Red Hat Enterprise Linux 10

Deploying and managing RHEL on Amazon Web Services

Obtaining Red Hat Enterprise Linux system images and creating RHEL instances on AWS

Last Updated: 2025-05-22

Red Hat Enterprise Linux 10 Deploying and managing RHEL on Amazon Web Services

Obtaining Red Hat Enterprise Linux system images and creating RHEL instances on AWS

Legal Notice

Abstract

To use Red Hat Enterprise Linux (RHEL) in a public cloud environment, you can create and deploy RHEL system images on various cloud platforms, including 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

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

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

Submitting feedback through Jira (account required)

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

CHAPTER 1. INTRODUCING RHEL ON PUBLIC CLOUD PLATFORMS

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

1.1. 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.

1.2. FREQUENT CONCERNS WHEN MIGRATING TO A PUBLIC CLOUD

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

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

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

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

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

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

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

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

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

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

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

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

1.3. OBTAINING RHEL FOR PUBLIC CLOUD DEPLOYMENTS

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

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

NOTE

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

2. Create a RHEL cloud instance on your chosen cloud platform. For details, see Methods for creating RHEL cloud instances.

3. To keep your RHEL deployment up-to-date, use Red Hat Update Infrastructure (RHUI).

Additional resources

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

1.4. METHODS FOR CREATING RHEL CLOUD INSTANCES

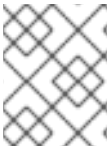
To deploy a RHEL instance on a public cloud platform, you can use one of the following methods:

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

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

Purchase a RHEL instance directly from the cloud provider marketplace

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



NOTE

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

Additional resources

- What is a golden image?

CHAPTER 2. PREPARING AND UPLOADING AMI IMAGES TO AWS

You can create custom images and can update them, either manually or automatically, to the AWS cloud with RHEL image builder.

2.1. PREPARING TO MANUALLY 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 python3-pip
```

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

```
# pip3 install awscli
```

3. 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 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:

a. Create a **trust-policy.json** file with the trust policy configuration, in the JSON format. For example:

```
{
   "Version": "2022-10-17",
   "Statement": [{
      "Effect": "Allow",
```

```
        "Principal": {
           "Service": "vmie.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
        "Condition": {
           "StringEquals": {
              "sts:Externalid": "vmimport"
           }
        }
     }]
}
```

b. Create a **role-policy.json** file with the role policy configuration, in the JSON format. For example:

```
{
   "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
```

c. Create a role for your Amazon Web Services account, by using the **trust-policy.json** file:

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

d. Embed an inline policy document, by using the **role-policy.json** file:

```
$ 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. MANUALLY UPLOADING AN AMI IMAGE TO AWS BY USING THE CLI

You can use RHEL image builder to build **ami** images and manually upload them directly to Amazon AWS Cloud service provider, by 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.

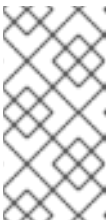
2. Save the file as *CONFIGURATION-FILE*.toml and close the text editor.

3. Start the compose to upload it to AWS:

```
# 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 AWS Identity and Access Management (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:

```
# 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. CREATING AND AUTOMATICALLY UPLOADING IMAGES TO THE AWS CLOUD AMI

You can create a **.raw** image by using RHEL image builder, and choose to check the **Upload to AWS** checkbox to automatically 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 RHEL image builder interface of the RHEL web console in a browser.
- You have created a blueprint. See Creating a 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. In the RHEL image builder dashboard, click the **blueprint name** that you previously created.
2. Select the tab **Images**.
3. Click **Create Image** to create your customized image.
   The **Create Image** window opens.
   a. From the **Type** drop-down menu list, select **Amazon Machine Image Disk (.raw)**.
   b. Check the **Upload to AWS** checkbox 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, 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 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. In a browser, access Service→EC2.

   a. On the AWS console dashboard menu, choose the correct region. The image must have the **Available** status, to indicate that it is uploaded.

   b. On the AWS dashboard, select your image and click **Launch**.

6. A new window opens. Choose an instance type according to the resources you need to start your image. Click **Review and Launch**.

7. Review your instance start details. You can edit each section if you need to make any changes. Click **Launch**.

8. 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.

9. Then, you can click **Launch Instance** to start your instance.
   You can check the status of the instance, which displays as **Initializing**.

10. After the instance status is **running**, the **Connect** button becomes available.

11. Click **Connect**. A window is displayed with instructions on how to connect by 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 by 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 over SSH.

Verification

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

Additional resources

- Open a case on Red Hat Customer Portal
- Connecting to your Linux instance by using SSH

CHAPTER 3. DEPLOYING A RHEL IMAGE AS AN EC2 INSTANCE ON AWS

To use RHEL image on an Amazon Web Services (AWS), convert the RHEL image to the AWS-compatible format and deploy a VM from the RHEL image to run as an Elastic Cloud Compute (EC2) instance. To create, customize, and deploy a RHEL Amazon Machine Image (AMI), you can use one of the following methods:

- Use the Red Hat Image Builder. For instructions, see Preparing and uploading AMI images to AWS
- Manually create and configure an AMI. This is a more complicated process but offers more granular customization options. For instructions, see the following sections.

Prerequisites

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

3.1. AVAILABLE RHEL IMAGE TYPES FOR PUBLIC CLOUD

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

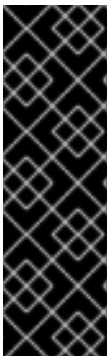
NOTE

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

Table 3.1. Image options

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

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



IMPORTANT

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

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

Additional resources

- Red Hat Ecosystem Cataglogue
- Red Hat Cloud Access Reference Guide
- Using gold images on Amazon Web Services (AWS)
- Amazon Web Services Red Hat Ecosystem Cataglogue
- Red Hat Enterprise Linux on Amazon EC2 FAQs

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

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

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

- Fully customizable
- High flexibility for any use case

- Lightweight - includes only the operating system and the required runtime libraries

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

NOTE

Vertify the following VM configurations.

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

Prerequisites

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

Procedure

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

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

6. 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"
```

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

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

Including these drivers prevents 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.

Verification

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

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

- Power down the VM.

3.3. UPLOADING A RHEL IMAGE TO AWS BY USING THE COMMAND LINE

To run a RHEL instance on Amazon Web Services (AWS), you must first upload a RHEL image to AWS. To configure and manage a RHEL EC2 instance on AWS, use the **awscli2** utility.

3.3.1. Installing AWSCLI2

You can use the AWS command line interface **awscli2** utility to configure and manage RHEL images and Red Hat high availability (HA) cluster on AWS.

Prerequisites

- You have access to an AWS Access Key ID and an AWS Secret Access Key. For details, see manage access keys.

Procedure

- Install **awscli2**:

```
# dnf install awscli2
```

Verification

1. Verify the installation:

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

2. Configure **awscli2** for AWS credentials and settings:

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

3.3.2. Converting and pushing an image to Amazon S3

You can convert a RHEL image in the **qcow2** image format to **OVA**, **VHD**, **VHDX**, **VMDK**, and **raw** by using the **qemu-img** utility, and then upload to the Amazon S3 storage. For details, see supported image formats by AWS.

Prerequisites

- You have created an Amazon S3 bucket by using awscli2 to upload the RHEL image.

Procedure

1. Run **qemu-img** to convert **.qcow2** image to **.raw** image format:

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

2. Upload the image to the Amazon S3 bucket:

```
$ aws s3 cp rhel-10.0-sample.raw s3://<example-s3-bucket-name>
```

Verification

- Monitor the AWS S3 Console to confirm successful upload.

Additional resources

- Create a S3 bucket

3.3.3. Managing a RHEL VM on AWS by using the command line

By using the **awscli2** utility, you can manage a RHEL EC2 VM on AWS through command line. In this case, you can use the **vmimport** role for managing RHEL EC2 image snapshot. With **awscli2**, you can also import a RHEL EC2 image snapshot, create an AMI, launch, and connect to a RHEL EC2 instance.

1. Use the **vmimport** role: An alternate to import the RHEL image to the Amazon S3 bucket is by using the **vimport** role. See Required permissions for VM Import/Export .

2. Import RHEL image as a snapshot: You can import RHEL VM image from Amazon S3 as a snapshot to Amazon EC2. For details, see Start an import snapshot task and Monitor an import snapshot task.

3. Create and launch a RHEL EC2 instance: You can create a RHEL Amazon Machine Image (AMI) from existing snapshot and launch a RHEL EC2 instance. For details, see create an AMI from snapshot by using awscli2 and launching, listing, and deleting RHEL instance by using awscli2 .

4. Configure the private key and connect to the RHEL EC2 instance: You can configure your ***<example-key>*.pem** file and connect to an RHEL EC2 instance. For details, see Create a key pair using Amazon EC2 and Connect using the AWS CLI.

Additional resources

- Amazon S3 general purpose buckets

- Amazon Web Services Red Hat Ecosystem Cataglogue

- Red Hat Enterprise Linux on Amazon EC2 FAQs

3.3.4. Attaching Red Hat subscriptions

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

Prerequisites

- You have an active Red Hat account.

Procedure

1. Register your system:

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

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

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

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

Additional resources

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

3.4. UPLOADING A RHEL IMAGE TO AWS BY USING THE AWS CONSOLE

To run a RHEL instance on Amazon Web Services (AWS), you must first upload the RHEL image to AWS. To configure and manage the RHEL EC2 instance on AWS, use the **awscli2** utility.

3.4.1. Converting and pushing an image to S3 by using the AWS console

You can convert a RHEL image in the **qcow2** image format to **OVA**, **VHD**, **VHDX**, **VMDK**, or **raw**, and upload it to an Amazon Elastic Cloud Computing (EC2) by using the **qemu-img** utility. For details, see supported image formats by AWS .

Prerequisites

- You have created an Amazon S3 bucket by using the Amazon S3 console to upload the RHEL image.

Procedure

1. Run **qemu-img** to convert **.qcow2** image to **.raw** image format:

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

2. Upload the image to the S3 bucket by using Amazon S3 console

Verification

- Monitor the AWS S3 Console to confirm successful upload.

Additional resources

- Create a S3 bucket

3.4.2. Managing a RHEL VM on AWS by using the AWS console

You can manage a RHEL EC2 VM on AWS by using the AWS console. You can create RHEL EC2 image snapshot, manage Amazon Machine Image (AMI), launch, and connect to a RHEL EC2 instance.

Procedure

1. Use the **vmimport** role: An alternate way to import the RHEL image to the Amazon S3 bucket is by using the **vmimport** role. See Import your VM as an image .

2. Import a RHEL image as a snapshot: You can import a RHEL VM image from Amazon S3 as a snapshot to Amazon EC2. For details, see Importing a disk as a snapshot using VM Import/Export and Monitor an import snapshot task .

3. Create and launch a RHEL EC2 instance: You can create a RHEL Amazon Machine Image (AMI) from an existing snapshot and launch a RHEL EC2 instance. For details, see Create an AMI from a snapshot and Launch an instance using defined parameters .

4. Configure the private key and connect to the RHEL EC2 instance: You can configure your ***<example-key>*.pem** file and connect to an RHEL EC2 instance. For details, see Create a key pair using Amazon EC2 and Connect using the Amazon EC2 console .

5. For Red Hat susbscriptions, see Attaching Red Hat subscriptions

Additional resources

- Amazon S3 general purpose buckets
- Red Hat Cloud Access Reference Guide
- Red Hat Enterprise Linux on Amazon EC2 - FAQs
- Red Hat on Amazon Web Services

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

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

To configure a Red Hat HA cluster on AWS that uses EC2 instances as cluster nodes, see the following sections. You have a number of options for obtaining RHEL images for the cluster. For details, see Available RHEL image types for public cloud .

Prerequisites

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

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

- **Pacemaker**, a cluster resource manager that provides both a command line utility (**pcs**) and a GUI (**pcsd**) to support multiple nodes
- **Corosync** and **Kronosnet** to create and manage HA clusters
- Resource agents to configure and manage custom applications

- Fencing agents to use cluster on platforms like bare-metal servers and virtual machines

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

4.2. INSTALLING THE HIGH AVAILABILITY PACKAGES AND AGENTS

On each of the nodes, you need to install the High Availability packages and agents to be able to configure a Red Hat High Availability cluster on AWS.

Prequisites

- You have completed the configuration for Uploading RHEL image to AWS by using the command line.

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 10 Server HA repositories.

```
# subscription-manager repos --enable=rhel-10-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 the AWS fencing agent 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

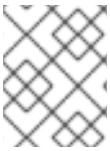
- Ensure the **pcs** service is running.

```
# 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.
```

4.3. CREATING A HIGH AVAILABILITY CLUSTER

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



NOTE

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

Procedure

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

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

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

option, which will start the cluster services on both nodes in the cluster.

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

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

NOTE

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

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

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

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

...
```

4.4. CONFIGURING FENCING IN A RED HAT HIGH AVAILABILITY CLUSTER

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

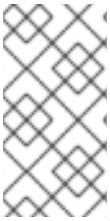
Without a fence device configured you do not have a way to know that the resources previously used by the disconnected cluster node have been released, and this could prevent the services from running on any of the other cluster nodes. Conversely, the system may assume erroneously that the cluster node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

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

High Availability cluster, see the Red Hat Knowledgebase solution Fencing in a Red Hat High Availability Cluster.

4.4.1. Displaying available fence agents and their options

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



NOTE

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

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

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

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

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

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

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

WARNING

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

4.4.2. Creating a fence device

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

```
pcs stonith create stonith_id stonith_device_type [stonith_device_options] [op operation_action operation_options]
```

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

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

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

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

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

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

4.4.3. General properties of fencing devices

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

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

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

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

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

Table 4.1. General Properties of Fencing Devices

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

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

Table 4.2. Advanced Properties of Fencing Devices

Field	Type	Default	Description

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

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

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

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

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

Table 4.3. Cluster Properties that Determine Fencing Behavior

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

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

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

4.4.4. Fencing delays

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

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

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

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

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

The pcmk_delay_base cluster property

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

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

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

The pcmk_delay_max cluster property

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

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

The priority-fencing-delay cluster property

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

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

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

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

Interaction of fencing delays

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

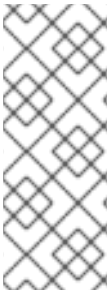
- Any delay set with the **priority-fencing-delay** property is added to any delay from the **pcmk_delay_base** and **pcmk_delay_max** fence device properties. This behavior allows some delay when both nodes have equal priority, or both nodes need to be fenced for some reason other than node loss, as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

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

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

4.4.5. Testing a fence device

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



NOTE

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

Procedure

Use the following procedure to test a fence device.

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

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

NOTE

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

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

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

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

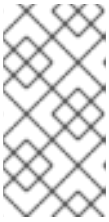
```
# fence_ipmilan -a ipaddress -l username -p password -o reboot
```

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

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

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

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



NOTE

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

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

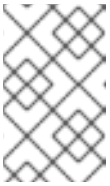
manually and the cluster has been started, test fencing with the **pcs stonith fence** command from any node (or even multiple times from different nodes), as in the following example. The **pcs stonith fence** command reads the cluster configuration from the CIB and calls the fence agent as configured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

```
# pcs stonith fence node_name
```

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

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

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

     NOTE

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

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

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

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

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

4.4.6. Configuring fencing levels

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

Pacemaker processes fencing levels as follows:

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

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

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

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

Prerequisites

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

Procedure

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

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

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

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

3. List the currently configured fencing levels.

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

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

4.4.6.1. Removing a fence level

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

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

4.4.6.2. Clearing fence levels

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

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

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

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

4.4.6.3. Verifying nodes and devices in fence levels

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

```
pcs stonith level verify
```

4.4.6.4. Specifying nodes in fencing topology

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

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

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

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

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

4.4.7. Configuring fencing for redundant power supplies

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

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

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

Procedure

1. Create the first fence device.

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

2. Create the second fence device.

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

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

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

4.4.8. Administering fence devices

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

4.4.8.1. Displaying configured fence devices

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

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

4.4.8.2. Exporting fence devices as pcs commands

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

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

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

4.4.8.3. Exporting fence level configuration

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

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

4.4.8.4. Modifying and deleting fence devices

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

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

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

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

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

```
pcs stonith delete stonith_id
```

4.4.8.5. Manually fencing a cluster node

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

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

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

down and free its resources for recovery.

WARNING

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

```
pcs stonith confirm node
```

4.4.8.6. Disabling a fence device

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

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

```
# pcs stonith disable myapc
```

4.4.8.7. Preventing a node from using a fencing device

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

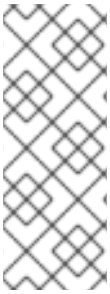
The following example prevents fence device **node1-ipmi** from running on **node1**.

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

4.4.9. Configuring ACPI for use with integrated fence devices

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

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



NOTE

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

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

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

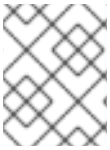
- Setting **HandlePowerKey=ignore** in the **/etc/systemd/logind.conf** file and verifying that the node node turns off immediately when fenced, as described in Disabling ACPI soft-off in the logind.conf file. This is the first alternate method of disabling ACPI Soft-Off.
- Appending **acpi=off** to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

IMPORTANT

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

4.4.9.1. Disabling ACPI Soft-Off with the BIOS

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



NOTE

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

Procedure

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

NOTE

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

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

BIOS CMOS Setup Utility:

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

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

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

4.4.9.2. Disabling ACPI Soft-Off in the logind.conf file

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

Procedure

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

```
HandlePowerKey=ignore
```

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

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

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

4.4.9.3. Disabling ACPI completely in the GRUB 2 file

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

IMPORTANT

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

Procedure

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

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

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

2. Reboot the node.

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

4.5. SETTING UP IP ADDRESS RESOURCES ON AWS

Clients use IP addresses to manage cluster resources across the network. To handle a failover of a cluster, include *IP address resources* in the cluster that use specific network resource agents. The RHEL HA Add-On provides a set of resource agents, which create IP address resources to manage various types of IP addresses on AWS. To decide which resource agent to configure, consider the type of AWS IP addresses in the HA cluster management. It includes the following ways to create a cluster resource for managing an IP address:

- Exposed to the internet: Use the **awseip** network resource.
- Limited to a single AWS Availability Zone (AZ): Use the **awsvip** and **IPaddr2** network resources.
- Reassigns to multiple AWS AZs within the same AWS region: Use the **aws-vpc-move-ip** network resource.

NOTE

If the HA cluster does not manage any IP addresses, the resource agents for managing virtual IP addresses on AWS are not required. If you need further guidance for your specific deployment, consult with AWS.

4.5.1. Creating an IP address resource to manage an IP address exposed to the internet

To ensure that high-availability (HA) clients can access a RHEL node that uses public-facing internet connections, configure an *AWS Secondary Elastic IP Address* (**awseip**) resource to use an elastic IP address.

Prerequisites

- You have a previously configured cluster.
- Your cluster nodes must have access to the RHEL HA repositories. For details, see Installing the High Availability packages and agents.
- You have set up the AWS CLI2. For details, see Installing AWSCLI2.

Procedure

1. Install the **resource-agents** package:

```
# dnf install resource-agents
```

2. 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. Optional: Display the description of **awseip**. This shows the options and default operations for this agent.

```
# pcs resource describe awseip
```

4. Create the Secondary Elastic IP address resource with the allocated IP address in the 2nd step:

```
# pcs resource create <resource-id> 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

1. Verify the cluster status to ensure resources are available:

```
[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
     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
```

In this example, **newcluster** is am active cluster where resources such as **vip** and **elastic** are part of the **networking-group** resource group.

2. Launch an SSH session from your local workstation to the elastic IP address that you previously created:

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

3. Verify that the SSH connected host is same as the host with the elastic resources.

4.5.2. Creating an IP address resource to manage a private IP address limited to a single AWS Availability Zone

You can configure an *AWS Secondary Private IP Address* (**awsvip**) resource to use a virtual IP address. With **awsvip**, high-availability (HA) clients on AWS can access a RHEL node to use a private IP address accessible in only a single availability zone (AZ). You can complete the following procedure on any node in the cluster.

Prerequisites

- You have a previously configured cluster.
- Your cluster nodes have access to the RHEL HA repositories. For details,see Installing the High Availability packages and agents.
- You have set up the AWS CLI. For instructions, see Installing AWSCLI2.

Procedure

1. Install the **resource-agents** package.

```
# dnf install resource-agents
```

2. Optional: View the options and default operations for **awsvip**:

```
# pcs resource describe awsvip
```

3. Create a Secondary Private IP address with an unused private IP address in the **VPC CIDR** block:

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

Here, secondary private IP address is a part of gets included in a resource group

4. Create a virtual IP resource with the **vip** resource ID and the **networking-group** group name:

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

This is a VPC IP address that maps from the fence node to the failover node, masking the failure of the fence node within the subnet. Ensure that the virtual IP belongs to the same resource group as the Secondary Private IP address you created in the previous step.

Verification

- Verify the cluster status to ensure resources are available:

```
[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
```

In this example, **newcluster** is an active cluster where resources such as **vip** and **elastic** are part of the **networking-group** resource group.

4.5.3. Creating an IP address resource to manage an IP address that can move across multiple AWS Availability Zones

You can configure The RHEL Overlay IP (**aws-vpc-move-ip**) resource agent to use an elastic IP address. With **aws-vpc-move-ip**, high-availability (HA) clients on AWS manage a RHEL node that can be moved across multiple availability zones (az) in a single region of AWS.

Prerequisites

- You have a previously configured cluster.
- Your cluster nodes have access to the RHEL HA repositories. For more information, see Installing the High Availability packages and agents .
- You have set up the AWS CLI. For instructions, see Installing AWSCLI2.
- An Identity and Access Management (IAM) user is configured on your cluster and has the following permissions:
  - Modify routing tables
  - Create security groups
  - Create IAM policies and roles

Procedure

1. Install the **resource-agents** package:

```
# dnf install resource-agents
```

2. Optional: View the options and default operations for **awsvip**:

```
# pcs resource describe aws-vpc-move-ip
```

3. Set up an **OverlayIPAgent** IAM policy for the IAM user.
   a. In the AWS console, navigate to **Services → IAM → Policies → Create OverlayIPAgent Policy**
   b. Input the following configuration, and change the *<region>*, *<account-id>*, and *<ClusterRouteTableID>* values to correspond with your cluster.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Stmt1424870324000",
         "Effect": "Allow",
         "Action":  "ec2:DescribeRouteTables",
         "Resource": "*"
      },
      {
         "Sid": "Stmt1424860166260",
         "Action": [
            "ec2:CreateRoute",
            "ec2:ReplaceRoute"
         ],
         "Effect": "Allow",
         "Resource": "arn:aws:ec2:<region>:<account-id>:route-table/<ClusterRouteTableID>"
      }
   ]
}
```

4. In the AWS console, disable the **Source/Destination Check** function on all nodes in the cluster. To do this, right-click each node → **Networking** → **Change Source/Destination Checks**. In the pop-up message that appears, click **Yes, Disable**.

5. Create a route table for the cluster. To do so, use the following command on one node in the cluster:

```
# aws ec2 create-route --route-table-id <ClusterRouteTableID> --destination-cidr-block
<NewCIDRblockIP/NetMask> --instance-id <ClusterNodeID>
```

   In the command, replace values as follows:

   - **ClusterRouteTableID**: The route table ID for the existing cluster VPC route table.
   - **NewCIDRblockIP/NetMask**: A new IP address and netmask outside of the VPC classless inter-domain routing (CIDR) block. For example, if the VPC CIDR block is **172.31.0.0/16**, the new IP address/netmask can be **192.168.0.15/32**.
   - **ClusterNodeID**: The instance ID for another node in the cluster.

6. On one of the nodes in the cluster, create a **aws-vpc-move-ip** resource that uses a free IP address that is accessible to the client. The following example creates a resource named **vpcip** that uses IP **192.168.0.15**.

```
# pcs resource create vpcip aws-vpc-move-ip ip=192.168.0.15 interface=eth0
routing_table=<ClusterRouteTableID>
```

7. On all nodes in the cluster, edit the /**etc**/**hosts**/ file, and add a line with the IP address of the newly created resource. For example:

```
192.168.0.15 vpcip
```

Verification

1. Test the failover ability of the new **aws-vpc-move-ip** resource:

```
# pcs resource move vpcip
```

2. If the failover succeeded, remove the automatically created constraint after the move of the **vpcip** resource:

```
# pcs resource clear vpcip
```

Additional resources

- Configure the OverlayIP Resource Agent (Red Hat Knowledgebase)
- IAM users

4.6. CONFIGURING SHARED BLOCK STORAGE

To create extra storage resources, you can use Amazon Elastic Block Storage (EBS) Multi-Attach volumes to configure shared block storage for a Red Hat High Availability cluster.

Prerequisites

- You have 3 RHEL instances running in a cluster as a three-node cluster with a 1 TB shared disk.
- You have access to an AWS Nitro System-based Amazon EC2 instance .

Procedure

1. Create a shared block 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"
}
```

2. For each instance in your cluster, attach a shared block volume. Use your **<instance_id>** and **<volume_id>**, for example **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. Verify the block device is available by using the **ssh** command with your instance IP **198.51.100.3** for each instance in the cluster:

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

node a
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 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
```