# Red Hat Enterprise Linux 10

## Deploying and managing RHEL on Google Cloud Platform

Obtaining RHEL system images and creating RHEL instances on GCP

Last Updated: 2025-05-22

# Red Hat Enterprise Linux 10 Deploying and managing RHEL on Google Cloud Platform

Obtaining RHEL system images and creating RHEL instances on GCP

## Legal Notice



## Abstract

To use Red Hat Enterprise Linux (RHEL) in a public cloud environment, you can create and deploy RHEL system images on various cloud platforms, including Google Cloud Platform (GCP). You can also create and configure a Red Hat High Availability (HA) cluster on GCP. The following chapters provide instructions for creating cloud RHEL instances and HA clusters on GCP. These processes include installing the required packages and agents, configuring fencing, and installing network resource agents.

# Table of Contents

# PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your feedback on our documentation. Let us know how we can improve it.

**Submitting feedback through Jira (account required)**

1. Log in to the Jira website.
2. Click **Create** in the top navigation bar
3. Enter a descriptive title in the **Summary** field.
4. Enter your suggestion for improvement in the **Description** field. Include links to the relevant parts of the documentation.
5. Click **Create** at the bottom of the dialogue.

# CHAPTER 1. INTRODUCING RHEL ON PUBLIC CLOUD PLATFORMS

Public cloud platforms provide computing resources as a service. Instead of using on-premises hardware, you can run your IT workloads, including Red Hat Enterprise Linux (RHEL) systems, as public cloud instances.

## 1.1. BENEFITS OF USING RHEL IN A PUBLIC CLOUD

RHEL as a cloud instance located on a public cloud platform has the following benefits over RHEL on-premises physical systems or VMs:

- **Flexible and fine-grained allocation of resources**
  A cloud instance of RHEL runs as a VM on a cloud platform, which means a cluster of remote servers maintained by the cloud service provider. Therefore, on the software level, allocating hardware resources to the instance is easily customizable, such as a specific type of CPU or storage.

  In comparison to a local RHEL system, you are also not limited by the capabilities of physical host. Instead, you can choose from a variety of features, based on selections offered by the cloud provider.

- **Space and cost efficiency**
  You do not need to own any on-premise servers to host cloud workloads. This avoids the space, power, and maintenance requirements associated with physical hardware.

  Instead, on public cloud platforms, you pay the cloud provider directly for using a cloud instance. The cost is typically based on the hardware allocated to the instance and the time you spend using it. Therefore, you can optimize your costs based on your requirements.

- **Software-controlled configurations**
  The entire configuration of a cloud instance is saved as data on the cloud platform, and is controlled by software. Therefore, you can easily create, remove, clone, or migrate the instance. A cloud instance is also operated remotely in a cloud provider console and is connected to remote storage by default.

  In addition, you can back up the current state of a cloud instance as a snapshot at any time. Afterwards, you can load the snapshot to restore the instance to the saved state.

- **Separation from the host and software compatibility**
  Similarly to a local VM, the RHEL guest operating system on a cloud instance runs on a virtualized kernel. This kernel is separate from the host operating system and from the *client* system that you use to connect to the instance.

  Therefore, any operating system can be installed on the cloud instance. This means that on a RHEL public cloud instance, you can run RHEL-specific applications that cannot be used on your local operating system.

  In addition, even if the operating system of the instance becomes unstable or is compromised, your client system is not affected in any way.

**Additional resources**

- What is public cloud?

- What is a hyperscaler?
- Types of cloud computing
- Public cloud use cases for RHEL
- Obtaining RHEL for public cloud deployments

## 1.2. PUBLIC CLOUD USE CASES FOR RHEL

Deploying on a public cloud provides many benefits, but might not be the most efficient solution in every scenario. If you are evaluating whether to migrate your RHEL deployments to the public cloud, consider whether your use case will benefit from the advantages of the public cloud.

**Beneficial use cases**

- Deploying public cloud instances is very effective for flexibly increasing and decreasing the active computing power of your deployments, also known as *scaling up* and *scaling down*. Therefore, using RHEL on public cloud is recommended in the following scenarios:
  - Clusters with high peak workloads and low general performance requirements. Scaling up and down based on your demands can be highly efficient in terms of resource costs.
  - Quickly setting up or expanding your clusters. This avoids high upfront costs of setting up local servers.
- Cloud instances are not affected by what happens in your local environment. Therefore, you can use them for backup and disaster recovery.

**Potentially problematic use cases**

- You are running an existing environment that cannot be adjusted. Customizing a cloud instance to fit the specific needs of an existing deployment may not be cost-effective in comparison with your current host platform.
- You are operating with a hard limit on your budget. Maintaining your deployment in a local data center typically provides less flexibility but more control over the maximum resource costs than the public cloud does.

**Next steps**

- Obtaining RHEL for public cloud deployments

**Additional resources**

- Should I migrate my application to the cloud? Here's how to decide.

## 1.3. FREQUENT CONCERNS WHEN MIGRATING TO A PUBLIC CLOUD

Moving your RHEL workloads from a local environment to a public cloud platform might raise concerns about the changes involved. The following are the most commonly asked questions.

**Will my RHEL work differently as a cloud instance than as a local virtual machine?**

In most respects, RHEL instances on a public cloud platform work the same as RHEL virtual machines on a local host, such as an on-premises server. Notable exceptions include:

- Instead of private orchestration interfaces, public cloud instances use provider-specific console interfaces for managing your cloud resources.
- Certain features, such as nested virtualization, may not work correctly. If a specific feature is critical for your deployment, check the feature's compatibility in advance with your chosen public cloud provider.

**Will my data stay safe in a public cloud as opposed to a local server?**

The data in your RHEL cloud instances is in your ownership, and your public cloud provider does not have any access to it.

In addition, major cloud providers support data encryption in transit, which improves the security of data when migrating your virtual machines to the public cloud.

The general security of RHEL public cloud instances is managed as follows:

- Your public cloud provider is responsible for the security of the cloud hypervisor
- Red Hat provides the security features of the RHEL guest operating systems in your instances
- You manage the specific security settings and practices in your cloud infrastructure

**What effect does my geographic region have on the functionality of RHEL public cloud instances?**

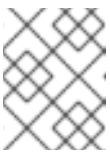
You can use RHEL instances on a public cloud platform regardless of your geographical location. Therefore, you can run your instances in the same region as your on-premises server.

However, hosting your instances in a physically distant region might cause high latency when operating them. In addition, depending on the public cloud provider, certain regions may provide additional features or be more cost-efficient. Before creating your RHEL instances, review the properties of the hosting regions available for your chosen cloud provider.

## 1.4. OBTAINING RHEL FOR PUBLIC CLOUD DEPLOYMENTS

To deploy a RHEL system in a public cloud environment, you need to:

1. Select the optimal cloud provider for your use case, based on your requirements and the current offer on the market.
   The certified cloud service providers (CCSP) for running RHEL instances are:
   - Amazon Web Services (AWS)
   - Google Cloud Platform (GCP)
   - Microsoft Azure

   **NOTE**

   This document specifically addresses the process of deploying RHEL on GCP.

2. Create a RHEL cloud instance on your chosen cloud platform. For details, see Methods for creating RHEL cloud instances.
3. To keep your RHEL deployment up-to-date, use Red Hat Update Infrastructure (RHUI).

**Additional resources**

- RHUI documentation
- Red Hat Open Hybrid Cloud
- Red Hat Ecosystem Cataglogue

## 1.5. METHODS FOR CREATING RHEL CLOUD INSTANCES

To deploy a RHEL instance on a public cloud platform, you can use one of the following methods:

**Create a RHEL system image and import it to the cloud platform**

- To create the system image, you can use the RHEL image builder or you can build the image manually.
- This method uses your existing RHEL subscription, and is also referred to as *bring your own subscription* (BYOS).
- You pre-pay a yearly subscription, and you can use your Red Hat customer discount.
- Your customer service is provided by Red Hat.
- For creating multiple images effectively, you can use the **cloud-init** tool.

**Purchase a RHEL instance directly from the cloud provider marketplace**

- You post-pay an hourly rate for using the service. Therefore, this method is also referred to as *pay as you go* (PAYG).
- Your customer service is provided by the cloud platform provider.

**NOTE**

For detailed instructions on using various methods to deploy RHEL instances on Google Cloud Platform, see the following chapters in this document.

**Additional resources**

- What is a golden image?

# CHAPTER 2. PREPARING AND UPLOADING CUSTOM GCE IMAGES TO GCP

With RHEL image builder, you can build a **gce** image, provide credentials for your user or GCP service account, and then upload the **gce** image directly to the GCP environment.

## 2.1. CONFIGURING AND UPLOADING A GCE IMAGE TO GCP BY USING THE CLI

Set up a configuration file with credentials to upload your **gce** image to GCP by using the RHEL image builder CLI.

**WARNING**

You cannot manually import **gce** image to GCP, because the image will not boot. You must use either **gcloud** or RHEL image builder to upload it.

**Prerequisites**

- You have a valid Google account and credentials to upload your image to GCP. The credentials can be from a user account or a service account. The account associated with the credentials must have at least the following IAM roles assigned:
  - **roles/storage.admin** - to create and delete storage objects
  - **roles/compute.storageAdmin** - to import a VM image to Compute Engine.
- You have an existing GCP bucket.

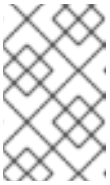
**Procedure**

1. Use a text editor to create a **gcp-config.toml** configuration file with the following content:

```
provider = "gcp"
[settings]
bucket = "<gcp_bucket>"
region = "<gcp_storage_region>"
object = "<object_key>"
credentials = "<gcp_credentials>"
```

   - ***<gcp_bucket>*** points to an existing bucket. It is used to store the intermediate storage object of the image which is being uploaded.
   - ***<gcp_storage_region>*** is both a regular Google storage region and a dual or multi region.
   - ***<object_key>*** is the name of an intermediate storage object. It must not exist before the upload, and it is deleted when the upload process is done. If the object name does not end with **.tar.gz**, the extension is automatically added to the object name.

- ***<gcp_credentials>*** is a **Base64-encoded** scheme of the credentials JSON file downloaded from GCP. The credentials determine which project the GCP uploads the image to.



**NOTE**

Specifying ***<gcp_credentials>*** in the **gcp-config.toml** file is optional if you use a different mechanism to authenticate with GCP. For other authentication methods, see Authenticating with GCP.

2. Retrieve the ***<gcp_credentials>*** from the JSON file downloaded from GCP.

```
$ sudo base64 -w 0 cee-gcp-nasa-476a1fa485b7.json
```

3. Create a compose with an additional image name and cloud provider profile:

```
$ sudo composer-cli compose start <blueprint_name> gce <image_key> gcp-config.toml
```

The image build, upload, and cloud registration processes can take up to ten minutes to complete.

**Verification**

- Verify that the image status is FINISHED:

```
$ sudo composer-cli compose status
```

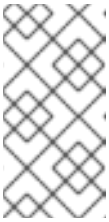
**Additional resources**

- Identity and Access Management
- Create storage buckets

## 2.2. HOW RHEL IMAGE BUILDER SORTS THE AUTHENTICATION ORDER OF DIFFERENT GCP CREDENTIALS

You can use several different types of credentials with RHEL image builder to authenticate with GCP. If RHEL image builder configuration is set to authenticate with GCP by using multiple sets of credentials, it uses the credentials in the following order of preference:

1. Credentials specified with the **composer-cli** command in the configuration file.
2. Credentials configured in the **osbuild-composer** worker configuration.
3. **Application Default Credentials** from the **Google GCP SDK** library, which tries to automatically find a way to authenticate by using the following options:
   a. If the *GOOGLE_APPLICATION_CREDENTIALS* environment variable is set, Application Default Credentials tries to load and use credentials from the file pointed to by the variable.
   b. Application Default Credentials tries to authenticate by using the service account attached to the resource that is running the code. For example, Google Compute Engine VM.



NOTE

You must use the GCP credentials to determine which GCP project to upload the image to. Therefore, unless you want to upload all of your images to the same GCP project, you always must specify the credentials in the **gcp-config.toml** configuration file with the **composer-cli** command.

## 2.3. SPECIFYING GCP CREDENTIALS WITH THE COMPOSER-CLI COMMAND

You can configure GCP authentication credentials to be used for GCP globally for all image builds. This way, if you want to import images to the same GCP project, you can use the same credentials for all image uploads to GCP.

**Procedure**

- In the **/etc/osbuild-worker/osbuild-worker.toml** worker configuration, set the following credential value:

```
[gcp]
credentials = "PATH_TO_GCP_ACCOUNT_CREDENTIALS"
```

## 2.4. SPECIFYING CREDENTIALS IN THE OSBUILD-COMPOSER WORKER CONFIGURATION

You can configure GCP authentication credentials to be used for GCP globally for all image builds. This way, if you want to import images to the same GCP project, you can use the same credentials for all image uploads to GCP.

**Procedure**

- In the **/etc/osbuild-worker/osbuild-worker.toml** worker configuration, set the following credential value:

```
[gcp]
credentials = "PATH_TO_GCP_ACCOUNT_CREDENTIALS"
```

# CHAPTER 3. DEPLOYING A RHEL IMAGE AS A GOOGLE COMPUTE ENGINE INSTANCE ON GCP

To use a RHEL image on Google Cloud Platform (GCP), convert the image to a GCP-compatible format and deploy a VM from the image to run as an Google Compute Engine (GCE) VM. To create, customize, and deploy a RHEL image to GCE supported formats, you can use one of the following methods:

- Use the Red Hat image builder. For instructions, see Preparing and uploading custom GCE images to GCP.
- Manually create and configure a supported format image . This is a more complicated process but offers more granular customization options. For details, see the following sections.

**NOTE**

For a list of Red Hat product certifications for GCP, see Red Hat on Google Cloud Platform.

**Prerequisites**

- You have created a Red Hat account .
- You have signed up and set up a GCP account .

## 3.1. AVAILABLE RHEL IMAGE TYPES FOR PUBLIC CLOUD

To deploy your RHEL virtual machine VM on a certified cloud service provider (CCSP), you can use a number of options. The following table lists the available image types, subscriptions, considerations, and sample scenarios for the image types.

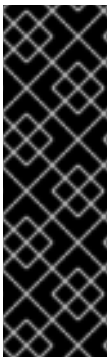
**NOTE**

To deploy customized ISO images, you can use Red Hat Image Builder. With Image Builder, you can create, upload, and deploy these custom images specific to your chosen CCSP.

**Table 3.1. Image options**

| Image types | Subscriptions | Considerations | Sample scenario |
|---|---|---|---|
| Deploy a Red Hat Golden Image | Use your existing Red Hat subscriptions | The subscriptions include the Red Hat product cost and support for Cloud Access images, while you pay the CCSP for all other instance costs | Select a Red Hat Golden Image on the CCSP. For details on Golden Images and how to access them on the CCSP, see the Red Hat Cloud Access Reference Guide |

| Image types | Subscriptions | Considerations | Sample scenario |
|---|---|---|---|
| Deploy a custom image that you move to the CCSP | Use your existing Red Hat subscriptions | The subscriptions includes the Red Hat product cost and support for custom RHEL image, while you pay the CCSP for all other instance costs | Upload your custom image and attach your subscriptions |
| Deploy an existing RHEL based custom machine image | The custom machine images include a RHEL image | You pay the CCSP on an hourly basis based on a *pay-as-you-go* model. For this model, on-demand images are available on the CCSP marketplace. The CCSP provides support for these images, while Red Hat handles updates. The CCSP provides updates through the Red Hat Update Infrastructure (RHUI) | Select a RHEL image when you launch an instance on the CCSP cloud management console, or choose an image from the CCSP marketplace. |



**IMPORTANT**

You cannot convert an on-demand instance to a custom RHEL instance. For migrating from an on-demand image to a custom RHEL bring your own subscription (BYOS) image:

- Create a new custom RHEL instance, then migrate data from your on-demand instance.
- When your data migration is completed, terminate the on-demand instance to avoid additional billing.

**Additional resources**

- Red Hat Ecosystem Cataglogue
- Red Hat Cloud Access Reference Guide
- Compute Engine images
- Create and start a Compute Engine instance

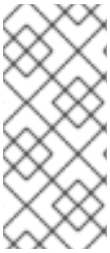
## 3.2. DEPLOYING A RHEL INSTANCE BY USING A CUSTOM BASE IMAGE

To manually configure a virtual machine (VM), first create a base (starter) image. Then, you can modify configuration settings and add the packages the VM requires to operate on the cloud. You can also make additional configuration changes for your specific application after you upload the image.

Creating a VM from a base image has the following advantages:

- Fully customizable
- High flexibility for any use case
- Lightweight - includes only the operating system and the required runtime libraries

To create a custom base image of RHEL from an ISO image, you can use the command line interface (CLI) or the web console for creating and configuring VM.



**NOTE**

Vertify the following VM configurations.

- ssh - ssh must be enabled to provide remote access to your VMs
- dhcp - the primary virtual adapter should be configured for dhcp.

**Prerequisites**

- You have enabled virtualization on the host machine.
- For web console, ensure the following options:
  - You have not checked the **Immediately Start VM** option.
  - You have already changed the **Memory** size to your preferred settings.
  - You have changed the **Model** option under **Virtual Network Interface Settings** to **virtio** and **vCPUs** to the capacity settings for the VM.

**Procedure**

1. Configure the Red Hat Enterprise Linux VM:
   a. To install from the command line (CLI), ensure that you set the default memory, network interfaces, and CPUs as per your requirement for the VM. For details, see Creating virtual machines by using the command line
   b. To install from the web console, see Creating virtual machines by using the web console
2. When the installation starts:
   a. Create a **root** password.
   b. Create an administrative user account.
3. After the installation completes, reboot the VM and log in to the **root** account.
4. After logging in as **root**, you can configure the image.
5. Register the VM and enable the RHEL repository:

```
# subscription-manager register --auto-attach
```

**Verification**

**Verification**

- Verify that the **cloud-init** package is installed and enabled:

```
# dnf install cloud-init
# systemctl enable --now cloud-init.service
```

- Power down the VM.

## 3.3. UPLOADING AND RUNNING A RHEL INSTANCE ON GCP

To run your RHEL instance on Google Cloud Platform (GCP), you need to configure and upload a RHEL image to GCP.

### 3.3.1. Installing the Google Cloud SDK

You can use Google Cloud SDK to manage Google Cloud Platform (GCP) resources and services from your command line by using Google Cloud CLI.

**Prerequisites**

- You have downloaded, extracted, and initialized the Google Cloud SDK.

**Procedure**

1. Use the **gcloud** CLI utility to manage project and instances.
2. Create a project:

```
# gcloud projects create <example-gcp-project-id> --name <example-gcp-project>
```

   The example creates a project with the project ID ***<example-gcp-project-id>*** and the project name ***<example-gcp-project>***.

3. Display project information:

```
# gcloud compute <example-project-info> describe --project <example-project-name>
```

**Additional resources**

- gcloud project create
- gcloud command-line tool overview

### 3.3.2. Creating a new project on GCP

To upload your RHEL image to Google Cloud Platform (GCP), You need to create new project on GCP. A project manages your assigned Google Cloud resources.

**Prerequisites**

- You have an account on Google Cloud Platform.

**Procedure**

1. Launch the GCP Console.
2. Click the drop-down menu to the right of **Google Cloud Platform**.
3. From the pop-up menu, click **NEW PROJECT**.
4. From the **New Project** window, enter a name for your new project.
5. Check **Organization**. Click the drop-down menu to change the organization, if necessary.
6. Confirm the **Location** of your parent organization or folder. Click **Browse** to search for and change this value, if necessary.
7. Click **CREATE** to create your new GCP project.

**Additional resources**

- Creating and Managing Resources in Google Cloud

### 3.3.3. Creating and uploading a RHEL image on Google Cloud Storage

Create a GCP storage bucket to import and store objects such as VM images.

**Procedure**

1. Log in to the Google cloud console:

```
# gcloud auth login
```

2. Create a storage bucket:

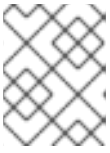
```
# gsutil mb gs://<example-bucket-name>
```

**NOTE**

Alternatively, you can use the Google Cloud Console to create a bucket. For details, see Create a bucket.

3. Specify the image name that you want to create, the existing bucket name, and the name of the image:

```
# **gcloud compute images create my-image-name --source-uri gs://__<example-bucket-name>__/disk.raw.tar.gz**
```



**NOTE**

Alternatively, you can use the Google Cloud Console to create an image. See Creating, deleting, and deprecating custom images for more information.

4. Optional: Find the image in the GCP Console.

a. Click the **Navigation** menu to the left of the **Google Cloud Console** banner.

b. Select **Compute Engine** and then **Images**.

5. Run the **qemu-img** command to convert your **qcow2** image to the **raw** format:

```
# qemu-img convert -f qcow2 -O raw rhel-10.0-sample.qcow2 disk.raw
```

6. Compress the image:

```
# tar --format=oldgnu -Sczf disk.raw.tar.gz disk.raw
```

7. Upload the image to the existing bucket:

```
# gsutil cp disk.raw.tar.gz gs://<example-bucket-name>
```

**NOTE**

Upload could take a few minutes.

8. From the **Google Cloud Platform** home screen, click the collapsed menu icon and select **Storage** and then select **Browser**.

9. Click the name of your bucket where **disk.raw.tar.gz** is now listed.

**NOTE**

You can also upload your image by using the **GCP Console**. To do so, click the bucket name and then click **Upload files**.

**Additional resources**

- gcloud compute images create
- Manually importing virtual disks
- Choosing an import method

### 3.3.4. Launching and connecting to a RHEL Google Compute Engine instance

To configure a GCE VM instance from an image, use the GCP Console.

**Procedure**

1. From the GCP Console Dashboard page , click the **Navigation** menu to the left of the Google **Cloud Console banner** and select **Compute Engine** and then select **Images**.

2. Select your image.

3. Click **Create Instance**.

4. On the **Create an instance** page, enter a **Name** for your instance.

5. Choose a **Region** and **Zone**.
6. Choose a **Machine configuration** that meets or exceeds the requirements of your workload.
7. Ensure that **Boot disk** specifies the name of your image.
8. Optional: Under **Firewall**, select **Allow HTTP traffic** or **Allow HTTPS traffic**.
9. Click **Create**.
10. Find your image under **VM instances**.
11. Click the **Navigation** menu to the left of the Google **Cloud Console banner** and select **Compute Engine** and then select **VM instances**.

NOTE

Alternatively, you can use the **gcloud compute instances create** command to create a GCE VM instance from an image.

```
# gcloud compute instances create myinstance3 --zone=us-central1-a --image
test-iso2-image
```

The example creates a VM instance named **myinstance3** in zone **us-central1-a** based upon the existing image **test-iso2-image**. For details, see gcloud compute instances create.

12. Use the **ssh-keygen** utility to generate an SSH key pair to use with GCE by using the public IP address:

```
# ssh-keygen -t rsa -f ~/.ssh/google_compute_engine.pub
```

13. From the GCP Console Dashboard page , click the **Navigation** menu to the left of the Google **Cloud Console banner** and select **Compute Engine** and then select **Metadata**.
14. Click **SSH Keys** and then click **Edit**.
15. Enter the output generated from the **~/.ssh/google_compute_engine.pub** file and click **Save**.
16. Connect to the instance:

```
# ssh -i ~/.ssh/google_compute_engine <username>@<instance_external_ip>
```

NOTE

Alternatively, you can run the **gcloud compute config-ssh** command to populate the config file with aliases for instances. The aliases allow simple SSH connections by instance name. For details, see gcloud compute config-ssh.

**Additional resources**

- Connecting to instances
- Creating and starting a VM instance

## 3.4. ATTACHING RED HAT SUBSCRIPTIONS

Using the **subscription-manager** command, you can register and attach your Red Hat subscription to a RHEL instance.

### Prerequisites

- You have an active Red Hat account.

### Procedure

1. Register your system:

```
# subscription-manager register --auto-attach
```

2. Attach your subscriptions:
   - You can use an activation key to attach subscriptions. See Creating Red Hat Customer Portal Activation Keys for more information.
   - Alternatively, you can manually attach a subscription by using the ID of the subscription pool (Pool ID). See Attaching a host-based subscription to hypervisors .
3. Optional: To collect various system metrics about the instance in the Red Hat Hybrid Cloud Console, you can register the instance with Red Hat Insights.

```
# insights-client register --display-name <display-name-value>
```

   For information on further configuration of Red Hat Insights, see Client Configuration Guide for Red Hat Insights.

### Additional resources

- Creating Red Hat Customer Portal Activation Keys
- Attaching a host-based subscription to hypervisors
- Client Configuration Guide for Red Hat Insights
- Red Hat Cloud Access Reference Guide

# CHAPTER 4. CONFIGURING A RED HAT HIGH AVAILABILITY CLUSTER ON GCP

By using the high availability (HA) cluster solution, you can create a cluster of RHEL nodes to automatically redistribute workloads if a node failure occurs. These HA clusters can also be deployed on public cloud platforms, such as Google Cloud Platform (GCP). Setting up HA clusters on GCP is similar to configuring them in non-cloud environments.

To configure a Red Hat HA cluster on GCP that uses Google Compute Engine (GCE) instances as cluster nodes, see the following sections. You have a number of options for obtaining RHEL images for the cluster. For details, see Available RHEL image types for public cloud .

**Prerequisites**

- You have created a Red Hat account
- You have signed up and set up a GCP account.
- Red Hat Enterprise Linux 10 Server: **rhel-10-server-rpms/8Server/x86_64**
- Red Hat Enterprise Linux 10 Server (High Availability): **rhel-10-server-ha-rpms/8Server/x86_64**
- Your project should have a service account that belongs to a VM instance and not an individual user. See Using the Compute Engine Default Service Account for information about using the default service account instead of creating a separate service account.

If you or your project administrator create a custom service account, the service account should be configured for the following roles.

- Cloud Trace Agent
- Compute Admin
- Compute Network Admin
- Cloud Datastore User
- Logging Admin
- Monitoring Editor
- Monitoring Metric Writer
- Service Account Administrator
- Storage Admin

## 4.1. BENEFITS OF USING HIGH-AVAILABILITY CLUSTERS ON PUBLIC CLOUD PLATFORMS

### Prerequisites

A high-availability (HA) cluster is a set of computers (called *nodes*) that are linked together to run a specific workload. The purpose of HA clusters is to provide redundancy in case of a hardware or software failure. If a node in the HA cluster fails, the Pacemaker cluster resource manager distributes

the workload to other nodes and no noticeable downtime occurs in the services that are running on the cluster.

You can also run HA clusters on public cloud platforms. In this case, you would use virtual machine (VM) instances in the cloud as the individual cluster nodes. Using HA clusters on a public cloud platform has the following benefits:

- *Improved availability*: In case of a VM failure, the workload is quickly redistributed to other nodes, so running services are not disrupted.
- *Scalability*: Additional nodes can be started when demand is high and stopped when demand is low.
- *Cost-effectiveness*: With the pay-as-you-go pricing, you pay only for nodes that are running.
- *Simplified management*: Some public cloud platforms offer management interfaces to make configuring HA clusters easier.

To enable HA on your RHEL systems, Red Hat offers a HA Add-On. You can configure a RHEL cluster with Red Hat HA Add-On to manage HA clusters with groups of RHEL servers. Red Hat HA Add-On provides access to integrated and streamlined tools. With cluster resource manager, fencing agents, and resource agents, you can set up and configure the cluster for automation. The Red Hat HA Add-On provides the following components for automation:

- **Pacemaker**, a cluster resource manager that provides both a command line utility ( **pcs**) and a GUI (**pcsd**) to support multiple nodes
- **Corosync** and **Kronosnet** to create and manage HA clusters
- Resource agents to configure and manage custom applications
- Fencing agents to use cluster on platforms like bare-metal servers and virtual machines

The Red Hat HA Add-On handles critical tasks such as node failures, load balancing, and node health checks to provide fault tolerance and system reliability.

## 4.2. REQUIRED SYSTEM PACKAGES

To create and configure a base image of RHEL, your host system must have the following packages installed.

**Table 4.1. System packages**

| Package | Repository | Description |
|---|---|---|
| libvirt | rhel-10-for-x86_64-appstream-rpms | Open source API, daemon, and management tool for managing platform virtualization |
| virt-install | rhel-10-for-x86_64-appstream-rpms | A command-line utility for building VMs |
| libguestfs | rhel-10-for-x86_64-appstream-rpms | A library for accessing and modifying VM file systems |

| Package | Repository | Description |
|---|---|---|
| guestfs-tools | rhel-10-for-x86_64-appstream-rpms | System administration tools for VMs; includes the **virt-customize** utility |

**Next steps**

- Follow the deployment steps in Deploying a RHEL instance by using a custom base image .

## 4.3. CREATING A CUSTOM VIRTUAL PRIVATE CLOUD NETWORK AND SUBNET

A custom virtual private cloud (VPC) network and subnet are required for a cluster to be configured with a High Availability (HA) function.

**Prerequisites**

- You have installed the Google Cloud SDK .
- You have created a new GCP project .

**Procedure**

1. Launch the GCP Console.
2. Select **VPC networks** under **Networking** in the left navigation pane.
3. Click **Create VPC Network**
4. Enter a name for the VPC network.
5. Under the **New subnet**, create a **Custom subnet** in the region where you want to create the cluster.
6. Click **Create**.

## 4.4. CREATING AND CONFIGURING A BASE GCP INSTANCE

To create and configure a GCP instance that complies with GCP operating and security requirements, complete the following steps.

**Procedure**

1. Create an image from the compressed file in the bucket.

```
$ gcloud compute images create BaseImageName --source-uri
gs://BucketName/BaseImageName.tar.gz
```

Example:

```
$ gcloud compute images create rhel-76-server --source-uri gs://user-rhelha/rhel-server-76.tar.gz
Created [https://www.googleapis.com/compute/v1/projects/MyProject/global/images/rhel-server-76].
NAME          PROJECT                FAMILY  DEPRECATED  STATUS
rhel-76-server  rhel-ha-testing-on-gcp                     READY
```

2. Create a template instance from the image. The minimum size required for a base RHEL instance is n1-standard-2. See gcloud compute instances create for additional configuration options.

```
$ gcloud compute instances create BaseInstanceName --can-ip-forward --machine-type n1-standard-2 --image BaseImageName --service-account ServiceAccountEmail
```

Example:

```
$ gcloud compute instances create rhel-76-server-base-instance --can-ip-forward --machine-type n1-standard-2 --image rhel-76-server --service-account account@project-name-on-gcp.iam.gserviceaccount.com
Created [https://www.googleapis.com/compute/v1/projects/rhel-ha-testing-on-gcp/zones/us-east1-b/instances/rhel-76-server-base-instance].
NAME  ZONE  MACHINE_TYPE  PREEMPTIBLE  INTERNAL_IP  EXTERNAL_IP  STATUS
rhel-76-server-base-instance  us-east1-bn1-standard-2          10.10.10.3   192.227.54.211  RUNNING
```

3. Connect to the instance with an SSH terminal session.

```
$ ssh root@PublicIPaddress
```

4. Update the RHEL software.

   a. Register with Red Hat Subscription Manager (RHSM).

   b. Enable a Subscription Pool ID (or use the **--auto-attach** command).

   c. Disable all repositories.

   ```
   # subscription-manager repos --disable=*
   ```

   d. Enable the following repository.

   ```
   # subscription-manager repos --enable=rhel-10-server-rpms
   ```

   e. Run the **dnf update** command.

   ```
   # dnf update -y
   ```

5. Install the GCP Linux Guest Environment on the running instance (in-place installation). See Install the guest environment in-place for instructions.

6. Select the **CentOS/RHEL** option.

7. Copy the command script and paste it at the command prompt to run the script immediately.
8. Make the following configuration changes to the instance. These changes are based on GCP recommendations for custom images. See gcloudcompute images list for more information.
   a. Edit the **/etc/chrony.conf** file and remove all NTP servers.
   b. Add the following NTP server.

```
metadata.google.internal iburst Google NTP server
```

   c. Remove any persistent network device rules.

```
# rm -f /etc/udev/rules.d/70-persistent-net.rules
# rm -f /etc/udev/rules.d/75-persistent-net-generator.rules
```

   d. Set the network service to start automatically.

```
# chkconfig network on
```

   e. Set the **sshd service** to start automatically.

```
# systemctl enable sshd
# systemctl is-enabled sshd
```

   f. Set the time zone to UTC.

```
# ln -sf /usr/share/zoneinfo/UTC /etc/localtime
```

   g. Optional: Edit the **/etc/ssh/ssh_config** file and add the following lines to the end of the file. This keeps your SSH session active during longer periods of inactivity.

```
# Server times out connections after several minutes of inactivity.
# Keep alive ssh connections by sending a packet every 7 minutes.
ServerAliveInterval 420
```

   h. Edit the **/etc/ssh/sshd_config** file and make the following changes, if necessary. The **ClientAliveInterval 420** setting is optional; this keeps your SSH session active during longer periods of inactivity.

```
PermitRootLogin no
PasswordAuthentication no
AllowTcpForwarding yes
X11Forwarding no
PermitTunnel no
# Compute times out connections after 10 minutes of inactivity.
# Keep ssh connections alive by sending a packet every 7 minutes.
ClientAliveInterval 420
```

9. Disable password access.

```
ssh_pwauth from 1 to 0.
ssh_pwauth: 0
```

**IMPORTANT**

Previously, you enabled password access to allow SSH session access to configure the instance. You must disable password access. All SSH session access must be passwordless.

10. Unregister the instance from the subscription manager.

```
# subscription-manager unregister
```

11. Clean the shell history. Keep the instance running for the next procedure.

```
# export HISTSIZE=0
```

## 4.5. CREATING A SNAPSHOT IMAGE

To preserve the configuration and disk data of a GCP HA instance, create a snapshot of it.

**Procedure**

1. On the running instance, synchronize data to disk.

```
# sync
```

2. On your host system, create the snapshot.

```
$ gcloud compute disks snapshot InstanceName --snapshot-names SnapshotName
```

3. On your host system, create the configured image from the snapshot.

```
$ gcloud compute images create ConfiguredImageFromSnapshot --source-snapshot
SnapshotName
```

**Additional resources**

- Creating Persistent Disk Snapshots

## 4.6. CREATING AN HA NODE TEMPLATE INSTANCE AND HA NODES

After you have configured an image from the snapshot, you can create a node template. Then, you can use this template to create all HA nodes.

**Procedure**

1. Create an instance template:

```
$ gcloud compute instance-templates create InstanceTemplateName --can-ip-forward --
machine-type n1-standard-2 --image ConfiguredImageFromSnapshot --service-account
ServiceAccountEmailAddress
```

Example:

```
-
```

```
$ gcloud compute instance-templates create rhel-10-instance-template --can-ip-forward --
machine-type n1-standard-2 --image rhel-10-gcp-image --service-account account@project-
name-on-gcp.iam.gserviceaccount.com
Created [https://www.googleapis.com/compute/v1/projects/project-name-on-
gcp/global/instanceTemplates/rhel-10-instance-template].
NAME  MACHINE_TYPE   PREEMPTIBLE  CREATION_TIMESTAMP
rhel-101-instance-template   n1-standard-2          2018-07-25T11:09:30.506-07:00
```

2. Create multiple nodes in one zone:

```
# gcloud compute instances create NodeName01 NodeName02 --source-instance-template
InstanceTemplateName --zone RegionZone --network=NetworkName --
subnet=SubnetName
```

Example:

```
$ gcloud compute instances create rhel10-node-01 rhel10-node-02 rhel10-node-03 --source-
instance-template rhel-10-instance-template --zone us-west1-b --network=projectVPC --
subnet=range0
Created [https://www.googleapis.com/compute/v1/projects/project-name-on-gcp/zones/us-
west1-b/instances/rhel81-node-01].
Created [https://www.googleapis.com/compute/v1/projects/project-name-on-gcp/zones/us-
west1-b/instances/rhel81-node-02].
Created [https://www.googleapis.com/compute/v1/projects/project-name-on-gcp/zones/us-
west1-b/instances/rhel81-node-03].
NAME          ZONE        MACHINE_TYPE   PREEMPTIBLE  INTERNAL_IP  EXTERNAL_IP
STATUS
rhel81-node-01  us-west1-b  n1-standard-2              10.10.10.4   192.230.25.81   RUNNING
rhel81-node-02  us-west1-b  n1-standard-2              10.10.10.5   192.230.81.253  RUNNING
rhel81-node-03  us-east1-b  n1-standard-2              10.10.10.6   192.230.102.15  RUNNING
```

## 4.7. INSTALLING THE HIGH AVAILABILITY PACKAGES AND AGENTS FOR GCP

On each of your nodes, you need to install the High Availability packages and agents to be able to configure a Red Hat High Availability cluster on Google Cloud Platform (GCP).

**Procedure**

1. In the Google Cloud Console, select **Compute Engine** and then select **VM instances**.
2. Select the instance, click the arrow next to **SSH**, and select the **View** gcloud command option.
3. Paste this command at a command prompt for passwordless access to the instance.
4. Enable sudo account access and register with Red Hat Subscription Manager.
5. Enable a Subscription Pool ID (or use the **--auto-attach** command).
6. Disable all repositories.

```
# subscription-manager repos --disable=*
```

7. Enable the following repositories.

```
# subscription-manager repos --enable=rhel-10-server-rpms
# subscription-manager repos --enable=rhel-10-for-x86_64-highavailability-rpms
```

8. Install **pcs pacemaker**, the fence agents, and the resource agents.

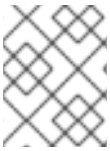
```
# dnf install -y pcs pacemaker fence-agents-gce resource-agents-gcp
```

9. Update all packages.

```
# dnf update -y
```

## 4.8. CREATING A HIGH AVAILABILITY CLUSTER

You create a Red Hat High Availability Add-On cluster with the following procedure. This example procedure creates a cluster that consists of the nodes **z1.example.com** and **z2.example.com**.



NOTE

To display the parameters of a **pcs** command and a description of those parameters, use the **-h** option of the **pcs** command.

**Procedure**

1. Authenticate the **pcs** user **hacluster** for each node in the cluster on the node from which you will be running **pcs**.
   The following command authenticates user **hacluster** on **z1.example.com** for both of the nodes in a two-node cluster that will consist of **z1.example.com** and **z2.example.com**.

```
[root@z1 ~]# pcs host auth z1.example.com z2.example.com
Username: hacluster
Password:
z1.example.com: Authorized
z2.example.com: Authorized
```

2. Execute the following command from **z1.example.com** to create the two-node cluster **my_cluster** that consists of nodes **z1.example.com** and **z2.example.com**. This will propagate the cluster configuration files to both nodes in the cluster. This command includes the **--start** option, which will start the cluster services on both nodes in the cluster.

```
[root@z1 ~]# pcs cluster setup my_cluster --start z1.example.com z2.example.com
```

3. Enable the cluster services to run on each node in the cluster when the node is booted.

NOTE

For your particular environment, you may choose to leave the cluster services disabled by skipping this step. This allows you to ensure that if a node goes down, any issues with your cluster or your resources are resolved before the node rejoins the cluster. If you leave the cluster services disabled, you will need to manually start the services when you reboot a node by executing the **pcs cluster start** command on that node.

```
[root@z1 ~]# pcs cluster enable --all
```

4. Display the status of the cluster you created with the **pcs cluster status** command. Because there may be a slight delay before the cluster is up and running when you start the cluster services with the **--start** option of the **pcs cluster setup** command, you should ensure that the cluster is up and running before performing any subsequent actions on the cluster and its configuration.

```
[root@z1 ~]# pcs cluster status
Cluster Status:
 Stack: corosync
 Current DC: z2.example.com (version 2.0.0-10.el8-b67d8d0de9) - partition with quorum
 Last updated: Thu Oct 11 16:11:18 2018
 Last change: Thu Oct 11 16:11:00 2018 by hacluster via crmd on z2.example.com
 2 Nodes configured
 0 Resources configured

...
```

## 4.9. CONFIGURING FENCING IN A RED HAT HIGH AVAILABILITY CLUSTER

If communication with a single node in the cluster fails, then other nodes in the cluster must be able to restrict or release access to resources that the failed cluster node may have access to. This cannot be accomplished by contacting the cluster node itself as the cluster node may not be responsive. Instead, you must provide an external method, which is called fencing with a fence agent. A fence device is an external device that can be used by the cluster to restrict access to shared resources by an errant node, or to issue a hard reboot on the cluster node.

Without a fence device configured you do not have a way to know that the resources previously used by the disconnected cluster node have been released, and this could prevent the services from running on any of the other cluster nodes. Conversely, the system may assume erroneously that the cluster node has released its resources and this can lead to data corruption and data loss. Without a fence device configured data integrity cannot be guaranteed and the cluster configuration will be unsupported.

When the fencing is in progress no other cluster operation is allowed to run. Normal operation of the cluster cannot resume until fencing has completed or the cluster node rejoins the cluster after the cluster node has been rebooted. For more information about fencing and its importance in a Red Hat High Availability cluster, see the Red Hat Knowledgebase solution Fencing in a Red Hat High Availability Cluster.

### 4.9.1. Displaying available fence agents and their options

The following commands can be used to view available fencing agents and the available options for specific fencing agents.

NOTE

Your system's hardware determines the type of fencing device to use for your cluster. For information about supported platforms and architectures and the different fencing devices, see the Red Hat Knowledgebase article Cluster Platforms and Architectures section of the article Support Policies for RHEL High Availability Clusters .

Run the following command to list all available fencing agents. When you specify a filter, this command displays only the fencing agents that match the filter.

```
pcs stonith list [filter]
```

Run the following command to display the options for the specified fencing agent.

```
pcs stonith describe [stonith_agent]
```

For example, the following command displays the options for the fence agent for APC over telnet/SSH.

```
# pcs stonith describe fence_apc
Stonith options for: fence_apc
  ipaddr (required): IP Address or Hostname
  login (required): Login Name
  passwd: Login password or passphrase
  passwd_script: Script to retrieve password
  cmd_prompt: Force command prompt
  secure: SSH connection
  port (required): Physical plug number or name of virtual machine
  identity_file: Identity file for ssh
  switch: Physical switch number on device
  inet4_only: Forces agent to use IPv4 addresses only
  inet6_only: Forces agent to use IPv6 addresses only
  ipport: TCP port to use for connection with device
  action (required): Fencing Action
  verbose: Verbose mode
  debug: Write debug information to given file
  version: Display version information and exit
  help: Display help and exit
  separator: Separator for CSV created by operation list
  power_timeout: Test X seconds for status change after ON/OFF
  shell_timeout: Wait X seconds for cmd prompt after issuing command
  login_timeout: Wait X seconds for cmd prompt after login
  power_wait: Wait X seconds after issuing ON/OFF
  delay: Wait X seconds before fencing is started
  retry_on: Count of attempts to retry power on
```

**WARNING**

For fence agents that provide a **method** option, with the exception of the **fence_sbd** agent a value of **cycle** is unsupported and should not be specified, as it may cause data corruption. Even for **fence_sbd**, however. you should not specify a method and instead use the default value.

### 4.9.2. Creating a fence device

The format for the command to create a fence device is as follows. For a listing of the available fence device creation options, see the **pcs stonith -h** display.

> pcs stonith create *stonith_id stonith_device_type* [*stonith_device_options*] [op *operation_action operation_options*]

The following command creates a single fencing device for a single node.

> **# pcs stonith create MyStonith fence_virt pcmk_host_list=f1 op monitor interval=30s**

Some fence devices can fence only a single node, while other devices can fence multiple nodes. The parameters you specify when you create a fencing device depend on what your fencing device supports and requires.

- Some fence devices can automatically determine what nodes they can fence.
- You can use the **pcmk_host_list** parameter when creating a fencing device to specify all of the machines that are controlled by that fencing device.
- Some fence devices require a mapping of host names to the specifications that the fence device understands. You can map host names with the **pcmk_host_map** parameter when creating a fencing device.

For information about the **pcmk_host_list** and **pcmk_host_map** parameters, see General properties of fencing devices.

After configuring a fence device, it is imperative that you test the device to ensure that it is working correctly. For information about testing a fence device, see Testing a fence device.

### 4.9.3. General properties of fencing devices

There are many general properties you can set for fencing devices, as well as various cluster properties that determine fencing behavior.

Any cluster node can fence any other cluster node with any fence device, regardless of whether the fence resource is started or stopped. Whether the resource is started controls only the recurring monitor for the device, not whether it can be used, with the following exceptions:

- You can disable a fencing device by running the **pcs stonith disable *stonith_id*** command. This will prevent any node from using that device.

- To prevent a specific node from using a fencing device, you can configure location constraints for the fencing resource with the **pcs constraint location ... avoids** command.
- Configuring **stonith-enabled=false** will disable fencing altogether. Note, however, that Red Hat does not support clusters when fencing is disabled, as it is not suitable for a production environment.

The following table describes the general properties you can set for fencing devices.

**Table 4.2. General Properties of Fencing Devices**

| Field | Type | Default | Description |
|---|---|---|---|
| **pcmk_host_map** | string | | A mapping of host names to port numbers for devices that do not support host names. For example: **node1:1;node2:2,3** tells the cluster to use port 1 for node1 and ports 2 and 3 for node2. The **pcmk_host_map** property supports special characters inside **pcmk_host_map** values using a backslash in front of the value. For example, you can specify **pcmk_host_map="node3:plug\ 1"** to include a space in the host alias. |
| **pcmk_host_list** | string | | A list of machines controlled by this device (Optional unless **pcmk_host_check=static-list**). |
| **pcmk_host_check** | string | * **static-list** if either **pcmk_host_list** or **pcmk_host_map** is set<br>* Otherwise, **dynamic-list** if the fence device supports the **list** action<br>* Otherwise, **status** if the fence device supports the **status** action<br>*Otherwise, **none**. | How to determine which machines are controlled by the device. Allowed values: **dynamic-list** (query the device), **static-list** (check the **pcmk_host_list** attribute), none (assume every device can fence every machine) |

The following table summarizes additional properties you can set for fencing devices. Note that these properties are for advanced use only.

**Table 4.3. Advanced Properties of Fencing Devices**

| Field | Type | Default | Description |
|---|---|---|---|

| Field | Type | Default | Description |
|---|---|---|---|
| **pcmk_host_argument** | string | port | An alternate parameter to supply instead of port. Some devices do not support the standard port parameter or may provide additional ones. Use this to specify an alternate, device-specific parameter that should indicate the machine to be fenced. A value of **none** can be used to tell the cluster not to supply any additional parameters. |
| **pcmk_reboot_action** | string | reboot | An alternate command to run instead of **reboot**. Some devices do not support the standard commands or may provide additional ones. Use this to specify an alternate, device-specific, command that implements the reboot action. |
| **pcmk_reboot_timeout** | time | 60s | Specify an alternate timeout to use for reboot actions instead of **stonith-timeout**. Some devices need much more/less time to complete than normal. Use this to specify an alternate, device-specific, timeout for reboot actions. |
| **pcmk_reboot_retries** | integer | 2 | The maximum number of times to retry the **reboot** command within the timeout period. Some devices do not support multiple connections. Operations may fail if the device is busy with another task so Pacemaker will automatically retry the operation, if there is time remaining. Use this option to alter the number of times Pacemaker retries reboot actions before giving up. |
| **pcmk_off_action** | string | off | An alternate command to run instead of **off**. Some devices do not support the standard commands or may provide additional ones. Use this to specify an alternate, device-specific, command that implements the off action. |
| **pcmk_off_timeout** | time | 60s | Specify an alternate timeout to use for off actions instead of **stonith-timeout**. Some devices need much more or much less time to complete than normal. Use this to specify an alternate, device-specific, timeout for off actions. |

| Field | Type | Default | Description |
|---|---|---|---|
| **pcmk_off_retries** | integer | 2 | The maximum number of times to retry the off command within the timeout period. Some devices do not support multiple connections. Operations may fail if the device is busy with another task so Pacemaker will automatically retry the operation, if there is time remaining. Use this option to alter the number of times Pacemaker retries off actions before giving up. |
| **pcmk_list_action** | string | list | An alternate command to run instead of **list**. Some devices do not support the standard commands or may provide additional ones. Use this to specify an alternate, device-specific, command that implements the list action. |
| **pcmk_list_timeout** | time | 60s | Specify an alternate timeout to use for list actions. Some devices need much more or much less time to complete than normal. Use this to specify an alternate, device-specific, timeout for list actions. |
| **pcmk_list_retries** | integer | 2 | The maximum number of times to retry the **list** command within the timeout period. Some devices do not support multiple connections. Operations may fail if the device is busy with another task so Pacemaker will automatically retry the operation, if there is time remaining. Use this option to alter the number of times Pacemaker retries list actions before giving up. |
| **pcmk_monitor_action** | string | monitor | An alternate command to run instead of **monitor**. Some devices do not support the standard commands or may provide additional ones. Use this to specify an alternate, device-specific, command that implements the monitor action. |
| **pcmk_monitor_timeout** | time | 60s | Specify an alternate timeout to use for monitor actions instead of **stonith-timeout**. Some devices need much more or much less time to complete than normal. Use this to specify an alternate, device-specific, timeout for monitor actions. |

| Field | Type | Default | Description |
|---|---|---|---|
| **pcmk_monitor_retries** | integer | 2 | The maximum number of times to retry the **monitor** command within the timeout period. Some devices do not support multiple connections. Operations may fail if the device is busy with another task so Pacemaker will automatically retry the operation, if there is time remaining. Use this option to alter the number of times Pacemaker retries monitor actions before giving up. |
| **pcmk_status_action** | string | status | An alternate command to run instead of **status**. Some devices do not support the standard commands or may provide additional ones. Use this to specify an alternate, device-specific, command that implements the status action. |
| **pcmk_status_timeout** | time | 60s | Specify an alternate timeout to use for status actions instead of **stonith-timeout**. Some devices need much more or much less time to complete than normal. Use this to specify an alternate, device-specific, timeout for status actions. |
| **pcmk_status_retries** | integer | 2 | The maximum number of times to retry the status command within the timeout period. Some devices do not support multiple connections. Operations may fail if the device is busy with another task so Pacemaker will automatically retry the operation, if there is time remaining. Use this option to alter the number of times Pacemaker retries status actions before giving up. |
| **pcmk_delay_base** | string | 0s | Enables a base delay for fencing actions and specifies a base delay value. You can specify different values for different nodes with the **pcmk_delay_base** parameter. For general information about fencing delay parameters and their interactions, see Fencing delays. |

| Field | Type | Default | Description |
|---|---|---|---|
| **pcmk_delay_max** | time | 0s | Enables a random delay for fencing actions and specifies the maximum delay, which is the maximum value of the combined base delay and random delay. For example, if the base delay is 3 and **pcmk_delay_max** is 10, the random delay will be between 3 and 10. For general information about fencing delay parameters and their interactions, see Fencing delays. |
| **pcmk_action_limit** | integer | 1 | The maximum number of actions that can be performed in parallel on this device. The cluster property **concurrent-fencing=true** needs to be configured first (this is the default value). A value of -1 is unlimited. |
| **pcmk_on_action** | string | on | For advanced use only: An alternate command to run instead of **on**. Some devices do not support the standard commands or may provide additional ones. Use this to specify an alternate, device-specific, command that implements the **on** action. |
| **pcmk_on_timeout** | time | 60s | For advanced use only: Specify an alternate timeout to use for **on** actions instead of **stonith-timeout**. Some devices need much more or much less time to complete than normal. Use this to specify an alternate, device-specific, timeout for **on** actions. |
| **pcmk_on_retries** | integer | 2 | For advanced use only: The maximum number of times to retry the **on** command within the timeout period. Some devices do not support multiple connections. Operations may **fail** if the device is busy with another task so Pacemaker will automatically retry the operation, if there is time remaining. Use this option to alter the number of times Pacemaker retries **on** actions before giving up. |

In addition to the properties you can set for individual fence devices, there are also cluster properties you can set that determine fencing behavior, as described in the following table.

**Table 4.4. Cluster Properties that Determine Fencing Behavior**

| Option | Default | Description |
|---|---|---|
| **stonith-enabled** | true | Indicates that failed nodes and nodes with resources that cannot be stopped should be fenced. Protecting your data requires that you set this **true**.<br><br>If **true**, or unset, the cluster will refuse to start resources unless one or more STONITH resources have been configured also.<br><br>Red Hat only supports clusters with this value set to **true**. |
| **stonith-action** | reboot | Action to send to fencing device. Allowed values: **reboot**, **off**. The value **poweroff** is also allowed, but is only used for legacy devices. |
| **stonith-timeout** | 60s | How long to wait for a STONITH action to complete. |
| **stonith-max-attempts** | 10 | How many times fencing can fail for a target before the cluster will no longer immediately re-attempt it. |
| **stonith-watchdog-timeout** | | The maximum time to wait until a node can be assumed to have been killed by the hardware watchdog. It is recommended that this value be set to twice the value of the hardware watchdog timeout. This option is needed only if watchdog-only SBD configuration is used for fencing. |
| **concurrent-fencing** | true | Allow fencing operations to be performed in parallel. |

| Option | Default | Description |
|---|---|---|
| **fence-reaction** | stop | Determines how a cluster node should react if notified of its own fencing. A cluster node may receive notification of its own fencing if fencing is misconfigured, or if fabric fencing is in use that does not cut cluster communication. Allowed values are **stop** to attempt to immediately stop Pacemaker and stay stopped, or **panic** to attempt to immediately reboot the local node, falling back to stop on failure.<br><br>Although the default value for this property is **stop**, the safest choice for this value is **panic**, which attempts to immediately reboot the local node. If you prefer the stop behavior, as is most likely to be the case in conjunction with fabric fencing, it is recommended that you set this explicitly. |
| **priority-fencing-delay** | 0 (disabled) | Sets a fencing delay that allows you to configure a two-node cluster so that in a split-brain situation the node with the fewest or least important resources running is the node that gets fenced. For general information about fencing delay parameters and their interactions, see Fencing delays. |

For information about setting cluster properties, see https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html/configuring_and_managing cluster-properties

### 4.9.4. Fencing delays

When cluster communication is lost in a two-node cluster, one node may detect this first and fence the other node. If both nodes detect this at the same time, however, each node may be able to initiate fencing of the other, leaving both nodes powered down or reset. By setting a fencing delay, you can decrease the likelihood of both cluster nodes fencing each other. You can set delays in a cluster with more than two nodes, but this is generally not of any benefit because only a partition with quorum will initiate fencing.

You can set different types of fencing delays, depending on your system requirements.

- **static fencing delays**
  A static fencing delay is a fixed, predetermined delay. Setting a static delay on one node makes that node more likely to be fenced because it increases the chances that the other node will initiate fencing first after detecting lost communication. In an active/passive cluster, setting a delay on a passive node makes it more likely that the passive node will be fenced when communication breaks down. You configure a static delay by using the **pcs_delay_base** cluster property. You can set this property when a separate fence device is used for each node or when a single fence device is used for all nodes.

- dynamic fencing delays
  A dynamic fencing delay is random. It can vary and is determined at the time fencing is needed. You configure a random delay and specify a maximum value for the combined base delay and random delay with the **pcs_delay_max** cluster property. When the fencing delay for each node is random, which node is fenced is also random. You may find this feature useful if your cluster is configured with a single fence device for all nodes in an active/active design.

- priority fencing delays
  A priority fencing delay is based on active resource priorities. If all resources have the same priority, the node with the fewest resources running is the node that gets fenced. In most cases, you use only one delay-related parameter, but it is possible to combine them. Combining delay-related parameters adds the priority values for the resources together to create a total delay. You configure a priority fencing delay with the **priority-fencing-delay** cluster property. You may find this feature useful in an active/active cluster design because it can make the node running the fewest resources more likely to be fenced when communication between the nodes is lost.

### The pcmk_delay_base cluster property

Setting the **pcmk_delay_base** cluster property enables a base delay for fencing and specifies a base delay value.

When you set the **pcmk_delay_max** cluster property in addition to the **pcmk_delay_base** property, the overall delay is derived from a random delay value added to this static delay so that the sum is kept below the maximum delay. When you set **pcmk_delay_base** but do not set **pcmk_delay_max**, there is no random component to the delay and it will be the value of **pcmk_delay_base**.

You can specify different values for different nodes with the **pcmk_delay_base** parameter. This allows a single fence device to be used in a two-node cluster, with a different delay for each node. You do not need to configure two separate devices to use separate delays. To specify different values for different nodes, you map the host names to the delay value for that node using a similar syntax to **pcmk_host_map**. For example, **node1:0;node2:10s** would use no delay when fencing **node1** and a 10-second delay when fencing **node2**.

### The pcmk_delay_max cluster property

Setting the **pcmk_delay_max** cluster property enables a random delay for fencing actions and specifies the maximum delay, which is the maximum value of the combined base delay and random delay. For example, if the base delay is 3 and **pcmk_delay_max** is 10, the random delay will be between 3 and 10.

When you set the **pcmk_delay_base** cluster property in addition to the **pcmk_delay_max** property, the overall delay is derived from a random delay value added to this static delay so that the sum is kept below the maximum delay. When you set **pcmk_delay_max** but do not set **pcmk_delay_base** there is no static component to the delay.

### The priority-fencing-delay cluster property

Setting the **priority-fencing-delay** cluster property allows you to configure a two-node cluster so that in a split-brain situation the node with the fewest or least important resources running is the node that gets fenced.

The **priority-fencing-delay** property can be set to a time duration. The default value for this property is 0 (disabled). If this property is set to a non-zero value, and the priority meta-attribute is configured for at least one resource, then in a split-brain situation the node with the highest combined priority of all resources running on it will be more likely to remain operational. For example, if you set **pcs resource defaults update priority=1** and **pcs property set priority-fencing-delay=15s** and no other priorities

are set, then the node running the most resources will be more likely to remain operational because the other node will wait 15 seconds before initiating fencing. If a particular resource is more important than the rest, you can give it a higher priority.

The node running the promoted role of a promotable clone gets an extra 1 point if a priority has been configured for that clone.

#### Interaction of fencing delays

Setting more than one type of fencing delay yields the following results:

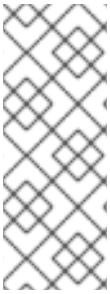
- Any delay set with the **priority-fencing-delay** property is added to any delay from the **pcmk_delay_base** and **pcmk_delay_max** fence device properties. This behavior allows some delay when both nodes have equal priority, or both nodes need to be fenced for some reason other than node loss, as when **on-fail=fencing** is set for a resource monitor operation. When setting these delays in combination, set the **priority-fencing-delay** property to a value that is significantly greater than the maximum delay from **pcmk_delay_base** and **pcmk_delay_max** to be sure the prioritized node is preferred. Setting this property to twice this value is always safe.

- Only fencing scheduled by Pacemaker itself observes fencing delays. Fencing scheduled by external code such as **dlm_controld** and fencing implemented by the **pcs stonith fence** command do not provide the necessary information to the fence device.

- Some individual fence agents implement a delay parameter, with a name determined by the agent and which is independent of delays configured with a **pcmk_delay_*** property. If both of these delays are configured, they are added together and would generally not be used in conjunction.

### 4.9.5. Testing a fence device

Fencing is a fundamental part of the Red Hat Cluster infrastructure and it is important to validate or test that fencing is working properly.



**NOTE**

When a Pacemaker cluster node or Pacemaker remote node is fenced a hard kill should occur and not a graceful shutdown of the operating system. If a graceful shutdown occurs when your system fences a node, disable ACPI soft-off in the **/etc/systemd/logind.conf** file so that your system ignores any power-button-pressed signal. For instructions on disabling ACPI soft-off in the **logind.conf** file, see Disabling ACPI soft-off in the logind.conf file

#### Procedure

Use the following procedure to test a fence device.

1. Use SSH, Telnet, HTTP, or whatever remote protocol is used to connect to the device to manually log in and test the fence device or see what output is given. For example, if you will be configuring fencing for an IPMI-enabled device,then try to log in remotely with **ipmitool**. Take note of the options used when logging in manually because those options might be needed when using the fencing agent.
   If you are unable to log in to the fence device, verify that the device is pingable, there is nothing such as a firewall configuration that is preventing access to the fence device, remote access is enabled on the fencing device, and the credentials are correct.

2. Run the fence agent manually, using the fence agent script. This does not require that the cluster services are running, so you can perform this step before the device is configured in the cluster. This can ensure that the fence device is responding properly before proceeding.

NOTE

These examples use the **fence_ipmilan** fence agent script for an iLO device. The actual fence agent you will use and the command that calls that agent will depend on your server hardware. You should consult the man page for the fence agent you are using to determine which options to specify. You will usually need to know the login and password for the fence device and other information related to the fence device.

The following example shows the format you would use to run the **fence_ipmilan** fence agent script with **-o status** parameter to check the status of the fence device interface on another node without actually fencing it. This allows you to test the device and get it working before attempting to reboot the node. When running this command, you specify the name and password of an iLO user that has power on and off permissions for the iLO device.

```
# fence_ipmilan -a ipaddress -l username -p password -o status
```

The following example shows the format you would use to run the **fence_ipmilan** fence agent script with the **-o reboot** parameter. Running this command on one node reboots the node managed by this iLO device.

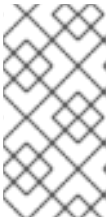
```
# fence_ipmilan -a ipaddress -l username -p password -o reboot
```

If the fence agent failed to properly do a status, off, on, or reboot action, you should check the hardware, the configuration of the fence device, and the syntax of your commands. In addition, you can run the fence agent script with the debug output enabled. The debug output is useful for some fencing agents to see where in the sequence of events the fencing agent script is failing when logging into the fence device.

```
# fence_ipmilan -a ipaddress -l username -p password -o status -D /tmp/$(hostname)-fence_agent.debug
```

When diagnosing a failure that has occurred, you should ensure that the options you specified when manually logging in to the fence device are identical to what you passed on to the fence agent with the fence agent script.

For fence agents that support an encrypted connection, you may see an error due to certificate validation failing, requiring that you trust the host or that you use the fence agent's **ssl-insecure** parameter. Similarly, if SSL/TLS is disabled on the target device, you may need to account for this when setting the SSL parameters for the fence agent.



NOTE

If the fence agent that is being tested is a **fence_drac**, **fence_ilo**, or some other fencing agent for a systems management device that continues to fail, then fall back to trying **fence_ipmilan**. Most systems management cards support IPMI remote login and the only supported fencing agent is **fence_ipmilan**.

3. Once the fence device has been configured in the cluster with the same options that worked

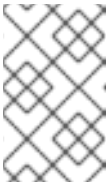
manually and the cluster has been started, test fencing with the **pcs stonith fence** command from any node (or even multiple times from different nodes), as in the following example. The **pcs stonith fence** command reads the cluster configuration from the CIB and calls the fence agent as configured to execute the fence action. This verifies that the cluster configuration is correct.

```
# pcs stonith fence node_name
```

If the **pcs stonith fence** command works properly, that means the fencing configuration for the cluster should work when a fence event occurs. If the command fails, it means that cluster management cannot invoke the fence device through the configuration it has retrieved. Check for the following issues and update your cluster configuration as needed.

- Check your fence configuration. For example, if you have used a host map you should ensure that the system can find the node using the host name you have provided.
- Check whether the password and user name for the device include any special characters that could be misinterpreted by the bash shell. Making sure that you enter passwords and user names surrounded by quotation marks could address this issue.
- Check whether you can connect to the device using the exact IP address or host name you specified in the **pcs stonith** command. For example, if you give the host name in the stonith command but test by using the IP address, that is not a valid test.
- If the protocol that your fence device uses is accessible to you, use that protocol to try to connect to the device. For example many agents use ssh or telnet. You should try to connect to the device with the credentials you provided when configuring the device, to see if you get a valid prompt and can log in to the device.
  If you determine that all your parameters are appropriate but you still have trouble connecting to your fence device, you can check the logging on the fence device itself, if the device provides that, which will show if the user has connected and what command the user issued. You can also search through the **/var/log/messages** file for instances of stonith and error, which could give some idea of what is transpiring, but some agents can provide additional information.

4. Once the fence device tests are working and the cluster is up and running, test an actual failure. To do this, take an action in the cluster that should initiate a token loss.
   - Take down a network. How you take a network depends on your specific configuration. In many cases, you can physically pull the network or power cables out of the host. For information about simulating a network failure, see the Red Hat Knowledgebase solution What is the proper way to simulate a network failure on a RHEL Cluster? .

     **NOTE**

     Disabling the network interface on the local host rather than physically disconnecting the network or power cables is not recommended as a test of fencing because it does not accurately simulate a typical real-world failure.

   - Block corosync traffic both inbound and outbound using the local firewall.
     The following example blocks corosync, assuming the default corosync port is used, **firewalld** is used as the local firewall, and the network interface used by corosync is in the default firewall zone:

```
# firewall-cmd --direct --add-rule ipv4 filter OUTPUT 2 -p udp --dport=5405 -j DROP
# firewall-cmd --add-rich-rule='rule family="ipv4" port port="5405" protocol="udp" drop
```

- Simulate a crash and panic your machine with **sysrq-trigger**. Note, however, that triggering a kernel panic can cause data loss; it is recommended that you disable your cluster resources first.

```
# echo c > /proc/sysrq-trigger
```

### 4.9.6. Configuring fencing levels

Pacemaker supports fencing nodes with multiple devices through a feature called fencing topologies. To implement topologies, create the individual devices as you normally would and then define one or more fencing levels in the fencing topology section in the configuration.

Pacemaker processes fencing levels as follows:

- Each level is attempted in ascending numeric order, starting at 1.
- If a device fails, processing terminates for the current level. No further devices in that level are exercised and the next level is attempted instead.
- If all devices are successfully fenced, then that level has succeeded and no other levels are tried.
- The operation is finished when a level has passed (success), or all levels have been attempted (failed).

Use the following command to add a fencing level to a node. The devices are given as a comma-separated list of **stonith** ids, which are attempted for the node at that level.

```
pcs stonith level add level node devices
```

The following example sets up fence levels so that if the device **my_ilo** fails and is unable to fence the node, then Pacemaker attempts to use the device **my_apc**.

#### Prerequisites

- You have configured an ilo fence device called **my_ilo** for node **rh7-2**.
- You have configured an apc fence device called **my_apc** for node **rh7-2**.

#### Procedure

1. Add a fencing level of 1 for fence device **my_ilo** on node **rh7-2**.

```
# pcs stonith level add 1 rh7-2 my_ilo
```

2. Add a fencing level of 2 for fence device **my_apc** on node **rh7-2**.

```
# pcs stonith level add 2 rh7-2 my_apc
```

3. List the currently configured fencing levels.

```
# pcs stonith level
 Node: rh7-2
  Level 1 - my_ilo
  Level 2 - my_apc
```

For information about testing fence devices, see Testing a fence device.

#### 4.9.6.1. Removing a fence level

The following command removes the fence level for the specified node and devices. If no nodes or devices are specified then the fence level you specify is removed from all nodes.

```
pcs stonith level remove level [node_id] [stonith_id] ... [stonith_id]
```

#### 4.9.6.2. Clearing fence levels

The following command clears the fence levels on the specified node or stonith id. If you do not specify a node or stonith id, all fence levels are cleared.

```
pcs stonith level clear [node]|stonith_id(s)]
```

If you specify more than one stonith id, they must be separated by a comma and no spaces, as in the following example.

```
# pcs stonith level clear dev_a,dev_b
```

#### 4.9.6.3. Verifying nodes and devices in fence levels

The following command verifies that all fence devices and nodes specified in fence levels exist.

```
pcs stonith level verify
```

#### 4.9.6.4. Specifying nodes in fencing topology

You can specify nodes in fencing topology by a regular expression applied on a node name and by a node attribute and its value. For example, the following commands configure nodes **node1**, **node2**, and **node3** to use fence devices **apc1** and **apc2**, and nodes **node4**, **node5**, and **node6** to use fence devices **apc3** and **apc4**.

```
# pcs stonith level add 1 "regexp%node[1-3]" apc1,apc2
# pcs stonith level add 1 "regexp%node[4-6]" apc3,apc4
```

The following commands yield the same results by using node attribute matching.

```
# pcs node attribute node1 rack=1
# pcs node attribute node2 rack=1
# pcs node attribute node3 rack=1
# pcs node attribute node4 rack=2
# pcs node attribute node5 rack=2
# pcs node attribute node6 rack=2
# pcs stonith level add 1 attrib%rack=1 apc1,apc2
# pcs stonith level add 1 attrib%rack=2 apc3,apc4
```

- 

For information about node attributes, see https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html-single/configuring_and_managing_high_availability_clusters/index#node-attributes

### 4.9.7. Configuring fencing for redundant power supplies

When configuring fencing for redundant power supplies, the cluster must ensure that when attempting to reboot a host, both power supplies are turned off before either power supply is turned back on.

If the node never completely loses power, the node may not release its resources. This opens up the possibility of nodes accessing these resources simultaneously and corrupting them.

You need to define each device only once and to specify that both are required to fence the node.

**Procedure**

1. Create the first fence device.

```
# pcs stonith create apc1 fence_apc_snmp ipaddr=apc1.example.com login=user
passwd='7a4D#1j!pz864'
pcmk_host_map="node1.example.com:1;node2.example.com:2"
```

2. Create the second fence device.

```
# pcs stonith create apc2 fence_apc_snmp ipaddr=apc2.example.com login=user
passwd='7a4D#1j!pz864'
pcmk_host_map="node1.example.com:1;node2.example.com:2"
```

3. Specify that both devices are required to fence the node.

```
# pcs stonith level add 1 node1.example.com apc1,apc2
# pcs stonith level add 1 node2.example.com apc1,apc2
```

### 4.9.8. Administering fence devices

The **pcs** command-line interface provides a variety of commands you can use to administer your fence devices after you have configured them.

#### 4.9.8.1. Displaying configured fence devices

The following command shows all currently configured fence devices. If a *stonith_id* is specified, the command shows the options for that configured fencing device only. If the **--full** option is specified, all configured fencing options are displayed.

```
pcs stonith config [stonith_id] [--full]
```

#### 4.9.8.2. Exporting fence devices as pcs commands

You can display the **pcs** commands that can be used to re-create configured fence devices on a different system using the **--output-format=cmd** option of the **pcs stonith config** command.

The following commands create a **fence_apc_snmp** fence device and display the **pcs** command you can use to re-create the device.

```
# pcs stonith create myapc fence_apc_snmp ip="zapc.example.com"
pcmk_host_map="z1.example.com:1;z2.example.com:2" username="apc" password="apc"
# pcs stonith config --output-format=cmd
Warning: Only 'text' output format is supported for stonith levels
pcs stonith create --no-default-ops --force -- myapc fence_apc_snmp \
  ip=zapc.example.com password=apc 'pcmk_host_map=z1.example.com:1;z2.example.com:2'
username=apc \
  op \
    monitor interval=60s id=myapc-monitor-interval-60s
```

#### 4.9.8.3. Exporting fence level configuration

The **pcs stonith config** and the **pcs stonith level config** commands support the **--output-format=** option to export the fencing level configuration in JSON format and as **pcs** commands.

- Specifying **--output-format=cmd** displays the **pcs** commands created from the current cluster configuration that configure fencing levels. You can use these commands to re-create configured fencing levels on a different system.
- Specifying **--output-format=json** displays the fencing level configuration in JSON format, which is suitable for machine parsing.

#### 4.9.8.4. Modifying and deleting fence devices

Modify or add options to a currently configured fencing device with the following command.

```
pcs stonith update stonith_id [stonith_device_options]
```

Updating a SCSI fencing device with the **pcs stonith update** command causes a restart of all resources running on the same node where the fencing resource was running. You can use either version of the following command to update SCSI devices without causing a restart of other cluster resources. SCSI fencing devices can be configured as multipath devices.

```
pcs stonith update-scsi-devices stonith_id set device-path1 device-path2
pcs stonith update-scsi-devices stonith_id add device-path1 remove device-path2
```

Use the following command to remove a fencing device from the current configuration.

```
pcs stonith delete stonith_id
```

#### 4.9.8.5. Manually fencing a cluster node

You can fence a node manually with the following command. If you specify the **--off** option this will use the **off** API call to stonith which will turn the node off instead of rebooting it.

```
pcs stonith fence node [--off]
```

In a situation where no fence device is able to fence a node even if it is no longer active, the cluster may not be able to recover the resources on the node. If this occurs, after manually ensuring that the node is powered down you can enter the following command to confirm to the cluster that the node is powered

down and free its resources for recovery.

**WARNING**

If the node you specify is not actually off, but running the cluster software or services normally controlled by the cluster, data corruption and cluster failure occurs.

```
pcs stonith confirm node
```

#### 4.9.8.6. Disabling a fence device

To disable a fencing device, run the **pcs stonith disable** command.

The following command disables the fence device **myapc**.

```
# pcs stonith disable myapc
```

#### 4.9.8.7. Preventing a node from using a fencing device

To prevent a specific node from using a fencing device, you can configure location constraints for the fencing resource.

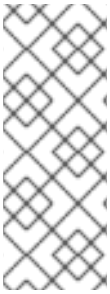
The following example prevents fence device **node1-ipmi** from running on **node1**.

```
# pcs constraint location node1-ipmi avoids node1
```

### 4.9.9. Configuring ACPI for use with integrated fence devices

If your cluster uses integrated fence devices, you must configure ACPI (Advanced Configuration and Power Interface) to ensure immediate and complete fencing.

If a cluster node is configured to be fenced by an integrated fence device, disable ACPI Soft-Off for that node. Disabling ACPI Soft-Off allows an integrated fence device to turn off a node immediately and completely rather than attempting a clean shutdown (for example, **shutdown -h now**). Otherwise, if ACPI Soft-Off is enabled, an integrated fence device can take four or more seconds to turn off a node (see the note that follows). In addition, if ACPI Soft-Off is enabled and a node panics or freezes during shutdown, an integrated fence device may not be able to turn off the node. Under those circumstances, fencing is delayed or unsuccessful. Consequently, when a node is fenced with an integrated fence device and ACPI Soft-Off is enabled, a cluster recovers slowly or requires administrative intervention to recover.



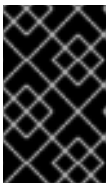
NOTE

The amount of time required to fence a node depends on the integrated fence device used. Some integrated fence devices perform the equivalent of pressing and holding the power button; therefore, the fence device turns off the node in four to five seconds. Other integrated fence devices perform the equivalent of pressing the power button momentarily, relying on the operating system to turn off the node; therefore, the fence device turns off the node in a time span much longer than four to five seconds.

- The preferred way to disable ACPI Soft-Off is to change the BIOS setting to "instant-off" or an equivalent setting that turns off the node without delay, as described in Disabling ACPI Soft-Off with the Bios".

Disabling ACPI Soft-Off with the BIOS may not be possible with some systems. If disabling ACPI Soft-Off with the BIOS is not satisfactory for your cluster, you can disable ACPI Soft-Off with one of the following alternate methods:

- Setting **HandlePowerKey=ignore** in the **/etc/systemd/logind.conf** file and verifying that the node node turns off immediately when fenced, as described in Disabling ACPI soft-off in the logind.conf file. This is the first alternate method of disabling ACPI Soft-Off.
- Appending **acpi=off** to the kernel boot command line, as described in Disabling ACPI completely in the GRUB 2 file. This is the second alternate method of disabling ACPI Soft-Off, if the preferred or the first alternate method is not available.



IMPORTANT

This method completely disables ACPI; some computers do not boot correctly if ACPI is completely disabled. Use this method *only* if the other methods are not effective for your cluster.

#### 4.9.9.1. Disabling ACPI Soft-Off with the BIOS

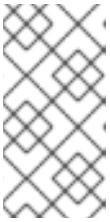
You can disable ACPI Soft-Off by configuring the BIOS of each cluster node with the following procedure.

NOTE

The procedure for disabling ACPI Soft-Off with the BIOS may differ among server systems. You should verify this procedure with your hardware documentation.

**Procedure**

1. Reboot the node and start the **BIOS CMOS Setup Utility** program.
2. Navigate to the Power menu (or equivalent power management menu).
3. At the Power menu, set the **Soft-Off by PWR-BTTN** function (or equivalent) to **Instant-Off** (or the equivalent setting that turns off the node by means of the power button without delay). The **BIOS CMOS Setup Utiliy** example below shows a Power menu with **ACPI Function** set to **Enabled** and **Soft-Off by PWR-BTTN** set to **Instant-Off**.



NOTE

The equivalents to **ACPI Function**, **Soft-Off by PWR-BTTN**, and **Instant-Off** may vary among computers. However, the objective of this procedure is to configure the BIOS so that the computer is turned off by means of the power button without delay.

4. Exit the **BIOS CMOS Setup Utility** program, saving the BIOS configuration.
5. Verify that the node turns off immediately when fenced. For information about testing a fence device, see Testing a fence device.

**BIOS CMOS Setup Utility:**

```
`Soft-Off by PWR-BTTN` set to
`Instant-Off`
```

```
+---------------------------------------------|-------------------+
|   ACPI Function             [Enabled]       |   Item Help       |
|   ACPI Suspend Type         [S1(POS)]       |-------------------|
| x Run VGABIOS if S3 Resume   Auto           |   Menu Level   *  |
|   Suspend Mode              [Disabled]      |                   |
|   HDD Power Down            [Disabled]      |                   |
|   Soft-Off by PWR-BTTN      [Instant-Off    |                   |
|   CPU THRM-Throttling       [50.0%]         |                   |
|   Wake-Up by PCI card       [Enabled]       |                   |
|   Power On by Ring          [Enabled]       |                   |
|   Wake Up On LAN            [Enabled]       |                   |
| x USB KB Wake-Up From S3     Disabled       |                   |
|   Resume by Alarm           [Disabled]      |                   |
| x  Date(of Month) Alarm       0             |                   |
| x  Time(hh:mm:ss) Alarm       0 :  0 :      |                   |
|   POWER ON Function         [BUTTON ONLY    |                   |
| x KB Power ON Password       Enter          |                   |
| x Hot Key Power ON           Ctrl-F1        |                   |
|                                 |                   |
|                                 |                   |
+---------------------------------------------|-------------------+
```

This example shows **ACPI Function** set to **Enabled**, and **Soft-Off by PWR-BTTN** set to **Instant-Off**.

#### 4.9.9.2. Disabling ACPI Soft-Off in the logind.conf file

To disable power-key handing in the **/etc/systemd/logind.conf** file, use the following procedure.

**Procedure**

1. Define the following configuration in the **/etc/systemd/logind.conf** file:

```
HandlePowerKey=ignore
```

2. Restart the **systemd-logind** service:

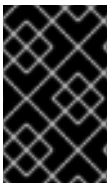
```
# systemctl restart systemd-logind.service
```

-

3. Verify that the node turns off immediately when fenced. For information about testing a fence device, see Testing a fence device.

#### 4.9.9.3. Disabling ACPI completely in the GRUB 2 file

You can disable ACPI Soft-Off by appending **acpi=off** to the GRUB menu entry for a kernel.



IMPORTANT

This method completely disables ACPI; some computers do not boot correctly if ACPI is completely disabled. Use this method *only* if the other methods are not effective for your cluster.

**Procedure**

Use the following procedure to disable ACPI in the GRUB 2 file:

1. Use the **--args** option in combination with the **--update-kernel** option of the **grubby** tool to change the **grub.cfg** file of each cluster node as follows:

```
# grubby --args=acpi=off --update-kernel=ALL
```

2. Reboot the node.

3. Verify that the node turns off immediately when fenced. For information about testing a fence device, see Testing a fence device.

## 4.10. CONFIGURING THE VIRTUAL IP MANAGEMENT RESOURCE AGENT

The **gcp-vpc-move-vip** resource agent attaches a secondary IP address (alias IP) to a running instance. This floating IP address can be assigned between different nodes in the cluster.

```
# pcs resource describe gcp-vpc-move-vip
```

You can configure the resource agent to use a primary subnet address range or a secondary subnet address range.

### 4.10.1. Configuring the primary subnet address range

If you need to automate or manage the assigned IP addresses allocation for VM or other resources within a subnet, you can use the primary subnet address range. It ensures that the primary address range is correctly set and configure to use as stable IP addresses for the primary virtual private network (VPC) subnet.

**Procedure**

1. Create the **aliasip** resource by including an unused internal IP address and the CIDR block:

```
# pcs resource create aliasip gcp-vpc-move-vip alias_ip=UnusedIPaddress/CIDRblock
```

Example:

```
[root@rhel81-node-01 ~]# pcs resource create aliasip gcp-vpc-move-vip
alias_ip=10.10.10.200/32
```

2. Create an **IPaddr2** resource for managing the IP on the node:

```
# pcs resource create vip IPaddr2 nic=interface ip=AliasIPaddress cidr_netmask=32
```

Example:

```
[root@rhel81-node-01 ~]# pcs resource create vip IPaddr2 nic=eth0 ip=10.10.10.200
cidr_netmask=32
```

3. Group the network resources under **vipgrp**:

```
# pcs resource group add vipgrp aliasip vip
```

**Verification**

1. Verify the active resources and under the **vipgrp** group:

```
# pcs status
```

2. Verify the movable resources across the nodes:

```
# pcs resource move vip Node
```

Example:

```
[root@rhel81-node-01 ~]# pcs resource move vip rhel81-node-03
```

3. Verify if the **vip** successfully started on a different node:

```
# pcs status
```

### 4.10.2. Configuring the secondary subnet address range

You can use the secondary subnet address range if you need to assign IP addresses from additional and predefined range within the same subnet, without creating a new subnet. It is useful for specific purposes such as custom routing. With secondary subnet address range, you can manage network traffic in a single subnet with multiple IP address ranges.

**Prerequisites**

- You have created a custom network and a subnet
- Optional: You have installed Google Cloud SDK. For instructions, see Installing the Google Cloud SDK.
  Note, however, that you can use the **gcloud** commands in the following procedure in the terminal that you can activate in the Google Cloud web console.

**Procedure**

1. Create a secondary subnet address range:

```
# gcloud compute networks subnets update SubnetName --region RegionName --add-secondary-ranges SecondarySubnetName=SecondarySubnetRange
```

Example:

```
# gcloud compute networks subnets update range0 --region us-west1 --add-secondary-ranges range1=10.10.20.0/24
```

2. Create the **aliasip** resource with an unused internal IP address in the secondary subnet address range and the CIDR block:

```
# pcs resource create aliasip gcp-vpc-move-vip alias_ip=UnusedIPaddress/CIDRblock
```

Example:

```
[root@rhel81-node-01 ~]# pcs resource create aliasip gcp-vpc-move-vip alias_ip=10.10.20.200/32
```

3. Create an **IPaddr2** resource for managing the IP on the node:

```
# pcs resource create vip IPaddr2 nic=interface ip=AliasIPaddress cidr_netmask=32
```

Example:

```
[root@rhel81-node-01 ~]# pcs resource create vip IPaddr2 nic=eth0 ip=10.10.20.200 cidr_netmask=32
```

4. Group the network resources under **vipgrp**:

```
# pcs resource group add vipgrp aliasip vip
```

### Verification

1. Verify that the active resources and and under the **vipgrp** group:

```
# pcs status
```

2. Verify that the movable resources across different nodes:

```
# pcs resource move vip Node
```

Example:

```
[root@rhel81-node-01 ~]# pcs resource move vip rhel81-node-03
```

3. Verify that the **vip** successfully started on a different node:

```
# pcs status
```

**Additional resources**

- Support Policies for Red Hat High Availability clusters - Transport Protocols
- Exploring Red Hat High Availability's Components, Concepts, and Features - Overview of Transport Protocols
- Design Guidance for Red Hat High Availability Clusters - Selecting the Transport Protocol