



Red Hat Enterprise Linux 9

Deploying RHEL 9 on Amazon Web Services

Obtaining RHEL system images and creating RHEL instances on AWS

Red Hat Enterprise Linux 9 Deploying RHEL 9 on Amazon Web Services

Obtaining RHEL system images and creating RHEL instances on AWS

Legal Notice

Copyright © 2023 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

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

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE	3
PROVIDING FEEDBACK ON RED HAT DOCUMENTATION	4
CHAPTER 1. INTRODUCING RHEL ON PUBLIC CLOUD PLATFORMS	5
1.1. BENEFITS OF USING RHEL IN A PUBLIC CLOUD	5
1.2. PUBLIC CLOUD USE CASES FOR RHEL	6
1.3. FREQUENT CONCERNS WHEN MIGRATING TO A PUBLIC CLOUD	7
1.4. OBTAINING RHEL FOR PUBLIC CLOUD DEPLOYMENTS	7
1.5. METHODS FOR CREATING RHEL CLOUD INSTANCES	8
CHAPTER 2. CREATING AND UPLOADING AWS AMI IMAGES	9
2.1. PREPARING TO UPLOAD AWS AMI IMAGES	9
2.2. UPLOADING AN AMI IMAGE TO AWS USING THE CLI	10
2.3. PUSHING IMAGES TO AWS CLOUD AMI	11
CHAPTER 3. DEPLOYING A RED HAT ENTERPRISE LINUX IMAGE AS AN EC2 INSTANCE ON AMAZON WEB SERVICES	14
3.1. RED HAT ENTERPRISE LINUX IMAGE OPTIONS ON AWS	14
3.2. UNDERSTANDING BASE IMAGES	16
3.2.1. Using a custom base image	16
3.2.2. Virtual machine configuration settings	16
3.3. CREATING A BASE VM FROM AN ISO IMAGE	16
3.3.1. Creating a VM from the RHEL ISO image	16
3.3.2. Completing the RHEL installation	17
3.4. UPLOADING THE RED HAT ENTERPRISE LINUX IMAGE TO AWS	18
3.4.1. Installing the AWS CLI	18
3.4.2. Creating an S3 bucket	19
3.4.3. Creating the vmimport role	20
3.4.4. Converting and pushing your image to S3	21
3.4.5. Importing your image as a snapshot	22
3.4.6. Creating an AMI from the uploaded snapshot	23
3.4.7. Launching an instance from the AMI	24
3.4.8. Attaching Red Hat subscriptions	25
3.4.9. Setting up automatic registration on AWS Gold Images	25
3.5. ADDITIONAL RESOURCES	26
CHAPTER 4. CONFIGURING A RED HAT HIGH AVAILABILITY CLUSTER ON AWS	27
4.1. THE BENEFITS OF USING HIGH-AVAILABILITY CLUSTERS ON PUBLIC CLOUD PLATFORMS	27
4.2. CREATING THE AWS ACCESS KEY AND AWS SECRET ACCESS KEY	28
4.3. INSTALLING THE AWS CLI	28
4.4. CREATING AN HA EC2 INSTANCE	29
4.5. CONFIGURING THE PRIVATE KEY	30
4.6. CONNECTING TO AN EC2 INSTANCE	31
4.7. INSTALLING THE HIGH AVAILABILITY PACKAGES AND AGENTS	31
4.8. CREATING A CLUSTER	32
4.9. CONFIGURING FENCING	33
4.10. INSTALLING THE AWS CLI ON CLUSTER NODES	36
4.11. INSTALLING NETWORK RESOURCE AGENTS	37
4.12. CONFIGURING SHARED BLOCK STORAGE	40
4.13. ADDITIONAL RESOURCES	41

MAKING OPEN SOURCE MORE INCLUSIVE

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

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

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

Submitting comments on specific passages

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

Submitting feedback through Jira (account required)

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

CHAPTER 1. INTRODUCING RHEL ON PUBLIC CLOUD PLATFORMS

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

To learn more about RHEL on public cloud platforms, see:

- [Benefits of using RHEL in a public cloud](#)
- [Public cloud use cases for RHEL](#)
- [Frequent concerns when migrating to a public cloud](#)
- [Obtaining RHEL for public cloud deployments](#)
- [Methods for creating RHEL cloud instances](#)

1.1. BENEFITS OF USING RHEL IN A PUBLIC CLOUD

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

- **Flexible and fine-grained allocation of resources**

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

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

- **Space and cost efficiency**

You do not need to own any on-premises servers to host your cloud workloads. This avoids the space, power, and maintenance requirements associated with physical hardware.

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

- **Software-controlled configurations**

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

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

- **Separation from the host and software compatibility**

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

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

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

Additional resources

- [What is public cloud?](#)
- [What is a hyperscaler?](#)
- [Types of cloud computing](#)
- [Public cloud use cases for RHEL](#)
- [Obtaining RHEL for public cloud deployments](#)
- [Why run Linux on AWS?](#)

1.2. PUBLIC CLOUD USE CASES FOR RHEL

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

Beneficial use cases

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

Potentially problematic use cases

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

Next steps

- [Obtaining RHEL for public cloud deployments](#)

Additional resources

- [Should I migrate my application to the cloud? Here's how to decide.](#)

1.3. FREQUENT CONCERNS WHEN MIGRATING TO A PUBLIC CLOUD

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

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

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

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

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

The data in your RHEL cloud instances is in your ownership, and your public cloud provider does not have any access to it. In addition, major cloud providers support data encryption in transit, which improves the security of data when migrating your virtual machines to the public cloud.

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

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

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

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

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

1.4. OBTAINING RHEL FOR PUBLIC CLOUD DEPLOYMENTS

To deploy a RHEL system in a public cloud environment:

1. Select the optimal cloud provider for your use case, based on your requirements and the current offer on the market.

The cloud providers currently certified for running RHEL instances are:

- [Amazon Web Services \(AWS\)](#)
- [Google Cloud Platform \(GCP\)](#)

- [Microsoft Azure](#)



NOTE

This document specifically talks about deploying RHEL on AWS.

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

Additional resources

- [RHUI documentation](#)
- [Red Hat Open Hybrid Cloud](#)

1.5. METHODS FOR CREATING RHEL CLOUD INSTANCES

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

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

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

Purchase a RHEL instance directly from the cloud provider marketplace.

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



NOTE

For detailed instructions on using various methods to deploy RHEL instances on Amazon Web Services, see the following chapters in this document.

Additional resources

- [What is a golden image?](#)
- [Configuring and managing cloud-init for RHEL 9](#)

CHAPTER 2. CREATING AND UPLOADING AWS AMI IMAGES

To use your customized RHEL system image in the Amazon Web Services (AWS) cloud, create the system image with Image Builder using the respective output type, configure your system for uploading the image, and upload the image to your AWS account.

2.1. PREPARING TO UPLOAD AWS AMI IMAGES

Before uploading an AWS AMI image, you must configure a system for uploading the images.

Prerequisites

- You must have an Access Key ID configured in the [AWS IAM account manager](#).
- You must have a writable [S3 bucket](#) prepared.

Procedure

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

```
# dnf install python3
# dnf install python3-pip
```

2. Install the [AWS command-line tools](#) with **pip**:

```
# pip3 install awscli
```

3. Run the following command to set your profile. The terminal prompts you to provide your credentials, region and output format:

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

4. Define a name for your bucket and use the following command to create a bucket:

```
$ BUCKET=bucketname
$ aws s3 mb s3://$BUCKET
```

Replace *bucketname* with the actual bucket name. It must be a globally unique name. As a result, your bucket is created.

5. To grant permission to access the S3 bucket, create a **vmimport** S3 Role in the AWS Identity and Access Management (IAM), if you have not already done so in the past:

```
$ printf '{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": { "Service":
"vmie.amazonaws.com" }, "Action": "sts:AssumeRole", "Condition": { "StringEquals":{
"sts:Externalid": "vmimport" } } } ] }' > trust-policy.json
$ printf '{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [
"s3:GetBucketLocation", "s3:GetObject", "s3:ListBucket" ], "Resource": [ "arn:aws:s3:::%s",
"arn:aws:s3:::%s/*" ] }, { "Effect": "Allow", "Action": [ "ec2:ModifySnapshotAttribute",
```

```
"ec2:CopySnapshot", "ec2:RegisterImage", "ec2:Describe*" ], "Resource": "*" } ] }' $BUCKET
$BUCKET > role-policy.json
$ aws iam create-role --role-name vmimport --assume-role-policy-document file://trust-
policy.json
$ aws iam put-role-policy --role-name vmimport --policy-name vmimport --policy-document
file://role-policy.json
```

Additional resources

- [Using high-level \(s3\) commands with the AWS CLI](#)

2.2. UPLOADING AN AMI IMAGE TO AWS USING THE CLI

You can use image builder to build **ami** images and push them directly to Amazon AWS Cloud service provider using the CLI.

Prerequisites

- You have an **Access Key ID** configured in the [AWS IAM](#) account manager.
- You have a writable [S3 bucket](#) prepared.
- You have a defined blueprint.

Procedure

1. Using the text editor, create a configuration file with the following content:

```
provider = "aws"

[settings]
accessKeyID = "AWS_ACCESS_KEY_ID"
secretAccessKey = "AWS_SECRET_ACCESS_KEY"
bucket = "AWS_BUCKET"
region = "AWS_REGION"
key = "IMAGE_KEY"
```

Replace values in the fields with your credentials for **accessKeyID**, **secretAccessKey**, **bucket**, and **region**. The **IMAGE_KEY** value is the name of your VM Image to be uploaded to EC2.

2. Save the file as *CONFIGURATION-FILE.toml* and close the text editor.
3. Start the compose:

```
# composer-cli compose start BLUEPRINT-NAME IMAGE-TYPE IMAGE_KEY
CONFIGURATION-FILE.toml
```

Replace:

- *BLUEPRINT-NAME* with the name of the blueprint you created
- *IMAGE-TYPE* with the **ami** image type.
- *IMAGE_KEY* with the name of your VM Image to be uploaded to EC2.

- `CONFIGURATION-FILE.toml` with the name of the configuration file of the cloud provider.



NOTE

You must have the correct IAM settings for the bucket you are going to send your customized image to. You have to set up a policy to your bucket before you are able to upload images to it.

4. Check the status of the image build and upload it to AWS:

```
# composer-cli compose status
```

After the image upload process is complete, you can see the "FINISHED" status.

Verification

To confirm that the image upload was successful:

1. Access [EC2](#) on the menu and select the correct region in the AWS console. The image must have the **available** status, to indicate that it was successfully uploaded.
2. On the dashboard, select your image and click **Launch**.

Additional Resources

- [Required service role to import a VM](#)

2.3. PUSHING IMAGES TO AWS CLOUD AMI

You can push the output image that you create directly to the **Amazon AWS Cloud AMI** service provider.

Prerequisites

- You must have **root** or **wheel** group user access to the system.
- You have opened the image builder interface of the RHEL web console in a browser.
- You have create a blueprint. See [Creating an image builder blueprint in the web console interface](#).
- You must have an Access Key ID configured in the [AWS IAM](#) account manager.
- You must have a writable [S3 bucket](#) prepared.

Procedure

1. Click the **blueprint name**.
2. Select the tab **Images**.
3. Click **Create Image** to create your customized image.
A pop-up window opens.
 - a. From the **Type** drop-down menu list, select **Amazon Machine Image Disk (.raw)**.

- b. Check the **Upload to AWS** check box to upload your image to the AWS Cloud and click **Next**.
- c. To authenticate your access to AWS, type your **AWS access key ID** and **AWS secret access key** in the corresponding fields. Click **Next**.

**NOTE**

You can view your AWS secret access key only when you create a new Access Key ID. If you do not know your Secret Key, generate a new Access Key ID.

- d. Type the name of the image in the **Image name** field, type the Amazon bucket name in the **Amazon S3 bucket name** field and type the **AWS region** field for the bucket you are going to add your customized image to. Click **Next**.
- e. Review the information and click **Finish**.
Optionally, you can click **Back** to modify any incorrect detail.

**NOTE**

You must have the correct IAM settings for the bucket you are going to send your customized image. This procedure uses the IAM Import and Export, so you have to set up a **policy** to your bucket before you are able to upload images to it. For more information, see [Required Permissions for IAM Users](#).

4. A small pop-up on the upper right informs you of the saving progress. It also informs that the image creation has been initiated, the progress of this image creation and the subsequent upload to the AWS Cloud.
After the process is complete, you can see the **Image build complete** status.
5. Click [Service→EC2](#) on the menu and choose the [correct region](#) in the AWS console. The image must have the **Available** status, to indicate that it is uploaded.
6. On the dashboard, select your image and click **Launch**.
7. A new window opens. Choose an instance type according to the resources you need to start your image. Click **Review and Launch**.
8. Review your instance start details. You can edit each section if you need to make any changes. Click **Launch**.
9. Before you start the instance, select a public key to access it.
You can either use the key pair you already have or you can create a new key pair.

Follow the next steps to create a new key pair in EC2 and attach it to the new instance.

- a. From the drop-down menu list, select **Create a new key pair**.
 - b. Enter the name to the new key pair. It generates a new key pair.
 - c. Click **Download Key Pair** to save the new key pair on your local system.
10. Then, you can click **Launch Instance** to start your instance.
You can check the status of the instance, which displays as **Initializing**.

11. After the instance status is **running**, the **Connect** button becomes available.
12. Click **Connect**. A pop-up window appears with instructions on how to connect using SSH.
 - a. Select **A standalone SSH client** as the preferred connection method to and open a terminal.
 - b. In the location you store your private key, ensure that your key is publicly viewable for SSH to work. To do so, run the command:

```
$ chmod 400 <your-instance-name.pem>_
```

- c. Connect to your instance using its Public DNS:

```
$ ssh -i "<_your-instance-name.pem_"> ec2-user@<_your-instance-IP-address_>
```

- d. Type **yes** to confirm that you want to continue connecting.
As a result, you are connected to your instance using SSH.

Verification

1. Check if you are able to perform any action while connected to your instance using SSH.

Additional resources

- [Open a case on Red Hat Customer Portal](#)
- [Connecting to your Linux instance using SSH](#)
- [Open support cases](#)

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) 9 image as an EC2 instance on Amazon Web Services (AWS). This chapter discusses your options for choosing an image and lists or refers to system requirements for your host system and virtual machine (VM). This chapter also provides procedures for creating a custom VM from an ISO image, uploading it to EC2, and launching an EC2 instance.

To deploy a Red Hat Enterprise Linux 9 (RHEL 9) as an EC2 instance on Amazon Web Services (AWS), follow the information below. This chapter:

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



IMPORTANT

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



NOTE

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

Prerequisites

- Sign up for a [Red Hat Customer Portal](#) account.
- Sign up for AWS and set up your AWS resources. See [Setting Up with Amazon EC2](#) for more information.

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
--------------	---------------	-----------------	----------------

Image option	Subscriptions	Sample scenario	Considerations
Deploy a Red Hat Gold Image.	Use your existing Red Hat subscriptions.	Select a Red Hat Gold Image on AWS. For details on Gold Images and how to access them on Azure, see the Red Hat Cloud Access Reference Guide .	The subscription includes the Red Hat product cost; you pay Amazon for all other instance costs. Red Hat provides support directly for Cloud Access images.
Deploy a custom image that you move to AWS.	Use your existing Red Hat subscriptions.	Upload your custom image, and attach your subscriptions.	The subscription includes the Red Hat product cost; you pay Amazon for all other instance costs. Red Hat provides support directly for custom RHEL images.
Deploy an existing Amazon image that includes RHEL.	The AWS EC2 images include a Red Hat product.	Select a RHEL image when you launch an instance on the AWS Management Console , or choose an image from the AWS Marketplace .	<p>You pay Amazon hourly on a <i>pay-as-you-go</i> model. Such images are called "on-demand" images. Amazon provides support for on-demand images.</p> <p>Red Hat provides updates to the images. AWS makes the updates available through the Red Hat Update Infrastructure (RHUI).</p>



NOTE

You can create a custom image for AWS using Red Hat Image Builder. See [Composing a Customized RHEL System Image](#) for more information.



IMPORTANT

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

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

Additional resources

- [Composing a Customized RHEL System Image](#)
- [AWS Management Console](#)
- [AWS Marketplace](#)

3.2. UNDERSTANDING BASE IMAGES

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

3.2.1. Using a custom base image

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

Additional resources

- [Red Hat Enterprise Linux](#)

3.2.2. Virtual machine configuration settings

Cloud VMs must have the following configuration settings.

Table 3.2. VM configuration settings

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

3.3. CREATING A BASE VM FROM AN ISO IMAGE

Follow the procedures in this section to create a RHEL 9 base image from an ISO image.

Prerequisites

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

3.3.1. Creating a VM from the RHEL ISO image

Procedure

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

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

- b. If you use the web console to create your VM, follow the procedure in [Creating virtual machines using the web console](#), with these caveats:
 - Do not check **Immediately Start VM**.
 - Change your **Memory** size to your preferred settings.
 - Before you start the installation, ensure that you have changed **Model** under **Virtual Network Interface Settings** to **virtio** and change your **vCPUs** to the capacity settings you want for the VM.

3.3.2. Completing the RHEL installation

Perform the following steps to complete the installation and to enable root access once the VM launches.

Procedure

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

5. Reboot the VM and log in as **root** once the installation completes.
6. Configure the image.
 - a. Register the VM and enable the Red Hat Enterprise Linux 9 repository.

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

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

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

7. **Important: This step is only for VMs you intend to upload to AWS.**

- a. For AMD64 or Intel 64 (x86_64) VMs, install the **nvme**, **xen-netfront**, and **xen-blkfront** drivers.

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

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

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

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

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

8. Power down the VM.

Additional resources

- [Understanding autoattaching subscriptions on the Customer Portal](#)
- [Introduction to cloud-init](#)

3.4. UPLOADING THE RED HAT ENTERPRISE LINUX IMAGE TO AWS

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

3.4.1. Installing the AWS CLI

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

Prerequisites

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

Procedure

1. Install the [AWS command line tools](#) using the **dnf** command.

```
# dnf install awscli
```

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

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

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

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

Additional resources

- [Quickly Configuring the AWS CLI](#)
- [AWS command line tools](#)

3.4.2. Creating an S3 bucket

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

Procedure

1. Launch the [Amazon S3 Console](#).
2. Click **Create Bucket**. The **Create Bucket** dialog appears.
3. In the **Name and region** view:
 - a. Enter a **Bucket name**.
 - b. Enter a **Region**.
 - c. Click **Next**.
4. In the **Configure options** view, select the desired options and click **Next**.
5. In the **Set permissions** view, change or accept the default options and click **Next**.
6. Review your bucket configuration.
7. Click **Create bucket**.



NOTE

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

Additional resources

- [Amazon S3 Console](#)
- [AWS CLI Command Reference](#)

3.4.3. Creating the vmimport role

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

Procedure

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

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

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

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

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

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketLocation",
```



```

        "s3:GetObject",
        "s3:ListBucket"
    ],
    "Resource": [
        "arn:aws:s3:::s3-bucket-name",
        "arn:aws:s3:::s3-bucket-name/*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "ec2:ModifySnapshotAttribute",
        "ec2:CopySnapshot",
        "ec2:RegisterImage",
        "ec2:Describe*"
    ],
    "Resource": "*"
}
]
}

```

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

```

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

```

Additional resources

- [VM Import Service Role](#)
- [Required Service Role](#)

3.4.4. Converting and pushing your image to S3

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

Procedure

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

```

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

```

2. Push the image to S3.

```

$ aws s3 cp rhel-9.0-sample.raw s3://s3-bucket-name

```

**NOTE**

This procedure could take a few minutes. After completion, you can check that your image uploaded successfully to your S3 bucket using the [AWS S3 Console](#).

Additional resources

- [How VM Import/Export Works](#)
- [AWS S3 Console](#)

3.4.5. Importing your image as a snapshot

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

Procedure

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

```
{
  "Description": "rhel-9.0-sample.raw",
  "Format": "raw",
  "UserBucket": {
    "S3Bucket": "s3-bucket-name",
    "S3Key": "s3-key"
  }
}
```

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

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

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

```
{
  "SnapshotTaskDetail": {
    "Status": "active",
    "Format": "RAW",
    "DiskImageSize": 0.0,
    "UserBucket": {
      "S3Bucket": "s3-bucket-name",
      "S3Key": "rhel-9.0-sample.raw"
    },
    "Progress": "3",
    "StatusMessage": "pending"
  },
  "ImportTaskId": "import-snap-06cea01fa0f1166a8"
}
```

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

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

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

Additional resources

- [Amazon S3 Console](#)
- [Importing a Disk as a Snapshot Using VM Import/Export](#)

3.4.6. Creating an AMI from the uploaded snapshot

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

Procedure

1. Go to the AWS EC2 Dashboard.
2. Under **Elastic Block Store**, select **Snapshots**.
3. Search for your snapshot ID (for example, **snap-0e718930bd72bcd0**).
4. Right-click on the snapshot and select **Create image**.
5. Name your image.
6. Under **Virtualization type**, choose **Hardware-assisted virtualization**.
7. Click **Create**. In the note regarding image creation, there is a link to your image.
8. Click on the image link. Your image shows up under **Images>AMIs**.

NOTE

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

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

You must specify the root device volume **/dev/sda1** as your **root-device-name**. For conceptual information about device mapping for AWS, see [Example block device mapping](#).

3.4.7. Launching an instance from the AMI

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

Procedure

1. From the AWS EC2 Dashboard, select **Images** and then **AMIs**.
2. Right-click on your image and select **Launch**.
3. Choose an **Instance Type** that meets or exceeds the requirements of your workload.
See [Amazon EC2 Instance Types](#) for information about 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 about tagging.

7. Click **Next: Configure Security Group**. Select the security group you created when [setting up your AWS environment](#).
8. Click **Review and Launch**. Verify your selections.
9. Click **Launch**. You are prompted to select an existing key pair or create a new key pair. Select the key pair you created when [setting up your AWS environment](#).



NOTE

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

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

**NOTE**

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

Additional resources

- [AWS Management Console](#)
- [Setting Up with Amazon EC2](#)
- [Amazon EC2 Instances](#)
- [Amazon EC2 Instance Types](#)

3.4.8. Attaching Red Hat subscriptions

To attach your Red Hat subscription to a RHEL instance, use the following steps.

Prerequisites

- You must have enabled your subscriptions.

Procedure

1. Register your system.

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

2. Attach your subscriptions.

- You can use an activation key to attach subscriptions. See [Creating Red Hat Customer Portal Activation Keys](#) for more information.
- Alternatively, you can manually attach a subscription using the ID of the subscription pool (Pool ID). See [Attaching and Removing Subscriptions Through the Command Line](#).

Additional resources

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

3.4.9. Setting up automatic registration on AWS Gold Images

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

Prerequisites

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



NOTE

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

Procedure

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

Verification

- In a RHEL 9 VM created using the above instructions, verify the system is registered to RHSM by executing the **subscription-manager identity** command. On a successfully registered system, this displays the UUID of the system. For example:

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

Additional resources

- [AWS Management Console](#)
- [Configuring cloud sources for Red Hat services](#)

3.5. 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](#)

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

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

This chapter includes:

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

Prerequisites

- Sign up for a [Red Hat Customer Portal](#) account.
- Sign up for AWS and set up your AWS resources. See [Setting Up with Amazon EC2](#) for more information.

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

A high-availability (HA) cluster is a set of computers (called *nodes*) that are linked together to run a specific workload. The purpose of HA clusters is to provide redundancy in case of a hardware or software failure. If a node in the HA cluster fails, the Pacemaker cluster resource manager distributes the workload to other nodes and no noticeable downtime occurs in the services that are running on the cluster.

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

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

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

Additional resources

- [High Availability Add-On overview](#)

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

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

Complete the following steps to create these keys.

Prerequisites

- Your IAM user account must have Programmatic access. See [Setting up the AWS Environment](#) for more information.

Procedure

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

4.3. INSTALLING THE AWS CLI

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

Prerequisites

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

Procedure

1. Install the [AWS command line tools](#) using the **dnf** command.

```
# dnf install awscli
```

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

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


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

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

Additional resources

- [Quickly Configuring the AWS CLI](#)
- [AWS command line tools](#)

4.4. CREATING AN HA EC2 INSTANCE

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

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

Prerequisites

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

Procedure

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



NOTE

Do not launch into an Auto Scaling Group.

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

**NOTE**

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

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

**NOTE**

Tags can help you manage your AWS resources. See [Tagging Your Amazon EC2 Resources](#) for information about tagging.

7. Click **Next: Configure Security Group**. Select the existing security group you created in [Setting up the AWS environment](#).
8. Click **Review and Launch** and verify your selections.
9. Click **Launch**. You are prompted to select an existing key pair or create a new key pair. Select the key pair you created when [Setting up the AWS environment](#).
10. Click **Launch Instances**.
11. Click **View Instances**. You can name the instance(s).

**NOTE**

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

Additional resources

- [AWS Management Console](#)
- [Setting Up with Amazon EC2](#)
- [Amazon EC2 Instances](#)
- [Amazon EC2 Instance Types](#)

4.5. CONFIGURING THE PRIVATE KEY

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

Procedure

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

2. Change the permissions of the key file so that only the root user can read it.

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

4.6. CONNECTING TO AN EC2 INSTANCE

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

Procedure

1. Launch the [AWS Console](#) and select the EC2 instance.
2. Click **Connect** and select **A standalone SSH client**.
3. From your SSH terminal session, connect to the instance using the AWS example provided in the pop-up window. Add the correct path to your **KeyName.pem** file if the path is not shown in the example.

4.7. INSTALLING THE HIGH AVAILABILITY PACKAGES AND AGENTS

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

Procedure

1. Remove the AWS Red Hat Update Infrastructure (RHUI) client.

```
$ sudo -i
# dnf -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 9 Server HA repositories.

```
# subscription-manager repos --enable=rhel-9-for-x86_64-highavailability-rpms
```

5. Update the RHEL AWS instance.

```
# dnf update -y
```

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

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

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

```
# passwd hacluster
```

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

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

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

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

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

Verification

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

```
# systemctl status pcsd.service

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

4.8. CREATING A CLUSTER

Complete the following steps to create the cluster of nodes.

Procedure

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

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

Example:

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

2. Create the cluster.

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

Example:

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

[...]

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

Verification

1. Enable the cluster.

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

2. Start the cluster.

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

4.9. CONFIGURING FENCING

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

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

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

Standard procedure

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

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

Example:

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

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

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

Example:

```
[root@ip-10-0-0-48 ~]# pcs stonith \
  create clusterfence fence_aws access_key=AKIAI123456MRMJA
  secret_key=a75EYIG4RVL3hdsdAslK7koQ8dzaDyn5yoIZ/ \
  region=us-east-1 pcmk_host_map="ip-10-0-0-48:i-07f1ac63af0ec0ac6;ip-10-0-0-46:i-063fc5fe93b4167b2;ip-10-0-0-58:i-08bd39eb03a6fd2c7" \
  power_timeout=240 pcmk_reboot_timeout=480 pcmk_reboot_retries=4
```

Alternate procedure

1. Obtain the VPC ID of the cluster.

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

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

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

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

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

```
[root@nodea ~]# pcs stonith config fence${CLUSTER}
```

```
Resource: clustername (class=stonith type=fence_aws)
Attributes: access_key=XXXXXXXXXXXXXXXXXXXXX pcmk_host_map=nodea:i-
0b02af8927a895137;nodeb:i-0cceb4ba8ab743b69;nodec:i-0502291ab38c762a5;
pcmk_reboot_retries=4 pcmk_reboot_timeout=480 power_timeout=240 region=xx-xxx-x
secret_key=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Operations: monitor interval=60s (clustername-monitor-interval-60s)
```

Verification

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

```
# pcs stonith fence awsnodename
```



NOTE

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

Example:

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

```
Node: ip-10-0-0-58 fenced
```

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

```
# pcs status
```

Example:

```
[root@ip-10-0-0-48 ~]# pcs status
```

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

```
3 nodes configured
1 resource configured
```

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

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

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

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

```
# pcs cluster start awshostname
```

4. Check the status to verify the node started.

```
# pcs status
```

Example:

```
[root@ip-10-0-0-48 ~]# pcs status

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

3 nodes configured
1 resource configured

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

Full list of resources:

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

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

4.10. INSTALLING THE AWS CLI ON CLUSTER NODES

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

Complete the following procedure on each cluster node.

Prerequisites

- You must have created an AWS Access Key and AWS Secret Access Key. See [Creating the AWS Access Key and AWS Secret Access Key](#) for more information.

Procedure

1. Install the AWS CLI. For instructions, see [Installing the AWS CLI](#).
2. Verify that the AWS CLI is configured properly. The instance IDs and instance names should display.

Example:

```
[root@ip-10-0-0-48 ~]# aws ec2 describe-instances --output text --query
'Reservations[].Instances[].[InstanceId,Tags[?Key==Name].Value]'
```



```
i-07f1ac63af0ec0ac6
ip-10-0-0-48
i-063fc5fe93b4167b2
ip-10-0-0-46
i-08bd39eb03a6fd2c7
ip-10-0-0-58
```

4.11. INSTALLING NETWORK RESOURCE AGENTS

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

Add the two resources to the [same group](#) to enforce **order** and **colocation** constraints.

Create a secondary private IP resource and virtual IP resource

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

Procedure

1. 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. 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

- 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. View the AWS Secondary Elastic IP Address resource agent (awseip) description. The following command shows the options and default operations for this agent.

pcs resource describe awseip

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

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

Example:

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

Verification

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

```
# pcs status
```

Example:

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

3 nodes configured
4 resources configured

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

Full list of resources:

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

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

Test the elastic IP address

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

Procedure

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

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

Example:

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

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

Additional resources

- [High Availability Add-On overview](#)
- [Configuring and managing high availability clusters](#)

4.12. CONFIGURING SHARED BLOCK STORAGE

To configure shared block storage for a Red Hat High Availability cluster with Amazon Elastic Block Storage (EBS) Multi-Attach volumes, use the following procedure. Note that this procedure is optional, and the steps below assume three instances (a three-node cluster) with a 1 TB shared disk.

Prerequisites

- You must be using an [AWS Nitro System-based Amazon EC2 instance](#).

Procedure

1. Create a shared block volume using the AWS command [create-volume](#).

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

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

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

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



NOTE

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

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

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

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

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

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

Verification

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

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

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

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

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

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

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

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

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

nodea
E: ID_SERIAL=Amazon Elastic Block Store_vol0fa5342e7aedef09f7
```

Additional resources

- [Configuring a GFS2 file system in a cluster](#)
- [Configuring GFS2 file systems](#)

4.13. 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](#)