# Red Hat Advanced Cluster Management for Kubernetes 2.6

## アドオン

クラスターでアドオンを使用する方法

Last Updated: 2023-11-01

# Red Hat Advanced Cluster Management for Kubernetes 2.6 アドオン

クラスターでアドオンを使用する方法

## 法律上の通知



## 概要

クラスターでアドオンを使用する方法

# 目次

# 第1章 アドオンの概要

Red Hat Advanced Cluster Management for Kubernetes アドオンは、パフォーマンスの一部の領域を改善し、アプリケーションを強化する機能を追加できます。以下のセクションでは、Red Hat Advanced Cluster Management で使用できるアドオンの概要を説明します。

- Submariner マルチクラスターネットワーキングおよびサービスディスカバリー
- VolSync 永続ボリューム複製サービス
- Kubernetes Operator のマルチクラスターエンジンからクラスターで klusterlet アドオンを有効にする

## 1.1. SUBMARINER マルチクラスターネットワーキングおよびサービスディスカバリー

Submariner は、Red Hat Advanced Cluster Management for Kubernetes で使用できるオープンソースツールであり、オンプレミスまたはクラウドのいずれかの環境で、2 つ以上のマネージドクラスター間で直接ネットワークおよびサービスディスカバリーを提供します。Submariner は Multi-Cluster Services API (Kubernetes Enhancements Proposal #1645) と互換性があります。Submariner の詳細は、Submariner のサイト を参照してください。

Red Hat Advanced Cluster Management for Kubernetes は、Submariner をハブクラスターのアドオンとして提供します。Submariner の詳細は、Submariner オープンソースプロジェクトのドキュメント を参照してください。

自動化コンソールデプロイメント でサポートされるインフラストラクチャープロバイダーおよび 手動デプロイメント が必要なインフラストラクチャープロバイダーに関する詳細は、Red Hat Advanced Cluster Management のサポートマトリックス を参照してください。

- 前提条件
- subctl コマンドユーティリティー
- Globalnet
- Submariner のデプロイ
  - コンソールを使用した Submariner のデプロイ
  - サブマリーナを手動でデプロイ
    - Submariner 向けに選択されたホストの準備
    - ManagedClusterAddOn API を使用した Submariner のデプロイ
    - Submariner デプロイメントのカスタマイズ
- Submariner のサービス検出の管理
- Submariner のアンインストール

### 1.1.1. 前提条件

Submariner を使用する前に、以下の前提条件があることを確認します。

- **cluster-admin** のパーミッションを使用してハブクラスターにアクセスするための認証情報。
- ゲートウェイノード間で IP 接続を設定している。2 つのクラスターを接続する場合に、最低でも 1 つのクラスターには、ゲートウェイノード専用のパブリックまたはプライベート IP アドレスを使用してゲートウェイノードにアクセスできる必要があります。詳細は、Submariner NAT Traversal を参照してください。
- OVN Kubernetes を使用している場合には、クラスターは Red Hat OpenShift Container Platform バージョン 4.11 以降を使用する必要があります。
- 各マネージドクラスターのすべてのノードにおけるファイアウォール設定は、両方の方向で 4800/UDP が許可されている。
- ゲートウェイノードのファイアウォール設定では、入力 8080/TCP が許可されているため、クラスター内の他のノードがアクセスできます。
- UDP/4500、およびゲートウェイノード上の IPsec トラフィックに使用されるその他のポート用にファイアウォール設定が開いている。
- ゲートウェイノードが間に NAT を介さずにプライベート IP 経由で直接到達できる場合は、ファイアウォール設定でゲートウェイノード上で ESP プロトコルが許可されていることを確認してください。
  **注記:** これは、クラスターが AWS または GCP 環境にデプロイされる際に自動的に設定されますが、他の環境のクラスター用に手動で設定し、プライベートクラウドを保護するファイアウォール用に設定する必要があります。
- **managedcluster** 名は、RFC1123 で定義されている DNS ラベル標準に従っている。これは、名前が次の基準を満たしている必要があることを意味します。
  - 最大 63 文字を含む。
  - 小文字の英数字またはハイフン (-) のみが含まれる。
  - 英数字で始まる。
  - 英数字で終わる。

### 1.1.2. Submariner ポートテーブル

次の表を参照して、有効にする必要がある Submariner ポートを確認してください。

| Name | デフォルト値 | カスタマイズ可能 | 任意または必須 |
|---|---|---|---|
| IPsec NATT | 4500/UDP | はい | 必須 |
| VXLAN | 4800/UDP | いいえ | 必須 |
| Submariner メトリクスポート | 8080/TCP | いいえ | 必須 |
| Globalnet メトリクスポート | 8081/TCP | いいえ | Globalnet が有効な場合に必要 |

前提条件の詳細は、Submariner アップストリームの前提条件のドキュメント を参照してください。

### 1.1.3. subctl コマンドユーティリティー

Submariner には、Submariner 環境でのタスクの実行を簡素化する追加コマンドを提供する **subctl** ユーティリティーが含まれています。

#### 1.1.3.1. subctl コマンドユーティリティーのインストール

**subctl** ユーティリティーは、コンテナーイメージで提供されます。**subctl** ユーティリティーをローカルにインストールするには、次の手順を実行します。

1. 次のコマンドを実行し、プロンプトが表示されたら認証情報を入力して、レジストリーにログインします。

```
oc registry login --registry registry.redhat.io
```

2. 次のコマンドを入力して、**subctl** コンテナー をダウンロードし、**subctl** バイナリーの圧縮バージョンを **/tmp** に展開します。

```
oc image extract registry.redhat.io/rhacm2/subctl-rhel8:v0.13 --path="/dist/subctl-v0.13*-linux-amd64.tar.xz":/tmp/ --confirm
```

3. 次のコマンドを入力して、**subctl** ユーティリティーを展開します。

```
tar -C /tmp/ -xf /tmp/subctl-v0.13*-linux-amd64.tar.xz
```

4. 次のコマンドを入力して、**subctl** ユーティリティーをインストールします。

```
install -m744 /tmp/subctl-v0.13*/subctl-v0.13*-linux-amd64 /$HOME/.local/bin/subctl
```

#### 1.1.3.2. subctl コマンドの使用

パスにユーティリティーを追加した後に使用可能なコマンドの簡単な説明については、次の表を参照してください。

| | |
|---|---|
| **export service** | 指定されたサービスの **ServiceExport** リソースを作成します。これにより、Submariner デプロイメント内の他のクラスターが対応するサービスを検出できるようになります。 |
| **unexport service** | 指定されたサービスの **ServiceExport** リソースを削除します。これにより、Submariner デプロイメント内の他のクラスターが対応するサービスを検出できなくなります。 |
| **show** | Submariner リソースに関する情報を提供します。 |
| **verify** | Submariner がクラスターのペア全体で設定されている場合は、接続性、サービスディスカバリー、およびその他のサブマリーナー機能を検証します。 |

| | |
|---|---|
| **benchmark** | Submariner で、または単一のクラスター内で有効になっているクラスターのペア全体のスループットおよびレイテンシーをベンチマークします。 |
| **diagnose** | チェックを実行して、Submariner デプロイメントが正しく機能しない原因となる問題を特定します。 |
| **gather** | クラスターから情報を収集して、Submariner デプロイメントのトラブルシューティングに役立てます。 |
| **version** | **subctl** バイナリーツールのバージョンの詳細を表示します。 |

**subctl** ユーティリティーとそのコマンドの詳細は、Submariner ドキュメントの**subctl** を参照してください。

### 1.1.4. Globalnet

Globalnet は、CIDR が重複しているクラスター間の接続をサポートする Submariner アドオンに含まれている機能です。Globalnet はクラスターセット全体の設定であり、最初のマネージドクラスターがクラスターセットに追加されたときに選択できます。Globalnet が有効になっている場合、各マネージドクラスターには、仮想グローバルプライベートネットワークからグローバル CIDR が割り当てられます。グローバル CIDR は、クラスター間通信をサポートするのに使用されます。

Submariner を実行しているクラスターで CIDR が重複している可能性がある場合は、Globalnet を有効にすることを検討してください。コンソールを使用する場合、**ClusterAdmin** は、クラスターセット内のクラスターに対して Submariner アドオンを有効にするときに、**Globalnet を有効にする** オプションを選択することにより、クラスターセットに対して Globalnet を有効にすることができます。Globalnet を有効にした後は、Submariner を削除せずに無効にすることはできません。

Red Hat Advanced Cluster Management API を使用する場合、**ClusterAdmin** は、**<ManagedClusterSet>-broker** namespace に **submariner-broker** オブジェクトを作成することで Globalnet を有効にできます。

**ClusterAdmin** ロールには、ブローカーの namespace にこのオブジェクトを作成するのに必要な権限があります。クラスターセットのプロキシー管理者として機能するのに作成されることがある **ManagedClusterSetAdmin** ロールには、必要な権限がありません。必要な権限を提供する場合は、**ClusterAdmin** が **access-to-brokers-submariner-crd** のロール権限を **ManagedClusterSetAdmin** ユーザーに関連付ける必要があります。

**submariner-broker** オブジェクトを作成するには、次の手順を実行します。

1. 次のコマンドを実行して **<broker-namespace>** を取得します。

```
oc get ManagedClusterSet <cluster-set-name> -o jsonpath="
{.metadata.annotations['cluster\.open-cluster-management\.io/submariner-broker-ns']}"
```

2. **submariner-broker** という名前の YAML ファイルを作成して、Globalnet 設定を指定する **submariner-broker** オブジェクトを作成します。次の行のようなコンテンツを YAML ファイルに追加します。

```
apiVersion: submariner.io/v1alpha1
kind: Broker
metadata:
  name: submariner-broker
  namespace: <broker-namespace>
spec:
  globalnetEnabled: <true-or-false>
```

**broker-namespace** を、ブローカーの namespace に置き換えます。

Globalnet を有効にするには、**true-or-false** を **true** に置き換えます。

**注:メタデータ名** パラメーターは **submariner-broker** である必要があります。

3. 次のコマンドを入力して、ファイルを YAML ファイルに適用します。

```
oc apply -f submariner-broker.yaml
```

Globalnet の詳細は、Submariner のドキュメントの Globalnet コントローラー を参照してください。

### 1.1.5. Submariner のデプロイ

Submariner は、次のプロバイダーのネットワーククラスターにデプロイできます。

自動デプロイメントプロセス:

- Amazon Web Services
- Google Cloud Platform
- Red Hat OpenStack Platform
- Microsoft Azure

手動デプロイメントプロセス:

- IBM Cloud
- VMware vSphere
- ベアメタル

#### 1.1.5.1. コンソールを使用した Submariner のデプロイ

Red Hat Advanced Cluster Management for Kubernetes コンソールを使用して、Amazon Web Services、Google Cloud Platform、および VMware vSphere にデプロイされた Red Hat OpenShift Container Platform マネージドクラスターに Submariner をデプロイできます。他のプロバイダーに Submariner をデプロイするには、Submariner の手動デプロイ を参照してください。Red Hat Advanced Cluster Management for Kubernetes コンソールで Submariner をデプロイするには、以下の手順を実行します。

**必要なアクセス権限:** クラスターの管理者

1. コンソールナビゲーションメニューから **Infrastructure** > **Clusters** を選択します。

2. **Clusters** ページで、**Cluster sets** タブを選択します。Submariner で有効にするクラスターは、同じクラスターセットにある必要があります。

3. Submariner をデプロイするクラスターがすでに同じクラスターセットにある場合は、手順 5 を省略して Submariner をデプロイします。

4. Submariner をデプロイするクラスターが同じクラスターセットにない場合は、以下の手順に従ってクラスターセットを作成します。

   a. **Create cluster set**を選択します。

   b. クラスターセットに名前を付け、**Create** を選択します。

   c. **Manage resource assignments** を選択して、クラスターセットに割り当てます。

   d. Submariner で接続するマネージドクラスターを選択して、クラスターセットに追加します。

   e. **Review** を選択して、選択したクラスターを表示し、確認します。

   f. **Save** を選択してクラスターセットを保存し、作成されるクラスターセットページを表示します。

5. クラスターセットページで、**Submariner add-on** タブを選択します。

6. **Install Submariner add-ons** を選択します。

7. Submariner をデプロイするクラスターを選択します。

8. **Install Submariner add-on** エディターに以下の情報を入力します。

   - **AWS Access Key ID** - このフィールドは、AWS クラスターをインポートする場合にのみ表示されます。

   - **AWS Secret Access Key**: このフィールドは、AWS クラスターをインポートする場合にのみ表示されます。

   - **Google Cloud Platform service account JSON key**: このフィールドは、Google Cloud Platform クラスターをインポートする場合にのみ表示されます。

   - **Instance type** - マネージドクラスターで作成されたゲートウェイノードの Amazon Web Services EC2 インスタンスタイプ。デフォルト値は **c5d.large** です。このフィールドは、マネージドクラスター環境が AWS の場合のみ表示されます。

   - **IPsec NAT-T ポート**: IPsec NAT トラバーサルポートのデフォルト値はポート **4500** です。マネージドクラスター環境が VMware vSphere の場合は、ファイアウォールでこのポートが開いていることを確認してください。

   - **ゲートウェイ数**: マネージドクラスターへの Submariner ゲートウェイコンポーネントのデプロイに使用されるワーカーノードの数。デフォルト値は **1** です。値が 1 を超える場合、Submariner ゲートウェイの High Availability (HA) は自動的に有効になります。

   - **ケーブルドライバー**: クラスター間トンネルを維持する Submariner ゲートウェイケーブルエンジンのコンポーネントです。デフォルト値は **Libreswan IPsec** です。

9. エディターの末尾で **Next** を選択して、次のクラスターのエディターに移動し、選択した残りのクラスターごとに、エディターを完了します。

10. 各マネージドクラスターの設定を確認します。

11. **Install** をクリックして、選択したマネージドクラスターに Submariner をデプロイします。インストールと設定が完了するまで数分かかる場合があります。**Submariner add-on** タブの一覧で Submariner ステータスを確認できます。

    - **Connection status** は、マネージドクラスターで確立される Submariner 接続の数を示します。
    - **Agent status** は、Submariner がマネージドクラスターに正常にデプロイされるかどうかを示します。コンソールでは、インストールと設定が完了するまで **Degraded** のステータスをレポートする場合があります。
    - **Gateway nodes labeled** は、マネージドクラスターの Submariner ゲートウェイラベル **submariner.io/gateway=true** が付いたワーカーノードの数を示します。

Submariner がクラスターにデプロイされました。

#### 1.1.5.2. サブマリーナを手動でデプロイ

Red Hat Advanced Cluster Management for Kubernetes に Submariner をデプロイする前に、接続用にホスト環境でクラスターを準備する必要があります。現時点で、**SubmarinerConfig** API を使用して、Amazon Web Services、Google Cloud Platform、および VMware vSphere のクラスターを自動的に準備できます。他のプラットフォームの場合は、手動で準備する必要があります。手順は、Submariner をデプロイする一部のホストの準備 を参照してください。

##### 1.1.5.2.1. Submariner をデプロイする一部のホストの準備

Red Hat Advanced Cluster Management for Kubernetes に Submariner をデプロイする前に、接続用にホスト環境でクラスターを手動で準備する必要があります。要件はホスティング環境によって異なるため、ホスティング環境の手順に従います。

###### 1.1.5.2.1.1. Submariner 向けの VMware vSphere の準備

Submariner は IPsec を使用して、ゲートウェイノード上のクラスター間でセキュアなトンネルを確立します。デフォルトのポートを使用するか、カスタムポートを指定できます。IPsec NATT ポートを指定せずにこの手順を実行すると、通信にはデフォルトのポートが自動的に使用されます。デフォルトのポートは 4500/UDP です。

Submariner は、仮想拡張可能な LAN (VXLAN) を使用して、ワーカーノードおよびマスターノードからゲートウェイノードに移行するときにトラフィックをカプセル化します。VXLAN ポートはカスタマイズできず、常にポート 4800/UDP を使用します。

Submariner は 8080/TCP を使用してクラスターのノード間でメトリクス情報を送信します。このポートはカスタマイズできません。

Submariner を有効にするには、VMWare vSphere 管理者が以下のポートを開放する必要があります。

**表1.1 VMware vSphere および Submariner ポート**

| Name | デフォルト値 | カスタマイズ可能 |
|---|---|---|
| IPsec NATT | 4500/UDP | はい |

| Name | デフォルト値 | カスタマイズ可能 |
|---|---|---|
| VXLAN | 4800/UDP | いいえ |
| Submariner メトリクス | 8080/TCP | いいえ |

Submariner をデプロイするように VMware vSphere を準備するには、以下の手順を実行します。

1. IPsec NATT、VXLAN、およびメトリクスポートが開放されていることを確認します。
2. 次の例のような YAML コンテンツをカスタマイズして適用します。

```
apiVersion: submarineraddon.open-cluster-management.io/v1alpha1
kind: SubmarinerConfig
metadata:
   name: submariner
   namespace: <managed-cluster-namespace>
spec:{}
```

**managed-cluster-namespace** は、マネージドクラスターの namespace に置き換えます。

**注記:** 以下の例のように、**SubmarinerConfig** の名前は **submariner** である必要があります。

この設定は、Submariner にデフォルトの NATT (Network Address Translation-Traversal) ポート (4500/UDP) を使用し、1 つのワーカーノードは vSphere クラスターの Submariner ゲートウェイとしてラベルが付けられています。

Submariner は IP セキュリティー (IPsec) を使用して、ゲートウェイノード上のクラスター間でセキュアなトンネルを確立します。デフォルトの IPsec NATT ポートを使用するか、設定した別のポートを指定できます。IPsec NATT を指定せずにこの手順を実行すると、4500/UDP のポートが通信に自動的に使用されます。

#### 1.1.5.2.1.2. Submariner 向けのベアメタルの準備

Submariner をデプロイするためのベアメタルクラスターを準備するには、次の手順を実行します。

1. IPsec NATT、VXLAN、およびメトリクスポートが開放されていることを確認します。
2. 次の例のような YAML コンテンツをカスタマイズして適用します。

```
apiVersion: submarineraddon.open-cluster-management.io/v1alpha1
kind: SubmarinerConfig
metadata:
   name: submariner
   namespace: <managed-cluster-namespace>
spec:{}
```

**managed-cluster-namespace** は、マネージドクラスターの namespace に置き換えます。

**注記:** 以下の例のように、**SubmarinerConfig** の名前は **submariner** である必要があります。

この設定は、Submariner にデフォルトの NATT (Network Address Translation-Traversal) ポート (4500/UDP) を使用し、1つのワーカーノードはベアメタルクラスターの Submariner ゲートウェイとしてラベルが付けられています。

Submariner は IP セキュリティー (IPsec) を使用して、ゲートウェイノード上のクラスター間でセキュアなトンネルを確立します。デフォルトの IPsec NATT ポートを使用するか、設定した別のポートを指定できます。IPsec NATT を指定せずにこの手順を実行すると、4500/UDP のポートが通信に自動的に使用されます。

カスタマイズオプションについては、Submariner デプロイメントのカスタマイズ を参照してください。

#### 1.1.5.2.2. ManagedClusterAddOn API を使用した Submariner のデプロイ

**ManagedClusterAddOn** API を使用して Submariner をデプロイするには、最初にホスティング環境でクラスターを準備する必要があります。詳細は Submariner をデプロイする一部のホストの準備 を参照してください。

クラスターを準備したら、次の手順を実行します。

1. ManagedClusterSets の作成と管理 のドキュメントの **ManagedClusterSets の作成と管理** のトピックに記載されている手順を使用して、ハブクラスターに **ManagedClusterSet** リソースを作成します。**ManagedClusterSet** のエントリーは以下のような内容になります。

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: ManagedClusterSet
metadata:
  name: <managed-cluster-set-name>
```

**managed-cluster-set-name** は、作成する **ManagedClusterSet** の名前に置き換えます。

**注記**: Kubernetes namespace の名前の最大長は 63 文字であるため、 **<managed-cluster-set-name>** の最大長さは 56 文字です。 **<managed-cluster-set-name>** の長さが 56 を超える場合には、**<managed-cluster-set-name>** は、先頭から 56 文字で省略されます。

**ManagedClusterSet** が作成されたら、 **submariner-addon** は **<managed-cluster-set-name>-broker** と呼ばれる namespace を作成し、その namespace に Submariner ブローカーをデプロイします。

2. 次の例のような YAML コンテンツをカスタマイズして適用することにより、**<managed-cluster-set-name>-broker** namespace のハブクラスターに **Broker** 設定を作成します。

```
apiVersion: submariner.io/v1alpha1
kind: Broker
metadata:
    name: submariner-broker
    namespace: <managed-cluster-set-name>-broker
spec:
    globalnetEnabled: false
```

**managed-cluster-set-name** は、マネージドクラスターの名前に置き換えます。

**ManagedClusterSet** で Submariner Globalnet を有効にする場合は、 **globalnetEnabled** の値を **true** に設定します。

3. 以下のコマンドを入力して、マネージドクラスターを 1 つ **ManagedClusterSet** に追加します。

```
oc label managedclusters <managed-cluster-name> "cluster.open-cluster-management.io/clusterset=<managed-cluster-set-name>" --overwrite
```

**managedcluster-name** は、**ManagedClusterSet** に追加するマネージドクラスターの名前に置き換えます。

**ManagedClusterSet-name** は、マネージドクラスターを追加する **ManagedClusterSet** の名前に置き換えます。

4. 次の例のような YAML コンテンツをカスタマイズして適用します。

```
apiVersion: submarineraddon.open-cluster-management.io/v1alpha1
kind: SubmarinerConfig
metadata:
   name: submariner
   namespace: <managed-cluster-namespace>
spec:{}
```

**managed-cluster-namespace** は、マネージドクラスターの namespace に置き換えます。

**注記:** 以下の例のように、**SubmarinerConfig** の名前は **submariner** である必要があります。

5. 次の例のような YAML コンテンツをカスタマイズして適用することにより、マネージドクラスターに Submariner をデプロイします。

```
apiVersion: addon.open-cluster-management.io/v1alpha1
kind: ManagedClusterAddOn
metadata:
    name: submariner
    namespace: <managed-cluster-name>
spec:
    installNamespace: submariner-operator
```

**managedcluster-name** は、Submariner で使用するマネージドクラスターの名前に置き換えます。

**ManagedClusterAddOn** の仕様の **installNamespace** フィールドは、Submariner をインストールするマネージドクラスター上の namespace に置き換えます。現在、**Submariner-operator** namespace に Submariner をインストールする必要があります。

**ManagedClusterAddOn** の作成後に、**submariner-addon** は Submariner をマネージドクラスターの **submariner-operator** namespace にデプロイします。この **ManagedClusterAddOn** のステータスから Submariner のデプロイメントステータスを表示できます。

**注記**: **ManagedClusterAddOn** の名前は **submariner** である必要があります。

6. Submariner を有効にするすべてのマネージドクラスターに対して、手順 3、4、および 5 を繰り返します。

7. マネージドクラスターに Submariner をデプロイしたら、以下のコマンドを入力して、Submariner **ManagedClusterAddOn** のステータスを確認して Submariner のデプロイメントのステータスを確認できます。

```
oc -n <managed-cluster-name> get managedclusteraddons submariner -oyaml
```

**cluster-name** は、マネージドクラスターの名前に置き換えます。

Submariner **ManagedClusterAddOn** のステータスの 3 つの条件により、Submariner のデプロイメントステータスが分かります。

- **SubmarinerGatewayNodesLabeled** の条件は、マネージドクラスターに Submariner ゲートウェイノードにラベル付けされているかどうかを示します。
- **SubmarinerAgentDegraded** の条件は、Submariner がマネージドクラスターに正常にデプロイされるかどうかを示します。
- **SubmarinerConnectionDegraded** の条件は、Submariner でマネージドクラスターで確立される接続の数を示します。

#### 1.1.5.2.3. Submariner デプロイメントのカスタマイズ

NATT (Network Address Translation-Traversal) ポート、ゲートウェイノードの数、ゲートウェイノードのインスタンスタイプなど、Submariner デプロイメントの設定の一部をカスタマイズできます。これらのカスタマイズは、すべてのプロバイダーで一貫しています。

##### 1.1.5.2.3.1. NATT ポート

NATT ポートをカスタマイズする場合は、プロバイダー環境に合わせて次の YAML コンテンツをカスタマイズして適用します。

```
apiVersion: submarineraddon.open-cluster-management.io/v1alpha1
kind: SubmarinerConfig
metadata:
   name: submariner
   namespace: <managed-cluster-namespace>
spec:
   credentialsSecret:
     name: <managed-cluster-name>-<provider>-creds
   IPSecNATTPort: <NATTPort>
```

- **managed-cluster-namespace** は、マネージドクラスターの namespace に置き換えます。
- **managed-cluster-name** は、マネージドクラスターの名前に置き換えます。
  - AWS: **provider** を **aws** に置き換えます。**<managed-cluster-name>-aws-creds** の値は、AWS の認証情報シークレット名で、この情報はハブクラスターのクラスター namespace にあります。
  - GCP: **provider** を **gcp** に置き換えます。**<managed-cluster-name>-gcp-creds** の値は、Google Cloud Platform 認証情報シークレット名を指し、ハブクラスターのクラスター namespace で見つけることができます。
  - OpenStack: **provider** を **osp** に置き換えます。**<managed-cluster-name>-osp-creds** の値は、ハブクラスターのクラスター namespace にある Red Hat OpenStack Platform 認証情報シークレット名です。
  - Azure: **provider** を **azure** に置き換えます。**<managed-cluster-name>-azure-creds** の値は、ハブクラスターのクラスター namespace で見つけることができる Microsoft Azure 認証情報シークレット名です。

- **managed-cluster-namespace** は、マネージドクラスターの namespace に置き換えます。
- **managed-cluster-name** は、マネージドクラスターの名前に置き換えます。**managed-cluster-name-gcp-creds** の値は、Google Cloud Platform 認証情報シークレット名を指し、ハブクラスターのクラスター namespace で見つけることができます。
- **NATTPort** は、使用する NATT ポートに置き換えます。

**注記:** 以下の例のように、**SubmarinerConfig** の名前は **submariner** である必要があります。

VMware vSphere 環境で NATT ポートをカスタマイズするには、次の YAML コンテンツをカスタマイズして適用します。

```
apiVersion: submarineraddon.open-cluster-management.io/v1alpha1
kind: SubmarinerConfig
metadata:
   name: submariner
   namespace: <managed-cluster-namespace>
spec:
   IPSecNATTPort: <NATTPort>
```

- **managed-cluster-namespace** は、マネージドクラスターの namespace に置き換えます。
- **NATTPort** は、使用する NATT ポートに置き換えます。

**注記:** 以下の例のように、**SubmarinerConfig** の名前は **submariner** である必要があります。

#### 1.1.5.2.3.2. ゲートウェイノードの数

ゲートウェイノードの数をカスタマイズする場合は、次の例のような YAML コンテンツをカスタマイズして適用します。

```
apiVersion: submarineraddon.open-cluster-management.io/v1alpha1
kind: SubmarinerConfig
metadata:
  name: submariner
  namespace: <managed-cluster-namespace>
spec:
  credentialsSecret:
    name: <managed-cluster-name>-<provider>-creds
 gatewayConfig:
    gateways: <gateways>
```

- **managed-cluster-namespace** は、マネージドクラスターの namespace に置き換えます。
- **managed-cluster-name** は、マネージドクラスターの名前に置き換えます。
  - AWS: **provider** を **aws** に置き換えます。**<managed-cluster-name>-aws-creds** の値は、AWS の認証情報シークレット名で、この情報はハブクラスターのクラスター namespace にあります。
  - GCP: **provider** を **gcp** に置き換えます。**<managed-cluster-name>-gcp-creds** の値は、Google Cloud Platform 認証情報シークレット名を指し、ハブクラスターのクラスター namespace で見つけることができます。

- OpenStack: **provider** を **osp** に置き換えます。**<managed-cluster-name>-osp-creds** の値は、ハブクラスターのクラスター namespace にある Red Hat OpenStack Platform 認証情報シークレット名です。

- Azure: **provider** を **azure** に置き換えます。**<managed-cluster-name>-azure-creds** の値は、ハブクラスターのクラスター namespace で見つけることができる Microsoft Azure 認証情報シークレット名です。

- **gateways** は、使用するゲートウェイ数に置き換えます。値が 1 より大きい場合には、Submariner ゲートウェイは高可用性を自動的に有効にします。

**注記:** 以下の例のように、**SubmarinerConfig** の名前は **submariner** である必要があります。

VMware vSphere 環境でゲートウェイノードの数をカスタマイズする場合は、次の例のような YAML コンテンツをカスタマイズして適用します。

```
apiVersion: submarineraddon.open-cluster-management.io/v1alpha1
kind: SubmarinerConfig
metadata:
  name: submariner
  namespace: <managed-cluster-namespace>
spec:
 gatewayConfig:
    gateways: <gateways>
```

- **managed-cluster-namespace** は、マネージドクラスターの namespace に置き換えます。

- **gateways** は、使用するゲートウェイ数に置き換えます。値が 1 より大きい場合には、Submariner ゲートウェイは高可用性を自動的に有効にします。

#### 1.1.5.2.3.3. ゲートウェイノードのインスタンスタイプ

ゲートウェイノードのインスタンスタイプをカスタマイズする場合は、次の例のような YAML コンテンツをカスタマイズして適用します。

```
apiVersion: submarineraddon.open-cluster-management.io/v1alpha1
kind: SubmarinerConfig
metadata:
  name: submariner
  namespace: <managed-cluster-namespace>
spec:
  credentialsSecret:
    name: <managed-cluster-name>-<provider>-creds
 gatewayConfig:
    instanceType: <instance-type>
```

- **managed-cluster-namespace** は、マネージドクラスターの namespace に置き換えます。

- **managed-cluster-name** は、マネージドクラスターの名前に置き換えます。

- AWS: **provider** を **aws** に置き換えます。**<managed-cluster-name>-aws-creds** の値は、AWS の認証情報シークレット名で、この情報はハブクラスターのクラスター namespace にあります。

- GCP: **provider** を **gcp** に置き換えます。**<managed-cluster-name>-gcp-creds** の値は、Google Cloud Platform 認証情報シークレット名を指し、ハブクラスターのクラスター namespace で見つけることができます。

- OpenStack: **provider** を **osp** に置き換えます。**<managed-cluster-name>-osp-creds** の値は、ハブクラスターのクラスター namespace にある Red Hat OpenStack Platform 認証情報シークレット名です。

- Azure: **provider** を **azure** に置き換えます。**<managed-cluster-name>-azure-creds** の値は、ハブクラスターのクラスター namespace で見つけることができる Microsoft Azure 認証情報シークレット名です。

- **instance-type** は、使用する AWS インスタンスタイプに置き換えます。

**注記:** 以下の例のように、**SubmarinerConfig** の名前は **submariner** である必要があります。

#### 1.1.5.2.3.4. ケーブルドライバー

Submariner Gateway Engine コンポーネントは、他のクラスターへの安全なトンネルを作成します。ケーブルドライバーコンポーネントは、ゲートウェイエンジンコンポーネントのプラグ可能なアーキテクチャーを使用してトンネルを維持します。ケーブルエンジンコンポーネントの **cableDriver** 設定には、Libreswan または VXLAN 実装を使用できます。以下の例を参照してください。

```
apiVersion: submarineraddon.open-cluster-management.io/v1alpha1
kind: SubmarinerConfig
metadata:
  name: submariner
  namespace: <managed-cluster-namespace>
spec:
  cableDriver: vxlan
  credentialsSecret:
    name: <managed-cluster-name>-<provider>-creds
```

**ベストプラクティス:** パブリックネットワークでは VXLAN ケーブルドライバーを使用しないでください。VXLAN ケーブルドライバーは暗号化されていません。プライベートネットワークでの不要な二重暗号化を避けるために、VXLAN のみを使用してください。たとえば、一部のオンプレミス環境では、専用の回線レベルのハードウェアデバイスを使用してトンネルの暗号化を処理する場合があります。

### 1.1.5.3. Submariner のサービス検出の管理

Submariner がマネージドクラスターと同じ環境にデプロイされた後、マネージドクラスターセット内のクラスター全体で Pod とサービス間の安全な IP ルーティングのためにルートが設定されます。

#### 1.1.5.3.1. Submariner のサービス検出の有効化

クラスターからのサービスをマネージドクラスターセット内の他のクラスターに表示および検出可能にするには、**ServiceExport** オブジェクトを作成する必要があります。**ServiceExport** オブジェクトでサービスをエクスポートすると、**<service>.<namespace>.svc.clusterset.local** 形式でサービスにアクセスできます。複数のクラスターが同じ名前で、同じ namespace からサービスをエクスポートすると、他のクラスターは、その複数のクラスターを 1 つの論理サービスとして認識します。

この例では、**default** の namespace で **nginx** サービスを使用しますが、Kubernetes の **ClusterIP** サービスまたはヘッドレスサービスを検出できます。

1. 以下のコマンドを入力して、**ManagedClusterSet** のマネージドクラスターに **nginx** サービスのインスタンスを適用します。

```
oc -n default create deployment nginx --image=nginxinc/nginx-unprivileged:stable-alpine
oc -n default expose deployment nginx --port=8080
```

2. 次のコマンドのような **subctl** ツールを使用してコマンドを入力し、**ServiceExport** エントリーを作成して、サービスをエクスポートします。

```
subctl export service --namespace <service-namespace> <service-name>
```

**service-namespace** を、サービスが置かれた namespace の名前に置き換えます。この例では、**default** になります。

**service-name** を、エクスポートするサービスの名前に置き換えます。この例では、**nginx** になります。

その他の使用可能なフラグの詳細については、Submariner ドキュメントの **export** を参照してください。

3. 別のマネージドクラスターから以下のコマンドを実行して、**nginx** サービスにアクセスできることを確認します。

```
oc -n default run --generator=run-pod/v1 tmp-shell --rm -i --tty --image
quay.io/submariner/nettest -- /bin/bash curl nginx.default.svc.clusterset.local:8080
```

これで、**nginx** サービス検出が Submariner に対して設定されました。

#### 1.1.5.3.2. Submariner のサービス検出の無効化

サービスが他のクラスターにエクスポートされないようにするには、**nginx** の次の例のようなコマンドを入力します。

```
subctl unexport service --namespace <service-namespace> <service-name>
```

**service-namespace** を、サービスが置かれた namespace の名前に置き換えます。

**service-name** を、エクスポートするサービスの名前に置き換えます。

その他の使用可能なフラグの詳細は、Submariner ドキュメントの **unexport** を参照してください。

このサービスは、クラスターによる検出に使用できなくなりました。

### 1.1.5.4. Submariner のアンインストール

Red Hat Advanced Cluster Management for Kubernetes コンソールまたはコマンドラインを使用して、クラスターから Submariner コンポーネントをアンインストールできます。0.12 より前の Submariner バージョンで、すべてのデータプレーンコンポーネントを完全に削除するには、追加の手順が必要です。Submariner のアンインストールはべき等であるため、問題なく手順を繰り返すことができます。

#### 1.1.5.4.1. コンソール方式

コンソールを使用してクラスターから Submariner をアンインストールするには、次の手順を実行します。

1. コンソールナビゲーションから、**Infrastructure** > **Clusters** を選択し、**Cluster sets** タブを選択します。
2. Submariner コンポーネントを削除するクラスターを含むクラスターセットを選択します。
3. **Submariner Add-ons** タブを選択して、Submariner がデプロイされているクラスターセット内のクラスターを表示します。
4. Submariner をアンインストールするクラスターの **Actions** メニューで、**Uninstall Add-on** を選択します。
5. Submariner をアンインストールするクラスターの **アクション** メニューで、**クラスターセットの削除** を選択します。
6. Submariner を削除する他のクラスターについても、これらの手順を繰り返します。
**ヒント:** 複数のクラスターを選択して **Actions** をクリックすると、同じクラスターセット内の複数のクラスターから Submariner アドオンを削除できます。**Uninstall Submariner add-ons** を選択します。

削除する Submariner のバージョンがバージョン 0.12 より前の場合は、Submariner の初期バージョンの手動削除手順 に進みます。Submariner のバージョンが 0.12 以降の場合、Submariner は削除されます。

**重要:** クラウドプロバイダーによる追加料金を回避するために、すべてのクラウドリソースがクラウドプロバイダーから削除されていることを確認してください。詳細は、Submariner リソースの削除の確認 を参照してください。

#### 1.1.5.4.2. コマンドライン方式

コマンドラインを使用して Submariner をアンインストールするには、次の手順を実行します。

1. 次のコマンドを実行して、クラスターの Submariner デプロイメントを削除します。

```
oc -n <managed-cluster-namespace> delete managedclusteraddon submariner
```

**managed-cluster-namespace** は、マネージドクラスターの namespace に置き換えます。

2. 次のコマンドを実行して、クラスターのクラウドリソースを削除します。

```
oc -n <managed-cluster-namespace> delete submarinerconfig submariner
```

**managed-cluster-namespace** は、マネージドクラスターの namespace に置き換えます。

3. 次のコマンドを実行して、クラスターセットを削除し、ブローカーの詳細を削除します。

```
oc delete managedclusterset <managedclusterset>
```

**managedclusterset** をマネージドクラスターセットの名前に置き換えます。

削除する Submariner のバージョンがバージョン 0.12 より前の場合は、Submariner の初期バージョンの手動削除手順 に進みます。Submariner のバージョンが 0.12 以降の場合、Submariner は削除されます。

**重要:** クラウドプロバイダーによる追加料金を回避するために、すべてのクラウドリソースがクラウドプロバイダーから削除されていることを確認してください。詳細は、Submariner リソースの削除の確認 を参照してください。

#### 1.1.5.4.3. Submariner の初期バージョンの手動削除手順

バージョン 0.12 より前のバージョンの Submariner をアンインストールする場合は、Submariner ドキュメントの 手動アンインストール セクションの手順 5 ~ 8 を実行してください。

これらの手順を完了すると、Submariner コンポーネントがクラスターから削除されます。

**重要:** クラウドプロバイダーによる追加料金を回避するために、すべてのクラウドリソースがクラウドプロバイダーから削除されていることを確認してください。詳細は、Submariner リソースの削除の確認 を参照してください。

#### 1.1.5.4.4. Submariner リソースの削除の確認

Submariner をアンインストールした後、すべての Submariner リソースがクラスターから削除されていることを確認します。それらがクラスターに残っている場合、一部のリソースはインフラストラクチャープロバイダーからの料金を引き続き発生させます。次の手順を実行して、クラスターに追加の Submariner リソースがないことを確認します。

1. 次のコマンドを実行して、クラスターに残っている Submariner リソースを一覧表示します。

   ```
   oc get cluster <CLUSTER_NAME> grep submariner
   ```

   **CLUSTER_NAME** をクラスターの名前に置き換えます。

2. 次のコマンドを入力して、リストのリソースをすべて削除します。

   ```
   oc delete resource <RESOURCE_NAME> cluster <CLUSTER_NAME>
   ```

   **RESOURCE_NAME** を、削除する Submariner リソースの名前に置き換えます。

3. 検索でリソースが特定されなくなるまで、クラスターごとに手順 1 ~ 2 を繰り返します。

Submariner リソースがクラスターから削除されます。

## 1.2. VOLSYNC の永続ボリューム複製サービス

VolSync は、レプリケーションの互換性がないストレージタイプが指定されたクラスター全体、または 1 つのクラスター内の永続ボリュームの非同期レプリケーションを有効にする Kubernetes Operator です。これは Container Storage Interface (CSI) を使用して互換性の制限を解消します。VolSync Operator を環境にデプロイした後、それを活用して永続データのコピーを作成および保守できます。VolSync は、バージョン 4.8 以降の Red Hat OpenShift Container Platform クラスターでのみ永続的なボリュームクレームを複製できます。

- VolSync を使用した永続ボリュームの複製
  - マネージドクラスターへの VolSync のインストール
  - Rsync レプリケーションの設定
  - restic バックアップの設定
  - Rclone レプリケーションの設定
- 複製されたイメージを使用可能な永続的なボリュームクレームに変換
- 同期のスケジューリング

### 1.2.1. VolSync を使用した永続ボリュームの複製

サポート対象の方法 3 つを使用して、VolSync で永続ボリュームを複製できます。Rsync、restic または Rclone など所有する同期ロケーションの数により異なります。

#### 1.2.1.1. 前提条件

クラスターに VolSync をインストールする前に、以下の要件が必要です。

- Red Hat Advanced Cluster Management バージョン 2.4 以降のハブクラスターで実行中で、設定済みの OpenShift Container Platform 環境。
- 同じ Red Hat Advanced Cluster Management ハブクラスターが管理する 2 つ以上のクラスター。
- VolSync で設定しているクラスター間のネットワーク接続。クラスターが同じネットワーク上にない場合は、Submariner マルチクラスターネットワークとサービスディスカバリー を設定し、**ServiceType** の **ClusterIP** 値を使用してクラスターをネットワーク化するか、**ServiceType** に **LoadBalancer** の値を指定してロードバランサーを使用できます。
- ソース永続ボリュームに使用するストレージドライバーは、CSI 互換であり、スナップショットをサポートできる必要があります。

#### 1.2.1.2. マネージドクラスターへの VolSync のインストール

VolSync が 1 つのクラスターの永続ボリューム要求を別のクラスターの Persistent Volume Claims に複製できるようにするには、ソースとターゲットの両方のマネージドクラスターに VolSync をインストールする必要があります。

VolSync は独自の namespace を作成しないため、他の OpenShift Container Platform のすべての namespace Operator と同じ namespace にあります。VolSync の Operator 設定に加えた変更は、チャネル更新の手動承認に変更した場合など、同じ namespace 内の他の Operator にも影響します。

2 つの方法のいずれかを使用して、環境内の 2 つのクラスターに VolSync をインストールできます。次のセクションで説明するように、ハブクラスター内の各マネージドクラスターにラベルを追加するか、**ManagedClusterAddOn** を手動で作成して適用することができます。

##### 1.2.1.2.1. ラベルを使用した VolSync のインストール

ラベルを追加して、マネージドクラスターに VolSync をインストールします。

- Red Hat Advanced Cluster Management コンソールから以下のステップを実行します。
  1. 詳細を表示するには、ハブクラスターコンソールの **Clusters** ページからマネージドクラスターの 1 つを選択します。
  2. **Labels** フィールドに、次のラベルを追加します。

     ```
     addons.open-cluster-management.io/volsync=true
     ```

     VolSync サービス Pod はマネージドクラスターにインストールされます。
  3. 他のマネージドクラスターに同じラベルを追加します。
  4. 各マネージドクラスターで次のコマンドを実行して、VolSync Operator がインストールされていることを確認します。

```
oc get csv -n openshift-operators
```

インストール時に VolSync の Operator がリストされています。

- コマンドラインインターフェイスから次の手順を実行します。

  1. ハブクラスターでコマンドラインセッションを開始します。
  2. 次のコマンドを入力して、最初のクラスターにラベルを追加します。

     ```
     oc label managedcluster <managed-cluster-1> "addons.open-cluster-management.io/volsync"="true"
     ```

     **managed-cluster-1** をマネージドクラスターの 1 つの名前に置き換えます。
  3. 次のコマンドを入力して、ラベルを 2 番目のクラスターに追加します。

     ```
     oc label managedcluster <managed-cluster-2> "addons.open-cluster-management.io/volsync"="true"
     ```

     **managed-cluster-2** を他のマネージドクラスターの名前に置き換えます。

     **ManagedClusterAddOn** リソースは、対応する各マネージドクラスターの namespace 内のハブクラスターに自動的に作成される必要があります。

#### 1.2.1.2.2. ManagedClusterAddOn を使用した VolSync のインストール

**ManagedClusterAddOn** を手動で追加して VolSync をマネージドクラスターにインストールするには、次の手順を実行します。

1. ハブクラスターで、次の例のようなコンテンツを含む **volsync-mcao.yaml** という YAML ファイルを作成します。

   ```
   apiVersion: addon.open-cluster-management.io/v1alpha1
   kind: ManagedClusterAddOn
   metadata:
     name: volsync
     namespace: <managed-cluster-1-namespace>
   spec: {}
   ```

   **managed-cluster-1-namespace** を、マネージドクラスターの 1 つの namespace に置き換えます。この namespace は、マネージドクラスターの名前と同じです。

   **注:** 名前は **volsync** である必要があります。

2. 次の例のようなコマンドを入力して、ファイルを設定に適用します。

   ```
   oc apply -f volsync-mcao.yaml
   ```

3. 他のマネージドクラスターに対して手順を繰り返します。
   **ManagedClusterAddOn** リソースは、対応する各マネージドクラスターの namespace 内のハブクラスターに自動的に作成される必要があります。

### 1.2.1.3. Rsync レプリケーションの設定

Rsync レプリケーションを使用して、永続ボリュームの 1:1 非同期レプリケーションを作成できます。Rsync ベースのレプリケーションを災害復旧やリモートサイトへのデータ送信に使用できます。

次の例は、Rsync メソッドを使用して設定する方法を示しています。Rsync の追加情報については、VolSync ドキュメントの Usage を参照してください。

#### 1.2.1.3.1. マネージドクラスター間での Rsync レプリケーションの設定

Rsync ベースのレプリケーションの場合は、ソースクラスターおよび宛先クラスターでカスタムリソースを設定します。カスタムリソースは、**address** 値を使用してソースを宛先に接続し、**sshKeys** を使用して転送されたデータがセキュアであることを確認します。

**注記: address** および **sshKeys** の値を宛先からソースにコピーし、ソースを設定する前に宛先を設定する必要があります。

この例では、**source-ns** namespace の **source** クラスターの永続ボリューム要求から **destination-ns** namespace の **destination** クラスターの永続ボリューム要求に Rsync レプリケーションを設定する手順を説明します。必要に応じて、これらの値を他の値に置き換えることができます。

1. 宛先クラスターを設定します。
   a. 宛先クラスターで次のコマンドを実行して、ネームスペースを作成します。

      ```
      oc create ns <destination-ns>
      ```

      **destination-ns** を、オンサイトのボリューム要求ごとに宛先が含まれる namespace の名前に置き換えます。

   b. 以下の YAML コンテンツをコピーし、**replication_destination.yaml** という名前の新規ファイルを作成します。

      ```
      apiVersion: volsync.backube/v1alpha1
      kind: ReplicationDestination
      metadata:
        name: <destination>
        namespace: <destination-ns>
      spec:
        rsync:
          serviceType: LoadBalancer
          copyMethod: Snapshot
          capacity: 2Gi
          accessModes: [ReadWriteOnce]
          storageClassName: gp2-csi
          volumeSnapshotClassName: csi-aws-vsc
      ```

      **注記: capacity** の値は、レプリケートされる永続ボリューム要求 (PVC) の容量と一致する必要があります。

      **destination** は、宛先 CR の名前に置き換えます。

      **destination-ns** は、宛先の namespace の名前に置き換えます。

      この例では、**LoadBalancer** の **ServiceType** 値が使用されます。ロードバランサーサービスはソースクラスターによって作成され、ソースマネージドクラスターが別の宛先マネージドクラスターに情報を転送できるようにします。ソースと宛先が同じクラスター上にあ

る場合、または Submariner ネットワークサービスが設定されている場合は、サービスタイプとして **ClusterIP** を使用できます。ソースクラスターを設定するときに参照するシークレットのアドレスと名前をメモします。

**storageClassName** および **volumeSnapshotClassName** は任意のパラメーターです。特に、環境のデフォルト値とは異なるストレージクラスおよびボリュームスナップショットクラス名を使用している場合は、環境の値を指定してください。

c. 宛先クラスターで以下のコマンドを実行し、**replicationdestination** リソースを作成します。

```
oc create -n <destination-ns> -f replication_destination.yaml
```

**destination-ns** は、宛先の namespace の名前に置き換えます。

**replicationdestination** リソースが作成されると、以下のパラメーターおよび値がリソースに追加されます。

| パラメーター | 値 |
|---|---|
| **.status.rsync.address** | 送信元クラスターと宛先クラスターが通信できるようにするために使用される宛先クラスターの IP アドレス。 |
| **.status.rsync.sshKeys** | ソースクラスターから宛先クラスターへの安全なデータ転送を可能にする SSH キーファイルの名前。 |

d. 以下のコマンドを実行して、ソースクラスターで使用する **.status.rsync.address** の値をコピーします。

```
ADDRESS=`oc get replicationdestination <destination> -n <destination-ns> --template=
{{.status.rsync.address}}`
echo $ADDRESS
```

**destination** は、レプリケーション先のカスタムリソースの名前に置き換えます。

**destination-ns** は、宛先の namespace の名前に置き換えます。

出力は、Amazon Web Services 環境の次の出力のように表示されます。

```
a831264645yhrjrjyer6f9e4a02eb2-5592c0b3d94dd376.elb.us-east-1.amazonaws.com
```

e. 以下のコマンドを実行して、シークレットの名前および **.status.rsync.sshKeys** の値として提供されるシークレットの内容をコピーします。

```
SSHKEYS=`oc get replicationdestination <destination> -n <destination-ns> --template=
{{.status.rsync.sshKeys}}`
echo $SSHKEYS
```

**destination** は、レプリケーション先のカスタムリソースの名前に置き換えます。

**destination-ns** は、宛先の namespace の名前に置き換えます。

ソースの設定時に、ソースクラスターで入力する必要があります。出力は、SSH キーシークレットファイルの名前である必要があります。これは、次の名前のようになります。

```
volsync-rsync-dst-src-destination-name
```

2. 複製するソース永続ボリュームクレームを特定します。
**注記:** ソース永続ボリューム要求は CSI ストレージクラスにある必要があります。

3. **ReplicationSource** アイテムを作成します。

   a. 以下の YAML コンテンツをコピーして、ソースクラスターに **replication_source.yaml** という名前の新規ファイルを作成します。

```
apiVersion: volsync.backube/v1alpha1
kind: ReplicationSource
metadata:
  name: <source>
  namespace: <source-ns>
spec:
  sourcePVC: <persistent_volume_claim>
  trigger:
    schedule: "*/3 * * * *" #/*
  rsync:
    sshKeys: <mysshkeys>
    address: <my.host.com>
    copyMethod: Snapshot
    storageClassName: gp2-csi
    volumeSnapshotClassName: gp2-csi
```

   **source** は、レプリケーションソースカスタムリソースの名前に置き換えます。これを自動的に置き換える方法については、この手順のステップ **3-vi** を参照してください。

   **source-ns** をソースが置かれている Persistent Volume Claim の namespace に置き換えます。これを自動的に置き換える方法については、この手順のステップ **3-vi** を参照してください。

   **persistent_volume_claim** は、ソース永続ボリューム要求の名前に置き換えます。

   **mysshkeys** は、設定時に **ReplicationDestination** の **.status.rsync.sshKeys** フィールドからコピーしたキーに置き換えます。

   **my.host.com** は、設定時に **ReplicationDestination** の **.status.rsync.address** フィールドからコピーしたホストアドレスに置き換えます。

   ストレージドライバーがクローン作成をサポートする場合は、**copyMethod** の値に **Clone** を使用すると、レプリケーションのより効率的なプロセスになる可能性があります。

   **storageClassName** および **volumeSnapshotClassName** はオプションのパラメーターです。ご使用の環境のデフォルトとは異なるストレージクラスおよびボリュームスナップショットクラス名を使用している場合は、それらの値を指定してください。

   永続ボリュームの同期方法を設定できるようになりました。

   b. 宛先クラスターに対して次のコマンドを入力して、宛先クラスターから SSH シークレットをコピーします。

```
oc get secret -n <destination-ns> $SSHKEYS -o yaml > /tmp/secret.yaml
```

**destination-ns** を、宛先が置かれている永続ボリューム要求の namespace に置き換えます。

c. 以下のコマンドを入力して、**vi** エディターでシークレットファイルを開きます。

```
vi /tmp/secret.yaml
```

d. 宛先クラスターのオープンシークレットファイルで、次の変更を行います。

- 名前空間をソースクラスターの名前空間に変更します。この例では、**source-ns** です。
- 所有者の参照を削除します (**.metadata.ownerReferences**)。

e. ソースクラスターで、ソースクラスターで次のコマンドを入力してシークレットファイルを作成します。

```
kubectl create -f /tmp/secret.yaml
```

f. ソースクラスターで、以下のコマンドを入力して **ReplicationSource** オブジェクトの **address** および **sshKeys** の値を、宛先クラスターから書き留めた値に置き換えて **replication_source.yaml** ファイルを変更します。

```
sed -i "s/<my.host.com>/$ADDRESS/g" replication_source.yaml
sed -i "s/<mysshkeys>/$SSHKEYS/g" replication_source.yaml
oc create -n <source> -f replication_source.yaml
```

**my.host.com** は、設定時に **ReplicationDestination** の **.status.rsync.address** フィールドからコピーしたホストアドレスに置き換えます。

**mysshkeys** は、設定時に **ReplicationDestination** の **.status.rsync.sshKeys** フィールドからコピーしたキーに置き換えます。

**source** を、ソース が置かれている永続ボリューム要求の名前に置き換えます。

**注記:** 複製する永続ボリューム要求と同じ namespace にファイルを作成する必要があります。

g. **ReplicationSource** オブジェクトで以下のコマンドを実行して、レプリケーションが完了したことを確認します。

```
kubectl describe ReplicationSource -n <source-ns> <source>
```

**source-ns** をソースが置かれている Persistent Volume Claim の namespace に置き換えます。

**source** は、レプリケーションソースのカスタムリソースの名前に置き換えます。

レプリケーションが成功した場合、出力は次の例のようになります。

```
Status:
  Conditions:
    Last Transition Time:  2021-10-14T20:48:00Z
```

```
  Message:               Synchronization in-progress
  Reason:                SyncInProgress
  Status:                True
  Type:                  Synchronizing
  Last Transition Time:  2021-10-14T20:41:41Z
  Message:               Reconcile complete
  Reason:                ReconcileComplete
  Status:                True
  Type:                  Reconciled
 Last Sync Duration:     5m20.764642395s
 Last Sync Time:         2021-10-14T20:47:01Z
 Next Sync Time:         2021-10-14T20:48:00Z
```

**Last Sync Time** に時間がリストされていない場合は、レプリケーションが完了していません。

元の永続ボリュームのレプリカがあります。

#### 1.2.1.4. restic バックアップの設定

Restic ベースのバックアップは、永続ボリュームの Restic ベースのバックアップコピーを、**restic-config.yaml** シークレットファイルで指定された場所にコピーします。Restic バックアップは、クラスター間でデータを同期しませんが、データをバックアップします。

次の手順を実行して、restic ベースのバックアップを設定します。

1. 次の YAML コンテンツのようなシークレットを作成して、バックアップイメージが保存されるリポジトリーを指定します。

```
apiVersion: v1
kind: Secret
metadata:
  name: restic-config
type: Opaque
stringData:
  RESTIC_REPOSITORY: <my-restic-repository>
  RESTIC_PASSWORD: <my-restic-password>
  AWS_ACCESS_KEY_ID: access
  AWS_SECRET_ACCESS_KEY: password
```

**my-restic-repository** は、バックアップファイルを保存する S3 バケットリポジトリーの場所に置き換えます。

**my-restic-password** は、リポジトリーへのアクセスに必要な暗号化キーに置き換えます。

必要に応じて、**アクセス** と **パスワード** は、プロバイダーのクレデンシャルに置き換えます。詳細は 新しいリポジトリーの準備 を参照してください。

**重要:** 複数の永続ボリューム要求を同じ S3 バケットにバックアップする場合には、バケットへのパスは永続ボリュームクレームごとに一意である必要があります。各永続ボリュームクレームは個別の **ReplicationSource** でバックアップされるので、個別の restic-config シークレットが必要です。

同じ S3 バケットを共有することで、各 **ReplicationSource** は S3 バケット全体への書き込みアクセスが割り当てられます。

2. 次の YAML コンテンツに似た **ReplicationSource** オブジェクトを作成して、バックアップポリシーを設定します。

```
apiVersion: volsync.backube/v1alpha1
kind: ReplicationSource
metadata:
  name: mydata-backup
spec:
  sourcePVC: <source>
  trigger:
    schedule: "*/30 * * * *" #/*
  restic:
    pruneIntervalDays: 14
    repository: <restic-config>
    retain:
      hourly: 6
      daily: 5
      weekly: 4
      monthly: 2
      yearly: 1
    copyMethod: Clone
  # The StorageClass to use when creating the PiT copy (same as source PVC if omitted)
  #storageClassName: my-sc-name
  # The VSC to use if the copy method is Snapshot (default if omitted)
  #volumeSnapshotClassName: my-vsc-name
```

**source** は、バックアップしている永続ボリュームクレームに置き換えます。

**schedule** の値は、バックアップを実行する頻度に置き換えます。この例では、30 分ごとにスケジュールが指定されています。詳しくは Synchronization のスケジュール を参照してください。

**PruneIntervalDays** の値は、インスタンスで次にデータの圧縮するまでの経過時間 (日数) に置き換えて、スペースを節約します。プルーニング操作は、実行中に大量の I/O トラフィックを生成する可能性があります。

**restic-config** は、ステップ 1 で作成したシークレットの名前に置き換えます。

**retain** の値は、バックアップしたイメージの保持ポリシーに設定します。

ベストプラクティス: **CopyMethod** の値に **Clone** を使用して、特定の時点のイメージが確実に保存されるようにします。

バックアップオプションの詳細は、VolSync ドキュメントの Backup options を参照してください。

#### 1.2.1.4.1. restic バックアップの復元

コピーされたデータを restic バックアップから新しい永続ボリューム要求に復元できます。**ベストプラクティス:** バックアップ 1 つだけを新しい永続ボリューム要求に復元します。Restic バックアップを復元するには、次の手順を実行します。

1. 次の例のように、新しいデータを含む新しい永続ボリュームクレームを作成します。

```
kind: PersistentVolumeClaim
apiVersion: v1
```

```
metadata:
  name: <pvc-name>
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 3Gi
```

**pvc-name** は、新しい永続ボリュームクレームの名前に置き換えます。

2. 次の例のような **ReplicationDestination** カスタムリソースを作成して、データの復元先を指定します。

```
apiVersion: volsync.backube/v1alpha1
kind: ReplicationDestination
metadata:
  name: <destination>
spec:
  trigger:
    manual: restore-once
  restic:
    repository: <restic-repo>
    destinationPVC: <pvc-name>
    copyMethod: Direct
```

**destination** は、宛先 CR の名前に置き換えます。

**restic-repo** は、ソースが保存されているリポジトリーへのパスに置き換えます。

**pvc-name** は、データを復元する新しい永続ボリュームクレームの名前に置き換えます。これには、新しいボリューム要求をプロビジョニングするのではなく、既存の永続ボリューム要求を使用してください。

復元プロセスは 1 回だけ完了する必要があります。この例では、最新のバックアップを復元します。復元オプションの詳細は、VolSync ドキュメントの Restore options を参照してください。

#### 1.2.1.5. Rclone レプリケーションの設定

Rclone バックアップは、Rclone を使用して AWS S3 などの中間オブジェクトストレージの場所を介して単一の永続ボリュームを複数の場所にコピーします。複数の場所にデータを配布する場合に役立ちます。

次の手順を実行して、Rclone レプリケーションを設定します。

1. 次の例のような **ReplicationSource** カスタムリソースを作成します。

```
apiVersion: volsync.backube/v1alpha1
kind: ReplicationSource
metadata:
  name: <source>
  namespace: <source-ns>
spec:
  sourcePVC: <source-pvc>
  trigger:
    schedule: "*/6 * * * *" #\*
```

```
  rclone:
    rcloneConfigSection: <intermediate-s3-bucket>
    rcloneDestPath: <destination-bucket>
    rcloneConfig: <rclone-secret>
    copyMethod: Snapshot
    storageClassName: <my-sc-name>
    volumeSnapshotClassName: <my-vsc>
```

**source-pvc** は、レプリケーションソースのカスタムリソースの名前に置き換えます。

**source-ns** をソースが置かれている Persistent Volume Claim の namespace に置き換えます。

**source** は、レプリケートしている永続ボリュームクレームに置き換えます。

**スケジュール** の値は、レプリケーションを実行する頻度に置き換えます。この例では、6 分ごとにスケジュールが指定されています。この値は引用符で囲む必要があります。詳しくは Synchronization のスケジュール を参照してください。

**Intermediate-s3-bucket** は、Rclone 設定ファイルの設定セクションへのパスに置き換えます。

**destination-bucket** は、レプリケートされたファイルをコピーするオブジェクトバケットへのパスに置き換えます。

**rclone-secret** は、Rclone 設定情報を含むシークレットの名前に置き換えます。

**copyMethod** の値は **Clone**、**Direct**、または **Snapshot** として設定します。この値は、ある特定の時点でのコピーを生成するかどうか、生成する場合は、生成方法を指定します。

**my-sc-name** は、ポイントインタイムコピーに使用するストレージクラスの名前に置き換えます。指定しない場合、ソースボリュームのストレージクラスが使用されます。

**スナップショット** を **copyMethod** として指定した場合は **my-vsc** を使用する **VolumeSnapshotClass** の名前に置き換えます。これは、他のタイプの **copyMethod** には必要ありません。

2. 次の例のような **ReplicationDestination** カスタムリソースを作成します。

```
apiVersion: volsync.backube/v1alpha1
kind: ReplicationDestination
metadata:
  name: database-destination
  namespace: dest
spec:
  trigger:
    schedule: "3,9,15,21,27,33,39,45,51,57 * * * *" #\*
  rclone:
    rcloneConfigSection: <intermediate-s3-bucket>
    rcloneDestPath: <destination-bucket>
    rcloneConfig: <rclone-secret>
    copyMethod: Snapshot
    accessModes: [ReadWriteOnce]
    capacity: 10Gi
    storageClassName: <my-sc>
    volumeSnapshotClassName: <my-vsc>
```

**スケジュール** の値は、レプリケーションを宛先に移動する頻度に置き換えます。移動元と宛先

のスケジュールをオフセットして、データが宛先からプルされる前に複製を完了できるようにする必要があります。この例では、オフセットは 3 分で、6 分間隔でスケジュールされています。この値は引用符で囲む必要があります。詳しくは Synchronization のスケジュール を参照してください。

**Intermediate-s3-bucket** は、Rclone 設定ファイルの設定セクションへのパスに置き換えます。

**destination-bucket** は、レプリケートされたファイルをコピーするオブジェクトバケットへのパスに置き換えます。

**rclone-secret** は、Rclone 設定情報を含むシークレットの名前に置き換えます。

**copyMethod** の値は **Clone**、**Direct**、または **Snapshot** として設定します。この値は、ある特定の時点でのコピーを生成するかどうか、生成する場合は、生成方法を指定します。

**accessModes** の値は、永続ボリュームクレームのアクセスモードを指定します。有効な値は **ReadWriteOnce** または **ReadWriteMany** です。

**容量** は宛先ボリュームのサイズを指定します。このサイズは、着信データを格納するのに十分な大きさに指定します。

**my-sc** は、特定の時点のコピーの宛先として使用するストレージクラスの名前に置き換えます。指定しない場合、システムストレージクラスが使用されます。

**スナップショット** を **copyMethod** として指定した場合は **my-vsc** を使用する **VolumeSnapshotClass** の名前に置き換えます。これは、他のタイプの **copyMethod** には必要ありません。含まれていない場合は、システムのデフォルトの **VolumeSnapshotClass** が使用されます。

### 1.2.2. 複製されたイメージを使用可能な永続的なボリュームクレームに変換

複製されたイメージを使用してデータを回復するか、永続的なボリュームクレームの新しいインスタンスを作成する必要がある場合があります。イメージのコピーは、使用する前に永続的なボリュームクレームに変換する必要があります。複製されたイメージを永続的なボリュームクレームに変換するには、次の手順を実行します。

1. レプリケーションが完了したら、以下のコマンドを入力して **ReplicationDestination** オブジェクトから最新のスナップショットを特定します。

```
$ kubectl get replicationdestination <destination> -n <destination-ns> --template=
{{.status.latestImage.name}}
```

永続的なボリュームクレームを作成するときの最新のスナップショットの値に注意してください。

**destination** は、レプリケーション先の名前に置き換えます。

**destination-ns** は、宛先の namespace に置き換えます。

2. 以下の例のような **pvc.yaml** ファイルを作成します。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: <pvc-name>
  namespace: <destination-ns>
```

```
spec:
  accessModes:
    - ReadWriteOnce
  dataSource:
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
    name: <snapshot_to_replace>
  resources:
    requests:
      storage: 2Gi
```

**pvc-name** は、新規の永続ボリューム要求 (PVC) の名前に置き換えます。

**destination-ns** を、永続ボリューム要求 (PVC) が置かれている namespace に置き換えます。

**snapshot_to_replace** を、直前の手順で確認した **VolumeSnapshot** 名に置き換えます。

**ベストプラクティス:** 値が少なくとも最初のソース永続ボリュームクレームと同じサイズである場合は、**resources.requests.storage** を異なる値で更新できます。

3. 次のコマンドを入力して、永続ボリュームクレームが環境で実行されていることを確認します。

```
$ kubectl get pvc -n <destination-ns>
```

元のバックアップイメージは、メインの永続ボリューム要求として実行されています。

### 1.2.3. 同期のスケジューリング

レプリケーションの開始方法を決定するときは、常に実行する、スケジュールどおりに実行する、または手動で実行するという 3 つのオプションから選択します。レプリケーションのスケジュールは、よく選択されるオプションです。

**スケジュール** オプションは、スケジュール時にレプリケーションを実行します。スケジュールは **cronspec** で定義されるため、スケジュールを時間間隔または特定の時間として設定できます。スケジュールの値の順序は次のとおりです。

**"minute (0-59) hour (0-23) day-of-month (1-31) month (1-12) day-of-week (0-6)"**

レプリケーションはスケジュールされた時間に開始されます。このレプリケーションオプションの設定は、以下の内容のようになります。

```
spec:
  trigger:
    schedule: "*/6 * * * *"
```

これらの方法のいずれかを有効にしたら、設定した方法に従って同期スケジュールが実行されます。

追加情報およびオプションについては、VolSync のドキュメントを参照してください。

## 1.3. KUBERNETES OPERATOR のマルチクラスターエンジンからクラスターで KLUSTERLET アドオンを有効にする

Red Hat Advanced Cluster Management for Kubernetes をインストールし、Kubernetes Operator 用のマルチクラスターエンジンを使用してクラスターを作成またはインポートした後、それらのマネージドクラスターに対して klusterlet アドオンを有効にできます。

Kubernetes Operator のマルチクラスターエンジンを使用してクラスターを作成またはインポートした場合、klusterlet アドオンはデフォルトで有効になっていません。さらに、Red Hat Advanced Cluster Management のインストール後、klusterlet アドオンはデフォルトで有効になりません。

次の利用可能な klusterlet アドオンを参照してください。

- application-manager
- cert-policy-controller
- config-policy-controller
- iam-policy-controller
- governance-policy-framework
- search-collector

Red Hat Advanced Cluster Management のインストール後に、マネージドクラスターの klusterlet アドオンを有効にするには、以下のステップを実行します。

1. 次の **KlusterletAddonConfig** に似た YAML ファイルを作成し、アドオンを表す **spec** 値を使用します。

```
apiVersion: agent.open-cluster-management.io/v1
kind: KlusterletAddonConfig
metadata:
  name: <cluster_name>
  namespace: <cluster_name>
spec:
  applicationManager:
    enabled: true
  certPolicyController:
    enabled: true
  iamPolicyController:
    enabled: true
  policyController:
    enabled: true
  searchCollector:
    enabled: true
```

**注: policy-controller** アドオンは、**governance-policy-framework** と **config-policy-controller** の 2 つのアドオンに分かれています。その結果、**policyController** は **governance-policy-framework** と **config-policy-controller managedClusterAddons** を制御します。

2. ファイルは **klusterlet-addon-config.yaml** として保存します。

3. ハブクラスターで次のコマンドを実行して、YAML を適用します。

```
oc apply -f klusterlet-addon-config.yaml
```

4. **KlusterletAddonConfig** の作成後に、有効な **managedClusterAddons** が作成されたかどうかを確認するには、次のコマンドを実行します。

```
oc get managedclusteraddons -n <cluster namespace>
```