Red Hat Enterprise Linux 9

Deploying RHEL 9 on Google Cloud Platform

Obtaining RHEL system images and creating RHEL instances on GCP

Last Updated: 2025-05-21

Red Hat Enterprise Linux 9 Deploying RHEL 9 on Google Cloud Platform

Obtaining RHEL system images and creating RHEL instances on GCP

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 virtual machines (VMs):

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

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

- **Space and cost efficiency**
  You do not need to own any on-premises servers to host your 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 your 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 cloud providers currently certified for running RHEL instances are:
   - Amazon Web Services (AWS)
   - Google Cloud Platform (GCP)
   - Microsoft Azure

   NOTE

   This document specifically talks about deploying RHEL on GCP.

2. Create a RHEL cloud instance on your chosen cloud platform. For more information, 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

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 system image of RHEL 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?
- Configuring and managing cloud-init for RHEL 9

CHAPTER 2. UPLOADING IMAGES TO GCP WITH RHEL IMAGE BUILDER

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 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.

2.2.1. Specifying GCP credentials with the composer-cli command

You can specify GCP authentication credentials in the upload target configuration **gcp-config.toml** file. Use a **Base64**-encoded scheme of the Google account credentials JSON file to save time.

Procedure

1. Get the encoded content of the Google account credentials file with the path stored in **GOOGLE_APPLICATION_CREDENTIALS** environment variable, by running the following command:

```
$ base64 -w 0 "${GOOGLE_APPLICATION_CREDENTIALS}"
```

2. In the upload target configuration **gcp-config.toml** file, set the credentials:

```
provider = "gcp"

[settings]
provider = "gcp"

[settings]
credentials = "GCP_CREDENTIALS"
```

2.2.2. 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 RED HAT ENTERPRISE LINUX IMAGE AS A GOOGLE COMPUTE ENGINE INSTANCE ON GOOGLE CLOUD PLATFORM

To set up a deployment of Red Hat Enterprise Linux 9 (RHEL 9) on Google Cloud Platform (GCP), you can deploy RHEL 9 as a Google Compute Engine (GCE) instance on GCP.

NOTE

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

IMPORTANT

You can create a custom VM from an ISO image, but Red Hat recommends that you use the *Red Hat Image Builder* product to create customized images for use on specific cloud providers. See Composing a Customized RHEL System Image for more information.

Prerequisites

- You need a Red Hat Customer Portal account to complete the procedures in this chapter.
- Create an account with GCP to access the Google Cloud Platform Console. See Google Cloud for more information.

3.1. RED HAT ENTERPRISE LINUX IMAGE OPTIONS ON GCP

You can use multiple types of images for deploying RHEL 9 on Google Cloud Platform. Based on your requirements, consider which option is optimal for your use case.

Table 3.1. Image options

Image option	Subscriptions	Sample scenario	Considerations
Deploy a Red Hat Gold Image.	Use your existing Red Hat subscriptions.	Select a Red Hat Gold Image on Google Cloud Platform. For details on Gold Images and how to access them on Google Cloud Platform, see the Red Hat Cloud Access Reference Guide.	The subscription includes the Red Hat product cost; you pay Google for all other instance costs. Red Hat provides support directly for custom RHEL images.
Deploy a custom image that you move to GCP.	Use your existing Red Hat subscriptions.	Upload your custom image and attach your subscriptions.	The subscription includes the Red Hat product cost; you pay all other instance costs. Red Hat provides support directly for custom RHEL images.

Image option	Subscriptions	Sample scenario	Considerations
Deploy an existing GCP image that includes RHEL.	The GCP images include a Red Hat product.	Choose a RHEL image when you launch an instance on the GCP Compute Engine, or choose an image from the Google Cloud Platform Marketplace.	You pay GCP hourly on a pay-as-you-go model. Such images are called "on-demand" images. GCP offers support for on-demand images through a support agreement.

NOTE

You can create a custom image for GCP by using Red Hat Image Builder. See Composing a Customized RHEL System Image for more information.

IMPORTANT

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

1. Create a new custom RHEL instance and migrate data from your on-demand instance.
2. Cancel your on-demand instance after you migrate your data to avoid double billing.

Additional resources

- Red Hat in the Public Cloud
- Compute Engine images
- Creating an instance from a custom image

3.2. UNDERSTANDING BASE IMAGES

To create a base VM from an ISO image, you can use preconfigured base images and their configuration settings.

3.2.1. Using a custom base image

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

Additional resources

- Red Hat Enterprise Linux

3.2.2. Virtual machine configuration settings

Cloud VMs must have the following configuration settings.

Table 3.2. VM configuration settings

Setting	Recommendation
ssh	ssh must be enabled to provide remote access to your VMs.
dhcp	The primary virtual adapter should be configured for dhcp.

3.3. CREATING A BASE VM FROM AN ISO IMAGE

To create a RHEL 9 base image from an ISO image, enable your host machine for virtualization and create a RHEL virtual machine (VM).

Prerequisites

- Virtualization is enabled on your host machine.
- You have downloaded the latest Red Hat Enterprise Linux ISO image from the Red Hat Customer Portal and moved the image to **/var/lib/libvirt/images**.

3.3.1. Creating a VM from the RHEL ISO image

Procedure

1. Ensure that you have enabled your host machine for virtualization. See Enabling virtualization in RHEL 9 for information and procedures.
2. Create and start a basic Red Hat Enterprise Linux VM. For instructions, see Creating virtual machines.
   a. If you use the command line to create your VM, ensure that you set the default memory and CPUs to the capacity you want for the VM. Set your virtual network interface to **virtio**.
      For example, the following command creates a **kvmtest** VM by using the **/home/username/Downloads/rhel9.iso** image:

      ```
      # virt-install \
          --name kvmtest --memory 2048 --vcpus 2 \
          --cdrom /home/username/Downloads/rhel9.iso,bus=virtio \
          --os-variant=rhel9.0
      ```

   b. If you use the web console to create your VM, follow the procedure in Creating virtual machines by using the web console, with these caveats:
      - Do not check **Immediately Start VM**.
      - Change your **Memory** size to your preferred settings.

- Before you start the installation, ensure that you have changed **Model** under **Virtual Network Interface Settings** to **virtio** and change your **vCPUs** to the capacity settings you want for the VM.

3.3.2. Completing the RHEL installation

To finish the installation of a RHEL system that you want to deploy on Amazon Web Services (AWS), customize the **Installation Summary** view, begin the installation, and enable root access once the VM launches.

Procedure

1. Choose the language you want to use during the installation process.
2. On the **Installation Summary** view:
   a. Click **Software Selection** and check **Minimal Install**.
   b. Click **Done**.
   c. Click **Installation Destination** and check **Custom** under **Storage Configuration**.
      - Verify at least 500 MB for **/boot**. You can use the remaining space for root **/**.
      - Standard partitions are recommended, but you can use Logical Volume Manager (LVM).
      - You can use xfs, ext4, or ext3 for the file system.
      - Click **Done** when you are finished with changes.
3. Click **Begin Installation**.
4. Set a **Root Password**. Create other users as applicable.
5. Reboot the VM and log in as **root** once the installation completes.
6. Configure the image.
   a. Register the VM and enable the Red Hat Enterprise Linux 9 repository.

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

   b. Ensure that the **cloud-init** package is installed and enabled.

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

7. Power down the VM.

Additional resources

- Introduction to cloud-init

3.4. UPLOADING THE RHEL IMAGE TO GCP

To run your RHEL 9 instance on Google Cloud Platform (GCP), you must upload your RHEL 9 image to GCP.

3.4.1. Creating a new project on GCP

To upload your Red Hat Enterprise Linux 9 image to Google Cloud Platform (GCP), you must first create a new project on GCP.

Prerequisites

- You must have an account with GCP. If you do not, see Google Cloud for more information.

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.

NOTE

Once you have installed the Google Cloud SDK, you can use the **gcloud projects create** CLI command to create a project. For example:

```
# gcloud projects create my-gcp-project3 --name project3
```

The example creates a project with the project ID **my-gcp-project3** and the project name **project3**. See gcloud project create for more information.

Additional resources

- Creating and Managing Resources in Google Cloud

3.4.2. Installing the Google Cloud SDK

Many of the procedures to manage HA clusters on Google Cloud Platform (GCP) require the tools in the Google Cloud SDK.

Procedure

1. Follow the GCP instructions for downloading and extracting the Google Cloud SDK archive. See the GCP document Quickstart for Linux for details.
2. Follow the same instructions for initializing the Google Cloud SDK.

NOTE

Once you have initialized the Google Cloud SDK, you can use the **gcloud** CLI commands to perform tasks and obtain information about your project and instances. For example, you can display project information with the **gcloud compute project-info describe --project <project-name>** command.

Additional resources

- Quickstart for Linux
- gcloud command reference
- gcloud command-line tool overview

3.4.3. Creating SSH keys for Google Compute Engine

Generate and register SSH keys with GCE so that you can SSH directly into an instance by using its public IP address.

Procedure

1. Use the **ssh-keygen** command to generate an SSH key pair for use with GCE.

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

2. 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**.
3. Click **SSH Keys** and then click **Edit**.
4. Enter the output generated from the **~/.ssh/google_compute_engine.pub** file and click **Save**. You can now connect to your instance by using standard SSH.

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

NOTE

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

Additional resources

- gcloud compute config-ssh
- Connecting to instances

3.4.4. Creating a storage bucket in GCP Storage

To import your RHEL 9 image to GCP, you must first create a GCP Storage Bucket.

Procedure

1. If you are not already logged in to GCP, log in with the following command.

```
# gcloud auth login
```

2. Create a storage bucket.

```
# gsutil mb gs://bucket_name
```

> **NOTE**
>
> Alternatively, you can use the Google Cloud Console to create a bucket. See Create a bucket for information.

Additional resources

- Create a bucket

3.4.5. Converting and uploading your image to your GCP Bucket

Before a local RHEL 9 image can be deployed in GCP, you must first convert and upload the image to your GCP Bucket. The following steps describe converting an **qcow2** image to **raw** format and then uploading the image as a **tar** archive. However, using different formats is possible as well.

Procedure

1. Run the **qemu-img** command to convert your image. The converted image must have the name **disk.raw**.

```
# qemu-img convert -f qcow2 -O raw rhel-9.0-sample.qcow2 disk.raw
```

2. Tar the image.

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

3. Upload the image to the bucket you created previously. Upload could take a few minutes.

```
# gsutil cp disk.raw.tar.gz gs://bucket_name
```

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

5. Click the name of your bucket.
   The tarred image is listed under your bucket name.

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

Additional resources

- Manually importing virtual disks
- Choosing an import method

3.4.6. Creating an image from the object in the GCP bucket

Before you can create a GCE image from an object that you uploaded to your GCP bucket, you must convert the object into a GCE image.

Procedure

1. Run the following command to create an image for GCE. Specify the name of the image you are creating, the bucket name, and the name of the tarred image.

   ```
   # gcloud compute images create my-image-name --source-uri gs://my-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.

2. 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**.

Additional resources

- Creating, deleting, and deprecating custom images
- gcloud compute images create

3.4.7. Creating a Google Compute Engine instance from an image

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

NOTE

See Creating and starting a VM instance for more information about GCE VM instances and their configuration options.

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**.

   NOTE

   These are the minimum configuration options necessary to create a basic instance. Review additional options based on your application requirements.

10. Find your image under **VM instances**.
11. From the GCP Console Dashboard, 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** CLI command to create a GCE VM instance from an image. A simple example follows.

   ```
   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**. See gcloud compute instances create for more information.

3.4.8. Connecting to your instance

Connect to your GCE instance by using its public IP address.

Procedure

1. Ensure that your instance is running. The following command lists information about your GCE instance, including whether the instance is running, and, if so, the public IP address of the running instance.

   ```
   # gcloud compute instances list
   ```

2. Connect to your instance by using standard SSH. The example uses the **google_compute_engine** key created earlier.

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

NOTE

GCP offers a number of ways to SSH into your instance. See Connecting to instances for more information. You can also connect to your instance using the root account and password you set previously.

Additional resources

- gcloud compute instances list
- Connecting to instances

3.4.9. Attaching Red Hat subscriptions

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

Prerequisites

- You must have enabled your subscriptions.

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

3.5. ADDITIONAL RESOURCES

- Red Hat in the Public Cloud

- Google Cloud

CHAPTER 4. CONFIGURING A RED HAT HIGH AVAILABILITY CLUSTER ON GOOGLE CLOUD PLATFORM

To create a cluster where RHEL nodes automatically redistribute their workloads if a node failure occurs, use the Red Hat High Availability Add-On. Such high availability (HA) clusters can also be hosted on public cloud platforms, including Google Cloud Platform (GCP). Creating RHEL HA clusters on GCP is similar to creating HA clusters in non-cloud environments, with certain specifics.

To configure a Red Hat HA cluster on Google Cloud Platform (GCP) using Google Compute Engine (GCE) virtual machine (VM) instances as cluster nodes, see the following sections.

These provide information on:

- Prerequisite procedures for setting up your environment for GCP. Once you have set up your environment, you can create and configure VM instances.
- Procedures specific to the creation of HA clusters, which transform individual nodes into a cluster of HA nodes on GCP. These include procedures for installing the High Availability packages and agents on each cluster node, configuring fencing, and installing network resource agents.

Prerequisites

- Red Hat Enterprise Linux 9 Server: rhel-9-server-rpms/8Server/x86_64
- Red Hat Enterprise Linux 9 Server (High Availability): rhel-9-server-ha-rpms/8Server/x86_64
    - You must belong to an active GCP project and have sufficient permissions to create resources in the project.
    - 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. THE BENEFITS OF USING HIGH-AVAILABILITY CLUSTERS ON PUBLIC CLOUD PLATFORMS

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 Red Hat Enterprise Linux (RHEL) systems, Red Hat offers a High Availability Add-On. The High Availability Add-On provides all necessary components for creating HA clusters on RHEL systems. The components include high availability service management and cluster administration tools.

Additional resources

- High Availability Add-On overview

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-9-for-x86_64-appstream-rpms	Open source API, daemon, and management tool for managing platform virtualization
virt-install	rhel-9-for-x86_64-appstream-rpms	A command-line utility for building VMs
libguestfs	rhel-9-for-x86_64-appstream-rpms	A library for accessing and modifying VM file systems

Package	Repository	Description
guestfs-tools	rhel-9-for-x86_64-appstream-rpms	System administration tools for VMs; includes the **virt-customize** utility

4.3. RED HAT ENTERPRISE LINUX IMAGE OPTIONS ON GCP

You can use multiple types of images for deploying RHEL 9 on Google Cloud Platform. Based on your requirements, consider which option is optimal for your use case.

Table 4.2. Image options

Image option	Subscriptions	Sample scenario	Considerations
Deploy a Red Hat Gold Image.	Use your existing Red Hat subscriptions.	Select a Red Hat Gold Image on Google Cloud Platform. For details on Gold Images and how to access them on Google Cloud Platform, see the Red Hat Cloud Access Reference Guide.	The subscription includes the Red Hat product cost; you pay Google for all other instance costs. Red Hat provides support directly for custom RHEL images.
Deploy a custom image that you move to GCP.	Use your existing Red Hat subscriptions.	Upload your custom image and attach your subscriptions.	The subscription includes the Red Hat product cost; you pay all other instance costs. Red Hat provides support directly for custom RHEL images.
Deploy an existing GCP image that includes RHEL.	The GCP images include a Red Hat product.	Choose a RHEL image when you launch an instance on the GCP Compute Engine, or choose an image from the Google Cloud Platform Marketplace.	You pay GCP hourly on a pay-as-you-go model. Such images are called "on-demand" images. GCP offers support for on-demand images through a support agreement.

NOTE

You can create a custom image for GCP by using Red Hat Image Builder. See Composing a Customized RHEL System Image for more information.

IMPORTANT

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

1. Create a new custom RHEL instance and migrate data from your on-demand instance.
2. Cancel your on-demand instance after you migrate your data to avoid double billing.

Additional resources

- Red Hat in the Public Cloud
- Compute Engine images
- Creating an instance from a custom image

4.4. INSTALLING THE GOOGLE CLOUD SDK

Many of the procedures to manage HA clusters on Google Cloud Platform (GCP) require the tools in the Google Cloud SDK.

Procedure

1. Follow the GCP instructions for downloading and extracting the Google Cloud SDK archive. See the GCP document Quickstart for Linux for details.
2. Follow the same instructions for initializing the Google Cloud SDK.

NOTE

Once you have initialized the Google Cloud SDK, you can use the **gcloud** CLI commands to perform tasks and obtain information about your project and instances. For example, you can display project information with the **gcloud compute project-info describe --project <project-name>** command.

Additional resources

- Quickstart for Linux
- gcloud command reference
- gcloud command-line tool overview

4.5. CREATING A GCP IMAGE BUCKET

The following document includes the minimum requirements for creating a multi-regional bucket in your default location.

Prerequisites

- GCP storage utility (gsutil)

Procedure

1. If you are not already logged in to Google Cloud Platform, log in with the following command.

   ```
   # gcloud auth login
   ```

2. Create a storage bucket.

   ```
   $ gsutil mb gs://BucketName
   ```

   Example:

   ```
   $ gsutil mb gs://rhel-ha-bucket
   ```

Additional resources

- Make buckets

4.6. 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.

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**.

4.7. PREPARING AND IMPORTING A BASE GCP IMAGE

Before a local RHEL 9 image can be deployed in GCP, you must first convert and upload the image to your GCP Bucket.

Procedure

1. Convert the file. Images uploaded to GCP must be in **raw** format and named **disk.raw**.

   ```
   $ qemu-img convert -f qcow2 ImageName.qcow2 -O raw disk.raw
   ```

2. Compress the **raw** file. Images uploaded to GCP must be compressed.

   ```
   $ tar -Sczf ImageName.tar.gz disk.raw
   ```

3. Import the compressed image to the bucket created earlier.

```
$ gsutil cp ImageName.tar.gz gs://BucketName
```

4.8. 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:

```
[admin@localhost ~] $ 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:

```
[admin@localhost ~] $ 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-9-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
```

4.9. 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

4.10. 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:

```
[admin@localhost ~] $ gcloud compute instance-templates create rhel-91-instance-template
--can-ip-forward --machine-type n1-standard-2 --image rhel-91-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-91-instance-template].
NAME  MACHINE_TYPE   PREEMPTIBLE  CREATION_TIMESTAMP
rhel-91-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:

```
[admin@localhost ~] $ gcloud compute instances create rhel81-node-01 rhel81-node-02
rhel81-node-03 --source-instance-template rhel-91-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
```

4.11. INSTALLING HA PACKAGES AND AGENTS

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-9-server-rpms
# subscription-manager repos --enable=rhel-9-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
```

4.12. CONFIGURING HA SERVICES

On each of your nodes, configure the HA services.

Procedure

1. The user **hacluster** was created during the **pcs** and **pacemaker** installation in the previous step. Create a password for the user **hacluster** on all cluster nodes. Use the same password for all nodes.

```
# passwd hacluster
```

2. If the **firewalld** service is installed, add the HA service.

```
# firewall-cmd --permanent --add-service=high-availability

# firewall-cmd --reload
```

3. Start the **pcs** service and enable it to start on boot.

```
# systemctl start pcsd.service

# systemctl enable pcsd.service

Created symlink from /etc/systemd/system/multi-user.target.wants/pcsd.service to
/usr/lib/systemd/system/pcsd.service.
```

Verification

1. Ensure the **pcsd** service is running.

```
# systemctl status pcsd.service

pcsd.service - PCS GUI and remote configuration interface
Loaded: loaded (/usr/lib/systemd/system/pcsd.service; enabled; vendor preset: disabled)
Active: active (running) since Mon 2018-06-25 19:21:42 UTC; 15s ago
Docs: man:pcsd(8)
man:pcs(8)
Main PID: 5901 (pcsd)
CGroup: /system.slice/pcsd.service
 └─5901 /usr/bin/ruby /usr/lib/pcsd/pcsd > /dev/null &
```

2. Edit the **/etc/hosts** file. Add RHEL host names and internal IP addresses for all nodes.

Additional resources

- How should the /etc/hosts file be set up on RHEL cluster nodes? (Red Hat Knowledgebase)

4.13. CREATING A CLUSTER

To convert multiple nodes into a cluster, use the following steps.

Procedure

1. On one of the nodes, authenticate the **pcs** user. Specify the name of each node in the cluster in the command.

```
# pcs host auth hostname1 hostname2 hostname3
Username: hacluster
Password:
hostname1: Authorized
hostname2: Authorized
hostname3: Authorized
```

2. Create the cluster.

```
# pcs cluster setup cluster-name hostname1 hostname2 hostname3
```

Verification

1. Run the following command to enable nodes to join the cluster automatically when started.

```
# pcs cluster enable --all
```

2. Start the cluster.

```
# pcs cluster start --all
```

4.14. CREATING A FENCING DEVICE

High Availability (HA) environments require a fencing device, which ensures that malfunctioning nodes are isolated and the cluster remains available if an outage occurs.

Note that for most default configurations, the GCP instance names and the RHEL host names are identical.

Procedure

1. Obtain GCP instance names. Note that the output of the following command also shows the internal ID for the instance.

```
# fence_gce --zone us-west1-b --project=rhel-ha-on-gcp -o list
```

Example:

```
[root@rhel81-node-01 ~]# fence_gce --zone us-west1-b --project=rhel-ha-testing-on-gcp
-o list

4435801234567893181,InstanceName-3
4081901234567896811,InstanceName-1
7173601234567893341,InstanceName-2
```

2. Create a fence device.

```
# pcs stonith create FenceDeviceName fence_gce zone=Region-Zone
project=MyProject
```

3. To ensure immediate and complete fencing, disable ACPI Soft-Off on all cluster nodes. For information about disabling ACPI Soft-Off, see Disabling ACPI for use with integrated fence device.

Verification

- Verify that the fence devices started.

```
# pcs status
```

Example:

```
[root@rhel81-node-01 ~]# pcs status

Cluster name: gcp-cluster
Stack: corosync
Current DC: rhel81-node-02 (version 1.1.18-11.el7_5.3-2b07d5c5a9) - partition with quorum
Last updated: Fri Jul 27 12:53:25 2018
Last change: Fri Jul 27 12:51:43 2018 by root via cibadmin on rhel81-node-01
```

```
3 nodes configured
3 resources configured

Online: [ rhel81-node-01 rhel81-node-02 rhel81-node-03 ]

Full list of resources:

us-west1-b-fence    (stonith:fence_gce):    Started rhel81-node-01

Daemon Status:
corosync: active/enabled
pacemaker: active/enabled
pcsd: active/enabled
```

4.15. CONFIGURING THE GCP-VCP-MOVE-VIP RESOURCE AGENT

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

To show more information about this resource:

```
# 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:

Primary subnet address range

Complete the following steps to configure the resource for the primary VPC subnet.

Procedure

1. Create the **aliasip** resource. Include an unused internal IP address. Include the CIDR block in the command.

   ```
   # 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 that the resources have started and are grouped under **vipgrp**.

   ```
   # pcs status
   ```

2. Verify that the resource can move to a different node.

   ```
   # 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
   ```

Secondary subnet address range

Complete the following steps to configure the resource for a secondary subnet address range.

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. Create an unused internal IP address in the secondary subnet address range. Include the CIDR block in the command.

   ```
   # 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 resources have started and are grouped under **vipgrp**.

```
# pcs status
```

2. Verify that the resource can move to a different node.

```
# 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
```

4.16. ADDITIONAL RESOURCES

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

CHAPTER 5. CONFIGURING RHEL ON GCP WITH SECURE BOOT

The Secure Boot mechanism in the Unified Extensible Firmware Interface (UEFI) specification controls execution of programs at the boot time. Secure Boot ensures the execution of only trusted and authorized programs, while preventing unauthorized programs by verifying digital signatures of the boot loader and other components at the boot time.

5.1. INTRODUCTION TO SECURE BOOT

The Secure Boot mechanism is a security protocol that provides authentication access for specific device paths by using defined interfaces. Successive authentication configuration overwrites the former configuration, by making it non-retrievable. It ensures that a trusted vendor has signed the boot loader and the kernel. Red Hat Enterprise Linux firmware checks the digital signature of the boot loader and related components against trusted keys stored on the hardware. If any component is tampered with or signed by an untrusted entity, the booting process aborts, which prevents potentially malicious software from taking control of the system. Additionally, the RHEL kernel offers the **lockdown** mode that ensures only a trusted vendor signed kernel modules are loaded.

5.2. COMPONENTS OF SECURE BOOT

Secure Boot consists of firmware, signature databases, cryptographic keys, boot loader, and hardware modules. The components of the UEFI trust sequence are listed below:

- Key Exchange Key database (KEK): Exchange of public keys to establish trust between the RHEL instance and the platform firmware such as Hardware Virtual Machine (HVM). You can also update Allowed Signature database (**db**) and Forbidden Signature database (**dbx**) by using these keys.

- Platform Key database (PK): A self-signed single key database to establish trust between the RHEL instance and the cloud service provider. It also updates KEK database.

- Allowed Signature database (**db**): A database that maintains a list of certificates or binary hashes to check whether the binary file is allowed to boot on the system or not. Additionally, you can import all certificates from **db** in the kernel **.platform** keyring. This feature allows you to add and load signed third-party kernel modules in the **lockdown** mode.

- Forbidden Signature database (**dbx**): A database that maintains a list of forbidden certificates or binary hashes to boot on the system.

NOTE

Binary files check against the **dbx** database as well as the Secure Boot Advanced Targeting (SBAT) mechanism. SBAT allows you to revoke older versions of specific binaries by keeping the certificate that has signed the binaries as valid.

Additional resources

- SBAT mechanism

5.3. STAGES OF BOOTING A RHEL INSTANCE ON A CLOUD PLATFORM

When a RHEL instance boots in the Unified Kernel Image (UKI) mode and with Secure Boot enabled, the RHEL instance interacts with cloud service infrastructure in the following sequence:

1. *Integrity verification* : Initially, when a cloud hosted firmware boots, it checks and verifies the RHEL instance integrity with Secure Boot. When the RHEL Kernel boots in the Secure Boot mode, it enters in a **lockdown** mode and also extends the kernel **.platform** keyring with an ephemeral key and other keys to sign third party modules.

2. *Variable store initialization* : Next, this firmware initializes UEFI variables from a variable store, which is a dedicated storage area for information that are necessary for the boot process and runtime operations. If the RHEL instance boots for the first time, the firmware initializes the variable store from default values of the VM image.

3. *Bootloader* : After that the firmware loads the **shim** boot loader for the RHEL instance in a x86 UEFI environment.

    a. The **shim** binary extends the list of trusted certificates with Red Hat Secure Boot CA, and optionally, with Machine Owner Key (**MOK**) that is neededfor bare metal platforms to update Secure Boot variables compatible with OEM vendors.

4. *UKI* : The **shim** binary loads the RHEL Unified Kernel Image (UKI), which is the **kernel-uki-virt** package. To use the UKI **cmdline** extensions, the RHEL kernel checks their signatures against Allowed Signature database (**db**) and **MOK** to ensure that they are signed by both Red Hat Enterprise Linux and the end user.

Additional resources

- Red Hat Enterprise Linux and Secure Boot in the cloud

5.4. CONFIGURING A RHEL INSTANCE WITH THE SECURE BOOT MECHANISM FROM A PUBLICLY AVAILABLE RHEL IMAGE ON GOOGLE CLOUD PLATFORM

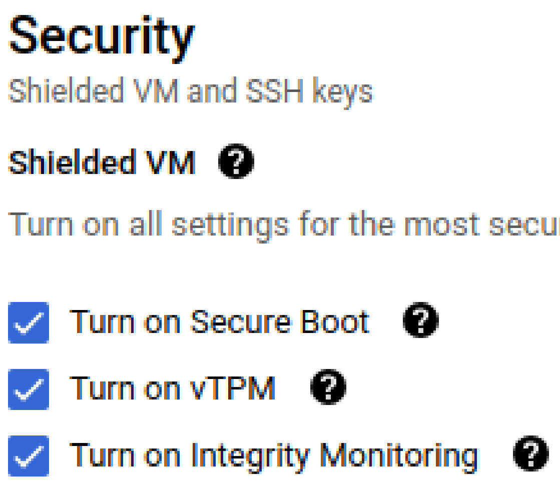
Publicly available Red Hat Enterprise Linux images on Google Cloud Platform can be booted in Secure Boot enabled state. By default, it contains the Allowed Signature database (**db**) with Microsoft certificates.

Prerequisites

- You have installed the **keyutils** package.

Procedure

- Launch a publicly available Red Hat Enterprise Linux instance from Google Cloud console with **Turn on Secure Boot** option enabled.


Security
Shielded VM and SSH keys
Shielded VM
Turn on all settings for the most secure configuration.
✓ Turn on Secure Boot
✓ Turn on vTPM
✓ Turn on Integrity Monitoring

Verification

1. Verify if **Secure Boot** is enabled:

```
$ mokutil --sb-state
SecureBoot enabled
```

2. Use the **keyctl** utility to verify the kernel keyring for the custom certificate:

```
$ sudo keyctl list %:.platform
4 keys in keyring:
12702216: ---lswrv     0     0 asymmetric: Microsoft Corporation UEFI CA 2011:
13adbf4309bd82709c8cd54f316ed522988a1bd4
50338534: ---lswrv     0     0 asymmetric: Red Hat Secure Boot CA 5:
cc6fa5e72868ba494e939bbd680b9144769a9f8f
681047026: ---lswrv     0     0 asymmetric: Microsoft Windows Production PCA 2011:
a92902398e16c49778cd90f99e4f9ae17c55af53
```

5.5. CONFIGURING A RHEL INSTANCE WITH THE SECURE BOOT MECHANISM FROM A CUSTOM RHEL IMAGE ON GOOGLE CLOUD PLATFORM

The following procedure configures a RHEL instance by including a custom certificate into the SecureBoot DB, which allows you to sign custom artifacts, such as third party kernel modules and UKI extensions.

Prerequisites

- You have installed the **python3**, **efivar**, **keyutils**, **openssl**, and **python3-virt-firmware** packages.
- You have installed the **google-cloud-cli** utility. See installing gcloud CLI on RHEL.

Procedure

1. Create a new random Universally Unique Identifier (UUID) and store it in a system generated random text file:

   ```
   $ uuidgen --random > GUID.txt
   ```

2. Generate a new RSA private key **PK.key** and a self-signed X.509 certificate **PK.cer** for the Platform Key database:

   ```
   $ openssl req -quiet -newkey rsa:4096 -nodes -keyout PK.key -new -x509 -sha256 -days
   3650 -subj "/CN=Platform key/" -outform DER -out PK.cer
   ```

   The **openssl** utility generates a common name **Platform key** for the certificate by setting output format to Distinguished Encoding Rules (DNR). DNR is a standardized binary format for data encoding.

3. Generate a new RSA private key **KEK.key** and a self-signed X.509 certificate **KEK.cer** for the Key Exchange Key database:

   ```
   $ openssl req -quiet -newkey rsa:4096 -nodes -keyout KEK.key -new -x509 -sha256 -days
   3650 -subj "/CN=Key Exchange Key/" -outform DER -out KEK.cer
   ```

4. Generate a custom certificate **custom_db.cer**:

   ```
   $ openssl req -quiet -newkey rsa:4096 -nodes -keyout custom_db.key -new -x509 -sha256 -
   days 3650 -subj "/CN=Signature Database key/" --outform DER -out custom_db.cer
   ```

5. Download the Microsoft certificate:

   ```
   $ wget https://go.microsoft.com/fwlink/p/?linkid=321194 --user-agent="Mozilla/5.0 (Windows
   NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.103
   Safari/537.36" -O MicCorUEFCA2011_2011-06-27.crt
   ```

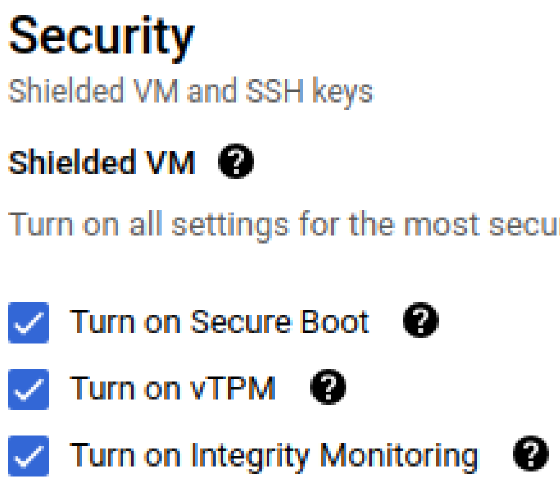
6. Download the updated forbidden signatures (dbx) UEFI Revocation List File for 64 bit system:

   ```
   $ wget https://uefi.org/sites/default/files/resources/x64_DBXUpdate.bin
   ```

7. Use the **google-cloud-cli** utility to create and register the image from a disk snapshot with the desired Secure Boot variables:

   ```
   $ gcloud compute images create <example-rhel-9-efi-image> --source-image
   projects/<example_project_id>/global/images/<example_image_name> --platform-key-
   file=PK.cer --key-exchange-key-file=KEK.cer --signature-database-
   file=custom_db.cer,MicCorUEFCA2011_2011-06-27.crt --forbidden-database-file
   x64_DBXUpdate.bin --guest-os-features="UEFI_COMPATIBLE"
   ```

8. Launch the instance of an **example-rhel-9-efi-image** image with the **Turn on Security Boot** feature from Google Cloud console.


Security

Shielded VM and SSH keys

Shielded VM

Turn on all settings for the most secure configuration.

- [x] Turn on Secure Boot
- [x] Turn on vTPM
- [x] Turn on Integrity Monitoring

Verification

1. Check if the newly created RHEL instance has Secure Boot enabled:

```
$ mokutil --sb-state
SecureBoot enabled
```

2. Use the **keyctl** utility to verify the kernel keyring for the custom certificate:

```
$ sudo keyctl list %:.platform
...
757453569: ---lswrv     0     0 asymmetric: Signature Database key:
f064979641c24e1b935e402bdbc3d5c4672a1acc
...
```