



Red Hat Advanced Cluster Security for Kubernetes 4.4

Installing

Installing Red Hat Advanced Cluster Security for Kubernetes

Red Hat Advanced Cluster Security for Kubernetes 4.4 Installing

Installing Red Hat Advanced Cluster Security for Kubernetes

Legal Notice

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

This document describes how to install Red Hat Advanced Cluster Security for Kubernetes by using the Operator, Helm charts, or the roxctl CLI.

Table of Contents

CHAPTER 1. HIGH-LEVEL RHACS INSTALLATION OVERVIEW	7
1.1. GENERAL INSTALLATION GUIDELINES	7
1.2. INSTALLATION METHODS FOR DIFFERENT PLATFORMS	7
1.3. INSTALLATION METHODS FOR DIFFERENT ARCHITECTURES	8
1.4. INSTALLATION STEPS FOR RHACS ON OPENSIFT CONTAINER PLATFORM	9
1.4.1. Installing RHACS on Red Hat OpenShift by using the RHACS Operator	9
1.4.2. Installing RHACS on Red Hat OpenShift by using Helm charts	9
1.4.3. Installing RHACS on Red Hat OpenShift by using the roxctl CLI	10
1.5. INSTALLATION STEPS FOR RHACS ON KUBERNETES	10
1.5.1. Installing RHACS on Kubernetes platforms by using Helm charts	10
1.5.2. Installing RHACS on Kubernetes platforms by using the roxctl CLI	10
CHAPTER 2. DEFAULT RESOURCE REQUIREMENTS FOR RED HAT ADVANCED CLUSTER SECURITY FOR KUBERNETES	12
2.1. GENERAL RHACS REQUIREMENTS	12
2.2. CENTRAL SERVICES (SELF-MANAGED)	13
2.2.1. Central	13
Memory, CPU, and storage requirements	14
2.2.2. Scanner	15
Memory and CPU requirements	15
StackRox Scanner	15
StackRox Scanner-DB	15
Scanner V4 (Technology Preview)	15
2.3. SECURED CLUSTER SERVICES	16
2.3.1. Sensor	16
Memory and CPU requirements	16
2.3.2. Admission controller	16
Memory and CPU requirements	16
2.3.3. Collector	17
Collection requirements	17
Memory and CPU requirements	17
Collector container	17
Compliance container	17
Node-inventory container	18
Total collector replica requirements	18
2.3.4. Scanner V4 (Technology Preview)	18
CHAPTER 3. RECOMMENDED RESOURCE REQUIREMENTS FOR RED HAT ADVANCED CLUSTER SECURITY FOR KUBERNETES	20
3.1. CENTRAL SERVICES (SELF-MANAGED)	20
3.1.1. Central	20
Memory and CPU requirements	20
3.1.2. Scanner	21
StackRox Scanner Memory and CPU requirements	21
3.2. SECURED CLUSTER SERVICES	21
3.2.1. Sensor	21
Memory and CPU requirements	22
3.2.2. Admission controller	22
Memory and CPU requirements	22
CHAPTER 4. INSTALLING RHACS ON RED HAT OPENSIFT	23
4.1. INSTALLING CENTRAL SERVICES FOR RHACS ON RED HAT OPENSIFT	23

4.1.1. Install Central using the Operator	23
4.1.1.1. Installing the Red Hat Advanced Cluster Security for Kubernetes Operator	23
4.1.1.2. Installing Central using the Operator method	24
4.1.1.3. Provisioning a database in your PostgreSQL instance	28
4.1.1.4. Installing Central with an external database using the Operator method	29
4.1.1.5. Verifying Central installation using the Operator method	32
4.1.2. Install Central using Helm charts	33
4.1.2.1. Install Central using Helm charts without customization	33
4.1.2.1.1. Adding the Helm chart repository	33
4.1.2.1.2. Installing the central-services Helm chart without customizations	34
4.1.2.2. Install Central using Helm charts with customizations	35
4.1.2.2.1. Private configuration file	36
4.1.2.2.1.1. Image pull secrets	36
4.1.2.2.1.2. Proxy configuration	36
4.1.2.2.1.3. Central	37
4.1.2.2.1.4. Scanner	38
4.1.2.2.2. Public configuration file	40
4.1.2.2.2.1. Image pull secrets	40
4.1.2.2.2.2. Image	41
4.1.2.2.2.3. Environment variables	41
4.1.2.2.2.4. Additional trusted certificate authorities	41
4.1.2.2.2.5. Central	42
4.1.2.2.2.6. StackRox Scanner	46
4.1.2.2.2.7. Scanner V4	47
4.1.2.2.2.8. Customization	50
4.1.2.2.2.9. Advanced customization	53
4.1.2.2.3. Declarative configuration values	53
4.1.2.2.4. Installing the central-services Helm chart	54
4.1.2.3. Changing configuration options after deploying the central-services Helm chart	54
4.1.3. Install Central using the roxctl CLI	55
4.1.3.1. Installing the roxctl CLI	55
4.1.3.1.1. Installing the roxctl CLI on Linux	55
4.1.3.1.2. Installing the roxctl CLI on macOS	56
4.1.3.1.3. Installing the roxctl CLI on Windows	57
4.1.3.2. Using the interactive installer	57
4.1.3.3. Running the Central installation scripts	59
4.2. CONFIGURING CENTRAL CONFIGURATION OPTIONS FOR RHACS USING THE OPERATOR	60
4.2.1. Central configuration options using the Operator	60
4.2.1.1. Central settings	60
4.2.1.2. StackRox Scanner settings	63
4.2.1.3. Scanner V4 settings (Technology Preview)	64
4.2.1.4. General and miscellaneous settings	66
4.2.2. Customizing the installation using the Operator with overlays	67
4.2.2.1. Overlays	67
4.2.2.1.1. Adding an overlay	68
4.2.2.2. Overlay examples	68
4.2.2.2.1. Specifying an EKS pod role ARN for the Central ServiceAccount	68
4.2.2.2.2. Injecting an environment variable into the Central deployment	68
4.2.2.2.3. Extending network policy with an ingress rule	69
4.2.2.2.4. Modifying ConfigMap data	69
4.2.2.2.5. Adding a container to the Central deployment	69
4.3. GENERATING AND APPLYING AN INIT BUNDLE FOR RHACS ON RED HAT OPENSIFT	70
4.3.1. Generating an init bundle	70

4.3.1.1. Generating an init bundle by using the RHACS portal	70
4.3.1.2. Generating an init bundle by using the roxctl CLI	71
4.3.1.3. Applying the init bundle on the secured cluster	72
4.3.2. Next steps	73
4.3.3. Additional resources	73
4.4. INSTALLING SECURED CLUSTER SERVICES FOR RHACS ON RED HAT OPENSIFT	73
4.4.1. Installing RHACS on secured clusters by using the Operator	73
4.4.1.1. Installing secured cluster services	73
4.4.2. Installing RHACS on secured clusters by using Helm charts	74
4.4.2.1. Installing RHACS on secured clusters by using Helm charts without customizations	74
4.4.2.1.1. Adding the Helm chart repository	75
4.4.2.1.2. Installing the secured-cluster-services Helm chart without customization	75
4.4.2.2. Configuring the secured-cluster-services Helm chart with customizations	76
4.4.2.2.1. Configuration parameters	76
4.4.2.2.1.1. Environment variables	83
4.4.2.2.2. Installing the secured-cluster-services Helm chart with customizations	84
4.4.2.3. Changing configuration options after deploying the secured-cluster-services Helm chart	85
4.4.3. Installing RHACS on secured clusters by using the roxctl CLI	86
4.4.3.1. Installing the roxctl CLI	86
4.4.3.1.1. Installing the roxctl CLI on Linux	86
4.4.3.1.2. Installing the roxctl CLI on macOS	87
4.4.3.1.3. Installing the roxctl CLI on Windows	88
4.4.3.2. Installing Sensor	88
4.4.3.2.1. Manifest installation method by using the web portal	88
4.4.3.2.2. Manifest installation by using the roxctl CLI	89
4.5. CONFIGURING SECURED CLUSTER SERVICES OPTIONS FOR RHACS USING THE OPERATOR	90
4.5.1. Secured Cluster services configuration options	90
4.5.1.1. Required Configuration Settings	90
4.5.1.2. Admission controller settings	91
4.5.1.3. Scanner configuration	92
4.5.1.4. Image configuration	93
4.5.1.5. Per node settings	93
4.5.1.6. Taint Tolerations settings	94
4.5.1.7. Sensor configuration	94
4.5.1.8. General and miscellaneous settings	95
4.5.2. Customizing the installation using the Operator with overlays	95
4.5.2.1. Overlays	95
4.5.2.1.1. Adding an overlay	96
4.5.2.2. Overlay examples	97
4.5.2.2.1. Specifying an EKS pod role ARN for the Central ServiceAccount	97
4.5.2.2.2. Injecting an environment variable into the Central deployment	97
4.5.2.2.3. Extending network policy with an ingress rule	97
4.5.2.2.4. Modifying ConfigMap data	98
4.5.2.2.5. Adding a container to the Central deployment	98
4.6. VERIFYING INSTALLATION OF RHACS ON RED HAT OPENSIFT	99
4.6.1. Verifying installation	99
CHAPTER 5. INSTALLING RHACS ON OTHER PLATFORMS	101
5.1. HIGH-LEVEL OVERVIEW OF INSTALLING RHACS ON OTHER PLATFORMS	101
5.2. INSTALLING CENTRAL SERVICES FOR RHACS ON OTHER PLATFORMS	101
5.2.1. Install Central using Helm charts	101
5.2.1.1. Install Central using Helm charts without customization	101
5.2.1.1.1. Adding the Helm chart repository	101

5.2.1.1.2. Installing the central-services Helm chart without customizations	102
5.2.1.2. Install Central using Helm charts with customizations	104
5.2.1.2.1. Private configuration file	104
5.2.1.2.1.1. Image pull secrets	104
5.2.1.2.1.2. Proxy configuration	105
5.2.1.2.1.3. Central	105
5.2.1.2.1.4. Scanner	107
5.2.1.2.2. Public configuration file	109
5.2.1.2.2.1. Image pull secrets	109
5.2.1.2.2.2. Image	109
5.2.1.2.2.3. Environment variables	109
5.2.1.2.2.4. Additional trusted certificate authorities	110
5.2.1.2.2.5. Central	110
5.2.1.2.2.6. StackRox Scanner	114
5.2.1.2.2.7. Scanner V4	116
5.2.1.2.2.8. Customization	119
5.2.1.2.2.9. Advanced customization	121
5.2.1.2.3. Declarative configuration values	122
5.2.1.2.4. Installing the central-services Helm chart	122
5.2.1.3. Changing configuration options after deploying the central-services Helm chart	123
5.2.2. Install Central using the roxctl CLI	124
5.2.2.1. Installing the roxctl CLI	124
5.2.2.1.1. Installing the roxctl CLI on Linux	124
5.2.2.1.2. Installing the roxctl CLI on macOS	125
5.2.2.1.3. Installing the roxctl CLI on Windows	125
5.2.2.2. Using the interactive installer	126
5.2.2.3. Running the Central installation scripts	127
5.3. GENERATING AND APPLYING AN INIT BUNDLE FOR RHACS ON OTHER PLATFORMS	128
5.3.1. Generating an init bundle	129
5.3.1.1. Generating an init bundle by using the RHACS portal	129
5.3.1.2. Generating an init bundle by using the roxctl CLI	129
5.3.1.3. Applying the init bundle on the secured cluster	130
5.3.2. Next steps	131
5.4. INSTALLING SECURED CLUSTER SERVICES FOR RHACS ON OTHER PLATFORMS	131
5.4.1. Installing RHACS on secured clusters by using Helm charts	131
5.4.1.1. Installing RHACS on secured clusters by using Helm charts without customizations	132
5.4.1.1.1. Adding the Helm chart repository	132
5.4.1.1.2. Installing the secured-cluster-services Helm chart without customization	132
5.4.1.2. Configuring the secured-cluster-services Helm chart with customizations	133
5.4.1.2.1. Configuration parameters	133
5.4.1.2.1.1. Environment variables	140
5.4.1.2.2. Installing the secured-cluster-services Helm chart with customizations	140
5.4.1.3. Changing configuration options after deploying the secured-cluster-services Helm chart	142
5.4.2. Installing RHACS on secured clusters by using the roxctl CLI	142
5.4.2.1. Installing the roxctl CLI	143
5.4.2.1.1. Installing the roxctl CLI on Linux	143
5.4.2.1.2. Installing the roxctl CLI on macOS	143
5.4.2.1.3. Installing the roxctl CLI on Windows	144
5.4.2.2. Installing Sensor	144
5.4.2.2.1. Manifest installation method by using the web portal	145
5.4.2.2.2. Manifest installation by using the roxctl CLI	146
5.5. VERIFYING INSTALLATION OF RHACS ON OTHER PLATFORMS	147
5.5.1. Verifying installation	147

CHAPTER 6. UNINSTALLING RED HAT ADVANCED CLUSTER SECURITY FOR KUBERNETES	148
6.1. DELETING NAMESPACE	148
6.2. DELETING GLOBAL RESOURCES	148
6.3. DELETING LABELS AND ANNOTATIONS	149

CHAPTER 1. HIGH-LEVEL RHACS INSTALLATION OVERVIEW

Red Hat Advanced Cluster Security for Kubernetes (RHACS) provides security services for your self-managed Red Hat OpenShift Kubernetes systems or platforms such as OpenShift Container Platform, Amazon Elastic Kubernetes Service (Amazon EKS), Google Kubernetes Engine (Google GKE), and Microsoft Azure Kubernetes Service (Microsoft AKS).

For information about supported platforms and architecture, see the [Red Hat Advanced Cluster Security for Kubernetes Support Matrix](#). For life cycle support information for RHACS, see the [Red Hat Advanced Cluster Security for Kubernetes Support Policy](#).

1.1. GENERAL INSTALLATION GUIDELINES

To ensure the best installation experience, follow these guidelines:

1. Understand the installation platforms and methods described in this module.
2. Understand [Red Hat Advanced Cluster Security for Kubernetes architecture](#).
3. Check the [default resource requirements](#).

1.2. INSTALLATION METHODS FOR DIFFERENT PLATFORMS

You can perform different types of installations on different platforms.



NOTE

Not all installation methods are supported for all platforms. See the [Red Hat Advanced Cluster Security for Kubernetes Support Matrix](#) for more information.

Table 1.1. Platforms and recommended installation methods

Platform type	Platform	Recommended installation methods	Installation steps
Managed service platform	Red Hat OpenShift Dedicated (OSD)	Operator (recommended), Helm charts, or roxctl CLI ^[1]	<ul style="list-style-type: none"> • Installing Central services for RHACS on Red Hat OpenShift • Installing Secured Cluster services for RHACS on Red Hat OpenShift
	Azure Red Hat OpenShift (ARO)		
	Red Hat OpenShift Service on AWS (ROSA)		
	Red Hat OpenShift on IBM Cloud		

Platform type	Platform	Recommended installation methods	Installation steps
	Amazon Elastic Kubernetes Service (Amazon EKS)	Helm charts (recommended), or roxctl CLI ^[1]	<ul style="list-style-type: none"> Installing Central services for RHACS on other platforms Installing Secured Cluster services for RHACS on other platforms
	Google Kubernetes Engine (Google GKE)		
	Microsoft Azure Kubernetes Service (Microsoft AKS)		
Self-managed platform	Red Hat OpenShift Container Platform (OCP)	Operator (recommended), Helm charts, or roxctl CLI ^[1]	<ul style="list-style-type: none"> Installing Central services for RHACS on Red Hat OpenShift Installing Secured Cluster services for RHACS on Red Hat OpenShift
	Red Hat OpenShift Kubernetes Engine (OKE)		

1. Do not use the **roxctl** installation method unless you have specific requirements for following this installation method.

1.3. INSTALLATION METHODS FOR DIFFERENT ARCHITECTURES

Red Hat Advanced Cluster Security for Kubernetes (RHACS) supports the following architectures. For information on supported platforms and architecture, see the [Red Hat Advanced Cluster Security for Kubernetes Support Matrix](#). Additionally, the following table gives information about installation methods available for each architecture.

Table 1.2. Architectures and supported installation methods for each architecture

Supported architectures	Supported installation methods
AMD64	Operator (preferred), Helm charts, or roxctl CLI (not recommended)
ppc64le (IBM Power)	Operator
s390x (IBM Z and IBM® LinuxONE)	

1.4. INSTALLATION STEPS FOR RHACS ON OPENSIFT CONTAINER PLATFORM

1.4.1. Installing RHACS on Red Hat OpenShift by using the RHACS Operator

1. On the Red Hat OpenShift cluster, install the RHACS Operator into the **rhacs-operator** project, or namespace.
2. On the Red Hat OpenShift cluster that will contain Central, called the central cluster, use the RHACS Operator to install Central services into the **stackrox** project. One central cluster can secure multiple clusters.
3. Log in to the RHACS web console from the central cluster, and then create an init bundle and download it. The init bundle is then installed on the cluster that you want to secure, called the secured cluster.
4. For the secured cluster:
 - a. Install the RHACS Operator into the **rhacs-operator** namespace.
 - b. On the secured cluster, apply the init bundle that you created in RHACS by performing one of these steps:
 - Use the OpenShift Container Platform web console to import the YAML file of the init bundle that you created. Make sure you are in the **stackrox** namespace.
 - In the terminal window, run the **oc create -f <init_bundle>.yaml -n <stackrox>** command, specifying the path to the downloaded YAML file of the init bundle.
 - c. On the secured cluster, use the RHACS Operator to install Secured Cluster services into the **stackrox** namespace. When creating these services, be sure to enter the address and port number of Central in the **Central Endpoint** field so that the secured cluster can communicate with Central.

1.4.2. Installing RHACS on Red Hat OpenShift by using Helm charts

1. Add the RHACS Helm charts repository.
2. Install the **central-services** Helm chart on the Red Hat OpenShift cluster that will contain Central, called the central cluster.
3. Log in to the RHACS web console on the Central cluster and create an init bundle.
4. For each cluster that you want to secure, log in to the secured cluster and perform the following steps:
 - a. Apply the init bundle you created with RHACS. To apply the init bundle on the secured cluster, perform one of these steps:
 - Use the OpenShift Container Platform web console to import the YAML file of the init bundle that you created. Make sure you are in the **stackrox** namespace.
 - In the terminal window, run the **oc create -f <init_bundle>.yaml -n <stackrox>** command, specifying the path to the downloaded YAML file of the init bundle.

- b. Install the **secured-cluster-services** Helm chart on the secured cluster, specifying the path to the init bundle that you created.

1.4.3. Installing RHACS on Red Hat OpenShift by using the roxctl CLI

This installation method is also called the *manifest installation method*.

1. Install the **roxctl** CLI.
2. On the Red Hat OpenShift cluster that will contain Central, perform these steps:
 - a. In the terminal window, run the interactive install command by using the **roxctl** CLI.
 - b. Run the setup shell script.
 - c. In the terminal window, create the Central resources by using the **oc create** command.
3. Perform one of the following actions:
 - In the RHACS web console, create and download the sensor YAML file and keys.
 - On the secured cluster, use the **roxctl sensor generate openshift** command.
4. On the secured cluster, run the sensor installation script.

1.5. INSTALLATION STEPS FOR RHACS ON KUBERNETES

1.5.1. Installing RHACS on Kubernetes platforms by using Helm charts

1. Add the RHACS Helm charts repository.
2. Install the **central-services** Helm chart on the cluster that will contain Central, called the Central cluster.
3. Log in to the RHACS web console from the Central cluster and create an init bundle that you will install on the cluster that you want to secure, called the secured cluster.
4. For each secured cluster:
 - a. Apply the init bundle you created with RHACS. Log in to the secured cluster and run the **kubectrl create -f <init_bundle>.yaml -n <stackrox>** command, specifying the path to the downloaded YAML file of the init bundle.
 - b. Install the **secured-cluster-services** Helm chart on the secured cluster, specifying the path to the init bundle that you created earlier.

1.5.2. Installing RHACS on Kubernetes platforms by using the roxctl CLI

This installation method is also called the *manifest installation method*.

1. Install the **roxctl** CLI.
2. On the Kubernetes cluster that will contain Central, perform these steps:
 - a. In the terminal window, run the interactive install command by using the **roxctl** CLI.

- b. Run the setup shell script.
 - c. In the terminal window, create the Central resources by using the **kubectl create** command.
3. Perform one of the following actions:
 - In the RHACS web console, create and download the sensor YAML file and keys.
 - On the cluster that you want to secure, called the secured cluster, use the **roxctl sensor generate openshift** command.
4. On the secured cluster, run the sensor installation script.

CHAPTER 2. DEFAULT RESOURCE REQUIREMENTS FOR RED HAT ADVANCED CLUSTER SECURITY FOR KUBERNETES

2.1. GENERAL RHACS REQUIREMENTS

Before you can install RHACS, your system must meet several requirements.



WARNING

You must not install RHACS 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.

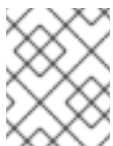
To install RHACS, you must have one of the following systems:

- 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 information about supported platforms and architecture, see the [Red Hat Advanced Cluster Security for Kubernetes Support Matrix](#). For life cycle support information for RHACS, see the [Red Hat Advanced Cluster Security for Kubernetes Support Policy](#).

The following minimum requirements and suggestions apply to cluster nodes.

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 are required.

Memory

6 GiB of RAM is required.

**NOTE**

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

Storage

A persistent volume claim (PVC) is required on the cluster where Central is installed. It is strongly recommended on the secured clusters where Scanner V4 is enabled. 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.

If you plan to install RHACS by using Helm charts, you must meet the following requirements:

- You must have Helm command-line interface (CLI) v3.2 or newer, if you are installing or configuring RHACS 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](#).

2.2. CENTRAL SERVICES (SELF-MANAGED)

**NOTE**

If you are using Red Hat Advanced Cluster Security Cloud Service (RHACS Cloud Service), you do not need to review the requirements for Central services, because they are managed by Red Hat. You only need to look at the requirements for secured cluster services.

Central services contain the following components:

- Central
- Scanner

2.2.1. Central

A containerized service called Central handles API interactions and RHACS web portal access while a containerized service called Central DB (PostgreSQL 13) handles data persistence.

Central DB and Scanner V4 require persistent storage in the cluster where Central is installed.

- You can provide storage with a persistent volume claim (PVC).

**NOTE**

You can use a hostPath volume for storage only if all your hosts (or a group of hosts) mount a shared file system, such as an NFS share or a storage appliance. Otherwise, your data is only saved on a single node. Red Hat does not recommend using a hostPath volume.

- Use Solid-State Drives (SSD) for best performance. However, you can use another storage type if you do not have SSDs available.
- If you use a web proxy or firewall, you must configure bypass rules to allow traffic for the **definitions.stackrox.io** and **collector-modules.stackrox.io** domains and enable Red Hat Advanced Cluster Security for Kubernetes to trust your web proxy or firewall. Otherwise, updates for vulnerability definitions and kernel support packages will fail.
Red Hat Advanced Cluster Security for Kubernetes requires access to:
 - **definitions.stackrox.io** for downloading updated vulnerability definitions. Vulnerability definition updates allow Red Hat Advanced Cluster Security for Kubernetes to maintain up-to-date vulnerability data when new vulnerabilities are discovered or additional data sources are added.
 - **collector-modules.stackrox.io** to download updated kernel support packages. Updated Kernel support packages ensure that Red Hat Advanced Cluster Security for Kubernetes can monitor the latest operating systems and collect data about the network traffic and processes running inside the containers. Without these updates, Red Hat Advanced Cluster Security for Kubernetes might fail to monitor containers if you add new nodes in your cluster or if you update your nodes' operating system.

**NOTE**

For security reasons, you should deploy Central in a cluster with limited administrative access.

Memory, CPU, and storage requirements

The following table lists the minimum memory and storage values required to install and run Central.

Central	CPU	Memory	Storage
Request	1.5 cores	4 GiB	100 GiB
Limit	4 cores	8 GiB	100 GiB

Central requires Central DB to store data. The following table lists the minimum memory and storage values required to install and run Central DB.

Central DB	CPU	Memory	Storage
Request	4 cores	8 GiB	100 GiB
Limit	8 cores	16 GiB	100 GiB

2.2.2. Scanner

Beginning with version 4.4, RHACS includes two image vulnerability scanners: the StackRox Scanner and Scanner V4. The StackRox Scanner is planned to be removed in a future release, but is required for version 4.4 to perform node and platform scanning. Scanner V4 is the preferred image scanner because it provides additional features over the StackRox Scanner, such as expanded language and operating system support and data from additional vulnerability databases.

Memory and CPU requirements

StackRox Scanner

The requirements in this table are based on the default of 2 replicas.

StackRox Scanner	CPU	Memory
Request	2 cores	3000 MiB
Limit	4 cores	8000 MiB

StackRox Scanner-DB

The StackRox Scanner requires Scanner-DB to store data. The following table lists the minimum memory and storage values required to install and run Scanner-DB.

Scanner-DB	CPU	Memory
Request	0.2 cores	512 MiB
Limit	2 cores	4000 MiB

Scanner V4 (Technology Preview)

Scanner V4 is optional. The requirements in this table are based on the default of 2 replicas.



IMPORTANT

Scanner V4 is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see [Technology Preview Features Support Scope](#).

Scanner V4 Indexer	CPU	Memory
Request	2 cores	3000 MiB
Limit	4 cores	6 GiB

The requirements in this table are based on the default of 2 replicas.

Scanner V4 Matcher	CPU	Memory
Request	2 cores	8 GiB
Limit	4 cores	10 GiB

Scanner V4 requires Scanner V4 DB to store data. The following table lists the minimum memory and storage values required to install and run Scanner V4 DB. For Scanner V4 DB, a PVC is required to ensure optimal performance. This PVC must be 50 GiB.

Scanner V4 DB	CPU	Memory
Request	0.2 cores	3 GiB
Limit	2 cores	4 GiB

2.3. SECURED CLUSTER SERVICES

Secured cluster services contain the following components:

- Sensor
- Admission controller
- Collector

2.3.1. 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

2.3.2. 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

2.3.3. 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 2.1. 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 collector service runs 3 replicas. The following tables list the request and limits for each replica and the total for the collector replicas.

Collector container

Type	CPU	Memory
Request	0.06 cores	320 MiB
Limit	0.9 cores	1000 MiB

Compliance container

Type	CPU	Memory
Request	0.01 cores	10 MiB

Type	CPU	Memory
Limit	1 core	2000 MiB

Node-inventory container

Type	CPU	Memory
Request	0.01 cores	10 MiB
Limit	1 core	500 MiB

Total collector replica requirements

Type	CPU	Memory
Request	0.07 cores	340 MiB
Limit	2.75 cores	3500 MiB

2.3.4. Scanner V4 (Technology Preview)

Scanner V4 is optional. If Scanner V4 is installed on secured clusters, the following requirements apply.

**IMPORTANT**

Scanner V4 is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see [Technology Preview Features Support Scope](#).

The requirements in this table are based on the default of 2 replicas.

Scanner V4 Indexer	CPU	Memory
Request	2 cores	3000 MiB
Limit	4 cores	6 GiB

Scanner V4 requires Scanner V4 DB to store data. The following table lists the minimum memory and storage values required to install and run Scanner V4 DB. For Scanner V4 DB, a PVC is strongly recommended because it ensures optimal performance. The PVC should be 10 GiB.

Scanner V4 DB	CPU	Memory
Request	0.2 cores	3 GiB
Limit	2 cores	4 GiB

CHAPTER 3. RECOMMENDED RESOURCE REQUIREMENTS FOR RED HAT ADVANCED CLUSTER SECURITY FOR KUBERNETES

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 is identified as a primary factor for increasing of used resources. And we are using the number of deployments for the estimation of required resources.

Additional resources

- [Default resource requirements](#)

3.1. CENTRAL SERVICES (SELF-MANAGED)



NOTE

If you are using Red Hat Advanced Cluster Security Cloud Service (RHACS Cloud Service), you do not need to review the requirements for Central services, because they are managed by Red Hat. You only need to look at the requirements for secured cluster services.

Central services contain the following components:

- Central
- Scanner



NOTE

For default resource requirements for the scanner, see the default resource requirements page.

3.1.1. Central

Memory and CPU requirements

The following table lists the minimum memory and CPU values required to run Central for one secured cluster. The table includes the number of concurrent web portal users.

Deployments	Concurrent web portal users	CPU	Memory
< 25,000	1 user	2 cores	8 GiB
< 25,000	< 5 users	6 cores	12 GiB
< 50,000	1 user	2 cores	12 GiB
< 50,000	< 5 users	6 cores	16 GiB

3.1.2. Scanner

StackRox Scanner Memory and CPU requirements

The following table lists the minimum memory and CPU values required for the StackRox Scanner deployment in the Central cluster. The table includes the number of unique images deployed in all secured clusters.

Unique Images	Replicas	CPU	Memory
< 100	1 replica	1 core	1.5 GiB
< 500	1 replica	2 cores	2.5 GiB
< 2000	2 replicas	2 cores	2.5 GiB
< 5000	3 replicas	2 cores	2.5 GiB

Additional resources

- [Default resource requirements](#)

3.2. 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.

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 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

3.2.2. 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 4. INSTALLING RHACS ON RED HAT OPENSIFT

4.1. INSTALLING CENTRAL SERVICES FOR RHACS ON RED HAT OPENSIFT

Central is the resource that contains the RHACS application management interface and services. It handles data persistence, API interactions, and RHACS portal access. You can use the same Central instance to secure multiple OpenShift Container Platform or Kubernetes clusters.

You can install Central on your OpenShift Container Platform or Kubernetes cluster by using one of the following methods:

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

4.1.1. Install Central using the Operator

4.1.1.1. Installing the Red Hat Advanced Cluster Security for Kubernetes Operator

Using the OperatorHub provided with OpenShift Container Platform is the easiest way to install Red Hat Advanced Cluster Security for Kubernetes.

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 information about supported platforms and architecture, see the [Red Hat Advanced Cluster Security for Kubernetes Support Matrix](#). For life cycle support information for RHACS, see the [Red Hat Advanced Cluster Security for Kubernetes Support Policy](#).

Procedure

1. In the web console, go 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**.

- Choose 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 choose 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 choose 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.



IMPORTANT

If you choose manual updates, you must update the RHACS Operator in all secured clusters when you update the RHACS Operator in the cluster where Central is installed. The secured clusters and the cluster where Central is installed must have the same version to ensure optimal functionality.

6. Click **Install**.

Verification

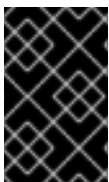
- After the installation completes, go to **Operators → Installed Operators** to verify that the Red Hat Advanced Cluster Security for Kubernetes Operator is listed with the status of **Succeeded**.

Next Step

- You installed the Operator into the **rhacs-operator** project. Using that Operator, install, configure, and deploy the **Central** custom resource into the **stackrox** project.

4.1.1.2. Installing Central using the Operator method

The main component of Red Hat Advanced Cluster Security for Kubernetes is called Central. You can install Central on OpenShift Container Platform by using the **Central** custom resource. You deploy Central only once, and you can monitor multiple separate clusters by using the same Central installation.



IMPORTANT

When you install Red Hat Advanced Cluster Security for Kubernetes for the first time, you must first install the **Central** custom resource because the **SecuredCluster** custom resource installation is dependent on certificates that Central generates.

Prerequisites

- You must be using OpenShift Container Platform 4.11 or later. For information about supported platforms and architecture, see the [Red Hat Advanced Cluster Security for Kubernetes Support Matrix](#). For life cycle support information for RHACS, see the [Red Hat Advanced Cluster Security for Kubernetes Support Policy](#).

Procedure

1. On the OpenShift Container Platform web console, go to the **Operators → Installed Operators** page.
2. Select the Red Hat Advanced Cluster Security for Kubernetes Operator from the list of installed Operators.
3. If you have installed the Operator in the recommended namespace, OpenShift Container Platform lists the project as **rhacs-operator**. Select **Project: rhacs-operator → Create project**.



WARNING

- If you have installed the Operator in a different namespace, OpenShift Container Platform shows the name of that namespace rather than **rhacs-operator**.
- You must install the Red Hat Advanced Cluster Security for Kubernetes **Central** custom resource in its own project and not in the **rhacs-operator** and **openshift-operator** projects, or in the project in which you have installed the Red Hat Advanced Cluster Security for Kubernetes Operator.

4. Enter the new project name (for example, **stackrox**), and click **Create**. Red Hat recommends that you use **stackrox** as the project name.
5. Under the **Provided APIs** section, select **Central**. Click **Create Central**.
6. Optional: If you are using declarative configuration, next to **Configure via**, click **YAML view** and add the information for the declarative configuration, such as shown in the following example:

```
...
spec:
  central:
    declarativeConfiguration:
      configMaps:
        - name: "<declarative-configs>" 1
      secrets:
        - name: "<sensitive-declarative-configs>" 2
...
```

- 1 Replace <declarative-configs> with the name of the config maps that you are using.
- 2 Replace <sensitive-declarative-configs> with the name of the secrets that you are using.

7. Enter a name for your **Central** custom resource and add any labels you want to apply. Otherwise, accept the default values for the available options.
8. You can configure available options for Central:
 - Central component settings:

Setting	Description
Administrator password	Secret that contains the administrator password. Use this field if you do not want RHACS to generate a password for you.
Exposure	Settings for exposing Central by using a route, load balancer, or node port. See the central.exposure.<parameter> information in the "Public configuration file" section in "Installing Central services for RHACS on Red Hat OpenShift".
User-facing TLS certificate secret	Use this field if you want to terminate TLS in Central and serve a custom server certificate.
Monitoring	Configures the monitoring endpoint for Central. See the central.exposeMonitoring parameter in the "Public configuration file" section in "Installing Central services for RHACS on Red Hat OpenShift".
Persistence	These fields configure how Central should store its persistent data. Use a persistent volume claim (PVC) for best results, especially if you are using Scanner V4. See the central.persistence.<parameter> information in the "Public configuration file" section in "Installing Central services for RHACS on Red Hat OpenShift".
Central DB Settings	Settings for Central DB, including data persistence. See the central.db.<parameter> information in the "Public configuration file" section in "Installing Central services for RHACS on Red Hat OpenShift".
Resources	Use these fields after consulting the documentation if you need to override the default settings for memory and CPU resources. For more information, see the "Default resource requirements for RHACS" and "Recommended resource requirements for RHACS" sections in the "Installation" chapter.
Tolerations	Use this parameter to configure Central to run only on specific nodes. See the central.tolerations parameter in the "Public configuration file" section in "Installing Central services for RHACS on Red Hat OpenShift".

- **Scanner Component Settings:** Settings for the default scanner, also called the StackRox Scanner. See the "Scanner" table in the "Public configuration file" section in "Installing Central services for RHACS on Red Hat OpenShift".
- **Scanner V4 Component Settings:** Settings for the optional Scanner V4 scanner, available in version 4.4 and later. It is not currently enabled by default. You can enable both the StackRox Scanner and Scanner V4 for concurrent use. See the "Scanner V4" table in the "Public configuration file" section in "Installing Central services for RHACS on Red Hat OpenShift".



IMPORTANT

Scanner V4 is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see [Technology Preview Features Support Scope](#).

When Scanner V4 is enabled, you can configure the following options:

Setting	Description
Indexer	The process that indexes images and creates a report of findings. You can configure replicas and autoscaling, resources, and tolerations. Before changing the default resource values, see the "Scanner V4" sections in the "Default resource requirements for RHACS" and "Recommended resource requirements for RHACS" sections in the "Installation" chapter.
Matcher	The process that performs vulnerability matching of the report from the indexer against vulnerability data stored in Scanner V4 DB. You can configure replicas and autoscaling, resources, and tolerations. Before changing the default resource values, see the "Scanner V4" sections in the "Default resource requirements for RHACS" and "Recommended resource requirements for RHACS" sections in the "Installation" chapter.
DB	The database that stores information for Scanner V4, including vulnerability data and index reports. You can configure persistence, resources, and tolerations. If you are using Scanner V4, a persistent volume claim (PVC) is required on Central clusters. A PVC is strongly recommended on secured clusters for best results. Before changing the default resource values, see the "Scanner V4" sections in the "Default resource requirements for RHACS" and "Recommended resource requirements for RHACS" sections in the "Installation" chapter.

- **Egress:** Settings for outgoing network traffic, including whether RHACS should run in online (connected) or offline (disconnected) mode.
- **TLS:** Use this field to add additional trusted root certificate authorities (CAs).
- **Advanced configuration:** You can use these fields to perform the following actions:
 - Specify additional image pull secrets
 - Add custom environment variables to set for managed pods' containers
 - Enable Red Hat OpenShift monitoring

9. Click **Create**.

**NOTE**

If you are using the cluster-wide proxy, Red Hat Advanced Cluster Security for Kubernetes uses that proxy configuration to connect to the external services.

Next Steps

1. Verify Central installation.
2. Optional: Configure Central options.
3. Generate an init bundle containing the cluster secrets that allows communication between the **Central** and **SecuredCluster** resources. You need to download this bundle, use it to generate resources on the clusters you want to secure, and securely store it.
4. Install secured cluster services on each cluster you want to monitor.

Additional resources

- [Default resource requirements for Red Hat Advanced Cluster Security for Kubernetes](#)
- [Recommended resource requirements for Red Hat Advanced Cluster Security for Kubernetes](#)
- [Public configuration file](#)

4.1.1.3. Provisioning a database in your PostgreSQL instance

This step is optional. You can use your existing PostgreSQL infrastructure to provision a database for RHACS. Use the instructions in this section for configuring a PostgreSQL database environment, creating a user, database, schema, role, and granting required permissions.

Procedure

1. Create a new user:

```
CREATE USER stackrox WITH PASSWORD <password>;
```

2. Create a database:

```
CREATE DATABASE stackrox;
```

3. Connect to the database:

```
\connect stackrox
```

4. Create user schema:

```
CREATE SCHEMA stackrox;
```

5. (Optional) Revoke rights on public:

```
REVOKE CREATE ON SCHEMA public FROM PUBLIC;
REVOKE USAGE ON SCHEMA public FROM PUBLIC;
REVOKE ALL ON DATABASE stackrox FROM PUBLIC;
```


6. Create a role:

```
CREATE ROLE readwrite;
```

7. Grant connection permission to the role:

```
GRANT CONNECT ON DATABASE stackrox TO readwrite;
```

8. Add required permissions to the **readwrite** role:

```
GRANT USAGE ON SCHEMA stackrox TO readwrite;
GRANT USAGE, CREATE ON SCHEMA stackrox TO readwrite;
GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA stackrox TO
readwrite;
ALTER DEFAULT PRIVILEGES IN SCHEMA stackrox GRANT SELECT, INSERT, UPDATE,
DELETE ON TABLES TO readwrite;
GRANT USAGE ON ALL SEQUENCES IN SCHEMA stackrox TO readwrite;
ALTER DEFAULT PRIVILEGES IN SCHEMA stackrox GRANT USAGE ON SEQUENCES
TO readwrite;
```

9. Assign the **readwrite** role to the **stackrox** user:

```
GRANT readwrite TO stackrox;
```

4.1.1.4. Installing Central with an external database using the Operator method

The main component of Red Hat Advanced Cluster Security for Kubernetes is called Central. You can install Central on OpenShift Container Platform by using the **Central** custom resource. You deploy Central only once, and you can monitor multiple separate clusters by using the same Central installation.



IMPORTANT

When you install Red Hat Advanced Cluster Security for Kubernetes for the first time, you must first install the **Central** custom resource because the **SecuredCluster** custom resource installation is dependent on certificates that Central generates.

For more information about RHACS databases, see the [Database Scope of Coverage](#).

Prerequisites

- You must be using OpenShift Container Platform 4.11 or later. For more information about supported OpenShift Container Platform versions, see the [Red Hat Advanced Cluster Security for Kubernetes Support Matrix](#).
- You must have a database in your database instance that supports PostgreSQL 13 and a user with the following permissions:
 - Connection rights to the database.
 - **Usage** and **Create** on the schema.
 - **Select**, **Insert**, **Update**, and **Delete** on all tables in the schema.

- **Usage** on all sequences in the schema.

Procedure

1. On the OpenShift Container Platform web console, go to the **Operators → Installed Operators** page.
2. Select the Red Hat Advanced Cluster Security for Kubernetes Operator from the list of installed Operators.
3. If you have installed the Operator in the recommended namespace, OpenShift Container Platform lists the project as **rhacs-operator**. Select **Project: rhacs-operator → Create project**.



WARNING

- If you have installed the Operator in a different namespace, OpenShift Container Platform shows the name of that namespace rather than **rhacs-operator**.
- You must install the Red Hat Advanced Cluster Security for Kubernetes **Central** custom resource in its own project and not in the **rhacs-operator** and **openshift-operator** projects, or in the project in which you have installed the Red Hat Advanced Cluster Security for Kubernetes Operator.

4. Enter the new project name (for example, **stackrox**), and click **Create**. Red Hat recommends that you use **stackrox** as the project name.
5. Create a password secret in the deployed namespace by using the OpenShift Container Platform web console or the terminal.
 - On the OpenShift Container Platform web console, go to the **Workloads → Secrets** page. Create a **Key/Value secret** with the key **password** and the value as the path of a plain text file containing the password for the superuser of the provisioned database.
 - Or, run the following command in your terminal:


```
$ oc create secret generic external-db-password \
  --from-file=password=<password.txt>
```

 - 1 If you use Kubernetes, enter **kubectl** instead of **oc**.
 - 2 Replace **password.txt** with the path of the file which has the plain text password.
6. Return to the Red Hat Advanced Cluster Security for Kubernetes operator page in the OpenShift Container Platform web console. Under the **Provided APIs** section, select **Central**. Click **Create Central**.
7. Optional: If you are using declarative configuration, next to **Configure via**, click **YAML view**.

8. Add the information for the declarative configuration, such as shown in the following example:

```
...
spec:
  central:
    declarativeConfiguration:
      configMaps:
        - name: <declarative-configs> ❶
      secrets:
        - name: <sensitive-declarative-configs> ❷
    ...
```

- ❶ Replace <declarative-configs> with the name of the config maps that you are using.
- ❷ Replace <sensitive-declarative-configs> with the name of the secrets that you are using.

9. Enter a name for your **Central** custom resource and add any labels you want to apply.
10. Go to **Central Component Settings → Central DB Settings**.
11. For **Administrator Password** specify the referenced secret as **external-db-password** (or the secret name of the password created previously).
12. For **Connection String** specify the connection string in **keyword=value** format, for example, **host=<host> port=5432 database=stackrox user=stackrox sslmode=verify-ca**
13. For **Persistence → PersistentVolumeClaim → Claim Name**, remove **central-db**.
14. If necessary, you can specify a Certificate Authority so that there is trust between the database certificate and Central. To add this, go to the YAML view and add a TLS block under the top-level spec, as shown in the following example:

```
spec:
  tls:
    additionalCAs:
      - name: db-ca
        content: |
          <certificate>
```

15. Click **Create**.



NOTE

If you are using the cluster-wide proxy, Red Hat Advanced Cluster Security for Kubernetes uses that proxy configuration to connect to the external services.

Next Steps

1. Verify Central installation.
2. Optional: Configure Central options.

3. Generate an init bundle containing the cluster secrets that allows communication between the **Central** and **SecuredCluster** resources. You need to download this bundle, use it to generate resources on the clusters you want to secure, and securely store it.
4. Install secured cluster services on each cluster you want to monitor.

Additional resources

- [Central configuration options](#)
- [PostgreSQL Connection String Docs](#)

4.1.1.5. Verifying Central installation using the Operator method

After Central finishes installing, log in to the RHACS portal to verify the successful installation of Central.

Procedure

1. On the OpenShift Container Platform web console, go to the **Operators → Installed Operators** page.
2. Select the Red Hat Advanced Cluster Security for Kubernetes Operator from the list of installed Operators.
3. Select the **Central** tab.
4. From the **Centrals** list, select **stackrox-central-services** to view its details.
5. To get the password for the **admin** user, you can either:
 - Click the link under **Admin Password Secret Reference**.
 - Use the Red Hat OpenShift CLI to enter the command listed under **Admin Credentials Info**:

```
$ oc -n stackrox get secret central-htpasswd -o go-template='{{index .data "password" | base64decode}}'
```

6. Find the link to the RHACS portal by using the Red Hat OpenShift CLI command:

```
$ oc -n stackrox get route central -o jsonpath='{.status.ingress[0].host}'
```

Alternatively, you can use the Red Hat Advanced Cluster Security for Kubernetes web console to find the link to the RHACS portal by performing the following commands:

- a. Go to **Networking → Routes**.
 - b. Find the **central** Route and click on the RHACS portal link under the **Location** column.
7. Log in to the RHACS portal using the username **admin** and the password that you retrieved in a previous step. Until RHACS is completely configured (for example, you have the **Central** resource and at least one **SecuredCluster** resource installed and configured), no data is available in the dashboard. The **SecuredCluster** resource can be installed and configured on the same cluster as the **Central** resource. Clusters with the **SecuredCluster** resource are similar to managed clusters in Red Hat Advanced Cluster Management (RHACM).

Next Steps

1. Optional: Configure central settings.
2. Generate an init bundle containing the cluster secrets that allows communication between the **Central** and **SecuredCluster** resources. You need to download this bundle, use it to generate resources on the clusters you want to secure, and securely store it.
3. Install secured cluster services on each cluster you want to monitor.

4.1.2. Install Central using Helm charts

You can install Central using Helm charts without any customization, using the default values, or by using Helm charts with additional customizations of configuration parameters.

4.1.2.1. Install Central using Helm charts without customization

You can install RHACS on your cluster without any customizations. You must add the Helm chart repository and install the **central-services** Helm chart to install the centralized components of Central and Scanner.

4.1.2.1.1. 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:

- Central services Helm chart (**central-services**) for installing the centralized components (Central and Scanner).



NOTE

You deploy centralized components only once and you can monitor multiple separate clusters by using the same installation.

- 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/
```

4.1.2.1.2. Installing the central-services Helm chart without customizations

Use the following instructions to install the **central-services** Helm chart to deploy the centralized components (Central and Scanner).

Prerequisites

- 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](#).

Procedure

- Run the following command to install Central services and expose Central using a route:

```
$ helm install -n stackrox \
  --create-namespace stackrox-central-services rhacs/central-services \
  --set imagePullSecrets.username=<username> \ 1
  --set imagePullSecrets.password=<password> \ 2
  --set central.exposure.route.enabled=true
```

- 1** Include the user name for your pull secret for Red Hat Container Registry authentication.
- 2** Include the password for your pull secret for Red Hat Container Registry authentication.

- Or, run the following command to install Central services and expose Central using a load balancer:

```
$ helm install -n stackrox \
  --create-namespace stackrox-central-services rhacs/central-services \
  --set imagePullSecrets.username=<username> \ 1
  --set imagePullSecrets.password=<password> \ 2
  --set central.exposure.loadBalancer.enabled=true
```

- 1** Include the user name for your pull secret for Red Hat Container Registry authentication.
- 2** Include the password for your pull secret for Red Hat Container Registry authentication.

- Or, run the following command to install Central services and expose Central using port forward:

```
$ helm install -n stackrox \
  --create-namespace stackrox-central-services rhacs/central-services \
  --set imagePullSecrets.username=<username> \ 1
  --set imagePullSecrets.password=<password> \ 2
```

- 1** Include the user name for your pull secret for Red Hat Container Registry authentication.
- 2** Include the password for your pull secret for Red Hat Container Registry authentication.

IMPORTANT

- If you are installing Red Hat Advanced Cluster Security for Kubernetes in a cluster that requires a proxy to connect to external services, you must specify your proxy configuration by using the **proxyConfig** parameter. For example:

```
env:
  proxyConfig: |
    url: http://proxy.name:port
    username: username
    password: password
    excludes:
      - some.domain
```

- If you already created one or more image pull secrets in the namespace in which you are installing, instead of using a username and password, you can use **--set imagePullSecrets.useExisting="<pull-secret-1;pull-secret-2>"**.
- Do not use image pull secrets:
 - If you are pulling your images from **quay.io/stackrox-io** or a registry in a private network that does not require authentication. Use **--set imagePullSecrets.allowNone=true** instead of specifying a username and password.
 - If you already configured image pull secrets in the default service account in the namespace you are installing. Use **--set imagePullSecrets.useFromDefaultServiceAccount=true** instead of specifying a username and password.

The output of the installation command includes:

- An automatically generated administrator password.
- Instructions on storing all the configuration values.
- Any warnings that Helm generates.

4.1.2.2. Install Central using Helm charts with customizations

You can install RHACS on your Red Hat OpenShift cluster with customizations by using Helm chart configuration parameters 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.
- Configuration file **declarative-config-values.yaml**: Create this file if you are using declarative configuration to add the declarative configuration mounts to Central.

4.1.2.2.1. Private configuration file

This section lists the configurable parameters of the **values-private.yaml** file. There are no default values for these parameters.

4.1.2.2.1.1. Image pull secrets

The credentials that are required for pulling images from the registry depend on the following factors:

- If you are using a custom registry, you must specify these parameters:
 - **imagePullSecrets.username**
 - **imagePullSecrets.password**
 - **image.registry**
- If you do not use a username and password to log in to the custom registry, you must specify one of the following parameters:
 - **imagePullSecrets.allowNone**
 - **imagePullSecrets.useExisting**
 - **imagePullSecrets.useFromDefaultServiceAccount**

Parameter	Description
imagePullSecrets.username	The username of the account that is used to log in to the registry.
imagePullSecrets.password	The password of the account that is used to log in to the registry.
imagePullSecrets.allowNone	Use true if you are using a custom registry and it allows pulling images without credentials.
imagePullSecrets.useExisting	A comma-separated list of secrets as values. For example, secret1, secret2, secretN . Use this option if you have already created pre-existing image pull secrets with the given name in the target namespace.
imagePullSecrets.useFromDefaultServiceAccount	Use true if you have already configured the default service account in the target namespace with sufficiently scoped image pull secrets.

4.1.2.2.1.2. Proxy configuration

If you are installing Red Hat Advanced Cluster Security for Kubernetes in a cluster that requires a proxy to connect to external services, you must specify your proxy configuration by using the **proxyConfig** parameter. For example:

```
env:
  proxyConfig: |
```



```
url: http://proxy.name:port
username: username
password: password
excludes:
- some.domain
```

Parameter	Description
env.proxyConfig	Your proxy configuration.

4.1.2.2.1.3. Central

Configurable parameters for Central.

For a new installation, you can skip the following parameters:

- **central.jwtSigner.key**
- **central.serviceTLS.cert**
- **central.serviceTLS.key**
- **central.adminPassword.value**
- **central.adminPassword.htpasswd**
- **central.db.serviceTLS.cert**
- **central.db.serviceTLS.key**
- **central.db.password.value**
- When you do not specify values for these parameters the Helm chart autogenerates values for them.
- If you want to modify these values you can use the **helm upgrade** command and specify the values using the **--set** option.

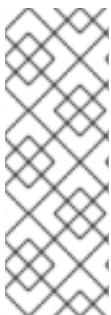


IMPORTANT

For setting the administrator password, you can only use either **central.adminPassword.value** or **central.adminPassword.htpasswd**, but not both.

Parameter	Description
central.jwtSigner.key	A private key which RHACS should use for signing JSON web tokens (JWTs) for authentication.
central.serviceTLS.cert	An internal certificate that the Central service should use for deploying Central.

Parameter	Description
central.serviceTLS.key	The private key of the internal certificate that the Central service should use.
central.defaultTLS.cert	<p>The user-facing certificate that Central should use. RHACS uses this certificate for RHACS portal.</p> <ul style="list-style-type: none"> For a new installation, you must provide a certificate, otherwise, RHACS installs Central by using a self-signed certificate. If you are upgrading, RHACS uses the existing certificate and its key.
central.defaultTLS.key	<p>The private key of the user-facing certificate that Central should use.</p> <ul style="list-style-type: none"> For a new installation, you must provide the private key, otherwise, RHACS installs Central by using a self-signed certificate. If you are upgrading, RHACS uses the existing certificate and its key.
central.db.password.value	Connection password for Central database.
central.adminPassword.value	Administrator password for logging into RHACS.
central.adminPassword.htpasswd	Administrator password for logging into RHACS. This password is stored in hashed format using bcrypt.
central.db.serviceTLS.cert	An internal certificate that the Central DB service should use for deploying Central DB.
central.db.serviceTLS.key	The private key of the internal certificate that the Central DB service should use.
central.db.password.value	The password used to connect to the Central DB.



NOTE

If you are using **central.adminPassword.htpasswd** parameter, you must use a bcrypt encoded password hash. You can run the command **htpasswd -nB admin** to generate a password hash. For example,

```
htpasswd: |
admin:<bcrypt-hash>
```

4.1.2.2.1.4. Scanner

Configurable parameters for the StackRox Scanner and Scanner V4 (Technology Preview).



IMPORTANT

Scanner V4 is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see [Technology Preview Features Support Scope](#).

For a new installation, you can skip the following parameters and the Helm chart autogenerated values for them. Otherwise, if you are upgrading to a new version, specify the values for the following parameters:

- **scanner.dbPassword.value**
- **scanner.serviceTLS.cert**
- **scanner.serviceTLS.key**
- **scanner.dbServiceTLS.cert**
- **scanner.dbServiceTLS.key**
- **scannerV4.db.password.value**
- **scannerV4.indexer.serviceTLS.cert**
- **scannerV4.indexer.serviceTLS.key**
- **scannerV4.matcher.serviceTLS.cert**
- **scannerV4.matcher.serviceTLS.key**
- **scannerV4.db.serviceTLS.cert**
- **scannerV4.db.serviceTLS.key**

Parameter	Description
scanner.dbPassword.value	The password to use for authentication with Scanner database. Do not modify this parameter because RHACS automatically creates and uses its value internally.
scanner.serviceTLS.cert	An internal certificate that the StackRox Scanner service should use for deploying the StackRox Scanner.
scanner.serviceTLS.key	The private key of the internal certificate that the Scanner service should use.
scanner.dbServiceTLS.cert	An internal certificate that the Scanner-db service should use for deploying Scanner database.

Parameter	Description
scanner.dbServiceTLS.key	The private key of the internal certificate that the Scanner-db service should use.
scannerV4.db.password.value	The password to use for authentication with the Scanner V4 database. Do not modify this parameter because RHACS automatically creates and uses its value internally.
scannerV4.db.serviceTLS.cert	An internal certificate that the Scanner V4 DB service should use for deploying the Scanner V4 database.
scannerV4.db.serviceTLS.key	The private key of the internal certificate that the Scanner V4 DB service should use.
scannerV4.indexer.serviceTLS.cert	An internal certificate that the Scanner V4 service should use for deploying the Scanner V4 Indexer.
scannerV4.indexer.serviceTLS.key	The private key of the internal certificate that the Scanner V4 Indexer should use.
scannerV4.matcher.serviceTLS.cert	An internal certificate that the Scanner V4 service should use for deploying the the Scanner V4 Matcher.
scannerV4.matcher.serviceTLS.key	The private key of the internal certificate that the Scanner V4 Matcher should use.

4.1.2.2.2. Public configuration file

This section lists the configurable parameters of the **values-public.yaml** file.

4.1.2.2.2.1. Image pull secrets

Image pull secrets are the credentials required for pulling images from your registry.

Parameter	Description
imagePullSecrets.allowNone	Use true if you are using a custom registry and it allows pulling images without credentials.
imagePullSecrets.useExisting	A comma-separated list of secrets as values. For example, secret1, secret2 . Use this option if you have already created pre-existing image pull secrets with the given name in the target namespace.

Parameter	Description
imagePullSecrets.useFromDefaultServiceAccount	Use true if you have already configured the default service account in the target namespace with sufficiently scoped image pull secrets.

4.1.2.2.2. Image

Image declares the configuration to set up the main registry, which the Helm chart uses to resolve images for the **central.image**, **scanner.image**, **scanner.dbImage**, **scannerV4.image**, and **scannerV4.db.image** parameters.

Parameter	Description
image.registry	Address of your image registry. Either use a hostname, such as registry.redhat.io , or a remote registry hostname, such as us.gcr.io/stackrox-mirror .

4.1.2.2.3. Environment variables

Red Hat Advanced Cluster Security for Kubernetes automatically detects your cluster environment and sets values for **env.openshift**, **env.istio**, and **env.platform**. Only set these values to override the automatic cluster environment detection.

Parameter	Description
env.openshift	Use true for installing on an OpenShift Container Platform cluster and overriding automatic cluster environment detection.
env.istio	Use true for installing on an Istio enabled cluster and overriding automatic cluster environment detection.
env.platform	The platform on which you are installing RHACS. Set its value to default or gke to specify cluster platform and override automatic cluster environment detection.
env.offlineMode	Use true to use RHACS in offline mode.

4.1.2.2.4. Additional trusted certificate authorities

The RHACS automatically references the system root certificates to trust. When Central, the StackRox Scanner, or Scanner V4 must reach out to services that use certificates issued by an authority in your organization or a globally trusted partner organization, you can add trust for these services by specifying the root certificate authority to trust by using the following parameter:

Parameter	Description
additionalCAs.<certificate_name>	Specify the PEM encoded certificate of the root certificate authority to trust.

4.1.2.2.5. Central

Configurable parameters for Central.

- You must specify a persistent storage option as either **hostPath** or **persistentVolumeClaim**.
- For exposing Central deployment for external access. You must specify one parameter, either **central.exposure.loadBalancer**, **central.exposure.nodePort**, or **central.exposure.route**. When you do not specify any value for these parameters, you must manually expose Central or access it by using port-forwarding.

The following table includes settings for an external PostgreSQL database.

Parameter	Description
central.declarativeConfiguration.mounts.configMaps	Mounts config maps used for declarative configurations.
Central.declarativeConfiguration.mounts.secrets	Mounts secrets used for declarative configurations.
central.endpointsConfig	The endpoint configuration options for Central.
central.nodeSelector	If the node selector selects tainted nodes, use this parameter to specify a taint toleration key, value, and effect for Central. This parameter is mainly used for infrastructure nodes.
central.tolerations	If the node selector selects tainted nodes, use this parameter to specify a taint toleration key, value, and effect for Central. This parameter is mainly used for infrastructure nodes.
central.exposeMonitoring	Specify true to expose Prometheus metrics endpoint for Central on port number 9090 .
central.image.registry	A custom registry that overrides the global image.registry parameter for the Central image.
central.image.name	The custom image name that overrides the default Central image name (main).

Parameter	Description
central.image.tag	The custom image tag that overrides the default tag for Central image. If you specify your own image tag during a new installation, you must manually increment this tag when you to upgrade to a new version by running the helm upgrade command. If you mirror Central images in your own registry, do not modify the original image tags.
central.image.fullRef	Full reference including registry address, image name, and image tag for the Central image. Setting a value for this parameter overrides the central.image.registry , central.image.name , and central.image.tag parameters.
central.resources.requests.memory	The memory request for Central.
central.resources.requests.cpu	The CPU request for Central.
central.resources.limits.memory	The memory limit for Central.
central.resources.limits.cpu	The CPU limit for Central.
central.persistence.hostPath	The path on the node where RHACS should create a database volume. Red Hat does not recommend using this option.
central.persistence.persistentVolumeClaim.claimName	The name of the persistent volume claim (PVC) you are using.
central.persistence.persistentVolumeClaim.createClaim	Use true to create a new PVC, or false to use an existing claim.
central.persistence.persistentVolumeClaim.size	The size (in GiB) of the persistent volume managed by the specified claim.
central.exposure.loadBalancer.enabled	Use true to expose Central by using a load balancer.
central.exposure.loadBalancer.port	The port number on which to expose Central. The default port number is 443.
central.exposure.nodePort.enabled	Use true to expose Central by using the node port service.

Parameter	Description
central.exposure.nodePort.port	The port number on which to expose Central. When you skip this parameter, OpenShift Container Platform automatically assigns a port number. Red Hat recommends that you do not specify a port number if you are exposing RHACS by using a node port.
central.exposure.route.enabled	Use true to expose Central by using a route. This parameter is only available for OpenShift Container Platform clusters.
central.db.external	Use true to specify that Central DB should not be deployed and that an external database will be used.
central.db.source.connectionString	<p>The connection string for Central to use to connect to the database. This is only used when central.db.external is set to true. The connection string must be in keyword/value format as described in the PostgreSQL documentation in "Additional resources".</p> <ul style="list-style-type: none"> ● Only PostgreSQL 13 is supported. ● Connections through PgBouncer are not supported. ● User must be superuser with ability to create and delete databases.
central.db.source.minConns	The minimum number of connections to the database to be established.
central.db.source.maxConns	The maximum number of connections to the database to be established.
central.db.source.statementTimeoutMs	The number of milliseconds a single query or transaction can be active against the database.
central.db.postgresConfig	The postgresql.conf to be used for Central DB as described in the PostgreSQL documentation in "Additional resources".
central.db.hbaConfig	The pg_hba.conf to be used for Central DB as described in the PostgreSQL documentation in "Additional resources".

Parameter	Description
central.db.nodeSelector	Specify a node selector label as label-key: label-value to force Central DB to only schedule on nodes with the specified label.
central.db.image.registry	A custom registry that overrides the global image.registry parameter for the Central DB image.
central.db.image.name	The custom image name that overrides the default Central DB image name (central-db).
central.db.image.tag	The custom image tag that overrides the default tag for Central DB image. If you specify your own image tag during a new installation, you must manually increment this tag when you to upgrade to a new version by running the helm upgrade command. If you mirror Central DB images in your own registry, do not modify the original image tags.
central.db.image.fullRef	Full reference including registry address, image name, and image tag for the Central DB image. Setting a value for this parameter overrides the central.db.image.registry , central.db.image.name , and central.db.image.tag parameters.
central.db.resources.requests.memory	The memory request for Central DB.
central.db.resources.requests.cpu	The CPU request for Central DB.
central.db.resources.limits.memory	The memory limit for Central DB.
central.db.resources.limits.cpu	The CPU limit for Central DB.
central.db.persistence.hostPath	The path on the node where RHACS should create a database volume. Red Hat does not recommend using this option.
central.db.persistence.persistentVolumeClaim.claimName	The name of the persistent volume claim (PVC) you are using.
central.db.persistence.persistentVolumeClaim.createClaim	Use true to create a new persistent volume claim, or false to use an existing claim.
central.db.persistence.persistentVolumeClaim.size	The size (in GiB) of the persistent volume managed by the specified claim.

4.1.2.2.6. StackRox Scanner

The following table lists the configurable parameters for the StackRox Scanner. This is the scanner used for node and platform scanning. If Scanner V4 is not enabled, the StackRox scanner also performs image scanning. Beginning with version 4.4, Scanner V4 can be enabled to provide image scanning. See the next table for Scanner V4 parameters.

Parameter	Description
scanner.disable	Use true to install RHACS without the StackRox Scanner. When you use it with the helm upgrade command, Helm removes the existing StackRox Scanner deployment.
scanner.exposeMonitoring	Specify true to expose Prometheus metrics endpoint for the StackRox Scanner on port number 9090 .
scanner.replicas	The number of replicas to create for the StackRox Scanner deployment. When you use it with the scanner.autoscaling parameter, this value sets the initial number of replicas.
scanner.logLevel	Configure the log level for the StackRox Scanner. Red Hat recommends that you not change the default log level value (INFO).
scanner.nodeSelector	Specify a node selector label as label-key: label-value to force the StackRox 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 the StackRox Scanner. This parameter is mainly used for infrastructure nodes.
scanner.autoscaling.disable	Use true to disable autoscaling for the StackRox Scanner deployment. When you disable autoscaling, the minReplicas and maxReplicas parameters do not have any effect.
scanner.autoscaling.minReplicas	The minimum number of replicas for autoscaling.
scanner.autoscaling.maxReplicas	The maximum number of replicas for autoscaling.
scanner.resources.requests.memory	The memory request for the StackRox Scanner.
scanner.resources.requests.cpu	The CPU request for the StackRox Scanner.
scanner.resources.limits.memory	The memory limit for the StackRox Scanner.

Parameter	Description
scanner.resources.limits.cpu	The CPU limit for the StackRox Scanner.
scanner.dbResources.requests.memory	The memory request for the StackRox Scanner database deployment.
scanner.dbResources.requests.cpu	The CPU request for the StackRox Scanner database deployment.
scanner.dbResources.limits.memory	The memory limit for the StackRox Scanner database deployment.
scanner.dbResources.limits.cpu	The CPU limit for the StackRox Scanner database deployment.
scanner.image.registry	A custom registry for the StackRox Scanner image.
scanner.image.name	The custom image name that overrides the default StackRox Scanner image name (scanner).
scanner.dbImage.registry	A custom registry for the StackRox Scanner DB image.
scanner.dbImage.name	The custom image name that overrides the default StackRox Scanner DB image name (scanner-db).
scanner.dbNodeSelector	Specify a node selector label as label-key: label-value to force the StackRox 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 the StackRox Scanner DB. This parameter is mainly used for infrastructure nodes.

4.1.2.2.7. Scanner V4

The following table lists the configurable parameters for Scanner V4.



IMPORTANT

Scanner V4 is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see [Technology Preview Features Support Scope](#).

Parameter	Description
scannerV4.db.persistence.persistentVolumeClaim.claimName	The name of the PVC to manage persistent data for Scanner V4. If no PVC with the given name exists, it is created. The default value is scanner-v4-db if not set. To prevent data loss, the PVC is not removed automatically when Central is deleted.
scannerV4.disable	Use false to enable Scanner V4. When setting this parameter, the StackRox Scanner must also be enabled by setting scanner.disable=false . Until feature parity between the StackRox Scanner and Scanner V4 is reached, Scanner V4 can only be used in combination with the StackRox Scanner. Enabling Scanner V4 without also enabling the StackRox Scanner is not supported. When you set this parameter to true with the helm upgrade command, Helm removes the existing Scanner V4 deployment.
scannerV4.exposeMonitoring	Specify true to expose Prometheus metrics endpoint for Scanner V4 on port number 9090 .
scannerV4.indexer.replicas	The number of replicas to create for the Scanner V4 Indexer deployment. When you use it with the scannerV4.indexer.autoscaling parameter, this value sets the initial number of replicas.
scannerV4.indexer.logLevel	Configure the log level for the Scanner V4 Indexer. Red Hat recommends that you not change the default log level value (INFO).
scannerV4.indexer.nodeSelector	Specify a node selector label as label-key: label-value to force the Scanner V4 Indexer to only schedule on nodes with the specified label.
scannerV4.indexer.tolerations	If the node selector selects tainted nodes, use this parameter to specify a taint toleration key, value, and effect for the Scanner V4 Indexer. This parameter is mainly used for infrastructure nodes.
scannerV4.indexer.autoscaling.disable	Use true to disable autoscaling for the Scanner V4 Indexer deployment. When you disable autoscaling, the minReplicas and maxReplicas parameters do not have any effect.
scannerV4.indexer.autoscaling.minReplicas	The minimum number of replicas for autoscaling.
scannerV4.indexer.autoscaling.maxReplicas	The maximum number of replicas for autoscaling.

Parameter	Description
scannerV4.indexer.resources.requests.memory	The memory request for the Scanner V4 Indexer.
scannerV4.indexer.resources.requests.cpu	The CPU request for the Scanner V4 Indexer.
scannerV4.indexer.resources.limits.memory	The memory limit for the Scanner V4 Indexer.
scannerV4.indexer.resources.limits.cpu	The CPU limit for the Scanner V4 Indexer.
scannerV4.matcher.replicas	The number of replicas to create for the Scanner V4 Matcher deployment. When you use it with the scannerV4.matcher.autoscaling parameter, this value sets the initial number of replicas.
scannerV4.matcher.logLevel	Red Hat recommends that you not change the default log level value (INFO).
scannerV4.matcher.nodeSelector	Specify a node selector label as label-key: label-value to force the Scanner V4 Matcher to only schedule on nodes with the specified label.
scannerV4.matcher.tolerations	If the node selector selects tainted nodes, use this parameter to specify a taint toleration key, value, and effect for the Scanner V4 Matcher. This parameter is mainly used for infrastructure nodes.
scannerV4.matcher.autoscaling.disable	Use true to disable autoscaling for the Scanner V4 Matcher deployment. When you disable autoscaling, the minReplicas and maxReplicas parameters do not have any effect.
scannerV4.matcher.autoscaling.minReplicas	The minimum number of replicas for autoscaling.
scannerV4.matcher.autoscaling.maxReplicas	The maximum number of replicas for autoscaling.
scannerV4.matcher.resources.requests.memory	The memory request for the Scanner V4 Matcher.
scannerV4.matcher.resources.requests.cpu	The CPU request for the Scanner V4 Matcher.
scannerV4.db.resources.requests.memory	The memory request for the Scanner V4 database deployment.
scannerV4.db.resources.requests.cpu	The CPU request for the Scanner V4 database deployment.

Parameter	Description
scannerV4.db.resources.limits.memory	The memory limit for the Scanner V4 database deployment.
scannerV4.db.resources.limits.cpu	The CPU limit for the Scanner V4 database deployment.
scannerV4.db.nodeSelector	Specify a node selector label as label-key: label-value to force the Scanner V4 DB to only schedule on nodes with the specified label.
scannerV4.db.tolerations	If the node selector selects tainted nodes, use this parameter to specify a taint toleration key, value, and effect for the Scanner V4 DB. This parameter is mainly used for infrastructure nodes.
scannerV4.db.image.registry	A custom registry for the Scanner V4 DB image.
scannerV4.db.image.name	The custom image name that overrides the default Scanner V4 DB image name (scanner-v4-db).
scannerV4.image.registry	A custom registry for the Scanner V4 image.
scannerV4.image.name	The custom image name that overrides the default Scanner V4 image name (scanner-v4).

4.1.2.2.2.8. Customization

Use these parameters to specify additional attributes for all objects that RHACS creates.

Parameter	Description
customize.labels	A custom label to attach to all objects.
customize.annotations	A custom annotation to attach to all objects.
customize.podLabels	A custom label to attach to all deployments.
customize.podAnnotations	A custom annotation to attach to all deployments.
customize.envVars	A custom environment variable for all containers in all objects.
customize.central.labels	A custom label to attach to all objects that Central creates.

Parameter	Description
customize.central.annotations	A custom annotation to attach to all objects that Central creates.
customize.central.podLabels	A custom label to attach to all Central deployments.
customize.central.podAnnotations	A custom annotation to attach to all Central deployments.
customize.central.envVars	A custom environment variable for all Central containers.
customize.scanner.labels	A custom label to attach to all objects that Scanner creates.
customize.scanner.annotations	A custom annotation to attach to all objects that Scanner creates.
customize.scanner.podLabels	A custom label to attach to all Scanner deployments.
customize.scanner.podAnnotations	A custom annotation to attach to all Scanner deployments.
customize.scanner.envVars	A custom environment variable for all Scanner containers.
customize.scanner-db.labels	A custom label to attach to all objects that Scanner DB creates.
customize.scanner-db.annotations	A custom annotation to attach to all objects that Scanner DB creates.
customize.scanner-db.podLabels	A custom label to attach to all Scanner DB deployments.
customize.scanner-db.podAnnotations	A custom annotation to attach to all Scanner DB deployments.
customize.scanner-db.envVars	A custom environment variable for all Scanner DB containers.
customize.scanner-v4-indexer.labels	A custom label to attach to all objects that Scanner V4 Indexer creates and into the pods belonging to them.

Parameter	Description
customize.scanner-v4-indexer.annotations	A custom annotation to attach to all objects that Scanner V4 Indexer creates and into the pods belonging to them.
customize.scanner-v4-indexer.podLabels	A custom label to attach to all objects that Scanner V4 Indexer creates and into the pods belonging to them.
customize.scanner-v4-indexer.podAnnotations	A custom annotation to attach to all objects that Scanner V4 Indexer creates and into the pods belonging to them.
customize.scanner-4v-indexer.envVars	A custom environment variable for all Scanner V4 Indexer containers and the pods belonging to them.
customize.scanner-v4-matcher.labels	A custom label to attach to all objects that Scanner V4 Matcher creates and into the pods belonging to them.
customize.scanner-v4-matcher.annotations	A custom annotation to attach to all objects that Scanner V4 Matcher creates and into the pods belonging to them.
customize.scanner-v4-matcher.podLabels	A custom label to attach to all objects that Scanner V4 Matcher creates and into the pods belonging to them.
customize.scanner-v4-matcher.podAnnotations	A custom annotation to attach to all objects that Scanner V4 Matcher creates and into the pods belonging to them.
customize.scanner-4v-matcher.envVars	A custom environment variable for all Scanner V4 Matcher containers and the pods belonging to them.
customize.scanner-v4-db.labels	A custom label to attach to all objects that Scanner V4 DB creates and into the pods belonging to them.
customize.scanner-v4-db.annotations	A custom annotation to attach to all objects that Scanner V4 DB creates and into the pods belonging to them.
customize.scanner-v4-db.podLabels	A custom label to attach to all objects that Scanner V4 DB creates and into the pods belonging to them.
customize.scanner-v4-db.podAnnotations	A custom annotation to attach to all objects that Scanner V4 DB creates and into the pods belonging to them.

Parameter	Description
customize.scanner-4v-db.envVars	A custom environment variable for all Scanner V4 DB containers and the pods belonging to them.

You can also use:

- the **customize.other.service/*.labels** and the **customize.other.service/*.annotations** parameters, to specify labels and annotations for all objects.
- or, provide a specific service name, for example, **customize.other.service/central-loadbalancer.labels** and **customize.other.service/central-loadbalancer.annotations** as parameters and set their value.

4.1.2.2.9. Advanced customization



IMPORTANT

The parameters specified in this section are for information only. Red Hat does not support RHACS instances with modified namespace and release names.

Parameter	Description
allowNonstandardNamespace	Use true to deploy RHACS into a namespace other than the default namespace stackrox .
allowNonstandardReleaseName	Use true to deploy RHACS with a release name other than the default stackrox-central-services .

4.1.2.2.3. Declarative configuration values

To use declarative configuration, you must create a YAML file (in this example, named "declarative-config-values.yaml") that adds the declarative configuration mounts to Central. This file is used in a Helm installation.

Procedure

1. Create the YAML file (in this example, named **declarative-config-values.yaml**) using the following example as a guideline:

```
central:
  declarativeConfiguration:
    mounts:
      configMaps:
        - declarative-configs
      secrets:
        - sensitive-declarative-configs
```

2. Install the Central services Helm chart as documented in the "Installing the central-services Helm chart", referencing the **declarative-config-values.yaml** file.

4.1.2.2.4. Installing the central-services Helm chart

After you configure the **values-public.yaml** and **values-private.yaml** files, install the **central-services** Helm chart to deploy the centralized components (Central and Scanner).

Procedure

- Run the following command:

```
$ helm install -n stackrox --create-namespace \
  stackrox-central-services rhacs/central-services \
  -f <path_to_values_public.yaml> -f <path_to_values_private.yaml> 1
```

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



NOTE

Optional: If using declarative configuration, add **-f <path_to_declarative-config-values.yaml>** to this command to mount the declarative configurations file in Central.

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

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

When using the **helm upgrade** command to make changes, the following guidelines and requirements apply:

- You can also specify configuration values using the **--set** or **--set-file** parameters. However, these options are not saved, and you must manually specify all the options again whenever you make changes.
- Some changes, such as enabling a new component like Scanner V4, require new certificates to be issued for the component. Therefore, you must provide a CA when making these changes.



IMPORTANT

Scanner V4 is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see [Technology Preview Features Support Scope](#).

- If the CA was generated by the Helm chart during the initial installation, you must retrieve these automatically generated values from the cluster and provide them to the **helm upgrade** command. The post-installation notes of the **central-services** Helm chart include a command for retrieving the automatically generated values.
- If the CA was generated outside of the Helm chart and provided during the installation of the **central-services** chart, then you must perform that action again when using the **helm**

upgrade command, for example, by using the **--reuse-values** flag with the **helm upgrade** command.

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-central-services rhacs/central-services \
  --reuse-values \
  -f <path_to_init_bundle_file \
  -f <path_to_values_public.yaml> \
  -f <path_to_values_private.yaml>
```

- 1 If you have modified values that are not included in the **values_public.yaml** and **values_private.yaml** files, include the **--reuse-values** parameter.

4.1.3. Install Central using the roxctl CLI



WARNING

For production environments, Red Hat recommends using the Operator or Helm charts to install RHACS. Do not use the **roxctl** install method unless you have a specific installation need that requires using this method.

4.1.3.1. Installing the roxctl CLI

To install Red Hat Advanced Cluster Security for Kubernetes you must install the **roxctl** CLI by downloading the binary. You can install **roxctl** on Linux, Windows, or macOS.

4.1.3.1.1. 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/4.4.3/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
```

4.1.3.1.2. 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/4.4.3/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
```

4.1.3.1.3. 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/4.4.3/bin/Windows/roxctl.exe
```

Verification

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

```
$ roxctl version
```

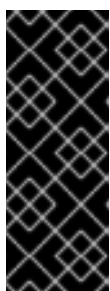
4.1.3.2. Using the interactive installer

Use the interactive installer to generate the required secrets, deployment configurations, and deployment scripts for your environment.

Procedure

1. Run the interactive install command:

```
$ roxctl central generate interactive
```



IMPORTANT

Installing RHACS using the **roxctl** CLI creates PodSecurityPolicy (PSP) objects by default for backward compatibility. If you install RHACS on Kubernetes versions 1.25 and newer or OpenShift Container Platform version 4.12 and newer, you must disable the PSP object creation. To do this, specify **--enable-pod-security-policies** option as **false** for the **roxctl central generate** and **roxctl sensor generate** commands.

2. Press **Enter** to accept the default value for a prompt or enter custom values as required. The following example shows the interactive installer prompts:

```
Enter path to the backup bundle from which to restore keys and certificates (optional):
Enter read templates from local filesystem (default: "false"):
Enter path to helm templates on your local filesystem (default: "/path"):
Enter PEM cert bundle file (optional): 1
Enter Create PodSecurityPolicy resources (for pre-v1.25 Kubernetes) (default: "true"): 2
Enter administrator password (default: autogenerated):
Enter orchestrator (k8s, openshift):
Enter default container images settings (development_build, stackrox.io, rhacs, opensource);
it controls repositories from where to download the images, image names and tags format
```

(default: "development_build"):
 Enter the directory to output the deployment bundle to (default: "central-bundle"):
 Enter the OpenShift major version (3 or 4) to deploy on (default: "0"):
 Enter whether to enable telemetry (default: "false"):
 Enter central-db image to use (if unset, a default will be used according to --image-defaults):
 Enter Istio version when deploying into an Istio-enabled cluster (leave empty when not running Istio) (optional):
 Enter the method of exposing Central (route, lb, np, none) (default: "none"): **3**
 Enter main image to use (if unset, a default will be used according to --image-defaults):
 Enter whether to run StackRox in offline mode, which avoids reaching out to the Internet (default: "false"):
 Enter list of secrets to add as declarative configuration mounts in central (default: "[]"): **4**
 Enter list of config maps to add as declarative configuration mounts in central (default: "[]"): **5**
 Enter the deployment tool to use (kubectl, helm, helm-values) (default: "kubectl"):
 Enter scanner-db image to use (if unset, a default will be used according to --image-defaults):
 Enter scanner image to use (if unset, a default will be used according to --image-defaults):
 Enter Central volume type (hostpath, pvc): **6**
 Enter external volume name for Central (default: "stackrox-db"):
 Enter external volume size in Gi for Central (default: "100"):
 Enter storage class name for Central (optional if you have a default StorageClass configured):
 Enter external volume name for Central DB (default: "central-db"):
 Enter external volume size in Gi for Central DB (default: "100"):
 Enter storage class name for Central DB (optional if you have a default StorageClass configured):

- 1** If you want to add a custom TLS certificate, provide the file path for the PEM-encoded certificate. When you specify a custom certificate the interactive installer also prompts you to provide a PEM private key for the custom certificate you are using.
- 2** If you are running Kubernetes version 1.25 or later, set this value to **false**.
- 3** To use the RHACS portal, you must expose Central by using a route, a load balancer or a node port.
- 4** For more information on using declarative configurations for authentication and authorization, see "Declarative configuration for authentication and authorization resources" in "Managing RBAC in Red Hat Advanced Cluster Security for Kubernetes".
- 5** For more information on using declarative configurations for authentication and authorization, see "Declarative configuration for authentication and authorization resources" in "Managing RBAC in Red Hat Advanced Cluster Security for Kubernetes".
- 6** If you plan to install Red Hat Advanced Cluster Security for Kubernetes on OpenShift Container Platform with a hostPath volume, you must modify the SELinux policy.



WARNING

On OpenShift Container Platform, for using a `hostPath` volume, you must modify the SELinux policy to allow access to the directory, which the host and the container share. It is because SELinux blocks directory sharing by default. To modify the SELinux policy, run the following command:

```
$ sudo chcon -Rt svirt_sandbox_file_t <full_volume_path>
```

However, Red Hat does not recommend modifying the SELinux policy, instead use PVC when installing on OpenShift Container Platform.

On completion, the installer creates a folder named `central-bundle`, which contains the necessary YAML manifests and scripts to deploy Central. In addition, it shows on-screen instructions for the scripts you need to run to deploy additional trusted certificate authorities, Central and Scanner, and the authentication instructions for logging into the RHACS portal along with the autogenerated password if you did not provide one when answering the prompts.

4.1.3.3. Running the Central installation scripts

After you run the interactive installer, you can run the **setup.sh** script to install Central.

Procedure

1. Run the **setup.sh** script to configure image registry access:

```
$ ./central-bundle/central/scripts/setup.sh
```

2. Create the necessary resources:

```
$ oc create -R -f central-bundle/central
```

3. Check the deployment progress:

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

4. After Central is running, find the RHACS portal IP address and open it in your browser. Depending on the exposure method you selected when answering the prompts, use one of the following methods to get the IP address.

Exposure method	Command	Address	Example
Route	oc -n stackrox get route central	The address under the HOST/PORT column in the output	https://central-stackrox.example.route

Exposure method	Command	Address	Example
Node Port	oc get node -owide && oc -n stackrox get svc central-loadbalancer	IP or hostname of any node, on the port shown for the service	https://198.51.100.0:31489
Load Balancer	oc -n stackrox get svc central-loadbalancer	EXTERNAL-IP or hostname shown for the service, on port 443	https://192.0.2.0
None	central-bundle/central/scripts/port-forward.sh 8443	https://localhost:8443	https://localhost:8443



NOTE

If you have selected autogenerated password during the interactive install, you can run the following command to see it for logging into Central:

```
$ cat central-bundle/password
```

4.2. CONFIGURING CENTRAL CONFIGURATION OPTIONS FOR RHACS USING THE OPERATOR

When installing the Central instance by using the Operator, you can configure optional settings.

4.2.1. Central configuration options using the Operator

When you create a Central instance, the Operator lists the following configuration options for the **Central** custom resource.

The following table includes settings for an external PostgreSQL database.

4.2.1.1. Central settings

Parameter	Description
central.adminPasswordSecret	Specify a secret that contains the administrator password in the password data item. If omitted, the operator autogenerated a password and stores it in the password item in the central-htpasswd secret.
central.defaultTLSCert	By default, Central only serves an internal TLS certificate, which means that you need to handle TLS termination at the ingress or load balancer level. If you want to terminate TLS in Central and serve a custom server certificate, you can specify a secret containing the certificate and private key.

Parameter	Description
central.adminPasswordGenerationDisabled	Set this parameter to true to disable the automatic administrator password generation. Use this only after you perform the first-time setup of alternative authentication methods. Do not use this for initial installation. Otherwise, you must reinstall the custom resource to log back in.
central.tolerations	If the node selector selects tainted nodes, use this parameter to specify a taint toleration key, value, and effect for Central. This parameter is mainly used for infrastructure nodes.
central.exposure.loadBalancer.enabled	Set this to true to expose Central through a load balancer.
central.exposure.loadBalancer.port	Use this parameter to specify a custom port for your load balancer.
central.exposure.loadBalancer.ip	Use this parameter to specify a static IP address reserved for your load balancer.
central.exposure.route.enabled	Set this to true to expose Central through a Red Hat OpenShift route. The default value is false .
central.exposure.route.host	Specify a custom hostname to use for Central's route. Leave this unset to accept the default value that OpenShift Container Platform provides.
central.exposure.nodePort.enabled	Set this to true to expose Central through a node port. The default value is false .
central.exposure.nodePort.port	Use this to specify an explicit node port.
central.monitoring.exposeEndpoint	Use Enabled to enable monitoring for Central. When you enable monitoring, RHACS creates a new monitoring service on port number 9090 . The default value is Disabled .
central.nodeSelector	If you want this component to only run on specific nodes, you can use this parameter to configure a node selector.
central.persistence.hostPath.path	Specify a host path to store persistent data in a directory on the host. Red Hat does not recommend using this. If you need to use host path, you must use it with a node selector.
central.persistence.persistentVolumeClaim.claimName	The name of the PVC to manage persistent data. If no PVC with the given name exists, it is created. The default value is stackrox-db if not set. To prevent data loss, the PVC is not removed automatically when Central is deleted.

Parameter	Description
central.persistence.persistentVolumeClaim.size	The size of the persistent volume when created through the claim. This is automatically generated by default.
central.persistence.persistentVolumeClaim.storageClassName	The name of the storage class to use for the PVC. If your cluster is not configured with a default storage class, you must provide a value for this parameter.
central.resources.limits	Use this parameter to override the default resource limits for the Central.
central.resources.requests	Use this parameter to override the default resource requests for the Central.
central.imagePullSecrets	Use this parameter to specify the image pull secrets for the Central image.
central.db.passwordSecret.name	Specify a secret that has the database password in the password data item. Only use this parameter if you want to specify a connection string manually. If omitted, the operator auto-generates a password and stores it in the password item in the central-db-password secret.
central.db.connectionString	Setting this parameter will not deploy Central DB, and Central will connect using the specified connection string. If you specify a value for this parameter, you must also specify a value for central.db.passwordSecret.name . This parameter has the following constraints: <ul style="list-style-type: none"> ● Connection string must be in keyword/value format as described in the PostgreSQL documentation. For more information, see the links in the Additional resources section. ● Only PostgreSQL 13 is supported. ● Connections through PGBouncer are not supported. ● User must be a superuser who can create and delete databases.
central.db.tolerations	If the node selector selects tainted nodes, use this parameter to specify a taint toleration key, value, and effect for Central DB. This parameter is mainly used for infrastructure nodes.
central.db.persistence.hostPath.path	Specify a host path to store persistent data in a directory on the host. Red Hat does not recommend using this. If you need to use host path, you must use it with a node selector.
central.db.persistence.persistentVolumeClaim.claimName	The name of the PVC to manage persistent data. If no PVC with the given name exists, it is created. The default value is central-db if not set. To prevent data loss, the PVC is not removed automatically when Central is deleted.

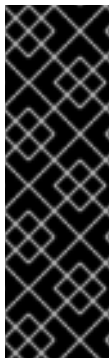
Parameter	Description
central.db.persistentVolumeClaim.size	The size of the persistent volume when created through the claim. This is automatically generated by default.
central.db.persistentVolumeClaim.storageClassName	The name of the storage class to use for the PVC. If your cluster is not configured with a default storage class, you must provide a value for this parameter.
central.db.resources.limits	Use this parameter to override the default resource limits for the Central DB.
central.db.resources.requests	Use this parameter to override the default resource requests for the Central DB.

4.2.1.2. StackRox Scanner settings

Parameter	Description
scanner.analyzer.nodeSelector	If you want this scanner to only run on specific nodes, you can use this parameter to configure a node selector.
scanner.analyzer.tolerations	If the node selector selects tainted nodes, use this parameter to specify a taint toleration key, value, and effect for the StackRox Scanner. This parameter is mainly used for infrastructure nodes.
scanner.analyzer.resources.limits	Use this parameter to override the default resource limits for the StackRox Scanner.
scanner.analyzer.resources.requests	Use this parameter to override the default resource requests for the StackRox Scanner.
scanner.analyzer.scaling.autoScaling	When enabled, the number of analyzer replicas is managed dynamically based on the load, within the limits specified.
scanner.analyzer.scaling.maxReplicas	Specifies the maximum replicas to be used in the analyzer autoscaling configuration
scanner.analyzer.scaling.minReplicas	Specifies the minimum replicas to be used in the analyzer autoscaling configuration
scanner.analyzer.scaling.replicas	When autoscaling is disabled, the number of replicas is always configured to match this value.
scanner.db.nodeSelector	If you want this component to only run on specific nodes, you can use this parameter to configure a node selector.

Parameter	Description
scanner.db.tolerations	If the node selector selects tainted nodes, use this parameter to specify a taint toleration key, value, and effect for the StackRox Scanner DB. This parameter is mainly used for infrastructure nodes.
scanner.db.resources.limits	Use this parameter to override the default resource limits for the StackRox Scanner DB.
scanner.db.resources.requests	Use this parameter to override the default resource requests for the StackRox Scanner DB.
scanner.monitoring.exposeEndpoint	Use Enabled to enable monitoring for the StackRox Scanner. When you enable monitoring, RHACS creates a new monitoring service on port number 9090 . The default value is Disabled .
scanner.scannerComponent	If you do not want to deploy the StackRox Scanner, you can disable it by using this parameter. If you disable the StackRox Scanner, all other settings in this section have no effect. Red Hat does not recommend disabling Red Hat Advanced Cluster Security for Kubernetes the StackRox Scanner. Do not disable the StackRox Scanner if you have enabled Scanner V4. Scanner V4 requires that the StackRox Scanner is also enabled to provide the necessary scanning capabilities.

4.2.1.3. Scanner V4 settings (Technology Preview)



IMPORTANT

Scanner V4 is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see [Technology Preview Features Support Scope](#).

Parameter	Description
scannerV4.db.nodeSelector	If you want this component to only run on specific nodes, you can use this parameter to configure a node selector.
scannerV4.db.tolerations	If the node selector selects tainted nodes, use this parameter to specify a taint toleration key, value, and effect for Scanner V4 DB. This parameter is mainly used for infrastructure nodes.
scannerV4.db.resources.limits	Use this parameter to override the default resource limits for Scanner V4 DB.

Parameter	Description
scannerV4.db.resources.requests	Use this parameter to override the default resource requests for Scanner V4 DB.
scannerV4.db.persistence.persistentVolumeClaim.claimName	The name of the PVC to manage persistent data for Scanner V4. If no PVC with the given name exists, it is created. The default value is scanner-v4-db if not set. To prevent data loss, the PVC is not removed automatically when Central is deleted.
scannerV4.indexer.nodeSelector	If you want this component to only run on specific nodes, you can use this parameter to configure a node selector.
scannerV4.indexer.tolerations	If the node selector selects tainted nodes, use this parameter to specify a taint toleration key, value, and effect for the Scanner V4 Indexer. This parameter is mainly used for infrastructure nodes.
scannerV4.indexer.resources.limits	Use this parameter to override the default resource limits for the Scanner V4 Indexer.
scannerV4.indexer.resources.requests	Use this parameter to override the default resource requests for the Scanner V4 Indexer.
scannerV4.indexer.scaling.autoScaling	When enabled, the number of Scanner V4 Indexer replicas is managed dynamically based on the load, within the limits specified.
scannerV4.indexer.scaling.maxReplicas	Specifies the maximum replicas to be used in the Scanner V4 Indexer autoscaling configuration.
scannerV4.indexer.scaling.minReplicas	Specifies the minimum replicas to be used in the Scanner V4 Indexer autoscaling configuration.
scannerV4.indexer.scaling.replicas	When autoscaling is disabled for the Scanner V4 Indexer, the number of replicas is always configured to match this value.
scannerV4.matcher.nodeSelector	If you want this component to only run on specific nodes, you can use this parameter to configure a node selector.
scannerV4.matcher.tolerations	If the node selector selects tainted nodes, use this parameter to specify a taint toleration key, value, and effect for the Scanner V4 Matcher. This parameter is mainly used for infrastructure nodes.
scannerV4.matcher.resources.limits	Use this parameter to override the default resource limits for the Scanner V4 Matcher.
scannerV4.matcher.resources.requests	Use this parameter to override the default resource requests for the Scanner V4 Matcher.

Parameter	Description
scannerV4.matcher.scaling.autoScaling	When enabled, the number of Scanner V4 Matcher replicas is managed dynamically based on the load, within the limits specified.
scannerV4.matcher.scaling.maxReplicas	Specifies the maximum replicas to be used in the Scanner V4 Matcher autoscaling configuration.
scannerV4.matcher.scaling.minReplicas	Specifies the minimum replicas to be used in the Scanner V4 Matcher autoscaling configuration.
scannerV4.matcher.scaling.replicas	When autoscaling is disabled for the Scanner V4 Matcher, the number of replicas is always configured to match this value.
scannerV4.monitoring.exposeEndpoint	Configures a monitoring endpoint for Scanner V4. The monitoring endpoint allows other services to collect metrics from Scanner V4, provided in a Prometheus-compatible format. Use Enabled to expose the monitoring endpoint. When you enable monitoring, RHACS creates a new service, monitoring , with port 9090, and a network policy allowing inbound connections to the port. By default, this is not enabled.
scannerV4.scannerComponent	Enables Scanner V4. The default value is default , which is disabled. To enable Scanner V4, set this parameter to Enabled .

4.2.1.4. General and miscellaneous settings

Parameter	Description
tls.additionalCAs	Additional Trusted CA certificates for the secured cluster to trust. These certificates are typically used when integrating with services using a private certificate authority.
misc.createSCCs	Specify true to create SecurityContextConstraints (SCCs) for Central. Setting to true might cause issues in some environments.
customize.annotations	Allows specifying custom annotations for the Central deployment.
customize.envVars	Advanced settings to configure environment variables.
egress.connectivityPolicy	Configures whether RHACS should run in online or offline mode. In offline mode, automatic updates of vulnerability definitions and kernel modules are disabled.
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.
overlays	See Customizing the installation using the operator with overlays

4.2.2. Customizing the installation using the Operator with overlays

Learn how to tailor the installation of RHACS using the Operator method with overlays.

4.2.2.1. Overlays

When **Central** or **SecuredCluster** custom resources don't expose certain low-level configuration options as parameters, you can use the **.spec.overlays** field for adjustments. Use this field to amend the Kubernetes resources generated by these custom resources.

The **.spec.overlays** field comprises a sequence of patches, applied in their listed order. These patches are processed by the Operator on the Kubernetes resources before deployment to the cluster.



WARNING

The **.spec.overlays** field in both **Central** and **SecuredCluster** allows users to modify low-level Kubernetes resources in arbitrary ways. Use this feature only when the desired customization is not available through the **SecuredCluster** or **Central** custom resources.

Support for the **.spec.overlays** feature is limited primarily because it grants the ability to make intricate and highly specific modifications to Kubernetes resources, which can vary significantly from one implementation to another. This level of customization introduces a complexity that goes beyond standard usage scenarios, making it challenging to provide broad support. Each modification can be unique, potentially interacting with the Kubernetes system in unpredictable ways across different versions and configurations of the product. This variability means that troubleshooting and guaranteeing the stability of these customizations require a level of expertise and understanding specific to each individual's setup.

Consequently, while this feature empowers tailoring Kubernetes resources to meet precise needs, greater responsibility must also be assumed to ensure the compatibility and stability of configurations, especially during upgrades or changes to the underlying product.

The following example shows the structure of an overlay:

```
overlays:
- apiVersion: v1      1
  kind: ConfigMap     2
  name: my-configmap  3
  patches:
  - path: .data       4
    value: |          5
      key1: data2
      key2: data2
```

1 Targeted Kubernetes resource ApiVersion, for example **apps/v1**, **v1**, **networking.k8s.io/v1**

2 Resource type (e.g., Deployment, ConfigMap, NetworkPolicy)

- 3 Name of the resource, for example **my-configmap**
- 4 JSONPath expression to the field, for example **spec.template.spec.containers[name:central].env[-1]**
- 5 YAML string for the new field value

4.2.2.1.1. Adding an overlay

For customizations, you can add overlays to **Central** or **SecuredCluster** custom resources. Use the OpenShift CLI (**oc**) or the OpenShift Container Platform web console for modifications.

If overlays do not take effect as expected, check the RHACS Operator logs for any syntax errors or issues logged.

4.2.2.2. Overlay examples

4.2.2.2.1. Specifying an EKS pod role ARN for the Central ServiceAccount

Add an Amazon Elastic Kubernetes Service (EKS) pod role Amazon Resource Name (ARN) annotation to the **central** ServiceAccount as shown in the following example:

```
apiVersion: platform.stackrox.io
kind: Central
metadata:
  name: central
spec:
  # ...
  overlays:
  - apiVersion: v1
    kind: ServiceAccount
    name: central
    patches:
    - path: metadata.annotations.eks\.amazonaws\.com/role-arn
      value: "\"arn:aws:iam:1234:role\""
```

4.2.2.2.2. Injecting an environment variable into the Central deployment

Inject an environment variable into the **central** deployment as shown in the following example:

```
apiVersion: platform.stackrox.io
kind: Central
metadata:
  name: central
spec:
  # ...
  overlays:
  - apiVersion: apps/v1
    kind: Deployment
    name: central
    patches:
    - path: spec.template.spec.containers[name:central].env[-1]
```



```
value: |
  name: MY_ENV_VAR
  value: value
```

4.2.2.2.3. Extending network policy with an ingress rule

Add an ingress rule to the **allow-ext-to-central** network policy for port 999 traffic as shown in the following example:

```
apiVersion: platform.stackrox.io
kind: Central
metadata:
  name: central
spec:
  # ...
  overlays:
  - apiVersion: networking.k8s.io/v1
    kind: NetworkPolicy
    name: allow-ext-to-central
    patches:
    - path: spec.ingress[-1]
      value: |
        ports:
        - port: 999
          protocol: TCP
```

4.2.2.2.4. Modifying ConfigMap data

Modify the **central-endpoints** ConfigMap data as shown in the following example:

```
apiVersion: platform.stackrox.io
kind: Central
metadata:
  name: central
spec:
  # ...
  overlays:
  - apiVersion: v1
    kind: ConfigMap
    name: central-endpoints
    patches:
    - path: data
      value: |
        endpoints.yaml: |
          disableDefault: false
```

4.2.2.2.5. Adding a container to theCentral deployment

Add a new container to the **central** deployment as shown in the following example:

```
apiVersion: platform.stackrox.io
kind: Central
metadata:
  name: central
```

```
spec:
  # ...
  overlays:
  - apiVersion: apps/v1
    kind: Deployment
    name: central
    patches:
    - path: spec.template.spec.containers[-1]
  value: |
    name: nginx
    image: nginx
    ports:
    - containerPort: 8000
      name: http
      protocol: TCP
```

Additional resources

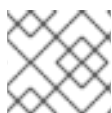
- [Connection Strings - PostgreSQL Docs](#)
- [Parameter Interaction via the Configuration File - PostgreSQL Docs](#)
- [The pg_hba.conf File - PostgreSQL Docs](#)

4.3. GENERATING AND APPLYING AN INIT BUNDLE FOR RHACS ON RED HAT OPENSIFT

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.

To configure an init bundle for RHACS Cloud Service, see the following resources:

- [Generating an init bundle for secured clusters \(Red Hat Cloud\)](#)
- [Applying an init bundle for secured clusters \(Red Hat Cloud\)](#)
- [Generating an init bundle for Kubernetes secured clusters](#)
- [Applying an init bundle for Kubernetes secured clusters](#)



NOTE

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

4.3.1. Generating an init bundle

4.3.1.1. Generating an init bundle by using the RHACS portal

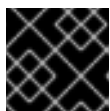
You can create an init bundle containing secrets by using the RHACS portal.

**NOTE**

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

Procedure

1. Find the address of the RHACS portal as described in "Verifying Central installation using the Operator method".
2. Log in to the RHACS portal.
3. If you do not have secured clusters, the **Platform Configuration → Clusters** page appears.
4. Click **Create init bundle**
5. Enter a name for the cluster init bundle.
6. Select your platform.
7. Select the installation method you will use for your secured clusters: **Operator** or **Helm chart**.
8. Click **Download** to generate and download the init bundle, which is created in the form of a YAML file. You can use one init bundle and its corresponding YAML file for all secured clusters if you are using the same installation method.

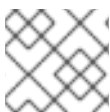
**IMPORTANT**

Store this bundle securely because it contains secrets.

9. Apply the init bundle by using it to create resources on the secured cluster.
10. Install secured cluster services on each cluster.

4.3.1.2. 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>
```

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.

4.3.1.3. Applying the init bundle on the secured cluster

Before you configure a secured cluster, you must apply the init bundle by using it to create the required resources on the cluster. Applying the init bundle allows the services on the secured cluster to communicate with Central.



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.
- You must have created the **stackrox** project, or namespace, on the cluster where secured cluster services will be installed. Using **stackrox** for the project is not required, but ensures that vulnerabilities for RHACS processes are not reported when scanning your clusters.

Procedure

To create resources, perform only one of the following steps:

- Create resources using the OpenShift Container Platform web console: In the OpenShift Container Platform web console, make sure that you are in the **stackrox** namespace. 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**. When the command is complete, the display shows that the **collector-tls**, **sensor-tls**, and **admission-control-tls`** resources were created.
- Create resources using the Red Hat OpenShift CLI: 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.

4.3.2. Next steps

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

4.3.3. Additional resources

- [Installing RHACS on secured clusters by using Helm charts](#)

4.4. INSTALLING SECURED CLUSTER SERVICES FOR RHACS ON RED HAT OPENSIFT

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

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

4.4.1. Installing RHACS on secured clusters by using the Operator

4.4.1.1. Installing secured cluster services

You can install Secured Cluster services on your clusters by using the Operator, which creates the **SecuredCluster** custom resource. You must install the Secured Cluster services on every cluster in your environment that you want to monitor.

Prerequisites

- If you are using OpenShift Container Platform, you must install version 4.11 or later.
- You have installed the RHACS Operator on the cluster that you want to secure, called the secured cluster.
- You have generated an init bundle and applied it to the cluster.

Procedure

1. On the OpenShift Container Platform web console for the secured cluster, go 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 by 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**.
 - Use the default value of **central.stackrox.svc:443** *only* if you are installing secured cluster services in the same cluster where Central is installed.
 - 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.
10. For the remaining fields, accept the default values or configure custom values if needed. For example, you might need to configure TLS if you are using custom certificates or untrusted CAs. See "Configuring Secured Cluster services options for RHACS using the Operator" for more information.
11. Click **Create**.
12. After a brief pause, the **SecuredClusters** page displays the status of **stackrox-secured-cluster-services**. You might see the following conditions:
 - **Conditions: Deployed, Initialized:** The secured cluster services have been installed and the secured cluster is communicating with Central.
 - **Conditions: Initialized, Irreconcilable:** The secured cluster is not communicating with Central. Make sure that you applied the init bundle you created in the RHACS web portal to the secured cluster.

Next steps

1. Configure additional secured cluster settings (optional).
2. Verify installation.

4.4.2. Installing RHACS 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.

4.4.2.1. Installing RHACS on secured clusters by using Helm charts without customizations

4.4.2.1.1. Adding the Helm chart repository

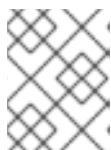
Procedure

- 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:

- Central services Helm chart (**central-services**) for installing the centralized components (Central and Scanner).



NOTE

You deploy centralized components only once and you can monitor multiple separate clusters by using the same installation.

- 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/
```

4.4.2.1.2. 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).

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 address and the port number that you are exposing the Central service on.

Procedure

- Run the following command on OpenShift Container Platform 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 scanner.disable=false 4
```

- 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** Specify the address and port number for Central. For example, **acs.domain.com:443**.
- 4** Set the value of the **scanner.disable** parameter to **false**, which means that Scanner-slim will be enabled during the installation. In Kubernetes, the secured cluster services now include Scanner-slim as an optional component.

Additional resources

- [Generating and applying an init bundle for RHACS on Red Hat OpenShift](#)

4.4.2.2. 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.

4.4.2.2.1. Configuration parameters

Parameter	Description
clusterName	Name of your cluster.

Parameter	Description
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.
image.main.registry	Address of the registry you are using for the main image.

Parameter	Description
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 CORE_BPF , EBPF (deprecated), 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. RHACS does not support this feature on OpenShift Container Platform 3.11.
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 with the EBPF collection method requires Central to provide the matching eBPF probe. 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.

4.4.2.2.1.1. 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).

4.4.2.2.2. Installing the secured-cluster-services Helm chart with customizations

After you configure the **values-public.yaml** and **values-private.yaml** files, install the **secured-cluster-services** Helm chart to deploy the following per-cluster and per-node components:

- Sensor
- Admission controller
- Collector
- Scanner: optional for secured clusters when the StackRox Scanner is installed
- Scanner DB: optional for secured clusters when the StackRox Scanner is installed
- Scanner V4 Indexer and Scanner V4 DB: optional for secured clusters when Scanner V4 is installed



IMPORTANT

Scanner V4 is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see [Technology Preview Features Support Scope](#).

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 address and the port number that you are exposing the Central service on.

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 and applying an init bundle for RHACS on Red Hat OpenShift](#)

4.4.2.3. 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.

When using the **helm upgrade** command to make changes, the following guidelines and requirements apply:

- You can also specify configuration values using the **--set** or **--set-file** parameters. However, these options are not saved, and you must manually specify all the options again whenever you make changes.
- Some changes, such as enabling a new component like Scanner V4, require new certificates to be issued for the component. Therefore, you must provide a CA when making these changes.



IMPORTANT

Scanner V4 is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see [Technology Preview Features Support Scope](#).

- If the CA was generated by the Helm chart during the initial installation, you must retrieve these automatically generated values from the cluster and provide them to the **helm upgrade** command. The post-installation notes of the **central-services** Helm chart include a command for retrieving the automatically generated values.
- If the CA was generated outside of the Helm chart and provided during the installation of the **central-services** chart, then you must perform that action again when using the **helm upgrade** command, for example, by using the **--reuse-values** flag with the **helm upgrade** command.

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** If you have modified values that are not included in the **values_public.yaml** and **values_private.yaml** files, include the **--reuse-values** parameter.

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

This method is also referred to as the manifest installation method.

Prerequisites

- If you plan to use the **roxctl** CLI command to generate the files used by the sensor installation script, you have installed the **roxctl** CLI.
- You have generated the files that will be used by the sensor installation script.

Procedure

- On the OpenShift Container Platform secured cluster, deploy the Sensor component by running the sensor installation script.

4.4.3.1. Installing the roxctl CLI

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

4.4.3.1.1. 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/4.4.3/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
```

4.4.3.1.2. 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/4.4.3/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
```

4.4.3.1.3. 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/4.4.3/bin/Windows/roxctl.exe
```

Verification

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

```
$ roxctl version
```

4.4.3.2. Installing Sensor

To monitor a cluster, you must deploy Sensor. You must deploy Sensor into each cluster that you want to monitor. This installation method is also called the manifest installation method.

To perform an installation by using the manifest installation method, follow *only one* of the following procedures:

- Use the RHACS web portal to download the cluster bundle, and then extract and run the sensor script.
- Use the **roxctl** CLI to generate the required sensor configuration for your OpenShift Container Platform cluster and associate it with your Central instance.

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).

4.4.3.2.1. Manifest installation method by using the web portal

Procedure

1. On your secured cluster, in the RHACS portal, go to **Platform Configuration → Clusters**.
2. Select **Secure a cluster → Legacy installation method**.

3. Specify a name for the cluster.
4. Provide appropriate values for the fields based on where you are deploying the Sensor.
 - If you are deploying Sensor in the same cluster, accept the default values for all the fields.
 - If you are deploying into a different cluster, replace **central.stackrox.svc:443** with a load balancer, node port, or other address, including the port number, that is accessible from the other cluster.
 - If you are using a non-gRPC capable load balancer, such as HAProxy, AWS Application Load Balancer (ALB), or AWS Elastic Load Balancing (ELB), use the WebSocket Secure (**wss**) protocol. To use **wss**:
 - Prefix the address with **wss://**.
 - Add the port number after the address, for example, **wss://stackrox-central.example.com:443**.
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, extract 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 help.

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

4.4.3.2.2. Manifest installation by using the roxctl CLI

Procedure

1. Generate the required sensor configuration for your OpenShift Container Platform cluster and associate it with your Central instance by running the following command:

```
$ roxctl sensor generate openshift --openshift-version <ocp_version> --name  
<cluster_name> --central "$ROX_ENDPOINT" ❶
```

- ❶ For the **--openshift-version** option, specify the major OpenShift Container Platform version number for your cluster. For example, specify **3** for OpenShift Container Platform version **3.x** and specify **4** for OpenShift Container Platform version **4.x**.

- From a system that has access to the monitored cluster, extract 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 help.

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

Verification

- 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
```

- 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.

4.5. CONFIGURING SECURED CLUSTER SERVICES OPTIONS FOR RHACS USING THE OPERATOR

When installing Secured Cluster services by using the Operator, you can configure optional settings.

4.5.1. Secured Cluster services configuration options

When you create a Central instance, the Operator lists the following configuration options for the **Central** custom resource.

4.5.1.1. Required Configuration Settings

Parameter	Description
-----------	-------------

Parameter	Description
centralEndpoint	The endpoint of Central instance to connect to, including the port number. If using a non-gRPC capable load balancer, use the WebSocket protocol by prefixing the endpoint address with wss:// . If you do not specify a value for this parameter, Sensor attempts to connect to a Central instance running in the same namespace.
clusterName	The unique name of this cluster, which shows up in the RHACS portal. After you set the name by using this parameter, you cannot change it again. To change the name, you must delete and re-create the object.

4.5.1.2. Admission controller settings

Parameter	Description
admissionControl.listenOnCreates	Specify true to enable preventive policy enforcement for object creations. The default value is true .
admissionControl.listenOnEvents	Specify true to enable monitoring and enforcement for Kubernetes events, such as port-forward and exec events. It is used to control access to resources through the Kubernetes API. The default value is true .
admissionControl.listenOnUpdates	Specify true to enable preventive policy enforcement for object updates. It will not have any effect unless Listen On Creates is set to true as well. The default value is true .
admissionControl.nodeSelector	If you want this component to only run on specific nodes, you can configure a node selector using this parameter.
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.resources.limits	Use this parameter to override the default resource limits for the admission controller.
admissionControl.resources.requests	Use this parameter to override the default resource requests for the admission controller.
admissionControl.bypass	<p>Use one of the following values to configure the bypassing of admission controller enforcement:</p> <ul style="list-style-type: none"> • BreakGlassAnnotation to enable bypassing the admission controller via the admission.stackrox.io/break-glass annotation. • Disabled to disable the ability to bypass admission controller enforcement for the secured cluster. <p>The default value is BreakGlassAnnotation.</p>

Parameter	Description
admissionControl.contactImageScanners	<p>Use one of the following values to specify if the admission controller must connect to the image scanner:</p> <ul style="list-style-type: none"> ● ScanIfMissing if the scan results for the image are missing. ● DoNotScanInline to skip scanning the image when processing the admission request. <p>The default value is DoNotScanInline.</p>
admissionControl.timeoutSeconds	<p>Use this parameter to specify the maximum number of seconds Red Hat Advanced Cluster Security for Kubernetes must wait for an admission review before marking it as fail open.</p>

4.5.1.3. Scanner configuration

Use Scanner configuration settings to modify the local cluster scanner for the OpenShift Container Registry (OCR).

Parameter	Description
scanner.analyzer.nodeSelector	Specify a node selector label as label-key: label-value to force Scanner to only schedule on nodes with the specified label.
scanner.analyzer.resources.requests.memory	The memory request for the Scanner container. Use this parameter to override the default value.
scanner.analyzer.resources.requests.cpu	The CPU request for the Scanner container. Use this parameter to override the default value.
scanner.analyzer.resources.limits.memory	The memory limit for the Scanner container. Use this parameter to override the default value.
scanner.analyzer.resources.limits.cpu	The CPU limit for the Scanner container. Use this parameter to override the default value.
scanner.scaling.autoScaling	If you set this option to Disabled , Red Hat Advanced Cluster Security for Kubernetes disables autoscaling on the Scanner deployment. The default value is Enabled .
scanner.scaling.minReplicas	The minimum number of replicas for autoscaling. The default value is 2 .
scanner.scaling.maxReplicas	The maximum number of replicas for autoscaling. The default value is 5 .

Parameter	Description
scanner.scaling.replicas	The default number of replicas. The default value is 3 .
scanner.Tolerations	If the node selector selects tainted nodes, use this parameter to specify a taint toleration key, value, and effect for Scanner.
scanner.db.nodeSelector	Specify a node selector label as label-key: label-value to force Scanner DB to only schedule on nodes with the specified label.
scanner.db.resources.requests.memory	The memory request for the Scanner DB container. Use this parameter to override the default value.
scanner.db.resources.requests.cpu	The CPU request for the Scanner DB container. Use this parameter to override the default value.
scanner.db.resources.limits.memory	The memory limit for the Scanner DB container. Use this parameter to override the default value.
scanner.db.resources.limits.cpu	The CPU limit for the Scanner DB container. Use this parameter to override the default value.
scanner.db.tolerations	If the node selector selects tainted nodes, use this parameter to specify a taint toleration key, value, and effect for Scanner DB.
scanner.scannerComponent	If you set this option to Disabled , Red Hat Advanced Cluster Security for Kubernetes does not deploy the Scanner deployment. Do not disable the Scanner on OpenShift Container Platform clusters. The default value is AutoSense .

4.5.1.4. Image configuration


Use image configuration settings when you are using a custom registry.

Parameter	Description
imagePullSecrets.name	Additional image pull secrets to be taken into account for pulling images.

4.5.1.5. Per node settings

Per node settings define the configuration settings for components that run on each node in a cluster to secure the cluster. These components are Collector and Compliance.

Parameter	Description
-----------	-------------

Parameter	Description
perNode.collector.collection	<p>The method for system-level data collection. The default value is CORE_BPF. Red Hat recommends using CORE_BPF for data collection. If you select NoCollection, Collector does not report any information about the network activity and the process executions. Available options are NoCollection, EBPF, and CORE_BPF.</p> <div>  <div> <p>NOTE</p> <p>Red Hat has deprecated the EBPF option and will remove it from future versions. Use CORE_BPF instead.</p> </div> </div>
perNode.collector.imageFlavor	<p>The image type to use for Collector. You can specify it as Regular or Slim. Regular images are bigger, but contain kernel modules for most kernels. If you use the Slim image type, you must ensure that your Central instance is connected to the internet, or regularly receives Collector support package updates. The default value is Slim.</p>
perNode.collector.resources.limits	<p>Use this parameter to override the default resource limits for Collector.</p>
perNode.collector.resources.requests	<p>Use this parameter to override the default resource requests for Collector.</p>
perNode.compliance.resources.requests	<p>Use this parameter to override the default resource requests for Compliance.</p>
perNode.compliance.resources.limits	<p>Use this parameter to override the default resource limits for Compliance.</p>

4.5.1.6. Taint Tolerations settings

Parameter	Description
taintToleration	<p>To ensure comprehensive monitoring of your cluster activity, Red Hat Advanced Cluster Security for Kubernetes runs services on every node in the cluster, including tainted nodes by default. If you do not want this behavior, specify AvoidTaints for this parameter.</p>

4.5.1.7. Sensor configuration

This configuration defines the settings of the Sensor components, which runs on one node in a cluster.

Parameter	Description
sensor.nodeSelector	If you want Sensor to only run on specific nodes, you can configure a node selector.
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.
sensor.resources.limits	Use this parameter to override the default resource limits for Sensor.
sensor.resources.requests	Use this parameter to override the default resource requests for Sensor.

4.5.1.8. General and miscellaneous settings

Parameter	Description
tls.additionalCAs	Additional trusted CA certificates for the secured cluster. These certificates are used when integrating with services using a private certificate authority.
misc.createSCCs	Set this to true to create SCCs for Central. It may cause issues in some environments.
customize.annotations	Allows specifying custom annotations for the Central deployment.
customize.envVars	Advanced settings to configure environment variables.
egress.connectivityPolicy	Configures whether Red Hat Advanced Cluster Security for Kubernetes should run in online or offline mode. In offline mode, automatic updates of vulnerability definitions and kernel modules are disabled.
overlays	See Customizing the installation using the operator with overlays

4.5.2. Customizing the installation using the Operator with overlays

Learn how to tailor the installation of RHACS using the Operator method with overlays.

4.5.2.1. Overlays

When **Central** or **SecuredCluster** custom resources don't expose certain low-level configuration options as parameters, you can use the **.spec.overlays** field for adjustments. Use this field to amend the Kubernetes resources generated by these custom resources.

The **.spec.overlays** field comprises a sequence of patches, applied in their listed order. These patches are processed by the Operator on the Kubernetes resources before deployment to the cluster.



WARNING

The **.spec.overlays** field in both **Central** and **SecuredCluster** allows users to modify low-level Kubernetes resources in arbitrary ways. Use this feature only when the desired customization is not available through the **SecuredCluster** or **Central** custom resources.

Support for the **.spec.overlays** feature is limited primarily because it grants the ability to make intricate and highly specific modifications to Kubernetes resources, which can vary significantly from one implementation to another. This level of customization introduces a complexity that goes beyond standard usage scenarios, making it challenging to provide broad support. Each modification can be unique, potentially interacting with the Kubernetes system in unpredictable ways across different versions and configurations of the product. This variability means that troubleshooting and guaranteeing the stability of these customizations require a level of expertise and understanding specific to each individual's setup. Consequently, while this feature empowers tailoring Kubernetes resources to meet precise needs, greater responsibility must also be assumed to ensure the compatibility and stability of configurations, especially during upgrades or changes to the underlying product.

The following example shows the structure of an overlay:

```
overlays:
- apiVersion: v1      1
  kind: ConfigMap     2
  name: my-configmap  3
  patches:
  - path: .data       4
    value: |          5
      key1: data2
      key2: data2
```

- 1 Targeted Kubernetes resource ApiVersion, for example **apps/v1**, **v1**, **networking.k8s.io/v1**
- 2 Resource type (e.g., Deployment, ConfigMap, NetworkPolicy)
- 3 Name of the resource, for example **my-configmap**
- 4 JSONPath expression to the field, for example **spec.template.spec.containers[name:central].env[-1]**
- 5 YAML string for the new field value

4.5.2.1.1. Adding an overlay

For customizations, you can add overlays to **Central** or **SecuredCluster** custom resources. Use the OpenShift CLI (**oc**) or the OpenShift Container Platform web console for modifications.

If overlays do not take effect as expected, check the RHACS Operator logs for any syntax errors or issues logged.

4.5.2.2. Overlay examples

4.5.2.2.1. Specifying an EKS pod role ARN for the Central ServiceAccount

Add an Amazon Elastic Kubernetes Service (EKS) pod role Amazon Resource Name (ARN) annotation to the **central** ServiceAccount as shown in the following example:

```
apiVersion: platform.stackrox.io
kind: Central
metadata:
  name: central
spec:
  # ...
  overlays:
    - apiVersion: v1
      kind: ServiceAccount
      name: central
      patches:
        - path: metadata.annotations.eks\.amazonaws\.com/role-arn
          value: "\arn:aws:iam:1234:role\""
```

4.5.2.2.2. Injecting an environment variable into the Central deployment

Inject an environment variable into the **central** deployment as shown in the following example:

```
apiVersion: platform.stackrox.io
kind: Central
metadata:
  name: central
spec:
  # ...
  overlays:
    - apiVersion: apps/v1
      kind: Deployment
      name: central
      patches:
        - path: spec.template.spec.containers[name:central].env[-1]
          value: |
            name: MY_ENV_VAR
            value: value
```

4.5.2.2.3. Extending network policy with an ingress rule

Add an ingress rule to the **allow-ext-to-central** network policy for port 999 traffic as shown in the following example:

```
apiVersion: platform.stackrox.io
kind: Central
metadata:
  name: central
```

```
spec:
  # ...
  overlays:
  - apiVersion: networking.k8s.io/v1
    kind: NetworkPolicy
    name: allow-ext-to-central
    patches:
    - path: spec.ingress[-1]
      value: |
        ports:
        - port: 999
          protocol: TCP
```

4.5.2.2.4. Modifying ConfigMap data

Modify the **central-endpoints** ConfigMap data as shown in the following example:

```
apiVersion: platform.stackrox.io
kind: Central
metadata:
  name: central
spec:
  # ...
  overlays:
  - apiVersion: v1
    kind: ConfigMap
    name: central-endpoints
    patches:
    - path: data
      value: |
        endpoints.yaml: |
          disableDefault: false
```

4.5.2.2.5. Adding a container to the Central deployment

Add a new container to the **central** deployment as shown in the following example:

```
apiVersion: platform.stackrox.io
kind: Central
metadata:
  name: central
spec:
  # ...
  overlays:
  - apiVersion: apps/v1
    kind: Deployment
    name: central
    patches:
    - path: spec.template.spec.containers[-1]
      value: |
        name: nginx
        image: nginx
        ports:
```

```
- containerPort: 8000
  name: http
  protocol: TCP
```

4.6. VERIFYING INSTALLATION OF RHACS ON RED HAT OPENSIFT

Provides steps to verify that RHACS is properly installed.

4.6.1. Verifying installation

After you complete the installation, run a few vulnerable applications and go to the RHACS portal to evaluate the results of security assessments and policy violations.



NOTE

The sample applications listed in the following section contain critical vulnerabilities and they are specifically designed to verify the build and deploy-time assessment features of Red Hat Advanced Cluster Security for Kubernetes.

To verify installation:

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. Go to **<https://localhost:18443/>**.

2. Using the Red Hat OpenShift CLI, create a new project:

```
$ oc new-project test
```

3. Start some applications with critical vulnerabilities:

```
$ oc run shell --labels=app=shellshock,team=test-team \
  --image=quay.io/stackrox-io/docs:example-vulnerables-cve-2014-6271 -n test
$ oc run samba --labels=app=rce \
  --image=quay.io/stackrox-io/docs:example-vulnerables-cve-2017-7494 -n test
```

Red Hat Advanced Cluster Security for Kubernetes automatically scans these deployments for security risks and policy violations as soon as they are submitted to the cluster. Go to the RHACS portal to view the violations. You can log in to the RHACS portal by using the default username **admin** and the

generated password.

CHAPTER 5. INSTALLING RHACS ON OTHER PLATFORMS

5.1. HIGH-LEVEL OVERVIEW OF INSTALLING RHACS ON OTHER PLATFORMS

Red Hat Advanced Cluster Security for Kubernetes (RHACS) provides security services for self-managed RHACS on platforms such as Amazon Elastic Kubernetes Service (Amazon EKS), Google Kubernetes Engine (Google GKE), and Microsoft Azure Kubernetes Service (Microsoft AKS).

Before you install:

- Understand the [installation methods for different platforms](#).
- Understand [Red Hat Advanced Cluster Security for Kubernetes architecture](#) .
- Check the [default resource requirements page](#).

The following list provides a high-level overview of installation steps:

1. Install [Central services](#) on a cluster using Helm charts or the **roxctl** CLI.
2. Generate and apply an [init bundle](#).
3. Install [secured cluster resources](#) on each of your secured clusters.

5.2. INSTALLING CENTRAL SERVICES FOR RHACS ON OTHER PLATFORMS

Central is the resource that contains the RHACS application management interface and services. It handles data persistence, API interactions, and RHACS portal access. You can use the same Central instance to secure multiple OpenShift Container Platform or Kubernetes clusters.

You can install Central by using one of the following methods:

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

5.2.1. Install Central using Helm charts

You can install Central using Helm charts without any customization, using the default values, or by using Helm charts with additional customizations of configuration parameters.

5.2.1.1. Install Central using Helm charts without customization

You can install RHACS on your Red Hat OpenShift cluster without any customizations. You must add the Helm chart repository and install the **central-services** Helm chart to install the centralized components of Central and Scanner.

5.2.1.1.1. 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:

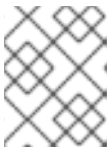
- Central services Helm chart (**central-services**) for installing the centralized components (Central and Scanner).



NOTE

You deploy centralized components only once and you can monitor multiple separate clusters by using the same installation.

- 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/
```

5.2.1.1.2. Installing the central-services Helm chart without customizations

Use the following instructions to install the **central-services** Helm chart to deploy the centralized components (Central and Scanner).

Prerequisites

- 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](#).

Procedure

- Run the following command to install Central services and expose Central using a route:

```
$ helm install -n stackrox \
  --create-namespace stackrox-central-services rhacs/central-services \
  --set imagePullSecrets.username=<username> 1
  --set imagePullSecrets.password=<password> 2
  --set central.exposure.route.enabled=true
```

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

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

- Or, run the following command to install Central services and expose Central using a load balancer:

```
$ helm install -n stackrox \
  --create-namespace stackrox-central-services rhacs/central-services \
  --set imagePullSecrets.username=<username> \ 1
  --set imagePullSecrets.password=<password> \ 2
  --set central.exposure.loadBalancer.enabled=true
```

- 1** Include the user name for your pull secret for Red Hat Container Registry authentication.
- 2** Include the password for your pull secret for Red Hat Container Registry authentication.

- Or, run the following command to install Central services and expose Central using port forward:

```
$ helm install -n stackrox \
  --create-namespace stackrox-central-services rhacs/central-services \
  --set imagePullSecrets.username=<username> \ 1
  --set imagePullSecrets.password=<password> \ 2
```

- 1** Include the user name for your pull secret for Red Hat Container Registry authentication.
- 2** Include the password for your pull secret for Red Hat Container Registry authentication.

IMPORTANT

- If you are installing Red Hat Advanced Cluster Security for Kubernetes in a cluster that requires a proxy to connect to external services, you must specify your proxy configuration by using the **proxyConfig** parameter. For example:

```
env:
  proxyConfig: |
    url: http://proxy.name:port
    username: username
    password: password
    excludes:
      - some.domain
```

- If you already created one or more image pull secrets in the namespace in which you are installing, instead of using a username and password, you can use **--set imagePullSecrets.useExisting="<pull-secret-1;pull-secret-2>"**.
- Do not use image pull secrets:
 - If you are pulling your images from **quay.io/stackrox-io** or a registry in a private network that does not require authentication. Use **--set imagePullSecrets.allowNone=true** instead of specifying a username and password.
 - If you already configured image pull secrets in the default service account in the namespace you are installing. Use **--set imagePullSecrets.useFromDefaultServiceAccount=true** instead of specifying a username and password.

The output of the installation command includes:

- An automatically generated administrator password.
- Instructions on storing all the configuration values.
- Any warnings that Helm generates.

5.2.1.2. Install Central using Helm charts with customizations

You can install RHACS on your Red Hat OpenShift cluster with customizations by using Helm chart configuration parameters 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.
- Configuration file **declarative-config-values.yaml**: Create this file if you are using declarative configuration to add the declarative configuration mounts to Central.

5.2.1.2.1. Private configuration file

This section lists the configurable parameters of the **values-private.yaml** file. There are no default values for these parameters.

5.2.1.2.1.1. Image pull secrets

The credentials that are required for pulling images from the registry depend on the following factors:

- If you are using a custom registry, you must specify these parameters:
 - **imagePullSecrets.username**
 - **imagePullSecrets.password**
 - **image.registry**
- If you do not use a username and password to log in to the custom registry, you must specify one of the following parameters:
 - **imagePullSecrets.allowNone**
 - **imagePullSecrets.useExisting**
 - **imagePullSecrets.useFromDefaultServiceAccount**

Parameter	Description
imagePullSecrets.username	The username of the account that is used to log in to the registry.
imagePullSecrets.password	The password of the account that is used to log in to the registry.
imagePullSecrets.allowNone	Use true if you are using a custom registry and it allows pulling images without credentials.
imagePullSecrets.useExisting	A comma-separated list of secrets as values. For example, secret1, secret2, secretN . Use this option if you have already created pre-existing image pull secrets with the given name in the target namespace.
imagePullSecrets.useFromDefaultServiceAccount	Use true if you have already configured the default service account in the target namespace with sufficiently scoped image pull secrets.

5.2.1.2.1.2. Proxy configuration

If you are installing Red Hat Advanced Cluster Security for Kubernetes in a cluster that requires a proxy to connect to external services, you must specify your proxy configuration by using the **proxyConfig** parameter. For example:

```
env:
  proxyConfig: |
    url: http://proxy.name:port
    username: username
    password: password
    excludes:
      - some.domain
```

Parameter	Description
env.proxyConfig	Your proxy configuration.

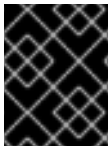
5.2.1.2.1.3. Central

Configurable parameters for Central.

For a new installation, you can skip the following parameters:

- **central.jwtSigner.key**
- **central.serviceTLS.cert**
- **central.serviceTLS.key**
- **central.adminPassword.value**

- **central.adminPassword.htpasswd**
- **central.db.serviceTLS.cert**
- **central.db.serviceTLS.key**
- **central.db.password.value**
- When you do not specify values for these parameters the Helm chart autogenerates values for them.
- If you want to modify these values you can use the **helm upgrade** command and specify the values using the **--set** option.



IMPORTANT

For setting the administrator password, you can only use either **central.adminPassword.value** or **central.adminPassword.htpasswd**, but not both.

Parameter	Description
central.jwtSigner.key	A private key which RHACS should use for signing JSON web tokens (JWTs) for authentication.
central.serviceTLS.cert	An internal certificate that the Central service should use for deploying Central.
central.serviceTLS.key	The private key of the internal certificate that the Central service should use.
central.defaultTLS.cert	<p>The user-facing certificate that Central should use. RHACS uses this certificate for RHACS portal.</p> <ul style="list-style-type: none"> • For a new installation, you must provide a certificate, otherwise, RHACS installs Central by using a self-signed certificate. • If you are upgrading, RHACS uses the existing certificate and its key.
central.defaultTLS.key	<p>The private key of the user-facing certificate that Central should use.</p> <ul style="list-style-type: none"> • For a new installation, you must provide the private key, otherwise, RHACS installs Central by using a self-signed certificate. • If you are upgrading, RHACS uses the existing certificate and its key.
central.db.password.value	Connection password for Central database.
central.adminPassword.value	Administrator password for logging into RHACS.

Parameter	Description
central.adminPassword.htpasswd	Administrator password for logging into RHACS. This password is stored in hashed format using bcrypt.
central.db.serviceTLS.cert	An internal certificate that the Central DB service should use for deploying Central DB.
central.db.serviceTLS.key	The private key of the internal certificate that the Central DB service should use.
central.db.password.value	The password used to connect to the Central DB.



NOTE

If you are using **central.adminPassword.htpasswd** parameter, you must use a bcrypt encoded password hash. You can run the command **htpasswd -nB admin** to generate a password hash. For example,

```
htpasswd: |
admin:<bcrypt-hash>
```

5.2.1.2.1.4. Scanner

Configurable parameters for the StackRox Scanner and Scanner V4 (Technology Preview).



IMPORTANT

Scanner V4 is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see [Technology Preview Features Support Scope](#).

For a new installation, you can skip the following parameters and the Helm chart autogenerates values for them. Otherwise, if you are upgrading to a new version, specify the values for the following parameters:

- **scanner.dbPassword.value**
- **scanner.serviceTLS.cert**
- **scanner.serviceTLS.key**
- **scanner.dbServiceTLS.cert**

- **scanner.dbServiceTLS.key**
- **scannerV4.db.password.value**
- **scannerV4.indexer.serviceTLS.cert**
- **scannerV4.indexer.serviceTLS.key**
- **scannerV4.matcher.serviceTLS.cert**
- **scannerV4.matcher.serviceTLS.key**
- **scannerV4.db.serviceTLS.cert**
- **scannerV4.db.serviceTLS.key**

Parameter	Description
scanner.dbPassword.value	The password to use for authentication with Scanner database. Do not modify this parameter because RHACS automatically creates and uses its value internally.
scanner.serviceTLS.cert	An internal certificate that the StackRox Scanner service should use for deploying the StackRox Scanner.
scanner.serviceTLS.key	The private key of the internal certificate that the Scanner service should use.
scanner.dbServiceTLS.cert	An internal certificate that the Scanner-db service should use for deploying Scanner database.
scanner.dbServiceTLS.key	The private key of the internal certificate that the Scanner-db service should use.
scannerV4.db.password.value	The password to use for authentication with the Scanner V4 database. Do not modify this parameter because RHACS automatically creates and uses its value internally.
scannerV4.db.serviceTLS.cert	An internal certificate that the Scanner V4 DB service should use for deploying the Scanner V4 database.
scannerV4.db.serviceTLS.key	The private key of the internal certificate that the Scanner V4 DB service should use.
scannerV4.indexer.serviceTLS.cert	An internal certificate that the Scanner V4 service should use for deploying the Scanner V4 Indexer.
scannerV4.indexer.serviceTLS.key	The private key of the internal certificate that the Scanner V4 Indexer should use.
scannerV4.matcher.serviceTLS.cert	An internal certificate that the Scanner V4 service should use for deploying the the Scanner V4 Matcher.

Parameter	Description
scannerV4.matcher.serviceTLS.key	The private key of the internal certificate that the Scanner V4 Matcher should use.

5.2.1.2.2. Public configuration file

This section lists the configurable parameters of the **values-public.yaml** file.

5.2.1.2.2.1. Image pull secrets

Image pull secrets are the credentials required for pulling images from your registry.

Parameter	Description
imagePullSecrets.allowNone	Use true if you are using a custom registry and it allows pulling images without credentials.
imagePullSecrets.useExisting	A comma-separated list of secrets as values. For example, secret1, secret2 . Use this option if you have already created pre-existing image pull secrets with the given name in the target namespace.
imagePullSecrets.useFromDefaultServiceAccount	Use true if you have already configured the default service account in the target namespace with sufficiently scoped image pull secrets.

5.2.1.2.2.2. Image

Image declares the configuration to set up the main registry, which the Helm chart uses to resolve images for the **central.image**, **scanner.image**, **scanner.dbImage**, **scannerV4.image**, and **scannerV4.db.image** parameters.

Parameter	Description
image.registry	Address of your image registry. Either use a hostname, such as registry.redhat.io , or a remote registry hostname, such as us.gcr.io/stackrox-mirror .

5.2.1.2.2.3. Environment variables

Red Hat Advanced Cluster Security for Kubernetes automatically detects your cluster environment and sets values for **env.openshift**, **env.istio**, and **env.platform**. Only set these values to override the automatic cluster environment detection.

Parameter	Description
env.openshift	Use true for installing on an OpenShift Container Platform cluster and overriding automatic cluster environment detection.
env.istio	Use true for installing on an Istio enabled cluster and overriding automatic cluster environment detection.
env.platform	The platform on which you are installing RHACS. Set its value to default or gke to specify cluster platform and override automatic cluster environment detection.
env.offlineMode	Use true to use RHACS in offline mode.

5.2.1.2.2.4. Additional trusted certificate authorities

The RHACS automatically references the system root certificates to trust. When Central, the StackRox Scanner, or Scanner V4 must reach out to services that use certificates issued by an authority in your organization or a globally trusted partner organization, you can add trust for these services by specifying the root certificate authority to trust by using the following parameter:

Parameter	Description
additionalCAs.<certificate_name>	Specify the PEM encoded certificate of the root certificate authority to trust.

5.2.1.2.2.5. Central

Configurable parameters for Central.

- You must specify a persistent storage option as either **hostPath** or **persistentVolumeClaim**.
- For exposing Central deployment for external access. You must specify one parameter, either **central.exposure.loadBalancer**, **central.exposure.nodePort**, or **central.exposure.route**. When you do not specify any value for these parameters, you must manually expose Central or access it by using port-forwarding.

The following table includes settings for an external PostgreSQL database.

Parameter	Description
central.declarativeConfiguration.mounts.configMaps	Mounts config maps used for declarative configurations.
Central.declarativeConfiguration.mounts.secrets	Mounts secrets used for declarative configurations.

Parameter	Description
central.endpointsConfig	The endpoint configuration options for Central.
central.nodeSelector	If the node selector selects tainted nodes, use this parameter to specify a taint toleration key, value, and effect for Central. This parameter is mainly used for infrastructure nodes.
central.tolerations	If the node selector selects tainted nodes, use this parameter to specify a taint toleration key, value, and effect for Central. This parameter is mainly used for infrastructure nodes.
central.exposeMonitoring	Specify true to expose Prometheus metrics endpoint for Central on port number 9090 .
central.image.registry	A custom registry that overrides the global image.registry parameter for the Central image.
central.image.name	The custom image name that overrides the default Central image name (main).
central.image.tag	The custom image tag that overrides the default tag for Central image. If you specify your own image tag during a new installation, you must manually increment this tag when you to upgrade to a new version by running the helm upgrade command. If you mirror Central images in your own registry, do not modify the original image tags.
central.image.fullRef	Full reference including registry address, image name, and image tag for the Central image. Setting a value for this parameter overrides the central.image.registry , central.image.name , and central.image.tag parameters.
central.resources.requests.memory	The memory request for Central.
central.resources.requests.cpu	The CPU request for Central.
central.resources.limits.memory	The memory limit for Central.
central.resources.limits.cpu	The CPU limit for Central.
central.persistence.hostPath	The path on the node where RHACS should create a database volume. Red Hat does not recommend using this option.

Parameter	Description
central.persistence.persistentVolumeClaim.claimName	The name of the persistent volume claim (PVC) you are using.
central.persistence.persistentVolumeClaim.createClaim	Use true to create a new PVC, or false to use an existing claim.
central.persistence.persistentVolumeClaim.size	The size (in GiB) of the persistent volume managed by the specified claim.
central.exposure.loadBalancer.enabled	Use true to expose Central by using a load balancer.
central.exposure.loadBalancer.port	The port number on which to expose Central. The default port number is 443.
central.exposure.nodePort.enabled	Use true to expose Central by using the node port service.
central.exposure.nodePort.port	The port number on which to expose Central. When you skip this parameter, OpenShift Container Platform automatically assigns a port number. Red Hat recommends that you do not specify a port number if you are exposing RHACS by using a node port.
central.exposure.route.enabled	Use true to expose Central by using a route. This parameter is only available for OpenShift Container Platform clusters.
central.db.external	Use true to specify that Central DB should not be deployed and that an external database will be used.
central.db.source.connectionString	<p>The connection string for Central to use to connect to the database. This is only used when central.db.external is set to true. The connection string must be in keyword/value format as described in the PostgreSQL documentation in "Additional resources".</p> <ul style="list-style-type: none"> ● Only PostgreSQL 13 is supported. ● Connections through PgBouncer are not supported. ● User must be superuser with ability to create and delete databases.
central.db.source.minConns	The minimum number of connections to the database to be established.

Parameter	Description
central.db.source.maxConns	The maximum number of connections to the database to be established.
central.db.source.statementTimeoutMs	The number of milliseconds a single query or transaction can be active against the database.
central.db.postgresConfig	The postgresql.conf to be used for Central DB as described in the PostgreSQL documentation in "Additional resources".
central.db.hbaConfig	The pg_hba.conf to be used for Central DB as described in the PostgreSQL documentation in "Additional resources".
central.db.nodeSelector	Specify a node selector label as label-key: label-value to force Central DB to only schedule on nodes with the specified label.
central.db.image.registry	A custom registry that overrides the global image.registry parameter for the Central DB image.
central.db.image.name	The custom image name that overrides the default Central DB image name (central-db).
central.db.image.tag	The custom image tag that overrides the default tag for Central DB image. If you specify your own image tag during a new installation, you must manually increment this tag when you to upgrade to a new version by running the helm upgrade command. If you mirror Central DB images in your own registry, do not modify the original image tags.
central.db.image.fullRef	Full reference including registry address, image name, and image tag for the Central DB image. Setting a value for this parameter overrides the central.db.image.registry , central.db.image.name , and central.db.image.tag parameters.
central.db.resources.requests.memory	The memory request for Central DB.
central.db.resources.requests.cpu	The CPU request for Central DB.
central.db.resources.limits.memory	The memory limit for Central DB.

Parameter	Description
central.db.resources.limits.cpu	The CPU limit for Central DB.
central.db.persistence.hostPath	The path on the node where RHACS should create a database volume. Red Hat does not recommend using this option.
central.db.persistence.persistentVolumeClaim.claimName	The name of the persistent volume claim (PVC) you are using.
central.db.persistence.persistentVolumeClaim.createClaim	Use true to create a new persistent volume claim, or false to use an existing claim.
central.db.persistence.persistentVolumeClaim.size	The size (in GiB) of the persistent volume managed by the specified claim.

5.2.1.2.2.6. StackRox Scanner

The following table lists the configurable parameters for the StackRox Scanner. This is the scanner used for node and platform scanning. If Scanner V4 is not enabled, the StackRox scanner also performs image scanning. Beginning with version 4.4, Scanner V4 can be enabled to provide image scanning. See the next table for Scanner V4 parameters.

Parameter	Description
scanner.disable	Use true to install RHACS without the StackRox Scanner. When you use it with the helm upgrade command, Helm removes the existing StackRox Scanner deployment.
scanner.exposeMonitoring	Specify true to expose Prometheus metrics endpoint for the StackRox Scanner on port number 9090 .
scanner.replicas	The number of replicas to create for the StackRox Scanner deployment. When you use it with the scanner.autoscaling parameter, this value sets the initial number of replicas.
scanner.logLevel	Configure the log level for the StackRox Scanner. Red Hat recommends that you not change the default log level value (INFO).
scanner.nodeSelector	Specify a node selector label as label-key: label-value to force the StackRox Scanner to only schedule on nodes with the specified label.

Parameter	Description
scanner.tolerations	If the node selector selects tainted nodes, use this parameter to specify a taint toleration key, value, and effect for the StackRox Scanner. This parameter is mainly used for infrastructure nodes.
scanner.autoscaling.disable	Use true to disable autoscaling for the StackRox Scanner deployment. When you disable autoscaling, the minReplicas and maxReplicas parameters do not have any effect.
scanner.autoscaling.minReplicas	The minimum number of replicas for autoscaling.
scanner.autoscaling.maxReplicas	The maximum number of replicas for autoscaling.
scanner.resources.requests.memory	The memory request for the StackRox Scanner.
scanner.resources.requests.cpu	The CPU request for the StackRox Scanner.
scanner.resources.limits.memory	The memory limit for the StackRox Scanner.
scanner.resources.limits.cpu	The CPU limit for the StackRox Scanner.
scanner.dbResources.requests.memory	The memory request for the StackRox Scanner database deployment.
scanner.dbResources.requests.cpu	The CPU request for the StackRox Scanner database deployment.
scanner.dbResources.limits.memory	The memory limit for the StackRox Scanner database deployment.
scanner.dbResources.limits.cpu	The CPU limit for the StackRox Scanner database deployment.
scanner.image.registry	A custom registry for the StackRox Scanner image.
scanner.image.name	The custom image name that overrides the default StackRox Scanner image name (scanner).
scanner.dbImage.registry	A custom registry for the StackRox Scanner DB image.
scanner.dbImage.name	The custom image name that overrides the default StackRox Scanner DB image name (scanner-db).

Parameter	Description
scanner.dbNodeSelector	Specify a node selector label as label-key: label-value to force the StackRox 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 the StackRox Scanner DB. This parameter is mainly used for infrastructure nodes.

5.2.1.2.2.7. Scanner V4

The following table lists the configurable parameters for Scanner V4.



IMPORTANT

Scanner V4 is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see [Technology Preview Features Support Scope](#).

Parameter	Description
scannerV4.db.persistence.persistentVolumeClaim.claimName	The name of the PVC to manage persistent data for Scanner V4. If no PVC with the given name exists, it is created. The default value is scanner-v4-db if not set. To prevent data loss, the PVC is not removed automatically when Central is deleted.
scannerV4.disable	Use false to enable Scanner V4. When setting this parameter, the StackRox Scanner must also be enabled by setting scanner.disable=false . Until feature parity between the StackRox Scanner and Scanner V4 is reached, Scanner V4 can only be used in combination with the StackRox Scanner. Enabling Scanner V4 without also enabling the StackRox Scanner is not supported. When you set this parameter to true with the helm upgrade command, Helm removes the existing Scanner V4 deployment.
scannerV4.exposeMonitoring	Specify true to expose Prometheus metrics endpoint for Scanner V4 on port number 9090 .

Parameter	Description
scannerV4.indexer.replicas	The number of replicas to create for the Scanner V4 Indexer deployment. When you use it with the scannerV4.indexer.autoscaling parameter, this value sets the initial number of replicas.
scannerV4.indexer.logLevel	Configure the log level for the Scanner V4 Indexer. Red Hat recommends that you not change the default log level value (INFO).
scannerV4.indexer.nodeSelector	Specify a node selector label as label-key: label-value to force the Scanner V4 Indexer to only schedule on nodes with the specified label.
scannerV4.indexer.tolerations	If the node selector selects tainted nodes, use this parameter to specify a taint toleration key, value, and effect for the Scanner V4 Indexer. This parameter is mainly used for infrastructure nodes.
scannerV4.indexer.autoscaling.disable	Use true to disable autoscaling for the Scanner V4 Indexer deployment. When you disable autoscaling, the minReplicas and maxReplicas parameters do not have any effect.
scannerV4.indexer.autoscaling.minReplicas	The minimum number of replicas for autoscaling.
scannerV4.indexer.autoscaling.maxReplicas	The maximum number of replicas for autoscaling.
scannerV4.indexer.resources.requests.memory	The memory request for the Scanner V4 Indexer.
scannerV4.indexer.resources.requests.cpu	The CPU request for the Scanner V4 Indexer.
scannerV4.indexer.resources.limits.memory	The memory limit for the Scanner V4 Indexer.
scannerV4.indexer.resources.limits.cpu	The CPU limit for the Scanner V4 Indexer.
scannerV4.matcher.replicas	The number of replicas to create for the Scanner V4 Matcher deployment. When you use it with the scannerV4.matcher.autoscaling parameter, this value sets the initial number of replicas.
scannerV4.matcher.logLevel	Red Hat recommends that you not change the default log level value (INFO).
scannerV4.matcher.nodeSelector	Specify a node selector label as label-key: label-value to force the Scanner V4 Matcher to only schedule on nodes with the specified label.

Parameter	Description
scannerV4.matcher.tolerations	If the node selector selects tainted nodes, use this parameter to specify a taint toleration key, value, and effect for the Scanner V4 Matcher. This parameter is mainly used for infrastructure nodes.
scannerV4.matcher.autoscaling.disable	Use true to disable autoscaling for the Scanner V4 Matcher deployment. When you disable autoscaling, the minReplicas and maxReplicas parameters do not have any effect.
scannerV4.matcher.autoscaling.minReplicas	The minimum number of replicas for autoscaling.
scannerV4.matcher.autoscaling.maxReplicas	The maximum number of replicas for autoscaling.
scannerV4.matcher.resources.requests.memory	The memory request for the Scanner V4 Matcher.
scannerV4.matcher.resources.requests.cpu	The CPU request for the Scanner V4 Matcher.
scannerV4.db.resources.requests.memory	The memory request for the Scanner V4 database deployment.
scannerV4.db.resources.requests.cpu	The CPU request for the Scanner V4 database deployment.
scannerV4.db.resources.limits.memory	The memory limit for the Scanner V4 database deployment.
scannerV4.db.resources.limits.cpu	The CPU limit for the Scanner V4 database deployment.
scannerV4.db.nodeSelector	Specify a node selector label as label-key: label-value to force the Scanner V4 DB to only schedule on nodes with the specified label.
scannerV4.db.tolerations	If the node selector selects tainted nodes, use this parameter to specify a taint toleration key, value, and effect for the Scanner V4 DB. This parameter is mainly used for infrastructure nodes.
scannerV4.db.image.registry	A custom registry for the Scanner V4 DB image.
scannerV4.db.image.name	The custom image name that overrides the default Scanner V4 DB image name (scanner-v4-db).
scannerV4.image.registry	A custom registry for the Scanner V4 image.

Parameter	Description
scannerV4.image.name	The custom image name that overrides the default Scanner V4 image name (scanner-v4).

5.2.1.2.2.8. Customization

Use these parameters to specify additional attributes for all objects that RHACS creates.

Parameter	Description
customize.labels	A custom label to attach to all objects.
customize.annotations	A custom annotation to attach to all objects.
customize.podLabels	A custom label to attach to all deployments.
customize.podAnnotations	A custom annotation to attach to all deployments.
customize.envVars	A custom environment variable for all containers in all objects.
customize.central.labels	A custom label to attach to all objects that Central creates.
customize.central.annotations	A custom annotation to attach to all objects that Central creates.
customize.central.podLabels	A custom label to attach to all Central deployments.
customize.central.podAnnotations	A custom annotation to attach to all Central deployments.
customize.central.envVars	A custom environment variable for all Central containers.
customize.scanner.labels	A custom label to attach to all objects that Scanner creates.
customize.scanner.annotations	A custom annotation to attach to all objects that Scanner creates.
customize.scanner.podLabels	A custom label to attach to all Scanner deployments.
customize.scanner.podAnnotations	A custom annotation to attach to all Scanner deployments.

Parameter	Description
customize.scanner.envVars	A custom environment variable for all Scanner containers.
customize.scanner-db.labels	A custom label to attach to all objects that Scanner DB creates.
customize.scanner-db.annotations	A custom annotation to attach to all objects that Scanner DB creates.
customize.scanner-db.podLabels	A custom label to attach to all Scanner DB deployments.
customize.scanner-db.podAnnotations	A custom annotation to attach to all Scanner DB deployments.
customize.scanner-db.envVars	A custom environment variable for all Scanner DB containers.
customize.scanner-v4-indexer.labels	A custom label to attach to all objects that Scanner V4 Indexer creates and into the pods belonging to them.
customize.scanner-v4-indexer.annotations	A custom annotation to attach to all objects that Scanner V4 Indexer creates and into the pods belonging to them.
customize.scanner-v4-indexer.podLabels	A custom label to attach to all objects that Scanner V4 Indexer creates and into the pods belonging to them.
customize.scanner-v4-indexer.podAnnotations	A custom annotation to attach to all objects that Scanner V4 Indexer creates and into the pods belonging to them.
customize.scanner-v4-indexer.envVars	A custom environment variable for all Scanner V4 Indexer containers and the pods belonging to them.
customize.scanner-v4-matcher.labels	A custom label to attach to all objects that Scanner V4 Matcher creates and into the pods belonging to them.
customize.scanner-v4-matcher.annotations	A custom annotation to attach to all objects that Scanner V4 Matcher creates and into the pods belonging to them.

Parameter	Description
customize.scanner-v4-matcher.podLabels	A custom label to attach to all objects that Scanner V4 Matcher creates and into the pods belonging to them.
customize.scanner-v4-matcher.podAnnotations	A custom annotation to attach to all objects that Scanner V4 Matcher creates and into the pods belonging to them.
customize.scanner-4v-matcher.envVars	A custom environment variable for all Scanner V4 Matcher containers and the pods belonging to them.
customize.scanner-v4-db.labels	A custom label to attach to all objects that Scanner V4 DB creates and into the pods belonging to them.
customize.scanner-v4-db.annotations	A custom annotation to attach to all objects that Scanner V4 DB creates and into the pods belonging to them.
customize.scanner-v4-db.podLabels	A custom label to attach to all objects that Scanner V4 DB creates and into the pods belonging to them.
customize.scanner-v4-db.podAnnotations	A custom annotation to attach to all objects that Scanner V4 DB creates and into the pods belonging to them.
customize.scanner-4v-db.envVars	A custom environment variable for all Scanner V4 DB containers and the pods belonging to them.

You can also use:

- the **customize.other.service/*.labels** and the **customize.other.service/*.annotations** parameters, to specify labels and annotations for all objects.
- or, provide a specific service name, for example, **customize.other.service/central-loadbalancer.labels** and **customize.other.service/central-loadbalancer.annotations** as parameters and set their value.

5.2.1.2.2.9. Advanced customization



IMPORTANT

The parameters specified in this section are for information only. Red Hat does not support RHACS instances with modified namespace and release names.

Parameter	Description
-----------	-------------

Parameter	Description
allowNonstandardNamespace	Use true to deploy RHACS into a namespace other than the default namespace stackrox .
allowNonstandardReleaseName	Use true to deploy RHACS with a release name other than the default stackrox-central-services .

5.2.1.2.3. Declarative configuration values

To use declarative configuration, you must create a YAML file (in this example, named "declarative-config-values.yaml") that adds the declarative configuration mounts to Central. This file is used in a Helm installation.

Procedure

1. Create the YAML file (in this example, named **declarative-config-values.yaml**) using the following example as a guideline:

```
central:
  declarativeConfiguration:
    mounts:
      configMaps:
        - declarative-configs
      secrets:
        - sensitive-declarative-configs
```

2. Install the Central services Helm chart as documented in the "Installing the central-services Helm chart", referencing the **declarative-config-values.yaml** file.

Additional resources

- [Connection Strings - PostgreSQL Docs](#)
- [Parameter Interaction via the Configuration File - PostgreSQL Docs](#)
- [The pg_hba.conf File - PostgreSQL Docs](#)

5.2.1.2.4. Installing the central-services Helm chart

After you configure the **values-public.yaml** and **values-private.yaml** files, install the **central-services** Helm chart to deploy the centralized components (Central and Scanner).

Procedure

- Run the following command:

```
$ helm install -n stackrox --create-namespace \
  stackrox-central-services rhacs/central-services \
  -f <path_to_values_public.yaml> -f <path_to_values_private.yaml> 1
```

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

**NOTE**

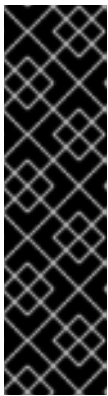
Optional: If using declarative configuration, add **-f <path_to_declarative-config-values.yaml>** to this command to mount the declarative configurations file in Central.

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

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

When using the **helm upgrade** command to make changes, the following guidelines and requirements apply:

- You can also specify configuration values using the **--set** or **--set-file** parameters. However, these options are not saved, and you must manually specify all the options again whenever you make changes.
- Some changes, such as enabling a new component like Scanner V4, require new certificates to be issued for the component. Therefore, you must provide a CA when making these changes.

**IMPORTANT**

Scanner V4 is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see [Technology Preview Features Support Scope](#).

- If the CA was generated by the Helm chart during the initial installation, you must retrieve these automatically generated values from the cluster and provide them to the **helm upgrade** command. The post-installation notes of the **central-services** Helm chart include a command for retrieving the automatically generated values.
- If the CA was generated outside of the Helm chart and provided during the installation of the **central-services** chart, then you must perform that action again when using the **helm upgrade** command, for example, by using the **--reuse-values** flag with the **helm upgrade** command.

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-central-services rhacs/central-services \
  --reuse-values 1 \
  -f <path_to_init_bundle_file> \
  -f <path_to_values_public.yaml> \
  -f <path_to_values_private.yaml>
```

1

If you have modified values that are not included in the **values_public.yaml** and **values_private.yaml** files, include the **--reuse-values** parameter.

5.2.2. Install Central using the roxctl CLI



WARNING

For production environments, Red Hat recommends using the Operator or Helm charts to install RHACS. Do not use the **roxctl** install method unless you have a specific installation need that requires using this method.

5.2.2.1. Installing the roxctl CLI

To install Red Hat Advanced Cluster Security for Kubernetes you must install the **roxctl** CLI by downloading the binary. You can install **roxctl** on Linux, Windows, or macOS.

5.2.2.1.1. 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/4.4.3/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
```

5.2.2.1.2. 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/4.4.3/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
```

5.2.2.1.3. 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/4.4.3/bin/Windows/roxctl.exe
```

Verification

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

```
$ roxctl version
```

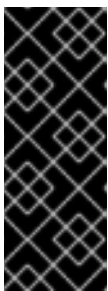
5.2.2.2. Using the interactive installer

Use the interactive installer to generate the required secrets, deployment configurations, and deployment scripts for your environment.

Procedure

1. Run the interactive install command:

```
$ roxctl central generate interactive
```



IMPORTANT

Installing RHACS using the **roxctl** CLI creates PodSecurityPolicy (PSP) objects by default for backward compatibility. If you install RHACS on Kubernetes versions 1.25 and newer or OpenShift Container Platform version 4.12 and newer, you must disable the PSP object creation. To do this, specify **--enable-pod-security-policies** option as **false** for the **roxctl central generate** and **roxctl sensor generate** commands.

2. Press **Enter** to accept the default value for a prompt or enter custom values as required. The following example shows the interactive installer prompts:

```
Enter path to the backup bundle from which to restore keys and certificates (optional):
Enter read templates from local filesystem (default: "false"):
Enter path to helm templates on your local filesystem (default: "/path"):
Enter PEM cert bundle file (optional): 1
Enter Create PodSecurityPolicy resources (for pre-v1.25 Kubernetes) (default: "true"): 2
Enter administrator password (default: autogenerated):
Enter orchestrator (k8s, openshift):
Enter default container images settings (development_build, stackrox.io, rhacs, opensource);
it controls repositories from where to download the images, image names and tags format
(default: "development_build"):
Enter the directory to output the deployment bundle to (default: "central-bundle"):
Enter the OpenShift major version (3 or 4) to deploy on (default: "0"):
Enter whether to enable telemetry (default: "false"):
Enter central-db image to use (if unset, a default will be used according to --image-defaults):
Enter Istio version when deploying into an Istio-enabled cluster (leave empty when not
running Istio) (optional):
Enter the method of exposing Central (route, lb, np, none) (default: "none"): 3
Enter main image to use (if unset, a default will be used according to --image-defaults):
Enter whether to run StackRox in offline mode, which avoids reaching out to the Internet
(default: "false"):
Enter list of secrets to add as declarative configuration mounts in central (default: "[]"): 4
Enter list of config maps to add as declarative configuration mounts in central (default: "[]"): 5
Enter the deployment tool to use (kubectl, helm, helm-values) (default: "kubectl"):
Enter scanner-db image to use (if unset, a default will be used according to --image-defaults):
Enter scanner image to use (if unset, a default will be used according to --image-defaults):
```

Enter Central volume type (hostpath, pvc): **6**
 Enter external volume name for Central (default: "stackrox-db"):
 Enter external volume size in Gi for Central (default: "100"):
 Enter storage class name for Central (optional if you have a default StorageClass configured):
 Enter external volume name for Central DB (default: "central-db"):
 Enter external volume size in Gi for Central DB (default: "100"):
 Enter storage class name for Central DB (optional if you have a default StorageClass configured):

- 1** If you want to add a custom TLS certificate, provide the file path for the PEM-encoded certificate. When you specify a custom certificate the interactive installer also prompts you to provide a PEM private key for the custom certificate you are using.
- 2** If you are running Kubernetes version 1.25 or later, set this value to **false**.
- 3** To use the RHACS portal, you must expose Central by using a route, a load balancer or a node port.
- 4** For more information on using declarative configurations for authentication and authorization, see "Declarative configuration for authentication and authorization resources" in "Managing RBAC in Red Hat Advanced Cluster Security for Kubernetes".
- 5** For more information on using declarative configurations for authentication and authorization, see "Declarative configuration for authentication and authorization resources" in "Managing RBAC in Red Hat Advanced Cluster Security for Kubernetes".
- 6** If you plan to install Red Hat Advanced Cluster Security for Kubernetes on OpenShift Container Platform with a hostPath volume, you must modify the SELinux policy.



WARNING

On OpenShift Container Platform, for using a hostPath volume, you must modify the SELinux policy to allow access to the directory, which the host and the container share. It is because SELinux blocks directory sharing by default. To modify the SELinux policy, run the following command:

```
$ sudo chcon -Rt svirt_sandbox_file_t <full_volume_path>
```

However, Red Hat does not recommend modifying the SELinux policy, instead use PVC when installing on OpenShift Container Platform.

On completion, the installer creates a folder named central-bundle, which contains the necessary YAML manifests and scripts to deploy Central. In addition, it shows on-screen instructions for the scripts you need to run to deploy additional trusted certificate authorities, Central and Scanner, and the authentication instructions for logging into the RHACS portal along with the autogenerated password if you did not provide one when answering the prompts.

5.2.2.3. Running the Central installation scripts

After you run the interactive installer, you can run the **setup.sh** script to install Central.

Procedure

1. Run the **setup.sh** script to configure image registry access:

```
$ ./central-bundle/central/scripts/setup.sh
```

2. Create the necessary resources:

```
$ oc create -R -f central-bundle/central
```

3. Check the deployment progress:

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

4. After Central is running, find the RHACS portal IP address and open it in your browser. Depending on the exposure method you selected when answering the prompts, use one of the following methods to get the IP address.

Exposure method	Command	Address	Example
Route	oc -n stackrox get route central	The address under the HOST/PORT column in the output	https://central-stackrox.example.route
Node Port	oc get node -owide && oc -n stackrox get svc central-loadbalancer	IP or hostname of any node, on the port shown for the service	https://198.51.100.0:31489
Load Balancer	oc -n stackrox get svc central-loadbalancer	EXTERNAL-IP or hostname shown for the service, on port 443	https://192.0.2.0
None	central-bundle/central/scripts/port-forward.sh 8443	https://localhost:8443	https://localhost:8443



NOTE

If you have selected autogenerated password during the interactive install, you can run the following command to see it for logging into Central:

```
$ cat central-bundle/password
```

5.3. GENERATING AND APPLYING AN INIT BUNDLE FOR RHACS ON OTHER PLATFORMS

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.

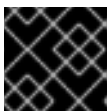


NOTE

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

Procedure

1. Find the address of the RHACS portal as described in "Verifying Central installation using the Operator method".
2. Log in to the RHACS portal.
3. If you do not have secured clusters, the **Platform Configuration → Clusters** page appears.
4. Click **Create init bundle**
5. Enter a name for the cluster init bundle.
6. Select your platform.
7. Select the installation method you will use for your secured clusters: **Operator** or **Helm chart**.
8. Click **Download** to generate and download the init bundle, which is created in the form of a YAML file. You can use one init bundle and its corresponding YAML file for all secured clusters if you are using the same installation method.



IMPORTANT

Store this bundle securely because it contains secrets.

9. Apply the init bundle by using it to create resources on the secured cluster.
10. 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:
 - Set the **ROX_API_TOKEN** by running the following command:

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

- Set the **ROX_CENTRAL_ADDRESS** environment variable by running the following command:

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

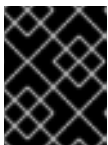
Procedure

- 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.

5.3.1.3. Applying the init bundle on the secured cluster

Before you configure a secured cluster, you must apply the init bundle by using it to create the required resources on the cluster. Applying the init bundle allows the services on the secured cluster to communicate with Central.

**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.
- You must have created the **stackrox** project, or namespace, on the cluster where secured cluster services will be installed. Using **stackrox** for the project is not required, but ensures that vulnerabilities for RHACS processes are not reported when scanning your clusters.

Procedure

To create resources, perform only one of the following steps:

- Create resources using the OpenShift Container Platform web console: In the OpenShift Container Platform web console, make sure that you are in the **stackrox** namespace. 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**. When the command is complete, the display shows that the **collector-tls**, **sensor-tls**, and **admission-control-tls** resources were created.
- Create resources using the Red Hat OpenShift CLI: Using the Red Hat OpenShift CLI, run the following command to create the resources:

```
$ oc create -f <init_bundle>.yaml \ 1
-n <stackrox> 2
```

- 1 Specify the file name of the init bundle containing the secrets.
- 2 Specify the name of the project where Central services are installed.

- 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**.

5.3.2. Next steps

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

5.4. INSTALLING SECURED CLUSTER SERVICES FOR RHACS ON OTHER PLATFORMS

You can install RHACS on your secured clusters for platforms such as Amazon Elastic Kubernetes Service (Amazon EKS), Google Kubernetes Engine (Google GKE), and Microsoft Azure Kubernetes Service (Microsoft AKS).

5.4.1. Installing RHACS 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.

5.4.1.1. Installing RHACS on secured clusters by using Helm charts without customizations

5.4.1.1.1. 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:

- Central services Helm chart (**central-services**) for installing the centralized components (Central and Scanner).



NOTE

You deploy centralized components only once and you can monitor multiple separate clusters by using the same installation.

- 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/
```

5.4.1.1.2. 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).

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 address and the port number that you are exposing the Central service on.

Additional resources

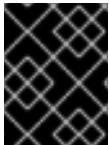
- [Generating and applying an init bundle for RHACS on other platforms](#)

5.4.1.2. 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.

5.4.1.2.1. 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.

Parameter	Description
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 CORE_BPF , EBPF (deprecated), or NO_COLLECTION .
collector.imagePullPolicy	Image pull policy for the Collector container.
collector.complianceImagePullPolicy	Image pull policy for the Compliance container.

Parameter	Description
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. RHACS does not support this feature on OpenShift Container Platform 3.11.
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 with the EBPF collection method requires Central to provide the matching eBPF probe. 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.
scanner.replicas	Resource specification for Collector's Compliance container.

Parameter	Description
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.
scanner.dbResources.requests.cpu	The CPU request for the Scanner DB container. Use this parameter to override the default value.

Parameter	Description
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.

5.4.1.2.1.1. Environment variables

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

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

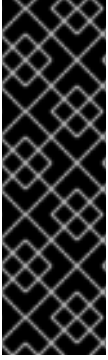
The **customize** setting allows you to specify custom Kubernetes metadata (labels and annotations) for all objects created by this Helm chart and additional pod labels, pod annotations, and container environment variables for workloads.

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).

5.4.1.2.2. Installing the secured-cluster-services Helm chart with customizations

After you configure the **values-public.yaml** and **values-private.yaml** files, install the **secured-cluster-services** Helm chart to deploy the following per-cluster and per-node components:

- Sensor
- Admission controller
- Collector
- Scanner: optional for secured clusters when the StackRox Scanner is installed
- Scanner DB: optional for secured clusters when the StackRox Scanner is installed
- Scanner V4 Indexer and Scanner V4 DB: optional for secured clusters when Scanner V4 is installed



IMPORTANT

Scanner V4 is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see [Technology Preview Features Support Scope](#).

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 address and the port number that you are exposing the Central service on.

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> ❶
--set imagePullSecrets.username=<username> ❷
--set imagePullSecrets.password=<password> ❸
```

- ❶ Use the **-f** option to specify the paths for your YAML configuration files.
- ❷ Include the user name for your pull secret for Red Hat Container Registry authentication.
- ❸ 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") ❶
```

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

Additional resources

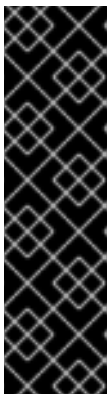
- [Generating and applying an init bundle for RHACS on other platforms](#)

5.4.1.3. 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.

When using the **helm upgrade** command to make changes, the following guidelines and requirements apply:

- You can also specify configuration values using the **--set** or **--set-file** parameters. However, these options are not saved, and you must manually specify all the options again whenever you make changes.
- Some changes, such as enabling a new component like Scanner V4, require new certificates to be issued for the component. Therefore, you must provide a CA when making these changes.



IMPORTANT

Scanner V4 is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see [Technology Preview Features Support Scope](#).

- If the CA was generated by the Helm chart during the initial installation, you must retrieve these automatically generated values from the cluster and provide them to the **helm upgrade** command. The post-installation notes of the **central-services** Helm chart include a command for retrieving the automatically generated values.
- If the CA was generated outside of the Helm chart and provided during the installation of the **central-services** chart, then you must perform that action again when using the **helm upgrade** command, for example, by using the **--reuse-values** flag with the **helm upgrade** command.

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** If you have modified values that are not included in the **values_public.yaml** and **values_private.yaml** files, include the **--reuse-values** parameter.

5.4.2. 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.

5.4.2.1. Installing the roxctl CLI

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

5.4.2.1.1. 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/4.4.3/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
```

5.4.2.1.2. 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/4.4.3/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
```

5.4.2.1.3. 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/4.4.3/bin/Windows/roxctl.exe
```

Verification

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

```
$ roxctl version
```

5.4.2.2. Installing Sensor

To monitor a cluster, you must deploy Sensor. You must deploy Sensor into each cluster that you want to monitor. This installation method is also called the manifest installation method.

To perform an installation by using the manifest installation method, follow *only one* of the following procedures:

- Use the RHACS web portal to download the cluster bundle, and then extract and run the sensor script.
- Use the **roxctl** CLI to generate the required sensor configuration for your OpenShift Container Platform cluster and associate it with your Central instance.

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).

5.4.2.2.1. Manifest installation method by using the web portal

Procedure

1. On your secured cluster, in the RHACS portal, go to **Platform Configuration → Clusters**.
2. Select **Secure a cluster → Legacy installation method**.
3. Specify a name for the cluster.
4. Provide appropriate values for the fields based on where you are deploying the Sensor.
 - If you are deploying Sensor in the same cluster, accept the default values for all the fields.
 - If you are deploying into a different cluster, replace **central.stackrox.svc:443** with a load balancer, node port, or other address, including the port number, that is accessible from the other cluster.
 - If you are using a non-gRPC capable load balancer, such as HAProxy, AWS Application Load Balancer (ALB), or AWS Elastic Load Balancing (ELB), use the WebSocket Secure (**wss**) protocol. To use **wss**:
 - Prefix the address with **wss://**.
 - Add the port number after the address, for example, **wss://stackrox-central.example.com:443**.
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, extract 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 help.

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

5.4.2.2.2. Manifest installation by using the roxctl CLI

Procedure

1. Generate the required sensor configuration for your OpenShift Container Platform cluster and associate it with your Central instance by running the following command:

```
$ roxctl sensor generate openshift --openshift-version <ocp_version> --name  
<cluster_name> --central "$ROX_ENDPOINT" 1
```

- 1 For the **--openshift-version** option, specify the major OpenShift Container Platform version number for your cluster. For example, specify **3** for OpenShift Container Platform version **3.x** and specify **4** for OpenShift Container Platform version **4.x**.

2. From a system that has access to the monitored cluster, extract 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 help.

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.

5.5. VERIFYING INSTALLATION OF RHACS ON OTHER PLATFORMS

Provides steps to verify that RHACS is properly installed.

5.5.1. Verifying installation

After you complete the installation, run a few vulnerable applications and go to the RHACS portal to evaluate the results of security assessments and policy violations.



NOTE

The sample applications listed in the following section contain critical vulnerabilities and they are specifically designed to verify the build and deploy-time assessment features of Red Hat Advanced Cluster Security for Kubernetes.

To verify installation:

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

- a. For a load balancer:

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

- b. For port forward:

- i. Run the following command:

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

- ii. Go to **<https://localhost:18443/>**.

2. Create a new namespace:

```
$ kubectl create namespace test
```

3. Start some applications with critical vulnerabilities:

```
$ kubectl run shell --labels=app=shellshock,team=test-team \
  --image=quay.io/stackrox-io/docs:example-vulnerables-cve-2014-6271 -n test
$ kubectl run samba --labels=app=rce \
  --image=quay.io/stackrox-io/docs:example-vulnerables-cve-2017-7494 -n test
```

Red Hat Advanced Cluster Security for Kubernetes automatically scans these deployments for security risks and policy violations as soon as they are submitted to the cluster. Go to the RHACS portal to view the violations. You can log in to the RHACS portal by using the default username **admin** and the generated password.

CHAPTER 6. UNINSTALLING RED HAT ADVANCED CLUSTER SECURITY FOR KUBERNETES

When you install Red Hat Advanced Cluster Security for Kubernetes, it creates:

- A namespace called **rhacs-operator** where the Operator is installed, if you chose the Operator method of installation
- A namespace called **stackrox**, or another namespace where you created the Central and SecuredCluster custom resources
- **PodSecurityPolicy** and Kubernetes role-based access control (RBAC) objects for all components
- Additional labels on namespaces, for use in generated network policies
- An application custom resource definition (CRD), if it does not exist

Uninstalling Red Hat Advanced Cluster Security for Kubernetes involves deleting all of these items.

6.1. DELETING NAMESPACE

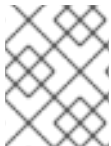
You can delete the namespace that Red Hat Advanced Cluster Security for Kubernetes creates by using the OpenShift Container Platform or Kubernetes command-line interface.

Procedure

- Delete the **stackrox** namespace:
 - On OpenShift Container Platform:


```
$ oc delete namespace stackrox
```
 - On Kubernetes:


```
$ kubectl delete namespace stackrox
```



NOTE

If you installed RHACS in a different namespace, use the name of that namespace in the **delete** command.

6.2. DELETING GLOBAL RESOURCES

You can delete the global resources that Red Hat Advanced Cluster Security for Kubernetes creates, by using the OpenShift Container Platform or Kubernetes command-line interface.

Procedure

- Delete global resources:
 - On OpenShift Container Platform:


```
$ oc get clusterrole,clusterrolebinding,role,rolebinding,psp -o name | grep stackrox |
xargs oc delete --wait
```

```
$ oc delete scc -l "app.kubernetes.io/name=stackrox"
```

```
$ oc delete ValidatingWebhookConfiguration stackrox
```

- On Kubernetes:

```
$ kubectl get clusterrole,clusterrolebinding,role,rolebinding,psp -o name | grep stackrox |
xargs kubectl delete --wait
```

```
$ kubectl delete ValidatingWebhookConfiguration stackrox
```

6.3. DELETING LABELS AND ANNOTATIONS

You can delete the labels and annotations that Red Hat Advanced Cluster Security for Kubernetes creates, by using the OpenShift Container Platform or Kubernetes command-line interface.

Procedure

- Delete labels and annotations:

- On OpenShift Container Platform:

```
$ for namespace in $(oc get ns | tail -n +2 | awk '{print $1}'); do   oc label namespace
$namespace namespace.metadata.stackrox.io/id-;   oc label namespace $namespace
namespace.metadata.stackrox.io/name-;   oc annotate namespace $namespace
modified-by.stackrox.io/namespace-label-patcher-; done
```

- On Kubernetes:

```
$ for namespace in $(kubectl get ns | tail -n +2 | awk '{print $1}'); do   kubectl label
namespace $namespace namespace.metadata.stackrox.io/id-;   kubectl label
namespace $namespace namespace.metadata.stackrox.io/name-;   kubectl annotate
namespace $namespace modified-by.stackrox.io/namespace-label-patcher-; done
```