Red Hat Advanced Cluster Security for Kubernetes 4.4

Upgrading

Upgrading Red Hat Advanced Cluster Security for Kubernetes

Last Updated: 2024-06-27

Red Hat Advanced Cluster Security for Kubernetes 4.4 Upgrading

Upgrading Red Hat Advanced Cluster Security for Kubernetes

Legal Notice

Abstract

This section provides instructions on upgrading Red Hat Advanced Cluster Security for Kubernetes by using Helm charts or the roxctl command-line interface.

Table of Contents

CHAPTER 1. UPGRADING BY USING THE OPERATOR 4
1.1. PREPARING TO UPGRADE 4
1.1.1. Changing the collection method 4
1.1.2. Setting the forceCollection parameter 5
1.2. MODIFYING CENTRAL CUSTOM RESOURCE 5
1.3. MODIFYING CENTRAL CUSTOM RESOURCE FOR EXTERNAL DATABASE 6
1.4. CHANGING SUBSCRIPTION CHANNEL 7
Changing the subscription channel by using the web console 7
Changing the subscription channel by using command line 8
1.5. REMOVING CENTRAL-ATTACHED PV AFTER UPGRADING TO VERSION 4.1 AND LATER 8
1.5.1. Removing Central-attached PV using the RHACS Operator for RHACS version 4.1 and later 9
1.6. ROLLING BACK AN OPERATOR UPGRADE 9
1.6.1. Rolling back an Operator upgrade by using the CLI 9
1.6.2. Rolling back an Operator upgrade by using the web console 12
1.7. TROUBLESHOOTING OPERATOR UPGRADE ISSUES 14
1.7.1. Central DB cannot be scheduled 14
1.7.2. Central or Secured cluster fails to deploy 14

CHAPTER 2. UPGRADING USING HELM CHARTS 17
2.1. UPGRADE SEQUENCE FROM RHACS RELEASE 3.74 AND EARLIER 17
2.2. BACKING UP THE CENTRAL DATABASE 17
2.3. OPTIMIZING CENTRAL DATABASE AND PVC 18
2.4. GENERATING ROOT CERTIFICATES FILE 18
2.5. UPDATING THE HELM CHART REPOSITORY 18
2.6. ADDITIONAL RESOURCES 19
2.7. RUNNING THE HELM UPGRADE COMMAND 19
2.8. REMOVING CENTRAL-ATTACHED PV AFTER UPGRADING TO VERSION 4.1 AND LATER 20
2.8.1. Removing Central-attached PV using Helm 20
2.9. ROLLING BACK A HELM UPGRADE 21

CHAPTER 3. MANUALLY UPGRADING USING THE ROXCTL CLI 22
3.1. BACKING UP THE CENTRAL DATABASE 22
3.2. UPGRADING THE ROXCTL CLI 23
3.2.1. Uninstalling the roxctl CLI 23
3.2.2. Installing the roxctl CLI on Linux 23
3.2.3. Installing the roxctl CLI on macOS 24
3.2.4. Installing the roxctl CLI on Windows 24
3.3. GENERATING CENTRAL DATABASE PROVISIONING BUNDLE 25
3.4. CREATING RESOURCES BY USING THE CENTRAL DB PROVISIONING BUNDLE 25
3.5. UPGRADING THE CENTRAL CLUSTER 26
3.5.1. Upgrading Central 26
3.5.1.1. Editing the GOMEMLIMIT environment variable for the Central deployment 26
3.5.2. Upgrading Scanner 27
3.5.2.1. Editing the GOMEMLIMIT environment variable for the Scanner deployment 27
3.5.3. Verifying the Central cluster upgrade 27
3.6. UPGRADING ALL SECURED CLUSTERS 28
3.6.1. Updating other images 28
3.6.2. Migrating SCCs during the manual upgrade 30
3.6.2.1. Editing the GOMEMLIMIT environment variable for the Sensor deployment 36
3.6.2.2. Editing the GOMEMLIMIT environment variable for the Collector deployment 36
3.6.2.3. Editing the GOMEMLIMIT environment variable for the Admission Controller deployment 37

3.6.2.4. Verifying secured cluster upgrade 37
3.7. ENABLING RHCOS NODE SCANNING 37
3.8. REMOVING CENTRAL-ATTACHED PV AFTER UPGRADING TO VERSION 4.1 AND LATER 39
3.8.1. Removing Central-attached PV using the roxctl CLI 39
3.9. ROLLING BACK CENTRAL 40
3.9.1. Rolling back Central normally 40
3.9.2. Rolling back Central forcefully 40
3.10. VERIFYING UPGRADES 41
3.11. REVOKING THE API TOKEN 42

CHAPTER 1. UPGRADING BY USING THE OPERATOR

Upgrades through the Red Hat Advanced Cluster Security for Kubernetes (RHACS) Operator are performed automatically or manually, depending on the **Update approval** option you chose at installation.

Follow these guidelines when upgrading:

- If the version for Central is earlier than 3.74, you must upgrade to 3.74 before upgrading to a 4.x version. For upgrading Central to version 3.74, see the upgrade documentation for version 3.74.
- When upgrading Operator-based Central deployments from version 3.74, first ensure the Operator upgrade mode is set to **Manual**. Then, upgrade the Operator to version 4.0 following the procedure in the upgrade documentation for version 4.0 and ensure that Central is online. After the upgrade to version 4.0 is complete, Red Hat recommends upgrading Central to the latest version for full functionality.

1.1. PREPARING TO UPGRADE

Before you upgrade the Red Hat Advanced Cluster Security for Kubernetes (RHACS) version, you must perform the following steps:

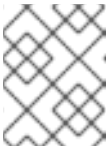
- If you are upgrading from version 3.74, verify that you are running the latest patch release version of the RHACS Operator 3.74.
- Backup your existing Central database.
- If the cluster you are upgrading contains the **SecuredCluster** custom resource (CR), change the collection method to EBPF or CORE_BPF. For more information, see "Changing the collection method".

1.1.1. Changing the collection method

If the cluster that you are upgrading contains the **SecuredCluster** CR, you must ensure that the per node collection setting is set to **CORE_BPF** before you upgrade, if you are upgrading from 4.1 or later. Otherwise, set the collection method to **EBPF**. To set the collection method to **EBPF**, you must set the **forceCollection** parameter to **true** after the upgrade and make sure that the collection method is **EBPF**.

Procedure

1. In the OpenShift Container Platform web console, go to the RHACS Operator page.
2. In the top navigation menu, select **Secured Cluster**.
3. Click the instance name, for example, **stackrox-secured-cluster-services**.
4. Use one of the following methods to change the setting:
   - In the **Form view**, under **Per Node Settings → Collector Settings → Collection**, select **CORE_BPF**.
   - Click **YAML** to open the YAML editor and locate the **spec.perNode.collector.collection** attribute. If the value is **KernelModule**, then change it to **CORE_BPF**.



NOTE

Only use **EBPF** if you are upgrading from a version earlier than 4.1 or if there is a specific reason to use it.

5. Click **Save.**

1.1.2. Setting the forceCollection parameter

When upgrading secured clusters, if you set the collection method to **EBPF**, you must set the **forceCollection** parameter to **true** after the upgrade. Then, make sure that the **spec.perNode.collector.collection** is still set to **EBPF** in the YAML editor.

Procedure

1. In the OpenShift Container Platform web console, go to the RHACS Operator page.
2. In the top navigation menu, select **Secured Cluster**.
3. Click the instance name, for example, **stackrox-secured-cluster-services**.
4. Click **YAML** to open the YAML editor.
5. Locate the **spec.perNode.collector.forceCollection** parameter and set it to **true**.
6. Click **Save.**

Additional resources

- Updating installed Operators
- Backing up Red Hat Advanced Cluster Security for Kubernetes

1.2. MODIFYING CENTRAL CUSTOM RESOURCE

The Central DB service requires persistent storage. If you have not configured a default storage class for the Central cluster that is an SSD or is high performance, you must update the **Central** custom resource to configure the storage class for the Central DB persistent volume claim (PVC).

NOTE

Skip this section if you have already configured a default storage class for Central.

Procedure

- Update the central custom resource with the following configuration:

```
spec:
  central:
    db:
      isEnabled: Default (1)
      persistence:
        persistentVolumeClaim: (2)
```

```
    claimName: central-db
    size: 100Gi
    storageClassName: <storage-class-name>
```

1. You must not change the value of **IsEnabled** to **Enabled**.
2. If this claim exists, your cluster uses the existing claim, otherwise it creates a new claim.

1.3. MODIFYING CENTRAL CUSTOM RESOURCE FOR EXTERNAL DATABASE

Prerequisites

- 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. 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 \ 1
       --from-file=password=<password.txt> 2
     ```

     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.

2. Go to the Red Hat Advanced Cluster Security for Kubernetes operator page in the OpenShift Container Platform web console. Select **Central** in the top navigation bar and select the instance you want to connect to the database.
3. Go to the **YAML editor** view.
4. For **db.passwordSecret.name** specify the referenced secret that you created in earlier steps. For example, **external-db-password**.
5. For **db.connectionString** specify the connection string in **keyword=value** format, for example, **host=<host> port=5432 database=stackrox user=stackrox sslmode=verify-ca**

6. For **db.persistence** delete the entire block.

7. If necessary, you can specify a Certificate Authority for Central to trust the database certificate by adding a TLS block under the top-level spec, as shown in the following example:

   - Update the central custom resource with the following configuration:

```
spec:
  tls:
    additionalCAs:
    - name: db-ca
      content: |
        <certificate>
  central:
    db:
      isEnabled: Default (1)
      connectionString: "host=<host> port=5432 user=<user> sslmode=verify-ca"
      passwordSecret:
        name: external-db-password
```

   (1) You must not change the value of **IsEnabled** to **Enabled**.

8. Click **Save**.

Additional resources

- Provisioning a database in your PostgreSQL instance

1.4. CHANGING SUBSCRIPTION CHANNEL

You can change the update channel for the RHACS Operator by using the OpenShift Container Platform web console or by using the command line. For upgrading to RHACS 4.0 from RHACS 3.74, you must change the update channel.

IMPORTANT

You must change the subscription channel for all clusters where you have installed RHACS Operator, including Central and all Secured clusters.

Prerequisites

- You must verify that you are using the latest RHACS 3.74 Operator and there are no pending manual Operator upgrades.
- You must verify that you have backed up your existing Central database.
- You have access to an OpenShift Container Platform cluster web console using an account with **cluster-admin** permissions.

Changing the subscription channel by using the web console

Use the following instructions for changing the subscription channel by using the web console:

Procedure

1. In the **Administrator** perspective of the OpenShift Container Platform web console, go to **Operators → Installed Operators**.
2. Locate the RHACS Operator and click on it.
3. Click the **Subscription** tab.
4. Click the name of the update channel under **Update Channel**.
5. Select **stable**, then click **Save**.
6. For subscriptions with an **Automatic** approval strategy, the update begins automatically. Go back to the **Operators → Installed Operators** page to monitor the progress of the update. When complete, the status changes to **Succeeded** and **Up to date**.
   For subscriptions with a **Manual** approval strategy, you can manually approve the update from the **Subscription** tab.

Changing the subscription channel by using command line

Use the following instructions for changing the subscription channel by using command line:

Procedure

- Run the following command to change the subscription channel to **stable**:

```
$ oc -n rhacs-operator \ 1
  patch subscriptions.operators.coreos.com rhacs-operator \
  --type=merge --patch='{ "spec": { "channel": "stable" }}'
```

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

During the update the RHACS Operator provisions a new deployment called **central-db** and your data begins migrating. It takes around 30 minutes and only happens once when you upgrade.

1.5. REMOVING CENTRAL-ATTACHED PV AFTER UPGRADING TO VERSION 4.1 AND LATER

Kubernetes and OpenShift Container Platform do not delete persistent volumes (PV) automatically. When you upgrade RHACS from earlier versions, the Central PV called **stackrox-db** remains mounted. However, in RHACS 4.1, Central does not need the previously attached PV anymore.

The PV has data and persistent files used by earlier RHACS versions. You can use the PV to roll back to an earlier version before RHACS 4.1. Or, if you have a large RocksDB backup bundle for Central, you can use the PV to restore that data.

After you complete the upgrade to 4.1, you can remove the Central-attached persistent volume claim (PVC) to free up the storage. Only remove the PVC if you do not plan to roll back or restore from earlier RocksDB backups.

WARNING

After removing PVC, you cannot roll back Central to an earlier version before RHACS 4.1 or restore large RocksDB backups created with RocksDB.

1.5.1. Removing Central-attached PV using the RHACS Operator for RHACS version 4.1 and later

Remove the Central-attached persistent volume claim (PVC) **stackrox-db** to free up storage space.

Procedure

- Add the following annotation to Central:

```
annotations:
  platform.stackrox.io/obsolete-central-pvc: "true"
```

Verification

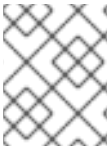
- Run the following command:

```
$ oc -n stackrox describe pvc stackrox-db | grep -i 'Used By'
Used By: <none> 1
```

1 Wait until you see **Used By: <none>**. It might take a few minutes.

1.6. ROLLING BACK AN OPERATOR UPGRADE

To roll back an Operator upgrade, you must perform the steps described in one of the following sections. You can roll back an Operator upgrade by using the CLI or the OpenShift Container Platform web console.



NOTE

If you are rolling back from RHACS 4.0, you can only rollback to the latest patch release version of RHACS 3.74.

1.6.1. Rolling back an Operator upgrade by using the CLI

You can roll back the Operator version by using CLI commands.

Procedure

1. Delete the OLM subscription by running the following command:
   - For OpenShift Container Platform, run the following command:

```
$ oc -n rhacs-operator delete subscription rhacs-operator
```

- For Kubernetes, run the following command:

```
$ kubectl -n rhacs-operator delete subscription rhacs-operator
```

2. Delete the cluster service version (CSV) by running the following command:
   - For OpenShift Container Platform, run the following command:

```
$ oc -n rhacs-operator delete csv -l operators.coreos.com/rhacs-operator.rhacs-operator
```

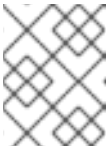
   - For Kubernetes, run the following command:

```
$ kubectl -n rhacs-operator delete csv -l operators.coreos.com/rhacs-operator.rhacs-operator
```

3. Determine the previous version you want to roll back to by choosing one of the following options:
   - If the current Central instance is running, query the RHACS API to get the rollback version by running the following command:

```
$ curl -k -s -u <user>:<password> https://<central hostname>/v1/centralhealth/upgradestatus | jq -r .upgradeStatus.forceRollbackTo
```

   - If the current Central instance is not running, perform the following steps:



NOTE

This procedure can only be used for RHACS release 3.74 and earlier when the **rocksdb** database is installed.

     a. Ensure the Central deployment is scaled down by running the following command:
        - For OpenShift Container Platform, run the following command:

```
$ oc scale -n <central namespace> –replicas=0 deploy/central
```

        - For Kubernetes, run the following command:

```
$ kubectl scale -n <central namespace> –replicas=0 deploy/central
```

     b. Save the following pod spec as a YAML file:

```
apiVersion: v1
kind: Pod
metadata:
  name: get-previous-db-version
spec:
  containers:
  - name: get-previous-db-version
    image: registry.redhat.io/advanced-cluster-security/rhacs-main-rhel8:<rollback version>
```

```
    command:
    - sh
    args:
    - '-c'
    - "cat /var/lib/stackrox/.previous/migration_version.yaml | grep '^image:' | cut -f 2 -d
: | tr -d ' '"
    volumeMounts:
    - name: stackrox-db
      mountPath: /var/lib/stackrox
  volumes:
  - name: stackrox-db
    persistentVolumeClaim:
      claimName: stackrox-db
```

c. Create a pod in your Central namespace by running the following command using the YAML file that you saved:

- For OpenShift Container Platform, run the following command:

```
$ oc create -n <central namespace> -f pod.yaml
```

- For Kubernetes, run the following command:

```
$ kubectl create -n <central namespace> -f pod.yaml
```

d. After pod creation is complete, get the version by running the following command:

- For OpenShift Container Platform, run the following command:

```
$ oc logs -n <central namespace> get-previous-db-version
```

- For Kubernetes, run the following command:

```
$ kubectl logs -n <central namespace> get-previous-db-version
```

4. Edit the **central-config.yaml ConfigMap** to set the **maintenance.forceRollBackVersion: <version>** parameter by running the following command:

- For OpenShift Container Platform, run the following command:

```
$ oc get configmap -n <central namespace> central-config -o yaml | sed -e
"s/forceRollbackVersion: none/forceRollbackVersion: <version>/" | oc -n <central
namespace> apply -f -
```

- For Kubernetes, run the following command:

```
$ kubectl get configmap -n <central namespace> central-config -o yaml | sed -e
"s/forceRollbackVersion: none/forceRollbackVersion: <version>/" | kubectl -n <central
namespace> apply -f -
```

5. Set the image for the Central deployment using the version string shown in Step 3 as the image tag. For example, run the following command:

- For OpenShift Container Platform, run the following command:

```
$ oc set image -n <central namespace> deploy/central
central=registry.redhat.io/advanced-cluster-security/rhacs-main-rhel8:<version>
```

- For Kubernetes, run the following command:

```
$ kubectl set image -n <central namespace> deploy/central
central=registry.redhat.io/advanced-cluster-security/rhacs-main-rhel8:<version>
```

Verification

1. Ensure that the Central pod starts and has a **ready** status. If the pod crashes, check the logs to see if the backup was restored. A successful log message appears similar to the following example:

```
Clone to Migrate ".previous", ""
```

2. Reinstall the Operator on the rolled back channel. For example, **3.74.2** is installed on the **rhacs-3.74** channel.

1.6.2. Rolling back an Operator upgrade by using the web console

You can roll back the Operator version by using the OpenShift Container Platform web console.

Prerequisites

- You have access to an OpenShift Container Platform cluster web console using an account with **cluster-admin** permissions.

Procedure

1. Go to the **Operators → Installed Operators** page.
2. Locate the RHACS Operator and click on it.
3. On the **Operator Details** page, select **Uninstall Operator** from the **Actions** list. Following this action, the Operator stops running and no longer receives updates.
4. Determine the previous version you want to roll back to by choosing one of the following options:
   - If the current Central instance is running, you can query the RHACS API to get the rollback version by running the following command from a terminal window:

```
$ curl -k -s -u <user>:<password> https://<central
hostname>/v1/centralhealth/upgradestatus | jq -r .upgradeStatus.forceRollbackTo
```

   - You can create a pod and extract the previous version by performing the following steps:

NOTE

This procedure can only be used for RHACS release 3.74 and earlier when the **rocksdb** database is installed.

a. Go to **Workloads → Deployments → central**.

b. Under **Deployment details**, click the down arrow next to the pod count to scale down the pod.

c. Go to **Workloads → Pods → Create Pod** and paste the contents of the pod spec as shown in the following example into the editor:

```
apiVersion: v1
kind: Pod
metadata:
  name: get-previous-db-version
spec:
  containers:
  - name: get-previous-db-version
    image: registry.redhat.io/advanced-cluster-security/rhacs-main-rhel8:<rollback
version>
    command:
    - sh
    args:
    - '-c'
    - "cat /var/lib/stackrox/.previous/migration_version.yaml | grep '^image:' | cut -f 2 -d
: | tr -d ' '"
    volumeMounts:
    - name: stackrox-db
      mountPath: /var/lib/stackrox
  volumes:
  - name: stackrox-db
    persistentVolumeClaim:
      claimName: stackrox-db
```

d. Click **Create**.

e. After the pod is created, click the **Logs** tab to get the version string.

5. Update the rollback configuration by performing the following steps:

a. Go to **Workloads → ConfigMaps → central-config** and select **Edit ConfigMap** from the **Actions** list.

b. Find the **forceRollbackVersion** line in the value of the **central-config.yaml** key.

c. Replace **none** with **3.73.3**, and then save the file.

6. Update Central to the earlier version by performing the following steps:

a. Go to **Workloads → Deployments → central** and select **Edit Deployment** from the Actions list.

b. Update the image name, and then save the changes.

Verification

1. Ensure that the Central pod starts and has a **ready** status. If the pod crashes, check the logs to see if the backup was restored. A successful log message appears similar to the following example:

```
Clone to Migrate ".previous", ""
```

2. Reinstall the Operator on the rolled back channel. For example, **3.74.2** is installed on the **rhacs-3.74** channel.

Additional resources

- Installing Central using the Operator method
- Operator Lifecycle Manager workflow
- Manually approving a pending Operator update

1.7. TROUBLESHOOTING OPERATOR UPGRADE ISSUES

Follow these instructions to investigate and resolve upgrade-related issues for the RHACS Operator.

1.7.1. Central DB cannot be scheduled

Follow the instructions here to troubleshoot a failing Central DB pod during an upgrade:

1. Check the status of the **central-db** pod:

```
$ oc -n <namespace> get pod -l app=central-db (1)
```

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

2. If the status of the pod is **Pending**, use the describe command to get more details:

```
$ oc -n <namespace> describe po/<central-db-pod-name> (1)
```

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

3. You might see the **FailedScheduling** warning message:

```
Type     Reason            Age   From               Message
----     ------            ----  ----               -------
Warning  FailedScheduling  54s   default-scheduler  0/7 nodes are available: 1 Insufficient memory, 3 node(s) had untolerated taint {node-role.kubernetes.io/master: }, 4 Insufficient cpu. preemption: 0/7 nodes are available: 3 Preemption is not helpful for scheduling, 4 No preemption victims found for incoming pod.
```

4. This warning message suggests that the scheduled node had insufficient memory to accommodate the pod's resource requirements. If you have a small environment, consider increasing resources on the nodes or adding a larger node that can support the database. Otherwise, consider decreasing the resource requirements for the **central-db** pod in the custom resource under **central → db → resources**. However, running central with fewer resources than the recommended minimum might lead to degraded performance for RHACS.

1.7.2. Central or Secured cluster fails to deploy

When RHACS Operator:

- fails to deploy Central or Secured Cluster.
- fails to apply CR changes to actual resources.

You must check the custom resource conditions to find the issue.

- For Central, run the following command to check the conditions:

```
$ oc -n rhacs-operator describe centrals.platform.stackrox.io 1
```

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

- For Secured clusters, run the following command to check the conditions:

```
$ oc -n rhacs-operator describe securedclusters.platform.stackrox.io 1
```

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

You can identify configuration errors from the conditions output:

Example output

```
 Conditions:
   Last Transition Time:  2023-04-19T10:49:57Z
   Status:                False
   Type:                  Deployed
   Last Transition Time:  2023-04-19T10:49:57Z
   Status:                True
   Type:                  Initialized
   Last Transition Time:  2023-04-19T10:59:10Z
   Message:               Deployment.apps "central" is invalid:
spec.template.spec.containers[0].resources.requests: Invalid value: "50": must be less than or equal
to cpu limit
   Reason:                ReconcileError
   Status:                True
   Type:                  Irreconcilable
   Last Transition Time:  2023-04-19T10:49:57Z
   Message:               No proxy configuration is desired
   Reason:                NoProxyConfig
   Status:                False
   Type:                  ProxyConfigFailed
   Last Transition Time:  2023-04-19T10:49:57Z
   Message:               Deployment.apps "central" is invalid:
spec.template.spec.containers[0].resources.requests: Invalid value: "50": must be less than or equal
to cpu limit
   Reason:                InstallError
   Status:                True
   Type:                  ReleaseFailed
```

Additionally, you can view RHACS pod logs to find more information about the issue. Run the following command to view the logs:

```
oc -n rhacs-operator logs deploy/rhacs-operator-controller-manager manager (1)
```

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

CHAPTER 2. UPGRADING USING HELM CHARTS

You must follow a specific upgrade path for RHACS depending on the release of RHACS that you are running. You must also back up your Central database before updating the Helm chart and peforming the upgrade.

2.1. UPGRADE SEQUENCE FROM RHACS RELEASE 3.74 AND EARLIER

When upgrading from earlier releases, follow this guidance:

- If the release for Central is earlier than 3.74, you must upgrade to the latest 3.74 patch before upgrading to a 4.x release. See the upgrade documentation for version 3.74 for information about upgrades from earlier versions to 3.74.
- When upgrading Helm-based installations from release 3.74, you can upgrade to any latest patch of RHACS version 4.0 through 4.4. However, for full functionality, upgrade to release 4.4.

If you have installed RHACS by using Helm charts, to upgrade to the latest version of RHACS perform the following steps:

1. Back up the Central database.
2. Optionally, optimize Central's database and Persistent Volume Claims (PVC).
3. Optionally, generate a **values-private.yaml** configuration file containing root certificates for the central-services Helm chart.
4. Update the Helm chart.
5. Run the **helm upgrade** command.

IMPORTANT

To ensure optimal functionality, use the same version for your secured-cluster-services Helm chart and central-services Helm chart.

2.2. BACKING UP THE CENTRAL DATABASE

You can back up the Central database and use that backup for rolling back from a failed upgrade or data restoration in the case of an infrastructure disaster.

Prerequisites

- You must have an API token with **read** permission for all resources of Red Hat Advanced Cluster Security for Kubernetes. The **Analyst** system role has **read** permissions for all resources.
- You have installed the **roxctl** CLI.
- You have configured the **ROX_API_TOKEN** and the **ROX_CENTRAL_ADDRESS** environment variables.

Procedure

- Run the backup command:
  -

```
$ roxctl -e "$ROX_CENTRAL_ADDRESS" central backup
```

Additional resources

- On-demand backups by using the roxctl CLI
- Installing the roxctl CLI

2.3. OPTIMIZING CENTRAL DATABASE AND PVC

When you upgrade to Red Hat Advanced Cluster Security for Kubernetes (RHACS) 4.0, RHACS creates a PostgreSQL instance called **central-db** with a default Persistent Volume Claims (PVC). Optionally, you can customize **central-db** or PVC configuration.

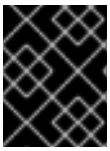
Red Hat recommends the following minimum memory and CPU requests:

```
central:
  db:
    resources:
      requests:
        memory: 16Gi
        cpu: 8
      limits:
        memory: 16Gi
        cpu: 8
```

2.4. GENERATING ROOT CERTIFICATES FILE

If you do not have access to your **values-private.yaml** configuration file that you have used to install Red Hat Advanced Cluster Security for Kubernetes (RHACS), use the following instruction to generate the **values-private.yaml** configuration file containing root certificates.

Skip the instruction here, if you have access to your **values-private.yaml** configuration file.



IMPORTANT

The generated **values-private.yaml** file has sensitive configuration options. Ensure that you store this file securely.

Procedure

1. Download the **create_certificate_values_file.sh** script.
2. Make the **create_certificate_values_file.sh** script executable:

```
$ chmod +x create_certificate_values_file.sh
```

3. Run the **create_certificate_values_file.sh** script file:

```
$ create_certificate_values_file.sh values-private.yaml
```

2.5. UPDATING THE HELM CHART REPOSITORY

You must always update Helm charts before upgrading to a new version of Red Hat Advanced Cluster Security for Kubernetes.

Prerequisites

- You must have already added the Red Hat Advanced Cluster Security for Kubernetes Helm chart repository.
- You must be using Helm version 3.8.3 or newer.

Procedure

- Update Red Hat Advanced Cluster Security for Kubernetes charts repository.

```
$ helm repo update
```

Verification

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

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

2.6. ADDITIONAL RESOURCES

- Installing Central using Helm charts
- Installing RHACS on secured clusters by using Helm charts

2.7. RUNNING THE HELM UPGRADE COMMAND

You can use the **helm upgrade** command to update Red Hat Advanced Cluster Security for Kubernetes (RHACS).

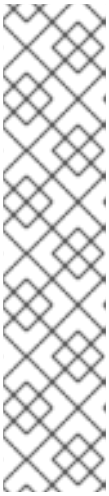
Prerequisites

- You must have access to the **values-private.yaml** configuration file that you have used to install Red Hat Advanced Cluster Security for Kubernetes (RHACS). Otherwise, you must generate the **values-private.yaml** configuration file containing root certificates before proceeding with these commands.

Procedure

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

```
$ helm upgrade -n stackrox stackrox-central-services \
  rhacs/central-services --version <current-rhacs-version> \ 1
  -f values-private.yaml \
  --set central.db.password.generate=true \
  --set central.db.serviceTLS.generate=true \
  --set central.db.persistence.persistentVolumeClaim.createClaim=true
```



NOTE

You might use the **--reuse-values** option to preserve the previously configured Helm values during the upgrade. If you do that, you must turn off **central-db** creation before you upgrade to the next version. See the following command example:

```
$ helm upgrade -n stackrox stackrox-central-services \
  rhacs/central-services --version <current-rhacs-version> --reuse-values \
  -f values-private.yaml \
  --set central.db.password.generate=false \
  --set central.db.serviceTLS.generate=false \
  --set central.db.persistence.persistentVolumeClaim.createClaim=false
```

2.8. REMOVING CENTRAL-ATTACHED PV AFTER UPGRADING TO VERSION 4.1 AND LATER

Kubernetes and OpenShift Container Platform do not delete persistent volumes (PV) automatically. When you upgrade RHACS from earlier versions, the Central PV called **stackrox-db** remains mounted. However, in RHACS 4.1, Central does not need the previously attached PV anymore.

The PV has data and persistent files used by earlier RHACS versions. You can use the PV to roll back to an earlier version before RHACS 4.1. Or, if you have a large RocksDB backup bundle for Central, you can use the PV to restore that data.

After you complete the upgrade to 4.1, you can remove the Central-attached persistent volume claim (PVC) to free up the storage. Only remove the PVC if you do not plan to roll back or restore from earlier RocksDB backups.

WARNING

After removing PVC, you cannot roll back Central to an earlier version before RHACS 4.1 or restore large RocksDB backups created with RocksDB.

2.8.1. Removing Central-attached PV using Helm

Remove the Central-attached persistent volume claim (PVC) **stackrox-db** to free up storage space.

Procedure

- Run the following command:

```
$ helm upgrade -n stackrox stackrox-central-services \
    rhacs/central-services --version <current-rhacs-version> \
    --set central.persistence.none=true
```

Verification

- Run the following command:

```
$ oc -n stackrox describe pvc stackrox-db | grep -i 'Used By'
Used By: <none> 1
```

1 1 Wait until you see **Used By: <none>**. It may take a few minutes.

2.9. ROLLING BACK A HELM UPGRADE

You can roll back to an earlier version of Central if the upgrade to a new version is unsuccessful.

Procedure

1. Run the following **helm upgrade** command:

```
$ helm upgrade -n stackrox \
  stackrox-central-services rhacs/central-services \
  --version <previous_rhacs_74_version> \ 1
  --set central.db.enabled=false
```

1 Replace **<previous_rhacs_74_version>** with the previously installed RHACS version.

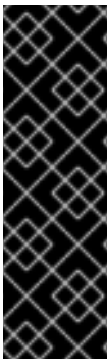
2. Delete the **central-db** persistent volume claim (PVC):

```
$ oc -n stackrox delete pvc central-db 1
```

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

CHAPTER 3. MANUALLY UPGRADING USING THE ROXCTL CLI

You can upgrade to the latest version of Red Hat Advanced Cluster Security for Kubernetes (RHACS) from a supported older version.



IMPORTANT

- You need to perform the manual upgrade procedure only if you used the **roxctl** CLI to install RHACS.
- There are manual steps for each version upgrade that must be followed, for example, from version 3.74 to version 4.0, and from version 4.0 to version 4.1. Therefore, Red Hat recommends upgrading first from 3.74 to 4.0, then from 4.0 to 4.1, then 4.1 to 4.2, until the selected version is installed. For full functionality, Red Hat recommends upgrading to the most recent version.

To upgrade RHACS to the latest version, perform the following steps:

1. Backup the Central database
2. Upgrade the **roxctl** CLI
3. Generate Central database provisioning bundle
4. Create resources by using the Central DB provisioning bundle
5. Upgrade the Central cluster
6. Upgrade all secured clusters

3.1. BACKING UP THE CENTRAL DATABASE

You can back up the Central database and use that backup for rolling back from a failed upgrade or data restoration in the case of an infrastructure disaster.

Prerequisites

- You must have an API token with **read** permission for all resources of Red Hat Advanced Cluster Security for Kubernetes. The **Analyst** system role has **read** permissions for all resources.
- You have installed the **roxctl** CLI.
- You have configured the **ROX_API_TOKEN** and the **ROX_CENTRAL_ADDRESS** environment variables.

Procedure

- Run the backup command:

```
$ roxctl -e "$ROX_CENTRAL_ADDRESS" central backup
```

Additional resources

- Authenticating by using the roxctl CLI

3.2. UPGRADING THE ROXCTL CLI

To upgrade the **roxctl** CLI to the latest version you must uninstall the existing version of **roxctl** CLI and then install the latest version of the **roxctl** CLI.

3.2.1. Uninstalling the roxctl CLI

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

Procedure

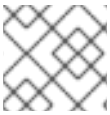
- Find and delete the **roxctl** binary:

```
$ ROXPATH=$(which roxctl) && rm -f $ROXPATH 1
```

1 Depending on your environment, you might need administrator rights to delete the **roxctl** binary.

3.2.2. 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
```

3.2.3. 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
```

3.2.4. 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
```

3.3. GENERATING CENTRAL DATABASE PROVISIONING BUNDLE

Before upgrading Central you must first generate a database provisioning bundle. This bundle is a **tar** archive that has a README file, a few YAML configuration files, and some scripts that aid in the installation process.

Prerequisites

- You must have an API token with the **Admin** role.
- You must have installed the **roxctl** CLI.

Procedure

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

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

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

2. Run the **central db generate** command:

```
$ roxctl -e $ROX_CENTRAL_ADDRESS central db generate \
  <cluster_type> \ 1
  <storage> \ 2
  --output-dir <bundle_dir> \ 3
  --central-db-image registry.redhat.io/advanced-cluster-security/rhacs-central-db-rhel8:4.4.3
```

1. **cluster-type** is the type of your cluster, specify **k8s** for Kubernetes and **openshift** for OpenShift Container Platform.
2. For **storage**, specify **hostpath** or **pvc**. If you use **pvc** you can use additional options to specify volume name, size, and storage class. Run **$ roxctl central db generate openshift pvc -h** for more details.
3. For **bundle-dir** specify the path where you want to save the generated provisioning bundle.

Next Step

- Use the Central DB provisioning bundle to create additional resources.

3.4. CREATING RESOURCES BY USING THE CENTRAL DB PROVISIONING BUNDLE

Before you upgrade the Central cluster, you must use the Central DB provisioning bundle to create additional resources that the Central cluster requires. This bundle is a **tar** archive that has a README file, a few YAML configuration files, and some scripts that aid in the installation process.

Prerequisites

- You must have generated a Central DB provisioning bundle.
- You must have extracted the **tar** archive bundle.

Procedure

1. Open the extracted bundle directory and run the **setup** script:

```
$ ./scripts/setup.sh
```

2. Run the **deploy-central-db** script:

```
$ ./deploy-central-db.sh
```

3.5. UPGRADING THE CENTRAL CLUSTER

After you have created a backup of the Central database and generated the necessary resources by using the provisioning bundle, the next step is to upgrade the Central cluster. This process involves upgrading Central and Scanner.

3.5.1. Upgrading Central

You can update Central to the latest version by downloading and deploying the updated images.

Procedure

- Run the following command to update the Central image:

```
$ oc -n stackrox set image deploy/central central=registry.redhat.io/advanced-cluster-security/rhacs-main-rhel8:4.4.3 1
```

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

Verification

- Verify that the new pods have deployed:

```
$ oc get deploy -n stackrox -o wide
```

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

3.5.1.1. Editing the GOMEMLIMIT environment variable for the Central deployment

Upgrading to version 4.4 requires that you manually replace the **GOMEMLIMIT** environment variable with the **ROX_MEMLIMIT** environment variable. You must edit this variable for each deployment.

Procedure

1. Run the following command to edit the variable for the Central deployment:

```
$ oc -n stackrox edit deploy/central
```
(1)

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

2. Replace the **GOMEMLIMIT** variable with **ROX_MEMLIMIT**.
3. Save the file.

3.5.2. Upgrading Scanner

You can update Scanner to the latest version by downloading and deploying the updated images.

Procedure

- Run the following command to update the Scanner image:

```
$ oc -n stackrox set image deploy/scanner scanner=registry.redhat.io/advanced-cluster-security/rhacs-scanner-rhel8:4.4.3
```
(1)

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

Verification

- Verify that the new pods have deployed:

```
$ oc get deploy -n stackrox -o wide
```

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

3.5.2.1. Editing the GOMEMLIMIT environment variable for the Scanner deployment

Upgrading to version 4.4 requires that you manually replace the **GOMEMLIMIT** environment variable with the **ROX_MEMLIMIT** environment variable. You must edit this variable for each deployment.

Procedure

1. Run the following command to edit the variable for the Scanner deployment:

```
$ oc -n stackrox edit deploy/scanner
```
(1)

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

2. Replace the **GOMEMLIMIT** variable with **ROX_MEMLIMIT**.
3. Save the file.

3.5.3. Verifying the Central cluster upgrade

After you have upgraded both Central and Scanner, verify that the Central cluster upgrade is complete.

Procedure

- Check the Central logs by running the following command:

```
$ oc logs -n stackrox deploy/central -c central 1
```

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

Sample output of a successful upgrade

```
No database restore directory found (this is not an error).
Migrator: 2023/04/19 17:58:54: starting DB compaction
Migrator: 2023/04/19 17:58:54: Free fraction of 0.0391 (40960/1048576) is < 0.7500. Will not
compact
badger 2023/04/19 17:58:54 INFO: All 1 tables opened in 2ms
badger 2023/04/19 17:58:55 INFO: Replaying file id: 0 at offset: 846357
badger 2023/04/19 17:58:55 INFO: Replay took: 50.324µs
badger 2023/04/19 17:58:55 DEBUG: Value log discard stats empty
Migrator: 2023/04/19 17:58:55: DB is up to date. Nothing to do here.
badger 2023/04/19 17:58:55 INFO: Got compaction priority: {level:0 score:1.73 dropPrefix:[]}
version: 2023/04/19 17:58:55.189866 ensure.go:49: Info: Version found in the DB was current. We're
good to go!
```

3.6. UPGRADING ALL SECURED CLUSTERS

After upgrading Central services, you must upgrade all secured clusters.

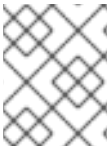
IMPORTANT

- If you are using automatic upgrades:
  - Update all your secured clusters by using automatic upgrades.
  - Skip the instructions in this section and follow the instructions in the Verify upgrades and Revoking the API token sections.
- If you are not using automatic upgrades, you must run the instructions in this section on all secured clusters including the Central cluster.
  - To ensure optimal functionality, use the same RHACS version for your secured clusters and the cluster on which Central is installed.

To complete manual upgrades of each secured cluster running Sensor, Collector, and Admission controller, follow the instructions in this section.

3.6.1. Updating other images

You must update the sensor, collector and compliance images on each secured cluster when not using automatic upgrades.



NOTE

If you are using Kubernetes, use **kubectl** instead of **oc** for the commands listed in this procedure.

Procedure

1. Update the Sensor image:

```
$ oc -n stackrox set image deploy/sensor sensor=registry.redhat.io/advanced-cluster-security/rhacs-main-rhel8:4.4.3 1
```

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

2. Update the Compliance image:

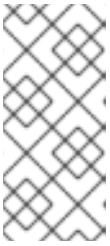
```
$ oc -n stackrox set image ds/collector compliance=registry.redhat.io/advanced-cluster-security/rhacs-main-rhel8:4.4.3 1
```

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

3. Update the Collector image:

```
$ oc -n stackrox set image ds/collector collector=registry.redhat.io/advanced-cluster-security/rhacs-collector-rhel8:4.4.3 1
```

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



NOTE

If you are using the collector slim image, run the following command instead:

```
$ oc -n stackrox set image ds/collector collector=registry.redhat.io/advanced-cluster-security/rhacs-collector-slim-rhel8:{rhacs-version}
```

4. Update the admission control image:

```
$ oc -n stackrox set image deploy/admission-control admission-control=registry.redhat.io/advanced-cluster-security/rhacs-main-rhel8:4.4.3
```

IMPORTANT

If you have installed RHACS on Red Hat OpenShift by using the **roxctl** CLI, you need to migrate the security context constraints (SCCs).

For more information, see "Migrating SCCs during the manual upgrade" in the "Additional resources" section.

Next steps

- Verifying secured cluster upgrade

Additional resources

- Migrating SCCs during the manual upgrade

3.6.2. Migrating SCCs during the manual upgrade

By migrating the security context constraints (SCCs) during the manual upgrade by using **roxctl** CLI, you can seamlessly transition the Red Hat Advanced Cluster Security for Kubernetes (RHACS) services to use the Red Hat OpenShift SCCs, ensuring compatibility and optimal security configurations across Central and all secured clusters.

Procedure

1. List all of the RHACS services that are deployed on Central and all secured clusters:

```
$ oc -n stackrox describe pods | grep 'openshift.io/scc\|^Name:'
```

Example output

```
Name:      admission-control-6f4dcc6b4c-2phwd
           openshift.io/scc: stackrox-admission-control
#...
Name:      central-575487bfcb-sjdx8
           openshift.io/scc: stackrox-central
Name:      central-db-7c7885bb-6bgbd
           openshift.io/scc: stackrox-central-db
Name:      collector-56nkr
           openshift.io/scc: stackrox-collector
#...
Name:      scanner-68fc55b599-f2wm6
           openshift.io/scc: stackrox-scanner
Name:      scanner-68fc55b599-fztlh
#...
Name:      sensor-84545f86b7-xgdwf
           openshift.io/scc: stackrox-sensor
#...
```

In this example, you can see that each pod has its own custom SCC, which is specified through the **openshift.io/scc** field.

2. Add the required roles and role bindings to use the Red Hat OpenShift SCCs instead of the RHACS custom SCCs.
To add the required roles and role bindings to use the Red Hat OpenShift SCCs for the Central cluster, perform the following steps:

   a. Create a file named **update-central.yaml** that defines the role and role binding resources by using the following content:

Example 3.1. Example YAML file

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role (1)
metadata:
```

```
  annotations:
    email: support@stackrox.com
    owner: stackrox
  labels:
    app.kubernetes.io/component: central
    app.kubernetes.io/instance: stackrox-central-services
    app.kubernetes.io/name: stackrox
    app.kubernetes.io/part-of: stackrox-central-services
    app.kubernetes.io/version: 4.4.0
  name: use-central-db-scc (2)
  namespace: stackrox (3)
Rules: (4)
- apiGroups:
  - security.openshift.io
  resourceNames:
  - nonroot-v2
  resources:
  - securitycontextconstraints
  verbs:
  - use
- - -
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  annotations:
    email: support@stackrox.com
    owner: stackrox
  labels:
    app.kubernetes.io/component: central
    app.kubernetes.io/instance: stackrox-central-services
    app.kubernetes.io/managed-by: Helm
    app.kubernetes.io/name: stackrox
    app.kubernetes.io/part-of: stackrox-central-services
    app.kubernetes.io/version: 4.4.0
  name: use-central-scc
  namespace: stackrox
rules:
- apiGroups:
  - security.openshift.io
  resourceNames:
  - nonroot-v2
  resources:
  - securitycontextconstraints
  verbs:
  - use
- - -
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  annotations:
    email: support@stackrox.com
    owner: stackrox
  labels:
    app.kubernetes.io/component: scanner
    app.kubernetes.io/instance: stackrox-central-services
    app.kubernetes.io/name: stackrox
```

```
    app.kubernetes.io/part-of: stackrox-central-services
    app.kubernetes.io/version: 4.4.0
  name: use-scanner-scc
  namespace: stackrox
rules:
- apiGroups:
  - security.openshift.io
  resourceNames:
  - nonroot-v2
  resources:
  - securitycontextconstraints
  verbs:
  - use
- - -
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding (5)
metadata:
  annotations:
    email: support@stackrox.com
    owner: stackrox
  labels:
    app.kubernetes.io/component: central
    app.kubernetes.io/instance: stackrox-central-services
    app.kubernetes.io/name: stackrox
    app.k ubernetes.io/part-of: stackrox-central-services
    app.kubernetes.io/version: 4.4.0
  name: central-db-use-scc (6)
  namespace: stackrox
roleRef: (7)
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: use-central-db-scc
subjects: (8)
- kind: ServiceAccount
  name: central-db
  namespace: stackrox
- - -
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  annotations:
    email: support@stackrox.com
    owner: stackrox
  labels:
    app.kubernetes.io/component: central
    app.kubernetes.io/instance: stackrox-central-services
    app.kubernetes.io/name: stackrox
    app.kubernetes.io/part-of: stackrox-central-services
    app.kubernetes.io/version: 4.4.0
  name: central-use-scc
  namespace: stackrox
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: use-central-scc
subjects:
```

```
- kind: ServiceAccount
  name: central
  namespace: stackrox
- - -
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  annotations:
    email: support@stackrox.com
    owner: stackrox
  labels:
    app.kubernetes.io/component: scanner
    app.kubernetes.io/instance: stackrox-central-services
    app.kubernetes.io/name: stackrox
    app.kubernetes.io/part-of: stackrox-central-services
    app.kubernetes.io/version: 4.4.0
  name: scanner-use-scc
  namespace: stackrox
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: use-scanner-scc
subjects:
- kind: ServiceAccount
  name: scanner
  namespace: stackrox
- - -
```

1. The type of Kubernetes resource, in this example, **Role**.
2. The name of the role resource.
3. The namespace in which the role is created.
4. Describes the permissions granted by the role resource.
5. The type of Kubernetes resource, in this example, **RoleBinding**.
6. The name of the role binding resource.
7. Specifies the role to bind in the same namespace.
8. Specifies the subjects that are bound to the role.

b. Create the role and role binding resources specified in the **update-central.yaml** file by running the following command:

```
$ oc -n stackrox create -f ./update-central.yaml
```

3. To add the required roles and role bindings to use the Red Hat OpenShift SCCs for all secured clusters, perform the following steps:

a. Create a file named **upgrade-scs.yaml** that defines the role and role binding resources by using the following content:

Example 3.2. Example YAML file

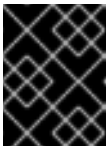
```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role (1)
metadata:
  annotations:
    email: support@stackrox.com
    owner: stackrox
  labels:
    app.kubernetes.io/component: collector
    app.kubernetes.io/instance: stackrox-secured-cluster-services
    app.kubernetes.io/name: stackrox
    app.kubernetes.io/part-of: stackrox-secured-cluster-services
    app.kubernetes.io/version: 4.4.0
    auto-upgrade.stackrox.io/component: sensor
  name: use-privileged-scc (2)
  namespace: stackrox (3)
rules: (4)
- apiGroups:
  - security.openshift.io
  resourceNames:
  - privileged
  resources:
  - securitycontextconstraints
  verbs:
  - use
- - -
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding (5)
metadata:
  annotations:
    email: support@stackrox.com
    owner: stackrox
  labels:
    app.kubernetes.io/component: collector
    app.kubernetes.io/instance: stackrox-secured-cluster-services
    app.kubernetes.io/name: stackrox
    app.kubernetes.io/part-of: stackrox-secured-cluster-services
    app.kubernetes.io/version: 4.4.0
    auto-upgrade.stackrox.io/component: sensor
  name: collector-use-scc (6)
  namespace: stackrox
roleRef: (7)
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: use-privileged-scc
subjects: (8)
- kind: ServiceAccount
  name: collector
  namespace: stackrox
- - -
```

(1) The type of Kubernetes resource, in this example, **Role**.

2. The name of the role resource.
3. The namespace in which the role is created.
4. Describes the permissions granted by the role resource.
5. The type of Kubernetes resource, in this example, **RoleBinding**.
6. The name of the role binding resource.
7. Specifies the role to bind in the same namespace.
8. Specifies the subjects that are bound to the role.

b. Create the role and role binding resources specified in the **upgrade-scs.yaml** file by running the following command:

```
$ oc -n stackrox create -f ./update-scs.yaml
```



IMPORTANT

You must run this command on each secured cluster to create the role and role bindings specified in the **upgrade-scs.yaml** file.

4. Delete the SCCs that are specific to RHACS:

a. To delete the SCCs that are specific to the Central cluster, run the following command:

```
$ oc delete scc/stackrox-central scc/stackrox-central-db scc/stackrox-scanner
```

b. To delete the SCCs that are specific to all secured clusters, run the following command:

```
$ oc delete scc/stackrox-admission-control scc/stackrox-collector scc/stackrox-sensor
```

IMPORTANT

You must run this command on each secured cluster to delete the SCCs that are specific to each secured cluster.

Verification

- Ensure that all the pods are using the correct SCCs by running the following command:

```
$ oc -n stackrox describe pods | grep 'openshift.io/scc\|^Name:'
```

Compare the output with the following table:

Component	Previous custom SCC	New Red Hat OpenShift 4 SCC
Central	**stackrox-central**	**nonroot-v2**
Central-db	**stackrox-central-db**	**nonroot-v2**
Scanner	**stackrox-scanner**	**nonroot-v2**
Scanner-db	**stackrox-scanner**	**nonroot-v2**
Admission Controller	**stackrox-admission-control**	**restricted-v2**
Collector	**stackrox-collector**	**privileged**
Sensor	**stackrox-sensor**	**restricted-v2**

3.6.2.1. Editing the GOMEMLIMIT environment variable for the Sensor deployment

Upgrading to version 4.4 requires that you manually replace the **GOMEMLIMIT** environment variable with the **ROX_MEMLIMIT** environment variable. You must edit this variable for each deployment.

Procedure

1. Run the following command to edit the variable for the Sensor deployment:

```
$ oc -n stackrox edit deploy/sensor (1)
```

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

2. Replace the **GOMEMLIMIT** variable with **ROX_MEMLIMIT**.
3. Save the file.

3.6.2.2. Editing the GOMEMLIMIT environment variable for the Collector deployment

Upgrading to version 4.4 requires that you manually replace the **GOMEMLIMIT** environment variable with the **ROX_MEMLIMIT** environment variable. You must edit this variable for each deployment.

Procedure

1. Run the following command to edit the variable for the Collector deployment:

```
$ oc -n stackrox edit deploy/collector (1)
```

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

2. Replace the **GOMEMLIMIT** variable with **ROX_MEMLIMIT**.

3. Save the file.

3.6.2.3. Editing the GOMEMLIMIT environment variable for the Admission Controller deployment

Upgrading to version 4.4 requires that you manually replace the **GOMEMLIMIT** environment variable with the **ROX_MEMLIMIT** environment variable. You must edit this variable for each deployment.

Procedure

1. Run the following command to edit the variable for the Admission Controller deployment:

```
$ oc -n stackrox edit deploy/admission-control 1
```

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

2. Replace the **GOMEMLIMIT** variable with **ROX_MEMLIMIT**.

3. Save the file.

3.6.2.4. Verifying secured cluster upgrade

After you have upgraded secured clusters, verify that the updated pods are working.

Procedure

- Check that the new pods have deployed:

```
$ oc get deploy,ds -n stackrox -o wide 1
```

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

```
$ oc get pod -n stackrox --watch 1
```

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

3.7. ENABLING RHCOS NODE SCANNING

If you use OpenShift Container Platform, you can enable scanning of Red Hat Enterprise Linux CoreOS (RHCOS) nodes for vulnerabilities by using Red Hat Advanced Cluster Security for Kubernetes (RHACS).

Prerequisites

- For scanning RHCOS node hosts of the Secured cluster, you must have installed Secured cluster on 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

Procedure

1. Run one of the following commands to update the compliance container.
   - For a default compliance container with metrics disabled, run the following command:

```
$ oc -n stackrox patch daemonset/collector -p '{"spec":{"template":{"spec":{"containers":
[{"name":"compliance","env":[{"name":"ROX_METRICS_PORT","value":"disabled"},
{"name":"ROX_NODE_SCANNING_ENDPOINT","value":"127.0.0.1:8444"},
{"name":"ROX_NODE_SCANNING_INTERVAL","value":"4h"},
{"name":"ROX_NODE_SCANNING_INTERVAL_DEVIATION","value":"24m"},
{"name":"ROX_NODE_SCANNING_MAX_INITIAL_WAIT","value":"5m"},
{"name":"ROX_RHCOS_NODE_SCANNING","value":"true"},
{"name":"ROX_CALL_NODE_INVENTORY_ENABLED","value":"true"}]}]}}}}'
```

   - For a compliance container with Prometheus metrics enabled, run the following command:

```
$ oc -n stackrox patch daemonset/collector -p '{"spec":{"template":{"spec":{"containers":
[{"name":"compliance","env":[{"name":"ROX_METRICS_PORT","value":":9091"},
{"name":"ROX_NODE_SCANNING_ENDPOINT","value":"127.0.0.1:8444"},
{"name":"ROX_NODE_SCANNING_INTERVAL","value":"4h"},
{"name":"ROX_NODE_SCANNING_INTERVAL_DEVIATION","value":"24m"},
{"name":"ROX_NODE_SCANNING_MAX_INITIAL_WAIT","value":"5m"},
{"name":"ROX_RHCOS_NODE_SCANNING","value":"true"},
{"name":"ROX_CALL_NODE_INVENTORY_ENABLED","value":"true"}]}]}}}}'
```

2. Update the Collector DaemonSet (DS) by taking the following steps:
   a. Add new volume mounts to Collector DS by running the following command:

```
$ oc -n stackrox patch daemonset/collector -p '{"spec":{"template":{"spec":{"volumes":
[{"name":"tmp-volume","emptyDir":{}},{"name":"cache-volume","emptyDir":
{"sizeLimit":"200Mi"}}]}}}}'
```

   b. Add the new **NodeScanner** container by running the following command:

```
$ oc -n stackrox patch daemonset/collector -p '{"spec":{"template":{"spec":{"containers":
[{"command":["/scanner","--nodeinventory","--config=",""],"env":
[{"name":"ROX_NODE_NAME","valueFrom":{"fieldRef":
{"apiVersion":"v1","fieldPath":"spec.nodeName"}}},
{"name":"ROX_CLAIR_V4_SCANNING","value":"true"},
{"name":"ROX_COMPLIANCE_OPERATOR_INTEGRATION","value":"true"},
{"name":"ROX_CSV_EXPORT","value":"false"},
{"name":"ROX_DECLARATIVE_CONFIGURATION","value":"false"},
{"name":"ROX_INTEGRATIONS_AS_CONFIG","value":"false"},
{"name":"ROX_NETPOL_FIELDS","value":"true"},
{"name":"ROX_NETWORK_DETECTION_BASELINE_SIMULATION","value":"true"},
{"name":"ROX_NETWORK_GRAPH_PATTERNFLY","value":"true"},
{"name":"ROX_NODE_SCANNING_CACHE_TIME","value":"3h36m"},
{"name":"ROX_NODE_SCANNING_INITIAL_BACKOFF","value":"30s"},
{"name":"ROX_NODE_SCANNING_MAX_BACKOFF","value":"5m"},
{"name":"ROX_PROCESSES_LISTENING_ON_PORT","value":"false"},
{"name":"ROX_QUAY_ROBOT_ACCOUNTS","value":"true"},
{"name":"ROX_ROXCTL_NETPOL_GENERATE","value":"true"},
{"name":"ROX_SOURCED_AUTOGENERATED_INTEGRATIONS","value":"false"},
{"name":"ROX_SYSLOG_EXTRA_FIELDS","value":"true"},
```

```
{"name":"ROX_SYSTEM_HEALTH_PF","value":"false"},
{"name":"ROX_VULN_MGMT_WORKLOAD_CVES","value":"false"}],"image":"registry.redhat.io/advanced-cluster-security/rhacs-scanner-slim-rhel8:4.4.3","imagePullPolicy":"IfNotPresent","name":"node-inventory","ports":[{"containerPort":8444,"name":"grpc","protocol":"TCP"}],"volumeMounts":[{"mountPath":"/host","name":"host-root-ro","readOnly":true},
{"mountPath":"/tmp/","name":"tmp-volume"},{"mountPath":"/cache","name":"cache-volume"}]}]}}}}'
```

Additional resources

- Scanning RHCOS node hosts

3.8. REMOVING CENTRAL-ATTACHED PV AFTER UPGRADING TO VERSION 4.1 AND LATER

Kubernetes and OpenShift Container Platform do not delete persistent volumes (PV) automatically. When you upgrade RHACS from earlier versions, the Central PV called **stackrox-db** remains mounted. However, in RHACS 4.1, Central does not need the previously attached PV anymore.

The PV has data and persistent files used by earlier RHACS versions. You can use the PV to roll back to an earlier version before RHACS 4.1. Or, if you have a large RocksDB backup bundle for Central, you can use the PV to restore that data.

After you complete the upgrade to 4.1, you can remove the Central-attached persistent volume claim (PVC) to free up the storage. Only remove the PVC if you do not plan to roll back or restore from earlier RocksDB backups.

WARNING

After removing PVC, you cannot roll back Central to an earlier version before RHACS 4.1 or restore large RocksDB backups created with RocksDB.

3.8.1. Removing Central-attached PV using the roxctl CLI

Remove the Central-attached persistent volume claim (PVC) **stackrox-db** to free up storage space.

Procedure

- Run the following command:

```
$ oc get deployment central -n stackrox -o json | jq '(.spec.template.spec.volumes[] | select(.name=="stackrox-db"))={"name": "stackrox-db", "emptyDir": {}}' | oc apply -f -
```

  It replaces the **stackrox-db**` entry in the **spec.template.spec.volumes** to a local emptyDir.

Verification

- Run the following command:

```
$ oc -n stackrox describe pvc stackrox-db | grep -i 'Used By'
Used By: <none> 1
```

1 Wait until you see **Used By: <none>**. It might take a few minutes.

3.9. ROLLING BACK CENTRAL

You can roll back to a previous version of Central if the upgrade to a new version is unsuccessful.

3.9.1. Rolling back Central normally

You can roll back to a previous version of Central if upgrading Red Hat Advanced Cluster Security for Kubernetes fails.

Prerequisites

- Before you can perform a rollback, you must have free disk space available on your persistent storage. Red Hat Advanced Cluster Security for Kubernetes uses disk space to keep a copy of databases during the upgrade. If the disk space is not enough to store a copy and the upgrade fails, you might not be able to roll back to an earlier version.

Procedure

- Run the following command to roll back to a previous version when an upgrade fails (before the Central service starts):

```
$ oc -n stackrox rollout undo deploy/central 1
```

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

3.9.2. Rolling back Central forcefully

You can use forced rollback to roll back to an earlier version of Central (after the Central service starts).

IMPORTANT

Using forced rollback to switch back to a previous version might result in loss of data and functionality.

Prerequisites

- Before you can perform a rollback, you must have free disk space available on your persistent storage. Red Hat Advanced Cluster Security for Kubernetes uses disk space to keep a copy of databases during the upgrade. If the disk space is not enough to store a copy and the upgrade fails, you will not be able to roll back to an earlier version.

Procedure

- Run the following commands to perform a forced rollback:
  - To forcefully rollback to the previously installed version:

```
$ oc -n stackrox rollout undo deploy/central (1)
```

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

- To forcefully rollback to a specific version:

  1. Edit Central's **ConfigMap**:

     ```
     $ oc -n stackrox edit configmap/central-config (1)
     ```

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

  2. Update the value of the **maintenance.forceRollbackVersion** key:

     ```
     data:
       central-config.yaml: |
         maintenance:
           safeMode: false
           compaction:
             enabled: true
             bucketFillFraction: .5
             freeFractionThreshold: 0.75
           forceRollbackVersion: <x.x.x.x> (1)
       ...
     ```

     (1) Specify the version that you want to roll back to.

  3. Update the Central image version:

     ```
     $ oc -n stackrox \ (1)
       set image deploy/central central=registry.redhat.io/advanced-cluster-security/rhacs-main-rhel8:<x.x.x.x> (2)
     ```

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

     (2) Specify the version that you want to roll back to. It must be the same version that you specified for the **maintenance.forceRollbackVersion** key in the **central-config** config map.

3.10. VERIFYING UPGRADES

The updated Sensors and Collectors continue to report the latest data from each secured cluster.

The last time Sensor contacted Central is visible in the RHACS portal.

Procedure

1. In the RHACS portal, go to **Platform Configuration → System Health**.
2. Check to ensure that Sensor Upgrade shows clusters up to date with Central.

3.11. REVOKING THE API TOKEN

For security reasons, Red Hat recommends that you revoke the API token that you have used to complete Central database backup.

Prerequisites

- After the upgrade, you must reload the RHACS portal page and re-accept the certificate to continue using the RHACS portal.

Procedure

1. In the RHACS portal, go to **Platform Configuration → Integrations**.
2. Scroll down to the **Authentication Tokens** category, and click **API Token**.
3. Select the checkbox in front of the token name that you want to revoke.
4. Click **Revoke**.
5. On the confirmation dialog box, click **Confirm**.