Red Hat Enterprise Linux 7

Deploying Red Hat Enterprise Linux 7 on public cloud platforms

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

Last Updated: 2023-04-04

Red Hat Enterprise Linux 7 Deploying Red Hat Enterprise Linux 7 on public cloud platforms

Creating custom Red Hat Enterprise Linux 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 describes two choices for creating images: Cloud Access images and on-demand (marketplace) images. It includes procedures for creating HA clusters, including installing required packages and agents, configuring fencing, and installing network resource agents. Each cloud provider has its own chapter that describes creating and deploying a custom image. There is also a separate chapter for configuring HA clusters for each cloud provider.

Table of Contents

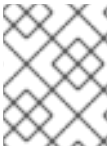
CHAPTER 1. DEPLOYING A RED HAT ENTERPRISE LINUX 7 IMAGE AS A VIRTUAL MACHINE ON MICROSOFT AZURE

You have a number of options for deploying a Red Hat Enterprise Linux (RHEL) 7 image on Azure. 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 Azure, and launching an Azure VM 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. See the Image Builder Guide for more information.

This chapter refers to the Azure documentation in a number of places. For many procedures, see the referenced Azure documentation for additional detail.



NOTE

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

Additional resources

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

1.1. RED HAT ENTERPRISE LINUX IMAGE OPTIONS ON AZURE

The following table lists image choices 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.	Leverage 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 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 Azure.	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 Microsoft for all other instance costs. Custom images that you move to Azure 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 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 the Image Builder Guide for more information.

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

Additional resources

- 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

1.2. UNDERSTANDING BASE IMAGES

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

1.2.1. Using a custom base image

To manually configure a VM, you start with a base (starter) VM image. Once you have created the base VM image, 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 Hyper-V cloud image of RHEL, see Prepare a RHEL 7 virtual machine from Hyper-V manager.

Additional resources

Red Hat Enterprise Linux

1.2.2. 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	Description	Command
qemu-kvm	This package provides the user-level KVM emulator and facilitates communication between hosts and guest VMs.	**# yum install qemu-kvm libvirt**

Package	Description	Command
qemu-img	This package provides disk management for guest VMs. The qemu-img package is installed as a dependency of the qemu-kvm package.	
libvirt	This package provides the server and host-side libraries for interacting with hypervisors and host systems and the libvirtd daemon that handles the library calls, manages VMs, and controls the hypervisor.	

Table 1.3. Additional Virtualization Packages

Package	Description	Command
virt-install	This package provides the **virt-install** command for creating VMs from the command line.	**# yum install virt-install libvirt-python virt-manager virt-install libvirt-client**
libvirt-python	This package contains a module that permits applications written in the Python programming language to use the interface supplied by the libvirt API.	
virt-manager	This package provides the virt-manager tool, also known as Virtual Machine Manager (VMM). VMM is a graphical tool for administering VMs. It uses the libvirt-client library as the management API.	
libvirt-client	This package provides the client-side APIs and libraries for accessing libvirt servers. The libvirt-client package includes the **virsh** command line tool to manage and control VMs and hypervisors from the command line or a special virtualization shell.	

Additional resources

- Installing Virtualization Packages Manually

1.2.3. 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 if you experience any errors.

Table 1.4. 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, running RHEL 7.5 and later, use Network Bound Disk Encryption (NBDE) for full disk encryption on Azure. NBDE is supported only on RHEL 7.5 and later.

1.2.4. 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.

Procedure

1. Download the latest Red Hat Enterprise Linux 7 Binary DVD ISO image from the Red Hat Customer Portal.
2. Ensure that you have enabled your host machine for virtualization. See the Virtualization Getting Started Guide for information and procedures.
3. Create and start a basic Red Hat Enterprise Linux VM. See the Getting Started with Virtualization Command-line Interface for instructions.
    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 isotest --memory 2048 --vcpus 2 --disk size=8,bus=virtio --location
rhel-7.0-x86_64-dvd.iso --os-variant=rhel7.0
```

b. If you use the VMM application to create your VM, follow the procedure in Getting Started with Virtual Machine Manager, with these caveats:

- Do not check **Immediately Start VM**.
- Change your **Memory** and **Storage 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.

4. 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 the 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**.

5. When the install starts:

- Create a **root** password.
- Create an administrative user account.

6. When installation is complete, reboot the VM and log in to the root account.

7. Once you are logged in as **root**, you can configure the image.

1.3. CONFIGURING THE BASE IMAGE FOR MICROSOFT AZURE

The base image requires configuration changes to serve as your RHEL 7 VM image in Azure. The following sections provide the additional configuration changes that Azure requires.

1.3.1. Installing Hyper-V device drivers

Microsoft provides network and storage device drivers as part of their Linux Integration Services (LIS) for the Hyper-V package. You may need to install Hyper-V device drivers on the VM image prior to provisioning it as an Azure 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.el7.x86_64/kernel/drivers/hv
-rw-r--r--   1 root     root        31272 Aug 11 08:45 usr/lib/modules/3.10.0-
932.el7.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.el7.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.el7.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 on your VM.

2. Create a file named **hv.conf** in **/etc/hv.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 "
```

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 exist in the environment.

4. Regenerate the **initramfs** image.

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

Verification steps

1. Reboot the machine.

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

1.3.2. Making additional configuration changes

The VM requires further configuration changes to operate in Azure. Perform the following procedure to make the additional changes.

Procedure

1. If necessary, power on the VM.

2. Register the VM and enable the Red Hat Enterprise Linux 7 repository.

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

Stopping and removing cloud-init (if present)

1. Stop the **cloud-init** service.

```
# systemctl stop cloud-init
```

2. Remove the **cloud-init** software.

```
# yum remove cloud-init
```

Completing other VM changes

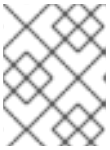
1. Edit the **/etc/ssh/sshd_config** file and enable password authentication.

```
PasswordAuthentication yes
```

2. Set a generic host name.

```
# hostnamectl set-hostname localhost.localdomain
```

3. Edit (or create) the **/etc/sysconfig/network-scripts/ifcfg-eth0** file. Use only the parameters listed below.



NOTE

The **ifcfg-eth0** file does not exist on the RHEL 7 DVD ISO image and must be created.

```
DEVICE="eth0"
ONBOOT="yes"
BOOTPROTO="dhcp"
TYPE="Ethernet"
USERCTL="yes"
PEERDNS="yes"
IPV6INIT="no"
```

4. Remove all 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
```

5. Set **ssh** to start automatically.

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

6. Modify the kernel boot parameters.

a. Add **crashkernel=256M** to the start of the **GRUB_CMDLINE_LINUX** line in the **/etc/default/grub** file. If **crashkernel=auto** is present, change it to **crashkernel=256M**.

b. Add the following lines to the end of the **GRUB_CMDLINE_LINUX** line (if not present).

```
earlyprintk=ttyS0
console=ttyS0
rootdelay=300
```

c. Remove the following options (if present).

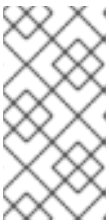
```
rhgb
quiet
```

7. Regenerate the **grub.cfg** file.

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

8. Install and enable the Windows Azure Linux Agent (WALinuxAgent).

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



NOTE

If you get the error message **No package WALinuxAgent available**, install the **rhel-7-server-extras-rpms** repository. Run the **# subscription-manager repos --enable=rhel-7-server-extras-rpms** command before trying the installation again.

9. Edit the following lines in the **/etc/waagent.conf** file to configure swap space for provisioned VMs. Set swap space for whatever is appropriate for your provisioned VMs.

```
Provisioning.DeleteRootPassword=n
ResourceDisk.Filesystem=ext4
ResourceDisk.EnableSwap=y
ResourceDisk.SwapSizeMB=2048
```

Preparing to provision

1. Unregister the VM from Red Hat Subscription Manager.

```
# subscription-manager unregister
```

2. Prepare the VM for Azure provisioning by cleaning up the existing provisioning details. Azure reprovisions the VM in Azure. This command generates data loss warnings, which are expected.

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

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

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

1.4. 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. Resize the image using the procedures in the **Aligning the image** section before proceeding to the next step.

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.

Aligning the image

Complete the following steps only if the **raw** file is not aligned.

1. 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>
```

2. 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.

1.5. INSTALLING THE AZURE CLI

Complete the following steps to install the Azure command line interface (Azure CLI 2.1) on your host machine. 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

1.6. 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:

```
$ 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:

```
$ 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;EndpointSuffi
x=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/azrhelclivnet1",
  "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/azrhelclivnet1/subnets/azrhelclisubnet1",
      "ipConfigurations": null,
      "name": "azrhelclisubnet1",
      "networkSecurityGroup": null,
      "provisioningState": "Succeeded",
      "resourceGroup": "azrhelclirsgrp",
      "resourceNavigationLinks": null,
      "routeTable": null
    }
  ],
```

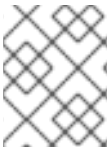
```
  "tags": {},
  "type": "Microsoft.Network/virtualNetworks",
  "virtualNetworkPeerings": null
  }
}
```

Additional resources

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

1.7. 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 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:

```
$ az storage blob upload --account-name azrhelclistact --container-name azrhelclistcont --
type page --file rhel-image-7.vhd --name rhel-image-7.vhd
Percent complete: %100.0
```

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

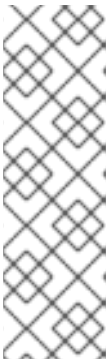
```
$ az storage blob url -c <container-name> -n <image-name>.vhd
```

Example:

```
$ az storage blob url -c azrhelclistcont -n rhel-image-7.vhd
"https://azrhelclistact.blob.core.windows.net/azrhelclistcont/rhel-image-7.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 rhel7 -g azrhelclirsgrp2 -l southcentralus --source
https://azrhelclistact.blob.core.windows.net/azrhelclistcont/rhel-image-7.vhd --os-type linux
```

1.8. 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.

```
$ 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>
```

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.

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 rhel7
{
  "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"
```

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.
```

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

You can now go to the 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 the **az --help** command 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.

1.9. 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>
```

1.10. 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. Refer to Creating Red Hat Customer Portal Activation Keys.

   - Alternatively, you can manually attach a subscription using the ID of the subscription pool (Pool ID). Refer to 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 2. CONFIGURING RED HAT HIGH AVAILABILITY CLUSTERS ON MICROSOFT AZURE

Red Hat supports High Availability (HA) on Red Hat Enterprise Linux (RHEL) 7.4 and later versions. This chapter includes information and procedures for configuring a Red Hat HA cluster on Microsoft Azure using 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 7 images to use for your cluster. For more information on image options for Azure, see Red Hat Enterprise Linux Image Options on 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.

This chapter also includes procedures specific to the creation of HA clusters, which transform individual VM 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.

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

Prerequisites

- You need to install the 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.

Additional resources

- Support Policies for RHEL High Availability Clusters - Microsoft Azure Virtual Machines as Cluster Members
- High Availability Add-On Overview

2.1. CREATING RESOURCES IN AZURE

Complete the following procedure to create an availability set. You need these resources to complete subsequent tasks in this chapter.

Procedure

- 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

2.2. CREATING AN AZURE ACTIVE DIRECTORY APPLICATION

Complete the following procedures 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

You need to 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

2.3. INSTALLING THE RED HAT HA PACKAGES AND AGENTS

Complete the following steps on all nodes.

Procedure

1. Register the VM with Red Hat.

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

2. Disable all repositories.

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

3. Enable the RHEL 7 Server and RHEL 7 Server HA repositories.

```
# subscription-manager repos --enable=rhel-7-server-rpms
# subscription-manager repos --enable=rhel-ha-for-rhel-7-server-rpms
```

4. Update all packages.

```
# yum update -y
```

5. Reboot if the kernel is updated.

```
# reboot
```

6. Install **pcs**, **pacemaker**, **fence agent**, **resource agent**, and **nmap-ncat**.

```
# yum install -y pcs pacemaker fence-agents-azure-arm resource-agents nmap-ncat
```

2.4. CONFIGURING HA SERVICES

Complete the following steps on all nodes.

Procedure

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

```
# passwd hacluster
```

2. Add the **high availability** service to the RHEL Firewall if **firewalld.service** is enabled.

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

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

```
# systemctl enable pcsd.service --now
```

Verification step

- Ensure the **pcs** service is running.

```
# systemctl is-active pcsd.service
```

2.5. 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**. 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 --name _hostname1_ _hostname2_ _hostname3_
```

Example:

```
[root@node01 clouduser]# pcs cluster setup --name newcluster 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 steps

1. Enable the cluster.

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

2. Start the cluster.

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

Example:

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

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

2.6. CREATING A FENCE DEVICE

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

Procedure

1. Identify the available instances that can be fenced.

```
# fence_azure_arm -l [appid] -p [authkey] --resourceGroup=[name] --subscriptionId=[name] --tenantId=[name] -o list
```

Example:

```
[root@node1 ~]# fence_azure_arm -l XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX -p XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX --resourceGroup=hacluster-rg --subscriptionId=XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX --tenantId=XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX -o list
```

```
node01-vm,
node02-vm,
node03-vm,
```

2. Create a fence device. Use the **pcmk_host_map** command to map the RHEL host name to the instance ID.

```
# pcs stonith create _clusterfence_ fence_azure_arm login=_AD-Application-ID_
passwd=_AD-passwd_ pcmk_host_map="_pcmk-host-map_ resourcegroup=
_myresourcegroup_ tenantid=_tenantid_ subscriptionid=_subscriptionid_
```

Verification steps

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

```
# pcs stonith fence _azurenodename_
```

Example:

```
[root@node01 ~]# pcs stonith fence fenceazure
 Resource: fenceazure (class=stonith type=fence_azure_arm)
  Attributes: login=XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX passwd=XXXXXXXX-
XXXX-XXXX-XXXX-XXXXXXXXXXXX pcmk_host_map=nodea:nodea-vm;nodeb:nodeb-
vm;nodec:nodec-vm pcmk_reboot_retries=4 pcmk_reboot_timeout=480 power_timeout=240
resourceGroup=rg subscriptionId=XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
tenantId=XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
  Operations: monitor interval=60s (fenceazure-monitor-interval-60s)
[root@node01 ~]# pcs stonith
 fenceazure     (stonith:fence_azure_arm):      Started nodea
```

2. Check the status to verify the node started.

```
# watch pcs status
```

Example:

```
[root@node01 ~]# watch pcs status
 fenceazure     (stonith:fence_azure_arm):      Started nodea
```

Additional resources

- Fencing in a Red Hat High Availability Cluster
- High Availability Add-On Administration

2.7. 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

Access to the 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 backend 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 a 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, use the default values that are prepopulated. Ensure to set **Floating IP (direct server return)** to **Enabled**.

2.8. CONFIGURING THE AZURE 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. Enter the **Azure id** command to view the Azure load balancer resource agent description. This shows the options and default operations for this agent.

```
# pcs resource describe _azure-id_
```

2. Create an **lpaddr2** resource for managing the IP on the node.

```
# pcs resource create _resource-id_ IPaddr2 ip=_virtual/floating-ip_
cidr_netmask=_virtual/floating-mask_ --group _group-id_ nic=_network-interface_ op monitor
interval=30s
```

   Example:

```
[root@node01 ~]# pcs resource create ClusterIP ocf:heartbeat:IPaddr2 ip=172.16.66.99
cidr_netmask=24 --group CloudIP nic=eth0 op monitor interval=30s
```

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

```
# pcs resource create _resource-loadbalancer-name_ azure-lb port=_port-number_ --group
_cluster-resources-group_
```

Verification step

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

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

Example:

```
[root@node01 ~]# pcs status
Cluster name: hacluster

WARNINGS:
No stonith devices and stonith-enabled is not false

Stack: corosync
Current DC: nodeb (version 1.1.22-1.el7-63d2d79005) - partition with quorum
Last updated: Wed Sep  9 16:47:07 2020
Last change: Wed Sep  9 16:44:32 2020 by hacluster via crmd on nodeb

3 nodes configured
0 resource instances configured

Online: [ node01 node02 node03 ]

No resources


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

Additional resources

- Cluster Operation

2.9. 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 1TB 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. 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 virtual machines 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 virtual machine 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-
7.0-20200713.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 steps

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_s "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**.

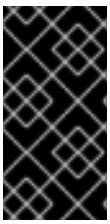
   ```
   # ssh 198.51.100.3 "hostname ; lsblk -d | grep ' 1T ' | awk '{print \$1}' | xargs -i udevadm info -
   -query=all --name=/dev/{} | grep '^E: ID_SERIAL='"

   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. For information on configuring resilient storage for a Red Hat High Availability cluster, see Configuring a GFS2 File System in a Cluster . For general information on GFS2 file systems, see Configuring and managing 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) 7 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). The chapter also 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 AMI (Amazon Machine Image) in the **ami** format. See the Image Builder Guide for more information.

This chapter refers to the Amazon documentation in a number of places. For many procedures, see the referenced Amazon documentation for additional detail.

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.

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

3.1. 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 the Image Builder Guide 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

- Using Red Hat Gold Images
- Red Hat Cloud Access program
- Image Builder Guide
- AWS Management Console
- AWS Marketplace

3.2. INSTALLING THE AWS CLI

Many of the procedures in this chapter include using the AWS CLI. Complete the following steps to install the AWS CLI.

Prerequisites

You need to have created and have access to an AWS Access Key ID and an AWS Secret Access Key. See Quickly Configuring the AWS CLI for information and instructions.

Procedure

1. Install Python 3 and the **pip** tool.

```
# yum install python3
# yum install python3-pip
```

2. Install the AWS command line tools with the **pip** command.

```
# pip3 install awscli
```

3. Run the **aws --version** command to verify that you installed the AWS CLI.

```
$ aws --version
aws-cli/1.16.182 Python/2.7.5 Linux/3.10.0-957.21.3.el7.x86_64 botocore/1.12.172
```

4. 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

3.3. 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 base image from an ISO image.

Prerequisites

Enable virtualization for your Red Hat Enterprise Linux 7 host machine by following the Virtualization Deployment and Administration Guide.

3.4.1. Downloading the ISO image

Procedure

1. Download the latest Red Hat Enterprise Linux ISO image from the Red Hat Customer Portal .
2. Move the image to the **/var/lib/libvirt/images** directory.

3.4.2. Creating a VM from an ISO image

Procedure

1. Ensure that you have enabled your host machine for virtualization. For information and procedures to install virutalization packages, see Installing virtualization packages on an existing Red Hat Enterprise Linux system
2. Create and start a basic Red Hat Enterprise Linux VM. For instructions to create VM, refer to Creating a virtual machine .

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 _vmname_ --memory 2048 --vcpus 2 --disk size=8,bus=virtio --location
rhel-7.0-x86_64-dvd.iso --os-variant=rhel7.0
```

b. If you use the virt-manager application to create your VM, follow the procedure in Creating guests with virt-manager, with these caveats:

- Do not check **Immediately Start VM**.
- Change your **Memory** and **Storage 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.

3.4.3. 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**.
5. Reboot the VM and log in as **root** once the installation completes.
6. Configure the image.

NOTE

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

7. Power down the VM.

3.5. UPLOADING THE RED HAT ENTERPRISE LINUX IMAGE TO AWS

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

3.5.1. Creating an S3 bucket

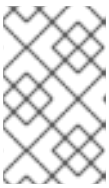
Importing to AWS requires an Amazon S3 bucket. An Amazon S3 bucket is an Amazon resource where you store objects. As part of the process for uploading your image, you create an S3 bucket and then move your image to the bucket. Complete the following steps to create a bucket.

Prerequisites

- You need to have AWS CLI installed. For more information, see Installing the AWS CLI.

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. Select **Region**. Enter your region into the field, or click the drop-down and select your region from all available regions.
   c. Click **Next**.
4. In the **Configure options** view, select 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, **aws s3 mb s3://my-new-bucket** creates an S3 bucket named **my-new-bucket**. See the AWS CLI Command Reference for information on the **mb** command.

Additional resources

- Amazon S3 Console
- AWS CLI Command Reference

3.5.2. 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. A sample follows.

```
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. A sample follows.

```
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

3.5.3. Converting and pushing an AMI to S3

Complete the following procedure to convert and push your Amazon Machine Image (AMI) 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. For more information, see How VM Import/Export Works on image formats that Amazon accepts.

Procedure

1. Run the **qemu-img** command to convert your image. A sample follows.

```
qemu-img convert -f qcow2 -O raw rhel-server-7.7-1-x86_64-kvm.qcow2 rhel-server-7.7-1-x86_64-kvm.raw
```

2. Push the image to S3.

```
aws s3 cp rhel-server-7.7.1-x86_64-kvm.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

3.5.4. Creating an AMI from a raw image

Perform the following procedure to create an AMI from the raw image.

Prerequisites

- You need to have AWS CLI installed. For more information, see Installing the AWS CLI.

Procedure

- You can run the **aws ec2 import-image** command on the AWS CLI to create an AMI from the raw image.

```
# aws ec2 import-image --platform Linux --license-type BYOL --no-encrypted --description
_imagedescription_ --architecture x86_64 --disk-containers Format=Raw,UserBucket="
{S3Bucket=virtqes1,S3Key=rhel-server-ec2-7.9-30.x86_64.raw}" --region _regionname_
```

Additional resources

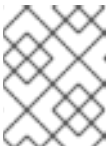
- Import Your VM as an Image

3.5.5. Launching an instance from the AMI

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

Procedure

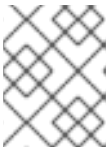
1. From the AWS EC2 Dashboard, select **Images** and then **AMIs**.
2. Right-click on your image and select **Launch**.
3. Choose an **Instance Type** that meets or exceeds the requirements of your workload. See Amazon EC2 Instance Types for information on instance types.
4. Click **Next: Configure Instance 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**.

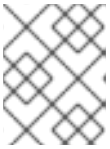


NOTE

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 command 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

3.5.6. 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. Refer to Creating Red Hat Customer Portal Activation Keys.
   - Alternatively, you can manually attach a subscription using the ID of the subscription pool (Pool ID). Refer to 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 4. CONFIGURING RED HAT HIGH AVAILABILITY CLUSTERS ON AWS

This chapter includes information and procedures for configuring a Red Hat High Availability (HA) cluster on Amazon Web Services (AWS) using EC2 instances as cluster nodes. 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.

This chapter also includes 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.

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

Prerequisites

- You need to install the AWS command line interface (CLI). For more information on installing AWS CLI, see Installing the AWS 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 AWS with full support from Red Hat.

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
- Support Policies for RHEL High Availability Clusters

4.1. 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

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 to 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.

4.2. INSTALLING THE HA PACKAGES AND AGENTS

Complete the following steps on all nodes to install the HA 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 7 Server and RHEL 7 Server HA repositories.

```
# subscription-manager repos --enable=rhel-7-server-rpms
# subscription-manager repos --enable=rhel-ha-for-rhel-7-server-rpms
```

5. Update all packages.

```
# yum update -y
```

6. Reboot if the kernel is updated.

```
# reboot
```

7. Install pcs, pacemaker, fence agent, and resource agent.

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

8. 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
```

9. Add the **high availability** service to the RHEL Firewall if **firewalld.service** is enabled.

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

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

```
# systemctl enable pcsd.service --now
```

Verification step

Ensure the **pcs** service is running.

```
# systemctl is-active pcsd.service
```

4.3. 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**. 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 --name _hostname1_ _hostname2_ _hostname3_
```

Example:

```
[root@node01 clouduser]# pcs cluster setup --name newcluster 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 steps

1. Enable the cluster.

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

2. Start the cluster.

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

Example:

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

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

4.4. CREATING A FENCING DEVICE

Complete the following steps to configure fencing.

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. Create a 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 you previously set up in Creating the AWS Access Key and AWS Secret Access Key .

```
# pcs stonith create cluster_fence 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"
```

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
```

Verification steps

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

```
# pcs stonith fence _awsnodename_
```

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.

```
# watch 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
```

4.5. INSTALLING THE AWS CLI ON CLUSTER NODES

Previously, you installed the AWS CLI on your host system. You now 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. For more information, see Creating the AWS Access Key and AWS Secret Access Key .

Procedure

1. Perform the procedure 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
```

4.6. 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 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 step

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 step

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
- High Availability Add-On Administration
- High Availability Add-On Reference

4.7. CONFIGURING SHARED BLOCK STORAGE

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

Procedure

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

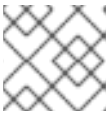
   ```
   $ aws ec2 create-volume --availability-zone availability_zone --no-encrypted --size 1024 --volume-type io1 --iops 51200 --multi-attach-enabled
   ```

   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 steps

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
```

After you have verified that the shared disk is attached to each instance, you can configure resilient storage for the cluster. For information on configuring resilient storage for a Red Hat High Availability cluster, see Configuring a GFS2 File System in a Cluster . For general information on GFS2 file systems, see Configuring and managing GFS2 file systems.

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

You have a number of options for deploying a Red Hat Enterprise Linux (RHEL) 7 image as a Google Compute Engine (GCE) instance on Google Cloud Platform (GCP). This chapter discusses your options for choosing an image and lists or refers to system requirements for your host system and VM. The chapter provides procedures for creating a custom VM from an ISO image, uploading to GCE, and launching an instance.

This chapter refers to the Google documentation in a number of places. For many procedures, see the referenced Google documentation for additional detail.

NOTE

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

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.

Additional resources

- Red Hat in the Public Cloud
- Google Cloud

5.1. RED HAT ENTERPRISE LINUX IMAGE OPTIONS ON GCP

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

Table 5.1. Image options

Image option	Subscriptions	Sample scenario	Considerations

Image option	Subscriptions	Sample scenario	Considerations
Choose to deploy a custom image that you move to GCP.	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 all other instance costs. Custom images that you move to GCP 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 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
- Images
- Red Hat Cloud Access Reference Guide
- Creating an instance from a custom image

5.2. UNDERSTANDING BASE IMAGES

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

5.2.1. Using a custom base image

To manually configure a VM, you start with a base (starter) VM image. Once you have created the base VM image, 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

5.2.2. 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.

5.3. CREATING A BASE VM FROM AN ISO IMAGE

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

Prerequisites

Enable virtualization for your Red Hat Enterprise Linux 7 host machine by following the Virtualization Deployment and Administration Guide.

5.3.1. Downloading the ISO image

Procedure

1. Download the latest Red Hat Enterprise Linux ISO image from the Red Hat Customer Portal .
2. Move the image to the **/var/lib/libvirt/images** directory.

5.3.2. Creating a VM from an ISO image

Procedure

1. Ensure that you have enabled your host machine for virtualization. For information and procedures to install virutalization packages, see Installing virtualization packages on an existing Red Hat Enterprise Linux system
2. Create and start a basic Red Hat Enterprise Linux VM. For instructions to create VM, refer to Creating a virtual machine .
    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 _vmname_ --memory 2048 --vcpus 2 --disk size=8,bus=virtio --location
rhel-7.0-x86_64-dvd.iso --os-variant=rhel7.0
```

b. If you use the virt-manager application to create your VM, follow the procedure in Creating guests with virt-manager, with these caveats:

- Do not check **Immediately Start VM**.
- Change your **Memory** and **Storage 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.

5.3.3. 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**.
5. Reboot the VM and log in as **root** once the installation completes.
6. Configure the image.

NOTE

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

7. Power down the VM.

5.4. UPLOADING THE RHEL IMAGE TO GCP

Follow the procedures in this section on your host machine to upload your image to GCP.

5.4.1. Creating a new project on GCP

Complete the following steps to create a new project on GCP.

Prerequisites

You must have created an account with GCP. If you have 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 the **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

5.4.2. Installing the Google Cloud SDK

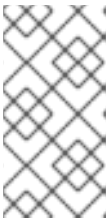
Complete the following steps to install the Google Cloud SDK.

Prerequisites

- Create a project on the GCP if you have not already done so. See Creating a new project on the Google Cloud Platform for more information.
- Ensure that your host system includes Python 2.7 or later. If it does not, install Python 2.7.

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

5.4.3. 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.

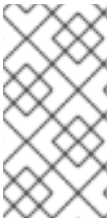
Procedure

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

5.4.4. 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

5.4.5. 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 a **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-sample.qcow2 disk.raw
```

2. Tar the image.

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

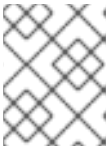
3. Upload the image to the bucket you created previously. Upload could take a few minutes.

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

Verification steps

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

2. 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

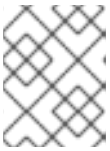
5.4.6. Creating an image from the object in the GCP bucket

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

Procedure

- 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 information.

- 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

5.4.7. Creating a Google Compute Engine instance from an image

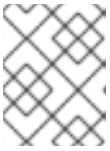
Complete the following steps to configure a GCE VM instance using the GCP Console.

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**, 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**, 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.

5.4.8. 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.

Additional resources

- gcloud compute instances list
- Connecting to instances

5.4.9. 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. Refer to Creating Red Hat Customer Portal Activation Keys.
   - Alternatively, you can manually attach a subscription using the ID of the subscription pool (Pool ID). Refer to 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 CLUSTERS ON GOOGLE CLOUD PLATFORM

This chapter includes 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.

The chapter includes prerequisite procedures for setting up your environment for GCP. Once you have set up your environment, you can create and configure GCP 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 GCP. These include procedures for installing the High Availability packages and agents on each cluster node, configuring fencing, and installing GCP network resource agents.

The chapter refers to GCP documentation in a number of places. For more information, see the referenced GCP documentation.

Prerequisites

- You need to install the GCP software development kit (SDK). For more information see, Installing the Google cloud SDK .
- Enable your subscriptions in the Red Hat Cloud Access program . The Red Hat Cloud Access program allows you to move your Red Hat Subscription from physical or on-premise systems onto GCP with full support from Red Hat.
- 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

Additional resources

- Support Policies for RHEL High Availability Clusters - Google Cloud Platform Virtual Machines as Cluster Members
- 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
- Quickstart for Red Hat and Centos

6.1. RED HAT ENTERPRISE LINUX IMAGE OPTIONS ON GCP

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

Table 6.1. Image options

Image option	Subscriptions	Sample scenario	Considerations
Choose to deploy a custom image that you move to GCP.	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 all other instance costs. Custom images that you move to GCP 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 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
- Images
- Red Hat Cloud Access Reference Guide
- Creating an instance from a custom image

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 6.2. System packages

Package	Description	Command
qemu-kvm	This package provides the user-level KVM emulator and facilitates communication between hosts and guest VMs.	**# yum install qemu-kvm libvirt**
qemu-img	This package provides disk management for guest VMs. The qemu-img package is installed as a dependency of the qemu-kvm package.	
libvirt	This package provides the server and host-side libraries for interacting with hypervisors and host systems and the libvirtd daemon that handles the library calls, manages VMs, and controls the hypervisor.	

Table 6.3. Additional Virtualization Packages

Package	Description	Command
virt-install	This package provides the **virt-install** command for creating VMs from the command line.	**# yum install virt-install libvirt-python virt-manager virt-install libvirt-client**

Package	Description	Command
libvirt-python	This package contains a module that permits applications written in the Python programming language to use the interface supplied by the libvirt API.	
virt-manager	This package provides the virt-manager tool, also known as Virtual Machine Manager (VMM). VMM is a graphical tool for administering VMs. It uses the libvirt-client library as the management API.	
libvirt-client	This package provides the client-side APIs and libraries for accessing libvirt servers. The libvirt-client package includes the **virsh** command line tool to manage and control VMs and hypervisors from the command line or a special virtualization shell.	

Additional resources

- Installing Virtualization Packages Manually

6.3. INSTALLING THE HA PACKAGES AND AGENTS

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

Procedure

1. Disable all repositories.

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

2. Enable RHEL 7 server and RHEL 7 server HA repositories.

```
# subscription-manager repos --enable=rhel-7-server-rpms
# subscription-manager repos --enable=rhel-ha-for-rhel-7-server-rpms
```

3. Update all packages.

```
# yum update -y
```

4. Install **pcs pacemaker** fence agent and resource agent.

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

5. Reboot the machine if the kernel is updated.

```
# reboot
```

6.4. CONFIGURING HA SERVICES

Complete the following steps on all nodes to configure High Availability 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 enabled, add the high availability service to RHEL.

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

# firewall-cmd --reload
```

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

```
# systemctl enable pcsd.service --now
```

Verification steps

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

```
# systemctl is-active pcsd.service
```

6.5. 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 **ha cluster**. Specify the name of each node in the cluster.

```
# pcs cluster auth  _hostname1_ _hostname2_ _hostname3_ -u hacluster
```

Example:

```
[root@node01 ~]# pcs cluster auth node01 node02 node03 -u hacluster
node01: Authorized
node02: Authorized
node03: Authorized
```

2. Create the cluster.

```
# pcs cluster setup --name cluster-name _hostname1_ _hostname2_  _hostname3_
```

Verification steps

1. Enable the cluster.

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

2. Start the cluster.

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

6.6. CREATING A FENCE DEVICE

For most default configurations, the GCP instance names and the RHEL host names are identical.

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

Procedure

1. Get the GCP instance names from any node in the cluster. Note that the output also shows the internal ID for the instance.

```
# fence_gce --zone _gcp_ _region_ --project= _gcp_ _project_ -o list
```

Example:

```
[root@rhel71-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. Create a fence device. Use the **pcmk_host-name** command to map the RHEL host name with the instance ID.

```
# pcs stonith create _clusterfence_ fence_gce pcmk_host_map=_pcmk-hpst-map_
fence_gce zone=_gcp-zone_ project=_gcpproject_
```

Example:

```
[root@node01 ~]# pcs stonith create fencegce fence_gce
pcmk_host_map="node01:node01-vm;node02:node02-vm;node03:node03-vm"
project=hacluster zone=us-east1-b
```

Verification steps

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

```
# pcs stonith fence gcp nodename
```

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

```
# watch pcs status
```

Example:

```
[root@node01 ~]# watch pcs status
Cluster name: gcp-cluster
Stack: corosync
Current DC: rhel71-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 rhel71-node-01

3 nodes configured
3 resources configured

Online: [ rhel71-node-01 rhel71-node-02 rhel71-node-03 ]

Full list of resources:

us-east1-b-fence    (stonith:fence_gce):    Started rhel71-node-01

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

6.7. 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
```

6.8. CONFIGURING THE GCP NETWORK RESOURCE AGENT

The cluster uses GCP network resource agents attached to 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.

Procedure

Enter the following command to view the GCP virtual IP address resource agent (gcp-vpc-move-vip) description. This shows the options and default operations for this agent.

```
# 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.

Primary subnet address range

Procedure

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

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

```
# pcs resource create aliasip gcp-vpc-move-vip  alias_ip=_UnusedIPaddress/CIDRblock_ --group _group-name_ --group _networking-group_
```

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

```
# pcs resource create vip IPaddr2 nic=_interface_ ip=_AliasIPaddress_ cidr_netmask=32 --group _group-name_ --group _networking-group_
```

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

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

Verification steps

1. Verify that the resources have started and are grouped under **vipgrp**.

```
# pcs status
```

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

```
# pcs resource move vip _Node_
```

Example:

```
# pcs resource move vip rhel71-node-03
```

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

```
# pcs status
```

Secondary subnet address range

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

Prerequisites

Create a custom network and subnet

Procedure

1. 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. 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_ --group _group-name_ --group _networking-group_
```

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

```
# pcs resource create vip IPaddr2 nic=_interface_ ip=_AliasIPaddress_ cidr_netmask=32 --group _group-name_ --group _networking-group_
```

Verification steps

1. Verify that the resources have started and are grouped under **vipgrp**.

```
# pcs status
```

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

```
# pcs resource move vip _Node_
```

Example:

```
[root@rhel71-node-01 ~]# pcs resource move vip rhel71-node-03
```

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

```
# pcs status
```