Red Hat Advanced Cluster Management for Kubernetes 2.5

애플리케이션

Git 리포지토리, Helm 리포지토리 및 오브젝트 스토리지 리포지토리를 사용하여 애플리케이션을 생성하는 방법을 알아보려면 자세한 내용을 확인하십시오.

Last Updated: 2023-07-14

Red Hat Advanced Cluster Management for Kubernetes 2.5 애플리케이션

Git 리포지토리, Helm 리포지토리 및 오브젝트 스토리지 리포지토리를 사용하여 애플리케이션을 생성하는 방법을 알아보려면 자세한 내용을 확인하십시오.

법적 공지

초록

Git 리포지토리, Helm 리포지토리 및 오브젝트 스토리지 리포지토리를 사용하여 애플리케이션을 생성하는 방법을 알아보려면 자세한 내용을 확인하십시오.

차례

1장. 애플리케이션 관리

다음 주제를 검토하여 애플리케이션 생성, 배포 및 관리에 대해 자세히 알아보십시오. 이 가이드에서는 Kubernetes 개념 및 용어에 대해 숙지하고 있습니다. 주요 Kubernetes 용어 및 구성 요소는 정의되지 않습니다. Kubernetes 개념에 대한 자세한 내용은 Kubernetes 설명서를 참조하십시오.

애플리케이션 관리 기능은 애플리케이션 및 애플리케이션 업데이트를 구성하고 배포하기 위한 통합 및 단순화 옵션을 제공합니다. 이러한 기능을 통해 개발자와 DevOps 인력은 채널 및 서브스크립션 기반 자동화를 통해 환경 전반에 걸쳐 애플리케이션을 생성하고 관리할 수 있습니다.

중요: 애플리케이션 이름은 37자를 초과할 수 없습니다.

다음 주제를 참조하십시오.

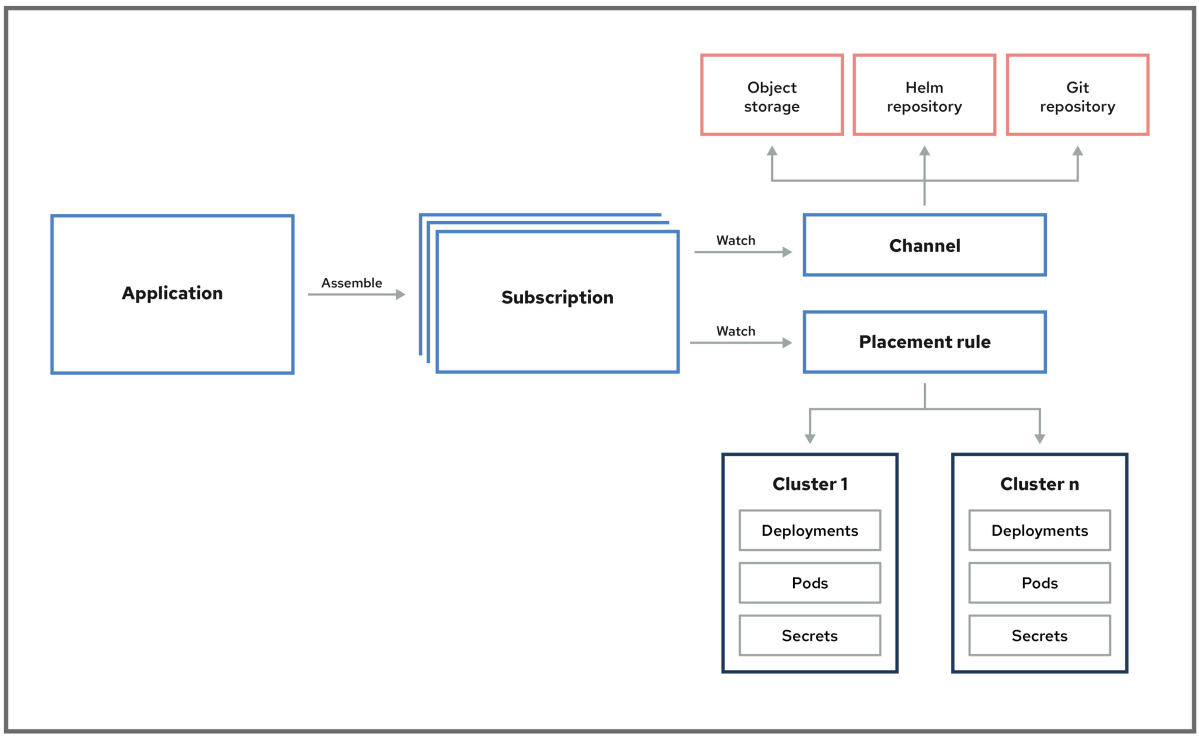
- 애플리케이션 모델 및 정의
- 애플리케이션 콘솔
- 서브스크립션 보고서
- 애플리케이션 리소스 관리
- Git 리포지토리를 사용하여 앱 관리
- Helm 리포지터리를 사용하여 앱 관리
- 오브젝트 스토리지 리포지토리를 사용하여 앱 관리
- 애플리케이션 고급 구성
- Git 리소스 구독
- 서브스크립션 관리자 권한 부여
- 서브스크립션 관리자로 허용 및 거부 목록 생성
- 조정 옵션 추가
- 보안 Git 연결에 대한 애플리케이션 채널 및 서브스크립션 구성
- Ansible Tower 작업 설정
- 관리형 클러스터에서 GitOps 구성
- 배포 예약
- 패키지 덮어쓰기 구성
- 채널 샘플
- 서브스크립션 샘플
- 배치 규칙 샘플
- 애플리케이션 샘플

1.1. 애플리케이션 모델 및 정의

애플리케이션 모델은 관리 클러스터에 배포된 리소스가 포함된 하나 이상의 Kubernetes 리소스 리포지토리(*채널* 리소스)에 가입하는 것을 기반으로 합니다. 단일 클러스터 애플리케이션 및 다중 클러스터 애플리케이션 모두 동일한 Kubernetes 사양을 사용하지만 멀티 클러스터 애플리케이션은 배포 및 애플리케이션 관리 라이프사이클의 자동화가 향상됩니다.

애플리케이션 모델에 대한 자세한 내용은 다음 이미지를 참조하십시오.

APPLICATION SUBSCRIPTION MODEL



다음 애플리케이션 리소스 섹션을 확인합니다.

- 애플리케이션
- 서브스크립션
- ApplicationSet
- 애플리케이션 문서

1.1.1. 애플리케이션

Kubernetes용 Red Hat Advanced Cluster Management의 애플리케이션(**application.app.k8s.io**)은 애플리케이션을 구성하는 Kubernetes 리소스를 그룹화하는 데 사용됩니다.

Kubernetes 애플리케이션용 Red Hat Advanced Cluster Management의 모든 애플리케이션 구성 요소 리소스는 YAML 파일 사양 섹션에서 정의합니다. 애플리케이션 구성 요소 리소스를 생성하거나 업데이트해야 하는 경우 리소스를 정의하는 레이블을 포함하도록 적절한 섹션을 생성하거나 편집해야 합니다.

검색된 애플리케이션(OpenShift Container Platform GitOps에서 검색한 애플리케이션 또는 클러스터에 설치된 Argo CD Operator)을 사용할 수도 있습니다. 동일한 리포지토리를 공유하는 애플리케이션은 이 보기에서 함께 그룹화됩니다.

1.1.2. 서브스크립션

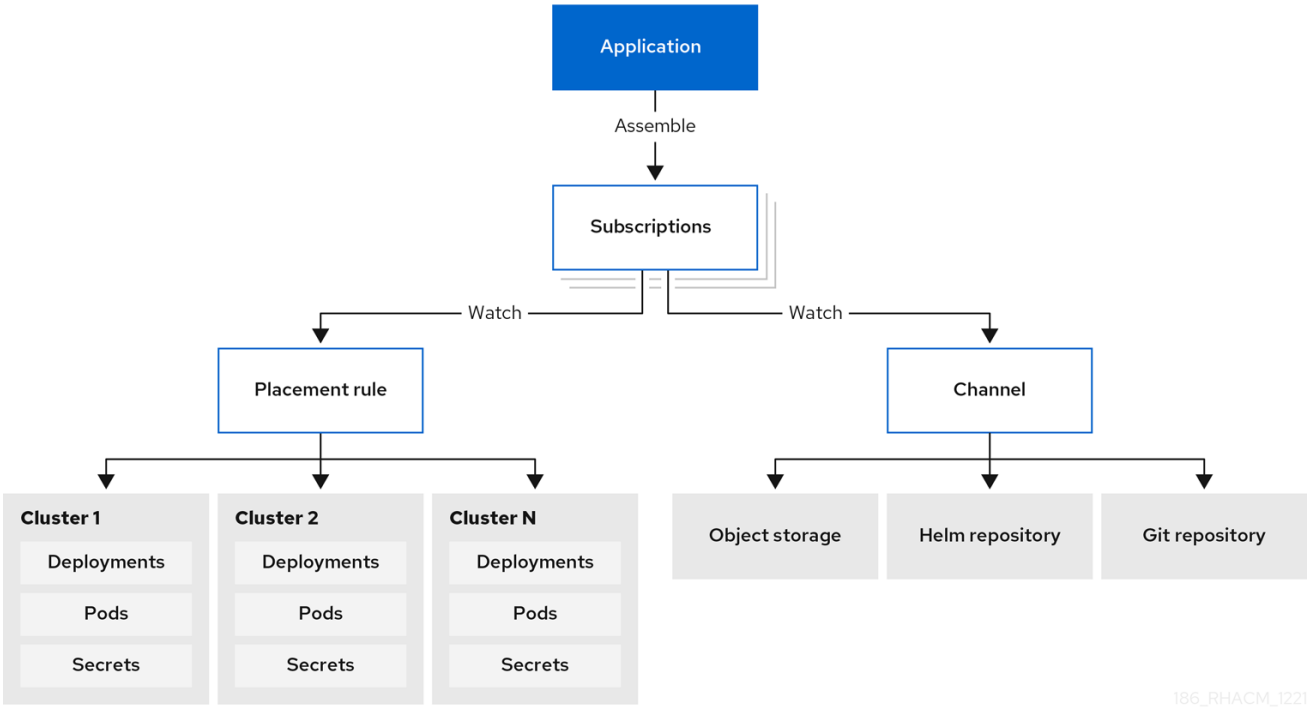
서브스크립션(**subscription.apps.open-cluster-management.io**)을 사용하면 클러스터에서 Git 리포지토리, Helm 릴리스 레지스트리 또는 오브젝트 스토리지 리포지토리의 유형일 수 있는 소스 리포지토리(채널)를 구독할 수 있습니다.

참고: 리소스가 hub 클러스터에 영향을 미칠 수 있으므로 hub 클러스터를 자체적으로 관리하는 것은 권장되지 않습니다.

허브 클러스터가 자체 관리되는 경우 서브스크립션을 통해 hub 클러스터에 애플리케이션 리소스를 로컬로 배포할 수 있습니다. 그런 다음 토폴로지에서 **로컬 클러스터** (자체 관리 허브 클러스터) 서브스크립션을 볼 수 있습니다. 리소스 요구 사항은 허브 클러스터 성능에 부정적인 영향을 미칠 수 있습니다.

서브스크립션은 새 리소스 템플릿 또는 업데이트된 리소스 템플릿을 식별하기 위한 채널 또는 스토리지 위치를 가리킬 수 있습니다. 서브스크립션 운영자는 스토리지 위치에서 직접 다운로드하여 허브 클러스터를 먼저 확인하지 않고도 대상 관리 클러스터에 배포할 수 있습니다. 서브스크립션을 통해 서브스크립션 운영자는 hub 클러스터 대신 새 리소스 또는 업데이트된 리소스에 대한 채널을 모니터링할 수 있습니다.

다음 서브스크립션 아키텍처 이미지를 참조하십시오.



1.1.2.1. 채널

채널(**channel.apps.open-cluster-management.io**)은 클러스터가 서브스크립션으로 구독할 수 있는 소스 리포지토리를 정의하고 hub 클러스터에서 Git, Helm 릴리스, 오브젝트 스토리지 리포지토리 및 리소스 템플릿과 같은 유형일 수 있습니다.

권한이 필요한 채널에 Kubernetes 리소스 또는 Helm 차트가 필요한 애플리케이션이 있는 경우(예: 권한이 부여된 Git 리포지토리) 보안을 사용하여 이러한 채널에 대한 액세스 권한을 제공할 수 있습니다. 서브스크립션은 데이터 보안을 유지하면서 이러한 채널에서 배포하기 위해 Kubernetes 리소스 및 Helm 차트에 액세스할 수 있습니다.

채널은 hub 클러스터 내의 네임스페이스를 사용하여 배포를 위해 리소스가 저장된 물리적 위치를 가리킵니다. 클러스터는 채널을 구독하여 각 클러스터에 배포할 리소스를 식별할 수 있습니다.

참고: 고유한 네임스페이스에 각 채널을 생성하는 것이 좋습니다. 그러나 Git 채널에서는 Git, Helm, 오브젝트 스토리지를 포함하여 다른 유형의 채널과 네임스페이스를 공유할 수 있습니다.

채널 내의 리소스는 해당 채널을 구독하는 클러스터에서만 액세스할 수 있습니다.

1.1.2.1.1. 지원되는 Git 리포지토리 서버

- GitHub
- GitLab
- Bitbucket
- Gogs

1.1.2.2. 배치 규칙

배치 규칙(**placementrule.apps.open-cluster-management.io**)은 리소스 템플릿을 배포할 수 있는 대상 클러스터를 정의합니다. 배치 규칙을 사용하면 배포 가능한 다중 클러스터를 쉽게 배포할 수 있습니다. 배치 규칙은 거버넌스 및 위험 정책에도 사용됩니다. 방법에 대한 자세한 내용은 Governance 를 참조하십시오.

1.1.3. ApplicationSet

ApplicationSet 은 GitOps Operator에서 지원하는 Argo CD의 하위 프로젝트입니다. **ApplicationSet** 은 Argo CD 애플리케이션에 대한 다중 클러스터 지원을 추가합니다. Red Hat Advanced Cluster Management 콘솔에서 애플리케이션 세트를 생성할 수 있습니다.

참고: ApplicationSet 을 배포하기 위한 사전 요구 사항에 대한 자세한 내용은 GitOps에 관리 클러스터 등록을 참조하십시오.

OpenShift Container Platform GitOps는 Argo CD를 사용하여 클러스터 리소스를 유지 관리합니다. Argo CD는 애플리케이션의 CI/CD(연속 통합 및 연속 배포)에 사용되는 오픈 소스 선언 도구입니다. OpenShift Container Platform GitOps는 Argo CD를 컨트롤러(OpenShift Container Platform GitOps Operator)로 구현하므로 Git 리포지토리에 정의된 애플리케이션 정의 및 구성을 지속적으로 모니터링합니다. 그러면 Argo CD에서 이러한 구성의 지정된 상태를 클러스터의 라이브 상태와 비교합니다.

ApplicationSet 컨트롤러는 GitOps Operator 인스턴스를 통해 클러스터에 설치되고 클러스터 관리자 중심 시나리오 지원에서 추가 기능을 추가하여 보완합니다. **ApplicationSet** 컨트롤러는 다음 기능을 제공합니다.

- 단일 Kubernetes 매니페스트를 사용하여 GitOps Operator가 있는 여러 Kubernetes 클러스터를 대상으로 하는 기능입니다.
- 단일 Kubernetes 매니페스트를 사용하여 GitOps Operator가 있는 하나 또는 여러 Git 리포지토리에서 여러 애플리케이션을 배포하는 기능
- Argo CD의 컨텍스트에서 단일 Git 리포지토리에 정의된 여러 Argo CD 애플리케이션 리소스인 monorepo에 대한 지원 개선
- 다중 테넌트 클러스터에서 권한 있는 클러스터 관리자를 활성화할 필요 없이 개별 클러스터 테넌트가 Argo CD를 사용하여 애플리케이션을 배포할 수 있는 기능이 향상되었습니다.

ApplicationSet Operator는 클러스터 의사 결정 생성기를 활용하여 사용자 정의 리소스별 논리를 사용하는 Kubernetes 사용자 정의 리소스를 사용하여 배포할 관리 클러스터를 결정합니다. 클러스터 결정 리소스는 관리 클러스터 목록을 생성한 후 **ApplicationSet** 리소스의 템플릿 필드로 렌더링됩니다. 이 작업은

참조된 Kubernetes 리소스의 전체 형태를 알 필요가 없는 duck-typing을 사용하여 수행됩니다.

ApplicationSet 내에서 **generators.clusterDecisionResource** 값의 다음 예제를 참조하십시오.

```
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: sample-application-set
  namespace: sample-gitops-namespace
spec:
  generators:
    - clusterDecisionResource:
        configMapRef: acm-placement
        labelSelector:
          matchLabels:
            cluster.open-cluster-management.io/placement: sample-application-placement
        requeueAfterSeconds: 180
  template:
    metadata:
      name: sample-application-{{name}}
    spec:
      project: default
      source:
        repoURL: https://github.com/sampleapp/apprepo.git
        targetRevision: main
        path: sample-application
      destination:
        namespace: sample-application
        server: "{{server}}"
      syncPolicy:
        syncOptions:
          - CreateNamespace=true
          - PruneLast=true
          - Replace=true
          - ApplyOutOfSyncOnly=true
          - Validate=false
        automated:
          prune: true
          allowEmpty: true
          selfHeal: true
```

다음 **배치**를 참조하십시오.

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: sample-application-placement
  namespace: sample-gitops-namespace
spec:
  clusterSets:
    - sampleclusterset
```

ApplicationSets 에 대한 자세한 내용은 Cluster Decision Resource Generator 를 참조하십시오.

1.1.4. 애플리케이션 문서

다음 문서에서 자세히 알아보십시오.

- 애플리케이션 콘솔
- 애플리케이션 리소스 관리
- Git 리포지토리를 사용하여 앱 관리
- Helm 리포지터리를 사용하여 앱 관리
- 오브젝트 스토리지 리포지토리를 사용하여 앱 관리
- 애플리케이션 고급 구성
- Git 리소스 구독
- Ansible Tower 작업 설정
- 채널 샘플
- 서브스크립션 샘플
- 배치 규칙 샘플
- 애플리케이션 샘플

1.2. 애플리케이션 콘솔

콘솔에는 애플리케이션 라이프사이클을 관리하기 위한 대시보드가 포함되어 있습니다. 콘솔 대시보드를 사용하여 애플리케이션을 생성 및 관리하고 애플리케이션 상태를 볼 수 있습니다. 향상된 기능은 개발자와 운영 인력이 클러스터 전체에서 애플리케이션을 생성, 배포, 업데이트, 관리 및 시각화하는 데 도움이 됩니다.

다음 목록의 콘솔 기능 중 일부를 확인하고 토폴로지를 읽는 방법에 대한 안내된 정보는 콘솔을 참조하십시오.

중요: 사용 가능한 작업은 할당된 역할을 기반으로 합니다. 역할 기반 액세스 제어 설명서에서 액세스 요구사항에 대해 알아보십시오.

- 관련 리소스 리포지토리, 서브스크립션 및 배치 구성을 포함하여 클러스터 전체에 배포된 애플리케이션을 시각화합니다.
- 애플리케이션을 생성 및 편집하고 리소스를 구독합니다. *작업* 메뉴에서 검색, 편집 또는 삭제할 수 있습니다. 필드를 업데이트할 때 **YAML:On** 을 선택하여 YAML을 보고 편집해야 합니다.
- 기본 *개요* 탭에서 애플리케이션 이름을 클릭하여 리소스 리포지토리, 서브스크립션, 배치, 배치 규칙, 배포된 리소스(Ansible Tower 작업 사용)를 사용하는 배포 후크와 같은 배포 리소스 등의 세부 정보와 애플리케이션 리소스를 볼 수 있습니다. 개요에서 애플리케이션을 생성할 수도 있습니다.
- **ApplicationSet** 및 **Subscription** 유형을 생성하고 봅니다. **ApplicationSet** 은 컨트롤러에서 생성되는 Argo 애플리케이션을 나타냅니다. ArgoCD **ApplicationSet** 을 생성하려면 **클러스터 상태가 동기화 정책에서 변경될 때 자동으로 동기화** 를 활성화해야 합니다.
- **참고: ApplicationSet** 을 생성하려면 GitOps 클러스터 리소스와 GitOps Operator가 설치되어 있어야 합니다. 이러한 사전 요구 사항이 없으면 콘솔에서 **ApplicationSet** 을 생성하기 위해 **Argo 서버** 옵션이 표시되지 않습니다.

- 기본 *개요* 에서 표의 애플리케이션 이름을 클릭하여 단일 애플리케이션 개요를 볼 때 다음 정보를 확인할 수 있습니다.
- 리소스 상태와 같은 클러스터 세부 정보
- 리소스 토폴로지
- 서브스크립션 세부 정보
- 편집하려면 편집기 탭에 액세스
- 프로젝트의 모든 애플리케이션 및 리소스를 시각적으로 표현하려면 *Topology* 탭을 클릭합니다. Helm 서브스크립션의 경우 적절한 **packageName 및 packageAlias 를 정의하여 정확한 토폴로지 디스플레이를 가져오도록 패키지 덮어쓰기** 구성을 참조하십시오.
- **고급 구성** 탭을 클릭하여 모든 애플리케이션에 대한 용어 및 리소스 테이블을 확인합니다. 리소스를 찾고 서브스크립션, 배치, 배치 규칙 및 채널을 필터링할 수 있습니다. 액세스할 수 있는 경우 편집, 검색 및 삭제와 같은 여러 **작업을** 클릭할 수도 있습니다.
- Ansible 작업을 배포된 애플리케이션의 prehook 또는 posthook로 사용하는 경우 성공적인 Ansible Tower 배포를 확인합니다.
- **Search에서 리소스 시작을** 클릭하여 관련 리소스를 검색합니다.
- *Search* 를 사용하여 각 리소스에 대한 구성 요소의 애플리케이션 리소스를 찾습니다. **리소스를 검색하려면 다음 값을 사용합니다.**

애플리케이션 리소스	종류(검색 매개변수)
서브스크립션	**서브스크립션**
채널	**채널**
Secret	**Secret**
placement	**placement**
배치 규칙	**PlacementRule**
애플리케이션	**애플리케이션**

이름, 네임스페이스, 클러스터, 레이블 등 다른 필드로 검색할 수도 있습니다. 검색 사용에 대한 자세한 내용은 콘솔에서 검색을 참조하십시오.

1.3. 서브스크립션 보고서

서브스크립션 보고서는 사용 중인 모든 관리 클러스터의 애플리케이션 상태 컬렉션입니다. 특히 상위 애플리케이션 리소스는 확장 가능한 양의 관리형 클러스터의 보고서를 보유할 수 있습니다.

자세한 애플리케이션 상태는 관리 클러스터에서 사용할 수 있는 반면 허브 클러스터의 **subscriptionReports** 는 가볍고 확장 가능합니다. 다음 세 가지 유형의 대체 상태 보고서를 참조하십시오.

- 패키지 수준 **SubscriptionStatus:** 이는 **appsub** 네임스페이스의 애플리케이션에서 배포한 모든 리소스에 대한 자세한 상태가 있는 관리 클러스터의 애플리케이션 패키지 상태입니다.
- 클러스터 수준 **SubscriptionReport:** 특정 클러스터에 배포된 모든 애플리케이션에 대한 전체 상태 보고서입니다.
- 애플리케이션 수준 **SubscriptionReport:** 특정 애플리케이션이 배포되는 모든 관리 클러스터에 대한 전반적인 상태 보고서입니다.
  - **SubscriptionStatus 패키지 수준**
  - **SubscriptionReport 클러스터 수준**
  - **SubscriptionReport 애플리케이션 수준**
  - **managedClusterView**
  - **CLI 애플리케이션 수준 상태**
  - **CLI 마지막 업데이트 시간**

1.3.1. SubscriptionStatus 패키지 수준

패키지 수준 관리형 클러스터 상태는 관리 클러스터의 **< namespace:<your-appsub-namespace >**에 있으며 애플리케이션에서 배포한 모든 리소스에 대한 세부 상태를 포함합니다. 관리형 클러스터에 배포된 모든 **appsub** 에 대해 관리 클러스터의 **appsub** 네임스페이스에 생성된 **SubscriptionStatus CR**이 있습니다. 모든 리소스는 오류가 있는 경우 자세한 오류와 함께 보고됩니다.

다음 SubscriptionStatus 샘플 YAML 파일을 참조하십시오.

```
apiVersion: apps.open-cluster-management.io/v1alpha1
kind: SubscriptionStatus
metadata:
  labels:
    apps.open-cluster-management.io/cluster: <your-managed-cluster>
    apps.open-cluster-management.io/hosting-subscription: <your-appsub-namespace>.<your-
appsub-name>
  name: <your-appsub-name>
  namespace: <your-appsub-namespace>
statuses:
  packages:
  - apiVersion: v1
    kind: Service
    lastUpdateTime: "2021-09-13T20:12:34Z"
    Message: <detailed error. visible only if the package fails>
    name: frontend
    namespace: test-ns-2
    phase: Deployed
  - apiVersion: apps/v1
    kind: Deployment
    lastUpdateTime: "2021-09-13T20:12:34Z"
    name: frontend
    namespace: test-ns-2
    phase: Deployed
  - apiVersion: v1
    kind: Service
    lastUpdateTime: "2021-09-13T20:12:34Z"
    name: redis-master
    namespace: test-ns-2
    phase: Deployed
  - apiVersion: apps/v1
    kind: Deployment
    lastUpdateTime: "2021-09-13T20:12:34Z"
    name: redis-master
    namespace: test-ns-2
    phase: Deployed
  - apiVersion: v1
    kind: Service
    lastUpdateTime: "2021-09-13T20:12:34Z"
    name: redis-slave
    namespace: test-ns-2
    phase: Deployed
  - apiVersion: apps/v1
    kind: Deployment
    lastUpdateTime: "2021-09-13T20:12:34Z"
    name: redis-slave
    namespace: test-ns-2
    phase: Deployed
```

1.3.2. SubscriptionReport 클러스터 수준

클러스터 수준 상태는 **hub** 클러스터의 **< namespace:<your-managed-cluster-1 >**에 있으며 해당 관리 클러스터의 각 애플리케이션에 대한 전체 상태만 포함합니다. **hub** 클러스터의 각 클러스터 네임스페이스의 **subscriptionReport** 는 다음 상태 중 하나를 보고합니다.

- **deployed**
- 실패
- **propagationFailed**

다음 **SubscriptionStatus** 샘플 **YAML** 파일을 참조하십시오.

```
apiVersion: apps.open-cluster-management.io/v1alpha1
kind: subscriptionReport
metadata:
  labels:
    apps.open-cluster-management.io/cluster: "true"
  name: <your-managed-cluster-1>
  namespace: <your-managed-cluster-1>
reportType: Cluster
results:
- result: deployed
  source: appsub-1-ns/appsub-1                  // appsub 1 to <your-managed-cluster-1>
  timestamp:
    nanos: 0
    seconds: 1634137362
- result: failed
  source: appsub-2-ns/appsub-2                  // appsub 2 to <your-managed-cluster-1>
  timestamp:
    nanos: 0
    seconds: 1634137362
- result: propagationFailed
  source: appsub-3-ns/appsub-3                  // appsub 3 to <your-managed-cluster-1>
  timestamp:
    nanos: 0
    seconds: 1634137362
```

1.3.3. SubscriptionReport 애플리케이션 수준

각 애플리케이션에 대한 하나의 애플리케이션 수준 **subscriptionReport** 는 **hub** 클러스터의 **appsub** 네임스페이스의 **< namespace:<your- appsub - namespace>**에 있으며 다음 정보를 포함합니다.

- 각 관리 클러스터의 애플리케이션 전체 상태
- 애플리케이션에 대한 모든 리소스 목록
- 총 클러스터 수가 포함된 보고서 요약
- 애플리케이션이 상태의 총 클러스터 수인 보고서 요약: **deployed,failed,propagationFailed** 및 **inProgress.**

참고: **inProcess** 상태는 배포된 총 빼기 , **- failed '** 및 **- 'propagationFailed**)입니다.

다음 **SubscriptionStatus** 샘플 **YAML** 파일을 참조하십시오.

```
apiVersion: apps.open-cluster-management.io/v1alpha1
kind: subscriptionReport
metadata:
  labels:
    apps.open-cluster-management.io/hosting-subscription: <your-appsub-namespace>.<your-
appsub-name>
  name: <your-appsub-name>
  namespace: <your-appsub-namespace>
reportType: Application
resources:
- apiVersion: v1
  kind: Service
  name: redis-master2
  namespace: playback-ns-2
- apiVersion: apps/v1
  kind: Deployment
  name: redis-master2
  namespace: playback-ns-2
- apiVersion: v1
  kind: Service
  name: redis-slave2
  namespace: playback-ns-2
- apiVersion: apps/v1
  kind: Deployment
  name: redis-slave2
  namespace: playback-ns-2
- apiVersion: v1
  kind: Service
  name: frontend2
  namespace: playback-ns-2
- apiVersion: apps/v1
```

```
  kind: Deployment
  name: frontend2
  namespace: playback-ns-2
results:
- result: deployed
  source: cluster-1                    //cluster 1 status
  timestamp:
    nanos: 0
    seconds: 0
- result: failed
  source: cluster-3                    //cluster 2 status
  timestamp:
    nanos: 0
    seconds: 0
- result: propagationFailed
  source: cluster-4                    //cluster 3 status
  timestamp:
    nanos: 0
    seconds: 0
summary:
  deployed: 8
  failed: 1
  inProgress: 0
  propagationFailed: 1
  clusters: 10
```

1.3.4. ManagedClusterView

ManagedClusterView CR은 첫 번째 실패한 클러스터에서 보고됩니다. 리소스 배포가 실패하는 여러 클러스터에 애플리케이션을 배포하면 허브 클러스터에서 첫 번째 실패한 클러스터 네임스페이스에 대해 하나의 **managedClusterView CR**만 생성됩니다. **managedClusterView CR**은 애플리케이션 소유자가 실패한 원격 클러스터에 액세스할 필요가 없도록 실패한 클러스터에서 자세한 서브스크립션 상태를 검색합니다.

상태를 가져오기 위해 실행할 수 있는 다음 명령을 확인합니다.

```
% oc get managedclusterview -n <failing-clusternamespace> "<app-name>-<app name>"
```

1.3.5. CLI 애플리케이션 수준 상태

관리형 클러스터에 액세스하여 서브스크립션 상태를 가져올 수 없는 경우 **CLI**를 사용할 수 있습니다. 클러스터 수준 또는 애플리케이션 수준 서브스크립션 보고서는 전체 상태를 제공하지만 애플리케이션에 대한 자세한 오류 메시지는 제공하지 않습니다.

1. multicloud-operators-subscription 에서 **CLI**를 다운로드합니다.

2. 다음 명령을 실행하여 **ManagedClusterView** 리소스를 생성하여 오류를 확인할 수 있도록 관리형 클러스터 애플리케이션 **SubscriptionStatus** 를 확인합니다.

```
% getAppSubStatus.sh -c <your-managed-cluster> -s <your-appsub-namespace> -n <your-appsub-name>
```

1.3.6. CLI 마지막 업데이트 시간

이 정보를 검색하기 위해 관리되는 각 클러스터에 로그인할 수 없는 경우 지정된 관리 클러스터에서 **AppSub**의 마지막 업데이트 시간을 가져올 수도 있습니다. 따라서 관리형 클러스터에서 **AppSub**의 마지막 업데이트 시간 검색을 단순화하기 위해 유틸리티 스크립트가 생성되었습니다. 이 스크립트는 **Hub** 클러스터에서 실행되도록 설계되었습니다. **ManagedClusterView** 리소스를 생성하여 관리 클러스터에서 **AppSub**를 가져오고 데이터를 구문 분석하여 마지막 업데이트 시간을 가져옵니다.

1. multicloud-operators-subscription 에서 **CLI**를 다운로드합니다.

2. 다음 명령을 실행하여 관리형 클러스터에서 **App Sub** 의 마지막 업데이트 시간을 검색합니다. 이 스크립트는 **hub** 클러스터에서 실행되도록 설계되었습니다. **ManagedClusterView** 리소스를 생성하여 관리 클러스터에서 **AppSub**를 가져오고 데이터를 구문 분석하여 마지막 업데이트 시간을 가져옵니다.

```
% getLastUpdateTime.sh -c <your-managed-cluster> -s <your-appsub-namespace> -n <your-appsub-name>
```

1.4. 애플리케이션 리소스 관리

콘솔에서 **Git** 리포지토리, **Helm** 리포지토리 및 오브젝트 스토리지 리포지토리를 사용하여 애플리케이션을 생성할 수 있습니다.

중요: **Git** 채널은 **Helm**, 오브젝트 스토리지 및 기타 **Git** 네임스페이스 등 다른 모든 채널 유형과 네임스페이스를 공유할 수 있습니다.

다음 주제를 참조하여 앱 관리를 시작하십시오.

- Git 리포지토리를 사용하여 앱 관리

- Helm 리포지터리를 사용하여 앱 관리
- 오브젝트 스토리지 리포지토리를 사용하여 앱 관리

1.4.1. Git 리포지토리를 사용하여 앱 관리

애플리케이션을 사용하여 Kubernetes 리소스를 배포할 때 리소스는 특정 리포지토리에 있습니다. 다음 절차에서 Git 리포지토리에서 리소스를 배포하는 방법을 알아봅니다. 애플리케이션 모델 및 정의에서 애플리케이션 모델에 대해 자세히 알아보십시오.

사용자 액세스 권한: 애플리케이션을 생성할 수 있는 사용자 역할입니다. 역할이 할당된 작업만 수행할 수 있습니다. 역할 기반 액세스 제어 설명서에서 액세스 요구 사항에 대해 알아보십시오.

1. 콘솔 탐색 메뉴에서 애플리케이션을 클릭하여 나열된 애플리케이션을 확인하고 새 애플리케이션을 생성합니다.
2. 선택 사항: 생성할 애플리케이션 유형을 선택한 후 YAML: On 을 선택하여 애플리케이션을 생성하고 편집할 콘솔에서 YAML을 볼 수 있습니다. 주제의 뒷부분에 있는 YAML 샘플을 참조하십시오.
3. 사용할 수 있는 리포지토리 목록에서 Git 을 선택하고 올바른 필드에 값을 입력합니다. 콘솔의 지침에 따라 입력에 따라 YAML 편집기 값이 변경되는지 확인합니다.

   참고:

   - 기존 Git 리포지토리 경로를 선택하는 경우 프라이빗 리포지토리인 경우 연결 정보를 지정할 필요가 없습니다. 연결 정보는 사전 설정되어 있으며 이러한 값을 볼 필요가 없습니다.
   - 새 Git 리포지토리 경로를 입력하면 선택적으로 개인 Git 리포지토리인 경우 Git 연결 정보를 입력할 수 있습니다.
   - 조정 옵션을 확인합니다. merge 옵션은 기본 선택 사항이며, 이는 새 필드가 추가되고 리소스에서 기존 필드가 업데이트됨을 의미합니다. 교체 를 선택할 수 있습니다. replace 옵션으로 기존 리소스가 Git 소스로 교체됩니다. 서브스크립션 조정 비율이 낮은 것으로 설정되면 서브스크립션된 애플리케이션 리소스를 조정하는 데 최대 1시간이 걸릴 수 있습니다. 단일

애플리케이션 보기의 카드에서 동기화 를 클릭하여 수동으로 조정합니다. **off** 로 설정하면 조정되지 않습니다.

4. 선택적 사전 배포 및 배포 후 작업을 설정합니다. 구독이 애플리케이션 리소스를 배포하기 전이나 후에 실행할 **Ansible Tower** 작업이 있는 경우 **Ansible Tower** 시크릿을 설정합니다. **Ansible** 작업을 정의하는 **Ansible Tower** 작업은 이 리포지토리의 **prehook** 및 **posthook** 폴더에 배치해야 합니다.

5. 콘솔을 사용하여 인증 정보를 추가해야 하는 경우 인증 정보 추가 를 클릭합니다. 콘솔의 지침을 따르십시오. **Manage credentials overview**에서 자세한 내용을 참조하십시오.

6. 생성을 클릭합니다.

7. 세부 사항 및 토폴로지를 볼 수 있는 *개요* 페이지로 리디렉션됩니다.

1.4.1.1. GitOps 패턴

클러스터를 관리하기 위해 **Git** 리포지토리를 조정하기 위한 모범 사례를 알아보십시오.

1.4.1.1.1. GitOps 예제 디렉터리

이 예제의 폴더는 정의되고 이름이 지정되며, 관리되는 클러스터에서 실행되는 애플리케이션 또는 구성이 포함된 각 폴더가 있습니다.

- 루트 폴더 **managed-subscriptions:** 공통 관리 폴더를 대상으로 하는 서브스크립션이 포함되어 있습니다.
- 하위 폴더 앱/: **managed-clusters** 에 배치를 사용하여 공통 관리 폴더의 애플리케이션을 구독하는 데 사용됩니다.
- 하위 폴더 **config/: managed-clusters** 에 배치를 사용하여 공통 관리 폴더의 구성을 구독하는 데 사용됩니다.
- 하위 폴더 정책/: 배치와 함께 관리 클러스터에 정책을 적용하는 데 사용됩니다.

- **folder root-subscription/: managed-subscriptions** 폴더를 서브스크립션하는 **hub** 클러스터의 초기 서브스크립션입니다.

디렉터리의 예제를 참조하십시오.

```
common-managed/
  apps/
    app-name-0/
    app-name-1/
  config/
    config001/
    config002/

managed-subscriptions
  apps/
  config/
  policies/

root-subscription/
```

1.4.1.1.2. GitOps flow

디렉터리 구조는 **root-subscription > managed-subscriptions > common-managed** 용으로 생성됩니다.

1. **root-subscription/** 의 단일 서브스크립션은 **CLI** 터미널에서 **hub** 클러스터로 적용됩니다.

2. 서브스크립션 및 정책은 **managed-subscription** 폴더의 **hub** 클러스터에 다운로드하여 적용됩니다.

   - 그런 다음 **managed-subscription** 폴더의 서브스크립션 및 정책은 배치에 따라 관리 클러스터에서 작업을 수행합니다.

   - **placement**는 각 서브스크립션 또는 정책에 영향을 미치는 관리 클러스터를 결정합니다.

   - 서브스크립션 또는 정책은 배치와 일치하는 클러스터에 대한 내용을 정의합니다.

3. 서브스크립션은 공통 관리 폴더의 내용을 배치 규칙과 일치하는 **managed-cluster** 에 적용

합니다. 이는 배치 규칙과 일치하는 모든 관리 클러스터에 공통 애플리케이션 및 구성도 적용합니다.

1.4.1.1.3. 더 많은 예

- **root-subscription/** 의 예는 **application-subscription-all** 을 참조하십시오.
- 동일한 리포지토리의 다른 폴더를 가리키는 서브스크립션의 예는 **subscribe-all** 을 참조하십시오.
- **nginx -apps** 리포지토리의 애플리케이션 아티팩트가 있는 공통 관리 폴더의 예제를 참조하십시오.
- 정책 컬렉션의 정책 예제를 참조하십시오.

1.4.2. Helm 리포지터리를 사용하여 앱 관리

애플리케이션을 사용하여 **Kubernetes** 리소스를 배포할 때 리소스는 특정 리포지토리에 있습니다. 다음 절차에서 **Helm** 리포지터리에서 리소스를 배포하는 방법을 알아봅니다. 애플리케이션 모델 및 정의에서 애플리케이션 모델에 대해 자세히 알아보십시오.

사용자 액세스 권한: 애플리케이션을 생성할 수 있는 사용자 역할입니다. 역할이 할당된 작업만 수행할 수 있습니다. 역할 기반 액세스 제어 설명서에서 액세스 요구 사항에 대해 알아보십시오.

1. 콘솔 탐색 메뉴에서 애플리케이션을 클릭하여 나열된 애플리케이션을 확인하고 새 애플리케이션을 생성합니다.
2. 선택 사항: 생성할 애플리케이션 유형을 선택한 후 **YAML: On** 을 선택하여 애플리케이션을 생성하고 편집할 콘솔에서 **YAML**을 볼 수 있습니다. 주제의 뒷부분에 있는 **YAML** 샘플을 참조하십시오.
3. 사용할 수 있는 리포지토리 목록에서 **Helm** 을 선택하고 올바른 필드에 값을 입력합니다. 콘솔의 지침에 따라 입력에 따라 **YAML** 편집기 값이 변경되는지 확인합니다.
4. 생성을 클릭합니다.

5. 세부 사항 및 토폴로지를 볼 수 있는 *개요* 페이지로 리디렉션됩니다.

1.4.2.1. YAML 샘플

다음 예제 채널 정의에서는 Helm 리포지터리를 채널로 추상화합니다.

참고: Helm의 경우 Helm 차트에 포함된 모든 Kubernetes 리소스에 레이블 릴리스가 있어야 합니다. 애플리케이션 토폴로지가 올바르게 표시되도록 하려면 {{ .release.Name }}' 입니다.

```
apiVersion: v1
kind: Namespace
metadata:
  name: hub-repo
---
apiVersion: apps.open-cluster-management.io/v1
kind: Channel
metadata:
  name: helm
  namespace: hub-repo
spec:
    pathname: [https://kubernetes-charts.storage.googleapis.com/] # URL points to a valid
chart URL.
    type: HelmRepo
```

다음 채널 정의는 Helm 리포지터리 채널의 또 다른 예를 보여줍니다.

```
apiVersion: apps.open-cluster-management.io/v1
kind: Channel
metadata:
  name: predev-ch
  namespace: ns-ch
  labels:
    app: nginx-app-details
spec:
  type: HelmRepo
  pathname: https://kubernetes-charts.storage.googleapis.com/
```

참고: REST API를 보려면 API를 사용합니다.

1.4.3. 오브젝트 스토리지 리포지토리를 사용하여 앱 관리

애플리케이션을 사용하여 Kubernetes 리소스를 배포할 때 리소스는 특정 리포지토리에 있습니다. 애플리케이션 모델 및 정의에서 애플리케이션 모델에 대해 자세히 알아보십시오.

사용자 액세스 권한: 애플리케이션을 생성할 수 있는 사용자 역할입니다. 역할이 할당된 작업만 수행할 수 있습니다. 역할 기반 액세스 제어 설명서에서 액세스 요구 사항에 대해 알아보십시오.

1. 콘솔 탐색 메뉴에서 애플리케이션을 클릭하여 나열된 애플리케이션을 확인하고 새 애플리케이션을 생성합니다.

2. 선택 사항: 생성할 애플리케이션 유형을 선택한 후 **YAML: On** 을 선택하여 애플리케이션을 생성하고 편집할 콘솔에서 **YAML**을 볼 수 있습니다. 주제의 뒷부분에 있는 **YAML** 샘플을 참조하십시오.

3. 사용할 수 있는 리포지토리 목록에서 오브젝트 저장소를 선택하고 올바른 필드에 값을 입력합니다. 콘솔의 지침에 따라 입력에 따라 **YAML** 편집기 값이 변경되는지 확인합니다.

4. 생성을 클릭합니다.

5. 세부 사항 및 토폴로지를 볼 수 있는 *개요* 페이지로 리디렉션됩니다.

1.4.3.1. YAML 샘플

다음 예제 채널 정의는 오브젝트 스토리지를 채널로 추상화합니다.

```
apiVersion: apps.open-cluster-management.io/v1
kind: Channel
metadata:
 name: dev
 namespace: ch-obj
spec:
 type: Object storage
 pathname: [http://sample-ip:#####/dev] # URL is appended with the valid bucket name, which matches the channel name.
 secretRef:
   name: miniosecret
 gates:
   annotations:
     dev-ready: true
```

참고: **REST API**를 보려면 **API**를 사용합니다.

1.4.3.2. AWS(Amazon Web Services) S3 오브젝트 스토리지 버킷 생성

서브스크립션을 설정하여 **Amazon Simple Storage Service(Amazon S3)** 오브젝트 스토리지 서비스에 정의된 리소스를 구독할 수 있습니다. 다음 절차를 참조하십시오.

1. AWS 계정, 사용자 이름 및 암호를 사용하여 AWS 콘솔에 로그인합니다.
2. **Amazon S3 > Buckets** 로 이동하여 버킷 홈 페이지로 이동합니다.
3. **Create Bucket** 을 클릭하여 버킷을 생성합니다.
4. **AWS S3** 오브젝트 버킷을 연결하는 데 필요한 **AWS** 리전 을 선택합니다.
5. 버킷 액세스 토큰을 생성합니다.
6. 탐색 모음에서 사용자 이름으로 이동한 다음 드롭다운 메뉴에서 **My Security Credentials** 를 선택합니다.
7. ***AWS IAM*** *인증 정보* 탭에서 ***CLI, SDK*** *및* ***API*** *액세스에 대한 액세스* 키로 이동하고 액세스 키 생성을 클릭합니다.
8. *액세스 키* ***ID*** *를 저장,* *시크릿 액세스 키*.
9. 오브젝트 **YAML** 파일을 버킷에 업로드합니다.

1.4.3.3. AWS 버킷의 오브젝트 구독

1. 시크릿을 사용하여 오브젝트 버킷 유형 채널을 생성하여 **AWS** 버킷 연결을 위한 ***AccessKeyID***,***SecretAccessKey*** 및 ***Region*** 을 지정합니다. 세 개의 필드는 **AWS** 버킷이 생성될 때 생성됩니다.
2. **URL**을 추가합니다. **URL**은 **URL**에 **s3 ://** 또는 **s 3** 및 **aws** 키워드가 포함된 경우 **AWS S3** 버킷에서 채널을 식별합니다. 예를 들어 다음 버킷 **URL**의 모든 **AWS s3** 버킷 식별자를 참조하십시

오.

```
https://s3.console.aws.amazon.com/s3/buckets/sample-bucket-1
s3://sample-bucket-1/
https://sample-bucket-1.s3.amazonaws.com/
```

참고: AWS S3 API와 버킷을 연결하는 데 AWS S3 오브젝트 버킷 URL이 필요하지 않습니다.

1.4.3.4. AWS 서브스크립션 샘플

다음 전체 AWS S3 오브젝트 버킷 채널 샘플 YAML 파일을 참조하십시오.

```
apiVersion: apps.open-cluster-management.io/v1
kind: Channel
metadata:
  name: object-dev
  namespace: ch-object-dev
spec:
  type: ObjectBucket
  pathname: https://s3.console.aws.amazon.com/s3/buckets/sample-bucket-1
  secretRef:
    name: secret-dev
---
apiVersion: v1
kind: Secret
metadata:
  name: secret-dev
  namespace: ch-object-dev
stringData:
  AccessKeyID: <your AWS bucket access key id>
  SecretAccessKey: <your AWS bucket secret access key>
  Region: <your AWS  bucket region>
type: Opaque
```

kind: PlacementRule 및 kind: Subscription 과 함께 다음 샘플 YAML에 표시된 대로 다른 AWS 서브스크립션 및 배치 규칙 오브젝트를 계속 생성할 수 있습니다.

```
apiVersion: apps.open-cluster-management.io/v1
kind: PlacementRule
metadata:
  name: towhichcluster
  namespace: obj-sub-ns
spec:
  clusterSelector: {}
---
apiVersion: apps.open-cluster-management.io/v1
kind: Subscription
```

```
metadata:
  name: obj-sub
  namespace: obj-sub-ns
spec:
  channel: ch-object-dev/object-dev
  placement:
    placementRef:
      kind: PlacementRule
      name: towhichcluster
```

오브젝트 버킷의 특정 하위 폴더 내의 오브젝트를 구독할 수도 있습니다. 하위 폴더 주석을 서브스크립션에 추가하여 오브젝트 버킷 서브스크립션이 하위 폴더의 모든 리소스만 적용하도록 강제 적용합니다.

subfolder-1 이 있는 주석을 **bucket-path** 로 참조하십시오.

```
annotations:
  apps.open-cluster-management.io/bucket-path: <subfolder-1>
```

하위 폴더에 대한 다음 전체 샘플을 참조하십시오.

```
apiVersion: apps.open-cluster-management.io/v1
kind: Subscription
metadata:
  annotations:
    apps.open-cluster-management.io/bucket-path: subfolder1
  name: obj-sub
  namespace: obj-sub-ns
  labels:
    name: obj-sub
spec:
  channel: ch-object-dev/object-dev
  placement:
    placementRef:
      kind: PlacementRule
      name: towhichcluster
```

1.5. 애플리케이션 고급 구성

Kubernetes용 **Red Hat Advanced Cluster Management** 내에서 애플리케이션은 여러 애플리케이션 리소스로 구성됩니다. 채널, 서브스크립션, 배치 및 배치 규칙 리소스를 사용하여 전체 애플리케이션을 배포, 업데이트 및 관리할 수 있습니다.

단일 클러스터 애플리케이션 및 다중 클러스터 애플리케이션 모두 동일한 **Kubernetes** 사양을 사용하지만 멀티 클러스터 애플리케이션은 배포 및 애플리케이션 관리 라이프사이클의 자동화가 향상됩니다.

Kubernetes 애플리케이션용 Red Hat Advanced Cluster Management의 모든 애플리케이션 구성 요소 리소스는 YAML 파일 사양 섹션에서 정의합니다. 애플리케이션 구성 요소 리소스를 생성하거나 업데이트해야 하는 경우 리소스를 정의하는 레이블을 포함하도록 적절한 섹션을 생성하거나 편집해야 합니다.

다음 애플리케이션 고급 구성 주제를 확인합니다.

- Git 리소스 구독
- 서브스크립션 관리자 권한 부여
- 서브스크립션 관리자로 허용 및 거부 목록 생성
- 조정 옵션 추가
- 보안 Git 연결에 대한 애플리케이션 채널 및 서브스크립션 구성
- Ansible Tower 작업 설정
- 관리형 클러스터에서 GitOps 구성
- 패키지 덮어쓰기 구성
- 채널 샘플 개요
- 서브스크립션 샘플 개요
- 배치 규칙 샘플 개요
- 애플리케이션 샘플 개요

1.5.1. Git 리소스 구독

기본적으로 서브스크립션이 서브스크립션된 애플리케이션을 대상 클러스터에 배포하면 애플리케이션 리소스가 다른 네임스페이스와 연결되어 있어도 애플리케이션은 해당 서브스크립션 네임스페이스에 배포됩니다. *서브스크립션 관리자*는 권한 부여 관리자 권한에 설명된 대로 기본 동작을 변경할 수 있습니다.

또한 클러스터에 애플리케이션 리소스가 있고 서브스크립션을 통해 생성되지 않은 경우 해당 기존 리소스에 새 리소스를 적용할 수 없습니다. 서브스크립션 관리자로 기본 설정을 변경하려면 다음 프로세스를 참조하십시오.

필수 액세스 권한: 클러스터 관리자

- Git에서 애플리케이션 리소스 생성
- 특정 Git 요소 가입
- 애플리케이션 네임스페이스 예
- 리소스 덮어쓰기 예

1.5.1.1. Git에서 애플리케이션 리소스 생성

구독할 때 리소스 YAML에서 apiVersion 의 전체 그룹 및 버전을 지정해야 합니다. 예를 들어 apiVersion: v1 을 구독하면 서브스크립션 컨트롤러가 서브스크립션의 유효성을 확인하지 못하고 Resource /v1, Kind=ImageStream이 지원되지 않습니다.

apiVersion 이 image.openshift.io/v1 로 변경되면 서브스크립션 컨트롤러의 유효성을 전달하고 리소스가 성공적으로 적용됩니다.

```
apiVersion: `image.openshift.io/v1`
kind: ImageStream
metadata:
  name: default
  namespace: default
spec:
  lookupPolicy:
    local: true
  tags:
```

```
  - name: 'latest'
    from:
      kind: DockerImage
      name: 'quay.io/repository/open-cluster-management/multicluster-operators-
subscription:community-latest'
```

다음으로 서브스크립션 관리자가 기본 동작을 변경하는 방법에 대한 유용한 예제를 참조하십시오.

1.5.1.2. 애플리케이션 네임스페이스 예

다음 예제에서는 서브스크립션 관리자로 로그인되어 있습니다.

1.5.1.2.1. 다른 네임스페이스로 애플리케이션

Git 리포지토리에서 샘플 리소스 YAML 파일을 서브스크립션하는 서브스크립션을 생성합니다. 예제 파일에는 다음과 같은 다른 네임스페이스 내에 있는 서브스크립션이 포함되어 있습니다.

해당 채널 유형: Git

- ConfigMap test-configmap-1 은 multins 네임스페이스에서 생성됩니다.
- ConfigMap test-configmap-2 는 기본 네임스페이스에서 생성됩니다.
- ConfigMap test-configmap-3 이 서브스크립션 네임스페이스에서 생성됩니다.

```
---
apiVersion: v1
kind: Namespace
metadata:
  name: multins
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: test-configmap-1
  namespace: multins
data:
  path: resource1
---
apiVersion: v1
kind: ConfigMap
metadata:
```

```
  name: test-configmap-2
  namespace: default
data:
  path: resource2
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: test-configmap-3
data:
  path: resource3
```

다른 사용자가 서브스크립션을 생성한 경우 서브스크립션과 동일한 네임스페이스에 모든 **ConfigMap**이 생성됩니다.

1.5.1.2.2. 동일한 네임 스페이스에 애플리케이션

서브스크립션 관리자는 모든 애플리케이션 리소스를 동일한 네임스페이스에 배포할 수 있습니다.

허용 및 거부 목록을 서브스크립션 관리자로 생성하여 모든 애플리케이션 리소스를 서브스크립션 네임스페이스에 배포할 수 있습니다.

apps.open-cluster-management.io/current-namespace-scoped: true 주석을 서브스크립션 **YAML**에 추가합니다. 예를 들어 서브스크립션 관리자가 다음 서브스크립션을 생성하면 이전 예제의 세 개의 **ConfigMap**이 모두 **subscription-ns** 네임스페이스에 생성됩니다.

```
apiVersion: apps.open-cluster-management.io/v1
kind: Subscription
metadata:
  name: subscription-example
  namespace: subscription-ns
  annotations:
    apps.open-cluster-management.io/git-path: sample-resources
    apps.open-cluster-management.io/reconcile-option: merge
    apps.open-cluster-management.io/current-namespace-scoped: "true"
spec:
  channel: channel-ns/somechannel
  placement:
    placementRef:
      name: dev-clusters
```

1.5.1.3. 리소스 덮어쓰기 예

해당 채널 유형: **Git, ObjectBucket(** 콘솔의 오브젝트 스토리지**)**

참고: helm 차트 리소스는 Helm에서 관리되므로 리소스 덮어쓰기 옵션은 Git 리포지토리의 helm 차트에 적용되지 않습니다.

이 예에서는 대상 클러스터에 다음 ConfigMap이 이미 있습니다.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: test-configmap-1
  namespace: sub-ns
data:
  name: user1
  age: 19
```

Git 리포지토리에서 다음 샘플 리소스 YAML 파일을 구독하고 기존 ConfigMap을 교체합니다. 데이터 사양의 변경 사항을 참조하십시오.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: test-configmap-1
  namespace: sub-ns
data:
  age: 20
```

1.5.1.3.1. 기본 병합 옵션

기본 apps.open-cluster-management.io/reconcile-option: 병합 주석을 사용하여 Git 리포지토리에서 다음 샘플 리소스 YAML 파일을 참조하십시오. 다음 예제를 참조하십시오.

```
apiVersion: apps.open-cluster-management.io/v1
kind: Subscription
metadata:
  name: subscription-example
  namespace: sub-ns
  annotations:
    apps.open-cluster-management.io/git-path: sample-resources
    apps.open-cluster-management.io/reconcile-option: merge
spec:
  channel: channel-ns/somechannel
  placement:
    placementRef:
      name: dev-clusters
```

서브스크립션 관리자가 이 서브스크립션을 생성하고 ConfigMap 리소스를 구독하면 다음 예제에서 볼 수 있듯이 기존 ConfigMap이 병합됩니다.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: test-configmap-1
  namespace: sub-ns
data:
  name: user1
  age: 20
```

merge 옵션을 사용하면 구독된 리소스의 항목이 기존 리소스에서 생성되거나 업데이트됩니다. 기존 리소스에서 항목이 제거되지 않습니다.

중요: 서브스크립션을 사용하여 덮어쓰려는 기존 리소스가 다른 **Operator** 또는 컨트롤러에 의해 자동으로 조정되면 서브스크립션과 컨트롤러 또는 **Operator** 둘 다에 의해 리소스 구성이 업데이트됩니다. 이 경우에는 이 방법을 사용하지 마십시오.

1.5.1.3.2. mergeAndOwn option

mergeAndOwn 을 사용하면 구독된 리소스의 항목이 기존 리소스에서 생성되거나 업데이트됩니다. 서브스크립션 관리자로 로그인하여 **apps.open-cluster-management.io/reconcile-option: mergeAndOwn** 주석을 사용하여 서브스크립션을 생성합니다. 다음 예제를 참조하십시오.

```
apiVersion: apps.open-cluster-management.io/v1
kind: Subscription
metadata:
  name: subscription-example
  namespace: sub-ns
  annotations:
    apps.open-cluster-management.io/git-path: sample-resources
    apps.open-cluster-management.io/reconcile-option: mergeAndOwn
spec:
  channel: channel-ns/somechannel
  placement:
    placementRef:
      name: dev-clusters
```

서브스크립션 관리자가 이 서브스크립션을 생성하고 **ConfigMap** 리소스를 구독하면 다음 예제에서 볼 수 있듯이 기존 **ConfigMap**이 병합됩니다.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: test-configmap-1
  namespace: sub-ns
  annotations:
```

```
    apps.open-cluster-management.io/hosting-subscription: sub-ns/subscription-example
data:
  name: user1
  age: 20
```

previosly mentioned와 같이 mergeAndOwn 옵션을 사용하면 구독된 리소스의 항목이 기존 리소스에서 생성하거나 업데이트됩니다. 기존 리소스에서 항목이 제거되지 않습니다. 또한 apps.open-cluster-management.io/hosting-subscription 주석을 추가하여 서브스크립션이 현재 리소스를 소유하고 있음을 나타냅니다. 서브스크립션을 삭제하면 ConfigMap이 삭제됩니다.

1.5.1.3.3. Replace 옵션

서브스크립션 관리자로 로그인하여 apps.open-cluster-management.io/reconcile-option: 주석을 교체 하여 서브스크립션을 생성합니다. 다음 예제를 참조하십시오.

```
apiVersion: apps.open-cluster-management.io/v1
kind: Subscription
metadata:
  name: subscription-example
  namespace: sub-ns
  annotations:
    apps.open-cluster-management.io/git-path: sample-resources
    apps.open-cluster-management.io/reconcile-option: replace
spec:
  channel: channel-ns/somechannel
  placement:
    placementRef:
      name: dev-clusters
```

이 서브스크립션을 서브스크립션 관리자가 생성하고 ConfigMap 리소스를 구독하면 기존 ConfigMap이 다음으로 교체됩니다.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: test-configmap-1
  namespace: sub-ns
data:
  age: 20
```

1.5.1.4. 특정 Git 요소 가입

특정 Git 분기, 커밋 또는 태그를 구독할 수 있습니다.

1.5.1.4.1. 특정 분기 구독

multicloud-operators-subscription 리포지토리에 포함된 서브스크립션 **Operator**는 **Git** 리포지토리의 기본 분기를 서브스크립션합니다. 다른 분기를 구독하려면 서브스크립션에 분기 이름 주석을 지정해야 합니다.

다음 예제에서는 **YAML** 파일에서 **apps.open-cluster-management.io/git-branch: <branch1 >**를 사용하여 다른 분기를 지정하는 방법을 표시합니다.

```
apiVersion: apps.open-cluster-management.io/v1
kind: Subscription
metadata:
  name: git-mongodb-subscription
  annotations:
    apps.open-cluster-management.io/git-path: stable/ibm-mongodb-dev
    apps.open-cluster-management.io/git-branch: <branch1>
```

1.5.1.4.2. 특정 커밋 구독

multicloud-operators-subscription 리포지토리에 포함된 서브스크립션 **Operator**는 기본적으로 **Git** 리포지토리의 지정된 분기의 최신 커밋을 서브스크립션합니다. 특정 커밋을 구독하려면 서브스크립션의 커밋 해시를 사용하여 원하는 커밋 주석을 지정해야 합니다.

다음 예제는 **YAML** 파일에서 **apps.open-cluster-management.io/git-desired-commit: <full commit number >**를 사용하여 다른 커밋을 지정하는 방법을 표시합니다.

```
apiVersion: apps.open-cluster-management.io/v1
kind: Subscription
metadata:
  name: git-mongodb-subscription
  annotations:
    apps.open-cluster-management.io/git-path: stable/ibm-mongodb-dev
    apps.open-cluster-management.io/git-desired-commit: <full commit number>
    apps.open-cluster-management.io/git-clone-depth: 100
```

git-clone-depth 주석은 선택 사항이며 기본적으로 **20** 으로 설정됩니다. 즉 서브스크립션 컨트롤러가 **Git** 리포지토리에서 이전 **20**개의 커밋 기록을 검색합니다. 훨씬 오래된 **git-desired-commit** 을 지정하는 경우 원하는 커밋에 따라 **git-clone-depth** 를 지정해야 합니다.

1.5.1.4.3. 특정 태그 구독

multicloud-operators-subscription 리포지토리에 포함된 서브스크립션 **Operator**는 기본적으로 **Git** 리포지토리의 지정된 분기의 최신 커밋을 서브스크립션합니다. 특정 태그를 구독하려면 서브스크립션에 태그 주석을 지정해야 합니다.

다음 예제에서는 YAML 파일에서 apps.open-cluster-management.io/git-tag: <v1.0 >을 사용하여 다른 태그를 지정하는 방법을 표시합니다.

```
apiVersion: apps.open-cluster-management.io/v1
kind: Subscription
metadata:
  name: git-mongodb-subscription
  annotations:
    apps.open-cluster-management.io/git-path: stable/ibm-mongodb-dev
    apps.open-cluster-management.io/git-tag: <v1.0>
    apps.open-cluster-management.io/git-clone-depth: 100
```

참고: Git 원하는 커밋 및 태그 주석이 모두 지정되면 태그가 무시됩니다.

git-clone-depth 주석은 선택 사항이며 기본적으로 20 으로 설정됩니다. 즉 서브스크립션 컨트롤러가 Git 리포지토리에서 이전 20 커밋 기록을 검색합니다. 이전 git-tag 를 많이 지정하는 경우 태그의 원하는 커밋에 맞게 git-clone-depth 를 지정해야 합니다.

1.5.2. 서브스크립션 관리자 권한 부여

서브스크립션 관리자 액세스 권한을 부여하는 방법에 대해 알아봅니다. *서브스크립션* 관리자는 기본 동작을 변경할 수 있습니다. 다음 프로세스에서 자세히 알아보십시오.

1. 콘솔에서 Red Hat OpenShift Container Platform 클러스터에 로그인합니다.

2. 하나 이상의 사용자를 생성합니다. 사용자 생성에 대한 자세한 내용은 사용자 준비를 참조하십시오. 그룹 또는 서비스 계정을 준비할 수도 있습니다.

   생성하는 사용자는 app.open-cluster-management.io/subscription 애플리케이션의 관리자입니다. OpenShift Container Platform을 사용하면 *서브스크립션* 관리자가 기본 동작을 변경할 수 있습니다. 이러한 사용자를 그룹화하여 이후 예제에서 설명하는 서브스크립션 관리 그룹을 나타낼 수 있습니다.

3. 터미널에서 Red Hat Advanced Cluster Management 클러스터에 로그인합니다.

4. open-cluster-management:subscription-admin ClusterRoleBinding이 없는 경우 이를 생성해야 합니다. 다음 예제를 참조하십시오.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: open-cluster-management:subscription-admin
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: open-cluster-management:subscription-admin
```

5. 다음 명령을 사용하여 **open-cluster-management:subscription-admin ClusterRoleBinding**에 다음 제목을 추가합니다.

```
oc edit clusterrolebinding open-cluster-management:subscription-admin
```

참고: 처음에는 **open-cluster-management:subscription-admin ClusterRoleBinding**에 제목이 없습니다.

제목이 다음 예로 표시될 수 있습니다.

```
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: example-name
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: example-group-name
- kind: ServiceAccount
  name: my-service-account
  namespace: my-service-account-namespace
# Service Account can be used as a user subject as well
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: 'system:serviceaccount:my-service-account-namespace:my-service-account'
```

1.5.3. 서브스크립션 관리자로 허용 및 거부 목록 생성

서브스크립션 관리자는 지정된 **Kubernetes** 종류 리소스만 배포할 수 있도록 허용 목록이 포함된 **Git** 리포지토리 애플리케이션 서브스크립션에서 애플리케이션을 생성할 수 있습니다. 애플리케이션 서브스크립션에 거부 목록을 생성하여 특정 **Kubernetes** 종류 리소스의 배포를 거부할 수도 있습니다.

기본적으로 **policy.open-cluster-management.io/v1** 리소스는 애플리케이션 서브스크립션에 의해 배포되지 않습니다. 이러한 기본 동작을 방지하려면 서브스크립션 관리자가 애플리케이션 서브스크립션을 배포해야 합니다.

다음 허용 및 거부 사양 예를 참조하십시오.

```
apiVersion: apps.open-cluster-management.io/v1
kind: Subscription
metadata:
  annotations:
    apps.open-cluster-management.io/github-path: sub2
  name: demo-subscription
  namespace: demo-ns
spec:
  channel: demo-ns/somechannel
  allow:
  - apiVersion: policy.open-cluster-management.io/v1
    kinds:
    - Policy
  - apiVersion: v1
    kinds:
    - Deployment
  deny:
  - apiVersion: v1
    kinds:
    - Service
    - ConfigMap
  placement:
    local: true
```

다음 애플리케이션 서브스크립션 YAML은 소스 리포지토리의 myapplication 디렉터리에서 애플리케이션을 배포할 때 소스 리포지토리에 다른 리소스가 있더라도 v1/Deployment 리소스만 배포하도록 지정합니다.

```
apiVersion: apps.open-cluster-management.io/v1
kind: Subscription
metadata:
  annotations:
    apps.open-cluster-management.io/github-path: myapplication
  name: demo-subscription
  namespace: demo-ns
spec:
  channel: demo-ns/somechannel
  deny:
  - apiVersion: v1
     kinds:
      - Service
      - ConfigMap
  placement:
    placementRef:
      name: demo-placement
      kind: PlacementRule
```

이 예제 애플리케이션 서브스크립션 YAML에서는 v1/Service 및 v1/ConfigMap 리소스를 제외한 모

든 유효한 리소스의 배포를 지정합니다. API 그룹 내에서 개별 리소스 종류를 나열하는 대신 "*" 를 추가하여 API 그룹에서 모든 리소스 종류를 허용하거나 거부할 수 있습니다.

1.5.4. 조정 옵션 추가

개별 리소스에서 apps.open-cluster-management.io/reconcile-option 주석을 사용하여 서브스크립션 수준 조정 옵션을 재정의할 수 있습니다.

예를 들어 서브스크립션된 Git 리포지토리의 리소스 YAML에 apps.open-cluster-management.io/reconcile-option: 주석을 교체하고 apps.open-cluster-management.io/reconcile-option: 병합 주석을 추가하면 다른 리소스가 교체되는 동안 대상 클러스터에서 리소스가 병합됩니다.

1.5.4.1. 빈도 Git 채널 조정

채널 구성에서 high,medium,low, off 등의 빈도 옵션을 선택하여 불필요한 리소스 조정을 방지하고 서브스크립션 운영자에 대한 과부하를 방지할 수 있습니다.

필수 액세스 권한: 관리자 및 클러스터 관리자

설정의 다음 정의를 참조하십시오:attribute:<value>:

- off: 배포된 리소스는 자동으로 조정되지 않습니다. Subscription 사용자 정의 리소스의 변경으로 인해 조정이 시작됩니다. 레이블 또는 주석을 추가하거나 업데이트할 수 있습니다.
- 낮음: 소스 Git 리포지토리에 변경이 없는 경우에도 배포된 리소스는 시간마다 자동으로 조정됩니다.
- 이 설정은 기본 설정입니다.This is the default setting. 서브스크립션 운영자는 현재 배포된 커밋 ID를 3분마다 소스 리포지토리의 최신 커밋 ID와 비교하고 대상 클러스터에 변경 사항을 적용합니다. 리포지토리에 변경이 없는 경우에도 모든 리소스가 소스 Git 리포지토리에서 대상 클러스터로 다시 적용됩니다.
- High: 소스 Git 리포지토리에 변경이 없는 경우에도 배포된 리소스는 2분마다 자동으로 조정됩니다.

서브스크립션에서 참조하는 채널 사용자 정의 리소스의 apps.open-cluster-

management.io/reconcile-rate 주석을 사용하여 이를 설정할 수 있습니다.

git-channel 예제를 참조하십시오.

```
apiVersion: apps.open-cluster-management.io/v1
kind: Channel
metadata:
  name: git-channel
  namespace: sample
  annotations:
    apps.open-cluster-management.io/reconcile-rate: <value from the list>
spec:
  type: GitHub
  pathname: <Git URL>
---
apiVersion: apps.open-cluster-management.io/v1
kind: Subscription
metadata:
  name: git-subscription
  annotations:
    apps.open-cluster-management.io/git-path: <application1>
    apps.open-cluster-management.io/git-branch: <branch1>
spec:
  channel: sample/git-channel
  placement:
    local: true
```

이전 예에서 sample/git-channel 을 사용하는 모든 서브스크립션에는 낮은 조정 빈도가 할당됩니다.

a. 서브스크립션 조정 비율이 낮은 것으로 설정되면 서브스크립션된 애플리케이션 리소스를 조정하는 데 최대 1시간이 걸릴 수 있습니다. 단일 애플리케이션 보기의 카드에서 동기화 를 클릭하여 수동으로 조정합니다. off 로 설정하면 조정되지 않습니다.

채널의 reconcile-rate 설정에 관계없이 서브스크립션에서 Subscription 사용자 정의 리소스에서 apps.open-cluster-management.io/reconcile-rate: off 주석을 지정하여 자동 조정을 해제할 수 있습니다.

다음 git-channel 예제를 참조하십시오.

```
apiVersion: apps.open-cluster-management.io/v1
kind: Channel
metadata:
  name: git-channel
  namespace: sample
  annotations:
```

```
    apps.open-cluster-management.io/reconcile-rate: high
spec:
  type: GitHub
  pathname: <Git URL>
---
apiVersion: apps.open-cluster-management.io/v1
kind: Subscription
metadata:
  name: git-subscription
  annotations:
    apps.open-cluster-management.io/git-path: application1
    apps.open-cluster-management.io/git-branch: branch1
    apps.open-cluster-management.io/reconcile-rate: "off"
spec:
  channel: sample/git-channel
  placement:
    local: true
```

조정- rate가 채널에서 high 로 설정되어 있어도 git-subscription 에서 배포한 리소스가 자동으로 조정 되지 않음을 확인합니다.

1.5.4.2. 빈도 Helm 채널 조정

15분마다 서브스크립션 Operator는 현재 배포된 Helm 차트의 해시와 소스 리포지터리의 해시를 비교합니다. 대상 클러스터에 변경 사항이 적용됩니다. 리소스 조정 빈도는 다른 애플리케이션 배포 및 업데이트의 성능에 영향을 미칩니다.

예를 들어 수백 개의 애플리케이션 서브스크립션이 있고 모든 서브스크립션을 더 자주 조정하려는 경우 조정 시간이 더 느립니다.

애플리케이션의 Kubernetes 리소스에 따라 적절한 조정 빈도가 성능을 향상시킬 수 있습니다.

- off: 배포된 리소스는 자동으로 조정되지 않습니다. Subscription 사용자 정의 리소스의 변경으로 인해 조정이 시작됩니다. 레이블 또는 주석을 추가하거나 업데이트할 수 있습니다.
- Low: Subscription Operator는 현재 배포된 해시를 매시간 소스 리포지토리의 해시와 비교하고 변경이 있을 때 대상 클러스터에 변경 사항을 적용합니다.
- 이 설정은 기본 설정입니다.This is the default setting. 서브스크립션 Operator는 현재 배포된 해시와 15분마다 소스 리포지토리의 해시와 비교하고 변경이 있을 때 대상 클러스터에 변경 사항을 적용합니다.

- **High: Subscription Operator**는 현재 배포된 해시를 **2**분마다 소스 리포지토리의 해시와 비교하고 변경이 있을 때 대상 클러스터에 변경 사항을 적용합니다.

서브스크립션에서 참조하는 **Channel** 사용자 정의 리소스에서 **apps.open-cluster-management.io/reconcile-rate** 주석을 사용하여 이 값을 설정할 수 있습니다. 다음 **helm-channel** 예제를 참조하십시오.

다음 **helm-channel** 예제를 참조하십시오.

```
apiVersion: apps.open-cluster-management.io/v1
kind: Channel
metadata:
  name: helm-channel
  namespace: sample
  annotations:
    apps.open-cluster-management.io/reconcile-rate: low
spec:
  type: HelmRepo
  pathname: <Helm repo URL>
---
apiVersion: apps.open-cluster-management.io/v1
kind: Subscription
metadata:
  name: helm-subscription
spec:
  channel: sample/helm-channel
  name: nginx-ingress
  packageOverrides:
  - packageName: nginx-ingress
    packageAlias: nginx-ingress-simple
    packageOverrides:
    - path: spec
      value:
        defaultBackend:
          replicaCount: 3
  placement:
    local: true
```

이 예에서 **sample/helm-channel** 을 사용하는 모든 서브스크립션에는 낮은 조정 빈도가 할당됩니다.

채널의 **reconcile-rate** 설정에 관계없이 서브스크립션은 다음 예에 표시된 대로 **Subscription** 사용자 정의 리소스에 **apps.open-cluster-management.io/reconcile-rate: off** 주석을 지정하여 자동 조정을 해제할 수 있습니다.

```
apiVersion: apps.open-cluster-management.io/v1
kind: Channel
```

```
metadata:
  name: helm-channel
  namespace: sample
  annotations:
    apps.open-cluster-management.io/reconcile-rate: high
spec:
  type: HelmRepo
  pathname: <Helm repo URL>
---
apiVersion: apps.open-cluster-management.io/v1
kind: Subscription
metadata:
  name: helm-subscription
  annotations:
    apps.open-cluster-management.io/reconcile-rate: "off"
spec:
  channel: sample/helm-channel
  name: nginx-ingress
  packageOverrides:
  - packageName: nginx-ingress
    packageAlias: nginx-ingress-simple
    packageOverrides:
    - path: spec
      value:
        defaultBackend:
          replicaCount: 3
  placement:
    local: true
```

이 예에서 조정- rate가 채널에서 high 로 설정되어 있어도 helm-subscription 에 의해 배포된 리소스는 자동으로 조정 되지 않습니다.

1.5.5. 보안 Git 연결에 대한 애플리케이션 채널 및 서브스크립션 구성

Git 채널 및 서브스크립션은 HTTPS 또는 SSH를 통해 지정된 Git 리포지토리에 연결됩니다. 보안 Git 연결에 다음 애플리케이션 채널 구성을 사용할 수 있습니다.

- 사용자 및 액세스 토큰을 사용하여 개인 리포지터리에 연결
- Git 서버에 비보안 HTTPS 연결
- 보안 HTTPS 연결에 사용자 정의 CA 인증서 사용
- Git 서버에 SSH 연결

- 인증서 및 SSH 키 업데이트

1.5.5.1. 사용자 및 액세스 토큰을 사용하여 개인 리포지터리에 연결

채널 및 서브스크립션을 사용하여 Git 서버에 연결할 수 있습니다. 사용자 및 액세스 토큰을 사용하여 개인 리포지토리에 연결하려면 다음 절차를 참조하십시오.

1. 채널과 동일한 네임스페이스에 보안을 생성합니다. user 필드를 Git 사용자 ID로 설정하고 accessToken 필드를 Git 개인 액세스 토큰으로 설정합니다. 값은 base64로 인코딩되어야 합니다. user 및 accessToken이 채워진 다음 샘플을 참조하십시오.

```
apiVersion: v1
kind: Secret
metadata:
  name: my-git-secret
  namespace: channel-ns
data:
  user: dXNlcgo=
  accessToken: cGFzc3dvcmQK
```

2. 시크릿을 사용하여 채널을 구성합니다. secretRef 가 채워진 다음 샘플을 참조하십시오.

```
apiVersion: apps.open-cluster-management.io/v1
kind: Channel
metadata:
  name: sample-channel
  namespace: channel-ns
spec:
    type: Git
    pathname: <Git HTTPS URL>
    secretRef:
      name: my-git-secret
```

1.5.5.2. Git 서버에 비보안 HTTPS 연결

개발 환경에서 다음 연결 방법을 사용하여 사용자 정의 또는 자체 서명된 인증 기관에서 서명한 SSL 인증서로 개인 호스팅 Git 서버에 연결할 수 있습니다. 그러나 다음 절차는 프로덕션에는 권장되지 않습니다.

채널 사양에 insecureSkipVerify: true 를 지정합니다. 그러지 않으면 Git 서버에 대한 연결이 실패하고 다음과 유사한 오류와 함께 실패합니다.

```
x509: certificate is valid for localhost.com, not localhost
```

이 방법에 대한 채널 사양 추가를 사용하여 다음 샘플을 참조하십시오.

```
apiVersion: apps.open-cluster-management.io/v1
ind: Channel
metadata:
labels:
  name: sample-channel
  namespace: sample
spec:
  type: GitHub
  pathname: <Git HTTPS URL>
  insecureSkipVerify: true
```

1.5.5.3. 보안 HTTPS 연결에 사용자 정의 CA 인증서 사용

이 연결 방법을 사용하여 사용자 정의 또는 자체 서명된 인증 기관에서 서명한 SSL 인증서로 개인 호스팅 Git 서버에 안전하게 연결할 수 있습니다.

1. Git 서버 루트 및 중간 CA 인증서를 PEM 형식으로 포함할 ConfigMap을 생성합니다. ConfigMap은 채널 CR과 동일한 네임스페이스에 있어야 합니다. 필드 이름은 caCerts 여야 하며 사용 | 이어야 합니다. 다음 샘플에서 caCerts 에는 루트 및 중간 CA와 같은 여러 인증서가 포함될 수 있습니다.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: git-ca
  namespace: channel-ns
data:
  caCerts: |
    # Git server root CA

    -----BEGIN CERTIFICATE-----
    MIIF5DCCA8wCCQDInYMol7LSDTANBgkqhkiG9w0BAQsFADCBszELMAkGA1UEBhMC

Q0ExCzAJBgNVBAgMAk9OMRAwDgYDVQQHDAdUb3JvbnRvMQ8wDQYDVQQKDAZSZW
RI

YXQxDDAKBgNVBAsMA0FDTTFFMEMGA1UEAww8Z29ncy1zdmMtZGVmYXVsdC5hcHBz
    LnJqdW5nLWh1YjEzLmRldjA2LnJlZC1jaGVzdGVyZmllbGQuY29tMR8wHQYJKoZI
    hvcNAQkBFhByb2tlakByZWRoYXQuY29tMB4XDTIwMTIwMzE4NTMxMloXDTIzMDky

MzE4NTMxMlowgbMxCzAJBgNVBAYTAkNBMQswCQYDVQQIDAJPTjEQMA4GA1UEBwwH


VG9yb250bzEPMA0GA1UECgwGUmVkSGF0MQwwCgYDVQQLDANBQ00xRTBDBgNVBA
MM
```

```
    PGdvZ3Mtc3ZjLWRlZmF1bHQuYXBwcy5yanVuZy1odWIxMy5kZXYwNi5yZWQtY2hl
    c3RlcmZpZWxkLmNvbTEfMB0GCSqGSIb3DQEJARYQcm9rZWpAcmVkaGF0LmNvbTCC
    AiIwDQYJKoZIhvcNAQEBBQADggIPADCCAgoCggIBAM3nPK4mOQzaDAo6S3ZJ0Ic3
    U9p/NLodnoTIC+cn0q8qNCAjf13zbGB3bfN9Zxl8Q5fv+wYwHrUOReCp6U/InyQy
    6OS3gj738F635inz1KdyhKtlWW2p9Ye9DUtx1IlfHkDVdXtynjHQbsFNIdRHcpQP
    upM5pwPC3BZXqvXChhlfAy2m4yu7vy0hO/oTzWIwNsoL5xt0Lw4mSyhlEip/t8lU
    xn2y8qhm7MiIUpXuwWhSYgCrEVqmTcB70Pc2YRZdSFolMN9Et70MjQN0TXjoktH8
    PyASJIKIRd+48yROIbUn8rj4aYYBsJuoSCjJNwujZPbqseqUr42+v+Qp2bBj1Sjw
    +SEZfHTvSv8AqX0T6eo6njr578+DgYlwsS1A1zcAdzp8qmDGqvJDzwcnQVFmvaoM
    gGHCdJihfy3vDhxuZRDse0V4Pz6tl6iklM+tHrJL/bdL0NdfJXNCqn2nKrM51fpw
    diNXs4Zn3QSStC2x2hKnK+Q1rwCSEg/lBawgxGUslTboFH77a+Kwu4Oug9ibtm5z
    ISs/JY4Kiy4C2XJOltOR2XZYkdKaX4x3ctbrGaD8Bj+QHiSAxaaSXIX+VbzkHF2N
    aD5ijFUopjQEKFrYh3O93DB/URIQ+wHVa6+Kvu3uqE0cg6pQsLpbFVQ/I8xHvt9L
    kYy6z6V/nj9ZYKQbq/kPAgMBAAEwDQYJKoZIhvcNAQELBQADggIBAKZuc+lewYAv
    jaaSeRDRoToTb/yN0Xsi69UfK0aBdvhCa7/0rPHcv8hmUBH3YgkZ+CSA5ygajtL4
    g2E8CwIO9ZjZ6l+pHCuqmNYoX1wdjaaDXlpwk8hGTSgy1LsOoYrC5ZysCi9Jilu9
    PQVGs/vehQRqLV9uZBigG6oZqdUqEimaLHrOcEAHB5RVcnFurz0qNbT+UySjsD63
    9yJdCeQbeKAR9SC4hG13EbM/RZh0lgFupkmGts7QYULzT+oA0cCJpPLQl6m6qGyE
    kh9aBB7FLykK1TeXVuANlNU4EMyJ/e+uhNkS9ubNJ3vuRuo+ECHsha058yi16JC9
    NkZqP+df4Hp85sd+xhrgYieq7QGX2KOXAjqAWo9htoBhOyW3mm783A7WcOiBMQv0
    2UGZxMsRjlP6UqB08LsV5ZBAefElR344sokJR1de/Sx2J9J/am7yOoqbtKpQotIA
    XSUkATuuQw4ctyZLDkUpzrDzgd2Bt+aawF6sD2YqycaGFwv2YD9t1YlD6F4Wh8Mc
    20Qu5EGrkQTCWZ9pOHNSa7YQdmJzwbxJC4hqBpBRAJFI2fAIqFtyum6/8ZN9nZ9K
    FSEKdlu+xeb6Y6xYt0mJJWF6mCRi4i7IL74EU/VNXwFmfP6ladliUOST3w5t92cB
    M26t73UCExXMXTCQvnp0ki84PeR1kRk4
    -----END CERTIFICATE-----

    # Git server intermediate CA 1

    -----BEGIN CERTIFICATE-----
    MIIF5DCCA8wCCQDInYMol7LSDTANBgkqhkiG9w0BAQsFADCBszELMAkGA1UEBhMC

Q0ExCzAJBgNVBAgMAk9OMRAwDgYDVQQHDAdUb3JvbnRvMQ8wDQYDVQQKDAZSZW
RI

YXQxDDAKBgNVBAsMA0FDTTFFMEMGA1UEAww8Z29ncy1zdmMtZGVmYXVsdC5hcHBz
    LnJqdW5nLWh1YjEzLmRldjA2LnJlZC1jaGVzdGVyZmllbGQuY29tMR8wHQYJKoZI
    hvcNAQkBFhByb2tlakByZWRoYXQuY29tMB4XDTIwMTIwMzE4NTMxMloXDTIzMDky

MzE4NTMxMlowgbMxCzAJBgNVBAYTAkNBMQswCQYDVQQIDAJPTjEQMA4GA1UEBwwH


VG9yb250bzEPMA0GA1UECgwGUmVkSGF0MQwwCgYDVQQLDANBQ00xRTBDBgNVBA
MM
    PGdvZ3Mtc3ZjLWRlZmF1bHQuYXBwcy5yanVuZy1odWIxMy5kZXYwNi5yZWQtY2hl
    c3RlcmZpZWxkLmNvbTEfMB0GCSqGSIb3DQEJARYQcm9rZWpAcmVkaGF0LmNvbTCC
    AiIwDQYJKoZIhvcNAQEBBQADggIPADCCAgoCggIBAM3nPK4mOQzaDAo6S3ZJ0Ic3
    U9p/NLodnoTIC+cn0q8qNCAjf13zbGB3bfN9Zxl8Q5fv+wYwHrUOReCp6U/InyQy
    6OS3gj738F635inz1KdyhKtlWW2p9Ye9DUtx1IlfHkDVdXtynjHQbsFNIdRHcpQP
    upM5pwPC3BZXqvXChhlfAy2m4yu7vy0hO/oTzWIwNsoL5xt0Lw4mSyhlEip/t8lU
    xn2y8qhm7MiIUpXuwWhSYgCrEVqmTcB70Pc2YRZdSFolMN9Et70MjQN0TXjoktH8
    PyASJIKIRd+48yROIbUn8rj4aYYBsJuoSCjJNwujZPbqseqUr42+v+Qp2bBj1Sjw
    +SEZfHTvSv8AqX0T6eo6njr578+DgYlwsS1A1zcAdzp8qmDGqvJDzwcnQVFmvaoM
    gGHCdJihfy3vDhxuZRDse0V4Pz6tl6iklM+tHrJL/bdL0NdfJXNCqn2nKrM51fpw
    diNXs4Zn3QSStC2x2hKnK+Q1rwCSEg/lBawgxGUslTboFH77a+Kwu4Oug9ibtm5z
    ISs/JY4Kiy4C2XJOltOR2XZYkdKaX4x3ctbrGaD8Bj+QHiSAxaaSXIX+VbzkHF2N
```

```
    aD5ijFUopjQEKFrYh3O93DB/URIQ+wHVa6+Kvu3uqE0cg6pQsLpbFVQ/I8xHvt9L
    kYy6z6V/nj9ZYKQbq/kPAgMBAAEwDQYJKoZIhvcNAQELBQADggIBAKZuc+lewYAv
    jaaSeRDRoToTb/yN0Xsi69UfK0aBdvhCa7/0rPHcv8hmUBH3YgkZ+CSA5ygajtL4
    g2E8CwIO9ZjZ6l+pHCuqmNYoX1wdjaaDXlpwk8hGTSgy1LsOoYrC5ZysCi9Jilu9
    PQVGs/vehQRqLV9uZBigG6oZqdUqEimaLHrOcEAHB5RVcnFurz0qNbT+UySjsD63
    9yJdCeQbeKAR9SC4hG13EbM/RZh0lgFupkmGts7QYULzT+oA0cCJpPLQl6m6qGyE
    kh9aBB7FLykK1TeXVuANlNU4EMyJ/e+uhNkS9ubNJ3vuRuo+ECHsha058yi16JC9
    NkZqP+df4Hp85sd+xhrgYieq7QGX2KOXAjqAWo9htoBhOyW3mm783A7WcOiBMQv0
    2UGZxMsRjlP6UqB08LsV5ZBAefElR344sokJR1de/Sx2J9J/am7yOoqbtKpQotIA
    XSUkATuuQw4ctyZLDkUpzrDzgd2Bt+aawF6sD2YqycaGFwv2YD9t1YlD6F4Wh8Mc
    20Qu5EGrkQTCWZ9pOHNSa7YQdmJzwbxJC4hqBpBRAJFl2fAIqFtyum6/8ZN9nZ9K
    FSEKdlu+xeb6Y6xYt0mJJWF6mCRi4i7IL74EU/VNXwFmfP6IadliUOST3w5t92cB
    M26t73UCExXMXTCQvnp0ki84PeR1kRk4
    -----END CERTIFICATE-----

    # Git server intermediate CA 2

    -----BEGIN CERTIFICATE-----
    MIIF5DCCA8wCCQDInYMol7LSDTANBgkqhkiG9w0BAQsFADCBszELMAkGA1UEBhMC

Q0ExCzAJBgNVBAgMAk9OMRAwDgYDVQQHDAdUb3JvbnRvMQ8wDQYDVQQKDAZSZW
RI

YXQxDDAKBgNVBAsMA0FDTTFFMEMGA1UEAww8Z29ncy1zdmMtZGVmYXVsdC5hcHBz
    LnJqdW5nLWh1YjEzLmRldjA2LnJlZC1jaGVzdGVyZmllbGQuY29tMR8wHQYJKoZI
    hvcNAQkBFhByb2tlakByZWRoYXQuY29tMB4XDTIwMTIwMzE4NTMxMloXDTIzMDky

MzE4NTMxMlowgbMxCzAJBgNVBAYTAkNBMQswCQYDVQQIDAJPTjEQMA4GA1UEBwwH


VG9yb250bzEPMA0GA1UECgwGUmVkSGF0MQwwCgYDVQQLDANBQ00xRTBDBgNVBA
MM
    PGdvZ3Mtc3ZjLWRlZmF1bHQuYXBwcy5yanVuZy1odWIxMy5kZXYwNi5yZWQtY2hl
    c3RlcmZpZWxkLmNvbTEfMB0GCSqGSIb3DQEJARYQcm9rZWpAcmVkaGF0LmNvbTCC
    AiIwDQYJKoZIhvcNAQEBBQADggIPADCCAgoCggIBAM3nPK4mOQzaDAo6S3ZJ0Ic3
    U9p/NLodnoTIC+cn0q8qNCAjf13zbGB3bfN9Zxl8Q5fv+wYwHrUOReCp6U/InyQy
    6OS3gj738F635inz1KdyhKtlWW2p9Ye9DUtx1IlfHkDVdXtynjHQbsFNIdRHcpQP
    upM5pwPC3BZXqvXChhlfAy2m4yu7vy0hO/oTzWlwNsoL5xt0Lw4mSyhlEip/t8lU
    xn2y8qhm7MiIUpXuwWhSYgCrEVqmTcB70Pc2YRZdSFolMN9Et70MjQN0TXjoktH8
    PyASJIKIRd+48yROIbUn8rj4aYYBsJuoSCjJNwujZPbqseqUr42+v+Qp2bBj1Sjw
    +SEZfHTvSv8AqX0T6eo6njr578+DgYlwsS1A1zcAdzp8qmDGqvJDzwcnQVFmvaoM
    gGHCdJihfy3vDhxuZRDse0V4Pz6tl6iklM+tHrJL/bdL0NdfJXNCqn2nKrM51fpw
    diNXs4Zn3QSStC2x2hKnK+Q1rwCSEg/lBawgxGUslTboFH77a+Kwu4Oug9ibtm5z
    ISs/JY4Kiy4C2XJOltOR2XZYkdKaX4x3ctbrGaD8Bj+QHiSAxaaSXIX+VbzkHF2N
    aD5ijFUopjQEKFrYh3O93DB/URIQ+wHVa6+Kvu3uqE0cg6pQsLpbFVQ/I8xHvt9L
    kYy6z6V/nj9ZYKQbq/kPAgMBAAEwDQYJKoZIhvcNAQELBQADggIBAKZuc+lewYAv
    jaaSeRDRoToTb/yN0Xsi69UfK0aBdvhCa7/0rPHcv8hmUBH3YgkZ+CSA5ygajtL4
    g2E8CwIO9ZjZ6l+pHCuqmNYoX1wdjaaDXlpwk8hGTSgy1LsOoYrC5ZysCi9Jilu9
    PQVGs/vehQRqLV9uZBigG6oZqdUqEimaLHrOcEAHB5RVcnFurz0qNbT+UySjsD63
    9yJdCeQbeKAR9SC4hG13EbM/RZh0lgFupkmGts7QYULzT+oA0cCJpPLQl6m6qGyE
    kh9aBB7FLykK1TeXVuANlNU4EMyJ/e+uhNkS9ubNJ3vuRuo+ECHsha058yi16JC9
    NkZqP+df4Hp85sd+xhrgYieq7QGX2KOXAjqAWo9htoBhOyW3mm783A7WcOiBMQv0
    2UGZxMsRjlP6UqB08LsV5ZBAefElR344sokJR1de/Sx2J9J/am7yOoqbtKpQotIA
    XSUkATuuQw4ctyZLDkUpzrDzgd2Bt+aawF6sD2YqycaGFwv2YD9t1YlD6F4Wh8Mc
    20Qu5EGrkQTCWZ9pOHNSa7YQdmJzwbxJC4hqBpBRAJFl2fAIqFtyum6/8ZN9nZ9K
```

```
FSEKdlu+xeb6Y6xYt0mJJWF6mCRi4i7IL74EU/VNXwFmfP6ladliUOST3w5t92cB
M26t73UCExXMXTCQvnp0ki84PeR1kRk4
-----END CERTIFICATE-----
```

2. 이 **ConfigMap**으로 채널을 구성합니다. 이전 단계의 **git-ca** 이름을 사용하여 다음 샘플을 참조하십시오.

```
apiVersion: apps.open-cluster-management.io/v1
kind: Channel
metadata:
  name: my-channel
  namespace: channel-ns
spec:
  configMapRef:
    name: git-ca
  pathname: <Git HTTPS URL>
  type: Git
```

1.5.5.4. Git 서버에 SSH 연결

1. 데이터의 **sshKey** 필드에 개인 **SSH** 키를 포함할 시크릿을 생성합니다. 키가 암호로 보호되는 경우 암호 필드에 암호 를 지정합니다. 이 보안은 채널 **CR**과 동일한 네임스페이스에 있어야 합니다. **oc** 명령을 사용하여 보안 일반 **git-ssh-key --from-file=sshKey=./.ssh/id_rsa** 를 생성한 다음 **base64** 인코딩 암호를 추가합니다. 다음 샘플을 참조하십시오.

```
apiVersion: v1
kind: Secret
metadata:
  name: git-ssh-key
  namespace: channel-ns
data:
  sshKey:
LS0tLS1CRUdJTiBPUEVOU1NIIFBSSVZBVEUgS0VZLS0tLS0KYjNCbGJuTnphQzFyWlh
rdGRqRUFBQUFBQ21GbGN6STFOaTFqZEhJQUFBQUdZbU55ZVhCMEFBQUFHQUFB
QUJDK3YySHhWSlwCm8zejh1endzV3NWODMvSFVkOEtGeVBmWk5OeE5TQUgcFA3Y
k1yR2tIRFFPd3J6MGlKOUlRM0tKVXQzWEE0Zmd6NVlrVFVhcTJsZWxxVk1HcXI2WHF2
UVJ5Mkc0NkRlRVlYUGpabVZMcGVuaGtRYU5HYmpaMmZOdQpWUGpiOVhZRmd4bTN
nYUpJU3BNeTFLWjQ5MzJvOFByaDZEdzRYVUF1a28wZGdBaDdndVpPaE53b0pVYnN
mYlZRc0xMS1RrCnQwblZ1anRvd2NEVGx4TlplUjcwbGVUSHdGQTYwekM0elpMNkRPc
3RMYjV2LzZhMjFHRlMwVmVXQ3YvMlpMOE1sbjVUZWwKSytoUWtxRnJBL3BUc1ozVX
NjSG1GUi9PV25FPQotLS0tLUVORCBPUEVOU1NIIFBSSVZBVEUgS0VZLS0tLS0K
  passphrase: cGFzc3cwcmQK
type: Opaque
```

2. 시크릿을 사용하여 채널을 구성합니다. 다음 샘플을 참조하십시오.

```
apiVersion: apps.open-cluster-management.io/v1
kind: Channel
```

```
metadata:
  name: my-channel
  namespace: channel-ns
spec:
  secretRef:
    name: git-ssh-key
  pathname: <Git SSH URL>
  type: Git
```

서브스크립션 컨트롤러는 제공된 Git 호스트 이름과 ssh-keyscan 을 수행하여 SSH 연결에서 MIT(Man-in-the-middle) 공격을 방지하기 위해 known_hosts 목록을 빌드합니다. 이 값을 건너뛰고 비보안 연결을 만들려면 채널 구성에서 insecureSkipVerify: true 를 사용합니다. 이 방법은 특히 프로덕션 환경에서는 모범 사례가 아닙니다.

```
apiVersion: apps.open-cluster-management.io/v1
kind: Channel
metadata:
  name: my-channel
  namespace: channel-ns
spec:
  secretRef:
    name: git-ssh-key
  pathname: <Git SSH URL>
  type: Git
  insecureSkipVerify: true
```

1.5.5.5. 인증서 및 SSH 키 업데이트

Git 채널 연결 구성에 CA 인증서, 인증 정보 또는 SSH 키와 같은 업데이트가 필요한 경우 동일한 네임스페이스에 새 시크릿 및 ConfigMap을 생성하고 해당 새 시크릿 및 ConfigMap을 참조하도록 채널을 업데이트해야 합니다. 자세한 내용은 보안 HTTPS 연결에 사용자 정의 CA 인증서 사용을 참조하십시오.

1.5.6. Ansible Tower 작업 설정

Red Hat Advanced Cluster Management는 Ansible Tower 자동화와 통합되어 Git 서브스크립션 애플리케이션 관리를 위한 prehook 및 posthook AnsibleJob 인스턴스를 생성할 수 있습니다. Ansible Tower 작업을 사용하면 작업을 자동화하고 Slack 및 PagerDuty 서비스와 같은 외부 서비스와 통합할 수 있습니다. Git 리포지토리 리소스 루트 경로에는 앱 배포, 앱 업데이트 또는 클러스터에서 앱을 제거하는 과정의 일부로 실행되는 Ansible Tower 작업에 대한 prehook 및 posthook 디렉터리가 포함됩니다.

필수 액세스 권한: 클러스터 관리자

- 사전 요구 사항

- Ansible Automation Platform Resource Operator 설치
- 인증 정보 설정
- Ansible 통합
- Ansible Operator 구성 요소
- Ansible 구성
- 보안 조정 설정
- Ansible 샘플 YAML

1.5.6.1. 사전 요구 사항

- OpenShift Container Platform 4.6 이상
- Ansible Tower 버전 3.7.3 또는 이후 버전이 설치되어 있어야 합니다. 지원되는 최신 Ansible Tower 버전을 설치하는 것이 좋습니다. 자세한 내용은 Red Hatanchor 설명서 를 참조하십시오.
- Ansible Automation Platform Resource Operator를 설치하여 Ansible 작업을 Git 서브스크립션 라이프사이클에 연결합니다. AnsibleJob을 사용하여 Ansible Tower 작업을 시작할 때 최상의 결과를 얻으려면 Ansible Tower 작업 템플릿이 멱등이어야 합니다.

INVENTORY 및 EXTRA VARIABLES 둘 다에 대해 템플릿에서 PROMPT ON >-<을 선택합니다. 자세한 내용은 작업 템플릿을 참조하십시오.

1.5.6.2. Ansible Automation Platform Resource Operator 설치

1. OpenShift Container Platform 클러스터 콘솔에 로그인합니다.

2. 콘솔 탐색에서 **OperatorHub** 를 클릭합니다.

3. ***Ansible Automation Platform Resource Operator*** 를 검색하고 설치합니다. 참고: **prehook** 및 **posthook AnsibleJobs** 를 제출하려면 다른 **OpenShift Container Platform** 버전에서 사용 가능한 해당 버전으로 **AAP(Ansible Automation Platform) Resource Operator**를 설치합니다.

- **OpenShift Container Platform 4.6**에는 **AAP (AAP) Resource Operator early-access**가 필요합니다.
- **OpenShift Container Platform 4.7**에는 **AAP (AAP) Resource Operator early-access, stable-2.1**이 필요합니다.
- **OpenShift Container Platform 4.8**에는 **AAP) Resource Operator early-access, stable-2.1, stable-2.2**가 필요합니다.
- **OpenShift Container Platform 4.9 (AAP) Resource Operator early-access, stable-2.1, stable-2.2** 필요
- **OpenShift Container Platform 4.10**에는 **AAP(Resource Operator) stable-2.1, stable-2.2**가 필요합니다.

1.5.6.3. 인증 정보 설정

콘솔의 인증 정보 페이지에서 필요한 인증 정보를 생성할 수 있습니다. *인증 정보 추가* 를 클릭하거나 *탐색에서 페이지에 액세스합니다. 인증 정보 정보는* ***Ansible Automation Platform에 대한 인증 정보 생성을*** *참조하십시오.*

1.5.6.4. Ansible 통합

Ansible Tower *작업을* ***Git*** *서브스크립션에 통합할 수 있습니다. 예를 들어 데이터베이스 프런트 엔드 및 백엔드 애플리케이션의 경우* ***Ansible*** *작업과 함께* ***Ansible Tower***를 사용하여 데이터베이스를 인스턴스화해야 하며 애플리케이션은 ***Git*** *서브스크립션을 통해 설치됩니다. 서브스크립션을 사용하여 프런트 엔드 및 백엔드 애플리케이션을 배포하기 전에 데이터베이스가 인스턴스화됩니다.*

애플리케이션 서브스크립션 ***Operator***가 향상되어 ***prehook*** 및 ***posthook*** 두 개의 하위 폴더를 정의할 수 있습니다. 두 폴더 모두 ***Git*** 리포지토리 리소스 루트 경로에 있으며, 각각 ***prehook*** 및 ***posthook***

Ansible 작업을 모두 포함합니다.

Git 서브스크립션이 생성되면 모든 사전 및 post AnsibleJob 리소스가 구문 분석하고 메모리에 오브젝트로 저장됩니다. 애플리케이션 서브스크립션 컨트롤러는 사전 및 AnsibleJob 인스턴스 생성 시기를 결정합니다.

1.5.6.5. Ansible Operator 구성 요소

서브스크립션 CR을 생성할 때 Git-branch 및 Git-path는 Git 리포지토리 루트 위치를 가리킵니다. Git 루트 위치에서 두 개의 하위 폴더 prehook 및 posthook 에는 Kind:AnsibleJob 리소스가 하나 이상 포함되어야 합니다.

1.5.6.5.1. Prehook

애플리케이션 서브스크립션 컨트롤러는 prehook AnsibleJob 오브젝트로 prehook 폴더의 모든 Kind:AnsibleJob CR을 검색한 다음 새 prehook AnsibleJob 인스턴스를 생성합니다. 새 인스턴스 이름은 prehook AnsibleJob 오브젝트 이름과 임의의 접미사 문자열입니다.

인스턴스 이름: database-sync-1-2913063 예제를 참조하십시오.

애플리케이션 서브스크립션 컨트롤러는 prehook AnsibleJob status.ansibleJobResult 를 확인하는 1분 루프에서 조정 요청을 다시 대기열에 넣습니다. prehook status.ansibleJobResult.status 가 성공 하면 애플리케이션 서브스크립션이 계속 기본 서브스크립션을 배포합니다.

1.5.6.5.2. Posthook

앱 서브스크립션 상태가 업데이트되면 서브스크립션 상태가 서브스크립션되거나 서브스크립션 상태의 모든 대상 클러스터에 전파되는 경우 앱 서브스크립션 컨트롤러는 posthook AnsibleJob 오브젝트로 모든 AnsibleJob Kind CR을 검색합니다. 그런 다음 새 posthook AnsibleJob 인스턴스를 생성합니다. 새 인스턴스 이름은 posthook AnsibleJob 오브젝트 이름과 임의의 접미사 문자열입니다.

인스턴스 이름 예: service-ticket-1-2913849 를 참조하십시오.

1.5.6.5.3. Ansible 배치 규칙

유효한 prehook AnsibleJob을 사용하면 배포 규칙 결정과 관계없이 사전hook AnsibleJob이 시작됩니다. 예를 들어 배치 규칙 서브스크립션을 전파하지 못한 prehook AnsibleJob이 있을 수 있습니다. 배치 규칙 결정이 변경되면 새로운 prehook 및 posthook AnsibleJob 인스턴스가 생성됩니다.

1.5.6.6. Ansible 구성

다음 작업을 사용하여 **Ansible Tower** 구성을 구성할 수 있습니다.

1.5.6.6.1. Ansible 시크릿

동일한 서브스크립션 네임스페이스에 **Ansible Tower** 시크릿 **CR**을 생성해야 합니다. **Ansible Tower** 시크릿은 동일한 서브스크립션 네임스페이스로 제한됩니다.

Ansible Tower 시크릿 이름 섹션을 작성하여 콘솔에서 시크릿 을 생성합니다. 터미널을 사용하여 시크릿을 생성하려면 다음 **yaml** 을 편집하고 적용합니다.

다음 명령을 실행하여 **YAML** 파일을 추가합니다.

```
oc apply -f
```

다음 **YAML** 샘플을 참조하십시오.

참고: 네임스페이스 는 서브스크립션 네임스페이스와 동일한 네임스페이스입니다. **stringData:token** 및 **host** 는 **Ansible Tower**에서 가져온 것입니다.

```
apiVersion: v1
kind: Secret
metadata:
  name: toweraccess
  namespace: same-as-subscription
type: Opaque
stringData:
  token: ansible-tower-api-token
  host: https://ansible-tower-host-url
```

앱 서브스크립션 컨트롤러에서 **prehook** 및 **posthook AnsibleJobs**를 생성할 때 서브스크립션 **spec.hooksecretref** 의 시크릿을 사용할 수 있는 경우 **AnsibleJob CR spec.tower_auth_secret** 으로 전송되고 **AnsibleJob**은 **Ansible Tower**에 액세스할 수 있습니다.

1.5.6.7. 보안 조정 설정

prehook 및 **posthook AnsibleJobs**가 있는 기본 서브스크립션의 경우 모든 **prehook** 및 **posthook AnsibleJobs** 또는 기본 서브스크립션이 **Git** 리포지토리에 업데이트된 후 **main-sub** 서브스크립션을 조

정해야 합니다.

***Prehook AnsibleJobs** 및 기본 서브스크립션은 **AnsibleJob** 이전 인스턴스를 지속적으로 조정하고 다시 시작합니다.*

1. ***AnsibleJob** 전이 완료되면 기본 서브스크립션을 다시 실행합니다.*

2. *기본 서브스크립션에 사양이 변경되면 서브스크립션을 다시 배포합니다. 재배포 프로세스에 맞게 기본 서브스크립션 상태를 업데이트해야 합니다.*

3. ***hub** 서브스크립션 상태를 **nil** 로 재설정합니다. 대상 클러스터의 서브스크립션 배포와 함께 서브스크립션이 새로 고쳐집니다.*

   *대상 클러스터에서 배포가 완료되면 대상 클러스터의 서브스크립션 상태가 "서브스크립션" 또는 " 실패" 로 업데이트되고 **hub** 클러스터 서브스크립션 상태와 동기화됩니다.*

4. *기본 서브스크립션이 완료되면 새 **post-AnsibleJob** 인스턴스를 다시 시작합니다.*

5. ***DONE** 서브스크립션이 업데이트되었는지 확인합니다. 다음 출력을 참조하십시오.*

   - ***subscription.status == "subscribed"***
   - ***subscription.status ==** 모든 대상 클러스터 "서브스크립션"*

***AnsibleJob CR**이 생성되면 대상 **Ansible Tower**와 통신하여 **Ansible Tower** 작업을 시작하기 위해 **Kubernetes** 작업 **CR**이 생성됩니다. 작업이 완료되면 작업의 최종 상태가 **AnsibleJob status.ansibleJobResult** 로 반환됩니다.*

참고:

***AnsibleJob status.conditions**는 **Kubernetes** 작업 결과 생성을 저장하기 위해 **Ansible** 작업 운영자가 예약합니다. **status.conditions**는 실제 **Ansible Tower** 작업 상태를 반영하지 않습니다.*

서브스크립션 컨트롤러는 **AnsibleJob.status.conditions** 대신 **AnsibleJob.status.ansibleJobResult** 에서 **Ansible Tower** 작업 상태를 확인합니다.

prehook 및 **posthook AnsibleJob** 워크플로우에서 언급한 것처럼 기본 서브스크립션이 **Git** 리포지토리에서 업데이트되면 새로운 **prehook** 및 **posthook AnsibleJob** 인스턴스가 생성됩니다. 따라서 하나의 기본 서브스크립션을 통해 여러 **AnsibleJob** 인스턴스에 연결할 수 있습니다.

subscription.status.ansibleJobs에 네 개의 필드가 정의됩니다.

- **lastPrehookJobs:** 최신 **prehook AnsibleJobs**
- **prehookJobsHistory:** 모든 **prehook AnsibleJobs history**
- **lastPosthookJobs:** 최신 **posthook AnsibleJobs**
- **posthookJobsHistory:** 모든 **posthook AnsibleJobs history**

1.5.6.8. Ansible 샘플 YAML

Git **prehook** 및 **posthook** 폴더에 있는 **AnsibleJob .yaml** 파일의 다음 샘플을 참조하십시오.

```
apiVersion: tower.ansible.com/v1alpha1
kind: AnsibleJob
metadata:
  name: demo-job-001
  namespace: default
spec:
  tower_auth_secret: toweraccess
  job_template_name: Demo Job Template
  extra_vars:
    cost: 6.88
    ghosts: ["inky","pinky","clyde","sue"]
    is_enable: false
    other_variable: foo
    pacman: mrs
    size: 8
    targets_list:
    - aaa
```

```
- bbb
- ccc
version: 1.23.45
```

1.5.7. OpenShift GitOps Operator용 Managed Clusters 구성

GitOps를 구성하려면 Kubernetes 관리 클러스터의 Red Hat Advanced Cluster Management 세트를 Red Hat OpenShift Container Platform GitOps Operator 인스턴스에 하나 이상 등록할 수 있습니다. 등록 후 해당 클러스터에 애플리케이션을 배포할 수 있습니다. 개발, 스테이징 및 프로덕션 환경의 클러스터 전체에서 애플리케이션 일관성을 자동화하도록 연속 GitOps 환경을 설정합니다.

1.5.7.1. 사전 요구 사항

1. Kubernetes용 Red Hat Advanced Cluster Management에 Red Hat OpenShift GitOps Operator 를 설치해야 합니다.

2. 하나 이상의 관리 클러스터를 가져옵니다.

1.5.7.2. 관리형 클러스터를 GitOps에 등록

1. 관리형 클러스터 세트를 생성하고 관리 대상 클러스터 세트에 관리 클러스터를 추가합니다. multicloud-integrations managedclusterset 에서 관리형 클러스터 세트의 예를 참조하십시오.

   자세한 내용은 ManagedClusterSets 생성 및 관리 설명서를 참조하십시오.

2. Red Hat OpenShift Container Platform GitOps가 배포된 네임스페이스에 대한 관리형 클러스터 세트 바인딩 을 생성합니다.

   openshift-gitops 네임스페이스에 바인딩되는 multicloud-integrations managedclustersetbinding 의 리포지토리 예제를 참조하십시오.

   자세한 내용은 ManagedClusterSetBinding 리소스 생성 설명서를 참조하십시오.

3. 관리형 클러스터 세트 바인딩에 사용되는 네임스페이스에서 배치 사용자 정의 리소스를 생성하여 OpenShift Container Platform GitOps Operator 인스턴스에 등록할 관리형 클러스터 세트를 선택합니다. 다중 클라우드 통합 배치에서 리포지터리의 예제를 사용할 수 있습니다.

   배치 정보는 ManagedClusterSets를 배치와 함께 사용을 참조하십시오.

참고: OpenShift Container Platform 클러스터만 다른 Kubernetes 클러스터가 아닌 Red Hat OpenShift Container Platform GitOps Operator 인스턴스에 등록됩니다.

4. GitOpsCluster 사용자 정의 리소스를 생성하여 배치 결정에 따라 관리되는 클러스터 세트를 Red Hat OpenShift Container Platform GitOps의 지정된 인스턴스에 등록합니다. 이를 통해 Red Hat OpenShift Container Platform GitOps 인스턴스를 통해 Red Hat Advanced Cluster Management 관리형 클러스터에 애플리케이션을 배포할 수 있습니다.

multicloud-integrations gitops 클러스터 의 리포지토리의 예제를 사용합니다.

참고: 참조된 배치 리소스는 GitOpsCluster 리소스와 동일한 네임스페이스에 있어야 합니다.

다음 샘플에서 placementRef.name 이 all-openshift-clusters 임을 확인하고 argoNamespace: openshift-gitops 에 설치된 GitOps 인스턴스의 대상 클러스터로 지정됩니다. argoServer.cluster 사양에는 local-cluster 값이 필요합니다.

```
apiVersion: apps.open-cluster-management.io/v1beta1
kind: GitOpsCluster
metadata:
  name: gitops-cluster-sample
  namespace: dev
spec:
  argoServer:
    cluster: local-cluster
    argoNamespace: openshift-gitops
  placementRef:
    kind: Placement
    apiVersion: cluster.open-cluster-management.io/v1beta1
    name: all-openshift-clusters
```

5. 변경 사항을 저장하십시오. 이제 GitOps 워크플로를 따라 애플리케이션을 관리할 수 있습니다. 자세한 내용은 GitOps 정보를 참조하십시오.

1.5.7.3. GitOps 토큰

placement 및 ManagedClusterSetBinding 사용자 정의 리소스를 통해 GitOps 네임스페이스에 바인딩된 모든 관리 클러스터에 대해 GitOps Operator와 통합할 때 네임스페이스에 ManagedCluster 에 액세스할 토큰이 있는 시크릿이 생성됩니다. GitOps 컨트롤러에서 관리 클러스터에 리소스를 동기화하는 데 필요합니다. 애플리케이션 라이프사이클 작업을 수행하기 위해 GitOps 네임스페이스에 대한 관리자 액세스 권한이 부여되면 사용자는 이 시크릿 및 관리 수준에 대한 액세스 권한도 부여됩니다.

이를 원하지 않는 경우 사용자를 네임스페이스 범위 **admin** 역할에 바인딩하는 대신 생성 및 사용자를 바인딩하는 데 사용할 수 있는 애플리케이션 리소스와 함께 더 제한적인 사용자 정의 역할을 사용합니다. 다음 **ClusterRole** 예제를 참조하십시오.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: application-set-admin
rules:
- apiGroups:
  - argoproj.io
  resources:
  - applicationsets
  verbs:
  - get
  - list
  - watch
  - update
  - delete
  - deletecollection
  - patch
```

1.5.8. 배포 예약

특정 시간에만 새로운 또는 **Helm** 차트 또는 기타 리소스를 배포해야 하는 경우 해당 리소스에 대한 서브스크립션을 정의하여 해당 특정 시간 동안만 배포를 시작할 수 있습니다. 또는 배포를 제한할 수 있습니다.

예를 들어 매주 금요일 오후 **10**시부터 오후 **11**시 사이에 시간 창을 정의하여 클러스터에 패치 또는 기타 애플리케이션 업데이트를 적용하기 위한 예정된 유지 관리 기간으로 사용할 수 있습니다.

최대 작업 시간 동안 예기치 않은 배포를 방지하기 위해 특정 시간 창에서 배포를 시작하거나 차단할 수 있습니다. 예를 들어 최대 시간을 방지하기 위해 서브스크립션 기간을 정의하여 오전 **8**시부터 오전 **8**시부터 오전 **8**시부터 오후 **8**시 사이에 배포 시작을 방지할 수 있습니다.

서브스크립션의 시간 창을 정의하면 모든 애플리케이션 및 클러스터에 대한 업데이트를 조정할 수 있습니다. 예를 들어 서브스크립션을 정의하여 오후 **6**시 ~ 오후 **11**시 **59**분 사이에 새 애플리케이션 리소스만 배포하고 다른 서브스크립션을 정의하여 기존 리소스의 업데이트된 버전만 오전 **12**시에서 **7**시 **59**분 사이로 배포할 수 있습니다.

서브스크립션에 대한 시간 창이 정의되면 서브스크립션이 활성으로 변경될 때의 시간 범위입니다. 시간 창을 정의하는 과정의 일부로 해당 창 중에 활성화 또는 차단 할 서브스크립션을 정의할 수 있습니다.

새 리소스 또는 변경된 리소스의 배포는 서브스크립션이 활성화된 경우에만 시작됩니다. 서브스크립션이 활성이든 차단되었는지 여부에 관계없이 서브스크립션은 새 리소스 또는 변경된 리소스에 대해 계속 모니터링합니다. 활성 및 차단 설정은 배포에만 영향을 미칩니다.

새 리소스 또는 변경된 리소스가 감지되면 시간 창 정의에 따라 서브스크립션에 대한 다음 작업이 결정됩니다.

- ***HelmRepo,ObjectBucket** 및 **Git** 유형 채널에 대한 서브스크립션의 경우:*
- *서브스크립션이 활성 상태일 때 시간 범위 중에 리소스가 감지되면 리소스 배포가 시작됩니다.*
- *서브스크립션이 배포 실행을 차단할 때 리소스가 시간 범위 외부에서 감지되면 리소스 배포 요청이 캐시됩니다. 서브스크립션이 활성화된 다음 시간 범위가 발생하면 캐시된 요청이 적용되고 모든 관련 배포가 시작됩니다.*
- *시간 창이 차단 되면 애플리케이션 서브스크립션에 의해 이전에 배포된 모든 리소스는 그대로 유지됩니다. 시간 창이 다시 활성화될 때까지 새 업데이트가 차단됩니다.*

최종 사용자는 앱 하위 시간 창이 차단될 때 배포된 모든 리소스가 삭제될 때 잘못 생각할 수 있습니다. 앱 하위 시간 창이 다시 활성화되면 다시 표시됩니다.

배포가 정의된 기간 동안 시작되고 정의된 기간이 종료될 때 실행되는 경우 배포가 계속 완료될 때까지 계속 실행됩니다.

*서브스크립션의 시간 창을 정의하려면 필요한 필드와 값을 서브스크립션 리소스 정의 **YAML**에 추가해야 합니다.*

- *시간 창을 정의하는 과정의 일부로 시간 창의 일 및 시간을 정의할 수 있습니다.*
- *또한 시간 창 유형을 정의하여 배포가 정의된 시간 프레임 동안 또는 외부에서 발생할 수 있는 기간을 결정할 수 있습니다.*
- *시간 창 유형이 활성 인 경우 정의된 기간 동안만 배포를 시작할 수 있습니다. 특정 유지 관리*

기간 내에만 배포를 수행하려는 경우 이 설정을 사용할 수 있습니다.

- 시간 창 유형이 **block** 인 경우 배포는 정의된 시간 프레임 동안 시작할 수 없지만 다른 시점에서 시작할 수 있습니다. 중요한 업데이트가 필요한 경우 이 설정을 사용할 수 있지만 특정 시간 범위 동안 배포하지 않도록 해야 합니다. 예를 들어 이 유형을 사용하여 오전 **10**시부터 오후 **2**시를 제외하고 보안 관련 업데이트를 언제든지 적용할 수 있도록 시간 창을 정의할 수 있습니다.

- 월요일과 수요일마다 시간 창을 정의하기 위해 서브스크립션의 여러 시간 창을 정의할 수 있습니다.

1.5.9. 패키지 덮어쓰기 구성

서브스크립션을 통해 구독한 **Helm** 차트 또는 **Kubernetes** 리소스의 서브스크립션 덮어쓰기 값에 대한 패키지 덮어쓰기를 구성합니다.

패키지 덮어쓰기를 구성하려면 **Kubernetes** 리소스 사양 내의 필드를 경로 필드의 값으로 덮어쓸 필드를 지정합니다. 교체 값을 **value** 필드의 값으로 지정합니다.

예를 들어 서브스크립션된 **Helm** 차트의 **Helm** 릴리스의 **spec** 내의 **values** 필드를 재정의해야 하는 경우 서브스크립션 정의의 **path** 필드 값을 **spec** 으로 설정해야 합니다.

```
packageOverrides:
- packageName: nginx-ingress
  packageOverrides:
  - path: spec
    value: my-override-values
```

value 필드의 콘텐츠는 **Helm** 사양의 **spec** 필드 내의 값을 재정의하는 데 사용됩니다.

- **Helm** 릴리스의 경우 **spec** 필드의 덮어쓰기 값이 **Helm** 릴리스 **values.yaml** 파일에 병합되어 기존 값을 덮어씁니다. 이 파일은 **Helm** 릴리스에 대한 구성 가능한 변수를 검색하는 데 사용됩니다.

- **Helm** 릴리스의 릴리스 이름을 재정의해야 하는 경우 **packageOverride** 섹션을 정의에 포함합니다. 다음 필드를 포함하여 **Helm** 릴리스의 **packageAlias** 를 정의합니다.

  - **Helm** 차트 를 식별하는 **PACKAGENAME.**

- ***packageAlias*** *는 릴리스 이름을 재정의하고 있음을 나타냅니다.*

  기본적으로 ***Helm*** *릴리스 이름이 지정되지 않은 경우 릴리스를 식별하는 데* ***Helm*** *차트 이름이 사용됩니다. 동일한 차트에 가입된 여러 릴리스가 있는 경우와 같이 충돌이 발생할 수 있습니다. 릴리스 이름은 네임스페이스 내의 서브스크립션 중에서 고유해야 합니다. 생성하는 서브스크립션의 릴리스 이름이 고유하지 않으면 오류가 발생합니다.* ***packageOverride*** *를 정의하여 서브스크립션에 대해 다른 릴리스 이름을 설정해야 합니다. 기존 서브스크립션 내에서 이름을 변경하려면 먼저 해당 서브스크립션을 삭제한 다음 기본 릴리스 이름으로 서브스크립션을 다시 생성해야 합니다.*

```
packageOverrides:
- packageName: nginx-ingress
  packageAlias: my-helm-release-name
```

1.5.10. 채널 샘플 개요

파일을 빌드하는 데 사용할 수 있는 샘플 및 ***YAML*** *정의를 확인합니다. 채널(****channel.apps.open-cluster-management.io****)은* ***Kubernetes*** *애플리케이션용* ***Red Hat Advanced Cluster Management****를 생성하고 관리하기 위한 지속적인 통합 및 연속 제공 기능을 제공합니다.*

OpenShift CLI *툴을 사용하려면 다음 절차를 참조하십시오.*

a. *원하는 편집 툴을 사용하여 애플리케이션* ***YAML*** *파일을 작성하고 저장합니다.*

b. *다음 명령을 실행하여 파일을* ***API*** *서버에 적용합니다. 파일 이름을 파일 이름으로 바꿉니다.*

```
oc apply -f filename.yaml
```

c. *다음 명령을 실행하여 애플리케이션 리소스가 생성되었는지 확인합니다.*

```
oc get application.app
```

- *채널* ***YAML*** *구조*
- *채널* ***YAML*** *테이블*

- *오브젝트 스토리지 버킷(ObjectBucket) 채널*
- *Helm 리포지터리(HelmRepo) 채널*
- *Git(Git) 리포지토리 채널*

1.5.10.1. 채널 YAML 구조

배포할 수 있는 애플리케이션 샘플은 stolostron 리포지토리를 참조하십시오.

다음 YAML 구조에는 채널의 필수 필드와 일부 공통 선택적 필드가 표시되어 있습니다. YAML 구조에는 몇 가지 필수 필드 및 값이 포함되어야 합니다. 애플리케이션 관리 요구 사항에 따라 다른 선택적 필드 및 값을 포함해야 할 수 있습니다. 모든 툴 및 제품 콘솔에서 고유한 YAML 콘텐츠를 작성할 수 있습니다.

```
apiVersion: apps.open-cluster-management.io/v1
kind: Channel
metadata:
  name:
  namespace: # Each channel needs a unique namespace, except Git channel.
spec:
  sourceNamespaces:
  type:
  pathname:
  secretRef:
    name:
  gates:
    annotations:
  labels:
```

1.5.10.2. 채널 YAML 테이블

필드	설명
apiVersion	필수 항목입니다. 값을 **apps.open-cluster-management.io/v1** 로 설정합니다.
kind	필수 항목입니다. 리소스가 채널임을 나타내기 위해 값을 **Channel** 로 설정합니다.
metadata.name	필수 항목입니다. 채널의 이름입니다.

필드	설명
metadata.namespace	필수 항목입니다. 채널의 네임스페이스입니다. 각 채널에는 Git 채널을 제외하고 고유한 네임스페이스가 필요합니다.
spec.sourceNamespaces	선택 사항: 채널 컨트롤러에서 채널을 검색하고 승격하기 위해 새 배포 또는 업데이트된 배포 가능 항목을 모니터링하는 네임스페이스를 식별합니다.
spec.type	필수 항목입니다. 채널 유형입니다. 지원되는 유형은 **HelmRepo**,**Git**, **ObjectBucket** (console의 오브젝트 스토리지)입니다.
spec.pathname	**HelmRepo**,**Git**,**ObjectBucket** 채널에 필요합니다. **HelmRepo** 채널의 경우 값을 Helm 리포지토리의 URL로 설정합니다. **ObjectBucket** 채널의 경우 해당 값을 오브젝트 스토리지의 URL로 설정합니다. **Git** 채널의 경우 해당 값을 Git 리포지토리의 HTTPS URL로 설정합니다.
spec.secretRef.name	선택 사항: 리포지토리 또는 차트에 액세스하는 것과 같이 인증에 사용할 Kubernetes Secret 리소스를 식별합니다. **HelmRepo**,**ObjectBucket** 및 **Git** 유형 채널만 사용한 인증에 보안을 사용할 수 있습니다.
spec.gates	선택 사항: 채널 내에서 배포 가능을 승격하기 위한 요구사항을 정의합니다. 요구 사항이 설정되지 않은 경우 채널 네임스페이스 또는 소스에 추가된 배포 가능이 채널로 승격됩니다. **게이트** 값은 **ObjectBucket** 채널 유형에만 적용되며 **HelmRepo** 및 **Git** 채널 유형에는 적용되지 않습니다.
spec.gates.annotations	선택 사항: 채널의 주석입니다. deployables에는 채널에 포함할 일치하는 주석이 있어야 합니다.
metadata.labels	선택 사항: 채널의 라벨입니다.
spec.insecureSkipVerify	선택 사항: 기본값은 **false** 입니다. **true** 로 설정된 경우 인증을 건너뛰어 채널 연결이 빌드됩니다.

*채널 정의 구조는 다음 **YAML** 콘텐츠와 유사합니다.*

```
apiVersion: apps.open-cluster-management.io/v1
kind: Channel
metadata:
  name: predev-ch
  namespace: ns-ch
  labels:
```

```
    app: nginx-app-details
spec:
  type: HelmRepo
  pathname: https://kubernetes-charts.storage.googleapis.com/
```

1.5.10.3. 오브젝트 스토리지 버킷(ObjectBucket) 채널

다음 예제 채널 정의는 **Object** 스토리지 버킷을 채널로 추상화합니다.

```
apiVersion: apps.open-cluster-management.io/v1
kind: Channel
metadata:
 name: dev
 namespace: ch-obj
spec:
 type: ObjectBucket
 pathname: [http://9.28.236.243:xxxx/dev] # URL is appended with the valid bucket name,
which matches the channel name.
 secretRef:
   name: miniosecret
 gates:
   annotations:
     dev-ready: true
```

1.5.10.4. Helm 리포지터리(HelmRepo) 채널

다음 예제 채널 정의에서는 **Helm** 리포지터리를 채널로 추상화합니다.

사용 중단 알림: 2.5의 경우 **Helm** 리포지터리 **SSL** 인증서를 건너뛰도록 채널 **ConfigMap** 참조에 **insecureSkipVerify: "true"** 를 지정하면 더 이상 사용되지 않습니다. 대신 채널에서 사용되는 **spec.insecureSkipVerify: true** 를 사용하여 다음 현재 샘플의 교체를 참조하십시오.

```
apiVersion: v1
kind: Namespace
metadata:
  name: hub-repo
---
apiVersion: apps.open-cluster-management.io/v1
kind: Channel
metadata:
  name: Helm
  namespace: hub-repo
spec:
    pathname: [https://9.21.107.150:8443/helm-repo/charts] # URL points to a valid chart URL.
    insecureSkipVerify: true
    type: HelmRepo
```

다음 채널 정의는 **Helm** 리포지터리 채널의 또 다른 예를 보여줍니다.

참고: **Helm** 차트에 포함된 모든 **Kubernetes** 리소스에는 애플리케이션 토폴로지가 올바르게 표시하려면 레이블 릴리스 **{{ .Release.Name }}** 가 있어야 합니다.

```
apiVersion: apps.open-cluster-management.io/v1
kind: Channel
metadata:
  name: predev-ch
  namespace: ns-ch
  labels:
    app: nginx-app-details
spec:
  type: HelmRepo
  pathname: https://kubernetes-charts.storage.googleapis.com/
```

1.5.10.5. Git(Git) 리포지토리 채널

다음 채널 정의에서는 **Git** 리포지토리의 채널 예를 보여줍니다. 다음 예에서 **secretRef** 는 경로 이름에 지정된 **Git** 리포지터리에 액세스하는 데 사용되는 사용자 **ID**를 나타냅니다. 공용 리포지터리가 있는 경우 **secretRef** 레이블 및 값이 필요하지 않습니다.

```
apiVersion: apps.open-cluster-management.io/v1
kind: Channel
metadata:
  name: hive-cluster-gitrepo
  namespace: gitops-cluster-lifecycle
spec:
  type: Git
  pathname: https://github.com/open-cluster-management/gitops-clusters.git
  secretRef:
    name: github-gitops-clusters
---
apiVersion: v1
kind: Secret
metadata:
  name: github-gitops-clusters
  namespace: gitops-cluster-lifecycle
data:
  user: dXNlcgo=          # Value of user and accessToken is Base 64 coded.
  accessToken: cGFzc3dvcmQ
```

1.5.11. 서브스크립션 샘플 개요

파일을 빌드하는 데 사용할 수 있는 샘플 및 **YAML** 정의를 확인합니다. 채널과 마찬가지로 서브스크립션(**subscription.apps.open-cluster-management.io**)은 애플리케이션 관리를 위한 지속적인 통합 및 지

속적인 제공 기능을 제공합니다.

OpenShift CLI 툴을 사용하려면 다음 절차를 참조하십시오.

a. 원하는 편집 툴을 사용하여 애플리케이션 YAML 파일을 작성하고 저장합니다.

b. 다음 명령을 실행하여 파일을 api 서버에 적용합니다. 파일 이름을 파일 이름으로 바꿉니다.

```
oc apply -f filename.yaml
```

c. 다음 명령을 실행하여 애플리케이션 리소스가 생성되었는지 확인합니다.

```
oc get application.app
```

- 서브스크립션 YAML 구조
- 서브스크립션 YAML 테이블
- 서브스크립션 파일 샘플
  - 서브스크립션 시간 창 예
  - 덮어쓰기가 포함된 서브스크립션 예
  - Helm 리포지터리 서브스크립션 예
  - Git 리포지토리 서브스크립션 예

1.5.11.1. 서브스크립션 YAML 구조

다음 YAML 구조는 서브스크립션 및 일부 공통 선택적 필드의 필수 필드를 보여줍니다. YAML 구조에는 특정 필수 필드 및 값이 포함되어야 합니다.

애플리케이션 관리 요구 사항에 따라 다른 선택적 필드 및 값을 포함해야 할 수 있습니다. 모든 툴을 사용하여 자체 YAML 콘텐츠를 작성할 수 있습니다.

```
apiVersion: apps.open-cluster-management.io/v1
kind: Subscription
metadata:
  name:
  namespace:
  labels:
spec:
  sourceNamespace:
  source:
  channel:
  name:
  packageFilter:
    version:
    labelSelector:
      matchLabels:
        package:
        component:
    annotations:
  packageOverrides:
  - packageName:
    packageAlias:
    - path:
      value:
  placement:
    local:
    clusters:
      name:
    clusterSelector:
    placementRef:
      name:
      kind: PlacementRule
  overrides:
    clusterName:
    clusterOverrides:
      path:
      value:
```

1.5.11.2. 서브스크립션 YAML 테이블

필드	설명
apiVersion	필수 항목입니다. 값을 **apps.open-cluster-management.io/v1** 로 설정합니다.
kind	필수 항목입니다. 리소스가 서브스크립션임을 나타내기 위해 값을 **Subscription** 으로 설정합니다.

필드	설명
metadata.name	필수 항목입니다. 서브스크립션을 식별하기 위한 이름입니다.
metadata.namespace	필수 항목입니다. 서브스크립션에 사용할 네임스페이스 리소스입니다.
metadata.labels	선택 사항: 서브스크립션의 라벨입니다.
spec.channel	선택 사항: 서브스크립션 채널을 정의하는 네임스페이스 이름("Namespace/Name")입니다. **채널** 또는 소스 또는 source 또는 **source Namespace** 필드를 정의합니다. 일반적으로 **source** 또는 **sourceNamespace** 필드를 사용하는 대신 **channel** 필드를 사용하여 채널을 가리킵니다. 둘 이상의 필드가 정의되면 정의된 첫 번째 필드가 사용됩니다.
spec.sourceNamespace	선택 사항: 배포 대상이 hub 클러스터에 저장되는 소스 네임스페이스입니다. 이 필드는 네임스페이스 채널에만 사용합니다. **채널** 또는 소스 또는 source 또는 **source Namespace** 필드를 정의합니다. 일반적으로 **source** 또는 **sourceNamespace** 필드를 사용하는 대신 **channel** 필드를 사용하여 채널을 가리킵니다.
spec.source	선택 사항: 배포 대상이 저장되는 Helm 리포지토리의 경로 이름("URL")입니다. Helm 리포지터리 채널에만 이 필드를 사용합니다. **채널** 또는 소스 또는 source 또는 **source Namespace** 필드를 정의합니다. 일반적으로 **source** 또는 **sourceNamespace** 필드를 사용하는 대신 **channel** 필드를 사용하여 채널을 가리킵니다.
spec.name	**HelmRepo** 유형 채널에 필요하지만 **ObjectBucket** 유형 채널에는 선택 사항입니다. 대상 Helm 차트의 특정 이름 또는 채널 내에서 배포할 수 있습니다. 필드가 선택적 채널 유형에 대해 **name** 또는 **packageFilter** 를 정의하지 않으면 모든 배포 가능 항목을 찾고 각 배포 가능 항목의 최신 버전이 검색됩니다.
spec.packageFilter	선택 사항: 대상 배포 가능 항목을 찾는 데 사용할 매개변수 또는 배포 가능 항목의 하위 집합을 정의합니다. 여러 필터 조건을 정의하는 경우 배포 가능은 모든 필터 조건을 충족해야 합니다.
spec.packageFilter.version	선택 사항: 배포 가능한 버전 또는 버전입니다. 버전 범위는 **> 1.0 , 또는 < 3.0** 에서 사용할 수 있습니다. 기본적으로 가장 최근의 "creationTimestamp" 값이 있는 버전이 사용됩니다.
spec.packageFilter.annotations	선택 사항: 배포 가능한 주석입니다.

필드	설명
spec.packageOverrides	선택 사항: 채널 내의 Helm 차트, 배포 가능 또는 기타 Kubernetes 리소스와 같이 서브스크립션을 통해 구독하는 Kubernetes 리소스에 대한 덮어쓰기를 정의하는 섹션입니다.
spec.packageOverrides.packageName	선택 사항이지만 재정의 설정에 필요합니다. 덮어쓰는 Kubernetes 리소스를 식별합니다.
spec.packageOverrides.packageAlias	선택 사항: 덮어쓰는 Kubernetes 리소스에 별칭을 지정합니다.
spec.packageOverrides.packageOverrides	선택 사항: Kubernetes 리소스를 재정의하는 데 사용할 매개변수 및 대체 값의 구성입니다.
spec.placement	필수 항목입니다. 배포 가능 항목을 배치해야 하는 서브스크립션 클러스터 또는 클러스터를 정의하는 배치 규칙을 식별합니다. 배치 구성을 사용하여 다중 클러스터 배포 값을 정의합니다.
spec.placement.local	선택 사항이지만 직접 관리하려는 독립 실행형 클러스터 또는 클러스터에 필요합니다. 서브스크립션을 로컬에 배포해야 하는지 여부를 정의합니다. 구독이 지정된 채널과 동기화되도록 값을 **true** 로 설정합니다. 서브스크립션이 지정된 채널에서 모든 리소스에 가입하지 못하도록 하려면 값을 **false** 로 설정합니다. 클러스터가 독립형 클러스터이거나 이 클러스터를 직접 관리하는 경우 이 필드를 사용합니다. 클러스터가 멀티 클러스터의 일부이고 클러스터를 직접 관리하지 않으려면 클러스터 ,**clusterSelector** 또는 **placementRef** 중 하나만 사용하여 서브스크립션을 배치할 위치를 정의합니다. **클러스터가 멀티 클러스터의 Hub이고 클러스터를 직접 관리하려면 서브스크립션 운영자가 리소스에 로컬에 가입하기 전에 Hub를 관리형 클러스터로 등록해야 합니다.**
spec.placement.clusters	선택 사항: 서브스크립션을 배치할 클러스터를 정의합니다. 클러스터 **, clusterSelector** 또는 **placementRef** 중 하나만 다중 클러스터에 대한 서브스크립션을 배치할 위치를 정의하는 데 사용됩니다. 클러스터가 허브 클러스터가 아닌 독립형 클러스터인 경우 **로컬 클러스터를** 사용할 수도 있습니다.
spec.placement.clusters.name	선택 사항이지만 서브스크립션 클러스터를 정의하는 데 필요합니다. 서브스크립션 클러스터의 이름 또는 이름입니다.

필드	설명
spec.placement.clusterSelector	선택 사항: 서브스크립션을 배치할 클러스터를 식별하는 데 사용할 라벨 선택기를 정의합니다. 클러스터 , **cluster Selector** 또는 **placementRef** 중 하나만 사용하여 멀티 클러스터의 서브스크립션을 배치할 위치를 정의합니다. 클러스터가 허브 클러스터가 아닌 독립형 클러스터인 경우 **로컬 클러스터를** 사용할 수도 있습니다.
spec.placement.placementRef	선택 사항: 서브스크립션에 사용할 배치 규칙을 정의합니다. 클러스터 , **cluster Selector** 또는 **placementRef** 중 하나만 사용하여 멀티 클러스터의 서브스크립션을 배치할 위치를 정의합니다. 클러스터가 Hub 클러스터가 아닌 독립형 클러스터인 경우 **로컬 클러스터를** 사용할 수도 있습니다.
spec.placement.placementRef.name	선택 사항이지만 배치 규칙을 사용하는 데 필요합니다. 서브스크립션에 대한 배치 규칙의 이름입니다.
spec.placement.placementRef.kind	선택 사항이지만 배치 규칙을 사용하는 데 필요합니다. 값을 **PlacementRule** 으로 설정하여 서브스크립션을 사용한 배포에 배치 규칙이 사용됨을 나타냅니다.
spec.overrides	선택 사항: 클러스터별 설정과 같이 재정의해야 하는 매개변수 및 값.
spec.overrides.clusterName	선택 사항: 매개변수와 값이 재정의되는 클러스터 또는 클러스터의 이름입니다.
spec.overrides.clusterOverrides	선택 사항: 재정의할 매개변수 및 값의 구성입니다.
spec.timeWindow	선택 사항: 구독이 활성 또는 차단될 때 시간 창을 구성하기 위한 설정을 정의합니다.
spec.timeWindow.type	선택 사항이지만 시간 창을 구성하는 데 필요합니다. 구성된 기간 동안 구독이 활성 상태인지 차단되는지 여부를 나타냅니다.Indicates whether the subscription is active or blocked during the configured time window. 서브스크립션에 대한 배포는 서브스크립션이 활성화된 경우에만 수행됩니다.
spec.timeWindow.location	선택 사항이지만 시간 창을 구성하는 데 필요합니다. 시간 창에 대해 구성된 시간 범위의 시간대입니다. 모든 시간대는 Time Zone (tz) 데이터베이스 이름 형식을 사용해야 합니다. 자세한 내용은 시간대 데이터베이스 을 참조하십시오.

필드	설명
spec.timeWindow.daysofweek	선택 사항이지만 시간 창을 구성하는 데 필요합니다. 시간 범위를 사용하여 시간 창을 만드는 데 적용되는 요일을 나타냅니다.Indicates the days of the week when the time range is applied to create a time window. 일 목록은 **요일: ["Monday", "Wednesday", "Friday"]** 와 같은 배열로 정의해야 합니다.
spec.timeWindow.hours	선택 사항이지만 시간 창을 구성하는 데 필요합니다. 시간 범위를 정의합니다. 시간 범위의 시작 시간과 종료 시간은 각 시간 창에 대해 정의해야 합니다. 서브스크립션에 대해 여러 시간 창 범위를 정의할 수 있습니다.
spec.timeWindow.hours.start	선택 사항이지만 시간 창을 구성하는 데 필요합니다. 시간 창의 시작을 정의하는 타임 스탬프입니다. 타임 스탬프는 Go 프로그래밍 언어 Kitchen 형식 **"hh:mmpm"** 을 사용해야 합니다. 자세한 내용은 Constants 를 참조하십시오.
spec.timeWindow.hours.end	선택 사항이지만 시간 창을 구성하는 데 필요합니다. 시간 창의 끝을 정의하는 타임 스탬프입니다. 타임 스탬프는 Go 프로그래밍 언어 Kitchen 형식 **"hh:mmpm"** 을 사용해야 합니다. 자세한 내용은 Constants 를 참조하십시오.

참고:

- *YAML을 정의할 때 서브스크립션은 **packageFilters** 를 사용하여 여러 **Helm** 차트, 배포 가능 또는 기타 **Kubernetes** 리소스를 가리킬 수 있습니다. 그러나 서브스크립션은 하나의 차트의 최신 버전 또는 배포 가능 또는 기타 리소스만 배포합니다.*

- *시간 **Windows**의 경우 창의 시간 범위를 정의할 때 시작 시간을 종료 시간 전에 발생하도록 설정해야 합니다. 서브스크립션에 대해 여러 시간 창을 정의하는 경우 창의 시간 범위가 중복될 수 없습니다. 실제 시간 범위는 **subscription-controller** 컨테이너 시간을 기반으로 하며, 이 컨테이너는 작업 중인 시간 및 위치와 다른 시간 및 위치로 설정할 수 있습니다.*

- *서브스크립션 사양에서 **Helm** 릴리스 배치를 서브스크립션 정의의 일부로 정의할 수도 있습니다. 각 서브스크립션은 기존 배치 규칙을 참조하거나 서브스크립션 정의 내에서 직접 배치 규칙을 정의할 수 있습니다.*

- *spec.placement 섹션에 서브스크립션을 배치할 위치를 정의하는 경우 다중 클러스터 환경에 클러스터 ,**clusterSelector** 또는 **placementRef** 중 하나만 사용하십시오.*

- 둘 이상의 배치 설정을 포함하는 경우 하나의 설정이 사용되며 다른 설정이 무시됩니다. 다음 우선순위는 서브스크립션 운영자가 사용하는 설정을 결정하는 데 사용됩니다.

  a. **placementRef**

  b. 클러스터

  c. **clusterSelector**

서브스크립션은 다음 **YAML** 콘텐츠와 유사합니다.

```
apiVersion: apps.open-cluster-management.io/v1
kind: Subscription
metadata:
  name: nginx
  namespace: ns-sub-1
  labels:
    app: nginx-app-details
spec:
  channel: ns-ch/predev-ch
  name: nginx-ingress
  packageFilter:
    version: "1.36.x"
  placement:
    placementRef:
      kind: PlacementRule
      name: towhichcluster
  overrides:
  - clusterName: "/"
    clusterOverrides:
    - path: "metadata.namespace"
      value: default
```

1.5.11.3. 서브스크립션 파일 샘플

배포할 수 있는 애플리케이션 샘플은 **stolostron** 리포지토리를 참조하십시오.

```
apiVersion: apps.open-cluster-management.io/v1
kind: Subscription
metadata:
  name: nginx
  namespace: ns-sub-1
  labels:
```

```
    app: nginx-app-details
spec:
  channel: ns-ch/predev-ch
  name: nginx-ingress
```

1.5.11.4. 보조 채널 샘플

미러링된 채널(애플리케이션 소스 저장소)이 있는 경우 서브스크립션 **YAML**에 **secondaryChannel**을 지정할 수 있습니다. 애플리케이션 서브스크립션이 기본 채널을 사용하여 리포지토리 서버에 연결하지 못하면 보조 채널을 사용하여 리포지토리 서버에 연결합니다. 보조 채널에 저장된 애플리케이션 매니페스트가 기본 채널과 동기화되었는지 확인합니다. **secondaryChannel**을 사용하여 다음 샘플 서브스크립션 **YAML**을 참조하십시오.

```
apiVersion: apps.open-cluster-management.io/v1
kind: Subscription
metadata:
  name: nginx
  namespace: ns-sub-1
  labels:
    app: nginx-app-details
spec:
  channel: ns-ch/predev-ch
  secondaryChannel: ns-ch-2/predev-ch-2
  name: nginx-ingress
```

1.5.11.4.1. 서브스크립션 시간 창 예

다음 예제 서브스크립션에는 구성된 여러 시간 창이 포함되어 있습니다. 시간 창은 매주 월요일, 수요일, 금요일 오전 **10**시~오후 **10**시 사이에 발생합니다. 시간 창은 월요일, 수요일, 금요일에 따라 오후 **12**시 ~ 오후 **1**시~오후 **1**시 사이에 발생합니다. 서브스크립션은 배포를 시작하기 위한 **6**일 동안의 기간 동안에만 활성화됩니다.

```
apiVersion: apps.open-cluster-management.io/v1
kind: Subscription
metadata:
  name: nginx
  namespace: ns-sub-1
  labels:
    app: nginx-app-details
spec:
  channel: ns-ch/predev-ch
  name: nginx-ingress
  packageFilter:
    version: "1.36.x"
  placement:
    placementRef:
      kind: PlacementRule
      name: towhichcluster
  timewindow:
    windowtype: "active" #Enter active or blocked depending on the purpose of the type.
```

```
    location: "America/Los_Angeles"
    daysofweek: ["Monday", "Wednesday", "Friday"]
    hours:
      - start: "10:20AM"
        end: "10:30AM"
      - start: "12:40PM"
        end: "1:40PM"
```

1.5.11.4.2. 덮어쓰기가 포함된 서브스크립션 예

다음 예제에는 Helm 차트에 대한 Helm 릴리스의 다른 릴리스 이름을 정의하는 패키지 덮어쓰기가 포함되어 있습니다. 패키지 덮어쓰기 설정은 my-nginx-ingress-releaseName 이름을 nginx-ingress Helm 릴리스의 다른 릴리스 이름으로 설정하는 데 사용됩니다.

```
apiVersion: apps.open-cluster-management.io/v1
kind: Subscription
metadata:
  name: simple
  namespace: default
spec:
  channel: ns-ch/predev-ch
  name: nginx-ingress
  packageOverrides:
  - packageName: nginx-ingress
    packageAlias: my-nginx-ingress-releaseName
    packageOverrides:
    - path: spec
      value:
        defaultBackend:
          replicaCount: 3
  placement:
    local: false
```

1.5.11.4.3. Helm 리포지터리 서브스크립션 예

다음 서브스크립션은 1.36.x 버전의 최신 nginx Helm 릴리스를 자동으로 가져옵니다. Helm 릴리스 배포 가능은 소스 Helm 리포지터리에서 새 버전을 사용할 수 있는 경우 my-development-cluster-1 클러스터에 배치됩니다.

spec.packageOverrides 섹션에는 Helm 릴리스의 값을 재정의하는 데 필요한 선택적 매개변수가 표시되어 있습니다. 덮어쓰기 값은 Helm 릴리스 values.yaml 파일에 병합되며 Helm 릴리스의 구성 가능한 변수를 검색하는 데 사용됩니다.

```
apiVersion: apps.open-cluster-management.io/v1
kind: Subscription
metadata:
  name: nginx
  namespace: ns-sub-1
  labels:
```

```
    app: nginx-app-details
spec:
  channel: ns-ch/predev-ch
  name: nginx-ingress
  packageFilter:
    version: "1.36.x"
  placement:
    clusters:
    - name: my-development-cluster-1
  packageOverrides:
  - packageName: my-server-integration-prod
    packageOverrides:
    - path: spec
      value:
        persistence:
          enabled: false
          useDynamicProvisioning: false
        license: accept
        tls:
          hostname: my-mcm-cluster.icp
        sso:
          registrationImage:
            pullSecret: hub-repo-docker-secret
```

1.5.11.4.4. Git 리포지토리 서브스크립션 예

1.5.11.4.4.1. Git 리포지토리의 특정 분기 및 디렉터리 구독

```
apiVersion: apps.open-cluster-management.io/v1
kind: Subscription
metadata:
  name: sample-subscription
  namespace: default
  annotations:
    apps.open-cluster-management.io/git-path: sample_app_1/dir1
    apps.open-cluster-management.io/git-branch: branch1
spec:
  channel: default/sample-channel
  placement:
    placementRef:
      kind: PlacementRule
      name: dev-clusters
```

이 예제 서브스크립션에서 주석 **apps.open-cluster-management.io/git-path** 는 서브스크립션이 채널에 지정된 **Git** 리포지토리의 **sample_app_1/dir1** 디렉터리 내의 모든 **Helm** 차트 및 **Kubernetes** 리소스에 가입함을 나타냅니다. 서브스크립션은 기본적으로 **master** 분기를 서브스크립션합니다. 이 예제 서브스크립션에서는 주석 **apps.open-cluster-management.io/git-branch: branch1** 이 지정되어 리포지토리의 **branch1** 분기를 서브스크립션합니다.

참고:

- *Helm 차트를 구독하는 Git 채널 서브스크립션을 사용하는 경우 리소스 토폴로지 보기에 추가 Helmrelease 리소스가 표시될 수 있습니다. 이 리소스는 내부 애플리케이션 관리 리소스이며 무시해도 됩니다.*

1.5.11.4.4.2. .kubernetesignore 파일 추가

Git 리포지토리 루트 디렉터리에 .kubernetesignore 파일을 포함하거나 서브스크립션 주석에 지정된 apps.open-cluster-management.io/git-path 디렉터리 내에 포함할 수 있습니다.

이 .kubernetesignore 파일을 사용하면 파일 또는 하위 디렉터리의 패턴 또는 두 가지 모두의 패턴을 지정하여 서브스크립션이 리포지터리에서 Kubernetes 리소스 또는 Helm 차트를 배포할 때 무시할 수 있습니다.

세분화된 필터링에 .kubernetesignore 파일을 사용하여 Kubernetes 리소스를 선택적으로 적용할 수도 있습니다. .kubernetesignore 파일의 패턴 형식은 .gitignore 파일과 동일합니다.

apps.open-cluster-management.io/git-path 주석이 정의되지 않은 경우 구독에서 리포지토리 루트 디렉터리에서 .kubernetesignore 파일을 찾습니다. apps.open-cluster-management.io/git-path 필드가 정의된 경우 서브스크립션은 apps.open-cluster-management.io/git-path 디렉터리에서 .kubernetesignore 파일을 찾습니다. 서브스크립션은 다른 디렉터리에서 .kubernetesignore 파일을 검색하지 않습니다.

1.5.11.4.4.3. Kustomize 적용

서브스크립션된 Git 폴더에 kustomization.yaml 또는 kustomization.yml 파일이 있는 경우 kustomize가 적용됩니다. spec.packageOverrides 를 사용하여 서브스크립션 배포 시 kustomization 을 덮어쓸 수 있습니다.

```
apiVersion: apps.open-cluster-management.io/v1
kind: Subscription
metadata:
  name: example-subscription
  namespace: default
spec:
  channel: some/channel
  packageOverrides:
  - packageName: kustomization
    packageOverrides:
    - value: |
patchesStrategicMerge:
- patch.yaml
```

kustomization.yaml 파일을 재정의하려면 packageOverrides 에 packageName: kustomization 이 필요합니다. 덮어쓰기는 새 항목을 추가하거나 기존 항목을 업데이트합니다. 기존 항목이 제거되지 않습니다.

1.5.11.4.4.4. Git WebHook 활성화

기본적으로 Git 채널 서브스크립션에서는 1분마다 채널에 지정된 Git 리포지토리를 복제하고 커밋 ID가 변경될 때 변경 사항을 적용합니다. 또는 Git 리포지토리에서 리포지터리 PUSH 및 PULL 웹 후크 이벤트 알림을 보내는 경우에만 변경 사항을 적용하도록 서브스크립션을 구성할 수 있습니다.

Git 리포지토리에 Webhook를 구성하려면 대상 Webhook 페이로드 URL과 필요한 경우 시크릿이 필요합니다.

1.5.11.4.4.4.1. 페이로드 URL

hub 클러스터에 경로(ingress)를 생성하여 서브스크립션 Operator의 웹 후크 이벤트 리스너 서비스를 노출합니다.

```
oc create route passthrough --service=multicluster-operators-subscription -n open-cluster-management
```

그런 다음 oc get route multicluster-operators-subscription -n open-cluster-management 명령을 사용하여 외부에서 연결할 수 있는 호스트 이름을 찾습니다.

Webhook 페이로드 URL은 https://<externally-reachable hostname>/webhook 입니다.

1.5.11.4.4.4.2. Webhook 보안

Webhook 보안은 선택 사항입니다. 채널 네임스페이스에 Kubernetes 시크릿을 생성합니다. 시크릿에는 data.secret 이 포함되어야 합니다.

다음 예제를 참조하십시오.

```
apiVersion: v1
kind: Secret
metadata:
  name: my-github-webhook-secret
data:
  secret: BASE64_ENCODED_SECRET
```

data.secret 의 값은 사용할 base-64 인코딩 WebHook 시크릿입니다.

모범 사례: 각 Git 리포지토리에 고유한 시크릿을 사용합니다.

1.5.11.4.4.4.3. Git 리포지토리에서 WebHook 구성

페이로드 URL 및 Webhook 시크릿을 사용하여 Git 리포지토리에 WebHook을 구성합니다.

1.5.11.4.4.4.4. 채널에서 WebHook 이벤트 알림 활성화

서브스크립션 채널에 주석을 답니다. 다음 예제를 참조하십시오.

```
oc annotate channel.apps.open-cluster-management.io <channel name> apps.open-cluster-management.io/webhook-enabled="true"
```

시크릿을 사용하여 WebHook을 구성한 경우 이를 사용하여 채널에 주석을 답니다. 여기서 < the_secret_name >은 Webhook 시크릿이 포함된 kubernetes 시크릿 이름입니다.

```
oc annotate channel.apps.open-cluster-management.io <channel name> apps.open-cluster-management.io/webhook-secret="<the_secret_name>"
```

1.5.11.4.4.4.5. Webhook 지원 채널 서브스크립션

서브스크립션에 Webhook별 구성이 필요하지 않습니다.

1.5.12. 배치 규칙 샘플 개요

배치 규칙(placementrule.apps.open-cluster-management.io)은 배포 가능 클러스터를 배포할 수 있는 대상 클러스터를 정의합니다. 배치 규칙을 사용하면 배포 가능한 다중 클러스터를 쉽게 배포할 수 있습니다.

OpenShift CLI 툴을 사용하려면 다음 절차를 참조하십시오.

a. *원하는 편집 툴을 사용하여 애플리케이션 YAML 파일을 작성하고 저장합니다.*

b. 다음 명령을 실행하여 파일을 **API** 서버에 적용합니다. 파일 이름을 파일 이름으로 바꿉니다.

```
oc apply -f filename.yaml
```

c. 다음 명령을 실행하여 애플리케이션 리소스가 생성되었는지 확인합니다.

```
oc get application.app
```

- 배치 규칙 **YAML** 구조
- 배치 규칙 **YAML** 값 테이블
- 배치 규칙 샘플 파일

1.5.12.1. 배치 규칙 YAML 구조

다음 **YAML** 구조는 배치 규칙 및 일부 공통 선택적 필드에 필요한 필드를 보여줍니다. **YAML** 구조에는 몇 가지 필수 필드 및 값이 포함되어야 합니다. 애플리케이션 관리 요구 사항에 따라 다른 선택적 필드 및 값을 포함해야 할 수 있습니다. 모든 툴 및 제품 콘솔에서 고유한 **YAML** 콘텐츠를 작성할 수 있습니다.

```
apiVersion: apps.open-cluster-management.io/v1
kind: PlacementRule
metadata:
  name:
  namespace:
  resourceVersion:
  labels:
    app:
    chart:
    release:
    heritage:
  selfLink:
  uid:
spec:
  clusterSelector:
    matchLabels:
      datacenter:
      environment:
  clusterReplicas:
  clusterConditions:
  ResourceHint:
```

```
  type:
  order:
Policies:
```

1.5.12.2. 배치 규칙 YAML 값 테이블

필드	설명
apiVersion	필수 항목입니다. 값을 **apps.open-cluster-management.io/v1** 로 설정합니다.
kind	필수 항목입니다. 값을 **PlacementRule** 으로 설정하여 리소스가 배치 규칙임을 나타냅니다.
metadata.name	필수 항목입니다. 배치 규칙을 식별하는 이름입니다.
metadata.namespace	필수 항목입니다. 배치 규칙에 사용할 네임스페이스 리소스입니다.
metadata.resourceVersion	선택 사항: 배치 규칙 리소스의 버전입니다.
metadata.labels	선택 사항: 배치 규칙의 레이블입니다.
spec.clusterSelector	선택 사항: 대상 클러스터를 식별하기 위한 라벨
spec.clusterSelector.matchLabels	선택 사항: 대상 클러스터에 존재해야 하는 레이블입니다.
spec.clusterSelector.matchExpressions	선택 사항: 대상 클러스터에 존재해야 하는 레이블입니다.
status.decisions	선택 사항: 배포 가능 항목이 배치되는 대상 클러스터를 정의합니다.
status.decisions.clusterName	선택 사항: 대상 클러스터의 이름
status.decisions.clusterNamespace	선택 사항: 대상 클러스터의 네임스페이스입니다.
spec.clusterReplicas	선택 사항: 생성할 복제본 수입니다.
spec.clusterConditions	선택 사항: 클러스터의 조건을 정의합니다.
spec.ResourceHint	선택 사항: 이전 필드에서 제공한 라벨 및 값과 둘 이상의 클러스터가 일치하는 경우, 클러스터를 선택하는 리소스별 기준을 지정할 수 있습니다. 예를 들어 가장 많은 CPU 코어를 사용하여 클러스터를 선택할 수 있습니다.

필드	설명
spec.ResourceHint.type	선택 사항: 사용 가능한 메모리 리소스에 따라 클러스터를 선택하려면 **cpu** 를 cpu로 설정하여 사용 가능한 CPU 코어 또는 **메모리** 를 기반으로 클러스터를 선택합니다.
spec.ResourceHint.order	선택 사항: 해당 값을 asc에 대해 **asc** 로 설정하거나, order에 대해 **desc** for desc를 설정합니다.
spec.Policies	선택 사항: 배치 규칙에 대한 정책 필터입니다.

1.5.12.3. 배치 규칙 샘플 파일

*배포할 수 있는 애플리케이션 샘플은 **stolostron** 리포지토리를 참조하십시오.*

*기존 배치 규칙에는 배치 규칙의 상태를 나타내는 다음 필드가 포함될 수 있습니다. 이 상태 섹션은 규칙에 대한 **YAML** 구조의 **spec** 섹션 뒤에 추가됩니다.*

```
status:
  decisions:
    clusterName:
    clusterNamespace:
```

필드	설명
status	배치 규칙의 상태 정보입니다.
status.decisions	배포 가능 항목이 배치되는 대상 클러스터를 정의합니다.
status.decisions.clusterName	대상 클러스터의 이름
status.decisions.clusterNamespace	대상 클러스터의 네임스페이스입니다.

- *예시 1*

```
apiVersion: apps.open-cluster-management.io/v1
kind: PlacementRule
metadata:
  name: gbapp-gbapp
```

```
  namespace: development
  labels:
    app: gbapp
spec:
  clusterSelector:
    matchLabels:
      environment: Dev
  clusterReplicas: 1
status:
  decisions:
    - clusterName: local-cluster
      clusterNamespace: local-cluster
```

- 예시 2

```
apiVersion: apps.open-cluster-management.io/v1
kind: PlacementRule
metadata:
  name: towhichcluster
  namespace: ns-sub-1
  labels:
    app: nginx-app-details
spec:
  clusterReplicas: 1
  clusterConditions:
    - type: ManagedClusterConditionAvailable
      status: "True"
  clusterSelector:
    matchExpressions:
    - key: environment
      operator: In
      values:
      - dev
```

1.5.13. 애플리케이션 샘플

파일을 빌드하는 데 사용할 수 있는 샘플 및 YAML 정의를 확인합니다. Kubernetes용 Red Hat Advanced Cluster Management의 애플리케이션(Application.app.k8s.io)은 애플리케이션 구성 요소를 확인하는 데 사용됩니다.

OpenShift CLI 툴을 사용하려면 다음 절차를 참조하십시오.

a. 원하는 편집 툴을 사용하여 애플리케이션 YAML 파일을 작성하고 저장합니다.

b. 다음 명령을 실행하여 파일을 API 서버에 적용합니다. 파일 이름을 파일 이름으로 바꿉니다.

```
oc apply -f filename.yaml
```

c. 다음 명령을 실행하여 애플리케이션 리소스가 생성되었는지 확인합니다.

```
oc get application.app
```

- 애플리케이션 YAML 구조
- 애플리케이션 YAML 테이블
- 애플리케이션 파일 샘플

1.5.13.1. 애플리케이션 YAML 구조

애플리케이션 리소스를 생성하거나 업데이트하기 위해 애플리케이션 정의 YAML 콘텐츠를 작성하려면 YAML 구조에 몇 가지 필수 필드 및 값을 포함해야 합니다. 애플리케이션 요구 사항 또는 애플리케이션 관리 요구 사항에 따라 다른 선택적 필드 및 값을 포함해야 할 수 있습니다.

다음 YAML 구조는 애플리케이션 및 일부 공통 선택적 필드에 대한 필수 필드를 보여줍니다.

```
apiVersion: app.k8s.io/v1beta1
kind: Application
metadata:
  name:
  namespace:
spec:
  selector:
    matchLabels:
      label_name: label_value
```

1.5.13.2. 애플리케이션 YAML 테이블

필드	현재의	설명
apiVersion	**app.k8s.io/v1beta1**	필수 항목
kind	**애플리케이션**	필수 항목
metadata		

필드	현재의	설명
	name: 애플리케이션 리소스를 식별하는 이름입니다.	필수 항목
	namespace: 애플리케이션에 사용할 네임스페이스 리소스입니다.	
spec		
selector.matchLabels	이 애플리케이션이 연결된 서브스크립션 또는 서브스크립션에 있는 Kubernetes 레이블 및 값인 **key:value** 쌍입니다. 레이블을 사용하면 애플리케이션 리소스에서 레이블 이름과 값과 일치하여 관련 서브스크립션을 찾을 수 있습니다.	필수 항목

*이러한 애플리케이션을 정의하는 사양은 **Kubernetes SIG(Special Interest Group)**에서 제공하는 애플리케이션 메타데이터 설명자 사용자 정의 리소스 정의를 기반으로 합니다. 표에 표시된 값만 필요합니다.*

*이 정의를 사용하면 자체 애플리케이션 **YAML** 콘텐츠를 구성하는 데 도움이 될 수 있습니다. 이 정의에 대한 자세한 내용은 **Kubernetes SIG Application CRD** 커뮤니티 사양 을 참조하십시오.*

1.5.13.3. 애플리케이션 파일 샘플

*배포할 수 있는 애플리케이션 샘플은 **stolostron** 리포지토리를 참조하십시오.*

*애플리케이션의 정의 구조는 다음 예제 **YAML** 콘텐츠와 유사합니다.*

```
apiVersion: app.k8s.io/v1beta1
kind: Application
metadata:
  name: my-application
  namespace: my-namespace
spec:
  selector:
    matchLabels:
      my-label: my-label-value
```