Red Hat Advanced Cluster Security for Kubernetes 4.3

RHACS Cloud Service

About the RHACS Cloud Service

Last Updated: 2024-07-16

Red Hat Advanced Cluster Security for Kubernetes 4.3 RHACS Cloud Service

About the RHACS Cloud Service

Legal Notice

Abstract

Guidance on understanding the RHACS Cloud Service.

Table of Contents

CHAPTER 1. RHACS CLOUD SERVICE SERVICE DESCRIPTION

1.1. INTRODUCTION TO RHACS

Red Hat Advanced Cluster Security for Kubernetes (RHACS) is an enterprise-ready, Kubernetes-native container security solution that helps you build, deploy, and run cloud-native applications more securely.

Red Hat Advanced Cluster Security Cloud Service (RHACS Cloud Service) provides Kubernetes-native security as a service. With RHACS Cloud Service, Red Hat maintains, upgrades, and manages your Central services.

Central services include the user interface (UI), data storage, RHACS application programming interface (API), and image scanning capabilities. You deploy your Central service through the Red Hat Hybrid Cloud Console. When you create a new ACS instance, Red Hat creates your individual control plane for RHACS.

RHACS Cloud Service allows you to secure self-managed clusters that communicate with a Central instance. The clusters you secure, called Secured Clusters, are managed by you, and not by Red Hat. Secured Cluster services include optional vulnerability scanning services, admission control services, and data collection services used for runtime monitoring and compliance. You install Secured Cluster services on any OpenShift or Kubernetes cluster you want to secure.

1.2. BILLING

Customers can purchase a RHACS Cloud Service subscription on the Amazon Web Services (AWS) marketplace. The service cost is charged hourly per secured core, or vCPU of a node belonging to a secured cluster.

> **Example 1.1. Subscription cost example**
>
> If you have established a connection to two secured clusters, each with 5 identical nodes with 8 vCPUs (such as Amazon EC2 m7g.2xlarge), the total number of secured cores is 80 (2 x 5 x 8 = 80).

1.3. SECURITY AND COMPLIANCE

All Red Hat Advanced Cluster Security Cloud Service (RHACS Cloud Service) data in the Central instance is encrypted in transit and at rest. The data is stored in secure storage with full replication and high availability together with regularly scheduled backups. The RHACS Cloud Service is available through cloud data centers that ensure optimal performance and the ability to meet data residency requirements.

1.3.1. Authentication provider

When you create a Central instance using Red Hat Hybrid Cloud Console , authentication for the cluster administrator is configured as part of the process. Customers must manage all access to the Central instance as part of their integrated solution. For more information about the available authentication methods, see Understanding authentication providers.

The default identity provider in RHACS Cloud Service is Red Hat Single Sign-On (SSO). For more information about authentication using Red Hat SSO, see Default access to ACS Console .

1.3.2. Regulatory compliance

For the latest regulatory compliance information, see Understanding process and security for OpenShift Dedicated.

1.4. METRICS AND LOGGING

1.4.1. Service metrics

Service metrics are internal only. Red Hat provides and maintains the service at the agreed upon level. Service metrics are accessible only to authorized Red Hat personnel. For more information, see PRODUCT APPENDIX 4 RED HAT ONLINE SERVICES .

1.4.2. Customer metrics

Core usage capacity metrics are available either through Subscription Watch or the Subscriptions page.

1.4.3. Service logging

System logs for all components of the Red Hat Advanced Cluster Security Cloud Service (RHACS Cloud Service) are internal and available only to Red Hat personnel. Red Hat does not provide user access to component logs. For more information, see PRODUCT APPENDIX 4 RED HAT ONLINE SERVICES .

1.5. SCALABILITY AND SERVICE LEVEL

Red Hat Advanced Cluster Security Cloud Service (RHACS Cloud Service) has set limits on the number of cores or clusters it can protect. The limits are based on the resources available in the Secured Cluster and the limits on usability.

1.5.1. Service Level Objective and Agreement

For more information about Service Level Objectives (SLOs) and Service Level Agreements (SLAs), see PRODUCT APPENDIX 4 RED HAT ONLINE SERVICES .

1.6. UPDATES AND UPGRADES

Red Hat makes a commercially reasonable effort to notify customers prior to updates and upgrades that impact service. The decision regarding the need for a Service update to the Central instance and its timing is the sole responsibility of Red Hat.

Customers have no control over when a Central service update occurs. For more information, see PRODUCT APPENDIX 4 RED HAT ONLINE SERVICES . Upgrades to the version of Red Hat Advanced Cluster Security Cloud Service (RHACS Cloud Service) are considered part of the service update.

Customers are responsible for timely RHACS Secured Cluster services upgrades that are required to maintain compatibility with RHACS Cloud Service.

Red Hat recommends enabling automatic upgrades for Secured Clusters that are connected to RHACS Cloud Service.

See the Red Hat Advanced Cluster Security for Kubernetes Support Matrix for more information about upgrade versions.

1.7. AVAILABILITY

Availability and disaster avoidance are extremely important aspects of any security platform. Red Hat Advanced Cluster Security Cloud Service (RHACS Cloud Service) provides numerous protections against failures at multiple levels. To account for possible cloud provider failures, Red Hat established multiple availability zones.

1.7.1. Backup and Disaster Recovery

All RHACS Cloud Service clusters are backed up using Database backups. This also applies to customer data stored in the Central database.

All snapshots are created using the appropriate cloud provider snapshot APIs, encrypted and then uploaded to secure object storage, which for Amazon Web Services (AWS) is an S3 bucket.

- Red Hat does not commit to a Recovery Point Objective (RPO) or Recovery Time Objective (RTO). For more information, see PRODUCT APPENDIX 4 RED HAT ONLINE SERVICES .
- Site Reliability Engineering performs backups only as a precautionary measure. They are stored in the same region as the cluster.
- Customers should deploy multiple availability zone Secured Clusters with workloads that follow Kubernetes best practices to ensure high availability within a region.

1.7.2. Getting Support

The RHACS Cloud Service includes Red Hat Standard and Premium support, which you can access using the Red Hat Customer Portal . You can open support tickets for the product "Red Hat Advanced Cluster Security Cloud Service".

Red Hat support responds to support tickets submitted by Limited Availability customers, while Red Hat Site Reliability Engineers (SREs) proactively monitor the health of Red Hat Advanced Cluster Security Cloud Service (RHACS Cloud Service).

In addition, a Red Hat Business Unit Solution Architect (BU SA) acts as a hands-on technical liaison between customer experts, Red Hat support, and Red Hat SREs, supporting customers with limited availability.

- For more information on what is covered with RHACS Cloud Service support, see Scope of Coverage Details.
- For more information on the terms of service for production support, see Production Support Terms of Service.

1.7.3. Service Removal

You can delete RHACS Cloud Service using the default delete operations from the Red Hat Hybrid Cloud Console. Deleting the RHACS Cloud Service Central instance automatically removes all RHACS components. Deleting is not reversible.

1.8. PRICING

Red Hat does charge a subscription fee for Red Hat Advanced Cluster Security Cloud Service (RHACS Cloud Service) during the Limited Availability. For more information, see PRODUCT APPENDIX 4 RED HAT ONLINE SERVICES.

1.9. SERVICE LEVEL AGREEMENT

For more information about the Service Level Agreements (SLAs) offered for Red Hat Advanced Cluster Security Cloud Service (RHACS Cloud Service), see PRODUCT APPENDIX 4 RED HAT ONLINE SERVICES.

1.10. TIMELINES

Limited Availability

Production support for Red Hat Advanced Cluster Security Cloud Service (RHACS Cloud Service) is offered to a limited number of customers. For more information, see PRODUCT APPENDIX 4 RED HAT ONLINE SERVICES.

General Availability

Production support for RHACS Cloud Service is offered to all RHACS Cloud Service customers. For more information, see PRODUCT APPENDIX 4 RED HAT ONLINE SERVICES .

CHAPTER 2. GETTING STARTED WITH RHACS CLOUD SERVICE

Red Hat Advanced Cluster Security Cloud Service (RHACS Cloud Service) provides security services for your Red Hat OpenShift and Kubernetes clusters. See the Red Hat Advanced Cluster Security for Kubernetes Support Matrix for more information on supported platforms for secured clusters.

Prerequisites

- Ensure that you can access the **Advanced Cluster Security** menu option from the Red Hat Hybrid Cloud Console.

> **NOTE**
>
> To access the RHACS Cloud Service console, you need your Red Hat Single Sign-On (SSO) credentials, or credentials for another identity provider if that has been configured. See Default access to the ACS console .

2.1. HIGH-LEVEL OVERVIEW OF INSTALLATION STEPS

The following sections provide an overview of installation steps and links to the relevant documentation.

2.1.1. Securing Red Hat OpenShift clusters

To secure Red Hat OpenShift clusters by using the Operator, perform the following steps:

1. Verify that the clusters you want to secure meet the requirements.
2. In the Red Hat Hybrid Cloud Console, create an **ACS Instance**.
3. On each Red Hat OpenShift cluster you want to secure, create a project named **stackrox**. This project will contain the resources for RHACS Cloud Service secured clusters.
4. In the ACS Console, create an init bundle. The init bundle contains secrets that allow communication between RHACS Cloud Service secured clusters and the ACS Console.
5. On each Red Hat OpenShift cluster, apply the init bundle by using it to create resources.
6. On each Red Hat OpenShift cluster, install the RHACS Operator.
7. On each Red Hat OpenShift cluster, install secured cluster resources in the **stackrox** project by using the Operator.
8. Verify installation by ensuring that your secured clusters can communicate with the ACS instance.

To secure Red Hat OpenShift clusters by using Helm charts or the **roxctl** CLI, perform the following steps:

1. Verify that the clusters you want to secure meet the requirements.
2. In the Red Hat Hybrid Cloud Console, create an **ACS Instance**.
3. On each Red Hat OpenShift cluster you want to secure, create a project named **stackrox**. This project will contain the resources for RHACS Cloud Service secured clusters.

4. In the ACS Console, create an init bundle. The init bundle contains secrets that allow communication between RHACS Cloud Service secured clusters and the ACS Console.

5. On each Red Hat OpenShift cluster, apply the init bundle by using it to create resources.

6. On each Red Hat OpenShift cluster, install secured cluster resources in the **stackrox** project by using Helm charts or by using the **roxctl** CLI.

7. Verify installation by ensuring that your secured clusters can communicate with the ACS instance.

2.1.2. Securing Kubernetes clusters

To secure Kubernetes clusters, perform the following steps:

1. Verify that the clusters you want to secure meet the requirements.

2. In the Red Hat Hybrid Cloud Console, create an **ACS Instance**.

3. In the ACS Console, create an init bundle. The init bundle contains secrets that allow communication between RHACS Cloud Service secured clusters and the ACS Console.

4. On each Kubernetes cluster, apply the init bundle by using it to create resources.

5. On each Kubernetes cluster, install secured cluster resources by using Helm charts or the **roxctl** CLI.

6. Verify installation by ensuring that your secured clusters can communicate with the ACS instance.

2.2. DEFAULT ACCESS TO THE ACS CONSOLE

By default, the authentication mechanism available to users is authentication by using Red Hat Single Sign-On (SSO). You cannot delete or change the Red Hat SSO authentication provider. However, you can change the minimum access role and add additional rules, or add another identity provider.

NOTE

To learn how authentication providers work in ACS, see Understanding authentication providers.

A dedicated OIDC client of **sso.redhat.com** is created for each ACS Console. All OIDC clients share the same **sso.redhat.com** realm. Claims from the token issued by **sso.redhat.com** are mapped to an ACS-issued token as follows:

- **realm_access.roles** to **groups**
- **org_id** to **rh_org_id**
- **is_org_admin** to **rh_is_org_admin**
- **sub** to **userid**

The built-in Red Hat SSO authentication provider has the required attribute **rh_org_id** set to the organization ID assigned to account of the user who created the RHACS Cloud Service instance. This is the ID of the organizational account the user is a part of. This can be thought of as the "tenant" the user is under and owned by. Only users with the same organizational account can access the ACS console by using the Red Hat SSO authentication provider.

NOTE

To gain more control over access to your ACS Console, configure another identity provider instead of relying on the Red Hat SSO authentication provider. For more information, see Understanding authentication providers. To configure the other authentication provider to be the first authentication option on the login page, its name should be lexicographically smaller than **Red Hat SSO**.

The minimum access role is set to **None**. Assigning a different value to this field gives access to the RHACS Cloud Service instance to all users with the same organizational account.

Other rules that are set up in the built-in Red Hat SSO authentication provider include the following:

- Rule mapping your **userid** to **Admin**
- Rules mapping administrators of the organization to **Admin**

You can add more rules to grant access to the ACS Console to someone else with the same organizational account. For example, you can use **email** as a key.

CHAPTER 3. DEFAULT RESOURCE REQUIREMENTS FOR RED HAT ADVANCED CLUSTER SECURITY CLOUD SERVICE

3.1. GENERAL REQUIREMENTS

RHACS has some system requirements that must be met before you can install it.

> **WARNING**
>
> You must not install Red Hat Advanced Cluster Security for Kubernetes on:
>
> - Amazon Elastic File System (Amazon EFS). Use the Amazon Elastic Block Store (Amazon EBS) with the default **gp2** volume type instead.
> - Older CPUs that do not have the Streaming SIMD Extensions (SSE) 4.2 instruction set. For example, Intel processors older than *Sandy Bridge* and AMD processors older than *Bulldozer*. (These processors were released in 2011.)

To install Red Hat Advanced Cluster Security for Kubernetes, you must have one of the following systems:

- OpenShift Container Platform version 4.11 or later, and cluster nodes with a supported operating system of Red Hat Enterprise Linux CoreOS (RHCOS) or Red Hat Enterprise Linux (RHEL).
- a supported managed Kubernetes platform, and cluster nodes with a supported operating system of Amazon Linux, CentOS, Container-Optimized OS from Google, Red Hat Enterprise Linux CoreOS (RHCOS), Debian, Red Hat Enterprise Linux (RHEL), or Ubuntu.
  For more information, see Red Hat Advanced Cluster Security for Kubernetes Support Policy .

Cluster nodes minimum requirements:

- **Architecture**: **amd64**, **ppc64le**, or **s390x**

  > **NOTE**
  >
  > Starting with RHACS 4.3, both Central and secured cluster services are supported on IBM Power(**ppc64le**), IBM Z(**s390x**), and IBM® LinuxONE(**s390x**) clusters.

- **Processor**: 3 CPU cores
- **Memory**: 6 GiB of RAM

  > **NOTE**
  >
  > See the default memory and CPU requirements for each component and ensure that the node size can support them.

Persistent storage by using persistent volume claim (PVC):

- Use Solid-State Drives (SSDs) for best performance. However, you can use another storage type if you do not have SSDs available.

IMPORTANT

You must not use Ceph FS storage with Red Hat Advanced Cluster Security for Kubernetes. Red Hat recommends using RBD block mode PVCs for Red Hat Advanced Cluster Security for Kubernetes.

To install using Helm charts:

- You must have Helm command-line interface (CLI) v3.2 or newer, if you are installing or configuring Red Hat Advanced Cluster Security for Kubernetes using Helm charts. Use the **helm version** command to verify the version of Helm you have installed.

- You must have access to the Red Hat Container Registry. For information about downloading images from **registry.redhat.io**, see Red Hat Container Registry Authentication .

3.2. SECURED CLUSTER SERVICES

Secured cluster services contain the following components:

- Sensor

- Admission controller

- Collector

3.2.1. Sensor

Sensor monitors your Kubernetes and OpenShift Container Platform clusters. These services currently deploy in a single deployment, which handles interactions with the Kubernetes API and coordinates with Collector.

Memory and CPU requirements

The following table lists the minimum memory and storage values required to install and run sensor on secured clusters.

Sensor	CPU	Memory
Request	2 cores	4 GiB
Limit	4 cores	8 GiB

3.2.2. Admission controller

The Admission controller prevents users from creating workloads that violate policies you configure.

Memory and CPU requirements

By default, the admission control service runs 3 replicas. The following table lists the request and limits for each replica.

Admission controller	CPU	Memory
Request	0.05 cores	100 MiB
Limit	0.5 cores	500 MiB

3.2.3. Collector

Collector monitors runtime activity on each node in your secured clusters. It connects to Sensor to report this information. The collector pod has three containers. The first container is collector, which actually monitors and reports the runtime activity on the node. The other two are compliance and node-inventory.

Collection requirements

To use the **CORE_BPF** collection method, the base kernel must support BTF, and the BTF file must be available to collector. In general, the kernel version must be later than 5.8 (4.18 for RHEL nodes) and the **CONFIG_DEBUG_INFO_BTF** configuration option must be set.

Collector looks for the BTF file in the standard locations shown in the following list:

Example 3.1. BTF file locations

```
/sys/kernel/btf/vmlinux
/boot/vmlinux-<kernel-version>
/lib/modules/<kernel-version>/vmlinux-<kernel-version>
/lib/modules/<kernel-version>/build/vmlinux
/usr/lib/modules/<kernel-version>/kernel/vmlinux
/usr/lib/debug/boot/vmlinux-<kernel-version>
/usr/lib/debug/boot/vmlinux-<kernel-version>.debug
/usr/lib/debug/lib/modules/<kernel-version>/vmlinux
```

If any of these files exists, it is likely that the kernel has BTF support and **CORE_BPF** is configurable.

Memory and CPU requirements

By default, the admission control service runs 3 replicas. The following table lists the request and limits for each replica.

Collector		CPU	Memory
Collector Container	**Request**	0.05 cores	320 MiB
	Limit	0.75 cores	1000 MiB
Compliance Container	**Request**	0.01 cores	10 MiB
	Limit	1 core	2000 MiB

Collector		CPU	Memory
Node-Inventory Container	**Request**	0.01 cores	10 MiB
	Limit	1 core	500 MiB
Total	**Request**	0.07 cores	340 MiB
	Limit	2.75 cores	3500 MiB

CHAPTER 4. RECOMMENDED RESOURCE REQUIREMENTS FOR RED HAT ADVANCED CLUSTER SECURITY CLOUD SERVICE

The recommended resource guidelines were developed by performing a focused test that created the following objects across a given number of namespaces:

- 10 deployments, with 3 pod replicas in a sleep state, mounting 4 secrets, 4 config maps
- 10 services, each one pointing to the TCP/8080 and TCP/8443 ports of one of the previous deployments
- 1 route pointing to the first of the previous services
- 10 secrets containing 2048 random string characters
- 10 config maps containing 2048 random string characters

During the analysis of results, the number of deployments was identified as a primary factor for increasing of used resources. The number of deployments was used for the estimation of required resources.

Additional resources

- Default resource requirements

4.1. SECURED CLUSTER SERVICES

Secured cluster services contain the following components:

- Sensor
- Admission controller
- Collector

NOTE

Collector component is not included on this page. Required resource requirements are listed on the default resource requirements page.

4.1.1. Sensor

Sensor monitors your Kubernetes and OpenShift Container Platform clusters. These services currently deploy in a single deployment, which handles interactions with the Kubernetes API and coordinates with Collector.

Memory and CPU requirements

The following table lists the minimum memory and CPU values required to run Sensor on a secured cluster.

Deployments	Pods per deployment	CPU	Memory
< 25,000	3	2 cores	8 GiB
< 50,000	3	2 cores	16 GiB

4.1.2. Admission controller

The admission controller prevents users from creating workloads that violate policies that you configure.

Memory and CPU requirements

The following table lists the minimum memory and CPU values required to run the admission controller on a secured cluster.

Deployments	Pods per deployment	CPU	Memory
< 25,000	3	0.5 cores	600 MiB
< 50,000	3	0.5 cores	1200 MiB

CHAPTER 5. SETTING UP RHACS CLOUD SERVICE WITH RED HAT OPENSHIFT SECURED CLUSTERS

5.1. CREATING A RHACS CLOUD INSTANCE ON RED HAT CLOUD

Access Red Hat Advanced Cluster Security Cloud Service (RHACS Cloud Service) by selecting an instance in the Red Hat Hybrid Cloud Console. An **ACS instance** contains the RHACS Cloud Service management interface and services that Red Hat configures and manages for you. The management interface connects to your secured clusters, which contain the services that scan and collect information about vulnerabilities. One instance can connect to and monitor many clusters.

5.1.1. Creating an instance in the console

In the Red Hat Hybrid Cloud Console, create an **ACS instance** to connect to your secured clusters.

Procedure

To create an **ACS instance**:

1. Log in to the Red Hat Hybrid Cloud Console.

2. From the navigation menu, select **Advanced Cluster Security → ACS Instances**.

3. Select **Create ACS instance** and enter information into the displayed fields or select the appropriate option from the drop-down list:

   - **Name**: Enter the name of your **ACS instance**. An **ACS instance** contains the RHACS Central component, also referred to as "Central", which includes the RHACS Cloud Service management interface and services that are configured and managed by Red Hat. You manage your secured clusters that communicate with Central. You can connect many secured clusters to one instance.

   - **Cloud provider**: The cloud provider where Central is located. Select **AWS**.

   - **Cloud region**: The region for your cloud provider where Central is located. Select one of the following regions:

     - US-East, N. Virginia

     - Europe, Ireland

   - **Availability zones**: Use the default value (**Multi**).

4. Click **Create instance**.

5.1.2. Next steps

- On each Red Hat OpenShift cluster you want to secure, create a project named **stackrox**. This project will contain the resources for RHACS Cloud Service secured clusters.

5.2. CREATING A PROJECT ON YOUR RED HAT OPENSHIFT SECURED CLUSTER

Create a project on each Red Hat OpenShift cluster that you want to secure. You then use this project to install RHACS Cloud Service resources by using the Operator or Helm charts.

5.2.1. Creating a project on your cluster

Procedure

- In your OpenShift Container Platform cluster, navigate to **Home** → **Projects** and create a project for RHACS Cloud Service. Use **stackrox** as the project **Name**.

5.2.2. Next steps

- In the ACS Console, create an init bundle. The init bundle contains secrets that allow communication between RHACS Cloud Service secured clusters and the ACS Console.

5.3. GENERATING AN INIT BUNDLE FOR SECURED CLUSTERS

Before you install the **SecuredCluster** resource on a cluster, you must create an init bundle. The cluster that has **SecuredCluster** installed and configured then uses this bundle to authenticate with Central. You can create an init bundle by using either the RHACS portal or the **roxctl** CLI. You then apply the init bundle by using it to create resources.

NOTE

You must have the **Admin** user role to create an init bundle.

5.3.1. Generating an init bundle

5.3.1.1. Generating an init bundle by using the RHACS portal

You can create an init bundle containing secrets by using the RHACS portal.

NOTE

You must have the **Admin** user role to create an init bundle.

Procedure

1. Find the address of the RHACS portal based on your exposure method:

   a. For a route:

   ```
   $ oc get route central -n stackrox
   ```

   b. For a load balancer:

   ```
   $ oc get service central-loadbalancer -n stackrox
   ```

   c. For port forward:

      i. Run the following command:

```
$ oc port-forward svc/central 18443:443 -n stackrox
```

ii. Navigate to **https://localhost:18443/**.

2. On the RHACS portal, navigate to **Platform Configuration → Integrations**.

3. Navigate to the **Authentication Tokens** section and click on **Cluster Init Bundle**.

4. Click **Generate bundle**.

5. Enter a name for the cluster init bundle and click **Generate**.

   a. If you are installing using Helm charts, click **Download Helm Values File** to download the generated bundle.

   b. If you are installing using the Operator, click **Download Kubernetes Secret File** to download the generated bundle.

IMPORTANT

Store this bundle securely because it contains secrets. You can use the same bundle to create multiple secured clusters.

Next steps

1. Apply the init bundle by creating a resource on the secured cluster.

2. Install secured cluster services on each cluster.

5.3.1.2. Generating an init bundle by using the roxctl CLI

You can create an init bundle with secrets by using the **roxctl** CLI.

NOTE

You must have the **Admin** user role to create init bundles.

Prerequisites

- You have configured the **ROX_API_TOKEN** and the **ROX_CENTRAL_ADDRESS** environment variables:

  a. Set the **ROX_API_TOKEN** by running the following command:

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

  b. Set the **ROX_CENTRAL_ADDRESS** environment variable by running the following command:

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

IMPORTANT

In RHACS Cloud Service, when using **roxctl** commands that require the Central address, use the **Central instance address** as displayed in the **Instance Details** section of the Red Hat Hybrid Cloud Console. For example, use **acs-ABCD12345.acs.rhcloud.com** instead of **acs-data-ABCD12345.acs.rhcloud.com**.

Procedure

- To generate a cluster init bundle containing secrets for Helm installations, run the following command:

  ```
  $ roxctl -e "$ROX_CENTRAL_ADDRESS" \
    central init-bundles generate <cluster_init_bundle_name> \
    --output cluster_init_bundle.yaml
  ```

- To generate a cluster init bundle containing secrets for Operator installations, run the following command:

  ```
  $ roxctl -e "$ROX_CENTRAL_ADDRESS" \
    central init-bundles generate <cluster_init_bundle_name> \
    --output-secrets cluster_init_bundle.yaml
  ```

  IMPORTANT

  Ensure that you store this bundle securely because it contains secrets. You can use the same bundle to set up multiple secured clusters.

5.3.2. Next steps

- Creating resources by using the init bundle

5.4. APPLYING AN INIT BUNDLE FOR SECURED CLUSTERS

Apply the init bundle by using it to create resources.

NOTE

You must have the **Admin** user role to apply an init bundle.

5.4.1. Creating resources by using the init bundle

Before you install secured clusters, you must use the init bundle to create the required resources on the cluster that will allow the services on the secured clusters to communicate with RHACS Cloud Service.

NOTE

If you are installing by using Helm charts, do not perform this step. Complete the installation by using Helm; See "Installing RHACS on secured clusters by using Helm charts" in the additional resources section.

Prerequisites

- You must have generated an init bundle containing secrets.

Procedure

To create resources, perform one of the following steps:

- In the OpenShift Container Platform web console, in the top menu, click **+** to open the **Import YAML** page. You can drag the init bundle file or copy and paste its contents into the editor, and then click **Create**.

- Using the Red Hat OpenShift CLI, run the following command to create the resources:

```
$ oc create -f <init_bundle>.yaml \ ❶
  -n <stackrox> ❷
```

❶ Specify the file name of the init bundle containing the secrets.

❷ Specify the name of the project where Central services are installed.

Verification

- Restart Sensor to pick up the new certificates.
  For more information about how to restart Sensor, see "Restarting the Sensor container" in the "Additional resources" section.

5.4.2. Next steps

- On each Red Hat OpenShift cluster, install the RHACS Operator.

- Install RHACS secured cluster services in all clusters that you want to monitor.

5.4.3. Additional resources

- Restarting the Sensor container

5.5. INSTALLING THE OPERATOR

Install the RHACS Operator on your secured clusters.

5.5.1. Installing the RHACS Operator for RHACS Cloud Service

Using the OperatorHub provided with OpenShift Container Platform is the easiest way to install the RHACS Operator.

Prerequisites

- You have access to an OpenShift Container Platform cluster using an account with Operator installation permissions.

- You must be using OpenShift Container Platform 4.11 or later. For more information, see Red Hat Advanced Cluster Security for Kubernetes Support Policy.

Procedure

1. Navigate in the web console to the **Operators → OperatorHub** page.

2. If Red Hat Advanced Cluster Security for Kubernetes is not displayed, enter **Advanced Cluster Security** into the **Filter by keyword** box to find the Red Hat Advanced Cluster Security for Kubernetes Operator.

3. Select the **Red Hat Advanced Cluster Security for Kubernetes Operator**to view the details page.

4. Read the information about the Operator, and then click **Install**.

5. On the **Install Operator** page:

    - Keep the default value for **Installation mode** as **All namespaces on the cluster**.

    - Select a specific namespace in which to install the Operator for the **Installed namespace** field. Install the Red Hat Advanced Cluster Security for Kubernetes Operator in the **rhacs-operator** namespace.

    - Select automatic or manual updates for **Update approval**.
      If you select automatic updates, when a new version of the Operator is available, Operator Lifecycle Manager (OLM) automatically upgrades the running instance of your Operator.

      If you select manual updates, when a newer version of the Operator is available, OLM creates an update request. As a cluster administrator, you must manually approve the update request to update the Operator to the latest version.

      Red Hat recommends enabling automatic upgrades for Operator in RHACS Cloud Service. See the Red Hat Advanced Cluster Security for Kubernetes Support Matrix for more information.

6. Click **Install**.

Verification

- After the installation completes, navigate to **Operators → Installed Operators** to verify that the Red Hat Advanced Cluster Security for Kubernetes Operator is listed with the status of **Succeeded**.

5.5.2. Next steps

- On each Red Hat OpenShift cluster, install secured cluster resources in the **stackrox** project.

5.6. INSTALLING SECURED CLUSTER RESOURCES FROM RHACS CLOUD SERVICE

You can install RHACS Cloud Service on your secured clusters by using the the Operator or Helm charts. You can also use the **roxctl** CLI to install it, but do not use this method unless you have a specific installation need that requires using it.

Prerequisites

- You have created your Red Hat OpenShift cluster and installed the Operator on it.

- In the ACS Console in RHACS Cloud Service, you have created and downloaded the init bundle.

- You applied the init bundle by using the **oc create** command.

- During installation, you noted the **Central API Endpoint**, including the address and the port number. You can view this information by choosing **Advanced Cluster Security → ACS Instances** from the cloud console navigation menu, and then clicking the ACS instance you created.

5.6.1. Installing RHACS on secured clusters by using the Operator

5.6.1.1. Installing secured cluster services

You can install secured cluster services on your clusters by using the **SecuredCluster** custom resource. You must install the secured cluster services on every cluster in your environment that you want to monitor.

CAUTION

When you install secured cluster services, Collector is also installed. To install Collector on systems that have Unified Extensible Firmware Interface (UEFI) and that have Secure Boot enabled, you must use eBPF probes because kernel modules are unsigned, and the UEFI firmware cannot load unsigned packages. Collector identifies Secure Boot status at the start and switches to eBPF probes if required.

Prerequisites

- If you are using OpenShift Container Platform, you must install version 4.11 or later.

- You have installed the RHACS Operator.

- You have generated an init bundle and applied it to the cluster.

Procedure

1. On the OpenShift Container Platform web console, navigate to the **Operators → Installed Operators** page.

2. Click the RHACS Operator.

3. Click **Secured Cluster** from the central navigation menu in the **Operator details** page.

4. Click **Create SecuredCluster**.

5. Select one of the following options in the **Configure via** field:

    - **Form view**: Use this option if you want to use the on-screen fields to configure the secured cluster and do not need to change any other fields.

    - **YAML view**: Use this view to set up the secured cluster using the YAML file. The YAML file is displayed in the window and you can edit fields in it. If you select this option, when you are finished editing the file, click **Create**.

6. If you are using **Form view**, enter the new project name by accepting or editing the default name. The default value is **stackrox-secured-cluster-services**.

7. Optional: Add any labels for the cluster.

8. Enter a unique name for your **SecuredCluster** custom resource.

9. For **Central Endpoint**, enter the address and port number of your Central instance. For example, if Central is available at **https://central.example.com**, then specify the central endpoint as **central.example.com:443**. The default value of **central.stackrox.svc:443** only works when you install secured cluster services and Central in the same cluster. Do not use the default value when you are configuring multiple clusters. Instead, use the hostname when configuring the **Central Endpoint** value for each cluster.

   - For RHACS Cloud Service use the **Central API Endpoint**, including the address and the port number. You can view this information by choosing **Advanced Cluster Security → ACS Instances** from the cloud console navigation menu, then clicking the ACS instance you created.

   - Only if you are installing secured cluster services and Central in the same cluster, use **central.stackrox.svc:443**.

10. Accept the default values or configure custom values if needed. For example, you may need to configure TLS if you are using custom certificates or untrusted CAs.

11. Click **Create**.

Next step

1. Optional: Configure additional secured cluster settings.

2. Verify installation.

5.6.2. Installing RHACS Cloud Service on secured clusters by using Helm charts

You can install RHACS on secured clusters by using Helm charts with no customization, using the default values, or with customizations of configuration parameters.

First, ensure that you add the Helm chart repository.

5.6.2.1. Adding the Helm chart repository

Procedure

- Add the RHACS charts repository.

```
$ helm repo add rhacs https://mirror.openshift.com/pub/rhacs/charts/
```

The Helm repository for Red Hat Advanced Cluster Security for Kubernetes includes Helm charts for installing different components, including:

- Secured Cluster Services Helm chart (**secured-cluster-services**) for installing the per-cluster and per-node components (Sensor, Admission Controller, Collector, and Scanner-slim).

  NOTE

  Deploy the per-cluster components into each cluster that you want to monitor and deploy the per-node components in all nodes that you want to monitor.

Verification

- Run the following command to verify the added chart repository:

```
$ helm search repo -l rhacs/
```

5.6.2.2. Installing RHACS Cloud Service on secured clusters by using Helm charts without customizations

5.6.2.2.1. Installing the secured-cluster-services Helm chart without customization

Use the following instructions to install the **secured-cluster-services** Helm chart to deploy the per-cluster and per-node components (Sensor, Admission controller, Collector, and Scanner-slim).

CAUTION

To install Collector on systems that have Unified Extensible Firmware Interface (UEFI) and that have Secure Boot enabled, you must use eBPF probes because kernel modules are unsigned, and the UEFI firmware cannot load unsigned packages. Collector identifies Secure Boot status at the start and switches to eBPF probes if required.

Prerequisites

- You must have generated an RHACS init bundle for your cluster.
- You must have access to the Red Hat Container Registry and a pull secret for authentication. For information about downloading images from **registry.redhat.io**, see Red Hat Container Registry Authentication.
- You must have the **Central API Endpoint**, including the address and the port number. You can view this information by choosing **Advanced Cluster Security → ACS Instances** from the cloud console navigation menu, then clicking the ACS instance you created.

Procedure

- Run the following command on your Kubernetes based clusters:

```
$ helm install -n stackrox --create-namespace \
   stackrox-secured-cluster-services rhacs/secured-cluster-services \
   -f <path_to_cluster_init_bundle.yaml> \ 1
   -f <path_to_pull_secret.yaml> \ 2
   --set clusterName=<name_of_the_secured_cluster> \
   --set centralEndpoint=<endpoint_of_central_service> 3
   --set imagePullSecrets.username=<your redhat.com username> \4
   --set imagePullSecrets.password=<your redhat.com password>5
```

1. Use the **-f** option to specify the path for the init bundle.

2. Use the -f option to specify the path for the pull secret for Red Hat Container Registry authentication.

3. Enter the Central API Endpoint, including the address and the port number. You can view this information again in the Red Hat Hybrid Cloud Console console by choosing Advanced Cluster Security → ACS Instances, and then clicking the ACS instance you created.

4 Include the user name for your pull secret for Red Hat Container Registry authentication.

5 Include the password for your pull secret for Red Hat Container Registry authentication.

Additional resources

- Generating an init bundle for secured clusters
- Applying an init bundle for secured clusters

5.6.2.3. Configuring the secured-cluster-services Helm chart with customizations

You can use Helm chart configuration parameters with the **helm install** and **helm upgrade** commands. Specify these parameters by using the **--set** option or by creating YAML configuration files.

Create the following files for configuring the Helm chart for installing Red Hat Advanced Cluster Security for Kubernetes:

- Public configuration file **values-public.yaml**: Use this file to save all non-sensitive configuration options.
- Private configuration file **values-private.yaml**: Use this file to save all sensitive configuration options. Ensure that you store this file securely.

IMPORTANT

When using the **secured-cluster-services** Helm chart, do not change the **values.yaml** file that is part of the chart.

5.6.2.3.1. Configuration parameters

Parameter	Description
clusterName	Name of your cluster.
centralEndpoint	Address, including port number, of the Central endpoint. If you are using a non-gRPC capable load balancer, use the WebSocket protocol by prefixing the endpoint address with **wss:**//. When configuring multiple clusters, use the hostname for the address (for example, **central.example.com:443**).
sensor.endpoint	Address of the Sensor endpoint including port number.
sensor.imagePullPolicy	Image pull policy for the Sensor container.
sensor.serviceTLS.cert	The internal service-to-service TLS certificate that Sensor uses.

Parameter	Description
sensor.serviceTLS.key	The internal service-to-service TLS certificate key that Sensor uses.
sensor.resources.requests.memory	The memory request for the Sensor container. Use this parameter to override the default value.
sensor.resources.requests.cpu	The CPU request for the Sensor container. Use this parameter to override the default value.
sensor.resources.limits.memory	The memory limit for the Sensor container. Use this parameter to override the default value.
sensor.resources.limits.cpu	The CPU limit for the Sensor container. Use this parameter to override the default value.
sensor.nodeSelector	Specify a node selector label as **label-key: label-value** to force Sensor to only schedule on nodes with the specified label.
sensor.tolerations	If the node selector selects tainted nodes, use this parameter to specify a taint toleration key, value, and effect for Sensor. This parameter is mainly used for infrastructure nodes.
image.main.name	The name of the **main** image.
image.collector.name	The name of the Collector image.
image.main.registry	Address of the registry you are using for the main image.
image.collector.registry	Address of the registry you are using for the Collector image.
image.main.pullPolicy	Image pull policy for **main** images.
image.collector.pullPolicy	Image pull policy for the Collector images.
image.main.tag	Tag of **main** image to use.
image.collector.tag	Tag of **collector** image to use.
collector.collectionMethod	Either **EBPF**, **CORE_BPF**, or **NO_COLLECTION**.
collector.imagePullPolicy	Image pull policy for the Collector container.

Parameter	Description
collector.complianceImagePullPolicy	Image pull policy for the Compliance container.
collector.disableTaintTolerations	If you specify **false**, tolerations are applied to Collector, and the collector pods can schedule onto all nodes with taints. If you specify it as **true**, no tolerations are applied, and the collector pods are not scheduled onto nodes with taints.
collector.resources.requests.memory	The memory request for the Collector container. Use this parameter to override the default value.
collector.resources.requests.cpu	The CPU request for the Collector container. Use this parameter to override the default value.
collector.resources.limits.memory	The memory limit for the Collector container. Use this parameter to override the default value.
collector.resources.limits.cpu	The CPU limit for the Collector container. Use this parameter to override the default value.
collector.complianceResources.requests.memory	The memory request for the Compliance container. Use this parameter to override the default value.
collector.complianceResources.requests.cpu	The CPU request for the Compliance container. Use this parameter to override the default value.
collector.complianceResources.limits.memory	The memory limit for the Compliance container. Use this parameter to override the default value.
collector.complianceResources.limits.cpu	The CPU limit for the Compliance container. Use this parameter to override the default value.
collector.serviceTLS.cert	The internal service-to-service TLS certificate that Collector uses.
collector.serviceTLS.key	The internal service-to-service TLS certificate key that Collector uses.
admissionControl.listenOnCreates	This setting controls whether Kubernetes is configured to contact Red Hat Advanced Cluster Security for Kubernetes with **AdmissionReview** requests for workload creation events.

Parameter	Description
admissionControl.listenOnUpdates	When you set this parameter as **false**, Red Hat Advanced Cluster Security for Kubernetes creates the **ValidatingWebhookConfiguration** in a way that causes the Kubernetes API server not to send object update events. Since the volume of object updates is usually higher than the object creates, leaving this as **false** limits the load on the admission control service and decreases the chances of a malfunctioning admission control service.
admissionControl.listenOnEvents	This setting controls whether the cluster is configured to contact Red Hat Advanced Cluster Security for Kubernetes with **AdmissionReview** requests for Kubernetes **exec** and **portforward** events. Red Hat Advanced Cluster Security for Kubernetes does not support this feature on OpenShift Container Platform 3.11. For more information, see Red Hat Advanced Cluster Security for Kubernetes Support Policy.
admissionControl.dynamic.enforceOnCreates	This setting controls whether Red Hat Advanced Cluster Security for Kubernetes evaluates policies; if it is disabled, all AdmissionReview requests are automatically accepted.
admissionControl.dynamic.enforceOnUpdates	This setting controls the behavior of the admission control service. You must specify **listenOnUpdates** as **true** for this to work.
admissionControl.dynamic.scanInline	If you set this option to **true**, the admission control service requests an image scan before making an admission decision. Since image scans take several seconds, enable this option only if you can ensure that all images used in your cluster are scanned before deployment (for example, by a CI integration during image build). This option corresponds to the **Contact image scanners** option in the RHACS Portal.
admissionControl.dynamic.disableBypass	Set it to **true** to disable bypassing the Admission controller.
admissionControl.dynamic.timeout	The maximum time, in seconds, Red Hat Advanced Cluster Security for Kubernetes should wait while evaluating admission review requests. Use this to set request timeouts when you enable image scanning. If the image scan runs longer than the specified time, Red Hat Advanced Cluster Security for Kubernetes accepts the request.

Parameter	Description
admissionControl.resources.requests.memory	The memory request for the Admission Control container. Use this parameter to override the default value.
admissionControl.resources.requests.cpu	The CPU request for the Admission Control container. Use this parameter to override the default value.
admissionControl.resources.limits.memory	The memory limit for the Admission Control container. Use this parameter to override the default value.
admissionControl.resources.limits.cpu	The CPU limit for the Admission Control container. Use this parameter to override the default value.
admissionControl.nodeSelector	Specify a node selector label as **label-key: label-value** to force Admission Control to only schedule on nodes with the specified label.
admissionControl.tolerations	If the node selector selects tainted nodes, use this parameter to specify a taint toleration key, value, and effect for Admission Control. This parameter is mainly used for infrastructure nodes.
admissionControl.serviceTLS.cert	The internal service-to-service TLS certificate that Admission Control uses.
admissionControl.serviceTLS.key	The internal service-to-service TLS certificate key that Admission Control uses.
registryOverride	Use this parameter to override the default **docker.io** registry. Specify the name of your registry if you are using some other registry.
collector.disableTaintTolerations	If you specify **false**, tolerations are applied to Collector, and the Collector pods can schedule onto all nodes with taints. If you specify it as **true**, no tolerations are applied, and the Collector pods are not scheduled onto nodes with taints.
createUpgraderServiceAccount	Specify **true** to create the **sensor-upgrader** account. 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 do not create this account, you must complete future upgrades manually if the Sensor does not have enough permissions.

Parameter	Description
createSecrets	Specify **false** to skip the orchestrator secret creation for the Sensor, Collector, and Admission controller.
collector.slimMode	Specify **true** if you want to use a slim Collector image for deploying Collector. Using slim Collector images requires Central to provide the matching eBPF probe or kernel module. If you are running Red Hat Advanced Cluster Security for Kubernetes in offline mode, you must download a kernel support package from stackrox.io and upload it to Central for slim Collectors to function. Otherwise, you must ensure that Central can access the online probe repository hosted at https://collector-modules.stackrox.io/.
sensor.resources	Resource specification for Sensor.
admissionControl.resources	Resource specification for Admission controller.
collector.resources	Resource specification for Collector.
collector.complianceResources	Resource specification for Collector’s Compliance container.
exposeMonitoring	If you set this option to **true**, Red Hat Advanced Cluster Security for Kubernetes exposes Prometheus metrics endpoints on port number 9090 for the Sensor, Collector, and the Admission controller.
auditLogs.disableCollection	If you set this option to **true**, Red Hat Advanced Cluster Security for Kubernetes disables the audit log detection features used to detect access and modifications to configuration maps and secrets.
scanner.disable	If you set this option to **false**, Red Hat Advanced Cluster Security for Kubernetes deploys a Scanner-slim and Scanner DB in the secured cluster to allow scanning images on OpenShift Container Registry. Enabling Scanner-slim is supported on OpenShift Container Platform and Kubernetes secured clusters. Defaults to **true**.
scanner.dbTolerations	If the node selector selects tainted nodes, use this parameter to specify a taint toleration key, value, and effect for Scanner DB.

Parameter	Description
scanner.replicas	Resource specification for Collector’s Compliance container.
scanner.logLevel	Setting this parameter allows you to modify the scanner log level. Use this option only for troubleshooting purposes.
scanner.autoscaling.disable	If you set this option to **true**, Red Hat Advanced Cluster Security for Kubernetes disables autoscaling on the Scanner deployment.
scanner.autoscaling.minReplicas	The minimum number of replicas for autoscaling. Defaults to 2.
scanner.autoscaling.maxReplicas	The maximum number of replicas for autoscaling. Defaults to 5.
scanner.nodeSelector	Specify a node selector label as **label-key: label-value** to force Scanner to only schedule on nodes with the specified label.
scanner.tolerations	If the node selector selects tainted nodes, use this parameter to specify a taint toleration key, value, and effect for Scanner.
scanner.dbNodeSelector	Specify a node selector label as **label-key: label-value** to force Scanner DB to only schedule on nodes with the specified label.
scanner.dbTolerations	If the node selector selects tainted nodes, use this parameter to specify a taint toleration key, value, and effect for Scanner DB.
scanner.resources.requests.memory	The memory request for the Scanner container. Use this parameter to override the default value.
scanner.resources.requests.cpu	The CPU request for the Scanner container. Use this parameter to override the default value.
scanner.resources.limits.memory	The memory limit for the Scanner container. Use this parameter to override the default value.
scanner.resources.limits.cpu	The CPU limit for the Scanner container. Use this parameter to override the default value.
scanner.dbResources.requests.memory	The memory request for the Scanner DB container. Use this parameter to override the default value.

Parameter	Description
scanner.dbResources.requests.cpu	The CPU request for the Scanner DB container. Use this parameter to override the default value.
scanner.dbResources.limits.memory	The memory limit for the Scanner DB container. Use this parameter to override the default value.
scanner.dbResources.limits.cpu	The CPU limit for the Scanner DB container. Use this parameter to override the default value.
monitoring.openshift.enabled	If you set this option to **false**, Red Hat Advanced Cluster Security for Kubernetes will not set up Red Hat OpenShift monitoring. Defaults to **true** on Red Hat OpenShift 4.

5.6.2.3.1.1. Environment variables

You can specify environment variables for Sensor and Admission controller in the following format:

```
customize:
  envVars:
    ENV_VAR1: "value1"
    ENV_VAR2: "value2"
```

The **customize** setting allows you to specify custom Kubernetes metadata (labels and annotations) for all objects created by this Helm chart and additional pod labels, pod annotations, and container environment variables for workloads.

The configuration is hierarchical, in the sense that metadata defined at a more generic scope (for example, for all objects) can be overridden by metadata defined at a narrower scope (for example, only for the Sensor deployment).

5.6.2.3.2. Installing the secured-cluster-services Helm chart

After you configure the **values-public.yaml** and **values-private.yaml** files, install the **secured-cluster-services** Helm chart to deploy the per-cluster and per-node components (Sensor, Admission controller, Collector, and Scanner-slim).

CAUTION

To install Collector on systems that have Unified Extensible Firmware Interface (UEFI) and that have Secure Boot enabled, you must use eBPF probes because kernel modules are unsigned, and the UEFI firmware cannot load unsigned packages. Collector identifies Secure Boot status at the start and switches to eBPF probes if required.

Prerequisites

- You must have generated an RHACS init bundle for your cluster.

- You must have access to the Red Hat Container Registry and a pull secret for authentication. For information about downloading images from **registry.redhat.io**, see Red Hat Container Registry Authentication.

- You must have the **Central API Endpoint**, including the address and the port number. You can view this information by choosing **Advanced Cluster Security → ACS Instances** from the cloud console navigation menu, then clicking the ACS instance you created.

Procedure

- Run the following command:

```
$ helm install -n stackrox \
  --create-namespace stackrox-secured-cluster-services rhacs/secured-cluster-services \
  -f <name_of_cluster_init_bundle.yaml> \
  -f <path_to_values_public.yaml> -f <path_to_values_private.yaml> \ 1
  --set imagePullSecrets.username=<username> \ 2
  --set imagePullSecrets.password=<password> 3
```

1 Use the **-f** option to specify the paths for your YAML configuration files.

2 Include the user name for your pull secret for Red Hat Container Registry authentication.

3 Include the password for your pull secret for Red Hat Container Registry authentication.

> NOTE
>
> To deploy **secured-cluster-services** Helm chart by using a continuous integration (CI) system, pass the init bundle YAML file as an environment variable to the **helm install** command:
>
> ```
> $ helm install ... -f <(echo "$INIT_BUNDLE_YAML_SECRET") 1
> ```
>
> 1 If you are using base64 encoded variables, use the **helm install ... -f <(echo "$INIT_BUNDLE_YAML_SECRET" | base64 --decode)** command instead.

Additional resources

- Generating an init bundle for secured clusters

- Applying an init bundle for secured clusters

5.6.2.4. Changing configuration options after deploying the secured-cluster-services Helm chart

You can make changes to any configuration options after you have deployed the **secured-cluster-services** Helm chart.

Procedure

1. Update the **values-public.yaml** and **values-private.yaml** configuration files with new values.

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

```
$ helm upgrade -n stackrox \
  stackrox-secured-cluster-services rhacs/secured-cluster-services \
  --reuse-values \ 1
  -f <path_to_values_public.yaml> \
  -f <path_to_values_private.yaml>
```

1 You must specify the **--reuse-values** parameter, otherwise the Helm upgrade command resets all previously configured settings.

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.

5.6.3. Installing RHACS on secured clusters by using the roxctl CLI

To install RHACS on secured clusters by using the CLI, perform the following steps:

1. Install the **roxctl** CLI.
2. Install Sensor.

5.6.3.1. Installing the roxctl CLI

You must first download the binary. You can install **roxctl** on Linux, Windows, or macOS.

5.6.3.1.1. Installing the roxctl CLI on Linux

You can install the **roxctl** CLI binary on Linux by using the following procedure.

NOTE

roxctl CLI for Linux is available for **amd64**, **ppc64le**, and **s390x** architectures.

Procedure

1. Determine the **roxctl** architecture for the target operating system:

```
$ arch="$(uname -m | sed "s/x86_64//")"; arch="${arch:+-$arch}"
```

2. Download the **roxctl** CLI:

```
$ curl -f -o roxctl "https://mirror.openshift.com/pub/rhacs/assets/4.3.8/bin/Linux/roxctl${arch}"
```

3. Make the **roxctl** binary executable:

```
$ chmod +x roxctl
```

4. Place the **roxctl** binary in a directory that is on your **PATH**:
To check your **PATH**, execute the following command:

```
$ echo $PATH
```

Verification

- Verify the **roxctl** version you have installed:

```
$ roxctl version
```

5.6.3.1.2. Installing the roxctl CLI on macOS

You can install the **roxctl** CLI binary on macOS by using the following procedure.

> NOTE
>
> **roxctl** CLI for macOS is available for the **amd64** architecture.

Procedure

1. Download the **roxctl** CLI:

```
$ curl -f -O https://mirror.openshift.com/pub/rhacs/assets/4.3.8/bin/Darwin/roxctl
```

2. Remove all extended attributes from the binary:

```
$ xattr -c roxctl
```

3. Make the **roxctl** binary executable:

```
$ chmod +x roxctl
```

4. Place the **roxctl** binary in a directory that is on your **PATH**:
   To check your **PATH**, execute the following command:

```
$ echo $PATH
```

Verification

- Verify the **roxctl** version you have installed:

```
$ roxctl version
```

5.6.3.1.3. Installing the roxctl CLI on Windows

You can install the **roxctl** CLI binary on Windows by using the following procedure.

> NOTE
>
> **roxctl** CLI for Windows is available for the **amd64** architecture.

Procedure

- Download the **roxctl** CLI:

```
$ curl -f -O https://mirror.openshift.com/pub/rhacs/assets/4.3.8/bin/Windows/roxctl.exe
```

Verification

- Verify the **roxctl** version you have installed:

```
$ roxctl version
```

5.6.3.2. Installing Sensor

To monitor a cluster, you must deploy Sensor. You must deploy Sensor into each cluster that you want to monitor. The following steps describe adding Sensor by using the RHACS portal.

Prerequisites

- You must have already installed Central services, or you can access Central services by selecting your **ACS instance** on Red Hat Advanced Cluster Security Cloud Service (RHACS Cloud Service).

Procedure

1. On your secured cluster, in the RHACS portal, navigate to **Platform Configuration → Clusters**.
2. Select **+ New Cluster**.
3. Specify a name for the cluster.
4. Provide appropriate values for the fields based on where you are deploying the Sensor.
   - Enter the Central API Endpoint, including the address and the port number. You can view this information again in the Red Hat Hybrid Cloud Console by choosing **Advanced Cluster Security → ACS Instances**, and then clicking the ACS instance you created.
5. Click **Next** to continue with the Sensor setup.
6. Click **Download YAML File and Keys**to download the cluster bundle (zip archive).

   IMPORTANT

   The cluster bundle zip archive includes unique configurations and keys for each cluster. Do not reuse the same files in another cluster.

7. From a system that has access to the monitored cluster, unzip and run the **sensor** script from the cluster bundle:

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

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

If you get a warning that you do not have the required permissions to deploy Sensor, follow the on-screen instructions, or contact your cluster administrator for assistance.

After Sensor is deployed, it contacts Central and provides cluster information.

Verification

1. Return to the RHACS portal and check if the deployment is successful. If successful, when viewing your list of clusters in **Platform Configuration → Clusters**, the cluster status displays a green checkmark and a **Healthy** status. If you do not see a green checkmark, use the following command to check for problems:
   - On OpenShift Container Platform, enter the following command:

     ```
     $ oc get pod -n stackrox -w
     ```

   - On Kubernetes, enter the following command:

     ```
     $ kubectl get pod -n stackrox -w
     ```

2. Click **Finish** to close the window.

After installation, Sensor starts reporting security information to RHACS and the RHACS portal dashboard begins showing deployments, images, and policy violations from the cluster on which you have installed the Sensor.

5.6.4. Next steps

- Verify installation by ensuring that your secured clusters can communicate with the ACS instance.

5.7. CONFIGURING THE PROXY FOR SECURED CLUSTER SERVICES IN RHACS CLOUD SERVICE

You must configure the proxy settings for secured cluster services within the Red Hat Advanced Cluster Security Cloud Service (RHACS Cloud Service) environment to establish a connection between the Secured Cluster and the specified proxy server. This ensures reliable data collection and transmission.

5.7.1. Specifying the environment variables in the SecuredCluster CR

To configure an egress proxy, you can either use the cluster-wide Red Hat OpenShift proxy or specify the **HTTP_PROXY**, **HTTPS_PROXY**, and **NO_PROXY** environment variables within the SecuredCluster Custom Resource (CR) configuration file to ensure proper use of the proxy and bypass for internal requests within the specified domain.

The proxy configuration applies to all running services: Sensor, Collector, Admission Controller and Scanner.

Procedure

- Specify the **HTTP_PROXY**, **HTTPS_PROXY**, and **NO_PROXY** environment variables under the customize specification in the SecuredCluster CR configuration file:
  For example:

  ```
  # proxy collector
  customize:
  ```

```
envVars:
  - name: HTTP_PROXY
    value: http://egress-proxy.stackrox.svc:xxxx ①
  - name: HTTPS_PROXY
    value: http://egress-proxy.stackrox.svc:xxxx ②
  - name: NO_PROXY
    value: .stackrox.svc ③
```

① The variable **HTTP_PROXY** is set to the value **http://egress-proxy.stackrox.svc:xxxx**. This is the proxy server used for HTTP connections.

② The variable **HTTPS_PROXY** is set to the value **http://egress-proxy.stackrox.svc:xxxx**. This is the proxy server used for HTTPS connections.

③ The variable **NO _PROXY** is set to **.stackrox.svc**. This variable is used to define the hostname or IP address that should not be accessed through the proxy server.

5.8. VERIFYING INSTALLATION OF SECURED CLUSTERS

After installing RHACS Cloud Service, you can perform some steps to verify that the installation was successful.

To verify installation, access your ACS Console from the Red Hat Hybrid Cloud Console. The Dashboard displays the number of clusters that RHACS Cloud Service is monitoring, along with information about nodes, deployments, images, and violations.

If no data appears in the ACS Console:

- Ensure that at least one secured cluster is connected to your RHACS Cloud Service instance. For more information, see Installing secured cluster resources from RHACS Cloud Service .

- Examine your Sensor pod logs to ensure that the connection to your RHACS Cloud Service instance is successful.

- In the Red Hat OpenShift cluster, navigate to **Platform Configuration → Clusters** to verify that the components are healthy and view additional operational information.

- Examine the values in the **SecuredCluster** API in the Operator on your local cluster to ensure that the **Central API Endpoint** has been entered correctly. This value should be the same value as shown in the **ACS instance** details in the Red Hat Hybrid Cloud Console.

CHAPTER 6. SETTING UP RHACS CLOUD SERVICE WITH KUBERNETES SECURED CLUSTERS

6.1. CREATING AN RHACS CLOUD SERVICE INSTANCE FOR KUBERNETES CLUSTERS

Access Red Hat Advanced Cluster Security Cloud Service (RHACS Cloud Service) by selecting an instance in the Red Hat Hybrid Cloud Console. An **ACS instance** contains the RHACS Cloud Service management interface and services that Red Hat configures and manages for you. The management interface connects to your secured clusters, which contain the services that scan and collect information about vulnerabilities. One instance can connect to and monitor many clusters.

6.1.1. Creating an instance in the console

In the Red Hat Hybrid Cloud Console, create an **ACS instance** to connect to your secured clusters.

Procedure

To create an **ACS instance**:

1. Log in to the Red Hat Hybrid Cloud Console.

2. From the navigation menu, select **Advanced Cluster Security → ACS Instances**.

3. Select **Create ACS instance** and enter information into the displayed fields or select the appropriate option from the drop-down list:

    - **Name**: Enter the name of your **ACS instance**. An **ACS instance** contains the RHACS Central component, also referred to as "Central", which includes the RHACS Cloud Service management interface and services that are configured and managed by Red Hat. You manage your secured clusters that communicate with Central. You can connect many secured clusters to one instance.

    - **Cloud provider**: The cloud provider where Central is located. Select **AWS**.

    - **Cloud region**: The region for your cloud provider where Central is located. Select one of the following regions:

        - US-East, N. Virginia

        - Europe, Ireland

    - **Availability zones**: Use the default value (**Multi**).

4. Click **Create instance**.

6.1.2. Next steps

- On each Kubernetes cluster you want to secure, install secured cluster resources by using Helm charts or the **roxctl** CLI.

6.2. GENERATING AN INIT BUNDLE FOR KUBERNETES SECURED CLUSTERS

Before you install the **SecuredCluster** resource on a cluster, you must create an init bundle. The cluster that has **SecuredCluster** installed and configured then uses this bundle to authenticate with the ACS Console. You can create an init bundle by using either the RHACS portal or the **roxctl** CLI. You then apply the init bundle by using it to create resources.

6.2.1. Generating an init bundle by using the RHACS portal

You can create an init bundle containing secrets by using the RHACS portal, also called the ACS Console.

NOTE

You must have the **Admin** user role to create an init bundle.

Procedure

1. On the RHACS portal, navigate to **Platform Configuration → Integrations**.
2. Navigate to the **Authentication Tokens** section and click on **Cluster Init Bundle**.
3. Click **Generate bundle**.
4. Enter a name for the cluster init bundle and click **Generate**.
    a. If you are installing using Helm charts, click **Download Helm Values File** to download the generated bundle.
    b. If you are installing using the Operator, click **Download Kubernetes Secret File** to download the generated bundle.

IMPORTANT

Store this bundle securely because it contains secrets. You can use the same bundle to create multiple secured clusters.

Next steps

1. Apply the init bundle by creating a resource on the secured cluster.
2. Install secured cluster services on each cluster.

6.2.2. Generating an init bundle by using the roxctl CLI

You can create an init bundle with secrets by using the **roxctl** CLI.

NOTE

You must have the **Admin** user role to create init bundles.

Prerequisites

- You have configured the **ROX_API_TOKEN** and the **ROX_CENTRAL_ADDRESS** environment variables:

a. Set the **ROX_API_TOKEN** by running the following command:

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

b. Set the **ROX_CENTRAL_ADDRESS** environment variable by running the following command:

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

IMPORTANT

In RHACS Cloud Service, when using **roxctl** commands that require the Central address, use the **Central instance address** as displayed in the **Instance Details** section of the Red Hat Hybrid Cloud Console. For example, use **acs-ABCD12345.acs.rhcloud.com** instead of **acs-data-ABCD12345.acs.rhcloud.com**.

Procedure

- To generate a cluster init bundle containing secrets for Helm installations, run the following command:

```
$ roxctl -e "$ROX_CENTRAL_ADDRESS" \
  central init-bundles generate <cluster_init_bundle_name> \
  --output cluster_init_bundle.yaml
```

- To generate a cluster init bundle containing secrets for Operator installations, run the following command:

```
$ roxctl -e "$ROX_CENTRAL_ADDRESS" \
  central init-bundles generate <cluster_init_bundle_name> \
  --output-secrets cluster_init_bundle.yaml
```

IMPORTANT

Ensure that you store this bundle securely because it contains secrets. You can use the same bundle to set up multiple secured clusters.

6.2.3. Next steps

- Creating resources by using the init bundle

6.3. APPLYING AN INIT BUNDLE FOR KUBERNETES SECURED CLUSTERS

Apply the init bundle by using it to create resources.

6.3.1. Creating resources by using the init bundle

Before you install secured clusters, you must use the init bundle to create the required resources on the cluster that will allow the services on the secured clusters to communicate with RHACS Cloud Service.

NOTE

If you are installing by using Helm charts, do not perform this step. Complete the installation by using Helm; See "Installing RHACS on secured clusters by using Helm charts" in the additional resources section.

Prerequisites

- You must have generated an init bundle containing secrets.

Procedure

- Using the **kubectl** CLI, run the following commands to create the resources:

```
$ kubectl create namespace stackrox 1
$ kubectl create -f <init_bundle>.yaml \ 2
  -n <stackrox> 3
```

1. Create the project where secured cluster resources will be installed. This example uses **stackrox**.
2. Specify the file name of the init bundle containing the secrets.
3. Specify the project name that you created. This example uses **stackrox**.

Verification

- Restart Sensor to pick up the new certificates.
  For more information about how to restart Sensor, see "Restarting the Sensor container" in the "Additional resources" section.

6.3.2. Next steps

- Install RHACS secured cluster services in all clusters that you want to monitor.

6.3.3. Additional resources

- Restarting the Sensor container

6.4. INSTALLING SECURED CLUSTER SERVICES FROM RHACS CLOUD SERVICE ON KUBERNETES CLUSTERS

You can install RHACS Cloud Service on your secured clusters by using one of the following methods:

- By using Helm charts
- By using the **roxctl** CLI (do not use this method unless you have a specific installation need that requires using it)

6.4.1. Installing RHACS Cloud Service on secured clusters by using Helm charts

You can install RHACS on secured clusters by using Helm charts with no customization, by using Helm charts with the default values, or by using Helm charts with customizations of configuration parameters.

First, ensure that you add the Helm chart repository.

6.4.1.1. Adding the Helm chart repository

Procedure

- Add the RHACS charts repository.

```
$ helm repo add rhacs https://mirror.openshift.com/pub/rhacs/charts/
```

The Helm repository for Red Hat Advanced Cluster Security for Kubernetes includes Helm charts for installing different components, including:

- Secured Cluster Services Helm chart (**secured-cluster-services**) for installing the per-cluster and per-node components (Sensor, Admission Controller, Collector, and Scanner-slim).

NOTE

Deploy the per-cluster components into each cluster that you want to monitor and deploy the per-node components in all nodes that you want to monitor.

Verification

- Run the following command to verify the added chart repository:

```
$ helm search repo -l rhacs/
```

6.4.1.2. Installing RHACS Cloud Service on secured clusters by using Helm charts without customizations

6.4.1.2.1. Installing the secured-cluster-services Helm chart without customization

Use the following instructions to install the **secured-cluster-services** Helm chart to deploy the per-cluster and per-node components (Sensor, Admission controller, Collector, and Scanner-slim).

CAUTION

To install Collector on systems that have Unified Extensible Firmware Interface (UEFI) and that have Secure Boot enabled, you must use eBPF probes because kernel modules are unsigned, and the UEFI firmware cannot load unsigned packages. Collector identifies Secure Boot status at the start and switches to eBPF probes if required.

Prerequisites

- You must have generated an RHACS init bundle for your cluster.

- You must have access to the Red Hat Container Registry and a pull secret for authentication. For information about downloading images from **registry.redhat.io**, see Red Hat Container Registry Authentication.

- You must have the **Central API Endpoint**, including the address and the port number. You can view this information by choosing **Advanced Cluster Security → ACS Instances** from the cloud console navigation menu, then clicking the ACS instance you created.

Procedure

- Run the following command on your Kubernetes based clusters:

```
$ helm install -n stackrox --create-namespace \
  stackrox-secured-cluster-services rhacs/secured-cluster-services \
  -f <path_to_cluster_init_bundle.yaml> \ 1
  -f <path_to_pull_secret.yaml> \ 2
  --set clusterName=<name_of_the_secured_cluster> \
  --set centralEndpoint=<endpoint_of_central_service> 3
  --set imagePullSecrets.username=<your redhat.com username> \ 4
  --set imagePullSecrets.password=<your redhat.com password> 5
```

1 Use the **-f** option to specify the path for the init bundle.

2 Use the -f option to specify the path for the pull secret for Red Hat Container Registry authentication.

3 Enter the Central API Endpoint, including the address and the port number. You can view this information again in the Red Hat Hybrid Cloud Console console by choosing Advanced Cluster Security → ACS Instances, and then clicking the ACS instance you created.

4 Include the user name for your pull secret for Red Hat Container Registry authentication.

5 Include the password for your pull secret for Red Hat Container Registry authentication.

6.4.1.3. Configuring the secured-cluster-services Helm chart with customizations

This section describes Helm chart configuration parameters that you can use with the **helm install** and **helm upgrade** commands. You can specify these parameters by using the **--set** option or by creating YAML configuration files.

Create the following files for configuring the Helm chart for installing Red Hat Advanced Cluster Security for Kubernetes:

- Public configuration file **values-public.yaml**: Use this file to save all non-sensitive configuration options.

- Private configuration file **values-private.yaml**: Use this file to save all sensitive configuration options. Ensure that you store this file securely.

IMPORTANT

While using the **secured-cluster-services** Helm chart, do not modify the **values.yaml** file that is part of the chart.

6.4.1.3.1. Configuration parameters

Parameter	Description
clusterName	Name of your cluster.
centralEndpoint	Address, including port number, of the Central endpoint. If you are using a non-gRPC capable load balancer, use the WebSocket protocol by prefixing the endpoint address with **wss:**//. When configuring multiple clusters, use the hostname for the address (for example, **central.example.com:443**).
sensor.endpoint	Address of the Sensor endpoint including port number.
sensor.imagePullPolicy	Image pull policy for the Sensor container.
sensor.serviceTLS.cert	The internal service-to-service TLS certificate that Sensor uses.
sensor.serviceTLS.key	The internal service-to-service TLS certificate key that Sensor uses.
sensor.resources.requests.memory	The memory request for the Sensor container. Use this parameter to override the default value.
sensor.resources.requests.cpu	The CPU request for the Sensor container. Use this parameter to override the default value.
sensor.resources.limits.memory	The memory limit for the Sensor container. Use this parameter to override the default value.
sensor.resources.limits.cpu	The CPU limit for the Sensor container. Use this parameter to override the default value.
sensor.nodeSelector	Specify a node selector label as **label-key: label-value** to force Sensor to only schedule on nodes with the specified label.
sensor.tolerations	If the node selector selects tainted nodes, use this parameter to specify a taint toleration key, value, and effect for Sensor. This parameter is mainly used for infrastructure nodes.
image.main.name	The name of the **main** image.
image.collector.name	The name of the Collector image.

Parameter	Description
image.main.registry	Address of the registry you are using for the main image.
image.collector.registry	Address of the registry you are using for the Collector image.
image.main.pullPolicy	Image pull policy for **main** images.
image.collector.pullPolicy	Image pull policy for the Collector images.
image.main.tag	Tag of **main** image to use.
image.collector.tag	Tag of **collector** image to use.
collector.collectionMethod	Either **EBPF**, **CORE_BPF**, or **NO_COLLECTION**.
collector.imagePullPolicy	Image pull policy for the Collector container.
collector.complianceImagePullPolicy	Image pull policy for the Compliance container.
collector.disableTaintTolerations	If you specify **false**, tolerations are applied to Collector, and the collector pods can schedule onto all nodes with taints. If you specify it as **true**, no tolerations are applied, and the collector pods are not scheduled onto nodes with taints.
collector.resources.requests.memory	The memory request for the Collector container. Use this parameter to override the default value.
collector.resources.requests.cpu	The CPU request for the Collector container. Use this parameter to override the default value.
collector.resources.limits.memory	The memory limit for the Collector container. Use this parameter to override the default value.
collector.resources.limits.cpu	The CPU limit for the Collector container. Use this parameter to override the default value.
collector.complianceResources.requests.memory	The memory request for the Compliance container. Use this parameter to override the default value.
collector.complianceResources.requests.cpu	The CPU request for the Compliance container. Use this parameter to override the default value.
collector.complianceResources.limits.memory	The memory limit for the Compliance container. Use this parameter to override the default value.

Parameter	Description
collector.complianceResources.limits.cpu	The CPU limit for the Compliance container. Use this parameter to override the default value.
collector.serviceTLS.cert	The internal service-to-service TLS certificate that Collector uses.
collector.serviceTLS.key	The internal service-to-service TLS certificate key that Collector uses.
admissionControl.listenOnCreates	This setting controls whether Kubernetes is configured to contact Red Hat Advanced Cluster Security for Kubernetes with **AdmissionReview** requests for workload creation events.
admissionControl.listenOnUpdates	When you set this parameter as **false**, Red Hat Advanced Cluster Security for Kubernetes creates the **ValidatingWebhookConfiguration** in a way that causes the Kubernetes API server not to send object update events. Since the volume of object updates is usually higher than the object creates, leaving this as **false** limits the load on the admission control service and decreases the chances of a malfunctioning admission control service.
admissionControl.listenOnEvents	This setting controls whether the cluster is configured to contact Red Hat Advanced Cluster Security for Kubernetes with **AdmissionReview** requests for Kubernetes **exec** and **portforward** events. Red Hat Advanced Cluster Security for Kubernetes does not support this feature on OpenShift Container Platform 3.11. For more information, see Red Hat Advanced Cluster Security for Kubernetes Support Policy.
admissionControl.dynamic.enforceOnCreates	This setting controls whether Red Hat Advanced Cluster Security for Kubernetes evaluates policies; if it is disabled, all AdmissionReview requests are automatically accepted.
admissionControl.dynamic.enforceOnUpdates	This setting controls the behavior of the admission control service. You must specify **listenOnUpdates** as **true** for this to work.

Parameter	Description
admissionControl.dynamic.scanInline	If you set this option to **true**, the admission control service requests an image scan before making an admission decision. Since image scans take several seconds, enable this option only if you can ensure that all images used in your cluster are scanned before deployment (for example, by a CI integration during image build). This option corresponds to the **Contact image scanners** option in the RHACS Portal.
admissionControl.dynamic.disableBypass	Set it to **true** to disable bypassing the Admission controller.
admissionControl.dynamic.timeout	The maximum time, in seconds, Red Hat Advanced Cluster Security for Kubernetes should wait while evaluating admission review requests. Use this to set request timeouts when you enable image scanning. If the image scan runs longer than the specified time, Red Hat Advanced Cluster Security for Kubernetes accepts the request.
admissionControl.resources.requests.memory	The memory request for the Admission Control container. Use this parameter to override the default value.
admissionControl.resources.requests.cpu	The CPU request for the Admission Control container. Use this parameter to override the default value.
admissionControl.resources.limits.memory	The memory limit for the Admission Control container. Use this parameter to override the default value.
admissionControl.resources.limits.cpu	The CPU limit for the Admission Control container. Use this parameter to override the default value.
admissionControl.nodeSelector	Specify a node selector label as **label-key: label-value** to force Admission Control to only schedule on nodes with the specified label.
admissionControl.tolerations	If the node selector selects tainted nodes, use this parameter to specify a taint toleration key, value, and effect for Admission Control. This parameter is mainly used for infrastructure nodes.
admissionControl.serviceTLS.cert	The internal service-to-service TLS certificate that Admission Control uses.

Parameter	Description
admissionControl.serviceTLS.key	The internal service-to-service TLS certificate key that Admission Control uses.
registryOverride	Use this parameter to override the default **docker.io** registry. Specify the name of your registry if you are using some other registry.
collector.disableTaintTolerations	If you specify **false**, tolerations are applied to Collector, and the Collector pods can schedule onto all nodes with taints. If you specify it as **true**, no tolerations are applied, and the Collector pods are not scheduled onto nodes with taints.
createUpgraderServiceAccount	Specify **true** to create the **sensor-upgrader** account. 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 do not create this account, you must complete future upgrades manually if the Sensor does not have enough permissions.
createSecrets	Specify **false** to skip the orchestrator secret creation for the Sensor, Collector, and Admission controller.
collector.slimMode	Specify **true** if you want to use a slim Collector image for deploying Collector. Using slim Collector images requires Central to provide the matching eBPF probe or kernel module. If you are running Red Hat Advanced Cluster Security for Kubernetes in offline mode, you must download a kernel support package from stackrox.io and upload it to Central for slim Collectors to function. Otherwise, you must ensure that Central can access the online probe repository hosted at https://collector-modules.stackrox.io/.
sensor.resources	Resource specification for Sensor.
admissionControl.resources	Resource specification for Admission controller.
collector.resources	Resource specification for Collector.
collector.complianceResources	Resource specification for Collector’s Compliance container.

Parameter	Description
exposeMonitoring	If you set this option to **true**, Red Hat Advanced Cluster Security for Kubernetes exposes Prometheus metrics endpoints on port number 9090 for the Sensor, Collector, and the Admission controller.
auditLogs.disableCollection	If you set this option to **true**, Red Hat Advanced Cluster Security for Kubernetes disables the audit log detection features used to detect access and modifications to configuration maps and secrets.
scanner.disable	If you set this option to **false**, Red Hat Advanced Cluster Security for Kubernetes deploys a Scanner-slim and Scanner DB in the secured cluster to allow scanning images on OpenShift Container Registry. Enabling Scanner-slim is supported on OpenShift Container Platform and Kubernetes secured clusters. Defaults to **true**.
scanner.dbTolerations	If the node selector selects tainted nodes, use this parameter to specify a taint toleration key, value, and effect for Scanner DB.
scanner.replicas	Resource specification for Collector's Compliance container.
scanner.logLevel	Setting this parameter allows you to modify the scanner log level. Use this option only for troubleshooting purposes.
scanner.autoscaling.disable	If you set this option to **true**, Red Hat Advanced Cluster Security for Kubernetes disables autoscaling on the Scanner deployment.
scanner.autoscaling.minReplicas	The minimum number of replicas for autoscaling. Defaults to 2.
scanner.autoscaling.maxReplicas	The maximum number of replicas for autoscaling. Defaults to 5.
scanner.nodeSelector	Specify a node selector label as **label-key: label-value** to force Scanner to only schedule on nodes with the specified label.
scanner.tolerations	If the node selector selects tainted nodes, use this parameter to specify a taint toleration key, value, and effect for Scanner.

Parameter	Description
scanner.dbNodeSelector	Specify a node selector label as **label-key: label-value** to force Scanner DB to only schedule on nodes with the specified label.
scanner.dbTolerations	If the node selector selects tainted nodes, use this parameter to specify a taint toleration key, value, and effect for Scanner DB.
scanner.resources.requests.memory	The memory request for the Scanner container. Use this parameter to override the default value.
scanner.resources.requests.cpu	The CPU request for the Scanner container. Use this parameter to override the default value.
scanner.resources.limits.memory	The memory limit for the Scanner container. Use this parameter to override the default value.
scanner.resources.limits.cpu	The CPU limit for the Scanner container. Use this parameter to override the default value.
scanner.dbResources.requests.memory	The memory request for the Scanner DB container. Use this parameter to override the default value.
scanner.dbResources.requests.cpu	The CPU request for the Scanner DB container. Use this parameter to override the default value.
scanner.dbResources.limits.memory	The memory limit for the Scanner DB container. Use this parameter to override the default value.
scanner.dbResources.limits.cpu	The CPU limit for the Scanner DB container. Use this parameter to override the default value.
monitoring.openshift.enabled	If you set this option to **false**, Red Hat Advanced Cluster Security for Kubernetes will not set up Red Hat OpenShift monitoring. Defaults to **true** on Red Hat OpenShift 4.

6.4.1.3.1.1. Environment variables

You can specify environment variables for Sensor and Admission controller in the following format:

```
customize:
  envVars:
    ENV_VAR1: "value1"
    ENV_VAR2: "value2"
```

The **customize** setting allows you to specify custom Kubernetes metadata (labels and annotations) for all objects created by this Helm chart and additional pod labels, pod annotations, and container environment variables for workloads.

The configuration is hierarchical, in the sense that metadata defined at a more generic scope (for example, for all objects) can be overridden by metadata defined at a narrower scope (for example, only for the Sensor deployment).

6.4.1.3.2. Installing the secured-cluster-services Helm chart

After you configure the **values-public.yaml** and **values-private.yaml** files, install the **secured-cluster-services** Helm chart to deploy the per-cluster and per-node components (Sensor, Admission controller, Collector, and Scanner-slim).

CAUTION

To install Collector on systems that have Unified Extensible Firmware Interface (UEFI) and that have Secure Boot enabled, you must use eBPF probes because kernel modules are unsigned, and the UEFI firmware cannot load unsigned packages. Collector identifies Secure Boot status at the start and switches to eBPF probes if required.

Prerequisites

- You must have generated an RHACS init bundle for your cluster.
- You must have access to the Red Hat Container Registry and a pull secret for authentication. For information about downloading images from **registry.redhat.io**, see Red Hat Container Registry Authentication.
- You must have the **Central API Endpoint**, including the address and the port number. You can view this information by choosing **Advanced Cluster Security → ACS Instances** from the cloud console navigation menu, then clicking the ACS instance you created.

Procedure

- Run the following command:

```
$ helm install -n stackrox \
  --create-namespace stackrox-secured-cluster-services rhacs/secured-cluster-services \
  -f <name_of_cluster_init_bundle.yaml> \
  -f <path_to_values_public.yaml> -f <path_to_values_private.yaml> \ 1
  --set imagePullSecrets.username=<username> \ 2
  --set imagePullSecrets.password=<password> 3
```

1 Use the **-f** option to specify the paths for your YAML configuration files.

2 Include the user name for your pull secret for Red Hat Container Registry authentication.

3 Include the password for your pull secret for Red Hat Container Registry authentication.

NOTE

To deploy **secured-cluster-services** Helm chart by using a continuous integration (CI) system, pass the init bundle YAML file as an environment variable to the **helm install** command:

```
$ helm install ... -f <(echo "$INIT_BUNDLE_YAML_SECRET") 1
```

1 If you are using base64 encoded variables, use the **helm install … -f <(echo "$INIT_BUNDLE_YAML_SECRET" | base64 --decode)** command instead.

6.4.1.4. Changing configuration options after deploying the secured-cluster-services Helm chart

You can make changes to any configuration options after you have deployed the **secured-cluster-services** Helm chart.

Procedure

1. Update the **values-public.yaml** and **values-private.yaml** configuration files with new values.
2. Run the **helm upgrade** command and specify the configuration files using the **-f** option:

```
$ helm upgrade -n stackrox \
  stackrox-secured-cluster-services rhacs/secured-cluster-services \
  --reuse-values \ 1
  -f <path_to_values_public.yaml> \
  -f <path_to_values_private.yaml>
```

1 You must specify the **--reuse-values** parameter, otherwise the Helm upgrade command resets all previously configured settings.

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.

6.4.2. Installing RHACS on secured clusters by using the roxctl CLI

To install RHACS on secured clusters by using the CLI, perform the following steps:

1. Install the **roxctl** CLI.
2. Install Sensor.

6.4.2.1. Installing the roxctl CLI

You must first download the binary. You can install **roxctl** on Linux, Windows, or macOS.

6.4.2.1.1. Installing the roxctl CLI on Linux

You can install the **roxctl** CLI binary on Linux by using the following procedure.

NOTE

roxctl CLI for Linux is available for **amd64**, **ppc64le**, and **s390x** architectures.

Procedure

1. Determine the **roxctl** architecture for the target operating system:

```
$ arch="$(uname -m | sed "s/x86_64//")"; arch="${arch:+-$arch}"
```

2. Download the **roxctl** CLI:

```
$ curl -f -o roxctl "https://mirror.openshift.com/pub/rhacs/assets/4.3.8/bin/Linux/roxctl${arch}"
```

3. Make the **roxctl** binary executable:

```
$ chmod +x roxctl
```

4. Place the **roxctl** binary in a directory that is on your **PATH**:
   To check your **PATH**, execute the following command:

```
$ echo $PATH
```

Verification

- Verify the **roxctl** version you have installed:

```
$ roxctl version
```

6.4.2.1.2. Installing the roxctl CLI on macOS

You can install the **roxctl** CLI binary on macOS by using the following procedure.

NOTE

roxctl CLI for macOS is available for the **amd64** architecture.

Procedure

1. Download the **roxctl** CLI:

```
$ curl -f -O https://mirror.openshift.com/pub/rhacs/assets/4.3.8/bin/Darwin/roxctl
```

2. Remove all extended attributes from the binary:

```
$ xattr -c roxctl
```

3. Make the **roxctl** binary executable:

```
$ chmod +x roxctl
```

4. Place the **roxctl** binary in a directory that is on your **PATH**:
   To check your **PATH**, execute the following command:

   ```
   $ echo $PATH
   ```

Verification

- Verify the **roxctl** version you have installed:

  ```
  $ roxctl version
  ```

6.4.2.1.3. Installing the roxctl CLI on Windows

You can install the **roxctl** CLI binary on Windows by using the following procedure.

> **NOTE**
>
> **roxctl** CLI for Windows is available for the **amd64** architecture.

Procedure

- Download the **roxctl** CLI:

  ```
  $ curl -f -O https://mirror.openshift.com/pub/rhacs/assets/4.3.8/bin/Windows/roxctl.exe
  ```

Verification

- Verify the **roxctl** version you have installed:

  ```
  $ roxctl version
  ```

6.4.2.2. Installing Sensor

To monitor a cluster, you must deploy Sensor. You must deploy Sensor into each cluster that you want to monitor. The following steps describe adding Sensor by using the RHACS portal.

Prerequisites

- You must have already installed Central services, or you can access Central services by selecting your **ACS instance** on Red Hat Advanced Cluster Security Cloud Service (RHACS Cloud Service).

Procedure

1. On your secured cluster, in the RHACS portal, navigate to **Platform Configuration → Clusters**.

2. Select **+ New Cluster**.

3. Specify a name for the cluster.

4. Provide appropriate values for the fields based on where you are deploying the Sensor.

- Enter the Central API Endpoint, including the address and the port number. You can view this information again in the Red Hat Hybrid Cloud Console by choosing **Advanced Cluster Security → ACS Instances**, and then clicking the ACS instance you created.

5. Click **Next** to continue with the Sensor setup.

6. Click **Download YAML File and Keys**to download the cluster bundle (zip archive).

IMPORTANT

The cluster bundle zip archive includes unique configurations and keys for each cluster. Do not reuse the same files in another cluster.

7. From a system that has access to the monitored cluster, unzip and run the **sensor** script from the cluster bundle:

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

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

If you get a warning that you do not have the required permissions to deploy Sensor, follow the on-screen instructions, or contact your cluster administrator for assistance.

After Sensor is deployed, it contacts Central and provides cluster information.

Verification

1. Return to the RHACS portal and check if the deployment is successful. If successful, when viewing your list of clusters in **Platform Configuration → Clusters**, the cluster status displays a green checkmark and a **Healthy** status. If you do not see a green checkmark, use the following command to check for problems:

   - On Kubernetes, enter the following command:

   ```
   $ kubectl get pod -n stackrox -w
   ```

2. Click **Finish** to close the window.

After installation, Sensor starts reporting security information to RHACS and the RHACS portal dashboard begins showing deployments, images, and policy violations from the cluster on which you have installed the Sensor.

6.5. VERIFYING INSTALLATION OF SECURED CLUSTERS

After installing RHACS Cloud Service, you can perform some steps to verify that the installation was successful.

To verify installation, access your ACS Console from the Red Hat Hybrid Cloud Console. The Dashboard displays the number of clusters that RHACS Cloud Service is monitoring, along with information about nodes, deployments, images, and violations.

If no data appears in the ACS Console:

- Ensure that at least one secured cluster is connected to your RHACS Cloud Service instance. For more information, see instructions for installing by using Helm charts or by using the **roxctl** CLI.

- Examine your Sensor pod logs to ensure that the connection to your RHACS Cloud Service instance is successful.

- Examine the values in the **SecuredCluster** API in the Operator on your local cluster to ensure that the **Central API Endpoint** has been entered correctly. This value should be the same value as shown in the **ACS instance** details in the Red Hat Hybrid Cloud Console.