Red Hat Advanced Cluster Management for Kubernetes 2.7

애드온

클러스터에 애드온을 사용하는 방법을 알아보려면 자세한 내용을 확인하십시오.

Last Updated: 2024-09-06

Red Hat Advanced Cluster Management for Kubernetes 2.7 애드온

클러스터에 애드온을 사용하는 방법을 알아보려면 자세한 내용을 확인하십시오.

법적 공지

초록

클러스터에 애드온을 사용하는 방법을 알아보려면 자세한 내용을 확인하십시오.

차례

1장. 애드온 개요

Red Hat Advanced Cluster Management for Kubernetes 애드온은 일부 성능 영역을 개선하고 애플리케이션 개선을 위한 기능을 추가할 수 있습니다. 다음 섹션에서는 Red Hat Advanced Cluster Management에서 사용할 수 있는 애드온에 대한 요약 정보를 제공합니다.

- Submariner 다중 클러스터 네트워킹 및 서비스 검색
- IKEvSync 영구 볼륨 복제 서비스
- 클러스터 관리를 위해 클러스터에서 klusterlet 애드온 활성화
- 기존 클러스터 애드온에서 클러스터 전체 프록시 활성화

1.1. SUBMARINER 다중 클러스터 네트워킹 및 서비스 검색

Submariner는 Kubernetes용 Red Hat Advanced Cluster Management와 함께 온-프레미스 또는 클라우드에서 두 개 이상의 관리형 클러스터 간에 직접 네트워킹 및 서비스 검색을 제공하는 오픈 소스 툴입니다. Submariner는 Multi-Cluster Services API와 호환됩니다(Kubernetes 기능 개선 사항 Proposal #1645). Submariner에 대한 자세한 내용은 Submariner 사이트를 참조하십시오.

자동화된 콘솔 배포에서 어떤 인프라 공급자를 지원하는지와 수동 배포가 필요한 인프라 공급자에 대한 자세한 내용은 Red Hat Advanced Cluster Management Support Matrix 를 참조하십시오.

Submariner 사용 방법에 대한 자세한 내용은 다음 주제를 참조하십시오.

- 연결이 끊긴 클러스터에 Submariner 배포
- Submariner 구성
- subctl 명령 유틸리티 설치
- 콘솔을 사용하여 Submariner 배포
- 수동으로 Submariner 배포
- Submariner 배포 사용자 정의
- Submariner의 서비스 검색 관리
- Submariner 설치 제거

1.1.1. 연결이 끊긴 클러스터에 Submariner 배포

연결이 끊긴 클러스터에 Submariner를 배포하면 클러스터에 대한 외부 공격의 위험을 줄임으로써 보안 문제가 발생할 수 있습니다. 연결이 끊긴 클러스터에서 Kubernetes용 Red Hat Advanced Cluster Management를 사용하여 Submariner를 배포하려면 먼저 연결이 끊긴 네트워크 환경에 설치에 설명된 단계를 완료해야 합니다.

1.1.1.1. 연결이 끊긴 클러스터에서 Submariner 구성

연결이 끊긴 네트워크 환경에 설치에 설명된 단계를 수행한 후 연결이 끊긴 클러스터의 배포를 지원하도록 설치 중에 Submariner를 구성해야 합니다. 다음 주제를 참조하십시오.

1.1.1.1.1. 로컬 레지스트리에서 이미지 미러링

연결이 끊긴 클러스터에 **Submariner**를 배포하기 전에 로컬 레지스트리에서 **Submariner** 번들 이미지를 미러링해야 합니다.

참고: Red Hat Advanced Cluster Management 2.7.2 이상을 사용하는 경우 **nettest-rhel8** 이미지도 미러링해야 합니다.

1.1.1.1.2. *catalogSource* 이름 사용자 정의

기본적으로 **submariner-addon** 은 이름이 **redhat-operators** 인 **catalogSource** 를 검색합니다. 다른 이름으로 **catalogSource** 를 사용하는 경우 **catalogSource** 의 사용자 정의 이름으로 관리 클러스터와 연결된 **SubmarinerConfig.Spec.subscriptionConfig.Source** 매개변수 값을 업데이트해야 합니다.

1.1.1.1.3. *SubmarinerConfig*에서 *airGappedDeployment* 활성화

Kubernetes 콘솔의 Red Hat Advanced Cluster Management에서 관리형 클러스터에 **submariner-addon** 을 설치할 때 Submariner가 외부 서버에 API 쿼리를 수행하지 않도록 **Disconnected** 클러스터 옵션을 선택할 수 있습니다.

API를 사용하여 Submariner를 설치하는 경우 관리형 클러스터와 연결된 **SubmarinerConfig** 에서 **airGappedDeployment** 매개변수를 **true** 로 설정해야 합니다.

1.1.2. Submariner 구성

Red Hat Advanced Cluster Management for Kubernetes는 hub 클러스터의 애드온으로 Submariner를 제공합니다. Submariner에 대한 자세한 내용은 Submariner 오픈 소스 프로젝트 문서에서 확인할 수 있습니다.

1.1.2.1. 사전 요구 사항

Submariner를 사용하기 전에 다음 사전 요구 사항이 있는지 확인하십시오.

- **cluster-admin** 권한이 있는 허브 클러스터에 액세스할 수 있는 인증 정보.
- 게이트웨이 노드 간에 IP 연결을 구성해야 합니다. 두 클러스터를 연결할 때 게이트웨이 노드에 지정된 공용 또는 개인 IP 주소를 사용하여 게이트웨이 노드에서 하나 이상의 클러스터에 액세스할 수 있어야 합니다. 자세한 내용은 Submariner NAT Traversal 을 참조하십시오.
- OVN Kubernetes를 사용하는 경우 클러스터는 Red Hat OpenShift Container Platform 버전 4.11 이상에 있어야 합니다.
- Red Hat OpenShift Container Platform 클러스터에서 OpenShift SDN CNI를 사용하는 경우 각 관리형 클러스터의 모든 노드에 있는 방화벽 구성은 두 방향 모두에서 4800/UDP를 허용해야 합니다.
- 방화벽 구성은 관리 클러스터 간에 터널을 설정하기 위해 게이트웨이 노드에서 4500/UDP 및 4490/UDP를 허용해야 합니다.
  참고: 클러스터가 Amazon Web Services, Google Cloud Platform, Microsoft Azure 또는 Red Hat OpenStack 환경에 배포되는 경우 자동으로 설정되지만 다른 환경의 클러스터와 프라이빗 클라우드를 보호하는 방화벽에 대해 수동으로 구성해야 합니다.
- **managedcluster** 이름은 RFC 1123에 정의된 DNS 라벨 표준을 따라야 하며 다음 요구사항을 충족해야 합니다.
  - 63자 이상을 포함합니다.
  - 소문자 영숫자 또는 '-'만 포함합니다.

- 영숫자 문자로 시작합니다.
- 영숫자 문자로 끝납니다.

1.1.2.2. Submariner 포트 테이블

다음 표에서 활성화해야 하는 Submariner 포트를 확인합니다.

이름	기본값	사용자 정의	선택적 또는 필수
IPsec NATT	4500/UDP	제공됨	필수 항목
VXLAN	4800/UDP	없음	필수 항목
NAT 검색 포트	4490/UDP	없음	필수 항목

사전 요구 사항에 대한 자세한 내용은 Submariner 업스트림 사전 요구 사항 설명서 를 참조하십시오.

1.1.2.3. Globalnet

Globalnet은 겹치는 CIDR이 있는 클러스터 간 연결을 지원하는 Submariner 애드온에 포함된 기능입니다. Globalnet은 클러스터 설정된 광범위한 구성이며 첫 번째 관리 클러스터가 클러스터 세트에 추가될 때 선택할 수 있습니다. Globalnet이 활성화되면 각 관리 대상 클러스터에 가상 글로벌 프라이빗 네트워크의 글로벌 CIDR이 할당됩니다. 글로벌 CIDR은 클러스터 간 통신을 지원하는 데 사용됩니다.

Submariner를 실행하는 클러스터의 CIDR이 중첩될 가능성이 있는 경우 Globalnet 활성화를 고려하십시오. 콘솔을 사용하는 경우 **ClusterAdmin** 은 클러스터 세트의 클러스터의 Submariner 애드온을 활성화할 때 **Enable Globalnet** 옵션을 선택하여 클러스터 세트의 Globalnet을 활성화할 수 있습니다. Globalnet을 활성화한 후에는 Submariner를 제거하지 않고 비활성화할 수 없습니다.

Red Hat Advanced Cluster Management API를 사용하는 경우 **ClusterAdmin** 은 < **ManagedClusterSet** >-broker 네임스페이스에 **submariner-broker** 개체를 생성하여 Globalnet을 활성화할 수 있습니다.

ClusterAdmin 역할에는 broker 네임스페이스에 이 오브젝트를 생성하는 데 필요한 권한이 있습니다. 클러스터 세트의 프록시 관리자 역할을 수행하기 위해 생성된 **ManagedClusterSetAdmin** 역할에는 필요한 권한이 없습니다. 필요한 권한을 제공하려면 **ClusterAdmin** 에서 **access-to-brokers-submariner-crd** 에 대한 역할 권한을 **ManagedClusterSetAdmin** 사용자에게 연결해야 합니다.

submariner-broker 오브젝트를 생성하려면 다음 단계를 완료합니다.

1. 다음 명령을 실행하여 < **broker-namespace** >를 검색합니다.

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

2. **submariner-broker** 라는 YAML 파일을 생성하여 Globalnet 구성을 지정하는 **submariner-broker** 오브젝트를 생성합니다. 다음 행과 유사한 콘텐츠를 YAML 파일에 추가합니다.

```
apiVersion: submariner.io/v1alpha1
kind: Broker
metadata:
  name: submariner-broker
```

```
  namespace: <broker-namespace>
spec:
  globalnetEnabled: <true-or-false>
```

broker-namespace 를 브로커 네임스페이스의 이름으로 교체합니다.

Globalnet을 활성화하려면 **true-or-false** 를 **true** 로 바꿉니다.

참고: **metadata name** 매개변수는 **submariner-broker** 여야 합니다.

3. 다음 명령을 입력하여 YAML 파일에 파일을 적용합니다.

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

Globalnet에 대한 자세한 내용은 Submariner 문서의 Globalnet 컨트롤러를 참조하십시오.

1.1.3. subctl 명령 유틸리티 설치

subctl 유틸리티는 컨테이너 이미지에 제공됩니다. **subctl** 유틸리티를 로컬로 설치하려면 다음 단계를 완료합니다.

1. 다음 명령을 입력하여 **subctl** 컨테이너 를 다운로드하고 **subctl** 바이너리의 압축된 버전을 **/tmp** 에 추출합니다.

```
oc image extract registry.redhat.io/rhacm2/subctl-rhel8:v0.14 --path="/dist/subctl-v0.14*-linux-amd64.tar.xz":/tmp/ --confirm
```

2. 다음 명령을 입력하여 **subctl** 유틸리티의 압축을 풉니다.

```
tar -C /tmp/ -xf /tmp/subctl-v0.14*-linux-amd64.tar.xz
```

3. 다음 명령을 입력하여 **subctl** 유틸리티를 설치합니다.

```
install -m744 /tmp/subctl-v0.14*/subctl-v0.14*-linux-amd64 /$HOME/.local/bin/subctl
```

1.1.3.1. subctl 명령 사용

경로에 유틸리티를 추가한 후 사용 가능한 명령에 대한 간략한 설명은 다음 표를 참조하십시오.

내보내기 서비스	지정된 서비스에 대한 **ServiceExport** 리소스를 생성하여 Submariner 배포의 다른 클러스터에서 해당 서비스를 검색할 수 있습니다.
내보내기 취소 서비스	지정된 서비스에 대한 **ServiceExport** 리소스를 제거하여 Submariner 배포의 다른 클러스터에서 해당 서비스를 검색하지 못하도록 합니다.
show	Submariner 리소스에 대한 정보를 제공합니다.
verify	Submariner가 클러스터 한 쌍에 걸쳐 구성된 경우 연결, 서비스 검색 및 기타 Submariner 기능을 확인합니다.

benchmark	Submariner 또는 단일 클러스터 내에서 활성화되는 클러스터 쌍에서 처리량과 대기 시간입니다.
diagnose	점검을 실행하여 Submariner 배포가 제대로 작동하지 않는 문제를 식별합니다.
gather	Submariner 배포 문제를 해결하는 데 도움이 되도록 클러스터에서 정보를 수집합니다.
version	**subctl** 바이너리 도구의 버전 세부 정보를 표시합니다.

subctl 유틸리티 및 해당 명령에 대한 자세한 내용은 Submariner 설명서의 **subctl** 을 참조하십시오.

1.1.4. 콘솔을 사용하여 Submariner 배포

Kubernetes용 Red Hat Advanced Cluster Management를 사용하여 Submariner를 배포하기 전에 호스팅 환경에서 클러스터를 준비해야 합니다. **SubmarinerConfig** API 또는 Kubernetes 콘솔용 Red Hat Advanced Cluster Management를 사용하여 다음 공급자에서 Red Hat OpenShift Container Platform 클러스터를 자동으로 준비할 수 있습니다.

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

참고: VMware vSphere에서는 비NSX 배포만 지원됩니다.

다른 공급자에 Submariner를 배포하려면 Submariner 배포의 지침을 따르십시오.

Red Hat Advanced Cluster Management for Kubernetes 콘솔을 사용하여 Submariner를 배포하려면 다음 단계를 완료합니다.

필수 액세스 권한: 클러스터 관리자

1. 콘솔에서 **Infrastructure** > **Clusters** 를 선택합니다.
2. *클러스터 페이지에서 클러스터 설정 탭을* 선택합니다. Submariner를 사용하여 활성화할 클러스터는 동일한 클러스터 세트에 있어야 합니다.
3. Submariner를 배포하려는 클러스터가 이미 동일한 클러스터에 설정된 경우 5 단계로 건너뜁니다.
4. Submariner를 배포하려는 클러스터가 동일한 클러스터 세트에 없는 경우 다음 단계를 완료하여 해당 클러스터에 대해 설정된 클러스터를 생성합니다.
   a. 클러스터 세트 생성을 선택합니다.
   b. 클러스터 세트의 이름을 지정하고 생성을 선택합니다.
   c. **Manage resource assignments** to assign clusters to the cluster set를 선택합니다.

d. Submariner에 연결할 관리 클러스터를 선택하여 클러스터 세트에 추가합니다.

e. 검토를 선택하여 선택한 클러스터를 보고 확인합니다.

f. 저장 을 선택하여 클러스터 세트를 저장하고 결과 클러스터 세트 페이지를 확인합니다.

5. 클러스터 세트 페이지에서 *Submariner 애드온* 탭을 선택합니다.

6. **Install Submariner Add-ons**를 선택합니다.

7. Submariner를 배포할 클러스터를 선택합니다.

8. 다음 표의 필드를 확인하고 *Install Submariner add-ons* 편집기에 필요한 정보를 입력합니다.

필드	참고
AWS 액세스 키 **ID**	AWS 클러스터를 가져올 때만 표시됩니다.
AWS Secret Access Key	AWS 클러스터를 가져올 때만 표시됩니다.
기본 도메인 리소스 그룹 이름	Azure 클러스터를 가져올 때만 표시됩니다.
클라이언트 **ID**	Azure 클러스터를 가져올 때만 표시됩니다.
클라이언트 시크릿	Azure 클러스터를 가져올 때만 표시됩니다.
서브스크립션 **ID**	Azure 클러스터를 가져올 때만 표시됩니다.
테넌트 **ID**	Azure 클러스터를 가져올 때만 표시됩니다.
Google Cloud Platform 서비스 계정 **JSON** 키	Google Cloud Platform 클러스터를 가져올 때만 표시됩니다.
인스턴스 유형	관리 클러스터에서 생성된 게이트웨이 노드의 인스턴스 유형입니다.
IPsec NAT-T 포트	IPsec NAT 트래버스 포트의 기본값은 포트 **4500** 입니다. 관리형 클러스터 환경이 VMware vSphere 인 경우 이 포트가 방화벽에서 열려 있는지 확인합니다.
게이트웨이 수	관리 클러스터에 배포할 게이트웨이 노드 수입니다. AWS, GCP, Azure 및 OpenStack 클러스터의 경우 전용 게이트웨이 노드가 배포됩니다. VWware 클러스터의 경우 기존 작업자 노드에 게이트웨이 노드로 태그가 지정됩니다. 기본값은 **1** 입니다. 값이 1보다 크면 Submariner 게이트웨이 HA(고가용성)가 자동으로 활성화됩니다.
Cable 드라이버	클러스터 간 터널을 유지하는 Submariner 게이트웨이 유선 엔진 구성 요소입니다. 기본값은 **Libreswan IPsec** 입니다.

필드	참고
연결이 끊긴 클러스터	활성화된 경우 공용 IP 확인을 위해 Submariner에게 외부 서버에 액세스하지 않도록 지시합니다.
Globalnet CIDR	클러스터 세트에서 Globalnet 구성이 선택된 경우에만 표시됩니다. 관리형 클러스터에 사용할 Globalnet CIDR입니다. 비어 있는 경우 클러스터 세트 풀에서 CIDR이 할당됩니다.

9. 편집기 끝에 있는 다음을 선택하여 다음 클러스터의 편집기로 이동하고 선택한 나머지 각 클러스터에 대해 편집기를 완료합니다.

10. 관리되는 각 클러스터의 구성을 확인합니다.

11. 선택한 관리 클러스터에 Submariner를 배포하려면 설치를 클릭합니다.
설치 및 구성을 완료하는 데 몇 분이 걸릴 수 있습니다. Submariner *추가 기능 탭의 목록에서 Submariner* 상태를 확인할 수 있습니다.

    - 연결 상태는 관리 클러스터에서 설정된 Submariner 연결 수를 나타냅니다.
    - 에이전트 상태는 Submariner가 관리 클러스터에 성공적으로 배포되었는지 여부를 나타냅니다. 콘솔은 설치 및 구성할 때까지 성능이 저하된 상태를 보고할 수 있습니다.
    - 라벨이 지정된 게이트웨이 노드는 관리 클러스터의 게이트웨이 노드 수를 나타냅니다.

이제 Submariner가 선택한 클러스터에 배포됩니다.

1.1.5. 수동으로 Submariner 배포

Kubernetes용 Red Hat Advanced Cluster Management를 사용하여 Submariner를 배포하기 전에 해당 연결을 위해 호스팅 환경에서 클러스터를 준비해야 합니다. 콘솔을 사용하여 지원되는 클러스터에 Submariner를 자동으로 배포하는 방법을 알아보려면 콘솔을 사용하여 Submariner 배포를 참조하십시오.

클러스터가 자동 Submariner 배포를 지원하지 않는 공급자에서 호스팅되는 경우 다음 섹션을 참조하여 인프라를 수동으로 준비합니다. 각 공급자에는 준비를 위한 고유한 단계가 있으므로 올바른 공급자를 선택해야 합니다.

1.1.5.1. Submariner 용 베어 메탈 준비

Submariner 배포를 위해 베어 메탈 클러스터를 준비하려면 다음 단계를 완료합니다.

1. 방화벽이 게이트웨이 노드에 대해 4500/UDP 및 4490/UDP 포트의 외부 클라이언트에 대한 인바운드/outbound 트래픽을 허용하고 로컬 클러스터 노드 내에서 인바운드/아웃 바인딩 UDP/4800 트래픽을 허용하는지 확인합니다.

2. 다음 예와 유사한 YAML 콘텐츠를 사용자 지정하고 적용합니다.

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

- **managed-cluster-namespace** 를 관리형 클러스터의 이름으로 교체합니다. **SubmarinerConfig** 의 이름은 예제에 표시된 대로 **submariner** 여야 합니다.

  이 구성에서는 작업자 노드 중 하나에 베어 메탈 클러스터의 Submariner 게이트웨이에 레이블을 지정합니다.

  기본적으로 Submariner는 IP 보안 (IPsec)을 사용하여 게이트웨이 노드의 클러스터 간 보안 터널을 설정합니다. 기본 IPsec NATT 포트를 사용하거나 구성한 다른 포트를 지정할 수 있습니다. IPsec NATT 포트를 지정하지 않고 이 절차를 실행하면 연결에 4500/UDP가 사용됩니다.

3. Submariner에서 구성한 게이트웨이 노드를 확인하고 방화벽 구성을 활성화하여 외부 트래픽에 대해 IPsec NATT(UDP/4500) 및 NatDiscovery(UDP/4490) 포트를 허용합니다.

사용자 지정 옵션에 대한 정보는 Submariner 배포 사용자 지정을 참조하십시오.

1.1.5.2. 콘솔을 사용하여 Microsoft Azure Red Hat OpenShift for Submariner 준비 (기술 프리뷰)

Microsoft Azure Red Hat OpenShift 서비스는 다양한 도구와 리소스를 결합하여 컨테이너 기반 애플리케이션을 빌드하는 프로세스를 단순화합니다. 콘솔을 사용하여 Submariner 배포를 위해 Azure Red Hat OpenShift 클러스터를 준비하려면 다음 단계를 완료하십시오.

1. PythonProof 및 CLI 확장을 다운로드합니다.
2. Azure CLI에서 다음 명령을 실행하여 확장을 설치합니다.

   ```
   az extension add --upgrade -s <path-to-extension>
   ```

   path-to-extension 을 **.w** >-< 확장 파일을 다운로드한 경로로 교체합니다.

3. 다음 명령을 실행하여 CLI 확장이 사용 중인지 확인합니다.

   ```
   az extension list
   ```

   확장 기능을 사용하는 경우 출력은 다음 예와 유사합니다.

   ```
   "experimental": false,
   "extensionType": "whl",
   "name": "aro",
   "path": "<path-to-extension>",
   "preview": true,
   "version": "1.0.x"
   ```

4. Azure CLI에서 다음 명령을 실행하여 프리뷰 기능을 등록합니다.

   ```
   az feature registration create --namespace Microsoft.RedHatOpenShift --name
   AdminKubeconfig
   ```

5. 다음 명령을 실행하여 관리자 **kubeconfig** 를 검색합니다.

   ```
   az aro get-admin-kubeconfig -g <resource group> -n <cluster resource name>
   ```

참고: **az aro** 명령은 **kubeconfig** 를 로컬 디렉터리에 저장하고 **kubeconfig** 라는 이름을 사용합니다. 이를 사용하려면 환경 변수 **KUBECONFIG** 가 파일의 경로와 일치하도록 설정합니다. 다음 예제를 참조하십시오.

```
export KUBECONFIG=<path-to-kubeconfig>
oc get nodes
```

6. Red Hat Advanced Cluster Management 콘솔에서 **Infrastructure** > **Clusters** > **Import cluster** 를 선택하여 Azure Red Hat OpenShift 클러스터를 클러스터 목록으로 가져옵니다.

7. **Kubeconfig Import** 모드를 선택하고 **Kubeconfig** 창의 **kubeconfig** 파일에서 콘텐츠를 입력합니다. 콘솔의 지침에 따라 가져오기를 완료합니다.
인프라 > 클러스터로 이동하여 Azure Red Hat OpenShift 클러스터를 성공적으로 가져왔는지 확인할 수 있습니다.

8. **Infrastructure** > **Clusters** & **gt**; Clusters로 이동하여 추가할 클러스터 세트의 이름을 선택합니다. 그런 다음 **Submariner Add-ons** 탭을 클릭합니다.

9. **Install Submariner Add-ons** 버튼을 클릭하고 Azure Red Hat OpenShift 클러스터를 대상 클러스터로 설정합니다. 콘솔의 지침에 따라 설치를 완료합니다.

10. 인프라 > 클러스터 > 클러스터 세트 > **Submariner** 애드온 으로 이동하여 Azure Red Hat OpenShift 클러스터 연결 상태가 정상 상태인지 확인합니다.

1.1.5.2.1. API를 사용하여 Microsoft Azure Red Hat OpenShift for Submariner 준비 (기술 프리뷰)

API를 사용하여 Submariner 배포를 위해 Azure Red Hat OpenShift 클러스터를 준비하려면 다음 예와 유사한 YAML 콘텐츠를 사용자 지정하고 적용하려면 다음을 수행합니다.

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

managed-cluster-namespace 를 관리형 클러스터의 이름으로 교체합니다.

SubmarinerConfig 의 이름은 예제에 표시된 대로 **submariner** 여야 합니다.

이 구성에서는 작업자 노드 중 하나에 Azure Red Hat OpenShift 클러스터의 Submariner 게이트웨이에 레이블을 지정합니다.

기본적으로 Submariner는 IP 보안 (IPsec)을 사용하여 게이트웨이 노드의 클러스터 간 보안 터널을 설정합니다. 기본 IPsec NATT 포트를 사용하거나 구성한 다른 포트를 지정할 수 있습니다. IPsec NATT 포트를 지정하지 않고 이 절차를 실행하면 포트 4500/UDP가 연결에 사용됩니다.

사용자 지정 옵션에 대한 정보는 Submariner 배포 사용자 지정을 참조하십시오.

1.1.5.3. 콘솔을 사용하여 Amazon for Submariner에 Red Hat OpenShift Service 준비 (기술 프리뷰)

AWS의 Red Hat OpenShift Service는 애플리케이션 개발 및 현대화를 위한 안정적이고 유연한 플랫폼을 제공합니다. Submariner 배포를 위해 AWS 클러스터에서 OpenShift Service를 준비하려면 다음 단계를 완료합니다.

1. 다음 명령을 실행하여 Submariner 게이트웨이를 실행할 새 노드를 만듭니다.

```
rosa create machinepool --cluster=<cluster_name> --name=sm-gw-mp --replicas=<number
of Submariner gateway > --labels='submariner.io/gateway=true'
```

2. 다음 명령을 실행하여 AWS에서 OpenShift Service에 로그인합니다.

```
rosa login
oc login <rosa-cluster-url>:6443 --username cluster-admin --password <password>
```

3. 다음 명령을 실행하여 AWS 클러스터에서 OpenShift Service의 **kubeconfig** 를 생성합니다.

```
oc config view --flatten=true > rosa_kube/kubeconfig
```

4. Red Hat Advanced Cluster Management 콘솔에서 **Infrastructure** > **Clusters** > **Import cluster** 를 선택하여 AWS의 OpenShift Service를 클러스터 목록으로 가져옵니다.

5. **Kubeconfig** Import 모드를 선택하고 **Kubeconfig** 창의 **kubeconfig** 파일에서 콘텐츠를 입력합니다. 콘솔의 지침에 따라 가져오기를 완료합니다.
Infrastructure > **Clusters** 로 이동하여 AWS 클러스터의 OpenShift Service를 성공적으로 가져왔는지 확인할 수 있습니다.

6. **Infrastructure** > **Clusters** & **gt**; Clusters로 이동하여 추가할 클러스터 세트의 이름을 선택합니다. 그런 다음 **Submariner Add-ons** 탭을 클릭합니다.

7. **Install Submariner Add-ons** 버튼을 클릭하고 AWS 클러스터에 OpenShift Service를 대상 클러스터로 설정합니다. 콘솔의 지침에 따라 설치를 완료합니다.

8. **Infrastructure** > **Clusters** > **Cluster sets** > **Submariner** 애드온 으로 이동하여 AWS 클러스터 연결 상태의 OpenShift 서비스가 정상 상태인지 확인합니다.

1.1.5.3.1. API를 사용하여 Amazon for Submariner에 Red Hat OpenShift Service 준비 (기술 프리뷰)

API를 사용하여 Submariner를 배포하기 위해 AWS 클러스터에서 OpenShift Service를 준비하려면 다음 단계를 완료하십시오.

1. 다음 명령을 실행하여 Submariner 게이트웨이를 실행할 새 노드를 만듭니다.

```
rosa create machinepool --cluster=<cluster_name> --name=sm-gw-mp --replicas=<number
of Submariner gateway > --labels='submariner.io/gateway=true'
```

2. 다음 예와 유사한 YAML 콘텐츠를 사용자 지정하고 적용합니다.

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

managed-cluster-namespace 를 관리형 클러스터의 이름으로 교체합니다.

SubmarinerConfig 의 이름은 예제에 표시된 대로 **submariner** 여야 합니다.

기본적으로 Submariner는 IP 보안 (IPsec)을 사용하여 게이트웨이 노드의 클러스터 간 보안 터널을 설정합니다. 기본 IPsec NATT 포트를 사용하거나 구성한 다른 포트를 지정할 수 있습니다. IPsec NATT 포트를 지정하지 않고 이 절차를 실행하면 포트 4500/UDP가 연결에 사용됩니다.

사용자 지정 옵션에 대한 정보는 Submariner 배포 사용자 지정을 참조하십시오.

1.1.5.4. ManagedClusterAddOn API를 사용하여 Submariner 배포

선택한 호스팅 환경을 수동으로 준비한 후 다음 단계를 완료하여 **ManagedClusterAddOn** API를 사용하여 Submariner를 배포할 수 있습니다.

1. **ManagedClusterSet** 설명서에 제공된 지침을 사용하여 hub 클러스터에 *ManagedClusterSet* 리소스를 생성합니다. **ManagedClusterSet** 의 항목이 다음 콘텐츠와 일치하는지 확인합니다.

```
apiVersion: cluster.open-cluster-management.io/v1beta2
kind: ManagedClusterSet
metadata:
  name: <managed-cluster-set-name>
```

managed-cluster-set-name 을 생성 중인 **ManagedClusterSet** 의 이름으로 교체합니다.

중요: Kubernetes 네임스페이스의 최대 문자 길이는 63자입니다. < **managed-cluster-set-name>**에 사용할 수 있는 최대 문자 길이는 CloudEvent자입니다. < **managed-cluster-set-name** >의 문자 길이가234자를 초과하면 < **managed-cluster-set-name** >이 헤드에서 차단됩니다.

ManagedClusterSet 이 생성된 후 **submariner-addon** 은 < **managed-cluster-set-name>-broker** 라는 네임스페이스를 생성하고 Submariner 브로커를 배포합니다.

2. 다음 예와 유사한 YAML 콘텐츠를 사용자 정의하고 적용하여 < **managed-cluster-set-name>-broker** 네임스페이스의 hub 클러스터에서 **Broker** 구성을 생성합니다.

```
apiVersion: submariner.io/v1alpha1
kind: Broker
metadata:
    name: submariner-broker
    namespace: <managed-cluster-set-name>-broker
    labels:
       cluster.open-cluster-management.io/backup: submariner
spec:
    globalnetEnabled: <true-or-false>
```

managed-cluster-set-name 을 관리형 클러스터의 이름으로 교체합니다.

ManagedClusterSet 에서 Submariner Globalnet을 활성화하려면 **globalnetEnabled** 의 값을 **true** 로 설정합니다.

3. 다음 명령을 실행하여 **ManagedClusterSet** 에 하나의 관리형 클러스터를 추가합니다.

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

& **lt;managed-cluster-name** >을 **ManagedClusterSet** 에 추가할 관리형 클러스터 이름으로 바꿉니다.

& **lt;managed-cluster-set-name** >을 관리 클러스터를 추가할 **ManagedClusterSet** 의 이름으로 바꿉니다.

4. 다음 예와 유사한 YAML 콘텐츠를 사용자 지정하고 적용합니다.

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

managed-cluster-namespace 를 관리 클러스터의 네임스페이스로 교체합니다.

참고**: SubmarinerConfig** 의 이름은 예제에 표시된 대로 **submariner** 여야 합니다.

5. 다음 예와 유사한 YAML 콘텐츠를 사용자 지정하고 적용하여 관리형 클러스터에 Submariner를 배포합니다.

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

managed-cluster-name 을 Submariner와 함께 사용하려는 관리형 클러스터의 이름으로 교체합니다.

ManagedClusterAddOn 사양의 **installNamespace** 필드는 Submariner를 설치하는 관리 클러스터의 네임스페이스입니다. 현재는 **submariner-operator** 네임스페이스에 Submariner가 설치되어 있어야 합니다.

ManagedClusterAddOn 이 생성된 후 **submariner-addon** 은 Submariner를 관리 클러스터의 **submariner-operator** 네임스페이스에 배포합니다. 이 **ManagedClusterAddOn** 의 상태에서 Submariner의 배포 상태를 볼 수 있습니다.

참고**: ManagedClusterAddOn** 의 이름은 **submariner** 여야 합니다.

6. Submariner를 활성화할 모든 관리형 클러스터에 대해 3단계, 4단계, 5단계를 반복합니다.

7. Submariner가 관리 클러스터에 배포된 후 다음 명령을 실행하여 잠수함 **ManagedClusterAddOn** 의 상태를 확인하여 Submariner 배포 상태를 확인할 수 있습니다.

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

managed-cluster-name 을 관리 클러스터의 이름으로 교체합니다.

Submariner **ManagedClusterAddOn** 의 상태에서 세 가지 조건은 Submariner 배포 상태를 나타냅니다.

- **SubmarinerGatewayNodesLabeled** 조건은 관리형 클러스터에 Submariner 게이트웨이 노드가 라벨이 있는지 여부를 나타냅니다.
- **SubmarinerAgentDegraded** 상태는 Submariner가 관리 클러스터에 성공적으로 배포되었는지 여부를 나타냅니다.
- **SubmarinerConnectionDegraded** 조건은 Submariner를 사용하여 관리 클러스터에서 설정된 연결 수를 나타냅니다.

1.1.6. Submariner 배포 사용자 정의

NATT(Network Address Translation-Traversal) 포트, 게이트웨이 노드 수 및 게이트웨이 노드의 인스턴스 유형을 포함하여 일부 Submariner 배포 설정을 사용자 지정할 수 있습니다. 이러한 사용자 지정은 모든 공급자에서 일관되게 유지됩니다.

1.1.6.1. NATT 포트

NATT 포트를 사용자 지정하려면 공급자 환경에 대해 다음 YAML 콘텐츠를 사용자 지정하고 적용하려면 다음을 수행합니다.

```
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** 를 관리 클러스터의 네임스페이스로 교체합니다.
- **managed-cluster-name** 을 관리 클러스터 이름으로 교체
  - AWS: 공급자 교체 **aws** 로 . < **managed-cluster-name>-aws-creds** 값은 허브 클러스터의 클러스터 네임스페이스에서 찾을 수 있는 AWS 인증 정보 시크릿 이름입니다.
  - GCP: 공급자를 **gcp** 로 바꿉니다. < **managed-cluster-name>-gcp-creds** 값은 허브 클러스터의 클러스터 네임스페이스에서 찾을 수 있는 Google Cloud Platform 인증 정보 시크릿 이름입니다.
  - OpenStack: 공급자를 **osp** 로 교체하십시오. < **managed-cluster-name>-osp-creds** 값은 hub 클러스터의 클러스터 네임스페이스에서 찾을 수 있는 Red Hat OpenStack Platform 인증 정보 시크릿 이름입니다.
  - Azure: 공급자를 **azure** 로 교체 . < **managed-cluster-name>-azure-creds** 값은 허브 클러스터의 클러스터 네임스페이스에서 찾을 수 있는 Microsoft Azure 인증 정보 시크릿 이름입니다.
- **managed-cluster-namespace** 를 관리 클러스터의 네임스페이스로 교체합니다.
- **managed-cluster-name** 을 관리 클러스터 이름으로 교체합니다. **managed-cluster-name-gcp-creds** 의 값은 hub 클러스터의 클러스터 네임스페이스에서 찾을 수 있는 Google Cloud Platform 인증 정보 시크릿 이름입니다.
- **NATTPort** 를 사용하려는 NATT 포트로 바꿉니다.

참고: **SubmarinerConfig** 의 이름은 예제에 표시된 대로 **submariner** 여야 합니다.

1.1.6.2. 게이트웨이 노드 수

게이트웨이 노드 수를 사용자 지정하려면 다음 예와 유사한 YAML 콘텐츠를 사용자 지정하고 적용하려면 다음을 수행합니다.

```
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** 를 관리 클러스터의 네임스페이스로 교체합니다.
- **managed-cluster-name** 을 관리 클러스터 이름으로 교체합니다.
  - AWS: 공급자 교체 **aws** 로 . < **managed-cluster-name>-aws-creds** 값은 허브 클러스터의 클러스터 네임스페이스에서 찾을 수 있는 AWS 인증 정보 시크릿 이름입니다.
  - GCP: 공급자를 **gcp** 로 바꿉니다. < **managed-cluster-name>-gcp-creds** 값은 허브 클러스터의 클러스터 네임스페이스에서 찾을 수 있는 Google Cloud Platform 인증 정보 시크릿 이름입니다.
  - OpenStack: 공급자를 **osp** 로 교체하십시오. < **managed-cluster-name>-osp-creds** 값은 hub 클러스터의 클러스터 네임스페이스에서 찾을 수 있는 Red Hat OpenStack Platform 인증 정보 시크릿 이름입니다.
  - Azure: 공급자를 **azure** 로 교체 . < **managed-cluster-name>-azure-creds** 값은 허브 클러스터의 클러스터 네임스페이스에서 찾을 수 있는 Microsoft Azure 인증 정보 시크릿 이름입니다.
- 게이트웨이를 사용하려는 게이트웨이 수로 바꿉니다. 값이 1보다 크면 Submariner 게이트웨이가 고가용성을 자동으로 활성화합니다.

참고: **SubmarinerConfig** 의 이름은 예제에 표시된 대로 **submariner** 여야 합니다.

1.1.6.3. 게이트웨이 노드의 인스턴스 유형

게이트웨이 노드의 인스턴스 유형을 사용자 지정하려면 다음 예와 유사한 YAML 콘텐츠를 사용자 지정하고 적용하려면 다음을 수행합니다.

```
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** 를 관리 클러스터의 네임스페이스로 교체합니다.
- **managed-cluster-name** 을 관리 클러스터 이름으로 교체합니다.
  - AWS: 공급자 교체 **aws** 로 . < **managed-cluster-name>-aws-creds** 값은 허브 클러스터의 클러스터 네임스페이스에서 찾을 수 있는 AWS 인증 정보 시크릿 이름입니다.
  - GCP: 공급자를 **gcp** 로 바꿉니다. < **managed-cluster-name>-gcp-creds** 값은 허브 클러스터의 클러스터 네임스페이스에서 찾을 수 있는 Google Cloud Platform 인증 정보 시크릿 이름입니다.
  - OpenStack: 공급자를 **osp** 로 교체하십시오. < **managed-cluster-name>-osp-creds** 값은 hub 클러스터의 클러스터 네임스페이스에서 찾을 수 있는 Red Hat OpenStack Platform 인증 정보 시크릿 이름입니다.
  - Azure: 공급자를 **azure** 로 교체 . < **managed-cluster-name>-azure-creds** 값은 허브 클러스터의 클러스터 네임스페이스에서 찾을 수 있는 Microsoft Azure 인증 정보 시크릿 이름입니다.
- 인스턴스 유형을 사용하려는 AWS 인스턴스 유형으로 교체합니다.

참고: **SubmarinerConfig** 의 이름은 예제에 표시된 대로 **submariner** 여야 합니다.

1.1.6.4. Cable 드라이버

Submariner Gateway Engine 구성 요소는 다른 클러스터에 대한 보안 터널을 생성합니다. 유선 드라이버 구성 요소는 게이트웨이 엔진 구성 요소에서 플러그형 아키텍처를 사용하여 터널을 유지 관리합니다. 배선 엔진 구성 요소의 channels **Driver** 구성에 Libreswan 또는 VXLAN 구현을 사용할 수 있습니다. 다음 예제를 참조하십시오.

```
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
```

모범 사례: 공용 네트워크에서 VXLAN 연결 드라이버를 사용하지 마십시오. VXLAN 연결 드라이버는 암호화되지 않습니다. 프라이빗 네트워크에서 불필요한 이중 암호화를 피하기 위해 VXLAN만 사용합니다. 예를 들어 일부 온프레미스 환경에서는 전용 라인 수준 하드웨어 장치를 사용하여 터널의 암호화를 처리할 수 있습니다.

1.1.7. Submariner의 서비스 검색 관리

Submariner가 관리 클러스터와 동일한 환경에 배포된 후 경로는 관리형 클러스터 세트의 클러스터 전체에서 Pod와 서비스 간의 보안 IP 라우팅을 위해 구성됩니다.

1.1.7.1. Submariner에 대한 서비스 검색 활성화

관리형 클러스터 세트의 다른 클러스터에 클러스터에서 서비스를 표시하고 검색할 수 있도록 하려면 **ServiceExport** 오브젝트를 생성해야 합니다. **ServiceExport** 오브젝트로 서비스를 내보낸 후 <service>.<**namespace>.svc.clusterset.local** 형식으로 서비스에 액세스할 수 있습니다. 여러 클러스터에서 동일한 이름의 서비스를 내보내고 동일한 네임스페이스에서 서비스를 내보내는 경우 다른 클러스터에서 단일 논리 서비스로 인식됩니다.

이 예제에서는 **default** 네임스페이스에서 **nginx** 서비스를 사용하지만 Kubernetes **ClusterIP** 서비스 또는 헤드리스 서비스를 검색할 수 있습니다.

1. 다음 명령을 입력하여 **ManagedClusterSet** 에 있는 관리형 클러스터에 **nginx** 서비스의 인스턴스를 적용합니다.

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

2. 다음 명령과 유사한 서브**ctl** 툴을 사용하여 명령을 입력하여 **ServiceExport** 항목을 생성하여 서비스를 내보냅니다.

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

service-namespace 를 서비스가 있는 네임스페이스의 이름으로 교체합니다. 이 예에서는 기본값 입니다.

service-name 을 내보낸 서비스 이름으로 교체합니다. 이 예제에서는 **nginx** 입니다.

사용 가능한 다른 플래그에 대한 자세한 내용은 Submariner 설명서에서 내보내기 를 참조하십시오.

3. 다른 관리 클러스터에서 다음 명령을 실행하여 **nginx** 서비스에 액세스할 수 있는지 확인합니다.

```
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
```

이제 **nginx** 서비스 검색이 Submariner에 대해 구성됩니다.

1.1.7.2. Submariner의 서비스 검색 비활성화

서비스를 다른 클러스터로 내보낼 수 없도록 하려면 **nginx** 의 다음 예제와 유사한 명령을 입력합니다.

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

service-namespace 를 서비스가 있는 네임스페이스의 이름으로 교체합니다.

service-name 을 내보낸 서비스 이름으로 교체합니다.

사용 가능한 다른 플래그에 대한 자세한 내용은 Submariner 설명서에서 내보내기 취소 를 참조하십시오.

클러스터에서 검색에 더 이상 서비스를 사용할 수 없습니다.

1.1.8. Submariner 설치 제거

Kubernetes 콘솔용 Red Hat Advanced Cluster Management 또는 명령줄을 사용하여 클러스터에서 Submariner 구성 요소를 설치 제거할 수 있습니다. Submariner 0.12 이전 버전의 경우 모든 데이터 플레인 구성 요소를 완전히 제거하려면 추가 단계가 필요합니다. Submariner uninstall는 멱등이므로 문제 없이 단

계를 반복할 수 있습니다.

1.1.8.1. 콘솔을 사용하여 Submariner 설치 제거

콘솔을 사용하여 클러스터에서 Submariner를 설치 제거하려면 다음 단계를 완료합니다.

1. 콘솔 탐색에서 인프라 > 클러스터를 선택하고 *클러스터 세트* 탭을 선택합니다.
2. Submariner 구성 요소를 제거할 클러스터가 포함된 클러스터 세트를 선택합니다.
3. **Submariner Add-ons** 탭을 선택하여 Submariner가 배포된 클러스터 세트의 클러스터를 확인합니다.
4. Submariner를 제거할 클러스터의 *작업* 메뉴에서 애드온 설치 제거를 선택합니다.
5. Submariner를 제거할 클러스터의 *작업* 메뉴에서 클러스터 세트 삭제 를 선택합니다.
6. Submariner를 제거할 다른 클러스터에 대해 해당 단계를 반복합니다.
   팁: 여러 클러스터를 선택하고 작업을 클릭하여 동일한 클러스터의 여러 클러스터에서 **Submariner** 애드온을 제거할 수 있습니다. **Submariner** 애드온 제거를 선택합니다.

제거할 Submariner 버전이 버전 0.12 이전인 경우 Submariner 제거를 수동으로 계속하십시오. Submariner 버전이 0.12 이상이면 Submariner가 제거됩니다.

중요: 클라우드 공급자의 추가 비용을 방지하기 위해 모든 클라우드 리소스가 클라우드 공급자에서 제거되었는지 확인합니다. 자세한 내용은 Submariner 리소스 제거 확인을 참조하십시오.

1.1.8.2. CLI를 사용하여 Submariner 설치 제거

명령줄을 사용하여 Submariner를 설치 제거하려면 다음 단계를 완료합니다.

1. 다음 명령을 실행하여 클러스터의 Submariner 배포를 제거합니다.

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

   managed-cluster-namespace 를 관리 클러스터의 네임스페이스로 교체합니다.

2. 다음 명령을 실행하여 클러스터의 클라우드 리소스를 제거합니다.

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

   managed-cluster-namespace 를 관리 클러스터의 네임스페이스로 교체합니다.

3. 다음 명령을 실행하여 브로커 세부 정보를 제거하도록 클러스터 세트를 삭제합니다.

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

   managedclusterset 를 관리 클러스터 세트의 이름으로 교체합니다.

제거할 Submariner 버전이 버전 0.12 이전인 경우 Submariner 제거를 수동으로 계속하십시오. Submariner 버전이 0.12 이상이면 Submariner가 제거됩니다.

중요: 클라우드 공급자의 추가 비용을 방지하기 위해 모든 클라우드 리소스가 클라우드 공급자에서 제거되었는지 확인합니다. 자세한 내용은 Submariner 리소스 제거 확인을 참조하십시오.

1.1.8.3. 수동으로 Submariner 설치 제거

버전 0.12 이하인 Submariner 버전을 제거할 때 Submariner 설명서의 Manual Uninstall 섹션에서 5-8 단계를 완료합니다.

해당 단계를 완료하면 Submariner 구성 요소가 클러스터에서 제거됩니다.

중요: 클라우드 공급자의 추가 비용을 방지하기 위해 모든 클라우드 리소스가 클라우드 공급자에서 제거되었는지 확인합니다. 자세한 내용은 Submariner 리소스 제거 확인을 참조하십시오.

1.1.8.4. Submariner 리소스 제거 확인

Submariner를 제거한 후 모든 Submariner 리소스가 클러스터에서 제거되었는지 확인합니다. 클러스터에 남아 있는 경우 일부 리소스는 인프라 공급자의 비용을 계속 청구합니다. 다음 단계를 완료하여 클러스터에 추가 Submariner 리소스가 없는지 확인합니다.

1. 다음 명령을 실행하여 클러스터에 남아 있는 Submariner 리소스를 나열합니다.

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

   CLUSTER_NAME 을 클러스터 이름으로 바꿉니다.

2. 다음 명령을 입력하여 목록의 리소스를 제거합니다.

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

   RESOURCE_NAME 을 삭제하려는 Submariner 리소스의 이름으로 바꿉니다.

3. 검색에서 리소스를 식별하지 못할 때까지 각 클러스터에 대해 1-2단계를 반복합니다.

Submariner 리소스는 클러스터에서 제거됩니다.

1.2. IKEVSYNC 영구 볼륨 복제 서비스

gRPCSync는 클러스터 내의 영구 볼륨을 비동기식 복제하거나 복제와 호환되지 않는 스토리지 유형이 있는 클러스터 전체에서 비동기식 복제를 가능하게 하는 Kubernetes Operator입니다. CSI(Container Storage Interface)를 사용하여 호환성 제한을 해결합니다. 사용 중인 환경에 presenceSync Operator를 배포한 후 이를 활용하여 영구 데이터의 복사본을 만들고 유지 관리할 수 있습니다. gRPCSync는 Red Hat OpenShift Container Platform 클러스터에서만 버전 4.8 이상에 있는 영구 볼륨 클레임을 복제할 수 있습니다.

중요: presenceSync는 파일 시스템의 **volumeMode** 를 사용하여 영구 볼륨 클레임 복제만 지원합니다. **volumeMode** 를 선택하지 않으면 기본값은 **Filesystem** 입니다.

- skopeoSync를 사용하여 영구 볼륨 복제
  - 관리형 클러스터에 IKEvSync 설치
  - Rsync 복제 구성
  - restic 백업 구성
  - Rclone 복제 구성
- 복제 이미지를 사용 가능한 영구 볼륨 클레임으로 변환

- 동기화 예약

1.2.1. skopeoSync를 사용하여 영구 볼륨 복제

지원되는 세 가지 방법을 사용하여 Rsync, restic 또는 Rclone과 같은 동기화 위치 수에 따라 IKEvSync를 사용하여 영구 볼륨을 복제할 수 있습니다.

1.2.1.1. 사전 요구 사항

클러스터에 segmentSync를 설치하기 전에 다음 요구 사항이 있어야 합니다.

- Red Hat Advanced Cluster Management 버전 2.4 이상을 실행하는 구성된 Red Hat OpenShift Container Platform 환경
- 동일한 Red Hat Advanced Cluster Management hub 클러스터에서 관리하는 구성된 두 개 이상의 클러스터
- funcSync로 구성 중인 클러스터 간 네트워크 연결입니다. 클러스터가 동일한 네트워크에 없는 경우 Submariner 다중 클러스터 네트워킹 및 서비스 검색을 구성하고 **ServiceType** 의 **ClusterIP** 값을 사용하여 클러스터를 네트워크로 설정하거나 **ServiceType** 에 **LoadBalancer** 값이 있는 로드 밸런서를 사용할 수 있습니다.
- 소스 영구 볼륨에 사용하는 스토리지 드라이버는 CSI와 호환되어야 하며 스냅샷을 지원할 수 있어야 합니다.

1.2.1.2. 관리형 클러스터에 IKEvSync 설치

IKEvSync가 한 클러스터에서 다른 클러스터의 영구 볼륨 클레임에 영구 볼륨 클레임을 복제할 수 있도록 하려면 소스 및 대상 관리 클러스터 모두에NodePolicySync를 설치해야 합니다.

grSync는 자체 네임스페이스를 생성하지 않으므로 다른 OpenShift Container Platform all-namespace Operator와 동일한 네임스페이스에 있습니다. WorkingSync의 Operator 설정을 변경하면 채널 업데이트에 대한 수동 승인으로 변경되는 경우와 같이 동일한 네임스페이스의 다른 연산자에도 영향을 미칩니다.

두 가지 방법 중 하나를 사용하여 사용자의 환경에 있는 두 클러스터에NodePolicySync를 설치할 수 있습니다. 다음 섹션에 설명된 대로 hub 클러스터의 각 관리 클러스터에 레이블을 추가하거나 수동으로 **ManagedClusterAddOn** 을 만들고 적용할 수 있습니다.

1.2.1.2.1. 라벨을 사용하여NodePolicySync 설치

레이블을 추가하여 관리 클러스터에 devolSync를 설치하려면 다음을 수행합니다.

- Red Hat Advanced Cluster Management 콘솔에서 다음 단계를 완료합니다.
  1. 허브 클러스터 콘솔의 클러스터 페이지에서 관리형 클러스터 중 하나를 선택하여 세부 정보를 확인합니다.
  2. 라벨 필드에 다음 라벨을 추가합니다.

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

     gRPCSync 서비스 pod가 관리형 클러스터에 설치되어 있습니다.
  3. 동일한 레이블을 다른 관리 클러스터에 추가합니다.

4. 각 관리 클러스터에서 다음 명령을 실행하여 gRPCSync Operator가 설치되었는지 확인합니다.

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

presenceSync가 설치될 때 사용할 수 있는 연산자가 있습니다.

- 명령줄 인터페이스에서 다음 단계를 완료합니다.

1. hub 클러스터에서 명령줄 세션을 시작합니다.
2. 다음 명령을 입력하여 첫 번째 클러스터에 레이블을 추가합니다.

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

managed-cluster-1 을 관리형 클러스터 중 하나의 이름으로 교체합니다.

3. 다음 명령을 입력하여 두 번째 클러스터에 레이블을 추가합니다.

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

managed-cluster-2 를 다른 관리 클러스터의 이름으로 교체합니다.

ManagedClusterAddOn 리소스는 각각의 관리 클러스터의 네임스페이스에 허브 클러스터에서 자동으로 생성해야 합니다.

1.2.1.2.2. ManagedClusterAddOn을 사용하여 gRPCSync 설치

ManagedClusterAddOn 을 수동으로 추가하여 ProvisionSync를 관리형 클러스터에 설치하려면 다음 단계를 완료하십시오.

1. hub 클러스터에서 다음 예와 유사한 콘텐츠가 포함된 **volsync-mcao.yaml** 이라는 YAML 파일을 만듭니다.

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

managed-cluster-1-namespace 를 관리형 클러스터 중 하나의 네임스페이스로 교체합니다. 이 네임스페이스는 관리 클러스터의 이름과 동일합니다.

참고: 이름은 **volsync** 여야 합니다.

2. 다음 예와 유사한 명령을 입력하여 구성에 파일을 적용합니다.

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

3. 다른 관리 클러스터에 대해 절차를 반복합니다.
ManagedClusterAddOn 리소스는 각각의 관리 클러스터의 네임스페이스에 허브 클러스터에서 자동으로 생성해야 합니다.

1.2.1.3. Rsync 복제 구성

Rsync 복제를 사용하여 영구 볼륨의 1:1 비동기 복제를 생성할 수 있습니다. 재해 복구 또는 원격 사이트로 데이터를 보내는 데 Rsync 기반 복제를 사용할 수 있습니다.

다음 예제에서는 Rsync 메서드를 사용하여 구성하는 방법을 보여줍니다. Rsync에 대한 자세한 내용은 Usage in the Rsync documentation를 참조하십시오.

1.2.1.3.1. 관리형 클러스터 전체에서 Rsync 복제 구성

Rsync 기반 복제의 경우 소스 및 대상 클러스터에서 사용자 정의 리소스를 구성합니다. 사용자 지정 리소스는 **address** 값을 사용하여 소스를 대상에 연결하고 **sshKeys** 를 사용하여 전송된 데이터의 보안을 유지합니다.

참고: 대상에서 소스로 **address** 및 **sshKeys** 의 값을 복사해야 하므로 소스를 구성하기 전에 대상을 구성하십시오.

이 예에서는 **source-ns** 네임스페이스의 소스 클러스터의 영구 볼륨 클레임에서 **destination -ns** 네임스페이스의 대상 클러스터의 영구 볼륨 클레임으로 Rsync 복제를 구성하는 단계를 제공합니다. 필요한 경우 해당 값을 다른 값으로 교체할 수 있습니다.

1. 대상 클러스터를 구성합니다.
   a. 대상 클러스터에서 다음 명령을 실행하여 네임스페이스를 생성합니다.

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

   destination-ns 를 대상 영구 볼륨 클레임을 포함할 네임스페이스의 이름으로 교체합니다.

   b. 다음 YAML 콘텐츠를 복사하여 **replication_destination.yaml** 이라는 새 파일을 생성합니다.

   ```
   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
   ```

   참고: 용량 값은 복제 중인 영구 볼륨 클레임의 용량과 일치해야 합니다.

   대상 을 복제 대상 CR의 이름으로 바꿉니다.

   destination-ns 를 대상이 있는 네임스페이스의 이름으로 바꿉니다.

   이 예에서는 **LoadBalancer** 의 **ServiceType** 값이 사용됩니다. 로드 밸런서 서비스는 소스 관리형 클러스터가 정보를 다른 대상 관리 클러스터로 전송할 수 있도록 소스 클러스터에서 생성합니다. 소스 및 대상이 동일한 클러스터에 있거나 Submariner 네트워크 서비스가 구성된 경우 **ClusterIP** 를 서비스 유형으로 사용할 수 있습니다. 소스 클러스터를 구성할 때 참조할 시크릿의 주소와 이름을 기록해 둡니다.

storageClassName 및 **volumeSnapshotClassName** 은 선택적 매개변수입니다. 특히 환경의 기본값과 다른 스토리지 클래스 및 볼륨 스냅샷 클래스 이름을 사용하는 경우 환경 값을 지정합니다.

c. 대상 클러스터에서 다음 명령을 실행하여 **replicationdestination** 리소스를 생성합니다.

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

destination-ns 를 대상이 있는 네임스페이스의 이름으로 바꿉니다.

replicationdestination 리소스가 생성되면 다음 매개변수 및 값이 리소스에 추가됩니다.

매개변수	현재의
.status.rsync.address	소스 및 대상 클러스터가 통신할 수 있도록 하는 데 사용되는 대상 클러스터의 IP 주소입니다.
.status.rsync.sshKeys	소스 클러스터에서 대상 클러스터로 안전하게 데이터를 전송할 수 있는 SSH 키 파일의 이름입니다.

d. 다음 명령을 실행하여 소스 클러스터에서 사용할 **.status.rsync.address** 값을 복사합니다.

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

대상 을 복제 대상 사용자 정의 리소스의 이름으로 바꿉니다.

destination-ns 를 대상이 있는 네임스페이스의 이름으로 바꿉니다.

출력은 Amazon Web Services 환경에 해당하는 다음 출력과 유사해야 합니다.

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

e. 다음 명령을 실행하여 보안 이름 및 **.status.rsync.sshKeys** 값으로 제공되는 보안의 콘텐츠를 복사합니다.

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

대상 을 복제 대상 사용자 정의 리소스의 이름으로 바꿉니다.

destination-ns 를 대상이 있는 네임스페이스의 이름으로 바꿉니다.

소스를 구성할 때 소스 클러스터에 입력해야 합니다. 출력은 다음 이름과 유사할 수 있는 SSH 키 보안 파일의 이름이어야 합니다.

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

2. 복제하려는 소스 영구 볼륨 클레임을 식별합니다.

참고: 소스 영구 볼륨 클레임은 CSI 스토리지 클래스에 있어야 합니다.

3. **ReplicationSource** 항목을 생성합니다.

   a. 다음 YAML 콘텐츠를 복사하여 소스 클러스터에서 **replication_source.yaml** 이라는 새 파일을 생성합니다.

```
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 를 복제 소스 사용자 정의 리소스의 이름으로 바꿉니다. 이를 자동으로 교체하는 방법에 대한 지침은 이 절차의 *3vi* 단계를 참조하십시오.

   source-ns 를 소스가 있는 영구 볼륨 클레임의 네임스페이스로 바꿉니다. 이를 자동으로 교체하는 방법에 대한 지침은 이 절차의 *3vi* 단계를 참조하십시오.

   persistent_volume_claim 을 소스 영구 볼륨 클레임의 이름으로 교체합니다.

   mysshkeys 를 구성할 때 **ReplicationDestination** 의 **.status.rsync.sshKeys** 필드에서 복사한 키로 교체합니다.

   my.host.com 을 구성할 때 **ReplicationDestination** 의 **.status.rsync.address** 필드에서 복사한 호스트 주소로 바꿉니다.

   스토리지 드라이버에서 복제를 지원하는 경우 **Clone** 을 **copyMethod** 값으로 사용하는 것이 복제를 위한 프로세스가 더 간소화될 수 있습니다.

   storageClassName 및 **volumeSnapshotClassName** 은 선택적 매개변수입니다. 해당 환경의 기본값과 다른 스토리지 클래스 및 볼륨 스냅샷 클래스 이름을 사용하는 경우 해당 값을 지정합니다.

   이제 영구 볼륨의 동기화 방법을 설정할 수 있습니다.

   b. 대상 클러스터에 대해 다음 명령을 입력하여 대상 클러스터에서 SSH 보안을 복사합니다.

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

   destination-ns 를 대상이 있는 영구 볼륨 클레임의 네임스페이스로 바꿉니다.

   c. 다음 명령을 입력하여 **vi** 편집기에서 시크릿 파일을 엽니다.

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

d. 대상 클러스터의 열려 있는 시크릿 파일에서 다음과 같이 변경합니다.

- 네임스페이스를 소스 클러스터의 네임스페이스로 변경합니다. 이 예제에서는 **source-ns** 입니다.
- 소유자 참조(**.metadata.ownerReferences**)를 제거합니다.

e. 소스 클러스터에서 소스 클러스터에 다음 명령을 입력하여 시크릿 파일을 생성합니다.

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

f. 소스 클러스터에서 **ReplicationSource** 오브젝트의 **address** 및 **sshKeys** 값을 다음 명령을 입력하여 대상 클러스터에서 지정한 값으로 교체하여 **replication_source.yaml** 파일을 수정합니다.

```
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 을 구성할 때 **ReplicationDestination** 의 **.status.rsync.address** 필드에서 복사한 호스트 주소로 바꿉니다.

mysshkeys 를 구성할 때 **ReplicationDestination** 의 **.status.rsync.sshKeys** 필드에서 복사한 키로 교체합니다.

source 를 소스가 있는 영구 볼륨 클레임의 이름으로 바꿉니다.

참고**:** 복제하려는 영구 볼륨 클레임과 동일한 네임스페이스에 파일을 생성해야 합니다.

g. **ReplicationSource** 오브젝트에서 다음 명령을 실행하여 복제가 완료되었는지 확인합니다.

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

source-ns 를 소스가 있는 영구 볼륨 클레임의 네임스페이스로 바꿉니다.

source 를 복제 소스 사용자 정의 리소스의 이름으로 교체합니다.

복제에 성공하면 출력이 다음 예와 유사해야 합니다.

```
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 에 시간이 나열되지 않은 경우 복제가 완료되지 않습니다.

원래 영구 볼륨 클레임의 복제본이 있습니다.

1.2.1.4. restic 백업 구성

restic 기반 백업은 restic-based 백업 사본을 **restic-config.yaml** 시크릿 파일에 지정된 위치에 복사합니다. restic 백업은 클러스터 간 데이터를 동기화하지 않지만 데이터 백업을 제공합니다.

restic 기반 백업을 구성하려면 다음 단계를 완료합니다.

1. 다음 YAML 콘텐츠와 유사한 보안을 생성하여 백업 이미지가 저장된 리포지터리를 지정합니다.

```
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 를 백업 파일을 저장하려는 S3 버킷 리포지토리의 위치로 교체합니다.

my-restic-password 를 리포지토리에 액세스하는 데 필요한 암호화 키로 교체합니다.

필요한 경우 액세스 및 암호를 공급자의 인증 정보로 바꿉니다. 자세한 내용은 새 리포지토리 준비를 참조하십시오.

새 리포지토리를 준비해야 하는 경우 프로시저를 위한 새 리포지토리 준비를 참조하십시오. 해당 절차를 사용하는 경우 **restic init** 명령을 실행하여 리포지토리를 초기화해야 하는 단계를 건너뜁니다. gRPCSync는 첫 번째 백업 중에 자동으로 저장소를 초기화합니다.

중요: 동일한 S3 버킷에 여러 영구 볼륨 클레임을 백업할 때 버킷 경로는 각 영구 볼륨 클레임마다 고유해야 합니다. 각 영구 볼륨 클레임은 별도의 **ReplicationSource** 로 백업되며 각각 별도의 restic-config 시크릿이 필요합니다.

동일한 S3 버킷을 공유하면 각 **ReplicationSource** 에 전체 S3 버킷에 대한 쓰기 액세스 권한이 있습니다.

2. 다음 YAML 콘텐츠와 유사한 **ReplicationSource** 오브젝트를 생성하여 백업 정책을 구성합니다.

```
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
```

소스를 백업 중인 영구 볼륨 클레임으로 교체합니다.

일정 값을 백업을 실행하는 빈도로 바꿉니다. 이 예제에는 30분마다 일정이 있습니다. 자세한 내용은 동기화 스케줄링 을 참조하십시오.

PruneIntervalDays 값을 데이터를 다시 압축하여 공간을 절약하기 위해 경과한 일 수로 바꿉니다. 정리 작업을 수행하면 실행 중에 상당한 I/O 트래픽을 생성할 수 있습니다.

restic-config 를 1단계에서 생성한 보안 이름으로 교체합니다.

백업된 이미지의 보존 정책으로 보존 값을 설정합니다.

모범 사례**: CopyMethod** 값으로 **Clone** 을 사용하여 지정 시간 이미지가 저장되도록 합니다**.**

백업 옵션에 대한 자세한 내용은 **WorkingSync** 설명서의 백업 옵션을 참조하십시오**.**

참고**: Restic movers**는 기본적으로 **root** 권한 없이 실행됩니다**. restic movers**를 **root**로 실행하려면 다음 명령을 실행하여 승격된 권한 주석을 네임스페이스에 추가합니다**.**

```
oc annotate namespace <namespace> volsync.backube/privileged-movers=true
```

& lt;namespace >를 네임스페이스 이름으로 바꿉니다**.**

1.2.1.4.1. restic 백업 복원

restic 백업에서 복사한 데이터를 새 영구 볼륨 클레임으로 복원할 수 있습니다**.** 모범 사례**:** 하나의 백업만 새 영구 볼륨 클레임으로 복원합니다**. restic** 백업을 복원하려면 다음 단계를 완료합니다**.**

1. 다음 예와 유사한 새 데이터를 포함하도록 새 영구 볼륨 클레임을 생성합니다**.**

```
kind: PersistentVolumeClaim
```

```
apiVersion: v1
metadata:
  name: <pvc-name>
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 3Gi
```

pvc-name 을 새 영구 볼륨 클레임의 이름으로 교체합니다.

2. 다음 예제와 유사한 **ReplicationDestination** 사용자 정의 리소스를 생성하여 데이터를 복원할 위치를 지정합니다.

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

대상 을 복제 대상 **CR**의 이름으로 바꿉니다.

restic-repo 를 소스가 저장된 리포지토리 경로로 바꿉니다.

pvc-name 을 데이터를 복원하려는 새 영구 볼륨 클레임의 이름으로 교체합니다. 새 영구 볼륨 클레임을 프로비저닝하지 않고 기존 영구 볼륨 클레임을 사용합니다.

복원 프로세스는 한 번만 완료해야 하며 이 예제에서는 최신 백업을 복원합니다. 복원 옵션에 대한 자세한 내용은**torSync** 설명서의 복원 옵션을 참조하십시오.

1.2.1.5. Rclone 복제 구성

Rclone 백업은 **AWS S3**와 같은 중간 오브젝트 스토리지 위치를 통해 **Rclone**을 사용하여 단일 영구 볼륨을 여러 위치에 복사합니다. 데이터를 여러 위치에 배포할 때 유용할 수 있습니다.

Rclone 복제를 구성하려면 다음 단계를 완료합니다.

1. 다음 예와 유사한 **ReplicationSource** 사용자 정의 리소스를 만듭니다.

```
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 를 복제 소스 사용자 정의 리소스의 이름으로 교체합니다.

source-ns 를 소스가 있는 영구 볼륨 클레임의 네임스페이스로 바꿉니다.

복제 중인 영구 볼륨 클레임으로 소스를 교체합니다.

일정 값을 복제 실행 빈도로 바꿉니다. 이 예제에는 **6**분마다 일정이 있습니다. 이 값은 따옴표 내에 있어야 합니다. 자세한 내용은 동기화 스케줄링 을 참조하십시오.

intermediate-s3-bucket 을 **Rclone** 구성 파일의 구성 섹션 경로로 바꿉니다.

destination-bucket 을 복제된 파일을 복사하려는 오브젝트 버킷의 경로로 교체합니다.

rclone-secret 을 **Rclone** 구성 정보가 포함된 시크릿 이름으로 교체합니다.

copyMethod 의 값을 **Clone,Direct, Snapshot** 으로 설정합니다. 이 값은 지정 시간 복사가

생성되는지 여부를 지정하고, 그렇다면 이를 생성하는 데 사용되는 메서드를 지정합니다.

my-sc-name 을 지정 시간 복사에 사용하려는 스토리지 클래스의 이름으로 바꿉니다. 지정하지 않으면 소스 볼륨의 스토리지 클래스가 사용됩니다.

Snapshot 을 **copyMethod** 로 지정한 경우 **my-vsc** 를 사용할 **VolumeSnapshotClass** 이름으로 교체합니다. 다른 유형의 **copyMethod** 에는 이 작업이 필요하지 않습니다.

2. 다음 예와 유사한 **ReplicationDestination** 사용자 정의 리소스를 만듭니다.

```
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>
```

일정 값을 복제를 대상으로 이동하는 빈도로 바꿉니다. 데이터가 대상에서 가져오기 전에 복제를 완료할 수 있도록 소스 및 대상의 일정을 오프셋해야 합니다. 이 예제에는 **6**분마다 일정이 있으며 **3**분 간격으로 오프셋됩니다. 이 값은 따옴표 내에 있어야 합니다. 자세한 내용은 동기화 스케줄링 을 참조하십시오.

intermediate-s3-bucket 을 **Rclone** 구성 파일의 구성 섹션 경로로 바꿉니다.

destination-bucket 을 복제된 파일을 복사하려는 오브젝트 버킷의 경로로 교체합니다.

rclone-secret 을 **Rclone** 구성 정보가 포함된 시크릿 이름으로 교체합니다.

copyMethod 의 값을 **Clone,Direct, Snapshot** 으로 설정합니다. 이 값은 지정 시간 복사가

생성되는지 여부와 같은 경우 이를 생성하는 데 사용되는 메서드를 지정합니다.

accessModes 값은 영구 볼륨 클레임의 액세스 모드를 지정합니다. 유효한 값은 **ReadWriteOnce** 또는 **ReadWriteMany** 입니다.

capacity 는 대상 볼륨의 크기를 지정합니다. 이 크기는 들어오는 데이터를 포함할 수 있을 만큼 커야 합니다.

my-sc 를 지정 시간 복사의 대상으로 사용할 스토리지 클래스의 이름으로 바꿉니다. 지정하지 않으면 시스템 스토리지 클래스가 사용됩니다.

Snapshot 을 **copyMethod** 로 지정한 경우 **my-vsc** 를 사용할 **VolumeSnapshotClass** 이름으로 교체합니다. 다른 유형의 **copyMethod** 에는 이 작업이 필요하지 않습니다. 포함되지 않은 경우 시스템 기본 **VolumeSnapshotClass** 가 사용됩니다.

참고**: Rclone movers**는 기본적으로 **root** 권한 없이 실행됩니다**. Rclone** 이동기를 **root**로 실행하려면 다음 명령을 실행하여 승격된 권한 주석을 네임스페이스에 추가합니다.

```
oc annotate namespace <namespace> volsync.backube/privileged-movers=true
```

& lt;namespace >를 네임스페이스 이름으로 바꿉니다.

1.2.2. 복제 이미지를 사용 가능한 영구 볼륨 클레임으로 변환

복제된 이미지를 사용하여 데이터를 복구하거나 영구 볼륨 클레임의 새 인스턴스를 생성해야 할 수 있습니다. 이미지 사본을 사용하려면 먼저 영구 볼륨 클레임으로 변환해야 합니다. 복제된 이미지를 영구 볼륨 클레임으로 변환하려면 다음 단계를 완료합니다.

1.

복제가 완료되면 다음 명령을 입력하여 **ReplicationDestination** 오브젝트에서 최신 스냅샷을 식별합니다.

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

영구 볼륨 클레임을 생성할 때의 최신 스냅샷 값을 기록해 둡니다.

대상 을 복제 대상의 이름으로 바꿉니다.

destination-ns 를 대상의 네임스페이스로 바꿉니다.

2. 다음 예와 유사한 **pvc.yaml** 파일을 생성합니다.

```
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 을 새 영구 볼륨 클레임의 이름으로 교체합니다.

destination-ns 를 영구 볼륨 클레임이 있는 네임스페이스로 교체합니다.

snapshot_to_replace 를 이전 단계에서 찾은 **VolumeSnapshot** 이름으로 교체합니다.

모범 사례**:** 값이 초기 소스 영구 볼륨 클레임과 적어도 동일한 크기인 경우 **resources.requests.storage** 를 다른 값으로 업데이트할 수 있습니다.

3. 다음 명령을 입력하여 영구 볼륨 클레임이 환경에서 실행 중인지 확인합니다.

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

원래 백업 이미지가 기본 영구 볼륨 클레임으로 실행되고 있습니다.

1.2.3. 동기화 예약

복제 시작 방법(항상 실행, 일정으로 또는 수동으로)을 결정할 때 세 가지 옵션 중에서 선택합니다. 복제 예약은 종종 선택되는 옵션입니다.

Schedule 옵션은 예약된 시간에 복제를 실행합니다. 스케줄은 **cronspec** 에 의해 정의되므로 일정은 시간 간격 또는 특정 시간으로 구성할 수 있습니다. 스케줄 값의 순서는 다음과 같습니다.

"분 **(0-59)** 시간 **(0-23)** 월 **(1-31)** 개월 **(1-12)** 일 주 **(0-6)**"

예약된 시간이 발생하면 복제가 시작됩니다. 이 복제 옵션에 대한 설정은 다음 내용과 유사합니다.

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

이러한 방법 중 하나를 활성화하면 구성한 메서드에 따라 동기화 일정이 실행됩니다.

자세한 내용 및 옵션은 **See Sync** 설명서를 참조하십시오.

1.3. 클러스터 관리를 위해 클러스터에서 KLUSTERLET 애드온 활성화

Kubernetes용 **Red Hat Advanced Cluster Management**를 설치한 후 다중 클러스터 엔진 **Operator**로 클러스터를 생성하거나 가져온 후 해당 관리 클러스터에 대해 **klusterlet** 애드온을 활성화할 수 있습니다. **Red Hat Advanced Cluster Management** 콘솔을 사용하여 생성하거나 가져오지 않는 한 **klusterlet** 애드온은 클러스터를 생성하거나 가져온 경우 기본적으로 활성화되지 않습니다. 다음과 같은 **klusterlet** 추가 기능을 참조하십시오.

- **application-manager**
- **cert-policy-controller**
- **config-policy-controller**

- **iam-policy-controller**
- **governance-policy-framework**
- **search-collector**

Red Hat Advanced Cluster Management가 설치된 후 관리 클러스터에 대해 **klusterlet** 애드온을 활성화하려면 다음 단계를 완료합니다**.**

1. 애드온을 나타내는 **spec** 값을 사용하여 다음 **KlusterletAddonConfig** 와 유사한 **YAML** 파일을 생성합니다**.**

```
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
```

참고**: policy-controller** 애드온은 두 개의 애드온**,** 즉 **governance-policy-framework** 및 **config-policy-controller** 로 나뉩니다**.** 결과적으로 **policyController** 는 **governance-policy-framework** 및 **config-policy-controller managedClusterAddons** 를 제어합니다**.**

2. 파일을 **klusterlet-addon-config.yaml** 로 저장합니다**.**

3. **hub** 클러스터에서 다음 명령을 실행하여 **YAML**을 적용합니다**.**

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

4.

KlusterletAddonConfig 가 생성된 후 활성화된 **managedClusterAddons** 가 생성되었는지 확인하려면 다음 명령을 실행합니다.

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

1.4. 기존 클러스터 애드온에서 클러스터 전체 프록시 활성화

클러스터 네임스페이스에서 **KlusterletAddonConfig** 를 구성하여 관리 **Red Hat OpenShift Container Platform** 클러스터의 모든 **klusterlet add-on Pod**에 프록시 환경 변수를 추가할 수 있습니다. **klusterletAddonConfig**의 **Pod**에 세 개의 환경 변수를 추가하도록 **KlusterletAddonConfig** 를 구성하려면 다음 단계를 완료합니다.

1. 프록시가 필요한 클러스터의 네임스페이스에 있는 **KlusterletAddonConfig** 파일을 편집합니다. 콘솔을 사용하여 리소스를 검색하거나 터미널에서 다음 명령을 사용하여 편집할 수 있습니다.

```
oc -n <my-cluster-name> edit klusterletaddonconfig <my-cluster-name>
```

참고**:** 하나의 클러스터로만 작업하는 경우 명령 끝에 **< my-cluster-name >**이 필요하지 않습니다. 다음 명령을 참조하십시오.

```
oc -n <my-cluster-name> edit klusterletaddonconfig
```

2. 다음 예와 같이 파일의 **.spec.proxyConfig** 섹션을 편집합니다. **spec.proxyConfig** 는 선택적 섹션입니다.

```
spec
  proxyConfig:
    httpProxy: "<proxy_not_secure>"
    httpsProxy: "<proxy_secure>"
    noProxy: "<no_proxy>"
```

proxy_not_secure 를 **http** 요청에 대해 프록시 서버의 주소로 바꿉니다. 예를 들어 **http://192.168.123.145:3128** 을 사용합니다.

proxy_secure 를 **https** 요청에 대해 프록시 서버의 주소로 바꿉니다. 예를 들어 **https://192.168.123.145:3128** 을 사용합니다.

no_proxy 를 프록시를 통해 트래픽을 라우팅하지 않는 **IP** 주소**,** 호스트 이름 및 도메인 이름 목록으로 바꿉니다. 예를 들어 **.cluster.local,.svc,10.128.0.0/14,example.com** 을 사용합니다.

허브 클러스터에 구성된 클러스터 전체 프록시를 사용하여 **OpenShift Container Platform** 클러스터를 생성하는 경우 다음 조건이 충족될 때 **klusterlet add-ons**의 **Pod**에 클러스터 전체 프록시 구성 값이 추가됩니다.

- 애드온 섹션의 **.spec.policyController.proxyPolicy** 가 활성화되고 **OCPGlobalProxy** 로 설정됩니다.

- **.spec.applicationManager.proxyPolocy** 가 활성화되고 **CustomProxy** 로 설정됩니다.

  참고**:** 애드온 섹션의 **proxyPolicy** 기본값은 **Disabled** 입니다.

  proxyPolicy 항목의 다음 예제를 참조하십시오.

```
apiVersion: agent.open-cluster-management.io/v1
  kind: KlusterletAddonConfig
  metadata:
    name: clusterName
    namespace: clusterName
  spec:
    proxyConfig:
      httpProxy: http://pxuser:12345@10.0.81.15:3128
      httpsProxy: http://pxuser:12345@10.0.81.15:3128
      noProxy: .cluster.local,.svc,10.128.0.0/14, example.com
    applicationManager:
      enabled: true
      proxyPolicy: CustomProxy
    policyController:
      enabled: true
      proxyPolicy: OCPGlobalProxy
    searchCollector:
      enabled: true
      proxyPolicy: Disabled
    certPolicyController:
      enabled: true
      proxyPolicy: Disabled
    iamPolicyController:
      enabled: true
      proxyPolicy: Disabled
```

중요**:** 글로벌 프록시 설정은 경고 전달에 영향을 미치지 않습니다. 클러스터 전체 프록시를 사용하여 **Red Hat Advanced Cluster Management hub** 클러스터에 대한 경고 전달을 설정하려면 자세한 내용은 경고 전달을 참조하십시오.