dynamically generated web pages. PHP also offers
built-in database integration for several commercial and non-commercial database
management systems, so writing a database-enabled webpage with PHP is fairly simple.
The most common
use of PHP coding is probably as a replacement for CGI scripts.

Tags: builder, php, php72, php-72
```

```
* A source build using source code from https://github.com/christianh814/openshift-php-upload-demo will be cr
eated
* The resulting image will be pushed to image stream tag "file-uploader:latest"
* Use 'oc start-build' to trigger a new build

--> Creating resources ...
    imagestream.image.openshift.io "file-uploader" created
    buildconfig.build.openshift.io "file-uploader" created
    deployment.apps "file-uploader" created
    service "file-uploader" created
--> Success
    Build scheduled, use 'oc logs -f buildconfig/file-uploader' to track its progress.

    Application is not exposed. You can expose services to the outside world by executing one or more of the commands below:
     'oc expose service/file-uploader'

    Run 'oc status' to view your app.
```

3. ビルドログを表示し、アプリケーションがデプロイされるまで待機します。

```
oc logs -f bc/file-uploader -n my-shared-storage
```

出力例:

```
Cloning "https://github.com/christianh814/openshift-php-upload-demo" ...

    [...]
    Generating dockerfile with builder image image-registry.openshift-image-regis
try.svc:5000/openshift/php@sha256:d97466f33999951739a76bce922ab17088885db610c
0e05b593844b41d5494ea
STEP 1: FROM image-registry.openshift-image-registry.svc:5000/openshift/php@s
ha256:d97466f33999951739a76bce922ab17088885db610c0e05b593844b41d5494ea
STEP 2: LABEL "io.openshift.build.commit.author"="Christian Hernandez <christ
ian.hernandez@yahoo.com>"       "io.openshift.build.commit.date"="Sun Oct 1 1
7:15:09 2017 -0700"       "io.openshift.build.commit.id"="288eda3dff43b02f7f7
b6b6b6f93396ffdf34cb2"       "io.openshift.build.commit.ref"="master"       "
io.openshift.build.commit.message"="trying to modularize"       "io.openshift
.build.source-location"="https://github.com/christianh814/openshift-php-uploa
d-demo"       "io.openshift.build.image"="image-registry.openshift-image-regi
stry.svc:5000/openshift/php@sha256:d97466f33999951739a76bce922ab17088885db610
c0e05b593844b41d5494ea"
STEP 3: ENV OPENSHIFT_BUILD_NAME="file-uploader-1"
OPENSHIFT_BUILD_NAMESP
ACE="my-shared-storage"     OPENSHIFT_BUILD_SOURCE="https://github.com/christ
ianh814/openshift-php-upload-demo"     OPENSHIFT_BUILD_COMMIT="288eda3dff43b0
2f7f7b6b6b6f93396ffdf34cb2"
STEP 4: USER root
STEP 5: COPY upload/src /tmp/src
STEP 6: RUN chown -R 1001:0 /tmp/src
STEP 7: USER 1001
STEP 8: RUN /usr/libexec/s2i/assemble
---> Installing application source...
=> sourcing 20-copy-config.sh ...
```

```
---> 17:24:39     Processing additional arbitrary httpd configuration provide
d by s2i ...
=> sourcing 00-documentroot.conf ...
=> sourcing 50-mpm-tuning.conf ...
=> sourcing 40-ssl-certs.sh ...
STEP 9: CMD /usr/libexec/s2i/run
STEP 10: COMMIT temp.builder.openshift.io/my-shared-storage/file-uploader-1:3
b83e447
Getting image source signatures

[...]
```

Push が成功したことを確認すると、コマンドプロンプトは tail モードから復帰します。

注記

new-app コマンドは、アプリケーションを git リポジトリーから直接デプロイして、OpenShift テンプレートを使用しないため、OpenShift ルートリソースはデフォルトでは作成されません。ルートを手動で作成する必要があります。

アプリケーションを 4 つのレプリカにスケールアップし、サービスを公開することで、アプリケーションのゾーンを認識し、利用できるようにしましょう。

```
oc expose svc/file-uploader -n my-shared-storage
oc scale --replicas=4 deploy/file-uploader -n my-shared-storage
oc get pods -o wide -n my-shared-storage
```

数分後には 4 つの file-uploader Pod ができあがります。4 つの file-uploader Pod の STATUS が稼働中になるまで、上記のコマンドを繰り返します。

PersistentVolumeClaim を作成し、oc set volume コマンドを使用して、これをアプリケーションに割り当てます。以下を実行します。

```
oc set volume deploy/file-uploader --add --name=my-shared-storage \
-t pvc --claim-mode=ReadWriteMany --claim-size=10Gi \
--claim-name=my-shared-storage --claim-class=ocs-storagecluster-cephfs \
--mount-path=/opt/app-root/src/uploaded \
-n my-shared-storage
```

このコマンドにより、以下が行われます。

- PersistentVolumeClaim の作成
- ボリューム定義を含めるようにアプリケーションデプロイメントを更新します。
- アプリケーションのデプロイメントを更新して、ボリュームマウントを指定されたマウントパスに割り当てます。
- 4 つの application Pod の新規デプロイメント

これで、ボリュームの追加の結果を見てみましょう。

```
oc get pvc -n my-shared-storage
```

出力例:

```
NAME                STATUS   VOLUME                                     CAPACITY   ACCESS MODES
STORAGECLASS                 AGE
my-shared-storage   Bound    pvc-5402cc8a-e874-4d7e-af76-1eb05bd2e7c7   10Gi       RWX
ocs-storagecluster-cephfs    52s
```

ACCESSMODE が RWX (ReadWriteMany) に設定されている点に注意してください。

4 つの file-uploaderPods はすべて、同じ RWX ボリュームを使用しています。この ACCESSMODE がないと、OpenShift は複数の Pod を同じ Persistent Volume に確実にアタッチしようとしません。RWO (ReadWriteOnce) 永続ボリュームを使用しているデプロイメントをスケールアップしようとすると、Pod は同じノードに配置されます。

5.2. ゾーンを意識したデプロイメントの変更

現在、**file-uploader** Deployment はゾーンを認識せず、すべての Pod を同じゾーンでスケジュールできます。この場合、サイトに障害が発生すると、アプリケーションが利用できなくなります。詳細は、Pod トポロジー分散制約の使用 を参照してください。

1. アプリケーションゾーンを認識させるには、アプリケーションのデプロイメント設定に Pod 配置ルールを追加する必要があります。次のコマンドを実行して、以下のように出力を確認します。次の手順では、以下の出力の下にある Start セクションおよび End セクションに示されるように、トポロジーゾーンラベルを使用するように Deployment を変更します。

   ```
   $ oc get deployment file-uploader -o yaml -n my-shared-storage | less
   ```

   出力例:

   ```
   [...]
   spec:
     progressDeadlineSeconds: 600
     replicas: 4
     revisionHistoryLimit: 10
     selector:
       matchLabels:
         deployment: file-uploader
     strategy:
       rollingUpdate:
         maxSurge: 25%
         maxUnavailable: 25%
       type: RollingUpdate
     template:
       metadata:
         annotations:
           openshift.io/generated-by: OpenShiftNewApp
         creationTimestamp: null
         labels:
           deployment: file-uploader
         spec:  # <-- Start inserted lines after here
           containers:  # <-- End inserted lines before here
           - image: image-registry.openshift-image-registry.svc:5000/my-shared-storage/file-
   uploader@sha256:a458ea62f990e431ad7d5f84c89e2fa27bdebdd5e29c5418c70c56eb81f0a26
   b
   ```

```
          imagePullPolicy: IfNotPresent
          name: file-uploader
[...]
```

2. デプロイメントを編集し、上記のように開始と終了の間に以下の行を追加します。

```
$ oc edit deployment file-uploader -n my-shared-storage
```

```
[...]
      spec:
        topologySpreadConstraints:
          - labelSelector:
              matchLabels:
                deployment: file-uploader
            maxSkew: 1
            topologyKey: topology.kubernetes.io/zone
            whenUnsatisfiable: DoNotSchedule
          - labelSelector:
              matchLabels:
                deployment: file-uploader
            maxSkew: 1
            topologyKey: kubernetes.io/hostname
            whenUnsatisfiable: ScheduleAnyway
        nodeSelector:
          node-role.kubernetes.io/worker: ""
        containers:
[...]
```

出力例:

```
deployment.apps/file-uploader edited
```

3. デプロイを**0** Pod に変更し、その後 **4** Pod に戻します。これは、デプロイメントが Pod の配置に関して変更されたために必要です。

```
oc scale deployment file-uploader --replicas=0 -n my-shared-storage
```

出力例:

```
deployment.apps/file-uploader scaled
```

次に 4 つの Pod に戻ります。

```
$ oc scale deployment file-uploader --replicas=4 -n my-shared-storage
```

出力例:

```
deployment.apps/file-uploader scaled
```

4. 4 つの Pod が datacenter1 および datacenter2 ゾーンの 4 つのノードに分散されていることを確認します。

```
$ oc get pods -o wide -n my-shared-storage | egrep '^file-uploader'| grep -v build | awk '{print $7}' | sort | uniq -c
```

出力例:

```
  1 perf1-mz8bt-worker-d2hdm
  1 perf1-mz8bt-worker-k68rv
  1 perf1-mz8bt-worker-ntkp8
  1 perf1-mz8bt-worker-qpwsr
```

```
$ oc get nodes -L topology.kubernetes.io/zone | grep datacenter | grep -v master
```

出力例:

```
perf1-mz8bt-worker-d2hdm   Ready    worker   35d   v1.20.0+5fbfd19   datacenter1
perf1-mz8bt-worker-k68rv   Ready    worker   35d   v1.20.0+5fbfd19   datacenter1
perf1-mz8bt-worker-ntkp8   Ready    worker   35d   v1.20.0+5fbfd19   datacenter2
perf1-mz8bt-worker-qpwsr   Ready    worker   35d   v1.20.0+5fbfd19   datacenter2
```

5. ブラウザーを使用して file-uploader Web アプリケーションを使用して新規ファイルをアップロードします。

   a. 作成されたルートを検索します。

   ```
   $ oc get route file-uploader -n my-shared-storage -o jsonpath --template="http://{.spec.host}{'\n'}"
   ```

   これにより、これと同様のルートが返されます。

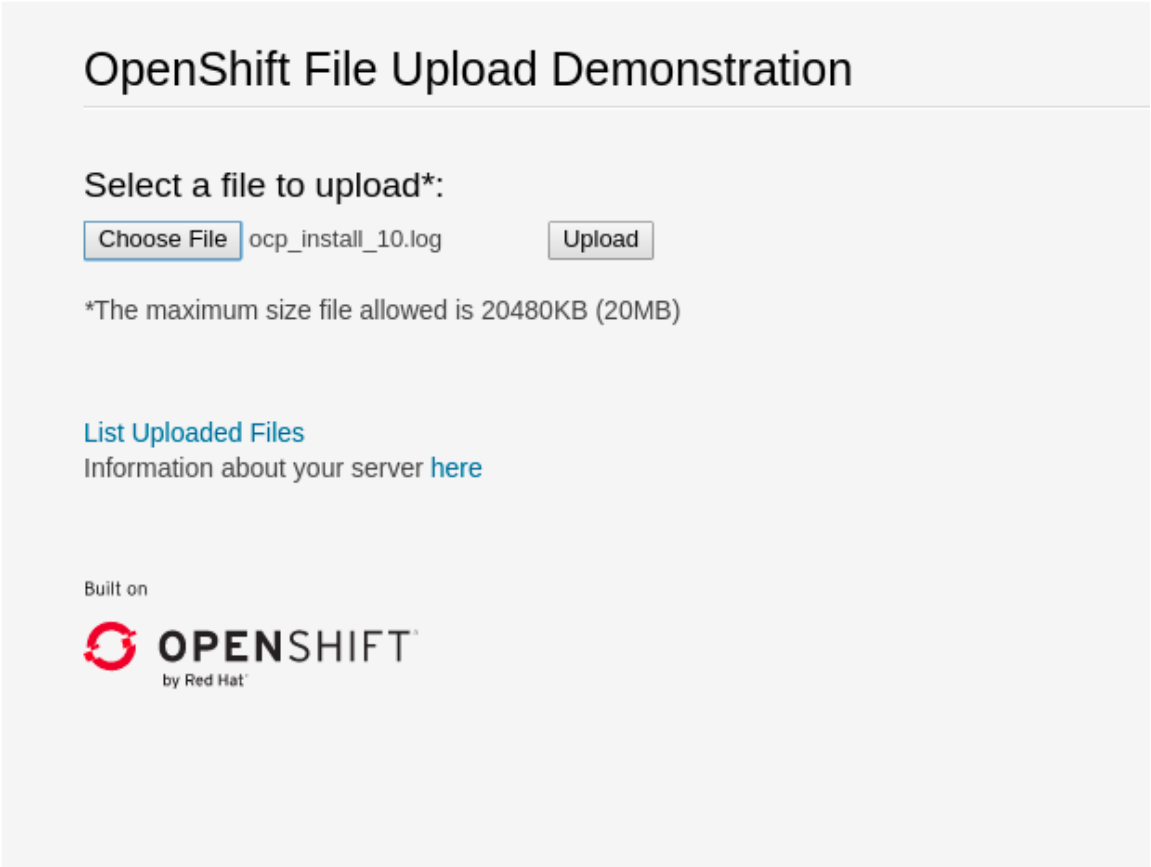
   出力サンプル

   ```
   http://file-uploader-my-shared-storage.apps.cluster-ocs4-abdf.ocs4-abdf.sandbox744.opentlc.com
   ```

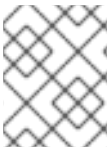
   b. 上記のルートを使用して、ブラウザーを Web アプリケーションに指定します。ルートは異なります。
   Web アプリケーションはアップロードされたすべてのファイルを一覧表示し、新しいファイルをアップロードしたり、既存のデータをダウンロードする機能を提供します。今は何もありません。

   c. ローカルマシンから任意のファイルを選択し、これをアプリケーションにアップロードします。

図5.1 簡単な PHP ベースのファイルのアップロードツール



d. **List uploaded files** をクリックし、現在アップロードされているファイルの一覧を表示します。



注記

OpenShift Container Platform イメージレジストリー、Ingress ルーティング、およびモニターリングサービスはゾーンを認識しません。