Red Hat Enterprise Linux 10-beta

Configuring and managing cloud-init for RHEL

Using cloud-init to automate the initialization of cloud instances

Last Updated: 2025-03-04

Red Hat Enterprise Linux 10-beta Configuring and managing cloud-init for RHEL

Using cloud-init to automate the initialization of cloud instances

Legal Notice

Abstract

You can efficiently create multiple cloud instances of RHEL by using the cloud-init package. This allows for consistent and repeatable deployment of RHEL on a variety of cloud platforms. In the following chapters, you can learn more about working of cloud-init, use cloud-init to initiate cloud instances, and Red Hat supported cloud-init use cases.

Table of Contents

RHEL BETA RELEASE

Red Hat provides Red Hat Enterprise Linux Beta access to all subscribed Red Hat accounts. The purpose of Beta access is to:

- Provide an opportunity to customers to test major features and capabilities prior to the general availability release and provide feedback or report issues.
- Provide Beta product documentation as a preview. Beta product documentation is under development and is subject to substantial change.

Note that Red Hat does not support the usage of RHEL Beta releases in production use cases. For more information, see the Red Hat Knowledgebase solution What does Beta mean in Red Hat Enterprise Linux and can I upgrade a RHEL Beta installation to a General Availability (GA) release?.

CHAPTER 1. INTRODUCING RHEL ON PUBLIC CLOUD PLATFORMS

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

1.1. BENEFITS OF USING RHEL IN A PUBLIC CLOUD

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

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

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

- **Space and cost efficiency**
  You do not need to own any on-premise servers to host cloud workloads. This avoids the space, power, and maintenance requirements associated with physical hardware.

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

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

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

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

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

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

Additional resources

- What is public cloud?

- What is a hyperscaler?
- Types of cloud computing
- Public cloud use cases for RHEL
- Obtaining RHEL for public cloud deployments

1.2. PUBLIC CLOUD USE CASES FOR RHEL

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

Beneficial use cases

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

Potentially problematic use cases

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

Next steps

- Obtaining RHEL for public cloud deployments

Additional resources

- Should I migrate my application to the cloud? Here's how to decide.

1.3. FREQUENT CONCERNS WHEN MIGRATING TO A PUBLIC CLOUD

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

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

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

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

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

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

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

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

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

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

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

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

1.4. OBTAINING RHEL FOR PUBLIC CLOUD DEPLOYMENTS

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

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

1.5. METHODS FOR CREATING RHEL CLOUD INSTANCES

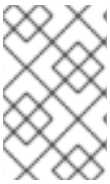
To deploy a RHEL instance on a public cloud platform, you can use one of the following methods:

Create a 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

see the following chapters in this document.

Additional resources

- What is a golden image?
- Configuring and managing cloud-init for RHEL

CHAPTER 2. INTRODUCTION TO CLOUD-INIT

The **cloud-init** utility automates the initialization of cloud instances during system boot. You can configure **cloud-init** to perform a variety of tasks:

- Configuring a host name
- Installing packages on an instance
- Running scripts
- Suppressing default virtual machine (VM) behavior

Prerequisites

- Sign up for a Red Hat Customer Portal account.

The **cloud-init** is available in various types of RHEL images. For example:

- If you download a KVM guest image from the Red Hat Customer Portal , the image comes preinstalled with the **cloud-init** package. After you launch the instance, the **cloud-init** package becomes enabled. KVM guest images on the Red Hat Customer Portal are intended to use with Red Hat Virtualization (RHV), the Red Hat OpenStack Platform (RHOSP), and Red Hat OpenShift Virtualization.
- You can also download the Red Hat ISO image from the Red Hat Customer Portal to create a custom guest image. In this case, you need to install the **cloud-init** package on the customized guest image.
- If you require to use an image from a cloud service provider (for example, AWS or Azure), use the *RHEL image builder* to create the image. Image builder images are customized for specific cloud providers. The following image types include **cloud-init** already installed:
  - Amazon Machine Image (AMI)
  - Virtual Hard Disk (VHD)
  - QEMU copy-on-write (qcow2)
    For details about the RHEL image builder, see Composing a customized RHEL system image.

Most cloud platforms support **cloud-init**, but configuration procedures and supported options vary. Alternatively, you can configure **cloud-init** for the NoCloud environment.

In addition, you can configure **cloud-init** on one VM and then use that VM as a template to create additional VMs or clusters of VMs.

Specific Red Hat products, for example, Red Hat Virtualization, have documented procedures to configure **cloud-init** for those products.

2.1. OVERVIEW OF THE CLOUD-INIT CONFIGURATION

The **cloud-init** utility uses YAML-formatted configuration files to apply user-defined tasks to instances. When an instance boots, the **cloud-init** service starts and executes the instructions from the YAML file. Depending on the configuration, tasks complete either during the first boot or on subsequent boots of the VM.

To define the specific tasks, configure the **/etc/cloud/cloud.cfg** file and add directives under the **/etc/cloud/cloud.cfg.d/** directory.

- The **cloud.cfg** file includes directives for various system configurations, such as user access, authentication, and system information.
  The file also includes default and optional modules for **cloud-init**. These modules execute in order in the following phases: .. The **cloud-init** initialization phase .. The configuration phase .. The final phase.

  + In the **cloud.cfg** file, the modules for the three phases are listed under **cloud_init_modules**, **cloud_config_modules**, and **cloud_final_modules** respectively.

- You can add additional directives for **cloud-init** in the **cloud.cfg.d** directory. When adding directives to the **cloud.cfg.d** directory, you need to add them to a custom file named ***.cfg**, and always include **#cloud-config** at the top of the file.

2.2. CLOUD-INIT OPERATES IN STAGES

During system boot, the **cloud-init** utility operates in five stages that determine whether **cloud-init** runs and where it finds its datasources, among other tasks. The stages are as follows:

1. **Generator stage**: By using the **systemd** service, this phase determines whether to run **cloud-init** utility at the time of boot.

2. **Local stage**: **cloud-init** searches local datasources and applies network configuration, including the DHCP-based fallback mechanism.

3. **Network stage**: **cloud-init** processes user data by running modules listed under **cloud_init_modules** in the **/etc/cloud/cloud.cfg** file. You can add, remove, enable, or disable modules in the **cloud_init_modules** section.

4. **Config stage**: **cloud-init** runs modules listed under **cloud_config_modules** section in the **/etc/cloud/cloud.cfg** file. You can add, remove, enable, or disable modules in the **cloud_config_modules** section.

5. **Final stage**: **cloud-init** runs modules and configurations included in the **cloud_final_modules** section of the **/etc/cloud/cloud.cfg** file. It can include the installation of specific packages, as well as triggering configuration management plug-ins and user-defined scripts. You can add, remove, enable, or disable modules in the **cloud_final_modules** section.

Additional resources

- Boot Stages of cloud-init

2.3. CLOUD-INIT MODULES EXECUTE IN PHASES

When **cloud-init** runs, it executes the modules within **cloud.cfg** in order within three phases:

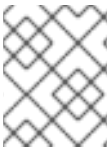
1. The network phase (**cloud_init_modules**)

2. The configuration phase (**cloud_config_modules**)

3. The final phase (**cloud_final_modules**)

When **cloud-init** runs for the first time on a VM, all the modules you have configured run in their respective phases. On a subsequent running of **cloud-init**, whether a module runs within a phase

depends on the *module frequency* of the individual module. Some modules run every time **cloud-init** runs; some modules only run the first time **cloud-init** runs, even if the instance ID changes.

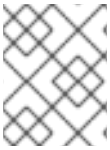


NOTE

An instance ID uniquely identifies an instance. When an instance ID changes, **cloud-init** treats the instance as a new instance.

The possible *module frequency* values are as follows:

- **Per instance** means that the module runs on first boot of an instance. For example, if you clone an instance or create a new instance from a saved image, the modules designated as per instance run again.
- **Per once** means that the module runs only once. For example, if you clone an instance or create a new instance from a saved image, the modules designated per once do not run again on those instances.
- **Per always** means the module runs on every boot.



NOTE

You can override a module's frequency when you configure the module or by using the command line.

2.4. CLOUD-INIT ACTS UPON USER DATA, METADATA, AND VENDOR DATA

The datasources that **cloud-init** consumes are user data, metadata, and vendor data.

- User data includes directives you specify in the **cloud.cfg** file and in the **cloud.cfg.d** directory, for example, user data can include files to run, packages to install, and shell scripts. Refer to the **cloud-init** Documentation section User-Data Formats for information about the types of user data that **cloud-init** allows.
- Metadata includes data associated with a specific datasource, for example, metadata can include a server name and instance ID. If you are using a specific cloud platform, the platform determines where your instances find user data and metadata. Your platform may require that you add metadata and user data to an HTTP service; in this case, when **cloud-init** runs it consumes metadata and user data from the HTTP service.
- Vendor data is optionally provided by the organization (for example, a cloud provider) and includes information that can customize the image to better fit the environment where the image runs. **cloud-init** acts upon optional vendor data and user data after it reads any metadata and initializes the system. By default, vendor data runs on the first boot. You can disable vendor data execution.
  Refer to the **cloud-init** Documentation section Instance Metadata for a description of metadata; Datasources for a list of datasources; and Vendor Data for more information about vendor data.

2.5. CLOUD-INIT IDENTIFIES THE CLOUD PLATFORM

cloud-init attempts to identify the cloud platform using the script **ds-identify**. The script runs on the first boot of an instance.

Adding a datasource directive can save time when **cloud-init** runs. You would add the directive in the **/etc/cloud/cloud.cfg** file or in the **/etc/cloud/cloud.cfg.d** directory. For example:

```
datasource_list:[Ec2]
```

Beyond adding the directive for your cloud platform, you can further configure **cloud-init** by adding additional configuration details, such as metadata URLs.

```
datasource_list: [Ec2]
datasource:
  Ec2:
    metadata_urls: ['http://169.254.169.254']
```

After **cloud-init** runs, you can view a log file (**run/cloud-init/ds-identify.log**) that provides detailed information about the platform.

Additional resources

- Datasources
- How to identify the datasource I'm using
- How can I debug my user data?

2.6. ADDITIONAL RESOURCES

- Upstream documentation for cloud-init