Red Hat Enterprise Linux 8

Deploying Red Hat Enterprise Linux 8 on public cloud platforms

Creating Red Hat Enterprise Linux system images and configuring a Red Hat High Availability cluster for public cloud platforms

Last Updated: 2022-12-14

Red Hat Enterprise Linux 8 Deploying Red Hat Enterprise Linux 8 on public cloud platforms

Creating Red Hat Enterprise Linux system images and configuring a Red Hat High Availability cluster for public cloud platforms

Legal Notice

Abstract

You can create and deploy custom Red Hat Enterprise Linux images to various cloud platforms, including Microsoft Azure, Amazon Web Services (AWS), and Google Cloud Platform (GCP). You can also create and configure a Red Hat High Availability cluster on each cloud platform. This document provides instructions for creating system images, and also for setting up high-availability (HA) clusters. including installing required packages and agents, configuring fencing, and installing network resource agents.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message .

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

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

Submitting comments on specific passages

1. View the documentation in the **Multi-page HTML** format and ensure that you see the **Feedback** button in the upper right corner after the page fully loads.
2. Use your cursor to highlight the part of the text that you want to comment on.
3. Click the **Add Feedback** button that appears near the highlighted text.
4. Add your feedback and click **Submit**.

Submitting feedback through Bugzilla (account required)

1. Log in to the Bugzilla website.
2. Select the correct version from the **Version** menu.
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 **Submit Bug**.

CHAPTER 1. DEPLOYING A RED HAT ENTERPRISE LINUX IMAGE AS A VIRTUAL MACHINE ON MICROSOFT AZURE

To deploy a Red Hat Enterprise Linux 8 (RHEL 8) image on Microsoft Azure, follow the information below. This chapter:

- Discusses your options for choosing an image
- Lists or refers to system requirements for your host system and virtual machine (VM)
- Provides procedures for creating a custom VM from an ISO image, uploading it to Azure, and launching an Azure VM instance

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. With Image Builder, you can create and upload an Azure Disk Image (VHD format). See Composing a Customized RHEL System Image for more information.

For a list of Red Hat products that you can use securely on Azure, refer to Red Hat on Microsoft Azure.

Prerequisites

- Sign up for a Red Hat Customer Portal account.
- Sign up for a Microsoft Azure account.
- Enable your subscriptions in the Red Hat Cloud Access program. The Red Hat Cloud Access program allows you to move your Red Hat subscriptions from physical or on-premise systems to Azure with full support from Red Hat.

1.1. ADDITIONAL RESOURCES

- Red Hat in the Public Cloud
- Red Hat Cloud Access Reference Guide
- Frequently Asked Questions and Recommended Practices for Microsoft Azure

1.2. RED HAT ENTERPRISE LINUX IMAGE OPTIONS ON AZURE

The following table lists image choices for RHEL 8 on Microsoft Azure, and notes the differences in the image options.

Table 1.1. Image options

Image option	Subscriptions	Sample scenario	Considerations

Image option	Subscriptions	Sample scenario	Considerations
Choose to deploy a Red Hat Gold Image.	Use your existing Red Hat subscriptions.	Enable subscriptions through the Red Hat Cloud Access program, and then choose a Red Hat Gold Image on Azure. See the Red Hat Cloud Access Reference Guide for details on Gold Images and how to access them on Azure.	The subscription includes the Red Hat product cost; you pay Microsoft for all other instance costs. Red Hat Gold Images are called "Cloud Access" images because you use your existing Red Hat subscriptions. Red Hat provides support directly for Cloud Access images.
Choose to deploy a custom image that you move to Azure.	Use your existing Red Hat subscriptions.	Enable subscriptions through the Red Hat Cloud Access program, upload your custom image, and attach your subscriptions.	The subscription includes the Red Hat product cost; you pay Microsoft for all other instance costs. Custom images that you move to Azure are "Cloud Access" images because you use your existing Red Hat subscriptions. Red Hat provides support directly for Cloud Access images.
Choose to deploy an existing Azure image that includes RHEL.	The Azure images include a Red Hat product.	Choose a RHEL image when you create a VM using the Azure console, or choose a VM from the Azure Marketplace.	You pay Microsoft hourly on a pay-as-you-go model. Such images are called "on-demand." Azure provides support for on-demand images through a support agreement. Red Hat provides updates to the images. Azure makes the updates available through the Red Hat Update Infrastructure (RHUI).

NOTE

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

The remainder of this chapter includes information and procedures pertaining to Red Hat Enterprise Linux custom images.

Additional resources

- Using Red Hat Gold Images on Microsoft Azure
- Red Hat Cloud Access program
- Azure Marketplace
- Billing options in the Azure Marketplace
- Red Hat Enterprise Linux Bring-Your-Own-Subscription Gold Images in Azure

1.3. UNDERSTANDING BASE IMAGES

This section includes information on using preconfigured base images and their configuration settings.

1.3.1. 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.

To prepare a cloud image of RHEL, follow the instructions in the sections below. To prepare a Hyper-V cloud image of RHEL, see the Prepare a Red Hat-based virtual machine from Hyper-V Manager .

1.3.2. Required system packages

The procedures in this chapter assume you are using a host system running Red Hat Enterprise Linux. To successfully complete the procedures, your host system must have the following packages installed.

Table 1.2. System packages

Package	Repository	Description
libvirt	rhel-8-for-x86_64-appstream-rpms	Open source API, daemon, and management tool for managing platform virtualization
virt-install	rhel-8-for-x86_64-appstream-rpms	A command-line utility for building VMs
libguestfs	rhel-8-for-x86_64-appstream-rpms	A library for accessing and modifying VM file systems
libguestfs-tools	rhel-8-for-x86_64-appstream-rpms	System administration tools for VMs; includes the **guestfish** utility

1.3.3. Azure VM configuration settings

Azure VMs must have the following configuration settings. Some of these settings are enabled during the initial VM creation. Other settings are set when provisioning the VM image for Azure. Keep these settings in mind as you move through the procedures. Refer to them as necessary.

Table 1.3. VM configuration settings

Setting	Recommendation
ssh	ssh must be enabled to provide remote access to your Azure VMs.
dhcp	The primary virtual adapter should be configured for dhcp (IPv4 only).
Swap Space	Do not create a dedicated swap file or swap partition. You can configure swap space with the Windows Azure Linux Agent (WALinuxAgent).
NIC	Choose **virtio** for the primary virtual network adapter.
encryption	For custom images, use Network Bound Disk Encryption (NBDE) for full disk encryption on Azure.

1.3.4. Creating a base image from an ISO image

The following procedure lists the steps and initial configuration requirements for creating a custom ISO image. Once you have configured the image, you can use the image as a template for creating additional VM instances.

Prerequisites

- Ensure that you have enabled your host machine for virtualization. See Enabling virtualization in RHEL 8 for information and procedures.

Procedure

1. Download the latest Red Hat Enterprise Linux 8 DVD ISO image from the Red Hat Customer Portal.
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**.
    A basic command-line sample follows.

    ```
    # virt-install --name kvmtest --memory 2048 --vcpus 2 --disk rhel-8.0-x86_64-kvm.qcow2,bus=virtio --import --os-variant=rhel8.0
    ```

    b. If you use the web console to create your VM, follow the procedure in Creating virtual machines 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. Review the following additional installation selection and modifications.
   - Select **Minimal Install** with the **standard RHEL** option.
   - For **Installation Destination**, select **Custom Storage Configuration**. Use the following configuration information to make your selections.
     - Verify at least 500 MB for **/boot**.
     - For file system, use xfs, ext4, or ext3 for both **boot** and **root** partitions.
     - Remove swap space. Swap space is configured on the physical blade server in Azure by the WALinuxAgent.
   - On the **Installation Summary** screen, select **Network and Host Name**. Switch **Ethernet** to **On**.
4. When the install starts:
   - Create a **root** password.
   - Create an administrative user account.
5. When installation is complete, reboot the VM and log in to the root account.
6. Once you are logged in as **root**, you can configure the image.

1.4. CONFIGURING A CUSTOM BASE IMAGE FOR MICROSOFT AZURE

To deploy a RHEL 8 virtual machine (VM) with specific settings in Azure, you can create a custom base image for the VM. The following sections describe additional configuration changes that Azure requires.

1.4.1. Installing Hyper-V device drivers

Microsoft provides network and storage device drivers as part of their Linux Integration Services (LIS) for Hyper-V package. You may need to install Hyper-V device drivers on the VM image prior to provisioning it as an Azure virtual machine (VM). Use the **lsinitrd | grep hv** command to verify that the drivers are installed.

Procedure

1. Enter the following **grep** command to determine if the required Hyper-V device drivers are installed.

   ```
   # lsinitrd | grep hv
   ```

   In the example below, all required drivers are installed.

   ```
   # lsinitrd | grep hv
   drwxr-xr-x   2 root     root           0 Aug 12 14:21 usr/lib/modules/3.10.0-
   ```

```
932.el8.x86_64/kernel/drivers/hv
-rw-r--r--   1 root     root        31272 Aug 11 08:45 usr/lib/modules/3.10.0-
932.el8.x86_64/kernel/drivers/hv/hv_vmbus.ko.xz
-rw-r--r--   1 root     root        25132 Aug 11 08:46 usr/lib/modules/3.10.0-
932.el8.x86_64/kernel/drivers/net/hyperv/hv_netvsc.ko.xz
-rw-r--r--   1 root     root         9796 Aug 11 08:45 usr/lib/modules/3.10.0-
932.el8.x86_64/kernel/drivers/scsi/hv_storvsc.ko.xz
```

If all the drivers are not installed, complete the remaining steps.

NOTE

An **hv_vmbus** driver may exist in the environment. Even if this driver is present, complete the following steps.

2. Create a file named **hv.conf** in **/etc/dracut.conf.d**.

3. Add the following driver parameters to the **hv.conf** file.

```
add_drivers+=" hv_vmbus "
add_drivers+=" hv_netvsc "
add_drivers+=" hv_storvsc "
add_drivers+=" nvme "
```

NOTE

Note the spaces before and after the quotes, for example, **add_drivers+=" hv_vmbus "**. This ensures that unique drivers are loaded in the event that other Hyper-V drivers already exist in the environment.

4. Regenerate the **initramfs** image.

```
# dracut -f -v --regenerate-all
```

Verification

1. Reboot the machine.

2. Run the **lsinitrd | grep hv** command to verify that the drivers are installed.

1.4.2. Making configuration changes required for a Microsoft Azure deployment

Before you deploy your custom base image to Azure, you must perform additional configuration changes to ensure that the virtual machine (VM) can properly operate in Azure.

Procedure

1. Log in to the VM.

2. Register the VM, and enable the Red Hat Enterprise Linux 8 repository.

```
# subscription-manager register --auto-attach
Installed Product Current Status:
```

```
Product Name: Red Hat Enterprise Linux for x86_64
Status: Subscribed
```

3. Ensure that the **cloud-init** and **hyperv-daemons** packages are installed.

   ```
   # yum install cloud-init hyperv-daemons -y
   ```

4. Create **cloud-init** configuration files that are needed for integration with Azure services:

   a. To enable logging to the Hyper-V Data Exchange Service (KVP), create the **/etc/cloud/cloud.cfg.d/10-azure-kvp.cfg** configuration file and add the following lines to that file.

   ```
   reporting:
       logging:
           type: log
       telemetry:
           type: hyperv
   ```

   b. To add Azure as a datasource, create the **/etc/cloud/cloud.cfg.d/91-azure_datasource.cfg** configuration file, and add the following lines to that file.

   ```
   datasource_list: [ Azure ]
   datasource:
       Azure:
           apply_network_config: False
   ```

5. To ensure that specific kernel modules are blocked from loading automatically, edit or create the **/etc/modprobe.d/blocklist.conf** file and add the following lines to that file.

   ```
   blacklist nouveau
   blacklist lbm-nouveau
   blacklist floppy
   blacklist amdgpu
   blacklist skx_edac
   blacklist intel_cstate
   ```

6. Modify **udev** network device rules:

   a. Remove the following persistent network device rules if present.

   ```
   # rm -f /etc/udev/rules.d/70-persistent-net.rules
   # rm -f /etc/udev/rules.d/75-persistent-net-generator.rules
   # rm -f /etc/udev/rules.d/80-net-name-slot-rules
   ```

   b. To ensure that Accelerated Networking on Azure works as intended, create a new network device rule **/etc/udev/rules.d/68-azure-sriov-nm-unmanaged.rules** and add the following line to it.

   ```
   SUBSYSTEM=="net", DRIVERS=="hv_pci", ACTION=="add",
   ENV{NM_UNMANAGED}="1"
   ```

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

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

8. Modify kernel boot parameters:

   a. Open the **/etc/default/grub** file, and ensure the **GRUB_TIMEOUT** line has the following value.

   ```
   GRUB_TIMEOUT=10
   ```

   b. Remove the following options from the end of the **GRUB_CMDLINE_LINUX** line if present.

   ```
   rhgb quiet
   ```

   c. Ensure the **/etc/default/grub** file contains the following lines with all the specified options.

   ```
   GRUB_CMDLINE_LINUX="loglevel=3 crashkernel=auto console=tty1 console=ttyS0
   earlyprintk=ttyS0 rootdelay=300"
   GRUB_TIMEOUT_STYLE=countdown
   GRUB_TERMINAL="serial console"
   GRUB_SERIAL_COMMAND="serial --speed=115200 --unit=0 --word=8 --parity=no --
   stop=1"
   ```

   d. Regenerate the **grub.cfg** file.
   On a BIOS-based machine:

   ```
   # grub2-mkconfig -o /boot/grub2/grub.cfg
   ```

   On a UEFI-based machine:

   ```
   # grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
   ```

   If your system uses a non-default location for **grub.cfg**, adjust the command accordingly.

9. Configure the Windows Azure Linux Agent (**WALinuxAgent**):

   a. Install and enable the **WALinuxAgent** package.

   ```
   # yum install WALinuxAgent -y
   # systemctl enable waagent
   ```

   b. To ensure that a swap partition is not used in provisioned VMs, edit the following lines in the **/etc/waagent.conf** file.

   ```
   Provisioning.DeleteRootPassword=y
   ResourceDisk.Format=n
   ResourceDisk.EnableSwap=n
   ```

10. Prepare the VM for Azure provisioning:

   a. Unregister the VM from Red Hat Subscription Manager.

   ```
   # subscription-manager unregister
   ```

b. Clean up the existing provisioning details.

```
# waagent -force -deprovision
```

NOTE

This command generates warnings, which are expected because Azure handles the provisioning of VMs automatically.

c. Clean the shell history and shut down the VM.

```
# export HISTSIZE=0
# poweroff
```

1.5. CONVERTING THE IMAGE TO A FIXED VHD FORMAT

All Microsoft Azure VM images must be in a fixed **VHD** format. The image must be aligned on a 1 MB boundary before it is converted to VHD. This section describes how to convert the image from **qcow2** to a fixed **VHD** format and align the image, if necessary. Once you have converted the image, you can upload it to Azure.

Procedure

1. Convert the image from **qcow2** to **raw** format.

```
$ qemu-img convert -f qcow2 -O raw <image-name>.qcow2 <image-name>.raw
```

2. Create a shell script using the contents below.

```
#!/bin/bash
MB=$((1024 * 1024))
size=$(qemu-img info -f raw --output json "$1" | gawk 'match($0, /"virtual-size": ([0-9]+),/, val)
{print val[1]}')
rounded_size=$((($size/$MB + 1) * $MB))
if [ $(($size % $MB)) -eq  0 ]
then
 echo "Your image is already aligned. You do not need to resize."
 exit 1
fi
echo "rounded size = $rounded_size"
export rounded_size
```

3. Run the script. This example uses the name **align.sh**.

```
$ sh align.sh <image-xxx>.raw
```

- If the message *"Your image is already aligned. You do not need to resize."* displays, proceed to the following step.
- If a value displays, your image is not aligned.

4. Use the following command to convert the file to a fixed **VHD** format.

The sample uses qemu-img version 2.12.0.

```
$ qemu-img convert -f raw -o subformat=fixed,force_size -O vpc <image-xxx>.raw
<image.xxx>.vhd
```

Once converted, the **VHD** file is ready to upload to Azure.

5. If the **raw** image is not aligned, complete the following steps to align it.
   a. Resize the **raw** file using the rounded value displayed when you ran the verification script.

      ```
      $ qemu-img resize -f raw <image-xxx>.raw <rounded-value>
      ```

   b. Convert the **raw** image file to a **VHD** format.
      The sample uses qemu-img version 2.12.0.

      ```
      $ qemu-img convert -f raw -o subformat=fixed,force_size -O vpc <image-xxx>.raw
      <image.xxx>.vhd
      ```

      Once converted, the **VHD** file is ready to upload to Azure.

1.6. INSTALLING THE AZURE CLI

Complete the following steps to install the Azure command line interface (Azure CLI 2.1). Azure CLI 2.1 is a Python-based utility that creates and manages VMs in Azure.

Prerequisites

- You need to have an account with Microsoft Azure before you can use the Azure CLI.
- The Azure CLI installation requires Python 3.x.

Procedure

1. Import the Microsoft repository key.

   ```
   $ sudo rpm --import https://packages.microsoft.com/keys/microsoft.asc
   ```

2. Create a local Azure CLI repository entry.

   ```
   $ sudo sh -c 'echo -e "[azure-cli]\nname=Azure
   CLI\nbaseurl=https://packages.microsoft.com/yumrepos/azure-
   cli\nenabled=1\ngpgcheck=1\ngpgkey=https://packages.microsoft.com/keys/microsoft.asc" >
   /etc/yum.repos.d/azure-cli.repo'
   ```

3. Update the **yum** package index.

   ```
   $ yum check-update
   ```

4. Check your Python version (**python --version**) and install Python 3.x, if necessary.

   ```
   $ sudo yum install python3
   ```

5. Install the Azure CLI.

```
$ sudo yum install -y azure-cli
```

6. Run the Azure CLI.

```
$ az
```

Additional resources

- Azure CLI
- Azure CLI command reference

1.7. CREATING RESOURCES IN AZURE

Complete the following procedure to create the Azure resources that you need before you can upload the **VHD** file and create the Azure image.

Procedure

1. Enter the following command to authenticate your system with Azure and log in.

```
$ az login
```

NOTE

If a browser is available in your environment, the CLI opens your browser to the Azure sign-in page. See Sign in with Azure CLI for more information and options.

2. Create a resource group in an Azure region.

```
$ az group create --name <resource-group> --location <azure-region>
```

Example:

```
[clouduser@localhost]$ az group create --name azrhelclirsgrp --location southcentralus
{
  "id": "/subscriptions//resourceGroups/azrhelclirsgrp",
  "location": "southcentralus",
  "managedBy": null,
  "name": "azrhelclirsgrp",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null
}
```

3. Create a storage account. See SKU Types for more information about valid SKU values.

```
$ az storage account create -l <azure-region> -n <storage-account-name> -g
<resource-group> --sku <sku_type>
```

Example:

```
[clouduser@localhost]$ az storage account create -l southcentralus -n azrhelclistact -g
azrhelclirsgrp --sku Standard_LRS
{
  "accessTier": null,
  "creationTime": "2017-04-05T19:10:29.855470+00:00",
  "customDomain": null,
  "encryption": null,
  "id":
"/subscriptions//resourceGroups/azrhelclirsgrp/providers/Microsoft.Storage/storageAccounts/azr
helclistact",
  "kind": "StorageV2",
  "lastGeoFailoverTime": null,
  "location": "southcentralus",
  "name": "azrhelclistact",
  "primaryEndpoints": {
    "blob": "https://azrhelclistact.blob.core.windows.net/",
    "file": "https://azrhelclistact.file.core.windows.net/",
    "queue": "https://azrhelclistact.queue.core.windows.net/",
    "table": "https://azrhelclistact.table.core.windows.net/"
},
"primaryLocation": "southcentralus",
"provisioningState": "Succeeded",
"resourceGroup": "azrhelclirsgrp",
"secondaryEndpoints": null,
"secondaryLocation": null,
"sku": {
  "name": "Standard_LRS",
  "tier": "Standard"
},
"statusOfPrimary": "available",
"statusOfSecondary": null,
"tags": {},
  "type": "Microsoft.Storage/storageAccounts"
}
```

4. Get the storage account connection string.

```
$ az storage account show-connection-string -n <storage-account-name> -g
<resource-group>
```

Example:

```
[clouduser@localhost]$ az storage account show-connection-string -n azrhelclistact -g
azrhelclirsgrp
{
  "connectionString":
"DefaultEndpointsProtocol=https;EndpointSuffix=core.windows.net;AccountName=azrhelclistact
AccountKey=NreGk...=="
}
```

5. Export the connection string by copying the connection string and pasting it into the following command. This string connects your system to the storage account.

```
$ export AZURE_STORAGE_CONNECTION_STRING="<storage-connection-string>"
```

Example:

```
[clouduser@localhost]$ export
AZURE_STORAGE_CONNECTION_STRING="DefaultEndpointsProtocol=https;Endpoi
ntSuffix=core.windows.net;AccountName=azrhelclistact;AccountKey=NreGk...=="
```

6. Create the storage container.

```
$ az storage container create -n <container-name>
```

Example:

```
[clouduser@localhost]$ az storage container create -n azrhelclistcont
{
  "created": true
}
```

7. Create a virtual network.

```
$ az network vnet create -g <resource group> --name <vnet-name> --subnet-name
<subnet-name>
```

Example:

```
[clouduser@localhost]$ az network vnet create --resource-group azrhelclirsgrp --name
azrhelclivnet1 --subnet-name azrhelclisubnet1
{
  "newVNet": {
    "addressSpace": {
      "addressPrefixes": [
      "10.0.0.0/16"
      ]
  },
  "dhcpOptions": {
    "dnsServers": []
  },
  "etag": "W/\"\"",
  "id":
"/subscriptions//resourceGroups/azrhelclirsgrp/providers/Microsoft.Network/virtualNetworks/azr
helclivnet1",
  "location": "southcentralus",
  "name": "azrhelclivnet1",
  "provisioningState": "Succeeded",
  "resourceGroup": "azrhelclirsgrp",
  "resourceGuid": "0f25efee-e2a6-4abe-a4e9-817061ee1e79",
  "subnets": [
    {
      "addressPrefix": "10.0.0.0/24",
      "etag": "W/\"\"",
      "id":
"/subscriptions//resourceGroups/azrhelclirsgrp/providers/Microsoft.Network/virtualNetworks/azr
helclivnet1/subnets/azrhelclisubnet1",
        "ipConfigurations": null,
        "name": "azrhelclisubnet1",
        "networkSecurityGroup": null,
        "provisioningState": "Succeeded",
        "resourceGroup": "azrhelclirsgrp",
        "resourceNavigationLinks": null,
        "routeTable": null
      }
    ],
    "tags": {},
    "type": "Microsoft.Network/virtualNetworks",
    "virtualNetworkPeerings": null
    }
}
```

Additional resources

- Azure Managed Disks Overview
- SKU Types

1.8. UPLOADING AND CREATING AN AZURE IMAGE

Complete the following steps to upload the **VHD** file to your container and create an Azure custom image.

> **NOTE**
>
> The exported storage connection string does not persist after a system reboot. If any of the commands in the following steps fail, export the connection string again.

Procedure

1. Upload the **VHD** file to the storage container. It may take several minutes. To get a list of storage containers, enter the **az storage container list** command.

```
$ az storage blob upload --account-name <storage-account-name> --container-name <container-name> --type page --file <path-to-vhd> --name <image-name>.vhd
```

Example:

```
[clouduser@localhost]$ az storage blob upload --account-name azrhelclistact --container-name azrhelclistcont --type page --file rhel-image-8.vhd --name rhel-image-8.vhd
Percent complete: %100.0
```

2. Get the URL for the uploaded **VHD** file to use in the following step.

```
$ az storage blob url -c <container-name> -n <image-name>.vhd
```

Example:

```
$ az storage blob url -c azrhelclistcont -n rhel-image-8.vhd
"https://azrhelclistact.blob.core.windows.net/azrhelclistcont/rhel-image-8.vhd"
```

3. Create the Azure custom image.

```
$ az image create -n <image-name> -g <resource-group> -l <azure-region> --source
<URL> --os-type linux
```

NOTE

The default hypervisor generation of the VM is V1. You can optionally specify a V2 hypervisor generation by including the option **--hyper-v-generation V2**. Generation 2 VMs use a UEFI-based boot architecture. See Support for generation 2 VMs on Azure for information on generation 2 VMs.

The command may return the error "Only blobs formatted as VHDs can be imported." This error may mean that the image was not aligned to the nearest 1 MB boundary before it was converted to **VHD**.

Example:

```
$ az image create -n rhel8 -g azrhelclirsgrp2 -l southcentralus --source
https://azrhelclistact.blob.core.windows.net/azrhelclistcont/rhel-image-8.vhd --os-type
linux
```

1.9. CREATING AND STARTING THE VM IN AZURE

The following steps provide the minimum command options to create a managed-disk Azure VM from the image. See az vm create for additional options.

Procedure

1. Enter the following command to create the VM.

NOTE

The option **--generate-ssh-keys** creates a private/public key pair. Private and public key files are created in ~**/.ssh** on your system. The public key is added to the **authorized_keys** file on the VM for the user specified by the **--admin-username** option. See Other authentication methods for additional information.

```
$ az vm create -g <resource-group> -l <azure-region> -n <vm-name> --vnet-name <vnet-
name> --subnet <subnet-name> --size Standard_A2 --os-disk-name <simple-name> --
admin-username <administrator-name> --generate-ssh-keys --image <path-to-image>
```

Example:

```
[clouduser@localhost]$ az vm create -g azrhelclirsgrp2 -l southcentralus -n rhel-azure-vm-1 -
-vnet-name azrhelclivnet1 --subnet azrhelclisubnet1  --size Standard_A2 --os-disk-name vm-
1-osdisk --admin-username clouduser --generate-ssh-keys --image rhel8

{
  "fqdns": "",
  "id":
"/subscriptions//resourceGroups/azrhelclirsgrp/providers/Microsoft.Compute/virtualMachines/rhe
```

```
l-azure-vm-1",
  "location": "southcentralus",
  "macAddress": "",
  "powerState": "VM running",
  "privateIpAddress": "10.0.0.4",
  "publicIpAddress": "<public-IP-address>",
  "resourceGroup": "azrhelclirsgrp2"
```

Note the **publicIpAddress**. You need this address to log in to the VM in the following step.

2. Start an SSH session and log in to the VM.

```
[clouduser@localhost]$ ssh  -i /home/clouduser/.ssh/id_rsa clouduser@<public-IP-address>.
The authenticity of host ',<public-IP-address>' can't be established.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '<public-IP-address>' (ECDSA) to the list of known hosts.

[clouduser@rhel-azure-vm-1 ~]$
```

If you see a user prompt, you have successfully deployed your Azure VM.

You can now go to the Microsoft Azure portal and check the audit logs and properties of your resources. You can manage your VMs directly in this portal. If you are managing multiple VMs, you should use the Azure CLI. The Azure CLI provides a powerful interface to your resources in Azure. Enter **az --help** in the CLI or see the Azure CLI command reference to learn more about the commands you use to manage your VMs in Microsoft Azure.

1.10. OTHER AUTHENTICATION METHODS

While recommended for increased security, using the Azure-generated key pair is not required. The following examples show two methods for SSH authentication.

Example 1: These command options provision a new VM without generating a public key file. They allow SSH authentication using a password.

```
$ az vm create -g <resource-group> -l <azure-region> -n <vm-name> --vnet-name <vnet-name> --
subnet <subnet-name> --size Standard_A2 --os-disk-name <simple-name> --authentication-type
password --admin-username <administrator-name> --admin-password <ssh-password> --image
<path-to-image>
```

```
$ ssh <admin-username>@<public-ip-address>
```

Example 2: These command options provision a new Azure VM and allow SSH authentication using an existing public key file.

```
$ az vm create -g <resource-group> -l <azure-region> -n <vm-name> --vnet-name <vnet-name> --
subnet <subnet-name> --size Standard_A2 --os-disk-name <simple-name> --admin-username
<administrator-name> --ssh-key-value <path-to-existing-ssh-key> --image <path-to-image>
```

```
$ ssh -i <path-to-existing-ssh-key> <admin-username>@<public-ip-address>
```

1.11. ATTACHING RED HAT SUBSCRIPTIONS

Complete the following steps to attach the subscriptions you previously enabled through the Red Hat Cloud Access program.

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 using the ID of the subscription pool (Pool ID). See Attaching and Removing Subscriptions Through the Command Line .

Additional resources

- Creating Red Hat Customer Portal Activation Keys
- Attaching and Removing Subscriptions Through the Command Line
- Using and Configuring Red Hat Subscription Manager

1.12. SETTING UP AUTOMATIC REGISTRATION ON AZURE GOLD IMAGES

To make deploying RHEL 8 virtual machines (VM) on Micorsoft Azure faster and more comfortable, you can set up Gold Images of RHEL 8 to be automatically registered to the Red Hat Subscription Manager (RHSM).

Prerequisites

- You have enabled an eligible Red Hat product subscription for Cloud Access on Azure, and RHEL 8 Gold Images are therefore available to you in Microsoft Azure. For instructions, see Using Gold Images on Azure.

> **NOTE**
>
> A Microsoft Azure account can only be attached to a single Red Hat account at a time. Therefore, ensure no other users require access to the Azure account before attaching it to your Red Hat one.

Procedure

1. Use the Gold Image to create a RHEL 8 VM in your Azure instance. For instructions, see Creating and starting the VM in Azure .

2. Start the created VM.

3. In the RHEL 8 VM, enable automatic registration.

   ```
   # subscription-manager config --rhsmcertd.auto_registration=1
   ```

4. Enable the **rhsmcertd** service.

   ```
   # systemctl enable rhsmcertd.service
   ```

5. Disable the **redhat.repo** repository.

   ```
   # subscription-manager config --rhsm.manage_repos=0
   ```

6. Power off the VM, and save it as a managed image on Azure. For instructions, see How to create a managed image of a virtual machine or VHD.

7. Create VMs using the managed image. They will be automatically subscribed to RHSM.

Verification

- In a RHEL 8 VM created using the above instructions, verify the system is registered to RHSM by executing the **subscription-manager identity** command. On a successfully registered system, this displays the UUID of the system. For example:

  ```
  # subscription-manager identity
  system identity: fdc46662-c536-43fb-a18a-bbcb283102b7
  name: 192.168.122.222
  org name: 6340056
  org ID: 6340056
  ```

Additional resources

- Red Hat Gold Images in Azure
- Overview of RHEL images in Azure
- Configuring cloud sources for Red Hat services

CHAPTER 2. CONFIGURING A RED HAT HIGH AVAILABILITY CLUSTER ON MICROSOFT AZURE

This chapter provides information and procedures for configuring a Red Hat High Availability (HA) cluster on Azure using Azure virtual machine (VM) instances as cluster nodes. The procedures in this chapter assume you are creating a custom image for Azure. You have a number of options for obtaining the RHEL 8 images you use for your cluster. See Red Hat Enterprise Linux Image Options on Azure for information on image options for Azure.

This chapter includes:

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

The chapter refers to the Azure documentation in a number of places. For many procedures, see the referenced Azure documentation for more information.

Prerequisites

- Sign up for a Red Hat Customer Portal account .
- Sign up for a Microsoft Azure account with administrator privileges.
- You need to install Azure command line interface (CLI). For more information, see Installing the Azure CLI.
- Enable your subscriptions in the Red Hat Cloud Access program . The Red Hat Cloud Access program allows you to move your Red Hat subscriptions from physical or on-premise systems onto Azure with full support from Red Hat.
    - Alternatively, RHEL images can be obtained on-demand using Azure Marketplace.

2.1. ADDITIONAL RESOURCES

- Support Policies for RHEL High Availability Clusters - Microsoft Azure Virtual Machines as Cluster Members
- Configuring and Managing High Availability Clusters

2.2. CREATING RESOURCES IN AZURE

Complete the following procedure to create a region, resource group, storage account, virtual network, and availability set. You need these resources to complete subsequent tasks in this chapter.

Procedure

1. Authenticate your system with Azure and log in.

```
$ az login
```

NOTE

If a browser is available in your environment, the CLI opens your browser to the Azure sign-in page.

Example:

```
[clouduser@localhost]$ az login

To sign in, use a web browser to open the page https://aka.ms/devicelogin and enter the code FDMSCMETZ to authenticate.
  [
    {
      "cloudName": "AzureCloud",
      "id": "Subscription ID",
      "isDefault": true,
      "name": "MySubscriptionName",
      "state": "Enabled",
      "tenantId": "Tenant ID",
      "user": {
        "name": "clouduser@company.com",
        "type": "user"
      }
    }
  ]
```

2. Create a resource group in an Azure region.

```
$ az group create --name resource-group --location azure-region
```

Example:

```
[clouduser@localhost]$ az group create --name azrhelclirsgrp --location southcentralus

{
  "id": "/subscriptions//resourceGroups/azrhelclirsgrp",
  "location": "southcentralus",
  "managedBy": null,
  "name": "azrhelclirsgrp",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null
}
```

3. Create a storage account.

```
$ az storage account create -l azure-region -n storage-account-name -g resource-group --sku sku_type --kind StorageV2
```

Example:

```
[clouduser@localhost]$ az storage account create -l southcentralus -n azrhelclistact -g azrhelclirsgrp --sku Standard_LRS --kind StorageV2
```

```
{
  "accessTier": null,
  "creationTime": "2017-04-05T19:10:29.855470+00:00",
  "customDomain": null,
  "encryption": null,
  "id":
"/subscriptions//resourceGroups/azrhelclirsgrp/providers/Microsoft.Storage/storageAccounts/azr
helclistact",
  "kind": "StorageV2",
  "lastGeoFailoverTime": null,
  "location": "southcentralus",
  "name": "azrhelclistact",
  "primaryEndpoints": {
    "blob": "https://azrhelclistact.blob.core.windows.net/",
    "file": "https://azrhelclistact.file.core.windows.net/",
    "queue": "https://azrhelclistact.queue.core.windows.net/",
    "table": "https://azrhelclistact.table.core.windows.net/"
},
"primaryLocation": "southcentralus",
"provisioningState": "Succeeded",
"resourceGroup": "azrhelclirsgrp",
"secondaryEndpoints": null,
"secondaryLocation": null,
"sku": {
  "name": "Standard_LRS",
  "tier": "Standard"
},
"statusOfPrimary": "available",
"statusOfSecondary": null,
"tags": {},
  "type": "Microsoft.Storage/storageAccounts"
}
```

4. Get the storage account connection string.

```
$ az storage account show-connection-string -n storage-account-name -g resource-
group
```

Example:

```
[clouduser@localhost]$ az storage account show-connection-string -n azrhelclistact -g
azrhelclirsgrp
{
  "connectionString":
"DefaultEndpointsProtocol=https;EndpointSuffix=core.windows.net;AccountName=azrhelclistact
AccountKey=NreGk...=="
}
```

5. Export the connection string by copying the connection string and pasting it into the following command. This string connects your system to the storage account.

```
$ export AZURE_STORAGE_CONNECTION_STRING="storage-connection-string"
```

Example:

```
[clouduser@localhost]$ export
AZURE_STORAGE_CONNECTION_STRING="DefaultEndpointsProtocol=https;Endpoi
ntSuffix=core.windows.net;AccountName=azrhelclistact;AccountKey=NreGk...=="
```

6. Create the storage container.

```
$ az storage container create -n container-name
```

Example:

```
[clouduser@localhost]$ az storage container create -n azrhelclistcont

{
  "created": true
}
```

7. Create a virtual network. All cluster nodes must be in the same virtual network.

```
$ az network vnet create -g resource group --name vnet-name --subnet-name subnet-
name
```

Example:

```
[clouduser@localhost]$ az network vnet create --resource-group azrhelclirsgrp --name
azrhelclivnet1 --subnet-name azrhelclisubnet1
{
  "newVNet": {
    "addressSpace": {
      "addressPrefixes": [
      "10.0.0.0/16"
      ]
  },
  "dhcpOptions": {
    "dnsServers": []
  },
  "etag": "W/\"\"",
  "id":
"/subscriptions//resourceGroups/azrhelclirsgrp/providers/Microsoft.Network/virtualNetworks/azr
helclivnet1",
  "location": "southcentralus",
  "name": "azrhelclivnet1",
  "provisioningState": "Succeeded",
  "resourceGroup": "azrhelclirsgrp",
  "resourceGuid": "0f25efee-e2a6-4abe-a4e9-817061ee1e79",
  "subnets": [
    {
      "addressPrefix": "10.0.0.0/24",
      "etag": "W/\"\"",
      "id":
"/subscriptions//resourceGroups/azrhelclirsgrp/providers/Microsoft.Network/virtualNetworks/azr
helclivnet1/subnets/azrhelclisubnet1",
      "ipConfigurations": null,
      "name": "azrhelclisubnet1",
      "networkSecurityGroup": null,
        "provisioningState": "Succeeded",
        "resourceGroup": "azrhelclirsgrp",
        "resourceNavigationLinks": null,
        "routeTable": null
      }
    ],
    "tags": {},
    "type": "Microsoft.Network/virtualNetworks",
    "virtualNetworkPeerings": null
    }
  }
```

8. Create an availability set. All cluster nodes must be in the same availability set.

```
$ az vm availability-set create --name MyAvailabilitySet --resource-group
MyResourceGroup
```

Example:

```
[clouduser@localhost]$ az vm availability-set create --name rhelha-avset1 --resource-
group azrhelclirsgrp
{
  "additionalProperties": {},
    "id":
"/subscriptions/.../resourceGroups/azrhelclirsgrp/providers/Microsoft.Compute/availabilitySets/rh
elha-avset1",
    "location": "southcentralus",
    "name": “rhelha-avset1",
    "platformFaultDomainCount": 2,
    "platformUpdateDomainCount": 5,

...omitted
```

Additional resources

- Sign in with Azure CLI
- SKU Types
- Azure Managed Disks Overview

2.3. REQUIRED SYSTEM PACKAGES FOR HIGH AVAILABILITY

The procedure assumes you are creating a VM image for Azure HA using Red Hat Enterprise Linux. To successfully complete the procedure, the following packages must be installed.

Table 2.1. System packages

Package	Repository	Description
libvirt	rhel-8-for-x86_64-appstream-rpms	Open source API, daemon, and management tool for managing platform virtualization

Package	Repository	Description
virt-install	rhel-8-for-x86_64-appstream-rpms	A command-line utility for building VMs
libguestfs	rhel-8-for-x86_64-appstream-rpms	A library for accessing and modifying VM file systems
libguestfs-tools	rhel-8-for-x86_64-appstream-rpms	System administration tools for VMs; includes the **guestfish** utility

2.4. AZURE VM CONFIGURATION SETTINGS

Azure VMs must have the following configuration settings. Some of these settings are enabled during the initial VM creation. Other settings are set when provisioning the VM image for Azure. Keep these settings in mind as you move through the procedures. Refer to them as necessary.

Table 2.2. VM configuration settings

Setting	Recommendation
ssh	ssh must be enabled to provide remote access to your Azure VMs.
dhcp	The primary virtual adapter should be configured for dhcp (IPv4 only).
Swap Space	Do not create a dedicated swap file or swap partition. You can configure swap space with the Windows Azure Linux Agent (WALinuxAgent).
NIC	Choose **virtio** for the primary virtual network adapter.
encryption	For custom images, use Network Bound Disk Encryption (NBDE) for full disk encryption on Azure.

2.5. INSTALLING HYPER-V DEVICE DRIVERS

Microsoft provides network and storage device drivers as part of their Linux Integration Services (LIS) for Hyper-V package. You may need to install Hyper-V device drivers on the VM image prior to provisioning it as an Azure virtual machine (VM). Use the **lsinitrd | grep hv** command to verify that the drivers are installed.

Procedure

1. Enter the following **grep** command to determine if the required Hyper-V device drivers are installed.

```
# lsinitrd | grep hv
```

In the example below, all required drivers are installed.

```
# lsinitrd | grep hv
drwxr-xr-x   2 root     root           0 Aug 12 14:21 usr/lib/modules/3.10.0-932.el8.x86_64/kernel/drivers/hv
-rw-r--r--   1 root     root       31272 Aug 11 08:45 usr/lib/modules/3.10.0-932.el8.x86_64/kernel/drivers/hv/hv_vmbus.ko.xz
-rw-r--r--   1 root     root       25132 Aug 11 08:46 usr/lib/modules/3.10.0-932.el8.x86_64/kernel/drivers/net/hyperv/hv_netvsc.ko.xz
-rw-r--r--   1 root     root        9796 Aug 11 08:45 usr/lib/modules/3.10.0-932.el8.x86_64/kernel/drivers/scsi/hv_storvsc.ko.xz
```

If all the drivers are not installed, complete the remaining steps.

NOTE

An **hv_vmbus** driver may exist in the environment. Even if this driver is present, complete the following steps.

2. Create a file named **hv.conf** in **/etc/dracut.conf.d**.

3. Add the following driver parameters to the **hv.conf** file.

```
add_drivers+=" hv_vmbus "
add_drivers+=" hv_netvsc "
add_drivers+=" hv_storvsc "
add_drivers+=" nvme "
```

NOTE

Note the spaces before and after the quotes, for example, **add_drivers+=" hv_vmbus "**. This ensures that unique drivers are loaded in the event that other Hyper-V drivers already exist in the environment.

4. Regenerate the **initramfs** image.

```
# dracut -f -v --regenerate-all
```

Verification

1. Reboot the machine.

2. Run the **lsinitrd | grep hv** command to verify that the drivers are installed.

2.6. MAKING CONFIGURATION CHANGES REQUIRED FOR A MICROSOFT AZURE DEPLOYMENT

Before you deploy your custom base image to Azure, you must perform additional configuration changes to ensure that the virtual machine (VM) can properly operate in Azure.

Procedure

1. Log in to the VM.
2. Register the VM, and enable the Red Hat Enterprise Linux 8 repository.

   ```
   # subscription-manager register --auto-attach
   Installed Product Current Status:
   Product Name: Red Hat Enterprise Linux for x86_64
   Status: Subscribed
   ```

3. Ensure that the **cloud-init** and **hyperv-daemons** packages are installed.

   ```
   # yum install cloud-init hyperv-daemons -y
   ```

4. Create **cloud-init** configuration files that are needed for integration with Azure services:
   a. To enable logging to the Hyper-V Data Exchange Service (KVP), create the **/etc/cloud/cloud.cfg.d/10-azure-kvp.cfg** configuration file and add the following lines to that file.

      ```
      reporting:
          logging:
              type: log
          telemetry:
              type: hyperv
      ```

   b. To add Azure as a datasource, create the **/etc/cloud/cloud.cfg.d/91-azure_datasource.cfg** configuration file, and add the following lines to that file.

      ```
      datasource_list: [ Azure ]
      datasource:
          Azure:
              apply_network_config: False
      ```

5. To ensure that specific kernel modules are blocked from loading automatically, edit or create the **/etc/modprobe.d/blocklist.conf** file and add the following lines to that file.

   ```
   blacklist nouveau
   blacklist lbm-nouveau
   blacklist floppy
   blacklist amdgpu
   blacklist skx_edac
   blacklist intel_cstate
   ```

6. Modify **udev** network device rules:
   a. Remove the following persistent network device rules if present.

      ```
      # rm -f /etc/udev/rules.d/70-persistent-net.rules
      # rm -f /etc/udev/rules.d/75-persistent-net-generator.rules
      # rm -f /etc/udev/rules.d/80-net-name-slot-rules
      ```

   b. [illegible]

b. To ensure that Accelerated Networking on Azure works as intended, create a new network device rule **/etc/udev/rules.d/68-azure-sriov-nm-unmanaged.rules** and add the following line to it.

```
SUBSYSTEM=="net", DRIVERS=="hv_pci", ACTION=="add",
ENV{NM_UNMANAGED}="1"
```

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

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

8. Modify kernel boot parameters:

a. Open the **/etc/default/grub** file, and ensure the **GRUB_TIMEOUT** line has the following value.

```
GRUB_TIMEOUT=10
```

b. Remove the following options from the end of the **GRUB_CMDLINE_LINUX** line if present.

```
rhgb quiet
```

c. Ensure the **/etc/default/grub** file contains the following lines with all the specified options.

```
GRUB_CMDLINE_LINUX="loglevel=3 crashkernel=auto console=tty1 console=ttyS0
earlyprintk=ttyS0 rootdelay=300"
GRUB_TIMEOUT_STYLE=countdown
GRUB_TERMINAL="serial console"
GRUB_SERIAL_COMMAND="serial --speed=115200 --unit=0 --word=8 --parity=no --
stop=1"
```

d. Regenerate the **grub.cfg** file.
On a BIOS-based machine:

```
# grub2-mkconfig -o /boot/grub2/grub.cfg
```

On a UEFI-based machine:

```
# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

If your system uses a non-default location for **grub.cfg**, adjust the command accordingly.

9. Configure the Windows Azure Linux Agent (**WALinuxAgent**):

a. Install and enable the **WALinuxAgent** package.

```
# yum install WALinuxAgent -y
# systemctl enable waagent
```

b. To ensure that a swap partition is not used in provisioned VMs, edit the following lines in the **/etc/waagent.conf** file.

```
Provisioning.DeleteRootPassword=y
ResourceDisk.Format=n
ResourceDisk.EnableSwap=n
```

10. Prepare the VM for Azure provisioning:

    a. Unregister the VM from Red Hat Subscription Manager.

    ```
    # subscription-manager unregister
    ```

    b. Clean up the existing provisioning details.

    ```
    # waagent -force -deprovision
    ```

    NOTE

    This command generates warnings, which are expected because Azure handles the provisioning of VMs automatically.

    c. Clean the shell history and shut down the VM.

    ```
    # export HISTSIZE=0
    # poweroff
    ```

2.7. CREATING AN AZURE ACTIVE DIRECTORY APPLICATION

Complete the following procedure to create an Azure Active Directory AD Application. The Azure AD Application authorizes and automates access for HA operations for all nodes in the cluster.

Prerequisites

Install the Azure Command Line Interface (CLI).

Procedure

1. Ensure you are an Administrator or Owner for the Microsoft Azure subscription. You need this authorization to create an Azure AD application.

2. Log in to your Azure account.

   ```
   $ az login
   ```

3. Enter the following command to create the Azure AD Application. To use your own password, add the **--password** option to the command. Ensure that you create a strong password.

   ```
   $ az ad sp create-for-rbac --name FencingApplicationName --role owner --scopes
   "/subscriptions/SubscriptionID/resourceGroups/MyResourseGroup"
   ```

   Example:

   ```
   [clouduser@localhost ~] $ az ad sp create-for-rbac --name FencingApp --role owner --
   scopes "/subscriptions/2586c64b-xxxxxx-xxxxxxx-xxxxxxx/resourceGroups/azrhelclirsgrp"
   ```

```
Retrying role assignment creation: 1/36
Retrying role assignment creation: 2/36
Retrying role assignment creation: 3/36
{
  "appId": "1a3dfe06-df55-42ad-937b-326d1c211739",
  "displayName": "FencingApp",
  "name": "http://FencingApp",
  "password": "43a603f0-64bb-482e-800d-402efe5f3d47",
  "tenant": "77ecefb6-xxxxxxxxxx-xxxxxxx-757a69cb9485"
}
```

4. Save the following information before proceeding. You need this information to set up the fencing agent.
   - Azure AD Application ID
   - Azure AD Application Password
   - Tenant ID
   - Microsoft Azure Subscription ID

Additional resources

- View the access a user has to Azure resources

2.8. CONVERTING THE IMAGE TO A FIXED VHD FORMAT

All Microsoft Azure VM images must be in a fixed **VHD** format. The image must be aligned on a 1 MB boundary before it is converted to VHD. This section describes how to convert the image from **qcow2** to a fixed **VHD** format and align the image, if necessary. Once you have converted the image, you can upload it to Azure.

Procedure

1. Convert the image from **qcow2** to **raw** format.

```
$ qemu-img convert -f qcow2 -O raw <image-name>.qcow2 <image-name>.raw
```

2. Create a shell script using the contents below.

```
#!/bin/bash
MB=$((1024 * 1024))
size=$(qemu-img info -f raw --output json "$1" | gawk 'match($0, /"virtual-size": ([0-9]+),/, val)
{print val[1]}')
rounded_size=$((($size/$MB + 1) * $MB))
if [ $(($size % $MB)) -eq  0 ]
then
 echo "Your image is already aligned. You do not need to resize."
 exit 1
fi
echo "rounded size = $rounded_size"
export rounded_size
```

3. Run the script. This example uses the name **align.sh**.

   ```
   $ sh align.sh <image-xxx>.raw
   ```

   - If the message *"Your image is already aligned. You do not need to resize."* displays, proceed to the following step.
   - If a value displays, your image is not aligned.

4. Use the following command to convert the file to a fixed **VHD** format.
   The sample uses qemu-img version 2.12.0.

   ```
   $ qemu-img convert -f raw -o subformat=fixed,force_size -O vpc <image-xxx>.raw
   <image.xxx>.vhd
   ```

   Once converted, the **VHD** file is ready to upload to Azure.

5. If the **raw** image is not aligned, complete the following steps to align it.

   a. Resize the **raw** file using the rounded value displayed when you ran the verification script.

   ```
   $ qemu-img resize -f raw <image-xxx>.raw <rounded-value>
   ```

   b. Convert the **raw** image file to a **VHD** format.
   The sample uses qemu-img version 2.12.0.

   ```
   $ qemu-img convert -f raw -o subformat=fixed,force_size -O vpc <image-xxx>.raw
   <image.xxx>.vhd
   ```

   Once converted, the **VHD** file is ready to upload to Azure.

2.9. UPLOADING AND CREATING AN AZURE IMAGE

Complete the following steps to upload the **VHD** file to your container and create an Azure custom image.

NOTE

The exported storage connection string does not persist after a system reboot. If any of the commands in the following steps fail, export the connection string again.

Procedure

1. Upload the **VHD** file to the storage container. It may take several minutes. To get a list of storage containers, enter the **az storage container list** command.

   ```
   $ az storage blob upload --account-name <storage-account-name> --container-name
   <container-name> --type page --file <path-to-vhd> --name <image-name>.vhd
   ```

   Example:

```
[clouduser@localhost]$ az storage blob upload --account-name azrhelclistact --container-
name azrhelclistcont --type page --file rhel-image-8.vhd --name rhel-image-8.vhd
Percent complete: %100.0
```

2. Get the URL for the uploaded **VHD** file to use in the following step.

```
$ az storage blob url -c <container-name> -n <image-name>.vhd
```

Example:

```
$ az storage blob url -c azrhelclistcont -n rhel-image-8.vhd
"https://azrhelclistact.blob.core.windows.net/azrhelclistcont/rhel-image-8.vhd"
```

3. Create the Azure custom image.

```
$ az image create -n <image-name> -g <resource-group> -l <azure-region> --source
<URL> --os-type linux
```

NOTE

The default hypervisor generation of the VM is V1. You can optionally specify a V2 hypervisor generation by including the option **--hyper-v-generation V2**. Generation 2 VMs use a UEFI-based boot architecture. See Support for generation 2 VMs on Azure for information on generation 2 VMs.

The command may return the error "Only blobs formatted as VHDs can be imported." This error may mean that the image was not aligned to the nearest 1 MB boundary before it was converted to **VHD**.

Example:

```
$ az image create -n rhel8 -g azrhelclirsgrp2 -l southcentralus --source
https://azrhelclistact.blob.core.windows.net/azrhelclistcont/rhel-image-8.vhd --os-type
linux
```

2.10. INSTALLING RED HAT HA PACKAGES AND AGENTS

Complete the following steps on all nodes.

Procedure

1. Launch an SSH terminal session and connect to the VM using the administrator name and public IP address.

```
$ ssh administrator@PublicIP
```

To get the public IP address for an Azure VM, open the VM properties in the Azure Portal or enter the following Azure CLI command.

```
$ az vm list -g <resource-group> -d --output table
```

Example:

```
[clouduser@localhost ~] $ az vm list -g azrhelclirsgrp -d --output table
Name    ResourceGroup          PowerState      PublicIps       Location
------  ---------------------  --------------  -------------   --------------
node01  azrhelclirsgrp         VM running      192.98.152.251  southcentralus
```

2. Register the VM with Red Hat.

```
$ sudo -i
# subscription-manager register --auto-attach
```

NOTE

If the **--auto-attach** command fails, manually register the VM to your subscription.

3. Disable all repositories.

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

4. Enable the RHEL 8 Server HA repositories.

```
# subscription-manager repos --enable=rhel-8-for-x86_64-highavailability-rpms
```

5. Update all packages.

```
# yum update -y
```

6. Install the Red Hat High Availability Add-On software packages, along with all available fencing agents from the High Availability channel.

```
# yum install pcs pacemaker fence-agents-azure-arm
```

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

```
# passwd hacluster
```

8. Add the **high availability** service to the RHEL Firewall if **firewalld.service** is installed.

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

9. 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

- Ensure the **pcs** 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 Fri 2018-02-23 11:00:58 EST; 1min 23s ago
Docs: man:pcsd(8)
          man:pcs(8)
Main PID: 46235 (pcsd)
  CGroup: /system.slice/pcsd.service
          └─46235 /usr/bin/ruby /usr/lib/pcsd/pcsd > /dev/null &
```

2.11. CREATING A CLUSTER

Complete the following steps to create the cluster of nodes.

Procedure

1. On one of the nodes, enter the following command to authenticate the pcs user **hacluster**. In the command, specify the name of each node in the cluster.

   ```
   # pcs host auth <hostname1> <hostname2> <hostname3>
   ```

   Example:

   ```
   [root@node01 clouduser]# pcs host auth node01 node02 node03
   Username: hacluster
   Password:
   node01: Authorized
   node02: Authorized
   node03: Authorized
   ```

2. Create the cluster.

   ```
   # pcs cluster setup <cluster_name> <hostname1> <hostname2> <hostname3>
   ```

   Example:

   ```
   [root@node01 clouduser]# pcs cluster setup new_cluster node01 node02 node03

   ...omitted

   Synchronizing pcsd certificates on nodes node01, node02, node03...
   node02: Success
   node03: Success
   node01: Success
   Restarting pcsd on the nodes in order to reload the certificates...
   node02: Success
   node03: Success
   node01: Success
   ```

Verification

1. Enable the cluster.

```
[root@node01 clouduser]# pcs cluster enable --all
node02: Cluster Enabled
node03: Cluster Enabled
node01: Cluster Enabled
```

2. Start the cluster.

```
[root@node01 clouduser]# pcs cluster start --all
node02: Starting Cluster...
node03: Starting Cluster...
node01: Starting Cluster...
```

2.12. FENCING OVERVIEW

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 node that is unresponsive may still be accessing data. The only way to be certain that your data is safe is to fence the node using STONITH. STONITH is an acronym for "Shoot The Other Node In The Head," and it protects your data from being corrupted by rogue nodes or concurrent access. Using STONITH, you can be certain that a node is truly offline before allowing the data to be accessed from another node.

Additional resources

- Fencing in Red Hat High Availability Cluster

2.13. CREATING A FENCING DEVICE

Complete the following steps to configure fencing. Complete these commands from any node in the cluster

Prerequisites

You need to set the cluster property **stonith-enabled** to **true**.

Procedure

1. Identify the Azure node name for each RHEL VM. You use the Azure node names to configure the fence device.

```
# fence_azure_arm -l AD-Application-ID -p AD-Password --resourceGroup
MyResourceGroup --tenantId Tenant-ID --subscriptionId Subscription-ID -o list
```

Example:

```
[root@node01 clouduser]# fence_azure_arm -l e04a6a49-9f00-xxxx-xxxx-a8bdda4af447 -
p z/a05AwCN0IzAjVwXXXXXXXEWIoeVp0xg7QT//JE= --resourceGroup azrhelclirsgrp --
```

```
tenantId 77ecefb6-cff0-XXXX-XXXX-757XXXX9485 --subscriptionId XXXXXXXX-38b4-
4527-XXXX-012d49dfc02c -o list
node01,
node02,
node03,
```

2. View the options for the Azure ARM STONITH agent.

```
# pcs stonith describe fence_azure_arm
```

Example:

```
# pcs stonith describe fence_apc
Stonith options:
password: Authentication key
password_script: Script to run to retrieve password
```

WARNING

For fence agents that provide a method option, do not specify a value of cycle as it is not supported and can cause data corruption.

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.

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.

You can use **pcmk_host_map** parameter when creating a fencing device to map host names to the specifications that comprehends the fence device.

3. Create a fence device.

```
# pcs stonith create clusterfence fence_azure_arm
```

4. Test the fencing agent for one of the other nodes.

```
# pcs stonith fence azurenodename
```

Example:

```
[root@node01 clouduser]# pcs status
Cluster name: newcluster
Stack: corosync
Current DC: node01 (version 1.1.18-11.el7-2b07d5c5a9) - partition with quorum
Last updated: Fri Feb 23 11:44:35 2018
Last change: Fri Feb 23 11:21:01 2018 by root via cibadmin on node01
```

```
3 nodes configured
1 resource configured

Online: [ node01 node03 ]
OFFLINE: [ node02 ]

Full list of resources:

  clusterfence  (stonith:fence_azure_arm):  Started node01

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

5. Start the node that was fenced in the previous step.

```
# pcs cluster start hostname
```

6. Check the status to verify the node started.

```
# pcs status
```

Example:

```
[root@node01 clouduser]# pcs status
Cluster name: newcluster
Stack: corosync
Current DC: node01 (version 1.1.18-11.el7-2b07d5c5a9) - partition with quorum
Last updated: Fri Feb 23 11:34:59 2018
Last change: Fri Feb 23 11:21:01 2018 by root via cibadmin on node01

3 nodes configured
1 resource configured

Online: [ node01 node02 node03 ]

Full list of resources:

clusterfence    (stonith:fence_azure_arm):  Started node01

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

Additional resources

- Fencing in a Red Hat High Availability Cluster
- General properties of fencing devices

2.14. CREATING AN AZURE INTERNAL LOAD BALANCER

The Azure internal load balancer removes cluster nodes that do not answer health probe requests.

Perform the following procedure to create an Azure internal load balancer. Each step references a specific Microsoft procedure and includes the settings for customizing the load balancer for HA.

Prerequisites

Azure control panel

Procedure

1. Create a Basic load balancer . Select **Internal load balancer**, the **Basic SKU**, and **Dynamic** for the type of IP address assignment.

2. Create a back-end address pool . Associate the backend pool to the availability set created while creating Azure resources in HA. Do not set any target network IP configurations.

3. Create a health probe . For the health probe, select **TCP** and enter port **61000**. You can use TCP port number that does not interfere with another service. For certain HA product applications (for example, SAP HANA and SQL Server), you may need to work with Microsoft to identify the correct port to use.

4. Create a load balancer rule . To create the load balancing rule, the default values are prepopulated. Ensure to set **Floating IP (direct server return)** to **Enabled**.

2.15. CONFIGURING THE LOAD BALANCER RESOURCE AGENT

After you have created the health probe, you must configure the **load balancer** resource agent. This resource agent runs a service that answers health probe requests from the Azure load balancer and removes cluster nodes that do not answer requests.

Procedure

1. Install the **nmap-ncat** resource agents on all nodes.

   ```
   # yum install nmap-ncat resource-agents
   ```

   Perform the following steps on a single node.

2. Create the **pcs** resources and group. Use your load balancer FrontendIP for the IPaddr2 address.

   ```
   # pcs resource create resource-name IPaddr2 ip="10.0.0.7" --group cluster-resources-group
   ```

3. Configure the **load balancer** resource agent.

   ```
   # pcs resource create resource-loadbalancer-name azure-lb port=port-number --group cluster-resources-group
   ```

Verification

- Run **pcs status** to see the results.

```
[root@node01 clouduser]# pcs status
```

Example output:

```
Cluster name: clusterfence01
Stack: corosync
Current DC: node02 (version 1.1.16-12.el7_4.7-94ff4df) - partition with quorum
Last updated: Tue Jan 30 12:42:35 2018
Last change: Tue Jan 30 12:26:42 2018 by root via cibadmin on node01

3 nodes configured
3 resources configured

Online: [ node01 node02 node03 ]

Full list of resources:

clusterfence (stonith:fence_azure_arm):      Started node01
Resource Group: g_azure
    vip_azure  (ocf::heartbeat:IPaddr2):       Started node02
    lb_azure   (ocf::heartbeat:azure-lb):      Started node02

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

2.16. CONFIGURING SHARED BLOCK STORAGE

This section provides an optional procedure for configuring shared block storage for a Red Hat High Availability cluster with Microsoft Azure Shared Disks. The procedure assumes three Azure VMs (a three-node cluster) with a 1 TB shared disk.

NOTE

This is a stand-alone sample procedure for configuring block storage. The procedure assumes that you have not yet created your cluster.

Prerequisites

- You must have installed the Azure CLI on your host system and created your SSH key(s).
- You must have created your cluster environment in Azure, which includes creating the following resources. Links are to the Microsoft Azure documentation.
    - Resource group
    - Virtual network
    - Network security group(s)
    - Network security group rules
    - Subnet(s)

- Load balancer (optional)
- Storage account
- Proximity placement group
- Availability set

Procedure

1. Create a shared block volume using the Azure command **az disk create**.

```
$ az disk create -g <resource_group> -n <shared_block_volume_name> --size-gb
<disk_size> --max-shares <number_vms> -l <location>
```

For example, the following command creates a shared block volume named **shared-block-volume.vhd** in the resource group **sharedblock** within the Azure Availability Zone **westcentralus**.

```
$ az disk create -g sharedblock-rg -n shared-block-volume.vhd --size-gb 1024 --max-
shares 3 -l westcentralus

{
  "creationData": {
    "createOption": "Empty",
    "galleryImageReference": null,
    "imageReference": null,
    "sourceResourceId": null,
    "sourceUniqueId": null,
    "sourceUri": null,
    "storageAccountId": null,
    "uploadSizeBytes": null
  },
  "diskAccessId": null,
  "diskIopsReadOnly": null,
  "diskIopsReadWrite": 5000,
  "diskMbpsReadOnly": null,
  "diskMbpsReadWrite": 200,
  "diskSizeBytes": 1099511627776,
  "diskSizeGb": 1024,
  "diskState": "Unattached",
  "encryption": {
    "diskEncryptionSetId": null,
    "type": "EncryptionAtRestWithPlatformKey"
  },
  "encryptionSettingsCollection": null,
  "hyperVgeneration": "V1",
  "id": "/subscriptions/12345678910-12345678910/resourceGroups/sharedblock-
rg/providers/Microsoft.Compute/disks/shared-block-volume.vhd",
  "location": "westcentralus",
  "managedBy": null,
  "managedByExtended": null,
  "maxShares": 3,
  "name": "shared-block-volume.vhd",
  "networkAccessPolicy": "AllowAll",
  "osType": null,
  "provisioningState": "Succeeded",
  "resourceGroup": "sharedblock-rg",
  "shareInfo": null,
  "sku": {
    "name": "Premium_LRS",
    "tier": "Premium"
  },
  "tags": {},
  "timeCreated": "2020-08-27T15:36:56.263382+00:00",
  "type": "Microsoft.Compute/disks",
  "uniqueId": "cd8b0a25-6fbe-4779-9312-8d9cbb89b6f2",
  "zones": null
}
```

2. Verify that you have created the shared block volume using the Azure command **az disk show**.

   ```
   $ az disk show -g <resource_group> -n <shared_block_volume_name>
   ```

   For example, the following command shows details for the shared block volume **shared-block-volume.vhd** within the resource group **sharedblock-rg**.

   ```
   $ az disk show -g sharedblock-rg -n shared-block-volume.vhd

   {
     "creationData": {
       "createOption": "Empty",
       "galleryImageReference": null,
       "imageReference": null,
       "sourceResourceId": null,
       "sourceUniqueId": null,
       "sourceUri": null,
       "storageAccountId": null,
       "uploadSizeBytes": null
     },
     "diskAccessId": null,
     "diskIopsReadOnly": null,
     "diskIopsReadWrite": 5000,
     "diskMbpsReadOnly": null,
     "diskMbpsReadWrite": 200,
     "diskSizeBytes": 1099511627776,
     "diskSizeGb": 1024,
     "diskState": "Unattached",
     "encryption": {
       "diskEncryptionSetId": null,
       "type": "EncryptionAtRestWithPlatformKey"
     },
     "encryptionSettingsCollection": null,
     "hyperVgeneration": "V1",
     "id": "/subscriptions/12345678910-12345678910/resourceGroups/sharedblock-
   rg/providers/Microsoft.Compute/disks/shared-block-volume.vhd",
     "location": "westcentralus",
     "managedBy": null,
     "managedByExtended": null,
     "maxShares": 3,
     "name": "shared-block-volume.vhd",
   ```

```
  "networkAccessPolicy": "AllowAll",
  "osType": null,
  "provisioningState": "Succeeded",
  "resourceGroup": "sharedblock-rg",
  "shareInfo": null,
  "sku": {
    "name": "Premium_LRS",
    "tier": "Premium"
  },
  "tags": {},
  "timeCreated": "2020-08-27T15:36:56.263382+00:00",
  "type": "Microsoft.Compute/disks",
  "uniqueId": "cd8b0a25-6fbe-4779-9312-8d9cbb89b6f2",
  "zones": null
}
```

3. Create three network interfaces using the Azure command **az network nic create**. Run the following command three times using a different **<nic_name>** for each.

   ```
   $ az network nic create -g <resource_group> -n <nic_name> --subnet <subnet_name> --
   vnet-name <virtual_network> --location <location> --network-security-group
   <network_security_group> --private-ip-address-version IPv4
   ```

   For example, the following command creates a network interface with the name **shareblock-nodea-vm-nic-protected**.

   ```
   $ az network nic create -g sharedblock-rg -n sharedblock-nodea-vm-nic-protected --subnet
   sharedblock-subnet-protected --vnet-name sharedblock-vn --location westcentralus --
   network-security-group sharedblock-nsg --private-ip-address-version IPv4
   ```

4. Create three VMs and attach the shared block volume using the Azure command **az vm create**. Option values are the same for each VM except that each VM has its own **<vm_name>**, **<new_vm_disk_name>**, and **<nic_name>**.

   ```
   $ az vm create -n <vm_name> -g <resource_group> --attach-data-disks
   <shared_block_volume_name> --data-disk-caching None --os-disk-caching ReadWrite --os-
   disk-name <new-vm-disk-name> --os-disk-size-gb <disk_size> --location <location> --size
   <virtual_machine_size> --image <image_name> --admin-username <vm_username> --
   authentication-type ssh --ssh-key-values <ssh_key> --nics <nic_name> --availability-set
   <availability_set> --ppg <proximity_placement_group>
   ```

   For example, the following command creates a VM named **sharedblock-nodea-vm**.

   ```
   $ az vm create -n sharedblock-nodea-vm -g sharedblock-rg --attach-data-disks shared-
   block-volume.vhd --data-disk-caching None --os-disk-caching ReadWrite --os-disk-
   name sharedblock-nodea-vm.vhd --os-disk-size-gb 64 --location westcentralus --size
   Standard_D2s_v3 --image /subscriptions/12345678910-
   12345678910/resourceGroups/sample-
   azureimagesgroupwestcentralus/providers/Microsoft.Compute/images/sample-azure-
   rhel-8.3.0-20200713.n.0.x86_64 --admin-username sharedblock-user --authentication-
   type ssh --ssh-key-values @sharedblock-key.pub --nics sharedblock-nodea-vm-nic-
   protected --availability-set sharedblock-as --ppg sharedblock-ppg

   {
   ```

```
  "fqdns": "",
  "id": "/subscriptions/12345678910-12345678910/resourceGroups/sharedblock-
rg/providers/Microsoft.Compute/virtualMachines/sharedblock-nodea-vm",
  "location": "westcentralus",
  "macAddress": "00-22-48-5D-EE-FB",
  "powerState": "VM running",
  "privateIpAddress": "198.51.100.3",
  "publicIpAddress": "",
  "resourceGroup": "sharedblock-rg",
  "zones": ""
}
```

Verification

1. For each VM in your cluster, verify that the block device is available by using the **ssh** command with your VM **<ip_address>**.

   ```
   # ssh <ip_address> "hostname ; lsblk -d | grep ' 1T '"
   ```

   For example, the following command lists details including the host name and block device for the VM IP **198.51.100.3**.

   ```
   # ssh 198.51.100.3 "hostname ; lsblk -d | grep ' 1T '"

   nodea
   sdb    8:16   0    1T  0 disk
   ```

2. Use the **ssh** command to verify that each VM in your cluster uses the same shared disk.

   ```
   # ssh <ip_address> "hostname ; lsblk -d | grep ' 1T ' | awk '{print \$1}' | xargs -i udevadm info
   --query=all --name=/dev/{} | grep '^E: ID_SERIAL='"
   ```

   For example, the following command lists details including the host name and shared disk volume ID for the instance IP address **198.51.100.3**.

   ```
   # *ssh 198.51.100.3 "hostname ; lsblk -d | grep ' 1T ' | awk '{print \$1}' | xargs -i udevadm info
   --query=all --name=/dev/{} | grep '^E: ID_SERIAL='"*

   nodea
   E: ID_SERIAL=3600224808dd8eb102f6ffc5822c41d89
   ```

After you have verified that the shared disk is attached to each VM, you can configure resilient storage for the cluster.

Additional resources

- Configuring a GFS2 file system in a cluster
- Configuring GFS2 file systems

CHAPTER 3. DEPLOYING A RED HAT ENTERPRISE LINUX IMAGE AS AN EC2 INSTANCE ON AMAZON WEB SERVICES

You have a number of options for deploying a Red Hat Enterprise Linux (RHEL) 8 image as an EC2 instance on Amazon Web Services (AWS). This chapter discusses your options for choosing an image and lists or refers to system requirements for your host system and virtual machine (VM). This chapter also provides procedures for creating a custom VM from an ISO image, uploading it to EC2, and launching an EC2 instance.

To deploy a Red Hat Enterprise Linux 8 (RHEL 8) as an EC2 instance on Amazon Web Services (AWS), follow the information below. This chapter:

- Discusses your options for choosing an image
- Lists or refers to system requirements for your host system and virtual machine (VM)
- Provides procedures for creating a custom VM from an ISO image, uploading it to EC2, and launching an EC2 instance

IMPORTANT

While you can create a custom VM from an ISO image, Red Hat recommends that you use the Red Hat Image Builder product to create customized images for use on specific cloud providers. With Image Builder, you can create and upload an Amazon Machine Image (AMI) in the **ami** format. See Composing a Customized RHEL System Image for more information.

NOTE

For a list of Red Hat products that you can use securely on AWS, see Red Hat on Amazon Web Services.

Prerequisites

- Sign up for a Red Hat Customer Portal account.
- Sign up for AWS and set up your AWS resources. See Setting Up with Amazon EC2 for more information.
- Enable your subscriptions in the Red Hat Cloud Access program . The Red Hat Cloud Access program allows you to move your Red Hat subscriptions from physical or on-premise systems onto AWS with full support from Red Hat.

3.1. ADDITIONAL RESOURCES

- Red Hat Cloud Access Reference Guide
- Red Hat in the Public Cloud
- Red Hat Enterprise Linux on Amazon EC2 - FAQs
- Setting Up with Amazon EC2
- Red Hat on Amazon Web Services

3.2. RED HAT ENTERPRISE LINUX IMAGE OPTIONS ON AWS

The following table lists image choices and notes the differences in the image options.

Table 3.1. Image options

Image option	Subscriptions	Sample scenario	Considerations
Choose to deploy a Red Hat Gold Image.	Leverage your existing Red Hat subscriptions.	Enable subscriptions through the Red Hat Cloud Access program, and then choose a Red Hat Gold Image on AWS.	The subscription includes the Red Hat product cost; you pay Amazon for all other instance costs. Red Hat Gold Images are called "Cloud Access" images because you leverage your existing Red Hat subscriptions. Red Hat provides support directly for Cloud Access images.
Choose to deploy a custom image that you move to AWS.	Leverage your existing Red Hat subscriptions.	Enable subscriptions through the Red Hat Cloud Access program, upload your custom image, and attach your subscriptions.	The subscription includes the Red Hat product cost; you pay Amazon for all other instance costs. Custom images that you move to AWS are "Cloud Access" images because you leverage your existing Red Hat subscriptions. Red Hat provides support directly for Cloud Access images.
Choose to deploy an existing Amazon image that includes RHEL.	The AWS EC2 images include a Red Hat product.	Choose a RHEL image when you launch an instance on the AWS Management Console, or choose an image from the AWS Marketplace.	You pay Amazon hourly on a pay-as-you-go model. Such images are called "on-demand" images. Amazon provides support for on-demand images. Red Hat provides updates to the images. AWS makes the updates available through the Red Hat Update Infrastructure (RHUI).

NOTE

You can create a custom image for AWS 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 Red Hat Cloud Access instance. To change from an on-demand image to a Red Hat Cloud Access bring-your-own-subscription (BYOS) image, create a new Red Hat Cloud Access instance and migrate data from your on-demand instance. Cancel your on-demand instance after you migrate your data to avoid double billing.

The remainder of this chapter includes information and procedures pertaining to custom images.

Additional resources

- Red Hat Cloud Access program
- Composing a Customized RHEL System Image
- AWS Management Console
- AWS Marketplace

3.3. UNDERSTANDING BASE IMAGES

This section includes information on using preconfigured base images and their configuration settings.

3.3.1. 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

3.3.2. 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.

3.4. CREATING A BASE VM FROM AN ISO IMAGE

Follow the procedures in this section to create a RHEL 8 base image from an ISO image.

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**.

3.4.1. 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 8 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**.
    A basic command-line sample follows.

    ```
    virt-install --name kvmtest --memory 2048 --vcpus 2 --disk rhel-8.0-x86_64-
    kvm.qcow2,bus=virtio --import --os-variant=rhel8.0
    ```

    b. If you use the web console to create your VM, follow the procedure in Creating virtual machines 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.

3.4.2. Completing the RHEL installation

Perform the following steps to complete the installation and to 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 Management (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 8 repository.

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

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

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

7. **Important: This step is only for VMs you intend to upload to AWS.**
   a. For AMD64 or Intel 64 (x86_64)VMs, install the **nvme**, **xen-netfront**, and **xen-blkfront** drivers.

      ```
      # dracut -f --add-drivers "nvme xen-netfront xen-blkfront"
      ```

   b. For ARM 64 (aarch64) VMs, install the **nvme** driver.

      ```
      # dracut -f --add-drivers "nvme"
      ```

      Including these drivers removes the possibility of a dracut time-out.

      Alternatively, you can add the drivers to **/etc/dracut.conf.d/** and then enter **dracut -f** to overwrite the existing **initramfs** file.

8. Power down the VM.

Additional resources

- Understanding autoattaching subscriptions on the Customer Portal
- Introduction to cloud-init

3.5. UPLOADING THE RED HAT ENTERPRISE LINUX IMAGE TO AWS

Follow the procedures in this section to upload your image to AWS.

3.5.1. Installing the AWS CLI

Many of the procedures required to manage HA clusters in AWS include using the AWS CLI. Complete the following steps to install the AWS CLI.

Prerequisites

- You have created an AWS Access Key ID and an AWS Secret Access Key, and have access to them. For instructions and details, see Quickly Configuring the AWS CLI.

Procedure

1. Install the AWS command line tools using the **yum** command.

```
# yum install awscli
```

2. Use the **aws --version** command to verify that you installed the AWS CLI.

```
$ aws --version
aws-cli/1.19.77 Python/3.6.15 Linux/5.14.16-201.fc34.x86_64 botocore/1.20.77
```

3. Configure the AWS command line client according to your AWS access details.

```
$ aws configure
AWS Access Key ID [None]:
AWS Secret Access Key [None]:
Default region name [None]:
Default output format [None]:
```

Additional resources

- Quickly Configuring the AWS CLI
- AWS command line tools

3.5.2. Creating an S3 bucket

Importing to AWS requires an Amazon S3 bucket. An Amazon S3 bucket is an Amazon resource where you store objects. As part of the process for uploading your image, you create an S3 bucket and then move your image to the bucket. Complete the following steps to create a bucket.

Procedure

1. Launch the Amazon S3 Console.
2. Click **Create Bucket**. The **Create Bucket** dialog appears.
3. In the **Name and region** view:
   a. Enter a **Bucket name**.
   b. Enter a **Region**.
   c. Click **Next**.

4. In the **Configure options** view, select the desired options and click **Next**.
5. In the **Set permissions** view, change or accept the default options and click **Next**.
6. Review your bucket configuration.
7. Click **Create bucket**.

NOTE

Alternatively, you can use the AWS CLI to create a bucket. For example, the **aws s3 mb s3://my-new-bucket** command creates an S3 bucket named **my-new-bucket**. See the AWS CLI Command Reference for more information on the **mb** command.

Additional resources

- Amazon S3 Console
- AWS CLI Command Reference

3.5.3. Creating the vmimport role

Perform the following procedure to create the **vmimport** role, which is required by VM import. See VM Import Service Role in the Amazon documentation for more information.

Procedure

1. Create a file named **trust-policy.json** and include the following policy. Save the file on your system and note its location.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": { "Service": "vmie.amazonaws.com" },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals":{
          "sts:Externalid": "vmimport"
        }
      }
    }
  ]
}
```

2. Use the **create role** command to create the **vmimport** role. Specify the full path to the location of the **trust-policy.json** file. Prefix **file://** to the path. For example:

```
aws iam create-role --role-name vmimport --assume-role-policy-document
file:///home/sample/ImportService/trust-policy.json
```

3. Create a file named **role-policy.json** and include the following policy. Replace **s3-bucket-name** with the name of your S3 bucket.

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
        "s3:GetBucketLocation",
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource":[
        "arn:aws:s3:::s3-bucket-name",
        "arn:aws:s3:::s3-bucket-name/*"
      ]
    },
    {
      "Effect":"Allow",
      "Action":[
        "ec2:ModifySnapshotAttribute",
        "ec2:CopySnapshot",
        "ec2:RegisterImage",
        "ec2:Describe*"
      ],
      "Resource":"*"
    }
  ]
}
```

4. Use the **put-role-policy** command to attach the policy to the role you created. Specify the full path of the **role-policy.json** file. For example:

```
aws iam put-role-policy --role-name vmimport --policy-name vmimport --policy-document
file:///home/sample/ImportService/role-policy.json
```

Additional resources

- VM Import Service Role
- Required Service Role

3.5.4. Converting and pushing your image to S3

Complete the following procedure to convert and push your image to S3. The samples are representative; they convert an image formatted in the **qcow2** file format to **raw** format. Amazon accepts images in **OVA**, **VHD**, **VHDX**, **VMDK**, and **raw** formats. See How VM Import/Export Works for more information on image formats that Amazon accepts.

Procedure

1. Run the **qemu-img** command to convert your image. For example:

```
qemu-img convert -f qcow2 -O raw rhel-8.0-sample.qcow2 rhel-8.0-sample.raw
```

2. Push the image to S3.

```
aws s3 cp rhel-8.0-sample.raw s3://s3-bucket-name
```

> **NOTE**
>
> This procedure could take a few minutes. After completion, you can check that your image uploaded successfully to your S3 bucket using the AWS S3 Console.

Additional resources

- How VM Import/Export Works
- AWS S3 Console

3.5.5. Importing your image as a snapshot

Perform the following procedure to import an image as a snapshot.

Procedure

1. Create a file to specify a bucket and path for your image. Name the file **containers.json**. In the sample that follows, replace **s3-bucket-name** with your bucket name and **s3-key** with your key. You can get the key for the image using the Amazon S3 Console.

```
{
   "Description": "rhel-8.0-sample.raw",
   "Format": "raw",
   "UserBucket": {
      "S3Bucket": "s3-bucket-name",
      "S3Key": "s3-key"
   }
}
```

2. Import the image as a snapshot. This example uses a public Amazon S3 file; you can use the Amazon S3 Console to change permissions settings on your bucket.

```
aws ec2 import-snapshot --disk-container file://containers.json
```

The terminal displays a message such as the following. Note the **ImportTaskID** within the message.

```
{
   "SnapshotTaskDetail": {
      "Status": "active",
      "Format": "RAW",
      "DiskImageSize": 0.0,
      "UserBucket": {
         "S3Bucket": "s3-bucket-name",
         "S3Key": "rhel-8.0-sample.raw"
      },
        "Progress": "3",
        "StatusMessage": "pending"
    },
    "ImportTaskId": "import-snap-06cea01fa0f1166a8"
}
```

3. Track the progress of the import using the **describe-import-snapshot-tasks** command. Include the **ImportTaskID**.

   ```
   aws ec2 describe-import-snapshot-tasks --import-task-ids import-snap-06cea01fa0f1166a8
   ```

   The returned message shows the current status of the task. When complete, **Status** shows **completed**. Within the status, note the snapshot ID.

Additional resources

- Amazon S3 Console
- Importing a Disk as a Snapshot Using VM Import/Export

3.5.6. Creating an AMI from the uploaded snapshot

Within EC2, you must choose an Amazon Machine Image (AMI) when launching an instance. Perform the following procedure to create an AMI from your uploaded snapshot.

Procedure

1. Go to the AWS EC2 Dashboard.
2. Under **Elastic Block Store**, select **Snapshots**.
3. Search for your snapshot ID (for example, **snap-0e718930bd72bcda0**).
4. Right-click on the snapshot and select **Create image**.
5. Name your image.
6. Under **Virtualization type**, choose **Hardware-assisted virtualization**.
7. Click **Create**. In the note regarding image creation, there is a link to your image.
8. Click on the image link. Your image shows up under **Images>AMIs**.

NOTE

Alternatively, you can use the AWS CLI **register-image** command to create an AMI from a snapshot. See register-image for more information. An example follows.

```
$ aws ec2 register-image --name "myimagename" --description
"myimagedescription" --architecture x86_64  --virtualization-type hvm --root-
device-name "/dev/sda1" --block-device-mappings "{\"DeviceName\":
\"/dev/sda1\",\"Ebs\": {\"SnapshotId\": \"snap-0ce7f009b69ab274d\"}}" --ena-
support
```

You must specify the root device volume **/dev/sda1** as your **root-device-name**. For conceptual information on device mapping for AWS, see Example block device mapping.

3.5.7. Launching an instance from the AMI

Perform the following procedure to launch and configure an instance from the AMI.

Procedure

1. From the AWS EC2 Dashboard, select **Images** and then **AMIs**.
2. Right-click on your image and select **Launch**.
3. Choose an **Instance Type** that meets or exceeds the requirements of your workload. See Amazon EC2 Instance Types for information on instance types.
4. Click **Next: Configure Instance Details**.
   a. Enter the **Number of instances** you want to create.
   b. For **Network**, select the VPC you created when setting up your AWS environment. Select a subnet for the instance or create a new subnet.
   c. Select **Enable** for Auto-assign Public IP.

   NOTE

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

5. Click **Next: Add Storage**. Verify that the default storage is sufficient.
6. Click **Next: Add Tags**.

   NOTE

   Tags can help you manage your AWS resources. See Tagging Your Amazon EC2 Resources for information on tagging.

7. Click **Next: Configure Security Group**. Select the security group you created when setting up your AWS environment.

8. Click **Review and Launch**. Verify your selections.
9. Click **Launch**. You are prompted to select an existing key pair or create a new key pair. Select the key pair you created when setting up your AWS environment.

    NOTE

    Verify that the permissions for your private key are correct. Use the command options **chmod 400 <keyname>.pem** to change the permissions, if necessary.

10. Click **Launch Instances**.
11. Click **View Instances**. You can name the instance(s).
    You can now launch an SSH session to your instance(s) by selecting an instance and clicking **Connect**. Use the example provided for **A standalone SSH client**.

    NOTE

    Alternatively, you can launch an instance using the AWS CLI. See Launching, Listing, and Terminating Amazon EC2 Instances in the Amazon documentation for more information.

Additional resources

- AWS Management Console
- Setting Up with Amazon EC2
- Amazon EC2 Instances
- Amazon EC2 Instance Types

3.5.8. Attaching Red Hat subscriptions

Complete the following steps to attach the subscriptions you previously enabled through the Red Hat Cloud Access program.

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 using the ID of the subscription pool (Pool ID). See Attaching and Removing Subscriptions Through the Command Line .

Additional resources

- Creating Red Hat Customer Portal Activation Keys
- Attaching and Removing Subscriptions Through the Command Line
- Using and Configuring Red Hat Subscription Manager

3.5.9. Setting up automatic registration on AWS Gold Images

To make deploying RHEL 8 virtual machines on Amazon Web Services (AWS) faster and more comfortable, you can set up Gold Images of RHEL 8 to be automatically registered to the Red Hat Subscription Manager (RHSM).

Prerequisites

- You have downloaded the latest RHEL 8 Gold Image for AWS. For instructions, see Using Gold Images on AWS.

NOTE

An AWS account can only be attached to a single Red Hat account at a time. Therefore, ensure no other users require access to the AWS account before attaching it to your Red Hat one.

Procedure

1. Upload the Gold Image to AWS. For instructions, see Uploading the Red Hat Enterprise Linux image to AWS.
2. Create VMs using the uploaded image. They will be automatically subscribed to RHSM.

Verification

- In a RHEL 8 VM created using the above instructions, verify the system is registered to RHSM by executing the **subscription-manager identity** command. On a successfully registered system, this displays the UUID of the system. For example:

```
# subscription-manager identity
system identity: fdc46662-c536-43fb-a18a-bbcb283102b7
name: 192.168.122.222
org name: 6340056
org ID: 6340056
```

Additional resources

- AWS Management Console
- Configuring cloud sources for Red Hat services

CHAPTER 4. CONFIGURING A RED HAT HIGH AVAILABILITY CLUSTER ON AWS

This chapter provides information and procedures for configuring a Red Hat High Availability (HA) cluster on Amazon Web Services (AWS) using EC2 instances as cluster nodes. Note that you have a number of options for obtaining the Red Hat Enterprise Linux (RHEL) images you use for your cluster. For information on image options for AWS, see Red Hat Enterprise Linux Image Options on AWS .

This chapter includes:

- Prerequisite procedures for setting up your environment for AWS. Once you have set up your environment, you can create and configure EC2 instances.
- Procedures specific to the creation of HA clusters, which transform individual nodes into a cluster of HA nodes on AWS. These include procedures for installing the High Availability packages and agents on each cluster node, configuring fencing, and installing AWS network resource agents.

Prerequisites

- Sign up for a Red Hat Customer Portal account.
- Sign up for AWS and set up your AWS resources. See Setting Up with Amazon EC2 for more information.
- Enable your subscriptions in the Red Hat Cloud Access program . The Red Hat Cloud Access program allows you to move your Red Hat subscriptions from physical or on-premise systems onto AWS with full support from Red Hat.

4.1. ADDITIONAL RESOURCES

- Red Hat Cloud Access Reference Guide
- Red Hat in the Public Cloud
- Red Hat Enterprise Linux on Amazon EC2 - FAQs
- Setting Up with Amazon EC2
- Red Hat on Amazon Web Services

4.2. CREATING THE AWS ACCESS KEY AND AWS SECRET ACCESS KEY

You need to create an AWS Access Key and AWS Secret Access Key before you install the AWS CLI. The fencing and resource agent APIs use the AWS Access Key and Secret Access Key to connect to each node in the cluster.

Complete the following steps to create these keys.

Prerequisites

- Your IAM user account must have Programmatic access. See Setting up the AWS Environment for more information.

Procedure

1. Launch the AWS Console.
2. Click on your AWS Account ID to display the drop-down menu and select **My Security Credentials**.
3. Click **Users**.
4. Select the user and open the **Summary** screen.
5. Click the **Security credentials** tab.
6. Click **Create access key**.
7. Download the **.csv** file (or save both keys). You need to enter these keys when creating the fencing device.

4.3. INSTALLING THE AWS CLI

Many of the procedures required to manage HA clusters in AWS include using the AWS CLI. Complete the following steps to install the AWS CLI.

Prerequisites

- You have created an AWS Access Key ID and an AWS Secret Access Key, and have access to them. For instructions and details, see Quickly Configuring the AWS CLI.

Procedure

1. Install the AWS command line tools using the **yum** command.

```
# yum install awscli
```

2. Use the **aws --version** command to verify that you installed the AWS CLI.

```
$ aws --version
aws-cli/1.19.77 Python/3.6.15 Linux/5.14.16-201.fc34.x86_64 botocore/1.20.77
```

3. Configure the AWS command line client according to your AWS access details.

```
$ aws configure
AWS Access Key ID [None]:
AWS Secret Access Key [None]:
Default region name [None]:
Default output format [None]:
```

Additional resources

- Quickly Configuring the AWS CLI
- AWS command line tools

4.4. CREATING AN HA EC2 INSTANCE

Complete the following steps to create the instances that you use as your HA cluster nodes. Note that you have a number of options for obtaining the RHEL images you use for your cluster. See Red Hat Enterprise Linux Image options on AWS for information on image options for AWS.

You can create and upload a custom image that you use for your cluster nodes, or you could choose a Gold Image (Cloud Access image) or an on-demand image.

Prerequisites

- You need to have set up an AWS environment. See Setting Up with Amazon EC2 for more information.

Procedure

1. From the AWS EC2 Dashboard, select **Images** and then **AMIs**.
2. Right-click on your image and select **Launch**.
3. Choose an **Instance Type** that meets or exceeds the requirements of your workload. Depending on your HA application, each instance may need to have higher capacity.
   See Amazon EC2 Instance Types for information on instance types.
4. Click **Next: Configure Instance Details**.
   a. Enter the **Number of instances** you want to create for the cluster. The examples in this chapter use three cluster nodes.

      NOTE

      Do not launch into an Auto Scaling Group.

   b. For **Network**, select the VPC you created in Set up the AWS environment. Select the subnet for the instance to create a new subnet.
   c. Select **Enable** for Auto-assign Public IP. These are the minimum selections you need to make for **Configure Instance Details**. Depending on your specific HA application, you may need to make additional selections.

      NOTE

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

5. Click **Next: Add Storage** and verify that the default storage is sufficient. You do not need to modify these settings unless your HA application requires other storage options.
6. Click **Next: Add Tags**.

   NOTE

   Tags can help you manage your AWS resources. See Tagging Your Amazon EC2 Resources for information on tagging.

7. Click **Next: Configure Security Group**. Select the existing security group you created in Setting up the AWS environment.

8. Click **Review and Launch** and verify your selections.

9. Click **Launch**. You are prompted to select an existing key pair or create a new key pair. Select the key pair you created when Setting up the AWS environment.

10. Click **Launch Instances**.

11. Click **View Instances**. You can name the instance(s).

NOTE

Alternatively, you can launch instances using the AWS CLI. See Launching, Listing, and Terminating Amazon EC2 Instances in the Amazon documentation for more information.

Additional resources

- AWS Management Console
- Setting Up with Amazon EC2
- Amazon EC2 Instances
- Amazon EC2 Instance Types

4.5. CONFIGURING THE PRIVATE KEY

Complete the following configuration tasks to use the private SSH key file (**.pem**) before it can be used in an SSH session.

Procedure

1. Move the key file from the **Downloads** directory to your **Home** directory or to your **~/.ssh directory**.

2. Enter the following command to change the permissions of the key file so that only the root user can read it.

```
# chmod 400 KeyName.pem
```

4.6. CONNECTING TO AN EC2 INSTANCE

Complete the following steps on all nodes to connect to an EC2 instance.

Procedure

1. Launch the AWS Console and select the EC2 instance.

2. Click **Connect** and select **A standalone SSH client**.

3. From your SSH terminal session, connect to the instance using the AWS example provided in the pop-up window. Add the correct path to your **KeyName.pem** file if the path is not shown in the example.

4.7. INSTALLING THE HIGH AVAILABILITY PACKAGES AND AGENTS

Complete the following steps on all nodes to install the High Availability packages and agents.

Procedure

1. Enter the following command to remove the AWS Red Hat Update Infrastructure (RHUI) client. Because you are going to use a Red Hat Cloud Access subscription, you should not use AWS RHUI in addition to your subscription.

```
$ sudo -i
# yum -y remove rh-amazon-rhui-client*
```

2. Register the VM with Red Hat.

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

3. Disable all repositories.

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

4. Enable the RHEL 8 Server HA repositories.

```
# subscription-manager repos --enable=rhel-8-for-x86_64-highavailability-rpms
```

5. Update the RHEL AWS instance.

```
# yum update -y
```

6. Install the Red Hat High Availability Add-On software packages, along with all available fencing agents from the High Availability channel.

```
# yum install pcs pacemaker fence-agents-aws
```

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

```
# passwd hacluster
```

8. Add the **high availability** service to the RHEL Firewall if **firewalld.service** is installed.

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

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

```
# systemctl start pcsd.service
# systemctl enable pcsd.service
```

10. Edit **/etc/hosts** and add RHEL host names and internal IP addresses. See How should the /etc/hosts file be set up on RHEL cluster nodes? for details.

Verification

- Ensure the **pcs** 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 Thu 2018-03-01 14:53:28 UTC; 28min ago
Docs: man:pcsd(8)
man:pcs(8)
Main PID: 5437 (pcsd)
CGroup: /system.slice/pcsd.service
    └─5437 /usr/bin/ruby /usr/lib/pcsd/pcsd > /dev/null &
Mar 01 14:53:27 ip-10-0-0-48.ec2.internal systemd[1]: Starting PCS GUI and remote
configuration interface…
Mar 01 14:53:28 ip-10-0-0-48.ec2.internal systemd[1]: Started PCS GUI and remote
configuration interface.
```

4.8. CREATING A CLUSTER

Complete the following steps to create the cluster of nodes.

Procedure

1. On one of the nodes, enter the following command to authenticate the pcs user **hacluster**. In the command, specify the name of each node in the cluster.

```
# pcs host auth <hostname1> <hostname2> <hostname3>
```

Example:

```
[root@node01 clouduser]# pcs host auth node01 node02 node03
Username: hacluster
Password:
node01: Authorized
node02: Authorized
node03: Authorized
```

2. Create the cluster.

```
# pcs cluster setup <cluster_name> <hostname1> <hostname2> <hostname3>
```

Example:

```
[root@node01 clouduser]# pcs cluster setup new_cluster node01 node02 node03

...omitted

Synchronizing pcsd certificates on nodes node01, node02, node03...
```

```
node02: Success
node03: Success
node01: Success
Restarting pcsd on the nodes in order to reload the certificates...
node02: Success
node03: Success
node01: Success
```

Verification

1. Enable the cluster.

```
[root@node01 clouduser]# pcs cluster enable --all
node02: Cluster Enabled
node03: Cluster Enabled
node01: Cluster Enabled
```

2. Start the cluster.

```
[root@node01 clouduser]# pcs cluster start --all
node02: Starting Cluster...
node03: Starting Cluster...
node01: Starting Cluster...
```

4.9. CONFIGURING FENCING

Fencing configuration ensures that a malfunctioning node on your AWS cluster is automatically isolated, which prevents the node from consuming the cluster's resources or compromising the cluster's functionality.

You can configure fencing on AWS cluster using multiple methods. This section provides the following:

- A standard procedure for default configuration.
- An alternate configuration procedure for more advanced configuration, focused on automation.

Standard procedure

1. Enter the following AWS metadata query to get the Instance ID for each node. You need these IDs to configure the fence device. See Instance Metadata and User Data for additional information.

```
# echo $(curl -s http://169.254.169.254/latest/meta-data/instance-id)
```

Example:

```
[root@ip-10-0-0-48 ~]# echo $(curl -s http://169.254.169.254/latest/meta-data/instance-
id) i-07f1ac63af0ec0ac6
```

2. Enter the following command to configure the fence device. Use the **pcmk_host_map** command to map the RHEL host name to the Instance ID. Use the AWS Access Key and AWS Secret Access Key that you previously set up.

```
# pcs stonith create name fence_aws access_key=access-key secret_key=secret-access-
key region=region pcmk_host_map="rhel-hostname-1:Instance-ID-1;rhel-hostname-
2:Instance-ID-2;rhel-hostname-3:Instance-ID-3" power_timeout=240
pcmk_reboot_timeout=480 pcmk_reboot_retries=4
```

Example:

```
[root@ip-10-0-0-48 ~]# pcs stonith create clusterfence fence_aws
access_key=AKIAI*******6MRMJA secret_key=a75EYIG4RVL3h*******K7koQ8dzaDyn5yoIZ/
region=us-east-1 pcmk_host_map="ip-10-0-0-48:i-07f1ac63af0ec0ac6;ip-10-0-0-46:i-
063fc5fe93b4167b2;ip-10-0-0-58:i-08bd39eb03a6fd2c7" power_timeout=240
pcmk_reboot_timeout=480 pcmk_reboot_retries=4
```

Alternate procedure

1. Obtain the VPC ID of the cluster.

```
# aws ec2 describe-vpcs --output text --filters "Name=tag:Name,Values=clustername-
vpc" --query 'Vpcs[*].VpcId'
vpc-06bc10ac8f6006664
```

2. Using the VPC ID of the cluster, obtain the VPC instances.

```
$ aws ec2 describe-instances --output text --filters "Name=vpc-id,Values=vpc-
06bc10ac8f6006664" --query 'Reservations[*].Instances[*].{Name:Tags[? Key==Name]|
[0].Value,Instance:InstanceId}' | grep "\-node[a-c]"
i-0b02af8927a895137     clustername-nodea-vm
i-0cceb4ba8ab743b69     clustername-nodeb-vm
i-0502291ab38c762a5     clustername-nodec-vm
```

3. Use the obtained instance IDs to configure fencing on each node on the cluster. For example:

```
[root@nodea ~]# CLUSTER=clustername && pcs stonith create fence${CLUSTER}
fence_aws access_key=XXXXXXXXXXXXXXXXXXXX pcmk_host_map=$(for NODE \
in node{a..c}; do ssh ${NODE} "echo -n \${HOSTNAME}:\$(curl -s
http://169.254.169.254/latest/meta-data/instance-id)\;"; done) \
pcmk_reboot_retries=4 pcmk_reboot_timeout=480 power_timeout=240 region=xx-xxxx-x
secret_key=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

[root@nodea ~]# pcs stonith config fence${CLUSTER}
Resource: clustername (class=stonith type=fence_aws)
Attributes: access_key=XXXXXXXXXXXXXXXXXXXX pcmk_host_map=nodea:i-
0b02af8927a895137;nodeb:i-0cceb4ba8ab743b69;nodec:i-0502291ab38c762a5;
pcmk_reboot_retries=4 pcmk_reboot_timeout=480 power_timeout=240 region=xx-xxxx-x
secret_key=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Operations: monitor interval=60s (clustername-monitor-interval-60s)
```

Verification

1. Test the fencing agent for one of the cluster nodes.

```
# pcs stonith fence awsnodename
```

NOTE

The command response may take several minutes to display. If you watch the active terminal session for the node being fenced, you see that the terminal connection is immediately terminated after you enter the fence command.

Example:

```
[root@ip-10-0-0-48 ~]# pcs stonith fence ip-10-0-0-58
Node: ip-10-0-0-58 fenced
```

2. Check the status to verify that the node is fenced.

```
# pcs status
```

Example:

```
[root@ip-10-0-0-48 ~]# pcs status
Cluster name: newcluster
Stack: corosync
Current DC: ip-10-0-0-46 (version 1.1.18-11.el7-2b07d5c5a9) - partition with quorum
Last updated: Fri Mar  2 19:55:41 2018
Last change: Fri Mar  2 19:24:59 2018 by root via cibadmin on ip-10-0-0-46

3 nodes configured
1 resource configured

Online: [ ip-10-0-0-46 ip-10-0-0-48 ]
OFFLINE: [ ip-10-0-0-58 ]

Full list of resources:
clusterfence  (stonith:fence_aws):    Started ip-10-0-0-46

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

3. Start the node that was fenced in the previous step.

```
# pcs cluster start awshostname
```

4. Check the status to verify the node started.

```
# pcs status
```

Example:

```
[root@ip-10-0-0-48 ~]# pcs status
Cluster name: newcluster
Stack: corosync
Current DC: ip-10-0-0-46 (version 1.1.18-11.el7-2b07d5c5a9) - partition with quorum
Last updated: Fri Mar  2 20:01:31 2018
Last change: Fri Mar  2 19:24:59 2018 by root via cibadmin on ip-10-0-0-48

3 nodes configured
1 resource configured

Online: [ ip-10-0-0-46 ip-10-0-0-48 ip-10-0-0-58 ]

Full list of resources:

  clusterfence  (stonith:fence_aws):    Started ip-10-0-0-46

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

4.10. INSTALLING THE AWS CLI ON CLUSTER NODES

Previously, you installed the AWS CLI on your host system. You need to install the AWS CLI on cluster nodes before you configure the network resource agents.

Complete the following procedure on each cluster node.

Prerequisites

- You must have created an AWS Access Key and AWS Secret Access Key. See Creating the AWS Access Key and AWS Secret Access Key for more information.

Procedure

1. Install the AWS CLI. For instructions, see Installing the AWS CLI.

2. Enter the following command to verify that the AWS CLI is configured properly. The instance IDs and instance names should display.
   Example:

   ```
   [root@ip-10-0-0-48 ~]# aws ec2 describe-instances --output text --query
   'Reservations[].Instances[].[InstanceId,Tags[?Key==Name].Value]'
   i-07f1ac63af0ec0ac6
   ip-10-0-0-48
   i-063fc5fe93b4167b2
   ip-10-0-0-46
   i-08bd39eb03a6fd2c7
   ip-10-0-0-58
   ```

4.11. INSTALLING NETWORK RESOURCE AGENTS

For HA operations to work, the cluster uses AWS networking resource agents to enable failover functionality. If a node does not respond to a heartbeat check in a set amount of time, the node is fenced and operations fail over to an additional node in the cluster. Network resource agents need to be configured for this to work.

Add the two resources to the same group to enforce **order** and **colocation** constraints.

Create a secondary private IP resource and virtual IP resource

Complete the following procedure to add a secondary private IP address and create a virtual IP. You can complete this procedure from any node in the cluster.

Procedure

1. Enter the following command to view the **AWS Secondary Private IP Address** resource agent (awsvip) description. This shows the options and default operations for this agent.

   ```
   # pcs resource describe awsvip
   ```

2. Enter the following command to create the Secondary Private IP address using an unused private IP address in the **VPC CIDR** block.

   ```
   # pcs resource create privip awsvip secondary_private_ip=Unused-IP-Address --group group-name
   ```

   Example:

   ```
   [root@ip-10-0-0-48 ~]# pcs resource create privip awsvip secondary_private_ip=10.0.0.68 --group networking-group
   ```

3. Create a virtual IP resource. This is a VPC IP address that can be rapidly remapped from the fenced node to the failover node, masking the failure of the fenced node within the subnet.

   ```
   # pcs resource create vip IPaddr2 ip=secondary-private-IP --group group-name
   ```

   Example:

   ```
   root@ip-10-0-0-48 ~]# pcs resource create vip IPaddr2 ip=10.0.0.68 --group networking-group
   ```

Verification

- Enter the **pcs status** command to verify that the resources are running.

  ```
  # pcs status
  ```

  Example:

  ```
  [root@ip-10-0-0-48 ~]# pcs status
  Cluster name: newcluster
  Stack: corosync
  Current DC: ip-10-0-0-46 (version 1.1.18-11.el7-2b07d5c5a9) - partition with quorum
  Last updated: Fri Mar  2 22:34:24 2018
  Last change: Fri Mar  2 22:14:58 2018 by root via cibadmin on ip-10-0-0-46

  3 nodes configured
  3 resources configured

  Online: [ ip-10-0-0-46 ip-10-0-0-48 ip-10-0-0-58 ]
  ```

```
Full list of resources:

clusterfence    (stonith:fence_aws):    Started ip-10-0-0-46
 Resource Group: networking-group
     privip (ocf::heartbeat:awsvip):    Started ip-10-0-0-48
     vip    (ocf::heartbeat:IPaddr2):   Started ip-10-0-0-58

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

Create an elastic IP address

An elastic IP address is a public IP address that can be rapidly remapped from the fenced node to the failover node, masking the failure of the fenced node.

Note that this is different from the virtual IP resource created earlier. The elastic IP address is used for public-facing Internet connections instead of subnet connections.

1. Add the two resources to the same group that was previously created to enforce **order** and **colocation** constraints.

2. Enter the following AWS CLI command to create an elastic IP address.

```
[root@ip-10-0-0-48 ~]# aws ec2 allocate-address --domain vpc --output text
eipalloc-4c4a2c45   vpc 35.169.153.122
```

3. Enter the following command to view the AWS Secondary Elastic IP Address resource agent (awseip) description. This shows the options and default operations for this agent.

```
# pcs resource describe awseip
```

4. Create the Secondary Elastic IP address resource using the allocated IP address created in Step 1.

```
# pcs resource create elastic awseip elastic_ip=Elastic-IP-Address
allocation_id=Elastic-IP-Association-ID --group networking-group
```

Example:

```
# pcs resource create elastic awseip elastic_ip=35.169.153.122 allocation_id=eipalloc-
4c4a2c45 --group networking-group
```

Verification

- Enter the **pcs status** command to verify that the resource is running.

```
# pcs status
```

Example:

```
[root@ip-10-0-0-58 ~]# pcs status
Cluster name: newcluster
```

```
Stack: corosync
Current DC: ip-10-0-0-58 (version 1.1.18-11.el7-2b07d5c5a9) - partition with quorum
Last updated: Mon Mar  5 16:27:55 2018
Last change: Mon Mar  5 15:57:51 2018 by root via cibadmin on ip-10-0-0-46

3 nodes configured
4 resources configured

Online: [ ip-10-0-0-46 ip-10-0-0-48 ip-10-0-0-58 ]

Full list of resources:

 clusterfence   (stonith:fence_aws):    Started ip-10-0-0-46
 Resource Group: networking-group
     privip (ocf::heartbeat:awsvip):  Started ip-10-0-0-48
     vip    (ocf::heartbeat:IPaddr2):    Started ip-10-0-0-48
     elastic (ocf::heartbeat:awseip):    Started ip-10-0-0-48

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

Test the elastic IP address

Enter the following commands to verify the virtual IP (awsvip) and elastic IP (awseip) resources are working.

Procedure

1. Launch an SSH session from your local workstation to the elastic IP address previously created.

   ```
   $ ssh -l ec2-user -i ~/.ssh/<KeyName>.pem elastic-IP
   ```

   Example:

   ```
   $ ssh -l ec2-user -i ~/.ssh/cluster-admin.pem 35.169.153.122
   ```

2. Verify that the host you connected to via SSH is the host associated with the elastic resource created.

Additional resources

- High Availability Add-On overview
- Configuring and managing high availability clusters

4.12. CONFIGURING SHARED BLOCK STORAGE

This section provides an optional procedure for configuring shared block storage for a Red Hat High Availability cluster with Amazon Elastic Block Storage (EBS) Multi-Attach volumes. The procedure assumes three instances (a three-node cluster) with a 1 TB shared disk.

Prerequisites

- You must be using an AWS Nitro System-based Amazon EC2 instance .

Procedure

1. Create a shared block volume using the AWS command create-volume.

```
$ aws ec2 create-volume --availability-zone <availability_zone> --no-encrypted --size
1024 --volume-type io1 --iops 51200 --multi-attach-enabled
```

For example, the following command creates a volume in the **us-east-1a** availability zone.

```
$ aws ec2 create-volume --availability-zone us-east-1a --no-encrypted --size 1024 --
volume-type io1 --iops 51200 --multi-attach-enabled

{
    "AvailabilityZone": "us-east-1a",
    "CreateTime": "2020-08-27T19:16:42.000Z",
    "Encrypted": false,
    "Size": 1024,
    "SnapshotId": "",
    "State": "creating",
    "VolumeId": "vol-042a5652867304f09",
    "Iops": 51200,
    "Tags": [ ],
    "VolumeType": "io1"
}
```

NOTE

You need the **VolumeId** in the next step.

2. For each instance in your cluster, attach a shared block volume using the AWS command attach-volume. Use your **<instance_id>** and **<volume_id>**.

```
$ aws ec2 attach-volume --device /dev/xvdd --instance-id <instance_id> --volume-id
<volume_id>
```

For example, the following command attaches a shared block volume **vol-042a5652867304f09** to **instance i-0eb803361c2c887f2**.

```
$ aws ec2 attach-volume --device /dev/xvdd --instance-id i-0eb803361c2c887f2 --
volume-id vol-042a5652867304f09

{
    "AttachTime": "2020-08-27T19:26:16.086Z",
    "Device": "/dev/xvdd",
    "InstanceId": "i-0eb803361c2c887f2",
    "State": "attaching",
    "VolumeId": "vol-042a5652867304f09"
}
```

Verification

1. For each instance in your cluster, verify that the block device is available by using the **ssh** command with your instance **<ip_address>**.

   ```
   # ssh <ip_address> "hostname ; lsblk -d | grep ' 1T '"
   ```

   For example, the following command lists details including the host name and block device for the instance IP **198.51.100.3**.

   ```
   # ssh 198.51.100.3 "hostname ; lsblk -d | grep ' 1T '"

   nodea
   nvme2n1 259:1    0   1T  0 disk
   ```

2. Use the **ssh** command to verify that each instance in your cluster uses the same shared disk.

   ```
   # ssh <ip_address> "hostname ; lsblk -d | grep ' 1T ' | awk '{print \$1}' | xargs -i
   udevadm info --query=all --name=/dev/{} | grep '^E: ID_SERIAL='"
   ```

   For example, the following command lists details including the host name and shared disk volume ID for the instance IP address **198.51.100.3**.

   ```
   # ssh 198.51.100.3 "hostname ; lsblk -d | grep ' 1T ' | awk '{print \$1}' | xargs -i udevadm
   info --query=all --name=/dev/{} | grep '^E: ID_SERIAL='"

   nodea
   E: ID_SERIAL=Amazon Elastic Block Store_vol0fa5342e7aedf09f7
   ```

Additional resources

- Configuring a GFS2 file system in a cluster
- Configuring GFS2 file systems

CHAPTER 5. DEPLOYING A RED HAT ENTERPRISE LINUX IMAGE AS A GOOGLE COMPUTE ENGINE INSTANCE ON GOOGLE CLOUD PLATFORM

To deploy a Red Hat Enterprise Linux 8 (RHEL 8) as a Google Compute Engine (GCE) instance on Google Cloud Platform (GCP), follow the information below. This chapter:

- Discusses your options for choosing an image
- Lists or refers to system requirements for your host system and virtual machine (VM)
- Provides procedures for creating a custom VM from an ISO image, uploading it to GCE, and launching an instance

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.
- Enable your Red Hat subscriptions through the Red Hat Cloud Access program. The Red Hat Cloud Access program allows you to move your Red Hat subscriptions from physical or on-premise systems onto GCP with full support from Red Hat.

5.1. ADDITIONAL RESOURCES

- Red Hat in the Public Cloud
- Google Cloud

5.2. RED HAT ENTERPRISE LINUX IMAGE OPTIONS ON GCP

The following table lists image choices for RHEL 8 on Google Cloud Platform and the differences in the image options.

Table 5.1. Image options

Image option	Subscriptions	Sample scenario	Considerations
Choose to deploy a Red Hat Gold Image.	Use your existing Red Hat subscriptions.	Enable subscriptions through the Red Hat Cloud Access program, and then choose a Red Hat Gold Image on Google Cloud Platform. See the Red Hat Cloud Access Reference Guide for details on Gold Images and how to access them on Google Cloud Platform.	The subscription includes the Red Hat product cost; you pay Google for all other instance costs. Red Hat Gold Images are called "Cloud Access" images because you use your existing Red Hat subscriptions. Red Hat provides support directly for Cloud Access images.
Choose to deploy a custom image that you move to GCP.	Use your existing Red Hat subscriptions.	Enable subscriptions through the Red Hat Cloud Access program, upload your custom image, and attach your subscriptions.	The subscription includes the Red Hat product cost; you pay all other instance costs. Custom images that you move to GCP are called "Cloud Access" images because you use your existing Red Hat subscriptions. Red Hat provides support directly for Cloud Access images.
Choose to 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.

IMPORTANT

You cannot convert an on-demand instance to a Red Hat Cloud Access instance. To change from an on-demand image to a Red Hat Cloud Access bring-your-own-subscription (BYOS) image, create a new Red Hat Cloud Access instance and migrate data from your on-demand instance. Cancel your on-demand instance after you migrate your data to avoid double billing.

The remainder of this chapter includes information and procedures pertaining to custom images.

Additional resources

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

5.3. UNDERSTANDING BASE IMAGES

This section includes information on using preconfigured base images and their configuration settings.

5.3.1. 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

5.3.2. Virtual machine configuration settings

Cloud VMs must have the following configuration settings.

Table 5.2. 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.

5.4. CREATING A BASE VM FROM AN ISO IMAGE

Follow the procedures in this section to create a RHEL 8 base image from an ISO image.

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**.

5.4.1. 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 8 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**.
   A basic command-line sample follows.

   ```
   virt-install --name kvmtest --memory 2048 --vcpus 2 --disk rhel-8.0-x86_64-
   kvm.qcow2,bus=virtio --import --os-variant=rhel8.0
   ```

   b. If you use the web console to create your VM, follow the procedure in Creating virtual machines 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.

5.4.2. Completing the RHEL installation

Perform the following steps to complete the installation and to 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 Management (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 8 repository.

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

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

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

7. Power down the VM.

Additional resources

- Understanding autoattaching subscriptions on the Customer Portal
- Introduction to cloud-init

5.5. UPLOADING THE RHEL IMAGE TO GCP

To upload your RHEL 8 image to Google Cloud Platform (GCP), follow the procedures in this section.

5.5.1. Creating a new project on GCP

Complete the following steps to create a new project on Google Cloud Platform (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. A simple example follows.

```
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

5.5.2. Installing the Google Cloud SDK

Complete the following steps to install 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

5.5.3. Creating SSH keys for Google Compute Engine

Perform the following procedure to generate and register SSH keys with GCE so that you can SSH directly into an instance 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 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 on the **gcloud compute config-ssh** command, see gcloud compute config-ssh.

Additional resources

- gcloud compute config-ssh
- Connecting to instances

5.5.4. Creating a storage bucket in GCP Storage

Importing to GCP requires a GCP Storage Bucket. Complete the following steps to create a 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

5.5.5. Converting and uploading your image to your GCP Bucket

Complete the following procedure to convert and upload your image to your GCP Bucket. The samples are representative; they convert an **qcow2** image to **raw** format and then tar that image for upload.

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-{ProductNumber}.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 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

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

Perform the following procedure to create an image from the object in your GCP bucket.

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. Optionally, 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

5.5.7. Creating a Google Compute Engine instance from an image

Complete the following steps to configure a GCE VM instance using the GCP Console.

NOTE

The following procedure provides instructions for creating a basic VM instance using the GCP Console. See Creating and starting a VM instance for more information on 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. Optionally, 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.

5.5.8. Connecting to your instance

Perform the following procedure to connect to your GCE instance using its public IP address.

Procedure

1. Run the following command to ensure that your instance is running. The 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 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

5.5.9. Attaching Red Hat subscriptions

Complete the following steps to attach the subscriptions you previously enabled through the Red Hat Cloud Access program.

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 using the ID of the subscription pool (Pool ID). See Attaching and Removing Subscriptions Through the Command Line .

Additional resources

- Creating Red Hat Customer Portal Activation Keys
- Attaching and Removing Subscriptions Through the Command Line
- Using and Configuring Red Hat Subscription Manager

CHAPTER 6. CONFIGURING RED HAT HIGH AVAILABILITY CLUSTER ON GOOGLE CLOUD PLATFORM

This chapter provides information and procedures for configuring a Red Hat High Availability (HA) cluster on Google Cloud Platform (GCP) using Google Compute Engine (GCE) virtual machine (VM) instances as cluster nodes.

This chapter includes:

- 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

- You must be enrolled in the Red Hat Cloud Access program and have unused RHEL subscriptions. The attached subscription must include access to the following repositories for each GCP instance.
  - Red Hat Enterprise Linux 8 Server: rhel-8-server-rpms/8Server/x86_64
  - Red Hat Enterprise Linux 8 Server (High Availability): rhel-8-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

6.1. 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

6.2. REQUIRED SYSTEM PACKAGES

The procedures in this chapter assume you are using a host system running Red Hat Enterprise Linux. To successfully complete the procedures, your host system must have the following packages installed.

Table 6.1. System packages

Package	Repository	Description
libvirt	rhel-8-for-x86_64-appstream-rpms	Open source API, daemon, and management tool for managing platform virtualization
virt-install	rhel-8-for-x86_64-appstream-rpms	A command-line utility for building VMs
libguestfs	rhel-8-for-x86_64-appstream-rpms	A library for accessing and modifying VM file systems
libguestfs-tools	rhel-8-for-x86_64-appstream-rpms	System administration tools for VMs; includes the **guestfish** utility

6.3. RED HAT ENTERPRISE LINUX IMAGE OPTIONS ON GCP

The following table lists image choices for RHEL 8 on Google Cloud Platform and the differences in the image options.

Table 6.2. Image options

Image option	Subscriptions	Sample scenario	Considerations

Image option	Subscriptions	Sample scenario	Considerations
Choose to deploy a Red Hat Gold Image.	Use your existing Red Hat subscriptions.	Enable subscriptions through the Red Hat Cloud Access program, and then choose a Red Hat Gold Image on Google Cloud Platform. See the Red Hat Cloud Access Reference Guide for details on Gold Images and how to access them on Google Cloud Platform.	The subscription includes the Red Hat product cost; you pay Google for all other instance costs. Red Hat Gold Images are called "Cloud Access" images because you use your existing Red Hat subscriptions. Red Hat provides support directly for Cloud Access images.
Choose to deploy a custom image that you move to GCP.	Use your existing Red Hat subscriptions.	Enable subscriptions through the Red Hat Cloud Access program, upload your custom image, and attach your subscriptions.	The subscription includes the Red Hat product cost; you pay all other instance costs. Custom images that you move to GCP are called "Cloud Access" images because you use your existing Red Hat subscriptions. Red Hat provides support directly for Cloud Access images.
Choose to 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.

IMPORTANT

You cannot convert an on-demand instance to a Red Hat Cloud Access instance. To change from an on-demand image to a Red Hat Cloud Access bring-your-own-subscription (BYOS) image, create a new Red Hat Cloud Access instance and migrate data from your on-demand instance. Cancel your on-demand instance after you migrate your data to avoid double billing.

The remainder of this chapter includes information and procedures pertaining to custom images.

Additional resources

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

6.4. INSTALLING THE GOOGLE CLOUD SDK

Complete the following steps to install 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

6.5. 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

6.6. CREATING A CUSTOM VIRTUAL PRIVATE CLOUD NETWORK AND SUBNET

Complete the following steps to create a custom virtual private cloud (VPC) network and subnet.

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**.

6.7. PREPARING AND IMPORTING A BASE GCP IMAGE

Complete the following steps to prepare a Red Hat Enterprise Linux 8 image for GCP.

Procedure

1. Enter the following command to 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. Enter the following command to 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
   ```

6.8. CREATING AND CONFIGURING A BASE GCP INSTANCE

Complete the following steps to create and configure a GCP instance that complies with GCP operating and security requirements.

Procedure

1. Enter the following command to 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. Enter the following command to 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-8-server-rpms
      ```

   e. Run the **yum update** command.

```
# yum 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. Enter the following command to 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. Enter the following command to disable password access. Edit the **/etc/cloud/cloud.cfg** file.

   ```
   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. Enter the following command to unregister the instance from the subscription manager.

    ```
    # subscription-manager unregister
    ```

11. Enter the following command to clean the shell history. Keep the instance running for the next procedure.

    ```
    # export HISTSIZE=0
    ```

6.9. CREATING A SNAPSHOT IMAGE

Complete the following steps to preserve the instance configuration settings and create a snapshot.

Procedure

1. On the running instance, enter the following command to synchronize data to disk.

   ```
   # sync
   ```

2. On your host system, enter the following command to create the snapshot.

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

3. On your host system, enter the following command to create the configured image from the snapshot.

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

Additional resources

- Creating Persistent Disk Snapshots

6.10. CREATING AN HA NODE TEMPLATE INSTANCE AND HA NODES

Once you have configured an image from the snapshot, you can create a node template. Use this template to create all HA nodes. Complete the following steps to create the template and HA nodes.

Procedure

1. Enter the following command to 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-81-instance-
template --can-ip-forward --machine-type n1-standard-2 --image rhel-81-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-81-instance-template].
NAME  MACHINE_TYPE   PREEMPTIBLE  CREATION_TIMESTAMP
rhel-81-instance-template   n1-standard-2         2018-07-25T11:09:30.506-07:00
```

2. Enter the following command to 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-81-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
```

6.11. INSTALLING HA PACKAGES AND AGENTS

Complete the following steps on all nodes.

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-8-server-rpms
   # subscription-manager repos --enable=rhel-8-for-x86_64-highavailability-rpms
   ```

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

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

9. Update all packages.

   ```
   # yum update -y
   ```

6.12. CONFIGURING HA SERVICES

Complete the following steps on all nodes to configure 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, enter the following command to add the HA service.

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

   # firewall-cmd --reload
   ```

3. Enter the following command to 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?

6.13. CREATING A CLUSTER

Complete the following steps to create the cluster of nodes.

Procedure

1. On one of the nodes, enter the following command to 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. Enter the following command to 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. Enter the following command to start the cluster.

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

6.14. CREATING A FENCING DEVICE

Complete the following steps to create a fencing device.

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

Procedure

1. Enter the following command to get GCP instance names. Note that the output 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
   44358**********3181,InstanceName-3
   40819**********6811,InstanceName-1
   71736**********3341,InstanceName-2
   ```

2. Enter the following command to create a fence device.

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

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
  ```

6.15. CONFIGURING GCP NODE AUTHORIZATION

Configure cloud SDK tools to use your account credentials to access GCP.

Procedure

Enter the following command on each node to initialize each node with your project ID and account credentials.

```
# gcloud-ra init
```

6.16. 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.

Enter the following command 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. This section includes procedures for both ranges.

Primary subnet address range

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

Procedure

1. Enter the following command to 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. Enter the following command to 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. Enter the following command to group the network resources under **vipgrp**.

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

Verification

1. Enter the following command to verify that the resources have started and are grouped under **vipgrp**.

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

2. Enter the following command to 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. Enter the following command to verify that the **vip** successfully started on a different node.

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

Secondary subnet address range

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

Prerequisites

- Create a custom network and subnet

Procedure

1. Enter the following command to create a secondary subnet address range.

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

   Example:

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

2. Enter the following command to 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. Enter the following command to 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 steps

1. Enter the following command to verify that the resources have started and are grouped under **vipgrp**.

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

2. Enter the following command to 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. Enter the following command to verify that the **vip** successfully started on a different node.

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