Red Hat Enterprise Linux 8

Configuring basic system settings

Set up the essential functions of your system and customize your system environment

Last Updated: 2024-04-09

Red Hat Enterprise Linux 8 Configuring basic system settings

Set up the essential functions of your system and customize your system environment

Legal Notice

Abstract

Perform basic system administration tasks, configure the environment settings, register your system, and configure network access and system security. Administer users, groups, and file permissions. Use system roles for managing system configurations interface on multiple RHEL systems. Use systemd for efficient service management. Configure the Network Time Protocol (NTP) with chrony. Backup and restore your system using ReaR. Install and use dynamic programming languages such as Python 3 and PHP.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE

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

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

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

Submitting 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. CONFIGURING AND MANAGING BASIC NETWORK ACCESS

This section describes only basic options on how to configure network settings in Red Hat Enterprise Linux.

1.1. CONFIGURING THE NETWORK AND HOST NAME IN THE GRAPHICAL INSTALLATION MODE

Follow the steps in this procedure to configure your network and host name.

Procedure

1. From the **Installation Summary** window, click **Network and Host Name**.
2. From the list in the left-hand pane, select an interface. The details are displayed in the right-hand pane.

    NOTE

    There are several types of network device naming standards used to identify network devices with persistent names, for example, **em1** and **wl3sp0**. For information about these standards, see the *Configuring and managing networking* document.

3. Toggle the **ON/OFF** switch to enable or disable the selected interface.

    NOTE

    The installation program automatically detects locally accessible interfaces, and you cannot add or remove them manually.

4. Click **+** to add a virtual network interface, which can be either: Team, Bond, Bridge, or VLAN.
5. Click **-** to remove a virtual interface.
6. Click **Configure** to change settings such as IP addresses, DNS servers, or routing configuration for an existing interface (both virtual and physical).
7. Type a host name for your system in the **Host Name** field.

> **NOTE**
>
> - The host name can either be a fully qualified domain name (FQDN) in the format ***hostname.domainname***, or a short host name without the domain. Many networks have a Dynamic Host Configuration Protocol (DHCP) service that automatically supplies connected systems with a domain name. To allow the DHCP service to assign the domain name to this system, specify only the short host name.
> - When using static IP and host name configuration, it depends on the planned system use case whether to use a short name or FQDN. Red Hat Identity Management configures FQDN during provisioning but some 3rd party software products may require short name. In either case, to ensure availability of both forms in all situations, add an entry for the host in **/etc/hosts** in the format **IP FQDN short-alias**.
> - The value **localhost** means that no specific static host name for the target system is configured, and the actual host name of the installed system is configured during the processing of the network configuration, for example, by NetworkManager using DHCP or DNS.
> - Host names can only contain alphanumeric characters and **-** or **.**. Host name should be equal to or less than 64 characters. Host names cannot start or end with **-** and **.**. To be compliant with DNS, each part of a FQDN should be equal to or less than 63 characters and the FQDN total length, including dots, should not exceed 255 characters.

8. Click **Apply** to apply the host name to the installer environment.
9. Alternatively, in the **Network and Hostname** window, you can choose the Wireless option. Click **Select network** in the right-hand pane to select your wifi connection, enter the password if required, and click **Done**.

Additional resources

- Performing an advanced RHEL 8 installation

1.2. CONFIGURING AN ETHERNET CONNECTION BY USING NMCLI

If you connect a host to the network over Ethernet, you can manage the connection's settings on the command line by using the **nmcli** utility.

Prerequisites

- A physical or virtual Ethernet Network Interface Controller (NIC) exists in the server's configuration.

Procedure

1. List the NetworkManager connection profiles:

```
# nmcli connection show
NAME                UUID                                  TYPE      DEVICE
Wired connection 1  a5eb6490-cc20-3668-81f8-0314a27f3f75  ethernet  enp1s0
```

By default, NetworkManager creates a profile for each NIC in the host. If you plan to connect this NIC only to a specific network, adapt the automatically-created profile. If you plan to connect this NIC to networks with different settings, create individual profiles for each network.

2. If you want to create an additional connection profile, enter:

```
# nmcli connection add con-name <connection-name> ifname <device-name> type ethernet
```

Skip this step to modify an existing profile.

3. Optional: Rename the connection profile:

```
# nmcli connection modify "Wired connection 1" connection.id "Internal-LAN"
```

On hosts with multiple profiles, a meaningful name makes it easier to identify the purpose of a profile.

4. Display the current settings of the connection profile:

```
# nmcli connection show Internal-LAN
...
connection.interface-name:     enp1s0
connection.autoconnect:        yes
ipv4.method:                   auto
ipv6.method:                   auto
...
```

5. Configure the IPv4 settings:

- To use DHCP, enter:

```
# nmcli connection modify Internal-LAN ipv4.method auto
```

Skip this step if **ipv4.method** is already set to **auto** (default).

- To set a static IPv4 address, network mask, default gateway, DNS servers, and search domain, enter:

```
# nmcli connection modify Internal-LAN ipv4.method manual ipv4.addresses 192.0.2.1/24 ipv4.gateway 192.0.2.254 ipv4.dns 192.0.2.200 ipv4.dns-search example.com
```

6. Configure the IPv6 settings:

- To use stateless address autoconfiguration (SLAAC), enter:

```
# nmcli connection modify Internal-LAN ipv6.method auto
```

Skip this step if **ipv6.method** is already set to **auto** (default).

- To set a static IPv6 address, network mask, default gateway, DNS servers, and search domain, enter:

```
# nmcli connection modify Internal-LAN ipv6.method manual ipv6.addresses 2001:db8:1::fffe/64 ipv6.gateway 2001:db8:1::fffe ipv6.dns 2001:db8:1::ffbb ipv6.dns-search example.com
```

7. To customize other settings in the profile, use the following command:

```
# nmcli connection modify <connection-name> <setting> <value>
```

Enclose values with spaces or semicolons in quotes.

8. Activate the profile:

```
# nmcli connection up Internal-LAN
```

Verification

1. Display the IP settings of the NIC:

```
# ip address show enp1s0
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 52:54:00:17:b8:b6 brd ff:ff:ff:ff:ff:ff
    inet 192.0.2.1/24 brd 192.0.2.255 scope global noprefixroute enp1s0
      valid_lft forever preferred_lft forever
    inet6 2001:db8:1::fffe/64 scope global noprefixroute
      valid_lft forever preferred_lft forever
```

2. Display the IPv4 default gateway:

```
# ip route show default
default via 192.0.2.254 dev enp1s0 proto static metric 102
```

3. Display the IPv6 default gateway:

```
# ip -6 route show default
default via 2001:db8:1::ffee dev enp1s0 proto static metric 102 pref medium
```

4. Display the DNS settings:

```
# cat /etc/resolv.conf
search example.com
nameserver 192.0.2.200
nameserver 2001:db8:1::ffbb
```

If multiple connection profiles are active at the same time, the order of **nameserver** entries depend on the DNS priority values in these profile and the connection types.

5. Use the **ping** utility to verify that this host can send packets to other hosts:

```
# ping <host-name-or-IP-address>
```

Troubleshooting

- Verify that the network cable is plugged-in to the host and a switch.
- Check whether the link failure exists only on this host or also on other hosts connected to the same switch.
- Verify that the network cable and the network interface are working as expected. Perform hardware diagnosis steps and replace defect cables and network interface cards.
- If the configuration on the disk does not match the configuration on the device, starting or restarting NetworkManager creates an in-memory connection that reflects the configuration of the device. For further details and how to avoid this problem, see the NetworkManager duplicates a connection after restart of NetworkManager service solution.

Additional resources

- **nm-settings(5)** man page
- Configuring NetworkManager to avoid using a specific profile to provide a default gateway
- Configuring the order of DNS servers

1.3. CONFIGURING AN ETHERNET CONNECTION BY USING NMTUI

If you connect a host to the network over Ethernet, you can manage the connection's settings in a text-based user interface by using the **nmtui** application. Use **nmtui** to create new profiles and to update existing ones on a host without a graphical interface.

> **NOTE**
>
> In **nmtui**:
>
> - Navigate by using the cursor keys.
> - Press a button by selecting it and hitting **Enter**.
> - Select and deselect checkboxes by using **Space**.

Prerequisites

- A physical or virtual Ethernet Network Interface Controller (NIC) exists in the server's configuration.

Procedure

1. If you do not know the network device name you want to use in the connection, display the available devices:

```
# nmcli device status
DEVICE     TYPE      STATE                  CONNECTION
enp1s0     ethernet  unavailable            --
...
```

2. Start **nmtui**:

```
# nmtui
```

3. Select **Edit a connection**, and press **Enter**.

4. Choose whether to add a new connection profile or to modify an existing one:

   - To create a new profile:

     i. Press the **Add** button.

     ii. Select **Ethernet** from the list of network types, and press **Enter**.

   - To modify an existing profile, select the profile from the list, and press **Enter**.

5. Optional: Update the name of the connection profile.
   On hosts with multiple profiles, a meaningful name makes it easier to identify the purpose of a profile.

6. If you create a new connection profile, enter the network device name into the **Device** field.

7. Depending on your environment, configure the IP address settings in the **IPv4 configuration** and **IPv6 configuration** areas accordingly. For this, press the button next to these areas, and select:

   - **Disabled**, if this connection does not require an IP address.

   - **Automatic**, if a DHCP server dynamically assigns an IP address to this NIC.

   - **Manual**, if the network requires static IP address settings. In this case, you must fill further fields:

     i. Press the **Show** button next to the protocol you want to configure to display additional fields.

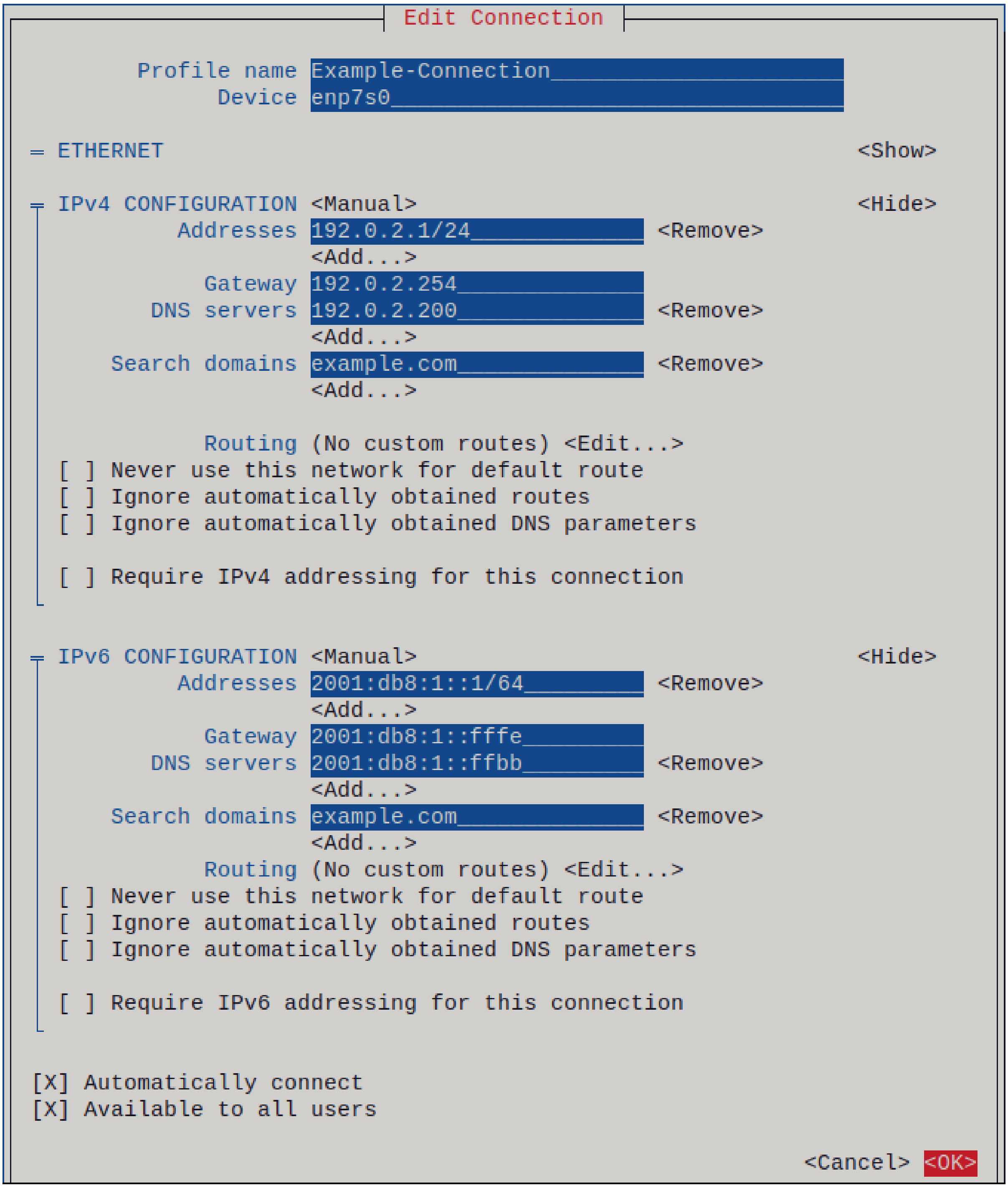
     ii. Press the **Add** button next to **Addresses**, and enter the IP address and the subnet mask in Classless Inter-Domain Routing (CIDR) format.
     If you do not specify a subnet mask, NetworkManager sets a /**32** subnet mask for IPv4 addresses and /**64** for IPv6 addresses.

     iii. Enter the address of the default gateway.

     iv. Press the **Add** button next to **DNS servers**, and enter the DNS server address.

     v. Press the **Add** button next to **Search domains**, and enter the DNS search domain.

Figure 1.1. Example of an Ethernet connection with static IP address settings



8. Press the **OK** button to create and automatically activate the new connection.

9. Press the **Back** button to return to the main menu.

10. Select **Quit**, and press **Enter** to close the **nmtui** application.

Verification

1. Display the IP settings of the NIC:

```
# ip address show enp1s0
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
group default qlen 1000
    link/ether 52:54:00:17:b8:b6 brd ff:ff:ff:ff:ff:ff
```

```
    inet 192.0.2.1/24 brd 192.0.2.255 scope global noprefixroute enp1s0
      valid_lft forever preferred_lft forever
    inet6 2001:db8:1::fffe/64 scope global noprefixroute
      valid_lft forever preferred_lft forever
```

2. Display the IPv4 default gateway:

```
# ip route show default
default via 192.0.2.254 dev enp1s0 proto static metric 102
```

3. Display the IPv6 default gateway:

```
# ip -6 route show default
default via 2001:db8:1::ffee dev enp1s0 proto static metric 102 pref medium
```

4. Display the DNS settings:

```
# cat /etc/resolv.conf
search example.com
nameserver 192.0.2.200
nameserver 2001:db8:1::ffbb
```

If multiple connection profiles are active at the same time, the order of **nameserver** entries depend on the DNS priority values in these profile and the connection types.

5. Use the **ping** utility to verify that this host can send packets to other hosts:

```
# ping <host-name-or-IP-address>
```

Troubleshooting

- Verify that the network cable is plugged-in to the host and a switch.
- Check whether the link failure exists only on this host or also on other hosts connected to the same switch.
- Verify that the network cable and the network interface are working as expected. Perform hardware diagnosis steps and replace defect cables and network interface cards.
- If the configuration on the disk does not match the configuration on the device, starting or restarting NetworkManager creates an in-memory connection that reflects the configuration of the device. For further details and how to avoid this problem, see the NetworkManager duplicates a connection after restart of NetworkManager service solution.

Additional resources

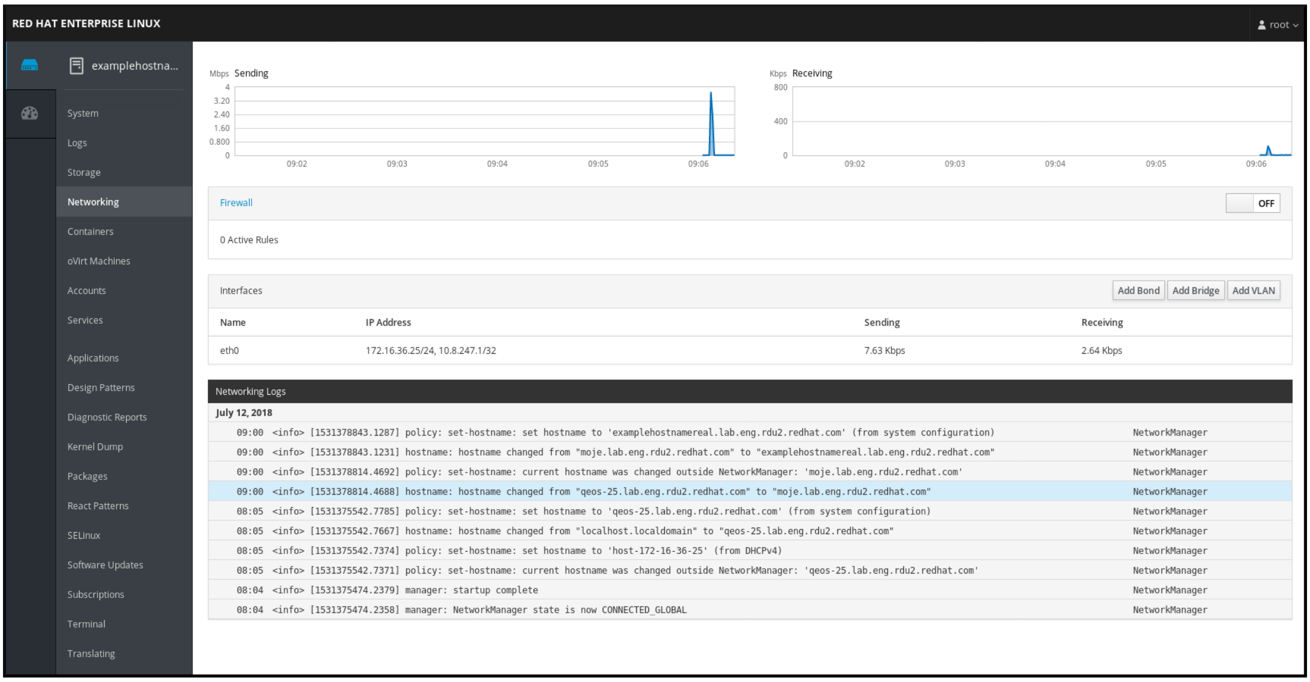
- Configuring NetworkManager to avoid using a specific profile to provide a default gateway
- Configuring the order of DNS servers

1.4. MANAGING NETWORKING IN THE RHEL WEB CONSOLE

In the web console, the **Networking** menu enables you:

- To display currently received and sent packets
- To display the most important characteristics of available network interfaces
- To display content of the networking logs.
- To add various types of network interfaces (bond, team, bridge, VLAN)

Figure 1.2. Managing Networking in the RHEL web console



1.5. MANAGING NETWORKING USING RHEL SYSTEM ROLES

You can configure the networking connections on multiple target machines using the **network** role.

The **network** role allows to configure the following types of interfaces:

- Ethernet
- Bridge
- Bonded
- VLAN
- MacVLAN
- InfiniBand

The required networking connections for each host are provided as a list within the **network_connections** variable.

> **WARNING**
>
> The **network** role updates or creates all connection profiles on the target system exactly as specified in the **network_connections** variable. Therefore, the **network** role removes options from the specified profiles if the options are only present on the system but not in the **network_connections** variable.

The following example shows how to apply the **network** role to ensure that an Ethernet connection with the required parameters exists:

An example playbook applying the network role to set up an Ethernet connection with the required parameters

```
# SPDX-License-Identifier: BSD-3-Clause
---
- hosts: managed-node-01.example.com
  vars:
    network_connections:

      # Create one Ethernet profile and activate it.
      # The profile uses automatic IP addressing
      # and is tied to the interface by MAC address.
      - name: prod1
        state: up
        type: ethernet
        autoconnect: yes
        mac: "00:00:5e:00:53:00"
        mtu: 1450

  roles:
    - rhel-system-roles.network
```

Additional resources

- Preparing a control node and managed nodes to use RHEL system roles

1.6. ADDITIONAL RESOURCES

- Configuring and managing networking

CHAPTER 2. REGISTERING THE SYSTEM AND MANAGING SUBSCRIPTIONS

Subscriptions cover products installed on Red Hat Enterprise Linux, including the operating system itself.

You can use a subscription to Red Hat Content Delivery Network to track:

- Registered systems
- Products installed on your systems
- Subscriptions attached to the installed products

2.1. REGISTERING THE SYSTEM AFTER THE INSTALLATION

Use the following procedure to register your system if you have not registered it during the installation process already.

Prerequisites

- A valid user account in the Red Hat Customer Portal.
- See the Create a Red Hat Login page.
- An active subscription for the RHEL system.
- For more information about the installation process, see Performing a standard RHEL 8 installation.

Procedure

1. Register and automatically subscribe your system in one step:

```
# subscription-manager register --username <username> --password <password> --auto-attach
Registering to: subscription.rhsm.redhat.com:443/subscription
The system has been registered with ID: 37to907c-ece6-49ea-9174-20b87ajk9ee7
The registered system name is: client1.idm.example.com
Installed Product Current Status:
Product Name: Red Hat Enterprise Linux for x86_64
Status:       Subscribed
```

   The command prompts you to enter your Red Hat Customer Portal user name and password.

   If the registration process fails, you can register your system with a specific pool. For guidance on how to do it, proceed with the following steps:

   a. Determine the pool ID of a subscription that you require:

```
# subscription-manager list --available
```

   This command displays all available subscriptions for your Red Hat account. For every subscription, various characteristics are displayed, including the pool ID.

b. Attach the appropriate subscription to your system by replacing *pool_id* with the pool ID determined in the previous step:

```
# subscription-manager attach --pool=pool_id
```

> **NOTE**
>
> To register the system with Red Hat Insights, you can use the **rhc connect** utility. See Setting up remote host configuration.

Additional resources

- Understanding autoattaching subscriptions on the Customer Portal
- Understanding the manual registration and subscription on the Customer Portal

2.2. REGISTERING SUBSCRIPTIONS WITH CREDENTIALS IN THE WEB CONSOLE

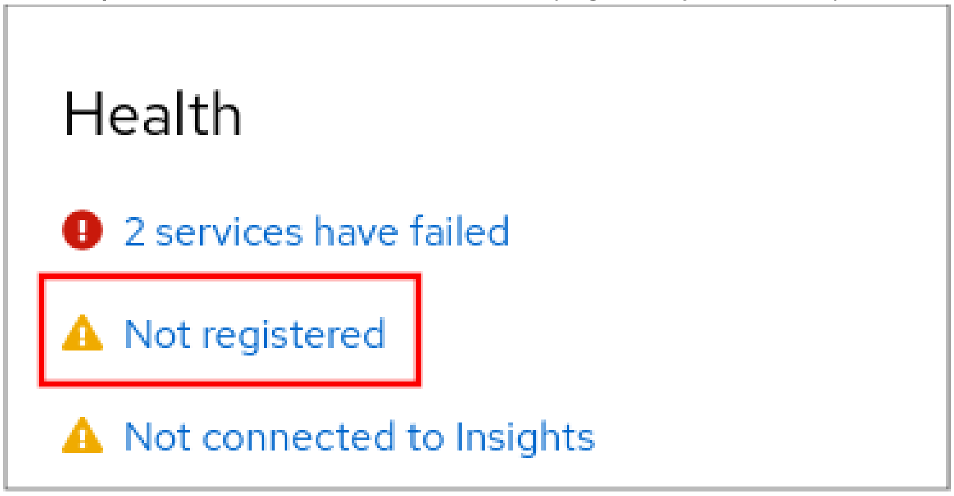
Use the following steps to register a newly installed Red Hat Enterprise Linux with account credentials using the RHEL web console.

Prerequisites

- A valid user account on the Red Hat Customer Portal.
  See the Create a Red Hat Login page.
- Active subscription for your RHEL system.

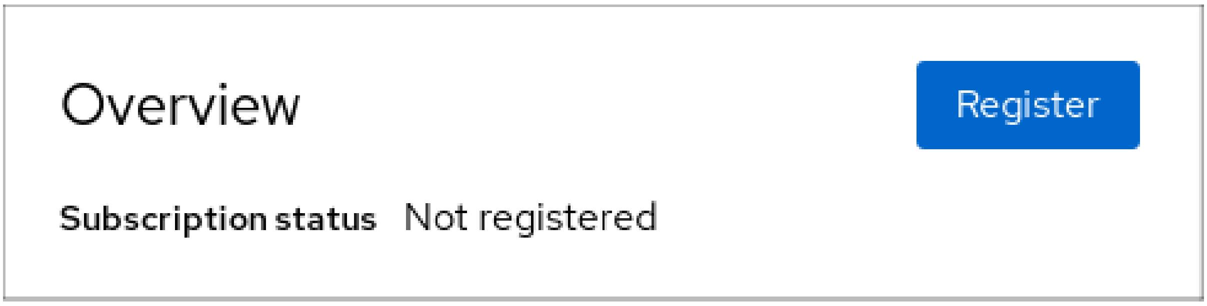
Procedure

1. Log in to the RHEL web console. For details, see Logging in to the web console.
2. In the **Health** filed in the **Overview** page, click the **Not registered** warning, or click **Subscriptions** in the main menu to move to page with your subscription information.

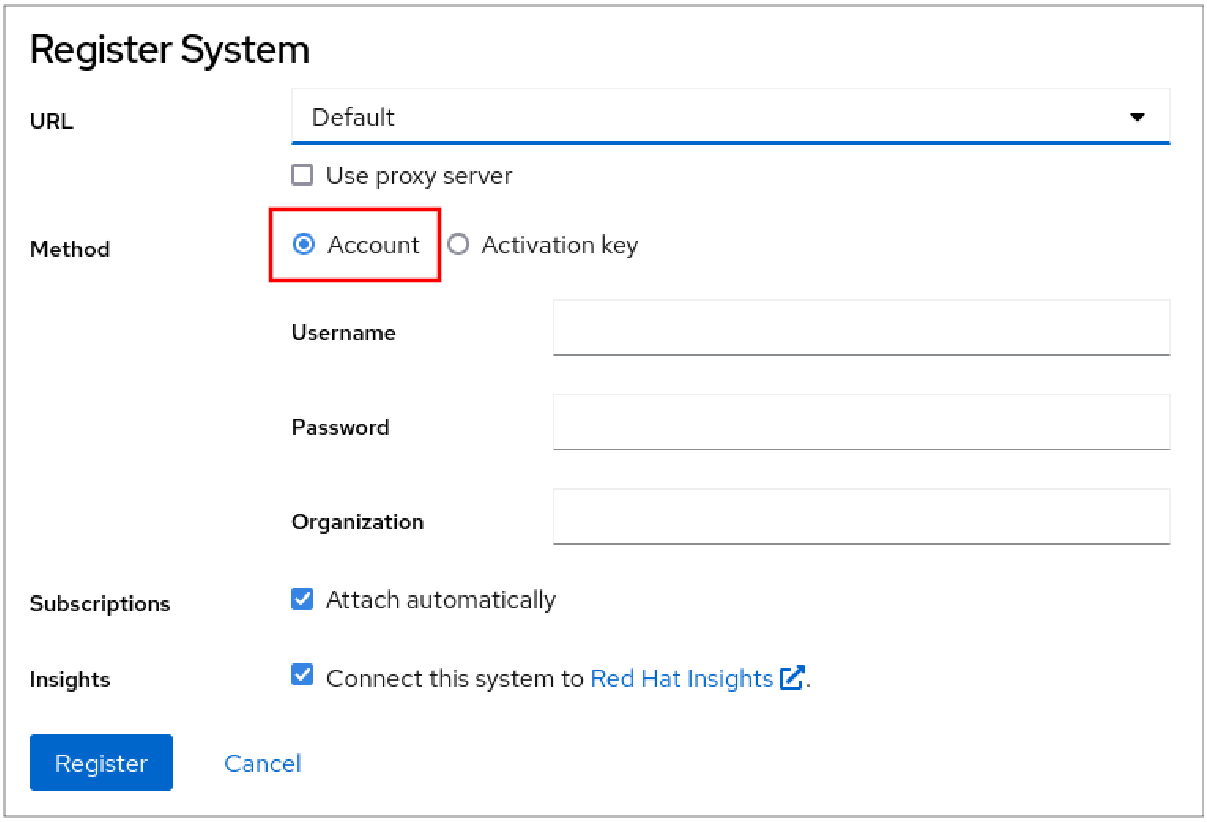


.

3. In the **Overview** filed, click **Register**.



4. In the **Register system** dialog box, select that you want to register using your account credentials.



5. Enter your username.

6. Enter your password.

7. Optionally, enter your organization's name or ID.
   If your account belongs to more than one organization on the Red Hat Customer Portal, you have to add the organization name or organization ID. To get the org ID, go to your Red Hat contact point.

   - If you do not want to connect your system to Red Hat Insights, clear the **Insights** check box.

8. Click the **Register** button.

At this point, your Red Hat Enterprise Linux Enterprise Linux system has been successfully registered.

2.3. REGISTERING A SYSTEM USING RED HAT ACCOUNT ON GNOME

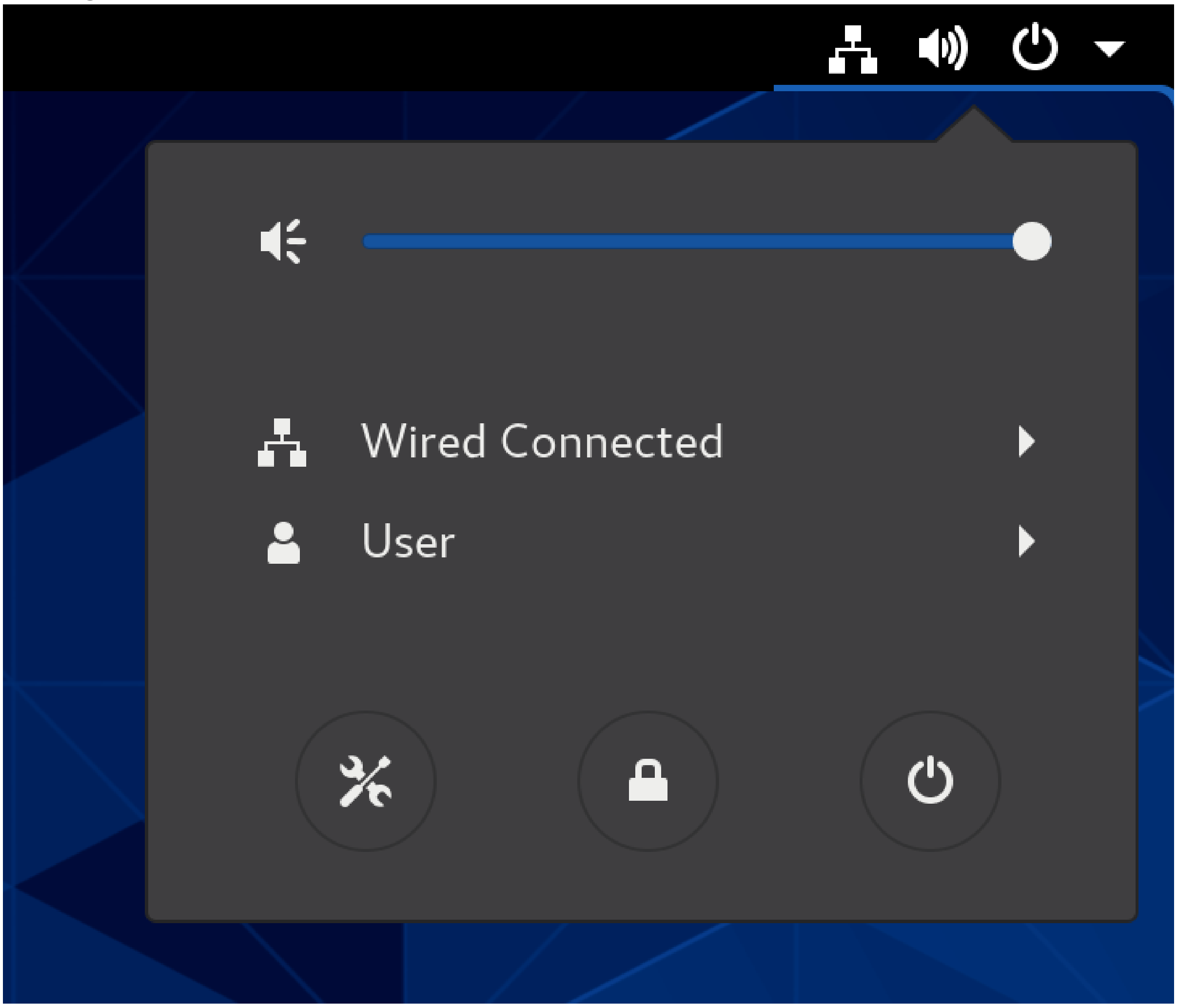
Follow the steps in this procedure to enroll your system with your Red Hat account.

Prerequisites

- A valid account on Red Hat customer portal.
  See the Create a Red Hat Login page for new user registration.

Procedure

1. Open the **system menu**, which is accessible from the upper-right screen corner, and click the **Settings** icon.



2. In the **Details → About** section, click **Register**.
3. Select **Registration Server**.
4. If you are not using the Red Hat server, enter the server address in the **URL** field.
5. In the **Registration Type** menu, select **Red Hat Account**
6. Under **Registration Details**:
   - Enter your Red Hat account user name in the **Login** field.
   - Enter your Red Hat account password in the **Password** field.
   - Enter the name of your organization in the **Organization** field.
7. Click **Register**.

2.4. REGISTERING A SYSTEM USING AN ACTIVATION KEY ON GNOME

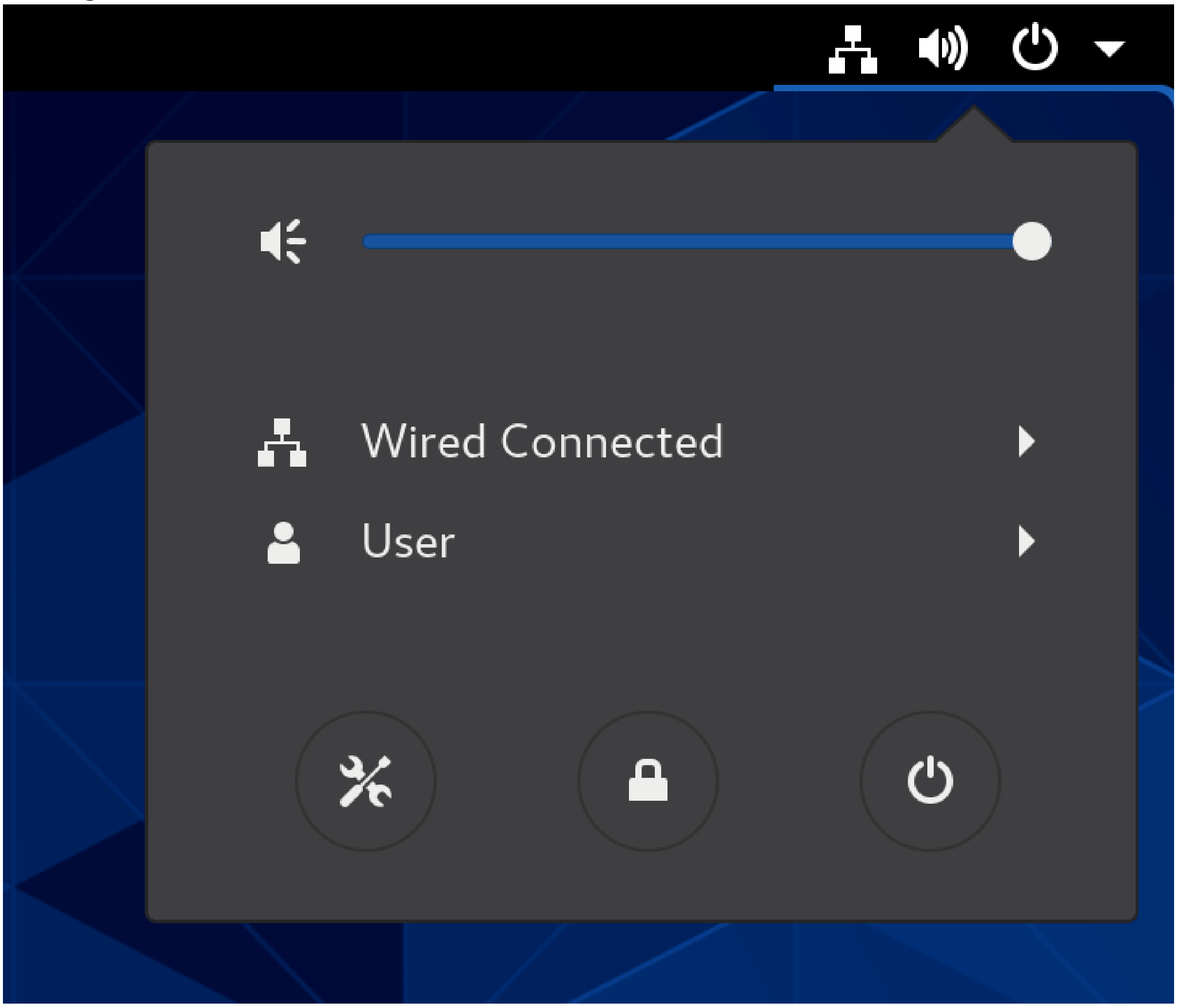
Follow the steps in this procedure to register your system with an activation key. You can get the activation key from your organization administrator.

Prerequisites

- Activation key or keys.
  See the Activation Keys page for creating new activation keys.

Procedure

1. Open the **system menu**, which is accessible from the upper-right screen corner, and click the **Settings** icon.



2. In the **Details → About** section, click **Register**.
3. Select **Registration Server**.
4. If you are not using the Red Hat server, enter the server address in the **URL** field.
5. In the **Registration Type** menu, select **Activation Keys**.
6. Under **Registration Details**:
   - Enter your activation keys in the **Activation Keys** field.
     Separate your keys by a comma (**,**).
   - Enter the name or ID of your organization in the **Organization** field.

7. Click **Register**.

2.5. REGISTERING RHEL 8 USING THE INSTALLER GUI

Use the following steps to register a Red Hat Enterprise Linux 8 using the RHEL installer GUI.

Prerequisites

- You have a valid user account on the Red Hat Customer Portal. See the Create a Red Hat Login page.
- You have a valid Activation Key and Organization id.

Procedure

1. From the **Installation Summary** screen, under **Software**, click **Connect to Red Hat**
2. Authenticate your Red Hat account using the **Account** or **Activation Key** option.
3. Optional: In the **Set System Purpose** field select the **Role**, **SLA**, and **Usage** attribute that you want to set from the drop-down menu.
   At this point, your Red Hat Enterprise Linux 8 system has been successfully registered.

CHAPTER 3. ACCESSING THE RED HAT SUPPORT

This section describes how to effectively troubleshoot your problems using Red Hat support and **sosreport**.

To obtain support from Red Hat, use the Red Hat Customer Portal, which provides access to everything available with your subscription.

3.1. OBTAINING RED HAT SUPPORT THROUGH RED HAT CUSTOMER PORTAL

The following section describes how to use the Red Hat Customer Portal to get help.

Prerequisites

- A valid user account on the Red Hat Customer Portal. See Create a Red Hat Login .
- An active subscription for the RHEL system.

Procedure

1. Access Red Hat support:
   a. Open a new support case.
   b. Initiate a live chat with a Red Hat expert.
   c. Contact a Red Hat expert by making a call or sending an email.

3.2. TROUBLESHOOTING PROBLEMS USING SOSREPORT

The **sosreport** command collects configuration details, system information and diagnostic information from a Red Hat Enterprise Linux system.

The following section describes how to use the **sosreport** command to produce reports for your support cases.

Prerequisites

- A valid user account on the Red Hat Customer Portal. See Create a Red Hat Login .
- An active subscription for the RHEL system.
- A support-case number.

Procedure

1. Install the **sos** package:

```
# yum install sos
```

> **NOTE**
>
> The default minimal installation of Red Hat Enterprise Linux does not include the **sos** package, which provides the **sosreport** command.

2. Generate a report:

```
# sosreport
```

3. Attach the report to your support case.
   See the How can I attach a file to a Red Hat support case? Red Hat Knowledgebase article for more information.

   Note that when attaching the report, you are prompted to enter the number of the relevant support case.

Additional resources

- What is an sosreport and how to create one in Red Hat Enterprise Linux?

CHAPTER 4. CHANGING BASIC ENVIRONMENT SETTINGS

Configuration of basic environment settings is a part of the installation process. The following sections guide you when you change them later. The basic configuration of the environment includes:

- Date and time
- System locales
- Keyboard layout
- Language

4.1. CONFIGURING THE DATE AND TIME

Accurate timekeeping is important for several reasons. In Red Hat Enterprise Linux, timekeeping is ensured by the **NTP** protocol, which is implemented by a daemon running in user space. The user-space daemon updates the system clock running in the kernel. The system clock can keep time by using various clock sources.

Red Hat Enterprise Linux 8 uses the **chronyd** daemon to implement **NTP**. **chronyd** is available from the **chrony** package. For more information, see Using the chrony suite to configure NTP .

4.1.1. Displaying the current date and time

To display the current date and time, use either of these steps.

Procedure

1. Enter the **date** command:

```
$ date
Mon Mar 30 16:02:59 CEST 2020
```

2. To see more details, use the **timedatectl** command:

```
$ timedatectl
Local time: Mon 2020-03-30 16:04:42 CEST
Universal time: Mon 2020-03-30 14:04:42 UTC
  RTC time: Mon 2020-03-30 14:04:41
 Time zone: Europe/Prague (CEST, +0200)
System clock synchronized: yes
NTP service: active
RTC in local TZ: no
```

Additional resources

- Configuring time settings using the web console
- **man date(1)** and **man timedatectl(1)**

4.2. CONFIGURING THE SYSTEM LOCALE

System-wide locale settings are stored in the **/etc/locale.conf** file that is read at early boot by the **systemd** daemon. Every service or user inherits the locale settings configured in **/etc/locale.conf**, unless individual programs or individual users override them.

Procedure

- To list available system locale settings:

```
$ localectl list-locales
C.utf8
aa_DJ
aa_DJ.iso88591
aa_DJ.utf8
...
```

- To display the current status of the system locales settings:

```
$ localectl status
```

- To set or change the default system locale settings, use a **localectl set-locale** sub-command as the **root** user. For example:

```
# localectl set-locale LANG=en_US
```

Additional resources

- **man localectl(1)**, **man locale(7)**, and **man locale.conf(5)**

4.3. CONFIGURING THE KEYBOARD LAYOUT

The keyboard layout settings control the layout used on the text console and graphical user interfaces.

Procedure

- To list available keymaps:

```
$ localectl list-keymaps
ANSI-dvorak
al
al-plisi
amiga-de
amiga-us
...
```

- To display the current status of keymaps settings:

```
$ localectl status
...
VC Keymap: us
...
```

- To set or change the default system keymap. For example:

```
# localectl set-keymap us
```

Additional resources

- **man localectl(1)**, **man locale(7)**, and **man locale.conf(5)**

4.4. CHANGING THE FONT SIZE IN TEXT CONSOLE MODE

You can change the font size in the virtual console by using the **setfont** command.

- Enter the **setfont** command with the name of the font, for example:

```
# setfont /usr/lib/kbd/consolefonts/LatArCyrHeb-19.psfu.gz
```

NOTE

The **setfont** command searches for multiple hard-coded paths by default. Therefore, **setfont** does not require the full name and path to the font.

- To double the size of the font horizontally and vertically, enter the **setfont** command with **-d** parameter:

```
# setfont -d LatArCyrHeb-16
```

NOTE

The maximum font size that you can double is 16x16 pixel.

- To preserve the selected font during the reboot of the system, use the **FONT** variable in the **/etc/vconsole.conf** file, for example:

```
# cat /etc/vconsole.conf
KEYMAP="us"
FONT="eurlatgr"
```

- You can find various fonts in the **kbd-misc** package, which is installed with the`kbd` package. For example, the font **LatArCyrHeb** has many variants:

```
# rpm -ql kbd-misc | grep LatAr

/usr/lib/kbd/consolefonts/LatArCyrHeb-08.psfu.gz
/usr/lib/kbd/consolefonts/LatArCyrHeb-14.psfu.gz
/usr/lib/kbd/consolefonts/LatArCyrHeb-16+.psfu.gz
/usr/lib/kbd/consolefonts/LatArCyrHeb-16.psfu.gz
/usr/lib/kbd/consolefonts/LatArCyrHeb-19.psfu.gz
```

NOTE

The maximum supported font size by the virtual console is 32 pixels. You can reduce the font readability problem by using smaller resolution for the console.

4.5. ADDITIONAL RESOURCES

- Performing a standard RHEL 8 installation

CHAPTER 5. USING SECURE COMMUNICATIONS BETWEEN TWO SYSTEMS WITH OPENSSH

SSH (Secure Shell) is a protocol which provides secure communications between two systems using a client-server architecture and allows users to log in to server host systems remotely. Unlike other remote communication protocols, such as FTP or Telnet, SSH encrypts the login session, which prevents intruders to collect unencrypted passwords from the connection.

Red Hat Enterprise Linux includes the basic **OpenSSH** packages: the general **openssh** package, the **openssh-server** package and the **openssh-clients** package. Note that the **OpenSSH** packages require the **OpenSSL** package **openssl-libs**, which installs several important cryptographic libraries that enable **OpenSSH** to provide encrypted communications.

5.1. SSH AND OPENSSH

SSH (Secure Shell) is a program for logging into a remote machine and executing commands on that machine. The SSH protocol provides secure encrypted communications between two untrusted hosts over an insecure network. You can also forward X11 connections and arbitrary TCP/IP ports over the secure channel.

The SSH protocol mitigates security threats, such as interception of communication between two systems and impersonation of a particular host, when you use it for remote shell login or file copying. This is because the SSH client and server use digital signatures to verify their identities. Additionally, all communication between the client and server systems is encrypted.

A host key authenticates hosts in the SSH protocol. Host keys are cryptographic keys that are generated automatically when OpenSSH is first installed, or when the host boots for the first time.

OpenSSH is an implementation of the SSH protocol supported by Linux, UNIX, and similar operating systems. It includes the core files necessary for both the OpenSSH client and server. The OpenSSH suite consists of the following user-space tools:

- **ssh** is a remote login program (SSH client).
- **sshd** is an OpenSSH SSH daemon.
- **scp** is a secure remote file copy program.
- **sftp** is a secure file transfer program.
- **ssh-agent** is an authentication agent for caching private keys.
- **ssh-add** adds private key identities to **ssh-agent**.
- **ssh-keygen** generates, manages, and converts authentication keys for **ssh**.
- **ssh-copy-id** is a script that adds local public keys to the **authorized_keys** file on a remote SSH server.
- **ssh-keyscan** gathers SSH public host keys.

Two versions of SSH currently exist: version 1, and the newer version 2. The OpenSSH suite in RHEL supports only SSH version 2. It has an enhanced key-exchange algorithm that is not vulnerable to exploits known in version 1.

OpenSSH, as one of core cryptographic subsystems of RHEL, uses system-wide crypto policies. This ensures that weak cipher suites and cryptographic algorithms are disabled in the default configuration. To modify the policy, the administrator must either use the **update-crypto-policies** command to adjust the settings or manually opt out of the system-wide crypto policies.

The OpenSSH suite uses two sets of configuration files: one for client programs (that is, **ssh**, **scp**, and **sftp**), and another for the server (the **sshd** daemon).

System-wide SSH configuration information is stored in the **/etc/ssh/** directory. User-specific SSH configuration information is stored in **~/.ssh/** in the user's home directory. For a detailed list of OpenSSH configuration files, see the **FILES** section in the **sshd(8)** man page.

Additional resources

- Man pages listed by using the **man -k ssh** command
- Using system-wide cryptographic policies

5.2. CONFIGURING AND STARTING AN OPENSSH SERVER

Use the following procedure for a basic configuration that might be required for your environment and for starting an OpenSSH server. Note that after the default RHEL installation, the **sshd** daemon is already started and server host keys are automatically created.

Prerequisites

- The **openssh-server** package is installed.

Procedure

1. Start the **sshd** daemon in the current session and set it to start automatically at boot time:

```
# systemctl start sshd
# systemctl enable sshd
```

2. To specify different addresses than the default **0.0.0.0** (IPv4) or **::** (IPv6) for the **ListenAddress** directive in the **/etc/ssh/sshd_config** configuration file and to use a slower dynamic network configuration, add the dependency on the **network-online.target** target unit to the **sshd.service** unit file. To achieve this, create the **/etc/systemd/system/sshd.service.d/local.conf** file with the following content:

```
[Unit]
Wants=network-online.target
After=network-online.target
```

3. Review if OpenSSH server settings in the **/etc/ssh/sshd_config** configuration file meet the requirements of your scenario.

4. Optionally, change the welcome message that your OpenSSH server displays before a client authenticates by editing the **/etc/issue** file, for example:

```
Welcome to ssh-server.example.com
Warning: By accessing this server, you agree to the referenced terms and conditions.
```

Ensure that the **Banner** option is not commented out in **/etc/ssh/sshd_config** and its value contains **/etc/issue**:

```
# less /etc/ssh/sshd_config | grep Banner
Banner /etc/issue
```

Note that to change the message displayed after a successful login you have to edit the **/etc/motd** file on the server. See the **pam_motd** man page for more information.

5. Reload the **systemd** configuration and restart **sshd** to apply the changes:

```
# systemctl daemon-reload
# systemctl restart sshd
```

Verification

1. Check that the **sshd** daemon is running:

```
# systemctl status sshd
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2019-11-18 14:59:58 CET; 6min ago
     Docs: man:sshd(8)
           man:sshd_config(5)
 Main PID: 1149 (sshd)
    Tasks: 1 (limit: 11491)
   Memory: 1.9M
   CGroup: /system.slice/sshd.service
           └─1149 /usr/sbin/sshd -D -oCiphers=aes128-ctr,aes256-ctr,aes128-cbc,aes256-cbc -
oMACs=hmac-sha2-256,>

Nov 18 14:59:58 ssh-server-example.com systemd[1]: Starting OpenSSH server daemon...
Nov 18 14:59:58 ssh-server-example.com sshd[1149]: Server listening on 0.0.0.0 port 22.
Nov 18 14:59:58 ssh-server-example.com sshd[1149]: Server listening on :: port 22.
Nov 18 14:59:58 ssh-server-example.com systemd[1]: Started OpenSSH server daemon.
```

2. Connect to the SSH server with an SSH client.

```
# ssh user@ssh-server-example.com
ECDSA key fingerprint is SHA256:dXbaS0RG/UzlTTku8GtXSz0S1++lPegSy31v3L/FAEc.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ssh-server-example.com' (ECDSA) to the list of known hosts.

user@ssh-server-example.com's password:
```

Additional resources

- **sshd(8)** and **sshd_config(5)** man pages.

5.3. SETTING AN OPENSSH SERVER FOR KEY-BASED AUTHENTICATION

To improve system security, enforce key-based authentication by disabling password authentication on your OpenSSH server.

Prerequisites

- The **openssh-server** package is installed.
- The **sshd** daemon is running on the server.

Procedure

1. Open the **/etc/ssh/sshd_config** configuration in a text editor, for example:

```
# vi /etc/ssh/sshd_config
```

2. Change the **PasswordAuthentication** option to **no**:

```
PasswordAuthentication no
```

   On a system other than a new default installation, check that **PubkeyAuthentication no** has not been set and the **ChallengeResponseAuthentication** directive is set to **no**. If you are connected remotely, not using console or out-of-band access, test the key-based login process before disabling password authentication.

3. To use key-based authentication with NFS-mounted home directories, enable the **use_nfs_home_dirs** SELinux boolean:

```
# setsebool -P use_nfs_home_dirs 1
```

4. Reload the **sshd** daemon to apply the changes:

```
# systemctl reload sshd
```

Additional resources

- **sshd(8)**, **sshd_config(5)**, and **setsebool(8)** man pages.

5.4. GENERATING SSH KEY PAIRS

Use this procedure to generate an SSH key pair on a local system and to copy the generated public key to an OpenSSH server. If the server is configured accordingly, you can log in to the OpenSSH server without providing any password.

IMPORTANT

If you complete the following steps as **root**, only **root** is able to use the keys.

Procedure

1. To generate an ECDSA key pair for version 2 of the SSH protocol:

```
$ ssh-keygen -t ecdsa
Generating public/private ecdsa key pair.
```

```
Enter file in which to save the key (/home/joesec/.ssh/id_ecdsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/joesec/.ssh/id_ecdsa.
Your public key has been saved in /home/joesec/.ssh/id_ecdsa.pub.
The key fingerprint is:
SHA256:Q/x+qms4j7PCQ0qFd09iZEFHA+SqwBKRNaU72oZfaCI
joesec@localhost.example.com
The key's randomart image is:
+---[ECDSA 256]---+
|.oo..o=++        |
|.. o .oo .       |
|. .. o. o        |
|....o.+...       |
|o.oo.o +S .      |
|.=.+.   .o       |
|E.*+.  .  . .    |
|.=..+ +..  o     |
|  .  oo*+o.      |
+----[SHA256]-----+
```

You can also generate an RSA key pair by using the **-t rsa** option with the **ssh-keygen** command or an Ed25519 key pair by entering the **ssh-keygen -t ed25519** command.

2. To copy the public key to a remote machine:

```
$ ssh-copy-id joesec@ssh-server-example.com
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are
already installed
joesec@ssh-server-example.com's password:
...
Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'joesec@ssh-server-example.com'" and check to
make sure that only the key(s) you wanted were added.
```

If you do not use the **ssh-agent** program in your session, the previous command copies the most recently modified **~/.ssh/id*.pub** public key if it is not yet installed. To specify another public-key file or to prioritize keys in files over keys cached in memory by **ssh-agent**, use the **ssh-copy-id** command with the **-i** option.

NOTE

If you reinstall your system and want to keep previously generated key pairs, back up the **~/.ssh/** directory. After reinstalling, copy it back to your home directory. You can do this for all users on your system, including **root**.

Verification

1. Log in to the OpenSSH server without providing any password:

```
$ ssh joesec@ssh-server-example.com
Welcome message.
...
Last login: Mon Nov 18 18:28:42 2019 from ::1
```

Additional resources

- **ssh-keygen(1)** and **ssh-copy-id(1)** man pages.

5.5. USING SSH KEYS STORED ON A SMART CARD

Red Hat Enterprise Linux enables you to use RSA and ECDSA keys stored on a smart card on OpenSSH clients. Use this procedure to enable authentication using a smart card instead of using a password.

Prerequisites

- On the client side, the **opensc** package is installed and the **pcscd** service is running.

Procedure

1. List all keys provided by the OpenSC PKCS #11 module including their PKCS #11 URIs and save the output to the *keys.pub* file:

```
$ ssh-keygen -D pkcs11: > keys.pub
$ ssh-keygen -D pkcs11:
ssh-rsa AAAAB3NzaC1yc2E...KKZMzcQZzx
pkcs11:id=%02;object=SIGN%20pubkey;token=SSH%20key;manufacturer=piv_II?module-
path=/usr/lib64/pkcs11/opensc-pkcs11.so
ecdsa-sha2-nistp256 AAA...J0hkYnnsM=
pkcs11:id=%01;object=PIV%20AUTH%20pubkey;token=SSH%20key;manufacturer=piv_II?
module-path=/usr/lib64/pkcs11/opensc-pkcs11.so
```

2. To enable authentication using a smart card on a remote server (*example.com*), transfer the public key to the remote server. Use the **ssh-copy-id** command with *keys.pub* created in the previous step:

```
$ ssh-copy-id -f -i keys.pub username@example.com
```

3. To connect to *example.com* using the ECDSA key from the output of the **ssh-keygen -D** command in step 1, you can use just a subset of the URI, which uniquely references your key, for example:

```
$ ssh -i "pkcs11:id=%01?module-path=/usr/lib64/pkcs11/opensc-pkcs11.so" example.com
Enter PIN for 'SSH key':
[example.com] $
```

4. You can use the same URI string in the **~/.ssh/config** file to make the configuration permanent:

```
$ cat ~/.ssh/config
IdentityFile "pkcs11:id=%01?module-path=/usr/lib64/pkcs11/opensc-pkcs11.so"
$ ssh example.com
Enter PIN for 'SSH key':
[example.com] $
```

   Because OpenSSH uses the **p11-kit-proxy** wrapper and the OpenSC PKCS #11 module is registered to PKCS#11 Kit, you can simplify the previous commands:

```
$ ssh -i "pkcs11:id=%01" example.com
Enter PIN for 'SSH key':
[example.com] $
```

If you skip the **id=** part of a PKCS #11 URI, OpenSSH loads all keys that are available in the proxy module. This can reduce the amount of typing required:

```
$ ssh -i pkcs11: example.com
Enter PIN for 'SSH key':
[example.com] $
```

Additional resources

- Fedora 28: Better smart card support in OpenSSH
- **p11-kit(8)**, **opensc.conf(5)**, **pcscd(8)**, **ssh(1)**, and **ssh-keygen(1)** man pages

5.6. MAKING OPENSSH MORE SECURE

The following tips help you to increase security when using OpenSSH. Note that changes in the **/etc/ssh/sshd_config** OpenSSH configuration file require reloading the **sshd** daemon to take effect:

```
# systemctl reload sshd
```

> **IMPORTANT**
>
> The majority of security hardening configuration changes reduce compatibility with clients that do not support up-to-date algorithms or cipher suites.

Disabling insecure connection protocols

- To make SSH truly effective, prevent the use of insecure connection protocols that are replaced by the OpenSSH suite. Otherwise, a user's password might be protected using SSH for one session only to be captured later when logging in using Telnet. For this reason, consider disabling insecure protocols, such as telnet, rsh, rlogin, and ftp.

Enabling key-based authentication and disabling password-based authentication

- Disabling passwords for authentication and allowing only key pairs reduces the attack surface and it also might save users' time. On clients, generate key pairs using the **ssh-keygen** tool and use the **ssh-copy-id** utility to copy public keys from clients on the OpenSSH server. To disable password-based authentication on your OpenSSH server, edit **/etc/ssh/sshd_config** and change the **PasswordAuthentication** option to **no**:

  ```
  PasswordAuthentication no
  ```

Key types

- Although the **ssh-keygen** command generates a pair of RSA keys by default, you can instruct it to generate ECDSA or Ed25519 keys by using the **-t** option. The ECDSA (Elliptic Curve Digital Signature Algorithm) offers better performance than RSA at the equivalent symmetric key strength. It also generates shorter keys. The Ed25519 public-key algorithm is an implementation of twisted Edwards curves that is more secure and also faster than RSA, DSA, and ECDSA. OpenSSH creates RSA, ECDSA, and Ed25519 server host keys automatically if they are missing. To configure the host key creation in RHEL, use the **sshd-keygen@.service** instantiated service. For example, to disable the automatic creation of the RSA key type:

```
# systemctl mask sshd-keygen@rsa.service
```

NOTE

In images with **cloud-init** enabled, the **ssh-keygen** units are automatically disabled. This is because the **ssh-keygen template** service can interfere with the **cloud-init** tool and cause problems with host key generation. To prevent these problems the **etc/systemd/system/sshd-keygen@.service.d/disable-sshd-keygen-if-cloud-init-active.conf** drop-in configuration file disables the **ssh-keygen** units if **cloud-init** is running.

- To exclude particular key types for SSH connections, comment out the relevant lines in **/etc/ssh/sshd_config**, and reload the **sshd** service. For example, to allow only Ed25519 host keys:

```
# HostKey /etc/ssh/ssh_host_rsa_key
# HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_ed25519_key
```

IMPORTANT

The Ed25519 algorithm is not FIPS-140-compliant, and OpenSSH does not work with Ed25519 keys in FIPS mode.

Non-default port

- By default, the **sshd** daemon listens on TCP port 22. Changing the port reduces the exposure of the system to attacks based on automated network scanning and therefore increase security through obscurity. You can specify the port using the **Port** directive in the **/etc/ssh/sshd_config** configuration file.
  You also have to update the default SELinux policy to allow the use of a non-default port. To do so, use the **semanage** tool from the **policycoreutils-python-utils** package:

```
# semanage port -a -t ssh_port_t -p tcp <port_number>
```

Furthermore, update **firewalld** configuration:

```
# firewall-cmd --add-port <port_number>/tcp
# firewall-cmd --remove-port=22/tcp
# firewall-cmd --runtime-to-permanent
```

In the previous commands, replace *<port_number>* with the new port number specified using the **Port** directive.

No root login

- If your particular use case does not require the possibility of logging in as the root user, you can set the **PermitRootLogin** configuration directive to **no** in the **/etc/ssh/sshd_config** file. By disabling the possibility of logging in as the root user, the administrator can audit which users run what privileged commands after they log in as regular users and then gain root rights.
  Alternatively, set **PermitRootLogin** to **prohibit-password**:

  ```
  PermitRootLogin prohibit-password
  ```

  This enforces the use of key-based authentication instead of the use of passwords for logging in as root and reduces risks by preventing brute-force attacks.

Using the X Security extension

- The X server in Red Hat Enterprise Linux clients does not provide the X Security extension. Therefore, clients cannot request another security layer when connecting to untrusted SSH servers with X11 forwarding. Most applications are not able to run with this extension enabled anyway.
  By default, the **ForwardX11Trusted** option in the **/etc/ssh/ssh_config.d/05-redhat.conf** file is set to **yes**, and there is no difference between the **ssh -X remote_machine** (untrusted host) and **ssh -Y remote_machine** (trusted host) command.

  If your scenario does not require the X11 forwarding feature at all, set the **X11Forwarding** directive in the **/etc/ssh/sshd_config** configuration file to **no**.

Restricting access to specific users, groups, or domains

- The **AllowUsers** and **AllowGroups** directives in the **/etc/ssh/sshd_config** configuration file server enable you to permit only certain users, domains, or groups to connect to your OpenSSH server. You can combine **AllowUsers** and **AllowGroups** to restrict access more precisely, for example:

  ```
  AllowUsers *@192.168.1.* *@10.0.0.* !*@192.168.1.2
  AllowGroups example-group
  ```

  The previous configuration lines accept connections from all users from systems in 192.168.1.* and 10.0.0.* subnets except from the system with the 192.168.1.2 address. All users must be in the **example-group** group. The OpenSSH server denies all other connections.

  The OpenSSH server permits only connections that pass all Allow and Deny directives in **/etc/ssh/sshd_config**. For example, if the **AllowUsers** directive lists a user that is not part of a group listed in the **AllowGroups** directive, then the user cannot log in.

  Note that using allowlists (directives starting with Allow) is more secure than using blocklists (options starting with Deny) because allowlists block also new unauthorized users or groups.

Changing system-wide cryptographic policies

- OpenSSH uses RHEL system-wide cryptographic policies, and the default system-wide cryptographic policy level offers secure settings for current threat models. To make your cryptographic settings more strict, change the current policy level:

  ```
  # update-crypto-policies --set FUTURE
  Setting system policy to FUTURE
  ```

> **WARNING**
>
> If your system communicates on the internet, you might face interoperability problems due to the strict setting of the **FUTURE** policy.

You can also disable only specific ciphers for the SSH protocol through the system-wide cryptographic policies. See the Customizing system-wide cryptographic policies with subpolicies section in the Security hardening document for more information.

To opt out of the system-wide cryptographic policies for your OpenSSH server, uncomment the line with the **CRYPTO_POLICY=** variable in the **/etc/sysconfig/sshd** file. After this change, values that you specify in the **Ciphers**, **MACs**, **KexAlgoritms**, and **GSSAPIKexAlgorithms** sections in the **/etc/ssh/sshd_config** file are not overridden.

See the **sshd_config(5)** man page for more information.

To opt out of system-wide cryptographic policies for your OpenSSH client, perform one of the following tasks:

- For a given user, override the global **ssh_config** with a user-specific configuration in the **~/.ssh/config** file.
- For the entire system, specify the cryptographic policy in a drop-in configuration file located in the **/etc/ssh/ssh_config.d/** directory, with a two-digit number prefix smaller than 5, so that it lexicographically precedes the **05-redhat.conf** file, and with a **.conf** suffix, for example, **04-crypto-policy-override.conf**.

Additional resources

- **sshd_config(5)**, **ssh-keygen(1)**, **crypto-policies(7)**, and **update-crypto-policies(8)** man pages.
- Using system-wide cryptographic policies in the Security hardening document.
- How to disable specific algorithms and ciphers for ssh service only article.

5.7. CONNECTING TO A REMOTE SERVER USING AN SSH JUMP HOST

Use this procedure for connecting your local system to a remote server through an intermediary server, also called jump host.

Prerequisites

- A jump host accepts SSH connections from your local system.
- A remote server accepts SSH connections only from the jump host.

Procedure

1. Define the jump host by editing the **~/.ssh/config** file on your local system, for example:

```
Host jump-server1
  HostName jump1.example.com
```

- The **Host** parameter defines a name or alias for the host you can use in **ssh** commands. The value can match the real host name, but can also be any string.
- The **HostName** parameter sets the actual host name or IP address of the jump host.

2. Add the remote server jump configuration with the **ProxyJump** directive to **~/.ssh/config** file on your local system, for example:

```
Host remote-server
  HostName remote1.example.com
  ProxyJump jump-server1
```

3. Use your local system to connect to the remote server through the jump server:

```
$ ssh remote-server
```

The previous command is equivalent to the **ssh -J jump-server1 remote-server** command if you omit the configuration steps 1 and 2.

> **NOTE**
>
> You can specify more jump servers and you can also skip adding host definitions to the configurations file when you provide their complete host names, for example:
>
> ```
> $ ssh -J jump1.example.com,jump2.example.com,jump3.example.com remote1.example.com
> ```
>
> Change the host name-only notation in the previous command if the user names or SSH ports on the jump servers differ from the names and ports on the remote server, for example:
>
> ```
> $ ssh -J johndoe@jump1.example.com:75,johndoe@jump2.example.com:75,johndoe@jump3.example.com:75 joesec@remote1.example.com:220
> ```

Additional resources

- **ssh_config(5)** and **ssh(1)** man pages.

5.8. CONNECTING TO REMOTE MACHINES WITH SSH KEYS USING SSH-AGENT

To avoid entering a passphrase each time you initiate an SSH connection, you can use the **ssh-agent** utility to cache the private SSH key. The private key and the passphrase remain secure.

Prerequisites

- You have a remote host with SSH daemon running and reachable through the network.

- You know the IP address or hostname and credentials to log in to the remote host.
- You have generated an SSH key pair with a passphrase and transferred the public key to the remote machine.

For more information, see Generating SSH key pairs.

Procedure

1. Optional: Verify you can use the key to authenticate to the remote host:

   a. Connect to the remote host using SSH:

   ```
   $ ssh example.user1@198.51.100.1 hostname
   ```

   b. Enter the passphrase you set while creating the key to grant access to the private key.

   ```
   $ ssh example.user1@198.51.100.1 hostname
    host.example.com
   ```

2. Start the **ssh-agent**.

   ```
   $ eval $(ssh-agent)
   Agent pid 20062
   ```

3. Add the key to **ssh-agent**.

   ```
   $ ssh-add ~/.ssh/id_rsa
   Enter passphrase for ~/.ssh/id_rsa:
   Identity added: ~/.ssh/id_rsa (example.user0@198.51.100.12)
   ```

Verification

- Optional: Log in to the host machine using SSH.

  ```
  $ ssh example.user1@198.51.100.1

  Last login: Mon Sep 14 12:56:37 2020
  ```

  Note that you did not have to enter the passphrase.

5.9. ADDITIONAL RESOURCES

- **sshd(8)**, **ssh(1)**, **scp(1)**, **sftp(1)**, **ssh-keygen(1)**, **ssh-copy-id(1)**, **ssh_config(5)**, **sshd_config(5)**, **update-crypto-policies(8)**, and **crypto-policies(7)** man pages.
- OpenSSH Home Page
- Configuring SELinux for applications and services with non-standard configurations
- Controlling network traffic using firewalld

CHAPTER 6. CONFIGURING BASIC SYSTEM SECURITY

Computer security is the protection of computer systems and their hardware, software, information, and services from theft, damage, disruption, and misdirection. Ensuring computer security is an essential task, in particular in enterprises that process sensitive data and handle business transactions.

This section covers only the basic security features that you can configure after installation of the operating system.

6.1. ENABLING THE FIREWALLD SERVICE

A firewall is a network security system that monitors and controls incoming and outgoing network traffic according to configured security rules. A firewall typically establishes a barrier between a trusted secure internal network and another outside network.

The **firewalld** service, which provides a firewall in Red Hat Enterprise Linux, is automatically enabled during installation.

To enable the **firewalld** service, follow this procedure.

Procedure

- Display the current status of **firewalld**:

```
$ systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
  Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled; vendor preset:
enabled)
  Active: inactive (dead)
...
```

- If **firewalld** is not enabled and running, switch to the **root** user, and start the **firewalld** service and enable to start it automatically after the system restarts:

```
# systemctl enable --now firewalld
```

Verification steps

- Check that **firewalld** is running and enabled:

```
$ systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
  Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; vendor preset:
enabled)
  Active: active (running)
...
```

Additional resources

- Using and configuring firewalld
- **man firewalld(1)**

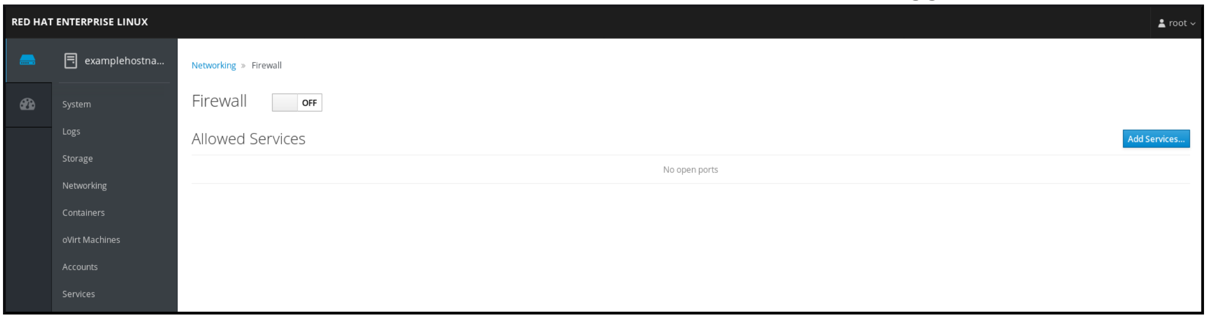
6.2. MANAGING FIREWALL IN THE RHEL 8 WEB CONSOLE

To configure the **firewalld** service in the web console, navigate to **Networking → Firewall**.

By default, the **firewalld** service is enabled.

Procedure

1. To enable or disable **firewalld** in the web console, switch the **Firewall** toggle button.



> **NOTE**
>
> Additionally, you can define more fine-grained access through the firewall to a service using the **Add services…** button.

6.3. MANAGING BASIC SELINUX SETTINGS

Security-Enhanced Linux (SELinux) is an additional layer of system security that determines which processes can access which files, directories, and ports. These permissions are defined in SELinux policies. A policy is a set of rules that guide the SELinux security engine.

SELinux has two possible states:

- Disabled
- Enabled

When SELinux is enabled, it runs in one of the following modes:

- Enabled
  - Enforcing
  - Permissive

In **enforcing mode**, SELinux enforces the loaded policies. SELinux denies access based on SELinux policy rules and enables only the interactions that are explicitly allowed. Enforcing mode is the safest SELinux mode and is the default mode after installation.

In **permissive mode**, SELinux does not enforce the loaded policies. SELinux does not deny access, but reports actions that break the rules to the **/var/log/audit/audit.log** log. Permissive mode is the default mode during installation. Permissive mode is also useful in some specific cases, for example when troubleshooting problems.

Additional resources

- Using SELinux

6.4. ENSURING THE REQUIRED STATE OF SELINUX

By default, SELinux operates in enforcing mode. However, in specific scenarios, you can set SELinux to permissive mode or even disable it.

IMPORTANT

Red Hat recommends to keep your system in enforcing mode. For debugging purposes, you can set SELinux to permissive mode.

Follow this procedure to change the state and mode of SELinux on your system.

Procedure

1. Display the current SELinux mode:

```
$ getenforce
```

2. To temporarily set SELinux:

   a. To Enforcing mode:

   ```
   # setenforce Enforcing
   ```

   b. To Permissive mode:

   ```
   # setenforce Permissive
   ```

   NOTE

   After reboot, SELinux mode is set to the value specified in the **/etc/selinux/config** configuration file.

3. To set SELinux mode to persist across reboots, modify the **SELINUX** variable in the **/etc/selinux/config** configuration file.
For example, to switch SELinux to enforcing mode:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=enforcing
...
```

> **WARNING**
>
> Disabling SELinux reduces your system security. Avoid disabling SELinux using the **SELINUX=disabled** option in the **/etc/selinux/config** file because this can result in memory leaks and race conditions causing kernel panics. Instead, disable SELinux by adding the **selinux=0** parameter to the kernel command line. For more information, see Changing SELinux modes at boot time.

Additional resources

- Changing SELinux states and modes

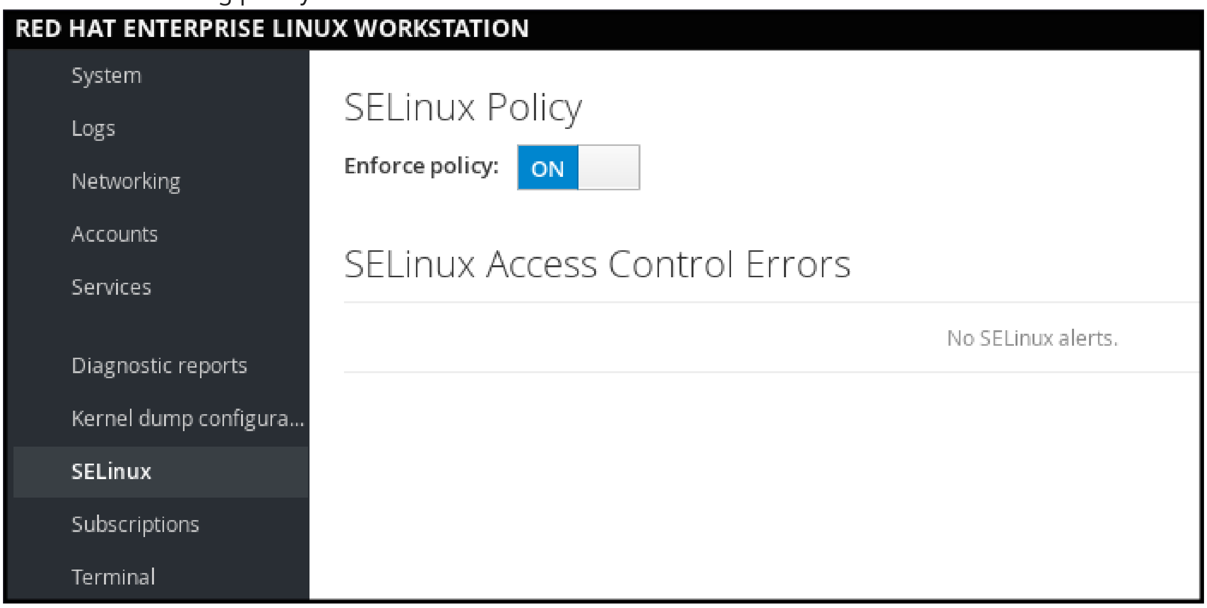
6.5. SWITCHING SELINUX MODES IN THE RHEL 8 WEB CONSOLE

You can set SELinux mode through the RHEL 8 web console in the **SELinux** menu item.

By default, SELinux enforcing policy in the web console is on, and SELinux operates in enforcing mode. By turning it off, you switch SELinux to permissive mode. Note that this selection is automatically reverted on the next boot to the configuration defined in the **/etc/sysconfig/selinux** file.

Procedure

1. In the web console, use the **Enforce policy** toggle button in the SELinux menu item to turn SELinux enforcing policy on or off.



6.6. ADDITIONAL RESOURCES

- Generating SSH key pairs
- Setting an OpenSSH server for key-based authentication
- Security hardening

- Using SELinux
- Securing networks
- Deploying the same SELinux configuration on multiple systems

CHAPTER 7. MANAGING SOFTWARE PACKAGES

7.1. SOFTWARE MANAGEMENT TOOLS IN RHEL 8

In RHEL 8, software installation is enabled by the new version of the **YUM** tool (**YUM v4**), which is based on the **DNF** technology.

> **NOTE**
>
> Upstream documentation identifies the technology as **DNF** and the tool is referred to as **DNF** in the upstream. As a result, some output returned by the new **YUM** tool in RHEL 8 mentions **DNF**.

Although **YUM v4** used in RHEL 8 is based on **DNF**, it is compatible with **YUM v3** used in RHEL 7. For software installation, the **yum** command and most of its options work the same way in RHEL 8 as they did in RHEL 7.

Selected **yum** plug-ins and utilities have been ported to the new DNF back end, and can be installed under the same names as in RHEL 7. Packages also provide compatibility symlinks, so the binaries, configuration files, and directories can be found in usual locations.

Note that the legacy Python API provided by **YUM v3** is no longer available. You can migrate your plug-ins and scripts to the new API provided by **YUM v4** (DNF Python API), which is stable and fully supported. See DNF API Reference for more information.

7.2. APPLICATION STREAMS

RHEL 8 introduces the concept of Application Streams. Multiple versions of user space components are now delivered and updated more frequently than the core operating system packages. This provides greater flexibility to customize Red Hat Enterprise Linux without impacting the underlying stability of the platform or specific deployments.

Components made available as Application Streams can be packaged as modules or RPM packages, and are delivered through the AppStream repository in RHEL 8. Each Application Stream has a given life cycle, either the same as RHEL 8 or shorter, more suitable to the particular application. Application Streams with a shorter life cycle are listed in the Red Hat Enterprise Linux 8 Application Streams Life Cycle page.

Modules are collections of packages representing a logical unit: an application, a language stack, a database, or a set of tools. These packages are built, tested, and released together.

Module streams represent versions of the Application Stream components. For example, two streams (versions) of the PostgreSQL database server are available in the postgresql module: PostgreSQL 10 (the default stream) and PostgreSQL 9.6. Only one module stream can be installed on the system. Different versions can be used in separate containers.

Detailed module commands are described in the Installing, managing, and removing user-space components document. For a list of modules available in AppStream, see the Package manifest.

7.3. SEARCHING FOR SOFTWARE PACKAGES

yum allows you to perform a complete set of operations with software packages.

The following section describes how to use **yum** to:

- Search for packages.
- List packages.
- List repositories.
- Display information about the packages.
- List package groups.
- Specify global expressions in yum input.

7.3.1. Searching packages with YUM

Use the following procedure to find a package providing a particular application or other content.

Procedure

- To search for a package, use:

  ```
  # yum search term
  ```

  Replace *term* with a term related to the package.

  Note that **yum search** command returns term matches within the name and summary of the packages. This makes the search faster and enables you to search for packages you do not know the name of, but for which you know a related term.

- To include term matches within package descriptions, use:

  ```
  # yum search --all term
  ```

  Replace *term* with a term you want to search for in a package name, summary, or description.

  Note that **yum search --all** enables a more exhaustive but slower search.

7.3.2. Listing packages with YUM

Use the following procedure to list installed and available packages.

Procedure

- To list information about all installed and available packages, use:

  ```
  # yum list --all
  ```

- To list all packages installed on your system, use:

  ```
  # yum list --installed
  ```

- To list all packages in all enabled repositories that are available to install, use:

  ```
  # yum list --available
  ```

Note that you can filter the results by appending global expressions as arguments. See Specifying global expressions in yum input

for more details.

7.3.3. Listing repositories with YUM

Use the following procedure to list enabled and disabled repositories.

Procedure

- To list all enabled repositories on your system, use:

```
# yum repolist
```

- To list all disabled repositories on your system, use:

```
# yum repolist --disabled
```

- To list both enabled and disabled repositories, use:

```
# yum repolist --all
```

- To list additional information about the repositories, use:

```
# yum repoinfo
```

Note that you can filter the results by passing the ID or name of repositories as arguments or by appending global expressions. See Specifying global expressions in yum input

for more details.

7.3.4. Displaying package information with YUM

You can display various types of information about a package using YUM, for example version, release, size, loaded plugins, and more.

Procedure

- To display information about one or more packages, use:

```
# yum info package-name
```

  Replace *package-name* with the name of the package.

Note that you can filter the results by appending global expressions as arguments. See Specifying global expressions in yum input

for more details.

7.3.5. Listing package groups with YUM

Use **yum** to view installed package groups and filter the listing results.

Procedure

- To view the number of installed and available groups, use:

  ```
  # yum group summary
  ```

- To list all installed and available groups, use:

  ```
  # yum group list
  ```

  Note that you can filter the results by appending command line options for the **yum group list** command (**--hidden**, **--available**). For more available options see the man pages.

- To list mandatory and optional packages contained in a particular group, use:

  ```
  # yum group info group-name
  ```

  Replace *group-name* with the name of the group.

Note that you can filter the results by appending global expressions as arguments. See Specifying global expressions in yum input

for more details.

7.3.6. Specifying global expressions in YUM input

yum commands allow you to filter the results by appending one or more *glob expressions* as arguments. You have to escape global expressions when passing them as arguments to the **yum** command.

Procedure

> To ensure global expressions are passed to **yum** as intended, use one of the following methods:

- Double-quote or single-quote the entire global expression.

  ```
  # yum provides "*/file-name"
  ```

  Replace *file-name* with the name of the file.

- Escape the wildcard characters by preceding them with a backslash (\) character.

  ```
  # yum provides \*/file-name
  ```

  Replace *file-name* with the name of the file.

7.4. INSTALLING SOFTWARE PACKAGES

The following section describes how to use **yum** to:

- Install packages.
- Install a package group.
- Specify a package name in yum input.

7.4.1. Installing packages with YUM

- To install a package and all the package dependencies, use:

  ```
  # yum install package-name
  ```

  Replace *package-name* with the name of the package.

- To install multiple packages and their dependencies simultaneously, use:

  ```
  # yum install package-name-1 package-name-2
  ```

  Replace *package-name-1* and *package-name-2* with the names of the packages.

- When installing packages on a *multilib* system (AMD64, Intel 64 machine), you can specify the architecture of the package by appending it to the package name:

  ```
  # yum install package-name.arch
  ```

  Replace *package-name.arch* with the name and architecture of the package.

- If you know the name of the binary you want to install, but not the package name, you can use the path to the binary as an argument:

  ```
  # yum install /usr/sbin/binary-file
  ```

  Replace ***/usr/sbin/binary-file*** with a path to the binary file.

  yum searches through the package lists, finds the package which provides ***/usr/sbin/binary-file***, and prompts you as to whether you want to install it.

- To install a previously-downloaded package from a local directory, use:

  ```
  # yum install /path/
  ```

  Replace */path/* with the path to the package.

Note that you can optimize the package search by explicitly defining how to parse the argument. See Section 7.4.3, "Specifying a package name in YUM input" for more details.

7.4.2. Installing a package group with YUM

The following procedure describes how to install a package group by a group name or by a groupID using **yum**.

Procedure

- To install a package group by a group name, use:

  ```
  # yum group install group-name
  ```

  Or

  ```
  # yum install @group-name
  ```

Replace *group-name* with the full name of the group or environmental group.

- To install a package group by the groupID, use:

  ```
  # yum group install groupID
  ```

  Replace *groupID* with the ID of the group.

7.4.3. Specifying a package name in YUM input

> To optimize the installation and removal process, you can append **-n**, **-na**, or **-nevra** suffixes to **yum install** and **yum remove** commands to explicitly define how to parse an argument:

- To install a package using its exact name, use:

  ```
  # yum install-n name
  ```

  Replace *name* with the exact name of the package.

- To install a package using its exact name and architecture, use:

  ```
  # yum install-na name.architecture
  ```

  Replace *name* and *architecture* with the exact name and architecture of the package.

- To install a package using its exact name, epoch, version, release, and architecture, use:

  ```
  # yum install-nevra name-epoch:version-release.architecture
  ```

  Replace *name*, *epoch*, *version*, *release*, and *architecture* with the exact name, epoch, version, release, and architecture of the package.

7.5. UPDATING SOFTWARE PACKAGES

yum allows you to check if your system has any pending updates. You can list packages that need updating and choose to update a single package, multiple packages, or all packages at once. If any of the packages you choose to update have dependencies, they are updated as well.

The following section describes how to use **yum** to:

- Check for updates.
- Update a single package.
- Update a package group.
- Update all packages and their dependencies.
- Apply security updates.
- Automate software updates.

7.5.1. Checking for updates with YUM

The following procedure describes how to check the available updates for packages installed on your system using **yum**.

Procedure

- To see which packages installed on your system have available updates, use:

```
# yum check-update
```

  The output returns the list of packages and their dependencies that have an update available.

7.5.2. Updating a single package with YUM

Use the following procedure to update a single package and its dependencies using **yum**.

> **IMPORTANT**
>
> When applying updates to kernel, **yum** always **installs** a new kernel regardless of whether you are using the **yum update** or **yum install** command.

- To update a package, use:

```
# yum update package-name
```

  Replace *package-name* with the name of the package.

> **IMPORTANT**
>
> If you upgraded the GRUB boot loader packages on a BIOS or IBM Power system, reinstall GRUB. See Reinstalling GRUB.

7.5.3. Updating a package group with YUM

Use the following procedure to update a group of packages and their dependencies using **yum**.

Procedure

- To update a package group, use:

```
# yum group update group-name
```

  Replace *group-name* with the name of the package group.

> **IMPORTANT**
>
> If you upgraded the GRUB boot loader packages on a BIOS or IBM Power system, reinstall GRUB. See Reinstalling GRUB.

7.5.4. Updating all packages and their dependencies with YUM

Use the following procedure to update all packages and their dependencies using **yum**.

Procedure

- To update all packages and their dependencies, use:

```
# yum update
```

> **IMPORTANT**
>
> If you upgraded the GRUB boot loader packages on a BIOS or IBM Power system, reinstall GRUB. See Reinstalling GRUB.

7.5.5. Updating security-related packages with YUM

Use the following procedure to update packages available packages that have security errata using **yum**.

Procedure

- To upgrade to the latest available packages that have security errata, use:

```
# yum update --security
```

- To upgrade to the last security errata packages, use:

```
# yum update-minimal --security
```

> **IMPORTANT**
>
> If you upgraded the GRUB boot loader packages on a BIOS or IBM Power system, reinstall GRUB. See Reinstalling GRUB.

7.5.6. Automating software updates

To check and download package updates automatically and regularly, you can use the **DNF Automatic** tool that is provided by the **dnf-automatic** package.

DNF Automatic is an alternative command-line interface to **yum** that is suited for automatic and regular execution using systemd timers, cron jobs and other such tools.

DNF Automatic synchronizes package metadata as needed and then checks for updates available. After, the tool can perform one of the following actions depending on how you configure it:

- Exit
- Download updated packages
- Download and apply the updates

The outcome of the operation is then reported by a selected mechanism, such as the standard output or email.

7.5.6.1. Installing DNF Automatic

The following procedure describes how to install the **DNF Automatic** tool.

Procedure

- To install the **dnf-automatic** package, use:

```
# yum install dnf-automatic
```

Verification steps

- To verify the successful installation, confirm the presence of the **dnf-automatic** package by running the following command:

```
# rpm -qi dnf-automatic
```

7.5.6.2. DNF Automatic configuration file

By default, **DNF Automatic** uses **/etc/dnf/automatic.conf** as its configuration file to define its behavior.

The configuration file is separated into the following topical sections:

- **[commands]** section
  Sets the mode of operation of **DNF Automatic**.

- **[emitters]** section
  Defines how the results of **DNF Automatic** are reported.

- **[command_email]** section
  Provides the email emitter configuration for an external command used to send email.

- **[email]** section
  Provides the email emitter configuration.

- **[base]** section
  Overrides settings from the main configuration file of yum.

With the default settings of the **/etc/dnf/automatic.conf** file, **DNF Automatic** checks for available updates, downloads them, and reports the results as standard output.

> **WARNING**
>
> Settings of the operation mode from the **[commands]** section are overridden by settings used by a systemd timer unit for all timer units except **dnf-automatic.timer**.

Additional resources

- For more details on particular sections, see DNF Automatic documentation.

- For more details on systemd timer units, see the **man dnf-automatic** manual pages.

- For the overview of the systemd timer units included in the **dnf-automatic package**, see Section Overview of the systemd timer units included in the dnf-automatic package Overview of the systemd timer units included in the dnf-automatic package

7.5.6.3. Enabling DNF Automatic

To run **DNF Automatic**, you always need to enable and start a specific systemd timer unit. You can use one of the timer units provided in the **dnf-automatic** package, or you can write your own timer unit depending on your needs.

The following section describes how to enable **DNF Automatic**.

Prerequisites

- You specified the behavior of DNF Automatic by modifying the **/etc/dnf/automatic.conf** configuration file.

For more information about **DNF Automatic** configuration file, see Section 2.5.6.2, “DNF Automatic configuration file”.

Procedure

- Select, enable and start a systemd timer unit that fits your needs:

```
# systemctl enable --now <unit>
```

where **<unit>** is one of the following timers:

- **dnf-automatic-download.timer**
- **dnf-automatic-install.timer**
- **dnf-automatic-notifyonly.timer**
- **dnf-automatic.timer**
  - To **download** available updates, use:

    ```
    # systemctl enable dnf-automatic-download.timer
    # systemctl start dnf-automatic-download.timer
    ```

  - To **download and install** available updates, use:

    ```
    # systemctl enable dnf-automatic-install.timer
    # systemctl start dnf-automatic-install.timer
    ```

  - To **report** about available updates, use:

    ```
    # systemctl enable dnf-automatic-notifyonly.timer
    # systemctl start dnf-automatic-notifyonly.timer
    ```

  - Optionally, you can use:

```
# systemctl enable dnf-automatic.timer
# systemctl start dnf-automatic.timer
```

In terms of downloading and applying updates, this timer unit behaves according to settings in the **/etc/dnf/automatic.conf** configuration file. The default behavior is similar to **dnf-automatic-download.timer**: it downloads the updated packages, but it does not install them.

NOTE

Alternatively, you can also run **DNF Automatic** by executing the **/usr/bin/dnf-automatic** file directly from the command line or from a custom script.

Verification steps

- To verify that the timer is enabled, run the following command:

```
# systemctl status <systemd timer unit>
```

Additional resources

- For more information about the dnf-automatic timers, see the **man dnf-automatic** manual pages.
- For the overview of the systemd timer units included in the **dnf-automatic** package, see Section Overview of the systemd timer units included in the dnf-automatic package

7.5.6.4. Overview of the systemd timer units included in the dnf-automatic package

The systemd timer units take precedence and override the settings in the **/etc/dnf/automatic.conf** configuration file concerning downloading and applying updates.

For example if you set the following option in the **/etc/dnf/automatic.conf** configuration file, but you have activated the **dnf-automatic-notifyonly.timer** unit, the packages will not be downloaded:

```
download_updates = yes
```

The **dnf-automatic** package includes the following systemd timer units:

Timer unit	Function	Overrides settings in the **/etc/dnf/automatic.conf** file?
dnf-automatic-download.timer	Downloads packages to cache and makes them available for updating. Note: This timer unit does not install the updated packages. To perform the installation, you have to execute the **dnf update** command.	Yes

Timer unit	Function	Overrides settings in the **/etc/dnf/automatic.conf** file?
dnf-automatic-install.timer	Downloads and installs updated packages.	Yes
dnf-automatic-notifyonly.timer	Downloads only repository data to keep repository cache up-to-date and notifies you about available updates. Note: This timer unit does not download or install the updated packages	Yes
dnf-automatic.timer	The behavior of this timer concerning downloading and applying updates is specified by the settings in the **/etc/dnf/automatic.conf** configuration file. Default behavior is the same as for the **dnf-automatic-download.timer** unit: it only downloads packages, but does not install them.	No

Additional resources

- For more information about the **dnf-automatic** timers, see the **man dnf-automatic** manual pages.
- For more information about the **/etc/dnf/automatic.conf** configuration file, see Section DNF Automatic configuration file

7.6. UNINSTALLING SOFTWARE PACKAGES

The following section describes how to use **yum** to:

- Remove packages.
- Remove a package group.
- Specify a package name in yum input.

7.6.1. Removing packages with YUM

Use the following procedure to remove a package either by the group name or the groupID.

Procedure

- To remove a particular package and all dependent packages, use:

```
# yum remove package-name
```

Replace *package-name* with the name of the package.

- To remove multiple packages and their dependencies simultaneously, use:

```
# yum remove package-name-1 package-name-2
```

Replace *package-name-1* and *package-name-2* with the names of the packages.

NOTE

yum is not able to remove a package without removing depending packages.

Note that you can optimize the package search by explicitly defining how to parse the argument. See Specifying a package name in yum input for more details.

7.6.2. Removing a package group with YUM

Use the following procedure to remove a package either by the group name or the groupID.

Procedure

- To remove a package group by the group name, use:

```
# yum group remove group-name
```

Or

```
# yum remove @group-name
```

Replace *group-name* with the full name of the group.

- To remove a package group by the groupID, use:

```
# yum group remove groupID
```

Replace *groupID* with the ID of the group.

7.6.3. Specifying a package name in YUM input

To optimize the installation and removal process, you can append **-n**, **-na**, or **-nevra** suffixes to **yum install** and **yum remove** commands to explicitly define how to parse an argument:

- To install a package using its exact name, use:

```
# yum install-n name
```

Replace *name* with the exact name of the package.

- To install a package using its exact name and architecture, use:

```
# yum install-na name.architecture
```

Replace *name* and *architecture* with the exact name and architecture of the package.

- To install a package using its exact name, epoch, version, release, and architecture, use:

```
# yum install-nevra name-epoch:version-release.architecture
```

Replace *name*, *epoch*, *version*, *release*, and *architecture* with the exact name, epoch, version, release, and architecture of the package.

7.7. MANAGING SOFTWARE PACKAGE GROUPS

A package group is a collection of packages that serve a common purpose (**System Tools**, **Sound and Video**). Installing a package group pulls a set of dependent packages, which saves time considerably.

The following section describes how to use **yum** to:

- List package groups.
- Install a package group.
- Remove a package group.
- Specify global expressions in yum input.

7.7.1. Listing package groups with YUM

Use **yum** to view installed package groups and filter the listing results.

Procedure

- To view the number of installed and available groups, use:

```
# yum group summary
```

- To list all installed and available groups, use:

```
# yum group list
```

Note that you can filter the results by appending command line options for the **yum group list** command (**--hidden**, **--available**). For more available options see the man pages.

- To list mandatory and optional packages contained in a particular group, use:

```
# yum group info group-name
```

Replace *group-name* with the name of the group.

Note that you can filter the results by appending global expressions as arguments. See Specifying global expressions in yum input

for more details.

7.7.2. Installing a package group with YUM

The following procedure describes how to install a package group by a group name or by a groupID using **yum**.

Procedure

- To install a package group by a group name, use:

  ```
  # yum group install group-name
  ```

  Or

  ```
  # yum install @group-name
  ```

  Replace *group-name* with the full name of the group or environmental group.

- To install a package group by the groupID, use:

  ```
  # yum group install groupID
  ```

  Replace *groupID* with the ID of the group.

7.7.3. Removing a package group with YUM

Use the following procedure to remove a package either by the group name or the groupID.

Procedure

- To remove a package group by the group name, use:

  ```
  # yum group remove group-name
  ```

  Or

  ```
  # yum remove @group-name
  ```

  Replace *group-name* with the full name of the group.

- To remove a package group by the groupID, use:

  ```
  # yum group remove groupID
  ```

  Replace *groupID* with the ID of the group.

7.7.4. Specifying global expressions in YUM input

yum commands allow you to filter the results by appending one or more *glob expressions* as arguments. You have to escape global expressions when passing them as arguments to the **yum** command.

Procedure

> To ensure global expressions are passed to **yum** as intended, use one of the following methods:

- Double-quote or single-quote the entire global expression.

```
# yum provides "*/file-name"
```

Replace *file-name* with the name of the file.

- Escape the wildcard characters by preceding them with a backslash (\) character.

```
# yum provides \*/file-name
```

Replace *file-name* with the name of the file.

7.8. HANDLING PACKAGE MANAGEMENT HISTORY

The **yum history** command allows you to review information about the timeline of **yum** transactions, dates and times they occurred, the number of packages affected, whether these transactions succeeded or were aborted, and if the RPM database was changed between transactions. **yum history** command can also be used to undo or redo the transactions.

The following section describes how to use **yum** to:

- List transactions.
- Revert transactions.
- Repeat transactions.
- Specify global expressions in yum input.

7.8.1. Listing transactions with YUM

Use the following procedure to list the latest transactions, the latest operations for a selected package, and details of a particular transaction.

Procedure

- To display a list of all the latest **yum** transactions, use:

```
# yum history
```

- To display a list of all the latest operations for a selected package, use:

```
# yum history list package-name
```

Replace *package-name* with the name of the package. You can filter the command output by appending global expressions. See Specifying global expressions in yum input for more details.

- To examine a particular transaction, use:

```
# yum history info transactionID
```

Replace *transactionID* with the ID of the transaction.

7.8.2. Reverting transactions with YUM

The following procedure describes how to revert a selected transaction or the last transaction using **yum**.

Procedure

- To revert a particular transaction, use:

  ```
  # yum history undo transactionID
  ```

  Replace *transactionID* with the ID of the transaction.

- To revert the last transaction, use:

  ```
  # yum history undo last
  ```

Note that the **yum history undo** command only reverts the steps that were performed during the transaction. If the transaction installed a new package, the **yum history undo** command uninstalls it. If the transaction uninstalled a package, the **yum history undo** command reinstalls it. **yum history undo** also attempts to downgrade all updated packages to their previous versions, if the older packages are still available.

7.8.3. Repeating transactions with YUM

Use the following procedure to repeat a selected transaction or the last transaction using **yum**.

Procedure

- To repeat a particular transaction, use:

  ```
  # yum history redo transactionID
  ```

  Replace *transactionID* with the ID of the transaction.

- To repeat the last transaction, use:

  ```
  # yum history redo last
  ```

Note that the **yum history redo** command only repeats the steps that were performed during the transaction.

7.8.4. Specifying global expressions in YUM input

yum commands allow you to filter the results by appending one or more *glob expressions* as arguments. You have to escape global expressions when passing them as arguments to the **yum** command.

Procedure

> To ensure global expressions are passed to **yum** as intended, use one of the following methods:

- Double-quote or single-quote the entire global expression.

```
# yum provides "*/file-name"
```

Replace *file-name* with the name of the file.

- Escape the wildcard characters by preceding them with a backslash (\) character.

```
# yum provides \*/file-name
```

Replace *file-name* with the name of the file.

7.9. MANAGING SOFTWARE REPOSITORIES

The configuration information for **yum** and related utilities are stored in the **/etc/yum.conf** file. This file contains one or more **[*repository*]** sections, which allow you to set repository-specific options.

It is recommended to define individual repositories in new or existing **.repo** files in the **/etc/yum.repos.d/** directory.

Note that the values you define in individual **[*repository*]** sections of the **/etc/yum.conf** file override values set in the **[main]** section.

The following section describes how to:

- Set **[*repository*]** options.
- Add a **yum** repository.
- Enable a **yum** repository.
- Disable a **yum** repository.

7.9.1. Setting YUM repository options

The **/etc/yum.conf** configuration file contains the **[*repository*]** sections, where *repository* is a unique repository ID. The **[*repository*]** sections allows you to define individual **yum** repositories.

> **NOTE**
>
> Do not give custom repositories names used by the Red Hat repositories to avoid conflicts.

For a complete list of available **[*repository*]** options, see the **[*repository*] OPTIONS** section of the **yum.conf**(5) manual page.

7.9.2. Adding a YUM repository

Procedure

To define a new repository, you can:

- Add a **[*repository*]** section to the **/etc/yum.conf** file.
- Add a **[*repository*]** section to a **.repo** file in the **/etc/yum.repos.d/** directory.
  yum repositories commonly provide their own **.repo** file.

> **NOTE**
>
> It is recommended to define your repositories in a **.repo** file instead of **/etc/yum.conf** as all files with the **.repo** file extension in this directory are read by **yum**.

- To add a repository to your system and enable it, use:

  ```
  # yum-config-manager --add-repo repository_URL
  ```

  Replace *repository_url* with URL pointing to the repository.

> **WARNING**
>
> Obtaining and installing software packages from unverified or untrusted sources other than Red Hat certificate-based **Content Delivery Network** (CDN) constitutes a potential security risk, and could lead to security, stability, compatibility, and maintainability issues.

7.9.3. Enabling a YUM repository

Once you added a **yum** repository to your system, enable it to ensure installation and updates.

Procedure

- To enable a repository, use:

  ```
  # yum-config-manager --enable repositoryID
  ```

  Replace *repositoryID* with the unique repository ID.

  To list available repository IDs, see Listing packages with yum.

7.9.4. Disabling a YUM repository

Disable a specific YUM repository to prevent particular packages from installation or update.

Procedure

- To disable a yum repository, use:

  ```
  # yum-config-manager --disable repositoryID
  ```

  Replace *repositoryID* with the unique repository ID.

  To list available repository IDs, see Listing packages with yum.

7.10. CONFIGURING YUM

The configuration information for **yum** and related utilities are stored in the **/etc/yum.conf** file. This file contains one mandatory **[main]** section, which enables you to set **yum** options that have global effect.

The following section describes how to:

- View the current **yum** configurations.
- Set **yum** [main] options.
- Use **yum** plug-ins.

7.10.1. Viewing the current YUM configurations

Use the following procedure to view the current **yum** configurations.

Procedure

- To display the current values of global **yum** options specified in the **[main]** section of the **/etc/yum.conf** file, use:

  ```
  # yum config-manager --dump
  ```

7.10.2. Setting YUM main options

The **/etc/yum.conf** configuration file contains one **[main]** section. The key-value pairs listed below affect how **yum** operates and treats repositories.

You can add additional options under the **[main]** section heading in **/etc/yum.conf**.

For a complete list of available **[main]** options, see the **[main] OPTIONS** section of the **yum.conf**(5) manual page.

7.10.3. Using YUM plug-ins

yum provides plug-ins that extend and enhance its operations. Certain plug-ins are installed by default.

The following section describes how to enable, configure, and disable **yum** plug-ins.

7.10.3.1. Managing YUM plug-ins

Procedure

The plug-in configuration files always contain a **[main]** section where the **enabled=** option controls whether the plug-in is enabled when you run **yum** commands. If this option is missing, you can add it manually to the file.

Every installed plug-in has its own configuration file in the **/etc/dnf/plugins/** directory. You can enable or disable plug-in specific options in these files.

7.10.3.2. Enabling YUM plug-ins

The following procedure describes how to disable or enable all **YUM** plug-ins, disable all plug-ins for a particular command, or certain **YUM** plug-ins for a single command.

Procedure

- To enable all yum plug-ins:

    1. Ensure a line beginning with **plugins=** is present in the **[main]** section of the **/etc/yum.conf** file.

    2. Set the value of **plugins=** to **1**.

```
plugins=1
```

7.10.3.3. Disabling YUM plug-ins

- To disable all yum plug-ins:

    1. Ensure a line beginning with **plugins=** is present in the **[main]** section of the **/etc/yum.conf** file.

    2. Set the value of **plugins=** to **0**.

```
plugins=0
```

> **IMPORTANT**
>
> Disabling all plug-ins is **not** advised. Certain plug-ins provide important yum services. In particular, the **product-id** and **subscription-manager** plug-ins provide support for the certificate-based **Content Delivery Network** (CDN). Disabling plug-ins globally is provided as a convenience option, and is advisable only when diagnosing a potential problem with **yum**.

- To disable all yum plug-ins for a particular command, append **--noplugins** option to the command.

```
# yum --noplugins update
```

- To disable certain yum plug-ins for a single command, append **--disableplugin=*plugin-name*** option to the command.

```
# yum update --disableplugin=plugin-name
```

Replace *plugin-name* with the name of the plug-in.

CHAPTER 8. INTRODUCTION TO RHEL SYSTEM ROLES

You can automate system administration across multiple systems with RHEL system roles.

RHEL system roles is a collection of Ansible roles and modules. By using RHEL system roles, you can remotely manage the system configurations of multiple RHEL systems across major versions of RHEL. To use it to configure systems, you must use the following components:

Control node

A control node is the system from which you run Ansible commands and playbooks. Your control node can be an Ansible Automation Platform, Red Hat Satellite, or a RHEL 9, 8, or 7 host. For more information, see Preparing a control node on RHEL 8 .

Managed node

Managed nodes are the servers and network devices that you manage with Ansible. Managed nodes are also sometimes called hosts. Ansible does not have to be installed on managed nodes. For more information, see Preparing a managed node .

Ansible playbook

In a playbook, you define the configuration you want to achieve on your managed nodes or a set of steps for the system on the managed node to perform. Playbooks are Ansible's configuration, deployment, and orchestration language.

Inventory

In an inventory file, you list the managed nodes and specify information such as IP address for each managed node. In the inventory, you can also organize the managed nodes by creating and nesting groups for easier scaling. An inventory file is also sometimes called a hostfile.

On Red Hat Enterprise Linux 8, you can use the following roles provided by the **rhel-system-roles** package, which is available in the **AppStream** repository:

Role name	Role description	Chapter title
certificate	Certificate Issuance and Renewal	Requesting certificates by using RHEL system roles
cockpit	Web console	Installing and configuring web console with the cockpit RHEL system role
crypto_policies	System-wide cryptographic policies	Setting a custom cryptographic policy across systems
firewall	Firewalld	Configuring firewalld by using system roles
ha_cluster	HA Cluster	Configuring a high-availability cluster by using system roles
kdump	Kernel Dumps	Configuring kdump by using RHEL system roles
kernel_settings	Kernel Settings	Using Ansible roles to permanently configure kernel parameters

Role name	Role description	Chapter title
logging	Logging	Using the logging system role
metrics	Metrics (PCP)	Monitoring performance by using RHEL system roles
microsoft.sql.server	Microsoft SQL Server	Configuring Microsoft SQL Server by using the microsoft.sql.server Ansible role
network	Networking	Using the network RHEL system role to manage InfiniBand connections
nbde_client	Network Bound Disk Encryption client	Using the nbde_client and nbde_server system roles
nbde_server	Network Bound Disk Encryption server	Using the nbde_client and nbde_server system roles
postfix	Postfix	Variables of the postfix role in system roles
postgresql	PostgreSQL	Installing and configuring PostgreSQL by using the postgresql RHEL system role
selinux	SELinux	Configuring SELinux by using system roles
ssh	SSH client	Configuring secure communication with the ssh system roles
sshd	SSH server	Configuring secure communication with the ssh system roles
storage	Storage	Managing local storage by using RHEL system roles
tlog	Terminal Session Recording	Configuring a system for session recording by using the tlog RHEL system role
timesync	Time Synchronization	Configuring time synchronization by using RHEL system roles
vpn	VPN	Configuring VPN connections with IPsec by using the vpn RHEL system role

Additional resources

- Automating system administration by using RHEL system roles
- Red Hat Enterprise Linux (RHEL) system roles
- **/usr/share/ansible/roles/rhel-system-roles.<*role_name*>/README.md** file

- **/usr/share/doc/rhel-system-roles/*<role_name>*/** directory

CHAPTER 9. CONFIGURING LOGGING

Most services in Red Hat Enterprise Linux log status messages, warnings, and errors. You can use the **rsyslogd** service to log these entries to local files or to a remote logging server.

9.1. CONFIGURING A REMOTE LOGGING SOLUTION

To ensure that logs from various machines in your environment are recorded centrally on a logging server, you can configure the **Rsyslog** application to record logs that fit specific criteria from the client system to the server.

9.1.1. The Rsyslog logging service

The Rsyslog application, in combination with the **systemd-journald** service, provides local and remote logging support in Red Hat Enterprise Linux. The **rsyslogd** daemon continuously reads **syslog** messages received by the **systemd-journald** service from the Journal. **rsyslogd** then filters and processes these **syslog** events and records them to **rsyslog** log files or forwards them to other services according to its configuration.

The **rsyslogd** daemon also provides extended filtering, encryption protected relaying of messages, input and output modules, and support for transportation using the TCP and UDP protocols.

In **/etc/rsyslog.conf**, which is the main configuration file for **rsyslog**, you can specify the rules according to which **rsyslogd** handles the messages. Generally, you can classify messages by their source and topic (facility) and urgency (priority), and then assign an action that should be performed when a message fits these criteria.

In **/etc/rsyslog.conf**, you can also see a list of log files maintained by **rsyslogd**. Most log files are located in the **/var/log/** directory. Some applications, such as **httpd** and **samba**, store their log files in a subdirectory within **/var/log/**.

Additional resources

- The **rsyslogd(8)** and **rsyslog.conf(5)** man pages.
- Documentation installed with the **rsyslog-doc** package in the **/usr/share/doc/rsyslog/html/index.html** file.

9.1.2. Installing Rsyslog documentation

The Rsyslog application has extensive online documentation that is available at https://www.rsyslog.com/doc/, but you can also install the **rsyslog-doc** documentation package locally.

Prerequisites

- You have activated the **AppStream** repository on your system.
- You are authorized to install new packages using **sudo**.

Procedure

- Install the **rsyslog-doc** package:

```
# yum install rsyslog-doc
```

Verification

- Open the **/usr/share/doc/rsyslog/html/index.html** file in a browser of your choice, for example:

```
$ firefox /usr/share/doc/rsyslog/html/index.html &
```

9.1.3. Configuring a server for remote logging over TCP

The Rsyslog application enables you to both run a logging server and configure individual systems to send their log files to the logging server. To use remote logging through TCP, configure both the server and the client. The server collects and analyzes the logs sent by one or more client systems.

With the Rsyslog application, you can maintain a centralized logging system where log messages are forwarded to a server over the network. To avoid message loss when the server is not available, you can configure an action queue for the forwarding action. This way, messages that failed to be sent are stored locally until the server is reachable again. Note that such queues cannot be configured for connections using the UDP protocol.

The **omfwd** plug-in provides forwarding over UDP or TCP. The default protocol is UDP. Because the plug-in is built in, it does not have to be loaded.

By default, **rsyslog** uses TCP on port **514**.

Prerequisites

- Rsyslog is installed on the server system.
- You are logged in as **root** on the server.
- The **policycoreutils-python-utils** package is installed for the optional step using the **semanage** command.
- The **firewalld** service is running.

Procedure

1. Optional: To use a different port for **rsyslog** traffic, add the **syslogd_port_t** SELinux type to port. For example, enable port **30514**:

```
# semanage port -a -t syslogd_port_t -p tcp 30514
```

2. Optional: To use a different port for **rsyslog** traffic, configure **firewalld** to allow incoming **rsyslog** traffic on that port. For example, allow TCP traffic on port **30514**:

```
# firewall-cmd --zone=<zone-name> --permanent --add-port=30514/tcp
success
# firewall-cmd --reload
```

3. Create a new file in the **/etc/rsyslog.d/** directory named, for example, **remotelog.conf**, and insert the following content:

```
# Define templates before the rules that use them
# Per-Host templates for remote systems
template(name="TmplAuthpriv" type="list") {
```

```
    constant(value="/var/log/remote/auth/")
    property(name="hostname")
    constant(value="/")
    property(name="programname" SecurePath="replace")
    constant(value=".log")
    }

template(name="TmplMsg" type="list") {
    constant(value="/var/log/remote/msg/")
    property(name="hostname")
    constant(value="/")
    property(name="programname" SecurePath="replace")
    constant(value=".log")
    }

# Provides TCP syslog reception
module(load="imtcp")

# Adding this ruleset to process remote messages
ruleset(name="remote1"){
     authpriv.*   action(type="omfile" DynaFile="TmplAuthpriv")
      *.info;mail.none;authpriv.none;cron.none
action(type="omfile" DynaFile="TmplMsg")
}

input(type="imtcp" port="30514" ruleset="remote1")
```

4. Save the changes to the **/etc/rsyslog.d/remotelog.conf** file.

5. Test the syntax of the **/etc/rsyslog.conf** file:

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-2.el8, config validation run...
rsyslogd: End of config validation run. Bye.
```

6. Make sure the **rsyslog** service is running and enabled on the logging server:

```
# systemctl status rsyslog
```

7. Restart the **rsyslog** service.

```
# systemctl restart rsyslog
```

8. Optional: If **rsyslog** is not enabled, ensure the **rsyslog** service starts automatically after reboot:

```
# systemctl enable rsyslog
```

Your log server is now configured to receive and store log files from the other systems in your environment.

Additional resources

- **rsyslogd(8)**, **rsyslog.conf(5)**, **semanage(8)**, and **firewall-cmd(1)** man pages.

- Documentation installed with the **rsyslog-doc** package in the **/usr/share/doc/rsyslog/html/index.html** file.

9.1.4. Configuring remote logging to a server over TCP

Follow this procedure to configure a system for forwarding log messages to a server over the TCP protocol. The **omfwd** plug-in provides forwarding over UDP or TCP. The default protocol is UDP. Because the plug-in is built in, you do not have to load it.

Prerequisites

- The **rsyslog** package is installed on the client systems that should report to the server.
- You have configured the server for remote logging.
- The specified port is permitted in SELinux and open in firewall.
- The system contains the **policycoreutils-python-utils** package, which provides the **semanage** command for adding a non-standard port to the SELinux configuration.

Procedure

1. Create a new file in the **/etc/rsyslog.d/** directory named, for example, **10-remotelog.conf**, and insert the following content:

```
*.* action(type="omfwd"
      queue.type="linkedlist"
      queue.filename="example_fwd"
      action.resumeRetryCount="-1"
      queue.saveOnShutdown="on"
      target="example.com" port="30514" protocol="tcp"
     )
```

Where:

- The **queue.type="linkedlist"** setting enables a LinkedList in-memory queue,
- The **queue.filename** setting defines a disk storage. The backup files are created with the **example_fwd** prefix in the working directory specified by the preceding global **workDirectory** directive.
- The **action.resumeRetryCount -1** setting prevents **rsyslog** from dropping messages when retrying to connect if server is not responding,
- The **queue.saveOnShutdown="on"** setting saves in-memory data if **rsyslog** shuts down.
- The last line forwards all received messages to the logging server. Port specification is optional.
  With this configuration, **rsyslog** sends messages to the server but keeps messages in memory if the remote server is not reachable. A file on disk is created only if **rsyslog** runs out of the configured memory queue space or needs to shut down, which benefits the system performance.

> **NOTE**
>
> Rsyslog processes configuration files **/etc/rsyslog.d/** in the lexical order.

2. Restart the **rsyslog** service.

```
# systemctl restart rsyslog
```

Verification

To verify that the client system sends messages to the server, follow these steps:

1. On the client system, send a test message:

```
# logger test
```

2. On the server system, view the **/var/log/messages** log, for example:

```
# cat /var/log/remote/msg/hostname/root.log
Feb 25 03:53:17 hostname root[6064]: test
```

Where *hostname* is the host name of the client system. Note that the log contains the user name of the user that entered the **logger** command, in this case **root**.

Additional resources

- **rsyslogd(8)** and **rsyslog.conf(5)** man pages.
- Documentation installed with the **rsyslog-doc** package in the **/usr/share/doc/rsyslog/html/index.html** file.

9.1.5. Configuring TLS-encrypted remote logging

By default, Rsyslog sends remote-logging communication in the plain text format. If your scenario requires to secure this communication channel, you can encrypt it using TLS.

To use encrypted transport through TLS, configure both the server and the client. The server collects and analyzes the logs sent by one or more client systems.

You can use either the **ossl** network stream driver (OpenSSL) or the **gtls** stream driver (GnuTLS).

> **NOTE**
>
> If you have a separate system with higher security, for example, a system that is not connected to any network or has stricter authorizations, use the separate system as the certifying authority (CA).

Prerequisites

- You have **root** access to both the client and server systems.
- The **rsyslog** and **rsyslog-openssl** packages are installed on the server and the client systems.

- If you use the **gtls** network stream driver, install the **rsyslog-gnutls** package instead of **rsyslog-openssl**.
- If you generate certificates using the **certtool** command, install the **gnutls-utils** package.
- On your logging server, the following certificates are in the **/etc/pki/ca-trust/source/anchors/** directory and your system configuration is updated by using the **update-ca-trust** command:
  - **ca-cert.pem** - a CA certificate that can verify keys and certificates on logging servers and clients.
  - **server-cert.pem** - a public key of the logging server.
  - **server-key.pem** - a private key of the logging server.
- On your logging clients, the following certificates are in the **/etc/pki/ca-trust/source/anchors/** directory and your system configuration is updated by using **update-ca-trust**:
  - **ca-cert.pem** - a CA certificate that can verify keys and certificates on logging servers and clients.
  - **client-cert.pem** - a public key of a client.
  - **client-key.pem** - a private key of a client.

Procedure

1. Configure the server for receiving encrypted logs from your client systems:
   a. Create a new file in the **/etc/rsyslog.d/** directory named, for example, **securelogser.conf**.
   b. To encrypt the communication, the configuration file must contain paths to certificate files on your server, a selected authentication method, and a stream driver that supports TLS encryption. Add the following lines to the **/etc/rsyslog.d/securelogser.conf** file:

```
# Set certificate files
global(
  DefaultNetstreamDriverCAFile="/etc/pki/ca-trust/source/anchors/ca-cert.pem"
  DefaultNetstreamDriverCertFile="/etc/pki/ca-trust/source/anchors/server-cert.pem"
  DefaultNetstreamDriverKeyFile="/etc/pki/ca-trust/source/anchors/server-key.pem"
)

# TCP listener
module(
  load="imtcp"
  PermittedPeer=["client1.example.com", "client2.example.com"]
  StreamDriver.AuthMode="x509/name"
  StreamDriver.Mode="1"
  StreamDriver.Name="ossl"
)

# Start up listener at port 514
input(
  type="imtcp"
  port="514"
)
```

NOTE

If you prefer the GnuTLS driver, use the **StreamDriver.Name="gtls"** configuration option. See the documentation installed with the **rsyslog-doc** package for more information about less strict authentication modes than **x509/name**.

c. Save the changes to the **/etc/rsyslog.d/securelogser.conf** file.

d. Verify the syntax of the **/etc/rsyslog.conf** file and any files in the **/etc/rsyslog.d/** directory:

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-2.el8, config validation run (level 1)...
rsyslogd: End of config validation run. Bye.
```

e. Make sure the **rsyslog** service is running and enabled on the logging server:

```
# systemctl status rsyslog
```

f. Restart the **rsyslog** service:

```
# systemctl restart rsyslog
```

g. Optional: If Rsyslog is not enabled, ensure the **rsyslog** service starts automatically after reboot:

```
# systemctl enable rsyslog
```

2. Configure clients for sending encrypted logs to the server:

a. On a client system, create a new file in the **/etc/rsyslog.d/** directory named, for example, **securelogcli.conf**.

b. Add the following lines to the **/etc/rsyslog.d/securelogcli.conf** file:

```
# Set certificate files
global(
  DefaultNetstreamDriverCAFile="/etc/pki/ca-trust/source/anchors/ca-cert.pem"
  DefaultNetstreamDriverCertFile="/etc/pki/ca-trust/source/anchors/client-cert.pem"
  DefaultNetstreamDriverKeyFile="/etc/pki/ca-trust/source/anchors/client-key.pem"
)


# Set up the action for all messages
*.* action(
  type="omfwd"
  StreamDriver="ossl"
  StreamDriverMode="1"
  StreamDriverPermittedPeers="server.example.com"
  StreamDriverAuthMode="x509/name"
  target="server.example.com" port="514" protocol="tcp"
)
```

> **NOTE**
>
> If you prefer the GnuTLS driver, use the **StreamDriver.Name="gtls"** configuration option.

c. Save the changes to the **/etc/rsyslog.d/securelogser.conf** file.

d. Verify the syntax of the **/etc/rsyslog.conf** file and other files in the **/etc/rsyslog.d/** directory:

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-2.el8, config validation run (level 1)...
rsyslogd: End of config validation run. Bye.
```

e. Make sure the **rsyslog** service is running and enabled on the logging server:

```
# systemctl status rsyslog
```

f. Restart the **rsyslog** service:

```
# systemctl restart rsyslog
```

g. Optional: If Rsyslog is not enabled, ensure the **rsyslog** service starts automatically after reboot:

```
# systemctl enable rsyslog
```

Verification

To verify that the client system sends messages to the server, follow these steps:

1. On the client system, send a test message:

```
# logger test
```

2. On the server system, view the **/var/log/messages** log, for example:

```
# cat /var/log/remote/msg/hostname/root.log
Feb 25 03:53:17 hostname root[6064]: test
```

Where ***hostname*** is the host name of the client system. Note that the log contains the user name of the user that entered the logger command, in this case **root**.

Additional resources

- **certtool(1)**, **openssl(1)**, **update-ca-trust(8)**, **rsyslogd(8)**, and **rsyslog.conf(5)** man pages.
- Documentation installed with the **rsyslog-doc** package at **/usr/share/doc/rsyslog/html/index.html**.
- Using the logging system role with TLS .

9.1.6. Configuring a server for receiving remote logging information over UDP

The **Rsyslog** application enables you to configure a system to receive logging information from remote systems. To use remote logging through UDP, configure both the server and the client. The receiving server collects and analyzes the logs sent by one or more client systems. By default, **rsyslog** uses UDP on port **514** to receive log information from remote systems.

Follow this procedure to configure a server for collecting and analyzing logs sent by one or more client systems over the UDP protocol.

Prerequisites

- Rsyslog is installed on the server system.
- You are logged in as **root** on the server.
- The **policycoreutils-python-utils** package is installed for the optional step using the **semanage** command.
- The **firewalld** service is running.

Procedure

1. Optional: To use a different port for **rsyslog** traffic than the default port **514**:
   a. Add the **syslogd_port_t** SELinux type to the SELinux policy configuration, replacing ***portno*** with the port number you want **rsyslog** to use:

      ```
      # semanage port -a -t syslogd_port_t -p udp portno
      ```

   b. Configure **firewalld** to allow incoming **rsyslog** traffic, replacing ***portno*** with the port number and **zone** with the zone you want **rsyslog** to use:

      ```
      # firewall-cmd --zone=zone --permanent --add-port=portno/udp
      success
      # firewall-cmd --reload
      ```

   c. Reload the firewall rules:

      ```
      # firewall-cmd --reload
      ```

2. Create a new **.conf** file in the **/etc/rsyslog.d/** directory, for example, **remotelogserv.conf**, and insert the following content:

   ```
   # Define templates before the rules that use them
   # Per-Host templates for remote systems
   template(name="TmplAuthpriv" type="list") {
       constant(value="/var/log/remote/auth/")
       property(name="hostname")
       constant(value="/")
       property(name="programname" SecurePath="replace")
       constant(value=".log")
       }

   template(name="TmplMsg" type="list") {
       constant(value="/var/log/remote/msg/")
       property(name="hostname")
   ```

```
    constant(value="/")
    property(name="programname" SecurePath="replace")
    constant(value=".log")
    }

# Provides UDP syslog reception
module(load="imudp")

# This ruleset processes remote messages
ruleset(name="remote1"){
     authpriv.*   action(type="omfile" DynaFile="TmplAuthpriv")
      *.info;mail.none;authpriv.none;cron.none
action(type="omfile" DynaFile="TmplMsg")
}

input(type="imudp" port="514" ruleset="remote1")
```

Where **514** is the port number **rsyslog** uses by default. You can specify a different port instead.

3. Verify the syntax of the **/etc/rsyslog.conf** file and all **.conf** files in the **/etc/rsyslog.d/** directory:

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-2.el8, config validation run...
```

4. Restart the **rsyslog** service.

```
# systemctl restart rsyslog
```

5. Optional: If **rsyslog** is not enabled, ensure the **rsyslog** service starts automatically after reboot:

```
# systemctl enable rsyslog
```

Additional resources

- **rsyslogd(8)** , **rsyslog.conf(5)**, **semanage(8)**, and **firewall-cmd(1)** man pages.
- Documentation installed with the **rsyslog-doc** package in the **/usr/share/doc/rsyslog/html/index.html** file.

9.1.7. Configuring remote logging to a server over UDP

Follow this procedure to configure a system for forwarding log messages to a server over the UDP protocol. The **omfwd** plug-in provides forwarding over UDP or TCP. The default protocol is UDP. Because the plug-in is built in, you do not have to load it.

Prerequisites

- The **rsyslog** package is installed on the client systems that should report to the server.
- You have configured the server for remote logging as described in Configuring a server for receiving remote logging information over UDP.

Procedure

1. Create a new **.conf** file in the **/etc/rsyslog.d/** directory, for example, **10-remotelogcli.conf**, and insert the following content:

```
*.* action(type="omfwd"
      queue.type="linkedlist"
      queue.filename="example_fwd"
      action.resumeRetryCount="-1"
      queue.saveOnShutdown="on"
      target="example.com" port="portno" protocol="udp"
     )
```

   Where:

   - The **queue.type="linkedlist"** setting enables a LinkedList in-memory queue.
   - The **queue.filename** setting defines a disk storage. The backup files are created with the ***example_fwd*** prefix in the working directory specified by the preceding global **workDirectory** directive.
   - The **action.resumeRetryCount -1** setting prevents **rsyslog** from dropping messages when retrying to connect if the server is not responding.
   - The **enabled queue.saveOnShutdown="on"** setting saves in-memory data if **rsyslog** shuts down.
   - The ***portno*** value is the port number you want **rsyslog** to use. The default value is **514**.
   - The last line forwards all received messages to the logging server, port specification is optional.
     With this configuration, **rsyslog** sends messages to the server but keeps messages in memory if the remote server is not reachable. A file on disk is created only if **rsyslog** runs out of the configured memory queue space or needs to shut down, which benefits the system performance.

   NOTE

   Rsyslog processes configuration files **/etc/rsyslog.d/** in the lexical order.

2. Restart the **rsyslog** service.

```
# systemctl restart rsyslog
```

3. Optional: If **rsyslog** is not enabled, ensure the **rsyslog** service starts automatically after reboot:

```
# systemctl enable rsyslog
```

Verification

To verify that the client system sends messages to the server, follow these steps:

1. On the client system, send a test message:

```
# logger test
```

2. On the server system, view the **/var/log/remote/msg/*hostname*/root.log** log, for example:

```
# cat /var/log/remote/msg/hostname/root.log
Feb 25 03:53:17 hostname root[6064]: test
```

Where **hostname** is the host name of the client system. Note that the log contains the user name of the user that entered the logger command, in this case **root**.

Additional resources

- **rsyslogd(8)** and **rsyslog.conf(5)** man pages.
- Documentation installed with the **rsyslog-doc** package at **/usr/share/doc/rsyslog/html/index.html**.

9.1.8. Load balancing helper in Rsyslog

The **RebindInterval** setting specifies an interval at which the current connection is broken and is re-established. This setting applies to TCP, UDP, and RELP traffic. The load balancers perceive it as a new connection and forward the messages to another physical target system.

The **RebindInterval** setting proves to be helpful in scenarios when a target system has changed its IP address. The Rsyslog application caches the IP address when the connection establishes, therefore, the messages are sent to the same server. If the IP address changes, the UDP packets will be lost until the Rsyslog service restarts. Re-establishing the connection will ensure the IP to be resolved by DNS again.

```
action(type=”omfwd” protocol=”tcp” RebindInterval=”250” target=”example.com” port=”514” …)

action(type=”omfwd” protocol=”udp” RebindInterval=”250” target=”example.com” port=”514” …)

action(type=”omrelp” RebindInterval=”250” target=”example.com” port=”6514” …)
```

9.1.9. Configuring reliable remote logging

With the Reliable Event Logging Protocol (RELP), you can send and receive **syslog** messages over TCP with a much reduced risk of message loss. RELP provides reliable delivery of event messages, which makes it useful in environments where message loss is not acceptable. To use RELP, configure the **imrelp** input module, which runs on the server and receives the logs, and the **omrelp** output module, which runs on the client and sends logs to the logging server.

Prerequisites

- You have installed the **rsyslog**, **librelp**, and **rsyslog-relp** packages on the server and the client systems.
- The specified port is permitted in SELinux and open in the firewall.

Procedure

1. Configure the client system for reliable remote logging:

   a. On the client system, create a new **.conf** file in the **/etc/rsyslog.d/** directory named, for example, **relpclient.conf**, and insert the following content:

   ```
   module(load="omrelp")
   *.* action(type="omrelp" target="_target_IP_" port="_target_port_")
   ```

Where:

- ***target_IP*** is the IP address of the logging server.
- ***target_port*** is the port of the logging server.

b. Save the changes to the **/etc/rsyslog.d/relpclient.conf** file.

c. Restart the **rsyslog** service.

```
# systemctl restart rsyslog
```

d. Optional: If **rsyslog** is not enabled, ensure the **rsyslog** service starts automatically after reboot:

```
# systemctl enable rsyslog
```

2. Configure the server system for reliable remote logging:

a. On the server system, create a new **.conf** file in the **/etc/rsyslog.d/** directory named, for example, **relpserv.conf**, and insert the following content:

```
ruleset(name="relp"){
*.* action(type="omfile" file="_log_path_")
}


module(load="imrelp")
input(type="imrelp" port="_target_port_" ruleset="relp")
```

Where:

- ***log_path*** specifies the path for storing messages.
- ***target_port*** is the port of the logging server. Use the same value as in the client configuration file.

b. Save the changes to the **/etc/rsyslog.d/relpserv.conf** file.

c. Restart the **rsyslog** service.

```
# systemctl restart rsyslog
```

d. Optional: If **rsyslog** is not enabled, ensure the **rsyslog** service starts automatically after reboot:

```
# systemctl enable rsyslog
```

Verification

To verify that the client system sends messages to the server, follow these steps:

1. On the client system, send a test message:

```
# logger test
```

2. On the server system, view the log at the specified ***log_path***, for example:

```
# cat /var/log/remote/msg/hostname/root.log
Feb 25 03:53:17 hostname root[6064]: test
```

Where **hostname** is the host name of the client system. Note that the log contains the user name of the user that entered the logger command, in this case **root**.

Additional resources

- **rsyslogd(8)** and **rsyslog.conf(5)** man pages.
- Documentation installed with the **rsyslog-doc** package in the **/usr/share/doc/rsyslog/html/index.html** file.

9.1.10. Supported Rsyslog modules

To expand the functionality of the Rsyslog application, you can use specific modules. Modules provide additional inputs (Input Modules), outputs (Output Modules), and other functionalities. A module can also provide additional configuration directives that become available after you load the module.

You can list the input and output modules installed on your system by entering the following command:

```
# ls /usr/lib64/rsyslog/{i,o}m*
```

You can view the list of all available **rsyslog** modules in the **/usr/share/doc/rsyslog/html/configuration/modules/idx_output.html** file after you install the **rsyslog-doc** package.

9.1.11. Configuring the netconsole service to log kernel messages to a remote host

When logging to disk or using a serial console is not possible, you can use the **netconsole** kernel module and the same-named service to log kernel messages over a network to a remote **rsyslog** service.

Prerequisites

- A system log service, such as **rsyslog** is installed on the remote host.
- The remote system log service is configured to receive incoming log entries from this host.

Procedure

1. Install the **netconsole-service** package:

```
# yum install netconsole-service
```

2. Edit the **/etc/sysconfig/netconsole** file and set the **SYSLOGADDR** parameter to the IP address of the remote host:

```
# SYSLOGADDR=192.0.2.1
```

3. Enable and start the **netconsole** service:

```
# systemctl enable --now netconsole
```

Verification steps

- Display the **/var/log/messages** file on the remote system log server.

Additional resources

Configuring a remote logging solution

9.1.12. Additional resources

- Documentation installed with the **rsyslog-doc** package in the **/usr/share/doc/rsyslog/html/index.html** file
- **rsyslog.conf(5)** and **rsyslogd(8)** man pages
- Configuring system logging without journald or with minimized journald usage Knowledgebase article
- Negative effects of the RHEL default logging setup on performance and their mitigations Knowledgebase article
- The Using the Logging system role chapter

9.2. USING THE LOGGING SYSTEM ROLE

As a system administrator, you can use the **logging** system role to configure a RHEL host as a logging server to collect logs from many client systems.

9.2.1. The logging system role

With the **logging** RHEL system role, you can deploy logging configurations on local and remote hosts.

Logging solutions provide multiple ways of reading logs and multiple logging outputs.

For example, a logging system can receive the following inputs:

- Local files
- **systemd/journal**
- Another logging system over the network

In addition, a logging system can have the following outputs:

- Logs stored in the local files in the **/var/log** directory
- Logs sent to Elasticsearch
- Logs forwarded to another logging system

With the **logging** RHEL system role, you can combine the inputs and outputs to fit your scenario. For example, you can configure a logging solution that stores inputs from **journal** in a local file, whereas inputs read from files are both forwarded to another logging system and stored in the local log files.

Additional resources

- **/usr/share/ansible/roles/rhel-system-roles.logging/README.md** file
- **/usr/share/doc/rhel-system-roles/logging/** directory
- RHEL system roles

9.2.2. Variables of the logging system role

In a **logging** RHEL system role playbook, you define the inputs in the **logging_inputs** parameter, outputs in the **logging_outputs** parameter, and the relationships between the inputs and outputs in the **logging_flows** parameter. The **logging** RHEL system role processes these variables with additional options to configure the logging system. You can also enable encryption or an automatic port management.

NOTE

Currently, the only available logging system in the **logging** RHEL system role is **Rsyslog**.

- **logging_inputs**: List of inputs for the logging solution.
  - **name**: Unique name of the input. Used in the **logging_flows**: inputs list and a part of the generated **config** file name.
  - **type**: Type of the input element. The type specifies a task type which corresponds to a directory name in **roles/rsyslog/{tasks,vars}/inputs/**.
    - **basics**: Inputs configuring inputs from **systemd** journal or **unix** socket.
      - **kernel_message**: Load **imklog** if set to **true**. Default to **false**.
      - **use_imuxsock**: Use **imuxsock** instead of **imjournal**. Default to **false**.
      - **ratelimit_burst**: Maximum number of messages that can be emitted within **ratelimit_interval**. Default to **20000** if **use_imuxsock** is false. Default to **200** if **use_imuxsock** is true.
      - **ratelimit_interval**: Interval to evaluate **ratelimit_burst**. Default to 600 seconds if **use_imuxsock** is false. Default to 0 if **use_imuxsock** is true. 0 indicates rate limiting is turned off.
      - **persist_state_interval**: Journal state is persisted every **value** messages. Default to **10**. Effective only when **use_imuxsock** is false.
    - **files**: Inputs configuring inputs from local files.
    - **remote**: Inputs configuring inputs from the other logging system over network.
  - **state**: State of the configuration file. **present** or **absent**. Default to **present**.
- **logging_outputs**: List of outputs for the logging solution.

  - **files**: Outputs configuring outputs to local files.
  - **forwards**: Outputs configuring outputs to another logging system.
  - **remote_files**: Outputs configuring outputs from another logging system to local files.

- **logging_flows**: List of flows that define relationships between **logging_inputs** and **logging_outputs**. The **logging_flows** variable has the following keys:
  - **name**: Unique name of the flow
  - **inputs**: List of **logging_inputs** name values
  - **outputs**: List of **logging_outputs** name values.
- **logging_manage_firewall**: If set to **true**, the **logging** RHEL system role uses the **firewall** RHEL system role to automatically manage port access.
- **logging_manage_selinux**: If set to **true**, the **logging** RHEL system role uses the **selinux** RHEL system role to automatically manage port access.

Additional resources

- **/usr/share/ansible/roles/rhel-system-roles.logging/README.md** file
- **/usr/share/doc/rhel-system-roles/logging/** directory

9.2.3. Applying a local logging system role

Prepare and apply an Ansible playbook to configure a logging solution on a set of separate machines. Each machine records logs locally.

Prerequisites

- You have prepared the control node and the managed nodes
- You are logged in to the control node as a user who can run playbooks on the managed nodes.
- The account you use to connect to the managed nodes has **sudo** permissions on them.

> **NOTE**
>
> You do not have to have the **rsyslog** package installed, because the RHEL system role installs **rsyslog** when deployed.

Procedure

1. Create a playbook file, for example **~/playbook.yml**, with the following content:

```
---
- name: Deploying basics input and implicit files output
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.logging
  vars:
```

```
logging_inputs:
  - name: system_input
    type: basics
logging_outputs:
  - name: files_output
    type: files
logging_flows:
  - name: flow1
    inputs: [system_input]
    outputs: [files_output]
```

2. Validate the playbook syntax:

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

Note that this command only validates the syntax and does not protect against a wrong but valid configuration.

3. Run the playbook:

```
$ ansible-playbook ~/playbook.yml
```

Verification

1. Test the syntax of the **/etc/rsyslog.conf** file:

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-6.el8, config validation run...
rsyslogd: End of config validation run. Bye.
```

2. Verify that the system sends messages to the log:

   a. Send a test message:

   ```
   # logger test
   ```

   b. View the **/var/log/messages** log, for example:

   ```
   # cat /var/log/messages
   Aug  5 13:48:31 <hostname> root[6778]: test
   ```

   Where ***<hostname>*** is the host name of the client system. Note that the log contains the user name of the user that entered the logger command, in this case **root**.

Additional resources

- **/usr/share/ansible/roles/rhel-system-roles.logging/README.md** file
- **/usr/share/doc/rhel-system-roles/logging/** directory

9.2.4. Filtering logs in a local logging system role

You can deploy a logging solution which filters the logs based on the **rsyslog** property-based filter.

Prerequisites

- You have prepared the control node and the managed nodes
- You are logged in to the control node as a user who can run playbooks on the managed nodes.
- The account you use to connect to the managed nodes has **sudo** permissions on them.

NOTE

You do not have to have the **rsyslog** package installed, because the RHEL system role installs **rsyslog** when deployed.

Procedure

1. Create a playbook file, for example ~/**playbook.yml**, with the following content:

```
---
- name: Deploying files input and configured files output
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.logging
  vars:
    logging_inputs:
      - name: files_input
        type: basics
    logging_outputs:
      - name: files_output0
        type: files
        property: msg
        property_op: contains
        property_value: error
        path: /var/log/errors.log
      - name: files_output1
        type: files
        property: msg
        property_op: "!contains"
        property_value: error
        path: /var/log/others.log
    logging_flows:
      - name: flow0
        inputs: [files_input]
        outputs: [files_output0, files_output1]
```

   Using this configuration, all messages that contain the **error** string are logged in **/var/log/errors.log**, and all other messages are logged in **/var/log/others.log**.

   You can replace the **error** property value with the string by which you want to filter.

   You can modify the variables according to your preferences.

2. Validate the playbook syntax:

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

Note that this command only validates the syntax and does not protect against a wrong but valid configuration.

3. Run the playbook:

```
$ ansible-playbook ~/playbook.yml
```

Verification

1. Test the syntax of the **/etc/rsyslog.conf** file:

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-6.el8, config validation run...
rsyslogd: End of config validation run. Bye.
```

2. Verify that the system sends messages that contain the **error** string to the log:

   a. Send a test message:

   ```
   # logger error
   ```

   b. View the **/var/log/errors.log** log, for example:

   ```
   # cat /var/log/errors.log
   Aug  5 13:48:31 hostname root[6778]: error
   ```

   Where ***hostname*** is the host name of the client system. Note that the log contains the user name of the user that entered the logger command, in this case **root**.

Additional resources

- **/usr/share/ansible/roles/rhel-system-roles.logging/README.md** file
- **/usr/share/doc/rhel-system-roles/logging/** directory

9.2.5. Applying a remote logging solution by using the logging system role

Follow these steps to prepare and apply a Red Hat Ansible Core playbook to configure a remote logging solution. In this playbook, one or more clients take logs from **systemd-journal** and forward them to a remote server. The server receives remote input from **remote_rsyslog** and **remote_files** and outputs the logs to local files in directories named by remote host names.

Prerequisites

- You have prepared the control node and the managed nodes
- You are logged in to the control node as a user who can run playbooks on the managed nodes.
- The account you use to connect to the managed nodes has **sudo** permissions on them.

NOTE

You do not have to have the **rsyslog** package installed, because the RHEL system role installs **rsyslog** when deployed.

Procedure

1. Create a playbook file, for example ~/**playbook.yml**, with the following content:

```
---
- name: Deploying remote input and remote_files output
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.logging
  vars:
    logging_inputs:
      - name: remote_udp_input
        type: remote
        udp_ports: [ 601 ]
      - name: remote_tcp_input
        type: remote
        tcp_ports: [ 601 ]
    logging_outputs:
      - name: remote_files_output
        type: remote_files
    logging_flows:
      - name: flow_0
        inputs: [remote_udp_input, remote_tcp_input]
        outputs: [remote_files_output]

- name: Deploying basics input and forwards output
  hosts: managed-node-02.example.com
  roles:
    - rhel-system-roles.logging
  vars:
    logging_inputs:
      - name: basic_input
        type: basics
    logging_outputs:
      - name: forward_output0
        type: forwards
        severity: info
        target: <host1.example.com>
        udp_port: 601
      - name: forward_output1
        type: forwards
        facility: mail
        target: <host1.example.com>
        tcp_port: 601
    logging_flows:
      - name: flows0
        inputs: [basic_input]
        outputs: [forward_output0, forward_output1]

[basic_input]
[forward_output0, forward_output1]
```

Where ***<host1.example.com>*** is the logging server.

NOTE

You can modify the parameters in the playbook to fit your needs.

WARNING

The logging solution works only with the ports defined in the SELinux policy of the server or client system and open in the firewall. The default SELinux policy includes ports 601, 514, 6514, 10514, and 20514. To use a different port, modify the SELinux policy on the client and server systems .

2. Validate the playbook syntax:

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

Note that this command only validates the syntax and does not protect against a wrong but valid configuration.

3. Run the playbook:

```
$ ansible-playbook ~/playbook.yml
```

Verification

1. On both the client and the server system, test the syntax of the **/etc/rsyslog.conf** file:

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-6.el8, config validation run (level 1), master config
/etc/rsyslog.conf
rsyslogd: End of config validation run. Bye.
```

2. Verify that the client system sends messages to the server:

   a. On the client system, send a test message:

   ```
   # logger test
   ```

   b. On the server system, view the **/var/log/*<host2.example.com>*/messages** log, for example:

   ```
   # cat /var/log/<host2.example.com>/messages
   Aug  5 13:48:31 <host2.example.com> root[6778]: test
   ```

   Where ***<host2.example.com>*** is the host name of the client system. Note that the log contains the user name of the user that entered the logger command, in this case **root**.

Additional resources

- **/usr/share/ansible/roles/rhel-system-roles.logging/README.md** file

- **/usr/share/doc/rhel-system-roles/logging/** directory

9.2.6. Using the logging system role with TLS

Transport Layer Security (TLS) is a cryptographic protocol designed to allow secure communication over the computer network.

As an administrator, you can use the **logging** RHEL system role to configure a secure transfer of logs using Red Hat Ansible Automation Platform.

9.2.6.1. Configuring client logging with TLS

You can use an Ansible playbook with the **logging** RHEL system role to configure logging on RHEL clients and transfer logs to a remote logging system using TLS encryption.

This procedure creates a private key and certificate, and configures TLS on all hosts in the clients group in the Ansible inventory. The TLS protocol encrypts the message transmission for secure transfer of logs over the network.

> **NOTE**
>
> You do not have to call the **certificate** RHEL system role in the playbook to create the certificate. The **logging** RHEL system role calls it automatically.
>
> In order for the CA to be able to sign the created certificate, the managed nodes must be enrolled in an IdM domain.

Prerequisites

- You have prepared the control node and the managed nodes
- You are logged in to the control node as a user who can run playbooks on the managed nodes.
- The account you use to connect to the managed nodes has **sudo** permissions on them.
- The managed nodes are enrolled in an IdM domain.

Procedure

1. Create a playbook file, for example **~/playbook.yml**, with the following content:

```
---
- name: Deploying files input and forwards output with certs
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.logging
  vars:
    logging_certificates:
      - name: logging_cert
        dns: ['localhost', 'www.example.com']
        ca: ipa
    logging_pki_files:
      - ca_cert: /local/path/to/ca_cert.pem
        cert: /local/path/to/logging_cert.pem
        private_key: /local/path/to/logging_cert.pem
```

```
logging_inputs:
  - name: input_name
    type: files
    input_log_path: /var/log/containers/*.log
logging_outputs:
  - name: output_name
    type: forwards
    target: your_target_host
    tcp_port: 514
    tls: true
    pki_authmode: x509/name
    permitted_server: 'server.example.com'
logging_flows:
  - name: flow_name
    inputs: [input_name]
    outputs: [output_name]
```

The playbook uses the following parameters:

logging_certificates

The value of this parameter is passed on to **certificate_requests** in the **certificate** RHEL system role and used to create a private key and certificate.

logging_pki_files

Using this parameter, you can configure the paths and other settings that logging uses to find the CA, certificate, and key files used for TLS, specified with one or more of the following sub-parameters: **ca_cert**, **ca_cert_src**, **cert**, **cert_src**, **private_key**, **private_key_src**, and **tls**.

> NOTE
>
> If you are using **logging_certificates** to create the files on the target node, do not use **ca_cert_src**, **cert_src**, and **private_key_src**, which are used to copy files not created by **logging_certificates**.

ca_cert

Represents the path to the CA certificate file on the target node. Default path is **/etc/pki/tls/certs/ca.pem** and the file name is set by the user.

cert

Represents the path to the certificate file on the target node. Default path is **/etc/pki/tls/certs/server-cert.pem** and the file name is set by the user.

private_key

Represents the path to the private key file on the target node. Default path is **/etc/pki/tls/private/server-key.pem** and the file name is set by the user.

ca_cert_src

Represents the path to the CA certificate file on the control node which is copied to the target host to the location specified by **ca_cert**. Do not use this if using **logging_certificates**.

cert_src

Represents the path to a certificate file on the control node which is copied to the target host to the location specified by **cert**. Do not use this if using **logging_certificates**.

private_key_src

Represents the path to a private key file on the control node which is copied to the target host to the location specified by **private_key**. Do not use this if using **logging_certificates**.

tls

Setting this parameter to **true** ensures secure transfer of logs over the network. If you do not want a secure wrapper, you can set **tls: false**.

2. Validate the playbook syntax:

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

Note that this command only validates the syntax and does not protect against a wrong but valid configuration.

3. Run the playbook:

```
$ ansible-playbook ~/playbook.yml
```

Additional resources

- **/usr/share/ansible/roles/rhel-system-roles.logging/README.md** file
- **/usr/share/doc/rhel-system-roles/logging/** directory
- Requesting certificates using RHEL system roles .

9.2.6.2. Configuring server logging with TLS

You can use an Ansible playbook with the **logging** RHEL system role to configure logging on RHEL servers and set them to receive logs from a remote logging system using TLS encryption.

This procedure creates a private key and certificate, and configures TLS on all hosts in the server group in the Ansible inventory.

> **NOTE**
>
> You do not have to call the **certificate** RHEL system role in the playbook to create the certificate. The **logging** RHEL system role calls it automatically.
>
> In order for the CA to be able to sign the created certificate, the managed nodes must be enrolled in an IdM domain.

Prerequisites

- You have prepared the control node and the managed nodes
- You are logged in to the control node as a user who can run playbooks on the managed nodes.
- The account you use to connect to the managed nodes has **sudo** permissions on them.
- The managed nodes are enrolled in an IdM domain.

Procedure

1. Create a playbook file, for example **~/playbook.yml**, with the following content:

```
---
- name: Deploying remote input and remote_files output with certs
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.logging
  vars:
    logging_certificates:
      - name: logging_cert
        dns: ['localhost', 'www.example.com']
        ca: ipa
    logging_pki_files:
      - ca_cert: /local/path/to/ca_cert.pem
        cert: /local/path/to/logging_cert.pem
        private_key: /local/path/to/logging_cert.pem
    logging_inputs:
      - name: input_name
        type: remote
        tcp_ports: 514
        tls: true
        permitted_clients: ['clients.example.com']
    logging_outputs:
      - name: output_name
        type: remote_files
        remote_log_path: /var/log/remote/%FROMHOST%/%PROGRAMNAME:::secpath-replace%.log
        async_writing: true
        client_count: 20
        io_buffer_size: 8192
    logging_flows:
      - name: flow_name
        inputs: [input_name]
        outputs: [output_name]
```

The playbook uses the following parameters:

logging_certificates

The value of this parameter is passed on to **certificate_requests** in the **certificate** RHEL system role and used to create a private key and certificate.

logging_pki_files

Using this parameter, you can configure the paths and other settings that logging uses to find the CA, certificate, and key files used for TLS, specified with one or more of the following sub-parameters: **ca_cert**, **ca_cert_src**, **cert**, **cert_src**, **private_key**, **private_key_src**, and **tls**.

> **NOTE**
>
> If you are using **logging_certificates** to create the files on the target node, do not use **ca_cert_src**, **cert_src**, and **private_key_src**, which are used to copy files not created by **logging_certificates**.

ca_cert

Represents the path to the CA certificate file on the target node. Default path is **/etc/pki/tls/certs/ca.pem** and the file name is set by the user.

cert

Represents the path to the certificate file on the target node. Default path is **/etc/pki/tls/certs/server-cert.pem** and the file name is set by the user.

private_key

Represents the path to the private key file on the target node. Default path is **/etc/pki/tls/private/server-key.pem** and the file name is set by the user.

ca_cert_src

Represents the path to the CA certificate file on the control node which is copied to the target host to the location specified by **ca_cert**. Do not use this if using **logging_certificates**.

cert_src

Represents the path to a certificate file on the control node which is copied to the target host to the location specified by **cert**. Do not use this if using **logging_certificates**.

private_key_src

Represents the path to a private key file on the control node which is copied to the target host to the location specified by **private_key**. Do not use this if using **logging_certificates**.

tls

Setting this parameter to **true** ensures secure transfer of logs over the network. If you do not want a secure wrapper, you can set **tls: false**.

2. Validate the playbook syntax:

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

Note that this command only validates the syntax and does not protect against a wrong but valid configuration.

3. Run the playbook:

```
$ ansible-playbook ~/playbook.yml
```

Additional resources

- **/usr/share/ansible/roles/rhel-system-roles.logging/README.md** file
- **/usr/share/doc/rhel-system-roles/logging/** directory
- Requesting certificates using RHEL system roles .

9.2.7. Using the logging system roles with RELP

Reliable Event Logging Protocol (RELP) is a networking protocol for data and message logging over the TCP network. It ensures reliable delivery of event messages and you can use it in environments that do not tolerate any message loss.

The RELP sender transfers log entries in form of commands and the receiver acknowledges them once they are processed. To ensure consistency, RELP stores the transaction number to each transferred command for any kind of message recovery.

You can consider a remote logging system in between the RELP Client and RELP Server. The RELP Client transfers the logs to the remote logging system and the RELP Server receives all the logs sent by the remote logging system.

Administrators can use the **logging** RHEL system role to configure the logging system to reliably send and receive log entries.

9.2.7.1. Configuring client logging with RELP

You can use the **logging** RHEL system role to configure logging in RHEL systems that are logged on a local machine and can transfer logs to the remote logging system with RELP by running an Ansible playbook.

This procedure configures RELP on all hosts in the **clients** group in the Ansible inventory. The RELP configuration uses Transport Layer Security (TLS) to encrypt the message transmission for secure transfer of logs over the network.

Prerequisites

- You have prepared the control node and the managed nodes
- You are logged in to the control node as a user who can run playbooks on the managed nodes.
- The account you use to connect to the managed nodes has **sudo** permissions on them.

Procedure

1. Create a playbook file, for example ~/**playbook.yml**, with the following content:

```
---
- name: Deploying basic input and relp output
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.logging
  vars:
    logging_inputs:
      - name: basic_input
        type: basics
    logging_outputs:
      - name: relp_client
        type: relp
        target: logging.server.com
        port: 20514
        tls: true
        ca_cert: /etc/pki/tls/certs/ca.pem
        cert: /etc/pki/tls/certs/client-cert.pem
        private_key: /etc/pki/tls/private/client-key.pem
        pki_authmode: name
        permitted_servers:
          - '*.server.example.com'
    logging_flows:
      - name: example_flow
        inputs: [basic_input]
        outputs: [relp_client]
```

The playbook uses following settings:

target

This is a required parameter that specifies the host name where the remote logging system is running.

port

Port number the remote logging system is listening.

tls

Ensures secure transfer of logs over the network. If you do not want a secure wrapper you can set the **tls** variable to **false**. By default **tls** parameter is set to true while working with RELP and requires key/certificates and triplets {**ca_cert**, **cert**, **private_key**} and/or {**ca_cert_src**, **cert_src**, **private_key_src**}.

- If the {**ca_cert_src**, **cert_src**, **private_key_src**} triplet is set, the default locations **/etc/pki/tls/certs** and **/etc/pki/tls/private** are used as the destination on the managed node to transfer files from control node. In this case, the file names are identical to the original ones in the triplet
- If the {**ca_cert**, **cert**, **private_key**} triplet is set, files are expected to be on the default path before the logging configuration.
- If both triplets are set, files are transferred from local path from control node to specific path of the managed node.

ca_cert

Represents the path to CA certificate. Default path is **/etc/pki/tls/certs/ca.pem** and the file name is set by the user.

cert

Represents the path to certificate. Default path is **/etc/pki/tls/certs/server-cert.pem** and the file name is set by the user.

private_key

Represents the path to private key. Default path is **/etc/pki/tls/private/server-key.pem** and the file name is set by the user.

ca_cert_src

Represents local CA certificate file path which is copied to the target host. If **ca_cert** is specified, it is copied to the location.

cert_src

Represents the local certificate file path which is copied to the target host. If **cert** is specified, it is copied to the location.

private_key_src

Represents the local key file path which is copied to the target host. If **private_key** is specified, it is copied to the location.

pki_authmode

Accepts the authentication mode as **name** or **fingerprint**.

permitted_servers

List of servers that will be allowed by the logging client to connect and send logs over TLS.

inputs

List of logging input dictionary.

outputs
List of logging output dictionary.

2. Validate the playbook syntax:

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

Note that this command only validates the syntax and does not protect against a wrong but valid configuration.

3. Run the playbook:

```
$ ansible-playbook ~/playbook.yml
```

Additional resources

- **/usr/share/ansible/roles/rhel-system-roles.logging/README.md** file
- **/usr/share/doc/rhel-system-roles/logging/** directory

9.2.7.2. Configuring server logging with RELP

You can use the **logging** RHEL system role to configure logging in RHEL systems as a server and can receive logs from the remote logging system with RELP by running an Ansible playbook.

This procedure configures RELP on all hosts in the **server** group in the Ansible inventory. The RELP configuration uses TLS to encrypt the message transmission for secure transfer of logs over the network.

Prerequisites

- You have prepared the control node and the managed nodes
- You are logged in to the control node as a user who can run playbooks on the managed nodes.
- The account you use to connect to the managed nodes has **sudo** permissions on them.

Procedure

1. Create a playbook file, for example **~/playbook.yml**, with the following content:

```
---
- name: Deploying remote input and remote_files output
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.logging
  vars:
    logging_inputs:
      - name: relp_server
        type: relp
        port: 20514
        tls: true
        ca_cert: /etc/pki/tls/certs/ca.pem
        cert: /etc/pki/tls/certs/server-cert.pem
```

```
        private_key: /etc/pki/tls/private/server-key.pem
        pki_authmode: name
        permitted_clients:
          - '*example.client.com'
    logging_outputs:
      - name: remote_files_output
        type: remote_files
    logging_flows:
      - name: example_flow
        inputs: relp_server
        outputs: remote_files_output
```

The playbooks uses the following settings:

port

Port number the remote logging system is listening.

tls

Ensures secure transfer of logs over the network. If you do not want a secure wrapper you can set the **tls** variable to **false**. By default **tls** parameter is set to true while working with RELP and requires key/certificates and triplets {**ca_cert**, **cert**, **private_key**} and/or {**ca_cert_src**, **cert_src**, **private_key_src**}.

- If the {**ca_cert_src**, **cert_src**, **private_key_src**} triplet is set, the default locations **/etc/pki/tls/certs** and **/etc/pki/tls/private** are used as the destination on the managed node to transfer files from control node. In this case, the file names are identical to the original ones in the triplet

- If the {**ca_cert**, **cert**, **private_key**} triplet is set, files are expected to be on the default path before the logging configuration.

- If both triplets are set, files are transferred from local path from control node to specific path of the managed node.

ca_cert

Represents the path to CA certificate. Default path is **/etc/pki/tls/certs/ca.pem** and the file name is set by the user.

cert

Represents the path to the certificate. Default path is **/etc/pki/tls/certs/server-cert.pem** and the file name is set by the user.

private_key

Represents the path to private key. Default path is **/etc/pki/tls/private/server-key.pem** and the file name is set by the user.

ca_cert_src

Represents local CA certificate file path which is copied to the target host. If **ca_cert** is specified, it is copied to the location.

cert_src

Represents the local certificate file path which is copied to the target host. If **cert** is specified, it is copied to the location.

private_key_src

Represents the local key file path which is copied to the target host. If **private_key** is specified, it is copied to the location.

pki_authmode
Accepts the authentication mode as **name** or **fingerprint**.

permitted_clients
List of clients that will be allowed by the logging server to connect and send logs over TLS.

inputs
List of logging input dictionary.

outputs
List of logging output dictionary.

2. Validate the playbook syntax:

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

Note that this command only validates the syntax and does not protect against a wrong but valid configuration.

3. Run the playbook:

```
$ ansible-playbook ~/playbook.yml
```

Additional resources

- **/usr/share/ansible/roles/rhel-system-roles.logging/README.md** file
- **/usr/share/doc/rhel-system-roles/logging/** directory

9.2.8. Additional resources

- Preparing a control node and managed nodes to use RHEL system roles
- Documentation installed with the **rhel-system-roles** package in **/usr/share/ansible/roles/rhel-system-roles.logging/README.html**.
- RHEL system roles
- **ansible-playbook(1)** man page.

CHAPTER 10. TROUBLESHOOTING PROBLEMS BY USING LOG FILES

Log files contain messages about the system, including the kernel, services, and applications running on it. These contain information that helps troubleshoot issues or monitor system functions. The logging system in Red Hat Enterprise Linux is based on the built-in **syslog** protocol. Particular programs use this system to record events and organize them into log files, which are useful when auditing the operating system and troubleshooting various problems.

10.1. SERVICES HANDLING SYSLOG MESSAGES

The following two services handle **syslog** messages:

- The **systemd-journald** daemon
- The **Rsyslog** service

The **systemd-journald** daemon collects messages from various sources and forwards them to **Rsyslog** for further processing. The **systemd-journald** daemon collects messages from the following sources:

- Kernel
- Early stages of the boot process
- Standard and error output of daemons as they start up and run
- **Syslog**

The **Rsyslog** service sorts the **syslog** messages by type and priority and writes them to the files in the **/var/log** directory. The **/var/log** directory persistently stores the log messages.

10.2. SUBDIRECTORIES STORING SYSLOG MESSAGES

The following subdirectories under the **/var/log** directory store **syslog** messages.

- **/var/log/messages** - all **syslog** messages except the following
- **/var/log/secure** - security and authentication-related messages and errors
- **/var/log/maillog** - mail server-related messages and errors
- **/var/log/cron** - log files related to periodically executed tasks
- **/var/log/boot.log** - log files related to system startup

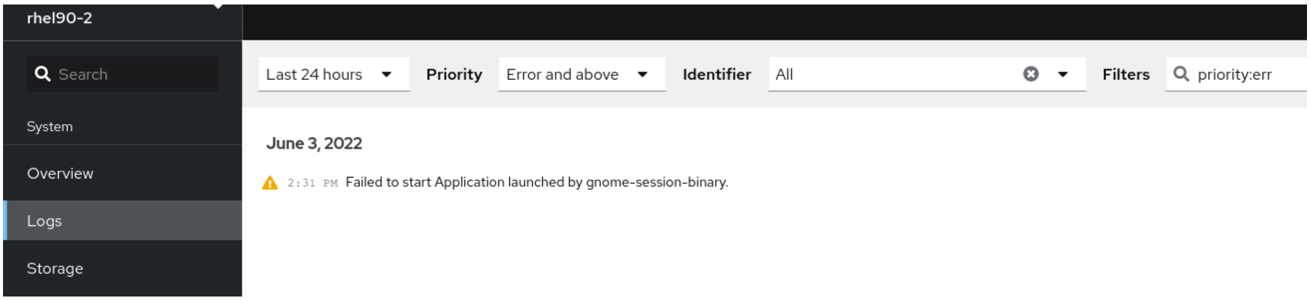
10.3. INSPECTING LOG FILES USING THE WEB CONSOLE

Follow the steps in this procedure to inspect the log files using the RHEL web console.

Procedure

1. Log into the RHEL web console. For details see Logging in to the web console .
2. Click **Logs**.

Figure 10.1. Inspecting the log files in the RHEL 8 web console



10.4. VIEWING LOGS USING THE COMMAND LINE

The Journal is a component of systemd that helps to view and manage log files. It addresses problems connected with traditional logging, closely integrated with the rest of the system, and supports various logging technologies and access management for the log files.

You can use the **journalctl** command to view messages in the system journal using the command line, for example:

```
$ journalctl -b | grep kvm
May 15 11:31:41 localhost.localdomain kernel: kvm-clock: Using msrs 4b564d01 and 4b564d00
May 15 11:31:41 localhost.localdomain kernel: kvm-clock: cpu 0, msr 76401001, primary cpu clock
...
```

Table 10.1. Viewing system information

Command	Description
journalctl	Shows all collected journal entries.
journalctl FILEPATH	Shows logs related to a specific file. For example, the **journalctl /dev/sda** command displays logs related to the **/dev/sda** file system.
journalctl -b	Shows logs for the current boot.
journalctl -k -b -1	Shows kernel logs for the current boot.

Table 10.2. Viewing information about specific services

Command	Description
journalctl -b _SYSTEMD_UNIT=*<name.service>*	Filters log to show entries matching the **systemd** service.
journalctl -b _SYSTEMD_UNIT=*<name.service>* _PID=*<number>*	Combines matches. For example, this command shows logs for **systemd-units** that match ***<name.service>*** and the PID ***<number>***.

Command	Description
journalctl -b _SYSTEMD_UNIT=*<name.service>* _PID=*<number>* + _SYSTEMD_UNIT=*<name2.service>*	The plus sign (+) separator combines two expressions in a logical OR. For example, this command shows all messages from the ***<name.service>*** service process with the **PID** plus all messages from the ***<name2.service>*** service (from any of its processes).
journalctl -b _SYSTEMD_UNIT=*<name.service>* _SYSTEMD_UNIT=*<name2.service>*	This command shows all entries matching either expression, referring to the same field. Here, this command shows logs matching a systemd-unit ***<name.service>*** or a systemd-unit ***<name2.service>***.

Table 10.3. Viewing logs related to specific boots

Command	Description
journalctl --list-boots	Shows a tabular list of boot numbers, their IDs, and the timestamps of the first and last message pertaining to the boot. You can use the ID in the next command to view detailed information.
journalctl --boot=ID _SYSTEMD_UNIT=*<name.service>*	Shows information about the specified boot ID.

10.5. ADDITIONAL RESOURCES

- **man journalctl(1)**
- Configuring a remote logging solution

CHAPTER 11. MANAGING USERS AND GROUPS

Preventing unauthorized access to files and processes requires an accurate user and group management. If you do not manage accounts centrally or you require a user account or group only on a specific system, you can create them locally on this host.

11.1. INTRODUCTION TO MANAGING USER AND GROUP ACCOUNTS

The control of users and groups is a core element of Red Hat Enterprise Linux (RHEL) system administration. Each RHEL user has distinct login credentials and can be assigned to various groups to customize their system privileges.

11.1.1. Introduction to users and groups

A user who creates a file is the owner of that file *and* the group owner of that file. The file is assigned separate read, write, and execute permissions for the owner, the group, and those outside that group. The file owner can be changed only by the **root** user. Access permissions to the file can be changed by both the **root** user and the file owner. A regular user can change group ownership of a file they own to a group of which they are a member of.

Each user is associated with a unique numerical identification number called *user ID* (**UID**). Each group is associated with a *group ID* (**GID**). Users within a group share the same permissions to read, write, and execute files owned by that group.

11.1.2. Configuring reserved user and group IDs

RHEL reserves user and group IDs below 1000 for system users and groups. You can find the reserved user and group IDs in the **setup** package. To view reserved user and group IDs, use:

```
cat /usr/share/doc/setup*/uidgid
```

It is recommended to assign IDs to the new users and groups starting at 5000, as the reserved range can increase in the future.

To make the IDs assigned to new users start at 5000 by default, modify the **UID_MIN** and **GID_MIN** parameters in the **/etc/login.defs** file.

Procedure

To modify and make the IDs assigned to new users start at 5000 by default:

1. Open the **/etc/login.defs** file in an editor of your choice.

2. Find the lines that define the minimum value for automatic UID selection.

```
# Min/max values for automatic uid selection in useradd
#
UID_MIN                  1000
```

3. Modify the **UID_MIN** value to start at 5000.

```
# Min/max values for automatic uid selection in useradd
#
UID_MIN                  5000
```

4. Find the lines that define the minimum value for automatic GID selection.

```
# Min/max values for automatic gid selection in groupadd
#
GID_MIN                  1000
```

5. Modify the **GID_MIN** value to start at 5000.

```
# Min/max values for automatic gid selection in groupadd
#
GID_MIN                  5000
```

The dynamically assigned UIDs and GIDs for the regular users now start at 5000.

> **NOTE**
>
> The UID's and GID's of users and groups created before you changed the UID_MIN and GID_MIN values do not change.

This will allow new user's group to have same 5000+ ID as UID and GID.

> **WARNING**
>
> Do not raise IDs reserved by the system above 1000 by changing **SYS_UID_MAX** to avoid conflict with systems that retain the 1000 limit.

11.1.3. User private groups

RHEL uses the *user private group* (**UPG**) system configuration, which makes UNIX groups easier to manage. A user private group is created whenever a new user is added to the system. The user private group has the same name as the user for which it was created and that user is the only member of the user private group.

UPGs simplify the collaboration on a project between multiple users. In addition, UPG system configuration makes it safe to set default permissions for a newly created file or directory, as it allows both the user, and the group this user is a part of, to make modifications to the file or directory.

A list of all groups is stored in the **/etc/group** configuration file.

11.2. GETTING STARTED WITH MANAGING USER ACCOUNTS

Red Hat Enterprise Linux is a multi-user operating system, which enables multiple users on different computers to access a single system installed on one machine. Every user operates under its own account, and managing user accounts thus represents a core element of Red Hat Enterprise Linux system administration.

The following are the different types of user accounts:

- **Normal user accounts:**

Normal accounts are created for users of a particular system. Such accounts can be added, removed, and modified during normal system administration.

- **System user accounts:**
  System user accounts represent a particular applications identifier on a system. Such accounts are generally added or manipulated only at software installation time, and they are not modified later.

  > **WARNING**
  >
  > System accounts are presumed to be available locally on a system. If these accounts are configured and provided remotely, such as in the instance of an LDAP configuration, system breakage and service start failures can occur.

  For system accounts, user IDs below 1000 are reserved. For normal accounts, you can use IDs starting at 1000. However, the recommended practice is to assign IDs starting at 5000. For assigning IDs, see the **/etc/login.defs** file.

- **Group:**
  A group is an entity which ties together multiple user accounts for a common purpose, such as granting access to particular files.

11.2.1. Managing accounts and groups using command line tools

Use the following basic command-line tools to manage user accounts and groups.

- To display user and group IDs:

  ```
  $ id
  uid=1000(example.user) gid=1000(example.user) groups=1000(example.user),10(wheel)
  context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
  ```

- To create a new user account:

  ```
  # useradd example.user
  ```

- To assign a new password to a user account belonging to *example.user*:

  ```
  # passwd example.user
  ```

- To add a user to a group:

  ```
  # usermod -a -G example.group example.user
  ```

Additional resources

- **man useradd(8)**, **man passwd(1)**, and **man usermod(8)**

11.2.2. System user accounts managed in the web console

With user accounts displayed in the RHEL web console you can:

- Authenticate users when accessing the system.
- Set the access rights to the system.

The RHEL web console displays all user accounts located in the system. Therefore, you can see at least one user account just after the first login to the web console.

After logging into the RHEL web console, you can perform the following operations:

- Create new users accounts.
- Change their parameters.
- Lock accounts.
- Terminate user sessions.

11.2.3. Adding new accounts using the web console

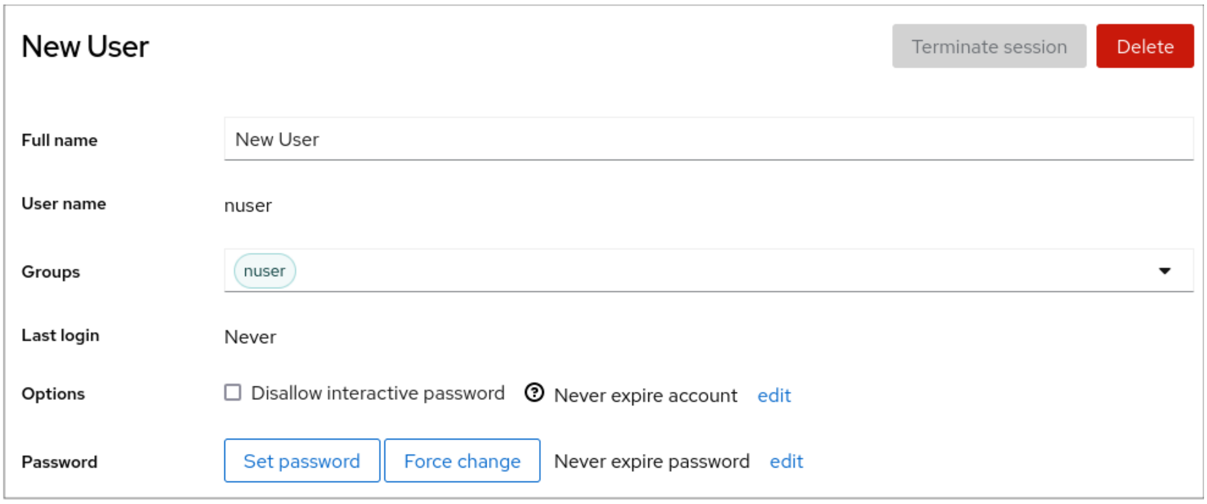
Use the following steps for adding user accounts to the system and setting administration rights to the accounts through the RHEL web console.

Prerequisites

- The RHEL web console must be installed and accessible. For details, see Installing the web console.

Procedure

1. Log in to the RHEL web console.
2. Click **Accounts**.
3. Click **Create New Account**.
4. In the **Full Name** field, enter the full name of the user.
   The RHEL web console automatically suggests a user name from the full name and fills it in the **User Name** field. If you do not want to use the original naming convention consisting of the first letter of the first name and the whole surname, update the suggestion.
5. In the **Password/Confirm** fields, enter the password and retype it for verification that your password is correct.
   The color bar below the fields shows you the security level of the entered password, which does not allow you to create a user with a weak password.
6. Click **Create** to save the settings and close the dialog box.
7. Select the newly created account.
8. In the **Groups** drop down menu, select the groups that you want to add to the new account.



Now you can see the new account in the **Accounts** settings and you can use its credentials to connect to the system.

11.3. MANAGING USERS FROM THE COMMAND LINE

You can manage users and groups using the command-line interface (**CLI**). This enables you to add, remove, and modify users and user groups in Red Hat Enterprise Linux environment.

11.3.1. Adding a new user from the command line

You can use the **useradd** utility to add a new user.

Prerequisites

- **Root** access

Procedure

- To add a new user, use:

  ```
  # useradd options username
  ```

  Replace *options* with the command-line options for the **useradd** command, and replace *username* with the name of the user.

  > **Example 11.1. Adding a new user**
  >
  > To add the user **sarah** with user ID **5000**, use:
  >
  > ```
  > # useradd -u 5000 sarah
  > ```

Verification steps

- To verify the new user is added, use the **id** utility.

  ```
  # id sarah
  ```

The output returns:

```
uid=5000(sarah) gid=5000(sarah) groups=5000(sarah)
```

Additional resources

- **useradd** man page

11.3.2. Adding a new group from the command line

You can use the **groupadd** utility to add a new group.

Prerequisites

- **Root** access

Procedure

- To add a new group, use:

  ```
  # groupadd options group-name
  ```

  Replace *options* with the command-line options for the **groupadd** command, and replace *group-name* with the name of the group.

  Example 11.2. Adding a new group

  To add the group **sysadmins** with group ID **5000**, use:

  ```
  # groupadd -g 5000 sysadmins
  ```

Verification steps

- To verify the new group is added, use the **tail** utility.

  ```
  # tail /etc/group
  ```

  The output returns:

  ```
  sysadmins:x:5000:
  ```

Additional resources

- **groupadd** man page

11.3.3. Adding a user to a supplementary group from the command line

You can add a user to a supplementary group to manage permissions or enable access to certain files or devices.

Prerequisites

- **root** access

Procedure

- To add a group to the supplementary groups of the user, use:

  ```
  # usermod --append -G group-name username
  ```

  Replace *group-name* with the name of the group, and replace *username* with the name of the user.

  Example 11.3. Adding a user to a supplementary group

  To add the user **sysadmin** to the group **system-administrators**, use:

  ```
  # usermod --append -G system-administrators sysadmin
  ```

Verification steps

- To verify the new groups is added to the supplementary groups of the user **sysadmin**, use:

  ```
  # groups sysadmin
  ```

  The output displays:

  ```
  sysadmin : sysadmin system-administrators
  ```

11.3.4. Creating a group directory

Under the UPG system configuration, you can apply the *set-group identification permission* (**setgid** bit) to a directory. The **setgid** bit makes managing group projects that share a directory simpler. When you apply the **setgid** bit to a directory, files created within that directory are automatically assigned to a group that owns the directory. Any user that has the permission to write and execute within this group can now create, modify, and delete files in the directory.

The following section describes how to create group directories.

Prerequisites

- **Root** access

Procedure

1. Create a directory:

   ```
   # mkdir directory-name
   ```

   Replace *directory-name* with the name of the directory.

2. Create a group:

```
# groupadd group-name
```

Replace *group-name* with the name of the group.

3. Add users to the group:

```
# usermod --append -G group-name username
```

Replace *group-name* with the name of the group, and replace *username* with the name of the user.

4. Associate the user and group ownership of the directory with the *group-name* group:

```
# chgrp group-name directory-name
```

Replace *group-name* with the name of the group, and replace *directory-name* with the name of the directory.

5. Set the write permissions to allow the users to create and modify files and directories and set the **setgid** bit to make this permission be applied within the *directory-name* directory:

```
# chmod g+rwxs directory-name
```

Replace *directory-name* with the name of the directory.

Now all members of the ***group-name*** group can create and edit files in the ***directory-name*** directory. Newly created files retain the group ownership of ***group-name*** group.

Verification steps

- To verify the correctness of set permissions, use:

```
# ls -ld directory-name
```

Replace *directory-name* with the name of the directory.

The output returns:

```
drwxrwsr-x. 2 root group-name 6 Nov 25 08:45 directory-name
```

11.3.5. Removing a user on the command line

You can remove a user account using the command line. In addition to removing the user account, you can optionally remove the user data and metadata, such as their home directory and configuration files.

Prerequisites

- You have **root** access.
- The user currently exists.
- Ensure that the user is logged out:

```
# loginctl terminate-user user-name
```

Procedure

- To only remove the user account, and not the user data:

```
# userdel user-name
```

- To remove the user, the data, and the metadata:

  a. Remove the user, their home directory, their mail spool, and their SELinux user mapping:

  ```
  # userdel --remove --selinux-user user-name
  ```

  b. Remove additional user metadata:

  ```
  # rm -rf /var/lib/AccountsService/users/user-name
  ```

  This directory stores information that the system needs about the user before the home directory is available. Depending on the system configuration, the home directory might not be available until the user authenticates at the login screen.

  IMPORTANT

  If you do not remove this directory and you later recreate the same user, the recreated user will still use certain settings inherited from the removed user.

Additional resources

- The **userdel(8)** man page.

11.4. MANAGING USER ACCOUNTS IN THE WEB CONSOLE

The RHEL web console offers a graphical interface that enables you to execute a wide range of administrative tasks without accessing your terminal directly. For example, you can add, edit or remove system user accounts.

After reading this section, you will know:

- From where the existing accounts come from.
- How to add new accounts.
- How to set password expiration.
- How and when to terminate user sessions.

Prerequisites

- Set up the RHEL web console. For details, see Getting started using the RHEL web console .
- Log in to the RHEL web console with an account that has administrator permissions assigned. For details, see Logging in to the RHEL web console .

11.4.1. System user accounts managed in the web console

With user accounts displayed in the RHEL web console you can:

- Authenticate users when accessing the system.
- Set the access rights to the system.

The RHEL web console displays all user accounts located in the system. Therefore, you can see at least one user account just after the first login to the web console.

After logging into the RHEL web console, you can perform the following operations:

- Create new users accounts.
- Change their parameters.
- Lock accounts.
- Terminate user sessions.

11.4.2. Adding new accounts using the web console

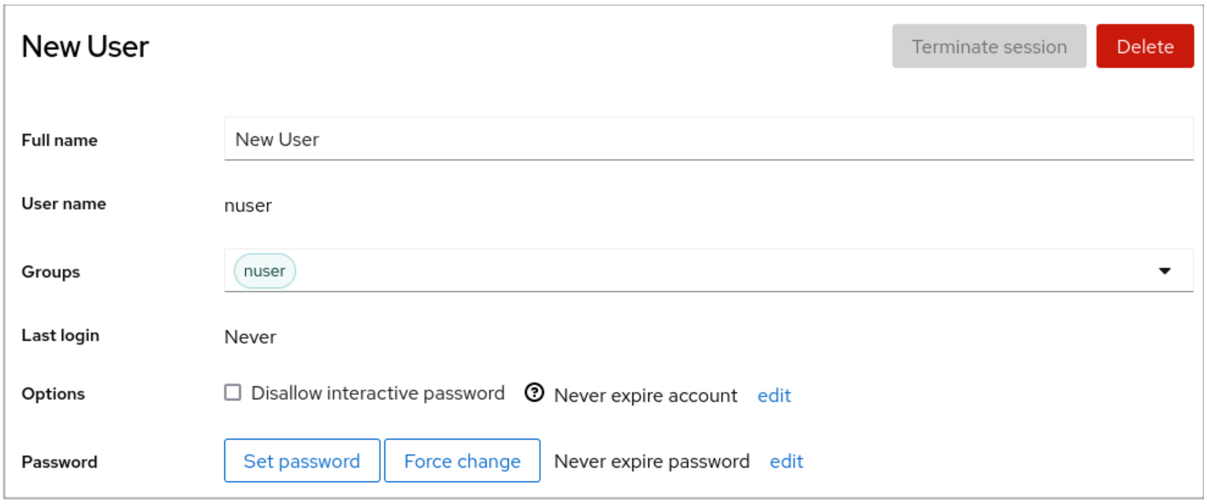
Use the following steps for adding user accounts to the system and setting administration rights to the accounts through the RHEL web console.

Prerequisites

- The RHEL web console must be installed and accessible. For details, see Installing the web console.

Procedure

1. Log in to the RHEL web console.
2. Click **Accounts**.
3. Click **Create New Account**.
4. In the **Full Name** field, enter the full name of the user.
   The RHEL web console automatically suggests a user name from the full name and fills it in the **User Name** field. If you do not want to use the original naming convention consisting of the first letter of the first name and the whole surname, update the suggestion.
5. In the **Password/Confirm** fields, enter the password and retype it for verification that your password is correct.
   The color bar below the fields shows you the security level of the entered password, which does not allow you to create a user with a weak password.
6. Click **Create** to save the settings and close the dialog box.
7. Select the newly created account.
8. In the **Groups** drop down menu, select the groups that you want to add to the new account.



Now you can see the new account in the **Accounts** settings and you can use its credentials to connect to the system.

11.4.3. Enforcing password expiration in the web console

By default, user accounts have set passwords to never expire. You can set system passwords to expire after a defined number of days. When the password expires, the next login attempt will prompt for a password change.

Procedure

1. Log in to the RHEL 8 web console.
2. Click **Accounts**.
3. Select the user account for which you want to enforce password expiration.
4. Click **edit** on the **Password** line.

5. In the **Password expiration** dialog box, select **Require password change every … days**and enter a positive whole number representing the number of days after which the password expires.
6. Click **Change**.
The web console immediately shows the date of the future password change request on the **Password** line.

11.4.4. Terminating user sessions in the web console

A user creates user sessions when logging into the system. Terminating user sessions means to log the user out from the system. It can be helpful if you need to perform administrative tasks sensitive to configuration changes, for example, system upgrades.

In each user account in the RHEL 8web console, you can terminate all sessions for the account except for the web console session you are currently using. This prevents you from loosing access to your system.

Procedure

1. Log in to the RHEL 8 web console.
2. Click **Accounts**.
3. Click the user account for which you want to terminate the session.
4. Click **Terminate Session**.
   If the **Terminate Session** button is inactive, the user is not logged in to the system.

   The RHEL web console terminates the sessions.

11.5. EDITING USER GROUPS USING THE COMMAND LINE

A user belongs to a certain set of groups that allow a logical collection of users with a similar access to files and folders. You can edit the primary and supplementary user groups from the command line to change the user's permissions.

11.5.1. Primary and supplementary user groups

A group is an entity which ties together multiple user accounts for a common purpose, such as granting access to particular files.

On Linux, user groups can act as primary or supplementary. Primary and supplementary groups have the following properties:

Primary group

- Every user has just one primary group at all times.
- You can change the user's primary group.

Supplementary groups

- You can add an existing user to an existing supplementary group to manage users with the same security and access privileges within the group.
- Users can be members of zero or multiple supplementary groups.

11.5.2. Listing the primary and supplementary groups of a user

You can list the groups of users to see which primary and supplementary groups they belong to.

Procedure

- Display the names of the primary and any supplementary group of a user:

  ```
  $ groups user-name
  ```

  Replace *user-name* with the name of the user. If you do not provide a user name, the command displays the group membership for the current user. The first group is the primary group followed by the optional supplementary groups.

Example 11.4. Listing of groups for user sarah:

```
$ groups sarah
```

The output displays:

```
sarah : sarah wheel developer
```

User **sarah** has a primary group **sarah** and is a member of supplementary groups **wheel** and **developer**.

Example 11.5. Listing of groups for user marc:

```
$ groups marc
```

The output displays:

```
marc : marc
```

User **marc** has only a primary group **marc** and no supplementary groups.

11.5.3. Changing the primary group of a user

You can change the primary group of an existing user to a new group.

Prerequisites:

1. **root** access
2. The new group must exist

Procedure

- Change the primary group of a user:

  ```
  # usermod -g group-name user-name
  ```

  Replace *group-name* with the name of the new primary group, and replace *user-name* with the name of the user.

> **NOTE**
>
> When you change a user's primary group, the command also automatically changes the group ownership of all files in the user's home directory to the new primary group. You must fix the group ownership of files outside of the user's home directory manually.

Example 11.6. Example of changing the primary group of a user:

> If the user **sarah** belongs to the primary group **sarah1**, and you want to change the primary group of the user to **sarah2**, use:
>
> ```
> # usermod -g sarah2 sarah
> ```

Verification steps

- Verify that you changed the primary group of the user:

  ```
  $ groups sarah
  ```

  The output displays:

  ```
  sarah : sarah2
  ```

11.5.4. Adding a user to a supplementary group from the command line

You can add a user to a supplementary group to manage permissions or enable access to certain files or devices.

Prerequisites

- **root** access

Procedure

- To add a group to the supplementary groups of the user, use:

  ```
  # usermod --append -G group-name username
  ```

  Replace *group-name* with the name of the group, and replace *username* with the name of the user.

  > **Example 11.7. Adding a user to a supplementary group**
  >
  > To add the user **sysadmin** to the group **system-administrators**, use:
  >
  > ```
  > # usermod --append -G system-administrators sysadmin
  > ```

Verification steps

- To verify the new groups is added to the supplementary groups of the user **sysadmin**, use:

  ```
  # groups sysadmin
  ```

  The output displays:

  ```
  sysadmin : sysadmin system-administrators
  ```

11.5.5. Removing a user from a supplementary group

You can remove an existing user from a supplementary group to limit their permissions or access to files and devices.

Prerequisites

- **root** access

Procedure

- Remove a user from a supplementary group:

  ```
  # gpasswd -d user-name group-name
  ```

  Replace *user-name* with the name of the user, and replace *group-name* with the name of the supplementary group.

  Example 11.8. Removing user from a supplementary group

  If the user sarah has a primary group **sarah2**, and belongs to the secondary groups **wheel** and **developers**, and you want to remove that user from the group **developers**, use:

  ```
  # gpasswd -d sarah developers
  ```

Verification steps

- Verify that you removed the user sarah from the secondary group developers:

  ```
  $ groups sarah
  ```

  The output displays:

  ```
  sarah : sarah2 wheel
  ```

11.5.6. Changing all of the supplementary groups of a user

You can overwrite the list of supplementary groups that you want the user to remain a member of.

Prerequisites

- **root** access
- The supplementary groups must exist

Procedure

- Overwrite a list of user's supplementary groups:

  ```
  # usermod -G group-names username
  ```

Replace *group-names* with the name of one or more supplementary groups. To add the user to several supplementary groups at once, separate the group names using commas and no intervening spaces. For example: **wheel,developer**.

Replace *user-name* with the name of the user.

> **IMPORTANT**
>
> If the user is currently a member of a group that you do not specify, the command removes the user from the group.

Example 11.9. Changing the list of supplementary groups of a user

If the user **sarah** has a primary group **sarah2**, and belongs to the supplementary group **wheel**, and you want the user to belong to three more supplementary groups **developer**, **sysadmin**, and **security**, use:

```
# usermod -G wheel,developer,sysadmin,security sarah
```

Verification steps

- Verify that you set the list of the supplementary groups correct:

```
# groups sarah
```

The output displays:

```
sarah : sarah2 wheel developer sysadmin security
```

11.6. CHANGING AND RESETTING THE ROOT PASSWORD

If the existing root password is no longer satisfactory or is forgotten, you can change or reset it both as the **root** user and a non-root user.

11.6.1. Changing the root password as the root user

You can use the **passwd** command to change the **root** password as the **root** user.

Prerequisites

- **Root** access

Procedure

- To change the **root** password, use:

```
# passwd
```

You are prompted to enter your current password before you can change it.

11.6.2. Changing or resetting the forgotten root password as a non-root user

You can use the **passwd** command to change or reset the forgotten **root** password as a non-root user.

Prerequisites

- You are able to log in as a non-root user.
- You are a member of the administrative **wheel** group.

Procedure

- To change or reset the **root** password as a non-root user that belongs to the **wheel** group, use:

  ```
  $ sudo passwd root
  ```

  You are prompted to enter your current non-root password before you can change the **root** password.

11.6.3. Resetting the root password on boot

If you are unable to log in as a non-root user or do not belong to the administrative **wheel** group, you can reset the root password on boot by switching into a specialized **chroot jail** environment.

Procedure

1. Reboot the system and, on the GRUB 2 boot screen, press the **e** key to interrupt the boot process.
   The kernel boot parameters appear.

   ```
   load_video
   set gfx_payload=keep
   insmod gzio
   linux ($root)/vmlinuz-4.18.0-80.e18.x86_64 root=/dev/mapper/rhel-root ro crash\
   kernel=auto resume=/dev/mapper/rhel-swap rd.lvm.lv/swap rhgb quiet
   initrd ($root)/initramfs-4.18.0-80.e18.x86_64.img $tuned_initrd
   ```

2. Go to the end of the line that starts with **linux**.

   ```
   linux ($root)/vmlinuz-4.18.0-80.e18.x86_64 root=/dev/mapper/rhel-root ro crash\
   kernel=auto resume=/dev/mapper/rhel-swap rd.lvm.lv/swap rhgb quiet
   ```

   Press **Ctrl+e** to jump to the end of the line.

3. Add **rd.break** to the end of the line that starts with **linux**.

   ```
   linux ($root)/vmlinuz-4.18.0-80.e18.x86_64 root=/dev/mapper/rhel-root ro crash\
   kernel=auto resume=/dev/mapper/rhel-swap rd.lvm.lv/swap rhgb quiet rd.break
   ```

4. Press **Ctrl+x** to start the system with the changed parameters.
   The **switch_root** prompt appears.

5. Remount the file system as writable:

```
mount -o remount,rw /sysroot
```

The file system is mounted as read-only in the **/sysroot** directory. Remounting the file system as writable allows you to change the password.

6. Enter the **chroot** environment:

```
chroot /sysroot
```

The **sh-4.4#** prompt appears.

7. Reset the **root** password:

```
passwd
```

Follow the instructions displayed by the command line to finalize the change of the **root** password.

8. Enable the SELinux relabeling process on the next system boot:

```
touch /.autorelabel
```

9. Exit the **chroot** environment:

```
exit
```

10. Exit the **switch_root** prompt:

```
exit
```

11. Wait until the SELinux relabeling process is finished. Note that relabeling a large disk might take a long time. The system reboots automatically when the process is complete.

Verification steps

1. To verify that the **root** password is successfully changed, log in as a normal user and open the Terminal.

2. Run the interactive shell as root:

```
$ su
```

3. Enter your new **root** password.

4. Print the user name associated with the current effective user ID:

```
# whoami
```

The output returns:

```
root
```

CHAPTER 12. MANAGING SUDO ACCESS

System administrators can grant **sudo** access to allow non-root users to execute administrative commands that are normally reserved for the root user. As a result, non-root users can execute such commands without logging in to the root user account.

12.1. USER AUTHORIZATIONS IN SUDOERS

The **/etc/sudoers** file specifies which users can use the **sudo** command to execute other commands. The rules can apply to individual users and user groups. You can also define rules for groups of hosts, commands, and even users more easily by using aliases. Default aliases are defined in the first part of the **/etc/sudoers** file.

When a user enters a command with **sudo** for which the user does not have authorization, the system records a message that contains the string ***<username>*** **: user NOT in sudoers** to the journal log.

The default /**etc/sudoers** file provides information and examples of authorizations. You can activate a specific example rule by uncommenting the corresponding line. The section with user authorizations is marked with the following introduction:

```
## Next comes the main part: which users can run what software on
## which machines  (the sudoers file can be shared between multiple
## systems).
```

You can create new **sudoers** authorizations and modify existing authorizations by using the following format:

```
<username> <hostname.example.com>=(<run_as_user>:<run_as_group>) <path/to/command>
```

Where:

- ***<username>*** is the user that enters the command, for example, **user1**. If the value starts with **%**, it defines a group, for example, **%group1**.
- ***<hostname.example.com>*** is the name of the host on which the rule applies.
- The section **(*<run_as_user>*:*<run_as_group>*)** defines the user or group as which the command is executed. If you omit this section, ***<username>*** can execute the command as root.
- ***<path/to/command>*** is the complete absolute path to the command. You can also limit the user to only performing a command with specific options and arguments by adding those options after the command path. If you do not specify any options, the user can use the command with all options.

You can apply the rule to all users, hosts, or commands by replacing any of these variables with **ALL**.

> **WARNING**
>
> With overly permissive rules, such as **ALL ALL=(ALL) ALL**, all users can run all commands as all users on all hosts. This presents serious security risks.

You can specify the arguments negatively by using the **!** operator. For example, **!root** specifies all users except root. Note that allowing specific users, groups, and commands is more secure than disallowing specific users, groups, and commands. This is because allow rules also block new unauthorized users or groups.

> **WARNING**
>
> Avoid using negative rules for commands because users can overcome such rules by renaming commands with the **alias** command.

The system reads the **/etc/sudoers** file from beginning to end. Therefore, if the file contains multiple entries for a user, the entries are applied in order. In case of conflicting values, the system uses the last match, even if it is not the most specific match.

To preserve the rules during system updates and for easier fixing of errors, enter new rules by creating new files in the **/etc/sudoers.d/** directory instead of entering rules directly to the **/etc/sudoers** file. The system reads the files in the **/etc/sudoers.d** directory when it reaches the following line in the **/etc/sudoers** file:

```
#includedir /etc/sudoers.d
```

Note that the number sign (**#**) at the beginning of this line is part of the syntax and does not mean the line is a comment. The names of files in that directory must not contain a period and must not end with a tilde (~).

Additional resources

- **sudo(8)** and **sudoers(5)** man pages

12.2. GRANTING SUDO ACCESS TO A USER

System administrators can allow non-root users to execute administrative commands by granting them **sudo** access. The **sudo** command provides users with administrative access without using the password of the root user.

When users need to perform an administrative command, they can precede that command with **sudo**. If the user has authorization for the command, the command is executed as if they were root.

Be aware of the following limitations:

- Only users listed in the **/etc/sudoers** configuration file can use the **sudo** command.
- The command is executed in the shell of the user, not in the root shell.

Prerequisites

- You have root access to the system.

Procedure

1. As root, open the **/etc/sudoers** file.

   ```
   # visudo
   ```

   The **/etc/sudoers** file defines the policies applied by the **sudo** command.

2. In the **/etc/sudoers** file, find the lines that grant **sudo** access to users in the administrative **wheel** group.

   ```
   ## Allows people in group wheel to run all commands
   %wheel        ALL=(ALL)       ALL
   ```

3. Make sure the line that starts with **%wheel** is not commented out with the number sign (**#**).

4. Save any changes, and exit the editor.

5. Add users you want to grant **sudo** access to into the administrative **wheel** group.

   ```
   # usermod --append -G wheel <username>
   ```

   Replace ***<username>*** with the name of the user.

Verification steps

- Verify that the user is in the administrative **wheel** group:

  ```
  # id <username>
  uid=5000(<username>) gid=5000(<username>) groups=5000(<username>),10(wheel)
  ```

Additional resources

- **sudo(8)**, **visudo(8)**, and **sudoers(5)** man pages

12.3. ENABLING UNPRIVILEGED USERS TO RUN CERTAIN COMMANDS

As an administrator, you can allow unprivileged users to enter certain commands on specific workstations by configuring a policy in the **/etc/sudoers.d/** directory.

Prerequisites

- You have root access to the system.

Procedure

1. As root, create a new **sudoers.d** directory under **/etc/**:

   ```
   # mkdir -p /etc/sudoers.d/
   ```

2. Create a new file in the **/etc/sudoers.d** directory:

   ```
   # visudo -f /etc/sudoers.d/<filename>
   ```

   The file opens automatically.

3. Add the following line to the **/etc/sudoers.d/*<filename>*** file:

```
<username> <hostname.example.com> = (<run_as_user>:<run_as_group>)
<path/to/command>
```

   - Replace ***<username>*** with the name of the user.
   - Replace ***<hostname.example.com>*** with the URL of the host.
   - Replace **(*<run_as_user>*:*<run_as_group>*)** with the user or group as which the command can be executed. If you omit this section, ***<username>*** can execute the command as root.
   - Replace ***<path/to/command>*** with the complete absolute path to the command. You can also limit the user to only performing a command with specific options and arguments by adding those options after the command path. If you do not specify any options, the user can use the command with all options.
   - To allow two and more commands on the same host on one line, you can list them separated by a comma followed by a space.
     For example, to allow **user1** to execute the **dnf** and **reboot** commands on **host1.example.com**, enter **user1 host1.example.com = /bin/dnf, /sbin/reboot**.

4. Optional: To receive email notifications every time the user attempts to use **sudo** privileges, add the following lines to the file:

```
Defaults    mail_always
Defaults    mailto="<email@example.com>"
```

5. Save the changes, and exit the editor.

Verification

1. To verify if a user can run a command with **sudo** privileges, switch the account:

```
# su <username> -
```

2. As the user, enter the command with the **sudo** command:

```
$ sudo <command>
[sudo] password for <username>:
```

   Enter the user's **sudo** password.

3. If the privileges are configured correctly, the system displays the list of commands and options. For example, with the **dnf** command, it shows the following output:

```
...
usage: dnf [options] COMMAND
...
```

   If the system returns the error message ***<username>* is not in the sudoers file. This incident will be reported**, the file for ***<username>*** in **/etc/sudoers.d/** does not exist.

If the system returns the error message ***<username>* is not allowed to run sudo on *<host.example.com>***, the configuration was not completed correctly. Ensure that you are logged in as root and that the configuration was performed correctly.

If the system returns the error message **Sorry, user *<username>* is not allowed to execute '*<path/to/command>*' as root on *<host.example.com>*.**, the command is not correctly defined in the rule for the user.

Additional resources

- **sudo(8)**, **visudo(8)**, and **sudoers(5)** man pages

CHAPTER 13. MANAGING FILE SYSTEM PERMISSIONS

File system permissions control the ability of user and group accounts to read, modify, and execute the contents of the files and to enter directories. Set permissions carefully to protect your data against unauthorized access.

13.1. MANAGING FILE PERMISSIONS

Every file or directory has three levels of ownership:

- User owner (**u**).
- Group owner (**g**).
- Others (**o**).

Each level of ownership can be assigned the following permissions:

- Read (**r**).
- Write (**w**).
- Execute (**x**).

Note that the execute permission for a file allows you to execute that file. The execute permission for a directory allows you to access the contents of the directory, but not execute it.

When a new file or directory is created, the default set of permissions are automatically assigned to it. The default permissions for a file or directory are based on two factors:

- Base permission.
- The *user file-creation mode mask* (**umask**).

13.1.1. Base file permissions

Whenever a new file or directory is created, a base permission is automatically assigned to it. Base permissions for a file or directory can be expressed in *symbolic* or *octal* values.

Permission	Symbolic value	Octal value
No permission	---	0
Execute	--x	1
Write	-w-	2
Write and execute	-wx	3
Read	r--	4
Read and execute	r-x	5

Read and write	rw-	6
Read, write, execute	rwx	7

The base permission for a directory is **777** (**drwxrwxrwx**), which grants everyone the permissions to read, write, and execute. This means that the directory owner, the group, and others can list the contents of the directory, create, delete, and edit items within the directory, and descend into it.

Note that individual files within a directory can have their own permission that might prevent you from editing them, despite having unrestricted access to the directory.

The base permission for a file is **666** (**-rw-rw-rw-**), which grants everyone the permissions to read and write. This means that the file owner, the group, and others can read and edit the file.

Example 13.1. Permissions for a file

If a file has the following permissions:

```
$ ls -l
-rwxrw----. 1 sysadmins sysadmins 2 Mar 2 08:43 file
```

- **-** indicates it is a file.
- **rwx** indicates that the file owner has permissions to read, write, and execute the file.
- **rw-** indicates that the group has permissions to read and write, but not execute the file.
- **---** indicates that other users have no permission to read, write, or execute the file.
- **.** indicates that the SELinux security context is set for the file.

Example 13.2. Permissions for a directory

If a directory has the following permissions:

```
$ ls -dl directory
drwxr-----. 1 sysadmins sysadmins 2 Mar 2 08:43 directory
```

- **d** indicates it is a directory.
- **rwx** indicates that the directory owner has the permissions to read, write, and access the contents of the directory.
  As a directory owner, you can list the items (files, subdirectories) within the directory, access the content of those items, and modify them.
- **r-x** indicates that the group has permissions to read the content of the directory, but not write - create new entries or delete files. The **x** permission means that you can also access the directory using the **cd** command.
- **---** indicates that other users have no permission to read, write, or access the contents of the directory.

As someone who is not a user owner, or as group owner of the directory, you cannot list the items within the directory, access information about those items, or modify them.

- **.** indicates that the SELinux security context is set for the directory.

NOTE

The base permission that is automatically assigned to a file or directory is **not** the default permission the file or directory ends up with. When you create a file or directory, the base permission is altered by the *umask*. The combination of the base permission and the *umask* creates the default permission for files and directories.

13.1.2. User file-creation mode mask

The user file-creation mode mask (*umask*) is variable that controls how file permissions are set for newly created files and directories. The *umask* automatically removes permissions from the base permission value to increase the overall security of a Linux system. The *umask* can be expressed in *symbolic* or *octal* values.

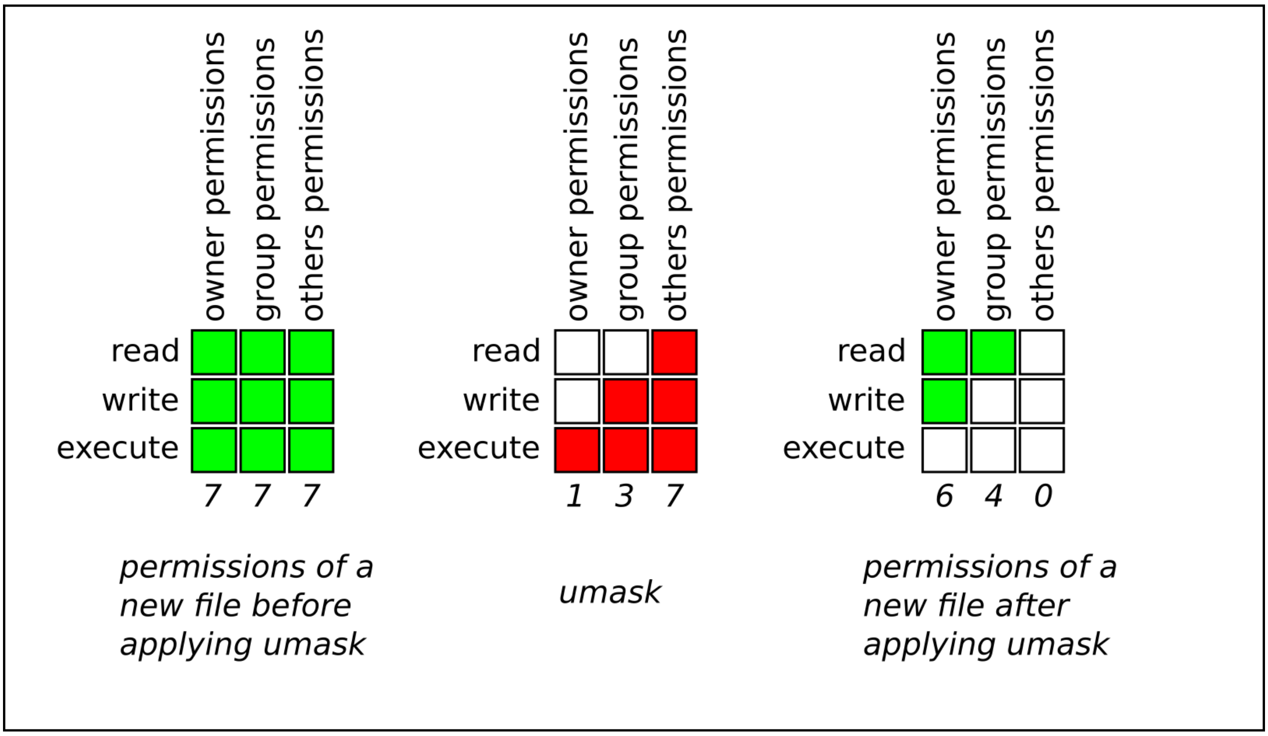
Permission	Symbolic value	Octal value
Read, write, and execute	rwx	0
Read and write	rw-	1
Read and execute	r-x	2
Read	r--	3
Write and execute	-wx	4
Write	-w-	5
Execute	--x	6
No permissions	---	7

The default *umask* for a standard user is **0002**. The default *umask* for a **root** user is **0022**.

The first digit of the *umask* represents special permissions (sticky bit,). The last three digits of the *umask* represent the permissions that are removed from the user owner (**u**), group owner (**g**), and others (**o**) respectively.

Example 13.3. Applying the umask when creating a file

The following example illustrates how the *umask* with an octal value of **0137** is applied to the file with the base permission of **777**, to create the file with the default permission of **640**.



13.1.3. Default file permissions

The default permissions are set automatically for all newly created files and directories. The value of the default permissions is determined by applying the *umask* to the base permission.

Example 13.4. Default permissions for a directory created by a standard user

When a **standard user** creates a new **directory**, the *umask* is set to **002** (**rwxrwxr-x**), and the base permissions for a directory are set to **777** (**rwxrwxrwx**). This brings the default permissions to **775** (**drwxrwxr-x**).

	Symbolic value	Octal value
Base permission	rwxrwxrwx	777
Umask	rwxrwxr-x	002
Default permission	rwxrwxr-x	775

This means that the directory owner and the group can list the contents of the directory, create, delete, and edit items within the directory, and descend into it. Other users can only list the contents of the directory and descend into it.

Example 13.5. Default permissions for a file created by a standard user

When a **standard user** creates a new **file**, the *umask* is set to **002** (**rwxrwxr-x**), and the base permissions for a file are set to **666** (**rw-rw-rw-**). This brings the default permissions to **664** (**-rw-rw-r--**).

	Symbolic value	Octal value
Base permission	rw-rw-rw-	666
Umask	rwxrwxr-x	002
Default permission	rw-rw-r--	664

This means that the file owner and the group can read and edit the file, while other users can only read the file.

Example 13.6. Default permissions for a directory created by the root user

When a **root user** creates a new **directory**, the *umask* is set to **022** (**rwxr-xr-x**), and the base permissions for a directory are set to **777** (**rwxrwxrwx**). This brings the default permissions to **755** (**rwxr-xr-x**).

	Symbolic value	Octal value
Base permission	rwxrwxrwx	777
Umask	rwxr-xr-x	022
Default permission	rwxr-xr-x	755

This means that the directory owner can list the contents of the directory, create, delete, and edit items within the directory, and descend into it. The group and others can only list the contents of the directory and descend into it.

Example 13.7. Default permissions for a file created by the root user

When a **root user** creates a new **file**, the *umask* is set to **022** (**rwxr-xr-x**), and the base permissions for a file are set to **666** (**rw-rw-rw-**). This brings the default permissions to **644** (**-rw-r—r--**).

	Symbolic value	Octal value
Base permission	rw-rw-rw-	666
Umask	rwxr-xr-x	022
Default permission	rw-r—r--	644

This means that the file owner can read and edit the file, while the group and others can only read the file.

> **NOTE**
>
> For security reasons, regular files cannot have execute permissions by default, even if the *umask* is set to **000** (**rwxrwxrwx**). However, directories can be created with execute permissions.

13.1.4. Changing file permissions using symbolic values

You can use the **chmod** utility with symbolic values (a combination letters and signs) to change file permissions for a file or directory.

You can assign the following *permissions*:

- Read (**r**)
- Write (**w**)
- Execute (**x**)

Permissions can be assigned to the following *levels of ownership*:

- User owner (**u**)
- Group owner (**g**)
- Other (**o**)
- All (**a**)

To add or remove permissions you can use the following *signs*:

- **+** to add the permissions on top of the existing permissions
- **-** to remove the permissions from the existing permission
- **=** to remove the existing permissions and explicitly define the new ones

Procedure

- To change the permissions for a file or directory, use:

  ```
  $ chmod <level><operation><permission> file-name
  ```

  Replace ***<level>*** with the level of ownership you want to set the permissions for. Replace ***<operation>*** with one of the signs. Replace ***<permission>*** with the permissions you want to assign. Replace *file-name* with the name of the file or directory. For example, to grant everyone the permissions to read, write, and execute (**rwx**) **my-script.sh**, use the **chmod a=rwx my-script.sh** command.

  See Base file permissions for more details.

Verification steps

- To see the permissions for a particular file, use:

  ```
  $ ls -l file-name
  ```

Replace *file-name* with the name of the file.

- To see the permissions for a particular directory, use:

```
$ ls -dl directory-name
```

Replace *directory-name* with the name of the directory.

- To see the permissions for all the files within a particular directory, use:

```
$ ls -l directory-name
```

Replace *directory-name* with the name of the directory.

Example 13.8. Changing permissions for files and directories

- To change file permissions for **my-file.txt** from **-rw-rw-r--** to **-rw------**, use:

1. Display the current permissions for **my-file.txt**:

```
$ ls -l my-file.txt
-rw-rw-r--. 1 username username 0 Feb 24 17:56 my-file.txt
```

2. Remove the permissions to read, write, and execute (**rwx**) the file from group owner (**g**) and others (**o**):

```
$ chmod go= my-file.txt
```

Note that any permission that is not specified after the equals sign (**=**) is automatically prohibited.

3. Verify that the permissions for **my-file.txt** were set correctly:

```
$ ls -l my-file.txt
-rw-------. 1 username username 0 Feb 24 17:56 my-file.txt
```

- To change file permissions for **my-directory** from **drwxrwx---** to **drwxrwxr-x**, use:

1. Display the current permissions for **my-directory**:

```
$ ls -dl my-directory
drwxrwx---. 2 username username 4096 Feb 24 18:12 my-directory
```

2. Add the read and execute (**r-x**) access for all users (**a**):

```
$ chmod o+rx my-directory
```

3. Verify that the permissions for **my-directory** and its content were set correctly:

```
$ ls -dl my-directory
drwxrwxr-x. 2 username username 4096 Feb 24 18:12 my-directory
```

13.1.5. Changing file permissions using octal values

You can use the **chmod** utility with octal values (numbers) to change file permissions for a file or directory.

Procedure

- To change the file permissions for an existing file or directory, use:

```
$ chmod octal_value file-name
```

Replace *file-name* with the name of the file or directory. Replace *octal_value* with an octal value. See Base file permissions for more details.

13.2. MANAGING THE ACCESS CONTROL LIST

Each file and directory can only have one user owner and one group owner at a time. If you want to grant a user permissions to access specific files or directories that belong to a different user or group while keeping other files and directories private, you can utilize Linux Access Control Lists (ACLs).

13.2.1. Displaying the current Access Control List

You can use the **getfacl** utility to display the current ACL.

Procedure

- To display the current ACL for a particular file or directory, use:

```
$ getfacl file-name
```

Replace *file-name* with the name of the file or directory.

13.2.2. Setting the Access Control List

You can use the **setfacl** utility to set the ACL for a file or directory.

Prerequisites

- **root** access.

Procedure

- To set the ACL for a file or directory, use:

```
# setfacl -m u:username:symbolic_value file-name
```

Replace *username* with the name of the user, *symbolic_value* with a symbolic value, and *file-name* with the name of the file or directory. For more information see the **setfacl** man page.

Example 13.9. Modifying permissions for a group project

The following example describes how to modify permissions for the **group-project** file owned by the **root** user that belongs to the **root** group so that this file is:

- Not executable by anyone.
- The user **andrew** has the **rw-** permissions.
- The user **susan** has the **---** permissions.
- Other users have the **r--** permissions.

Procedure

```
# setfacl -m u:andrew:rw- group-project
# setfacl -m u:susan:--- group-project
```

Verification steps

- To verify that the user **andrew** has the **rw-** permission, the user **susan** has the **---** permission, and other users have the **r--** permission, use:

  ```
  $ getfacl group-project
  ```

  The output returns:

  ```
  # file: group-project
  # owner: root
  # group: root
  user:andrew:rw-
  user:susan:---
  group::r--
  mask::rw-
  other::r--
  ```

13.3. MANAGING THE UMASK

You can use the **umask** utility to display, set, or change the current or default value of the *umask*.

13.3.1. Displaying the current value of the umask

You can use the **umask** utility to display the current value of the *umask* in symbolic or octal mode.

Procedure

- To display the current value of the *umask* in symbolic mode, use:

  ```
  $ umask -S
  ```

- To display the current value of the *umask* in the octal mode, use:

  ```
  $ umask
  ```

> **NOTE**
>
> When displaying the *umask* in octal mode, you may notice it displayed as a four digit number (**0002** or **0022**). The first digit of the *umask* represents a special bit (sticky bit, SGID bit, or SUID bit). If the first digit is set to **0**, the special bit is not set.

13.3.2. Displaying the default bash umask

There are a number of shells you can use, such as **bash**, **ksh**, **zsh** and **tcsh**. Those shells can behave as login or non-login shells. You can invoke the login shell by opening a native or a GUI terminal.

To determine whether you are executing a command in a login or a non-login shell, use the **echo $0** command.

Example 13.10. Determining if you are working in a login or a non-login bash shell

- If the output of the **echo $0** command returns **bash**, you are executing the command in a non-login shell.

  ```
  $ echo $0
  bash
  ```

  The default *umask* for the non-login shell is set in the **/etc/bashrc** configuration file.

- If the output of the **echo $0** command returns **-bash**, you are executing the command in a login shell.

  ```
  # echo $0
  -bash
  ```

  The default *umask* for the login shell is set in the **/etc/profile** configuration file.

Procedure

- To display the default **bash** *umask* for the non-login shell, use:

  ```
  $ grep umask /etc/bashrc
  ```

  The output returns:

  ```
  # By default, we want umask to get set. This sets it for non-login shell.
         umask 002
         umask 022
  ```

- To display the default **bash** *umask* for the login shell, use:

  ```
  $ grep umask /etc/profile
  ```

  The output returns:

```
# By default, we want umask to get set. This sets it for login shell
        umask 002
        umask 022
```

13.3.3. Setting the umask using symbolic values

You can use the **umask** utility with symbolic values (a combination letters and signs) to set the *umask* for the current shell session

You can assign the following *permissions*:

- Read (**r**)
- Write (**w**)
- Execute (**x**)

Permissions can be assigned to the following *levels of ownership*:

- User owner (**u**)
- Group owner (**g**)
- Other (**o**)
- All (**a**)

To add or remove permissions you can use the following *signs*:

- **+** to add the permissions on top of the existing permissions
- **-** to remove the permissions from the existing permission
- **=** to remove the existing permissions and explicitly define the new ones

NOTE

Any permission that is not specified after the equals sign (**=**) is automatically prohibited.

Procedure

- To set the *umask* for the current shell session, use:

  ```
  $ umask -S <level><operation><permission>
  ```

  Replace ***<level>*** with the level of ownership you want to set the umask for. Replace ***<operation>*** with one of the signs. Replace ***<permission>*** with the permissions you want to assign. For example, to set the *umask* to **u=rwx,g=rwx,o=rwx**, use **umask -S a=rwx**.

  See User file-creation mode for more details.

NOTE

The *umask* is only valid for the current shell session.

13.3.4. Setting the umask using octal values

You can use the **umask** utility with octal values (numbers) to set the *umask* for the current shell session.

Procedure

- To set the *umask* for the current shell session, use:

```
$ umask octal_value
```

Replace *octal_value* with an octal value. See User file-creation mode mask for more details.

> **NOTE**
>
> The *umask* is only valid for the current shell session.

13.3.5. Changing the default umask for the non-login shell

You can change the default **bash** *umask* for standard users by modifying the **/etc/bashrc** file.

Prerequisites

- **root** access

Procedure

1. As **root**, open the **/etc/bashrc** file in the editor.
2. Modify the following sections to set a new default **bash** *umask*:

```
if [ $UID -gt 199 ] && [ “id -gn” = “id -un” ]; then
  umask 002
else
  umask 022
fi
```

Replace the default octal value of the *umask* (**002**) with another octal value. See User file-creation mode mask for more details.

3. Save the changes and exit the editor.

13.3.6. Changing the default umask for the login shell

You can change the default **bash** *umask* for the **root** user by modifying the **/etc/profile** file.

Prerequisites

- **root** access

Procedure

1. As **root**, open the **/etc/profile** file in the editor.
2. Modify the following sections to set a new default **bash** *umask*:

```
if [ $UID -gt 199 ] && [ "/usr/bin/id -gn" = "/usr/bin/id -un" ]; then
   umask 002
else
   umask 022
fi
```

Replace the default octal value of the *umask* (**022**) with another octal value. See User file-creation mode mask for more details.

3. Save the changes and exit the editor.

13.3.7. Changing the default umask for a specific user

You can change the default *umask* for a specific user by modifying the **.bashrc** for that user.

Procedure

- Append the line that specifies the octal value of the *umask* into the **.bashrc** file for the particular user.

```
$ echo 'umask octal_value' >> /home/username/.bashrc
```

Replace *octal_value* with an octal value and replace *username* with the name of the user. See User file-creation mode mask for more details.

13.3.8. Setting default permissions for newly created home directories

You can change the permission modes for home directories of newly created users by modifying the **/etc/login.defs** file.

Procedure

1. As **root**, open the **/etc/login.defs** file in the editor.

2. Modify the following section to set a new default *HOME_MODE*:

```
# HOME_MODE is used by useradd(8) and newusers(8) to set the mode for new
# home directories.
# If HOME_MODE is not set, the value of UMASK is used to create the mode.
HOME_MODE       0700
```

Replace the default octal value (**0700**) with another octal value. The selected mode will be used to create the permissions for the home directory.

3. If *HOME_MODE* is set, save the changes and exit the editor.

4. If *HOME_MODE* is not set, modify the *UMASK* to set the mode for the newly created home directories:

```
# Default initial "umask" value used by login(1) on non-PAM enabled systems.
# Default "umask" value for pam_umask(8) on PAM enabled systems.
# UMASK is also used by useradd(8) and newusers(8) to set the mode for new
# home directories if HOME_MODE is not set.
# 022 is the default value, but 027, or even 077, could be considered
```

```
# for increased privacy. There is no One True Answer here: each sysadmin
# must make up their mind.

UMASK       022
```

Replace the default octal value (**022**) with another octal value. See User file-creation mode mask for more details.

5. Save the changes and exit the editor.

CHAPTER 14. MANAGING SYSTEMD

As a system administrator, you can manage critical aspects of your system with **systemd**. Serving as a system and service manager for Linux operating systems, **systemd** software suite provides tools and services for controlling, reporting, and system initialization. Key features of **systemd** include:

- Parallel start of system services during boot
- On-demand activation of daemons
- Dependency-based service control logic

The basic object that **systemd** manages is a *systemd unit*, a representation of system resources and services. A **systemd** unit consists of a name, type and a configuration file that defines and manages a particular task. You can use unit files to configure system behavior. See the following examples of various systemd unit types:

Service

Controls and manages individual system services.

Target

Represents a group of units that define system states.

Device

Manages hardware devices and their availability.

Mount

Handles file system mounting.

Timer

Schedules tasks to run at specific intervals.

NOTE

To display all available unit types:

```
# systemctl -t help
```

14.1. SYSTEMD UNIT FILES LOCATIONS

You can find the unit configuration files in one of the following directories:

Table 14.1. systemd unit files locations

Directory	Description
/usr/lib/systemd/system/	**systemd** unit files distributed with installed RPM packages.
/run/systemd/system/	**systemd** unit files created at run time. This directory takes precedence over the directory with installed service unit files.

Directory	Description
/etc/systemd/system/	**systemd** unit files created by using the **systemctl enable** command as well as unit files added for extending a service. This directory takes precedence over the directory with runtime unit files.

The default configuration of **systemd** is defined during the compilation and you can find the configuration in the **/etc/systemd/system.conf** file. By editing this file, you can modify the default configuration by overriding values for **systemd** units globally.

For example, to override the default value of the timeout limit, which is set to 90 seconds, use the **DefaultTimeoutStartSec** parameter to input the required value in seconds.

```
DefaultTimeoutStartSec=required value
```

14.2. MANAGING SYSTEM SERVICES WITH SYSTEMCTL

As a system administrator, you can manage system services by using the **systemctl** utility. You can perform various tasks, such as starting, stopping, restarting running services, enabling and disabling services to start at boot, listing available services, and displaying system services statuses.

14.2.1. Listing system services

You can list all currently loaded service units and display the status of all available service units.

Procedure

Use the **systemctl** command to perform any of the following tasks:

- List all currently loaded service units:

```
$ systemctl list-units --type service
UNIT                     LOAD   ACTIVE SUB     DESCRIPTION
abrt-ccpp.service        loaded active exited  Install ABRT coredump hook
abrt-oops.service        loaded active running ABRT kernel log watcher
abrtd.service            loaded active running ABRT Automated Bug Reporting Tool
...
systemd-vconsole-setup.service loaded active exited  Setup Virtual Console
tog-pegasus.service      loaded active running OpenPegasus CIM Server

LOAD   = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, or a generalization of SUB.
SUB    = The low-level unit activation state, values depend on unit type.

46 loaded units listed. Pass --all to see loaded but inactive units, too.
To show all installed unit files use 'systemctl list-unit-files'
```

By default, the **systemctl list-units** command displays only active units. For each service unit file, the command provides an overview of the following parameters:

UNIT

The full name of the service unit

LOAD

The load state of the configuration file

ACTIVE or **SUB**

The current high-level and low-level unit file activation state

DESCRIPTION

A short description of the unit's purpose and functionality

- List **all loaded units regardless of their state** by using the following command with the **--all** or **-a** command line option:

```
$ systemctl list-units --type service --all
```

- List the status (**enabled** or **disabled**) of all available service units:

```
$ systemctl list-unit-files --type service
UNIT FILE                    STATE
abrt-ccpp.service            enabled
abrt-oops.service            enabled
abrtd.service                enabled
...
wpa_supplicant.service       disabled
ypbind.service               disabled

208 unit files listed.
```

For each service unit, this command displays:

UNIT FILE

The full name of the service unit

STATE

The information whether the service unit is enabled or disabled to start automatically during boot

Additional resources

- Displaying system service status

14.2.2. Displaying system service status

You can inspect any service unit to get detailed information and verify the state of the service, whether it is enabled to start during boot or currently running. You can also view services that are ordered to start after or before a particular service unit.

Procedure

Use the **systemctl** command to perform any of the following tasks:

- Display detailed information about a service unit that corresponds to a system service:

```
$ systemctl status <name>.service
```

–

Replace ***<name>*** with the name of the service unit you want to inspect (for example, **gdm**).

This command displays the following information:

- The name of the selected service unit followed by a short description
- One or more fields described in Available service unit information
- The execution of the service unit: if the unit is executed by the **root** user
- The most recent log entries

Table 14.2. Available service unit information

Field	Description
Loaded	Information whether the service unit has been loaded, the absolute path to the unit file, and a note whether the unit is enabled to start during boot.
Active	Information whether the service unit is running followed by a time stamp.
Main PID	The process ID and the name of the corresponding system service.
Status	Additional information about the corresponding system service.
Process	Additional information about related processes.
CGroup	Additional information about related control groups (**cgroups**).

Example 14.1. Displaying service status

The service unit for the GNOME Display Manager is named **gdm.service**. To determine the current status of this service unit, type the following at a shell prompt:

```
# systemctl status gdm.service
gdm.service - GNOME Display Manager
   Loaded: loaded (/usr/lib/systemd/system/gdm.service; enabled)
   Active: active (running) since Thu 2013-10-17 17:31:23 CEST; 5min ago
 Main PID: 1029 (gdm)
   CGroup: /system.slice/gdm.service
           ├─1029 /usr/sbin/gdm
           └─1047 /usr/bin/Xorg :0 -background none -verbose -auth /r...

Oct 17 17:31:23 localhost systemd[1]: Started GNOME Display Manager.
```

- Verify that a particular service unit is running:

```
$ systemctl is-active <name>.service
```

- Determine whether a particular service unit is enabled to start during boot:

```
$ systemctl is-enabled <name>.service
```

> **NOTE**
>
> Both **systemctl is-active** and **systemctl is-enabled** commands return an exit status of **0** if the specified service unit is running or enabled.

- Check what services **systemd** orders to start before the specified service unit

```
# systemctl list-dependencies --after <name>.service
```

For example, to view the list of services ordered to start before **gdm**, enter:

```
# systemctl list-dependencies --after gdm.service
gdm.service
 ├─dbus.socket
 ├─getty@tty1.service
 ├─livesys.service
 ├─plymouth-quit.service
 ├─system.slice
 ├─systemd-journald.socket
 ├─systemd-user-sessions.service
 └─basic.target
[output truncated]
```

- Check what services **systemd** orders to start after the specified service unit:

```
# systemctl list-dependencies --before <name>.service
```

For example, to view the list of services **systemd** orders to start after **gdm**, enter:

```
# systemctl list-dependencies --before gdm.service
gdm.service
 ├─dracut-shutdown.service
 ├─graphical.target
 │ ├─systemd-readahead-done.service
 │ ├─systemd-readahead-done.timer
 │ └─systemd-update-utmp-runlevel.service
 └─shutdown.target
   ├─systemd-reboot.service
   └─final.target
     └─systemd-reboot.service
```

Additional resources

- Listing system services

14.2.3. Starting a system service

You can start system service in the current session by using the **start** command.

Prerequisites

- Root access

Procedure

- Start a system service in the current session:

  ```
  # systemctl start <name>.service
  ```

  Replace ***<name>*** with the name of the service unit you want to start (for example, **httpd.service**).

> **NOTE**
>
> In **systemd**, positive and negative dependencies between services exist. Starting a particular service may require starting one or more other services (**positive dependency**) or stopping one or more services (**negative dependency**).
>
> When you attempt to start a new service, **systemd** resolves all dependencies automatically, without explicit notification to the user. This means that if you are already running a service, and you attempt to start another service with a negative dependency, the first service is automatically stopped.
>
> For example, if you are running the **postfix** service, and you attempt to start the **sendmail** service, **systemd** first automatically stops **postfix**, because these two services are conflicting and cannot run on the same port.

Additional resources

- **systemctl(1)** man page
- Enabling a system service to start at boot
- Displaying system service status

14.2.4. Stopping a system service

If you want to stop a system service in the current session, use the **stop** command.

Prerequisites

- Root access

Procedure

- Stop a system service:

  ```
  # systemctl stop <name>.service
  ```

Replace ***<name>*** with the name of the service unit you want to stop (for example, **bluetooth**).

Additional resources

- **systemctl(1)** man page
- Disabling a system service to start at boot
- Displaying system service status

14.2.5. Restarting a system service

You can restart system service in the current session using the **restart** command to perform the following actions:

- Stop the selected service unit in the current session and immediately start it again.
- Restart a service unit only if the corresponding service is already running.
- Reload configuration of a system service without interrupting its execution.

Prerequisites

- Root access

Procedure

- Restart a system service:

  ```
  # systemctl restart <name>.service
  ```

  Replace ***<name>*** with the name of the service unit you want to restart (for example, **httpd**).

  > **NOTE**
  >
  > If the selected service unit is not running, this command starts it too.

- Optional: Restart a service unit only if the corresponding service is already running:

  ```
  # systemctl try-restart <name>.service
  ```

- Optional: Reload the configuration without interrupting service execution:

  ```
  # systemctl reload <name>.service
  ```

  > **NOTE**
  >
  > System services that do not support this feature, ignore this command. To restart such services, use the **reload-or-restart** and **reload-or-try-restart** commands instead.

Additional resources

- **systemctl** man page
- Displaying system service status

14.2.6. Enabling a system service to start at boot

You can enable a service to start automatically at boot, these changes apply with the next reboot.

Prerequisites

- Root access
- The service you want to enable must not be masked. If you have a masked service, unmask it first:

```
# systemctl unmask <name>.service
```

Procedure

- Enable a service to start at boot:

```
# systemctl enable <name>.service
```

  Replace ***<name>*** with the name of the service unit you want to enable (for example, **httpd**).

- Optional: You can also enable and start a service by using a single command:

```
# systemctl enable --now <name>.service
```

Additional resources

- **systemctl (1)** man page
- Displaying system service status
- Starting a system service

14.2.7. Disabling a system service to start at boot

You can prevent a service unit from starting automatically at boot time. If you disable a service, it will not start at boot, but you can start it manually. You can also mask a service, so that it cannot be started manually. Masking is a way of disabling a service that makes the service permanently unusable until it is unmasked again.

Prerequisites

- Root access

Procedure

- Disable a service to start at boot:

```
# systemctl disable <name>.service
```

Replace ***<name>*** with the name of the service unit you want to disable (for example, **bluetooth**).

- Optional: If you want to make a service permanently unusable, mask the service:

  ```
  # systemctl mask <name>.service
  ```

  This command replaces the **/etc/systemd/system/*name*.service** file with a symbolic link to **/dev/null**, rendering the actual unit file inaccessible to **systemd**.

Additional resources

- **systemctl (1)** man page
- Displaying system service status
- Stopping a system service

14.3. BOOTING INTO A TARGET SYSTEM STATE

As a system administrator, you can control the boot process of your system, and define the state you want your system to boot into. This is called a **systemd** target, and it is a set of **systemd** units that your system starts to reach a certain level of functionality. While working with systemd targets, you can view the default target, select a target at runtime, change the default boot target, boot into emergency or rescue target.

14.3.1. Target unit files

Targets in **systemd** are groups of related units that act as synchronization points during the start of your system. Target unit files, which end with the **.target** file extension, represent the **systemd** targets. The purpose of target units is to group together various **systemd** units through a chain of dependencies.

Consider the following examples:

- The **graphical.target unit** for starting a graphical session, starts system services such as the GNOME Display Manager (**gdm.service**) or Accounts Service (**accounts-daemon.service**), and also activates the **multi-user.target unit**.
- Similarly, the **multi-user.target** unit starts other essential system services such as NetworkManager (**NetworkManager.service**) or D-Bus (**dbus.service**) and activates another target unit named **basic.target**.

You can set the following **systemd** targets as default or current targets:

Table 14.3. Common systemd targets

rescue	unit target that pulls in the base system and spawns a rescue shell
multi-user	unit target for setting up a multi-user system
graphical	unit target for setting up a graphical login screen
emergency	unit target that starts an emergency shell on the main console

Additional resources

- **systemd.special(7)** man page
- **systemd.target(5)** man page

14.3.2. Changing the default target to boot into

When a system starts, **systemd** activates the **default.target** symbolic link, which points to the true target unit. You can find the currently selected default target unit in the **/etc/systemd/system/default.target** file. Each target represents a certain level of functionality and is used for grouping other units. Additionally, target units serve as synchronization points during boot. You can change the default target your system boots into. When you set a default target unit, the current target remains unchanged until the next reboot.

Prerequisites

- Root access

Procedure

1. Determine the current default target unit **systemd** uses to start the system:

```
# systemctl get-default
graphical.target
```

2. List the currently loaded targets:

```
# systemctl list-units --type target
```

3. Configure the system to use a different target unit by default:

```
# systemctl set-default <name>.target
```

   Replace ***<name>*** with the name of the target unit you want to use by default.

```
Example:
# systemctl set-default multi-user.target
Removed /etc/systemd/system/default.target
Created symlink /etc/systemd/system/default.target -> /usr/lib/systemd/system/multi-user.target
```

4. Verify the default target unit:

```
# systemctl get-default
multi-user.target
```

5. Apply the changes by rebooting:

```
# reboot
```

Additional resources

- **systemctl(1)** man page
- **systemd.special(7)** man page
- **bootup(7)** man page

14.3.3. Changing the current target

On a running system, you can change the target unit in the current boot without reboot. If you switch to a different target, **systemd** starts all services and their dependencies that this target requires, and stops all services that the new target does not enable. Isolating a different target affects only the current boot.

Procedure

1. Optional: Determine the current target:

```
# systemctl get-default
graphical.target
```

2. Optional: Display the list of targets you can select:

```
# systemctl list-units --type target
```

> **NOTE**
>
> You can only isolate targets that have the **AllowIsolate=yes** option set in the unit files.

3. Change to a different target unit in the current boot:

```
# systemctl isolate <name>.target
```

Replace *<name>* with the name of the target unit you want to use in the current boot.

```
Example:
# systemctl isolate multi-user.target
```

This command starts the target unit named **multi-user** and all dependent units, and immediately stops all other unit.

Additional resources

- **systemctl(1)** man page

14.3.4. Booting to rescue mode

You can boot to the *rescue mode* that provides a single-user environment for troubleshooting or repair if the system cannot get to a later target, and the regular booting process fails. In rescue mode, the system attempts to mount all local file systems and start certain important system services, but it does not activate network interfaces.

Prerequisites

- Root access

Procedure

- To enter the rescue mode, change the current target in the current session:

```
# systemctl rescue

Broadcast message from root@localhost on pts/0 (Fri 2023-03-24 18:23:15 CEST):

The system is going down to rescue mode NOW!
```

> **NOTE**
>
> This command is similar to **systemctl isolate rescue.target**, but it also sends an informative message to all users that are currently logged into the system.
>
> To prevent **systemd** from sending a message, enter the following command with the **--no-wall** command-line option:
>
> ```
> # systemctl --no-wall rescue
> ```

Troubleshooting steps

If your system is not able to enter the rescue mode, you can boot to *emergency mode*, which provides the most minimal environment possible. In emergency mode, the system mounts the root file system only for reading, does not attempt to mount any other local file systems, does not activate network interfaces, and only starts a few essential services.

14.3.5. Troubleshooting the boot process

As a system administrator, you can select a non-default target at boot time to troubleshoot the boot process. Changing the target at boot time affects only a single boot. You can boot to *emergency mode*, which provides the most minimal environment possible.

Procedure

1. Reboot the system, and interrupt the boot loader menu countdown by pressing any key except the Enter key, which would initiate a normal boot.

2. Move the cursor to the kernel entry that you want to start.

3. Press the E key to edit the current entry.

4. Move to the end of the line that starts with **linux** and press Ctrl+E to jump to the end of the line:

```
linux ($root)/vmlinuz-5.14.0-70.22.1.e19_0.x86_64 root=/dev/mapper/rhel-root ro crash\
kernel=auto resume=/dev/mapper/rhel-swap rd.lvm.lv/swap rhgb quiet
```

5. To choose an alternate boot target, append the **systemd.unit=** parameter to the end of the line that starts with **linux**:

```
linux ($root)/vmlinuz-5.14.0-70.22.1.e19_0.x86_64 root=/dev/mapper/rhel-root ro crash\
kernel=auto resume=/dev/mapper/rhel-swap rd.lvm.lv/swap rhgb quiet
systemd.unit=<name>.target
```

Replace ***<name>*** with the name of the target unit you want to use. For example, **systemd.unit=emergency.target**

6. Press Ctrl+X to boot with these settings.

14.4. SHUTTING DOWN, SUSPENDING, AND HIBERNATING THE SYSTEM

As a system administrator, you can use different power management options to manage power consumption, perform a proper shutdown to ensure that all data is saved, or restart the system to apply changes and updates.

14.4.1. System shutdown

To shut down the system, you can either use the **systemctl** utility directly, or call this utility through the **shutdown** command.

Using the **shutdown** has the following advantages:

- You can schedule a shutdown by using the **time** argument. This also gives users warning that a system shutdown has been scheduled.
- You can cancel the shutdown.

Additional resources

- Overview of the power management commands with systemctl

14.4.2. Scheduling a system shutdown

As a system administrator, you can schedule a delayed shutdown to give users time to save their work and log off the system. Use the **shutdown** command to perform the following operations:

- Shut down the system and power off the machine at a certain time
- Shut down and halt the system without powering off the machine
- Cancel a pending shutdown

Prerequisites

- Root access

Procedure

Use the **shutdown** command to perform any of the following tasks:

- Specify the time at which you want to shut down the system and power off the machine:

```
# shutdown --poweroff hh:mm
```

Where ***hh:mm*** is the time in the 24-hour time notation. To prevent new logins, the **/run/nologin** file is created 5 minutes before system shutdown.

When you use the time argument, you can notify users logged in to the system of the planned shutdown by specifying an optional *wall message*, for example **shutdown --poweroff 13:59 "Attention. The system will shut down at 13:59"**.

- Shut down and halt the system after a delay, without powering off the machine:

```
# shutdown --halt +m
```

Where ***+m*** is the delay time in minutes. You can use the **now** keyword as an alias for **+0**.

- Cancel a pending shutdown:

```
# shutdown -c
```

Additional resources

- **shutdown(8)** manual page
- Shutting down the system using the systemctl command

14.4.3. Shutting down the system using the systemctl command

As a system administrator, you can shut down the system and power off the machine or shut down and halt the system without powering off the machine by using the **systemctl** command.

Prerequisites

- Root access

Procedure

Use the **systemctl** command to perform any of the following tasks:

- Shut down the system and power off the machine:

```
# systemctl poweroff
```

- Shut down and halt the system without powering off the machine:

```
# systemctl halt
```

> **NOTE**
>
> By default, running either of these commands causes **systemd** to send an informative message to all users that are currently logged into the system. To prevent **systemd** from sending this message, run the selected command with the **--no-wall** command line option.

14.4.4. Restarting the system

When you restart the system, **systemd** stops all running programs and services, the system shuts down, and then immediately starts again. Restarting the system can be helpful in the following situations:

- After installing new software or updates
- After making changes to system settings
- When troubleshooting system issues

Prerequisites

- Root access

Procedure

- Restart the system:

```
# systemctl reboot
```

> **NOTE**
>
> By default, when you use this command, **systemd** sends an informative message to all users that are currently logged into the system. To prevent **systemd** from sending this message, run this command with the **--no-wall** option.

14.4.5. Optimizing power consumption by suspending and hibernating the system

As a system administrator, you can manage power consumption, save energy on your systems, and preserve the current state of your system. To do so, apply one of the following modes:

Suspend

Suspending saves the system state in RAM and with the exception of the RAM module, powers off most of the devices in the machine. When you turn the machine back on, the system then restores its state from RAM without having to boot again. Because the system state is saved in RAM and not on the hard disk, restoring the system from suspend mode is significantly faster than from hibernation. However, the suspended system state is also vulnerable to power outages.

Hibernate

Hibernating saves the system state on the hard disk drive and powers off the machine. When you turn the machine back on, the system then restores its state from the saved data without having to boot again. Because the system state is saved on the hard disk and not in RAM, the machine does not have to maintain electrical power to the RAM module. However, as a consequence, restoring the system from hibernation is significantly slower than restoring it from suspend mode.

Hybrid sleep

This combines elements of both hibernation and suspending. The system first saves the current state on the the hard disk drive, and enters a low-power state similar to suspending, which allows the system to resume more quickly. The benefit of hybrid sleep is that if the system loses power during the sleep state, it can still recover the previous state from the saved image on the hard disk, similar to hibernation.

Suspend-then-hibernate

This mode first suspends the system, which results in saving the current system state to RAM and putting the system in a low-power mode. The system hibernates if it remains suspended for a specific period of time that you can define in the **HibernateDelaySec** parameter. Hibernation saves the system state to the hard disk drive and shuts down the system completely. The suspend-then-hibernate mode provides the benefit of conserving battery power while you are still able to quickly resume work. Additionally, this mode ensures that your data is saved in case of a power failure.

Prerequisites

- Root access

Procedure

Choose the appropriate method for power saving:

- Suspend the system:

```
# systemctl suspend
```

- Hibernate the system:

```
# systemctl hibernate
```

- Hibernate and suspend the system:

```
# systemctl hybrid-sleep
```

- Suspend and then hibernate the system:

```
# systemctl suspend-then-hibernate
```

14.4.6. Overview of the power management commands with systemctl

You can use the following list of the **systemctl** commands to control the power management of your system.

Table 14.4. Overview of the systemctl power management commands

systemctl command	Description
systemctl halt	Halts the system.
systemctl poweroff	Powers off the system.
systemctl reboot	Restarts the system.
systemctl suspend	Suspends the system.
systemctl hibernate	Hibernates the system.
systemctl hybrid-sleep	Hibernates and suspends the system.

14.4.7. Changing how your system behaves when you press the power button

When you press the power button on your computer, it shuts down the system by default. You can customize this behavior according to your preferences.

Prerequisites

- Administrative access.

Procedure

1. Open the **/etc/systemd/logind.conf** configuration file.
2. Look for the line that says **HandlePowerKey=poweroff**.
3. If the line starts with the **#** symbol, remove it to enable the setting.
4. Replace **poweroff** with one of the following options:
    - **poweroff** to shut down the computer.
    - **reboot** for a system reboot.
    - **halt** to initiate a system halt.
    - **kexec** for a kexec-based reboot.
    - **suspend** to suspend the system.
    - **hibernate** to initiate system hibernation.
    - **ignore** to do nothing.

    For example, to reboot the system upon pressing the power button, use this setting:

    ```
    HandlePowerKey=reboot
    ```

5. Save your changes and close the editor.

CHAPTER 15. CONFIGURING TIME SYNCHRONIZATION

Accurate timekeeping in an IT environment is important. A consistent time across all network devices improves the traceability of log files and certain protocols rely on synchronized clocks. For example, Kerberos uses time stamps to prevent replay attacks.

15.1. USING THE CHRONY SUITE TO CONFIGURE NTP

Accurate timekeeping is important for several reasons in IT. In networking for example, accurate time stamps in packets and logs are required. In Linux systems, the **NTP** protocol is implemented by a daemon running in user space.

The user space daemon updates the system clock running in the kernel. The system clock can keep time by using various clock sources. Usually, the *Time Stamp Counter* (**TSC**) is used. The TSC is a CPU register which counts the number of cycles since it was last reset. It is very fast, has a high resolution, and there are no interruptions.

Starting with Red Hat Enterprise Linux 8, the **NTP** protocol is implemented by the **chronyd** daemon, available from the repositories in the **chrony** package.

The following sections describe how to use the **chrony** suite to configure NTP.

15.1.1. Introduction to chrony suite

chrony is an implementation of the **Network Time Protocol (NTP)**. You can use **chrony**:

- To synchronize the system clock with **NTP** servers
- To synchronize the system clock with a reference clock, for example a GPS receiver
- To synchronize the system clock with a manual time input
- As an **NTPv4(RFC 5905)** server or peer to provide a time service to other computers in the network

chrony performs well in a wide range of conditions, including intermittent network connections, heavily congested networks, changing temperatures (ordinary computer clocks are sensitive to temperature), and systems that do not run continuously, or run on a virtual machine.

Typical accuracy between two machines synchronized over the Internet is within a few milliseconds, and for machines on a LAN within tens of microseconds. Hardware timestamping or a hardware reference clock may improve accuracy between two machines synchronized to a sub-microsecond level.

chrony consists of **chronyd**, a daemon that runs in user space, and **chronyc**, a command line program which can be used to monitor the performance of **chronyd** and to change various operating parameters when it is running.

The **chrony** daemon, **chronyd**, can be monitored and controlled by the command line utility **chronyc**. This utility provides a command prompt which allows entering a number of commands to query the current state of **chronyd** and make changes to its configuration. By default, **chronyd** accepts only commands from a local instance of **chronyc**, but it can be configured to accept monitoring commands also from remote hosts. The remote access should be restricted.

15.1.2. Using chronyc to control chronyd

You can control **chronyd** by using the **chronyc** command line utility.

Procedure

1. To make changes to the local instance of **chronyd** using the command line utility **chronyc** in interactive mode, enter the following command as **root**:

```
# chronyc
```

   chronyc must run as **root** if some of the restricted commands are to be used.

   The **chronyc** command prompt will be displayed as follows:

```
chronyc>
```

2. To list all of the commands, type **help**.

3. Alternatively, the utility can also be invoked in non-interactive command mode if called together with a command as follows:

```
chronyc command
```

NOTE

Changes made using **chronyc** are not permanent, they will be lost after a **chronyd** restart. For permanent changes, modify **/etc/chrony.conf**.

15.1.3. Migrating to chrony

In Red Hat Enterprise Linux 7, users could choose between **ntp** and **chrony** to ensure accurate timekeeping. For differences between **ntp** and **chrony**, **ntpd** and **chronyd**, see Differences between ntpd and chronyd.

Starting with Red Hat Enterprise Linux 8, **ntp** is no longer supported. **chrony** is enabled by default. For this reason, you might need to migrate from **ntp** to **chrony**.

Migrating from **ntp** to **chrony** is straightforward in most cases. The corresponding names of the programs, configuration files and services are:

Table 15.1. Corresponding names of the programs, configuration files and services when migrating from ntp to chrony

ntp name	chrony name
/etc/ntp.conf	/etc/chrony.conf
/etc/ntp/keys	/etc/chrony.keys
ntpd	chronyd
ntpq	chronyc

ntp name	chrony name
ntpd.service	chronyd.service
ntp-wait.service	chrony-wait.service

The **ntpdate** and **sntp** utilities, which are included in the **ntp** distribution, can be replaced with **chronyd** using the **-q** option or the **-t** option. The configuration can be specified on the command line to avoid reading **/etc/chrony.conf**. For example, instead of running **ntpdate ntp.example.com**, **chronyd** could be started as:

```
# chronyd -q 'server ntp.example.com iburst'
2018-05-18T12:37:43Z chronyd version 3.3 starting (+CMDMON +NTP +REFCLOCK +RTC
+PRIVDROP +SCFILTER +SIGND +ASYNCDNS +SECHASH +IPV6 +DEBUG)
2018-05-18T12:37:43Z Initial frequency -2.630 ppm
2018-05-18T12:37:48Z System clock wrong by 0.003159 seconds (step)
2018-05-18T12:37:48Z chronyd exiting
```

The **ntpstat** utility, which was previously included in the **ntp** package and supported only **ntpd**, now supports both **ntpd** and **chronyd**. It is available in the **ntpstat** package.

15.1.3.1. Migration script

A Python script called **ntp2chrony.py** is included in the documentation of the **chrony** package (**/usr/share/doc/chrony**). The script automatically converts an existing **ntp** configuration to **chrony**. It supports the most common directives and options in the **ntp.conf** file. Any lines that are ignored in the conversion are included as comments in the generated **chrony.conf** file for review. Keys that are specified in the **ntp** key file, but are not marked as trusted keys in **ntp.conf** are included in the generated **chrony.keys** file as comments.

By default, the script does not overwrite any files. If **/etc/chrony.conf** or **/etc/chrony.keys** already exist, the **-b** option can be used to rename the file as a backup. The script supports other options. The **--help** option prints all supported options.

An example of an invocation of the script with the default **ntp.conf** provided in the **ntp** package is:

```
# python3 /usr/share/doc/chrony/ntp2chrony.py -b -v
Reading /etc/ntp.conf
Reading /etc/ntp/crypto/pw
Reading /etc/ntp/keys
Writing /etc/chrony.conf
Writing /etc/chrony.keys
```

The only directive ignored in this case is **disable monitor**, which has a chrony equivalent in the **noclientlog** directive, but it was included in the default **ntp.conf** only to mitigate an amplification attack.

The generated **chrony.conf** file typically includes a number of **allow** directives corresponding to the restrict lines in **ntp.conf**. If you do not want to run **chronyd** as an **NTP** server, remove all **allow** directives from **chrony.conf**.

15.2. USING CHRONY

The following sections describe how to install, start, and stop **chronyd**, and how to check if **chrony** is synchronized. Sections also describe how to manually adjust System Clock.

15.2.1. Managing chrony

The following procedure describes how to install, start, stop, and check the status of **chronyd**.

Procedure

1. The **chrony** suite is installed by default on Red Hat Enterprise Linux. To ensure that it is, run the following command as **root**:

```
# yum install chrony
```

The default location for the **chrony** daemon is **/usr/sbin/chronyd**. The command line utility will be installed to **/usr/bin/chronyc**.

2. To check the status of **chronyd**, issue the following command:

```
$ systemctl status chronyd
chronyd.service - NTP client/server
  Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled)
  Active: active (running) since Wed 2013-06-12 22:23:16 CEST; 11h ago
```

3. To start **chronyd**, issue the following command as **root**:

```
# systemctl start chronyd
```

To ensure **chronyd** starts automatically at system start, issue the following command as **root**:

```
# systemctl enable chronyd
```

4. To stop **chronyd**, issue the following command as **root**:

```
# systemctl stop chronyd
```

To prevent **chronyd** from starting automatically at system start, issue the following command as **root**:

```
# systemctl disable chronyd
```

15.2.2. Checking if chrony is synchronized

The following procedure describes how to check if **chrony** is synchronized with the use of the **tracking**, **sources**, and **sourcestats** commands.

Procedure

1. To check **chrony** tracking, issue the following command:

```
$ chronyc tracking
Reference ID    : CB00710F (ntp-server.example.net)
```

```
Stratum         : 3
Ref time (UTC)  : Fri Jan 27 09:49:17 2017
System time     :  0.000006523 seconds slow of NTP time
Last offset     : -0.000006747 seconds
RMS offset      : 0.000035822 seconds
Frequency       : 3.225 ppm slow
Residual freq   : 0.000 ppm
Skew            : 0.129 ppm
Root delay      : 0.013639022 seconds
Root dispersion : 0.001100737 seconds
Update interval : 64.2 seconds
Leap status     : Normal
```

2. The sources command displays information about the current time sources that **chronyd** is accessing. To check **chrony** sources, issue the following command:

```
$ chronyc sources
 210 Number of sources = 3
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
====
#* GPS0                          0   4   377    11   -479ns[ -621ns] /-  134ns
^? a.b.c                         2   6   377    23   -923us[ -924us] +/-   43ms
^ d.e.f                          1   6   377    21  -2629us[-2619us] +/-   86ms
```

You can specify the optional **-v** argument to print more verbose information. In this case, extra caption lines are shown as a reminder of the meanings of the columns.

3. The **sourcestats** command displays information about the drift rate and offset estimation process for each of the sources currently being examined by **chronyd**. To check **chrony** source statistics, issue the following command:

```
$  chronyc sourcestats
210 Number of sources = 1
Name/IP Address            NP  NR  Span  Frequency  Freq Skew  Offset  Std Dev
===============================================================================
====
abc.def.ghi                11   5   46m     -0.001      0.045      1us    25us
```

The optional argument **-v** can be specified, meaning verbose. In this case, extra caption lines are shown as a reminder of the meanings of the columns.

Additional resources

- **chronyc(1)** man page

15.2.3. Manually adjusting the System Clock

The following procedure describes how to manually adjust the System Clock.

Procedure

1. To step the system clock immediately, bypassing any adjustments in progress by slewing, issue the following command as **root**:

```
# chronyc makestep
```

If the **rtcfile** directive is used, the real-time clock should not be manually adjusted. Random adjustments would interfere with **chrony**'s need to measure the rate at which the real-time clock drifts.

15.2.4. Disabling a chrony dispatcher script

The **chrony** dispatcher script manages the online and offline state of the NTP servers. As a system administrator, you can disable the dispatcher script to keep **chronyd** polling the servers constantly.

If you enable NetworkManager on your system to manage networking configuration, the NetworkManager executes the **chrony** dispatcher script during interface reconfiguration, stop or start operations. However, if you configure certain interfaces or routes outside of NetworkManager, you can encounter the following situation:

1. The dispatcher script might run when no route to the NTP servers exists, causing the NTP servers to switch to the offline state.

2. If you establish the route later, the script does not run again by default, and the NTP servers remain in the offline state.

To ensure that **chronyd** can synchronize with your NTP servers, which have separately managed interfaces, disable the dispatcher script.

Prerequisites

- You installed NetworkManager on your system and enabled it.
- Root access

Procedure

1. To disable the **chrony** dispatcher script, edit the **/etc/NetworkManager/dispatcher.d/20-chrony-onoffline** file as follows:

```
#!/bin/sh
exit 0
```

> **NOTE**
>
> When you upgrade or reinstall the **chrony** package, the packaged version of the dispatcher script replaces your modified dispatcher script.

15.2.5. Setting up chrony for a system in an isolated network

For a network that is never connected to the Internet, one computer is selected to be the primary timeserver. The other computers are either direct clients of the server, or clients of clients. On the server, the drift file must be manually set with the average rate of drift of the system clock. If the server is rebooted, it will obtain the time from surrounding systems and calculate an average to set its system clock. Thereafter it resumes applying adjustments based on the drift file. The drift file will be updated automatically when the **settime** command is used.

The following procedure describes how to set up **chrony** for a system in an isolated network.

Procedure

1. On the system selected to be the server, using a text editor running as **root**, edit **/etc/chrony.conf** as follows:

```
driftfile /var/lib/chrony/drift
commandkey 1
keyfile /etc/chrony.keys
initstepslew 10 client1 client3 client6
local stratum 8
manual
allow 192.0.2.0
```

   Where **192.0.2.0** is the network or subnet address from which the clients are allowed to connect.

2. On the systems selected to be direct clients of the server, using a text editor running as **root**, edit the **/etc/chrony.conf** as follows:

```
server ntp1.example.net
driftfile /var/lib/chrony/drift
logdir /var/log/chrony
log measurements statistics tracking
keyfile /etc/chrony.keys
commandkey 24
local stratum 10
initstepslew 20 ntp1.example.net
allow 192.0.2.123
```

   Where **192.0.2.123** is the address of the server, and **ntp1.example.net** is the host name of the server. Clients with this configuration will resynchronize with the server if it restarts.

On the client systems which are not to be direct clients of the server, the **/etc/chrony.conf** file should be the same except that the **local** and **allow** directives should be omitted.

In an isolated network, you can also use the **local** directive that enables a local reference mode, which allows **chronyd** operating as an **NTP** server to appear synchronized to real time, even when it was never synchronized or the last update of the clock happened a long time ago.

To allow multiple servers in the network to use the same local configuration and to be synchronized to one another, without confusing clients that poll more than one server, use the **orphan** option of the **local** directive which enables the orphan mode. Each server needs to be configured to poll all other servers with **local**. This ensures that only the server with the smallest reference ID has the local reference active and other servers are synchronized to it. When the server fails, another one will take over.

15.2.6. Configuring remote monitoring access

chronyc can access **chronyd** in two ways:

- Internet Protocol, IPv4 or IPv6.
- Unix domain socket, which is accessible locally by the **root** or **chrony** user.

By default, **chronyc** connects to the Unix domain socket. The default path is **/var/run/chrony/chronyd.sock**. If this connection fails, which can happen for example when **chronyc** is running under a non-root user, **chronyc** tries to connect to 127.0.0.1 and then ::1.

Only the following monitoring commands, which do not affect the behavior of **chronyd**, are allowed from the network:

- activity
- manual list
- rtcdata
- smoothing
- sources
- sourcestats
- tracking
- waitsync

The set of hosts from which **chronyd** accepts these commands can be configured with the **cmdallow** directive in the configuration file of **chronyd**, or the **cmdallow** command in **chronyc**. By default, the commands are accepted only from localhost (127.0.0.1 or ::1).

All other commands are allowed only through the Unix domain socket. When sent over the network, **chronyd** responds with a **Not authorised** error, even if it is from localhost.

The following procedure describes how to access chronyd remotely with **chronyc**.

Procedure

1. Allow access from both IPv4 and IPv6 addresses by adding the following to the **/etc/chrony.conf** file:

   ```
   bindcmdaddress 0.0.0.0
   ```

   or

   ```
   bindcmdaddress ::
   ```

2. Allow commands from the remote IP address, network, or subnet by using the **cmdallow** directive.
   Add the following content to the **/etc/chrony.conf** file:

   ```
   cmdallow 192.168.1.0/24
   ```

3. Open port 323 in the firewall to connect from a remote system:

   ```
   # firewall-cmd --zone=public --add-port=323/udp
   ```

   Optionally, you can open port 323 permanently using the **--permanent** option:

   ```
   # firewall-cmd --permanent --zone=public --add-port=323/udp
   ```

4. If you opened port 323 permanently, reload the firewall configuration:

```
# firewall-cmd --reload
```

Additional resources

- **chrony.conf(5)** man page

15.2.7. Managing time synchronization using RHEL system roles

You can manage time synchronization on multiple target machines using the **timesync** role. The **timesync** role installs and configures an NTP or PTP implementation to operate as an NTP or PTP client to synchronize the system clock.

Note that using the **timesync** role also facilitates migration to chrony, because you can use the same playbook on all versions of Red Hat Enterprise Linux starting with RHEL 6 regardless of whether the system uses **ntp** or **chrony** to implement the NTP protocol.

> **WARNING**
>
> The **timesync** role replaces the configuration of the given or detected provider service on the managed host. Previous settings are lost, even if they are not specified in the role variables. The only preserved setting is the choice of provider if the **timesync_ntp_provider** variable is not defined.

The following example shows how to apply the **timesync** role in a situation with just one pool of servers.

Example 15.1. An example playbook applying the timesync role for a single pool of servers

```
---
- hosts: timesync-test
  vars:
    timesync_ntp_servers:
      - hostname: 2.rhel.pool.ntp.org
        pool: yes
        iburst: yes
  roles:
    - rhel-system-roles.timesync
```

For a detailed reference on **timesync** role variables, install the **rhel-system-roles** package, and see the **README.md** or **README.html** files in the **/usr/share/doc/rhel-system-roles/timesync** directory.

Additional resources

- Preparing a control node and managed nodes to use RHEL system roles

15.2.8. Additional resources

- **chronyc(1)** man page

- **chronyd(8)** man page
- Frequently Asked Questions

15.3. CHRONY WITH HW TIMESTAMPING

Hardware timestamping is a feature supported in some Network Interface Controller (NICs) which provides accurate timestamping of incoming and outgoing packets. **NTP** timestamps are usually created by the kernel and **chronyd** with the use of the system clock. However, when HW timestamping is enabled, the NIC uses its own clock to generate the timestamps when packets are entering or leaving the link layer or the physical layer. When used with **NTP**, hardware timestamping can significantly improve the accuracy of synchronization. For best accuracy, both **NTP** servers and **NTP** clients need to use hardware timestamping. Under ideal conditions, a sub-microsecond accuracy may be possible.

Another protocol for time synchronization that uses hardware timestamping is **PTP**.

Unlike **NTP**, **PTP** relies on assistance in network switches and routers. If you want to reach the best accuracy of synchronization, use **PTP** on networks that have switches and routers with **PTP** support, and prefer **NTP** on networks that do not have such switches and routers.

The following sections describe how to:

- Verify support for hardware timestamping
- Enable hardware timestamping
- Configure client polling interval
- Enable interleaved mode
- Configure server for large number of clients
- Verify hardware timestamping
- Configure PTP-NTP bridge

15.3.1. Verifying support for hardware timestamping

To verify that hardware timestamping with **NTP** is supported by an interface, use the **ethtool -T** command. An interface can be used for hardware timestamping with **NTP** if **ethtool** lists the **SOF_TIMESTAMPING_TX_HARDWARE** and **SOF_TIMESTAMPING_TX_SOFTWARE** capabilities and also the **HWTSTAMP_FILTER_ALL** filter mode.

Example 15.2. Verifying support for hardware timestamping on a specific interface

```
# ethtool -T eth0
```

Output:

```
Timestamping parameters for eth0:
Capabilities:
        hardware-transmit     (SOF_TIMESTAMPING_TX_HARDWARE)
        software-transmit     (SOF_TIMESTAMPING_TX_SOFTWARE)
        hardware-receive      (SOF_TIMESTAMPING_RX_HARDWARE)
        software-receive      (SOF_TIMESTAMPING_RX_SOFTWARE)
```

```
        software-system-clock (SOF_TIMESTAMPING_SOFTWARE)
        hardware-raw-clock    (SOF_TIMESTAMPING_RAW_HARDWARE)
PTP Hardware Clock: 0
Hardware Transmit Timestamp Modes:
        off                   (HWTSTAMP_TX_OFF)
        on                    (HWTSTAMP_TX_ON)
Hardware Receive Filter Modes:
        none                  (HWTSTAMP_FILTER_NONE)
        all                   (HWTSTAMP_FILTER_ALL)
        ptpv1-l4-sync         (HWTSTAMP_FILTER_PTP_V1_L4_SYNC)
        ptpv1-l4-delay-req    (HWTSTAMP_FILTER_PTP_V1_L4_DELAY_REQ)
        ptpv2-l4-sync         (HWTSTAMP_FILTER_PTP_V2_L4_SYNC)
        ptpv2-l4-delay-req    (HWTSTAMP_FILTER_PTP_V2_L4_DELAY_REQ)
        ptpv2-l2-sync         (HWTSTAMP_FILTER_PTP_V2_L2_SYNC)
        ptpv2-l2-delay-req    (HWTSTAMP_FILTER_PTP_V2_L2_DELAY_REQ)
        ptpv2-event           (HWTSTAMP_FILTER_PTP_V2_EVENT)
        ptpv2-sync            (HWTSTAMP_FILTER_PTP_V2_SYNC)
        ptpv2-delay-req       (HWTSTAMP_FILTER_PTP_V2_DELAY_REQ)
```

15.3.2. Enabling hardware timestamping

To enable hardware timestamping, use the **hwtimestamp** directive in the **/etc/chrony.conf** file. The directive can either specify a single interface, or a wildcard character can be used to enable hardware timestamping on all interfaces that support it. Use the wildcard specification in case that no other application, like **ptp4l** from the **linuxptp** package, is using hardware timestamping on an interface. Multiple **hwtimestamp** directives are allowed in the chrony configuration file.

Example 15.3. Enabling hardware timestamping by using the hwtimestamp directive

```
hwtimestamp eth0
hwtimestamp eth1
hwtimestamp *
```

15.3.3. Configuring client polling interval

The default range of a polling interval (64-1024 seconds) is recommended for servers on the Internet. For local servers and hardware timestamping, a shorter polling interval needs to be configured in order to minimize offset of the system clock.

The following directive in **/etc/chrony.conf** specifies a local **NTP** server using one second polling interval:

```
server ntp.local minpoll 0 maxpoll 0
```

15.3.4. Enabling interleaved mode

NTP servers that are not hardware **NTP** appliances, but rather general purpose computers running a software **NTP** implementation, like **chrony**, will get a hardware transmit timestamp only after sending a packet. This behavior prevents the server from saving the timestamp in the packet to which it corresponds. In order to enable **NTP** clients receiving transmit timestamps that were generated after the transmission, configure the clients to use the **NTP** interleaved mode by adding the **xleave** option to the server directive in **/etc/chrony.conf**:

```
server ntp.local minpoll 0 maxpoll 0 xleave
```

15.3.5. Configuring server for large number of clients

The default server configuration allows a few thousands of clients at most to use the interleaved mode concurrently. To configure the server for a larger number of clients, increase the **clientloglimit** directive in **/etc/chrony.conf**. This directive specifies the maximum size of memory allocated for logging of clients' access on the server:

```
clientloglimit 100000000
```

15.3.6. Verifying hardware timestamping

To verify that the interface has successfully enabled hardware timestamping, check the system log. The log should contain a message from **chronyd** for each interface with successfully enabled hardware timestamping.

Example 15.4. Log messages for interfaces with enabled hardware timestamping

```
chronyd[4081]: Enabled HW timestamping on eth0
chronyd[4081]: Enabled HW timestamping on eth1
```

When **chronyd** is configured as an **NTP** client or peer, you can have the transmit and receive timestamping modes and the interleaved mode reported for each **NTP** source by the **chronyc ntpdata** command:

Example 15.5. Reporting the transmit, receive timestamping and interleaved mode for each NTP source

```
# chronyc ntpdata
```

Output:

```
Remote address  : 203.0.113.15 (CB00710F)
Remote port     : 123
Local address   : 203.0.113.74 (CB00714A)
Leap status     : Normal
Version         : 4
Mode            : Server
Stratum         : 1
Poll interval   : 0 (1 seconds)
Precision       : -24 (0.000000060 seconds)
Root delay      : 0.000015 seconds
Root dispersion : 0.000015 seconds
Reference ID    : 47505300 (GPS)
Reference time  : Wed May 03 13:47:45 2017
Offset          : -0.000000134 seconds
```

```
Peer delay      : 0.000005396 seconds
Peer dispersion : 0.000002329 seconds
Response time   : 0.000152073 seconds
Jitter asymmetry: +0.00
NTP tests       : 111 111 1111
Interleaved     : Yes
Authenticated   : No
TX timestamping : Hardware
RX timestamping : Hardware
Total TX        : 27
Total RX        : 27
Total valid RX  : 27
```

Example 15.6. Reporting the stability of NTP measurements

```
# chronyc sourcestats
```

With hardware timestamping enabled, stability of **NTP** measurements should be in tens or hundreds of nanoseconds, under normal load. This stability is reported in the **Std Dev** column of the output of the **chronyc sourcestats** command:

Output:

```
210 Number of sources = 1
Name/IP Address            NP  NR  Span  Frequency  Freq Skew  Offset  Std Dev
ntp.local                  12   7    11     +0.000      0.019    +0ns    49ns
```

15.3.7. Configuring PTP-NTP bridge

If a highly accurate Precision Time Protocol (**PTP**) primary timeserver is available in a network that does not have switches or routers with **PTP** support, a computer may be dedicated to operate as a **PTP** client and a stratum-1 **NTP** server. Such a computer needs to have two or more network interfaces, and be close to the primary timeserver or have a direct connection to it. This will ensure highly accurate synchronization in the network.

Configure the **ptp4l** and **phc2sys** programs from the **linuxptp** packages to use one interface to synchronize the system clock using **PTP**.

Configure **chronyd** to provide the system time using the other interface:

Example 15.7. Configuring chronyd to provide the system time using the other interface

```
bindaddress 203.0.113.74
hwtimestamp eth1
local stratum 1
```

15.4. ACHIEVING SOME SETTINGS PREVIOUSLY SUPPORTED BY NTP IN CHRONY

Some settings that were in previous major version of Red Hat Enterprise Linux supported by **ntp**, are not supported by **chrony**. The following sections list such settings, and describe ways to achieve them on a system with **chrony**.

15.4.1. Monitoring by ntpq and ntpdc

chronyd cannot be monitored by the **ntpq** and **ntpdc** utilities from the **ntp** distribution, because **chrony** does not support the **NTP** modes 6 and 7. It supports a different protocol and **chronyc** is the client implementation. For more information, see the **chronyc(1)** man page.

To monitor the status of the system clock sychronized by **chronyd**, you can:

- Use the tracking command
- Use the **ntpstat** utility, which supports **chrony** and provides a similar output as it used to with **ntpd**

Example 15.8. Using the tracking command

```
$ chronyc -n tracking
Reference ID    : 0A051B0A (10.5.27.10)
Stratum         : 2
Ref time (UTC)  : Thu Mar 08 15:46:20 2018
System time     : 0.000000338 seconds slow of NTP time
Last offset     : +0.000339408 seconds
RMS offset      : 0.000339408 seconds
Frequency       : 2.968 ppm slow
Residual freq   : +0.001 ppm
Skew            : 3.336 ppm
Root delay      : 0.157559142 seconds
Root dispersion : 0.001339232 seconds
Update interval : 64.5 seconds
Leap status     : Normal
```

Example 15.9. Using the ntpstat utility

```
$ ntpstat
synchronised to NTP server (10.5.27.10) at stratum 2
   time correct to within 80 ms
   polling server every 64 s
```

15.4.2. Using authentication mechanism based on public key cryptography

In Red Hat Enterprise Linux 7, **ntp** supported **Autokey**, which is an authentication mechanism based on public key cryptography.

In Red Hat Enterprise Linux 8, **chronyd** supports Network Time Security (NTS), a modern secure authentication mechanism, instead of **Autokey**. For more information, see Overview of Network Time Security (NTS) in chrony.

15.4.3. Using ephemeral symmetric associations

In Red Hat Enterprise Linux 7, **ntpd** supported ephemeral symmetric associations, which can be mobilized by packets from peers which are not specified in the **ntp.conf** configuration file. In Red Hat Enterprise Linux 8, **chronyd** needs all peers to be specified in **chrony.conf**. Ephemeral symmetric associations are not supported.

Note that using the client/server mode enabled by the **server** or **pool** directive is more secure compared to the symmetric mode enabled by the **peer** directive.

15.4.4. multicast/broadcast client

Red Hat Enterprise Linux 7 supported the broadcast/multicast **NTP** mode, which simplifies configuration of clients. With this mode, clients can be configured to just listen for packets sent to a multicast/broadcast address instead of listening for specific names or addresses of individual servers, which may change over time.

In Red Hat Enterprise Linux 8, **chronyd** does not support the broadcast/multicast mode. The main reason is that it is less accurate and less secure than the ordinary client/server and symmetric modes.

There are several options of migration from an **NTP** broadcast/multicast setup:

- Configure DNS to translate a single name, such as ntp.example.com, to multiple addresses of different servers
  Clients can have a static configuration using only a single pool directive to synchronize with multiple servers. If a server from the pool becomes unreacheable, or otherwise unsuitable for synchronization, the clients automatically replace it with another server from the pool.

- Distribute the list of **NTP** servers over DHCP
  When NetworkManager gets a list of **NTP** servers from the DHCP server, **chronyd** is automatically configured to use them. This feature can be disabled by adding **PEERNTP=no** to the **/etc/sysconfig/network** file.

- Use the **Precision Time Protocol (PTP)**
  This option is suitable mainly for environments where servers change frequently, or if a larger group of clients needs to be able to synchronize to each other without having a designated server.

  PTP was designed for multicast messaging and works similarly to the **NTP** broadcast mode. A **PTP** implementation is available in the **linuxptp** package.

  PTP normally requires hardware timestamping and support in network switches to perform well. However, **PTP** is expected to work better than **NTP** in the broadcast mode even with software timestamping and no support in network switches.

  In networks with very large number of **PTP** clients in one communication path, it is recommended to configure the **PTP** clients with the **hybrid_e2e** option to reduce the amount of network traffic generated by the clients. You can configure a computer running **chronyd** as an **NTP** client, and possibly **NTP** server, to operate also as a primary **PTP** timeserver to distribute synchronized time to a large number of computers using multicast messaging.

15.5. OVERVIEW OF NETWORK TIME SECURITY (NTS) IN CHRONY

Network Time Security (NTS) is an authentication mechanism for Network Time Protocol (NTP), designed to scale substantial clients. It verifies that the packets received from the server machines are unaltered while moving to the client machine. Network Time Security (NTS) includes a Key Establishment (NTS-KE) protocol that automatically creates the encryption keys used between the server and its clients.

> **WARNING**
>
> NTS is not compatible with the FIPS and OSPP profile. When you enable the FIPS and OSPP profile, **chronyd** that is configured with NTS can abort with a fatal message. You can disable the OSPP profile and FIPS mode for **chronyd** service by adding the **GNUTLS_FORCE_FIPS_MODE=0** to the **/etc/sysconfig/chronyd** file.

15.5.1. Enabling Network Time Security (NTS) in the client configuration file

By default, Network Time Security (NTS) is not enabled. You can enable NTS in the **/etc/chrony.conf**. For that, perform the following steps:

Prerequisites

- Server with the NTS support

Procedure

In the client configuration file:

1. Specify the server with the **nts** option in addition to the recommended **iburst** option.

   ```
   For example:
   server time.example.com iburst nts
   server nts.netnod.se iburst nts
   server ptbtime1.ptb.de iburst nts
   ```

2. To avoid repeating the Network Time Security-Key Establishment (NTS-KE) session during system boot, add the following line to **chrony.conf**, if it is not present:

   ```
   ntsdumpdir /var/lib/chrony
   ```

3. Add the following line to **/etc/sysconfig/network** to disable synchronization with Network Time Protocol (NTP) servers provided by **DHCP**:

   ```
   PEERNTP=no
   ```

4. Save your changes.
5. Restart the **chronyd** service:

   ```
   systemctl restart chronyd
   ```

Verification

- Verify if the **NTS** keys were successfully established:

  ```
  # chronyc -N authdata

  Name/IP address  Mode KeyID Type KLen Last Atmp  NAK Cook CLen
  ```

```
=========================================================================
time.example.com  NTS    1   15  256  33m    0    0    8  100
nts.sth1.ntp.se   NTS    1   15  256  33m    0    0    8  100
nts.sth2.ntp.se   NTS    1   15  256  33m    0    0    8  100
```

The **KeyID**, **Type**, and **KLen** should have non-zero values. If the value is zero, check the system log for error messages from **chronyd**.

- Verify the client is making NTP measurements:

```
# chronyc -N sources

MS Name/IP address Stratum Poll Reach LastRx Last sample
=================================================================
time.example.com   3       6   377    45   +355us[ +375us] +/-   11ms
nts.sth1.ntp.se    1       6   377    44   +237us[ +237us] +/-   23ms
nts.sth2.ntp.se    1       6   377    44   -170us[ -170us] +/-   22ms
```

The **Reach** column should have a non-zero value; ideally 377. If the value rarely gets 377 or never gets to 377, it indicates that NTP requests or responses are getting lost in the network.

Additional resources

- **chrony.conf(5)** man page

15.5.2. Enabling Network Time Security (NTS) on the server

If you run your own Network Time Protocol (NTP) server, you can enable the server Network Time Security (NTS) support to facilitate its clients to synchronize securely.

If the NTP server is a client of other servers, that is, it is not a Stratum 1 server, it should use NTS or symmetric key for its synchronization.

Prerequisites

- Server private key in **PEM** format
- Server certificate with required intermediate certificates in **PEM** format

Procedure

1. Specify the private key and the certificate file in **chrony.conf**. For example:

```
ntsserverkey /etc/pki/tls/private/<ntp-server.example.net>.key
ntsservercert /etc/pki/tls/certs/<ntp-server.example.net>.crt
```

2. Ensure that both the key and certificate files are readable by the chrony system user, by setting the group ownership. For example:

```
# chown :chrony /etc/pki/tls//<ntp-server.example.net>.
```

3. Ensure the **ntsdumpdir /var/lib/chrony** directive is present in the **chrony.conf**.

4. Restart the **chronyd** service:

```
# systemctl restart chronyd
```

IMPORTANT

If the server has a firewall, it needs to allow both the **UDP 123** and **TCP 4460** ports for NTP and Network Time Security-Key Establishment (NTS-KE).

Verification

- Perform a quick test from a client machine with the following command:

```
$ chronyd -Q -t 3 'server

ntp-server.example.net iburst nts maxsamples 1'
2021-09-15T13:45:26Z chronyd version 4.1 starting (+CMDMON +NTP +REFCLOCK +RTC
+PRIVDROP +SCFILTER +SIGND +ASYNCDNS +NTS +SECHASH +IPV6 +DEBUG)
2021-09-15T13:45:26Z Disabled control of system clock
2021-09-15T13:45:28Z System clock wrong by 0.002205 seconds (ignored)
2021-09-15T13:45:28Z chronyd exiting
```

The **System clock wrong** message indicates the NTP server is accepting NTS-KE connections and responding with NTS-protected NTP messages.

- Verify the NTS-KE connections and authenticated NTP packets observed on the server:

```
# chronyc serverstats

NTP packets received       : 7
NTP packets dropped        : 0
Command packets received   : 22
Command packets dropped    : 0
Client log records dropped : 0
NTS-KE connections accepted: 1
NTS-KE connections dropped : 0
Authenticated NTP packets: 7
```

If the value of the **NTS-KE connections accepted** and **Authenticated NTP packets** field is a non-zero value, it means that at least one client was able to connect to the NTS-KE port and send an authenticated NTP request.

CHAPTER 16. USING LANGPACKS

Langpacks are meta-packages which install extra add-on packages containing translations, dictionaries and locales for every package installed on the system.

On a Red Hat Enterprise Linux 8 system, **langpacks** installation is based on the **langpacks-<langcode>** language meta-packages and RPM weak dependencies (Supplements tag).

There are two prerequisites to be able to use **langpacks** for a selected language. If these prerequisites are fulfilled, the language meta-packages pull their langpack for the selected language automatically in the transaction set.

Prerequisites

- The **langpacks-<langcode>** language meta-package for the selected language has been installed on the system.
  On Red Hat Enterprise Linux 8, the langpacks meta packages are installed automatically with the initial installation of the operating system using the Anaconda installer, because these packages are available in the in Application Stream repository.

  For more information, see Checking languages that provide langpacks.

- The base package, for which you want to search the locale packages, has already been installed on the system.

16.1. CHECKING LANGUAGES THAT PROVIDE LANGPACKS

Folow this procedure to check which languages provide langpacks.

Procedure

- Execute the following command:

  ```
  # yum list langpacks-*
  ```

16.2. WORKING WITH RPM WEAK DEPENDENCY-BASED LANGPACKS

This section describes multiple actions that you may want to perform when querying RPM weak dependency-based langpacks, installing or removing language support.

16.2.1. Listing already installed language support

To list the already installed language support, use this procedure.

Procedure

- Execute the following command:

  ```
  # yum list installed langpacks*
  ```

16.2.2. Checking the availability of language support

To check if language support is available for any language, use the following procedure.

Procedure

- Execute the following command:

```
# yum list available langpacks*
```

16.2.3. Listing packages installed for a language

To list what packages get installed for any language, use the following procedure:

Procedure

- Execute the following command:

  ```
  # yum repoquery --whatsupplements langpacks-<locale_code>
  ```

16.2.4. Installing language support

To add new a language support, use the following procedure.

Procedure

- Execute the following command:

  ```
  # yum install langpacks-<locale_code>
  ```

16.2.5. Removing language support

To remove any installed language support, use the following procedure.

Procedure

- Execute the following command:

  ```
  # yum remove langpacks-<locale_code>
  ```

16.3. SAVING DISK SPACE BY USING GLIBC-LANGPACK-<LOCALE_CODE>

Currently, all locales are stored in the **/usr/lib/locale/locale-archive** file, which requires a lot of disk space.

On systems where disk space is a critical issue, such as containers and cloud images, or only a few locales are needed, you can use the glibc locale langpack packages (**glibc-langpack-<locale_code>**).

To install locales individually, and thus gain a smaller package installation footprint, use the following procedure.

Procedure

- Execute the following command:

```
# yum install glibc-langpack-<locale_code>
```

When installing the operating system with Anaconda, **glibc-langpack-<locale_code>** is installed for the language you used during the installation and also for the languages you selected as additional languages. Note that **glibc-all-langpacks**, which contains all locales, is installed by default, so some locales are duplicated. If you installed **glibc-langpack-<locale_code>** for one or more selected languages, you can delete **glibc-all-langpacks** after the installation to save the disk space.

Note that installing only selected **glibc-langpack-<locale_code>** packages instead of **glibc-all-langpacks** has impact on run time performance.

NOTE

If disk space is not an issue, keep all locales installed by using the **glibc-all-langpacks** package.

CHAPTER 17. DUMPING A CRASHED KERNEL FOR LATER ANALYSIS

To analyze why a system crashed, you can use the **kdump** service to save the contents of the system's memory for later analysis. This section provides a brief introduction to **kdump**, and information about configuring **kdump** using the RHEL web console or using the corresponding RHEL system role.

17.1. WHAT IS KDUMP

kdump is a service which provides a crash dumping mechanism and generates a dump file, known as crash dump or a **vmcore** file. The **vmcore** file has the contents of the system memory that helps in analysis and troubleshooting. **kdump** uses the **kexec** system call to boot into the second kernel, a *capture kernel* without a reboot and then captures the contents of the crashed kernel's memory and saves it into a file. The second kernel is available in a reserved part of the system memory.

IMPORTANT

A kernel crash dump can be the only information available if a system failure occur. Therefore, operational **kdump** is important in mission-critical environments. Red Hat advises to regularly update and test **kexec-tools** in your normal kernel update cycle. This is especially important when you install new kernel features.

You can enable **kdump** for all installed kernels on a machine or only for specified kernels. This is useful when there are multiple kernels used on a machine, some of which are stable enough that there is no concern that they could crash. When you install **kdump**, a default **/etc/kdump.conf** file is created. The **/etc/kdump.conf** file includes the default minimum **kdump** configuration, which you can edit to customize the **kdump** configuration.

17.2. CONFIGURING KDUMP MEMORY USAGE AND TARGET LOCATION IN WEB CONSOLE

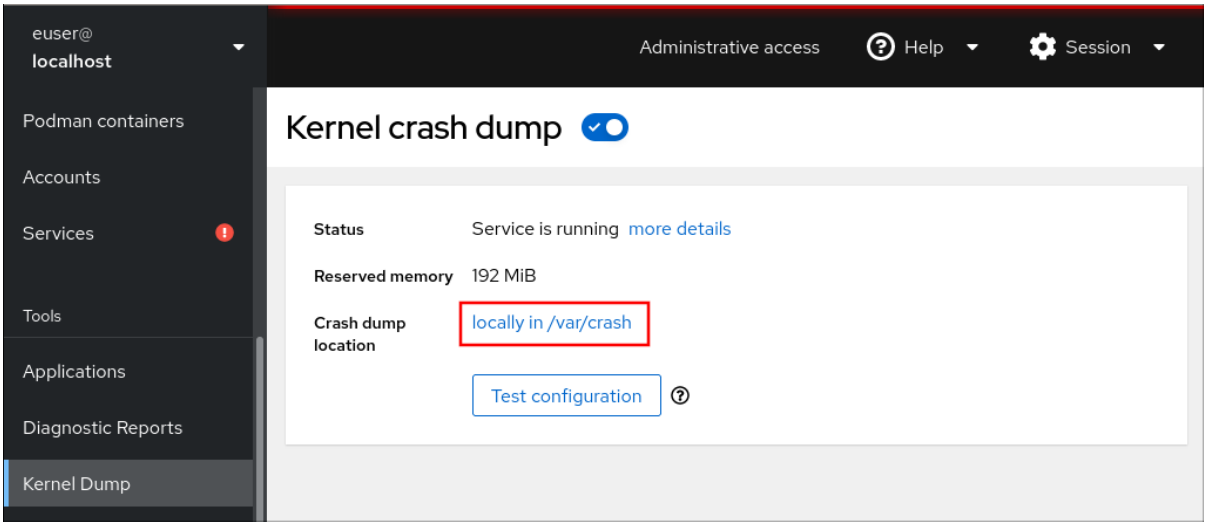
You can configure the memory reserve for the **kdump** kernel and also specify the target location to capture the **vmcore** dump file with the RHEL web console interface.

Procedure

1. In the web console, open the **Kernel Dump** tab and start the **kdump** service by setting the **Kernel crash dump** switch to on.

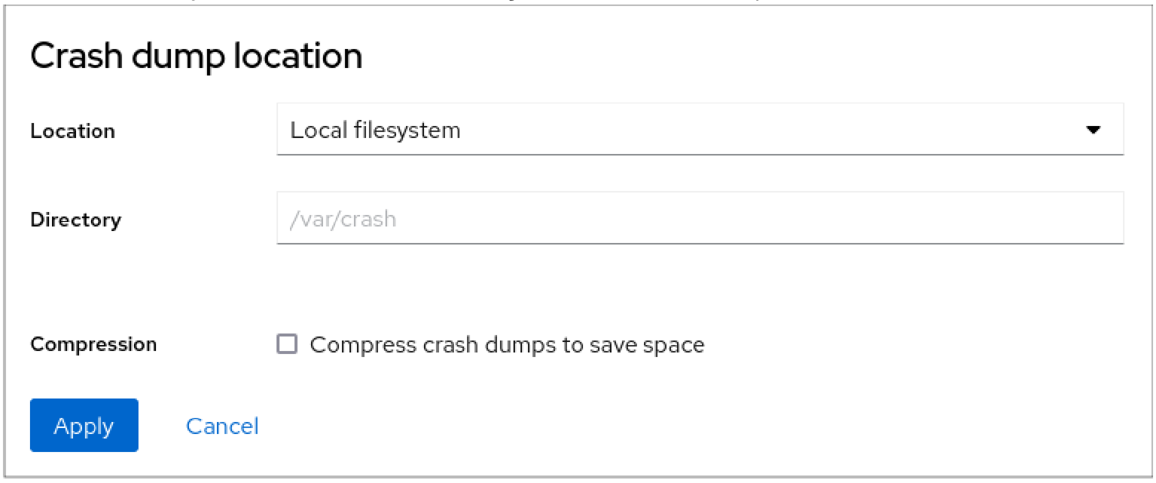
2. Configure the **kdump** memory usage in the command line.

3. In the **Kernel Dump** tab, go to **Crash dump location** and click the link with the path to the dump location.



4. Specify the target directory for saving the **vmcore** dump file:

- For a local filesystem, select **Local Filesystem** from the drop-down menu.

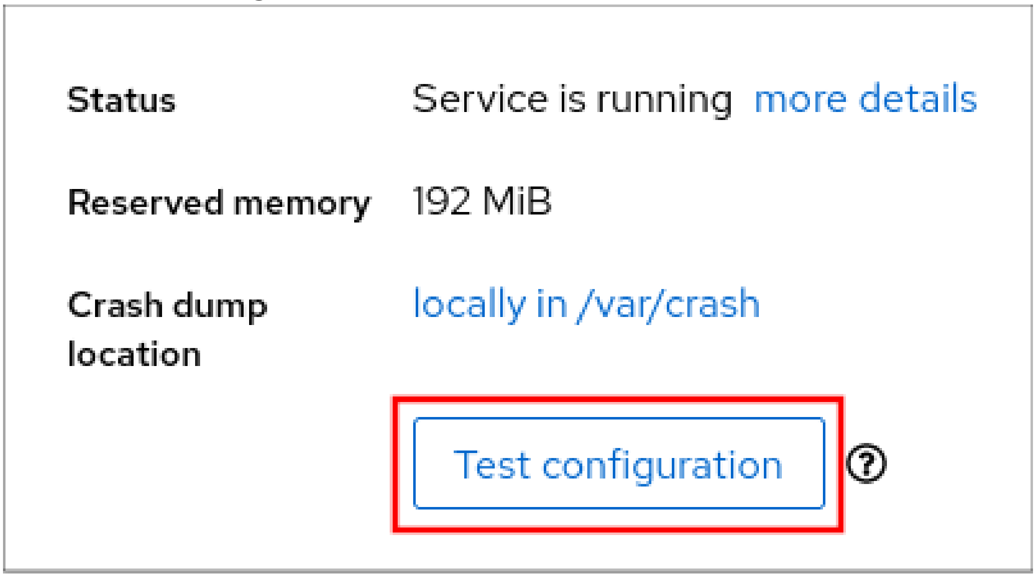


- For a remote system by using the SSH protocol, select **Remote over SSH** from the drop-down menu and specify the following fields:
  - In the **Server** field, enter the remote server address
  - In the **ssh key** field, enter the ssh key location
  - In the **Directory** field, enter the target directory
- For a remote system by using the NFS protocol, select **Remote over NFS** from the drop-down menu and specify the following fields:
  - In the **Server** field, enter the remote server address
  - In the **Export** field, enter the location of the shared folder of an NFS server
  - In the **Directory** field, enter the target directory

> **NOTE**
>
> You can reduce the size of the **vmcore** file by selecting the **Compression** check box.

Verification

1. Click the **Test configuration**.



2. Click **Crash system** under **Test kdump settings**.

> **WARNING**
>
> When you initiate the system crash, the kernel's operation stops and results in a system crash with data loss.

Additional resources

- Supported kdump targets
- Using secure communications between two systems with OpenSSH
- Getting started using the RHEL web console

17.3. KDUMP USING RHEL SYSTEM ROLES

RHEL system roles is a collection of Ansible roles and modules that provide a consistent configuration interface to remotely manage multiple RHEL systems. The **kdump** role enables you to set basic kernel dump parameters on multiple systems.

> **WARNING**
>
> The **kdump** role replaces the **kdump** configuration of the managed hosts entirely by replacing the **/etc/kdump.conf** file. Additionally, if the **kdump** role is applied, all previous **kdump** settings are also replaced, even if they are not specified by the role variables, by replacing the **/etc/sysconfig/kdump** file.

The following example playbook shows how to apply the **kdump** system role to set the location of the crash dump files:

```
---
- hosts: kdump-test
  vars:
    kdump_path: /var/crash
  roles:
    - rhel-system-roles.kdump
```

For a detailed reference on **kdump** role variables, install the **rhel-system-roles** package, and see the **README.md** or **README.html** files in the **/usr/share/doc/rhel-system-roles/kdump** directory.

Additional resources

- Introduction to RHEL system roles

17.4. ADDITIONAL RESOURCES

- Installing kdump
- Configuring kdump on the command line
- Configuring kdump in the web console

CHAPTER 18. RECOVERING AND RESTORING A SYSTEM

To recover and restore a system using an existing backup, Red Hat Enterprise Linux provides the Relax-and-Recover (ReaR) utility.

You can use the utility as a disaster recovery solution and also for system migration.

The utility enables you to perform the following tasks:

- Produce a bootable image and restore the system from an existing backup, using the image.
- Replicate the original storage layout.
- Restore user and system files.
- Restore the system to a different hardware.

Additionally, for disaster recovery, you can also integrate certain backup software with ReaR.

Setting up ReaR involves the following high-level steps:

1. Install ReaR.
2. Modify ReaR configuration file, to add backup method details.
3. Create rescue system.
4. Generate backup files.

18.1. SETTING UP REAR

Use the following steps to install the package for using the Relax-and-Recover (ReaR) utility, create a rescue system, configure and generate a backup.

Prerequisites

- Necessary configurations as per the backup restore plan are ready.
  Note that you can use the **NETFS** backup method, a fully-integrated and built-in method with ReaR.

Procedure

1. Install the ReaR utility by running the following command:

   ```
   # yum install rear
   ```

2. Modify the ReaR configuration file in an editor of your choice, for example:

   ```
   # vi /etc/rear/local.conf
   ```

3. Add the backup setting details to **/etc/rear/local.conf**. For example, in the case of the **NETFS** backup method, add the following lines:

   ```
   BACKUP=NETFS
   BACKUP_URL=backup.location
   ```

Replace *backup.location* by the URL of your backup location.

4. To configure ReaR to keep the previous backup archive when the new one is created, also add the following line to the configuration file:

```
NETFS_KEEP_OLD_BACKUP_COPY=y
```

5. To make the backups incremental, meaning that only the changed files are backed up on each run, add the following line:

```
BACKUP_TYPE=incremental
```

6. Create a rescue system:

```
# rear mkrescue
```

7. Take a backup as per the restore plan. For example, in the case of the **NETFS** backup method, run the following command:

```
# rear mkbackuponly
```

Alternatively, you can create the rescue system and the backup in a single step by running the following command:

```
# rear mkbackup
```

This command combines the functionality of the **rear mkrescue** and **rear mkbackuponly** commands.

18.2. USING A REAR RESCUE IMAGE ON THE 64-BIT IBM Z ARCHITECTURE

Basic Relax and Recover (ReaR) functionality is now available on the 64-bit IBM Z architecture as a Technology Preview. You can create a ReaR rescue image on IBM Z only in the z/VM environment. Backing up and recovering logical partitions (LPARs) has not been tested.

IMPORTANT

ReaR on the 64-bit IBM Z architecture is supported only with the **rear** package version 2.6-9.el8 or later. Earlier versions are available as a Technology Preview feature only. For more information about the support scope of Red Hat Technology Preview features, see https://access.redhat.com/support/offerings/techpreview.

The only output method currently available is Initial Program Load (IPL). IPL produces a kernel and an initial RAM disk (initrd) that can be used with the **zIPL** boot loader.

Prerequisites

- ReaR is installed.
  - To install ReaR, run the **yum install rear** command

Procedure

Add the following variables to the **/etc/rear/local.conf** to configure ReaR for producing a rescue image on the 64-bit IBM Z architecture:

1. To configure the **IPL** output method, add **OUTPUT=IPL**.
2. To configure the backup method and destination, add **BACKUP** and **BACKUP_URL** variables. For example:

```
BACKUP=NETFS

BACKUP_URL=nfs://<nfsserver name>/<share path>
```

> **IMPORTANT**
>
> The local backup storage is currently not supported on the 64-bit IBM Z architecture.

3. Optionally, you can also configure the **OUTPUT_URL** variable to save the kernel and **initrd** files. By default, the **OUTPUT_URL** is aligned with **BACKUP_URL**.
4. To perform backup and rescue image creation:

```
# rear mkbackup
```

5. This creates the kernel and initrd files at the location specified by the **BACKUP_URL** or **OUTPUT_URL** (if set) variable, and a backup using the specified backup method.
6. To recover the system, use the ReaR kernel and initrd files created in step 3, and boot from a Direct Attached Storage Device (DASD) or a Fibre Channel Protocol (FCP)-attached SCSI device prepared with the **zipl** boot loader, kernel, and **initrd**. For more information, see Using a Prepared DASD.
7. When the rescue kernel and **initrd** get booted, it starts the ReaR rescue environment. Proceed with system recovery.

> **WARNING**
>
> Currently, the rescue process reformats all the DASDs (Direct Attached Storage Devices) connected to the system. Do not attempt a system recovery if there is any valuable data present on the system storage devices. This also includes the device prepared with the zipl boot loader, ReaR kernel, and initrd that were used to boot into the rescue environment. Ensure to keep a copy.

Additional resources

- Installing under z/VM
- Using a Prepared DASD

CHAPTER 19. INSTALLING AND USING DYNAMIC PROGRAMMING LANGUAGES

Red Hat provides different programming languages, such as Python, PHP, and Tcl/TK. Use them to develop own applications and services.

19.1. INTRODUCTION TO PYTHON

Python is a high-level programming language that supports multiple programming paradigms, such as object-oriented, imperative, functional, and procedural paradigms. Python has dynamic semantics and can be used for general-purpose programming.

With Red Hat Enterprise Linux, many packages that are installed on the system, such as packages providing system tools, tools for data analysis, or web applications, are written in Python. To use these packages, you must have the **python*** packages installed.

19.1.1. Python versions

Two incompatible versions of Python are widely used, Python 2.x and Python 3.x. RHEL 8 provides the following versions of Python.

Table 19.1. Python versions in RHEL 8

Version	Package to install	Command examples	Available since	Life cycle
Python 3.6	**python3**, **python36**	**python3**, **python3.6**, **pip3**, **pip3.6**	RHEL 8.0	full RHEL 8
Python 2.7	**python2**	**python2**, **pip2**	RHEL 8.0	shorter
Python 3.8	**python38**	**python3.8**, **pip3.8**	RHEL 8.2	shorter
Python 3.9	**python39**	**python3.9**, **pip3.9**	RHEL 8.4	shorter
Python 3.11	**python3.11**	**python3.11**, **pip3.11**	RHEL 8.8	shorter

For details about the length of support, see Red Hat Enterprise Linux Life Cycle and Red Hat Enterprise Linux Application Streams Life Cycle.

Each of the Python versions up to 3.9 is distributed in a separate module. Python 3.11 is distributed as a suite of non-modular RPM packages, including the **python3.11** package.

You can install multiple Python versions in parallel on the same RHEL 8 system.

IMPORTANT

Always specify the version of Python when installing it, invoking it, or otherwise interacting with it. For example, use **python3** instead of **python** in package and command names. All Python-related commands must also include the version, for example, **pip3**, **pip2**, **pip3.8**, **pip3.9**, or **pip3.11**.

The unversioned **python** command (**/usr/bin/python**) is not available by default in RHEL 8. You can configure it using the **alternatives** command; for instructions, see Configuring the unversioned Python

Any manual changes to **/usr/bin/python**, except changes made using the **alternatives** command, might be overwritten upon an update.

As a system administrator, use Python 3 for the following reasons:

- Python 3 represents the main development direction of the Python project.
- Support for Python 2 in the upstream community ended in 2020.
- Popular Python libraries are discontinuing Python 2 support in upstream.
- Python 2 in Red Hat Enterprise Linux 8 will have a shorter life cycle and aims to facilitate a smoother transition to **Python 3** for customers.

For developers, Python 3 has the following advantages over Python 2:

- Python 3 enables you to write expressive, maintainable, and correct code more easily.
- Code written in Python 3 will have greater longevity.
- Python 3 has new features, including **asyncio**, f-strings, advanced unpacking, keyword-only arguments, and chained exceptions.

However, legacy software might require **/usr/bin/python** to be configured to Python 2. For this reason, no default **python** package is distributed with Red Hat Enterprise Linux 8, and you can choose between using Python 2 and 3 as **/usr/bin/python**, as described in Configuring the unversioned Python.

IMPORTANT

System tools in Red Hat Enterprise Linux 8 use Python version 3.6 provided by the internal **platform-python** package, which is not intended to be used directly by customers. It is recommended to use the **python3** or **python3.6** command from the **python36** package for Python 3.6, or to use later Python versions.

Do not remove the **platform-python** package from RHEL 8 because other packages require it.

19.1.2. Notable differences between Python versions

Python versions included in RHEL 8 differ in various aspects.

Python bindings

The **python38** and **python39** modules and the **python3.11** package suite do not include the same bindings to system tools (RPM, DNF, SELinux, and others) that are provided for the **python36** module.

Therefore, use **python36** in instances where the greatest compatibility with the base operating system or binary compatibility is necessary. In unique instances where system bindings are necessary together with later versions of various Python modules, use the **python36** module in combination with third-party upstream Python modules installed through **pip** into Python's **venv** or **virtualenv** environments.

Python 3.11 virtual environments must be created using venv instead of virtualenv

The **virtualenv** utility in RHEL 8, provided by the **python3-virtualenv** package, is not compatible with Python 3.11. An attempt to create a virtual environment by using **virtualenv** will fail with the following error message:

```
$ virtualenv -p python3.11 venv3.11
Running virtualenv with interpreter /usr/bin/python3.11
ERROR: Virtual environments created by virtualenv < 20 are not compatible with Python 3.11.
ERROR: Use python3.11 -m venv instead.
```

To create Python 3.11 virtual environments, use the **python3.11 -m venv** command instead, which uses the **venv** module from the standard library.

19.2. INSTALLING AND USING PYTHON

In Red Hat Enterprise Linux 8, Python 3 is distributed in versions 3.6, 3.8, and 3.9, provided by the **python36**, **python38**, and **python39** modules, and the **python3.11** package suite in the AppStream repository.

> **WARNING**
>
> Using the unversioned **python** command to install or run Python does not work by default due to ambiguity. Always specify the version of Python, or configure the system default version by using the **alternatives** command.

19.2.1. Installing Python 3

By design, you can install RHEL 8 modules in parallel, including the **python27**, **python36**, **python38**, and **python39** modules, and the **python3.11** package suite.

You can install Python 3.8, Python 3.9, and Python 3.11, including packages built for each version, in parallel with Python 3.6 on the same system, with the exception of the **mod_wsgi** module. Due to a limitation of the Apache HTTP Server, only one of the **python3-mod_wsgi**, **python38-mod_wsgi**, **python39-mod_wsgi**, or **python3.11-mod_wsgi** packages can be installed on a system.

Procedure

- To install Python 3.6 from the **python36** module, use:

  ```
  # yum install python3
  ```

  The **python36:3.6** module stream is enabled automatically.

- To install Python 3.8 from the **python38** module, use:

```
# yum install python38
```

The **python38:3.8** module stream is enabled automatically.

- To install Python 3.9 from the **python39** module, use:

```
# yum install python39
```

The **python39:3.9** module stream is enabled automatically.

- To install Python 3.11 from the **python3.11** RPM package, use:

```
# yum install python3.11
```

Verification steps

- To verify the Python version installed on your system, use the **--version** option with the **python** command specific for your required version of Python.
  - For Python 3.6:

```
$ python3 --version
```

  - For Python 3.8:

```
$ python3.8 --version
```

  - For Python 3.9:

```
$ python3.9 --version
```

  - For Python 3.11:

```
$ python3.11 --version
```

Additional resources

- Installing, managing, and removing user-space components

19.2.2. Installing additional Python 3 packages

Packages with add-on modules for Python 3.6 generally use the **python3-** prefix, packages for Python 3.8 include the **python38-** prefix, packages for Python 3.9 include the **python39-** prefix, and packages for Python 3.11 include the **python3.11-** prefix. Always include the prefix when installing additional Python packages, as shown in the examples below.

Procedure

- To install the **Requests** module for Python 3.6, use:

```
# yum install python3-requests
```

- To install the **Cython** extension to Python 3.8, use:

  ```
  # yum install python38-Cython
  ```

- To install the **pip** package installer from Python 3.9, use:

  ```
  # yum install python39-pip
  ```

- To install the **pip** package installer from Python 3.11, use:

  ```
  # yum install python3.11-pip
  ```

Additional resources

- Upstream documentation about Python add-on modules

19.2.3. Installing additional Python 3 tools for developers

Additional Python tools for developers are distributed mostly through the CodeReady Linux Builder (CRB) repository in the respective **python38-devel** or **python39-devel** module, or the **python3.11-*** packages.

The **python3-pytest** package (for Python 3.6) and its dependencies are available in the AppStream repository.

The CRB repository provides:

- The **python38-devel** module, which contains the **python38-pytest** package and its dependencies.
- The **python39-devel** module, which contains the **python39-pytest** package and its dependencies, and the **python39-debug** and **python39-Cython** packages.
- The **python3.11-*** packages, which include:
  - **python3.11-pytest** and its dependencies
  - **python3.11-idle**
  - **python3.11-debug**
  - **python3.11-Cython**

IMPORTANT

The content in the CodeReady Linux Builder repository is unsupported by Red Hat.

NOTE

Not all upstream Python-related packages are available in RHEL.

To install the **python3*-pytest** package, use the following procedure.

Procedure

1. For Python 3.8 and later, enable the CodeReady Linux Builder repository:

```
# subscription-manager repos --enable codeready-builder-for-rhel-8-x86_64-rpms
```

2. For Python 3.8 or 3.9, enable the respective **python3*-devel** module, for example:

```
# yum module enable python39-devel
```

3. Install the **python3*-pytest** package:

   - For Python 3.6:

   ```
   # yum install python3-pytest
   ```

   - For Python 3.8:

   ```
   # yum install python38-pytest
   ```

   - For Python 3.9:

   ```
   # yum install python39-pytest
   ```

   - For Python 3.11:

   ```
   # yum install python3.11-pytest
   ```

Additional resources

- How to enable and make use of content within CodeReady Linux Builder
- Package manifest

19.2.4. Installing Python 2

Some applications and scripts have not yet been fully ported to Python 3 and require Python 2 to run. Red Hat Enterprise Linux 8 allows parallel installation of Python 3 and Python 2. If you need the Python 2 functionality, install the **python27** module, which is available in the AppStream repository.

> **WARNING**
>
> Note that Python 3 is the main development direction of the Python project. Support for Python 2 is being phased out. The **python27** module has a shorter support period than Red Hat Enterprise Linux 8.

Procedure

- To install Python 2.7 from the **python27** module, use:

  ```
  # yum install python2
  ```

  The **python27:2.7** module stream is enabled automatically.

Packages with add-on modules for Python 2 generally use the **python2-** prefix. Always include the prefix when installing additional Python packages, as shown in the examples below.

- To install the **Requests** module for Python 2, use:

  ```
  # yum install python2-requests
  ```

- To install the **Cython** extension to Python 2, use:

  ```
  # yum install python2-Cython
  ```

Verification steps

- To verify the Python version installed on your system, use:

  ```
  $ python2 --version
  ```

NOTE

By design, you can install RHEL 8 modules in parallel, including the **python27**, **python36**, **python38**, and **python39** modules.

Additional resources

- Installing, managing, and removing user-space components in RHEL 8

19.2.5. Migrating from Python 2 to Python 3

As a developer, you may want to migrate your former code that is written in Python 2 to Python 3.

For more information about how to migrate large code bases to Python 3, see The Conservative Python 3 Porting Guide.

Note that after this migration, the original Python 2 code becomes interpretable by the Python 3 interpreter and stays interpretable for the Python 2 interpreter as well.

19.2.6. Using Python

When running the Python interpreter or Python-related commands, always specify the version.

Prerequisites

- Ensure that the required version of Python is installed.
- If you want to download and install third-party applications for Python 3.11, install the **python3.11-pip** package.

Procedure

- To run the Python 3.6 interpreter or related commands, use, for example:

```
$ python3
$ python3 -m venv --help
$ python3 -m pip install package
$ pip3 install package
```

- To run the Python 3.8 interpreter or related commands, use, for example:

```
$ python3.8
$ python3.8 -m venv --help
$ python3.8 -m pip install package
$ pip3.8 install package
```

- To run the Python 3.9 interpreter or related commands, use, for example:

```
$ python3.9
$ python3.9 -m venv --help
$ python3.9 -m pip install package
$ pip3.9 install package
```

- To run the Python 3.11 interpreter or related commands, use, for example:

```
$ python3.11
$ python3.11 -m venv --help
$ python3.11 -m pip install package
$ pip3.11 install package
```

- To run the Python 2 interpreter or related commands, use, for example:

```
$ python2
$ python2 -m pip install package
$ pip2 install package
```

19.3. CONFIGURING THE UNVERSIONED PYTHON

System administrators can configure the unversioned **python** command, located at **/usr/bin/python**, using the **alternatives** command. Note that the required package, **python3**, **python38**, **python39**, **python3.11**, or **python2**, must be installed before configuring the unversioned command to the respective version.

IMPORTANT

The **/usr/bin/python** executable is controlled by the **alternatives** system. Any manual changes may be overwritten upon an update.

Additional Python-related commands, such as **pip3**, do not have configurable unversioned variants.

19.3.1. Configuring the unversioned python command directly

You can configure the unversioned **python** command directly to a selected version of Python.

Prerequisites

- Ensure that the required version of Python is installed.

Procedure

- To configure the unversioned **python** command to Python 3.6, use:

  ```
  # alternatives --set python /usr/bin/python3
  ```

- To configure the unversioned **python** command to Python 3.8, use:

  ```
  # alternatives --set python /usr/bin/python3.8
  ```

- To configure the unversioned **python** command to Python 3.9, use:

  ```
  # alternatives --set python /usr/bin/python3.9
  ```

- To configure the unversioned **python** command to Python 3.11, use:

  ```
  # alternatives --set python /usr/bin/python3.11
  ```

- To configure the unversioned **python** command to Python 2, use:

  ```
  # alternatives --set python /usr/bin/python2
  ```

19.3.2. Configuring the unversioned python command to the required Python version interactively

You can configure the unversioned **python** command to the required Python version interactively.

Prerequisites

- Ensure that the required version of Python is installed.

Procedure

1. To configure the unversioned **python** command interactively, use:

   ```
   # alternatives --config python
   ```

2. Select the required version from the provided list.

3. To reset this configuration and remove the unversioned **python** command, use:

   ```
   # alternatives --auto python
   ```

19.3.3. Additional resources

- **alternatives(8)** and **unversioned-python(1)** man pages

19.4. PACKAGING PYTHON 3 RPMS

Most Python projects use Setuptools for packaging, and define package information in the **setup.py** file. For more information about Setuptools packaging, see the Setuptools documentation.

You can also package your Python project into an RPM package, which provides the following advantages compared to Setuptools packaging:

- Specification of dependencies of a package on other RPMs (even non-Python)
- Cryptographic signing
  With cryptographic signing, content of RPM packages can be verified, integrated, and tested with the rest of the operating system.

19.4.1. SPEC file description for a Python package

A SPEC file contains instructions that the **rpmbuild** utility uses to build an RPM. The instructions are included in a series of sections. A SPEC file has two main parts in which the sections are defined:

- Preamble (contains a series of metadata items that are used in the Body)
- Body (contains the main part of the instructions)

An RPM SPEC file for Python projects has some specifics compared to non-Python RPM SPEC files. Most notably, a name of any RPM package of a Python library must always include the prefix determining the version, for example, **python3** for Python 3.6, **python38** for Python 3.8, **python39** for Python 3.9, or **python3.11** for Python 3.11.

Other specifics are shown in the following SPEC file **example for the python3-detox package**. For description of such specifics, see the notes below the example.

```
%global modname detox                                          1

Name:           python3-detox                                  2
Version:        0.12
Release:        4%{?dist}
Summary:        Distributing activities of the tox tool
License:        MIT
URL:            https://pypi.io/project/detox
Source0:        https://pypi.io/packages/source/d/%{modname}/%{modname}-%{version}.tar.gz

BuildArch:      noarch

BuildRequires:  python36-devel                                 3
BuildRequires:  python3-setuptools
BuildRequires:  python36-rpm-macros
BuildRequires:  python3-six
BuildRequires:  python3-tox
BuildRequires:  python3-py
BuildRequires:  python3-eventlet

%?python_enable_dependency_generator                           4
```

```
%description

Detox is the distributed version of the tox python testing tool. It makes efficient use of multiple CPUs
by running all possible activities in parallel.
Detox has the same options and configuration that tox has, so after installation you can run it in the
same way and with the same options that you use for tox.

    $ detox

%prep
%autosetup -n %{modname}-%{version}

%build
%py3_build                          5

%install
%py3_install

%check
%{__python3} setup.py test          6

%files -n python3-%{modname}
%doc CHANGELOG
%license LICENSE
%{_bindir}/detox
%{python3_sitelib}/%{modname}/
%{python3_sitelib}/%{modname}-%{version}*

%changelog
...
```

1. The **modname** macro contains the name of the Python project. In this example it is **detox**.
2. When packaging a Python project into RPM, the **python3** prefix always needs to be added to the original name of the project. The original name here is **detox** and the **name of the RPM** is **python3-detox**.
3. **BuildRequires** specifies what packages are required to build and test this package. In BuildRequires, always include items providing tools necessary for building Python packages: **python36-devel** and **python3-setuptools**. The **python36-rpm-macros** package is required so that files with **/usr/bin/python3** interpreter directives are automatically changed to **/usr/bin/python3.6**.
4. Every Python package requires some other packages to work correctly. Such packages need to be specified in the SPEC file as well. To specify the **dependencies**, you can use the **%python_enable_dependency_generator** macro to automatically use dependencies defined in the **setup.py** file. If a package has dependencies that are not specified using Setuptools, specify them within additional **Requires** directives.
5. The **%py3_build** and **%py3_install** macros run the **setup.py build** and **setup.py install** commands, respectively, with additional arguments to specify installation locations, the interpreter to use, and other details.
6. The **check** section provides a macro that runs the correct version of Python. The **%{__python3}** macro contains a path for the Python 3 interpreter, for example **/usr/bin/python3**. We recommend to always use the macro rather than a literal path.

19.4.2. Common macros for Python 3 RPMs

In a SPEC file, always use the macros that are described in the following Macros for Python 3 RPMs table rather than hardcoding their values.

In macro names, always use **python3** or **python2** instead of unversioned **python**. Configure the particular Python 3 version in the **BuildRequires** section of the SPEC file to **python36-rpm-macros**, **python38-rpm-macros**, **python39-rpm-macros**, or **python3.11-rpm-macros**.

Table 19.2. Macros for Python 3 RPMs

Macro	Normal Definition	Description
%{__python3}	/usr/bin/python3	Python 3 interpreter
%{python3_version}	3.6	The full version of the Python 3 interpreter.
%{python3_sitelib}	/usr/lib/python3.6/site-packages	Where pure-Python modules are installed.
%{python3_sitearch}	/usr/lib64/python3.6/site-packages	Where modules containing architecture-specific extensions are installed.
%py3_build		Runs the **setup.py build** command with arguments suitable for a system package.
%py3_install		Runs the **setup.py install** command with arguments suitable for a system package.

19.4.3. Automatic provides for Python RPMs

When packaging a Python project, make sure that the following directories are included in the resulting RPM if these directories are present:

- **.dist-info**
- **.egg-info**
- **.egg-link**

From these directories, the RPM build process automatically generates virtual **pythonX.Ydist** provides, for example, **python3.6dist(detox)**. These virtual provides are used by packages that are specified by the **%python_enable_dependency_generator** macro.

19.5. HANDLING INTERPRETER DIRECTIVES IN PYTHON SCRIPTS

In Red Hat Enterprise Linux 8, executable Python scripts are expected to use interpreter directives (also known as hashbangs or shebangs) that explicitly specify at a minimum the major Python version. For example:

```
#!/usr/bin/python3
#!/usr/bin/python3.6
#!/usr/bin/python3.8
#!/usr/bin/python3.9
#!/usr/bin/python3.11
#!/usr/bin/python2
```

The **/usr/lib/rpm/redhat/brp-mangle-shebangs** buildroot policy (BRP) script is run automatically when building any RPM package, and attempts to correct interpreter directives in all executable files.

The BRP script generates errors when encountering a Python script with an ambiguous interpreter directive, such as:

```
#!/usr/bin/python
```

or

```
#!/usr/bin/env python
```

19.5.1. Modifying interpreter directives in Python scripts

Modify interpreter directives in the Python scripts that cause the build errors at RPM build time.

Prerequisites

- Some of the interpreter directives in your Python scripts cause a build error.

Procedure

To modify interpreter directives, complete one of the following tasks:

- Apply the **pathfix.py** script from the **platform-python-devel** package:

  ```
  # pathfix.py -pn -i %{__python3} PATH ...
  ```

  Note that multiple ***PATH*s** can be specified. If a ***PATH*** is a directory, **pathfix.py** recursively scans for any Python scripts matching the pattern **^[a-zA-Z0-9_]+\.py$**, not only those with an ambiguous interpreter directive. Add this command to the **%prep** section or at the end of the **%install** section.

- Modify the packaged Python scripts so that they conform to the expected format. For this purpose, **pathfix.py** can be used outside the RPM build process, too. When running **pathfix.py** outside an RPM build, replace ***%{__python3}*** from the example above with a path for the interpreter directive, such as **/usr/bin/python3**.

If the packaged Python scripts require a version other than Python 3.6, adjust the preceding commands to include the required version.

19.5.2. Changing /usr/bin/python3 interpreter directives in your custom packages

By default, interpreter directives in the form of **/usr/bin/python3** are replaced with interpreter directives pointing to Python from the **platform-python** package, which is used for system tools with Red Hat Enterprise Linux. You can change the **/usr/bin/python3** interpreter directives in your custom packages to point to a specific version of Python that you have installed from the AppStream repository.

Procedure

- To build your package for a specific version of Python, add the ***python**-rpm-macros** subpackage of the respective **python** package to the **BuildRequires** section of the SPEC file. For example, for Python 3.6, include the following line:

  ```
  BuildRequires:  python36-rpm-macros
  ```

  As a result, the **/usr/bin/python3** interpreter directives in your custom package are automatically converted to **/usr/bin/python3.6**.

> **NOTE**
>
> To prevent the BRP script from checking and modifying interpreter directives, use the following RPM directive:
>
> ```
> %undefine __brp_mangle_shebangs
> ```

19.6. USING THE PHP SCRIPTING LANGUAGE

Hypertext Preprocessor (PHP) is a general-purpose scripting language mainly used for server-side scripting, which enables you to run the PHP code using a web server.

In RHEL 8, the PHP scripting language is provided by the **php** module, which is available in multiple streams (versions).

Depending on your use case, you can install a specific profile of the selected module stream:

- **common** - The default profile for server-side scripting using a web server. It includes several widely used extensions.
- **minimal** - This profile installs only the command-line interface for scripting with PHP without using a web server.
- **devel** - This profile includes packages from the **common** profile and additional packages for development purposes.

19.6.1. Installing the PHP scripting language

You can install a selected version of the **php** module.

Procedure

- To install a **php** module stream with the default profile, use:

  ```
  # yum module install php:stream
  ```

  Replace *stream* with the version of PHP you wish to install.

  For example, to install PHP 8.0:

  ```
  # yum module install php:8.0
  ```

The default **common** profile installs also the **php-fpm** package, and preconfigures PHP for use with the **Apache HTTP Server** or **nginx**.

- To install a specific profile of a **php** module stream, use:

  ```
  # yum module install php:stream/profile
  ```

  Replace *stream* with the desired version and *profile* with the name of the profile you wish to install.

  For example, to install PHP 8.0 for use without a web server:

  ```
  # yum module install php:8.0/minimal
  ```

Additional resources

- If you want to upgrade from an earlier version of PHP available in RHEL 8, see Switching to a later stream.
- For more information about managing RHEL 8 modules and streams, see Installing, managing, and removing user-space components.

19.6.2. Using the PHP scripting language with a web server

19.6.2.1. Using PHP with the Apache HTTP Server

In Red Hat Enterprise Linux 8, the **Apache HTTP Server** enables you to run PHP as a FastCGI process server. FastCGI Process Manager (FPM) is an alternative PHP FastCGI daemon that allows a website to manage high loads. PHP uses FastCGI Process Manager by default in RHEL 8.

You can run the PHP code using the FastCGI process server.

Prerequisites

- The PHP scripting language is installed on your system.
  See Installing the PHP scripting language.

Procedure

1. Install the **httpd** module:

   ```
   # yum module install httpd:2.4
   ```

2. Start the **Apache HTTP Server**:

   ```
   # systemctl start httpd
   ```

   Or, if the **Apache HTTP Server** is already running on your system, restart the **httpd** service after installing PHP:

   ```
   # systemctl restart httpd
   ```

3. Start the **php-fpm** service:

```
# systemctl start php-fpm
```

4. Optional: Enable both services to start at boot time:

```
# systemctl enable php-fpm httpd
```

5. To obtain information about your PHP settings, create the **index.php** file with the following content in the **/var/www/html/** directory:

```
# echo '<?php phpinfo(); ?>' > /var/www/html/index.php
```

6. To run the **index.php** file, point the browser to:

```
http://<hostname>/
```

7. Optional: Adjust configuration if you have specific requirements:
   - **/etc/httpd/conf/httpd.conf** - generic **httpd** configuration
   - **/etc/httpd/conf.d/php.conf** - PHP-specific configuration for **httpd**
   - **/usr/lib/systemd/system/httpd.service.d/php-fpm.conf** - by default, the **php-fpm** service is started with **httpd**
   - **/etc/php-fpm.conf** - FPM main configuration
   - **/etc/php-fpm.d/www.conf** - default **www** pool configuration

Example 19.1. Running a "Hello, World!" PHP script using the Apache HTTP Server

1. Create a **hello** directory for your project in the **/var/www/html/** directory:

```
# mkdir hello
```

2. Create a **hello.php** file in the **/var/www/html/hello/** directory with the following content:

```
# <!DOCTYPE html>
<html>
<head>
<title>Hello, World! Page</title>
</head>
<body>
<?php
    echo 'Hello, World!';
?>
</body>
</html>
```

3. Start the **Apache HTTP Server**:

```
# systemctl start httpd
```

4. To run the **hello.php** file, point the browser to:

```
http://<hostname>/hello/hello.php
```

As a result, a web page with the “Hello, World!” text is displayed.

Additional resources

- Setting up the Apache HTTP web server

19.6.2.2. Using PHP with the nginx web server

You can run PHP code through the **nginx** web server.

Prerequisites

- The PHP scripting language is installed on your system.
  See Installing the PHP scripting language.

Procedure

1. Install an **nginx** module stream:

   ```
   # yum module install nginx:stream
   ```

   Replace *stream* with the version of **nginx** you wish to install.

   For example, to install **nginx** version 1.18:

   ```
   # yum module install nginx:1.18
   ```

2. Start the **nginx** server:

   ```
   # systemctl start nginx
   ```

   Or, if the **nginx** server is already running on your system, restart the **nginx** service after installing PHP:

   ```
   # systemctl restart nginx
   ```

3. Start the **php-fpm** service:

   ```
   # systemctl start php-fpm
   ```

4. Optional: Enable both services to start at boot time:

   ```
   # systemctl enable php-fpm nginx
   ```

5. To obtain information about your PHP settings, create the **index.php** file with the following content in the **/usr/share/nginx/html/** directory:

   ```
   # echo '<?php phpinfo(); ?>' > /usr/share/nginx/html/index.php
   ```

6. To run the **index.php** file, point the browser to:

```
http://<hostname>/
```

7. Optional: Adjust configuration if you have specific requirements:

   - **/etc/nginx/nginx.conf** - **nginx** main configuration
   - **/etc/nginx/conf.d/php-fpm.conf** - FPM configuration for **nginx**
   - **/etc/php-fpm.conf** - FPM main configuration
   - **/etc/php-fpm.d/www.conf** - default **www** pool configuration

Example 19.2. Running a "Hello, World!" PHP script using the nginx server

1. Create a **hello** directory for your project in the **/usr/share/nginx/html/** directory:

```
# mkdir hello
```

2. Create a **hello.php** file in the **/usr/share/nginx/html/hello/** directory with the following content:

```
# <!DOCTYPE html>
<html>
<head>
<title>Hello, World! Page</title>
</head>
<body>
<?php
    echo 'Hello, World!';
?>
</body>
</html>
```

3. Start the **nginx** server:

```
# systemctl start nginx
```

4. To run the **hello.php** file, point the browser to:

```
http://<hostname>/hello/hello.php
```

As a result, a web page with the “Hello, World!” text is displayed.

Additional resources

- Setting up and configuring NGINX

19.6.3. Running a PHP script using the command-line interface

A PHP script is usually run using a web server, but also can be run using the command-line interface.

If you want to run **php** scripts using only command-line, install the **minimal** profile of a **php** module stream.

See Installing the PHP scripting language.

Prerequisites

- The PHP scripting language is installed on your system.
  See Installing the PHP scripting language.

Procedure

1. In a text editor, create a ***filename*.php** file
   Replace *filename* with the name of your file.

2. Execute the created ***filename*.php** file from the command line:

```
# php filename.php
```

Example 19.3. Running a "Hello, World!" PHP script using the command-line interface

1. Create a **hello.php** file with the following content using a text editor:

```
<?php
    echo 'Hello, World!';
?>
```

2. Execute the **hello.php** file from the command line:

```
# php hello.php
```

As a result, “Hello, World!” is printed.

19.6.4. Additional resources

- **httpd(8)** — The manual page for the **httpd** service containing the complete list of its command-line options.
- **httpd.conf(5)** — The manual page for **httpd** configuration, describing the structure and location of the **httpd** configuration files.
- **nginx(8)** — The manual page for the **nginx** web server containing the complete list of its command-line options and list of signals.
- **php-fpm(8)** — The manual page for PHP FPM describing the complete list of its command-line options and configuration files.

19.7. GETTING STARTED WITH TCL/TK

19.7.1. Introduction to Tcl/Tk

Tool command language (Tcl) is a dynamic programming language. The interpreter for this language, together with the C library, is provided by the **tcl** package.

Using **Tcl** paired with **Tk** (**Tcl/Tk**) enables creating cross-platform GUI applications. **Tk** is provided by the **tk** package.

Note that **Tk** can refer to any of the following:

- A programming toolkit for multiple languages
- A Tk C library bindings available for multiple languages, such as C, Ruby, Perl and Python
- A wish interpreter that instantiates a Tk console
- A Tk extension that adds a number of new commands to a particular Tcl interpreter

For more information about Tcl/Tk, see the Tcl/Tk manual or Tcl/Tk documentation web page.

19.7.2. Notable changes in Tcl/Tk 8.6

Red Hat Enterprise Linux 7 used **Tcl/Tk 8.5**. With Red Hat Enterprise Linux 8, **Tcl/Tk version 8.6** is provided in the Base OS repository.

Major changes in **Tcl/Tk 8.6** compared to **Tcl/Tk 8.5** are:

- Object-oriented programming support
- Stackless evaluation implementation
- Enhanced exceptions handling
- Collection of third-party packages built and installed with Tcl
- Multi-thread operations enabled
- SQL database-powered scripts support
- IPv6 networking support
- Built-in Zlib compression
- List processing
  Two new commands, **lmap** and **dict map** are available, which allow the expression of transformations over **Tcl** containers.
- Stacked channels by script
  Two new commands, **chan push** and **chan pop** are available, which allow to add or remove transformations to or from I/O channels.

Major changes in **Tk** include:

- Built-in PNG image support
- Busy windows
  A new command, **tk busy** is available, which disables user interaction for a window or a widget and shows the busy cursor.

- New font selection dialog interface
- Angled text support
- Moving things on a canvas support

For the detailed list of changes between **Tcl 8.5** and **Tcl 8.6**, see Changes in Tcl/Tk 8.6.

19.7.3. Migrating to Tcl/Tk 8.6

Red Hat Enterprise Linux 7 used **Tcl/Tk 8.5**. With Red Hat Enterprise Linux 8, **Tcl/Tk version 8.6** is provided in the Base OS repository.

This section describes migration path to **Tcl/Tk 8.6** for:

- Developers writing **Tcl** extensions or embedding **Tcl** interpreter into their applications
- Users scripting tasks with **Tcl/Tk**

19.7.3.1. Migration path for developers of Tcl extensions

To make your code compatible with **Tcl 8.6**, use the following procedure.

Procedure

1. Rewrite the code to use the **interp** structure. For example, if your code reads **interp→errorLine**, rewrite it to use the following function:

   ```
   Tcl_GetErrorLine(interp)
   ```

   This is necessary because **Tcl 8.6** limits direct access to members of the **interp** structure.

2. To make your code compatible with both **Tcl 8.5** and **Tcl 8.6**, use the following code snippet in a header file of your C or C++ application or extension that includes the **Tcl** library:

   ```
   # include <tcl.h>
   # if !defined(Tcl_GetErrorLine)
   # define Tcl_GetErrorLine(interp) (interp→errorLine)
   # endif
   ```

19.7.3.2. Migration path for users scripting their tasks with Tcl/Tk

In **Tcl 8.6**, most scripts work the same way as with the previous version of **Tcl**.

To migrate you code into **Tcl 8.6**, use this procedure.

Procedure

- When writing a portable code, make sure to not use the commands that are no longer supported in **Tk 8.6**:

  ```
  tkIconList_Arrange
  tkIconList_AutoScan
  tkIconList_Btn1
  ```

```
tkIconList_Config
tkIconList_Create
tkIconList_CtrlBtn1
tkIconList_Curselection
tkIconList_DeleteAll
tkIconList_Double1
tkIconList_DrawSelection
tkIconList_FocusIn
tkIconList_FocusOut
tkIconList_Get
tkIconList_Goto
tkIconList_Index
tkIconList_Invoke
tkIconList_KeyPress
tkIconList_Leave1
tkIconList_LeftRight
tkIconList_Motion1
tkIconList_Reset
tkIconList_ReturnKey
tkIconList_See
tkIconList_Select
tkIconList_Selection
tkIconList_ShiftBtn1
tkIconList_UpDown
```

Note that you can check the list of unsupported commands also in the **/usr/share/tk8.6/unsupported.tcl** file.