Red Hat OpenShift Data Foundation 4.10

Metro-DR 向け Advanced Cluster Management と OpenShift Data Foundation の設定

開発者プレビュー: Metro-DR 機能を備えた OpenShift Data Foundation の設定手順。このソリューションは開発者向けのプレビュー機能であり、実稼働環境で実行することを目的としたものではありません。

Last Updated: 2023-09-20

Red Hat OpenShift Data Foundation 4.10 Metro-DR 向け Advanced Cluster Management と OpenShift Data Foundation の設定

開発者プレビュー: Metro-DR 機能を備えた OpenShift Data Foundation の設定手順。このソリューションは開発者向けのプレビュー機能であり、実稼働環境で実行することを目的としたものではありません。

法律上の通知

概要

このソリューションガイドの目的は、災害復旧向けに OpenShift Data Foundation を Advanced Cluster Management と共にデプロイするのに必要な手順の詳細を提供し、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することです。 Configuring OpenShift Data Foundation for Metro-DR with Advanced Cluster Management is a Developer Preview feature and is subject to Developer Preview support limitations. Developer Preview releases are not intended to be run in production environments and are not supported through the Red Hat Customer Portal case management system. If you need assistance with Developer Preview features, reach out to the ocs-devpreview@redhat.com mailing list and a member of the Red Hat Development Team will assist you as quickly as possible based on their availability and work schedules.

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、Red Hat 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 METRO-DR の概要

障害復旧は、自然または人が原因の障害からビジネスクリティカルなアプリケーションを復旧し、継続する機能です。これは、重大な有害事象の発生時に事業の継続性を維持するために設計されている、主要な組織における事業継続ストラテジー全体の設定要素です。

Metro-DR 機能は、同じ地理的領域にあるサイト間でボリュームの永続的なデータとメタデータのレプリケーションを提供します。パブリッククラウドでは、これらはアベイラビリティーゾーンの障害からの保護に似ています。Metro-DR は、データセンターが利用できない場合でも、データを失うことなくビジネスの継続性を保証します。これは通常、目標復旧時点 (RPO) および目標復旧時間 (RTO) で表されます。

- RPO は、永続データのバックアップまたはスナップショットを作成する頻度の尺度です。実際には、RPO は、停止後に失われるか、再入力する必要があるデータの量を示します。Metro-DR ソリューションは、データが同期的に複製されるため、RPO がゼロであることを保証します。

- RTO は、企業が許容できるダウンタイムの量です。RTO は、ビジネスの中断が通知されてからシステムが回復するまでにどのくらいの時間がかかりますか？という質問に答えます。

このガイドの目的は、アプリケーションを Red Hat OpenShift Container Platform クラスターから別のクラスターにフェイルオーバーし、そのアプリケーションを元のプライマリークラスターにフォールバックできるようにするために必要な Metro Disaster Recovery (Metro-DR) の手順とコマンドを詳しく説明することです。この場合、RHOCP クラスターは Red Hat Advanced Cluster Management (RHACM) を使用して作成またはインポートされ、**RHOCP クラスター間の距離は RTT レイテンシーが 10 ミリ秒未満に制限されます**。

アプリケーションの永続ストレージは、2 つの場所の間に拡張された外部 **Red Hat Ceph Storage**クラスターによって提供され、RHOCP インスタンスはこのストレージクラスターに接続されます。サイトが停止した場合に Red Hat Ceph Storage クラスターのクォーラムを確立するには、3 番目の場所 (RHOCP インスタンスが展開されている場所とは異なる場所) にストレージモニターサービスを備えたアービターノードが必要になります。3 番目の場所では、RHOCP インスタンスに接続されたストレージクラスターから最大 100 ミリ秒の RTT レイテンシーの値をサポートする、レイテンシー要件が緩和されています。

1.1. METRO-DR ソリューションのコンポーネント

Metro-DR は、Red Hat Advanced Cluster Management for Kubernetes、Red Hat Ceph Storage、および OpenShift Data Foundation コンポーネントで設定され、OpenShift Container Platform クラスター全体でアプリケーションとデータのモビリティを提供します。

Red Hat Advanced Cluster Management for Kubernetes

Red Hat Advanced Cluster Management (RHACM) は、複数のクラスターとアプリケーションのライフサイクルを管理する機能を提供します。したがって、マルチクラスター環境でのコントロールプレーンとして機能します。

RHACM は 2 つの部分に分かれています。

- RHACM Hub: マルチクラスターコントロールプレーンで実行されるコンポーネント

- マネージドクラスター: マネージドクラスターで実行されるコンポーネント

この製品の詳細については、RHACM のドキュメント および RHACM のアプリケーションの管理 を参照してください。

Red Hat Ceph Storage

Red Hat Ceph Storage は、非常にスケーラブルでオープンなソフトウェア定義のストレージプラットフォームであり、最も安定したバージョンの Ceph ストレージシステムと Ceph 管理プラットフォーム、デプロイメントユーティリティー、およびサポートサービスを組み合わせたものです。エンタープライズデータの保存コストを大幅に削減し、組織が指数関数的なデータの成長を管理できるようにします。このソフトウェアは、パブリックまたはプライベートのクラウドデプロイメント向けの堅牢かつ最新のペタバイトスケールのストレージプラットフォームです。

OpenShift Data Foundation

OpenShift Data Foundation は、OpenShift Container Platform クラスターでステートフルなアプリケーション用のストレージをプロビジョニングし、管理する機能を提供します。これは、OpenShift Data Foundation コンポーネントスタックで Rook によって管理されるストレージプロバイダーとして Ceph によってサポートされ、Ceph-CSI はステートフルアプリケーションの永続ボリュームのプロビジョニングおよび管理を行います。

OpenShift Data Foundation スタックは、永続ボリューム要求ミラーリングごとに管理される **csi-addons** を提供する機能と共に強化されています。

OpenShift DR

OpenShift DR は、RHACM を使用してデプロイおよび管理される一連のピア OpenShift クラスター全体のステートフルアプリケーションの障害復旧オーケストレーターであり、永続ボリュームでのアプリケーションの状態のライフサイクルのオーケストレーションを行うためのクラウドネイティブインターフェイスを提供します。これらには以下が含まれます。

- OpenShift クラスター間でアプリケーションの状態の関係を保護する
- アプリケーションの状態をピアクラスターに移フェイルオーバーする
- アプリケーションの状態を以前にデプロイされたクラスターに再配置する

OpenShift DR は 2 つのコンポーネントに分割されます。

- **OpenShift DR Hub Operator**: アプリケーションのフェイルオーバーと再配置を管理するためにハブクラスターにインストールされます。
- **OpenShift DR Cluster Operator**: 各マネージドクラスターにインストールされ、アプリケーションのすべての PVC のライフサイクルを管理します。

1.2. METRO-DR デプロイメントワークフロー

このセクションでは、OpenShift Data Foundation バージョン 4.10、RHCS 5、および RHACM の最新バージョンを使用して Metro-DR 機能を 2 つの異なる OpenShift Container Platform クラスターに設定およびデプロイするために必要な手順の概要を説明します。2 つのマネージドクラスターに加えて、Advanced Cluster Management をデプロイするのに、3 つ目の OpenShift Container Platform クラスターが必要です。

インフラストラクチャーを設定するには、指定した順序で以下の手順を実行します。

1. RHACM オペレーターのインストール、OpenShift Container Platform の RHACM ハブおよびネットワーク設定への作成またはインポートを含む Metro-DR の各要件を満たしていることを確認してください。Requirements for enabling Metro-DR を参照します。

2. アービターを使用して Red Hat Ceph Storage ストレッチクラスターをデプロイするための要件を満たしていることを確認してください。Requirements for deploying Red Hat Ceph Storage を参照してください。

3. Red Hat Ceph Storage ストレッチクラスターモードを設定します。ストレッチモード機能を使用して 2 つの異なるデータセンターに Ceph クラスターを設定する方法については、Configuring Red Hat Ceph Storage stretch cluster を参照してください。

4. OpenShift Data Foundation 4.10 をプライマリーおよびセカンダリーマネージドクラスターにインストールします。マネージドクラスターへの OpenShift Data Foundation のインストール を参照してください。

5. Openshift DR Hub Operator をハブクラスターにインストールします。Installing OpenShift DR Hub Operator on Hub cluster を参照してください。

6. マネージドクラスターとハブクラスターを設定します。マネージドクラスターとハブクラスターの設定 を参照してください。

7. マネージドクラスター全体でワークロードをデプロイ、フェイルオーバー、および再配置するために使用されるハブクラスターに DRPolicy リソースを作成します。Creating Disaster Recovery Policy on Hub cluster を参照してください。

8. OpenShift DR Cluster オペレーターの自動インストールと、マネージドクラスターでの S3 シークレットの自動転送を有効にします。手順は、OpenShift DR クラスターオペレーターの自動インストールの有効化 および マネージドクラスターでの S3 シークレットの自動転送の有効化 を参照してください。

9. フェイルオーバーと再配置のテストをテストするために、RHACM コンソールを使用してサンプルアプリケーションを作成します。手順については、サンプルアプリケーションの作成、アプリケーションフェイルオーバー、およびマネージドクラスター間での アプリケーションの再配置 を参照してください。

第2章 METRO-DR を有効にする要件

Red Hat OpenShift Data Foundation でサポートされる障害復旧機能では、障害復旧ソリューションを正常に実装するために以下の前提条件がすべて必要になります。

- サブスクリプションの要件
  - 有効な Red Hat OpenShift Data Foundation Advanced エンタイトルメント
  - 有効な Red Hat Advanced Cluster Management for Kubernetes サブスクリプション

  OpenShift Data Foundation のサブスクリプションがどのように機能するかを知るには、OpenShift Data Foundation subscriptions に関するナレッジベースの記事 を参照してください。

- 相互にネットワーク接続が可能な 3 つの OpenShift クラスターが必要です。
  - **ハブクラスター**: Kubernetes のための高度なクラスター管理 (RHACM 演算子) と OpenShift DR ハブコントローラーがインストールされています。
  - **プライマリーマネージドクラスター**: OpenShift Data Foundation、OpenShift-DR クラスターコントローラー、およびアプリケーションがインストールされています。
  - **セカンダリーマネージドクラスター**: は、OpenShift Data Foundation、OpenShift-DR クラスターコントローラー、およびアプリケーションがインストールされています。

- RHACM オペレーターと Multiclusterhub がハブクラスターにインストールされていることを確認します。手順は、RHACM インストールガイド を参照してください。
  - デプロイメントが完了したら、OpenShift 認証情報を使用して RHACM コンソールにログインします。
  - Advanced Cluster Manager コンソール向けに作成されたルートを検索します。

    ```
    $ oc get route multicloud-console -n open-cluster-management -o jsonpath --
    template="https://{.spec.host}/multicloud/clusters{'\n'}"
    ```

    出力例:

    ```
    https://multicloud-console.apps.perf3.example.com/multicloud/clusters
    ```

    OpenShift 認証情報を使用してログインした後に、ローカルクラスターがインポートされたことが確認できるはずです。

- RHACM コンソールを使用して、**プライマリーマネージドクラスター**および**セカンダリーマネージドクラスター**をインポートまたは作成してください。環境に適したオプションを選択します。マネージドクラスターが正常に作成またはインポートされると、コンソールでインポートまたは作成されたクラスターの一覧を確認できます。

第3章 アービターを使用して RED HAT CEPH STORAGE ストレッチクラスターをデプロイするための要件

Red Hat Ceph Storage は、標準的で経済的なサーバーおよびディスク上で統一されたソフトウェア定義ストレージを提供するオープンソースエンタープライズプラットフォームです。ブロック、オブジェクト、ファイルストレージを 1 つのプラットフォームに統合することで、Red Hat Ceph Storage はすべてのデータを効率的かつ自動的に管理します。そのため、それを使用するアプリケーションおよびワークロードに集中することができます。

このセクションでは、Red Hat Ceph Storage デプロイメントの基本的な概要を説明します。より複雑なデプロイメントの詳細は、RHCS 5 の公式ドキュメントガイド を参照してください。

注記

劣化すると **min_size=1** で実行されるため、Flash メディアのみがサポートされています。ストレッチモードは、オールフラッシュ OSD でのみ使用してください。オールフラッシュ OSD を使用すると、接続が復元された後の回復に必要な時間が最小限に抑えられるため、データ損失の可能性が最小限に抑えられます。

重要

イレイジャーコーディングされたプールは、ストレッチモードでは使用できません。

3.1. ハードウェア要件

Red Hat Ceph Storage をデプロイするための最小ハードウェア要件については、Minimum hardware recommendations for containerized Ceph を参照してください。

表3.1 Red Hat Ceph Storage クラスターデプロイメントの物理サーバーの場所と Ceph コンポーネントのレイアウト:

ノード名	データセンター	Ceph コンポーネント
ceph1	DC1	OSD+MON+MGR
ceph2	DC1	OSD+MON
ceph3	DC1	OSD+MDS+RGW
ceph4	DC2	OSD+MON+MGR
ceph5	DC2	OSD+MON
ceph6	DC2	OSD+MDS+RGW
ceph7	DC3	MON

3.2. ソフトウェア要件

Red Hat Ceph Storage 5の最新のソフトウェアバージョンを使用してください。

Red Hat Ceph Storage でサポートされているオペレーティングシステムのバージョンの詳細については、Red Hat Ceph Storage: Supported configurations に関するナレッジベースの記事を参照してください。

3.3. ネットワーク設定要件

推奨される Red Hat Ceph Storage 設定は次のとおりです。

- パブリックネットワークとプライベートネットワークを 1 つずつ、合計 2 つのネットワークが必要です。
- すべてのデータセンターの Cephs プライベートネットワークとパブリックネットワークの VLAN とサブネットをサポートする 3 つの異なるデータセンターが必要です。

注記

データセンターごとに異なるサブネットを使用できます。

- Red Hat Ceph Storage Object Storage Devices (OSD) を実行している 2 つのデータセンター間のレイテンシーは、RTT で 10 ミリ秒を超えることはできません。**アービター** データセンターの場合、これは、他の 2 つの OSD データセンターに対して 100 ミリ秒の RTT という高い値でテストされました。

このガイドで使用した基本的なネットワーク設定の例を次に示します。

- **DC1: Ceph パブリック/プライベートネットワーク:**10.0.40.0/24
- **DC2: Ceph パブリック/プライベートネットワーク:**10.0.40.0/24
- **DC3: Ceph パブリック/プライベートネットワーク:**10.0.40.0/24

必要なネットワーク環境の詳細については、Ceph network configuration を参照してください。

3.4. ノードのプリデプロイメント要件

Red Hat Ceph Storage クラスターをインストールする前に、必要なすべての要件を満たすために、以下の手順を実施します。

1. 全ノードを Red Hat Network または Red Hat Satellite に登録し、有効なプールにサブスクライブします。

```
subscription-manager register
subscription-manager subscribe --pool=8a8XXXXXX9e0
```

2. 次のリポジトリーの Ceph クラスター内のすべてのノードへのアクセスを有効にします。
   - **rhel-8-for-x86_64-baseos-rpms**
   - **rhel-8-for-x86_64-appstream-rpms**

```
subscription-manager repos --disable="*" --enable="rhel-8-for-x86_64-baseos-rpms" --
enable="rhel-8-for-x86_64-appstream-rpms"
```

3. オペレーティングシステムの RPM を最新バージョンに更新し、必要に応じて再起動します。

```
dnf update -y
reboot
```

4. クラスターからノードを選択して、ブートストラップノードにします。**ceph1** は、この例の今後のブートストラップノードです。
ブートストラップノード **ceph1** でのみ、**ansible-2.9-for-rhel-8-x86_64-rpms** および **rhceph-5-tools-for-rhel-8-x86_64-rpms** リポジトリーを有効にします。

```
subscription-manager repos --enable="ansible-2.9-for-rhel-8-x86_64-rpms" --
enable="rhceph-5-tools-for-rhel-8-x86_64-rpms"
```

5. すべてのホストでベア/短縮ホスト名を使用して **hostname** を設定します。

```
hostnamectl set-hostname <short_name>
```

6. Red Hat Ceph Storage を使用して Red Hat Ceph Storage をデプロイするためのホスト名設定を確認します。

```
$ hostname
```

出力例:

```
ceph1
```

7. /etc/hosts ファイルを変更し、DNS ドメイン名を使用して DOMAIN 変数を設定して、fqdn エントリーを 127.0.0.1IP に追加します。

```
DOMAIN="example.domain.com"

cat <<EOF >/etc/hosts
127.0.0.1 $(hostname).${DOMAIN} $(hostname) localhost localhost.localdomain localhost4
localhost4.localdomain4
::1       $(hostname).${DOMAIN} $(hostname) localhost6 localhost6.localdomain6
EOF
```

8. **hostname -f** オプションを使用して、**fqdn** の長いホスト名を確認します。

```
$ hostname -f
```

出力例:

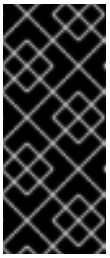
```
ceph1.example.domain.com
```

注: これらの変更が必要な理由の詳細については、完全修飾ドメイン名とベアホスト名 を参照してください。

9. ブートストラップノードで次の手順を実行します。この例では、ブートストラップノードは **ceph1** です。

   a. **cephadm-ansible** RPM パッケージをインストールします。

```
$ sudo dnf install -y cephadm-ansible
```



重要

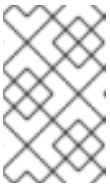
Ansible Playbook を実行するには、Red Hat Ceph Storage クラスターに設定されているすべてのノードに **ssh** パスワードなしでアクセスできる必要があります。設定されたユーザー (たとえば、**deployment-user**) が、パスワードを必要とせずに **sudo** コマンドを呼び出すための root 権限を持っていることを確認してください。

b. カスタムキーを使用するには、選択したユーザー (たとえば、**deployment-user**) の ssh 設定ファイルを設定して、ssh 経由でノードに接続するために使用される ID/キーを指定します。

```
cat <<EOF > ~/.ssh/config
Host ceph*
   User deployment-user
   IdentityFile ~/.ssh/ceph.pem
EOF
```

c. Ansible インベントリーを構築します。

```
cat <<EOF > /usr/share/cephadm-ansible/inventory
ceph1
ceph2
ceph3
ceph4
ceph5
ceph6
ceph7
[admin]
ceph1
EOF
```



注記

インベントリーファイルの admin グループの一部として設定されたホストは、**cephadm** によって **_admin** としてタグ付けされるため、ブートストラッププロセス中に admin ceph キーリングを受け取ります。

d. プリフライト Playbook を実行する前に、**ansible** が ping モジュールを使用してすべてのノードにアクセスできることを確認します。

```
$ ansible -i /usr/share/cephadm-ansible/inventory -m ping all -b
```

出力例:

```
ceph6 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
ceph4 | SUCCESS => {
    "ansible_facts": {
```

```
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
ceph3 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
ceph2 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
ceph5 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
ceph1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
ceph7 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
```

e. 以下の Ansible Playbook を実行します。

```
$ ansible-playbook -i /usr/share/cephadm-ansible/inventory /usr/share/cephadm-
ansible/cephadm-preflight.yml --extra-vars "ceph_origin=rhcs"
```

プリフライト Playbook は Red Hat Ceph Storage **dnf** リポジトリーを設定し、ブートストラップ用にストレージクラスターを準備します。また、podman、lvm2、chronyd、および cephadm もインストールします。**cephadm-ansible** および **cephadm-** preflight.yml のデフォルトの場所は **/usr/share/cephadm-ansible** です。

3.5. CEPHADM を使用したクラスターのブートストラップとサービスのデプロイメント

cephadm ユーティリティーは、cephadm ブートストラップコマンドが実行されているローカルノード上に、新しい Red Hat Ceph Storage クラスターの単一の Ceph Monitor デーモンと Ceph Manager デーモンをインストールし、開始します。

注記

ブートストラッププロセスの詳細については、Bootstrapping a new storage cluster を参照してください。

手順

1. 次のように、json ファイルを使用してコンテナーレジストリーに対して認証を行うための json ファイルを作成します。

```
$ cat <<EOF > /root/registry.json
{
 "url":"registry.redhat.io",
 "username":"User",
 "password":"Pass"
}
EOF
```

2. RHCS クラスターにノードを追加し、表 3.1 に従ってサービスを実行する場所に特定のラベルを設定する **cluster-spec.yaml** を作成します。

```
cat <<EOF > /root/cluster-spec.yaml
service_type: host
addr: 10.0.40.78  ## <XXX.XXX.XXX.XXX>
hostname: ceph1   ##  <ceph-hostname-1>
location:
  root: default
  datacenter: DC1
labels:
  - osd
  - mon
  - mgr
---
service_type: host
addr: 10.0.40.35
hostname: ceph2
location:
  datacenter: DC1
labels:
  - osd
  - mon
---
service_type: host
addr: 10.0.40.24
hostname: ceph3
location:
  datacenter: DC1
labels:
  - osd
  - mds
  - rgw
```

```
---
service_type: host
addr: 10.0.40.185
hostname: ceph4
location:
  root: default
  datacenter: DC2
labels:
  - osd
  - mon
  - mgr
---
service_type: host
addr: 10.0.40.88
hostname: ceph5
location:
  datacenter: DC2
labels:
  - osd
  - mon
---
service_type: host
addr: 10.0.40.66
hostname: ceph6
location:
  datacenter: DC2
labels:
  - osd
  - mds
  - rgw
---
service_type: host
addr: 10.0.40.221
hostname: ceph7
labels:
  - mon
---
service_type: mon
placement:
  label: "mon"
---
service_type: mds
service_id: fs_name
placement:
  label: "mds"
---
service_type: mgr
service_name: mgr
placement:
  label: "mgr"
---
service_type: osd
service_id: all-available-devices
service_name: osd.all-available-devices
placement:
  label: "osd"
```

```
spec:
  data_devices:
    all: true
---
service_type: rgw
service_id: objectgw
service_name: rgw.objectgw
placement:
  count: 2
  label: "rgw"
spec:
  rgw_frontend_port: 8080
EOF
```

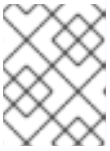
3. ブートストラップノードから RHCS パブリックネットワークが設定されている NIC の IP を取得します。**10.0.40.0** を ceph パブリックネットワークで定義したサブネットに置き換えた後、次のコマンドを実行します。

```
$ ip a | grep 10.0.40
```

出力例:

```
10.0.40.78
```

4. クラスター内の最初の Monitor ノードとなるノードで、**root** ユーザーとして **Cephadm** bootstrap コマンドを実行します。**IP_ADDRESS** オプションは、**cephadm bootstrap** コマンドの実行に使用しているノードの IP アドレスです。



注記

パスワードなしの SSH アクセス用に **root** ではなく別のユーザーを設定した場合は、**cepadm bootstrap** コマンドで **--ssh-user=** フラグを使用します。

```
$ cephadm  bootstrap --ssh-user=deployment-user --mon-ip 10.0.40.78 --apply-spec
/root/cluster-spec.yaml --registry-json /root/registry.json
```

重要

ローカルノードが完全修飾ドメイン名 (FQDN) を使用する場合は、コマンドラインで **--allow-fqdn-hostname** オプションを **cephadm bootstrap** に追加します。

ブートストラップが終了すると、前の cephadm bootstrap コマンドから次の出力が表示されます。

```
You can access the Ceph CLI with:

 sudo /usr/sbin/cephadm shell --fsid dd77f050-9afe-11ec-a56c-029f8148ea14 -c
/etc/ceph/ceph.conf -k /etc/ceph/ceph.client.admin.keyring

Please consider enabling telemetry to help improve Ceph:
```

```
 ceph telemetry on

For more information see:

 https://docs.ceph.com/docs/pacific/mgr/telemetry/
```

5. ceph1 の Ceph CLI クライアントを使用して、Red Hat Ceph Storage クラスターデプロイメントのステータスを確認します。

```
$ ceph -s
```

出力例:

```
cluster:
  id:     3a801754-e01f-11ec-b7ab-005056838602
  health: HEALTH_OK

services:
  mon: 5 daemons, quorum ceph1,ceph2,ceph4,ceph5,ceph7 (age 4m)
  mgr: ceph1.khuuot(active, since 5m), standbys: ceph4.zotfsp
  osd: 12 osds: 12 up (since 3m), 12 in (since 4m)
  rgw: 2 daemons active (2 hosts, 1 zones)

data:
  pools:   5 pools, 107 pgs
  objects: 191 objects, 5.3 KiB
  usage:   105 MiB used, 600 GiB / 600 GiB avail
           105 active+clean
```

注記

すべてのサービスが開始されるまでに数分かかる場合があります。

osds が設定されていないときに、グローバルリカバリーイベントが発生するのは正常です。

ceph orch ps および **ceph orch ls** を使用して、サービスのステータスをさらに確認できます。

6. すべてのノードが **cephadm** クラスターの一部であるかどうかを確認します。

```
$ ceph orch host ls
```

出力例:

```
HOST   ADDR         LABELS  STATUS
ceph1  10.0.40.78    _admin osd mon mgr
ceph2  10.0.40.35    osd mon
ceph3  10.0.40.24    osd mds rgw
ceph4  10.0.40.185   osd mon mgr
ceph5  10.0.40.88    osd mon
ceph6  10.0.40.66    osd mds rgw
ceph7  10.0.40.221   mon
```

注記

ceph1 は [admin] グループの一部として **cephadm-ansible** インベントリーで設定されているため、ホストから Ceph コマンドを直接実行できます。Ceph 管理キーは、**cephadm bootstrap** プロセス中にホストにコピーされました。

7. データセンターでの Ceph モニターサービスの現在の配置を確認します。

```
$ ceph orch ps | grep mon | awk '{print $1 " " $2}'
```

出力例:

```
mon.ceph1 ceph1
mon.ceph2 ceph2
mon.ceph4 ceph4
mon.ceph5 ceph5
mon.ceph7 ceph7
```

8. データセンターでの Ceph 管理サービスの現在の配置を確認します。

```
$ ceph orch ps | grep mgr | awk '{print $1 " " $2}'
```

出力例:

```
mgr.ceph2.ycgwyz ceph2
mgr.ceph5.kremtt ceph5
```

9. ceph osd クラッシュマップレイアウトをチェックして、各ホストに 1 つの OSD が設定され、そのステータスが **UP** であることを確認します。また、表 3.1 で指定されているように、各ノードが適切なデータセンターバケットの下にあることを再確認してください。

```
$ ceph osd tree
```

出力例:

```
ID   CLASS  WEIGHT   TYPE NAME          STATUS  REWEIGHT  PRI-AFF
-1         0.87900  root default
-16        0.43950      datacenter DC1
-11        0.14650          host ceph1
  2    ssd  0.14650              osd.2       up   1.00000  1.00000
 -3        0.14650          host ceph2
  3    ssd  0.14650              osd.3       up   1.00000  1.00000
-13        0.14650          host ceph3
  4    ssd  0.14650              osd.4       up   1.00000  1.00000
-17        0.43950      datacenter DC2
 -5        0.14650          host ceph4
  0    ssd  0.14650              osd.0       up   1.00000  1.00000
 -9        0.14650          host ceph5
  1    ssd  0.14650              osd.1       up   1.00000  1.00000
 -7        0.14650          host ceph6
  5    ssd  0.14650              osd.5       up   1.00000  1.00000
```

10. 新しい RDB ブロックプールを作成して有効にします。

```
$ ceph osd pool create rbdpool 32 32
$ ceph osd pool application enable rbdpool rbd
```

注記

コマンドの最後にある 32 という数字は、このプールに割り当てられている PG の数です。PG の数は、クラスター内の OSD の数、プールの予想使用率など、いくつかの要因によって異なります。次の計算機を使用して、必要な PG の数を決定できます。プール計算機ごとの Ceph 配置グループ (PG)。

11. RBD プールが作成されたことを確認します。

```
$ ceph osd lspools | grep rbdpool
```

出力例:

```
 3 rbdpool
```

12. MDS サービスがアクティブであり、各データセンターに 1 つのサービスが配置されていることを確認します。

```
$ ceph orch ps | grep mds
```

出力例:

```
mds.cephfs.ceph3.cjpbqo    ceph3          running (17m)   117s ago  17m    16.1M       -
16.2.9
mds.cephfs.ceph6.lqmgqt    ceph6          running (17m)   117s ago  17m    16.1M       -
16.2.9
```

13. CephFS ボリュームを作成します。

```
$ ceph fs volume create cephfs
```

注記

ceph fs volume create コマンドは、必要なデータとメタ CephFS プールも作成します。詳細については、Configuring and Mounting Ceph File Systems を参照してください。

14. **Ceph** のステータスを確認して、MDS デーモンがどのようにデプロイされたかを確認します。状態がアクティブで、**ceph6** がこのファイルシステムのプライマリー MDS で、**ceph3** がセカンダリー MDS であることを確認します。

```
$ ceph fs status
```

出力例:

```
cephfs - 0 clients
======
RANK  STATE        MDS          ACTIVITY     DNS    INOS   DIRS   CAPS
```

```
 0    active  cephfs.ceph6.ggjywj  Reqs:    0 /s    10     13     12      0
       POOL           TYPE     USED  AVAIL
cephfs.cephfs.meta  metadata  96.0k   284G
cephfs.cephfs.data    data       0    284G
   STANDBY MDS
cephfs.ceph3.ogcqkl
```

15. RGW サービスがアクティブであることを確認します。

```
$ ceph orch ps | grep rgw
```

出力例:

```
rgw.objectgw.ceph3.kkmxgb  ceph3  *:8080       running (7m)      3m ago   7m    52.7M        -
16.2.9
rgw.objectgw.ceph6.xmnpah  ceph6  *:8080       running (7m)      3m ago   7m    53.3M        -
16.2.9
```

第4章 RED HAT CEPH STORAGE ストレッチクラスターの設定

cephadm を使用して Red Hat Ceph Storage クラスターが完全にデプロイされたら、次の手順でストレッチクラスターモードを設定します。新しいストレッチモードは、2 サイトのケースを処理するように設計されています。

手順

1. ceph mon dump コマンドを使用して、モニターが使用している現在の選挙戦略を確認します。ceph クラスターのデフォルトでは、接続はクラシックに設定されています。

```
ceph mon dump | grep election_strategy
```

出力例:

```
dumped monmap epoch 9
election_strategy: 1
```

2. モニターの選択を接続に変更します。

```
ceph mon set election_strategy connectivity
```

3. 前の cephmondump コマンドを再度実行して、election_strategy 値を確認します。

```
$ ceph mon dump | grep election_strategy
```

出力例:

```
dumped monmap epoch 10
election_strategy: 3
```

さまざまな選択戦略の詳細については、Configuring monitor election strategy を参照してください。

4. すべての Ceph モニターの場所を設定します。

```
ceph mon set_location ceph1 datacenter=DC1
ceph mon set_location ceph2 datacenter=DC1
ceph mon set_location ceph4 datacenter=DC2
ceph mon set_location ceph5 datacenter=DC2
ceph mon set_location ceph7 datacenter=DC3
```

5. 各モニターに適切な場所があることを確認します。

```
$ ceph mon dump
```

出力例:

```
epoch 17
fsid dd77f050-9afe-11ec-a56c-029f8148ea14
last_changed 2022-03-04T07:17:26.913330+0000
created 2022-03-03T14:33:22.957190+0000
min_mon_release 16 (pacific)
```

```
election_strategy: 3
0: [v2:10.0.143.78:3300/0,v1:10.0.143.78:6789/0] mon.ceph1; crush_location
{datacenter=DC1}
1: [v2:10.0.155.185:3300/0,v1:10.0.155.185:6789/0] mon.ceph4; crush_location
{datacenter=DC2}
2: [v2:10.0.139.88:3300/0,v1:10.0.139.88:6789/0] mon.ceph5; crush_location
{datacenter=DC2}
3: [v2:10.0.150.221:3300/0,v1:10.0.150.221:6789/0] mon.ceph7; crush_location
{datacenter=DC3}
4: [v2:10.0.155.35:3300/0,v1:10.0.155.35:6789/0] mon.ceph2; crush_location
{datacenter=DC1}
```

6. **crushtool** コマンドを使用するために **ceph-base** RPM パッケージをインストールして、この OSD クラッシュトポロジーを利用する CRUSH ルールを作成します。

```
$ dnf -y install ceph-base
```

CRUSH ルールセットの詳細については、Ceph CRUSH ruleset を参照してください。

7. コンパイルされた CRUSH マップをクラスターから取得します。

```
$ ceph osd getcrushmap > /etc/ceph/crushmap.bin
```

8. CRUSH マップを逆コンパイルし、これをテキストファイルに変換して編集できるようにします。

```
$ crushtool -d /etc/ceph/crushmap.bin -o /etc/ceph/crushmap.txt
```

9. ファイルの末尾にあるテキストファイル **/etc/ceph/crushmap.txt** を編集して、以下のルールを CRUSH マップに追加します。

```
$ vim /etc/ceph/crushmap.txt
```

```
rule stretch_rule {
        id 1
        type replicated
        min_size 1
        max_size 10
        step take DC1
        step chooseleaf firstn 2 type host
        step emit
        step take DC2
        step chooseleaf firstn 2 type host
        step emit
}

# end crush map
```

注記

ルール **id** は一意である必要があります。この例では、id 0 のクラッシュルールがもう 1 つしかないため、id 1 を使用しています。デプロイメントにさらにルールが作成されている場合は、次の空き ID を使用します。

宣言された CRUSH ルールには、次の情報が含まれています。

- **Rule name**:
  - 説明: ルールを識別する一意の完全な名前。
  - 値: **stretch_rule**
- **id**:
  - 説明: ルールを識別する一意の整数。
  - 値: **1**
- **type**:
  - 説明: レプリケートまたはイレイジャーコーディングされたストレージドライブのルールを説明しています。
  - 値: **replicated**
- **min_size**:
  - 説明: プールがこの数よりも小さいレプリカを使用する場合、CRUSH はこのルールを選択しません。
  - 値: **1**
- **max_size**:
  - 説明: プールがこの数よりも大きいレプリカを使用する場合、CRUSH はこのルールを選択しません。
  - 値: **10**
- **step take DC1**
  - 説明: バケット名 (DC1) を取り、ツリーを下ってイテレートを開始します。
- **step chooseleaf firstn 2 type host**
  - 説明: 指定のタイプのバケット数を選択します。この例では、DC1 にある 2 つの異なるホストになります。
- **step emit**
  - 説明: 現在の値を出力し、スタックを除算します。通常、ルールの最後に使用されますが、同じルール内の異なるツリーを選択する際に使用することもできます。
- **step take DC2**
  - 説明: バケット名 (DC2) を取り、ツリーを下ってイテレートを開始します。
- **step chooseleaf firstn 2 type host**
  - 説明: 指定のタイプのバケット数を選択します。この例では、DC2 にある 2 つの異なるホストになります。
- **step emit**

- 説明: 現在の値を出力し、スタックを除算します。通常、ルールの最後に使用されますが、同じルール内の異なるツリーを選択する際に使用することもできます。

10. ファイル **/etc/ceph/crushmap.txt** から新しい CRUSH マップをコンパイルし、これを **/etc/ceph/crushmap2.bin** というバイナリーファイルに変換します。

```
$ crushtool -c /etc/ceph/crushmap.txt -o /etc/ceph/crushmap2.bin
```

11. 作成した新しいクラッシュマップをクラスターに注入します。

```
$ ceph osd setcrushmap -i /etc/ceph/crushmap2.bin
```

出力例:

```
17
```

注記

数字の 17 はカウンターであり、クラッシュマップに加えた変更に応じて増加します (18、19 など)。

12. 作成したストレッチルールが使用可能になったことを確認します。

```
ceph osd crush rule ls
```

出力例:

```
replicated_rule
stretch_rule
```

13. ストレッチクラスターモードを有効にします。

```
$ ceph mon enable_stretch_mode ceph7 stretch_rule datacenter
```

この例では、**ceph7** がアービターノード、**stretch_rule** が前の手順で作成したクラッシュルール、**datacenter** が分割バケットです。

14. すべてのプールが、Ceph クラスターに作成した **stretch_rule** CRUSH ルールを使用していることを確認します。

```
$ for pool in $(rados lspools);do echo -n "Pool: ${pool}; ";ceph osd pool get ${pool}
crush_rule;done
```

出力例:

```
Pool: device_health_metrics; crush_rule: stretch_rule
Pool: cephfs.cephfs.meta; crush_rule: stretch_rule
Pool: cephfs.cephfs.data; crush_rule: stretch_rule
Pool: .rgw.root; crush_rule: stretch_rule
Pool: default.rgw.log; crush_rule: stretch_rule
```

```
Pool: default.rgw.control; crush_rule: stretch_rule
Pool: default.rgw.meta; crush_rule: stretch_rule
Pool: rbdpool; crush_rule: stretch_rule
```

これは、アービターモードで稼働中の Red Hat Ceph Storage ストレッチクラスターが利用可能になったことを示しています。

第5章 マネージドクラスターへの OPENSHIFT DATA FOUNDATION のインストール

2 つの OpenShift Container Platform クラスター間でストレージレプリケーションを設定するには、以下のとおり先に OpenShift Data Foundation を各マネージドクラスターにインストールする必要があります。

1. 各マネージドクラスターに最新の OpenShift Data Foundation をインストールします。
2. Operator のインストール後に、**外部ストレージプラットフォームとの接続**オプションを使用して StorageSystem を作成します。
   詳しい手順は、外部モードでの OpenShift Data Foundation のデプロイ を参照してください。
3. OpenShift Data Foundation の正常なデプロイメントを検証します。
   a. 各マネージドクラスターで以下のコマンドを実行します。

```
$ oc get storagecluster -n openshift-storage ocs-external-storagecluster -o
jsonpath='{.status.phase}{"\n"}'
```

   b. Multicloud Gateway (MCG) の場合:

```
$ oc get noobaa -n openshift-storage noobaa -o jsonpath='{.status.phase}{"\n"}'
```

   ステータス結果が **Primary managed cluster** と **Secondary managed cluster** の両方のクエリーに対して **Ready** である場合は、次の手順に進みます。

注記

OpenShift Data Foundation のインストールの成功は、**Storage** と **Data Foundation** に移動して OpenShift Container Platform Web コンソールで検証することもできます。

第6章 ハブクラスターへの OPENSHIFT-DR HUB OPERATOR のインストール

手順

1. ハブクラスターで OperatorHub に移動し、**OpenShift-DR Hub Operator** の検索フィルターを使用します。
2. 画面の指示に従って、Operator をプロジェクト **openshift-dr-system** にインストールします。
3. 次のコマンドを使用して、オペレーター Pod が **Running** 状態にあることを確認します。

```
$ oc get pods -n openshift-dr-system
```

出力例:

```
NAME                                    READY   STATUS    RESTARTS   AGE
ramen-hub-operator-898c5989b-96k65    2/2     Running   0          4m14s
```

第7章 マネージドクラスターおよびハブクラスターの設定

7.1. S3 エンドポイント間の SSL アクセスの設定

s3 endpoints 間のネットワーク (SSL) アクセスを設定して、を安全なトランスポートプロトコルを使用してメタデータを **MCG object bucket** の代替クラスターに保存できるようにします。さらに、**ハブクラスター**はオブジェクトバケットへのアクセスを確認する必要があります。

注記

すべての OpenShift クラスターが環境の署名済みの有効な証明書セットを使用してデプロイされる場合は、このセクションを省略できます。

手順

1. プライマリーマネージドクラスター の Ingress 証明書を展開し、出力を **primary.crt** に保存します。

```
$ oc get cm default-ingress-cert -n openshift-config-managed -o jsonpath="{['data']['ca-bundle\.crt']}" > primary.crt
```

2. セカンダリーマネージドクラスターの Ingress 証明書を抽出し、出力を **secondary.crt** に保存します。

```
$ oc get cm default-ingress-cert -n openshift-config-managed -o jsonpath="{['data']['ca-bundle\.crt']}" > secondary.crt
```

3. **プライマリーマネージドクラスター**、**セカンダリーマネージドクラスター**、および **ハブクラスター** 上のファイル名 **cm-clusters-crt.yaml** を使用して、リモートクラスターの証明書バンドルを保持する新しい **ConfigMap** を作成します。

注記

この例のように、クラスターごとに 3 つ以上の証明書が存在する可能性があります。また、以前作成した **primary.crt** ファイルおよび **secondary.crt** ファイルから、証明書の内容をコピーして貼り付けた後に、証明書の内容が正しくインデントされていることを確認します。

```
apiVersion: v1
data:
  ca-bundle.crt: |
    -----BEGIN CERTIFICATE-----
    <copy contents of cert1 from primary.crt here>
    -----END CERTIFICATE-----

    -----BEGIN CERTIFICATE-----
    <copy contents of cert2 from primary.crt here>
    -----END CERTIFICATE-----

    -----BEGIN CERTIFICATE-----
    <copy contents of cert3 primary.crt here>
    -----END CERTIFICATE----
```

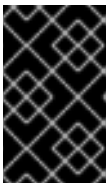
```
    -----BEGIN CERTIFICATE-----
    <copy contents of cert1 from secondary.crt here>
    -----END CERTIFICATE-----

    -----BEGIN CERTIFICATE-----
    <copy contents of cert2 from secondary.crt here>
    -----END CERTIFICATE-----

    -----BEGIN CERTIFICATE-----
    <copy contents of cert3 from secondary.crt here>
    -----END CERTIFICATE-----
kind: ConfigMap
metadata:
  name: user-ca-bundle
  namespace: openshift-config
```

4. **プライマリーマネージドクラスター**、**セカンダリーマネージドクラスター**、および**ハブクラスター**に ConfigMap ファイルを作成します。

```
$ oc create -f cm-clusters-crt.yaml
```

出力例:

```
configmap/user-ca-bundle created
```



重要

ハブクラスターが **DRPolicy** リソースを使用してオブジェクトバケットへのアクセスを確認するには、同じ **ConfigMap cm-clusters-crt.yaml** をハブクラスターに作成する必要もあります。

5. **プライマリーマネージドクラスター**、**セカンダリーマネージドクラスター**、および **ハブクラスター** 上のデフォルトのプロキシーリソースにパッチを適用します。

```
$ oc patch proxy cluster --type=merge  --patch='{"spec":{"trustedCA":{"name":"user-ca-bundle"}}}'
```

出力例:

```
proxy.config.openshift.io/cluster patched
```

7.2. オブジェクトバケットおよび S3STOREPROFILES の作成

OpenShift DR には、マネージドクラスターからのワークロードの関連クラスターデータを保存するため、およびフェイルオーバーまたは再配置アクション中のワークロード復旧のオーケストレーションを行うために、S3 ストアが必要です。これらの手順は、Multicloud Object Gateway (MCG) を使用して必要なオブジェクトバケットを作成するために適用できます。OpenShift Data Foundation のインストールにより、MCG がすでにインストールされているはずです。

手順

1. プライマリーおよびセカンダリーマネージドクラスターの両方で永続ボリュームメタデータを保存するために使用する、MCG オブジェクトバケットまたは OBC を作成します。

   a. 以下の YAML ファイルを、ファイル名 **odrbucket.yaml** にコピーします。

   ```
   apiVersion: objectbucket.io/v1alpha1
   kind: ObjectBucketClaim
   metadata:
     name: odrbucket
     namespace: openshift-storage
   spec:
     generateBucketName: "odrbucket"
     storageClassName: openshift-storage.noobaa.io
   ```

   b. **プライマリーマネージドクラスター** および **セカンダリーマネージドクラスター** の両方で、MCG バケット **odrbucket** を作成します。

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

   出力例:

   ```
   objectbucketclaim.objectbucket.io/odrbucket created
   ```

2. 以下のコマンドを使用して、各マネージドクラスターの **odrbucket OBC** アクセスキーを **base-64 でエンコード** された値として展開します。

   ```
   $ oc get secret odrbucket -n openshift-storage -o jsonpath='{.data.AWS_ACCESS_KEY_ID}
   {"\n"}'
   ```

   出力例:

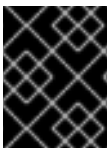
   ```
   cFpIYTZWN1NhemJjbEUyWlpwN1E=
   ```

3. 以下のコマンドを使用して、各マネージドクラスターの **odrbucket OBC** シークレットキーを **base-64 でエンコード** された値として展開します。

   ```
   $ oc get secret odrbucket -n openshift-storage -o
   jsonpath='{.data.AWS_SECRET_ACCESS_KEY}{"\n"}'
   ```

   出力例:

   ```
   V1hUSnMzZUoxMHRRTXdGMU9jQXRmUlAyMmd5bGwwYjNvMHprZVhtNw==
   ```



重要

アクセスキーとシークレットキーは、**プライマリーマネージドクラスター**と**セカンダリーマネージドクラスター**の両方の **odrbucket OBC** で取得する必要があります。

7.3. MULTICLOUD OBJECT GATEWAY オブジェクトバケットの S3 シークレットの作成

前のセクションでオブジェクトバケットに必要な情報が抽出されたので、**ハブクラスター**上に新し

いシークレットを作成する必要があります。これらの新しいシークレットは、ハブクラスター上の両方のマネージドクラスターの MCG オブジェクトバケットのアクセスキーとシークレットキーを格納します。

手順

1. **プライマリーマネージドクラスター**用の以下の S3 シークレット YAML 形式を、ファイル名 **odr-s3secret-primary.yaml** にコピーします。

```
apiVersion: v1
data:
  AWS_ACCESS_KEY_ID: <primary cluster base-64 encoded access key>
  AWS_SECRET_ACCESS_KEY: <primary cluster base-64 encoded secret access key>
kind: Secret
metadata:
  name: odr-s3secret-primary
  namespace: openshift-dr-system
```

2. このシークレットを**ハブクラスター**上に作成します。

```
$ oc create -f odr-s3secret-primary.yaml
```

出力例:

```
secret/odr-s3secret-primary created
```

3. **セカンダリーマネージドクラスター**用の以下の S3 シークレット YAML 形式を、ファイル名 **odr-s3secret-secondary.yaml** にコピーします。

```
apiVersion: v1
data:
  AWS_ACCESS_KEY_ID: <secondary cluster base-64 encoded access key>
  AWS_SECRET_ACCESS_KEY: <secondary cluster base-64 encoded secret access key>
kind: Secret
metadata:
  name: odr-s3secret-secondary
  namespace: openshift-dr-system
```

4. このシークレットを**ハブクラスター**上に作成します。

```
$ oc create -f odr-s3secret-secondary.yaml
```

出力例:

```
secret/odr-s3secret-secondary created
```

重要

アクセスキーおよびシークレットキーの値は **base-64 でエンコーディングされている** 必要があります。キーのエンコードされた値は、前のセクションで取得されています。

7.4. OPENSHIFT DR HUB OPERATOR の S3STOREPROFILES の設定

MCG の s3CompatibleEndpoint または route を見つけるには、プライマリーマネージドクラスターとセカンダリーマネージドクラスターで以下のコマンドを実行します。

手順

a. 以下のコマンドを使用して、外部 S3 エンドポイント **s3CompatibleEndpoint** または各マネージドクラスターで MCG のルートを検索します。

```
$ oc get route s3 -n openshift-storage -o jsonpath --template="https://{.spec.host}{'\n'}"
```

出力例:

```
https://s3-openshift-storage.apps.perf1.example.com
```

重要

一意の **s3CompatibleEndpoint** ルートまたは **s3-openshift-storage.apps.<primary clusterID>.<baseDomain>** および **s3-openshift-storage.apps.<secondary clusterID>.<baseDomain>** は、**プライマリーマネージドクラスター** と **セカンダリーマネージドクラスター** の両方で取得する必要があります。

b. **odrbucket OBC** バケットの正確な名前を検索します。

```
$ oc get configmap odrbucket -n openshift-storage -o jsonpath='{.data.BUCKET_NAME}
{"\n"}'
```

出力例:

```
odrbucket-2f2d44e4-59cb-4577-b303-7219be809dcd
```

重要

一意の **s3Bucket** 名 **odrbucket-<your value1>** と **odrbucket-<your value2>** は、**プライマリーマネージドクラスター** と **セカンダリーマネージドクラスター** の両方で取得する必要があります。

c. ハブクラスターの ConfigMap **ramen-hub-operator-config** を変更して、新しいコンテンツを追加します。

```
$ oc edit configmap ramen-hub-operator-config -n openshift-dr-system
```

d. **s3StoreProfiles** で開始する以下の新規コンテンツを、ハブクラスターの **ConfigMap** のみに追加します。

```
[...]
data:
  ramen_manager_config.yaml: |
    apiVersion: ramendr.openshift.io/v1alpha1
    kind: RamenConfig
[...]
    ramenControllerType: "dr-hub"
    ### Start of new content to be added
```

```
    s3StoreProfiles:
    - s3ProfileName: s3-primary
      s3CompatibleEndpoint: https://s3-openshift-storage.apps.<primary clusterID>.
<baseDomain>
      s3Region: primary
      s3Bucket: odrbucket-<your value1>
      s3SecretRef:
        name: odr-s3secret-primary
        namespace: openshift-dr-system
    - s3ProfileName: s3-secondary
      s3CompatibleEndpoint: https://s3-openshift-storage.apps.<secondary clusterID>.
<baseDomain>
      s3Region: secondary
      s3Bucket: odrbucket-<your value2>
      s3SecretRef:
        name: odr-s3secret-secondary
        namespace: openshift-dr-system
[...]
```

第8章 ハブクラスターでの障害復旧ポリシーの作成

OpenShift DR は、RHACM ハブクラスターで Disaster Recovery Policy (DRPolicy) リソース (クラスタースコープ) を使用して、マネージドクラスター間でワークロードをデプロイ、フェイルオーバー、および再配置します。

前提条件

- 2 つのクラスターのセットがあることを確認します。
- ポリシーの各クラスターに、OpenShift-DR Cluster および Hub Operator の ConfigMap で設定される S3 プロファイル名が割り当てられている必要があります。

手順

1. **ハブクラスター**で、**openshift-dr-system** プロジェクトで Installed Operators に移動し、**OpenShift DR Hub Operator** をクリックします。2 つの利用可能な API (**DRPolicy** と **DRPlacementControl**) が表示されるはずです。
2. DRPolicy の **Create instance** をクリックし、**YAML view** をクリックします。
3. <cluster1> および <cluster2> を **RHACM** のマネージドクラスターの正しい名前に置き換えてから、以下の YAML を、ファイル名 **drpolicy.yaml** に保存します。<string_value> を任意の値 (つまり metro) に置き換えます。

```
apiVersion: ramendr.openshift.io/v1alpha1
kind: DRPolicy
metadata:
  name: odr-policy
spec:
  drClusterSet:
  - name: <cluster1>
    region: <string_value>
    s3ProfileName: s3-primary
    clusterFence: Unfenced
  - name: <cluster2>
    region: <string_value>
    s3ProfileName: s3-secondary
    clusterFence: Unfenced
```

注記

DRPolicy はクラスタースコープのリソースであるため、このリソースを作成するために namespace を指定する必要はありません。

4. 一意の **drpolicy.yaml** ファイルの内容を YAML ビューにコピーします。元のコンテンツを完全に置き換える必要があります。
5. YAML ビュー画面の **Create** をクリックします。
6. **DRPolicy** が正常に作成され、前に作成したシークレットを使用して MCG オブジェクトバケットにアクセスできることを検証するには、**ハブクラスター**で次のコマンドを実行します。

```
$ oc get drpolicy odr-policy -n openshift-dr-system -o jsonpath='{.status.conditions[].reason}
{"\n"}'
```

出力例:

```
Succeeded
```

第9章 OPENSHIFT DR クラスターオペレーターの自動インストールを有効にする

DRPolicy が正常に作成されると、**OpenShift DR クラスターオペレーター** を **openshift-dr-system** 名前空間のプライマリーマネージドクラスターおよびセカンダリーマネージドクラスターにインストールできます。

手順

1. ハブクラスターで ConfigMag **ramen-hub-operator-config** を編集し、次のように **deploymentAutomationEnabled=false** の値を **true** に変更します。

```
$ oc edit configmap ramen-hub-operator-config -n openshift-dr-system
```

```
apiVersion: v1
data:
  ramen_manager_config.yaml: |
  [...]
    drClusterOperator:
      deploymentAutomationEnabled: true  ## <-- Change value to "true" if it is set to "false"
      channelName: stable-4.10
      packageName: odr-cluster-operator
      namespaceName: openshift-dr-system
      catalogSourceName: redhat-operators
      catalogSourceNamespaceName: openshift-marketplace
      clusterServiceVersionName: odr-cluster-operator.v4.10.0
[...]
```

2. **プライマリーマネージドクラスター**でインストールが成功したことを確認し、**セカンダリーマネージドクラスター**で次のコマンドを実行します。

```
$ oc get csv,pod -n openshift-dr-system
```

出力例:

```
NAME                                                              DISPLAY                       VERSION
REPLACES   PHASE
clusterserviceversion.operators.coreos.com/odr-cluster-operator.v4.10.0   Openshift DR
Cluster Operator   4.10.0              Succeeded

NAME                                         READY   STATUS    RESTARTS   AGE
pod/ramen-dr-cluster-operator-5564f9d669-f6lbc   2/2     Running   0          5m32s
```

各マネージドクラスターの Operator Hub に移動して、**OpenShift DR Cluster Operator** がインストールされているかどうかを確認することもできます。

第10章 マネージドクラスターへの S3SECRETS の自動転送の有効化

この手順に従って、必要な OpenShift DR クラスターコンポーネントへの s3Secrets の自動転送を有効にします。OpenShift DR 設定マップ内の s3Profiles にアクセスするために必要な s3Secrets を使用して、OpenShift DR クラスターの名前空間を更新します。

手順

1. 次のように、ハブクラスターの ConfigMag **ramen-hub-operator-config** 設定を編集して、**s3SecretDistributionEnabled=true** を追加します。

```
$ oc edit configmap ramen-hub-operator-config -n openshift-dr-system
```

```
apiVersion: v1
data:
  ramen_manager_config.yaml: |
    apiVersion: ramendr.openshift.io/v1alpha1
    drClusterOperator:
      deploymentAutomationEnabled: true
      s3SecretDistributionEnabled: true  ## <-- Add to enable automatic transfer of s3secrets
      catalogSourceName: redhat-operators
      catalogSourceNamespaceName: openshift-marketplace
      channelName: stable-4.10
      clusterServiceVersionName: odr-cluster-operator.v4.10.0
      namespaceName: openshift-dr-system
      packageName: odr-cluster-operator
[...]
```

2. 両方のマネージドクラスターでこのコマンドを実行して、シークレットの転送が成功したことを確認します。

```
$ oc get secrets -n openshift-dr-system | grep Opaque
```

出力例:

```
8b3fb9ed90f66808d988c7edfa76eba35647092   Opaque       2      11m
af5f82f21f8f77faf3de2553e223b535002e480   Opaque       2      11m
```

第11章 サンプルアプリケーションの作成

プライマリーマネージドクラスターからセカンダリーマネージドクラスターへの **failover** およびフェイルバックをテストするには、単純なアプリケーションが必要です。**busybox** というサンプルアプリケーションを例として使用します。

手順

1. **busybox** サンプルアプリケーションの **ハブクラスター** で **namespace** または **プロジェクト** を作成します。

```
$ oc new-project busybox-sample
```

注記

必要に応じて、**busybox-sample** 以外のプロジェクト名を使用できます。Advanced Cluster Manager コンソールでサンプルアプリケーションをデプロイする場合は、この手順で作成したものと同じプロジェクト名を使用するようにしてください。

2. **DRPlacementControl** リソースを作成します。
DRPlacementControl は、ハブクラスターに OpenShift DR Hub Operator をインストールした後に利用可能な API です。これは、概説としては、DRPolicy の一部であるクラスター間でのデータ可用性に基づいて配置の決定をオーケストレーションする Advanced Cluster Manager PlacementRule リコンサイラーです。

   a. ハブクラスターで、**busybox-sample** プロジェクトで Installed Operators に移動し、**OpenShift DR Hub Operator** をクリックします。2 つの利用可能な API (DRPolicy と DRPlacementControl) が表示されるはずです。

   b. **DRPlacementControl** のインスタンスを作成してから、YAML ビューに移動します。**busybox-sample** プロジェクトが選択されていることを確認します。

   c. **<cluster1>** を Advanced Cluster Manager のマネージドクラスターの正しい名前に置き換えたあと、以下の YAML をファイル名 **busybox-drpc.yaml** にコピーして保存します。

```
apiVersion: ramendr.openshift.io/v1alpha1
kind: DRPlacementControl
metadata:
  labels:
    app: busybox-sample
  name: busybox-drpc
spec:
  drPolicyRef:
    name: odr-policy
  placementRef:
    kind: PlacementRule
    name: busybox-placement
  preferredCluster: <cluster1>
  pvcSelector:
    matchLabels:
      appname: busybox
```

d. 一意の **busybox-drpc.yaml** ファイルの内容を YAML ビューにコピーします (元のコンテンツを完全に置き換え)。

e. YAML ビュー画面の **Create** をクリックします。
以下の CLI コマンドを使用してこのリソースを作成することもできます。

```
$ oc create -f busybox-drpc.yaml -n busybox-sample
```

出力例:

```
drplacementcontrol.ramendr.openshift.io/busybox-drpc created
```

重要

このリソースは、**busybox-sample** namespace (または先に作成した namespace) に作成する必要があります。

3. リソーステンプレートのデプロイ先のターゲットクラスターを定義する **Placement Rule** リソースを作成します。配置ルールを使用すると、アプリケーションのマルチクラスターデプロイメントが容易になります。

a. 以下の YAML をファイル名 **busybox-placementrule.yaml** にコピーし、保存します。

```
apiVersion: apps.open-cluster-management.io/v1
kind: PlacementRule
metadata:
  labels:
    app: busybox-sample
  name: busybox-placement
spec:
  clusterConditions:
  - status: "True"
    type: ManagedClusterConditionAvailable
  clusterReplicas: 1
  schedulerName: ramen
```

b. **busybox-sample** アプリケーションの PlacementRule リソースを作成します。

```
$ oc create -f busybox-placementrule.yaml -n busybox-sample
```

出力例:

```
placementrule.apps.open-cluster-management.io/busybox-placement created
```

重要

このリソースは、**busybox-sample** namespace (または先に作成した namespace) に作成する必要があります。

4. RHACM コンソールを使用した **サンプルアプリケーション** の作成

a. まだログインしていない場合は、OpenShift 認証情報を使用して RHACM コンソールにログインします。

```
$ oc get route multicloud-console -n open-cluster-management -o jsonpath --
template="https://{.spec.host}/multicloud/applications{"\n"}"
```

出力例:

```
https://multicloud-console.apps.perf3.example.com/multicloud/applications
```

b. **Applications** に移動し、**Create application** をクリックします。

c. 種類は **Subscription**を選択します。

d. アプリケーションの **Name** (**busybox** など) および **Namespace** (**busybox-sample** など) を入力します。

e. Repository location for resources セクションで **Repository type Git** を選択します。

f. サンプルアプリケーションの github **Branch** および **Path** で、 Git リポジトリー URL を入力します。リソース **busybox** Pod および PVC が作成されます。
Branch が **main** で、**Path** は **busybox-odr-metro** である **https://github.com/RamenDR/ocm-ramen-samples** として、サンプルアプリケーションリポジトリーを使用します。

g. **Select clusters to deploy to**セクションまでフォームを下にスクロールして、**Select an existing placement configuration** をクリックします。

h. ドロップダウンリストから **Existing Placement Rule** (**busybox-placement** など) を選択します。

i. **Save** をクリックします。
次に表示される画面で下部までスクロールします。アプリケーショントポロジーのチェックマークがすべて緑であることが確認できるはずです。

注記

詳細な情報を表示するには、トポロジー要素のいずれかをクリックすると、トポロジービューの右側にウィンドウが表示されます。

5. サンプルアプリケーションのデプロイメントおよびレプリケーションを確認します。
busybox アプリケーションが (DRPlacementControl で指定された) preferredCluster にデプロイされたので、デプロイメントを検証できるようになりました。

a. **busybox** が RHACM によってデプロイされたマネージドクラスターにログオンします。

```
$ oc get pods,pvc -n busybox-sample
```

出力例:

```
NAME          READY   STATUS    RESTARTS   AGE
pod/busybox   1/1     Running   0          6m

NAME                                 STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS                  AGE
persistentvolumeclaim/busybox-pvc    Bound    pvc-a56c138a-a1a9-4465-927f-
af02afbbff37   1Gi        RWO            ocs-storagecluster-ceph-rbd   6m
```

b. レプリケーションリソースも **busybox** PVC に作成されていることを確認します。

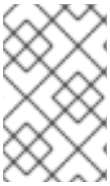
```
$ oc get volumereplicationgroup -n busybox-sample
```

出力例:

```
NAME                                                  AGE
volumereplicationgroup.ramendr.openshift.io/busybox-drpc   6m
```

11.1. サンプルアプリケーションの削除

RHACM コンソールを使用してサンプルアプリケーション **busybox** を削除できます。

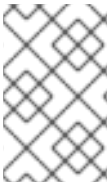


注記

サンプルアプリケーションを削除する手順は、フェイルオーバーとフォールバック (再配置) のテストが完了し、アプリケーションを RHACM とマネージドクラスターから削除する準備ができるまで実行しないでください。

手順

1. RHACM コンソールで、**Applications** に移動します。
2. 削除するサンプルアプリケーションを検索します (例: **busybox**)。
3. 削除するアプリケーションの横にある Action メニュー **(⋮)** をクリックします。
4. **Delete application** をクリックします。
   Delete application を選択すると、アプリケーション関連のリソースも削除すべきかどうかを求める新規画面が表示されます。
5. **Remove application related resources**チェックボックスを選択して、Subscription および PlacementRule を削除します。
6. **Delete** をクリックします。これにより、Primary マネージドクラスター (またはアプリケーションが実行しているクラスター) の busybox アプリケーションが削除されます。
7. RHACM コンソールを使用して削除されたリソースのほかに、**busybox** アプリケーションの削除直後に **DRPlacementControl** も削除する必要があります。
   a. ハブクラスターの OpenShift Web コンソールにログインし、プロジェクト **busybox-sample** の Installed Operators に移動します。
   b. **OpenShift DR Hub Operator** をクリックした後、**DRPlacementControl** タブをクリックします。
   c. 削除する **busybox** アプリケーション DRPlacementControl の横にあるアクションメニュー **(⋮)** をクリックします。
   d. **Delete DRPlacementControl** をクリックします。
   e. **Delete** をクリックします。



注記

このプロセスを使用して、**DRPlacementControl** リソースでアプリケーションを削除できます。**DRPlacementControl** リソースは、CLI を使用してアプリケーション namespace で削除することもできます。

第12章 マネージドクラスター間のアプリケーションのフェイルオーバー

本セクションでは、busybox サンプルアプリケーションをフェイルオーバーする方法を説明します。Metro-DR のフェイルオーバー方法は、アプリケーションベースです。この方法で保護される各アプリケーションには、Create Sample Application for DR testing セクションで説明されているように、対応する **DRPlacementControl** リソースとアプリケーション **namespace** で作成された **PlacementRule** リソースが必要です。

手順

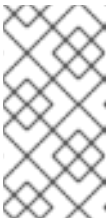
1. **NetworkFence** リソースを作成し、**フェンシング**を有効にします。
   ネットワークフェンシング操作が実行される CIDR ブロックまたは IP アドレスのリストを指定します。この場合は、外部 RHCS クラスターの使用からフェンスする必要があるクラスター内にあるすべての OpenShift ノードの EXTERNAL-IP になります。

   a. このコマンドを実行して、**プライマリーマネージドクラスター** の IP アドレスを取得します。

   ```
   $ oc get nodes -o jsonpath='{range .items[*]}{.status.addresses[?
   (@.type=="ExternalIP")].address}{"\n"}{end}'
   ```

   出力例:

   ```
   10.70.56.118
   10.70.56.193
   10.70.56.154
   10.70.56.242
   10.70.56.136
   10.70.56.99
   ```

   注記

   サイトが停止する前に、すべての OpenShift ノードの現在の IP アドレスを収集します。ベストプラクティスは、**NetworkFence** YAML ファイルを作成し、それを障害回復イベントに使用できるようにして最新の状態にすることです。

   以下に示すように、すべてのノードの IP アドレスが **NetworkFence** サンプルリソースに追加されます。この例は 6 つのノードを対象としていますが、クラスター内にさらに多くのノードが存在する可能性があります。

   ```
   apiVersion: csiaddons.openshift.io/v1alpha1
   kind: NetworkFence
   metadata:
     name: network-fence-<cluster1>
   spec:
     driver: openshift-storage.rbd.csi.ceph.com
     cidrs:
       - <IP_Address1>/32
       - <IP_Address2>/32
       - <IP_Address3>/32
       - <IP_Address4>/32
   ```

```
    - <IP_Address5>/32
    - <IP_Address6>/32
    [...]
  secret:
    name: rook-csi-rbd-provisioner
    namespace: openshift-storage
  parameters:
    clusterID: openshift-storage
```

b. 上記の YAML ファイルの例では、IP アドレスを変更し、プライマリーマネージドクラスターの **RHACM** で検出されるクラスター名となる正しい **<cluster1>** を指定します。これをファイル名 **network-fence-<cluster1>.yaml** に保存します。

重要

NetworkFence は、フェイルオーバーの前に、アプリケーションが現在実行されている反対側のマネージドクラスターから作成する必要があります。この場合は**セカンダリーマネージドクラスター**です。

```
$ oc create -f network-fence-<cluster1>.yaml
```

出力例:

```
networkfences.csiaddons.openshift.io/network-fence-ocp4perf1 created
```

重要

NetworkFence が作成されると、アプリケーションから OpenShift Data Foundation ストレージへのすべての通信が失敗し、一部の Pod は、現在フェンスでされているクラスター上で異常な状態 (CreateContainerError、CrashLoopBackOff など) になります。

c. **NetworkFence** が作成された場所と同じクラスターで、ステータスが Succeeded であることを確認します。<cluster1> を正しく変更します。

```
export NETWORKFENCE=network-fence-<cluster1>
oc get networkfences.csiaddons.openshift.io/$NETWORKFENCE -n openshift-dr-system
-o jsonpath='{.status.result}{"\n"}'
```

出力例:

```
Succeeded
```

2. **fenced** クラスターの **DRPolicy** を変更します。

a. ハブクラスターの **DRPolicy** を編集し、**<cluster1>** (例: ocp4perf1) を **Unfenced** から **ManuallyFenced** に変更します。

```
$ oc edit drpolicy odr-policy
```

出力例:

```
[...]
spec:
  drClusterSet:
  - clusterFence: ManuallyFenced  ## <-- Modify from Unfenced to ManuallyFenced
    name: ocp4perf1
    region: metro
    s3ProfileName: s3-primary
  - clusterFence: Unfenced
    name: ocp4perf2
    region: metro
    s3ProfileName: s3-secondary
[...]
```

出力例:

```
drpolicy.ramendr.openshift.io/odr-policy edited
```

b. ハブクラスターの **DRPolicy** ステータスが、**プライマリーマネージドクラスター**で **Fenced** に変更されていることを確認します。

```
$ oc get drpolicies.ramendr.openshift.io odr-policy -o yaml | grep -A 6 drClusters
```

出力例:

```
  drClusters:
    ocp4perf1:
      status: Fenced
      string: ocp4perf1
    ocp4perf2:
      status: Unfenced
      string: ocp4perf2
```

3. **failover** するように **DRPlacementControl** を変更します

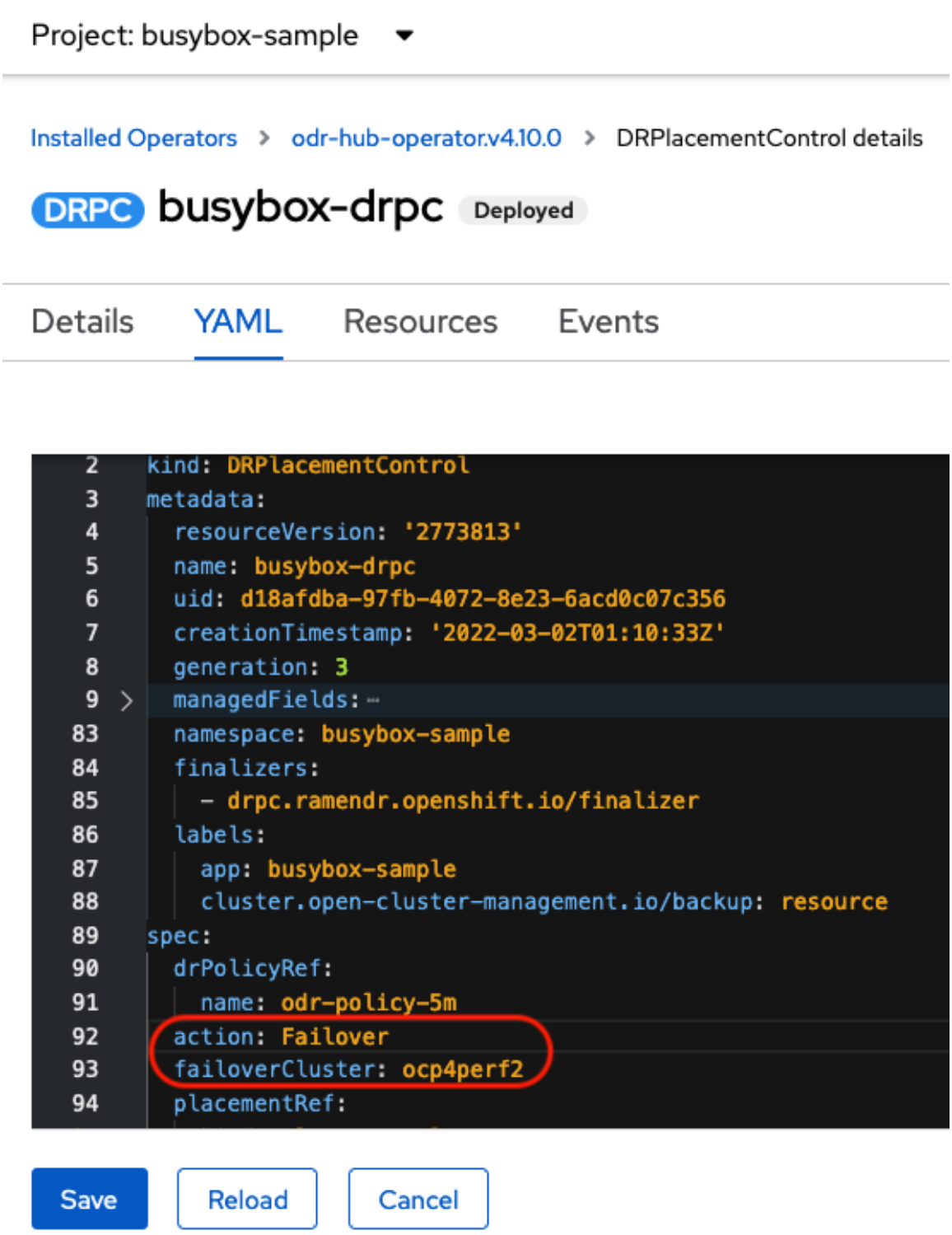
a. ハブクラスターで Installed Operators に移動し、**Openshift DR Hub Operator** をクリックします。

b. **DRPlacementControl** タブをクリックします。

c. DRPC **busybox-drpc** をクリックしてから、YAML ビューをクリックします。

d. 以下のスクリーンショットのように、**action** および **failoverCluster** の詳細を追加します。**failoverCluster** はセカンダリーマネージドクラスターの ACM クラスター名である必要があります。

DRPlacementControl add action Failover



e. **Save** をクリックします。

4. アプリケーションの **busybox** がセカンダリーマネージドクラスター (YAML ファイルに指定されるフェイルオーバークラスター **ocp4perf2**) で実行されていることを確認します。

```
$ oc get pods,pvc -n busybox-sample
```

出力例:

```
NAME          READY   STATUS    RESTARTS   AGE
pod/busybox   1/1     Running   0          35s

NAME                                STATUS   VOLUME                                     CAPACITY   ACCESS
```

```
MODES   STORAGECLASS                 AGE
persistentvolumeclaim/busybox-pvc   Bound    pvc-79f2a74d-6e2c-48fb-9ed9-666b74cfa1bb
5Gi         RWO             ocs-storagecluster-ceph-rbd   35s
```

5. **busybox** がプライマリーマネージドクラスターで実行していないことを確認します。

```
$ oc get pods,pvc -n busybox-sample
```

出力例:

```
No resources found in busybox-sample namespace.
```

重要

リリースノートの 既知の問題 に記載されている既知の Metro-DR の問題に注意してください。

第13章 マネージドクラスター間のアプリケーションの再配置

再配置操作はフェイルオーバーと非常に似ています。再配置はアプリケーションベースで、**DRPlacementControl** を使用して再配置をトリガーします。フォールバックの主な違いは、アプリケーションが failoverCluster でスケールダウンされるため、**NetworkFence** を作成する必要がないことです。

手順

1. **NetworkFence** リソースを削除し、**Fencing** を無効にします。
   フォールバックまたは再配置アクションを成功させる前に、プライマリーマネージドクラスターの **NetworkFence** を削除する必要があります。

   a. **セカンダリーマネージドクラスター**でこのコマンドを実行し、前のセクションで作成した **NetworkFenceYAML** ファイル名に合わせて **<cluster1>** を変更します。

   ```
   $ oc delete -f network-fence-<cluster1>.yaml
   ```

   出力例:

   ```
   networkfence.csiaddons.openshift.io "network-fence-ocp4perf1" deleted
   ```

   b. **Fenced** になっている OpenShift Container Platform ノードを再起動します。
   以前のフェンスされたクラスター (この場合はプライマリーマネージドクラスター) 上の一部のアプリケーション Pod が異常な状態にあるため (例: CreateContainerError、CrashLoopBackOff)、この手順が必要です。これは、すべてのワーカー OpenShift ノードを一度に 1 つずつ再起動することで最も簡単に修正できます。

   注記

   OpenShift Web Console ダッシュボードと **概要** ページを使用して、アプリケーションと外部ストレージの正常性を評価することもできます。OpenShiftDataFoundation ダッシュボードの詳細は、**Storage → Data Foundation** に移動すると表示されます。

   c. すべての OpenShift ノードが再起動されてステータスが **Ready** になった後、**プライマリーマネージドクラスター**でこのコマンドを実行して、すべての Pod が正常な状態にあることを確認します。このクエリーの出力はゼロ Pod である必要があります。

   ```
   $ oc get pods -A | egrep -v 'Running|Completed'
   ```

   出力例:

   ```
   NAMESPACE                                  NAME
   READY   STATUS     RESTARTS      AGE
   ```

   重要

   ストレージ通信が切断されたために Pod がまだ異常な状態にある場合は、続行する前にトラブルシューティングを行って解決してください。ストレージクラスターは OpenShift の外部にあるため、OpenShift アプリケーションを正常に動作させるには、サイトの停止後にストレージクラスターを適切に復元する必要もあります。

2. **DRPolicy** を **Unfenced** ステータスに変更します。
   ODR HUB 演算子が、プライマリーマネージドクラスターの **NetworkFence** が削除されたことを知るには、新しい **Unfenced** クラスターの **DRPolicy** を変更する必要があります。

   a. ハブクラスターの **DRPolicy** を編集し、**<cluster1>** (例: **ocp4perf1**) を **ManuallyFenced** から **Unfenced** に変更します。

   ```
   $ oc edit drpolicy odr-policy
   ```

   出力例:

   ```
   [...]
   spec:
     drClusterSet:
     - clusterFence: Unfenced  ## <-- Modify from ManuallyFenced to Unfenced
       name: ocp4perf1
       region: metro
       s3ProfileName: s3-primary
     - clusterFence: Unfenced
       name: ocp4perf2
       region: metro
       s3ProfileName: s3-secondary
   [...]
   ```

   出力例:

   ```
   drpolicy.ramendr.openshift.io/odr-policy edited
   ```

   b. **プライマリーマネージドクラスターのハブクラスター**で、**DRPolicy** のステータスが **Unfenced** に変更されていることを確認します。

   ```
   $ oc get drpolicies.ramendr.openshift.io odr-policy -o yaml | grep -A 6 drClusters
   ```

   出力例:

   ```
     drClusters:
       ocp4perf1:
         status: Unfenced
         string: ocp4perf1
       ocp4perf2:
         status: Unfenced
         string: ocp4perf2
   ```

3. **DRPlacementControl** を **failback** に変更します

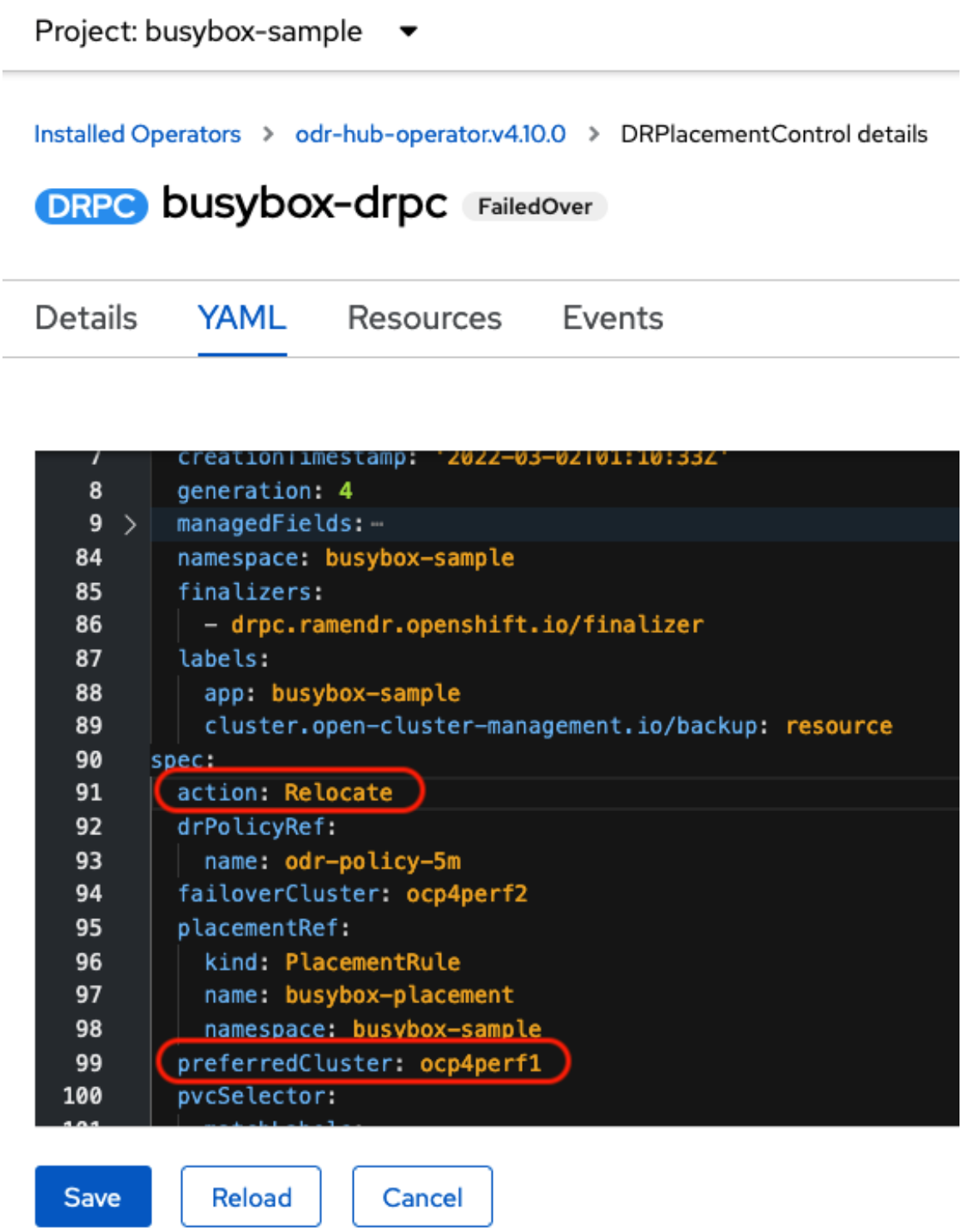
   a. ハブクラスターで Installed Operators に移動し、**Openshift DR Hub Operator** をクリックします。

   b. **DRPlacementControl** タブをクリックします。

   c. DRPC **busybox-drpc** をクリックしてから、YAML ビューをクリックします。

   d. **action** を **Relocate** に変更します

DRPlacementControl modify action to Relocate



e. **Save** をクリックします。

f. アプリケーション **busybox** がプライマリーマネージドクラスターで実行されているかどうかを確認します。フェイルオーバー操作の前にアプリケーションが実行していた YAML ファイルで指定されている **preferredCluster ocp4perf1** へのフェイルバックが行われます。

```
$ oc get pods,pvc -n busybox-sample
```

出力例:

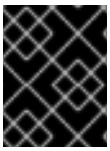
```
NAME         READY   STATUS    RESTARTS   AGE
pod/busybox   1/1     Running   0          60s

NAME                                  STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS                  AGE
persistentvolumeclaim/busybox-pvc   Bound    pvc-79f2a74d-6e2c-48fb-9ed9-
666b74cfa1bb   5Gi        RWO            ocs-storagecluster-ceph-rbd   61s
```

g. **busybox** がセカンダリーマネージドクラスターで実行しているかどうかを確認します。busybox アプリケーションは、このマネージドクラスターでは実行しないようにしてください。

```
$ oc get pods,pvc -n busybox-sample
```

出力例:

```
No resources found in busybox-sample namespace.
```



重要

リリースノートの 既知の問題 に記載されている既知の Metro-DR の問題に注意してください。