Red Hat OpenShift GitOps 1.12

安全性

使用安全功能配置安全通信，并保护传输中可能存在的敏感数据

Last Updated: 2024-05-15

Red Hat OpenShift GitOps 1.12 安全性

使用安全功能配置安全通信，并保护传输中可能存在的敏感数据

法律通告

摘要

本文档提供在 OpenShift GitOps 中使用传输层安全(TLS)加密的说明。它还讨论了如何配置与 Redis 的安全通信，以保护传输中可能存在的敏感数据。

目录

第 1 章 配置与 REDIS 的安全通信

在 Red Hat OpenShift GitOps 中使用传输层安全 (TLS) 加密，您可以保护 Argo CD 组件和 Redis 缓存之间的通信，并保护传输中潜在的敏感数据。

您可以使用以下配置之一保护与 Redis 的通信：

- 启用 **autotls** 设置，为 TLS 加密发布适当的证书。
- 通过使用密钥和证书对创建 **argocd-operator-redis-tls** secret，手动配置 TLS 加密。

启用或没有启用高可用性 (HA) 时都可以使用这两个配置。

1.1. 先决条件

- 您可以使用 **cluster-admin** 权限访问集群。
- 访问 OpenShift Container Platform web 控制台。
- 在集群中安装了 Red Hat OpenShift GitOps Operator。

1.2. 为启用了 AUTOTLS 的 REDIS 配置 TLS

您可以通过在新的或已有的 Argo CD 实例中启用 **autotls** 设置来为 Redis 配置 TLS 加密。配置会自动置备 **argocd-operator-redis-tls** secret，且不需要进一步的步骤。目前，OpenShift Container Platform 是唯一受支持的 secret 供应商。

注意

默认情况下禁用 **autotls** 设置。

流程

1. 登陆到 OpenShift Container Platform Web 控制台。
2. 创建启用了 **autotls** 的 Argo CD 实例：
   a. 在 Web 控制台的 **Administrator** 视角中，使用左侧导航面板进入 **Administration → CustomResourceDefinitions**。
   b. 搜索 **argocds.argoproj.io** 并点 **ArgoCD** 自定义资源定义 (CRD)。
   c. 在 **CustomResourceDefinition 详情**页面中，点 **Instances** 选项卡，然后点 **Create ArgoCD**。
   d. 编辑或替换类似以下示例的 YAML：

启用 autotls 的 Argo CD CR 示例

```
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 实例的名称。
2. 要运行 Argo CD 实例的命名空间。
3. 启用 **autotls** 设置并为 Redis 创建 TLS 证书的标记。
4. 启用 HA 功能的 flag 值。如果不启用 HA，请不要包含此行，或者将标志值设为 **false**。

提示

另外，您可以通过运行以下命令来在已经存在的 Argo CD 实例上启用 **autotls** 设置：

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

e. 点 **Create**。

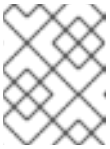
f. 验证 Argo CD pod 是否已就绪并在运行：

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

1. 指定运行 Argo CD 实例的命名空间，如 **openshift-gitops**。

禁用 HA 的输出示例

```
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
```



注意

启用 HA 的 TLS 配置需要一个至少有三个 worker 节点的集群。如果您启用了使用 HA 配置的 Argo CD 实例，可能需要几分钟时间才会显示输出。

启用了 HA 的输出示例

```
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. 验证 **argocd-operator-redis-tls** secret 是否已创建：

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

(1) 指定运行 Argo CD 实例的命名空间，如 **openshift-gitops**。

输出示例

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

secret 必须是 **kubernetes.io/tls** 类型，大小为 **2**。

1.3. 为禁用了 AUTOTLS 的 REDIS 配置 TLS

您可以使用密钥和证书对创建 **argocd-operator-redis-tls** secret，为 Redis 手动配置 TLS 加密。另外，您必须注解 secret 以指示它属于适当的 Argo CD 实例。对于启用了高可用性 (HA) 的实例，创建证书和 secret 的步骤会有所不同。

流程

1. 登陆到 OpenShift Container Platform Web 控制台。
2. 创建 Argo CD 实例：
   a. 在 Web 控制台的 **Administrator** 视角中，使用左侧导航面板进入 **Administration → CustomResourceDefinitions**。
   b. 搜索 **argocds.argoproj.io** 并点 **ArgoCD** 自定义资源定义 (CRD)。
   c. 在 **CustomResourceDefinition 详情**页面中，点 **Instances** 选项卡，然后点 **Create ArgoCD**。
   d. 编辑或替换类似以下示例的 YAML：

禁用 autotls 的 ArgoCD CR 示例

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

(1) Argo CD 实例的名称。

(2) 要运行 Argo CD 实例的命名空间。

3 启用 HA 功能的 flag 值。如果不启用 HA，请不要包含此行，或者将标志值设为 **false**。

e. 点 **Create**。

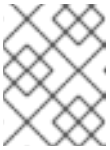
f. 验证 Argo CD pod 是否已就绪并在运行：

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

1 指定运行 Argo CD 实例的命名空间，如 **openshift-gitops**。

禁用 HA 的输出示例

```
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
```



注意

启用 HA 的 TLS 配置需要一个至少有三个 worker 节点的集群。如果您启用了使用 HA 配置的 Argo CD 实例，可能需要几分钟时间才会显示输出。

启用了 HA 的输出示例

```
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. 根据您的 HA 配置，使用以下选项之一为 Redis 服务器创建一个自签名证书：

- 对于禁用了 HA 的 Argo CD 实例，请运行以下命令：

```
$ 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 指定运行 Argo CD 实例的命名空间，如 **openshift-gitops**。

输出示例

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

- 对于启用了 HA 的 Argo CD 实例，运行以下命令：

```
$ 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 指定运行 Argo CD 实例的命名空间，如 **openshift-gitops**。

输出示例

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

4. 运行以下命令，验证生成的证书和密钥是否在 **/tmp** 目录中可用：

```
$ cd /tmp
```

```
$ ls
```

禁用 HA 的输出示例

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

启用了 HA 的输出示例

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

5. 根据您的 HA 配置，使用以下选项之一创建 **argocd-operator-redis-tls** secret：

- 对于禁用了 HA 的 Argo CD 实例，请运行以下命令：

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

- 对于启用了 HA 的 Argo CD 实例，运行以下命令：

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

输出示例

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

6. 注解 secret 以表示它属于 Argo CD CR：

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

1 指定 Argo CD 实例的名称，如 **argocd**。

输出示例

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

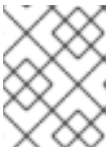
7. 验证 Argo CD pod 是否已就绪并在运行：

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

1 指定运行 Argo CD 实例的命名空间，如 **openshift-gitops**。

禁用 HA 的输出示例

```
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
```



注意

如果您启用了使用 HA 配置的 Argo CD 实例，可能需要几分钟时间才会显示输出。

启用了 HA 的输出示例

```
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
```

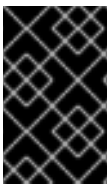
```
argocd-redis-ha-server-2                  2/2     Running   0          53s
argocd-repo-server-576499d46d-8hgbh       1/1     Running   0          10m
argocd-server-9486f88b7-dk2ks             1/1     Running   0          10m
```

第 2 章 使用带有 GITOPS 的 SECRET STORE CSI 驱动程序安全地管理 SECRET

本指南指导您在 OpenShift Container Platform 4.14 及之后的版本中将 Secret Store Container Storage Interface (SSCSI)驱动程序与 GitOps Operator 集成。

2.1. 使用带有 GITOPS 的 SECRET STORE CSI 驱动程序管理 SECRET 概述

有些应用程序需要敏感信息，如密码和用户名，它们必须被视为良好的安全实践。如果公开敏感信息，因为集群中没有正确配置基于角色的访问控制(RBAC)，则具有 API 或 etcd 访问权限的任何人都可以检索或修改 secret。



重要

授权在命名空间中创建 pod 的任何人都可以使用该 RBAC 读取该命名空间中的任何 secret。使用 SSCSI Driver Operator 时，您可以使用外部 secret 存储来安全地存储并为 pod 提供敏感信息。

将 OpenShift Container Platform SSCSI 驱动程序与 GitOps Operator 集成的过程包括以下步骤：

1. 在 GitOps 仓库中存储 AWS Secrets Manager 资源
2. 配置 SSCSI 驱动程序从 AWS Secrets Manager 挂载 secret
3. 配置 GitOps 受管资源以使用挂载的 secret

2.1.1. 优点

将 SSCSI 驱动程序与 GitOps Operator 集成有以下优点：

- 提高 GitOps 工作流的安全性和效率
- 有助于将 secret 的安全作为卷附加到部署 pod 中
- 确保安全高效地访问敏感信息

2.1.2. Secret 存储供应商

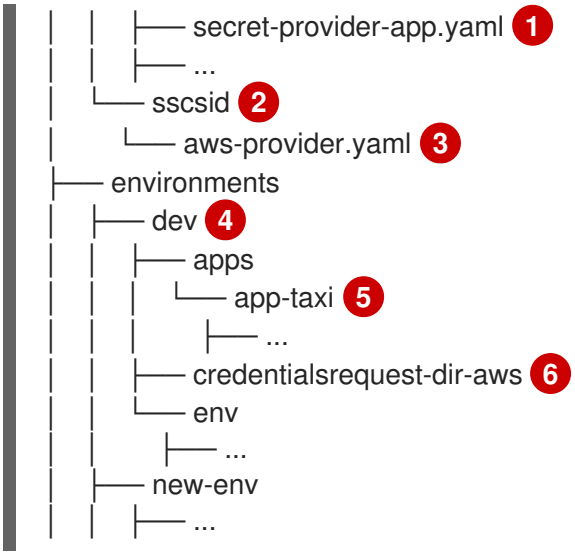
以下 secret 存储供应商可用于 Secret Store CSI Driver Operator：

- AWS Secrets Manager
- AWS Systems Manager Parameter Store
- Microsoft Azure Key Vault

例如，假设您使用 AWS Secrets Manager 作为带有 SSCSI Driver Operator 的 secret 存储供应商。以下示例显示了 GitOps 存储库中的目录结构，它可供使用 AWS Secrets Manager 中的 secret：

GitOps 存储库中的目录结构示例

```
├── config
│   ├── argocd
│   │   ├── argo-app.yaml
```



2 存储 **aws-provider.yaml** 文件的目录。

3 安装 AWS Secrets Manager 供应商并为它部署资源的配置文件。

1 创建用于 AWS Secret Manager 的应用程序和部署资源的配置文件。

4 存储部署 Pod 和凭据请求的目录。

5 存储 **SecretProviderClass** 资源的目录，以定义您的 secret 存储供应商。

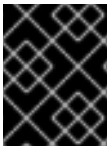
6 存储 **credentialsrequest.yaml** 文件的文件夹。此文件包含凭据请求的配置，以将 secret 挂载到部署 pod。

2.2. 先决条件

- 您可以使用 **cluster-admin** 权限访问集群。
- 访问 OpenShift Container Platform web 控制台。
- 您已提取并准备好 **ccoctl** 二进制文件。
- 已安装 **jq** CLI 工具。
- 您的集群安装在 AWS 上，并使用 AWS 安全令牌服务 (STS)。
- 您已将 AWS Secrets Manager 配置为存储所需的 secret。
- SSCSI Driver Operator 已安装在集群中。
- 在集群中安装了 Red Hat OpenShift GitOps Operator。
- 您有一个 GitOps 存储库可供使用 secret。
- 您可以使用 Argo CD admin 帐户登录到 Argo CD 实例。

2.3. 在 GITOPS 仓库中存储 AWS SECRETS MANAGER 资源

本指南提供了示例，可帮助您将 GitOps 工作流与 Secret Store Container Storage Interface (SSCSI) Driver Operator 搭配使用，将 secret 从 AWS Secrets Manager 挂载到 OpenShift Container Platform 中的 CSI 卷。



重要

托管 control plane 集群不支持将 SSCSI Driver Operator 与 AWS Secrets Manager 搭配使用。

先决条件

- 您可以使用 **cluster-admin** 权限访问集群。
- 访问 OpenShift Container Platform web 控制台。
- 您已提取并准备好 **ccoctl** 二进制文件。
- 已安装 **jq** CLI 工具。
- 您的集群安装在 AWS 上，并使用 AWS 安全令牌服务 (STS)。
- 您已将 AWS Secrets Manager 配置为存储所需的 secret。
- SSCSI Driver Operator 已安装在集群中。
- 在集群中安装了 Red Hat OpenShift GitOps Operator。
- 您有一个 GitOps 存储库可供使用 secret。
- 您可以使用 Argo CD admin 帐户登录到 Argo CD 实例。

流程

1. 安装 AWS Secrets Manager 供应商并添加资源：
   a. 在 GitOps 存储库中，使用以下配置创建一个目录并添加 **aws-provider.yaml** 文件，以便为 AWS Secrets Manager 供应商部署资源：

重要

SSCSI 驱动程序的 AWS Secrets Manager 供应商是一个上游供应商。

此配置会根据上游 AWS 文档中提供的配置进行修改，以便它可以与 OpenShift Container Platform 正常工作。对此配置的更改可能会影响功能。

aws-provider.yaml 文件示例

```
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. 在 GitOps 仓库中添加 **secret-provider-app.yaml** 文件，以便为 AWS Secrets Manager 创建应用程序和部署资源：

secret-provider-app.yaml 文件示例

```
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) 更新 **repoURL** 字段的值以指向 GitOps 存储库。

2. 将资源与默认 Argo CD 实例同步，以在集群中部署它们：

a. 为应用程序部署到 **openshift-cluster-csi-drivers** 命名空间添加标签，以便 **openshift-gitops** 命名空间中的 Argo CD 实例可以管理它：

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

b. 将 GitOps 存储库中的资源应用到集群，包括您刚才推送的 **aws-provider.yaml** 文件：

输出示例

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

在 Argo CD UI 中，您可以观察 **csi-secrets-store-provider-aws** daemonset 继续同步资源。要解决这个问题，您必须配置 SSCSI 驱动程序来从 AWS Secrets Manager 挂载 secret。

2.4. 配置 SSCSI 驱动程序从 AWS SECRETS MANAGER 挂载 SECRET

要安全地存储和管理您的 secret，请使用 GitOps 工作流并配置 Secret Store Container Storage Interface (SSCSI) Driver Operator，将 secret 从 AWS Secrets Manager 挂载到 OpenShift Container Platform 中的 CSI 卷。例如，请考虑您要将 secret 挂载到 **dev** 命名空间下的部署 pod，该命名空间位于 **/environments/dev/** 目录中。

先决条件

- 您有存储在 GitOps 存储库中的 AWS Secrets Manager 资源。

流程

1. 运行以下命令，授予 **csi-secrets-store-provider-aws** 服务帐户的特权访问权限：

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

输出示例

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

2. 授予服务帐户读取 AWS secret 对象的权限：

a. 在 GitOps 仓库的命名空间范围的目录中创建一个 **credentialsrequest-dir-aws** 文件夹，因为凭证请求是命名空间范围。例如，运行以下命令，在 **dev** 命名空间中创建一个 **credentialsrequest-dir-aws** 文件夹，该文件夹位于 **/environments/dev/** 目录中：

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

b. 在 **/environments/dev/credentialsrequest-dir-aws/** 路径中为凭证请求创建一个 YAML 文件，以将 secret 挂载到 **dev** 命名空间中的部署 pod：

credentialsrequest.yaml 文件示例

■

```
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 secret 引用的命名空间。根据您的项目部署设置，更新此 **namespace** 字段的值。

1 集群所在区域中的 secret 的 ARN。& **lt;aws_ secret_arn> 的 <aws_** region> 必须与集群区域匹配。如果不匹配，请在集群所在的区域中创建 secret 的复制。

提示

要查找集群区域，请运行以下命令：

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

输出示例

```
us-west-2
```

c. 运行以下命令来检索 OIDC 供应商：

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

输出示例

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

从输出中复制 OIDC 供应商名称 **<oidc_provider_name>**，在下一步中使用。

d. 运行以下命令，使用 **ccoctl** 工具处理凭证请求：

```
$ 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
```

输出示例

```
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
```

从输出中复制 **<aws_role_arn>** 以在下一步中使用。例如，**arn:aws:iam::<aws_account_id>:role/my-role-my-namespace-aws-creds**。

e. 检查 AWS 上的角色策略，以确认角色策略中的 **<aws_region** > **"Resource"** 与集群区域匹配：

角色策略示例

```
{
 "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. 运行以下命令，使用角色 ARN 绑定服务帐户：

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

示例命令

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

输出示例

```
serviceaccount/default annotated
```

3. 创建一个命名空间范围的 **SecretProviderClass** 资源来定义您的 secret 存储供应商。例如，您可以在 GitOps 存储库的 **/environments/dev/apps/app-taxi/services/taxi/base/config** 目录中创建一个 **SecretProviderClass** 资源。

   a. 在目标部署所在的同一目录中创建一个 **secret-provider-class-aws.yaml** 文件：

secret-provider-class-aws.yaml示例

```
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. secret 供应商类的名称。
2. secret 供应商类的命名空间。命名空间必须与将使用 secret 的资源的命名空间匹配。
3. secret 存储供应商的名称。
4. 指定特定于供应商的配置参数。
5. 您在 AWS 中创建的 secret 名称。

b. 在将此 YAML 文件推送到 GitOps 存储库后，命名空间范围内的 **SecretProviderClass** 资源会在 Argo CD UI 的目标应用程序页面中填充。

注意

如果应用程序的 Sync Policy 没有设置为 **Auto**，您可以通过点 Argo CD UI 中的 **Sync** 来手动同步 **SecretProviderClass** 资源。

2.5. 配置 GITOPS 受管资源以使用挂载的 SECRET

您必须通过将卷挂载配置添加到部署并配置容器 pod 来使用挂载的 secret 来配置 GitOps 受管资源。

先决条件

- 您有存储在 GitOps 存储库中的 AWS Secrets Manager 资源。
- 您已将 Secret Store Container Storage Interface (SSCSI)驱动程序配置为从 AWS Secret Manager 挂载 secret。

流程

1. 配置 GitOps 受管资源。例如，假设您想将卷挂载配置添加到 **app-taxi** 应用程序的部署中，而 **100-deployment.yaml** 文件位于 **/environments/dev/apps/app-taxi/services/taxi/base/config/** 目录中。

   a. 将卷挂载添加到部署 YAML 文件中，并将容器 pod 配置为使用 secret 供应商类资源和挂载的 secret：

   YAML 文件示例

```
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. 部署的命名空间。这必须与 secret 供应商类相同。
2. 在卷挂载中挂载 secret 的路径。
3. secret 供应商类的名称。

b. 将更新的资源 YAML 文件推送到 GitOps 存储库。

2. 在 Argo CD UI 中，点目标应用程序页面中的 **REFRESH** 应用更新的部署清单。
3. 验证所有资源是否在目标应用程序页面中成功同步。
4. 验证您可以从 pod 卷挂载中的 AWS Secrets manager 访问 secret：

a. 列出 pod 挂载中的 secret：

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

示例命令

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

输出示例

```
<secret_name>
```

b. 查看 pod 挂载中的 secret：

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

示例命令

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

输出示例

```
<secret_value>
```

2.6. 其他资源

- 获取 **ccoctl** 工具
- 配置 Cloud Credential Operator 工具
- 将 AWS 集群配置为使用 AWS STS
- 配置 AWS Secrets Manager 以存储所需的 secret
- 关于 Secret Store CSI Driver Operator
- 将 secret 从外部 secret 存储挂载到 CSI 卷