Red Hat Advanced Cluster Security for Kubernetes 4.1

Configuring

Configuring Red Hat Advanced Cluster Security for Kubernetes

Last Updated: 2024-04-01

Red Hat Advanced Cluster Security for Kubernetes 4.1 Configuring

Configuring Red Hat Advanced Cluster Security for Kubernetes

Legal Notice

Abstract

This document describes how to perform common configuration tasks, including configuring certificates, automatic upgrades, and proxy settings. It also includes information on enabling monitoring and logging.

Table of Contents

CHAPTER 1. ADDING CUSTOM CERTIFICATES

Learn how to use a custom TLS certificate with Red Hat Advanced Cluster Security for Kubernetes. After you set up a certificate, users and API clients do not have to bypass the certificate security warnings when connecting to Central.

1.1. ADDING A CUSTOM SECURITY CERTIFICATE

You can apply a security certificate during the installation or on an existing Red Hat Advanced Cluster Security for Kubernetes deployment.

1.1.1. Prerequisites for adding custom certificates

Prerequisites

- You must already have PEM-encoded private key and certificate files.
- The certificate file should begin and end with human-readable blocks. For example:

```
-----BEGIN CERTIFICATE-----
MIICLDCCAdKgAwIBAgIBADAKBggqhkjOPQQDAjB9MQswCQYDVQQGEwJCRTEPMA0G
...
l4wOuDwKQa+upc8GftXE2C//4mKANBC6It01gUaTIpo=
-----END CERTIFICATE-----
```

- The certificate file can contain either a single (leaf) certificate, or a certificate chain.

WARNING

- If the certificate is not directly signed by a trusted root, you must provide the full certificate chain, including any intermediate certificates.
- All certificates in the chain must be in order so that the leaf certificate is the first and the root certificate is the last in the chain.

- If you are using a custom certificate that is not globally trusted, you must also configure the Sensor to trust your custom certificate.

1.1.2. Adding a custom certificate during a new installation

Procedure

- If you are installing Red Hat Advanced Cluster Security for Kubernetes using the Operator:
  1. Create a **central-default-tls-cert** secret that contains the appropriate TLS certificates in the namespace where the Central service will be installed by entering the following command:

```
oc -n <namespace> create secret tls central-default-tls-cert --cert <tls-cert.pem> --key
<tls-key.pem>
```

- If you are installing Red Hat Advanced Cluster Security for Kubernetes using Helm:

  1. Add your custom certificate and its key in the **values-private.yaml** file:

```
central:
  # Configure a default TLS certificate (public cert + private key) for central
  defaultTLS:
    cert: |
      -----BEGIN CERTIFICATE-----

EXAMPLE!MIIMIICLDCCAdKgAwIBAgIBADAKBggqhkjOPQQDAjB9MQswCQYDVQQGE
wJCRTEPMA0G
      ...
      -----END CERTIFICATE-----
    key: |
      -----BEGIN EC PRIVATE KEY-----
      EXAMPLE!MHcl4wOuDwKQa+upc8GftXE2C//4mKANBC6It01gUaTIpo=
      ...
      -----END EC PRIVATE KEY-----
```

  2. Provide the configuration file during the installation:

```
$ helm install -n stackrox --create-namespace stackrox-central-services rhacs/central-
services -f values-private.yaml
```

- If you are installing Red Hat Advanced Cluster Security for Kubernetes using the **roxctl** CLI, provide the certificate and key files when you run the installer:

  - For the non-interactive installer, use the **--default-tls-cert** and **--default-tls-key** options:

```
$ roxctl central generate --default-tls-cert "cert.pem" --default-tls-key "key.pem"
```

  - For the interactive installer, provide the certificate and key files when you enter answers for the prompts:

```
...
Enter PEM cert bundle file (optional): <cert.pem>
Enter PEM private key file (optional): <key.pem>
Enter administrator password (default: autogenerated):
Enter orchestrator (k8s, openshift): openshift
...
```

1.1.3. Adding a custom certificate for an existing instance

Procedure

- If you have installed Red Hat Advanced Cluster Security for Kubernetes using the Operator:

  1. Create a **central-default-tls-cert** secret that contains the appropriate TLS certificates in the namespace where the Central service is installed by entering the following command:

```
oc -n <namespace> create secret tls central-default-tls-cert --cert <tls-cert.pem> --key
<tls-key.pem>
```

- If you have installed Red Hat Advanced Cluster Security for Kubernetes using Helm:

  1. Add your custom certificate and its key in the **values-private.yaml** file:

```
central:
  # Configure a default TLS certificate (public cert + private key) for central
  defaultTLS:
    cert: |
      -----BEGIN CERTIFICATE-----

EXAMPLE!MIIMIICLDCCAdKgAwIBAgIBADAKBggqhkjOPQQDAjB9MQswCQYDVQQGE
wJCRTEPMA0G
      ...
      -----END CERTIFICATE-----
    key: |
      -----BEGIN EC PRIVATE KEY-----
      EXAMPLE!MHcl4wOuDwKQa+upc8GftXE2C//4mKANBC6It01gUaTIpo=
      ...
      -----END EC PRIVATE KEY-----
```

  2. Use the **helm upgrade** command and provide the updated configuration file:

```
$ helm upgrade -n stackrox --create-namespace stackrox-central-services rhacs/central-
services -f values-private.yaml
```

- If you have installed Red Hat Advanced Cluster Security for Kubernetes using the **roxctl** CLI:

  - Create and apply a TLS secret from the PEM-encoded key and certificate files:

```
$ oc -n stackrox create secret tls central-default-tls-cert \
  --cert <server_cert.pem> \
  --key <server_key.pem> \
  --dry-run -o yaml | oc apply -f -
```

    After you run this command, Central automatically applies the new key and certificate without requiring the pod to be restarted. It might take up to a minute to propagate the changes.

1.1.4. Updating the custom certificate for an existing instance

If you use a custom certificate for Central, you can update the certificate by performing the following procedure.

Procedure

1. Delete the existing custom certificate's secret:

```
$ oc delete secret central-default-tls-cert
```

2. Create a new secret:

```
$ oc -n stackrox create secret tls central-default-tls-cert \
  --cert <server_cert.pem> \
  --key <server_key.pem> \
  --dry-run -o yaml | oc apply -f -
```

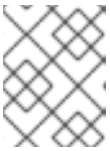
3. Restart the Central container.

1.1.4.1. Restarting the Central container

You can restart the Central container by killing the Central container or by deleting the Central pod.

Procedure

- Run the following command to kill the Central container:



NOTE

You must wait for at least 1 minute, until OpenShift Container Platform propagates your changes and restarts the Central container.

```
$ oc -n stackrox exec deploy/central -c central -- kill 1
```

- Or, run the following command to delete the Central pod:

```
$ oc -n stackrox delete pod -lapp=central
```

1.2. CONFIGURING SENSOR TO TRUST CUSTOM CERTIFICATES

If you are using a custom certificate that is not trusted globally, you must configure the Sensor to trust your custom certificate. Otherwise, you might get errors. The specific type of error may vary based on your setup and the certificate you use. Usually, it is an **x509 validation** related error.

NOTE

You do not need to configure the Sensor to trust your custom certificate if you are using a globally trusted certificate.

1.2.1. Downloading a Sensor bundle

The Sensor bundle includes the necessary configuration files and scripts to install Sensor. You can download the Sensor bundle from the RHACS portal.

Procedure

1. Navigate to the RHACS portal.
2. Go to **Platform Configuration → Clusters**.
3. Click **New Cluster** and specify a name for the cluster.
4. If you are deploying the Sensor in the same cluster, accept the default values for all the fields. Otherwise, if you are deploying into a different cluster, replace the address

central.stackrox.svc:443 with a load balancer, node port, or other address (including the port number) that is accessible from the other cluster in which you are planning to install.

NOTE

If you are using a non-gRPC capable load balancer, such as HAProxy, AWS Application Load Balancer (ALB), or AWS Elastic Load Balancing (ELB) use the WebSocket Secure (**wss**) protocol. To use **wss**:

1. Prefix the address with **wss://**, and
2. Add the port number after the address, for example, **wss://stackrox-central.example.com:443**.

5. Click **Next** to continue.
6. Click **Download YAML File and Keys**.

1.2.2. Configuring Sensor to trust custom certificates when deploying a new Sensor

Prerequisites

- You have downloaded the Sensor bundle.

Procedure

- If you are using the **sensor.sh** script:
  1. Unzip the Sensor bundle:

     ```
     $ unzip -d sensor sensor-<cluster_name>.zip
     ```

  2. Run the **sensor.sh** script:

     ```
     $ ./sensor/sensor.sh
     ```

     The certificates are automatically applied when you run the sensor (**./sensor/sensor.sh**) script. You can also place additional custom certificates in the **sensor/additional-cas/** directory before you run the **sensor.sh** script.

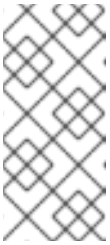
- If you are not using the **sensor.sh** script:
  1. Unzip the Sensor bundle:

     ```
     $ unzip -d sensor sensor-<cluster_name>.zip
     ```

  2. Run the following command to create the secret:

     ```
     $ ./sensor/ca-setup-sensor.sh -d sensor/additional-cas/ 1
     ```

     1 Use the **-d** option to specify a directory containing custom certificates.



NOTE

If you get the "secret already exists" error message, re-run the script with the **-u** option:

```
$ ./sensor/ca-setup-sensor.sh -d sensor/additional-cas/ -u
```

3. Continue Sensor deployment by using the YAML files.

1.2.3. Configuring an existing Sensor to trust custom certificates

Prerequisites

- You have downloaded the Sensor bundle.

Procedure

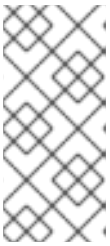
1. Unzip the Sensor bundle:

```
$ unzip -d sensor sensor-<cluster_name>.zip
```

2. Run the following command to create the secret:

```
$ ./sensor/ca-setup-sensor.sh -d sensor/additional-cas/ 1
```

1 Use the **-d** option to specify a directory containing custom certificates.



NOTE

If you get the "secret already exists" error message, re-run the script with the **-u** option:

```
$ ./sensor/ca-setup-sensor.sh -d sensor/additional-cas/ -u
```

3. Continue Sensor deployment by using the YAML files.

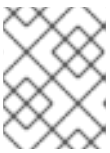
If you added the certificates to an existing sensor, you must restart the Sensor container.

1.2.3.1. Restarting the Sensor container

You can restart the Sensor container either by killing the container or by deleting the Sensor pod.

Procedure

- Run the following command to kill the Sensor container:



NOTE

You must wait for at least 1 minute, until OpenShift Container Platform or Kubernetes propagates your changes and restarts the Sensor container.

- On OpenShift Container Platform:

```
$ oc -n stackrox deploy/sensor -c sensor -- kill 1
```

- On Kubernetes:

```
$ kubectl -n stackrox deploy/sensor -c sensor -- kill 1
```

- Or, run the following command to delete the Sensor pod:
  - On OpenShift Container Platform:

```
$ oc -n stackrox delete pod -lapp=sensor
```

  - On Kubernetes:

```
$ kubectl -n stackrox delete pod -lapp=sensor
```

CHAPTER 2. ADDING TRUSTED CERTIFICATE AUTHORITIES

Learn how to add custom trusted certificate authorities to Red Hat Advanced Cluster Security for Kubernetes.

If you are using an enterprise certificate authority (CA) on your network, or self-signed certificates, you must add the CA's root certificate to Red Hat Advanced Cluster Security for Kubernetes as a trusted root CA.

Adding trusted root CAs allows:

- Central and Scanner to trust remote servers when you integrate with other tools.
- Sensor to trust custom certificates you use for Central.

You can add additional CAs during the installation or on an existing deployment.

NOTE

You must first configure your trusted CAs in the cluster where you have deployed Central and then propagate the changes to Scanner and Sensor.

2.1. CONFIGURING ADDITIONAL CAS

To add custom CAs:

Procedure

1. Download the **ca-setup.sh** script.

NOTE

- If you are doing a new installation, you can find the **ca-setup.sh** script in the **scripts** directory at **central-bundle/central/scripts/ca-setup.sh**.
- You must run the **ca-setup.sh** script in the same terminal from which you logged into your OpenShift Container Platform cluster.

2. Make the **ca-setup.sh** script executable:

```
$ chmod +x ca-setup.sh
```

3. To add:
   a. A single certificate, use the **-f** (file) option:

```
$ ./ca-setup.sh -f <certificate>
```

NOTE

- You must use a PEM-encoded certificate file (with any extension).
- You can also use the **-u** (update) option along with the **-f** option to update any previously added certificate.

b. Multiple certificates at once, move all certificates in a directory, and then use the **-d** (directory) option:

```
$ ./ca-setup.sh -d <directory_name>
```

NOTE

- You must use PEM-encoded certificate files with a **.crt** or **.pem** extension.
- Each file must only contain a single certificate.
- You can also use the **-u** (update) option along with the **-d** option to update any previously added certificates.

2.2. PROPAGATING CHANGES

After you configure trusted CAs, you must make Red Hat Advanced Cluster Security for Kubernetes services trust them.

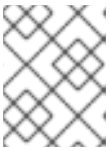
- If you have configured trusted CAs after the installation, you must restart Central.
- Additionally, if you are also adding certificates for integrating with image registries, you must restart both Central and Scanner.

2.2.1. Restarting the Central container

You can restart the Central container by killing the Central container or by deleting the Central pod.

Procedure

- Run the following command to kill the Central container:



NOTE

You must wait for at least 1 minute, until OpenShift Container Platform propagates your changes and restarts the Central container.

```
$ oc -n stackrox exec deploy/central -c central -- kill 1
```

- Or, run the following command to delete the Central pod:

```
$ oc -n stackrox delete pod -lapp=central
```

2.2.2. Restarting the Scanner container

You can restart the Scanner container by deleting the pod.

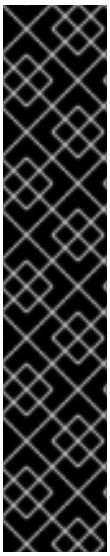
Procedure

- Run the following command to delete the Scanner pod:
  - On OpenShift Container Platform:

```
$ oc delete pod -n stackrox -l app=scanner
```

  - On Kubernetes:

```
$ kubectl delete pod -n stackrox -l app=scanner
```



IMPORTANT

After you have added trusted CAs and configured Central, the CAs are included in any new Sensor deployment bundles that you create.

- If an existing Sensor reports problems while connecting to Central, you must generate a Sensor deployment YAML file and update existing clusters.
- If you are deploying a new Sensor using the **sensor.sh** script, run the following command before you run the **sensor.sh** script:

```
$ ./ca-setup-sensor.sh -d ./additional-cas/
```

- If you are deploying a new Sensor using Helm, you do not have to run any additional scripts.

CHAPTER 3. REISSUING INTERNAL CERTIFICATES

Each component of Red Hat Advanced Cluster Security for Kubernetes uses an X.509 certificate to authenticate itself to other components. These certificates have expiration dates, and you must reissue them before they expire. You can view the certificate expiry dates in the **Platform Configuration → Clusters** view from the RHACS portal.

3.1. REISSUING INTERNAL CERTIFICATES FOR CENTRAL

Central uses a built-in server certificate for authentication when communicating with other Red Hat Advanced Cluster Security for Kubernetes services. This certificate is unique to your Central installation. The RHACS portal shows an information banner when the Central certificate is about to expire.

NOTE

The information banner only appears 15 days before the certificate expiry date.

Prerequisites

- To reissue certificates, you must have **write** permission for the **ServiceIdentity** resource.

Procedure

1. Click on the link in the banner to download a YAML configuration file, which contains a new OpenShift Container Platform secret, including the certificate and key values.
2. Apply the new YAML configuration file to the cluster where you have installed Central.

```
$ oc apply -f <secret_file.yaml>
```

3. Restart Central to apply the changes.

3.1.1. Restarting the Central container

You can restart the Central container by killing the Central container or by deleting the Central pod.

Procedure

- Run the following command to kill the Central container:

NOTE

You must wait for at least 1 minute, until OpenShift Container Platform propagates your changes and restarts the Central container.

```
$ oc -n stackrox exec deploy/central -c central -- kill 1
```

- Or, run the following command to delete the Central pod:

```
$ oc -n stackrox delete pod -lapp=central
```

3.2. REISSUING INTERNAL CERTIFICATES FOR SCANNER

Scanner has a built-in certificate that it uses to communicate with Central.

The RHACS portal shows an information banner when the Scanner certificate is about to expire.

NOTE

The information banner only appears 15 days before the certificate expiry date.

Prerequisites

- To reissue certificates, you must have **write** permission for the **ServiceIdentity** resource.

Procedure

1. Click on the link in the banner to download a YAML configuration file, which contains a new OpenShift Container Platform secret, including the certificate and key values.
2. Apply the new YAML configuration file to the cluster where you installed Scanner.

```
$ oc apply -f <secret_file.yaml>
```

3. Restart Scanner to apply the changes.

3.2.1. Restarting the Scanner and Scanner DB containers

You can restart the Scanner and Scanner DB container by deleting the pods.

Procedure

- To delete the Scanner and Scanner DB pods, run the following command:
  - On OpenShift Container Platform:

```
$ oc delete pod -n stackrox -l app=scanner; oc -n stackrox delete pod -l app=scanner-db
```

  - On Kubernetes:

```
$ kubectl delete pod -n stackrox -l app=scanner; kubectl -n stackrox delete pod -l
app=scanner-db
```

3.3. REISSUING INTERNAL CERTIFICATES FOR SENSOR, COLLECTOR, AND ADMISSION CONTROLLER

Sensor, Collector, and Admission controller use certificates to communicate with each other, and with Central.

To replace the certificates, use one of the following methods:

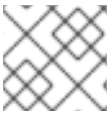
- Create, download, and install an init bundle on the secured cluster. You must have the **Admin** user role to create an init bundle.

- Use the automatic upgrades functionality. Automatic upgrades are available only for static manifest deployments using the **roxctl** CLI.

3.3.1. Reissuing internal certificates for Secured Clusters using init bundles

Secured clusters contain the Collector, Sensor, and Admission Control components. These components use a built-in server certificate for authentication when communicating with other Red Hat Advanced Cluster Security for Kubernetes components.

The RHACS portal shows an information banner when the Central certificate is about to expire.

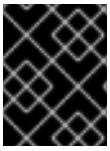


NOTE

The information banner only appears 15 days before the certificate expiry date.

Prerequisites

- To reissue certificates, you must have **write** permission for the **ServiceIdentity** resource.



IMPORTANT

Store this bundle securely because it contains secrets. You can use the same bundle on multiple secured clusters. You must have the **Admin** user role to create init bundles.

Procedure

- To generate an init bundle using the RHACS portal:
  a. Select **Platform Configuration → Clusters**.
  b. Click **Manage Tokens**.
  c. Navigate to the **Authentication Tokens** section, and click **Cluster Init Bundle**.
  d. Click **Generate bundle**.
  e. Enter a name for the cluster init bundle and click **Generate**.
  f. To download the generated bundle, click **Download Kubernetes secrets file**.
- To generate an init bundle using the **roxctl** CLI, run the following command:

```
$ roxctl -e <endpoint> -p <admin_password> central init-bundle generate <bundle_name> --output-secrets init-bundle.yaml
```

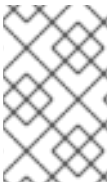
Next steps

- To create the necessary resources on each secured cluster, run the following command:

```
$ oc -n stackrox apply -f <init-bundle.yaml>
```

3.3.2. Reissuing internal certificates for secured clusters by using automatic upgrades

You can reissue internal certificates for Sensor, Collector, and Admission controller by using automatic upgrades.



NOTE

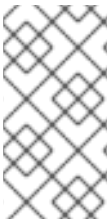
Automatic upgrades are only applicable to static manifest-based deployments using the **roxctl** CLI. See "Installing Central" in the "Installing by using the roxctl CLI" section of the *Installing* chapter.

Prerequisites

- You must have enabled automatic upgrades for all clusters.
- To reissue certificates, you must have **write** permission for the **ServiceIdentity** resource.

Procedure

1. In the RHACS portal, navigate to **Platform Configuration → Clusters**.
2. In the **Clusters** view, select a **Cluster** to view its details.
3. From the cluster details panel, select the link to **Apply credentials by using an automatic upgrade**.



NOTE

When you apply an automatic upgrade, Red Hat Advanced Cluster Security for Kubernetes creates new credentials in the selected cluster. However, you will still see a notification. The notification goes away when each Red Hat Advanced Cluster Security for Kubernetes service begins using the new credentials after the service restarts.

CHAPTER 4. ADDING SECURITY NOTICES

With Red Hat Advanced Cluster Security for Kubernetes, you can add security notices that users see when they log in. You can also set up an organization-wide message or disclaimers on the top or bottom of the RHACS portal.

This message can serve as a reminder of corporate policies and notify employees of the appropriate policies. Alternatively, you might want to display these messages for legal reasons, for example, to warn users that their actions are audited.

4.1. ADDING A CUSTOM LOGIN MESSAGE

The display of a warning message before login warns malicious or uninformed users about the consequences of their actions.

Prerequisites

- You must have the **Config** role with **read** permission to view the login message configuration options.
- You must have the **Config** role with **write** permission to modify, enable or disable the login message.

Procedure

1. On the RHACS portal, navigate to **Platform Configuration → System Configuration**.
2. On the **System Configuration** view header, click **Edit**.
3. Enter your login message in the **Login Configuration** section.
4. To enable the login message, turn on the toggle in the **Login Configuration** section.
5. Click **Save**.

4.2. ADDING A CUSTOM HEADER AND FOOTER

You can place custom text in a header and footer and configure the text and its background color.

Prerequisites

- You must have the **Config** role with **read** permission to view the custom header and footer configuration options.
- You must have the **Config** role with **write** permission to modify, enable or disable the custom header and footer.

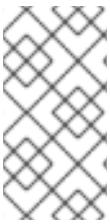
Procedure

1. On the RHACS portal, navigate to **Platform Configuration → System Configuration**.
2. On the **System Configuration** view header, click **Edit**.

3. Under the **Header Configuration** and **Footer Configuration** sections, enter the header and footer text.
4. Customize the header and footer **Text Color**, **Size**, and **Background Color**.
5. To enable the header, turn on the toggle in the **Header Configuration** section.
6. To enable the footer, turn on the toggle in the **Footer Configuration** section.
7. Click **Save**.

CHAPTER 5. ENABLING OFFLINE MODE

You can use Red Hat Advanced Cluster Security for Kubernetes for clusters that are not connected to the internet by enabling the offline mode. In offline mode, Red Hat Advanced Cluster Security for Kubernetes components do not connect to addresses or hosts on the internet.



NOTE

Red Hat Advanced Cluster Security for Kubernetes does not determine if the user-supplied hostnames, IP addresses, or other resources are on the internet. For example, if you try to integrate with a Docker registry hosted on the internet, Red Hat Advanced Cluster Security for Kubernetes will not block this request.

To deploy and operate Red Hat Advanced Cluster Security for Kubernetes in offline mode:

1. Download RHACS images and install them in your clusters. If you are using OpenShift Container Platform, you can use Operator Lifecycle Manager (OLM) and OperatorHub to download images to a workstation that is connected to the internet. The workstation then pushes images to a mirror registry that is also connected to your secured cluster. For other platforms, you can use a program such as Skopeo or Docker to pull the images from the remote registry and push them to your own private registry, as described in Downloading images directly.
2. Enable offline mode during installation.
3. (Optional) Routinely update Scanner's vulnerability list by uploading a new definitions file.
4. (Optional) When required, add support for runtime collection on more kernel versions by uploading new kernel support packages.

IMPORTANT

You can only enable offline mode during the installation, and not during an upgrade.

5.1. DOWNLOADING IMAGES FOR OFFLINE USE

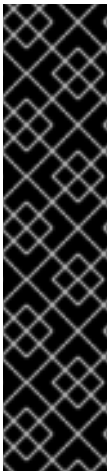
5.1.1. Downloading images directly

You can manually pull, retag, and push Red Hat Advanced Cluster Security for Kubernetes images to your registry. The images included in the current version of the image bundles are:

- **registry.redhat.io/advanced-cluster-security/rhacs-main-rhel8:4.1.5**
- **registry.redhat.io/advanced-cluster-security/rhacs-scanner-rhel8:4.1.5**
- **registry.redhat.io/advanced-cluster-security/rhacs-scanner-db-rhel8:4.1.5**
- **registry.redhat.io/advanced-cluster-security/rhacs-collector-rhel8:4.1.5**
- **registry.redhat.io/advanced-cluster-security/rhacs-collector-slim-rhel8:4.1.5**

5.1.1.1. Retagging images

You can download and retag images using the Docker command-line interface.



IMPORTANT

When you retag an image, you must maintain the name of the image and the tag. For example, use:

```
$ docker tag registry.redhat.io/advanced-cluster-security/rhacs-main-rhel8:4.1.5
<your_registry>/rhacs-main-rhel8:4.1.5
```

and do not retag like the following example:

```
$ docker tag registry.redhat.io/advanced-cluster-security/rhacs-main-rhel8:4.1.5
<your_registry>/other-name:latest
```

Procedure

1. Log in to the registry:

```
$ docker login registry.redhat.io
```

2. Pull the image:

```
$ docker pull <image>
```

3. Retag the image:

```
$ docker tag <image> <new_image>
```

4. Push the updated image to your registry:

```
$ docker push <new_image>
```

5.2. ENABLING OFFLINE MODE DURING INSTALLATION

You can enable offline mode during the installation of Red Hat Advanced Cluster Security for Kubernetes.

5.2.1. Enabling offline mode by using Helm configuration

You can enable offline mode during the installation when you are installing Red Hat Advanced Cluster Security for Kubernetes by using a Helm chart.

Procedure

1. When installing the central-services Helm chart, set the value of the **env.offlineMode** environmental variable to **true** in the **values-public.yaml** configuration file.

2. When installing the secured-cluster-services Helm chart, set the value of the **config.offlineMode** parameter to **true** in the **values-public.yaml** configuration file.

5.2.2. Enabling offline mode by using the roxctl CLI

You can enable offline mode when you are installing Red Hat Advanced Cluster Security for Kubernetes by using the **roxctl** CLI.

Procedure

1. If you are using a registry other than the default internet-connected registry (**registry.redhat.io**), provide the locations where you have pushed the Red Hat Advanced Cluster Security for Kubernetes images when answering the **image to use** prompts:

```
Enter main image to use (if unset, the default will be used): <your_registry>/rhacs-main-rhel8:4.1.5
```

NOTE

The default image depends on your answer for the prompt **Enter default container images settings:**. If you entered **rhacs**, the default option, the default image will be **registry.redhat.io/advanced-cluster-security/rhacs-main-rhel8:4.1.5**.

```
Enter Scanner DB image to use (if unset, the default will be used): <your_registry>/rhacs-scanner-db-rhel8:4.1.5
```

```
Enter Scanner image to use (if unset, the default will be used): <your_registry>/rhacs-scanner-rhel8:4.1.5
```

2. To enable the offline mode, enter **true** when answering the **Enter whether to run StackRox in offline mode** prompt:

```
Enter whether to run StackRox in offline mode, which avoids reaching out to the internet (default: "false"): true
```

3. Later, when you add Sensor to a remote cluster in the **Platform Configuration → Clusters** view in the RHACS portal, you must specify your the Collector image name in the **Collector Image Repository** field.

5.3. UPDATING SCANNER DEFINITIONS IN OFFLINE MODE

Scanner contains a local vulnerability definitions database. When Red Hat Advanced Cluster Security for Kubernetes runs in normal mode (connected to the internet), Scanner fetches new vulnerability definitions from the internet and updates its database.

However, when you are using Red Hat Advanced Cluster Security for Kubernetes in offline mode, you must manually update Scanner definitions by uploading them to Central.

When Red Hat Advanced Cluster Security for Kubernetes runs in offline mode, Scanner checks for new definitions from Central. If new definitions are available, Scanner downloads the new definitions from Central, marks them as default, and then uses the updated definitions for scanning images.

To update the definitions in offline mode:

1. Download the definitions.
2. Upload the definitions to Central.

5.3.1. Downloading Scanner definitions

If you are running Red Hat Advanced Cluster Security for Kubernetes in offline mode, you can download the vulnerability definitions database that Scanner uses and then upload it to Central.

Prerequisites

- To download Scanner definitions, you need a system with internet access.

Procedure

- Navigate to https://install.stackrox.io/scanner/scanner-vuln-updates.zip to download the definitions.

5.3.2. Uploading definitions to Central

To upload Scanner definitions to Central, you can either use an API token or your administrator password. Red Hat recommends using an authentication token in a production environment because each token is assigned specific access control permissions.

5.3.2.1. Uploading definitions to Central by using an API token

You can upload the vulnerability definitions database that Scanner uses to Central by using an API token.

Prerequisites

- You must have an API token with the administrator role.
- You must have installed the **roxctl** command-line interface (CLI).

Procedure

1. Set the **ROX_API_TOKEN** and the **ROX_CENTRAL_ADDRESS** environment variables:

```
$ export ROX_API_TOKEN=<api_token>
```

```
$ export ROX_CENTRAL_ADDRESS=<address>:<port_number>
```

2. Run the following command to upload the definitions file:

```
$ roxctl scanner upload-db \
  -e "$ROX_CENTRAL_ADDRESS" \
  --scanner-db-file=<compressed_scanner_definitions.zip>
```

5.3.2.1.1. Additional resources

- Authenticating by using the roxctl CLI

5.3.2.2. Uploading definitions to Central by using the administrator password

You can upload the vulnerability definitions database that Scanner uses to Central by using your Red Hat Advanced Cluster Security for Kubernetes administrator password.

Prerequisites

- You must have the administrator password.
- You must have installed the **roxctl** command-line interface (CLI).

Procedure

1. Set the **ROX_CENTRAL_ADDRESS** environment variable:

```
$ export ROX_CENTRAL_ADDRESS=<address>:<port_number>
```

2. Run the following command to upload the definitions file:

```
$ roxctl scanner upload-db \
  -p <your_administrator_password> \
  -e "$ROX_CENTRAL_ADDRESS" \
  --scanner-db-file=<compressed_scanner_definitions.zip>
```

5.4. UPDATING KERNEL SUPPORT PACKAGES IN OFFLINE MODE

Collector monitors the runtime activity for each node in your secured clusters. To monitor the activities, Collector requires probes. These probes are eBPF programs or kernel modules specific to the Linux kernel version installed on the host. The Collector image contains a set of built-in probes.

When Red Hat Advanced Cluster Security for Kubernetes runs in normal mode (connected to the internet), Collector automatically downloads a new probe if the required probe is not built in.

In offline mode, you can manually download packages containing probes for all recent and supported Linux kernel versions and upload them to Central. Collectors then download these probes from Central.

Collector checks for the new probes in the following order. It checks:

1. The existing Collector image.
2. The kernel support package (if you have uploaded one to Central).
3. A Red Hat-operated server available on the internet. Collector uses Central's network connection to check and download the probes.

If Collector does not get new probes after checking, it reports a **CrashLoopBackoff** event.

If your network configuration restricts outbound traffic, you can manually download packages containing probes for all recent and supported Linux kernel versions and upload them to Central. Collectors then download these probes from Central, thus avoiding any outbound internet access.

5.4.1. Downloading kernel support packages

If you are running Red Hat Advanced Cluster Security for Kubernetes in offline mode, you can download packages containing probes for all recent and supported Linux kernel versions and then upload them to Central.

Procedure

- View and download available support packages from

https://install.stackrox.io/collector/support-packages/index.html. The kernel support packages list categorizes support packages based on Red Hat Advanced Cluster Security for Kubernetes version.

5.4.2. Uploading kernel support packages to Central

You can upload the kernel support packages containing probes for all recent and supported Linux kernel versions to Central.

Prerequisites

- You must have an API token with the administrator role.
- You must have installed the **roxctl** command-line interface (CLI).

Procedure

1. Set the **ROX_API_TOKEN** and the **ROX_CENTRAL_ADDRESS** environment variables:

```
$ export ROX_API_TOKEN=<api_token>
```

```
$ export ROX_CENTRAL_ADDRESS=<address>:<port_number>
```

2. Run the following command to upload the kernel support packages:

```
$ roxctl collector support-packages upload <package_file> \
  -e "$ROX_CENTRAL_ADDRESS"
```

NOTE

- When you upload a new support package that includes content uploaded to Central previously, only new files are uploaded.
- When you upload a new support package that includes files with the same name but different contents than those present on the Central, **roxctl** shows a warning message and does not overwrite files.
  - You can use the **--overwrite** option with the upload command to overwrite the files.
- When you upload a support package that contains a required probe, Central does not make any outbound requests (to the internet) for downloading this probe. Central uses the probe from the support package.

CHAPTER 6. ENABLING ALERT DATA RETENTION

Learn how to configure a retention period for Red Hat Advanced Cluster Security for Kubernetes alerts.

With Red Hat Advanced Cluster Security for Kubernetes, you can configure the time to keep historical alerts stored. Red Hat Advanced Cluster Security for Kubernetes then deletes the older alerts after the specified time.

By automatically deleting alerts that are no longer needed, you can save storage costs.

The alerts for which you can configure the retention period include:

- Runtime alerts, both unresolved (active) and resolved.
- Stale deploy-time alerts that do not apply to the current deployment.

NOTE

- Data retention settings are enabled by default. You can change these settings after the installation.
- When you upgrade Red Hat Advanced Cluster Security for Kubernetes, data retention settings are not applied unless you have enabled them before.
- You can configure alert retention settings by using the RHACS portal or the API.
- The deletion process runs every hour. Currently, you cannot change this.

6.1. CONFIGURING ALERT DATA RETENTION

You can configure alert retention settings by using the RHACS portal.

Prerequisites

- You must have the **Config** role with **read** and **write** permissions to configure data retention.

Procedure

1. On the RHACS portal, navigate to **Platform Configuration → System Configuration**.
2. On the **System Configuration** view header, click **Edit**.
3. Under the **Data Retention Configuration** section, update the number of days for each type of data:
   - **All Runtime Violations**
   - **Resolved Deploy-Phase Violations**
   - **Runtime Violations For Deleted Deployments**
   - **Images No Longer Deployed**

NOTE

To save a type of data forever, set the retention period to **0** days.

4. Click **Save**.

NOTE

To configure alert data retention by using Red Hat Advanced Cluster Security for Kubernetes API, view the **PutConfig** API and related APIs in the **ConfigService** group in the API reference documentation.

CHAPTER 7. EXPOSING THE RHACS PORTAL OVER HTTP

Enable an unencrypted HTTP server to expose the RHACS portal through ingress controllers, Layer 7 load balancers, Istio, or other solutions.

If you use an ingress controller, Istio, or a Layer 7 load balancer that prefers unencrypted HTTP back ends, you can configure Red Hat Advanced Cluster Security for Kubernetes to expose the RHACS portal over HTTP. Doing this makes the RHACS portal available over a plaintext back end.

IMPORTANT

To expose the RHACS portal over HTTP, you must be using an ingress controller, a Layer 7 load balancer, or Istio to encrypt external traffic with HTTPS. It is insecure to expose the RHACS portal directly to external clients by using plain HTTP.

You can expose the RHACS portal over HTTP during installation or on an existing deployment.

7.1. PREREQUISITES

- To specify an HTTP endpoint you must use an **<endpoints_spec>**. It is a comma-separated list of single endpoint specifications in the form of **<type>@<addr>:<port>**, where:
  - **type** is **grpc** or **http**. Using **http** as type works in most use cases. For advanced use cases, you can either use **grpc** or omit its value. If you omit the value for **type**, you can configure two endpoints in your proxy, one for gRPC and the other for HTTP. Both these endpoints point to the same exposed HTTP port on Central. However, most proxies do not support carrying both gRPC and HTTP traffic on the same external port.
  - **addr** is the IP address to expose Central on. You can omit this, or use **localhost** or **127.0.0.1** if you need an HTTP endpoint which is only accessible by using port-forwarding.
  - **port** is the port to expose Central on.
  - The following are several valid **<endpoints_spec>** values:
    - **8080**
    - **http@8080**
    - **:8081**
    - **grpc@:8081**
    - **localhost:8080**
    - **http@localhost:8080**
    - **http@8080,grpc@8081**
    - **8080, grpc@:8081, http@0.0.0.0:8082**

7.2. EXPOSING THE RHACS PORTAL OVER HTTP DURING THE INSTALLATION

If you are installing Red Hat Advanced Cluster Security for Kubernetes using the **roxctl** CLI, use the **--plaintext-endpoints** option with the **roxctl central generate interactive** command to enable the HTTP server during the installation.

Procedure

- Run the following command to specify an HTTP endpoint during the interactive installation process:

```
$ roxctl central generate interactive \
  --plaintext-endpoints=<endpoints_spec> 1
```

1 Endpoint specifications in the form of **<type>@<addr>:<port>**. See the Prerequisites section for details.

7.3. EXPOSING THE RHACS PORTAL OVER HTTP FOR AN EXISTING DEPLOYMENT

You can enable the HTTP server on an existing Red Hat Advanced Cluster Security for Kubernetes deployment.

Procedure

1. Create a patch and define a **ROX_PLAINTEXT_ENDPOINTS** environment variable:

```
$ CENTRAL_PLAINTEXT_PATCH='
spec:
  template:
    spec:
      containers:
      - name: central
        env:
        - name: ROX_PLAINTEXT_ENDPOINTS
          value: <endpoints_spec> 1
'
```

1 Endpoint specifications in the form of **<type>@<addr>:<port>**. See the Prerequisites section for details.

2. Add the **ROX_PLAINTEXT_ENDPOINTS** environment variable to the Central deployment:

```
$ oc -n stackrox patch deploy/central -p "$CENTRAL_PLAINTEXT_PATCH"
```

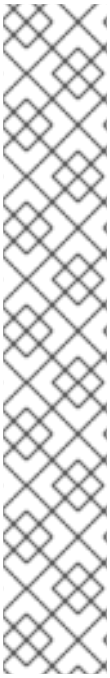
CHAPTER 8. CONFIGURING AUTOMATIC UPGRADES FOR SECURED CLUSTERS

You can automate the upgrade process for each secured cluster and view the upgrade status from the RHACS portal.

Automatic upgrades make it easier to stay up-to-date by automating the manual task of upgrading each secured cluster.

With automatic upgrades, after you upgrade Central; Sensor, Collector, and Compliance services in all secured clusters, automatically upgrade to the latest version.

Red Hat Advanced Cluster Security for Kubernetes also enables centralized management of all your secured clusters from within the RHACS portal. The new **Clusters** view displays information about all your secured clusters, the Sensor version for every cluster, and upgrade status messages. You can also use this view to selectively upgrade your secured clusters or change their configuration.



NOTE

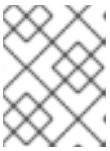
- The automatic upgrade feature is enabled by default.
- If you are using a private image registry, you must first push the Sensor and Collector images to your private registry.
- The Sensor must run with the default RBAC permissions.
- Automatic upgrades do not preserve any patches that you have made to any Red Hat Advanced Cluster Security for Kubernetes services running in your cluster. However, it preserves all labels and annotations that you have added to any Red Hat Advanced Cluster Security for Kubernetes object.
- By default, Red Hat Advanced Cluster Security for Kubernetes creates a service account called **sensor-upgrader** in each secured cluster. This account is highly privileged but is only used during upgrades. If you remove this account, Sensor does not have enough permissions, and you must complete future upgrades manually.

8.1. ENABLING AUTOMATIC UPGRADES

You can enable automatic upgrades for all secured clusters to automatically upgrade Collector and Compliance services in all secured clusters to the latest version.

Procedure

1. In the RHACS portal, navigate to **Platform Configuration → Clusters**.
2. Turn on the **Automatically upgrade secured clusters** toggle.



NOTE

For new installations, the **Automatically upgrade secured clusters** toggle is enabled by default.

8.2. DISABLING AUTOMATIC UPGRADES

If you want to manage your secured cluster upgrades manually, you can disable automatic upgrades.

Procedure

1. In the RHACS portal, navigate to **Platform Configuration → Clusters**.
2. Turn off the **Automatically upgrade secured clusters** toggle.

NOTE

For new installations, the **Automatically upgrade secured clusters** toggle is enabled by default.

8.3. AUTOMATIC UPGRADE STATUS

The **Clusters** view lists all clusters and their upgrade statuses.

Upgrade status	Description
Up to date with Central version	The secured cluster is running the same version as Central.
Upgrade available	A new version is available for the Sensor and Collector.
Upgrade failed. Retry upgrade.	The previous automatic upgrade failed.
Manual upgrade required	The Sensor and Collector version is older than version 2.5.29.0. You must manually upgrade your secured clusters.
Pre-flight checks complete	The upgrade is in progress. Before performing automatic upgrade, the upgrade installer runs a pre-flight check. During the pre-flight check, the installer verifies if certain conditions are satisfied and then only starts the upgrade process.

8.4. AUTOMATIC UPGRADE FAILURE

Sometimes, Red Hat Advanced Cluster Security for Kubernetes automatic upgrades might fail to install. When an upgrade fails, the status message for the secured cluster changes to **Upgrade failed. Retry upgrade**. To view more information about the failure and understand why the upgrade failed, you can check the secured cluster row in the **Clusters** view.

Some common reasons for the failure are:

- The sensor-upgrader deployment might not have run because of a missing or a non-schedulable image.
- The pre-flight checks may have failed, either because of insufficient RBAC permissions or because the cluster state is not recognizable. This can happen if you have edited Red Hat Advanced Cluster Security for Kubernetes service configurations or the **auto-**

upgrade.stackrox.io/component label is missing.

- There might be errors in executing the upgrade. If this happens, the upgrade installer automatically attempts to roll back the upgrade.

NOTE

Sometimes, the rollback can fail as well. For such cases view the cluster logs to identify the issue or contact support.

After you identify and fix the root cause for the upgrade failure, you can use the **Retry Upgrade** option to upgrade your secured cluster.

8.5. UPGRADING SECURED CLUSTERS MANUALLY FROM THE RHACS PORTAL

If you do not want to enable automatic upgrades, you can manage your secured cluster upgrades by using the **Clusters** view.

To manually trigger upgrades for your secured clusters:

Procedure

1. In the RHACS portal, navigate to **Platform Configuration → Clusters**.
2. Select the **Upgrade available** option under the **Upgrade status** column for the cluster you want to upgrade.
3. To upgrade multiple clusters at once, select the checkboxes in the **Cluster** column for the clusters you want to update.
4. Click **Upgrade**.

CHAPTER 9. CONFIGURING AUTOMATIC REMOVAL OF NONACTIVE CLUSTERS FROM RHACS

Red Hat Advanced Cluster Security for Kubernetes (RHACS) provides the option to configure your system to automatically remove nonactive clusters from RHACS so that you can monitor active clusters only. Note that only clusters that were installed and performed a handshake with Central at least one time are monitored initially. If this feature is enabled, when Central has been unable to reach Sensor in a cluster for the period of time configured in the **Decomissioned cluster age** field, the cluster is considered nonactive in RHACS. Central will then no longer monitor nonactive clusters. You can configure the **Decommissioned cluster age** field in the **Platform Configuration → System Configuration** page. When configuring this feature, you can add a label for the cluster so that RHACS continues to monitor the cluster even if it becomes nonactive.

Removal of nonactive clusters from RHACS is disabled by default. To enable this setting, enter a nonzero number in the **Decommissioned cluster age** field, as described in the following procedure. The **Decommissioned cluster age** field indicates the number of days that a cluster can remain unreachable before it is considered nonactive. When a cluster is nonactive, the **Clusters** page displays the status of the cluster. Nonactive clusters are indicated with the **unhealthy** label and the window shows the number of days after which the cluster will be removed from RHACS if it continues to remain nonactive. After a cluster is removed from RHACS, that action is documented with an **info** log in the Central logs.

NOTE

There is a 24-hour grace period after enabling this setting before clusters are removed. The cluster that hosts Central is never removed.

9.1. CONFIGURING CLUSTER DECOMMISSIONING

You can configure RHACS to automatically remove nonactive clusters from RHACS. Nonactive clusters are those that were installed and performed a handshake with Central at least once but have been unreachable by Sensor for a specified period of time. You can also label clusters so that they are not removed when they are unreachable.

Procedure

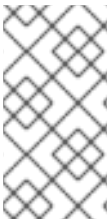
1. On the RHACS portal, navigate to **Platform Configuration → System Configuration**.
2. In the **System Configuration** header, click **Edit**.
3. In the **Cluster deletion** section, you can configure the following fields:
   - **Decommissioned cluster age**: The number of days that a cluster is unreachable before it is considered for removal from RHACS. If Central cannot reach Sensor on the cluster for this number of days, the cluster and all of its resources are removed from RHACS. To leave this feature disabled, which is the default behavior, enter **0** in this field. To enable this feature, enter a nonzero number, such as **90**, to configure the number of unreachable days.
   - **Ignore clusters which have labels**: To prevent a cluster from being removed, you can configure a label by entering a key and value in this section. Clusters with this label will not be removed, even if they have been unreachable for the number of days set in the **Decommissioned cluster age** field.
     - **Key**: Enter the label to use for the cluster.
     - **Value**: Enter the value associated with the key.

For example, to preserve production clusters from removal, you can configure a key of **cluster-type** and a value of **production**.

NOTE

In the **Cluster deletion** section, click **Clusters which have Sensor Status: Unhealthy** to go to the **Clusters** list page. This page is filtered to show nonactive clusters that are subject to removal and a timeframe for the removal from RHACS.

4. Click **Save**.



NOTE

To view and configure this option by using the API, use the **decommissionedClusterRetention** settings in the request payload for the **/v1/config** and **/v1/config/private** endpoints. For more information, see the API documentation for the **ConfigService** object by navigating to **Help → API reference** in the RHACS portal.

9.2. VIEWING NONACTIVE CLUSTERS

Nonactive clusters are clusters that were installed and performed a handshake with Central at least once but have been unreachable by Sensor for a specified period of time. Use this procedure to view a list of these clusters.

Procedure

1. On the RHACS portal, navigate to **Platform Configuration → System Configuration**.
2. In the **Cluster deletion** section, click **Clusters which have Sensor Status: Unhealthy** to go to the **Clusters** list page. This page is filtered to show nonactive clusters that are subject to removal from RHACS and a timeframe for the removal.

NOTE

If this feature is enabled after a cluster is considered nonactive, the counting of days towards removal starts from the time the cluster became nonactive, not from the time that the feature was enabled. If there are any nonactive clusters that you do not want removed, you can configure a label as described in the "Configuring cluster decommissioning" section. Clusters with those labels are ignored when the system removes nonactive clusters.

CHAPTER 10. CONFIGURING A PROXY FOR EXTERNAL NETWORK ACCESS

If your network configuration restricts outbound traffic through proxies, you can configure proxy settings in Red Hat Advanced Cluster Security for Kubernetes to route traffic through a proxy.

When you use a proxy with Red Hat Advanced Cluster Security for Kubernetes:

- All outgoing HTTP, HTTPS, and other TCP traffic from Central and Scanner goes through the proxy.
- Traffic between Central and Scanner does not go through the proxy.
- The proxy configuration does not affect the other Red Hat Advanced Cluster Security for Kubernetes components.
- When you are not using the offline mode, and a Collector running in a secured cluster needs to download an additional eBPF probe or kernel module at runtime:
  - The collector attempts to download them by contacting Sensor.
  - The Sensor then forwards this request to Central.
  - Central uses the proxy to locate the module or probe at **https://collector-modules.stackrox.io**.

10.1. CONFIGURING A PROXY ON AN EXISTING DEPLOYMENT

To configure a proxy in an existing deployment, you must export the **proxy-config** secret as a YAML file, update your proxy configuration in that file, and upload it as a secret.

NOTE

If you have configured a global proxy on your OpenShift Container Platform cluster, the Operator Lifecycle Manager (OLM) automatically configures Operators that it manages with the cluster-wide proxy. However, you can also configure installed Operators to override the global proxy or inject a custom certificate authority (CA) certificate.

For more information, see Configuring proxy support in Operator Lifecycle Manager .

Procedure

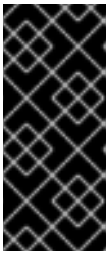
1. Save the existing secret as a YAML file:

```
$ oc -n stackrox get secret proxy-config \
  -o go-template='{{index .data "config.yaml" | \
  base64decode}}{{"\n"}}' > /tmp/proxy-config.yaml
```

2. Edit the fields you want to modify in the YAML configuration file, as specified in the Configure proxy during installation section.
3. After you save the changes, run the following command to replace the secret:

```
$ oc -n stackrox create secret generic proxy-config \
```

```
--from-file=config.yaml=/tmp/proxy-config.yaml -o yaml --dry-run | \
oc label -f - --local -o yaml app.kubernetes.io/name=stackrox | \
oc apply -f -
```



IMPORTANT

- You must wait for at least 1 minute, until OpenShift Container Platform propagates your changes to Central and Scanner.
- If you see any issues with outgoing connections after changing the proxy configuration, you must restart your Central and Scanner pods.

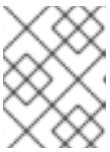
10.2. CONFIGURING A PROXY DURING INSTALLATION

When you are installing Red Hat Advanced Cluster Security for Kubernetes by using the **roxctl** command-line interface (CLI) or Helm, you can specify your proxy configuration during the installation.

When you run the installer by using the **roxctl central generate** command, the installer generates the secrets and deployment configuration files for your environment. You can configure a proxy by editing the generated configuration secret (YAML) file. Currently, you cannot configure proxies by using the **roxctl** CLI. The configuration is stored in a Kubernetes secret and it is shared by both Central and Scanner.

Procedure

1. Open the configuration file **central/proxy-config-secret.yaml** from your deployment bundle directory.



NOTE

If you are using Helm the configuration file is at **central/templates/proxy-config-secret.yaml**.

2. Edit the fields you want to modify in the configuration file:

```
apiVersion: v1
kind: Secret
metadata:
  namespace: stackrox
  name: proxy-config
type: Opaque
stringData:
  config.yaml: |- (1)
    # # NOTE: Both central and scanner should be restarted if this secret is changed.
    # # While it is possible that some components will pick up the new proxy configuration
    # # without a restart, it cannot be guaranteed that this will apply to every possible
    # # integration etc.
    # url: http://proxy.name:port (2)
    # username: username (3)
    # password: password (4)
    # # If the following value is set to true, the proxy wil NOT be excluded for the default hosts:
    # # - *.stackrox, *.stackrox.svc
    # # - localhost, localhost.localdomain, 127.0.0.0/8, ::1
```

```
    # # - *.local
    # omitDefaultExcludes: false
    # excludes:  # hostnames (may include * components) for which you do not 5
    # # want to use a proxy, like in-cluster repositories.
    # - some.domain
    # # The following configuration sections allow specifying a different proxy to be used for
HTTP(S) connections.
    # # If they are omitted, the above configuration is used for HTTP(S) connections as well as
TCP connections.
    # # If only the `http` section is given, it will be used for HTTPS connections as well.
    # # Note: in most cases, a single, global proxy configuration is sufficient.
    # http:
    #   url: http://http-proxy.name:port 6
    #   username: username 7
    #   password: password 8
    # https:
    #   url: http://https-proxy.name:port 9
    #   username: username 10
    #   password: password 11
```

3 4 7 8 10 11 Adding a **username** and a **password** is optional, both at the beginning and in the **http** and **https** sections.

2 6 9 The **url** option supports the following URL schemes:

- **http:**// for an HTTP proxy.
- **https:**// for a TLS-enabled HTTP proxy.
- **socks5:**// for a SOCKS5 proxy.

5 The **excludes** list can contain DNS names (with or without * wildcards), IP addresses, or IP blocks in CIDR notation (for example, **10.0.0.0/8**). The values in this list are applied to all outgoing connections, regardless of protocol.

1 The **|-** line in the **stringData** section indicates the start of the configuration data.

NOTE

- When you first open the file, all values are commented out (by using the **#** sign at the beginning of the line). Lines starting with double hash signs **# #** contain explanation of the configuration keys.
- Make sure that when you edit the fields, you maintain an indentation level of two spaces relative to the **config.yaml: |-** line.

3. After editing the configuration file, you can proceed with your usual installation. The updated configuration instructs Red Hat Advanced Cluster Security for Kubernetes to use the proxy running on the provided address and the port number.

CHAPTER 11. GENERATING A DIAGNOSTIC BUNDLE

You can generate a diagnostic bundle and send that data to enable the support team to provide insights into the status and health of Red Hat Advanced Cluster Security for Kubernetes components.

Red Hat might request you to send the diagnostic bundle during investigation of your issues with Red Hat Advanced Cluster Security for Kubernetes. You can generate a diagnostic bundle and inspect its data before sending.

NOTE

The diagnostic bundle is unencrypted, and depending upon the number of clusters in your environment, the bundle size is between 100 KB and 1 MB. Always use an encrypted channel to transfer this data back to Red Hat.

11.1. DIAGNOSTIC BUNDLE DATA

When you generate a diagnostic bundle, it includes the following data:

- Central heap profile.
- System logs: Logs of all Red Hat Advanced Cluster Security for Kubernetes components (for the last 20 minutes) and logs of recently crashed components (from up to 20 minutes before the crash). System logs depend on the size of your environment. For large deployments, data includes log files for components with critical errors only, such as a high restart count.
- YAML definitions for Red Hat Advanced Cluster Security for Kubernetes components: This data does not include Kubernetes secrets.
- OpenShift Container Platform or Kubernetes events: Details about the events that relate to the objects in the **stackrox** namespace.
- Online Telemetry data, which includes:
    - Storage information: Details about the database size and the amount of free space available in attached volumes.
    - Red Hat Advanced Cluster Security for Kubernetes components health information: Details about Red Hat Advanced Cluster Security for Kubernetes components versions, their memory usage, and any reported errors.
    - Coarse-grained usage statistics: Details about API endpoint invocation counts and reported error statuses. It does not include the actual data sent in API requests.
    - Nodes information: Details about the nodes in each secured cluster. It includes kernel and operating system versions, resource pressure, and taints.
    - Environment information: Details about each secured cluster, including Kubernetes or OpenShift Container Platform version, Istio version (if applicable), cloud provider type and other similar information.

11.2. GENERATING A DIAGNOSTIC BUNDLE BY USING THE RHACS PORTAL

You can generate a diagnostic bundle by using the system health dashboard on the RHACS portal.

Prerequisites

- To generate a diagnostic bundle, you need **read** permission for the **DebugLogs** resource.

Procedure

1. On the RHACS portal, select **Platform Configuration → System Health**.
2. On the **System Health** view header, click **Generate Diagnostic Bundle**.
3. For the **Filter by clusters** drop-down menu, select the clusters for which you want to generate the diagnostic data.
4. For **Filter by starting time**, specify the date and time (in UTC format) from which you want to include the diagnostic data.
5. Click **Download Diagnostic Bundle**.

11.3. GENERATING A DIAGNOSTIC BUNDLE BY USING THE ROXCTL CLI

You can generate a diagnostic bundle with the Red Hat Advanced Cluster Security for Kubernetes (RHACS) administrator password or API token and central address by using the **roxctl** CLI.

Prerequisites

- To generate a diagnostic bundle, you need **read** permission for the **Administration** resource. This is required for versions of the **DebugLogs** resource older than version 3.73.0.
- You must have configured the RHACS administrator password or API token and central address.

Procedure

- To generate a diagnostic bundle by using the RHACS administrator password, perform the following steps:
  1. Run the following command to configure the **ROX_PASSWORD** and **ROX_CENTRAL_ADDRESS** environment variables:

     ```
     $ export ROX_PASSWORD=<rox_password> && export ROX_CENTRAL_ADDRESS=<address>:<port_number>
     ```
     (1)

     (1) For **<rox_password>**, specify the RHACS administrator password.

  2. Run the following command to generate a diagnostic bundle by using the RHACS administrator password:

     ```
     $ roxctl -e "$ROX_CENTRAL_ADDRESS" -p "$ROX_PASSWORD" central debug download-diagnostics
     ```
- To generate a diagnostic bundle by using the API token, perform the following steps:
  1. Run the following command to configure the **ROX_API_TOKEN** environment variable:

```
$ export ROX_API_TOKEN=<api_token>
```

2. Run the following command to generate a diagnostic bundle by using the API token:

```
$ roxctl -e "$ROX_CENTRAL_ADDRESS" central debug download-diagnostics
```

CHAPTER 12. CONFIGURING ENDPOINTS

Learn how to configure endpoints for Red Hat Advanced Cluster Security for Kubernetes (RHACS) by using a YAML configuration file.

You can use a YAML configuration file to configure exposed endpoints. You can use this configuration file to define one or more endpoints for Red Hat Advanced Cluster Security for Kubernetes and customize the TLS settings for each endpoint, or disable the TLS for specific endpoints. You can also define if client authentication is required, and which client certificates to accept.

12.1. CUSTOM YAML CONFIGURATION

Red Hat Advanced Cluster Security for Kubernetes uses the YAML configuration as a **ConfigMap**, making configurations easier to change and manage.

When you use the custom YAML configuration file, you can configure the following for each endpoint:

- The protocols to use, such as **HTTP**, **gRPC**, or both.
- Enable or disable TLS.
- Specify server certificates.
- Client Certificate Authorities (CA) to trust for client authentication.
- Specify if client certificate authentication (**mTLS**) is required.

You can use the configuration file to specify endpoints either during the installation or on an existing instance of Red Hat Advanced Cluster Security for Kubernetes. However, if you expose any additional ports other than the default port **8443**, you must create network policies that allow traffic on those additional ports.

The following is a sample **endpoints.yaml** configuration file for Red Hat Advanced Cluster Security for Kubernetes:

```
# Sample endpoints.yaml configuration for Central.
#
# # CAREFUL: If the following line is uncommented, do not expose the default endpoint on port 8443 by default.
# #         This will break normal operation.
# disableDefault: true # if true, do not serve on :8443 (1)
endpoints: (2)
  # Serve plaintext HTTP only on port 8080
  - listen: ":8080" (3)
    # Backend protocols, possible values are 'http' and 'grpc'. If unset or empty, assume both.
    protocols: (4)
      - http
    tls: (5)
      # Disable TLS. If this is not specified, assume TLS is enabled.
      disable: true (6)
  # Serve HTTP and  gRPC for sensors only on port 8444
  - listen: ":8444" (7)
    tls: (8)
      # Which TLS certificates to serve, possible values are 'service' (For  service certificates that Red
```

```
Hat Advanced Cluster Security for Kubernetes generates)
    # and 'default' (user-configured default TLS certificate). If unset or empty, assume both.
    serverCerts: 9
      - default
      - service
    # Client authentication settings.
    clientAuth: 10
      # Enforce TLS client authentication. If unset, do not enforce, only request certificates
      # opportunistically.
      required: true 11
      # Which TLS client CAs to serve, possible values are 'service' (CA for service
      # certificates that Red Hat Advanced Cluster Security for Kubernetes generates) and 'user' (CAs
for PKI auth providers). If unset or empty, assume both.
      certAuthorities: 12
      # if not set, assume ["user", "service"]
        - service
```

1 Use **true** to disable exposure on the default port number **8443**. The default value is **false**; changing it to **true** might break existing functionality.

2 A list of additional endpoints for exposing Central.

3 7 The address and port number on which to listen. You must specify this value if you are using **endpoints**. You can use the format **port**, **:port**, or **address:port** to specify values. For example,

- **8080** or **:8080** - listen on port **8080** on all interfaces.
- **0.0.0.0:8080** - listen on port **8080** on all IPv4 (not IPv6) interfaces.
- **127.0.0.1:8080** - listen on port **8080** on the local loopback device only.

4 Protocols to use for the specified endpoint. Acceptable values are **http** and **grpc**. If you do not specify a value, Central listens to both HTTP and gRPC traffic on the specified port. If you want to expose an endpoint exclusively for the RHACS portal, use **http**. However, you will not be able to use the endpoint for service-to-service communication or for the **roxctl** CLI, because these clients require both gRPC and HTTP. Red Hat recommends that you do not specify a value of this key, to enable both HTTP and gRPC protocols for the endpoint. If you want to restrict an endpoint to Red Hat Advanced Cluster Security for Kubernetes services only, use the **clientAuth** option.

5 8 Use it to specify the TLS settings for the endpoint. If you do not specify a value, Red Hat Advanced Cluster Security for Kubernetes enables TLS with the default settings for all the following nested keys.

6 Use **true** to disable TLS on the specified endpoint. The default value is **false**. When you set it to **true**, you cannot specify values for **serverCerts** and **clientAuth**.

9 Specify a list of sources from which to configure server TLS certificates. The **serverCerts** list is order-dependent, it means that the first item in the list determines the certificate that Central uses by default, when there is no matching SNI (Server Name Indication). You can use this to specify multiple certificates and Central automatically selects the right certificate based on SNI. Acceptable values are:

- **default**: use the already configured custom TLS certificate if it exists.
- **service**: use the internal service certificate that Red Hat Advanced Cluster Security for Kubernetes generates.

10 Use it to configure the behavior of the TLS-enabled endpoint's client certificate authentication.

11 Use **true** to only allow clients with a valid client certificate. The default value is **false**. You can use **true** in conjunction with a the **certAuthorities** setting of **service** to only allow Red Hat Advanced Cluster Security for Kubernetes services to connect to this endpoint.

12 A list of CA to verify client certificates. The default value is **["service", "user"]**. The **certAuthorities** list is order-independent, it means that the position of the items in this list does not matter. Also, setting it as empty list **[]** disables client certificate authentication for the endpoint, which is different from leaving this value unset. Acceptable values are:

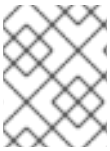
- **service**: CA for service certificates that Red Hat Advanced Cluster Security for Kubernetes generates.
- **user**: CAs configured by PKI authentication providers.

12.2. CONFIGURING ENDPOINTS DURING A NEW INSTALLATION

When you install Red Hat Advanced Cluster Security for Kubernetes by using the **roxctl** CLI, it creates a folder named **central-bundle**, which contains the necessary YAML manifests and scripts to deploy Central.

Procedure

1. After you generate the **central-bundle**, open the **./central-bundle/central/02-endpoints-config.yaml** file.
2. In this file, add your custom YAML configuration under the **data:** section of the key **endpoints.yaml**. Make sure that you maintain a 4 space indentation for the YAML configuration.
3. Continue the installation instructions as usual. Red Hat Advanced Cluster Security for Kubernetes uses the specified configuration.



NOTE

If you expose any additional ports other than the default port **8443**, you must create network policies that allow traffic on those additional ports.

12.3. CONFIGURING ENDPOINTS FOR AN EXISTING INSTANCE

You can configure endpoints for an existing instance of Red Hat Advanced Cluster Security for Kubernetes.

Procedure

1. Download the existing config map:

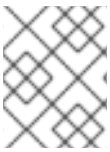
```
$ oc -n stackrox get cm/central-endpoints -o go-template='{{index .data "endpoints.yaml"}}' >
<directory_path>/central_endpoints.yaml
```

2. In the downloaded **central_endpoints.yaml** file, specify your custom YAML configuration.

3. Upload and apply the modified **central_endpoints.yaml** configuration file:

```
$ oc -n stackrox create cm central-endpoints --from-file=endpoints.yaml=<directory-
path>/central-endpoints.yaml -o yaml --dry-run | \
oc label -f - --local -o yaml app.kubernetes.io/name=stackrox | \
oc apply -f -
```

4. Restart Central.



NOTE

If you expose any additional ports other than the default port **8443**, you must create network policies that allow traffic on those additional ports.

12.3.1. Restarting the Central container

You can restart the Central container by killing the Central container or by deleting the Central pod.

Procedure

- Run the following command to kill the Central container:

NOTE

You must wait for at least 1 minute, until OpenShift Container Platform propagates your changes and restarts the Central container.

```
$ oc -n stackrox exec deploy/central -c central -- kill 1
```

- Or, run the following command to delete the Central pod:

```
$ oc -n stackrox delete pod -lapp=central
```

12.4. ENABLING TRAFFIC FLOW THROUGH CUSTOM PORTS

If you are exposing a port to another service running in the same cluster or to an ingress controller, you must only allow traffic from the services in your cluster or from the proxy of the ingress controller. Otherwise, if you are exposing a port by using a load balancer service, you might want to allow traffic from all sources, including external sources. Use the procedure listed in this section to allow traffic from all sources.

Procedure

1. Clone the **allow-ext-to-central** Kubernetes network policy:

```
$ oc -n stackrox get networkpolicy.networking.k8s.io/allow-ext-to-central -o yaml >
<directory_path>/allow-ext-to-central-custom-port.yaml
```

2. Use it as a reference to create your network policy, and in that policy, specify the port number you want to expose. Make sure to change the name of your network policy in the **metadata** section of the YAML file, so that it does not interfere with the built-in **allow-ext-to-central** policy.

CHAPTER 13. MONITORING WITH PROMETHEUS

Prometheus is an open-source monitoring and alerting platform. You can use it to monitor health and availability of Central and Sensor components of Red Hat Advanced Cluster Security for Kubernetes (RHACS). When you enable monitoring, RHACS creates a new monitoring service on port number **9090** and a network policy allowing inbound connections to that port.

13.1. MONITORING CENTRAL SERVICES USING THE RHACS OPERATOR

You can monitor Central services, Central and Scanner, by changing the configuration of the **Central** custom resource.

Procedure

1. On the OpenShift Container Platform web console, go to the **Operators → Installed Operators** page.
2. Select the Red Hat Advanced Cluster Security for Kubernetes Operator from the list of installed Operators.
3. Click on the **Central** tab.
4. From the list of Central instances, click on a Central instance for which you want to enable monitoring for.
5. Click on the **YAML** tab and update the YAML configuration:
   - For monitoring Central, enable the **central.monitoring.exposeEndpoint** configuration option for the **Central** custom resource.

     ```
     apiVersion: platform.stackrox.io/v1alpha1
     kind: Central
     ...
     spec:
       central:
         monitoring:
           exposeEndpoint: Enabled
     ...
     ```

   - For monitoring Scanner, enable the **scanner.monitoring.exposeEndpoint** configuration option for the **Central** custom resource.

     ```
     apiVersion: platform.stackrox.io/v1alpha1
     kind: Central
     ...
     spec:
       scanner:
         monitoring:
           exposeEndpoint: Enabled
     ...
     ```

6. Click **Save**.

Additional resources

Additional resources

- Central configuration options using the Operator

13.2. MONITORING CENTRAL SERVICES USING HELM CHARTS

You can monitor Central services, Central and Scanner, by changing the configuration options in the **central-services** Helm chart.

Procedure

1. Update the **values-public.yaml** configuration file with the following values:

```
central.exposeMonitoring: true
scanner.exposeMonitoring: true
```

2. Run the **helm upgrade** command and specify the configuration files using the **-f** option:

```
$ helm upgrade -n stackrox \
  stackrox-central-services rhacs/central-services \
  -f <path_to_values_public.yaml>
```

NOTE

You can also specify configuration values using the **--set** or **--set-file** parameters. However, these options are not saved, and it requires you to manually specify all the options again whenever you make changes.

Additional resources

- Changing configuration options after deploying the central-services Helm chart

13.3. MONITORING CENTRAL USING PROMETHEUS SERVICE MONITOR

If you are using the Prometheus Operator, you can use a service monitor to scrape the metrics from Red Hat Advanced Cluster Security for Kubernetes (RHACS).

NOTE

If you are not using the Prometheus operator, you must edit the Prometheus configuration files to receive the data from RHACS.

Procedure

1. Create a new **servicemonitor.yaml** file with the following content:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: prometheus-stackrox
  namespace: stackrox
spec:
```

```
endpoints:
  - interval: 30s
    port: monitoring
    scheme: http
selector:
  matchLabels:
    app.kubernetes.io/name: <stackrox-service> 1
```

1 The labels must match with the **Service** resource that you want to monitor. For example, **central** or **scanner**.

2. Apply the YAML to the cluster:

```
$ oc apply -f servicemonitor.yaml 1
```

1 If you use Kubernetes, enter **kubectl** instead of **oc**.

Verification

- Run the following command to check the status of service monitor:

```
$ oc get servicemonitor --namespace stackrox 1
```

1 If you use Kubernetes, enter **kubectl** instead of **oc**.

13.4. MANUALLY ENABLE MONITORING

IMPORTANT

If you have already enabled monitoring using **central.monitoring.exposeEndpoint: Enabled** or by using the **central.exposeMonitoring: true** Helm chart customization option, do not run the commands in this section.

Before you can monitor Red Hat Advanced Cluster Security for Kubernetes, you must enable monitoring.

Procedure

1. Patch the services to expose the port number **9090**.
   a. Patch the Sensor service:

```
$ oc -n stackrox patch svc/sensor -p '{"spec":{"ports":
[{"name":"monitoring","port":9090,"protocol":"TCP","targetPort":9090}]}}' 1
```

1 If you use Kubernetes, enter **kubectl** instead of **oc**.

   b. Patch the Central service:

```
$ oc -n stackrox patch svc/central -p '{"spec":{"ports":
[{"name":"monitoring","port":9090,"protocol":"TCP","targetPort":9090}]}}'
```

2. Modify network policies to allow ingress.

```
$ oc apply -f - <<EOF (1)
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  labels:
    app.kubernetes.io/name: stackrox
  name: allow-monitoring
  namespace: stackrox
spec:
  ingress:
  - ports:
    - port: 9090
      protocol: TCP
  podSelector:
    matchExpressions:
    - {key: app, operator: In, values: [central, sensor, collector]}
  policyTypes:
  - Ingress
EOF
```

(1) If you use Kubernetes, enter **kubectl** instead of **oc**.

13.5. CUSTOMIZING THE DEFAULT PORT

To customize the port used for Prometheus metrics in Red Hat Advanced Cluster Security for Kubernetes Central and Sensor, you can use the **ROX_METRICS_PORT** environment variable.

Procedure

- Set the **ROX_METRICS_PORT** environment variable:

```
$ oc -n stackrox set env deploy/central ROX_METRICS_PORT=<value> (1)
```

(1) If you use Kubernetes, enter **kubectl** instead of **oc**.

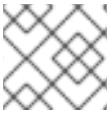
NOTE

You can specify the **<value>** for the **ROX_METRICS_PORT** environment variable as:

- **disabled** to disable monitoring.
- **:<port_number>** to bind it to a wildcard address.
- **<address>:<port_number>** to use specific address and port number. You can also specify an IPv6 address by using square brackets, for example, **[2001:db8::1234]:9090**.

CHAPTER 14. CONFIGURING AUDIT LOGGING

Red Hat Advanced Cluster Security for Kubernetes provides audit logging features that you can use to check all the changes made in Red Hat Advanced Cluster Security for Kubernetes. The audit log captures all the **PUT** and **POST** events, which are modifications to Red Hat Advanced Cluster Security for Kubernetes. Use this information to troubleshoot a problem or to keep a record of important events, such as changes to roles and permissions. With audit logging you get a complete picture of all normal and abnormal events that happened on Red Hat Advanced Cluster Security for Kubernetes.



NOTE

Audit logging is not enabled by default. You must enable audit logging manually.

WARNING

Currently there is no message delivery guarantee for audit log messages.

14.1. ENABLING AUDIT LOGGING

When you enable audit logging, every time there is a modification, Red Hat Advanced Cluster Security for Kubernetes sends an HTTP POST message (in JSON format) to the configured system.

Prerequisites

- Configure Splunk or another webhook receiver to handle Red Hat Advanced Cluster Security for Kubernetes log messages.
- You must have **write** permission enabled on the **Notifiers** resource for your role.

Procedure

1. On the RHACS portal, navigate to **Platform Configuration → Integrations**.
2. Scroll down to the **Notifier Integrations** section and select **Generic Webhook** or **Splunk**.
3. Fill in the required information and turn on the **Enable Audit Logging** toggle.

14.2. SAMPLE AUDIT LOG MESSAGE

The log message has the following format:

```
{
  "headers": {
    "Accept-Encoding": [
      "gzip"
    ],
    "Content-Length": [
      "586"
    ],
```

```
    "Content-Type": [
      "application/json"
    ],
    "User-Agent": [
      "Go-http-client/1.1"
    ]
  },
  "data": {
    "audit": {
      "interaction": "CREATE",
      "method": "UI",
      "request": {
        "endpoint": "/v1/notifiers",
        "method": "POST",
        "payload": {
          "@type": "storage.Notifier",
          "enabled": true,
          "generic": {
            "auditLoggingEnabled": true,
            "endpoint": "http://samplewebhookserver.com:8080"
          },
          "id": "b53232ee-b13e-47e0-b077-1e383c84aa07",
          "name": "Webhook",
          "type": "generic",
          "uiEndpoint": "https://localhost:8000"
        }
      },
      "status": "REQUEST_SUCCEEDED",
      "time": "2019-05-28T16:07:05.500171300Z",
      "user": {
        "friendlyName": "John Doe",
        "role": {
          "globalAccess": "READ_WRITE_ACCESS",
          "name": "Admin"
        },
        "username": "john.doe@example.com"
      }
    }
  }
}
```

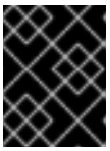
CHAPTER 15. CONFIGURING API TOKENS

Red Hat Advanced Cluster Security for Kubernetes (RHACS) requires API tokens for some system integrations, authentication processes, and system functions. You can configure tokens using the RHACS web interface.

15.1. CREATING AN API TOKEN

Procedure

1. In the RHACS portal, navigate to **Platform Configuration → Integrations**.
2. Scroll to the **Authentication Tokens** category, and then click **API Token**.
3. Click **Generate Token**.
4. Enter a name for the token and select a role that provides the required level of access (for example, **Continuous Integration** or **Sensor Creator**).
5. Click **Generate**.



IMPORTANT

Copy the generated token and securely store it. You will not be able to view it again.

15.2. ABOUT API TOKEN EXPIRATION

API tokens expire one year from the creation date. RHACS alerts you in the web interface and by sending log messages to Central when a token will expire in less than one week. The log message process runs once an hour. Once a day, the process lists the tokens that are expiring and creates a log message for each one. Log messages are issued once a day and appear in Central logs.

Logs have the format as shown in the following example:

```
Warn: API Token [token name] (ID [token ID]) will expire in less than X days.
```

You can change the default settings for the log message process by configuring the environment variables shown in the following table:

Environment variable	Default value	Description
ROX_TOKEN_EXPIRATION_NOTIFIER_INTERVAL	1h (1 hour)	The frequency at which the log message background loop that lists tokens and creates the logs will run.
ROX_TOKEN_EXPIRATION_NOTIFIER_BACKOFF_INTERVAL	24h (1 day)	The frequency at which the loop lists tokens and issues notifications.

ROX_TOKEN_EXPIRATION_DETECTION_WINDOW	168h (1 week)	The time period before expiration of the token that will cause the notification to be generated.

CHAPTER 16. USING DECLARATIVE CONFIGURATION

With declarative configuration, you can update configurations by storing them in files in repositories and apply them to the system. Declarative configuration is useful, for example, if you are using a GitOps workflow. You can currently use declarative configuration in Red Hat Advanced Cluster Security for Kubernetes (RHACS) for authentication and authorization resources such as authentication providers, roles, permission sets, and access scopes.

To use declarative configuration, you create YAML files that contain configuration information about authentication and authorization resources. These files, or configurations, are added to RHACS by using a mount point during Central installation. See the installation documentation in the "Additional references" section for more information on configuring mount points when installing RHACS.

The configuration files used with declarative configuration are stored in config maps or secrets, depending on the type of resource. Store configurations for authentication providers in a secret for greater security. You can store other configurations in config maps.

A single config map or secret can contain more than one configuration of multiple resource types. This allows you to limit the number of volume mounts for the Central instance.

16.1. RESTRICTIONS FOR RESOURCES CREATED FROM DECLARATIVE CONFIGURATION

Because resources can reference other resources (for example, a role can reference both a permission set and access scope), the following restrictions for references apply:

- A declarative configuration can only reference a resource that is either also created declaratively or a system RHACS resource; for example, a resource such as the **Admin** or **Analyst** system role or permission set.
- All references between resources use names to identify the resource; therefore, all names within the same resource type must be unique.
- Resources created from declarative configuration can only be modified or deleted by altering the declarative configuration files. You cannot change these resources by using the RHACS portal or the API.

16.2. CREATING DECLARATIVE CONFIGURATIONS

Use **roxctl** to create the YAML files that store the configurations, create a config map from the files, and apply the config map.

Prerequisites

- You have added the mount for the config map or secret during the installation of Central. In this example, the config map is called "declarative-configs". See the installation documentation listed in the "Additional resources" section for more information.

Procedure

1. Create the permission set by entering the following command. This example creates a permission set named "restricted" and is saved as the **permission-set.yaml** file. It sets read and write access for the **Administration** resource and read access to the Access resource.

```
$ roxctl declarative-config create permission-set \
--name="restricted" \
--description="Restriction permission set that only allows \
access to Administration and Access resources" \
--resource-with-access=Administration=READ_WRITE_ACCESS \
--resource-with-access=Access=READ_ACCESS > permission-set.yaml
```

2. Create the role that allows access to the **Administration** and **Access** resources by entering the following command. This example creates a role named "restricted" and is saved as the **role.yaml** file.

```
$ roxctl declarative-config create role \
--name="restricted" \
--description="Restricted role that only allows access to Administration and Access" \
--permission-set="restricted" \
--access-scope="Unrestricted" > role.yaml
```

3. Create a config map from the two YAML files that were created in the earlier steps by entering the following command. This example creates the **declarative-configurations** config map.

```
$ kubectl create configmap declarative-configurations \ 1
--from-file permission-set.yaml --from-file role.yaml \
-o yaml --namespace=stackrox > declarative-configs.yaml
```

1 For OpenShift Container Platform, use **oc create**.

4. Apply the config map by entering the following command:

```
$ kubectl apply -f declarative-configs.yaml 1
```

1 For OpenShift Container Platform, use **oc apply**.

After you apply the config map, configuration information extracted from Central creates the resources.

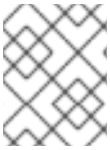
NOTE

Although the watch interval is 5 seconds, as described in the following paragraph, there can be a delay in propagating changes from the config map to the Central mount.

You can configure the following intervals to specify how declarative configurations interact with Central:

- Configuration watch interval: The interval for Central to check for changes is every 5 seconds. You can configure this interval by using the **ROX_DECLARATIVE_CONFIG_WATCH_INTERVAL** environment variable.
- Reconciliation interval: By default, declarative configuration reconciliation with Central occurs every 20 seconds. You can configure this interval by using the **ROX_DECLARATIVE_CONFIG_RECONCILE_INTERVAL** environment variable.

After creating authentication and authorization resources by using declarative configuration, you can view them in the **Access Control** page in the RHACS web portal. The **Origin** field indicates **Declarative** if the resource was created by using declarative configuration.



NOTE

You cannot edit resources created from declarative configurations in the RHACS web portal. You must edit the configuration files directly to make changes to these resources.

You can view the status of declarative configurations by navigating to **Platform Configuration → System Health** and scrolling to the **Declarative configuration** section.

16.3. DECLARATIVE CONFIGURATION EXAMPLES

You can create declarative configurations by using the following examples as a guide. Use the **roxctl declarative-config lint** command to verify that your configurations are valid.

16.3.1. Declarative configuration authentication provider example

Declarative configuration authentication provider example

```
name: A sample auth provider
minimumRole: Analyst (1)
uiEndpoint: central.custom-domain.com:443 (2)
extraUIEndpoints: (3)
  - central-alt.custom-domain.com:443
groups: (4)
  - key: email (5)
    value: example@example.com
    role: Admin (6)
  - key: groups
    value: reviewers
    role: Analyst
requiredAttributes: (7)
  - key: org_id
    value: "12345"
oidc: (8)
  issuer: sample.issuer.com (9)
  mode: auto (10)
  clientID: CLIENT_ID
  clientSecret: CLIENT_SECRET
clientSecret: CLIENT_SECRET
iap: (11)
  audience: audience
saml: (12)
  spIssuer: sample.issuer.com
  metadataURL: sample.provider.com/metadata
saml: (13)
  spIssuer: sample.issuer.com
  cert: | (14)
  ssoURL: saml.provider.com
  idpIssuer: idp.issuer.com
```

```
userpki:
  certificateAuthorities: | (15)
  certificate (16)
```

1. Identifies the minimum role that will be assigned by default to any user logging in. If left blank, the value is **None**.
2. Use the user interface endpoint of your Central instance.
3. If your Central instance is exposed to different endpoints, specify them here.
4. These fields map users to specific roles, based on their attributes.
5. The key can be any claim returned from the authentication provider.
6. Identifies the role that the users are given. You can use a default role or a declaratively-created role.
7. Optional: Use these fields if attributes returned from the authentication provider are required; for example, if the audience is limited to a specific organization or group.
8. This section is required only for OpenID Connect (OIDC) authentication providers.
9. Identifies the expected issuer for the token.
10. Identifies the OIDC callback mode. Possible values are **auto**, **post**, **query**, and **fragment**. The preferred value is **auto**.
11. This section is required only for Google Identity-Aware Proxy (IAP) authentication providers.
12. This section is required only for Security Assertion Markup Language (SAML) 2.0 dynamic configuration authentication providers.
13. This section is required only for SAML 2.0 static configuration authentication providers.
14. Include the certificate in Privacy Enhanced Mail (PEM) format.
15. This section is required only for authentication with user certificates.
16. Include the certificate in PEM format.

16.3.2. Declarative configuration permission set example

Declarative configuration permission set example

```
name: A sample permission set
description: A sample permission set created declaratively
resources:
- resource: Integration (1)
  access: READ_ACCESS (2)
- resource: Administration
  access: READ_WRITE_ACCESS
```

1. For a full list of supported resources, navigate to **Access Control → Permission Sets**.

2 Access can be either **READ_ACCESS** or **READ_WRITE_ACCESS**.

16.3.3. Declarative configuration access scope example

Declarative configuration access scope example

```
name: A sample access scope
description: A sample access scope created declaratively
rules:
  included:
    - cluster: secured-cluster-A 1
      namespaces:
        - namespaceA
    - cluster: secured-cluster-B 2
  clusterLabelSelectors:
    - requirements:
        key: kubernetes.io/metadata.name
        operator: IN 3
        values:
        - production
        - staging
        - environment
```

1 Identifies a cluster where only specific namespaces are included within the access scope.

2 Identifies a cluster where all namespaces are included within the access scope.

3 Identifies the Operator to use for the label selection. Valid values are **IN**, **NOT_IN**, **EXISTS**, and **NOT_EXISTS**.

16.3.4. Declarative configuration role example

Declarative configuration role example

```
name: A sample role
description: A sample role created declaratively
permissionSet: A sample permission set 1
accessScope: Unrestricted 2
```

1 Name of the permission set; can be either one of the system permission sets or a declaratively-created permission set.

2 Name of the access scope; can be either one of the system access scopes or a declaratively-created access scope.

16.4. TROUBLESHOOTING DECLARATIVE CONFIGURATION

You can use the error messages displayed in the **Declarative configuration** section of the **Platform Configuration → System Health** page to help in troubleshooting. The **roxctl declarative-config** command also includes a **lint** option to validate the configuration file and help you detect errors.

The error messages displayed in the **Declarative configuration** section of the **Platform Configuration → System Health** page provide information about issues with declarative configurations. Problems with declarative configurations can be caused by the following conditions:

- The format of the configuration file is not in valid YAML.
- The configuration file contains invalid values, such as invalid access within a permission set.
- Invalid storage constraints exist, such as resource names are not unique or the configuration contains invalid references to a resource.

To validate configuration files, check for errors in configuration files, and make sure that there are no invalid storage constraints when creating and updating configuration files, use the **roxctl declarative-config lint** command.

To troubleshoot a storage constraint during deletion, check if the resource has been marked as **Declarative Orphaned**. This indicates that the declarative configuration referenced by a resource was deleted (for example, if the declarative configuration for a permission set that was referenced by a role was deleted). To correct this error, edit the resource to point to a new permission set, or restore the declarative configuration that was deleted.

16.5. ADDITIONAL RESOURCES

- Install Central using Helm charts with customizations (Red Hat OpenShift)
- Install Central using Helm charts with customizations (other Kubernetes platforms)