



# Red Hat OpenShift Service on AWS 4

## Install ROSA with HCP clusters

Installing, accessing, and deleting Red Hat OpenShift Service on AWS (ROSA) clusters.



## Red Hat OpenShift Service on AWS 4 Install ROSA with HCP clusters

---

Installing, accessing, and deleting Red Hat OpenShift Service on AWS (ROSA) clusters.

## 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<sup>®</sup> is the registered trademark of Linus Torvalds in the United States and other countries.

Java<sup>®</sup> is a registered trademark of Oracle and/or its affiliates.

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

MySQL<sup>®</sup> is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js<sup>®</sup> 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<sup>®</sup> 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 provides information on how to install Red Hat OpenShift Service on AWS (ROSA) clusters that use hosted control planes.

## Table of Contents

<b>CHAPTER 1. CREATING ROSA WITH HCP CLUSTERS USING THE DEFAULT OPTIONS</b> .....	<b>4</b>
Considerations regarding auto creation mode	4
1.1. OVERVIEW OF THE DEFAULT CLUSTER SPECIFICATIONS	5
1.2. ROSA WITH HCP PREREQUISITES	6
1.2.1. Creating a Virtual Private Cloud for your ROSA with HCP clusters	6
Creating a Virtual Private Cloud using Terraform	7
Creating a Virtual Private Cloud manually	8
Tagging your subnets	8
1.2.2. Creating the account-wide STS roles and policies	9
1.2.3. Creating an OpenID Connect configuration	10
1.2.4. Creating Operator roles and policies	12
1.3. CREATING A ROSA WITH HCP CLUSTER USING THE CLI	14
1.4. NEXT STEPS	16
1.5. ADDITIONAL RESOURCES	16
<b>CHAPTER 2. CREATING A ROSA CLUSTER USING TERRAFORM</b> .....	<b>18</b>
2.1. CREATING A DEFAULT ROSA CLUSTER USING TERRAFORM	18
2.1.1. Overview of Terraform	18
Considerations when using Terraform	20
2.1.2. Overview of the default cluster specifications	20
2.1.3. Creating a default ROSA cluster using Terraform	22
2.1.3.1. Preparing your environment for Terraform	22
2.1.3.2. Creating your Terraform files locally	22
2.1.3.3. Using Terraform to create your ROSA cluster	27
2.1.3.4. Deleting your ROSA cluster with Terraform	29
<b>CHAPTER 3. CREATING ROSA WITH HCP CLUSTERS USING A CUSTOM AWS KMS ENCRYPTION KEY</b> ..	<b>31</b>
3.1. ROSA WITH HCP PREREQUISITES	31
3.1.1. Creating a Virtual Private Cloud for your ROSA with HCP clusters	31
Creating a Virtual Private Cloud using Terraform	31
Creating a Virtual Private Cloud manually	32
3.1.2. Creating the account-wide STS roles and policies	33
3.1.3. Creating an OpenID Connect configuration	34
3.1.4. Creating Operator roles and policies	35
3.1.5. Creating a ROSA cluster using a custom AWS KMS key	38
3.2. NEXT STEPS	41
3.3. ADDITIONAL RESOURCES	41
<b>CHAPTER 4. CREATING A PRIVATE CLUSTER ON ROSA WITH HCP</b> .....	<b>42</b>
4.1. CREATING AN AWS PRIVATE CLUSTER	42
4.2. CONFIGURING AWS SECURITY GROUPS TO ACCESS THE API	43
4.3. NEXT STEPS	44
4.4. ADDITIONAL RESOURCES	44
<b>CHAPTER 5. CREATING ROSA WITH HCP CLUSTERS WITH EXTERNAL AUTHENTICATION</b> .....	<b>45</b>
5.1. ROSA WITH HCP PREREQUISITES	45
5.2. CREATING A ROSA WITH HCP CLUSTER THAT USES EXTERNAL AUTHENTICATION PROVIDERS	45
5.3. CREATING AN EXTERNAL AUTHENTICATION PROVIDER	47
5.4. CREATING A BREAK GLASS CREDENTIAL FOR A ROSA WITH HCP CLUSTER	50
5.5. ACCESSING A ROSA WITH HCP CLUSTER BY USING A BREAK GLASS CREDENTIAL	53
5.6. REVOKING A BREAK GLASS CREDENTIAL FOR A ROSA WITH HCP CLUSTER	54
5.7. DELETING AN EXTERNAL AUTHENTICATION PROVIDER	55

5.8. ADDITIONAL RESOURCES	56
<b>CHAPTER 6. USING THE NODE TUNING OPERATOR ON ROSA WITH HCP CLUSTERS</b> .....	<b>58</b>
Purpose	58
6.1. CUSTOM TUNING SPECIFICATION	58
6.2. CREATING NODE TUNING CONFIGURATIONS ON ROSA WITH HCP	63
6.3. MODIFYING YOUR NODE TUNING CONFIGURATIONS FOR ROSA WITH HCP	65
6.4. DELETING NODE TUNING CONFIGURATIONS ON ROSA WITH HCP	67
<b>CHAPTER 7. DELETING A ROSA WITH HCP CLUSTER</b> .....	<b>68</b>
7.1. DELETING A ROSA WITH HCP CLUSTER AND THE CLUSTER-SPECIFIC IAM RESOURCES	68
7.2. DELETING THE ACCOUNT-WIDE IAM RESOURCES	70
7.2.1. Deleting the account-wide IAM roles and policies	71
7.2.2. Unlinking and deleting the OpenShift Cluster Manager and user IAM roles	73



# CHAPTER 1. CREATING ROSA WITH HCP CLUSTERS USING THE DEFAULT OPTIONS



## NOTE

If you are looking for a quickstart guide for ROSA Classic, see [Red Hat OpenShift Service on AWS quickstart guide](#).

Red Hat OpenShift Service on AWS (ROSA) with hosted control planes (HCP) offers a more efficient and reliable architecture for creating Red Hat OpenShift Service on AWS (ROSA) clusters. With ROSA with HCP, each cluster has a dedicated control plane that is isolated in a ROSA service account.

Create a ROSA with HCP cluster quickly by using the default options and automatic AWS Identity and Access Management (IAM) resource creation. You can deploy your cluster by using the ROSA CLI (**rosa**).



## IMPORTANT

Since it is not possible to upgrade or convert existing ROSA clusters to a hosted control planes architecture, you must create a new cluster to use ROSA with HCP functionality.



## IMPORTANT

[Sharing VPCs across multiple AWS accounts](#) is not currently supported for ROSA with HCP. Do not install a ROSA with HCP cluster into subnets shared from another AWS account. See "[Are multiple ROSA clusters in a single VPC supported?](#)" for more information.



## NOTE

ROSA with HCP clusters only support AWS Security Token Service (STS) authentication.

### Further reading

- For a comparison between ROSA with HCP and ROSA Classic, see the [Comparing architecture models](#) documentation.
- See the AWS documentation for information about [Getting started with ROSA with HCP using the ROSA CLI in auto mode](#).

### Additional resources

For a full list of the supported certificates, see the [Compliance](#) section of "Understanding process and security for Red Hat OpenShift Service on AWS".

### Considerations regarding auto creation mode

The procedures in this document use the **auto** mode in the ROSA CLI to immediately create the required IAM resources using the current AWS account. The required resources include the account-wide IAM roles and policies, cluster-specific Operator roles and policies, and OpenID Connect (OIDC) identity provider.

Alternatively, you can use **manual** mode, which outputs the **aws** commands needed to create the IAM resources instead of deploying them automatically.



## Next steps



- Ensure that you have completed the [AWS prerequisites](#).

## 1.1. OVERVIEW OF THE DEFAULT CLUSTER SPECIFICATIONS

You can quickly create a ROSA with HCP cluster with the Security Token Service (STS) by using the default installation options. The following summary describes the default cluster specifications.

**Table 1.1. Default ROSA with HCP cluster specifications**

Component	Default specifications
Accounts and roles	<ul style="list-style-type: none"> <li>• Default IAM role prefix: <b>ManagedOpenShift</b></li> <li>• No cluster admin role created</li> </ul>
Cluster settings	<ul style="list-style-type: none"> <li>• Default cluster version: Latest</li> <li>• Default AWS region for installations using the ROSA CLI (<b>rosa</b>): Defined by your <b>aws</b> CLI configuration</li> <li>• Availability: Single zone for the data plane</li> <li>• Default EC2 IMDS endpoints (both v1 and v2) are enabled</li> <li>• Monitoring for user-defined projects: Enabled</li> </ul>
Encryption	<ul style="list-style-type: none"> <li>• Cloud storage is encrypted at rest</li> <li>• Additional etcd encryption is not enabled</li> <li>• The default AWS Key Management Service (KMS) key is used as the encryption key for persistent data</li> </ul>
Compute node machine pool	<ul style="list-style-type: none"> <li>• Compute node instance type: m5.xlarge (4 vCPU 16, GiB RAM)</li> <li>• Compute node count: 2</li> <li>• Autoscaling: Not enabled</li> <li>• No additional node labels</li> </ul>
Networking configuration	<ul style="list-style-type: none"> <li>• Cluster privacy: Public</li> <li>• You must have configured your own Virtual Private Cloud (VPC)</li> <li>• No cluster-wide proxy is configured</li> </ul>

Component	Default specifications
Classless Inter-Domain Routing (CIDR) ranges	<ul style="list-style-type: none"> <li>• Machine CIDR: 10.0.0.0/16</li> <li>• Service CIDR: 172.30.0.0/16</li> <li>• Pod CIDR: 10.128.0.0/16</li> <li>• Host prefix: /23</li> </ul>  <p><b>NOTE</b></p> <p>When using ROSA with HCP, the static IP address <b>172.20.0.1</b> is reserved for the internal Kubernetes API address. The machine, pod, and service CIDRs ranges must not conflict with this IP address.</p>
Cluster roles and policies	<ul style="list-style-type: none"> <li>• Mode used to create the Operator roles and the OpenID Connect (OIDC) provider: <b>auto</b></li> </ul>  <p><b>NOTE</b></p> <p>For installations that use OpenShift Cluster Manager on the Hybrid Cloud Console, the <b>auto</b> mode requires an admin-privileged OpenShift Cluster Manager role.</p> <ul style="list-style-type: none"> <li>• Default Operator role prefix: <b>&lt;cluster_name&gt;-&lt;4_digit_random_string&gt;</b></li> </ul>
Cluster update strategy	<ul style="list-style-type: none"> <li>• Individual updates</li> <li>• 1 hour grace period for node draining</li> </ul>

## 1.2. ROSA WITH HCP PREREQUISITES

To create a ROSA with HCP cluster, you must have the following items:

- A configured virtual private cloud (VPC)
- Account-wide roles
- An OIDC configuration
- Operator roles

### 1.2.1. Creating a Virtual Private Cloud for your ROSA with HCP clusters

You must have a Virtual Private Cloud (VPC) to create ROSA with HCP cluster. You can use the following methods to create a VPC:

- Create a VPC by using a Terraform template
- Manually create the VPC resources in the AWS console



## NOTE

The Terraform instructions are for testing and demonstration purposes. Your own installation requires some modifications to the VPC for your own use. You should also ensure that when you use this Terraform script it is in the same region that you intend to install your cluster. In these examples, use **us-east-2**.

## Creating a Virtual Private Cloud using Terraform

Terraform is a tool that allows you to create various resources using an established template. The following process uses the default options as required to create a ROSA with HCP cluster. For more information about using Terraform, see the additional resources.

### Prerequisites

- You have installed Terraform version 1.4.0 or newer on your machine.
- You have installed Git on your machine.

### Procedure

1. Open a shell prompt and clone the Terraform VPC repository by running the following command:

```
$ git clone https://github.com/openshift-cs/terraform-vpc-example
```

2. Navigate to the created directory by running the following command:

```
$ cd terraform-vpc-example
```

3. Initiate the Terraform file by running the following command:

```
$ terraform init
```

A message confirming the initialization appears when this process completes.

4. To build your VPC Terraform plan based on the existing Terraform template, run the **plan** command. You must include your AWS region. You can choose to specify a cluster name. A **rosa.tfplan** file is added to the **hypershift-tf** directory after the **terraform plan** completes. For more detailed options, see the [Terraform VPC repository's README file](#).

```
$ terraform plan -out rosa.tfplan -var region=<region>
```

5. Apply this plan file to build your VPC by running the following command:

```
$ terraform apply rosa.tfplan
```

- a. Optional: You can capture the values of the Terraform-provisioned private, public, and machinepool subnet IDs as environment variables to use when creating your ROSA with HCP cluster by running the following commands:

```
$ export SUBNET_IDS=$(terraform output -raw cluster-subnets-string)
```

- b. Verify that the variables were correctly set with the following command:

```
$ echo $SUBNET_IDS
```

### Example output

```
$ subnet-0a6a57e0f784171aa,subnet-078e84e5b10ecf5b0
```

### Additional resources

- See the [Terraform VPC](#) repository for a detailed list of all options available when customizing the VPC for your needs.

### Creating a Virtual Private Cloud manually

If you choose to manually create your Virtual Private Cloud (VPC) instead of using Terraform, go to [the VPC page in the AWS console](#). Your VPC must meet the requirements shown in the following table.

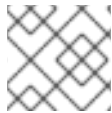
**Table 1.2. Requirements for your VPC**

Requirement	Details
VPC name	You need to have the specific VPC name and ID when creating your cluster.
CIDR range	Your VPC CIDR range should match your machine CIDR.
Availability zone	You need one availability zone for a single zone, and you need three for availability zones for multi-zone.
Public subnet	You must have one public subnet with a NAT gateway for public clusters. Private clusters do not need a public subnet.
DNS hostname and resolution	You must ensure that the DNS hostname and resolution are enabled.

### Tagging your subnets

Before you can use your VPC to create a ROSA with HCP cluster, you must tag your VPC subnets. Automated service preflight checks verify that these resources are tagged correctly before you can use these resources. The following table shows how your resources should be tagged as the following:

Resource	Key	Value
Public subnet	<b>kubernetes.io/role/elb</b>	<b>1</b> or no value
Private subnet	<b>kubernetes.io/role/internal-elb</b>	<b>1</b> or no value



## NOTE

You must tag at least one private subnet and, if applicable, and one public subnet.

### Prerequisites

- You have created a VPC.
- You have installed the **aws** CLI.

### Procedure

1. Tag your resources in your terminal by running the following commands:

- a. For public subnets, run:

```
$ aws ec2 create-tags --resources <public-subnet-id> --tags
Key=kubernetes.io/role/elb,Value=1
```

- b. For private subnets, run:

```
$ aws ec2 create-tags --resources <private-subnet-id> --tags
Key=kubernetes.io/role/internal-elb,Value=1
```

### Verification

- Verify that the tag is correctly applied by running the following command:

```
$ aws ec2 describe-tags --filters "Name=resource-id,Values=<subnet_id>"
```

### Example output

```
TAGS   Name           <subnet-id>   subnet <prefix>-subnet-public1-us-east-1a
TAGS   kubernetes.io/role/elb <subnet-id>   subnet 1
```

### Additional resources

- [Get Started with Amazon VPC](#)
- [HashiCorp Terraform documentation](#)
- [Subnet Auto Discovery](#)

## 1.2.2. Creating the account-wide STS roles and policies

Before using the Red Hat OpenShift Service on AWS (ROSA) CLI (**rosa**) to create Red Hat OpenShift Service on AWS (ROSA) with hosted control planes (HCP) clusters, create the required account-wide roles and policies, including the Operator policies.



## NOTE

ROSA with HCP clusters require account and Operator roles with AWS managed policies attached. Customer managed policies are not supported. For more information regarding AWS managed policies for ROSA with HCP clusters, see [AWS managed policies for ROSA account roles](#).

### Prerequisites

- You have completed the AWS prerequisites for ROSA with HCP.
- You have available AWS service quotas.
- You have enabled the ROSA service in the AWS Console.
- You have installed and configured the latest ROSA CLI (**rosa**) on your installation host.
- You have logged in to your Red Hat account by using the ROSA CLI.

### Procedure

1. If they do not exist in your AWS account, create the required account-wide STS roles and attach the policies by running the following command:

```
$ rosa create account-roles --hosted-cp
```

2. Optional: Set your prefix as an environmental variable by running the following command:

```
$ export ACCOUNT_ROLES_PREFIX=<account_role_prefix>
```

- View the value of the variable by running the following command:

```
$ echo $ACCOUNT_ROLES_PREFIX
```

### Example output

```
ManagedOpenShift
```

For more information regarding AWS managed IAM policies for ROSA, see [AWS managed IAM policies for ROSA](#).

## 1.2.3. Creating an OpenID Connect configuration

When using a ROSA with HCP cluster, you must create the OpenID Connect (OIDC) configuration prior to creating your cluster. This configuration is registered to be used with OpenShift Cluster Manager.

### Prerequisites

- You have completed the AWS prerequisites for ROSA with HCP.
- You have completed the AWS prerequisites for Red Hat OpenShift Service on AWS.
- You have installed and configured the latest Red Hat OpenShift Service on AWS (ROSA) CLI, **rosa**, on your installation host.

## Procedure

1. To create your OIDC configuration alongside the AWS resources, run the following command:

```
$ rosa create oidc-config --mode=auto --yes
```

This command returns the following information.

### Example output

```
? Would you like to create a Managed (Red Hat hosted) OIDC Configuration Yes
I: Setting up managed OIDC configuration
I: To create Operator Roles for this OIDC Configuration, run the following command and
remember to replace <user-defined> with a prefix of your choice:
  rosa create operator-roles --prefix <user-defined> --oidc-config-id 13cdr6b
If you are going to create a Hosted Control Plane cluster please include '--hosted-cp'
I: Creating OIDC provider using 'arn:aws:iam::4540112244:user/userName'
? Create the OIDC provider? Yes
I: Created OIDC provider with ARN 'arn:aws:iam::4540112244:oidc-
provider/dvbwgdztaeq9o.cloudfront.net/13cdr6b'
```

When creating your cluster, you must supply the OIDC config ID. The CLI output provides this value for **--mode auto**, otherwise you must determine these values based on **aws** CLI output for **--mode manual**.

2. Optional: you can save the OIDC configuration ID as a variable to use later. Run the following command to save the variable:

```
$ export OIDC_ID=<oidc_config_id> 1
```

- 1** In the example output above, the OIDC configuration ID is 13cdr6b.

- View the value of the variable by running the following command:

```
$ echo $OIDC_ID
```

### Example output

```
13cdr6b
```

## Verification

- You can list the possible OIDC configurations available for your clusters that are associated with your user organization. Run the following command:

```
$ rosa list oidc-config
```

### Example output

```
ID                MANAGED ISSUER URL
SECRET ARN
2330dbs0n8m3chkk25gkkcd8pnj3lk2 true
```

```
https://dvbwgdztaeq9o.cloudfront.net/2330dbs0n8m3chkk25gkkcd8pnj3lk2
233hvnjrjoqu14jltk6lhbhf2tj11f8un false https://oidc-r7u1.s3.us-east-1.amazonaws.com
aws:secretsmanager:us-east-1:242819244:secret:rosa-private-key-oidc-r7u1-tM3MDN
```

## 1.2.4. Creating Operator roles and policies

When using a ROSA with HCP cluster, you must create the Operator IAM roles that are required for Red Hat OpenShift Service on AWS (ROSA) with hosted control planes (HCP) deployments. The cluster Operators use the Operator roles to obtain the temporary permissions required to carry out cluster operations, such as managing back-end storage, cloud provider credentials, and external access to a cluster.

### Prerequisites

- You have completed the AWS prerequisites for ROSA with HCP.
- You have installed and configured the latest Red Hat OpenShift Service on AWS ROSA CLI (**rosa**), on your installation host.
- You created the account-wide AWS roles.

### Procedure

1. Set your prefix name to an environment variable using the following command:

```
$ export OPERATOR_ROLES_PREFIX=<prefix_name>
```

2. To create your Operator roles, run the following command:

```
$ rosa create operator-roles --hosted-cp --prefix=$OPERATOR_ROLES_PREFIX --oidc-
config-id=$OIDC_ID --installer-role-arn
arn:aws:iam::${AWS_ACCOUNT_ID}:role/${ACCOUNT_ROLES_PREFIX}-HCP-ROSA-
Installer-Role
```

The following breakdown provides options for the Operator role creation.

```
$ rosa create operator-roles --hosted-cp
--prefix=$OPERATOR_ROLES_PREFIX 1
--oidc-config-id=$OIDC_ID 2
--installer-role-arn
arn:aws:iam::${AWS_ACCOUNT_ID}:role/${ACCOUNT_ROLES_PREFIX}-HCP-ROSA-
Installer-Role 3
```

- 1** You must supply a prefix when creating these Operator roles. Failing to do so produces an error. See the Additional resources of this section for information on the Operator prefix.
- 2** This value is the OIDC configuration ID that you created for your ROSA with HCP cluster.
- 3** This value is the installer role ARN that you created when you created the ROSA account roles.

You must include the **--hosted-cp** parameter to create the correct roles for ROSA with HCP clusters. This command returns the following information.



## Example output

```

? Role creation mode: auto
? Operator roles prefix: <pre-filled_prefix> 1
? OIDC Configuration ID: 23soa2bgvpek9kmes9s7os0a39i13qm4 |
https://dvbwgdztaeq9o.cloudfront.net/23soa2bgvpek9kmes9s7os0a39i13qm4 2
? Create hosted control plane operator roles: Yes
W: More than one Installer role found
? Installer role ARN: arn:aws:iam::4540112244:role/<prefix>-HCP-ROSA-Installer-Role
? Permissions boundary ARN (optional):
I: Reusable OIDC Configuration detected. Validating trusted relationships to operator roles:
I: Creating roles using 'arn:aws:iam::4540112244:user/<userName>'
I: Created role '<prefix>-openshift-cluster-csi-drivers-ebs-cloud-credentials' with ARN
'arn:aws:iam::4540112244:role/<prefix>-openshift-cluster-csi-drivers-ebs-cloud-credentials'
I: Created role '<prefix>-openshift-cloud-network-config-controller-cloud-credenti' with ARN
'arn:aws:iam::4540112244:role/<prefix>-openshift-cloud-network-config-controller-cloud-
credenti'
I: Created role '<prefix>-kube-system-kube-controller-manager' with ARN
'arn:aws:iam::4540112244:role/<prefix>-kube-system-kube-controller-manager'
I: Created role '<prefix>-kube-system-capa-controller-manager' with ARN
'arn:aws:iam::4540112244:role/<prefix>-kube-system-capa-controller-manager'
I: Created role '<prefix>-kube-system-control-plane-operator' with ARN
'arn:aws:iam::4540112244:role/<prefix>-kube-system-control-plane-operator'
I: Created role '<prefix>-kube-system-kms-provider' with ARN
'arn:aws:iam::4540112244:role/<prefix>-kube-system-kms-provider'
I: Created role '<prefix>-openshift-image-registry-installer-cloud-credentials' with ARN
'arn:aws:iam::4540112244:role/<prefix>-openshift-image-registry-installer-cloud-credentials'
I: Created role '<prefix>-openshift-ingress-operator-cloud-credentials' with ARN
'arn:aws:iam::4540112244:role/<prefix>-openshift-ingress-operator-cloud-credentials'
I: To create a cluster with these roles, run the following command:
  rosa create cluster --sts --oidc-config-id 23soa2bgvpek9kmes9s7os0a39i13qm4 --operator-
roles-prefix <prefix> --hosted-cp

```

- 1 This field is prepopulated with the prefix that you set in the initial creation command.
- 2 This field requires you to select an OIDC configuration that you created for your ROSA with HCP cluster.

The Operator roles are now created and ready to use for creating your ROSA with HCP cluster.

## Verification

- You can list the Operator roles associated with your ROSA account. Run the following command:

```
$ rosa list operator-roles
```

## Example output

```

I: Fetching operator roles
ROLE PREFIX AMOUNT IN BUNDLE
<prefix> 8
? Would you like to detail a specific prefix Yes 1

```

```

? Operator Role Prefix: <prefix>
ROLE NAME                                ROLE ARN
VERSION MANAGED
<prefix>-kube-system-capac-controller-manager
arn:aws:iam::4540112244:role/<prefix>-kube-system-capac-controller-manager
4.13 No
<prefix>-kube-system-control-plane-operator
arn:aws:iam::4540112244:role/<prefix>-kube-system-control-plane-operator
4.13 No
<prefix>-kube-system-kms-provider
arn:aws:iam::4540112244:role/<prefix>-kube-system-kms-provider
No 4.13
<prefix>-kube-system-kube-controller-manager
arn:aws:iam::4540112244:role/<prefix>-kube-system-kube-controller-manager
4.13 No
<prefix>-openshift-cloud-network-config-controller-cloud-credenti
arn:aws:iam::4540112244:role/<prefix>-openshift-cloud-network-config-controller-cloud-
credenti 4.13 No
<prefix>-openshift-cluster-csi-drivers-ebs-cloud-credentials
arn:aws:iam::4540112244:role/<prefix>-openshift-cluster-csi-drivers-ebs-cloud-credentials
4.13 No
<prefix>-openshift-image-registry-installer-cloud-credentials
arn:aws:iam::4540112244:role/<prefix>-openshift-image-registry-installer-cloud-credentials
4.13 No
<prefix>-openshift-ingress-operator-cloud-credentials
arn:aws:iam::4540112244:role/<prefix>-openshift-ingress-operator-cloud-credentials
4.13 No

```

- 1 After the command runs, it displays all the prefixes associated with your AWS account and notes how many roles are associated with this prefix. If you need to see all of these roles and their details, enter "Yes" on the detail prompt to have these roles listed out with specifics.

### Additional resources

- See [About custom Operator IAM role prefixes](#) for information on the Operator prefixes.

## 1.3. CREATING A ROSA WITH HCP CLUSTER USING THE CLI

When using the Red Hat OpenShift Service on AWS (ROSA) CLI, **rosa**, to create a cluster, you can select the default options to create the cluster quickly.

### Prerequisites

- You have completed the AWS prerequisites for ROSA with HCP.
- You have available AWS service quotas.
- You have enabled the ROSA service in the AWS Console.
- You have installed and configured the latest ROSA CLI (**rosa**) on your installation host. Run **rosa version** to see your currently installed version of the ROSA CLI. If a newer version is available, the CLI provides a link to download this upgrade.

- You have logged in to your Red Hat account by using the ROSA CLI.
- You have created an OIDC configuration.
- You have verified that the AWS Elastic Load Balancing (ELB) service role exists in your AWS account.

## Procedure

1. Use one of the following commands to create your ROSA with HCP cluster:



### NOTE

When creating a ROSA with HCP cluster, the default machine Classless Inter-Domain Routing (CIDR) is **10.0.0.0/16**. If this does not correspond to the CIDR range for your VPC subnets, add **--machine-cidr <address\_block>** to the following commands. To learn more about the default CIDR ranges for Red Hat OpenShift Service on AWS, see [CIDR range definitions](#).

- If you did not set environmental variables, run the following command:

```
$ rosa create cluster --cluster-name=<cluster_name> \ <.>
  --mode=auto --hosted-cp [--private] \ <.>
  --operator-roles-prefix <operator-role-prefix> \ <.>
  --oidc-config-id <id-of-oidc-configuration> \
  --subnet-ids=<public-subnet-id>,<private-subnet-id>
```

<.> Specify the name of your cluster. If your cluster name is longer than 15 characters, it will contain an autogenerated domain prefix as a subdomain for your provisioned cluster on openshiftapps.com. To customize the subdomain, use the **--domain-prefix** flag. The domain prefix cannot be longer than 15 characters, must be unique, and cannot be changed after cluster creation. <.> Optional: The **--private** argument is used to create private ROSA with HCP clusters. If you use this argument, ensure that you only use your private subnet ID for **--subnet-ids**. <.> By default, the cluster-specific Operator role names are prefixed with the cluster name and a random 4-digit hash. You can optionally specify a custom prefix to replace **<cluster\_name>-<hash>** in the role names. The prefix is applied when you create the cluster-specific Operator IAM roles. For information about the prefix, see *About custom Operator IAM role prefixes*.



### NOTE

If you specified custom ARN paths when you created the associated account-wide roles, the custom path is automatically detected. The custom path is applied to the cluster-specific Operator roles when you create them in a later step.

- If you set the environmental variables, create a cluster with a single, initial machine pool, using either a publicly or privately available API, and a publicly or privately available Ingress by running the following command:

```
$ rosa create cluster --private --cluster-name=<cluster_name> \
  --mode=auto --hosted-cp --operator-roles-prefix=$OPERATOR_ROLES_PREFIX \
  --oidc-config-id=$OIDC_ID --subnet-ids=$SUBNET_IDS
```

- If you set the environmental variables, create a cluster with a single, initial machine pool, a publicly available API, and a publicly available Ingress by running the following command:

```
$ rosa create cluster --cluster-name=<cluster_name> --mode=auto \
  --hosted-cp --operator-roles-prefix=$OPERATOR_ROLES_PREFIX \
  --oidc-config-id=$OIDC_ID --subnet-ids=$SUBNET_IDS
```

2. Check the status of your cluster by running the following command:

```
$ rosa describe cluster --cluster=<cluster_name>
```

The following **State** field changes are listed in the output as the cluster installation progresses:

- **pending (Preparing account)**
- **installing (DNS setup in progress)**
- **installing**
- **ready**



#### NOTE

If the installation fails or the **State** field does not change to **ready** after more than 10 minutes, check the installation troubleshooting documentation for details. For more information, see *Troubleshooting installations*. For steps to contact Red Hat Support for assistance, see *Getting support for Red Hat OpenShift Service on AWS*.

3. Track the progress of the cluster creation by watching the Red Hat OpenShift Service on AWS installation program logs. To check the logs, run the following command:

```
$ rosa logs install --cluster=<cluster_name> --watch \<.>
```

<.> Optional: To watch for new log messages as the installation progresses, use the **--watch** argument.

## 1.4. NEXT STEPS

- [Accessing a ROSA cluster](#)
- [Adding notification contacts](#)

## 1.5. ADDITIONAL RESOURCES

- For steps to deploy a ROSA cluster using manual mode, see [Creating a cluster using customizations](#).
- For more information about the AWS Identity Access Management (IAM) resources required to deploy Red Hat OpenShift Service on AWS with STS, see [About IAM resources for clusters that use STS](#).
- See [Additional custom security groups](#) for information about security group requirements.

- For details about optionally setting an Operator role name prefix, see [About custom Operator IAM role prefixes](#).
- For information about the prerequisites to installing ROSA with STS, see [AWS prerequisites for ROSA with STS](#).
- For details about using the **auto** and **manual** modes to create the required STS resources, see [Understanding the auto and manual deployment modes](#).
- For more information about using OpenID Connect (OIDC) identity providers in AWS IAM, see [Creating OpenID Connect \(OIDC\) identity providers](#) in the AWS documentation.
- For more information about troubleshooting ROSA cluster installations, see [Troubleshooting installations](#).
- For steps to contact Red Hat Support for assistance, see [Getting support for Red Hat OpenShift Service on AWS](#).

## CHAPTER 2. CREATING A ROSA CLUSTER USING TERRAFORM

### 2.1. CREATING A DEFAULT ROSA CLUSTER USING TERRAFORM

Create a Red Hat OpenShift Service on AWS (ROSA) cluster quickly by using a Terraform cluster template that is configured with the default cluster options.

The cluster creation process described below uses a Terraform configuration that prepares a ROSA with HCP cluster with the following resources:

- An OIDC provider with a managed **oidc-config** configuration
- Prerequisite IAM Operator roles with associated AWS Managed ROSA Policies
- IAM account roles with associated AWS Managed ROSA Policies
- All other AWS resources required to create a ROSA with STS cluster

#### 2.1.1. Overview of Terraform

Terraform is an infrastructure-as-code tool that provides a way to configure your resources once and replicate those resources as desired. Terraform accomplishes the creation tasks by using declarative language. You declare what you want the final state of the infrastructure resource to be, and Terraform creates these resources to your specifications.

#### Prerequisites

To use [the Red Hat Cloud Services provider](#) inside your Terraform configuration, you must meet the following prerequisites:

- You have installed the Red Hat OpenShift Service on AWS (ROSA) command-line interface (CLI) tool.
- You have your offline [Red Hat OpenShift Cluster Manager token](#).
- You have installed [Terraform version 1.4.6](#) or newer.
- You have created your AWS account-wide IAM roles.  
The specific account-wide IAM roles and policies provide the STS permissions required for ROSA support, installation, control plane, and compute functionality. This includes account-wide Operator policies. See the Additional resources for more information on the AWS account roles.
- You have an [AWS account](#) and [associated credentials](#) that allow you to create resources. The credentials are configured for the AWS provider. See the [Authentication and Configuration](#) section in AWS Terraform provider documentation.
- You have, at minimum, the following permissions in your AWS IAM role policy that is operating Terraform. Check for these permissions in the AWS console.

#### Example 2.1. Minimum AWS permissions for Terraform

```
{
  "Version": "2012-10-17",
  "Statement": [
```



```

{
  "Sid": "VisualEditor0",
  "Effect": "Allow",
  "Action": [
    "iam:GetPolicyVersion",
    "iam:DeletePolicyVersion",
    "iam:CreatePolicyVersion",
    "iam:UpdateAssumeRolePolicy",
    "secretsmanager:DescribeSecret",
    "iam:ListRoleTags",
    "secretsmanager:PutSecretValue",
    "secretsmanager:CreateSecret",
    "iam:TagRole",
    "secretsmanager>DeleteSecret",
    "iam:UpdateOpenIDConnectProviderThumbprint",
    "iam:DeletePolicy",
    "iam:CreateRole",
    "iam:AttachRolePolicy",
    "iam:ListInstanceProfilesForRole",
    "secretsmanager:GetSecretValue",
    "iam:DetachRolePolicy",
    "iam:ListAttachedRolePolicies",
    "iam:ListPolicyTags",
    "iam:ListRolePolicies",
    "iam>DeleteOpenIDConnectProvider",
    "iam>DeleteInstanceProfile",
    "iam:GetRole",
    "iam:GetPolicy",
    "iam>ListEntitiesForPolicy",
    "iam>DeleteRole",
    "iam:TagPolicy",
    "iam>CreateOpenIDConnectProvider",
    "iam>CreatePolicy",
    "secretsmanager:GetResourcePolicy",
    "iam>ListPolicyVersions",
    "iam:UpdateRole",
    "iam:GetOpenIDConnectProvider",
    "iam:TagOpenIDConnectProvider",
    "secretsmanager:TagResource",
    "sts:AssumeRoleWithWebIdentity",
    "iam>ListRoles"
  ],
  "Resource": [
    "arn:aws:secretsmanager:*:<ACCOUNT_ID>:secret:*",
    "arn:aws:iam::<ACCOUNT_ID>:instance-profile/*",
    "arn:aws:iam::<ACCOUNT_ID>:role/*",
    "arn:aws:iam::<ACCOUNT_ID>:oidc-provider/*",
    "arn:aws:iam::<ACCOUNT_ID>:policy/*"
  ]
},
{
  "Sid": "VisualEditor1",
  "Effect": "Allow",
  "Action": [
    "s3:*"
  ],

```





Component	Default specifications
Compute node machine pool	<ul style="list-style-type: none"> <li>● Compute node instance type: m5.xlarge (4 vCPU 16, GiB RAM)</li> <li>● Compute node count: 3</li> <li>● Autoscaling: Not enabled</li> <li>● No additional node labels</li> </ul>
Networking configuration	<ul style="list-style-type: none"> <li>● Cluster privacy: public or private</li> <li>● You can choose to create a new VPC during the Terraform cluster creation process.</li> <li>● You must have configured your own Virtual Private Cloud (VPC)</li> <li>● No cluster-wide proxy is configured</li> </ul>
Classless Inter-Domain Routing (CIDR) ranges	<ul style="list-style-type: none"> <li>● Machine CIDR: 10.0.0.0/16</li> <li>● Service CIDR: 172.30.0.0/16</li> <li>● Pod CIDR: 10.128.0.0/14</li> <li>● Host prefix: /23</li> </ul> <div data-bbox="592 1144 699 1339" style="display: inline-block; vertical-align: middle;">  </div> <div data-bbox="778 1149 868 1180" style="display: inline-block; vertical-align: middle; margin-left: 10px;"> <p><b>NOTE</b></p> </div> <div data-bbox="778 1216 1406 1339" style="display: inline-block; vertical-align: middle; margin-left: 10px;"> <p>When using ROSA with HCP, the static IP address <b>172.20.0.1</b> is reserved for the internal Kubernetes API address. The machine, pod, and service CIDRs ranges must not conflict with this IP address.</p> </div>
Cluster roles and policies	<ul style="list-style-type: none"> <li>● Mode used to create the Operator roles and the OpenID Connect (OIDC) provider: <b>auto</b></li> </ul> <div data-bbox="592 1570 699 1733" style="display: inline-block; vertical-align: middle;">  </div> <div data-bbox="778 1576 868 1608" style="display: inline-block; vertical-align: middle; margin-left: 10px;"> <p><b>NOTE</b></p> </div> <div data-bbox="778 1644 1422 1733" style="display: inline-block; vertical-align: middle; margin-left: 10px;"> <p>For installations that use OpenShift Cluster Manager on the Hybrid Cloud Console, the <b>auto</b> mode requires an admin-privileged OpenShift Cluster Manager role.</p> </div> <ul style="list-style-type: none"> <li>● Default Operator role prefix: <b>rosa-<code>&lt;6-digit-alphanumeric-string&gt;</code></b></li> </ul>
Cluster update strategy	<ul style="list-style-type: none"> <li>● Individual updates</li> <li>● 1 hour grace period for node draining</li> </ul>

### 2.1.3. Creating a default ROSA cluster using Terraform

The cluster creation process outlined below shows how to use Terraform to create your account-wide IAM roles and a ROSA cluster with a managed OIDC configuration.

#### 2.1.3.1. Preparing your environment for Terraform

Before you can create your Red Hat OpenShift Service on AWS cluster by using Terraform, you need to export your [offline Red Hat OpenShift Cluster Manager token](#) .

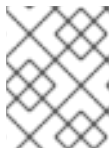
##### Procedure

1. **Optional:** Because the Terraform files get created in your current directory during this procedure, you can create a new directory to store these files and navigate into it by running the following command:

```
$ mkdir terraform-cluster && cd terraform-cluster
```

2. Grant permissions to your account by using [an offline Red Hat OpenShift Cluster Manager token](#).
3. Copy your offline token, and set the token as an environmental variable by running the following command:

```
$ export RHCS_TOKEN=<your_offline_token>
```



##### NOTE

This environmental variable resets at the end of each session, such as restarting your machine or closing the terminal.

##### Verification

- After you export your token, verify the value by running the following command:

```
$ echo $RHCS_TOKEN
```

#### 2.1.3.2. Creating your Terraform files locally

After you set up your [offline Red Hat OpenShift Cluster Manager token](#) , you need to create the Terraform files locally to build your cluster. You can create these files by using the following code templates.

##### Procedure

1. Create the **main.tf** file by running the following command:

```
$ cat<<-EOF>main.tf
#
# Copyright (c) 2023 Red Hat, Inc.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
```

```

# You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = ">= 4.20.0"
    }
    rhcs = {
      version = ">= 1.6.2"
      source = "terraform-redhat/rhcs"
    }
  }
}

# Export token using the RHCS_TOKEN environment variable
provider "rhcs" {}

provider "aws" {
  region = var.aws_region
  ignore_tags {
    key_prefixes = ["kubernetes.io/"]
  }
  default_tags {
    tags = var.default_aws_tags
  }
}

data "aws_availability_zones" "available" {}

locals {
  # Extract availability zone names for the specified region, limit it to 3 if multi az or 1 if single
  region_azs = var.multi_az ? slice([for zone in data.aws_availability_zones.available.names :
format("%s", zone)], 0, 3) : slice([for zone in data.aws_availability_zones.available.names :
format("%s", zone)], 0, 1)
}

resource "random_string" "random_name" {
  length = 6
  special = false
  upper = false
}

locals {
  worker_node_replicas = var.multi_az ? 3 : 2
  # If cluster_name is not null, use that, otherwise generate a random cluster name
  cluster_name = coalesce(var.cluster_name, "rosa-${random_string.random_name.result}")
}

```

```

# The network validator requires an additional 60 seconds to validate Terraform clusters.
resource "time_sleep" "wait_60_seconds" {
  count = var.create_vpc ? 1 : 0
  depends_on = [module.vpc]
  create_duration = "60s"
}

module "rosa-hcp" {
  source          = "terraform-redhat/rosa-hcp/rhcs"
  version         = "1.6.2"
  cluster_name    = local.cluster_name
  openshift_version = var.openshift_version
  account_role_prefix = local.cluster_name
  operator_role_prefix = local.cluster_name
  replicas        = local.worker_node_replicas
  aws_availability_zones = local.region_azs
  create_oidc      = true
  private          = var.private_cluster
  aws_subnet_ids   = var.create_vpc ? var.private_cluster ?
module.vpc[0].private_subnets : concat(module.vpc[0].public_subnets,
module.vpc[0].private_subnets) : var.aws_subnet_ids
  create_account_roles = true
  create_operator_roles = true

  depends_on = [time_sleep.wait_60_seconds]
}
EOF

```

2. Create the **variables.tf** file by running the following command:



#### NOTE

Copy and edit this file *before* running the command to build your cluster.

```

$ cat<<-EOF>variables.tf
#
# Copyright (c) 2023 Red Hat, Inc.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#
variable "openshift_version" {
  type    = string
  default = "4.14.20"
  description = "Desired version of OpenShift for the cluster, for example '4.14.20'. If version is

```

```
greater than the currently running version, an upgrade will be scheduled."
}

variable "create_vpc" {
  type    = bool
  description = "If you would like to create a new VPC, set this value to 'true'. If you do not
want to create a new VPC, set this value to 'false'."
}

# ROSA Cluster info
variable "cluster_name" {
  default    = null
  type      = string
  description = "The name of the ROSA cluster to create"
}

variable "additional_tags" {
  default = {
    Terraform = "true"
    Environment = "dev"
  }
  description = "Additional AWS resource tags"
  type        = map(string)
}

variable "multi_az" {
  type      = bool
  description = "Multi AZ Cluster for High Availability"
  default   = true
}

variable "worker_node_replicas" {
  default    = 3
  description = "Number of worker nodes to provision. Single zone clusters need at least 2
nodes, multizone clusters need at least 3 nodes"
  type      = number
}

variable "aws_subnet_ids" {
  type      = list(any)
  description = "A list of either the public or public + private subnet IDs to use for the cluster
blocks to use for the cluster"
  default   = ["subnet-01234567890abcdef", "subnet-01234567890abcdef", "subnet-
01234567890abcdef"]
}

variable "private_cluster" {
  type      = bool
  description = "If you want to create a private cluster, set this value to 'true'. If you want a
publicly available cluster, set this value to 'false'."
}

#VPC Info
variable "vpc_name" {
  type      = string
  description = "VPC Name"
```

```

default    = "tf-qs-vpc"
}

variable "vpc_cidr_block" {
  type     = string
  description = "value of the CIDR block to use for the VPC"
  default  = "10.0.0.0/16"
}

variable "private_subnet_cidrs" {
  type     = list(any)
  description = "The CIDR blocks to use for the private subnets"
  default  = ["10.0.1.0/24", "10.0.2.0/24", "10.0.3.0/24"]
}

variable "public_subnet_cidrs" {
  type     = list(any)
  description = "The CIDR blocks to use for the public subnets"
  default  = ["10.0.101.0/24", "10.0.102.0/24", "10.0.103.0/24"]
}

variable "single_nat_gateway" {
  type     = bool
  description = "Single NAT or per NAT for subnet"
  default  = false
}

#AWS Info
variable "aws_region" {
  type     = string
  default  = "us-east-2"
}

variable "default_aws_tags" {
  type     = map(string)
  description = "Default tags for AWS"
  default  = {}
}
EOF

```

3. Create the **vpc.tf** file by running the following command:

```

$ cat<<-EOF>>vpc.tf
#
# Copyright (c) 2023 Red Hat, Inc.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and

```

```
# limitations under the License.
#
module "vpc" {
  source = "terraform-aws-modules/vpc/aws"
  version = "5.1.2"

  count = var.create_vpc ? 1 : 0
  name = var.vpc_name
  cidr = var.vpc_cidr_block

  azs          = local.region_azs
  private_subnets = var.multi_az ? var.private_subnet_cidrs : [var.private_subnet_cidrs[0]]
  public_subnets = var.multi_az ? var.public_subnet_cidrs : [var.public_subnet_cidrs[0]]

  enable_nat_gateway = true
  single_nat_gateway = var.single_nat_gateway
  enable_dns_hostnames = true
  enable_dns_support = true

  tags = var.additional_tags
}
EOF
```

You are ready to initiate Terraform.

### 2.1.3.3. Using Terraform to create your ROSA cluster

After you create the Terraform files, you must initiate Terraform to provide all of the required dependencies. Then apply the Terraform plan.



#### IMPORTANT

Do not modify Terraform state files. For more information, see [Considerations when using Terraform](#)

#### Procedure

1. Set up Terraform to create your resources based on your Terraform files, run the following command:

```
$ terraform init
```

2. **Optional:** Verify that the Terraform you copied is correct by running the following command:

```
$ terraform validate
```

#### Example output

```
Success! The configuration is valid.
```

3. Create your cluster with Terraform by running the following command:

```
$ terraform apply
```

The Terraform interface asks two questions to create your cluster, similar to the following:

### Example output

```
var.create_vpc
  If you would like to create a new VPC, set this value to 'true'. If you do not want to create a
  new VPC, set this value to 'false'.

  Enter a value:

var.private_cluster
  If you want to create a private cluster, set this value to 'true'. If you want a publicly available
  cluster, set this value to 'false'.

  Enter a value:
```

4. Enter **yes** to proceed or **no** to cancel when the Terraform interface lists the resources to be created or changed and prompts for confirmation:

### Example output

```
Plan: 63 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.
```

If you enter **yes**, your Terraform plan starts, creating your AWS account roles, Operator roles, and your ROSA Classic cluster.

## Verification

1. Verify that your cluster was created by running the following command:

```
$ rosa list clusters
```

### Example output showing a cluster's ID, name, and status:

```
ID                NAME                STATE TOPOLOGY
27c3snjsupa9obua74ba8se5kcj11269  rosa-tf-demo  ready Classic (STS)
```

2. Verify that your account roles were created by running the following command:

```
$ rosa list account-roles
```

### Example output

```
I: Fetching account roles
ROLE NAME                ROLE TYPE  ROLE ARN
OPENSIFT VERSION  AWS Managed
ROSA-demo-Installer-Role  Installer  arn:aws:iam::<ID>:role/ROSA-demo-
Installer-Role          4.14      No
ROSA-demo-Support-Role   Support    arn:aws:iam::<ID>:role/ROSA-demo-
```



Support-Role	4.14	No	
ROSA-demo-Worker-Role		Worker	arn:aws:iam::<ID>:role/ROSA-demo-
Worker-Role	4.14	No	

- Verify that your Operator roles were created by running the following command:

```
$ rosa list operator-roles
```

#### Example output showing Terraform-created Operator roles:

```
I: Fetching operator roles
ROLE PREFIX  AMOUNT IN BUNDLE
rosa-demo    8
```

### 2.1.3.4. Deleting your ROSA cluster with Terraform

Use the **terraform destroy** command to remove all of the resources that were created with the **terraform apply** command.



#### NOTE

Do not modify your Terraform **.tf** files before destroying your resources. These variables are matched to resources to delete.

#### Procedure

- In the directory where you ran the **terraform apply** command to create your cluster, run the following command to delete the cluster:

```
$ terraform destroy
```

The Terraform interface prompts you for two variables. These should match the answers you provided when creating a cluster:

```
var.create_vpc
```

If you would like to create a new VPC, set this value to 'true.' If you do not want to create a new VPC, set this value to 'false.'

Enter a value:

```
var.private_cluster
```

If you want to create a private cluster, set this value to 'true.' If you want a publicly available cluster, set this value to 'false.'

Enter a value:

- Enter **yes** to start the role and cluster deletion:

#### Example output

```
Plan: 0 to add, 0 to change, 63 to destroy.
```

```
Do you really want to destroy all resources?
```

```
Terraform will destroy all your managed infrastructure, as shown above.  
There is no undo. Only 'yes' will be accepted to confirm.
```

```
Enter a value: yes
```

## Verification

1. Verify that your cluster was destroyed by running the following command:

```
$ rosa list clusters
```

### Example output showing no cluster

```
I: No clusters available
```

2. Verify that the account roles were destroyed by running the following command:

```
$ rosa list account-roles
```

### Example output showing no Terraform-created account roles:

```
I: Fetching account roles  
I: No account roles available
```

3. Verify that the Operator roles were destroyed by running the following command:

```
$ rosa list operator-roles
```

### Example output showing no Terraform-created Operator roles:

```
I: Fetching operator roles  
I: No operator roles available
```

## CHAPTER 3. CREATING ROSA WITH HCP CLUSTERS USING A CUSTOM AWS KMS ENCRYPTION KEY

Create a Red Hat OpenShift Service on AWS (ROSA) with a hosted control planes (HCP) cluster using a custom AWS Key Management Service (KMS) key.

### 3.1. ROSA WITH HCP PREREQUISITES

To create a ROSA with HCP cluster, you must have the following items:

- A configured virtual private cloud (VPC)
- Account-wide roles
- An OIDC configuration
- Operator roles

#### 3.1.1. Creating a Virtual Private Cloud for your ROSA with HCP clusters

You must have a Virtual Private Cloud (VPC) to create ROSA with HCP cluster. You can use the following methods to create a VPC:

- Create a VPC by using a Terraform template
- Manually create the VPC resources in the AWS console



#### NOTE

The Terraform instructions are for testing and demonstration purposes. Your own installation requires some modifications to the VPC for your own use. You should also ensure that when you use this Terraform script it is in the same region that you intend to install your cluster. In these examples, use **us-east-2**.



#### IMPORTANT

[Sharing VPCs across multiple AWS accounts](#) is not currently supported for ROSA with HCP. Do not install a ROSA with HCP cluster into subnets shared from another AWS account. See "[Are multiple ROSA clusters in a single VPC supported?](#)" for more information.

#### Creating a Virtual Private Cloud using Terraform

Terraform is a tool that allows you to create various resources using an established template. The following process uses the default options as required to create a ROSA with HCP cluster. For more information about using Terraform, see the additional resources.

#### Prerequisites

- You have installed Terraform version 1.4.0 or newer on your machine.
- You have installed Git on your machine.

#### Procedure

1. Open a shell prompt and clone the Terraform VPC repository by running the following command:

```
$ git clone https://github.com/openshift-cs/terraform-vpc-example
```

2. Navigate to the created directory by running the following command:

```
$ cd terraform-vpc-example
```

3. Initiate the Terraform file by running the following command:

```
$ terraform init
```

A message confirming the initialization appears when this process completes.

4. To build your VPC Terraform plan based on the existing Terraform template, run the **plan** command. You must include your AWS region. You can choose to specify a cluster name. A **rosa.tfplan** file is added to the **hypershift-tf** directory after the **terraform plan** completes. For more detailed options, see the [Terraform VPC repository's README file](#).

```
$ terraform plan -out rosa.tfplan -var region=<region>
```

5. Apply this plan file to build your VPC by running the following command:

```
$ terraform apply rosa.tfplan
```

- a. Optional: You can capture the values of the Terraform-provisioned private, public, and machinepool subnet IDs as environment variables to use when creating your ROSA with HCP cluster by running the following commands:

```
$ export SUBNET_IDS=$(terraform output -raw cluster-subnets-string)
```

- b. Verify that the variables were correctly set with the following command:

```
$ echo $SUBNET_IDS
```

### Example output

```
$ subnet-0a6a57e0f784171aa,subnet-078e84e5b10ecf5b0
```

### Additional resources

- See the [Terraform VPC](#) repository for a detailed list of all options available when customizing the VPC for your needs.

### Creating a Virtual Private Cloud manually

If you choose to manually create your Virtual Private Cloud (VPC) instead of using Terraform, go to [the VPC page in the AWS console](#). Your VPC must meet the requirements shown in the following table.

Table 3.1. Requirements for your VPC

Requirement	Details
VPC name	You need to have the specific VPC name and ID when creating your cluster.
CIDR range	Your VPC CIDR range should match your machine CIDR.
Availability zone	You need one availability zone for a single zone, and you need three for availability zones for multi-zone.
Public subnet	You must have one public subnet with a NAT gateway for public clusters. Private clusters do not need a public subnet.
DNS hostname and resolution	You must ensure that the DNS hostname and resolution are enabled.

### Additional resources

- [Get Started with Amazon VPC](#)
- [HashiCorp Terraform documentation](#)

### 3.1.2. Creating the account-wide STS roles and policies

Before using the Red Hat OpenShift Service on AWS (ROSA) CLI (**rosa**) to create Red Hat OpenShift Service on AWS (ROSA) with hosted control planes (HCP) clusters, create the required account-wide roles and policies, including the Operator policies.



#### NOTE

ROSA with HCP clusters require account and Operator roles with AWS managed policies attached. Customer managed policies are not supported. For more information regarding AWS managed policies for ROSA with HCP clusters, see [AWS managed policies for ROSA account roles](#).

### Prerequisites

- You have completed the AWS prerequisites for ROSA with HCP.
- You have available AWS service quotas.
- You have enabled the ROSA service in the AWS Console.
- You have installed and configured the latest ROSA CLI (**rosa**) on your installation host.
- You have logged in to your Red Hat account by using the ROSA CLI.

### Procedure

1. If they do not exist in your AWS account, create the required account-wide STS roles and attach the policies by running the following command:

```
$ rosa create account-roles --hosted-cp
```

2. Optional: Set your prefix as an environmental variable by running the following command:

```
$ export ACCOUNT_ROLES_PREFIX=<account_role_prefix>
```

- View the value of the variable by running the following command:

```
$ echo $ACCOUNT_ROLES_PREFIX
```

### Example output

```
ManagedOpenShift
```

For more information regarding AWS managed IAM policies for ROSA, see [AWS managed IAM policies for ROSA](#).

### 3.1.3. Creating an OpenID Connect configuration

When using a ROSA with HCP cluster, you must create the OpenID Connect (OIDC) configuration prior to creating your cluster. This configuration is registered to be used with OpenShift Cluster Manager.

#### Prerequisites

- You have completed the AWS prerequisites for ROSA with HCP.
- You have completed the AWS prerequisites for Red Hat OpenShift Service on AWS.
- You have installed and configured the latest Red Hat OpenShift Service on AWS (ROSA) CLI, **rosa**, on your installation host.

#### Procedure

1. To create your OIDC configuration alongside the AWS resources, run the following command:

```
$ rosa create oidc-config --mode=auto --yes
```

This command returns the following information.

### Example output

```
? Would you like to create a Managed (Red Hat hosted) OIDC Configuration Yes
I: Setting up managed OIDC configuration
I: To create Operator Roles for this OIDC Configuration, run the following command and
remember to replace <user-defined> with a prefix of your choice:
  rosa create operator-roles --prefix <user-defined> --oidc-config-id 13cdr6b
If you are going to create a Hosted Control Plane cluster please include '--hosted-cp'
I: Creating OIDC provider using 'arn:aws:iam::4540112244:user/userName'
```

```
? Create the OIDC provider? Yes
```

```
I: Created OIDC provider with ARN 'arn:aws:iam::4540112244:oidc-provider/dvbwgdztaeq9o.cloudfront.net/13cdr6b'
```

When creating your cluster, you must supply the OIDC config ID. The CLI output provides this value for **--mode auto**, otherwise you must determine these values based on **aws** CLI output for **--mode manual**.

- Optional: you can save the OIDC configuration ID as a variable to use later. Run the following command to save the variable:

```
$ export OIDC_ID=<oidc_config_id> 1
```

- In the example output above, the OIDC configuration ID is 13cdr6b.

- View the value of the variable by running the following command:

```
$ echo $OIDC_ID
```

#### Example output

```
13cdr6b
```

### Verification

- You can list the possible OIDC configurations available for your clusters that are associated with your user organization. Run the following command:

```
$ rosa list oidc-config
```

#### Example output

```
ID                MANAGED ISSUER URL
SECRET ARN
2330dbs0n8m3chkk25gkkcd8pnj3lk2 true
https://dvbwgdztaeq9o.cloudfront.net/2330dbs0n8m3chkk25gkkcd8pnj3lk2
233hvnjrjoqu14jltk6lhbhf2tj11f8un false https://oidc-r7u1.s3.us-east-1.amazonaws.com
aws:secretsmanager:us-east-1:242819244:secret:rosa-private-key-oidc-r7u1-tM3MDN
```

### 3.1.4. Creating Operator roles and policies

When using a ROSA with HCP cluster, you must create the Operator IAM roles that are required for Red Hat OpenShift Service on AWS (ROSA) with hosted control planes (HCP) deployments. The cluster Operators use the Operator roles to obtain the temporary permissions required to carry out cluster operations, such as managing back-end storage, cloud provider credentials, and external access to a cluster.

#### Prerequisites

- You have completed the AWS prerequisites for ROSA with HCP.

- You have installed and configured the latest Red Hat OpenShift Service on AWS ROSA CLI (**rosa**), on your installation host.
- You created the account-wide AWS roles.

## Procedure

1. Set your prefix name to an environment variable using the following command:

```
$ export OPERATOR_ROLES_PREFIX=<prefix_name>
```

2. To create your Operator roles, run the following command:

```
$ rosa create operator-roles --hosted-cp --prefix=$OPERATOR_ROLES_PREFIX --oidc-
config-id=$OIDC_ID --installer-role-arn
arn:aws:iam::${AWS_ACCOUNT_ID}:role/${ACCOUNT_ROLES_PREFIX}-HCP-ROSA-
Installer-Role
```

The following breakdown provides options for the Operator role creation.

```
$ rosa create operator-roles --hosted-cp
--prefix=$OPERATOR_ROLES_PREFIX 1
--oidc-config-id=$OIDC_ID 2
--installer-role-arn
arn:aws:iam::${AWS_ACCOUNT_ID}:role/${ACCOUNT_ROLES_PREFIX}-HCP-ROSA-
Installer-Role 3
```

- 1** You must supply a prefix when creating these Operator roles. Failing to do so produces an error. See the Additional resources of this section for information on the Operator prefix.
- 2** This value is the OIDC configuration ID that you created for your ROSA with HCP cluster.
- 3** This value is the installer role ARN that you created when you created the ROSA account roles.

You must include the **--hosted-cp** parameter to create the correct roles for ROSA with HCP clusters. This command returns the following information.

## Example output

```
? Role creation mode: auto
? Operator roles prefix: <pre-filled_prefix> 1
? OIDC Configuration ID: 23soa2bgvpek9kmes9s7os0a39i13qm4 |
https://dvbwgdztaeq9o.cloudfront.net/23soa2bgvpek9kmes9s7os0a39i13qm4 2
? Create hosted control plane operator roles: Yes
W: More than one Installer role found
? Installer role ARN: arn:aws:iam::4540112244:role/<prefix>-HCP-ROSA-Installer-Role
? Permissions boundary ARN (optional):
I: Reusable OIDC Configuration detected. Validating trusted relationships to operator roles:
I: Creating roles using 'arn:aws:iam::4540112244:user/<userName>'
I: Created role '<prefix>-openshift-cluster-csi-drivers-ebs-cloud-credentials' with ARN
'arn:aws:iam::4540112244:role/<prefix>-openshift-cluster-csi-drivers-ebs-cloud-credentials'
I: Created role '<prefix>-openshift-cloud-network-config-controller-cloud-credenti' with ARN
'arn:aws:iam::4540112244:role/<prefix>-openshift-cloud-network-config-controller-cloud-
```



```

credenti'
I: Created role '<prefix>-kube-system-kube-controller-manager' with ARN
'arn:aws:iam::4540112244:role/<prefix>-kube-system-kube-controller-manager'
I: Created role '<prefix>-kube-system-capa-controller-manager' with ARN
'arn:aws:iam::4540112244:role/<prefix>-kube-system-capa-controller-manager'
I: Created role '<prefix>-kube-system-control-plane-operator' with ARN
'arn:aws:iam::4540112244:role/<prefix>-kube-system-control-plane-operator'
I: Created role '<prefix>-kube-system-kms-provider' with ARN
'arn:aws:iam::4540112244:role/<prefix>-kube-system-kms-provider'
I: Created role '<prefix>-openshift-image-registry-installer-cloud-credentials' with ARN
'arn:aws:iam::4540112244:role/<prefix>-openshift-image-registry-installer-cloud-credentials'
I: Created role '<prefix>-openshift-ingress-operator-cloud-credentials' with ARN
'arn:aws:iam::4540112244:role/<prefix>-openshift-ingress-operator-cloud-credentials'
I: To create a cluster with these roles, run the following command:
  rosa create cluster --sts --oidc-config-id 23soa2bgvpek9kmes9s7os0a39i13qm4 --operator-
  roles-prefix <prefix> --hosted-cp

```

- 1 This field is prepopulated with the prefix that you set in the initial creation command.
- 2 This field requires you to select an OIDC configuration that you created for your ROSA with HCP cluster.

The Operator roles are now created and ready to use for creating your ROSA with HCP cluster.

## Verification

- You can list the Operator roles associated with your ROSA account. Run the following command:

```
$ rosa list operator-roles
```

## Example output

```

I: Fetching operator roles
ROLE PREFIX AMOUNT IN BUNDLE
<prefix> 8
? Would you like to detail a specific prefix Yes 1
? Operator Role Prefix: <prefix>
ROLE NAME ROLE ARN
VERSION MANAGED
<prefix>-kube-system-capa-controller-manager
arn:aws:iam::4540112244:role/<prefix>-kube-system-capa-controller-manager
4.13 No
<prefix>-kube-system-control-plane-operator
arn:aws:iam::4540112244:role/<prefix>-kube-system-control-plane-operator
4.13 No
<prefix>-kube-system-kms-provider
arn:aws:iam::4540112244:role/<prefix>-kube-system-kms-provider 4.13
No
<prefix>-kube-system-kube-controller-manager
arn:aws:iam::4540112244:role/<prefix>-kube-system-kube-controller-manager
4.13 No
<prefix>-openshift-cloud-network-config-controller-cloud-credenti
arn:aws:iam::4540112244:role/<prefix>-openshift-cloud-network-config-controller-cloud-

```

```

credenti 4.13 No
<prefix>-openshift-cluster-csi-drivers-ebs-cloud-credentials
arn:aws:iam::4540112244:role/<prefix>-openshift-cluster-csi-drivers-ebs-cloud-credentials
4.13 No
<prefix>-openshift-image-registry-installer-cloud-credentials
arn:aws:iam::4540112244:role/<prefix>-openshift-image-registry-installer-cloud-credentials
4.13 No
<prefix>-openshift-ingress-operator-cloud-credentials
arn:aws:iam::4540112244:role/<prefix>-openshift-ingress-operator-cloud-credentials
4.13 No

```

- 1 After the command runs, it displays all the prefixes associated with your AWS account and notes how many roles are associated with this prefix. If you need to see all of these roles and their details, enter "Yes" on the detail prompt to have these roles listed out with specifics.

### 3.1.5. Creating a ROSA cluster using a custom AWS KMS key

You can create a Red Hat OpenShift Service on AWS (ROSA) cluster with a customer-provided KMS key that is used to encrypt either node root volumes, the etcd database, or both. A different KMS key ARN can be provided for each option.



#### NOTE

ROSA with HCP does not automatically configure the **default** storage class to encrypt persistent volumes with the customer-provided KMS key. This is something that can be configured in-cluster after installation.

#### Procedure

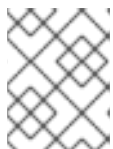
1. Create a custom AWS customer-managed KMS key by running the following command:

```

$ KMS_ARN=$(aws kms create-key --region $AWS_REGION --description 'Custom ROSA Encryption Key' --tags TagKey=red-hat,TagValue=true --query KeyMetadata.Arn --output text)

```

This command saves the Amazon Resource Name (ARN) output of this custom key for further steps.



#### NOTE

Customers must provide the **--tags TagKey=red-hat,TagValue=true** argument that is required for a customer KMS key.

2. Verify the KMS key has been created by running the following command:

```

$ echo $KMS_ARN

```

3. Set your AWS account ID to an environment variable.

```

$ AWS_ACCOUNT_ID=<aws_account_id>

```

4. Add the ARN for the account-wide installer role and operator roles that you created in the preceding step to the **Statement.Principal.AWS** section in the file. In the following example, the ARN for the default **ManagedOpenShift-HCP-ROSA-Installer-Role** role is added:

```
{
  "Version": "2012-10-17",
  "Id": "key-rosa-policy-1",
  "Statement": [
    {
      "Sid": "Enable IAM User Permissions",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::${AWS_ACCOUNT_ID}:root"
      },
      "Action": "kms:*",
      "Resource": "*"
    },
    {
      "Sid": "Installer Permissions",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::${AWS_ACCOUNT_ID}:role/ManagedOpenShift-HCP-ROSA-Installer-Role"
      },
      "Action": [
        "kms:CreateGrant",
        "kms:DescribeKey",
        "kms:GenerateDataKeyWithoutPlaintext"
      ],
      "Resource": "*"
    },
    {
      "Sid": "ROSA KubeControllerManager Permissions",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::${AWS_ACCOUNT_ID}:role/<operator_role_prefix>-kubernetes-kube-controller-manager"
      },
      "Action": "kms:DescribeKey",
      "Resource": "*"
    },
    {
      "Sid": "ROSA KMS Provider Permissions",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::${AWS_ACCOUNT_ID}:role/<operator_role_prefix>-kubernetes-kms-provider"
      },
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:DescribeKey"
      ],
      "Resource": "*"
    }
  ]
}
```

```

    {
      "Sid": "ROSA NodeManager Permissions",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::${AWS_ACCOUNT_ID}:role/<operator_role_prefix>-kube-
system-capa-controller-manager"
      },
      "Action": [
        "kms:DescribeKey",
        "kms:GenerateDataKeyWithoutPlaintext",
        "kms:CreateGrant"
      ],
      "Resource": "*"
    }
  ]
}

```

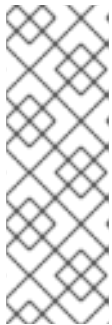
- Confirm the details of the policy file created by running the following command:

```
$ cat rosa-key-policy.json
```

- Apply the newly generated key policy to the custom KMS key by running the following command:

```
$ aws kms put-key-policy --key-id $KMS_ARN \
--policy file://rosa-key-policy.json \
--policy-name default
```

- Create the cluster by running the following command:



#### NOTE

If your cluster name is longer than 15 characters, it will contain an autogenerated domain prefix as a sub-domain for your provisioned cluster on **\*.openshiftapps.com**.

To customize the subdomain, use the **--domain-prefix** flag. The domain prefix cannot be longer than 15 characters, must be unique, and cannot be changed after cluster creation.

```
$ rosa create cluster --cluster-name <cluster_name> \
--subnet-ids <private_subnet_id>,<public_subnet_id> \
--sts \
--mode auto \
--machine-cidr 10.0.0.0/16 \
--compute-machine-type m5.xlarge \
--hosted-cp \
--region <aws_region> \
--oidc-config-id $OIDC_ID \
--kms-key-arn $KMS_ARN \ 1 \
--etcd-encryption-kms-arn $KMS_ARN \ 2 \
--operator-roles-prefix $OPERATOR_ROLES_PREFIX
```

- 1 This KMS key ARN is used to encrypt all worker node root volumes. It is not required if only etcd database encryption is needed.
- 2 This KMS key ARN is used to encrypt the etcd database. The etcd database is always encrypted by default with an AES cipher block, but can be encrypted instead with a KMS key. It is not required if only node root volume encryption is needed.

## Verification

You can verify that your KMS key works by using [OpenShift Cluster Manager](#).

1. Navigate to [OpenShift Cluster Manager](#) and select **Instances**.
2. Select your instance.
3. Click the **Storage** tab.
4. Copy the **KMS key ID**.
5. Search and select **Key Management Service**.
6. Enter your copied *KMS key ID* in the **Filter** field.

## 3.2. NEXT STEPS

- [Accessing a ROSA cluster](#)

## 3.3. ADDITIONAL RESOURCES

- For information on using the CLI to create a cluster, see [Creating a ROSA with HCP cluster using the CLI](#).
- For steps to deploy a ROSA cluster using manual mode, see [Creating a cluster using customizations](#).
- For more information about the AWS Identity Access Management (IAM) resources required to deploy Red Hat OpenShift Service on AWS with STS, see [About IAM resources for clusters that use STS](#).
- For details about optionally setting an Operator role name prefix, see [About custom Operator IAM role prefixes](#).
- For information about the prerequisites to installing ROSA with STS, see [AWS prerequisites for ROSA with STS](#).
- For details about using the **auto** and **manual** modes to create the required STS resources, see [Understanding the auto and manual deployment modes](#).
- For more information about using OpenID Connect (OIDC) identity providers in AWS IAM, see [Creating OpenID Connect \(OIDC\) identity providers](#).
- For more information about troubleshooting ROSA cluster installations, see [Troubleshooting installations](#).
- For steps to contact Red Hat Support for assistance, see [Getting support for Red Hat OpenShift Service on AWS](#).

## CHAPTER 4. CREATING A PRIVATE CLUSTER ON ROSA WITH HCP

This document describes how to create a Red Hat OpenShift Service on AWS (ROSA) with hosted control planes (HCP) private cluster.

### 4.1. CREATING AN AWS PRIVATE CLUSTER

You can create a private cluster with multiple availability zones (Multi-AZ) on ROSA with HCP using the ROSA command line interface (CLI), **rosa**.

#### Prerequisites

- You have available AWS service quotas.
- You have enabled the ROSA service in the AWS Console.
- You have installed and configured the latest version of the ROSA CLI on your installation host.

#### Procedure

Creating a cluster with hosted control planes can take around 10 minutes.

1. Create a VPC with at least one private subnet. Ensure that your machine's classless inter-domain routing (CIDR) matches your virtual private cloud's CIDR. For more information, see [Requirements for using your own VPC](#) and [VPC Validation](#).



#### IMPORTANT

If you use a firewall, you must configure it so that ROSA can access the sites that required to function.

For more information, see the "AWS PrivateLink firewall prerequisites" section.

2. Create the account-wide IAM roles by running the following command:

```
$ rosa create account-roles --hosted-cp
```

3. Create the OIDC configuration by running the following command:

```
$ rosa create oidc-config --mode=auto --yes
```

Save the OIDC configuration ID because you need it to create the Operator roles.

#### Example output

```
I: Setting up managed OIDC configuration
I: To create Operator Roles for this OIDC Configuration, run the following command and
remember to replace <user-defined> with a prefix of your choice:
  rosa create operator-roles --prefix <user-defined> --oidc-config-id
  28s4avcdt2l318r1jbk3ifmimkurk384
If you are going to create a Hosted Control Plane cluster please include '--hosted-cp'
```

```
I: Creating OIDC provider using 'arn:aws:iam::46545644412:user/user'
I: Created OIDC provider with ARN 'arn:aws:iam::46545644412:oidc-provider/oidc.op1.openshiftapps.com/28s4avcdt2l318r1jbk3ifmimkurk384'
```

4. Create the Operator roles by running the following command:

```
$ rosa create operator-roles --hosted-cp --prefix <operator_roles_prefix> --oidc-config-id
<oidc_config_id> --installer-role-arn
arn:aws:iam::$<account_roles_prefix>:role/$<account_roles_prefix>-HCP-ROSA-Installer-
Role
```

5. Create a private ROSA with HCP cluster by running the following command:

```
$ rosa create cluster --private --cluster-name=<cluster-name> --sts --mode=auto --hosted-cp
--operator-roles-prefix <operator_role_prefix> --oidc-config-id <oidc_config_id> [--machine-
cidr=<VPC CIDR>/16] --subnet-ids=<private-subnet-id1>[,<private-subnet-id2>,<private-
subnet-id3>]
```

6. Enter the following command to check the status of your cluster. During cluster creation, the **State** field from the output will transition from **pending** to **installing**, and finally, to **ready**.

```
$ rosa describe cluster --cluster=<cluster_name>
```



#### NOTE

If installation fails or the **State** field does not change to **ready** after 10 minutes, see the "Troubleshooting Red Hat OpenShift Service on AWS installations" documentation in the Additional resources section.

7. Enter the following command to follow the OpenShift installer logs to track the progress of your cluster:

```
$ rosa logs install --cluster=<cluster_name> --watch
```

## 4.2. CONFIGURING AWS SECURITY GROUPS TO ACCESS THE API

With ROSA with HCP private clusters, the AWS PrivateLink endpoint exposed in the customer's VPC has a default security group. This security group has access to the PrivateLink endpoint that is limited to only those resources that exist within the VPC or resources that are present with an IP address associated with the VPC CIDR range. In order to grant access to any entities outside of the VPC, through VPC peering and transit gateway, you must create and attach another security group to the PrivateLink endpoint to grant the necessary access.

### Prerequisites

- Your corporate network or other VPC has connectivity.
- You have permission to create and attach security groups within the VPC.

### Procedure

1. Set your cluster name as an environmental variable by running the following command:

-

```
$ export CLUSTER_NAME=<cluster_name>
```

You can verify that the variable has been set by running the following command:

```
$ echo $CLUSTER_NAME
```

### Example output

```
hcp-private
```

2. Find the VPC endpoint (VPCE) ID and VPC ID by running the following command:

```
$ read -r VPCE_ID VPC_ID <<< $(aws ec2 describe-vpc-endpoints --filters  
"Name=tag:api.openshift.com/id,Values=$(rosa describe cluster -c ${CLUSTER_NAME} -o  
yaml | grep '^id: ' | cut -d' ' -f2)" --query 'VpcEndpoints[][VpcEndpointId,VpcId]' --output text)
```

3. Create your security group by running the following command:

```
$ export SG_ID=$(aws ec2 create-security-group --description "Granting API access to  
${CLUSTER_NAME} from outside of VPC" --group-name "${CLUSTER_NAME}-api-sg" --  
vpc-id $VPC_ID --output text)
```

4. Add an ingress rule to the security group by running the following command:

```
$ aws ec2 authorize-security-group-ingress --group-id $SG_ID --ip-permissions  
FromPort=443,ToPort=443,IpProtocol=tcp,IpRanges=[{CidrIp=0.0.0.0/0}]
```

5. Add the new security group to the VPCE by running the following command:

```
$ aws ec2 modify-vpc-endpoint --vpc-endpoint-id $VPCE_ID --add-security-group-ids  
$SG_ID
```

You now can access the API with your ROSA with HCP private cluster.

## 4.3. NEXT STEPS

[Configuring identity providers](#)

## 4.4. ADDITIONAL RESOURCES

- [AWS PrivateLink firewall prerequisites](#)
- [Overview of the ROSA with STS deployment workflow](#)
- [Deleting a ROSA cluster](#)
- [ROSA architecture models](#)
- [Troubleshooting Red Hat OpenShift Service on AWS installations](#)



## CHAPTER 5. CREATING ROSA WITH HCP CLUSTERS WITH EXTERNAL AUTHENTICATION

You can create Red Hat OpenShift Service on AWS (ROSA) with hosted control planes (HCP) clusters that use external authentication to issue your access tokens.



### IMPORTANT

Since it is not possible to upgrade or convert existing ROSA clusters to a hosted control planes architecture, you must create a new cluster to use ROSA with HCP functionality. You also cannot convert a cluster that was created to use external authentication providers to use the internal OAuth2 server. You must also create a new cluster.



### IMPORTANT

[Sharing VPCs across multiple AWS accounts](#) is not currently supported for ROSA with HCP. Do not install a ROSA with HCP cluster into subnets shared from another AWS account. See "[Are multiple ROSA clusters in a single VPC supported?](#)" for more information.



### NOTE

ROSA with HCP clusters only support Security Token Service (STS) authentication.

### Further reading

- For a comparison between ROSA with HCP and ROSA Classic, see the [Comparing architecture models](#) documentation.
- See the AWS documentation for information about [Getting started with ROSA with HCP using the ROSA CLI in auto mode](#).

### Additional resources

For a full list of the supported certificates, see the [Compliance](#) section of "Understanding process and security for Red Hat OpenShift Service on AWS".

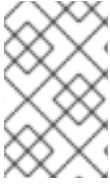
## 5.1. ROSA WITH HCP PREREQUISITES

To create a ROSA with HCP cluster, you must have completed the following steps:

- Completed the [AWS prerequisites](#)
- [Configured virtual private cloud \(VPC\)](#)
- Created [Account-wide roles](#)
- Created an [OIDC configuration](#)
- Created [Operator roles](#)

## 5.2. CREATING A ROSA WITH HCP CLUSTER THAT USES EXTERNAL AUTHENTICATION PROVIDERS

Use the **--external-auth-providers-enabled** flag in the ROSA CLI to create a cluster that uses an external authentication service.



## NOTE

When creating a ROSA with HCP cluster, the default machine Classless Inter-Domain Routing (CIDR) is **10.0.0.0/16**. If this does not correspond to the CIDR range for your VPC subnets, add **--machine-cidr <address\_block>** to the following commands.

## Procedure

- If you used the **OIDC\_ID**, **SUBNET\_IDS**, and **OPERATOR\_ROLES\_PREFIX** variables to prepare your environment, you can continue to use those variables when creating your cluster. For example, run the following command:

```
$ rosa create cluster --hosted-cp --subnet-ids=$SUBNET_IDS \
  --oidc-config-id=$OIDC_ID --cluster-name=<cluster_name> \
  --operator-roles-prefix=$OPERATOR_ROLES_PREFIX \
  --external-auth-providers-enabled
```

- If you did not set environmental variables, run the following command:

```
$ rosa create cluster --cluster-name=<cluster_name> --sts --mode=auto \
  --hosted-cp --operator-roles-prefix <operator-role-prefix> \
  --oidc-config-id <ID-of-OIDC-configuration> \
  --external-auth-providers-enabled \
  --subnet-ids=<public-subnet-id>,<private-subnet-id>
```

## Verification

- Verify that your external authentication is enabled in the cluster details by running the following command:

```
$ rosa describe cluster --cluster=<cluster_name>
```

```
Name:                rosa-ext-test
Display Name:        rosa-ext-test
ID:                  <cluster_id>
External ID:         <cluster_ext_id>
Control Plane:       ROSA Service Hosted
OpenShift Version:   4.16.3
Channel Group:       stable
DNS:                 <dns>
AWS Account:         <AWS_id>
AWS Billing Account: <AWS_id>
API URL:             <ocm_api>
Console URL:
Region:              us-east-1
Availability:
- Control Plane:     MultiAZ
- Data Plane:        SingleAZ

Nodes:
- Compute (desired): 2
```

```

- Compute (current):    0
Network:
- Type:                 OVNKubernetes
- Service CIDR:         <service_cidr>
- Machine CIDR:         <machine_cidr>
- Pod CIDR:             <pod_cidr>
- Host Prefix:          /23
- Subnets:             <subnet_ids>
EC2 Metadata Http Tokens: optional
Role (STS) ARN:         arn:aws:iam::<AWS_id>:role/<account_roles_prefix>-HCP-ROSA-
Installer-Role
Support Role ARN:       arn:aws:iam::<AWS_id>:role/<account_roles_prefix>-HCP-ROSA-
Support-Role
Instance IAM Roles:
- Worker:               arn:aws:iam::<AWS_id>:role/<account_roles_prefix>-HCP-ROSA-
Worker-Role
Operator IAM Roles:
- arn:aws:iam::<AWS_id>:role/<operator_roles_prefix>-openshift-cloud-network-config-
controller-clo
- arn:aws:iam::<AWS_id>:role/<operator_roles_prefix>-kube-system-capac-controller-
manager
- arn:aws:iam::<AWS_id>:role/<operator_roles_prefix>-kube-system-control-plane-operator
- arn:aws:iam::<AWS_id>:role/<operator_roles_prefix>-kube-system-kms-provider
- arn:aws:iam::<AWS_id>:role/<operator_roles_prefix>-kube-system-kube-controller-
manager
- arn:aws:iam::<AWS_id>:role/<operator_roles_prefix>-openshift-image-registry-installer-
cloud-cred
- arn:aws:iam::<AWS_id>:role/<operator_roles_prefix>-openshift-ingress-operator-cloud-
credentials
- arn:aws:iam::<AWS_id>:role/<operator_roles_prefix>-openshift-cluster-csi-drivers-obs-
cloud-crede
Managed Policies:      Yes
State:                  ready
Private:                No
Created:                Mar 29 2024 14:25:52 UTC
User Workload Monitoring: Enabled
Details Page:           https://<url>
OIDC Endpoint URL:      https://<endpoint> (Managed)
Audit Log Forwarding:   Disabled
External Authentication: Enabled ❶

```

- ❶ The **External Authentication** flag is enabled, and you can now create an external authentication provider.

### 5.3. CREATING AN EXTERNAL AUTHENTICATION PROVIDER

After you have created a ROSA with HCP cluster with the enabled option for external authentication providers, you must create a provider using the ROSA CLI.



#### NOTE

Similar to the **rosa create|delete|list idp[s]** command in the ROSA CLI, you cannot edit an existing identity provider that you created using **rosa create external-auth-provider**. Instead, you must delete the external authentication provider and create a new one.

The following table shows the possible CLI flags you can use when creating your external authentication provider:

CLI Flag	Description
<b>--cluster</b>	The name or the ID of your cluster.
<b>--name</b>	A name that is used to refer to the external authentication provider.
<b>--console-client-secret</b>	This string is the client secret that is used to associate your account with the application. If you do not include the client secret, this command uses a public OIDC OAuthClient.
<b>--issuer-audiences</b>	This is a comma-separated list of token audiences.
<b>--issuer-url</b>	The URL of the token issuer.
<b>--claim-mapping-username-claim</b>	The name of the claim that should be used to construct user names for the cluster identity.
<b>--claim-mapping-groups-claim</b>	The name of the claim that should be used to construct group names for the cluster identity.

## Procedure

- To use the interactive command interface, run the following commands:

```
$ rosa create external-auth-provider -c <cluster_name>
```

```
I: Enabling interactive mode
```

```
? Name: 1
```

```
? Issuer audiences: 2
```

```
? The serving url of the token issuer: 3
```

```
? CA file path (optional): 4
```

```
? Claim mapping username: 5
```

```
? Claim mapping groups: 6
```

```
? Claim validation rule (optional): 7
```

```
? Console client id (optional): 8
```

- The name of your external authentication provider. This name should be a lower-case with numbers and dashes.
- The audience IDs that this authentication provider issues tokens for.
- The issuer's URL that serves the token.
- Optional: The certificate file to use when making requests.

- 5 The name of the claim that is used to construct the user names for cluster identity, such as using **email**.
  - 6 The method with which to transform the ID token into a cluster identity, such as using **groups**.
  - 7 Optional: The rules that help validate token claims which authenticate your users. This field should be formatted as **:<required\_value>**.
  - 8 Optional: The application or client ID that your app registration uses for the console.
- You can include the required IDs to create your external authentication provider with the following command:

```
rosa create external-auth-provider --cluster=<cluster_id> \
  --name=<provider_name> --issuer-url=<issuing_url> \
  --issuer-audiences=<audience_id> \
  --claim-mapping-username-claim=email \
  --claim-mapping-groups-claim=groups \
  --console-client-id=<client_id_for_app_registration> \
  --console-client-secret=<client_secret>
```

### Example output

```
I: Successfully created an external authentication provider for cluster '<cluster_id>'
```

### Verification

- To verify your external authentication provider, run one of the following options:
  - List the external authentication configuration on a specified cluster with the following command:

```
$ rosa list external-auth-provider -c <cluster_name>
```

### Example output

The following example shows a configured Microsoft Entra ID external authentication provider:

```
NAME      ISSUER URL
m-entra-id https://login.microsoftonline.com/<group_id>/v2.0
```

- Display the external authentication configuration on a specified cluster by using the following command:

```
$ rosa describe external-auth-provider \
  -c <cluster_name> --name <name_of_external_authentication>
```

### Example output

```
ID:                ms-entra-id
Cluster ID:        <cluster_id>
```

Issuer audiences:

- <audience\_id>

Issuer Url: `https://login.microsoftonline.com/<group_id>/v2.0`

Claim mappings group: `groups`

Claim mappings username: `email`

### Additional resources

- For more information about configuring Entra ID for your IDP, see [What is Microsoft Entra ID?](#) in the Azure documentation or the [Configuring Microsoft Entra ID \(formerly Azure Active Directory\) as an identity provider](#) tutorial section of the documentation.
- For information about the similar **idps** tool in the ROSA CLI, see [create idp](#).
- For more information about options in the ROSA CLI, see [create external-auth-provider](#), [list external-auth-provider](#), and [delete external-auth-provider](#).

## 5.4. CREATING A BREAK GLASS CREDENTIAL FOR A ROSA WITH HCP CLUSTER

As a ROSA with HCP cluster owner, you can use the break glass credential to create temporary administrative client credentials to access your clusters that are configured with custom OpenID Connect (OIDC) token issuers. Creating a break glass credential generates a new cluster-admin **kubeconfig** file. The **kubeconfig** file contains information about the cluster that the CLI uses to connect a client to the correct cluster and API server. You can use the newly generated **kubeconfig** file to allow access to the ROSA with HCP cluster.

### Prerequisites

- You have created a ROSA with HCP cluster with external authentication enabled. For more information, see *Creating a ROSA with HCP with HCP cluster that uses external authentication providers*.
- You have created an external authentication provider. For more information, see *Creating an external authentication provider*.
- You have an account with **cluster admin** permissions.

### Procedure

1. Create a break glass credential by using one of the following commands:
  - To create a break glass credential by using the interactive command interface to interactively specify custom settings, run the following command:

```
$ rosa create break-glass-credential -c <cluster_name> -i 1
```

- 1** Replace <cluster\_name> with the name of your cluster.

This command starts an interactive CLI process:

### Example output

```
I: Enabling interactive mode
? Username (optional): 1
? Expiration duration (optional): 2
I: Successfully created a break glass credential for cluster 'ac-hcp-test'.
```

- 1 If left blank, the value in the **username** will have a randomly generated username value.
- 2 The minimum validity of the break glass credential is 10 minutes, and the maximum validity is 24 hours. If left blank, the expiration duration value defaults to 24 hours.

- To create a break glass credential for cluster called **mycluster** with specified values:

```
$ rosa create break-glass-credential -c mycluster --username test-username --expiration 1h
```

2. List the break glass credential IDs, status, and associated users that are available for a cluster called **mycluster** by running the following command:

```
$ rosa list break-glass-credential -c mycluster
```

### Example output

```
ID                USERNAME  STATUS
2a7jli9n4phe6c02ul7ti91djt2o51d test-user issued
```



### NOTE

You can also view the credentials in a JSON output by adding the **-o json** argument to the command.

3. To view the status of a break glass credential, run the following command, replacing `<break_glass_credential_id>` with the break glass credential ID:

```
$ rosa describe break-glass-credential <break_glass_credential_id> -c <cluster_name>
```

### Example output

```
ID:                2a7jli9n4phe6c02ul7ti91djt2o51d
Username:          test-user
Expire at:         Dec 28 2026 10:23:05 EDT
Status:            issued
```

The following is a list of possible **Status** field values:

- **issued** The break glass credential has been issued and is ready to use.
- **expired** The break glass credential has expired and can no longer be used.
- **failed** The break glass credential has failed to create. In this case, you receive a service log detailing the failure. For more information about service logs, see *Accessing the service logs*

for Red Hat OpenShift Service on AWS clusters. For steps to contact Red Hat Support for assistance, see *Getting support*.

- **awaiting\_revocation** The break glass credential is currently being revoked, meaning it cannot be used.
- **revoked** The break glass credential has been revoked and can no longer be used.

4. To retrieve the **kubeconfig**, run the following commands:

- Create a **kubeconfigs** directory:

```
$ mkdir ~/kubeconfigs
```

- Export the newly generated **kubeconfig** file, replacing <cluster\_name> with the name of your cluster:

```
$ export CLUSTER_NAME=<cluster_name> && export
KUBECONFIG=~/.kubeconfigs/break-glass-{$CLUSTER_NAME}.kubeconfig
```

- View the **kubeconfig**:

```
$ rosa describe break-glass-credential <break_glass_credential_id> -c mycluster --
kubeconfig
```

### Example output

```
apiVersion: v1
clusters:
- cluster:
  server: <server_url>
  name: cluster
contexts:
- context:
  cluster: cluster
  namespace: default
  user: test-username
  name: admin
current-context: admin
kind: Config
preferences: {}
users:
- name: test-user
  user:
    client-certificate-data: <client-certificate-data> 1
    client-key-data: <client-key-data> 2
```

**1** The client-certificate contains a certificate for the user signed by the Kubernetes certificate authorities (CA).

**2** The client-key contains the key that signed the client certificate.

5. Optional: To save the **kubeconfig**, run the following command :

■



```
$ rosa describe break-glass-credential <break_glass_credential_id> -c mycluster --
kubeconfig > $KUBECONFIG
```

### Additional resources

- For more information about creating a ROSA with HCP cluster with external authentication enabled, see [Creating a ROSA with HCP cluster that uses external authentication providers](#) .
- For more information about CLI configurations, see [Managing CLI profiles](#).

## 5.5. ACCESSING A ROSA WITH HCP CLUSTER BY USING A BREAK GLASS CREDENTIAL

Use the new **kubeconfig** from the break glass credential to gain temporary admin access to a ROSA with HCP cluster.

### Prerequisites

- You have access to a ROSA with HCP cluster with external authentication enabled. For more information, see *Creating a ROSA with HCP cluster that uses external authentication providers* .
- You have installed the **oc** and the **kubectl** CLIs.
- You have configured the new **kubeconfig**. For more information, see *Creating a break glass credential for a ROSA with HCP cluster*.

### Procedure

1. Access the details for the cluster:

```
$ rosa describe break-glass-credential <break_glass_credential_id> -c <cluster_name> --
kubeconfig > $KUBECONFIG
```

2. List the nodes from the cluster:

```
$ oc get nodes
```

### Example output

```
NAME                STATUS ROLES AGE VERSION
ip-10-0-0-27.ec2.internal Ready  worker 8m v1.28.7+f1b5f6c
ip-10-0-0-67.ec2.internal Ready  worker 9m v1.28.7+f1b5f6c
```

3. Verify you have the correct credentials:

```
$ kubectl auth whoami
```

### Example output

```
ATTRIBUTE VALUE
Username system:customer-break-glass:test-user
Groups [system:masters system:authenticated]
```

4. Apply the **ClusterRoleBinding** for the groups defined in the external OIDC provider. The **ClusterRoleBinding** maps the **rosa-hcp-admins** group that is created in Microsoft Entra ID to a group in the ROSA with HCP cluster.

```
$ oc apply -f - <<EOF
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: rosa-hcp-admins
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: f715c264-ab90-45d5-8a29-2e91a609a895
EOF
```

### Example output

```
clusterrolebinding.rbac.authorization.k8s.io/rosa-hcp-admins created
```



### NOTE

After the **ClusterRoleBinding** has been applied, the ROSA with HCP cluster is configured, and the **rosa** CLI and the [Red Hat Hybrid Cloud Console](#) are authenticated through the external OpenID Connect (OIDC) provider. You can now start assigning roles and deploying applications on the cluster.

### Additional resources

- For more information about cluster role binding, see [Using RBAC to define and apply permissions](#).

## 5.6. REVOKING A BREAK GLASS CREDENTIAL FOR A ROSA WITH HCP CLUSTER

You can revoke access to any break glass credentials that you have provisioned at any time by using the **revoke break-glass-credentials** command.

### Prerequisites

- You have created a break glass credential.
- You are the cluster owner.

### Procedure

- Revoke the break glass credentials for a ROSA with HCP cluster by running the following command.

**IMPORTANT**

Running this command will revoke access for all break glass credentials related to the cluster.

```
$ rosa revoke break-glass-credentials -c <cluster_name> 1
```

- 1** Replace <cluster\_name> with the name of your cluster.

**Example output**

```
? Are you sure you want to revoke all the break glass credentials on cluster 'my-cluster?': Yes
l: Successfully requested revocation for all break glass credentials from cluster 'my-cluster'
```

**Verification**

- The revocation process can take several minutes. You can verify that the break glass credentials for your clusters have been revoked by running one of the following commands:
  - List all break glass credentials and check the status of each:

```
$ rosa list break-glass-credential -c <cluster_name>
```

**Example output**

```
ID                USERNAME  STATUS
2330db50n8m3chkkr25gkkcd8pnj3lk2 test-user  awaiting_revocation
```

- You can also verify the status by checking the individual credential:

```
$ rosa describe break-glass-credential <break_glass_credential_id> -c <cluster_name>
```

**Example output**

```
ID:                2330db50n8m3chkkr25gkkcd8pnj3lk2
Username:          test-user
Expire at:         Dec 28 2026 10:23:05 EDT
Status:            issued
Revoked at:        Dec 27 2026 15:30:33 EDT
```

**5.7. DELETING AN EXTERNAL AUTHENTICATION PROVIDER**

Delete external authentication providers by using the ROSA CLI.

**Procedure**

1. Display your external authentication provider on your cluster by running the following command:

```
$ rosa list external-auth-provider -c <cluster_name>
```

### Example output

```
NAME      ISSUER URL
entra-test https://login.microsoftonline.com/<group_id>/v2.0
```

2. Delete the external authentication provider by running the following command:

```
$ rosa delete external-auth-provider <name_of_provider> -c <cluster_name>
```

### Example output

```
? Are you sure you want to delete external authentication provider entra-test on cluster rosa-ext-test? Yes
I: Successfully deleted external authentication provider 'entra-test' from cluster 'rosa-ext-test'
```

### Verification

1. Query for any external authentication providers on your cluster by running the following command:

```
$ rosa list external-auth-provider -c <cluster_name>
```

### Example output

```
E: there are no external authentication providers for this cluster
```

## 5.8. ADDITIONAL RESOURCES

- For steps to deploy a ROSA cluster using manual mode, see [Creating a cluster using customizations](#).
- For more information about the AWS Identity Access Management (IAM) resources required to deploy Red Hat OpenShift Service on AWS with STS, see [About IAM resources for clusters that use STS](#).
- To learn more about the default CIDR ranges for Red Hat OpenShift Service on AWS, see [CIDR range definitions](#).
- For details about optionally setting an Operator role name prefix, see [About custom Operator IAM role prefixes](#).
- For information about the prerequisites to installing ROSA with STS, see [AWS prerequisites for ROSA with STS](#).
- For details about using the **auto** and **manual** modes to create the required STS resources, see [Understanding the auto and manual deployment modes](#).
- For more information about using OpenID Connect (OIDC) identity providers in AWS IAM, see [Creating OpenID Connect \(OIDC\) identity providers](#) in the AWS documentation.
- For more information about troubleshooting ROSA cluster installations, see [Troubleshooting installations](#).

- For steps to contact Red Hat Support for assistance, see [Getting support for Red Hat OpenShift Service on AWS](#).

## CHAPTER 6. USING THE NODE TUNING OPERATOR ON ROSA WITH HCP CLUSTERS

Red Hat OpenShift Service on AWS (ROSA) with hosted control planes (HCP) supports the Node Tuning Operator to improve performance of your nodes on your ROSA with HCP clusters. Prior to creating a node tuning configuration, you must create a custom tuning specification.

### Purpose

The Node Tuning Operator helps you manage node-level tuning by orchestrating the TuneD daemon and achieves low latency performance by using the Performance Profile controller. The majority of high-performance applications require some level of kernel tuning. The Node Tuning Operator provides a unified management interface to users of node-level sysctls and more flexibility to add custom tuning specified by user needs.

The Operator manages the containerized TuneD daemon for Red Hat OpenShift Service on AWS as a Kubernetes daemon set. It ensures the custom tuning specification is passed to all containerized TuneD daemons running in the cluster in the format that the daemons understand. The daemons run on all nodes in the cluster, one per node.

Node-level settings applied by the containerized TuneD daemon are rolled back on an event that triggers a profile change or when the containerized TuneD daemon is terminated gracefully by receiving and handling a termination signal.

The Node Tuning Operator uses the Performance Profile controller to implement automatic tuning to achieve low latency performance for Red Hat OpenShift Service on AWS applications.

The cluster administrator configures a performance profile to define node-level settings such as the following:

- Updating the kernel to kernel-rt.
- Choosing CPUs for housekeeping.
- Choosing CPUs for running workloads.

The Node Tuning Operator is part of a standard Red Hat OpenShift Service on AWS installation in version 4.1 and later.



### NOTE

In earlier versions of Red Hat OpenShift Service on AWS, the Performance Addon Operator was used to implement automatic tuning to achieve low latency performance for OpenShift applications. In Red Hat OpenShift Service on AWS 4.11 and later, this functionality is part of the Node Tuning Operator.

## 6.1. CUSTOM TUNING SPECIFICATION

The custom resource (CR) for the Operator has two major sections. The first section, **profile:**, is a list of TuneD profiles and their names. The second, **recommend:**, defines the profile selection logic.

Multiple custom tuning specifications can co-exist as multiple CRs in the Operator's namespace. The existence of new CRs or the deletion of old CRs is detected by the Operator. All existing custom tuning specifications are merged and appropriate objects for the containerized TuneD daemons are updated.

### Management state

The Operator Management state is set by adjusting the default Tuned CR. By default, the Operator is in the Managed state and the **spec.managementState** field is not present in the default Tuned CR. Valid values for the Operator Management state are as follows:

- Managed: the Operator will update its operands as configuration resources are updated
- Unmanaged: the Operator will ignore changes to the configuration resources
- Removed: the Operator will remove its operands and resources the Operator provisioned

### Profile data

The **profile:** section lists TuneD profiles and their names.

```
{
  "profile": [
    {
      "name": "tuned_profile_1",
      "data": "# TuneD profile specification\n[main]\nsummary=Description of tuned_profile_1\nprofile\n\n[sysctl]\nnet.ipv4.ip_forward=1\n# ... other sysctl's or other TuneD daemon plugins\nsupported by the containerized TuneD\n"
    },
    {
      "name": "tuned_profile_n",
      "data": "# TuneD profile specification\n[main]\nsummary=Description of tuned_profile_n\nprofile\n\n# tuned_profile_n profile settings\n"
    }
  ]
}
```

### Recommended profiles

The **profile:** selection logic is defined by the **recommend:** section of the CR. The **recommend:** section is a list of items to recommend the profiles based on a selection criteria.

```
"recommend": [
  {
    "recommend-item-1": details_of_recommendation,
    # ...
    "recommend-item-n": details_of_recommendation,
  }
]
```

The individual items of the list:

```
{
  "profile": [
    {
      # ...
    }
  ],
  "recommend": [
    {
      "profile": <tuned_profile_name>, 1
      "priority": { <priority>, 2

```

```

    },
    "match": [ 3
      {
        "label": <label_information> 4
      },
    ],
  ],
},
]
}

```

- 1 A TuneD profile to apply on a match. For example **tuned\_profile\_1**.
- 2 Profile ordering priority. Lower numbers mean higher priority (**0** is the highest priority).
- 3 If omitted, profile match is assumed unless a profile with a higher priority matches first.
- 4 The label for the profile matched items.

**<match>** is an optional list recursively defined as follows:

```

"match": [
  {
    "label": 1
  },
]

```

- 1 Node or pod label name.

If **<match>** is not omitted, all nested **<match>** sections must also evaluate to **true**. Otherwise, **false** is assumed and the profile with the respective **<match>** section will not be applied or recommended. Therefore, the nesting (child **<match>** sections) works as logical AND operator. Conversely, if any item of the **<match>** list matches, the entire **<match>** list evaluates to **true**. Therefore, the list acts as logical OR operator.

### Example: Node or pod label based matching

```

[
  {
    "match": [
      {
        "label": "tuned.openshift.io/elasticsearch",
        "match": [
          {
            "label": "node-role.kubernetes.io/master"
          },
          {
            "label": "node-role.kubernetes.io/infra"
          }
        ],
      },
    ],
    "type": "pod"
  }
],
"priority": 10,
"profile": "openshift-control-plane-es"

```



```

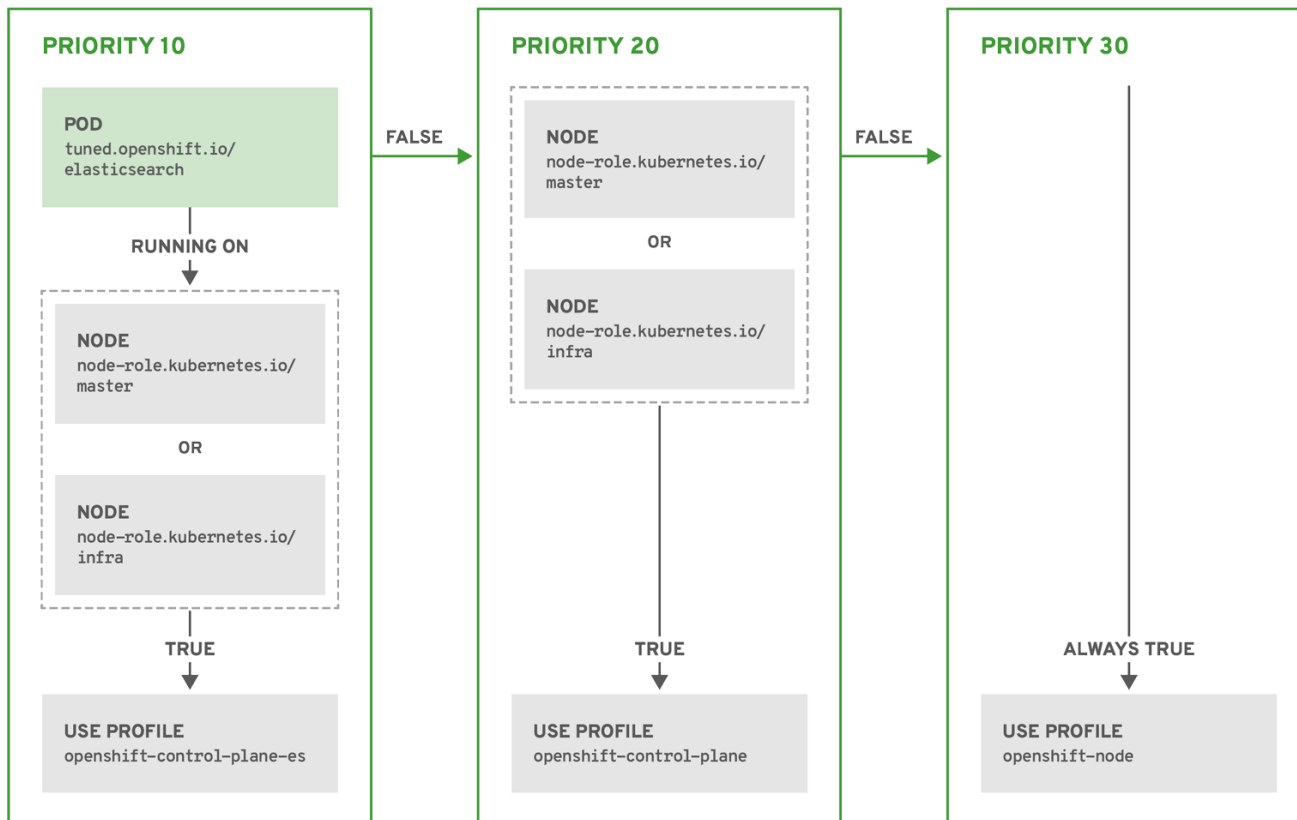
    },
    {
      "match": [
        {
          "label": "node-role.kubernetes.io/master"
        },
        {
          "label": "node-role.kubernetes.io/infra"
        }
      ],
      "priority": 20,
      "profile": "openshift-control-plane"
    },
    {
      "priority": 30,
      "profile": "openshift-node"
    }
  ]
}

```

The CR above is translated for the containerized TuneD daemon into its **recommend.conf** file based on the profile priorities. The profile with the highest priority (**10**) is **openshift-control-plane-es** and, therefore, it is considered first. The containerized TuneD daemon running on a given node looks to see if there is a pod running on the same node with the **tuned.openshift.io/elasticsearch** label set. If not, the entire **<match>** section evaluates as **false**. If there is such a pod with the label, in order for the **<match>** section to evaluate to **true**, the node label also needs to be **node-role.kubernetes.io/master** or **node-role.kubernetes.io/infra**.

If the labels for the profile with priority **10** matched, **openshift-control-plane-es** profile is applied and no other profile is considered. If the node/pod label combination did not match, the second highest priority profile (**openshift-control-plane**) is considered. This profile is applied if the containerized TuneD pod runs on a node with labels **node-role.kubernetes.io/master** or **node-role.kubernetes.io/infra**.

Finally, the profile **openshift-node** has the lowest priority of **30**. It lacks the **<match>** section and, therefore, will always match. It acts as a profile catch-all to set **openshift-node** profile, if no other profile with higher priority matches on a given node.



OPENSIFT\_10\_0319

### Example: Machine pool based matching

```

{
  "apiVersion": "tuned.openshift.io/v1",
  "kind": "Tuned",
  "metadata": {
    "name": "openshift-node-custom",
    "namespace": "openshift-cluster-node-tuning-operator"
  },
  "spec": {
    "profile": [
      {
        "data": "[main]\nsummary=Custom OpenShift node profile with an additional kernel\nparameter\ninclude=openshift-node\n[bootloader]\ncmdline_openshift_node_custom=+skew_tick=1\n",
        "name": "openshift-node-custom"
      }
    ],
    "recommend": [
      {
        "priority": 20,
        "profile": "openshift-node-custom"
      }
    ]
  }
}

```

### Cloud provider-specific Tuned profiles

With this functionality, all Cloud provider-specific nodes can conveniently be assigned a TuneD profile specifically tailored to a given Cloud provider on a Red Hat OpenShift Service on AWS cluster. This can be accomplished without adding additional node labels or grouping nodes into machine pools.

This functionality takes advantage of **spec.providerID** node object values in the form of **<cloud-provider>://<cloud-provider-specific-id>** and writes the file **/var/lib/ocp-tuned/provider** with the value **<cloud-provider>** in NTO operand containers. The content of this file is then used by TuneD to load **provider-<cloud-provider>** profile if such profile exists.

The **openshift** profile that both **openshift-control-plane** and **openshift-node** profiles inherit settings from is now updated to use this functionality through the use of conditional profile loading. Neither NTO nor TuneD currently include any Cloud provider-specific profiles. However, it is possible to create a custom profile **provider-<cloud-provider>** that will be applied to all Cloud provider-specific cluster nodes.

### Example GCE Cloud provider profile

```
{
  "apiVersion": "tuned.openshift.io/v1",
  "kind": "Tuned",
  "metadata": {
    "name": "provider-gce",
    "namespace": "openshift-cluster-node-tuning-operator"
  },
  "spec": {
    "profile": [
      {
        "data": "[main]\nsummary=GCE Cloud provider-specific profile\n# Your tuning for GCE Cloud
provider goes here.\n",
        "name": "provider-gce"
      }
    ]
  }
}
```



#### NOTE

Due to profile inheritance, any setting specified in the **provider-<cloud-provider>** profile will be overwritten by the **openshift** profile and its child profiles.

## 6.2. CREATING NODE TUNING CONFIGURATIONS ON ROSA WITH HCP

You can create tuning configurations using the Red Hat OpenShift Service on AWS (ROSA) CLI, **rosa**.

### Prerequisites

- You have downloaded the latest version of the ROSA CLI.
- You have a cluster on the latest version.
- You have a specification file configured for node tuning.

### Procedure

1. Run the following command to create your tuning configuration:

```
$ rosa create tuning-config -c <cluster_id> --name <name_of_tuning> --spec-path
<path_to_spec_file>
```

You must supply the path to the **spec.json** file or the command returns an error.

### Example output

```
$ I: Tuning config 'sample-tuning' has been created on cluster 'cluster-example'.
$ I: To view all tuning configs, run 'rosa list tuning-configs -c cluster-example'
```

### Validation

- You can verify the existing tuning configurations that are applied by your account with the following command:

```
$ rosa list tuning-configs -c <cluster_name> [-o json]
```

You can specify the type of output you want for the configuration list.

- Without specifying the output type, you see the ID and name of the tuning configuration:

#### Example output without specifying output type

```
ID                                NAME
20468b8e-edc7-11ed-b0e4-0a580a800298  sample-tuning
```

- If you specify an output type, such as **json**, you receive the tuning configuration as JSON text:



#### NOTE

The following JSON output has hard line-returns for the sake of reading clarity. This JSON output is invalid unless you remove the newlines in the JSON strings.

#### Example output specifying JSON output

```
[
  {
    "kind": "TuningConfig",
    "id": "20468b8e-edc7-11ed-b0e4-0a580a800298",
    "href":
"/api/clusters_mgmt/v1/clusters/23jbsevqb22l0m58ps39ua4trff9179e/tuning_configs/20468b8e-edc7-11ed-b0e4-0a580a800298",
    "name": "sample-tuning",
    "spec": {
      "profile": [
        {
          "data": "[main]\nsummary=Custom OpenShift profile\ninclude=openshift-node\n\n[sysctl]\nvm.dirty_ratio=55\n",
          "name": "tuned-1-profile"
        }
      ]
    }
  }
]
```

```

    }
  ],
  "recommend": [
    {
      "priority": 20,
      "profile": "tuned-1-profile"
    }
  ]
}
]

```

## 6.3. MODIFYING YOUR NODE TUNING CONFIGURATIONS FOR ROSA WITH HCP

You can view and update the node tuning configurations using the Red Hat OpenShift Service on AWS (ROSA) CLI, **rosa**.

### Prerequisites

- You have downloaded the latest version of the ROSA CLI.
- You have a cluster on the latest version
- Your cluster has a node tuning configuration added to it

### Procedure

1. You view the tuning configurations with the **rosa describe** command:

```

$ rosa describe tuning-config -c <cluster_id> ❶
  --name <name_of_tuning> ❷
  [-o json] ❸

```

The following items in this spec file are:

- ❶ Provide the cluster ID for the cluster that you own that you want to apply a node tuning configurations.
- ❷ Provide the name of your tuning configuration.
- ❸ Optionally, you can provide the output type. If you do not specify any outputs, you only see the ID and name of the tuning configuration.

### Example output without specifying output type

```

Name:  sample-tuning
ID:    20468b8e-edc7-11ed-b0e4-0a580a800298
Spec:  {
  "profile": [
    {
      "data": "[main]\nsummary=Custom OpenShift profile\ninclude=openshift-
node\n\n[sysctl]\nvm.dirty_ratio=\"55\"\n",

```

```

        "name": "tuned-1-profile"
      }
    ],
    "recommend": [
      {
        "priority": 20,
        "profile": "tuned-1-profile"
      }
    ]
  }
}

```

### Example output specifying JSON output

```

{
  "kind": "TuningConfig",
  "id": "20468b8e-edc7-11ed-b0e4-0a580a800298",
  "href":
"/api/clusters_mgmt/v1/clusters/23jbsevqb2210m58ps39ua4trff9179e/tuning_configs/20468b8e-
edc7-11ed-b0e4-0a580a800298",
  "name": "sample-tuning",
  "spec": {
    "profile": [
      {
        "data": "[main]\nsummary=Custom OpenShift profile\ninclude=openshift-
node\n\n[sysctl]\nvm.dirty_ratio=\"55\"\n",
        "name": "tuned-1-profile"
      }
    ],
    "recommend": [
      {
        "priority": 20,
        "profile": "tuned-1-profile"
      }
    ]
  }
}

```

2. After verifying the tuning configuration, you edit the existing configurations with the **rosa edit** command:

```

$ rosa edit tuning-config -c <cluster_id> --name <name_of_tuning> --spec-path
<path_to_spec_file>

```

In this command, you use the **spec.json** file to edit your configurations.

### Verification

- Run the **rosa describe** command again, to see that the changes you made to the **spec.json** file are updated in the tuning configurations:

```

$ rosa describe tuning-config -c <cluster_id> --name <name_of_tuning>

```

### Example output

```

Name: sample-tuning
ID: 20468b8e-edc7-11ed-b0e4-0a580a800298
Spec: {
  "profile": [
    {
      "data": "[main]\nsummary=Custom OpenShift profile\ninclude=openshift-
node\n\n[sysctl]\nvm.dirty_ratio=55\n",
      "name": "tuned-2-profile"
    }
  ],
  "recommend": [
    {
      "priority": 10,
      "profile": "tuned-2-profile"
    }
  ]
}

```

## 6.4. DELETING NODE TUNING CONFIGURATIONS ON ROSA WITH HCP

You can delete tuning configurations by using the Red Hat OpenShift Service on AWS (ROSA) CLI, **rosa**.



### NOTE

You cannot delete a tuning configuration referenced in a machine pool. You must first remove the tuning configuration from all machine pools before you can delete it.

### Prerequisites

- You have downloaded the latest version of the ROSA CLI.
- You have a cluster on the latest version .
- Your cluster has a node tuning configuration that you want delete.

### Procedure

- To delete the tuning configurations, run the following command:

```
$ rosa delete tuning-config -c <cluster_id> <name_of_tuning>
```

The tuning configuration on the cluster is deleted

### Example output

```
? Are you sure you want to delete tuning config sample-tuning on cluster sample-cluster? Yes
l: Successfully deleted tuning config 'sample-tuning' from cluster 'sample-cluster'
```

## CHAPTER 7. DELETING A ROSA WITH HCP CLUSTER

If you want to delete a Red Hat OpenShift Service on AWS (ROSA) with hosted control planes (HCP) cluster, you can use either the Red Hat OpenShift Cluster Manager or the ROSA command line interface (CLI) (**rosa**). After deleting your cluster, you can also delete the AWS Identity and Access Management (IAM) resources that are used by the cluster.

### 7.1. DELETING A ROSA WITH HCP CLUSTER AND THE CLUSTER-SPECIFIC IAM RESOURCES

You can delete a ROSA with HCP cluster by using the ROSA command line interface (CLI) (**rosa**) or Red Hat OpenShift Cluster Manager.

After deleting the cluster, you can clean up the cluster-specific Identity and Access Management (IAM) resources in your AWS account by using the ROSA CLI. The cluster-specific resources include the Operator roles and the OpenID Connect (OIDC) provider.



#### NOTE

The cluster deletion must complete before you remove the IAM resources, because the resources are used in the cluster deletion and clean up processes.

If add-ons are installed, the cluster deletion takes longer because add-ons are uninstalled before the cluster is deleted. The amount of time depends on the number and size of the add-ons.

#### Prerequisites

- You have installed a ROSA with HCP cluster.
- You have installed and configured the latest ROSA CLI (**rosa**) on your installation host.

#### Procedure

1. Get the cluster ID, the Amazon Resource Names (ARNs) for the cluster-specific Operator roles, and the endpoint URL for the OIDC provider by running the following command:

```
$ rosa describe cluster --cluster=<cluster_name>
```

#### Example output

```
Name:                test_cluster
Domain Prefix:      test_cluster
Display Name:       test_cluster
ID:                 <cluster_id> 1
External ID:        <external_id>
Control Plane:      ROSA Service Hosted
OpenShift Version:  4.16.0
Channel Group:      stable
DNS:                test_cluster.l3cn.p3.openshiftapps.com
AWS Account:        <AWS_id>
AWS Billing Account: <AWS_id>
API URL:            https://api.test_cluster.l3cn.p3.openshiftapps.com:443
Console URL:
```



```

Region:                us-east-1
Availability:
- Control Plane:      MultiAZ
- Data Plane:         SingleAZ

Nodes:
- Compute (desired):  2
- Compute (current):  0

Network:
- Type:               OVNKubernetes
- Service CIDR:       172.30.0.0/16
- Machine CIDR:       10.0.0.0/16
- Pod CIDR:           10.128.0.0/14
- Host Prefix:        /23
- Subnets:           <subnet_ids>
EC2 Metadata Http Tokens: optional
Role (STS) ARN:        arn:aws:iam::<AWS_id>:role/test_cluster-HCP-ROSA-Installer-Role
Support Role ARN:      arn:aws:iam::<AWS_id>:role/test_cluster-HCP-ROSA-Support-
Role
Instance IAM Roles:
- Worker:             arn:aws:iam::<AWS_id>:role/test_cluster-HCP-ROSA-Worker-Role
Operator IAM Roles: 2
- arn:aws:iam::<AWS_id>:role/test_cluster-openshift-cloud-network-config-controller-cloud-
crede
- arn:aws:iam::<AWS_id>:role/test_cluster-openshift-image-registry-installer-cloud-
credentials
- arn:aws:iam::<AWS_id>:role/test_cluster-openshift-ingress-operator-cloud-credentials
- arn:aws:iam::<AWS_id>:role/test_cluster-kube-system-kube-controller-manager
- arn:aws:iam::<AWS_id>:role/test_cluster-kube-system-capac-controller-manager
- arn:aws:iam::<AWS_id>:role/test_cluster-kube-system-control-plane-operator
- arn:aws:iam::<AWS_id>:role/hcpcluster-kube-system-kms-provider
- arn:aws:iam::<AWS_id>:role/test_cluster-openshift-cluster-csi-drivers-efs-cloud-
credentials
Managed Policies:     Yes
State:                 ready
Private:               No
Created:               Apr 16 2024 20:32:06 UTC
User Workload Monitoring: Enabled
Details Page:          https://console.redhat.com/openshift/details/s/<cluster_id>
OIDC Endpoint URL:     https://oidc.op1.openshiftapps.com/<cluster_id> (Managed) 3
Audit Log Forwarding:  Disabled
External Authentication: Disabled

```

- 1 Lists the cluster ID.
- 2 Specifies the ARNs for the cluster-specific Operator roles. For example, in the sample output the ARN for the role required by the Machine Config Operator is **arn:aws:iam::<aws\_account\_id>:role/mycluster-x4q9-openshift-machine-api-aws-cloud-credentials**.
- 3 Displays the endpoint URL for the cluster-specific OIDC provider.


**IMPORTANT**

After the cluster is deleted, you need the cluster ID to delete the cluster-specific STS resources using the ROSA CLI.

2. Delete the cluster by using either the OpenShift Cluster Manager or the ROSA CLI (**rosa**):

- To delete the cluster by using the OpenShift Cluster Manager:

a. Navigate to the [OpenShift Cluster Manager](#).

b. Click the Options menu  next to your cluster and select **Delete cluster**.

c. Type the name of your cluster into the prompt and click **Delete**.

- To delete the cluster using the ROSA CLI:

a. Run the following command, replacing **<cluster\_name>** with the name or ID of your cluster:

```
$ rosa delete cluster --cluster=<cluster_name> --watch
```

**IMPORTANT**

You must wait for cluster deletion to complete before you remove the Operator roles and the OIDC provider.

3. Delete the cluster-specific Operator IAM roles by running the following command:

```
$ rosa delete operator-roles --prefix <operator_role_prefix>
```

4. Delete the OIDC provider by running the following command:

```
$ rosa delete oidc-provider --oidc-config-id <oidc_config_id>
```

**Troubleshooting**

- If the cluster cannot be deleted because of missing IAM roles, see [Repairing a cluster that cannot be deleted](#).
- If the cluster cannot be deleted for other reasons:
  - Ensure that there are no add-ons for your cluster pending in the [Hybrid Cloud Console](#).
  - Ensure that all AWS resources and dependencies have been deleted in the Amazon Web Console.

**7.2. DELETING THE ACCOUNT-WIDE IAM RESOURCES**

After you have deleted all Red Hat OpenShift Service on AWS (ROSA) with hosted control planes (HCP) clusters that depend on the account-wide AWS Identity and Access Management (IAM) resources, you can delete the account-wide resources.

If you no longer need to install a ROSA with HCP cluster by using Red Hat OpenShift Cluster Manager, you can also delete the OpenShift Cluster Manager and user IAM roles.



### IMPORTANT

The account-wide IAM roles and policies might be used by other ROSA with HCP clusters in the same AWS account. Only remove the resources if they are not required by other clusters.

The OpenShift Cluster Manager and user IAM roles are required if you want to install, manage, and delete other Red Hat OpenShift Service on AWS clusters in the same AWS account by using OpenShift Cluster Manager. Only remove the roles if you no longer need to install Red Hat OpenShift Service on AWS clusters in your account by using OpenShift Cluster Manager. For more information about repairing your cluster if these roles are removed before deletion, see "Repairing a cluster that cannot be deleted" in *Troubleshooting cluster deployments*.

#### Additional resources

- [Repairing a cluster that cannot be deleted](#)

### 7.2.1. Deleting the account-wide IAM roles and policies

This section provides steps to delete the account-wide IAM roles and policies that you created for ROSA with HCP deployments, along with the account-wide Operator policies. You can delete the account-wide AWS Identity and Access Management (IAM) roles and policies only after deleting all of the ROSA with HCP clusters that depend on them.



### IMPORTANT

The account-wide IAM roles and policies might be used by other Red Hat OpenShift Service on AWS in the same AWS account. Only remove the roles if they are not required by other clusters.

#### Prerequisites

- You have account-wide IAM roles that you want to delete.
- You have installed and configured the latest ROSA CLI (**rosa**) on your installation host.

#### Procedure

1. Delete the account-wide roles:
  - a. List the account-wide roles in your AWS account by using the ROSA CLI (**rosa**):

```
$ rosa list account-roles
```

#### Example output

```
I: Fetching account roles
ROLE NAME                ROLE TYPE  ROLE ARN
OPENSHIFT VERSION AWS Managed
ManagedOpenShift-HCP-ROSA-Installer-Role  Installer  arn:aws:iam::
```

```

<aws_account_id>:role/ManagedOpenShift-HCP-ROSA-Installer-Role 4.16      Yes
ManagedOpenShift-HCP-ROSA-Support-Role  Support      arn:aws:iam::
<aws_account_id>:role/ManagedOpenShift-HCP-ROSA-Support-Role 4.16
Yes
ManagedOpenShift-HCP-ROSA-Worker-Role  Worker      arn:aws:iam::
<aws_account_id>:role/ManagedOpenShift-HCP-ROSA-Worker-Role 4.16
Yes

```

- b. Delete the account-wide roles:

```
$ rosa delete account-roles --prefix <prefix> --mode auto 1
```

- 1 You must include the `--<prefix>` argument. Replace `<prefix>` with the prefix of the account-wide roles to delete. If you did not specify a custom prefix when you created the account-wide roles, specify the default prefix, **ManagedOpenShift**.



### IMPORTANT

The account-wide IAM roles might be used by other ROSA clusters in the same AWS account. Only remove the roles if they are not required by other clusters.

### Example output

```

W: There are no classic account roles to be deleted
I: Deleting hosted CP account roles
? Delete the account role 'delete-rosa-HCP-ROSA-Installer-Role'? Yes
I: Deleting account role 'delete-rosa-HCP-ROSA-Installer-Role'
? Delete the account role 'delete-rosa-HCP-ROSA-Support-Role'? Yes
I: Deleting account role 'delete-rosa-HCP-ROSA-Support-Role'
? Delete the account role 'delete-rosa-HCP-ROSA-Worker-Role'? Yes
I: Deleting account role 'delete-rosa-HCP-ROSA-Worker-Role'
I: Successfully deleted the hosted CP account roles

```

2. Delete the account-wide in-line and Operator policies:

- a. Under the **Policies** page in the [AWS IAM Console](#), filter the list of policies by the prefix that you specified when you created the account-wide roles and policies.



### NOTE

If you did not specify a custom prefix when you created the account-wide roles, search for the default prefix, **ManagedOpenShift**.

- b. Delete the account-wide in-line policies and Operator policies by using the [AWS IAM Console](#). For more information about deleting IAM policies by using the AWS IAM Console, see [Deleting IAM policies](#) in the AWS documentation.

**IMPORTANT**

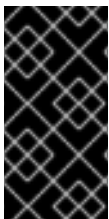
The account-wide in-line and Operator IAM policies might be used by other ROSA with HCP in the same AWS account. Only remove the roles if they are not required by other clusters.

**Additional resources**

- [About IAM resources for ROSA clusters that use STS](#)

**7.2.2. Unlinking and deleting the OpenShift Cluster Manager and user IAM roles**

When you install a ROSA with HCP cluster by using Red Hat OpenShift Cluster Manager, you also create OpenShift Cluster Manager and user Identity and Access Management (IAM) roles that link to your Red Hat organization. After deleting your cluster, you can unlink and delete the roles by using the ROSA CLI (**rosa**).

**IMPORTANT**

The OpenShift Cluster Manager and user IAM roles are required if you want to use OpenShift Cluster Manager to install and manage other ROSA with HCP in the same AWS account. Only remove the roles if you no longer need to use the OpenShift Cluster Manager to install ROSA with HCP clusters.

**Prerequisites**

- You created OpenShift Cluster Manager and user IAM roles and linked them to your Red Hat organization.
- You have installed and configured the latest ROSA CLI (**rosa**) on your installation host.
- You have organization administrator privileges in your Red Hat organization.

**Procedure**

1. Unlink the OpenShift Cluster Manager IAM role from your Red Hat organization and delete the role:
  - a. List the OpenShift Cluster Manager IAM roles in your AWS account:

```
$ rosa list ocm-roles
```

**Example output**

```
I: Fetching ocm roles
ROLE NAME                               ROLE ARN
LINKED ADMIN AWS Managed
ManagedOpenShift-OCM-Role-<red_hat_organization_external_id>  arn:aws:iam::
<aws_account_id>:role/ManagedOpenShift-OCM-Role-
<red_hat_organization_external_id> Yes   Yes   Yes
```

- b. If your OpenShift Cluster Manager IAM role is listed as linked in the output of the preceding command, unlink the role from your Red Hat organization by running the following command:

```
$ rosa unlink ocm-role --role-arn <arn> 1
```

- 1** Replace **<arn>** with the Amazon Resource Name (ARN) for your OpenShift Cluster Manager IAM role. The ARN is specified in the output of the preceding command. In the preceding example, the ARN is in the format **arn:aws:iam::<aws\_account\_id>:role/ManagedOpenShift-OCM-Role-<red\_hat\_organization\_external\_id>**.

### Example output

```
I: Unlinking OCM role
? Unlink the 'arn:aws:iam::<aws_account_id>:role/ManagedOpenShift-OCM-Role-
<red_hat_organization_external_id>' role from organization '<red_hat_organization_id>'?
Yes
I: Successfully unlinked role-arn 'arn:aws:iam::
<aws_account_id>:role/ManagedOpenShift-OCM-Role-
<red_hat_organization_external_id>' from organization account
'<red_hat_organization_id>'
```

- c. Delete the OpenShift Cluster Manager IAM role and policies:

```
$ rosa delete ocm-role --role-arn <arn>
```

### Example output

```
I: Deleting OCM role
? OCM Role ARN: arn:aws:iam::<aws_account_id>:role/ManagedOpenShift-OCM-Role-
<red_hat_organization_external_id>
? Delete 'arn:aws:iam::<aws_account_id>:role/ManagedOpenShift-OCM-Role-
<red_hat_organization_external_id>' ocm role? Yes
? OCM role deletion mode: auto 1
I: Successfully deleted the OCM role
```

- 1** Specifies the deletion mode. You can use **auto** mode to automatically delete the OpenShift Cluster Manager IAM role and policies. In **manual** mode, the ROSA CLI generates the **aws** commands needed to delete the role and policies. **manual** mode enables you to review the details before running the **aws** commands manually.

2. Unlink the user IAM role from your Red Hat organization and delete the role:

- a. List the user IAM roles in your AWS account:

```
$ rosa list user-roles
```

### Example output

```
I: Fetching user roles
ROLE NAME                               ROLE ARN
LINKED
ManagedOpenShift-User-<ocm_user_name>-Role arn:aws:iam::
<aws_account_id>:role/ManagedOpenShift-User-<ocm_user_name>-Role Yes
```

- b. If your user IAM role is listed as linked in the output of the preceding command, unlink the role from your Red Hat organization:

```
$ rosa unlink user-role --role-arn <arn> 1
```

- 1** Replace **<arn>** with the Amazon Resource Name (ARN) for your user IAM role. The ARN is specified in the output of the preceding command. In the preceding example, the ARN is in the format **arn:aws:iam::<aws\_account\_id>:role/ManagedOpenShift-User-<ocm\_user\_name>-Role**.

### Example output

```
I: Unlinking user role
? Unlink the 'arn:aws:iam::<aws_account_id>:role/ManagedOpenShift-User-
<ocm_user_name>-Role' role from the current account '<ocm_user_account_id>'? Yes
I: Successfully unlinked role ARN 'arn:aws:iam::
<aws_account_id>:role/ManagedOpenShift-User-<ocm_user_name>-Role' from account
'<ocm_user_account_id>'
```

- c. Delete the user IAM role:

```
$ rosa delete user-role --role-arn <arn>
```

### Example output

```
I: Deleting user role
? User Role ARN: arn:aws:iam::<aws_account_id>:role/ManagedOpenShift-User-
<ocm_user_name>-Role
? Delete the 'arn:aws:iam::<aws_account_id>:role/ManagedOpenShift-User-
<ocm_user_name>-Role' role from the AWS account? Yes
? User role deletion mode: auto 1
I: Successfully deleted the user role
```

- 1** Specifies the deletion mode. You can use **auto** mode to automatically delete the user IAM role. In **manual** mode, the ROSA CLI generates the **aws** command needed to delete the role. **manual** mode enables you to review the details before running the **aws** command manually.