Red Hat OpenShift Service on AWS 4

Install ROSA with HCP clusters

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

CHAPTER 1. CREATING ROSA WITH HCP CLUSTERS USING THE DEFAULT OPTIONS ................................ 3
1.1. COMPARING ROSA WITH HOSTED CONTROL PLANES AND ROSA CLASSIC ............................ 3
   1.1.1. ROSA architecture network comparisons ....................................................................................... 5
   Considerations regarding auto creation mode ......................................................................................... 6
1.2. OVERVIEW OF THE DEFAULT CLUSTER SPECIFICATIONS ...................................................... 6
1.3. ROSA WITH HCP PREREQUISITES ................................................................................................. 8
   1.3.1. Creating a Virtual Private Cloud for your ROSA with HCP clusters .................................................. 8
   Creating a Virtual Private Cloud using Terraform .................................................................................... 9
   Creating a Virtual Private Cloud manually ............................................................................................ 10
   1.3.2. Creating the account-wide STS roles and policies ....................................................................... 10
   1.3.3. Creating an OpenID Connect configuration ................................................................................. 11
   1.3.4. Creating Operator roles and policies ............................................................................................. 12
1.4. CREATING A ROSA WITH HCP CLUSTER USING THE CLI .................................................... 15
1.5. NEXT STEPS ....................................................................................................................................... 16
1.6. ADDITIONAL RESOURCES ............................................................................................................... 16

CHAPTER 2. USING THE NODE TUNING OPERATOR ON ROSA WITH HCP CLUSTERS ..................... 18
2.1. CUSTOM TUNING SPECIFICATION ................................................................................................. 18
2.2. CREATING NODE TUNING CONFIGURATIONS ON ROSA WITH HCP .................................... 24
2.3. MODIFYING YOUR NODE TUNING CONFIGURATIONS FOR ROSA WITH HCP ..................... 25
2.4. DELETING NODE TUNING CONFIGURATIONS ON ROSA WITH HCP ...................................... 27
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.

**IMPORTANT**
Red Hat OpenShift Service on AWS (ROSA) with hosted control planes (HCP) is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

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

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.

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

### 1.1. COMPARING ROSA WITH HOSTED CONTROL PLANES AND ROSA CLASSIC

Red Hat OpenShift Service on AWS (ROSA) with hosted control planes (HCP) offers a different way to create a managed Red Hat OpenShift Service on AWS (ROSA) cluster. ROSA with HCP offers a reduced-cost solution with focuses on reliability and efficiency. With a focus on efficiency, you can quickly create a new cluster and deploy applications in minutes.

ROSA with HCP requires only a minimum of two nodes making it ideal for smaller projects while still being able to scale to support larger projects and enterprises.

Table 1.1. ROSA architectures comparison table
<table>
<thead>
<tr>
<th>Hosted Control Plane</th>
<th>Classic</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cluster infrastructure hosting</strong></td>
<td>ROSA with HCP deploys control plane components, such as etcd, API server, and oauth, that are hosted separately on AWS in a Red Hat-owned and managed account.</td>
</tr>
<tr>
<td><strong>Provisioning Time</strong></td>
<td>Approximately 10 minutes</td>
</tr>
<tr>
<td><strong>Architecture</strong></td>
<td>- Underlying control plane infrastructure is fully managed and directly unavailable to end customers except through dedicated and explicitly exposed endpoints &lt;br&gt; - Work nodes are hosted in the customer’s AWS account</td>
</tr>
<tr>
<td><strong>Minimum Amazon EC2 footprint</strong></td>
<td>One cluster requires a minimum of two nodes</td>
</tr>
<tr>
<td><strong>Deployment</strong></td>
<td>- Deploy using the ROSA CLI (<strong>rosa</strong>) &lt;br&gt; - Customers provision &quot;Hosted Clusters&quot; that deploy the control plane components into Red Hat’s AWS account &lt;br&gt; - Customers provision &quot;Machine Pools&quot; that deploy worker nodes into the customer’s AWS account</td>
</tr>
<tr>
<td><strong>Upgrades</strong></td>
<td>Selectively upgrade control plane and machine pools separately</td>
</tr>
<tr>
<td><strong>Regional Availability</strong></td>
<td>- Europe - Frankfort (eu-central-1) &lt;br&gt; - Europe - Ireland (eu-west-1) &lt;br&gt; - US East - N. Virginia (us-east-1) &lt;br&gt; - US East - Ohio (us-east-2) &lt;br&gt; - US West - Oregon (us-west-2) &lt;br&gt; - Asia Pacific - Jakarta (ap-southeast-3)</td>
</tr>
</tbody>
</table>
Compliance certifications and FIPS are not yet available. Compliance specifics are located in the Red Hat OpenShift Service on AWS documentation.

### 1.1.1. ROSA architecture network comparisons

ROSA Classic and ROSA with HCP offer options to install your cluster on public and private networks. The following images show the differences between these options.

**Figure 1.1. ROSA Classic deployed on public and private networks**

**Figure 1.2. ROSA with HCP deployed on a public network**

**Figure 1.3. ROSA with HCP deployed on a private network**
Figure 1.3. ROSA with HCP deployed on a private network

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. For steps to deploy a ROSA with HCP cluster by using manual mode or with customizations, see Creating a cluster using customizations.

Next steps

- Ensure that you have completed the AWS prerequisites.

1.2. OVERVIEW OF THE DEFAULT CLUSTER SPECIFICATIONS

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

Table 1.2. Default ROSA with HCP cluster specifications

<table>
<thead>
<tr>
<th>Component</th>
<th>Default specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accounts and roles</td>
<td>● Default IAM role prefix: ManagedOpenShift</td>
</tr>
<tr>
<td>Component</td>
<td>Default specifications</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Cluster settings        | - Default cluster version: Latest  
- Default AWS region for installations using the ROSA CLI (rosa): Defined by your aws CLI configuration  
- Default EC2 IMDS endpoints (both v1 and v2) are enabled  
- Availability: Single zone for the data plane  
- Monitoring for user-defined projects: Enabled |
| Encryption              | - Cloud storage is encrypted at rest  
- Additional etcd encryption is not enabled  
- The default AWS Key Management Service (KMS) key is used as the encryption key for persistent data |
| Compute node machine pool | - Compute node instance type: m5.xlarge (4 vCPU, 16 GiB RAM)  
- Compute node count: 2  
- Autoscaling: Not enabled  
- No additional node labels |
| Networking configuration | - Cluster privacy: Public  
- You must have configured your own Virtual Private Cloud (VPC)  
- No cluster-wide proxy is configured |
### Component | Default specifications
--- | ---
Classless Inter-Domain Routing (CIDR) ranges | - Machine CIDR: 10.0.0.0/16  
- Service CIDR: 172.30.0.0/16  
- Pod CIDR: 10.128.0.0/16  
- Host prefix: /23  
**NOTE**  
When using ROSA with HCP, the static IP address 172.20.0.1 is reserved for the internal Kubernetes API address. The machine, pod, and service CIDRs ranges must not conflict with this IP address.

Cluster roles and policies | - Mode used to create the Operator roles and the OpenID Connect (OIDC) provider: **auto**  
**NOTE**  
For installations using the OpenShift Cluster Manager Hybrid Cloud Console, the auto mode requires an admin-privileged OpenShift Cluster Manager role.  
- Default Operator role prefix: `<cluster_name>-<4_digit_random_string>`

Cluster update strategy | - Individual updates  
- 1 hour grace period for node draining

### 1.3. 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

### 1.3.1. 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:

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

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

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

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

   ```bash
   $ 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.

   ```bash
   $ terraform plan -out rosa.tfplan -var region=<region> [-var cluster_name=<cluster_name>]
   ```

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

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

6. 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)

Verification

- You can verify that the variables were correctly set with the following command:

  $ echo $SUBNET_IDS

  Sample 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 1.3. Requirements for your VPC

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPC name</td>
<td>You need to have the specific VPC name and ID when creating your cluster.</td>
</tr>
<tr>
<td>CIDR range</td>
<td>Your VPC CIDR range should match your machine CIDR.</td>
</tr>
<tr>
<td>Availability zone</td>
<td>You need one availability zone for a single zone, and you need three for multi-zone.</td>
</tr>
<tr>
<td>Public subnet</td>
<td>You must have one public subnet with a NAT gateway.</td>
</tr>
<tr>
<td>DNS hostname and resolution</td>
<td>You must ensure that the DNS hostname and resolution are enabled.</td>
</tr>
</tbody>
</table>

Additional resources

- Get Started with Amazon VPC
- HashiCorp Terraform documentation

1.3.2. 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.
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.

Note

To successfully install ROSA with HCP clusters, use the latest version of the ROSA CLI (rosa).

- 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 policies by running the following command:

   $ rosa create account-roles --force-policy-creation

   The --force-policy-creation parameter updates any existing roles and policies that are present. If no roles and policies are present, the command creates these resources instead.

1.3.3. 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 OCM.

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.

Procedure

- 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.

Sample 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
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:

  ```bash
  $ export OIDC_ID=30f5dqmk
  $ echo $OIDC_ID
  $ 30f5dqmk
  ```

**Verification**

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

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

**1.3.4. 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**

- To create your Operator roles, run the following command:

```
$ rosa create operator-roles --hosted-cp --prefix <prefix-name> --oidc-config-id <oidc-config-id>
```

The following breakdown provides options for the Operator role creation.

```
$ rosa create operator-roles --hosted-cp
   --prefix <prefix-name>  1
   --oidc-config-id <oidc-config-id>  2
```

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.

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

**Sample output**

```
? Role creation mode: auto
? Operator roles prefix: <pre-filled_prefix> 1
? OIDC Configuration ID: 23soa2bgpek9kmes9s7os0a39i13qm4 | https://dvbwgdztaeq9o.cloudfront.net/23soa2bgpek9kmes9s7os0a39i13qm4 2
? Create hosted control plane operator roles: Yes
W: More than one Installer role found
? Installer role ARN: arn:aws:iam::4540112244:role/<prefix>-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>-kube-system-kube-controller-manager' with ARN 'arn:aws:iam::4540112244:role/<prefix>-kube-system-kube-controller-manager'
I: Created role '<prefix>-kube-system-capac-controller-manager' with ARN 'arn:aws:iam::4540112244:role/<prefix>-kube-system-capac-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'
```
This field is prefilled with the prefix that you set in the initial creation command.

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

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

   $ rosa list operator-roles

   **Sample output**

   I: Fetching operator roles
   ROLE PREFIX AMOUNT IN BUNDLE
   <prefix> 8
   ? Would you like to detail a specific prefix Yes
   ? Operator Role Prefix: <prefix>
   ROLE NAME ROLE ARN
   VERSION MANAGED
   <prefix>-kube-system-capab-controller-manager
   arn:aws:iam::4540112244:role/<prefix>-kube-system-capab-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-configuration
   <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
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.

1.4. 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.

NOTE

To successfully install ROSA clusters, use the latest version of the ROSA CLI (rosa). 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. You can create your ROSA with HCP cluster with one of the following commands.

- Create a cluster with a single, initial machine pool, publicly available API, and publicly available Ingress by running the following command:

  ```bash
  $ 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> --subnet-ids=<public-subnet-id>,<private-subnet-id>
  ```

- Create a cluster with a single, initial machine pool, privately available API, and privately available Ingress by running the following command:
If you used variables like `OIDC_ID` and `SUBNET_IDS`, you can use those references when creating your cluster. For example, run the following command:

```bash
$ rosa create cluster --private --cluster-name=<cluster_name> \
   --sts --mode=auto --hosted-cp --subnet-ids=<private-subnet-id>
```

- If you used variables like `OIDC_ID` and `SUBNET_IDS`, you can use those references when creating your cluster. For example, run the following command:

```bash
$ rosa create cluster --hosted-cp --subnet-ids=$SUBNET_IDS --oidc-config-id=$OIDC_ID --cluster-name=<cluster_name>
```

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

```bash
$ 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:

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

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

### 1.5. NEXT STEPS

- Accessing a ROSA cluster

### 1.6. 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*. 
• 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. 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.

**IMPORTANT**

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

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

**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.

**NOTE**

Currently, disabling CPU load balancing is not supported by cgroup v2. As a result, you might not get the desired behavior from performance profiles if you have cgroup v2 enabled. Enabling cgroup v2 is not recommended if you are using performance profiles.

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.

2.1. 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.

```json
{
  "profile": [
    {
      "name": "tuned_profile_1",
      "data": "# TuneD profile specification
[main]
summary=Description of tuned_profile_1
profile

[sysctl]
net.ipv4.ip_forward=1
# ... other sysctl's or other TuneD daemon plugins supported by the containerized TuneD
"},
    {
      "name": "tuned_profile_n",
      "data": "# TuneD profile specification
[main]
summary=Description of tuned_profile_n
profile

# tuned_profile_n profile settings
"}
  ]
}
```

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.

```json
"recommend": [
  "recommend-item-1": details_of_recommendation,
]```
The individual items of the list:

```
[  
  "profile": [  
    # ...  
  ],
  "recommend": [  
    {  
      "profile": <tuned_profile_name>,  
      "priority": <priority>,  
      "machineConfigLabels": { <Key_Pair_for_MachineConfig>  
    },
    "match": [  
      {  
        "label": <label_information>  
      },
    ],
  ]
]
```

1. Profile ordering priority. Lower numbers mean higher priority (0 is the highest priority).
2. A TuneD profile to apply on a match. For example tuned_profile_1.
3. Optional: A dictionary of key-value pairs MachineConfig labels. The keys must be unique.
4. If omitted, profile match is assumed unless a profile with a higher priority matches first or machineConfigLabels is set.
5. The label for the profile matched items.

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

```
"match": [  
  {  
    "label":  
  }
]
```

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.

If `machineConfigLabels` is defined, machine config pool based matching is turned on for the given `recommend` list item. `<mcLabels>` specifies the labels for a machine config. The machine config is created automatically to apply host settings, such as kernel boot parameters, for the profile `<tuned_profile_name>`. This involves finding all machine config pools with machine config selector matching `<mcLabels>` and setting the profile `<tuned_profile_name>` on all nodes that are assigned the found machine config pools. To target nodes that have both master and worker roles, you must use the master role.

The list items `match` and `machineConfigLabels` are connected by the logical OR operator. The `match` item is evaluated first in a short-circuit manner. Therefore, if it evaluates to `true`, the `machineConfigLabels` item is not considered.

**IMPORTANT**

When using machine config pool based matching, it is advised to group nodes with the same hardware configuration into the same machine config pool. Not following this practice might result in TuneD operands calculating conflicting kernel parameters for two or more nodes sharing the same machine config pool.

Example: node or pod label based matching

```json
[  
  {  
    "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"
    }
  ]
]```
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.

**Example: machine config pool based matching**

```json
{
  "apiVersion": "tuned.openshift.io/v1",
  "kind": "Tuned",
  "priority": 30,
  "profile": "openshift-node"
}
```
To minimize node reboots, label the target nodes with a label the machine config pool’s node selector will match, then create the Tuned CR above and finally create the custom machine config pool itself.

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 config 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/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

```json
{
  "apiVersion": "tuned.openshift.io/v1",
  "kind": "Tuned",
  "metadata": {
    "name": "provider-gce",
    "namespace": "openshift-cluster-node-tuning-operator"
  },
  "spec": {
    "profile": [
```
Due to profile inheritance, any setting specified in the `profile-cloud-provider` profile will be overwritten by the `openshift` profile and its child profiles.

### 2.2. 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:

   ```bash
   $ 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.

   **Sample output**

   ```bash
   $ 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:

  ```bash
  $ 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:

  **Sample output without specifying output type**
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.
```

Sample output specifying JSON output

```
[  
   
   
   "kind": "TuningConfig",  
   "id": "20468b8e-edc7-11ed-b0e4-0a580a800298",  
   "href":  
   
   "api/clusters_mgmt/v1/clusters/23jbsevqb22i0m58ps39ua4trff9179e/tuning_configs/20468b8e-edc7-11ed-b0e4-0a580a800298",  
   "name": "sample-tuning",  
   "spec": {  
      "profile": [  
         {  
            "data": "{\"main\":\nsummary=Custom OpenShift profile\ninclude=openshift-node\n\n\n[\"sysctl\"]\nvmm.dirty_ratio=\"55\"\n",  
            "name": "tuned-1-profile"  
         }  
      ],  
      "recommend": [  
         {  
            "priority": 20,  
            "profile": "tuned-1-profile"  
         }  
      ]  
   }  
]
```

2.3. 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>  
   --name <name_of_tuning>  
   [-o json]
   ```

   The following items in this spec file are:
   
   1. Provide the cluster ID for the cluster that you own that you want to apply a node tuning configurations.
   2. Provide the name of your tuning configuration.
   3. 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.

Sample 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
[sysctl]\nvm.dirty_ratio="55"\n",
         "name": "tuned-1-profile"
      }
   ],  
   "recommend": [  
      {  
         "priority": 20,  
         "profile": "tuned-1-profile"
      }
   ]
}

Sample output specifying JSON output

```json
{
    "kind": "TuningConfig",
    "id": "20468b8e-edc7-11ed-b0e4-0a580a800298",
    "href": "/api/clusters_mgmt/v1/clusters/23bsevqb22l0m58ps39ua4trff9179e/tuning_configs/20468b8e-edc7-11ed-b0e4-0a580a800298",
    "name": "sample-tuning",
    "spec": {
        "profile": [
            {
                "data": 
                    "[main]\nsummary=Custom OpenShift profile\ninclude=openshift-node\n
[sysctl]\nvm.dirty_ratio="55"\n",
                "name": "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>
```

**Sample output**

```
Name:  sample-tuning
ID:    20468b8e-edc7-11ed-b0e4-0a580a800298
Spec:  {
    "profile": [
        {
            "data": ">[main]
            summary=Custom OpenShift profile
            include=openshift-node
            [sysctl]
            vm.dirty_ratio="55"
        },
        "name": "tuned-2-profile"
    ],
    "recommend": [
        {"priority": 10,
        "profile": "tuned-2-profile"
    ]
}
```

## 2.4. 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`. 

[27]
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

Sample output

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