Red Hat OpenShift GitOps 1.10

Security

Using security features to configure secure communication and protect the possibly sensitive data in transit

Last Updated: 2024-01-11

Red Hat OpenShift GitOps 1.10 Security

Using security features to configure secure communication and protect the possibly sensitive data in transit

Legal Notice

Abstract

This document provides instructions for using the Transport Layer Security (TLS) encryption with the OpenShift GitOps. It also discusses how to configure secure communication with Redis to protect the possibly sensitive data in transit.

Table of Contents

CHAPTER 1. CONFIGURING SECURE COMMUNICATION WITH REDIS 3
1.1. PREREQUISITES 3
1.2. CONFIGURING TLS FOR REDIS WITH AUTOTLS ENABLED 3
1.3. CONFIGURING TLS FOR REDIS WITH AUTOTLS DISABLED 5

CHAPTER 1. CONFIGURING SECURE COMMUNICATION WITH REDIS

Using the Transport Layer Security (TLS) encryption with Red Hat OpenShift GitOps, you can secure the communication between the Argo CD components and Redis cache and protect the possibly sensitive data in transit.

You can secure communication with Redis by using one of the following configurations:

- Enable the **autotls** setting to issue an appropriate certificate for TLS encryption.
- Manually configure the TLS encryption by creating the **argocd-operator-redis-tls** secret with a key and certificate pair.

Both configurations are possible with or without the High Availability (HA) enabled.

1.1. PREREQUISITES

- You have access to the cluster with **cluster-admin** privileges.
- You have access to the OpenShift Container Platform web console.
- Red Hat OpenShift GitOps Operator is installed on your cluster.

1.2. CONFIGURING TLS FOR REDIS WITH AUTOTLS ENABLED

You can configure TLS encryption for Redis by enabling the **autotls** setting on a new or already existing Argo CD instance. The configuration automatically provisions the **argocd-operator-redis-tls** secret and does not require further steps. Currently, OpenShift Container Platform is the only supported secret provider.

NOTE

By default, the **autotls** setting is disabled.

Procedure

1. Log in to the OpenShift Container Platform web console.
2. Create an Argo CD instance with **autotls** enabled:
   a. In the **Administrator** perspective of the web console, use the left navigation panel to go to **Administration → CustomResourceDefinitions**.
   b. Search for **argocds.argoproj.io** and click **ArgoCD** custom resource definition (CRD).
   c. On the **CustomResourceDefinition details** page, click the **Instances** tab, and then click **Create ArgoCD**.
   d. Edit or replace the YAML similar to the following example:

   Example Argo CD CR with autotls enabled

```
apiVersion: argoproj.io/v1beta1
```

```
kind: ArgoCD
metadata:
  name: argocd (1)
  namespace: openshift-gitops (2)
spec:
  redis:
    autotls: openshift (3)
  ha:
    enabled: true (4)
```

1. The name of the Argo CD instance.
2. The namespace where you want to run the Argo CD instance.
3. The flag that enables the **autotls** setting and creates a TLS certificate for Redis.
4. The flag value that enables the HA feature. If you do not want to enable HA, do not include this line or set the flag value as **false**.

TIP

Alternatively, you can enable the **autotls** setting on an already existing Argo CD instance by running the following command:

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

e. Click **Create**.

f. Verify that the Argo CD pods are ready and running:

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

1. Specify a namespace where the Argo CD instance is running, for example **openshift-gitops**.

Example output with HA disabled

```
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
```

NOTE

The HA-enabled TLS configuration requires a cluster with at least three worker nodes. It can take a few minutes for the output to appear if you have enabled the Argo CD instances with HA configuration.

Example output with HA enabled

```
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. Verify that the **argocd-operator-redis-tls** secret is created:

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

(1) Specify a namespace where the Argo CD instance is running, for example **openshift-gitops**.

Example output

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

The secret must be of the **kubernetes.io/tls** type and a size of **2**.

1.3. CONFIGURING TLS FOR REDIS WITH AUTOTLS DISABLED

You can manually configure TLS encryption for Redis by creating the **argocd-operator-redis-tls** secret with a key and certificate pair. In addition, you must annotate the secret to indicate that it belongs to the appropriate Argo CD instance. The steps to create a certificate and secret vary for instances with High Availability (HA) enabled.

Procedure

1. Log in to the OpenShift Container Platform web console.
2. Create an Argo CD instance:
   a. In the **Administrator** perspective of the web console, use the left navigation panel to go to **Administration → CustomResourceDefinitions**.
   b. Search for **argocds.argoproj.io** and click **ArgoCD** custom resource definition (CRD).
   c. On the **CustomResourceDefinition details** page, click the **Instances** tab, and then click **Create ArgoCD**.
   d. Edit or replace the YAML similar to the following example:

Example ArgoCD CR with autotls disabled

```
apiVersion: argoproj.io/v1beta1
kind: ArgoCD
metadata:
```

```
  name: argocd (1)
  namespace: openshift-gitops (2)
spec:
  ha:
    enabled: true (3)
```

(1) The name of the Argo CD instance.

(2) The namespace where you want to run the Argo CD instance.

(3) The flag value that enables the HA feature. If you do not want to enable HA, do not include this line or set the flag value as **false**.

e. Click **Create**.

f. Verify that the Argo CD pods are ready and running:

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

(1) Specify a namespace where the Argo CD instance is running, for example **openshift-gitops**.

Example output with HA disabled

```
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
```

NOTE

The HA-enabled TLS configuration requires a cluster with at least three worker nodes. It can take a few minutes for the output to appear if you have enabled the Argo CD instances with HA configuration.

Example output with HA enabled

```
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. Create a self-signed certificate for the Redis server by using one of the following options depending on your HA configuration:

- For the Argo CD instance with HA disabled, run the following command:

```
$ 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 Specify a namespace where the Argo CD instance is running, for example **openshift-gitops**.

Example output

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

- For the Argo CD instance with HA enabled, run the following command:

```
$ 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 Specify a namespace where the Argo CD instance is running, for example **openshift-gitops**.

Example output

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

4. Verify that the generated certificate and key are available in the **/tmp** directory by running the following commands:

```
$ cd /tmp
```

```
$ ls
```

Example output with HA disabled

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

Example output with HA enabled

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

5. Create the **argocd-operator-redis-tls** secret by using one of the following options depending on your HA configuration:

   - For the Argo CD instance with HA disabled, run the following command:

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

   - For the Argo CD instance with HA enabled, run the following command:

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

     Example output

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

6. Annotate the secret to indicate that it belongs to the Argo CD CR:

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

   (1) Specify a name of the Argo CD instance, for example **argocd**.

   Example output

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

7. Verify that the Argo CD pods are ready and running:

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

   (1) Specify a namespace where the Argo CD instance is running, for example **openshift-gitops**.

   Example output with HA disabled

```
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
```

NOTE

It can take a few minutes for the output to appear if you have enabled the Argo CD instances with HA configuration.

Example output with HA enabled

```
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
```