Red Hat OpenShift GitOps 1.13

보안

여기에 간단한 설명을 입력합니다.

Last Updated: 2024-12-10

Red Hat OpenShift GitOps 1.13 보안

여기에 간단한 설명을 입력합니다.

법적 공지

초록

이 책의 주제와 목적에 대한 간략한 개요와 요약, 일반적으로 하나 이상의 단락이 없습니다.

차례

1장. REDIS를 사용하여 보안 통신 구성

Red Hat OpenShift GitOps와 함께 TLS(Transport Layer Security) 암호화를 사용하면 Argo CD 구성 요소와 Redis 캐시 간의 통신을 보호하고 전송 시 중요한 데이터를 보호할 수 있습니다.

다음 구성 중 하나를 사용하여 Redis와의 통신을 보호할 수 있습니다.

- **autotls** 설정을 활성화하여 TLS 암호화에 적절한 인증서를 발급합니다.
- 키 및 인증서 쌍으로 **argocd-operator-redis-tls** 시크릿을 생성하여 TLS 암호화를 수동으로 구성합니다.

두 구성 모두 HA(고가용성)를 활성화하거나 사용하지 않고 사용할 수 있습니다.

1.1. 사전 요구 사항

- **cluster-admin** 권한이 있는 클러스터에 액세스할 수 있습니다.
- OpenShift Container Platform 웹 콘솔에 액세스할 수 있습니다.
- Red Hat OpenShift GitOps Operator가 클러스터에 설치되어 있습니다.

1.2. AUTOTLS가 활성화된 REDIS의 TLS 구성

새 또는 기존 Argo CD 인스턴스에서 **autotls** 설정을 활성화하여 Redis에 대한 TLS 암호화를 구성할 수 있습니다. 이 구성은 **argocd-operator-redis-tls** 시크릿을 자동으로 프로비저닝하고 추가 단계가 필요하지 않습니다. 현재 OpenShift Container Platform은 지원되는 유일한 시크릿 공급자입니다.

참고

기본적으로 **autotls** 설정은 비활성화되어 있습니다.

프로세스

1. OpenShift Container Platform 웹 콘솔에 로그인합니다.
2. **autotls** 가 활성화된 Argo CD 인스턴스를 생성합니다.
   a. 웹 콘솔의 **관리자** 화면에서 왼쪽 탐색 패널을 사용하여 **Administration → CustomResourceDefinitions** 로 이동합니다.
   b. **argocds.argoproj.io** 를 검색하고 **ArgoCD** CRD(사용자 정의 리소스 정의)를 클릭합니다.
   c. **CustomResourceDefinition 세부 정보** 페이지에서 **Instances** 탭을 클릭한 다음 **Create ArgoCD** 를 클릭합니다.
   d. 다음 예와 유사한 YAML을 편집하거나 교체합니다.

autotls가 활성화된 Argo CD CR의 예

```
apiVersion: argoproj.io/v1beta1
kind: ArgoCD
metadata:
  name: argocd (1)
```

```
  namespace: openshift-gitops (2)
spec:
  redis:
    autotls: openshift (3)
  ha:
    enabled: true (4)
```

(1) Argo CD 인스턴스의 이름입니다.

(2) Argo CD 인스턴스를 실행할 네임스페이스입니다.

(3) **autotls** 설정을 활성화하고 Redis에 대한 TLS 인증서를 생성하는 플래그입니다.

(4) HA 기능을 활성화하는 플래그 값입니다. HA를 활성화하지 않으려면 이 행을 포함하지 않거나 플래그 값을 **false** 로 설정합니다.

작은 정보

또는 다음 명령을 실행하여 기존 Argo CD 인스턴스에서 **autotls** 설정을 활성화할 수 있습니다.

```
$ oc patch argocds.argoproj.io <instance-name> --type=merge -p '{"spec":{"redis":
{"autotls":"openshift"}}}'
```

e. **생성**을 클릭합니다.

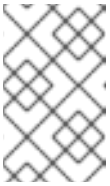
f. Argo CD Pod가 준비되어 실행 중인지 확인합니다.

```
$ oc get pods -n <namespace> (1)
```

(1) Argo CD 인스턴스가 실행 중인 네임스페이스를 지정합니다(예: **openshift-gitops**).

HA가 비활성화된 출력 예

```
NAME                                READY   STATUS    RESTARTS   AGE
argocd-application-controller-0     1/1     Running   0          26s
argocd-redis-84b77d4f58-vp6zm       1/1     Running   0          37s
argocd-repo-server-5b959b57f4-znxjq 1/1     Running   0          37s
argocd-server-6b8787d686-wv9zh      1/1     Running   0          37s
```



참고

HA 지원 TLS 구성에는 작업자 노드가 3개 이상인 클러스터가 필요합니다. HA 구성으로 Argo CD 인스턴스를 활성화한 경우 출력이 표시되는 데 몇 분이 걸릴 수 있습니다.

HA가 활성화된 출력 예

```
NAME                                READY   STATUS    RESTARTS   AGE
argocd-application-controller-0     1/1     Running   0          10m
```

```
argocd-redis-ha-haproxy-669757fdb7-5xg8h   1/1     Running   0          10m
argocd-redis-ha-server-0                   2/2     Running   0          9m9s
argocd-redis-ha-server-1                   2/2     Running   0          98s
argocd-redis-ha-server-2                   2/2     Running   0          53s
argocd-repo-server-576499d46d-8hgbh        1/1     Running   0          10m
argocd-server-9486f88b7-dk2ks              1/1     Running   0          10m
```

3. **argocd-operator-redis-tls** 보안이 생성되었는지 확인합니다.

```
$ oc get secrets argocd-operator-redis-tls -n <namespace> (1)
```

(1) Argo CD 인스턴스가 실행 중인 네임스페이스를 지정합니다(예: **openshift-gitops**).

출력 예

```
NAME                       TYPE                DATA   AGE
argocd-operator-redis-tls  kubernetes.io/tls   2      30s
```

시크릿은 **kubernetes.io/tls** 유형이어야 하며 크기는 **2** 여야 합니다.

1.3. AUTOTLS가 비활성화된 상태로 REDIS의 TLS 구성

키 및 인증서 쌍으로 **argocd-operator-redis-tls** 시크릿을 생성하여 Redis에 대한 TLS 암호화를 수동으로 구성할 수 있습니다. 또한 적절한 Argo CD 인스턴스에 속해 있음을 나타내려면 시크릿에 주석을 달아야 합니다. 인증서 및 보안을 생성하는 단계는 HA(고가용성)가 활성화된 인스턴스에 따라 다릅니다.

프로세스

1. OpenShift Container Platform 웹 콘솔에 로그인합니다.
2. Argo CD 인스턴스를 생성합니다.
   a. 웹 콘솔의 **관리자** 화면에서 왼쪽 탐색 패널을 사용하여 **Administration → CustomResourceDefinitions** 로 이동합니다.
   b. **argocds.argoproj.io** 를 검색하고 **ArgoCD** CRD(사용자 정의 리소스 정의)를 클릭합니다.
   c. **CustomResourceDefinition 세부 정보** 페이지에서 **Instances** 탭을 클릭한 다음 **Create ArgoCD** 를 클릭합니다.
   d. 다음 예와 유사한 YAML을 편집하거나 교체합니다.

autotls가 비활성화된 ArgoCD CR의 예

```
apiVersion: argoproj.io/v1beta1
kind: ArgoCD
metadata:
  name: argocd (1)
  namespace: openshift-gitops (2)
spec:
  ha:
    enabled: true (3)
```

1. Argo CD 인스턴스의 이름입니다.
2. Argo CD 인스턴스를 실행할 네임스페이스입니다.
3. HA 기능을 활성화하는 플래그 값입니다. HA를 활성화하지 않으려면 이 행을 포함하지 않거나 플래그 값을 **false** 로 설정합니다.

e. **생성**을 클릭합니다.

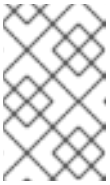
f. Argo CD Pod가 준비되어 실행 중인지 확인합니다.

```
$ oc get pods -n <namespace> 1
```

1. Argo CD 인스턴스가 실행 중인 네임스페이스를 지정합니다(예: **openshift-gitops**).

HA가 비활성화된 출력 예

```
NAME                                READY   STATUS    RESTARTS   AGE
argocd-application-controller-0     1/1     Running   0          26s
argocd-redis-84b77d4f58-vp6zm       1/1     Running   0          37s
argocd-repo-server-5b959b57f4-znxjq 1/1     Running   0          37s
argocd-server-6b8787d686-wv9zh      1/1     Running   0          37s
```



참고

HA 지원 TLS 구성에는 작업자 노드가 3개 이상인 클러스터가 필요합니다. HA 구성으로 Argo CD 인스턴스를 활성화한 경우 출력이 표시되는 데 몇 분이 걸릴 수 있습니다.

HA가 활성화된 출력 예

```
NAME                                          READY   STATUS    RESTARTS   AGE
argocd-application-controller-0               1/1     Running   0          10m
argocd-redis-ha-haproxy-669757fdb7-5xg8h      1/1     Running   0          10m
argocd-redis-ha-server-0                      2/2     Running   0          9m9s
argocd-redis-ha-server-1                      2/2     Running   0          98s
argocd-redis-ha-server-2                      2/2     Running   0          53s
argocd-repo-server-576499d46d-8hgbh           1/1     Running   0          10m
argocd-server-9486f88b7-dk2ks                 1/1     Running   0          10m
```

3. HA 구성에 따라 다음 옵션 중 하나를 사용하여 Redis 서버에 대한 자체 서명 인증서를 생성합니다.

- HA가 비활성화된 Argo CD 인스턴스의 경우 다음 명령을 실행합니다.

```
$ openssl req -new -x509 -sha256 \
  -subj "/C=XX/ST=XX/O=Testing/CN=redis" \
  -reqexts SAN -extensions SAN \
  -config <(printf "\n[SAN]\nsubjectAltName=DNS:argocd-redis.
<namespace>.svc.cluster.local\n[req]\ndistinguished_name=req") \ 1
  -keyout /tmp/redis.key \
  -out /tmp/redis.crt \
```

```
  -newkey rsa:4096 \
  -nodes \
  -sha256 \
  -days 10
```

1 Argo CD 인스턴스가 실행 중인 네임스페이스를 지정합니다(예: **openshift-gitops**).

출력 예

```
Generating a RSA private key
..............++++
.........................++++
writing new private key to '/tmp/redis.key'
```

- HA가 활성화된 Argo CD 인스턴스의 경우 다음 명령을 실행합니다.

```
$ openssl req -new -x509 -sha256 \
  -subj "/C=XX/ST=XX/O=Testing/CN=redis" \
  -reqexts SAN -extensions SAN \
  -config <(printf "\n[SAN]\nsubjectAltName=DNS:argocd-redis-ha-haproxy.
<namespace>.svc.cluster.local\n[req]\ndistinguished_name=req") \ 1
  -keyout /tmp/redis-ha.key \
  -out /tmp/redis-ha.crt \
  -newkey rsa:4096 \
  -nodes \
  -sha256 \
  -days 10
```

1 Argo CD 인스턴스가 실행 중인 네임스페이스를 지정합니다(예: **openshift-gitops**).

출력 예

```
Generating a RSA private key
..............++++
.........................++++
writing new private key to '/tmp/redis-ha.key'
```

4. 다음 명령을 실행하여 생성된 인증서 및 키를 **/tmp** 디렉토리에서 사용할 수 있는지 확인합니다.

```
$ cd /tmp
```

```
$ ls
```

HA가 비활성화된 출력 예

```
...
redis.crt
redis.key
...
```

HA가 활성화된 출력 예

```
...
redis-ha.crt
redis-ha.key
...
```

5. HA 구성에 따라 다음 옵션 중 하나를 사용하여 **argocd-operator-redis-tls** 시크릿을 생성합니다.
   - HA가 비활성화된 Argo CD 인스턴스의 경우 다음 명령을 실행합니다.

     ```
     $ oc create secret tls argocd-operator-redis-tls --key=/tmp/redis.key --cert=/tmp/redis.crt
     ```

   - HA가 활성화된 Argo CD 인스턴스의 경우 다음 명령을 실행합니다.

     ```
     $ oc create secret tls argocd-operator-redis-tls --key=/tmp/redis-ha.key --cert=/tmp/redis-ha.crt
     ```

     출력 예

     ```
     secret/argocd-operator-redis-tls created
     ```

6. Argo CD CR에 속함을 나타내기 위해 보안에 주석을 답니다.

   ```
   $ oc annotate secret argocd-operator-redis-tls argocds.argoproj.io/name=<instance-name> 1
   ```

   1 Argo CD 인스턴스의 이름을 지정합니다(예: **argocd**).

   출력 예

   ```
   secret/argocd-operator-redis-tls annotated
   ```

7. Argo CD Pod가 준비되어 실행 중인지 확인합니다.

   ```
   $ oc get pods -n <namespace> 1
   ```

   1 Argo CD 인스턴스가 실행 중인 네임스페이스를 지정합니다(예: **openshift-gitops**).

   HA가 비활성화된 출력 예

   ```
   NAME                                READY   STATUS    RESTARTS   AGE
   argocd-application-controller-0     1/1     Running   0          26s
   argocd-redis-84b77d4f58-vp6zm       1/1     Running   0          37s
   argocd-repo-server-5b959b57f4-znxjq 1/1     Running   0          37s
   argocd-server-6b8787d686-wv9zh      1/1     Running   0          37s
   ```

참고

HA 구성으로 Argo CD 인스턴스를 활성화한 경우 출력이 표시되는 데 몇 분이 걸릴 수 있습니다.

HA가 활성화된 출력 예

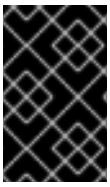
```
NAME                                READY   STATUS    RESTARTS   AGE
argocd-application-controller-0     1/1     Running   0          10m
argocd-redis-ha-haproxy-669757fdb7-5xg8h   1/1     Running   0          10m
argocd-redis-ha-server-0            2/2     Running   0          9m9s
argocd-redis-ha-server-1            2/2     Running   0          98s
argocd-redis-ha-server-2            2/2     Running   0          53s
argocd-repo-server-576499d46d-8hgbh        1/1     Running   0          10m
argocd-server-9486f88b7-dk2ks       1/1     Running   0          10m
```

2장. GITOPS와 함께 SECRETS STORE CSI 드라이버를 사용하여 안전하게 보안 관리

이 가이드에서는 OpenShift Container Platform 4.14 이상의 GitOps Operator와 SSCSI(Secrets Store Container Storage Interface) 드라이버를 통합하는 프로세스를 안내합니다.

2.1. GITOPS와 함께 SECRETS STORE CSI 드라이버를 사용하여 보안 관리 개요

일부 애플리케이션에는 암호 및 사용자 이름과 같은 민감한 정보가 필요합니다. 이 정보는 적절한 보안 관행으로 숨겨져 있어야 합니다. RBAC(역할 기반 액세스 제어)가 클러스터에 제대로 구성되지 않았기 때문에 중요한 정보가 노출되면 API 또는 etcd 액세스 권한이 있는 모든 사용자가 시크릿을 검색하거나 수정할 수 있습니다.



중요

네임스페이스에서 Pod를 생성할 권한이 있는 모든 사용자는 해당 RBAC를 사용하여 해당 네임스페이스의 보안을 읽을 수 있습니다. SSCSI Driver Operator를 사용하면 외부 시크릿 저장소를 사용하여 중요한 정보를 안전하게 저장하고 Pod에 제공할 수 있습니다.

OpenShift Container Platform SSCSI 드라이버를 GitOps Operator와 통합하는 프로세스는 다음 절차로 구성됩니다.

1. GitOps 리포지토리에 AWS Secrets Manager 리소스 저장
2. AWS Secrets Manager에서 시크릿을 마운트하도록 SSCSI 드라이버 구성
3. 마운트된 보안을 사용하도록 GitOps 관리 리소스 구성

2.1.1. 혜택

SSCSI 드라이버를 GitOps Operator와 통합하면 다음과 같은 이점이 있습니다.

- GitOps 워크플로우의 보안 및 효율성 향상
- 배포 Pod에 대한 보안 보안 연결을 볼륨으로 용이하게 합니다.
- 민감한 정보에 안전하고 효율적으로 액세스 할 수 있도록

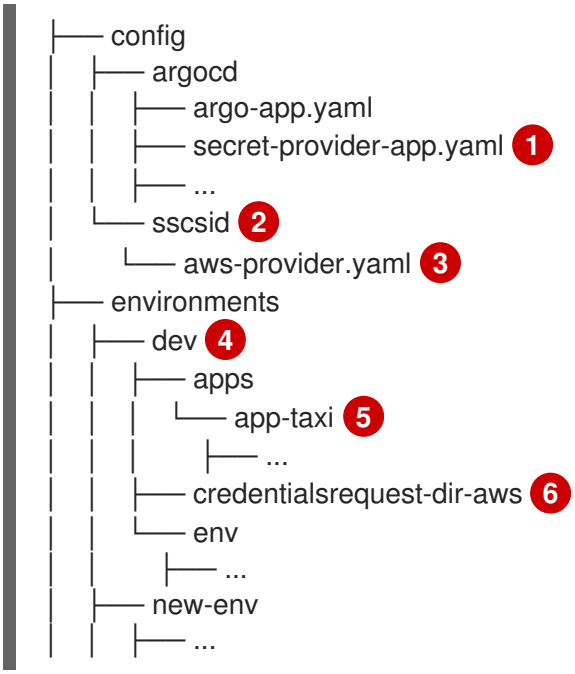
2.1.2. 보안 저장소 공급자

Secrets Store CSI Driver Operator와 함께 다음 보안 저장소 공급자를 사용할 수 있습니다.

- AWS Secrets Manager
- AWS Systems Manager 매개변수 저장소
- Microsoft Azure Key Vault

예를 들어 SSCSI Driver Operator가 있는 시크릿 저장소 공급자로 AWS Secrets Manager를 사용하는 것이 좋습니다. 다음 예제에서는 AWS Secrets Manager의 시크릿을 사용할 준비가 된 GitOps 리포지토리의 디렉터리 구조를 보여줍니다.

GitOps 리포지토리의 디렉터리 구조 예



2 **aws-provider.yaml** 파일을 저장하는 디렉터리입니다.

3 AWS Secrets Manager 공급자를 설치하고 해당 공급자를 위한 리소스를 배포하는 구성 파일입니다.

1 애플리케이션을 생성하고 AWS Secrets Manager용 리소스를 배포하는 구성 파일입니다.

4 배포 Pod 및 인증 정보 요청을 저장하는 디렉터리입니다.

5 **SecretProviderClass** 리소스를 저장하여 시크릿 저장소 공급자를 정의하는 디렉터리입니다.

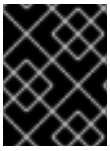
6 **credentialsrequest.yaml** 파일을 저장하는 폴더입니다. 이 파일에는 배포 Pod에 보안을 마운트하기 위한 인증 정보 요청 구성이 포함되어 있습니다.

2.2. 사전 요구 사항

- **cluster-admin** 권한이 있는 클러스터에 액세스할 수 있습니다.
- OpenShift Container Platform 웹 콘솔에 액세스할 수 있습니다.
- **ccoctl** 바이너리를 추출하여 준비했습니다.
- **jq** CLI 툴을 설치했습니다.
- 클러스터가 AWS에 설치되어 있으며 AWS STS(보안 토큰 서비스)를 사용합니다.
- 필요한 보안을 저장하도록 AWS Secrets Manager를 구성했습니다.
- SSCSI Driver Operator가 클러스터에 설치되어 있습니다.
- Red Hat OpenShift GitOps Operator가 클러스터에 설치되어 있습니다.
- secrets를 사용할 수 있는 GitOps 리포지토리가 있습니다.
- Argo CD 관리자 계정을 사용하여 Argo CD 인스턴스에 로그인했습니다.

2.3. GITOPS 리포지토리에 AWS SECRETS MANAGER 리소스 저장

이 가이드에서는 CSI(Secrets Store Container Storage Interface) Driver Operator와 함께 GitOps 워크플로우를 사용하여 AWS Secrets Manager의 시크릿을 OpenShift Container Platform의 CSI 볼륨에 마운트하는 데 도움이 되는 예제를 제공합니다.



중요

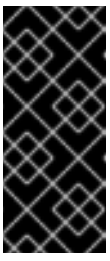
SSCSI Driver Operator를 AWS Secrets Manager와 함께 사용하는 것은 호스트된 컨트롤 플레인 클러스터에서 지원되지 않습니다.

사전 요구 사항

- **cluster-admin** 권한이 있는 클러스터에 액세스할 수 있습니다.
- OpenShift Container Platform 웹 콘솔에 액세스할 수 있습니다.
- **ccoctl** 바이너리를 추출하여 준비했습니다.
- **jq** CLI 툴을 설치했습니다.
- 클러스터가 AWS에 설치되어 있으며 AWS STS(보안 토큰 서비스)를 사용합니다.
- 필요한 보안을 저장하도록 AWS Secrets Manager를 구성했습니다.
- SSCSI Driver Operator가 클러스터에 설치되어 있습니다.
- Red Hat OpenShift GitOps Operator가 클러스터에 설치되어 있습니다.
- secrets를 사용할 수 있는 GitOps 리포지토리가 있습니다.
- Argo CD 관리자 계정을 사용하여 Argo CD 인스턴스에 로그인했습니다.

프로세스

1. AWS Secrets Manager 공급자를 설치하고 리소스를 추가합니다.
   a. GitOps 리포지토리에서 디렉터리를 생성하고 다음 구성으로 **aws-provider.yaml** 파일을 추가하여 AWS Secrets Manager 공급자에 대한 리소스를 배포합니다.



중요

SSCSI 드라이버의 AWS Secrets Manager 공급자는 업스트림 공급자입니다.

이 구성은 OpenShift Container Platform과 제대로 작동하도록 업스트림 AWS 설명서에 제공된 구성에서 수정됩니다. 이 구성을 변경하면 기능에 영향을 미칠 수 있습니다.

aws-provider.yaml 파일의 예

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: csi-secrets-store-provider-aws
  namespace: openshift-cluster-csi-drivers
```

```
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: csi-secrets-store-provider-aws-cluster-role
rules:
- apiGroups: [""]
  resources: ["serviceaccounts/token"]
  verbs: ["create"]
- apiGroups: [""]
  resources: ["serviceaccounts"]
  verbs: ["get"]
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get"]
- apiGroups: [""]
  resources: ["nodes"]
  verbs: ["get"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: csi-secrets-store-provider-aws-cluster-rolebinding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: csi-secrets-store-provider-aws-cluster-role
subjects:
- kind: ServiceAccount
  name: csi-secrets-store-provider-aws
  namespace: openshift-cluster-csi-drivers
---
apiVersion: apps/v1
kind: DaemonSet
metadata:
  namespace: openshift-cluster-csi-drivers
  name: csi-secrets-store-provider-aws
  labels:
    app: csi-secrets-store-provider-aws
spec:
  updateStrategy:
    type: RollingUpdate
  selector:
    matchLabels:
      app: csi-secrets-store-provider-aws
  template:
    metadata:
      labels:
        app: csi-secrets-store-provider-aws
    spec:
      serviceAccountName: csi-secrets-store-provider-aws
      hostNetwork: false
      containers:
        - name: provider-aws-installer
          image: public.ecr.aws/aws-secrets-manager/secrets-store-csi-driver-provider-
aws:1.0.r2-50-g5b4aca1-2023.06.09.21.19
```

```
        imagePullPolicy: Always
        args:
          - --provider-volume=/etc/kubernetes/secrets-store-csi-providers
        resources:
          requests:
            cpu: 50m
            memory: 100Mi
          limits:
            cpu: 50m
            memory: 100Mi
        securityContext:
          privileged: true
        volumeMounts:
          - mountPath: "/etc/kubernetes/secrets-store-csi-providers"
            name: providervol
          - name: mountpoint-dir
            mountPath: /var/lib/kubelet/pods
            mountPropagation: HostToContainer
      tolerations:
      - operator: Exists
      volumes:
        - name: providervol
          hostPath:
            path: "/etc/kubernetes/secrets-store-csi-providers"
        - name: mountpoint-dir
          hostPath:
            path: /var/lib/kubelet/pods
            type: DirectoryOrCreate
      nodeSelector:
        kubernetes.io/os: linux
```

b. GitOps 리포지토리에 **secret-provider-app.yaml** 파일을 추가하여 애플리케이션을 생성하고 AWS Secrets Manager의 리소스를 배포합니다.

secret-provider-app.yaml 파일 예

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: secret-provider-app
  namespace: openshift-gitops
spec:
  destination:
    namespace: openshift-cluster-csi-drivers
    server: https://kubernetes.default.svc
  project: default
  source:
    path: path/to/aws-provider/resources
    repoURL: https://github.com/<my-domain>/<gitops>.git (1)
  syncPolicy:
    automated:
    prune: true
    selfHeal: true
```

(1) GitOps 리포지토리를 가리키도록 **repoURL** 필드의 값을 업데이트합니다.

2. 리소스를 기본 Argo CD 인스턴스와 동기화하여 클러스터에 배포합니다.

   a. **openshift-gitops** 네임스페이스의 Argo CD 인스턴스에서 관리할 수 있도록 애플리케이션이 배포된 **openshift-cluster-csi-drivers** 네임스페이스에 레이블을 추가합니다.

   ```
   $ oc label namespace openshift-cluster-csi-drivers argocd.argoproj.io/managed-by=openshift-gitops
   ```

   b. 방금 내보낸 **aws-provider.yaml** 파일을 포함하여 GitOps 리포지토리의 리소스를 클러스터에 적용합니다.

   출력 예

   ```
   application.argoproj.io/argo-app created
   application.argoproj.io/secret-provider-app created
   ...
   ```

Argo CD UI에서 **csi-secrets-store-provider-aws** 데몬 세트에서 리소스를 계속 동기화하는지 확인할 수 있습니다. 이 문제를 해결하려면 AWS Secrets Manager에서 시크릿을 마운트하도록 SSCSI 드라이버를 구성해야 합니다.

2.4. AWS SECRETS MANAGER에서 시크릿을 마운트하도록 SSCSI 드라이버 구성

보안을 안전하게 저장하고 관리하려면 GitOps 워크플로우를 사용하고 Secrets Store Container Storage Interface (SSCSI) Driver Operator를 구성하여 AWS Secrets Manager의 시크릿을 OpenShift Container Platform의 CSI 볼륨에 마운트합니다. 예를 들어 **/environments/dev/** 디렉터리에 있는 **dev** 네임스페이스 아래의 배포 Pod에 시크릿을 마운트하려고 합니다.

사전 요구 사항

- AWS Secrets Manager 리소스가 GitOps 리포지토리에 저장되어 있습니다.

프로세스

1. 다음 명령을 실행하여 **csi-secrets-store-provider-aws** 서비스 계정에 대한 권한 권한을 부여합니다.

   ```
   $ oc adm policy add-scc-to-user privileged -z csi-secrets-store-provider-aws -n openshift-cluster-csi-drivers
   ```

   출력 예

   ```
   clusterrole.rbac.authorization.k8s.io/system:openshift:scc:privileged added: "csi-secrets-store-provider-aws"
   ```

2. 서비스 계정에서 AWS 시크릿 오브젝트를 읽을 수 있는 권한을 부여합니다.

   a. 인증 정보 요청이 네임스페이스 범위이므로 GitOps 리포지토리의 네임스페이스 범위 디렉터리에 **credentialsrequest-dir-aws** 폴더를 생성합니다. 예를 들어 다음 명령을 실행하여 **/environments/dev/** 디렉터리에 있는 **dev** 네임스페이스에 **credentialsrequest-dir-aws** 폴더를 생성합니다.

```
$ mkdir credentialsrequest-dir-aws
```

b. **/environments/dev/credentialsrequest-dir-aws/** 경로에서 인증 정보 요청에 대한 다음 구성으로 YAML 파일을 생성하여 **dev** 네임스페이스의 배포 Pod에 보안을 마운트합니다.

credentialsrequest.yaml 파일 예

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: aws-provider-test
  namespace: openshift-cloud-credential-operator
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
    - action:
      - "secretsmanager:GetSecretValue"
      - "secretsmanager:DescribeSecret"
      effect: Allow
      resource: "<aws_secret_arn>" 1
secretRef:
  name: aws-creds
  namespace: dev 2
serviceAccountNames:
  - default
```

2 시크릿 참조의 네임스페이스입니다. 프로젝트 배포 설정에 따라 이 **네임스페이스** 필드의 값을 업데이트합니다.

1 클러스터가 있는 리전의 시크릿 ARN입니다. &lt **;aws_ secret_arn>의 <aws_** region> 은 클러스터 리전과 일치해야 합니다. 일치하지 않는 경우 클러스터가 있는 리전에 시크릿 복제를 생성합니다.

작은 정보

클러스터 리전을 찾으려면 다음 명령을 실행합니다.

```
$ oc get infrastructure cluster -o jsonpath='{.status.platformStatus.aws.region}'
```

출력 예

```
us-west-2
```

c. 다음 명령을 실행하여 OIDC 공급자를 검색합니다.

```
$ oc get --raw=/.well-known/openid-configuration | jq -r '.issuer'
```

출력 예

```
https://<oidc_provider_name>
```

다음 단계에서 사용할 출력에서 OIDC **공급자 이름 <oidc_provider_name >**을 복사합니다.

d. **ccoctl** 툴을 사용하여 다음 명령을 실행하여 인증 정보 요청을 처리합니다.

```
$ ccoctl aws create-iam-roles \
    --name my-role --region=<aws_region> \
    --credentials-requests-dir=credentialsrequest-dir-aws \
    --identity-provider-arn arn:aws:iam::<aws_account>:oidc-
provider/<oidc_provider_name> --output-dir=credrequests-ccoctl-output
```

출력 예

```
2023/05/15 18:10:34 Role arn:aws:iam::<aws_account_id>:role/my-role-my-namespace-
aws-creds created
2023/05/15 18:10:34 Saved credentials configuration to: credrequests-ccoctl-
output/manifests/my-namespace-aws-creds-credentials.yaml
2023/05/15 18:10:35 Updated Role policy for Role my-role-my-namespace-aws-creds
```

다음 단계에서 사용할 출력에서 < **aws_role_arn** >을 복사합니다. 예를 들어 **arn:aws:iam::<aws_account_id>:role/my-role-my-namespace-aws-creds**.

e. AWS의 역할 정책을 확인하여 역할 정책의 < **aws_region** >의 <aws_region>이 클러스터 리전과 일치하는지 확인합니다.

역할 정책 예

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "secretsmanager:GetSecretValue",
    "secretsmanager:DescribeSecret"
   ],
   "Resource": "arn:aws:secretsmanager:<aws_region>:
<aws_account_id>:secret:my-secret-xxxxxx"
  }
 ]
}
```

f. **다음 명령을 실행하여 서비스 계정을 ARN 역할과 바인딩합니다.**

```
$ oc annotate -n <namespace> sa/<app_service_account>
eks.amazonaws.com/role-arn="<aws_role_arn>"
```

명령 예

```
$ oc annotate -n dev sa/default eks.amazonaws.com/role-arn="<aws_role_arn>"
```

출력 예

```
serviceaccount/default annotated
```

3. 네임스페이스 범위의 **SecretProviderClass** 리소스를 생성하여 시크릿 저장소 공급자를 정의합니다. 예를 들어 **GitOps** 리포지토리의 **/environments/dev/apps/app-taxi/services/taxi/base/config** 디렉터리에 **SecretProviderClass** 리소스를 생성합니다.

   a. 대상 배포가 **GitOps** 리포지토리에 있는 동일한 디렉터리에 **secret-provider-class-aws.yaml** 파일을 생성합니다.

secret-provider-class-aws.yaml의 예

```
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: my-aws-provider 1
  namespace: dev 2
spec:
  provider: aws 3
  parameters: 4
    objects: |
      - objectName: "testSecret" 5
        objectType: "secretsmanager"
```

1

시크릿 공급자 클래스의 이름입니다.

2

시크릿 공급자 클래스의 네임스페이스입니다. 네임스페이스는 보안을 사용할 리소스의 네임스페이스와 일치해야 합니다.

3

시크릿 저장소 공급자의 이름입니다.

4

공급자별 구성 매개변수를 지정합니다.

5

AWS에서 생성한 시크릿 이름입니다.

b. 이 YAML 파일을 GitOps 리포지토리로 내보낸 후 Argo CD UI의 대상 애플리케이션 페이지에 네임스페이스 범위의 SecretProviderClass 리소스가 채워져 있는지 확인합니다.

참고

애플리케이션의 동기화 정책이 Auto 로 설정되지 않은 경우 Argo CD UI에서 동기화를 클릭하여 SecretProviderClass 리소스를 수동으로 동기화 할 수 있습니다.

2.5. 마운트된 보안을 사용하도록 GITOPS 관리 리소스 구성

배포에 볼륨 마운트 구성을 추가하고 마운트된 보안을 사용하도록 컨테이너 Pod를 구성하여 GitOps 관리 리소스를 구성해야 합니다.

사전 요구 사항

- AWS Secrets Manager 리소스가 GitOps 리포지토리에 저장되어 있습니다.

- AWS Secrets Manager에서 보안을 마운트하도록 SSCSI(Secrets Store Container Storage Interface) 드라이버가 구성되어 있습니다.

프로세스

1. GitOps 관리 리소스를 구성합니다. 예를 들어 app-taxi 애플리케이션 배포에 볼륨 마운트 구성을 추가하고 100-deployment.yaml 파일은 /environments/dev/apps/app-taxi/services/config/ 디렉터리에 있다고 가정합니다.

   a. 배포 YAML 파일에 볼륨 마운트를 추가하고 시크릿 공급자 클래스 리소스 및 마운트된 보안을 사용하도록 컨테이너 Pod를 구성합니다.

YAML 파일의 예

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: taxi
  namespace: dev (1)
spec:
  replicas: 1
  template:
    metadata:
# ...
    spec:
      containers:
        - image: nginxinc/nginx-unprivileged:latest
          imagePullPolicy: Always
          name: taxi
          ports:
            - containerPort: 8080
          volumeMounts:
            - name: secrets-store-inline
              mountPath: "/mnt/secrets-store" (2)
              readOnly: true
          resources: {}
      serviceAccountName: default
      volumes:
        - name: secrets-store-inline
          csi:
            driver: secrets-store.csi.k8s.io
            readOnly: true
            volumeAttributes:
```

```
        secretProviderClass: "my-aws-provider" 3
    status: {}
# ...
```

1

배포를 위한 네임스페이스입니다. 이는 시크릿 공급자 클래스와 동일한 네임스페이스여야 합니다.

2

볼륨 마운트에 시크릿을 마운트하는 경로입니다.

3

시크릿 공급자 클래스의 이름입니다.

b. 업데이트된 리소스 YAML 파일을 GitOps 리포지토리로 내보냅니다.

2. Argo CD UI에서 대상 애플리케이션 페이지에서 REFRESH 를 클릭하여 업데이트된 배포 매니페스트를 적용합니다.

3. 모든 리소스가 대상 애플리케이션 페이지에서 성공적으로 동기화되었는지 확인합니다.

4. Pod 볼륨 마운트의 AWS Secrets 관리자에서 시크릿에 액세스할 수 있는지 확인합니다.

a. Pod 마운트의 보안을 나열합니다.

```
$ oc exec <deployment_name>-<hash> -n <namespace> -- ls /mnt/secrets-store/
```

명령 예

```
$ oc exec taxi-5959644f9-t847m -n dev -- ls /mnt/secrets-store/
```

출력 예

```
<secret_name>
```

b. Pod 마운트에서 보안을 확인합니다.

```
$ oc exec <deployment_name>-<hash> -n <namespace> -- cat /mnt/secrets-store/<secret_name>
```

명령 예

```
$ oc exec taxi-5959644f9-t847m -n dev -- cat /mnt/secrets-store/testSecret
```

출력 예

```
<secret_value>
```

2.6. 추가 리소스

- ccoctl 툴 가져오기
- Cloud Credential Operator 정보

- Cloud Credential Operator 모드 확인
- AWS STS를 사용하도록 AWS 클러스터 구성
- 필요한 시크릿을 저장하도록 AWS Secrets Manager 구성
- Secrets Store CSI Driver Operator 정보
- 외부 시크릿 저장소에서 CSI 볼륨으로 보안 마운트