Red Hat Enterprise Linux 8

配置基本系统设置

设置系统的基本功能并自定义您的系统环境

Last Updated: 2024-07-10

Red Hat Enterprise Linux 8 配置基本系统设置

设置系统的基本功能并自定义您的系统环境

法律通告

摘要

执行基本的系统管理任务、配置环境设置、注册您的系统以及配置网络访问和系统安全性。管理用户、组和文件权限。使用系统角色来管理多个 RHEL 系统上的系统配置接口。使用 systemd 进行有效的服务管理。使用 chrony 配置网络时间协议(NTP)。使用 ReaR 备份和恢复您的系统。安装和使用动态编程语言，如 Python 3 和 PHP。

目录

对红帽文档提供反馈

我们感谢您对我们文档的反馈。让我们了解如何改进它。

通过 Jira 提交反馈（需要帐户）

1. 登录到 Jira 网站。
2. 单击顶部导航栏中的 **Create**。
3. 在 **Summary** 字段中输入描述性标题。
4. 在 **Description** 字段中输入您的改进建议。包括文档相关部分的链接。
5. 点对话框底部的 **Create**。

第 1 章 配置和管理基本网络访问

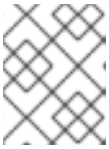
这部分只论述了如何在 Red Hat Enterprise Linux 中配置网络设置的基本选项。

1.1. 在图形安装模式中配置网络和主机名

按照以下步骤来配置您的网络和主机名。

步骤

1. 在 **Installation Summary** 窗口中点击 **Network and Host Name**。
2. 在左侧窗格的列表中选择一个接口。详情显示在右侧方框中。



注意

有几个可用来使用持久名称识别网络设备的网络设备命名标准，例如： **em1** 和 **wl3sp0**。有关这些标准的详情，请查看 *配置和管理联网文档*。

3. 使用 **ON/OFF** 开关来启用或禁用所选接口。

注意

安装程序自动检测到本地可访问的界面，您无法手动添加或删除它们。

4. 点 **+** 添加虚拟网络接口，可以是 Team、Bond、Bridge 或 VLAN。
5. 点 **-** 删除虚拟接口。
6. 点 **Configure** 更改设置，如 IP 地址、DNS 服务器或者现有接口的路由配置（虚拟和物理）。
7. 在 **Host Name** 字段中输入您系统的主机名。

注意

- 主机名可以是完全限定域名(FQDN)，格式为 ***hostname.domainname***，也可以是没有域的短主机名。许多网络具有动态主机配置协议(DHCP)服务，该服务自动为连接的系统提供域名。要允许 DHCP 服务为这个系统分配域名，请只指定简短主机名。
- 使用静态 IP 和主机名配置时，它取决于计划的系统用例，无论是使用短名称还是 FQDN。红帽身份管理在置备过程中配置 FQDN，但有些第三方软件产品可能需要短名称。在任何一种情况下，要确保在所有情况下两种形式都可用，请在 **/etc/hosts** 中为主机添加一个条目，格式为 **IP FQDN 短别名**。
- **localhost** 值意味着没有为目标系统配置特定的静态主机名，安装的系统的实际主机名是在处理网络配置的过程中配置的，例如，通过使用 DHCP 或 DNS 的 NetworkManager。
- 主机名只能包含字母数字字符和 **-** 或 **.**。主机名应等于或小于 64 个字符。主机名不能以 **-** 和 **.** 开头或结尾要兼容 DNS，FQDN 的每个部分都应等于或小于 63 个字符，以及 FQDN 总计长度（包括点数）不应超过 255 个字符。

8. 单击 **Apply**，将主机名应用到安装程序环境。

9. 或者，在 **Network and Hostname** 窗口中，您可以选择 Wireless 选项。单击右侧窗格中的 **Select network** 以选择您的 wifi 连接，如需要请输入密码，然后点击 **Done**。

其他资源

- 执行高级 RHEL 8 安装

1.2. 使用 NMCLI 配置以太网连接

如果您通过以太网将主机连接到网络，您可以使用 **nmcli** 工具在命令行上管理连接的设置。

先决条件

- 服务器配置中存在物理或虚拟以太网网络接口控制器(NIC)。

流程

1. 列出 NetworkManager 连接配置文件：

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

默认情况下，NetworkManager 为主机中的每个 NIC 创建一个配置文件。如果您计划仅将这个 NIC 连接到特定的网络，请调整自动创建的配置文件。如果您计划将这个 NIC 连接到具有不同设置的网络，请为每个网络创建单独的配置文件。

2. 如果要创建额外的连接配置文件，请输入：

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

跳过此步骤以修改现有配置文件。

3. 可选：重命名连接配置文件：

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

在具有多个配置文件的主机上，有意义的名称可以更容易识别配置文件的用途。

4. 显示连接配置文件的当前设置：

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

5. 配置 IPv4 设置：

- 要使用 DHCP，请输入：

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

如果 **ipv4.method** 已设置为 **auto**（默认），请跳过这一步。

- 要设置静态 IPv4 地址、网络掩码、默认网关、DNS 服务器和搜索域，请输入：

```
# 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. 配置 IPv6 设置：

- 要使用无状态地址自动配置(SLAAC)，请输入：

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

如果 **ipv6.method** 已设置为 **auto**（默认），请跳过这一步。

- 要设置静态 IPv6 地址、网络掩码、默认网关、DNS 服务器和搜索域，请输入：

```
# 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. 要在配置文件中自定义其他设置，请使用以下命令：

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

将值用空格或分号括在引号中。

8. 激活配置文件：

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

验证

1. 显示 NIC 的 IP 设置：

```
# 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. 显示 IPv4 默认网关：

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

3. 显示 IPv6 默认网关：

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

4. 显示 DNS 设置：

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

如果多个连接配置文件同时处于活动状态，则 **nameserver** 条目的顺序取决于这些配置文件中的 DNS 优先级值和连接类型。

5. 使用 **ping** 程序来验证这个主机是否可以向其它主机发送数据包：

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

故障排除

- 验证网线是否插入到主机和交换机。
- 检查链路失败是否只存在于此主机上，或者也存在于其它连接到同一交换机的主机上。
- 验证网络电缆和网络接口是否如预期工作。执行硬件诊断步骤并替换有缺陷的电缆和网络接口卡。
- 如果磁盘中的配置与设备中的配置不匹配，则启动或重启 NetworkManager 会创建一个代表该设备的配置的内存连接。有关详情以及如何避免这个问题，请参阅 NetworkManager 服务重启后会复制一个连接 解决方案。

其他资源

- **nm-settings(5)** 手册页

1.3. 使用 NMTUI配置以太网连接

如果通过以太网将主机连接到网络，您可以使用 **nmtui** 工具在基于文本的用户界面中管理连接的设置。使用 **nmtui** 创建新配置文件，并在没有图形界面的主机上更新现有配置文件。

注意

在 **nmtui** 中：

- 使用光标键导航。
- 选择按钮并按 **Enter** 键。
- 使用空格选择和 **清除复选框**。

先决条件

- 服务器配置中存在物理或虚拟以太网网络接口控制器(NIC)。

流程

1. 如果您不知道连接中使用的网络设备名称，显示可用的设备：

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

2. 启动 **nmtui** ：

```
# nmtui
```

3. 选择 **Edit a connection**，然后按 **Enter**。

4. 选择是否添加一个新连接配置文件或修改现有连接配置文件：

   - 要创建一个新配置文件：

     i. 按 **添加**。

     ii. 从网络类型列表中选择 **Ethernet**，然后按 **Enter** 键。

   - 要修改现有的配置文件，请从列表中选择配置文件，然后按 **Enter**。

5. 可选：更新连接配置文件的名称。
   在具有多个配置文件的主机上，有意义的名称可以更容易识别配置文件的用途。

6. 如果创建新连接配置文件，请在 **Device** 字段中输入网络设备名称。

7. 根据您的环境，相应地在 **IPv4 configuration** 和 **IPv6 configuration** 区中配置 IP 地址。为此，请按这些区域旁边的按钮，并选择：

   - **Disabled**，如果此连接不需要 IP 地址。

   - **Automatic**，如果 DHCP 服务器为这个 NIC 动态分配了一个 IP 地址。

   - **Manual**，如果网络需要静态 IP 地址设置。在这种情况下，您必须填写更多字段：

     i. 在您要配置的协议旁边按 **Show** 以显示其他字段。

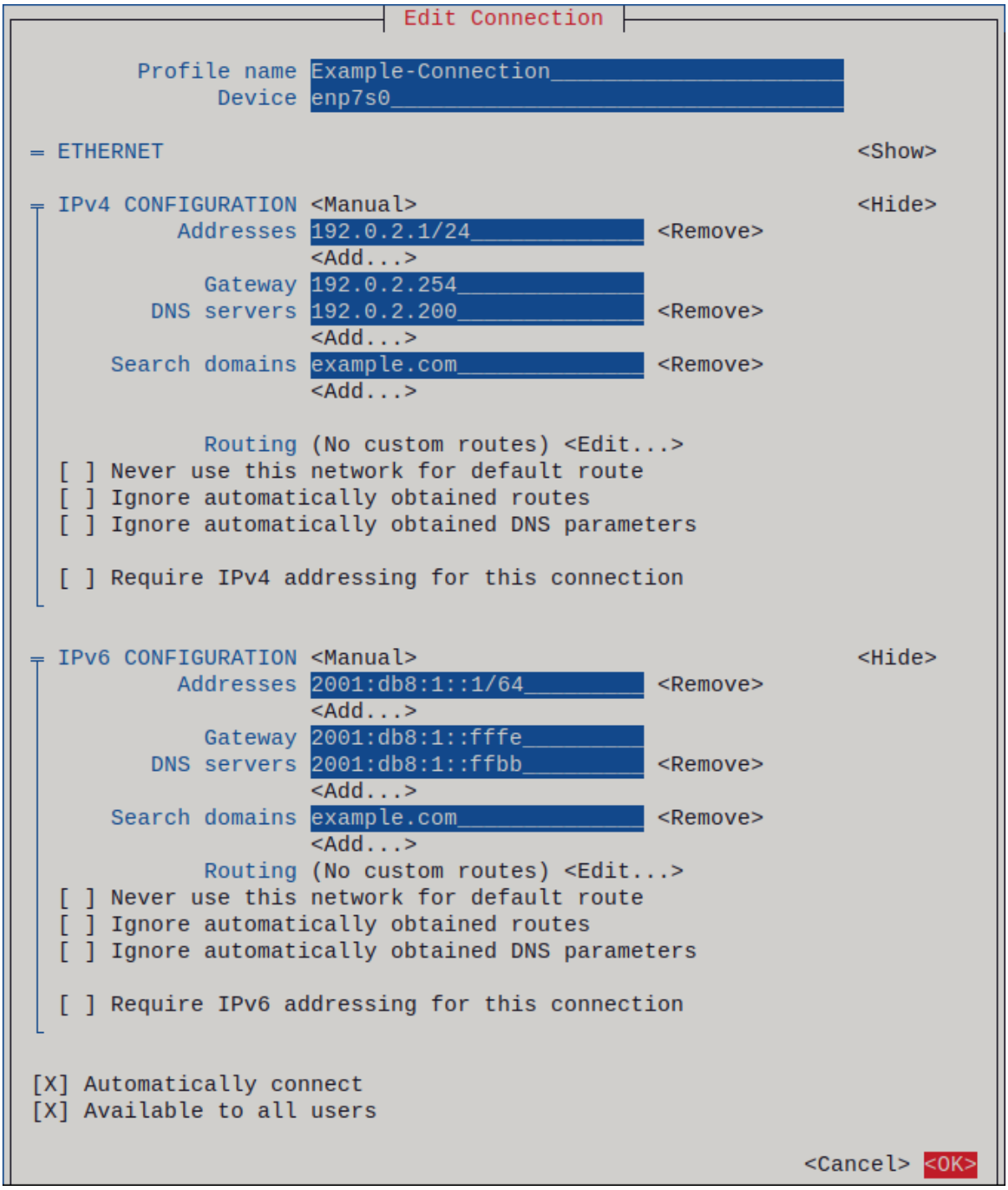
     ii. 按 **Addresses** 旁边的 **Add**，并以无类别域间路由(CIDR)格式输入 IP 地址和子网掩码。如果没有指定子网掩码，NetworkManager 会为 IPv4 地址设置 **/32** 子网掩码，并为 IPv6 地址设置 **/64**。

     iii. 输入默认网关的地址。

     iv. 按 **DNS 服务器** 旁边的 **Add**，并输入 DNS 服务器地址。

     v. 按 **搜索域** 旁边的 **Add**，并输入 DNS 搜索域。

图 1.1. 使用静态 IP 地址设置的以太网连接示例



8. 按 **OK** 创建并自动激活新连接。
9. 按 **Back** 返回到主菜单。
10. 选择 **Quit**，然后按 **Enter** 关闭 **nmtui** 应用程序。

验证

1. 显示 NIC 的 IP 设置：

```
# 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. 显示 IPv4 默认网关：

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

3. 显示 IPv6 默认网关：

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

4. 显示 DNS 设置：

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

   如果多个连接配置文件同时处于活动状态，则 **nameserver** 条目的顺序取决于这些配置文件中的 DNS 优先级值和连接类型。

5. 使用 **ping** 程序来验证这个主机是否可以向其它主机发送数据包：

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

故障排除

- 验证网线是否插入到主机和交换机。
- 检查链路失败是否只存在于此主机上，或者也存在于其它连接到同一交换机的主机上。
- 验证网络电缆和网络接口是否如预期工作。执行硬件诊断步骤并替换有缺陷的电缆和网络接口卡。
- 如果磁盘中的配置与设备中的配置不匹配，则启动或重启 NetworkManager 会创建一个代表该设备的配置的内存连接。有关详情以及如何避免这个问题，请参阅 NetworkManager 服务重启后会复制一个连接 解决方案。

其他资源

- 配置 NetworkManager 以避免使用特定配置集提供默认网关
- 配置 DNS 服务器顺序

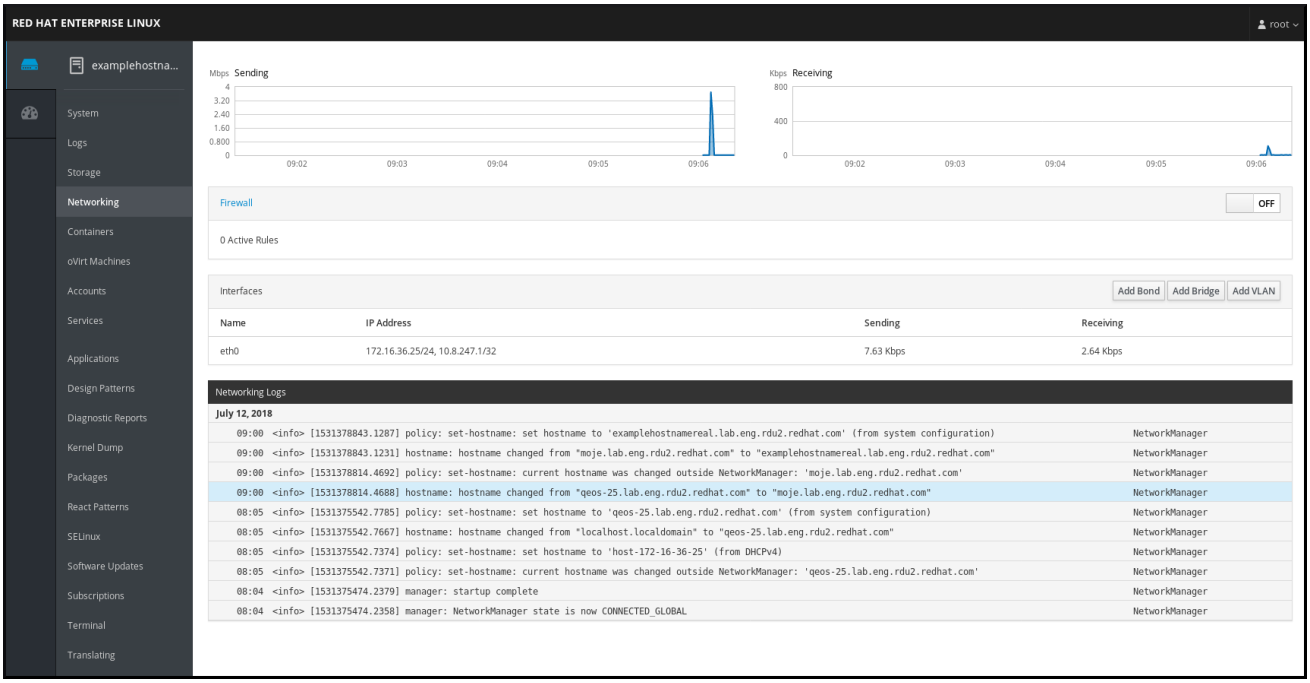
1.4. 在 RHEL WEB 控制台中管理网络

在 Web 控制台中，使用 **Networking** 菜单可以：

- 显示当前接收并发送的数据包

- 显示可用网络接口最重要的信息
- 显示网络日志的内容。
- 添加各种网络接口类型（bond、team、bridge、VLAN）

图 1.2. 在 RHEL web 控制台中管理网络



1.5. 使用 RHEL 系统角色管理网络

您可以使用 **network** 角色在多目标机器上配置网络连接。

network 角色可以配置以下类型的接口：

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

每个主机所需的网络连接都作为 **network_connections** 变量中的列表提供。

警告

network 角色更新或者创建目标系统中的所有连接配置集，与在 **network_connections** 变量中指定的方法完全相同。因此，如果选项只在系统中出现而没有出现在 **network_connections** 变量中，**network** 角色会从指定的配置集中删除选项。

以下示例演示了如何应用 **network** 角色来确保存在与所需参数的以太网连接：

应用网络角色的 playbook 示例，以设置使用所需参数的以太网连接

```
# 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
```

其他资源

- 准备一个控制节点和受管节点以使用 RHEL 系统角色

1.6. 其他资源

- 配置和管理网络

第 2 章 注册系统并管理订阅

订阅涵盖了安装在 Red Hat Enterprise Linux 上的产品，包括操作系统本身。

您可以使用 Red Hat Content Delivery Network 订阅来跟踪：

- 注册的系统
- 在您的系统中安装的产品
- 附加到安装产品的订阅

2.1. 安装后注册系统

如果您在安装过程中还没有注册系统，请使用以下步骤注册您的系统。

先决条件

- 红帽客户门户网站中的一个有效的用户帐户。
- 请参阅 创建红帽登录 页面。
- RHEL 系统的有效订阅。
- 有关安装过程的更多信息，请参阅 执行标准的 RHEL 8 安装。

流程

1. 注册并自动订阅您的系统。

```
# 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
```

该命令提示您输入您的红帽客户门户网站用户名和密码。

如果注册过程失败，您可以使用一个特定的池来注册您的系统。有关如何操作的指南，请执行以下步骤：

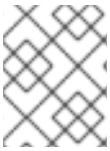
a. 确定您需要的订阅池 ID：

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

这个命令会显示您的红帽账户中的所有可用订阅。对于每个订阅，会显示各种相关信息，包括池 ID。

b. 通过使用上一步中决定的池 ID 替换 *pool_id* 来为您的系统附加适当的订阅：

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



注意

要将系统注册到 Red Hat Insights，您可以使用 **rhc connect** 工具。请参阅 设置远程主机配置。

其他资源

- 了解客户门户网站上的自动附加订阅
- 在客户门户网站中了解手动注册和订阅

2.2. 在 WEB 控制台中使用凭证注册订阅

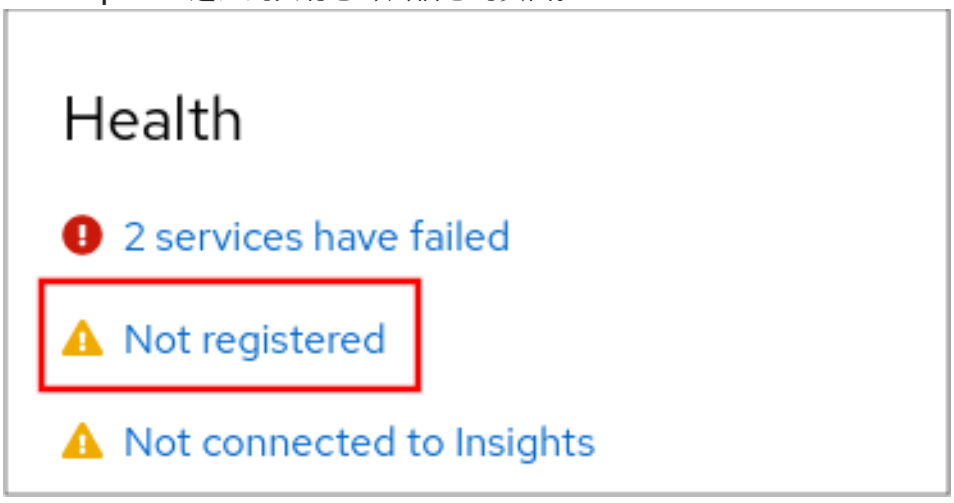
使用以下步骤通过 RHEL web 控制台使用帐户凭证注册新安装的 Red Hat Enterprise Linux。

先决条件

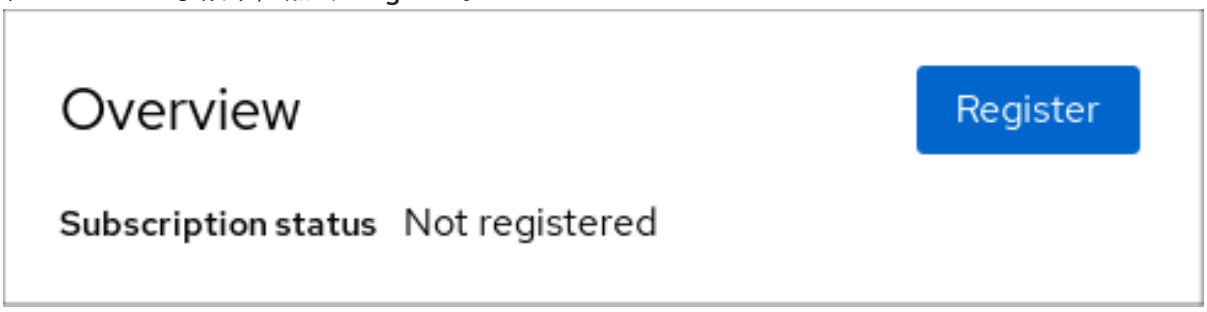
- 红帽客户门户网站中的有效用户帐户。
  请参阅 创建红帽登录 页面。
- RHEL 系统的有效订阅。

流程

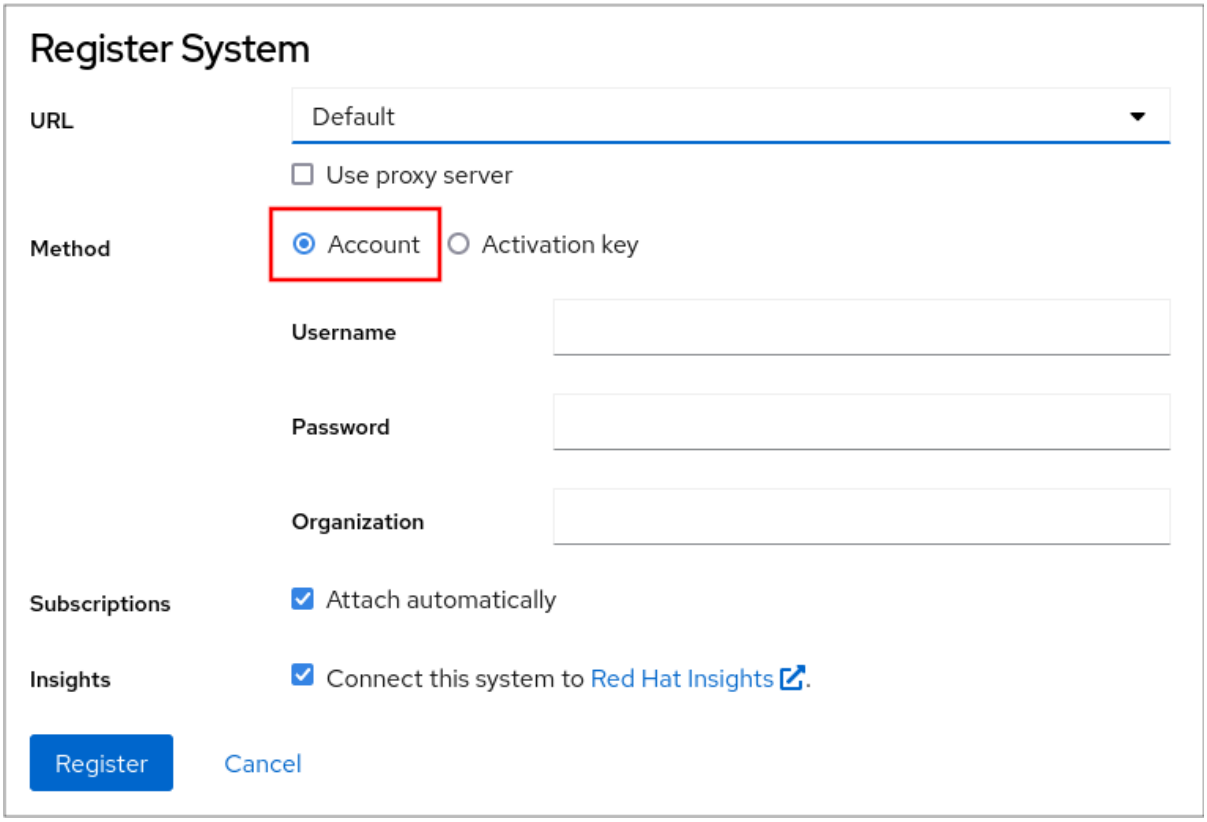
1. 登录到 RHEL web 控制台。详情请参阅 Web 控制台的日志记录。
2. 在 **Overview** 页面中的 **Health** 字段中，点击 **Not registered** 警告，或者点击主菜单中的 **Subscriptions** 进入到具有您订阅信息的页面。



3. 在 **Overview** 字段中，点击 **Register**。



4. 在 **Register system** 对话框中，选择您要使用您的帐户凭证进行注册的系统。



5. 输入您的用户名。
6. 输入您的密码。
7. （可选）输入您的机构名称或 ID。
   如果您的帐户属于红帽客户门户网站中的多个机构，您必须添加机构名称或机构 ID。要获得机构 ID，请联系您的红帽相关人员。
   - 如果您不想将您的系统连接到 Red Hat Insights，请清除 **Insights** 复选框。
8. 点击 **Register** 按钮。

此时您的 Red Hat Enterprise Linux Enterprise Linux 系统已被成功注册。

2.3. 在 GNOME 中使用红帽帐户注册系统

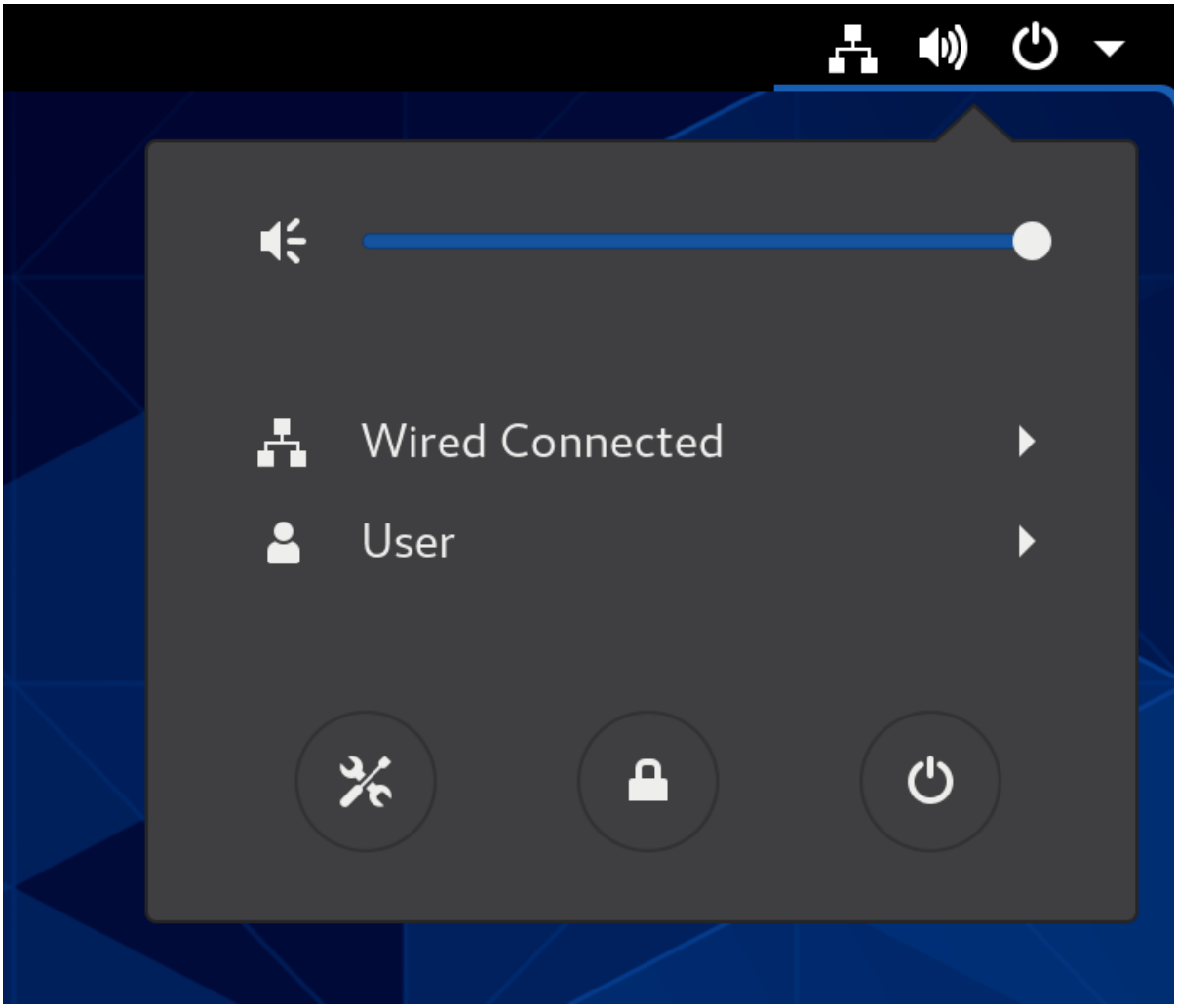
按照以下步骤将您的系统注册到您的红帽帐户中。

先决条件

- 红帽客户门户网站中的有效帐户。
  对于新用户注册的详情，请参阅创建红帽登陆页。

流程

1. 打开 **系统菜单**，该菜单可从右上角访问，然后单击 **Settings** 图标。



2. 在 Details → About 部分，点 **Register**。
3. 选择 **Registration Server**。
4. 如果没有使用红帽服务器，在 **URL** 项中输入服务器地址。
5. 在 **Registration Type** 菜单中选 **Red Hat Account**。
6. 在 **Registration Details** 中：
   - 在 **Login** 字段中输入您的红帽帐户用户名。
   - 在 **Password** 字段中输入您的红帽帐户密码。
   - 在 **Organization** 项中输入您的机构名称。
7. 点 **Register**。

2.4. 在 GNOME 中使用激活码注册系统

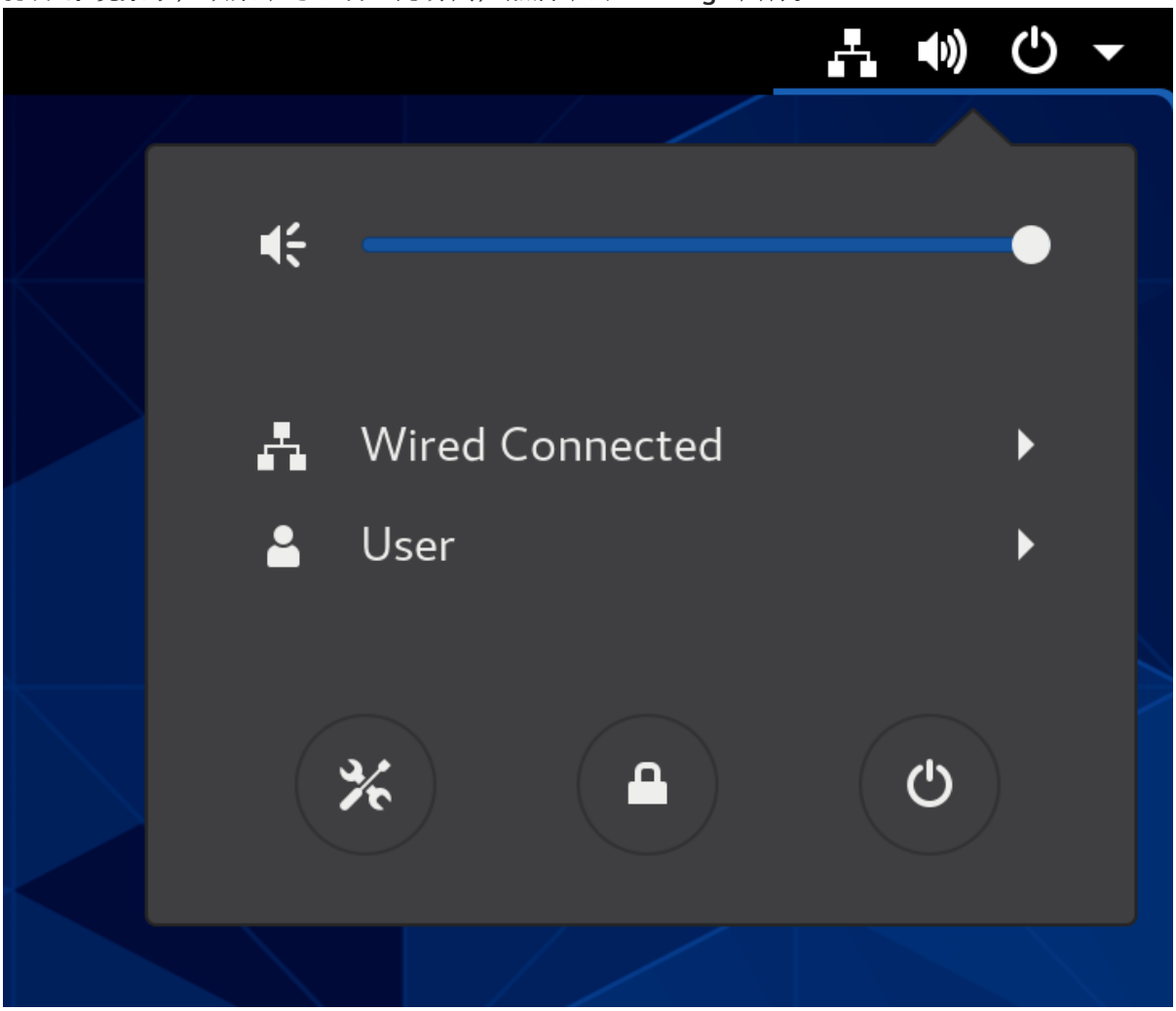
按照以下步骤，使用激活码注册您的系统。您可从您的机构管理员获得激活码。

先决条件

- 激活码。
  有关生成新激活键的详情，查看 Activation Keys 页。

流程

1. 打开 **系统菜单**，该菜单可从右上角访问，然后单击 **Settings** 图标。



2. 在 **Details → About** 部分，点 **Register**。
3. 选择 **Registration Server**。
4. 如果没有使用红帽服务器，在 **URL** 项中输入服务器地址。
5. 在 **Registration Type** 菜单中选 **Activation Keys**。
6. 在 **Registration Details** 中：
   - 在 **Activation Keys** 字段中输入您的激活码。
     以逗号 (,) 分隔您的激活码。
   - 在 **Organization** 字段中输入您的机构名称或者 ID。
7. 点 **Register**。

2.5. 使用安装程序 GUI 注册 RHEL 8

您可以使用 RHEL 安装程序 GUI 注册 Red Hat Enterprise Linux 8。

先决条件

- 您在红帽客户门户网站中有一个有效的用户帐户。请参阅 创建一个红帽登录页面。
- 您有一个有效的激活码和机构 ID。

流程

1. 从 **Installation Summary** 屏幕，在 **Software** 下，点击 **Connect to Red Hat**。
2. 使用 **帐户**或 **激活密码** 选项验证您的红帽帐户。
3. 可选：在 **Set System Purpose** 字段中，选择您要从下拉菜单中设置的 **Role**、**SLA** 和 **Usage** 属性。
   此时您的 Red Hat Enterprise Linux 8 系统已被成功注册。

第 3 章 获得红帽支持

本节介绍了如何使用红帽支持有效解决问题以及 **sosreport**。

要获得红帽支持，请使用 红帽客户门户网站，它提供对您订阅所有可用资源的访问。

3.1. 通过红帽客户门户网站获得红帽支持

下面的部分论述了如何使用红帽客户门户网站获得帮助。

先决条件

- 红帽客户门户网站中的有效用户帐户。请参阅 创建红帽登录帐号。
- RHEL 系统的有效订阅。

流程

1. 访问 红帽支持:
   a. 创建新的支持问题单。
   b. 与红帽支持专家启动实时聊天。
   c. 通过致电或发送电子邮件与红帽专家联系。

3.2. 使用 SOSREPORT 进行故障排除

sosreport 命令收集 Red Hat Enterprise Linux 系统的配置详情、系统信息和诊断信息。

以下小节论述了如何使用 **sosreport** 命令为您的支持问题单生成报告。

先决条件

- 红帽客户门户网站中的有效用户帐户。请参阅 创建红帽登录帐号。
- RHEL 系统的有效订阅。
- 支持问题单号。

流程

1. 安装 **sos** 软件包：

```
# yum install sos
```

注意

Red Hat Enterprise Linux 的默认最小安装不包括 **sos** 软件包，该软件包提供 **sosreport** 命令。

2. 生成一个报告：

```
# sosreport
```

3. 将报告附加到您的支持问题单中。
   请参阅 如何将文件附加到红帽支持问题单？更多信息请参阅红帽知识库文章。

   请注意，在附加报告时会提示您输入相关问题单的号码。

其他资源

- 什么是 sosreport 以及如何在 Red Hat Enterprise Linux 中创建一个？

第 4 章 更改基本环境设置

配置基本环境设置是安装过程的一部分。以下部分介绍了在稍后修改时的信息。环境的基本配置包括：

- 日期和时间
- 系统区域设置
- 键盘布局
- 语言

4.1. 配置日期和时间

因为以下原因，准确计时非常重要。在 Red Hat Enterprise Linux 中，时间保持是由**NTP** 协议来保证的，该协议由一个运行在用户空间的守护进程来实现。user-space 守护进程更新内核中运行的系统时钟。系统时钟可以通过使用不同的时钟源来维护系统的时间。

Red Hat Enterprise Linux 8 使用 **chronyd** 守护进程来实现 **NTP**。**chronyd** 包括在 **chrony** 软件包中。如需更多信息，请参阅使用 chrony 来配置 NTP。

4.1.1. 显示当前日期和时间

要显示当前日期和时间，请使用这些步骤之 一。

流程

1. 输入 **date** 命令：

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

2. 要查看更多详细信息，请使用 **timedatectl** 命令：

```
$ 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
```

其他资源

- 使用 Web 控制台配置时间设置
- **man date(1)** 和 **man timedatectl(1)**

4.2. 配置系统区域设置

系统范围的区域设置存储在 **/etc/locale.conf** 文件中，其在早期引导时被 **systemd** 守护进程读取。每个服务或用户都会继承在 **/etc/locale.conf** 中配置的 locale 设置，单独程序或个人用户可以单独覆盖它们。

流程

- 要列出系统可用区域设置，请执行以下操作：

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

- 显示系统区域设置的当前状态：

```
$ localectl status
```

- 要设置或更改默认的系统区域设置，请使用 **localectl set-locale** 子命令（使用 **root** 用户）。例如：

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

其他资源

- **man localectl(1)**、**man locale(7)** 和 **man locale.conf(5)**

4.3. 配置键盘布局

键盘布局设置控制文本控制台和图形用户界面中的布局。

流程

- 要列出可用的键映射：

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

- 显示 keymaps 设置的当前状态：

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

- 要设置或更改默认系统 keymap ：例如：

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

其他资源

- **man localectl(1)**、**man locale(7)** 和 **man locale.conf(5)**

4.4. 在文本控制台模式下更改字体大小

您可以使用 **setfont** 命令更改虚拟控制台中的字体大小。

- 输入带有字体名称的 **setfont** 命令，例如：

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

注意

setfont 命令默认搜索多个硬编码的路径。因此，**setfont** 不需要全名和字体的路径。

- 要水平和垂直地将字体大小增加一倍，请输入带有 **-d** 参数的 **setfont** 命令：

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

注意

可以增加一倍的最大字体大小为 16x16 像素。

- 要在系统重启过程中保留所选字体，请在 **/etc/vconsole.conf** 文件中使用 **FONT** 变量，例如：

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

- 您可以在 **kbd-misc** 软件包中找到各种字体，该软件包是使用'kbd' 软件包安装的。例如，字体 **LatArCyrHeb** 有很多变体：

```
# 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
```

注意

虚拟控制台支持的最大字体大小为 32 像素。您可以通过对控制台使用较小的分辨率来减少字体可读性问题。

4.5. 其他资源

- 执行标准的 RHEL 8 安装

第 5 章 使用 OPENSSH 的两个系统间使用安全通讯

SSH(Secure Shell)是一种协议，它使用客户端-服务器架构在两个系统之间提供安全通信，并允许用户远程登录到服务器主机系统。和其它远程沟通协议，如 FTP 或 Telnet 不同，SSH 会加密登录会话，它会阻止入侵者从连接中收集未加密的密码。

Red Hat Enterprise Linux 包括基本的 **OpenSSH** 软件包：常规 **openssh** 软件包、**openssh-server** 软件包和 **openssh-clients** 软件包。请注意，**OpenSSH** 软件包需要 **OpenSSL** 软件包 **openssl-libs**，它会安装几个重要的加密库来启用 **OpenSSH** 对通讯进行加密。

5.1. SSH 和 OPENSSH

SSH（安全 Shell）是一个登录远程机器并在该机器上执行命令的程序。SSH 协议通过不安全的网络在两个不可信主机间提供安全加密的通讯。您还可以通过安全频道转发 X11 连接和任意 TCP/IP 端口。

当使用 SSH 协议进行远程 shell 登录或文件复制时，SSH 协议可以缓解威胁，例如，拦截两个系统之间的通信和模拟特定主机。这是因为 SSH 客户端和服务器使用数字签名来验证其身份。另外，所有客户端和服务器系统之间的沟通都是加密的。

主机密钥验证使用 SSH 协议的主机。当首次安装 OpenSSH 或主机第一次引导时，主机密钥是自动生成的加密密钥。

OpenSSH 是 Linux、UNIX 和类似操作系统支持的 SSH 协议的实现。它包括 OpenSSH 客户端和服务器需要的核心文件。OpenSSH 组件由以下用户空间工具组成：

- **ssh** 是一个远程登录程序（SSH 客户端）.
- **sshd** 是一个 OpenSSH SSH 守护进程。
- **scp** 是一个安全的远程文件复制程序。
- **sftp** 是一个安全的文件传输程序。
- **ssh-agent** 是用于缓存私钥的身份验证代理。
- **ssh-add** 为 **ssh-agent**添加私钥身份。
- **ssh-keygen** 生成、管理并转换 **ssh** 验证密钥。
- **ssh-copy-id** 是一个将本地公钥添加到远程 SSH 服务器上的 **authorized_keys** 文件中的脚本。
- **ssh-keyscan** 可以收集 SSH 公共主机密钥。

RHEL 中的 OpenSSH 套件仅支持 SSH 版本 2。它有一个增强的密钥交换算法，其不会受到较旧版本 1 中已知的漏洞的影响。

OpenSSH 作为 RHEL 的核心加密子系统之一，使用系统范围的加密策略。这样可确保在默认配置中禁用弱密码套件和加密算法。要修改策略，管理员必须使用 **update-crypto-policies** 命令来调整设置，或者手动选择不使用系统范围的加密策略。

OpenSSH 套件使用两组配置文件：一个用于客户端程序（即 **ssh**、**scp** 和 **sftp**），另一个用于服务器（**sshd** 守护进程）。

系统范围的 SSH 配置信息保存在 **/etc/ssh/** 目录中。用户特定的 SSH 配置信息保存在用户主目录中的 **~/.ssh/** 中。有关 OpenSSH 配置文件的详细列表，请查看 **sshd(8)** man page 中的 **FILES** 部分。

其他资源

其他资源

- 使用 **man -k ssh** 命令显示 man page
- 使用系统范围的加密策略

5.2. 配置并启动 OPENSSH 服务器

您可以更改 OpenSSH 服务器的欢迎横幅和 IP 地址。您还可以切换到较慢的动态网络配置。

请注意，在默认 RHEL 安装后，**sshd** 守护进程已经启动，服务器主机密钥会被自动创建。

先决条件

- 已安装 **openssh-server** 软件包。

流程

1. 如果 **sshd** 服务还没有运行，请在当前会话中启动它，并将其设置为在引导时自动启动：

```
# systemctl enable --now sshd
```

2. 要为 **/etc/ssh/sshd_config** 配置文件中的 **ListenAddress** 指令指定默认地址 **0.0.0.0**（IPv4）或 **::**（IPv6），并使用较慢的动态网络配置，将 **network-online.target** 目标单元的依赖关系添加到 **sshd.service** 单元文件中。要做到这一点，使用以下内容创建 **/etc/systemd/system/sshd.service.d/local.conf** 文件：

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

3. 查看 **/etc/ssh/sshd_config** 配置文件中的 OpenSSH 服务器设置是否满足您的情况要求。

4. 另外，还可通过编辑 **/etc/issue** 文件来更改您的 OpenSSH 服务器在客户端验证前显示的欢迎信息，例如：

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

确保 **/etc/ssh/sshd_config** 中未注释掉 **Banner** 选项，并且其值包含 **/etc/issue** ：

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

请注意：要在成功登录后改变显示的信息，您必须编辑服务器上的 **/etc/motd** 文件。详情请查看 **pam_motd** man page。

5. 重新载入 **systemd** 配置，并重启 **sshd** 以应用修改：

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

验证

1. 检查 **sshd** 守护进程是否正在运行：

```
# 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. 使用 SSH 客户端连接到 SSH 服务器。

```
# 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:
```

其他资源

- **sshd(8)** 和 **sshd_config(5)** 手册页。

5.3. 为基于密钥的身份验证设置 OPENSSH 服务器

要提高系统安全性，通过在 OpenSSH 服务器上禁用密码身份验证来强制进行基于密钥的身份验证。

先决条件

- 已安装 **openssh-server** 软件包。
- **sshd** 守护进程正在服务器中运行。

流程

1. 在文本编辑器中打开 **/etc/ssh/sshd_config** 配置，例如：

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

2. 将 **PasswordAuthentication** 选项改为 **no**:

```
PasswordAuthentication no
```

在新默认安装以外的系统中，检查 **PubkeyAuthentication** 没有被设置，并且将 **ChallengeResponseAuthentication** 指令设为 **no**。如果您要进行远程连接，而不使用控制台或带外访问，在禁用密码验证前测试基于密钥的登录过程。

3. 要在 NFS 挂载的主目录中使用基于密钥的验证，启用 **use_nfs_home_dirs** SELinux 布尔值：

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

4. 重新载入 **sshd** 守护进程以应用更改：

```
# systemctl reload sshd
```

其他资源

- **sshd(8)**, **sshd_config(5)** 和 **setsebool(8)** 手册页。

5.4. 生成 SSH 密钥对

您可以通过在本地系统上生成 SSH 密钥对并将生成的公钥复制到 OpenSSH 服务器来在没有提供密码的情况下登录到 OpenSSH 服务器。必须将服务器配置为允许此选项。

先决条件

- 您以 Linux 用户身份登录，该用户将连接到 OpenSSH 服务器。如果以 **root** 身份完成以下步骤，则只有 **root** 用户才能使用该密钥。

流程

1. 为 SSH 协议的版本 2 生成 ECDSA 密钥对：

```
$ ssh-keygen -t ecdsa
Generating public/private ecdsa key pair.
Enter file in which to save the key (/home/<username>/.ssh/id_ecdsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/<username>/.ssh/id_ecdsa.
Your public key has been saved in /home/<username>/.ssh/id_ecdsa.pub.
The key fingerprint is:
SHA256:Q/x+qms4j7PCQ0qFd09iZEFHA+SqwBKRNaU72oZfaCI
<username>@<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]-----+
```

您还可以通过输入 **ssh-keygen -t ed25519** 命令，在 **ssh-keygen** 命令或 Ed25519 密钥对中使用 **-t rsa** 选项生成 RSA 密钥对。

2. 将公钥复制到远程机器中：

```
$ ssh-copy-id <username>@<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
<username>@<ssh-server-example.com>'s password:
...
Number of key(s) added: 1

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

将 & ***lt;username> @ <ssh-server-example.com&*** gt; 替换为您的凭证。

如果您没有在会话中使用 **ssh-agent** 程序，上一个命令会复制最新修改的 **~/.ssh/id*.pub** 公钥。要指定另一个公钥文件，或在 **ssh-agent** 内存中缓存的密钥优先选择文件中的密钥，使用带有 **-i** 选项的 **ssh-copy-id** 命令。

3. 可选：要在重新安装系统之间保留之前生成的密钥对，备份 **~/.ssh/** 目录。重新安装后，将其复制到主目录中。您可以为系统中的所有用户（包括 **root** 用户）进行此操作。

验证

1. 在不提供任何密码的情况下登录到 OpenSSH 服务器：

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

其他资源

- **ssh-keygen(1)** 和 **ssh-copy-id(1)** 手册页。

5.5. 使用保存在智能卡中的 SSH 密钥

您可以使用保存在 OpenSSH 客户端智能卡中的 RSA 和 ECDSA 密钥。使用智能卡进行验证替换了默认密码验证。

先决条件

- 在客户端中安装了 **opensc** 软件包，**pcscd** 服务正在运行。

流程

1. 列出所有由 OpenSC PKCS #11 模块提供的密钥，包括其 PKCS #11 URIs，并将输出保存到 *key.pub* 文件：

```
$ 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. 要使用远程服务器上的智能卡（*example.com*）启用验证，将公钥传送到远程服务器。使用带有上一步中创建的 *key.pub* 的 **ssh-copy-id** 命令：

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

3. 要使用在第 1 步的 **ssh-keygen -D** 命令输出中的 ECDSA 密钥连接到 *example.com*，您只能使用 URI 中的一个子集，它是您的密钥的唯一参考，例如：

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

4. 您可以使用 **~/.ssh/config** 文件中的同一 URI 字符串使配置持久：

```
$ 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] $
```

因为 OpenSSH 使用 **p11-kit-proxy** 包装器，并且 OpenSC PKCS #11 模块是注册到 PKCS#11 Kit 的，所以您可以简化前面的命令：

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

如果您跳过 PKCS #11 URI 的 **id=** 部分，则 OpenSSH 会加载代理模块中可用的所有密钥。这可减少输入所需的数量：

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

其它资源

- Fedora 28:在 OpenSSH 中提供更强大的智能卡
- **p11-kit (8)**, **opensc.conf (5)**, **pcscd (8)**, **ssh (1)**, 和 **ssh-keygen (1)** man page

5.6. 使 OPENSSH 更安全

在使用 OpenSSH 时，您可以调整系统以提高安全性。

请注意，**/etc/ssh/sshd_config** OpenSSH 配置文件的更改需要重新载入 **sshd** 守护进程才能生效：

```
# systemctl reload sshd
```

-

警告

大多数安全强化配置更改会降低与不支持最新算法或密码套件的客户端的兼容性。

禁用不安全的连接协议

- 要使 SSH 生效，防止使用由 OpenSSH 套件替代的不安全连接协议。否则，用户的密码可能只会在一个会话中被 SSH 保护，可能会在以后使用 Telnet 登录时被捕获。因此，请考虑禁用不安全的协议，如 telnet、rsh、rlogin 和 ftp。

启用基于密钥的身份验证并禁用基于密码的身份验证

- 禁用密码验证并只允许密钥对可减少安全攻击面，还可节省用户的时间。在客户端中，使用 **ssh-keygen** 工具生成密钥对，并使用 **ssh-copy-id** 工具从 OpenSSH 服务器的客户端复制公钥。要在 OpenSSH 服务器中禁用基于密码的验证，请编辑 **/etc/ssh/sshd_config**，并将 **PasswordAuthentication** 选项改为 **no**:

```
PasswordAuthentication no
```

密钥类型

- 虽然 **ssh-keygen** 命令会默认生成一组 RSA 密钥，但您可以使用 **-t** 选项指定它生成 ECDSA 或者 Ed25519 密钥。ECDSA(Elliptic Curve Digital Signature Algorithm)能够在同等的对称密钥强度下，提供比 RSA 更好的性能。它还会生成较短的密钥。Ed25519 公钥算法是 一种变形的 Edwards 曲线的实现，其比 RSA、DSA 和 ECDSA 更安全，也更快。
  如果没有这些密钥，OpenSSH 会自动创建 RSA、ECDSA 和 Ed25519 服务器主机密钥。要在 RHEL 中配置主机密钥创建，请使用 **sshd-keygen@.service** 实例化服务。例如，禁用自动创建 RSA 密钥类型：

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

注意

在启用了 **cloud-init** 的镜像中，**ssh-keygen** 单元会自动禁用。这是因为 **ssh-keygen 模板** 服务可能会干扰 **cloud-init** 工具，并导致主机密钥生成问题。要防止这些问题 **etc/systemd/system/sshd-keygen@.service.d/disable-sshd-keygen-if-cloud-init-active.conf** drop-in 配置文件禁用 **ssh-keygen** 单元（如果 **cloud-init** 正在运行）。

- 要排除 SSH 连接的特定密钥类型，注释 **/etc/ssh/sshd_config** 中的相关行，并重新载入 **sshd** 服务。例如，只允许 Ed25519 主机密钥：

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

重要

Ed25519 算法不符合 FIPS-140，OpenSSH 在 FIPS 模式下无法使用 Ed25519 密钥。

非默认端口

- 默认情况下，**sshd** 守护进程侦听 TCP 端口 22。更改端口可降低系统因自动网络扫描而受到攻击的风险，从而提高安全性。您可以使用 **/etc/ssh/sshd_config** 配置文件中的 **Port** 指令指定端口。
  您还必须更新默认 SELinux 策略以允许使用非默认端口。要做到这一点，使用 **policycoreutils-python-utils** 软件包中的 **semanage** 工具：

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

  另外，更新 **firewalld** 配置：

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

  在前面的命令中，将 *<port_number>* 替换为使用 **Port** 指令指定的新端口号。

没有 root 登录

- 如果您的特定用例不需要以 root 用户身份登录，您可以在 **/etc/ssh/sshd_config** 文件中将 **PermitRootLogin** 配置指令设置为 **no**。通过禁止以 root 用户身份登录，管理员可以审核哪些用户在以普通用户身份登录后运行了哪些特权命令，然后获得了 root 权限。
  或者，将 **PermitRootLogin** 设置为 **prohibit-password**：

```
PermitRootLogin prohibit-password
```

  这强制使用基于密钥的身份验证，而不是使用密码以 root 身份登录，并通过防止暴力攻击来降低风险。

使用 X 安全性扩展

- Red Hat Enterprise Linux 客户端中的 X 服务器不提供 X 安全性扩展。因此，当连接到带有 X11 转发的不可信 SSH 服务器时，客户端无法请求另一个安全层。大多数应用程序都无法在启用此扩展时运行。
  默认情况下，**/etc/ssh/ssh_config.d/05-redhat.conf** 文件中的 **ForwardX 11Trusted** 选项被设置为 **yes**，且 **ssh -X remote_machine**（不信任主机）和 **ssh -Y remote_machine**（可信主机）命令之间没有区别。

  如果您的场景根本不需要 X11 转发功能，请将 **/etc/ssh/sshd_config** 配置文件中的 **X11Forwarding** 指令设置为 **no**。

限制对特定用户、组群或者域的访问

- **/etc/ssh/sshd_config** 配置文件服务器中的 **AllowUsers** 和 **AllowGroups** 指令可让您只允许某些用户、域或组连接到您的 OpenSSH 服务器。您可以组合 **AllowUsers** 和 **Allow Groups** 来更准确地限制访问，例如：

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

以上配置行接受来自 192.168.1.* 和 10.0.0.* 子网中所有用户的连接，但 192.168.1.2 地址的系统除外。所有用户都必须在 **example-group** 组中。OpenSSH 服务器拒绝所有其他连接。

OpenSSH 服务器仅允许通过 **/etc/ssh/sshd_config** 中所有 Allow 和 Deny 指令的连接。例如，如果 **AllowUsers** 指令列出的用户不是 **AllowGroups** 指令中列出的组的一部分，则用户无法登录。

请注意，使用允许列表（以 Allow 开头的指令）比使用阻止列表（以 Deny 开始的选项）更安全，因为允许列表也会阻止新的未授权的用户或组。

更改系统范围的加密策略

- OpenSSH 使用 RHEL 系统范围的加密策略，默认的系统范围的加密策略级别为当前威胁模型提供了安全设置。要使您的加密设置更严格，请更改当前的策略级别：

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

警告

如果您的系统在互联网上进行通信，则可能会因为 **FUTURE** 策略的严格设置而面临互操作性问题。

您还可以只通过系统范围的加密策略为 SSH 协议禁用特定的密码。如需更多信息，请参阅 安全强化 文档中的 使用子策略自定义系统范围的加密策略 部分。

要为您的 OpenSSH 服务器选择不使用系统范围的加密策略，请在 **/etc/sysconfig/sshd** 文件中取消具有 **CRYPTO_POLICY=** 变量的行的注释。更改后，您在 **/etc/ssh/sshd_config** 文件中的 **Ciphers**、**MAC**、**KexAlgoritms** 和 **GSSAPIKexAlgorithms** 部分指定的值不会被覆盖。

详情请查看 **sshd_config(5)** 手册页。

要为您的 OpenSSH 客户端选择不使用系统范围的加密策略，请执行以下任务之一：

- 对于给定的用户，使用 **~/.ssh/config** 文件中特定于用户的配置覆盖全局 **ssh_config**。
- 对于整个系统，在 **/etc/ssh/ssh_config.d/** 目录中的置入配置文件中指定加密策略，使用小于 5 的两位数字前缀，以便其在字典顺序上位于 **05-redhat.conf** 文件之前，并带有 **.conf** 后缀，例如 **04-crypto-policy-override.conf**。

其他资源

- **sshd_config(5)**、**ssh-keygen(1)**、 **crypto-policies(7)** 和 **update-crypto-policies(8)** 手册页。
- 安全强化 文档中的 使用系统范围的加密策略 。
- 如何只为 ssh 服务禁用特定的算法和密码 文章。

5.7. 使用 SSH 跳过主机连接到远程服务器

您可以通过中间服务器（也称为跳过主机）将本地系统连接到远程服务器。

先决条件

- 跳过主机接受来自本地系统的 SSH 连接。
- 远程服务器只接受来自跳过主机的 SSH 连接。

流程

1. 通过编辑本地系统中的 **~/.ssh/config** 文件来定义跳板主机，例如：

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

   - **Host** 参数定义您可以在 **ssh** 命令中使用的主机的名称或别名。该值可以匹配真实的主机名，但也可以是任意字符串。
   - **HostName** 参数设置跳过主机的实际主机名或 IP 地址。

2. 使用 **ProxyJump** 指令将远程服务器跳板配置添加到本地系统上的 **~/.ssh/config** 文件中，例如：

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

3. 使用您的本地系统通过跳过服务器连接到远程服务器：

```
$ ssh remote-server
```

   如果省略了配置步骤 1 和 2，则上一命令等同于 **ssh -J skip-server1 remote-server** 命令。

4. 您可以指定更多的跳板服务器，您也可以在提供其完整主机名时跳过在配置文件中添加主机定义，例如：

```
$ ssh -J jump1.example.com,jump2.example.com,jump3.example.com remote1.example.com
```

   如果跳板服务器上的用户名或 SSH 端口与远程服务器上的用户名和端口不同，请只修改上一命令中的主机名表示法，例如：

```
$ ssh -J
johndoe@jump1.example.com:75,johndoe@jump2.example.com:75,johndoe@jump3.example
.com:75 joesec@remote1.example.com:220
```

其他资源

- **ssh_config(5)** 和 **ssh(1)** 手册页.

5.8. 通过 SSH-AGENT，使用 SSH 密钥连接到远程机器

为了避免在每次发起 SSH 连接时输入密语，您可以使用 **ssh-agent** 工具缓存 SSH 私钥。确保私钥和密语安全。

先决条件

- 您有一个运行 SSH 守护进程的远程主机，并且可通过网络访问。
- 您知道登录到远程主机的 IP 地址或者主机名以及凭证。
- 您已用密码生成了 SSH 密钥对，并将公钥传送到远程机器。

如需更多信息，请参阅 生成 SSH 密钥对。

流程

1. 可选：验证您是否可以使用密钥来对远程主机进行身份验证：
   a. 使用 SSH 连接到远程主机：

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

   b. 输入您在创建密钥时设定的密码短语以授予对私钥的访问权限。

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

2. 启动 **ssh-agent**。

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

3. 将密钥添加到 **ssh-agent**。

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

验证

- 可选：使用 SSH 登录到主机机器。

  ```
  $ ssh example.user1@198.51.100.1

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

  请注意您不必输入密码短语。

5.9. 配置与 SSH 系统角色的安全通信

作为管理员，您可以使用 **sshd** 系统角色配置 SSH 服务器和 **ssh** 系统角色，以使用 Red Hat Ansible Automation Platform 同时在任意数量的 RHEL 系统上配置 SSH 客户端。

5.9.1. sshd RHEL 系统角色的变量

在 **sshd** 系统角色 playbook 中，您可以根据您的偏好和限制为 SSH 配置文件定义参数。

如果没有配置这些变量，系统角色会生成一个与 RHEL 默认值匹配的 **sshd_config** 文件。

在所有情况下，布尔值在 **sshd** 配置中都正确呈现为 **yes** 和 **no**。您可以使用 list 来定义多行配置项。例如：

```
sshd_ListenAddress:
  - 0.0.0.0
  - '::'
```

呈现为：

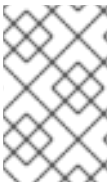
```
ListenAddress 0.0.0.0
ListenAddress ::
```

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.sshd/README.md** 文件
- **/usr/share/doc/rhel-system-roles/sshd/** 目录

5.9.2. 使用 sshd RHEL 系统角色配置 OpenSSH 服务器

您可以通过运行 Ansible playbook，使用 **sshd** RHEL 系统角色来配置多个 SSH 服务器。



注意

您可以将 **sshd** RHEL 系统角色用于更改 SSH 和 SSHD 配置的其他 RHEL 系统角色，如身份管理 RHEL 系统角色。要防止配置被覆盖，请确保 **sshd** 角色使用命名空间(RHEL 8 和更早的版本)或 drop-in 目录(RHEL 9)。

先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。

流程

1. 创建一个包含以下内容的 playbook 文件，如 ~/**playbook.yml** ：

```
---
- name: SSH server configuration
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure sshd to prevent root and password login except from particular subnet
      ansible.builtin.include_role:
        name: rhel-system-roles.sshd
      vars:
        sshd:
          PermitRootLogin: no
```

```
        PasswordAuthentication: no
        Match:
          - Condition: "Address 192.0.2.0/24"
            PermitRootLogin: yes
            PasswordAuthentication: yes
```

playbook 将受管节点配置为 SSH 服务器，以便：

- 禁用密码和 **root** 用户登录
- 只对子网 **192.0.2.0/24** 启用密码和 **root** 用户登录

2. 验证 playbook 语法：

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

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

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

验证

1. 登录到 SSH 服务器：

```
$ ssh <username>@<ssh_server>
```

2. 验证 SSH 服务器上 **sshd_config** 文件的内容：

```
$ cat /etc/ssh/sshd_config
...
PasswordAuthentication no
PermitRootLogin no
...
Match Address 192.0.2.0/24
  PasswordAuthentication yes
  PermitRootLogin yes
...
```

3. 检查您是否可以以 root 用户身份从 **192.0.2.0/24** 子网连接到服务器：

   a. 确定您的 IP 地址：

   ```
   $ hostname -I
   192.0.2.1
   ```

   如果 IP 地址在 **192.0.2.1** - **192.0.2.254** 范围内，您可以连接到服务器。

   b. 以 **root** 用户身份连接到服务器：

   ```
   $ ssh root@<ssh_server>
   ```

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.sshd/README.md** 文件
- **/usr/share/doc/rhel-system-roles/sshd/** 目录

5.9.3. ssh RHEL 系统角色的变量

在 **ssh** 系统角色 playbook 中，您可以根据您的偏好和限制为客户端 SSH 配置文件定义参数。

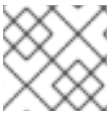
如果没有配置这些变量，系统角色会生成一个与 RHEL 默认值匹配的全局 **ssh_config** 文件。

在所有情况下，布尔值在 **ssh** 配置中都正确地呈现为 **yes** 或 **no**。您可以使用 list 来定义多行配置项。例如：

```
LocalForward:
  - 22 localhost:2222
  - 403 localhost:4003
```

呈现为：

```
LocalForward 22 localhost:2222
LocalForward 403 localhost:4003
```



注意

配置选项区分大小写。

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.ssh/README.md** 文件
- **/usr/share/doc/rhel-system-roles/ssh/** 目录

5.9.4. 使用 ssh RHEL 系统角色配置 OpenSSH 客户端

您可以通过运行 Ansible playbook，使用 **ssh** RHEL 系统角色来配置多个 SSH 客户端。

注意

您可以将 **ssh** RHEL 系统角色用于更改 SSH 和 SSHD 配置的其他系统角色，如身份管理 RHEL 系统角色。要防止配置被覆盖，请确保 **ssh** 角色使用置入目录（在 RHEL 8 及更新版本中默认使用）。

先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。

流程

1. 创建一个包含以下内容的 playbook 文件，如 **~/playbook.yml** ：

```
---
- name: SSH client configuration
  hosts: managed-node-01.example.com
  tasks:
    - name: "Configure ssh clients"
      ansible.builtin.include_role:
        name: rhel-system-roles.ssh
      vars:
        ssh_user: root
        ssh:
          Compression: true
          GSSAPIAuthentication: no
          ControlMaster: auto
          ControlPath: ~/.ssh/.cm%C
          Host:
            - Condition: example
              Hostname: server.example.com
              User: user1
        ssh_ForwardX11: no
```

此 playbook 使用以下配置在受管节点上配置 **root** 用户的 SSH 客户端首选项：

- 压缩已启用。
- ControlMaster 多路复用设置为 **auto**。
- 连接到 **server.example.com** 主机的 **example** 别名是 **user1**。
- 创建了 **example** 主机别名，它代表使用 **user1** 用户名到 **server.example.com** 主机的连接。
- X11 转发被禁用。

2. 验证 playbook 语法：

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

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

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

验证

- 通过显示 SSH 配置文件来验证受管节点是否有正确的配置：

```
# cat ~/root/.ssh/config
# Ansible managed
Compression yes
ControlMaster auto
ControlPath ~/.ssh/.cm%C
ForwardX11 no
GSSAPIAuthentication no
```

```
Host example
  Hostname example.com
  User user1
```

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.ssh/README.md** 文件
- **/usr/share/doc/rhel-system-roles/ssh/** 目录

5.9.5. 对非独占配置使用 sshd RHEL 系统角色

通常，应用 **sshd** 系统角色会覆盖整个配置。如果您之前已调整了配置，例如使用不同的系统角色或 playbook，这可能会有问题。要只对所选的配置选项应用 **sshd** 系统角色，同时保留其他选项，您可以使用非独占配置。

您可以应用一个非独占配置：

- 在 RHEL 8 及更早版本中，使用配置片断。
- 在 RHEL 9 及更高版本中，使用置入目录中的文件。默认配置文件已放入随时可访问的目录中，存为 **/etc/ssh/sshd_config.d/00-ansible_system_role.conf**。

先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。

流程

1. 创建一个包含以下内容的 playbook 文件，如 ~/**playbook.yml** ：
   - 对于运行 RHEL 8 或更早版本的受管节点：

```
---
- name: Non-exclusive sshd configuration
  hosts: managed-node-01.example.com
  tasks:
    - name: <Configure SSHD to accept some useful environment variables>
      ansible.builtin.include_role:
        name: rhel-system-roles.sshd
      vars:
        sshd_config_namespace: <my-application>
        sshd:
          # Environment variables to accept
          AcceptEnv:
            LANG
            LS_COLORS
            EDITOR
```

   - 对于运行 RHEL 9 或更高版本的受管节点：

```
- name: Non-exclusive sshd configuration
  hosts: managed-node-01.example.com
  tasks:
    - name: <Configure sshd to accept some useful environment variables>
      ansible.builtin.include_role:
        name: rhel-system-roles.sshd
      vars:
        sshd_config_file: /etc/ssh/sshd_config.d/<42-my-application>.conf
        sshd:
          # Environment variables to accept
          AcceptEnv:
            LANG
            LS_COLORS
            EDITOR
```

在 **sshd_config_file** 变量中，定义 **sshd** 系统角色在其中写入配置选项的 **.conf** 文件。使用两位前缀，例如 ***42-*** 来指定应用配置文件的顺序。

2. 验证 playbook 语法：

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

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

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

验证

- 验证 SSH 服务器上的配置：
  - 对于运行 RHEL 8 或更早版本的受管节点：

```
# cat /etc/ssh/sshd_config.d/42-my-application.conf
# Ansible managed
#
AcceptEnv LANG LS_COLORS EDITOR
```

  - 对于运行 RHEL 9 或更高版本的受管节点：

```
# cat /etc/ssh/sshd_config
...
# BEGIN sshd system role managed block: namespace <my-application>
Match all
  AcceptEnv LANG LS_COLORS EDITOR
# END sshd system role managed block: namespace <my-application>
```

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.sshd/README.md** 文件
- **/usr/share/doc/rhel-system-roles/sshd/** 目录

5.10. 其他资源

- **sshd(8)**、**ssh(1)**、**scp(1)**、**sftp(1)**、**ssh-keygen(1)**、**ssh-copy-id(1)**、**ssh_config(5)**、**ssh_config(5)**、**update-crypto-policies(8)** 和 **crypto-policies(7)** 手册页
- 为使用非标准配置的应用程序和服务配置 SELinux
- 使用 firewalld 控制网络流量

第 6 章 配置基本系统安全性

计算机安全性涉及到对硬件、软件、信息和服务的保护。计算机安全性是一项非常关键的任务，特别是对于那些处理敏感数据并处理商业事务的企业。

这部分只论述安装操作系统后您可以配置的基本安全功能。

6.1. 启用 FIREWALLD 服务

防火墙是一个网络安全系统，它可根据配置的安全规则监控并控制进入和离开的网络流量。防火墙通常在可信内部网络和其它网络间建立一个屏障。

在安装过程中，Red Hat Enterprise Linux 中提供防火墙的 **firewalld** 服务会自动启用。

要启用 **firewalld** 服务，请按照以下步骤执行。

流程

- 显示 **firewalld** 的当前状态：

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

- 如果没有启用并运行 **firewalld**，切换到 **root** 用户，启动 **firewalld** 服务并在系统重启后自动启动它：

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

验证步骤

- 检查 **firewalld** 已在运行并启用：

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

其他资源

- 使用和配置 firewalld
- **man firewalld(1)**

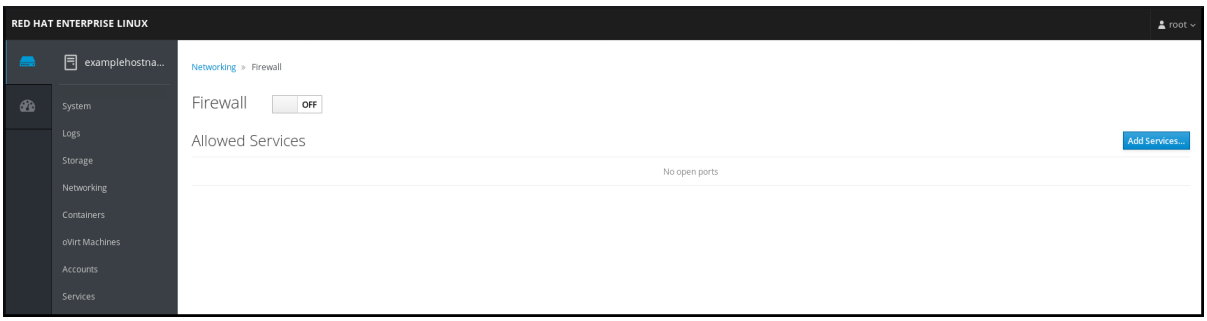
6.2. 在 RHEL 8 WEB 控制台中管理防火墙

要在 web 控制台中配置 **firewalld** 服务，请导航至 **Networking → Firewall**。

默认情况下 **firewalld** 服务是启用的。

流程

1. 要在 web 控制台中启用或禁用 **firewalld**，请切换 **Firewall** 切换按钮。



注意

另外，您可以使用 **Add services...** 按钮、通过防火墙定义更精细的访问。

6.3. 管理基本 SELINUX 设置

Security-Enhanced Linux（SELinux）是系统安全性的额外层，可决定哪些进程可访问哪些文件、目录和端口。这些权限在 SELinux 策略中定义。策略是一组指导 SELinux 安全引擎的规则。

SELinux 有两个可能的状态：

- Disabled
- Enabled

启用 SELinux 时，它以以下模式之一运行：

- Enabled
  - Enforcing
  - Permissive

在 **enforcing 模式** 中，SELinux 强制执行载入的策略。SELinux 会基于 SELinux 策略规则来拒绝访问，只有明确指定允许的操作才可以被接受。Enforcing 模式是最安全的 SELinux 模式，它是安装后的默认模式。

在 **permissive 模式** 中，SELinux 不强制执行载入的策略。SELinux 不会拒绝访问，但会在 **/var/log/audit/audit.log** 日志中报告违反了规则的操作。Permissive 模式是安装过程中的默认模式。在一些特殊情况下，permissive 模式也很有用，如进行故障排除时。

其他资源

- 使用 SELinux

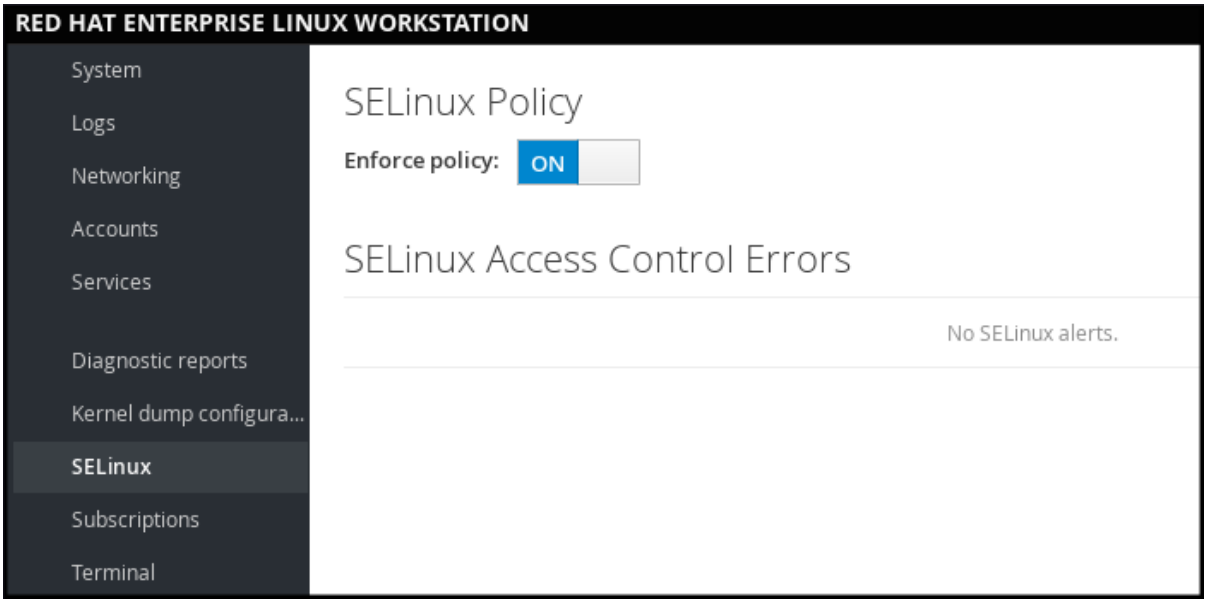
6.4. 在 RHEL 8 WEB 控制台中切换 SELINUX 模式

您可以在 SELinux 菜单项中通过 RHEL 8 web 控制台设置 **SELinux** 模式。

默认情况下，web 控制台中的 SELinux enforcing 策略是 on，SELinux 在 enforcing 模式下运行。关闭 SELinux 后，您需要将 SELinux 切换到 permissive 模式。请注意，此选择会在下次引导时自动恢复到 **/etc/sysconfig/selinux** 文件中定义的配置。

流程

1. 在 web 控制台中，使用 SELinux 菜单项中的 **Enforce policy** 切换按钮打开或关闭 SELinux 强制策略。



6.5. 其他资源

- 生成 SSH 密钥对
- 为基于密钥的身份验证设置 OpenSSH 服务器
- 安全强化
- 使用 SELinux
- 保护网络
- 在多个系统中部署相同的 SELinux 配置

第 7 章 管理软件包

7.1. RHEL 8 中的软件管理工具

在 RHEL 8 中，软件安装是通过新版本的 **YUM** 工具(**YUM v4**)启用的，该工具基于 **DNF** 技术。

注意

上游文档将此技术称为 **DNF**，这个工具在上游社区被称为 **DNF**。因此，RHEL 8 中新的 **YUM** 工具返回的一些输出中会包括 **DNF**。

尽管 RHEL 8 中使用的 **YUM v4** 是基于 **DNF**的，但它与 RHEL 7 中使用的 **YUM v3** 兼容。对于软件安装，**yum** 命令及其大多数选项在 RHEL 8 中的工作方式与 RHEL 7 中相同。

所选的 **yum** 插件和工具已移植到新的 DNF 后端，并可以安装在 RHEL 7 中相同的名称下。软件包也提供兼容性符号链接，因此可在通常的位置找到 二进制文件、配置文件和目录。

请注意，**YUM v3** 提供的旧版本的 Python API 不再可用。您可以将插件和脚本迁移到 **YUM v4** 提供的新 API 中(DNF Python API)，该 API 稳定且完全被支持。如需更多信息，请参阅 DNF API 参考。

7.2. 应用程序流

RHEL 8 引入了应用程序流的概念。现在，用户空间组件的多个版本的发布和更新频率会比核心操作系统的发布和更新频率快。应用程序流为用户提供了，在不影响底层平台或特定部署的稳定性的情况下，对 Red Hat Enterprise Linux 进行定制的更大的灵活性。

作为应用程序流提供的组件可以打包为模块或 RPM 软件包，并通过 RHEL 8 中的 AppStream 存储库交付。每个 Application Stream 都有一个生命周期，它与 RHEL 8 的生命周期相同或更短。Red Hat Enterprise Linux 8 应用程序流生命周期 页中列出了生命周期较短的应用程序流。

模块是代表逻辑单元的软件包集合： 应用程序、语言堆栈、数据库或一组工具。这些软件包被一同构建、测试并发布。

模块流代表 Application Stream 组件的版本。例如，PostgreSQL 数据库服务器的两个流（版本）位于 postgresql 模块中： PostgreSQL 10（默认流）和 PostgreSQL 9.6。在系统中只能安装 一 个模块流。不同的容器可以使用不同的版本。

详细的模块命令，请参考 安装、管理和删除用户空间组件文档。有关 AppStream 中可用的模块列表，请查看 软件包清单。

7.3. 搜索软件包

yum 允许您使用软件包执行一组完整的操作。

下面的部分描述了如何使用 **yum**:

- 搜索软件包。
- 列出软件包。
- 列出软件仓库。
- 显示软件包信息。

- 列出软件包组。
- 在 yum 输入中指定全局表达式。

7.3.1. 使用 YUM 搜索软件包

使用以下流程查找提供特定应用程序或其他内容的软件包。

流程

- 要搜索软件包，使用：

```
# yum search term
```

使用与该软件包相关的术语替换 *term*。

请注意，**yum search** 命令返回与软件包名称和概述中的术语匹配。这样可加快搜索速度，您可以搜索您不知道名称但您了解相关术语的软件包。

- 要在软件包描述中包含匹配名，请使用：

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

用您要在软件包名称、概述或描述中搜索的术语替换 *term*。

请注意 **yum search --all** 可进行更详细但更慢的搜索。

7.3.2. 使用 YUM 列出软件包

使用以下流程列出已安装和可用的软件包。

流程

- 要列出所有已安装的和可用软件包的信息，请使用：

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

- 要列出安装在您的系统中的所有软件包，请使用：

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

- 要列出所有启用的存储库中可以安装的软件包，请使用：

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

请注意，您可以通过将全局表达式附加为参数来过滤结果。请参阅 在 yum 输入中指定全局表达式

如需更多详细信息。

7.3.3. 使用 YUM 列出存储库

使用以下流程列出启用和禁用的存储库。

流程

- 要列出您系统中所有启用的库，请使用：

```
# yum repolist
```

- 要列出系统中所有禁用的软件仓库，请使用：

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

- 要列出启用和禁用的存储库，请使用：

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

- 要列出有关存储库的附加信息，请使用：

```
# yum repoinfo
```

请注意，您可以通过传递 ID 或库名称作为参数或者附加全局表达式来过滤结果。请参阅 在 yum 输入中指定全局表达式

如需更多详细信息。

7.3.4. 使用 YUM 显示软件包信息

您可以使用 YUM 显示软件包的各种信息，如版本、发行、大小、载入的插件等。

流程

- 要显示一个或多个软件包的信息，请使用：

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

使用软件包名称替换 *package-name*。

请注意，您可以通过将全局表达式附加为参数来过滤结果。请参阅 在 yum 输入中指定全局表达式

如需更多详细信息。

7.3.5. 使用 YUM 列出软件包组

使用 **yum** 查看已安装的软件包组，并过滤列出的结果。

流程

- 要查看已安装的和可用组的数量，请使用：

```
# yum group summary
```

- 要列出所有安装的和可用的组，请使用：

```
# yum group list
```

请注意，您可以通过为 **yum group list** 命令添加命令行选项（**--hidden**，**--available**）过滤结果。更多可用选项请查看 man page。

- 要列出特定组群中包含的强制和可选软件包，请使用：

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

用组群的名称替换 *group-name*。

请注意，您可以通过将全局表达式附加为参数来过滤结果。请参阅 在 yum 输入中指定全局表达式

如需更多详细信息。

7.3.6. 在 YUM 输入中指定全局表达式

yum 命令允许您将一个或多个 *glob 表达式* 作为参数过滤。当将全局表达式作为参数传递给 **yum** 命令时，您必须退出全局表达式。

流程

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

- 使用双引号或单引号包括整个全局表达式。

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

用文件名替换 *file-name*。

- 在它们前面使用反斜杠(\)符号转义通配符字符。

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

用文件名替换 *file-name*。

7.4. 安装软件包

下面的部分描述了如何使用 **yum**:

- 安装软件包。
- 安装软件包组。
- 在 yum 输入中指定软件包名称。

7.4.1. 使用 YUM 安装软件包

- 要安装软件包以及所有软件包的依赖软件包，请使用：

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

使用软件包名称替换 *package-name*。

- 要同时安装多个软件包及其依赖软件包，请使用：

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

使用软件包名称替换 *package-name-1* 和 *package-name-2*。

- 当在一个 *multilib* 系统（AMD64、Intel 64 机器）中安装软件包时，您可以指定软件包的构架，方法是将其附加到软件包名称中：

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

使用软件包的名称和构架替换 *package-name.arch*。

- 如果您知道要安装的二进制代码的名称，但不知道软件包名，则可以使用到这个二进制代码的路径作为一个参数：

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

使用二进制文件路径替换 ***/usr/sbin/binary-file***。

yum 搜索软件包列表，找到提供 ***/usr/sbin/binary-file*** 的软件包，并提示您是否安装该软件包。

- 要从本地目录中安装之前下载的软件包，请使用：

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

使用到该软件包的路径替换 */path/*。

请注意，您可以通过显式定义如何解析参数来优化软件包搜索。详情请查看 第 7.4.3 节 “在 YUM 输入中指定软件包名称”。

7.4.2. 使用 YUM 安装软件包组

下面的流程描述了如何使用 **yum** 根据组群名称或 groupID 安装软件包组。

流程

- 要根据组名称安装软件包组，请使用：

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

或者

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

使用组群或者环境组群的完整名称替换 *group-name*。

- 要根据 groupID 安装软件包组，请使用：

```
# yum group install groupID
```

使用组 ID 替换 *groupID*。

7.4.3. 在 YUM 输入中指定软件包名称

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:

- 要使用其确切的名称安装软件包，请使用：

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

  使用具体软件包名称替换 *name*。

- 要使用其确切的名称和构架安装软件包，请使用：

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

  使用软件包的实际名称和构架替换 *name* 和 *architecture*。

- 要使用其确切的名称、epoch、版本、发行版和架构安装软件包，请使用：

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

  用软件包的名称、epoch、版本、发行和架构替换 *name*、*epoch*、*version*、*release*和 *architecture*

7.5. 更新软件包

yum 允许您检查您的系统是否有待处理的更新。您可以列出需要更新的软件包，并选择更新单个软件包、多个软件包或者所有软件包。如果您选择更新的软件包有依赖项，它们也会被更新。

下面的部分描述了如何使用 **yum**:

- 检查更新。
- 更新单个软件包。
- 更新软件包组。
- 更新所有软件包及其依赖项。
- 应用安全更新。
- 自动软件更新。

7.5.1. 使用 YUM 检查更新

以下流程描述了如何使用 **yum** 检查系统上安装的软件包的可用更新。

流程

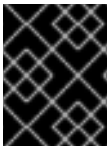
- 要查看您系统中安装的软件包是否有可用的更新，请使用：

  ```
  # yum check-update
  ```

  输出返回有可用更新的软件包及其依赖项列表。

7.5.2. 使用 YUM 更新单个软件包

使用以下流程通过 **yum** 更新单个软件包及其依赖项。



重要

在对内核应用更新时，**yum** 总会**安装**一个新内核，无论是否使用了 **yum update** 或 **yum install** 命令。

- 要更新软件包，请使用：

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

使用软件包名称替换 *package-name*。

重要

如果您在 BIOS 或 IBM Power 系统上升级了 GRUB 引导装载程序软件包，请重新安装 GRUB。请参阅 重新安装 GRUB。

7.5.3. 使用 YUM 更新软件包组

使用以下流程通过 **yum** 更新一组软件包及其依赖项。

流程

- 要更新软件包组，请使用：

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

使用软件包组的名称替换 *group-name*。

重要

如果您在 BIOS 或 IBM Power 系统上升级了 GRUB 引导装载程序软件包，请重新安装 GRUB。请参阅 重新安装 GRUB。

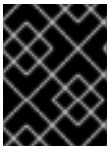
7.5.4. 使用 YUM 更新所有软件包及其依赖项

使用以下流程通过 **yum** 更新所有软件包及其依赖项。

流程

- 要更新所有软件包及其依赖项，请使用：

```
# yum update
```



重要

如果您在 BIOS 或 IBM Power 系统上升级了 GRUB 引导装载程序软件包，请重新安装 GRUB。请参阅 重新安装 GRUB。

7.5.5. 使用 YUM 更新与安全相关的软件包

使用以下流程，使用 **yum** 更新具有安全勘误的软件包。

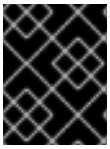
流程

- 要升级到有安全勘误的最新可用软件包，请使用：

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

- 要升级到最后一个安全勘误软件包，请使用：

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



重要

如果您在 BIOS 或 IBM Power 系统上升级了 GRUB 引导装载程序软件包，请重新安装 GRUB。请参阅 重新安装 GRUB。

7.5.6. 自动化软件更新

要自动并定期检查和下载软件包更新，您可以使用 **dnf-automatic** 软件包提供的 DNF Automatic 工具。

DNF Automatic 是 **yum** 的替代命令行界面，它适用于使用 systemd 计时器、cron 任务和其它此类工具自动和常规执行。

DNF Automatic 根据需要同步软件包元数据,然后检查可用更新。之后，该工具可以根据其配置的方式来执行以下操作之一：

- 退出
- 下载更新的软件包
- 下载并应用更新

然后，通过选定的机制（如标准输出或电子邮件）报告操作的结果。

7.5.6.1. 安装 DNF Automatic

以下流程描述了如何安装 DNF Automatic 工具。

流程

- 要安装 **dnf-automatic** 软件包，请使用：

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

验证步骤

- 要验证安装是否成功，请运行以下命令来确认 **dnf-automatic** 软件包是否存在：

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

7.5.6.2. DNF Automatic 配置文件

默认情况下，DNF Automatic 使用 **/etc/dnf/automatic.conf** 作为其配置文件来定义其行为。

配置文件被分隔为以下主题部分：

- **[commands]** 部分
  设置 DNF Automatic 的操作模式。
- **[emitters]** 部分
  定义如何报告 DNF Automatic 的结果。
- **[command_email]** 部分
  为用来发送电子邮件的外部命令提供电子邮件发布程序配置。
- **[email]** 部分
  提供电子邮件发布程序配置。
- **[base]** 部分
  覆盖 yum 主配置文件中的设置。

使用 **/etc/dnf/automatic.conf** 文件的默认设置，DNF Automatic 会检查可用的更新，下载这些更新，并以标准输出的形式报告结果。

警告

[commands] 部分中操作模式的设置会被所有计时器单元的 systemd 定时器单元覆盖，**dnf-automatic.timer** 除外。

其他资源

- 如需具体部分的详情，请参阅 DNF Automatic 文档。
- 有关 systemd 计时器单元的详情，请查看 **man dnf-automatic** 手册页。
- 有关 **dnf-automatic 软件包** 中包含的 systemd 计时器单元的概述，请参阅 dnf-automatic 软件包中所含 的systemd 计时器单元的概述 部分 dnf-automatic 软件包中所含 的systemd 计时器单元的概述

7.5.6.3. 启用 DNF Automatic

要运行 DNF Automatic，您始终需要启用并启动特定的 systemd 计时器单元。您可以使用 **dnf-automatic** 软件包中提供的计时器单元，或者您可以根据需要编写您自己的计时器单元。

下面的部分论述了如何启用 DNF Automatic。

先决条件

- 您可以通过修改 **/etc/dnf/automatic.conf** 配置文件来指定 DNF Automatic 的行为。

有关 DNF Automatic 配置文件的更多信息，请参阅 "DNF 自动配置文件"的 2.5.6.2 部分。

流程

- 选择、启用并启动一个符合您需要的 systemd 计时器单元：

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

其中 **<unit>** 是以下计时器之一：

- **dnf-automatic-download.timer**
- **dnf-automatic-install.timer**
- **dnf-automatic-notifyonly.timer**
- **dnf-automatic.timer**
  - 要 **下载** 可用的更新，请使用：

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

  - 要 **下载并安装** 可用的更新，请使用：

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

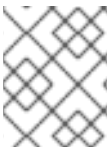
  - 要 **报告** 可用的更新，请使用：

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

  - 另外，您可以使用：

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

就下载和应用更新而言，这个计时器单元的行为取决于 **/etc/dnf/automatic.conf** 配置文件中的设置。默认行为与 **dnf-automatic-download.timer** 类似：它会下载更新的软件包，但不安装它们。



注意

或者，您还可以从命令行或从自定义脚本，通过直接执行 **/usr/bin/dnf-automatic** 文件来运行 DNF Automatic。

验证步骤

- 要验证是否启用了计时器，请运行以下命令：

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

其他资源

- 有关 dnf-automatic 计时器的详情，请参考 **man dnf-automatic** 手册页。

- 有关 **dnf-automatic** 软件包中所含的 systemd 计时器单元的概述，请参阅 dnf-automatic 软件包中所含的 systemd 计时器单元的概述 部分

7.5.6.4. dnf-automatic 软件包中包含的 systemd 计时器单元的概述

systemd 定时器单元优先并覆盖 **/etc/dnf/automatic.conf** 配置文件中有关下载和应用更新的设置。

例如，如果您在 **/etc/dnf/automatic.conf** 配置文件中设置了以下选项，但已激活了 **dnf-automatic-notifyonly.timer** 单元，则软件包将不会被下载：

```
download_updates = yes
```

dnf-automatic 软件包包括以下 systemd 计时器单元：

计时器单元	功能	覆盖 /etc/dnf/automatic.conf 文件中的设置？
dnf-automatic-download.timer	下载软件包以便进行更新。 注：这个计时器单元没有安装更新的软件包。要执行安装，您必须执行 **dnf update** 命令。	是
dnf-automatic-install.timer	下载并安装更新的软件包。	是
dnf-automatic-notifyonly.timer	仅下载存储库数据，以保持存储库缓存最新，并通知您有关可用的更新。 注：这个计时器单元不下载或安装更新的软件包	是
dnf-automatic.timer	此计时器有关下载和应用更新的行为由 **/etc/dnf/automatic.conf** 配置文件中的设置指定。 默认行为与 **dnf-automatic-download.timer** 单元的行为相同：它仅下载软件包，但不安装它们。	否

其他资源

- 有关 **dnf-automatic** 计时器的详情，请参考 **man dnf-automatic** 手册页。
- 有关 **/etc/dnf/automatic.conf** 配置文件的更多信息，请参阅 DNF 自动配置文件

7.6. 卸载软件包

下面的部分描述了如何使用 **yum**:

- 删除软件包。

- 删除软件包组。
- 在 yum 输入中指定软件包名称。

7.6.1. 使用 YUM 删除软件包

使用以下流程，根据组名或 groupID 删除软件包。

流程

- 要删除某个软件包以及所有相依性软件包，请使用：

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

使用软件包名称替换 *package-name*。

- 要同时删除多个软件包及其依赖项，请使用：

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

使用软件包名称替换 *package-name-1* 和 *package-name-2*。

注意

在删除其依赖软件包前。yum 无法删除软件包。

请注意，您可以通过显式定义如何解析参数来优化软件包搜索。如需了解更多详细信息，请参阅 在 yum 输入中指定软件包名称。

7.6.2. 使用 YUM 删除软件包组

使用以下流程，根据组名或 groupID 删除软件包。

流程

- 要根据组群名称删除软件包组，请使用：

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

或者

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

使用组群的全名替换 *group-name*。

- 要通过 groupID 删除软件包组，请使用：

```
# yum group remove groupID
```

使用组 ID 替换 *groupID*。

7.6.3. 在 YUM 输入中指定软件包名称

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:

- 要使用其确切的名称安装软件包，请使用：

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

  使用具体软件包名称替换 *name*。

- 要使用其确切的名称和构架安装软件包，请使用：

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

  使用软件包的实际名称和构架替换 *name* 和 *architecture*。

- 要使用其确切的名称、epoch、版本、发行版和架构安装软件包，请使用：

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

  用软件包的名称、epoch、版本、发行和架构替换 *name*、*epoch*、*version*、*release*和 *architecture*

7.7. 管理软件包组

软件包组是用于共同目的的软件包集合（System Tools、Sound and Video）。安装软件包组会拉取一组依赖软件包，这可节省大量时间。

下面的部分描述了如何使用 yum:

- 列出软件包组。
- 安装软件包组。
- 删除软件包组。
- 在 yum 输入中指定全局表达式。

7.7.1. 使用 YUM 列出软件包组

使用 **yum** 查看已安装的软件包组，并过滤列出的结果。

流程

- 要查看已安装的和可用组的数量，请使用：

  ```
  # yum group summary
  ```

- 要列出所有安装的和可用的组，请使用：

  ```
  # yum group list
  ```

  请注意，您可以通过为 **yum group list** 命令添加命令行选项（**--hidden**，**--available**）过滤结果。更多可用选项请查看 man page。

- 要列出特定组群中包含的强制和可选软件包，请使用：

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

用组群的名称替换 *group-name*。

请注意，您可以通过将全局表达式附加为参数来过滤结果。请参阅 在 yum 输入中指定全局表达式 如需更多详细信息。

7.7.2. 使用 YUM 安装软件包组

下面的流程描述了如何使用 **yum** 根据组群名称或 groupID 安装软件包组。

流程

- 要根据组名称安装软件包组，请使用：

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

或者

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

使用组群或者环境组群的完整名称替换 *group-name*。

- 要根据 groupID 安装软件包组，请使用：

```
# yum group install groupID
```

使用组 ID 替换 *groupID*。

7.7.3. 使用 YUM 删除软件包组

使用以下流程，根据组名或 groupID 删除软件包。

流程

- 要根据组群名称删除软件包组，请使用：

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

或者

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

使用组群的全名替换 *group-name*。

- 要通过 groupID 删除软件包组，请使用：

```
# yum group remove groupID
```

使用组 ID 替换 *groupID*。

7.7.4. 在 YUM 输入中指定全局表达式

yum 命令允许您将一个或多个 *glob 表达式* 作为参数过滤。当将全局表达式作为参数传递给 **yum** 命令时，您必须退出全局表达式。

流程

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

- 使用双引号或单引号包括整个全局表达式。

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

  用文件名替换 *file-name*。

- 在它们前面使用反斜杠(\)符号转义通配符字符。

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

  用文件名替换 *file-name*。

7.8. 处理软件包管理历史记录

yum history 命令允许您查看有关 **yum** 事务的时间线、所发生的日期和时间、受影响的软件包数量、这些事务是成功还是被中止，以及是否在事务间更改了 RPM 数据库的信息。**yum history** 命令也可用于撤销或重做事务。

下面的部分描述了如何使用 **yum**:

- 列出事务。
- 恢复事务。
- 重复事务。
- 在 yum 输入中指定全局表达式。

7.8.1. 使用 YUM 列出事务

使用以下流程列出最新的事务、对所选软件包的最新操作以及特定事务的详情。

流程

- 要显示所有最新的 **yum** 事务列表，请使用：

  ```
  # yum history
  ```

- 要显示所选软件包的最新操作列表，请使用：

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

  使用软件包名称替换 *package-name*。您可以通过附加全局表达式来过滤命令输出。如需了解更多详细信息，请参阅 在 yum 输入中指定全局表达式。

- 要检查特定的事务，请使用：

```
# yum history info transactionID
```

用事务的 ID 替换 *transactionID*。

7.8.2. 使用 YUM 恢复事务

以下流程描述了如何使用 **yum** 恢复所选事务或最后一个事务。

流程

- 要恢复特定的事务，请使用：

```
# yum history undo transactionID
```

用事务的 ID 替换 *transactionID*。

- 要恢复到最后的事务，请使用：

```
# yum history undo last
```

请注意，**yum history undo** 命令只恢复事务期间执行的操作步骤。如果事务安装了一个新的软件包，**yum history undo** 命令会卸载它。如果事务卸载了一个软件包，则 **yum history undo** 命令会重新安装它。**yum history undo** 还会尝试将所有更新的软件包降级到它们之前的版本（如果旧包仍然可用）。

7.8.3. 使用 YUM 重复事务

使用以下流程，通过 **yum** 重复所选事务或最后的事务。

流程

- 要重复特定的事务，请使用：

```
# yum history redo transactionID
```

用事务的 ID 替换 *transactionID*。

- 要重复最后的事务，请使用：

```
# yum history redo last
```

请注意， **yum history redo** 命令只重复事务期间执行的步骤。

7.8.4. 在 YUM 输入中指定全局表达式

yum 命令允许您将一个或多个 *glob 表达式* 作为参数过滤。当将全局表达式作为参数传递给 **yum** 命令时，您必须退出全局表达式。

流程

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

-

- 使用双引号或单引号包括整个全局表达式。

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

  用文件名替换 *file-name*。

- 在它们前面使用反斜杠(\)符号转义通配符字符。

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

  用文件名替换 *file-name*。

7.9. 管理软件存储库

yum 及相关工具的配置信息保存在 **/etc/yum.conf** 文件中。此文件包含一个或多个 **[*repository*]** 部分，用于设置特定存储库选项。

建议您在 **/etc/yum.repos.d/** 目录的新的或现有 **.repo** 文件中定义单独的库。

请注意：您在 **/etc/yum.conf** 文件的单独 **[*repository*]** 部分定义的值会覆盖 **[main]** 部分中设置的值。

下面的部分描述了如何：

- 设置 **[*repository*]** 选项。
- 添加 **yum** 软件仓库。
- 启用 **yum** 软件仓库。
- 禁用 **yum** 软件仓库。

7.9.1. 设置 YUM 存储库选项

/etc/yum.conf 配置文件包含 **[*repository*]** 部分，其中 *repository* 是唯一的软件仓库 ID。**[*repository*]** 项可以用来定义独立的 **yum** 软件仓库。

注意

不要给出红帽软件仓库使用的自定义软件仓库名称以避免冲突。

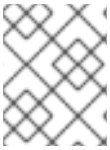
有关可用 **[*存储库*]** 选项 的完整列表，请查看 **yum.conf**(5)手册页中的 **[*repositor*] OPTIONS** 部分。

7.9.2. 添加一个 YUM 存储库

流程

要定义新软件仓库，您可以：

- 在 **/etc/yum.conf** 文件中添加 **[*repository*]** 部分。
- 在 **/etc/yum.repos.d/** 目录中的 **.repo** 文件中添加 **[*repository*]** 部分。
  yum repositories 通常提供它们自己的 **.repo** 文件。



注意

建议您在 **.repo** 文件中而不是 **/etc/yum.conf** 中定义您的程序仓库，因为在这个目录中的所有带有 **.repo** 文件扩展名的文件都被 **yum** 读取。

- 要在您的系统中添加库并启用该程序，请使用：

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

使用指向库的 URL 替换 *repository_url*。

警告

从未验证或不受信任的源而不是基于红帽证书的 **内容交付网络** (CDN)来获取和安装软件包构成了潜在的安全风险，并可能导致安全性、稳定性、兼容性和可维护性问题。

7.9.3. 启用一个 YUM 存储库

在您的系统中添加了 **yum** 存储库后，启用它以确保安装和更新。

流程

- 要启用存储库，请使用：

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

使用唯一的存储库 ID 替换 *repositoryID*。

要列出可用存储库的 ID，请参阅 使用 yum 列出软件包。

7.9.4. 禁用一个 YUM 存储库

禁用特定的 YUM 存储库，以防止安装或更新特定的软件包。

流程

- 要禁用 yum 软件仓库，请使用：

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

使用唯一的存储库 ID 替换 *repositoryID*。

要列出可用存储库的 ID，请参阅 使用 yum 列出软件包。

7.10. 配置 YUM

yum 及相关工具的配置信息保存在 **/etc/yum.conf** 文件中。这个文件包含一个必须的 **[main]** 部分,它可让您设置具有全局效果的 **yum** 选项。

下面的部分描述了如何：

- 查看当前的 **yum** 配置。
- 设置 **yum** [main] options。
- 使用 **yum** 插件。

7.10.1. 查看当前的 YUM 配置

使用以下流程查看当前的 **yum** 配置。

流程

- 要显示 **/etc/yum.conf** 文件的 **[main]** 部分中指定的全局 **yum** 选项的当前值，请使用：

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

7.10.2. 设置 YUM 主要选项

/etc/yum.conf 配置文件包含一个 **[main]** 部分。以下列出的键-值对会影响 **yum** 如何操作，以及如何对待存储库。

您可以在 **/etc/yum.conf** 文件 **[main]** 部分标题下添加附加选项。

有关可用的 **[main]** 选项的完整列表，请查看 **yum.conf**(5)手册页中的 **[main] OPTIONS** 部分。

7.10.3. 使用 YUM 插件

yum 提供扩展和增强操作的插件。默认安装某些插件。

下面的部分论述了如何启用、配置和禁用 **yum** 插件。

7.10.3.1. 管理 YUM 插件

流程

插件配置文件始终包含一个 **[main]** 部分，其中 **enabled=** 选项控制在运行 **yum** 命令时插件是否启用。如果缺少这个选项，您可以手动将其添加到该文件中。

每个安装的插件在 **/etc/dnf/plugins/** 目录中都有自己的配置文件。您可以在这些文件中启用或禁用特定插件选项。

7.10.3.2. 启用 YUM 插件

以下流程描述了如何禁用或启用所有 **YUM** 插件，为特定的命令禁用所有插件，或为单个命令禁用某些 **YUM** 插件。

流程

- 启用所有 yum 插件：
  1. 请确定在 **/etc/yum.conf** 文件的 **[main]** 部分有以 **plugins=** 开头的行。
  2. 将 **plugins=** 的值设置为 **1**。

```
plugins=1
```

7.10.3.3. 禁用 YUM 插件

- 禁用所有 yum 插件：

  1. 请确定在 **/etc/yum.conf** 文件的 **[main]** 部分有以 **plugins=** 开头的行。
  2. 将 **plugins=** 的值设置为 **0**。

     ```
     plugins=0
     ```

     重要

     不建议禁用所有插件。某些插件提供重要的 yum 服务。特别是 **product-id** 和 **subscription-manager** 插件，它们为基于证书的 **内容发布网络** (CDN)提供支持。全局禁用插件是作为一个方便选项提供的，仅在使用 **yum** 诊断潜在的问题时才建议使用。

- 要禁用特定命令的所有 yum 插件，请在该命令中附加 **--noplugins** 选项。

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

- 要在一个命令中禁用特定的 yum 插件，请在命令中附加 **--disableplugin=*plugin-name*** 选项。

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

  使用插件的名称替换 *plugin-name*。

第 8 章 RHEL 系统角色简介

通过使用 RHEL 系统角色，您可以远程管理跨 RHEL 主版本的多个 RHEL 系统的系统配置。

重要术语和概念

下面描述了 Ansible 环境中的重要术语和概念：

控制节点

控制节点是您运行 Ansible 命令和 playbook 的系统。您的控制节点可以是 Ansible Automation Platform、Red Hat Satellite 或 RHEL 9、8 或 7 主机。如需更多信息，请参阅 在 RHEL 8 上准备一个控制节点。

受管节点

受管节点是您使用 Ansible 管理的服务器和网络设备。受管节点有时也称为主机。Ansible 不必安装在受管节点上。如需更多信息，请参阅 准备一个受管节点。

Ansible playbook

在 playbook 中，您定义要在受管节点上实现的配置，或受管节点上的系统要执行的一组步骤。Playbook 是 Ansible 的配置、部署和编配语言。

清单（Inventory）

在清单文件中，您列出受管节点，并指定信息，如每个受管节点的 IP 地址等。在清单中，您还可以通过创建和嵌套组来组织受管节点，以便于扩展。清单文件有时也称为主机文件。

Red Hat Enterprise Linux 8 控制节点上的可用角色

在 Red Hat Enterprise Linux 8 控制节点上，**rhel-system-roles** 软件包提供以下角色：

角色名称	角色描述	章节标题
certificate	证书问题和续订	使用 RHEL 系统角色请求证书
cockpit	Web 控制台	使用 cockpit RHEL 系统角色安装和配置 Web 控制台
crypto_policies	系统范围的加密策略	设置跨系统的自定义加密策略
firewall	Firewalld	使用系统角色配置 firewalld
ha_cluster	HA 集群	使用系统角色配置高可用性集群
kdump	内核转储	使用 RHEL 系统角色配置 kdump
kernel_settings	内核设置	使用 Ansible 角色永久配置内核参数
logging	日志记录	使用 logging 系统角色
metrics	指标(PCP)	使用 RHEL 系统角色监控性能
microsoft.sql.server	Microsoft SQL Server	使用 microsoft.sql.server Ansible 角色配置 Microsoft SQL Server

角色名称	角色描述	章节标题
network	网络	使用 network RHEL 系统角色管理 InfiniBand 连接
nbde_client	网络绑定磁盘加密客户端	使用 nbde_client 和 nbde_server 系统角色
nbde_server	网络绑定磁盘加密服务器	使用 nbde_client 和 nbde_server 系统角色
postfix	postfix	系统角色中 postfix 角色的变量
postgresql	PostgreSQL	使用 postgresql RHEL 系统角色安装和配置 PostgreSQL
selinux	SELinux	使用系统角色配置 SELinux
ssh	SSH 客户端	使用 ssh 系统角色配置安全通信
sshd	SSH 服务器	使用 ssh 系统角色配置安全通信
storage	存储	使用 RHEL 系统角色管理本地存储
tlog	终端会话记录	使用 tlog RHEL 系统角色为会话记录配置系统
timesync	时间同步	使用 RHEL 系统角色配置时间同步
vpn	VPN	使用 vpn RHEL 系统角色配置带有 IPsec 的 VPN 连接

其他资源

- 使用 RHEL 系统角色自动化系统管理
- Red Hat Enterprise Linux (RHEL)系统角色
- **/usr/share/ansible/roles/rhel-system-roles.*<role_name>*/README.md** 文件
- **/usr/share/doc/rhel-system-roles/*<role_name>*/** 目录

第 9 章 配置日志记录

Red Hat Enterprise Linux 中的大多数服务都会记录状态消息、警告和错误。您可以使用 **rsyslogd** 服务将这些条目记录到本地文件或远程日志记录服务器。

9.1. 配置远程日志记录解决方案

要确保在日志记录服务器中集中记录来自环境中各种机器的日志，您可以将 Rsyslog 应用程序配置为将适合客户端系统中特定条件的日志记录到服务器。

9.1.1. Rsyslog 日志记录服务

Rsyslog 应用程序与 **systemd-journald** 服务相结合，在 Red Hat Enterprise Linux 中提供了本地和远程记录支持。**rsyslogd** 守护进程会持续读取 **systemd-journald** 服务从 Journal 接收的 **syslog** 消息。然后 **rsyslogd** 会过滤并处理这些 **syslog** 事件，并将它们记录到 **rsyslog** 日志文件，或者根据自己的配置将它们转发到其他服务。

rsyslogd 守护进程还提供扩展的过滤、加密受保护的转发消息、输入和输出模块，并支持使用 TCP 和 UDP 协议进行传输。

在 **/etc/rsyslog.conf** 中，是 **rsyslog** 的主要配置文件，您可以根据 **rsyslogd** 处理消息来指定规则。通常，您可以通过其来源和主题（设施）和紧急情况（优先级）对消息进行分类，然后分配在消息适合这些条件时应执行的操作。

在 **/etc/rsyslog.conf** 中，您还可以看到 **rsyslogd** 维护的日志文件列表。大多数日志文件位于 **/var/log/** 目录中。**httpd** 和 **samba** 等一些应用将其日志文件存储在 **/var/log/** 中的子目录中。

其他资源

- **rsyslogd (8)** 和 **rsyslog.conf (5)** man page。
- 在 **/usr/share/doc/rsyslog/html/index.html** 文件中通过 **rsyslog-doc** 软件包安装的文档。

9.1.2. 安装 Rsyslog 文档

Rsyslog 应用程序在 https://www.rsyslog.com/doc/ 上提供了广泛的在线文档，但您也可以在本地安装 **rsyslog-doc** 文档软件包。

先决条件

- 您已在系统中激活了 **AppStream** 软件仓库。
- 您有权使用 **sudo** 安装新软件包。

流程

- 安装 **rsyslog-doc** 软件包：

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

验证

- 在您选择的浏览器中打开 **/usr/share/doc/rsyslog/html/index.html** 文件，例如：

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

9.1.3. 通过 TCP 配置服务器进行远程记录

Rsyslog 应用程序可让您运行日志服务器并配置各个系统将其日志文件发送到日志记录服务器。要通过 TCP 使用远程日志，请同时配置服务器和客户端。服务器收集和分析由一个或多个客户端系统发送的日志。

使用 Rsyslog 应用程序，您可以维护一个集中的日志系统，该系统可通过网络将日志消息转发到服务器。为了避免服务器不可用时消息丢失，您可以为转发操作配置操作队列。这样，无法发送的消息将存储在本地，直到服务器再次可访问为止。请注意，此类队列无法针对使用 UDP 协议的连接配置。

omfwd 插件通过 UDP 或 TCP 提供转发。默认协议是 UDP。由于插件内置在内，因此不必加载它。

默认情况下，**rsyslog** 使用端口 **514** 上的 TCP。

先决条件

- rsyslog 已安装在服务器系统上。
- 您以 **root** 身份登录到服务器中。
- 使用 **semanage** 命令，为可选步骤安装 **policycoreutils-python-utils** 软件包。
- **firewalld** 服务在运行。

流程

1. 可选： 要为 **rsyslog** 流量使用不同的端口，请将 **syslogd_port_t** SELinux 类型添加到端口。例如，启用端口 **30514** ：

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

2. 可选： 要为 **rsyslog** 流量使用不同的端口，请将 **firewalld** 配置为允许该端口上传入的 **rsyslog** 流量。例如，允许端口 **30514** 上的 TCP 流量：

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

3. 在 **/etc/rsyslog.d/** 目录中创建一个新文件（例如，**remotelog.conf**），并插入以下内容：

```
# 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. 将更改保存到 **/etc/rsyslog.d/remotelog.conf** 文件。

5. 测试 **/etc/rsyslog.conf** 文件的语法：

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

6. 确保 **rsyslog** 服务在日志记录服务器中运行并启用：

```
# systemctl status rsyslog
```

7. 重新启动 **rsyslog** 服务。

```
# systemctl restart rsyslog
```

8. 可选： 如果没有启用 **rsyslog**，请确保 **rsyslog** 服务在重启后自动启动：

```
# systemctl enable rsyslog
```

您的日志服务器现在已配置为从您环境中的其他系统接收和存储日志文件。

其他资源

- **rsyslogd(8)**, **rsyslog.conf(5)**, **semanage(8)**, 和 **firewall-cmd(1)** man pages.
- 在 **/usr/share/doc/rsyslog/html/index.html** 文件中通过 **rsyslog-doc** 软件包安装的文档。

9.1.4. 通过 TCP 配置远程日志记录到服务器

您可以配置系统，以通过 TCP 协议将日志消息转发到服务器。**omfwd** 插件通过 UDP 或 TCP 提供转发。默认协议是 UDP。因为插件内置在内，所以不必加载它。

先决条件

- **rsyslog** 软件包安装在应该向服务器报告的客户端系统上。

- 您已为远程日志记录配置了服务器。
- 在 SELinux 中允许指定的端口并在防火墙中打开。
- 系统包含 **policycoreutils-python-utils** 软件包，它为 SELinux 配置中添加非标准端口提供 **semanage** 命令。

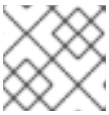
流程

1. 在 **/etc/rsyslog.d/** 目录中创建一个名为 的新文件，如 **10-remotelog.conf**，并插入以下内容：

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

其中：

- **queue.type="linkedlist"** 设置启用 LinkedList 内存中队列，
- **queue.filename** 设置定义磁盘存储。备份文件是使用 **example_fwd** 前缀，在之前全局 **workDirectory** 指令指定的工作目录中创建的。
- **action.resumeRetryCount -1** 设置防止 **rsyslog** 在服务器没有响应而重试连接时丢弃消息，
- 如果 **rsyslog** 关闭，**queue.saveOnShutdown="on"** 设置会保存内存中的数据。
- 最后一行将所有收到的消息转发到日志记录服务器。端口规格是可选的。
  使用这个配置，**rsyslog** 会向服务器发送消息，但如果远程服务器无法访问，则会将消息保留在内存中。只有 **rsyslog** 耗尽配置的内存队列空间或需要关闭时，才能创建磁盘上的文件，从而让系统性能受益。



注意

rsyslog 按照一般顺序处理配置文件 **/etc/rsyslog.d/**。

2. 重新启动 **rsyslog** 服务。

```
# systemctl restart rsyslog
```

验证

要验证客户端系统向服务器发送信息，请按照以下步骤执行：

1. 在客户端系统中发送测试信息：

```
# logger test
```

2. 在服务器系统上，查看 **/var/log/messages** 日志，例如：

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

-

其中 *hostname* 是客户端系统的主机名。请注意，日志包含输入 **logger** 命令的用户的用户名，本例中为 **root**。

其他资源

- **rsyslogd(8)** 和 **rsyslog.conf(5)** 手册页。
- 在 **/usr/share/doc/rsyslog/html/index.html** 文件中通过 **rsyslog-doc** 软件包安装的文档。

9.1.5. 配置 TLS 加密的远程日志记录

默认情况下，Rsyslog 以纯文本格式发送 remote-logging 通信。如果您的场景需要保护这个通信频道，您可以使用 TLS 加密它。

要通过 TLS 使用加密传输，请同时配置服务器和客户端。服务器收集和分析由一个或多个客户端系统发送的日志。

您可以使用 **ossl** 网络流驱动程序(OpenSSL)或 **gtls** 流驱动程序(GnuTLS)。

注意

如果您的系统具有更高的安全性，例如，没有连接到任何网络或有严格授权的系统，请使用独立的系统作为认证授权(CA)。

您可以使用 **全局**、**模块** 和 **输入** 级别在服务器端使用流驱动程序在 **全局** 和操作级别上自定义连接设置。**更为具体的配置会覆盖更常规的配置。例如，您可以对大多数连接和 gtls 在全局设置中使用 ossl，而只为特定连接使用 gtls。**

先决条件

- **有对客户端和服务器系统的 root 访问权限。**
- **以下软件包安装在服务器和客户端系统中：**
  - **rsyslog 软件包。**
  - **对于 ossl 网络流驱动程序，rsyslog-openssl 软件包。**
  - **对于 gtls 网络流驱动程序，rsyslog-gnutls 软件包。**
  - **要使用 certtool 命令(gnutls-utils 软件包)生成证书。**

- 在您的日志服务器中，以下证书位于 /etc/pki/ca-trust/source/anchors/ 目录中，并使用 update-ca-trust 命令更新您的系统配置：
  - ca-cert.pem - 一个 CA 证书，它可以在日志记录服务器和客户端上验证密钥和证书。
  - server-cert.pem - 日志记录服务器的公钥。
  - server-key.pem - 日志记录服务器的私钥。
- 在您的日志记录客户端中，以下证书位于 /etc/pki/ca-trust/source/anchors/ 目录中，并使用 update-ca-trust 来更新您的系统配置：
  - ca-cert.pem - 一个 CA 证书，它可以在日志记录服务器和客户端上验证密钥和证书。
  - client-cert.pem - 客户端的公钥。
  - client-key.pem - 客户端的私钥。

流程

1. 配置服务器以从您的客户端系统接收加密日志：
   a. 在 /etc/rsyslog.d/ 目录中创建一个新文件，例如 securelogser.conf。
   b. 要加密通信，配置文件必须包含指向服务器的证书文件的路径、所选身份验证方法，以及支持 TLS 加密的流驱动程序。在 /etc/rsyslog.d/securelogser.conf 文件中添加以下行：

```
# 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"
)
```

注意

如果您更喜欢 GnuTLS 驱动程序，请使用 StreamDriver.Name="gtls" 配置选项。有关比 x509/name 严格性低的验证模式的更多信息，请参阅使用 rsyslog-doc 软件包安装的文档。

c. 将更改保存到 /etc/rsyslog.d/securelogser.conf 文件。

d. 验证 /etc/rsyslog.conf 文件的语法以及 /etc/rsyslog.d/ 目录中的任何文件：

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

e. 确保 rsyslog 服务在日志记录服务器中运行并启用：

```
# systemctl status rsyslog
```

f. 重启 rsyslog 服务：

```
# systemctl restart rsyslog
```

g. 可选：如果没有启用 Rsyslog，请确保 rsyslog 服务在重启后自动启动：

```
# systemctl enable rsyslog
```

2. **配置客户端以将加密日志发送到服务器：**

a. **在客户端系统上，在 /etc/rsyslog.d/ 目录中创建一个名为 的新文件，如 securelogcli.conf。**

b. **在 /etc/rsyslog.d/securelogcli.conf 文件中添加以下行：**

```
# 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"
)
```

注意

如果您更喜欢 GnuTLS 驱动程序，请使用 StreamDriver.Name="gtls" 配置选项。

c. **将更改保存到 /etc/rsyslog.d/securelogcli.conf 文件。**

d. **验证 /etc/rsyslog.conf 文件的语法以及 /etc/rsyslog.d/ 目录中的其他文件：**

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

e. **确保 rsyslog 服务在日志记录服务器中运行并启用：**

```
# systemctl status rsyslog
```

f. 重启 rsyslog 服务：

```
# systemctl restart rsyslog
```

g. 可选： 如果没有启用 Rsyslog，请确保 rsyslog 服务在重启后自动启动：

```
# systemctl enable rsyslog
```

验证

要验证客户端系统向服务器发送信息，请按照以下步骤执行：

1. 在客户端系统中发送测试信息：

```
# logger test
```

2. 在服务器系统上，查看 /var/log/messages 日志，例如：

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

其中 <hostname> 是客户端系统的主机名。请注意，该日志包含输入 logger 命令的用户的用户名，本例中为 root。

其他资源

- certtool (1), openssl (1), update-ca-trust (8), rsyslogd (8), 和 rsyslog.conf (5) 手册页。
- 在 /usr/share/doc/rsyslog/html/index.html 上安装了 rsyslog-doc 软件包的文档。
- 将 logging 系统角色与 TLS 一起使用。

9.1.6. 配置服务器以通过 UDP 接收远程日志信息

Rsyslog 应用程序可让您将系统配置为从远程系统接收日志信息。要通过 UDP 使用远程日志记录，请

同时配置服务器和客户端。接收服务器收集并分析一个或多个客户端系统发送的日志。默认情况下，rsyslog 使用端口 514 上的 UDP 从远程系统接收日志信息。

按照以下步骤配置服务器，以通过 UDP 协议收集和分析一个或多个客户端系统发送的日志。

先决条件

- rsyslog 已安装在服务器系统上。
- 您以 root 身份登录到服务器中。
- 使用 semanage 命令，为可选步骤安装 policycoreutils-python-utils 软件包。
- firewalld 服务在运行。

流程

1. 可选：要将不同的端口用于 rsyslog 流量，而不是默认端口 514 ：

   a. 将 syslogd_port_t SELinux 类型添加到 SELinux 策略配置中，使用您要 rsyslog 的端口号替换 *portno* ：

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

   b. 配置 firewalld 以允许传入的 rsyslog 流量，使用您要 rsyslog 使用的端口替换 *portno*，区替换 zone：

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

   c. 重新载入防火墙规则：

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

2.

在 /etc/rsyslog.d/ 目录中创建一个新的 .conf 文件，如 remotelogserv.conf，并插入以下内容：

```
# 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")
```

其中 514 是 rsyslog 默认使用的端口号。您可以指定不同的端口。

3. 验证 /etc/rsyslog.conf 文件以及 /etc/rsyslog.d/ 目录中的所有 .conf 文件的语法：

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

4. 重新启动 rsyslog 服务。

```
# systemctl restart rsyslog
```

5. 可选： 如果没有启用 rsyslog，请确保 rsyslog 服务在重启后自动启动：

```
# systemctl enable rsyslog
```

其他资源

- **rsyslogd(8) , rsyslog.conf(5), semanage(8), 和 firewall-cmd(1) man page。**
- **在 /usr/share/doc/rsyslog/html/index.html 文件中通过 rsyslog-doc 软件包安装的文档。**

9.1.7. 通过 UDP 配置远程日志记录到服务器

您可以配置系统，以通过 UDP 协议将日志消息转发到服务器。omfwd 插件通过 UDP 或 TCP 提供转发。默认协议是 UDP。因为插件内置在内，所以不必加载它。

先决条件

- **rsyslog 软件包安装在应该向服务器报告的客户端系统上。**
- **您已为远程日志记录配置了服务器，如配置服务器 以通过 UDP 接收远程日志信息。**

流程

1. **在 /etc/rsyslog.d/ 目录中创建一个新的 .conf 文件，如 10-remotelogcli.conf，并插入以下内容：**

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

其中：

- **queue.type="linkedlist" 设置启用一个 LinkedList 内存中队列。**
- **queue.filename 设置定义磁盘存储。备份文件使用之前全局 workDirectory 指令指定**

的工作目录中的 *example_fwd* 前缀创建。

- action.resumeRetryCount -1 设置可防止 rsyslog 在重试时丢弃消息（如果服务器没有响应）。
- 如果 rsyslog 关闭，启用的 queue.saveOnShutdown="on" 设置会保存内存中的数据。
- *portno* 值是您要 rsyslog 使用的端口号。默认值为 514。
- 最后一行将所有收到的消息转发到日志记录服务器，端口规格是可选的。

使用这个配置，rsyslog 会向服务器发送消息，但如果远程服务器无法访问，则会将消息保留在内存中。只有 rsyslog 耗尽配置的内存队列空间或需要关闭时，才能创建磁盘上的文件，从而让系统性能受益。

注意

rsyslog 按照一般顺序处理配置文件 /etc/rsyslog.d/。

2. 重新启动 rsyslog 服务。

```
# systemctl restart rsyslog
```

3. 可选： 如果没有启用 rsyslog，请确保 rsyslog 服务在重启后自动启动：

```
# systemctl enable rsyslog
```

验证

要验证客户端系统向服务器发送信息，请按照以下步骤执行：

1. 在客户端系统中发送测试信息：

```
# logger test
```

2. 在服务器系统中，查看 /var/log/remote/msg/*hostname*/root.log 日志，例如：

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

其中 hostname 是客户端系统的主机名。请注意，该日志包含输入 logger 命令的用户的用户名，本例中为 root。

其他资源

- rsyslogd(8) 和 rsyslog.conf(5) 手册页。
- 在 /usr/share/doc/rsyslog/html/index.html 上安装了 rsyslog-doc 软件包的文档。

9.1.8. Rsyslog 中的负载均衡帮助程序

RebindInterval 设置指定当前连接中断的时间间隔，并被重新建立。此设置适用于 TCP、UDP 和 RELP 流量。负载平衡器将信息作为新连接，并将消息转发到另一个物理目标系统。

当目标系统更改其 IP 地址时，RebindInterval 设置非常有用。Rsyslog 应用程序在连接建立时缓存 IP 地址，因此信息会发送到同一服务器。如果 IP 地址更改，UDP 数据包将会丢失，直到 Rsyslog 服务重启为止。重新建立连接将确保 DNS 再次解析 IP。

```
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. 配置可靠的远程日志记录

通过可靠的事件日志记录协议(RELP)，您可以降低消息丢失的风险通过 TCP 发送和接收 syslog 消息。RELP 提供可靠的事件消息交付，这对于无法接受消息丢失的环境中非常有用。要使用 RELP，请配置服务器上运行的 imrelp 输入模块并接收日志，以及在客户端上运行的 omrelp 输出模块，并将日志发送到日志记录服务器。

先决条件

- 您已在服务器和客户端系统中安装了 rsyslog、librelp 和 rsyslog-relp 软件包。
- 在 SELinux 中允许指定的端口并在防火墙中打开。

流程

1. 配置客户端系统以可靠远程记录：

   a. 在客户端系统上，在 /etc/rsyslog.d/ 目录中创建一个新的 .conf 文件，例如 relpclient.conf，并插入以下内容：

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

   其中：

   - *target_IP* 是日志记录服务器的 IP 地址。
   - *target_port* 是日志记录服务器的端口。

   b. 保存对 /etc/rsyslog.d/relpclient.conf 文件的更改。

   c. 重新启动 rsyslog 服务。

   ```
   # systemctl restart rsyslog
   ```

   d. 可选：如果没有启用 rsyslog，请确保 rsyslog 服务在重启后自动启动：

   ```
   # systemctl enable rsyslog
   ```

2. 配置服务器系统以可靠远程记录：

   a. 在服务器系统中，在 /etc/rsyslog.d/ 目录中创建一个新的 .conf 文件，例如

reserv.conf，并插入以下内容：

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


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

其中：

- *log_path* 指定存储消息的路径。
- *target_port* 是日志记录服务器的端口。使用与客户端配置文件中相同的值。

b. 保存对 /etc/rsyslog.d/relpserv.conf 文件的更改。

c. 重新启动 rsyslog 服务。

```
# systemctl restart rsyslog
```

d. 可选：如果没有启用 rsyslog，请确保 rsyslog 服务在重启后自动启动：

```
# systemctl enable rsyslog
```

验证

要验证客户端系统向服务器发送信息，请按照以下步骤执行：

1. 在客户端系统中发送测试信息：

```
# logger test
```

2. 在服务器系统中，查看指定 *log_path* 的日志，例如：

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

其中 hostname 是客户端系统的主机名。请注意，该日志包含输入 logger 命令的用户的用户名，本例中为 root。

其他资源

- rsyslogd(8) 和 rsyslog.conf(5) 手册页。
- 在 /usr/share/doc/rsyslog/html/index.html 文件中通过 rsyslog-doc 软件包安装的文档。

9.1.10. 支持的 Rsyslog 模块

要扩展 Rsyslog 应用程序的功能，您可以使用特定的模块。模块提供额外的输入(Input 模块)、输出（输出模块）和其他功能。模块也可以提供在加载模块后可用的其他配置指令。

您可以输入以下命令列出您系统中安装的输入和输出模块：

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

在安装了 rsyslog-doc 软件包后，您可以在 /usr/share/doc/rsyslog/html/configuration/modules/idx_output.html 文件中查看所有可用的 rsyslog 模块。

9.1.11. 配置 netconsole 服务为将内核信息记录到远程主机

当无法登录到磁盘或使用串行控制台时，您可以使用 netconsole 内核模块和同名服务将内核消息通过网络记录到远程 rsyslog 服务。

先决条件

- 远程主机上已安装系统日志服务，如 rsyslog。
- 远程系统日志服务被配置为接收来自此主机的日志条目。

流程

1. 安装 netconsole-service 软件包：

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

2. 编辑 /etc/sysconfig/netconsole 文件，并将 SYSLOGADDR 参数设为远程主机的 IP 地址：

```
# SYSLOGADDR=192.0.2.1
```

3. 启用并启动 netconsole 服务：

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

验证步骤

- 在远程系统日志服务器上显示 /var/log/messages 文件。

其他资源

配置远程日志记录解决方案

9.1.12. 其他资源

- 在 /usr/share/doc/rsyslog/html/index.html 文件中安装有 rsyslog-doc 软件包的文档
- rsyslog.conf (5) 和 rsyslogd (8) man page
- 在不使用 journald 或最小化 journald 的情况下配置系统日志记录知识库文章。
- RHEL 默认日志设置对性能的负面影响及其缓解措施
- 使用日志记录系统角色 章节

9.2. 使用 LOGGING 系统角色

作为系统管理员，您可以使用 logging 系统角色将 Red Hat Enterprise Linux 主机配置为日志服务器，以从多个客户端系统收集日志。

9.2.1. logging RHEL 系统角色

使用 logging RHEL 系统角色，您可以在本地和远程主机上部署日志记录配置。

日志记录解决方案提供多种读取日志和多个日志记录输出的方法。

例如，日志记录系统可接受以下输入：

- 本地文件
- systemd/journal
- 网络上的另一个日志记录系统

另外，日志记录系统还可有以下输出：

- 日志存储在 /var/log 目录中的本地文件中
- 日志被发送到 Elasticsearch
- 日志被转发到另一个日志系统

使用 logging RHEL 系统角色，您可以组合输入和输出以适应您的场景。例如，您可以配置一个日志解决方案，将来自 日志 的输入存储在本地文件中，而从文件读取的输入则被转发到另一个日志系统，并存储在本地日志文件中。

其他资源

- /usr/share/ansible/roles/rhel-system-roles.logging/README.md 文件
- /usr/share/doc/rhel-system-roles/logging/ 目录
- RHEL 系统角色

9.2.2. 应用本地 日志记录 RHEL 系统角色

准备和应用 Ansible playbook，以便在一组独立的机器上配置日志记录解决方案。每台机器在本地记录日志。

先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 sudo 权限。

注意

您不必安装 rsyslog 软件包，因为 RHEL 系统角色在部署时安装了 rsyslog。

流程

1. 创建一个包含以下内容的 playbook 文件，如 ~/playbook.yml ：

```
---
- 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. 验证 playbook 语法：

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

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

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

验证

1. 测试 /etc/rsyslog.conf 文件的语法：

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

2. 验证系统是否向日志发送信息：

   a. 发送测试信息：

   ```
   # logger test
   ```

   b. 查看 /var/log/messages 日志，例如：

   ```
   # cat /var/log/messages
   Aug  5 13:48:31 <hostname> root[6778]: test
   ```

   其中 *<hostname>* 是客户端系统的主机名。请注意，该日志包含输入 logger 命令的用户的用户名，本例中为 root。

其他资源

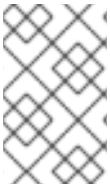
- /usr/share/ansible/roles/rhel-system-roles.logging/README.md 文件
- /usr/share/doc/rhel-system-roles/logging/ 目录

9.2.3. 过滤本地 日志 RHEL 系统角色中的日志

您可以部署一个日志解决方案，该方案基于 rsyslog 属性的过滤器过滤日志。

先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 sudo 权限。



注意

您不必安装 rsyslog 软件包，因为 RHEL 系统角色在部署时安装了 rsyslog。

流程

1. 创建一个包含以下内容的 playbook 文件，如 ~/playbook.yml ：

```
---
- 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]
```

使用这个配置，所有包含 error 字符串的消息都记录在 /var/log/errors.log 中，所有其他消息都记录在 /var/log/others.log 中。

您可以将 error 属性值替换为您要用来过滤的字符串。

您可以根据您的偏好修改变量。

2. 验证 playbook 语法：

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

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

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

验证

1. 测试 /etc/rsyslog.conf 文件的语法：

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

2. 验证系统是否向日志发送包含 error 字符串的信息：

a. 发送测试信息：

```
# logger error
```

b. 查看 /var/log/errors.log 日志，例如：

```
# cat /var/log/errors.log
Aug  5 13:48:31 hostname root[6778]: error
```

其中 *hostname* 是客户端系统的主机名。请注意，该日志包含输入 logger 命令的用户的用户名，本例中为 root。

其他资源

- /usr/share/ansible/roles/rhel-system-roles.logging/README.md 文件
- /usr/share/doc/rhel-system-roles/logging/ 目录

9.2.4. 使用 logging RHEL 系统角色应用远程日志解决方案

按照以下步骤准备并应用 Red Hat Ansible Core playbook 来配置远程日志记录解决方案。在本 playbook 中，一个或多个客户端从 systemd-journal 获取日志，并将它们转发到远程服务器。服务器从 remote_rsyslog 和 remote_files 接收远程输入，并将日志输出到由远程主机名命名的目录中的本地文件。

先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 sudo 权限。

注意

您不必安装 rsyslog 软件包，因为 RHEL 系统角色在部署时安装了 rsyslog。

流程

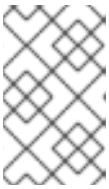
1. 创建一个包含以下内容的 playbook 文件，如 ~/playbook.yml ：

```
---
- 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]
```

其中 *<host1.example.com>* 是日志记录服务器。



注意

您可以修改 playbook 中的参数以符合您的需要。

警告

日志解决方案只适用于在服务器或者客户端系统的 SELinux 策略中定义的端口并在防火墙中打开。默认 SELinux 策略包括端口 601、514、6514、10514 和 20514。要使用其他端口，请修改客户端和服务器系统上的 SELinux 策略。

2. 验证 playbook 语法：

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

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

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

验证

1. 在客户端和服务器系统上测试 /etc/rsyslog.conf 文件的语法：

```
# 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. 验证客户端系统向服务器发送信息：

a. 在客户端系统中发送测试信息：

```
# logger test
```

b. 在服务器系统上，查看 /var/log/*<host2.example.com>*/messages 日志，例如：

```
# cat /var/log/<host2.example.com>/messages
Aug  5 13:48:31 <host2.example.com> root[6778]: test
```

其中 *<host2.example.com>* 是客户端系统的主机名。请注意，该日志包含输入 logger 命令的用户的用户名，本例中为 root。

其他资源

- /usr/share/ansible/roles/rhel-system-roles.logging/README.md 文件
- /usr/share/doc/rhel-system-roles/logging/ 目录

9.2.5. 使用带有 TLS 的 logging RHEL 系统角色

传输层安全性(TLS)是一种加密协议，旨在允许计算机网络上的安全通信。

作为管理员，您可以使用 logging RHEL 系统角色来使用 Red Hat Ansible Automation Platform 配置日志的安全传输。

9.2.5.1. 配置带有 TLS 的客户端日志

您可以使用 Ansible playbook 和 logging RHEL 系统角色，来在 RHEL 客户端上配置日志记录，并使用 TLS 加密将日志传送到远程日志记录系统。

此流程创建一个私钥和证书，并在 Ansible 清单中客户端组的所有主机上配置 TLS。TLS 对信息的传输进行加密，确保日志在网络安全传输。

注意

您不必在 playbook 中调用 certificate RHEL 系统角色来创建证书。logging RHEL 系统角色会自动调用它。

要让 CA 能够为创建的证书签名，受管节点必须在 IdM 域中注册。

先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 sudo 权限。
- 受管节点已在 IdM 域中注册。

流程

1. 创建一个包含以下内容的 playbook 文件，如 ~/playbook.yml ：

```
---
- 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]
```

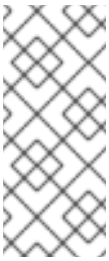
playbook 使用以下参数：

logging_certificates

此参数的值被传给 certificate RHEL 系统角色中的 certificate_requests，并用来创建私钥和证书。

logging_pki_files

使用这个参数，您可以配置日志记录用来查找 CA 证书和用于 TLS 的密钥文件的路径和其他设置，使用以下子参数指定：ca_cert、ca_cert_src 、cert、cert_src、private_key、private_key_src 和 tls。



注意

如果您使用 logging_certificates 在目标节点上创建文件，请不要使用 ca_cert_src、cert_src 和 private_key_src，它们用于复制由 logging_certificates 创建的文件。

ca_cert

代表目标节点上 CA 证书文件的路径。默认路径为 /etc/pki/tls/certs/ca.pem，文件名由用户设置。

cert

表示目标节点上证书文件的路径。默认路径为 /etc/pki/tls/certs/server-cert.pem，文件名由用户设置。

private_key

代表目标节点上私钥文件的路径。默认路径为 /etc/pki/tls/private/server-key.pem，文件名由用户设置。

ca_cert_src

代表控制节点上 CA 证书文件的路径，该路径被复制到目标主机上由 ca_cert 指定的位置。如果使用 logging_certificates，请不要使用它。

cert_src

代表控制节点上证书文件的路径，该文件的路径被复制到目标主机上由 cert 指定的位置。如果使用 logging_certificates，请不要使用它。

private_key_src

代表控制节点上私钥文件的路径，该路径被复制到目标主机上由 private_key 指定的位置。如果使用 logging_certificates，请不要使用它。

tls

将此参数设为 true 以确保通过网络安全地传输日志。如果您不想要一个安全的包装程序，您可以设置 tls: false。

2. 验证 playbook 语法：

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

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

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

其他资源

- /usr/share/ansible/roles/rhel-system-roles.logging/README.md 文件
- /usr/share/doc/rhel-system-roles/logging/ 目录
- 使用 RHEL 系统角色请求证书。

9.2.5.2. 配置带有 TLS 的服务器日志

您可以使用 Ansible playbook 和 logging RHEL 系统角色，来在 RHEL 服务器上配置日志记录，并

将它们设置为使用 TLS 加密从远程日志记录系统接收日志。

此流程创建一个私钥和证书，并在 Ansible 清单中服务器组中的所有主机上配置 TLS。

注意

您不必在 playbook 中调用 certificate RHEL 系统角色来创建证书。logging RHEL 系统角色会自动调用它。

要让 CA 能够为创建的证书签名，受管节点必须在 IdM 域中注册。

先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 sudo 权限。
- 受管节点已在 IdM 域中注册。

流程

1. 创建一个包含以下内容的 playbook 文件，如 ~/playbook.yml ：

```
---
- 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]
```

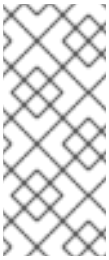
playbook 使用以下参数：

logging_certificates

此参数的值被传给 certificate RHEL 系统角色中的 certificate_requests，并用来创建私钥和证书。

logging_pki_files

使用这个参数，您可以配置日志记录用来查找 CA 证书和用于 TLS 的密钥文件的路径和其他设置，使用以下子参数指定：ca_cert、ca_cert_src、cert、cert_src、private_key、private_key_src 和 tls。



注意

如果您使用 logging_certificates 在目标节点上创建文件，请不要使用 ca_cert_src、cert_src 和 private_key_src，它们用于复制由 logging_certificates 创建的文件。

ca_cert

代表目标节点上 CA 证书文件的路径。默认路径为 /etc/pki/tls/certs/ca.pem，文件名由用户设置。

cert

表示目标节点上证书文件的路径。默认路径为 /etc/pki/tls/certs/server-cert.pem，文件名由用户设置。

private_key

代表目标节点上私钥文件的路径。默认路径为 **/etc/pki/tls/private/server-key.pem**，文件名由用户设置。

ca_cert_src

代表控制节点上 **CA** 证书文件的路径，该路径被复制到目标主机上由 **ca_cert** 指定的位置。如果使用 **logging_certificates**，请不要使用它。

cert_src

代表控制节点上证书文件的路径，该文件的路径被复制到目标主机上由 **cert** 指定的位置。如果使用 **logging_certificates**，请不要使用它。

private_key_src

代表控制节点上私钥文件的路径，该路径被复制到目标主机上由 **private_key** 指定的位置。如果使用 **logging_certificates**，请不要使用它。

tls

将此参数设为 **true** 以确保通过网络安全地传输日志。如果您不想要一个安全的包装程序，您可以设置 **tls: false**。

2. 验证 **playbook** 语法：

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

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 **playbook**：

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

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.logging/README.md** 文件
- **/usr/share/doc/rhel-system-roles/logging/** 目录

- 使用 RHEL 系统角色请求证书。

9.2.6. 使用带有 RELP 的日志记录 RHEL 系统角色

可靠的事件日志协议(RELP)是一种通过 TCP 网络记录数据和消息的网络协议。它确保了事件消息的可靠传递，您可以在不容许任何消息丢失的环境中使用它。

RELP 发送者以命令的形式传输日志条目，接收者在处理后确认这些条目。为确保一致性，RELP 将事务数保存到传输的命令中，以便进行任何类型的消息恢复。

您可以考虑在 RELP 客户端和 RELP Server 间的远程日志系统。RELP 客户端将日志传送给远程日志系统，RELP 服务器接收由远程日志系统发送的所有日志。

管理员可以使用 logging RHEL 系统角色将日志记录系统配置为可靠地发送和接收日志条目。

9.2.6.1. 配置带有 RELP 的客户端日志

您可以使用 logging RHEL 系统角色在 RHEL 系统中配置日志记录，这些日志记录在本地机器上，并可以通过运行 Ansible playbook 将日志转发到带有 RELP 的远程日志记录系统。

此流程对 Ansible 清单中 客户端 组中的所有主机配置 RELP。RELP 配置使用传输层安全(TLS)来加密消息传输，保证日志在网络上安全传输。

先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 sudo 权限。

流程

1. 创建一个包含以下内容的 playbook 文件，如 ~/playbook.yml ：

```
---
- 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]
```

playbook 使用以下设置：

target

这是一个必需的参数，用于指定运行远程日志系统的主机名。

port

远程日志记录系统正在监听的端口号。

tls

确保日志在网络上安全地传输。如果您不想要安全打包程序，可以将 tls 变量设置为 false。在与 RELP 工作时，默认的 tls 参数被设置为 true，且需要密钥/证书和 triplets {ca_cert、cert、private_key} 和/或 {ca_cert_src,cert_src,private_key_src}。

- 如果设置了 {ca_cert_src,cert_src,private_key_src} 三元组，则默认位置 /etc/pki/tls/certs 和 /etc/pki/tls/private 被用作受管节点上的目的地，以便从控制节点传输文件。在这种情况下，文件名与 triplet 中的原始名称相同

- 如果设置了 {ca_cert,cert,private_key} 三元组，则文件在日志配置前应位于默认路径上。

- 如果两个三元组都设置了，则文件将从控制节点的本地路径传输到受管节点的特定路径。

ca_cert

表示 CA 证书的路径。默认路径为 /etc/pki/tls/certs/ca.pem，文件名由用户设置。

cert

表示证书的路径。默认路径为 /etc/pki/tls/certs/server-cert.pem，文件名由用户设置。

private_key

表示私钥的路径。默认路径为 /etc/pki/tls/private/server-key.pem，文件名由用户设置。

ca_cert_src

表示复制到目标主机的本地 CA 证书文件路径。如果指定了 ca_cert，则会将其复制到该位置。

cert_src

表示复制到目标主机的本地证书文件路径。如果指定了 cert，则会将其复制到该位置。

private_key_src

表示复制到目标主机的本地密钥文件的路径。如果指定了 private_key，则会将其复制到该位置。

pki_authmode

接受身份验证模式为 name 或 fingerprint。

permitted_servers

日志客户端允许通过 TLS 连接和发送日志的服务器列表。

输入

日志输入字典列表。

输出

日志输出字典列表。

2. 验证 playbook 语法：

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

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

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

其他资源

- /usr/share/ansible/roles/rhel-system-roles.logging/README.md 文件
- /usr/share/doc/rhel-system-roles/logging/ 目录

9.2.6.2. 配置带有 RELP 的服务器日志

您可以使用 logging RHEL 系统角色在作为服务器的 RHEL 系统中配置日志记录，并可以通过运行 Ansible playbook 从具有 RELP 的远程日志记录系统接收日志。

此流程对 Ansible 清单中 服务器 组中的所有主机配置 RELP。RELP 配置使用 TLS 加密消息传输，以保证在网络上安全地传输日志。

先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。

- 用于连接到受管节点的帐户具有 sudo 权限。

流程

1. 创建一个包含以下内容的 playbook 文件，如 ~/playbook.yml ：

```
---
- 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
```

playbook 使用以下设置：

port

远程日志记录系统正在监听的端口号。

tls

确保日志在网络上安全地传输。如果您不想要安全打包程序，可以将 tls 变量设置为 false。在与 RELP 工作时，默认的 tls 参数被设置为 true，且需要密钥/证书和 triplets {ca_cert、cert、private_key} 和/或 {ca_cert_src,cert_src,private_key_src}。

- 如果设置了 {ca_cert_src,cert_src,private_key_src} 三元组，则默认位置 /etc/pki/tls/certs 和 /etc/pki/tls/private 被用作受管节点上的目的地，以便从控制节点传输文件。在这种情况下，文件名与 triplet 中的原始名称相同

- 如果设置了 {ca_cert,cert,private_key} 三元组，则文件在日志配置前应位于默认路径上。

- 如果两个三元组都设置了，则文件将从控制节点的本地路径传输到受管节点的特定路径。

ca_cert

表示 CA 证书的路径。默认路径为 /etc/pki/tls/certs/ca.pem，文件名由用户设置。

cert

表示证书的路径。默认路径为 /etc/pki/tls/certs/server-cert.pem，文件名由用户设置。

private_key

表示私钥的路径。默认路径为 /etc/pki/tls/private/server-key.pem，文件名由用户设置。

ca_cert_src

表示复制到目标主机的本地 CA 证书文件路径。如果指定了 ca_cert，则会将其复制到该位置。

cert_src

表示复制到目标主机的本地证书文件路径。如果指定了 cert，则会将其复制到该位置。

private_key_src

表示复制到目标主机的本地密钥文件的路径。如果指定了 private_key，则会将其复制到该位置。

pki_authmode

接受身份验证模式为 name 或 fingerprint。

permitted_clients

日志记录服务器允许通过 TLS 连接和发送日志的客户端列表。

输入

日志输入字典列表。

输出

日志输出字典列表。

2. 验证 playbook 语法：

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

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

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

其他资源

- /usr/share/ansible/roles/rhel-system-roles.logging/README.md 文件
- /usr/share/doc/rhel-system-roles/logging/ 目录

9.2.7. 其他资源

- 准备一个控制节点和受管节点以使用 RHEL 系统角色
- 随 rhel-system-roles 软件包安装在 /usr/share/ansible/roles/rhel-system-roles.logging/README.html 中的文档。
- RHEL 系统角色
- ansible-playbook(1) 手册页。

第 10 章 使用日志文件对问题进行故障排除

日志文件包含有关系统的消息，包括内核、服务及其上运行的应用。这些信息可帮助故障排除问题或监控系统功能。Red Hat Enterprise Linux 中的日志记录系统是基于内置的 syslog 协议的。特定的程序使用这个系统记录事件并将其整理到日志文件中，这些文件在审核操作系统和故障排除各种问题时非常有用。

10.1. 处理 SYSLOG 信息的服务

以下两个服务处理 syslog 信息：

- systemd-journald 守护进程
- Rsyslog 服务

systemd-journald 守护进程收集来自各种来源的信息并将其转发到 Rsyslog 以便进一步处理。systemd-journald 守护进程从以下来源收集信息：

- 内核
- 引导过程的早期阶段
- 启动并运行守护进程的标准和错误输出
- Syslog

Rsyslog 服务根据类型和优先权对 syslog 信息进行排序，并将其写入 /var/log 目录下的文件中。/var/log 目录会永久保存日志信息。

10.2. 存储 SYSLOG 信息的子目录

/var/log 下的以下子目录保存了 syslog 信息。

- /var/log/messages - 除以下外的所有 syslog 信息
- /var/log/secure - 与安全和验证相关的信息和错误
- /var/log/maillog - 与邮件服务器相关的信息和错误
- /var/log/cron - 与定期执行的任务相关的日志文件
- /var/log/boot.log - 与系统启动相关的日志文件

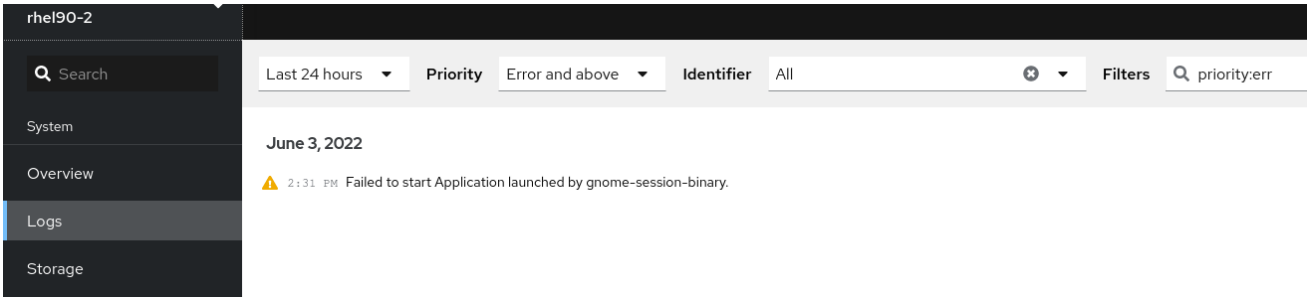
10.3. 使用 WEB 控制台检查日志文件

按照此流程中的步骤，使用 RHEL web 控制台来检查日志文件。

流程

1. 登录到 RHEL web 控制台。详情请参阅 登录到 Web 控制台。
2. 点 Logs。

图 10.1. 在 RHEL 8 web 控制台中检查日志文件



10.4. 使用命令行查看日志

Journal 是 systemd 的一个组件，可帮助查看和管理日志文件。它解决了与传统日志记录相关的问题，与系统的其余部分紧密集成，并且支持各种日志记录技术以及日志文件的访问管理。

您可以通过命令行，使用 journalctl 命令查看系统日志中的信息，例如：

```
$ 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
...
```

表 10.1. 查看系统信息

命令	描述
journalctl	显示所有收集的日志条目。
journalctl FILEPATH	显示与特定文件相关的日志。例如：**journalctl /dev/sda** 命令显示与 **/dev/sda** 文件系统相关的日志。
journalctl -b	显示当前引导的日志。
journalctl -k -b -1	显示当前引导的内核日志。

表 10.2. 查看有关特定服务的信息

命令	描述
journalctl -b _SYSTEMD_UNIT=*<name.service>*	过滤日志以显示与 **systemd** 服务匹配的条目。
journalctl -b _SYSTEMD_UNIT=*<name.service>* _PID=*<number>*	合并匹配。例如，这个命令显示与***<name.service>*** 和 PID ***<number>*** 匹配的 **systemd-units** 的日志。
journalctl -b _SYSTEMD_UNIT=*<name.service>* _PID=*<number>* + _SYSTEMD_UNIT=*<name2.service>*	加号(+)分隔符将两个表达式按逻辑 OR 组合在一起。例如，这个命令显示来自带有 **PID** 的 ***<name.service>*** 服务进程的所有消息，加上来自 ***<name2.service>*** 服务（来自其任何进程）的所有消息。
journalctl -b _SYSTEMD_UNIT=*<name.service>* _SYSTEMD_UNIT=*<name2.service>*	此命令显示与引用同一字段的任一表达式匹配的所有条目。在这里，这个命令会显示与 systemd-unit ***<name.service>*** 或 systemd-unit ***<name2.service>*** 匹配的日志。

表 10.3. 查看与特定引导相关的日志

命令	描述

命令	描述
journalctl --list-boots	显示引导号、其 ID 以及与引导相关的第一条和最后一个消息的时间戳列表。您可以在下一个命令中使用 ID 来查看详细信息。
journalctl --boot=ID _SYSTEMD_UNIT=*<name.service>*	显示有关指定的引导 ID 的信息。

10.5. 其他资源

- **man journalctl(1)**
- **配置远程日志记录解决方案**

第 11 章 管理用户和组

防止对文件和流程的未经授权的访问需要一个准确的用户和组管理。如果您不集中管理帐户，或者仅需要特定系统上的用户帐户或组，则您可以在此主机上本地创建它们。

11.1. 管理用户和组帐户简介

用户和组群的控制是 Red Hat Enterprise Linux(RHEL)系统管理的核心元素。每个 RHEL 用户都有不同的登录凭证，并可分配给不同的组以自定义其系统权限。

11.1.1. 用户和组介绍

创建文件的用户是该文件的拥有者*以及*该文件的组所有者。这个文件会单独为拥有者、组和组以外的成员分配读、写和执行权限。文件所有者只能由 root 用户更改。root 用户和文件拥有者都可以更改对该文件的访问权限。常规用户可以将他们拥有的文件的组群所有权改为他们所属的组。

每个用户都与一个唯一数字身份号关联，称为 ***user ID*** (UID)。每个组都与一个 ***group ID*** (GID)关联。组群中的用户共享相同的读取、写入和执行该组所拥有的文件的权限。

11.1.2. 配置保留的用户和组群 ID

RHEL 为系统用户和组保留在 1000 以下的用户和组群 ID。您可以在 setup 软件包中找到保留的用户和组群 ID。要查看保留的用户和组群 ID，请使用：

```
cat /usr/share/doc/setup*/uidgid
```

建议从 5000 开始将 ID 分配给新用户和组，因为保留范围将来可能会增加。

要使分配给新用户的 ID 默认从 5000 开始，修改 /etc/login.defs 文件中的 UID_MIN 和 GID_MIN 参数。

流程

要修改并使 ID 默认分配给从 5000 开始的新用户：

1. 在您选择的编辑器中打开 /etc/login.defs 文件。

2. 找到为自动 UID 选择定义最小值的行。

```
# Min/max values for automatic uid selection in useradd
#
UID_MIN                  1000
```

3. 修改 UID_MIN 值从 5000 开始。

```
# Min/max values for automatic uid selection in useradd
#
UID_MIN                  5000
```

4. 找到自动选择 GID 最小值的行。

```
# Min/max values for automatic gid selection in groupadd
#
GID_MIN                  1000
```

5. 修改 GID_MIN 值，以从 5000 开始。

```
# Min/max values for automatic gid selection in groupadd
#
GID_MIN                  5000
```

常规用户动态分配的 UID 和 GID 现在从 5000 开始。

注意

在更改 UID_MIN 和 GID_MIN 值之前创建的用户和组的 UID 和 GID 不会更改。

这将允许新用户的组拥有与 UID 和 GID 相同的 5000+ ID。

警告

不要通过更改 SYS_UID_MAX 来提高系统 1000 以上保留的 ID，以避免与保留 1000 限制的系统冲突。

11.1.3. 用户私人组群

RHEL 使用 *用户私人组群*（UPG）系统配置，这可让 UNIX 组更容易管理。无论何时在系统中添加新用户，都会创建一个用户私人组群。用户私人组群的名称与为其创建的用户的名称相同，该用户是该用户私人组群中的唯一成员。

UPG 简化了多个用户之间在项目上的协作。此外，UPG 系统配置可以安全地为新创建的文件或目录设置默认权限，因为它允许该用户和此用户所属的组对文件或目录进行修改。

所有组群列表都保存在 /etc/group 配置文件中。

11.2. 管理用户帐户入门

Red Hat Enterprise Linux 是一个多用户操作系统，可使不同计算机上的多个用户访问安装在一台计算机上的一个系统。每个用户都在自己的帐户下操作，因此管理用户帐户代表 Red Hat Enterprise Linux 系统管理的一个核心元素。

以下是不同类型的用户帐户：

- 普通用户帐户：

  为特定系统用户创建普通帐户。这些帐户可以在正常的系统管理过程中添加、删除和修改。

- 系统用户帐户：

  系统用户帐户代表系统中的特定应用程序标识符。此类帐户通常仅在软件安装时添加或操作，且不会在以后进行修改。

警告

系统帐户假定在一个系统中本地可用。如果这些帐户是远程配置和提供的，如 LDAP 配置的实例中，则可能会出现系统中断和服务启动故障。

对于系统帐户，1000 以下的用户 ID 被保留。对于普通帐户，使用从 1000 开始的 ID。但推荐做法是使用从 5000 开始的 ID。有关分配 ID 的信息，请查看 /etc/login.defs 文件。

- 组：

  组是出于共同目的将多个用户帐户连接在一起的实体，例如对特定文件授予访问权限。

11.2.1. 使用命令行工具管理帐户和组

使用以下基本命令行工具管理用户帐户和组。

- 显示用户和组群 ID:

```
$ 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
```

- 要创建新用户帐户，请执行以下操作：

```
# useradd example.user
```

- 为 *example.user* 所属用户帐户分配新密码：

```
# passwd example.user
```

- 将用户添加到组中：

```
# usermod -a -G example.group example.user
```

其他资源

- man useradd(8)、man passwd(1) 和 man usermod(8)

11.2.2. Web 控制台中管理的系统用户帐户

您可在 RHEL web 控制台中显示用户帐户：

- 在访问系统时验证用户。
- 设置系统的访问权限。

RHEL web 控制台显示系统中的所有用户帐户。因此，在首次登录 web 控制台后，至少可以看到一个可用的用户帐户。

登录到 RHEL web 控制台后，您可以执行以下操作：

- 创建新用户帐户。
- 更改其参数。
- 锁定帐户。
- 终止用户会话。

11.2.3. 使用 Web 控制台添加新帐户

您可以将用户帐户添加到系统，并通过 RHEL web 控制台为帐户设置管理权限。

先决条件

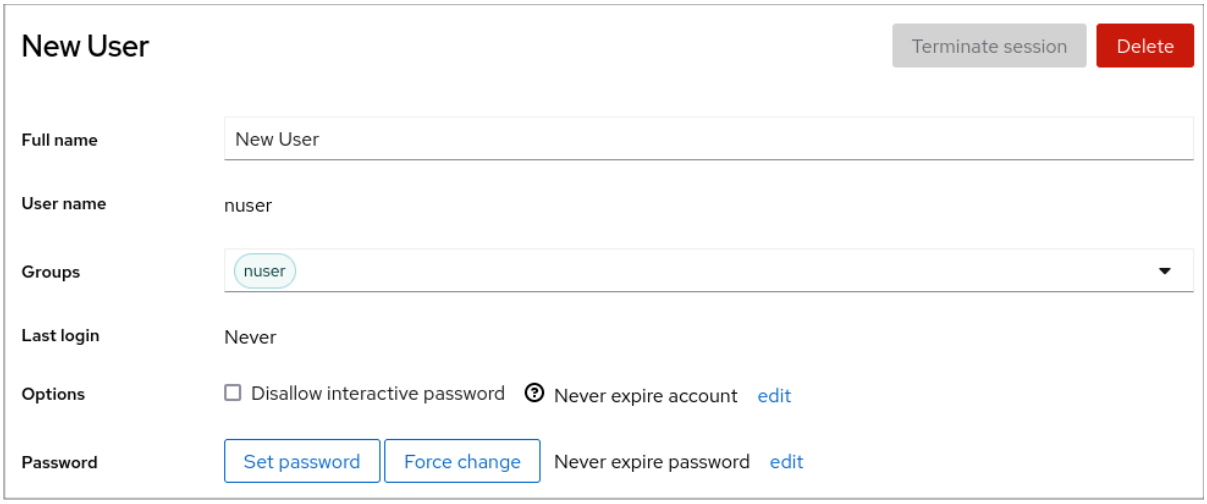
- 必须安装并可以访问 RHEL web 控制台。详情请参阅安装 Web 控制台。

流程

1. 登录到 RHEL web 控制台。
2. 点 Account。
3. 点 Create New Account。
4. 在 Full Name 字段中输入用户全名。

   RHEL web 控制台会自动在全名中推荐用户名并在 User Name 字段中填充该用户名。如果您不想使用原始命名规则（由名的第一个字母和完整的姓组成），对它进行更新。
5. 在 Password/Confirm 字段中输入密码并重新输入该密码以便验证您的密码是否正确。

   下面的颜色栏显示您输入密码的安全级别，这不允许您创建使用弱密码的用户。
6. 点 Create 保存设置并关闭对话框。
7. 选择新创建的帐户。
8. 在 Groups 下拉菜单中选择您要添加到新帐户的组。



现在，您可以在 Accounts 设置中看到新帐户，您可以使用其凭证连接到该系统。

11.3. 从命令行管理用户

您可以使用命令行界面（CLI）来管理用户和组。这可让您在 Red Hat Enterprise Linux 环境中添加、删除和修改用户和用户组。

11.3.1. 使用命令行添加新用户

您可以使用 useradd 工具添加新用户。

先决条件

- 根 访问权限

流程

- 要添加新用户，请使用：

```
# useradd options username
```

使用 useradd 命令的选项替换 *options*，并使用用户名称替换 *username*。

例 11.1. 添加新用户

添加用户 ID 为 5000 的用户 sarah，使用：

```
# useradd -u 5000 sarah
```

验证步骤

- 要验证新用户是否已添加，使用 id 工具程序。

```
# id sarah
```

输出返回：

```
uid=5000(sarah) gid=5000(sarah) groups=5000(sarah)
```

其他资源

- useradd 手册页

11.3.2. 使用命令行添加新组

您可以使用 groupadd 工具添加新组。

先决条件

- 根 访问权限

流程

- 要添加新组，请使用：

```
# groupadd options group-name
```

使用 groupadd 命令的命令行选项替换 *options*，并使用 *group-name* 替换 group-name。

例 11.2. 添加新组

要添加组 ID 为 5000 的组 sysadmins，请使用：

```
# groupadd -g 5000 sysadmins
```

验证步骤

- 要验证新组是否已添加，使用 tail 实用程序。

```
# tail /etc/group
```

输出返回：

```
sysadmins:x:5000:
```

其他资源

- groupadd 手册页

11.3.3. 从命令行将用户添加到补充组中

您可以将用户添加到补充组中，以管理权限或启用对特定文件或设备的访问权限。

先决条件

- root 访问权限

流程

- 要在用户的附加组中添加一个组，请使用：

```
# usermod --append -G group-name username
```

使用组群名称替换 *group- name*，并将 *group-name* 替换为组的名称。

例 11.3. 将用户添加到补充组中

要将用户 sysadmin 添加到 system-administrators 组中，请使用：

```
# usermod --append -G system-administrators sysadmin
```

验证步骤

- 要验证新的组被添加到用户 sysadmin 的附加组中，请使用：

```
# groups sysadmin
```

输出显示：

```
sysadmin : sysadmin system-administrators
```

11.3.4. 创建组目录

在 UPG 系统配置下，您可以将 *set-group 身份权限*（setgid 位）应用到目录。setgid 位使得管理共享目录的组项目变得更加简单。当您将 setgid 位应用到某个目录中时，在该目录中创建的文件会自动分配给拥有该目录的组群。在此组中具有写和执行权限的任何用户现在可以在目录中创建、修改和删除文件。

下面的部分论述了如何创建组目录。

先决条件

- 根 访问权限

流程

1. 创建目录：

```
# mkdir directory-name
```

使用目录名替换 *directory-name*。

2. 创建组：

```
# groupadd group-name
```

用组群的名称替换 *group-name*。

3. 向组中添加用户：

```
# usermod --append -G group-name username
```

使用组群名称替换 *group- name*，并将 *group-name* 替换为组的名称。

4. 将目录的用户和组群所有权与 *group-name* 组关联：

```
# chgrp group-name directory-name
```

用组群名称替换 *group-name*，并用 目录名替换 *directory-name*。

5. 设置写入权限，允许用户创建和修改文件和目录，并设置 setgid 位使其在 *directory-name* 目录中应用这个权限：

```
# chmod g+rwxs directory-name
```

使用目录名替换 *directory-name*。

现在，*group-name* 组的所有成员都可以在 *directory-name* 目录中创建并编辑文件。新创建的文件保留 *group-name* 组的组群所有权。

验证步骤

- 要验证设置权限的正确性，请使用：

```
# ls -ld directory-name
```

使用目录名替换 ***directory-name***。

输出返回：

```
drwx*rws*r-x. 2 root *group-name* 6 Nov 25 08:45 *directory-name*
```

11.3.5. 在命令行上删除用户

您可以使用命令行删除用户帐户。除了删除用户帐户外，您还可以选择删除用户数据和元数据，如其主目录和配置文件。

先决条件

- 您有 **root** 访问权限。
- 用户当前存在。
- 确保用户已退出登录：

```
# loginctl terminate-user user-name
```

流程

- 要仅删除用户帐户，而不删除用户数据：

```
# userdel user-name
```

- 要删除用户、数据和元数据：

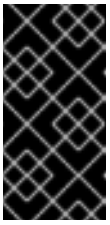
a. 删除用户、其主目录、其邮件假脱机及其 SELinux 用户映射：

```
# userdel --remove --selinux-user user-name
```

b. 删除其他用户元数据：

```
# rm -rf /var/lib/AccountsService/users/user-name
```

此目录存储系统在主目录可用之前所需的有关用户的信息。根据系统配置，用户在登录屏幕进行身份验证之前，主目录可能无法使用。



重要

如果您没有删除此目录，并在稍后重新创建同一用户，则重新创建的用户仍可以使用从已删除的用户继承来的某些设置。

其他资源

- userdel(8) 手册页。

11.4. 在 WEB 控制台中管理用户帐户

RHEL web 控制台提供了一个图形界面，可让您执行各种管理任务，而无需直接访问终端。例如，您可以添加、编辑或删除系统用户帐户。

在阅读这个部分后，您将了解：

- 现有帐户来自哪里。
- 如何添加新帐户。
- 如何设置密码过期。
- 如何和何时终止用户会话。

先决条件

- 设置 RHEL web 控制台。详情请参阅 开始使用 RHEL web 控制台。
- 使用分配了管理员权限的帐户登录到 RHEL web 控制台。详情请参阅 RHEL web 控制台的日

志记录。

11.4.1. Web 控制台中管理的系统用户帐户

您可在 RHEL web 控制台中显示用户帐户：

- 在访问系统时验证用户。
- 设置系统的访问权限。

RHEL web 控制台显示系统中的所有用户帐户。因此，在首次登录 web 控制台后，至少可以看到一个可用的用户帐户。

登录到 RHEL web 控制台后，您可以执行以下操作：

- 创建新用户帐户。
- 更改其参数。
- 锁定帐户。
- 终止用户会话。

11.4.2. 使用 Web 控制台添加新帐户

您可以将用户帐户添加到系统，并通过 RHEL web 控制台为帐户设置管理权限。

先决条件

- 必须安装并可以访问 RHEL web 控制台。详情请参阅安装 Web 控制台。

流程

1. 登录到 RHEL web 控制台。

2. 点 Account。

3. 点 Create New Account。

4. 在 Full Name 字段中输入用户全名。

   RHEL web 控制台会自动在全名中推荐用户名并在 User Name 字段中填充该用户名。如果您不想使用原始命名规则（由名的第一个字母和完整的姓组成），对它进行更新。

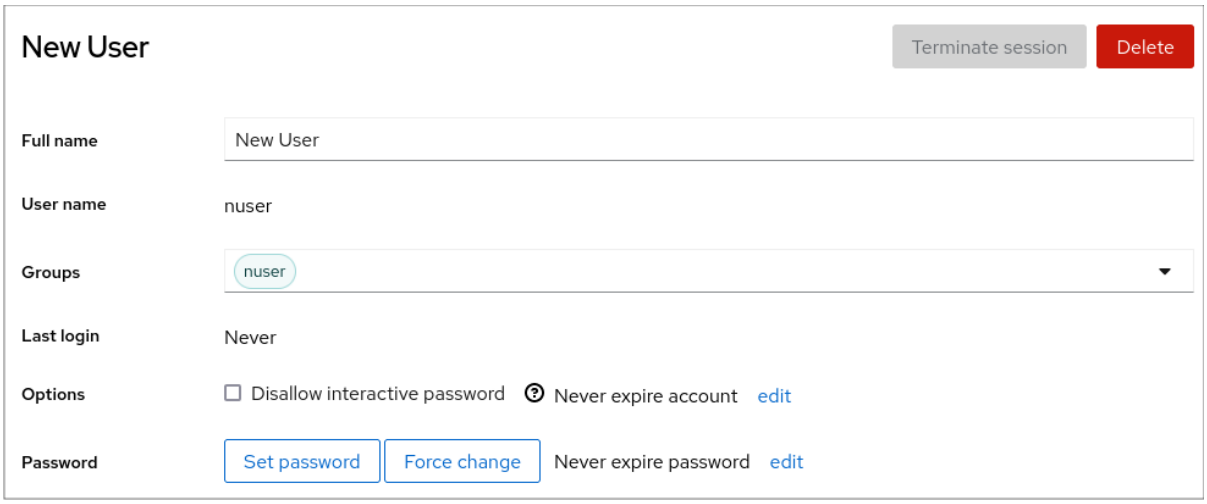
5. 在 Password/Confirm 字段中输入密码并重新输入该密码以便验证您的密码是否正确。

   下面的颜色栏显示您输入密码的安全级别，这不允许您创建使用弱密码的用户。

6. 点 Create 保存设置并关闭对话框。

7. 选择新创建的帐户。

8. 在 Groups 下拉菜单中选择您要添加到新帐户的组。



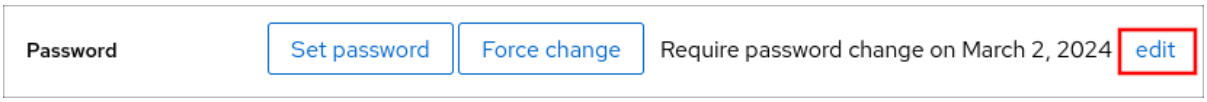
现在，您可以在 Accounts 设置中看到新帐户，您可以使用其凭证连接到该系统。

11.4.3. 在 web 控制台中强制密码过期

默认情况下，用户帐户将密码设定为永远不会过期。您可以设置系统密码在指定的天数后过期。当密码过期时，下次登录尝试会提示密码更改。

流程

1. 登录到 RHEL 8 web 控制台。
2. 点 Account。
3. 选择您要强制密码过期的用户帐户。
4. 点 Password 行上的 edit。



5. 在 Password expiration 对话框中，选择 Require password change every ... days，然后输入一个正整数，代表密码过期的天数。
6. 点 Change。

Web 控制台会立即在 Password 行上显示未来密码更改请求的日期。

11.4.4. 在 web 控制台中终止用户会话

用户在登录系统时创建用户会话。终止用户会话意味着从系统中注销用户。如果您需要执行对配置更改敏感的管理任务，比如升级系统，这非常有用。

在 RHEL 8web 控制台中的每个用户帐户中，您可以终止该帐户的所有会话，但您当前使用的 web 控制台会话除外。这可防止您失去对系统的访问。

流程

1. 登录到 RHEL 8 web 控制台。
2. 点 Account。
3. 点击要终止会话的用户帐户。
4. 点 Terminate Session。

   如果 Terminate Session 按钮不可用，这个用户就不能登录到系统。

   RHEL web 控制台会终止会话。

11.5. 使用命令行编辑用户组

用户属于某个组集合，允许用户的逻辑组集合对文件和文件夹具有类似的访问权限。您可以从命令行编辑主和补充用户组，以更改用户的权限。

11.5.1. 主和补充用户组

组是出于共同目的将多个用户帐户连接在一起的实体，例如对特定文件授予访问权限。

在 Linux 上，用户组可以充当主或补充组。主和补充组具有以下属性：

主组

- 每个用户始终只有一个主组。

- 您可以更改用户的主组。

补充组

- 您可以将现有用户添加到现有的补充组中，以使用相同的安全和访问权限管理组中的用户。

- 用户可以是零个或多个补充组的成员。

11.5.2. 列出用户的主和补充组

您可以列出用户的组，以查看他们所属的主和补充组。

流程

- 显示用户的主组以及任何补充组的名称：

```
$ groups user-name
```

使用用户名称替换 ***user-name***。如果不提供用户名，则命令将显示当前用户的组成员身份。第一个组是主组，后跟可选的补充组。

例 11.4. 列出用户 sarah 的组：

```
$ groups sarah
```

输出显示：

```
sarah : sarah wheel developer
```

用户 arah 有一个主组 sarah，它是补充组 wheel 和 developer 的成员。

例 11.5. 列出用户 marc 的组：

```
$ groups marc
```

输出显示：

```
marc : marc
```

用户 marc 仅有一个主组 marc，没有补充组。

11.5.3. 更改用户的主组

您可以将现有用户的主组更改为一个新组。

先决条件：

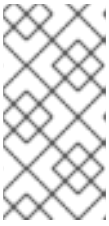
1. root 访问权限
2. 新组必须存在

流程

- 更改用户的主组：

```
# usermod -g group-name user-name
```

使用新主组的名称替换 *group-name*，并使用用户名替换 *user-name*。



注意

更改用户的主组时，命令还会将用户主目录中所有文件的组所有权自动更改为新的主组。您必须手动修复用户主目录外文件的组所有权。

例 11.6. 更改用户的主组的示例：

如果用户 arah 属于主组 sarah1，且您想将用户的主组更改为 sarah2，请使用：

```
# usermod -g sarah2 sarah
```

验证步骤

- 验证您是否更改了用户的主组：

```
$ groups sarah
```

输出显示：

```
sarah : sarah2
```

11.5.4. 从命令行将用户添加到补充组中

您可以将用户添加到补充组中，以管理权限或启用对特定文件或设备的访问权限。

先决条件

- root 访问权限

流程

- 要在用户的附加组中添加一个组，请使用：

```
# usermod --append -G group-name username
```

使用组群名称替换 *group- name*，并将 *group-name* 替换为组的名称。

例 11.7. 将用户添加到补充组中

要将用户 sysadmin 添加到 system-administrators 组中，请使用：

```
# usermod --append -G system-administrators sysadmin
```

验证步骤

- 要验证新的组被添加到用户 sysadmin 的附加组中，请使用：

```
# groups sysadmin
```

输出显示：

```
sysadmin : sysadmin system-administrators
```

11.5.5. 从补充组中删除用户

您可以从补充组中删除现有的用户，以限制他们对文件和设备的权限或访问。

先决条件

- root 访问权限

流程

- 从补充组中删除用户：

```
# gpasswd -d user-name group-name
```

使用用户名替换 *user-name*，并使用补充组的名称替换 *group-name*。

例 11.8. 从补充组中删除用户

如果用户 sarah 有一个主组 sarah2，并且属于次要组 wheel 和 developers，并且您想从组 developers 中删除该用户，请使用：

```
# gpasswd -d sarah developers
```

验证步骤

- 验证您是否从次要组 developers 中删除了用户 sarah：

```
$ groups sarah
```

-

输出显示：

```
sarah : sarah2 wheel
```

11.5.6. 更改用户的所有补充组

您可以覆盖您希望用户保留其成员的补充组的列表。

先决条件

- root 访问权限
- 补充组必须存在

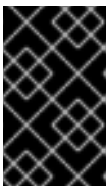
流程

- 覆盖用户的补充组的列表：

```
# usermod -G group-names username
```

使用一个或多个补充组的名称替换 *group-names*。要将用户一次添加到多个补充组中，请使用逗号分隔组名称，并且没有插入空格。例如：wheel,developer。

使用用户名称替换 *user-name*。



重要

如果用户是当前您未指定的组的成员，则该命令会从组中删除该用户。

例 11.9. 更改用户的补充组的列表

如果用户 sarah 有一个主组 sarah2，并且属于补充组 wheel，您希望用户属于多个补充组 developer、sysadmin 和 security，请使用：

```
# usermod -G wheel,developer,sysadmin,security sarah
```

验证步骤

- 验证您是否正确设置了补充组列表：

```
# groups sarah
```

输出显示：

```
sarah : sarah2 wheel developer sysadmin security
```

11.6. 更改和重置根密码

如果需要更改现有的根密码，可以以 root 用户或一个非 root 用户重置它。

11.6.1. 作为 root 用户更改 root 密码

您可以使用 passwd 命令以 root 用户身份更改 root 密码。

先决条件

- 根 访问权限

流程

- 要更改 root 密码，使用：

```
# passwd
```

在修改前，会提示您输入您当前的密码。

11.6.2. 以非 root 用户的身份更改或重置根密码

您可以使用 passwd 命令以非 root 用户身份更改或重置忘掉的root 密码。

先决条件

- 您可以以非 root 用户身份登录。
- 您是管理 wheel 组的成员。

流程

- 以wheel 组中的非 root 用户身份修改或重置 root 密码，请使用：

```
$ sudo passwd root
```

此时会提示您输入当前的非 root 密码，然后才能更改 root 密码。

11.6.3. 在引导时重置 root 密码

如果您无法以非 root 用户身份登录或者不属于管理 wheel 组，则可以通过切换到一个特殊的 chroot jail 环境在引导时重置 root 密码。

流程

1. 重启系统，在 GRUB 2 引导屏幕上按 e 键中断引导过程。

   此时会出现内核引导参数。

```
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. 进入以 linux 开头的行的末尾。

```
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
```

按 Ctrl+e 键跳到这一行的末尾。

3. 在以 linux 开头的行的最后添加 rd.break。

```
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. 按 Ctrl+x 使用更改的参数启动系统。

此时会出现 switch_root 提示符。

5. 将文件系统重新挂载为可写：

```
mount -o remount,rw /sysroot
```

文件系统以只读模式挂载到 /sysroot 目录中。将文件系统重新挂载为可写才可以更改密码。

6. 进入 chroot 环境：

```
chroot /sysroot
```

此时会出现 sh-4.4# 提示符。

7. 重置 root 密码：

```
passwd
```

按照命令行中的步骤完成 root 密码的更改。

8. 在下次系统引导时启用 SELinux 重新标记进程：

```
touch /.autorelabel
```

9. 退出 chroot 环境：

```
exit
```

10. 退出 switch_root 提示符：

```
exit
```

11. 等待 SELinux 重新标记过程完成。请注意，重新标记一个大磁盘可能需要很长时间。系统会在这个过程完成后自动重启。

验证步骤

1. 要验证 root 密码是否已成功更改，请以普通用户身份登录并打开 Terminal。

2. 以 root 用户身份运行交互式 shell:

```
$ su
```

3. 输入新的 root 密码。

4. 显示与当前有效用户 ID 关联的用户名：

```
# whoami
```

输出返回：

```
root
```

第 12 章 管理 SUDO 访问

系统管理员可以授予 sudo 访问权限，以允许非 root 用户执行通常为 root 用户保留的管理命令。因此，非 root 用户可以执行此类命令，而无需登录到 root 用户帐户。

12.1. SUDOERS 中的用户授权

/etc/sudoers 文件指定哪些用户可以使用 sudo 命令来执行其他命令。规则可应用到单个用户和用户组。您还可以使用别名轻松地为主机组、命令，甚至用户定义规则。默认别名定义在 /etc/sudoers 文件的第一部分中。

当用户为没有授权的用户输入带有 sudo 的命令时，系统会将包含字符串 ***<username>*** **: user NOT in sudoers** 的消息记录到日志中。

默认的 /etc/sudoers 文件提供授权信息和示例。您可以通过取消相应行的注释来激活特定的示例规则。带有用户授权的部分标有以下介绍：

```
## Next comes the main part: which users can run what software on
## which machines  (the sudoers file can be shared between multiple
## systems).
```

您可以使用以下格式创建新的 sudoers 授权并修改现有授权：

```
<username> <hostname.example.com>=(<run_as_user>:<run_as_group>) <path/to/command>
```

其中：

- ***<username>*** 是输入命令的用户，如 user1。如果该值以 % 开头，它定义一个组，例如 %group1。
- ***<hostname.example.com>*** 是应用规则的主机的名称。
- (***<run_as_user>***:***<run_as_group>***) 部分 定义以其身份执行命令的用户或组。如果省略这个部分，***<username>*** 可以以 root 身份执行命令。
-

<path/to/command> 是命令的完整的绝对路径。您还可以通过在命令路径后面添加这些选项，将用户限制为仅使用特定的选项和参数执行命令。如果没有指定任何选项，用户可以使用带有所有选项的命令。

您可以通过将任何这些变量替换为 ALL，来将规则应用到所有用户、主机或命令。

警告

使用过于宽松的规则（如 ALL=(ALL) ALL ），所有用户都可以在所有主机上以所有用户的身份运行所有命令。这带来了严重的安全风险。

您可以使用！操作符来指定参数的反。例如，!root 指定除 root 以外的所有用户。请注意，允许特定的用户、组和命令比禁止特定的用户、组和命令更安全。这是因为允许规则还阻止新的未授权的用户或组。

警告

避免对命令使用负规则，因为用户可以通过使用 alias 命令重命名命令来克服此类规则。

系统会从头到尾读取 /etc/sudoers 文件。因此，如果文件中包含用户的多个条目，则按顺序应用条目。如果值冲突，系统将使用最后匹配的项，即使它不是最具体的匹配。

要在系统更新期间保留规则，并更轻松地修复错误，请在 /etc/sudoers.d/ 目录中创建新文件来输入新规则，而不是直接在 /etc/sudoers 文件中输入规则。当系统在 /etc/sudoers 文件中达到以下行时，会读取 /etc/sudoers.d 目录中的文件：

```
#includedir /etc/sudoers.d
```

请注意，此行开头的数字符号(#)是语法的一部分，并不意味着该行是一个注释行。该目录中文件的名称不能包含句点，也不能以波形符(~)结尾。

其他资源

- **sudo (8) 和 sudoers (5) 手册页**

12.2. 为用户授予 SUDO 访问权限

系统管理员可以通过授予他们 sudo 访问权限来允许非 root 用户执行管理命令。sudo 命令为用户提供了管理访问权限，而无需使用 root 用户的密码。

当用户需要执行管理命令时，您可以在使用 sudo 命令前执行该命令。如果用户有命令的授权，则可以执行命令，就像他们是 root 一样。

请注意以下限制：

- 只有 /etc/sudoers 配置文件中列出的用户才能使用 sudo 命令。
- 命令在用户的 shell 下执行，而不是在 root shell 下执行。

先决条件

- 有对系统的 root 访问权限。

流程

1. 以 root 身份，打开 /etc/sudoers 文件。

```
# visudo
```

/etc/sudoers 文件定义 sudo 命令应用的策略。

2. 在 /etc/sudoers 文件中，找到为 wheel 管理组中用户授予 sudo 访问权限的行。

```
## Allows people in group wheel to run all commands
%wheel        ALL=(ALL)       ALL
```

3. 确保以 %wheel 开头的行没有用数字符号(#)注释掉。

4. 保存所有更改并退出编辑器。

5. 将您要授予 sudo 访问权限的用户添加到 wheel 管理组中。

```
# usermod --append -G wheel <username>
```

将 ***<username>*** 替换为用户的名称。

验证步骤

- 验证用户是否在 wheel 管理组中：

```
# id <username>
uid=5000(<username>) gid=5000(<username>) groups=5000(<username>),10(wheel)
```

其他资源

- sudo (8)、visudo (8) 和 sudoers (5) 手册页

12.3. 使非特权用户运行某些命令

作为管理员，您可以通过在 /etc/sudoers.d/ 目录中配置策略来允许非特权用户在特定工作站上输入某些命令。

先决条件

- 有对系统的 root 访问权限。

流程

1. 以 root 用户身份，在 /etc/ 下创建一个新的 sudoers.d 目录：

```
# mkdir -p /etc/sudoers.d/
```

2. 在 /etc/sudoers.d 目录中创建一个新文件：

```
# visudo -f /etc/sudoers.d/<filename>
```

文件会自动打开。

3. 将以下行添加到 /etc/sudoers.d/*<filename>* 文件中：

```
<username> <hostname.example.com> = (<run_as_user>:<run_as_group>)
<path/to/command>
```

- 将 ***<username>*** 替换为用户的名称。
- 将 ***<hostname.example.com>*** 替换为主机的 URL。
- 将 (***<run_as_user>***:***<run_as_group>***) 替换为可以以其身份执行命令的用户或组。如果省略这个部分，***<username>*** 可以以 root 身份执行命令。
- 将 ***<path/to/command>*** 替换为命令的完整的绝对路径。您还可以通过在命令路径后面添加这些选项，将用户限制为仅使用特定的选项和参数执行命令。如果没有指定任何选项，用户可以使用带有所有选项的命令。
- 要在一行上允许同一主机上的两个和多个命令，您可以用逗号后跟一个空格把它们分开来列出它们。

  例如，要允许 user1 在 host1.example.com 上执行 dnf 和 reboot 命令，请输入 user1 host1.example.com = /bin/dnf, /sbin/reboot。

4. 可选：要在用户每次尝试使用 sudo 特权时收到电子邮件通知，请在文件中添加以下行：

```
Defaults    mail_always
Defaults    mailto="<email@example.com>"
```

5. 保存更改，再退出编辑器。

验证

1. 要验证用户是否可以使用 sudo 特权运行命令，请切换帐户：

```
# su <username> -
```

2. 以用户身份，输入带有 sudo 的命令：

```
$ sudo <command>
[sudo] password for <username>:
```

输入用户的 sudo 密码。

3. 如果正确配置了特权，系统会显示命令和选项的列表。例如，使用 dnf 命令，它显示以下输出：

```
...
usage: dnf [options] COMMAND
...
```

如果系统返回错误信息 ***<username>*** **is not in the sudoers file。**这个事件会被报告，**the file for *<username>* in /etc/sudoers.d/ does not exist。**

如果系统返回错误消息 ***<username>*** **is not allowed to run sudo on *<host.example.com>*，**则配置没有正确完成。确保您已以 **root** 身份登录，并且配置被正确执行。

如果系统返回错误消息 **Sorry, user *<username>* is not allowed to execute '*<path/to/command>*' as root on *<host.example.com>*.，**则命令没有在用户的规则中正确定义。

其他资源

- **sudo (8)、visudo (8) 和 sudoers (5) 手册页**

第 13 章 管理文件系统权限

文件系统权限控制用户和组帐户读、修改和执行文件的内容以及进入目录的能力。仔细设置权限以保护您的数据免于未授权访问。

13.1. 管理文件权限

每个文件或目录都有三个级别的所有权：

- 用户所有者（u）。
- 组所有者（g）。
- 其他（o）。

可为每个级别的所有权分配以下权限：

- 读（r）。
- 写（w）。
- 执行（x）。

请注意，文件的执行权限允许执行该文件。目录的执行权限允许访问目录中的内容，但不执行它。

创建新文件或目录时，会自动为其分配默认权限集。文件或目录的默认权限基于两个因素：

- 基本权限。
- *user file-creation mode mask*（umask）。

13.1.1. 基本文件权限

每当创建新文件或目录时，会自动为其分配基本权限。文件或目录的基本权限可以使用*符号*或者*数值*表示。

权限	符号	数值
无权限	---	0
执行	--x	1
写	-w-	2
写和执行	-wx	3
读	r--	4
读和执行	r-x	5
读写	rw-	6
读、写、执行	rwx	7

目录的基本权限是 777（drwxrwxrwxrwx），它为任何人都授予读、写和执行的权限。这意味着目录所有者、组和其它可以列出目录的内容，并可以在该目录下（以及其子目录）中创建、删除和编辑项。

请注意，一个目录中的单个文件可以有它们自己的权限，例如可以阻止用户您编辑它们，即使用户对该目录有非受限的访问权限。

文件的基本权限为 666（-rw-rw-rw-），它为所有人都授予读取和写入的权限。这意味着文件所有者、组和其它用户都可以读和编辑该文件。

例 13.1. 文件的权限

如果文件有以下权限：

```
$ ls -l
-rwxrw----. 1 sysadmins sysadmins 2 Mar 2 08:43 file
```

- **- 表示它是文件。**
- **rwx 表示文件所有者有读、写和执行文件的权限。**
- **rw- 表示组有读写权限，但不能执行文件。**
- **--- 表示其他用户没有读、写或执行文件的权限。**
- **. 表示为该文件设定了 SELinux 安全上下文。**

例 13.2. 目录的权限

如果一个目录有以下权限：

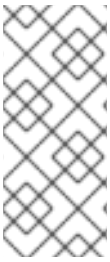
```
$ ls -dl directory
drwxr-----. 1 sysadmins sysadmins 2 Mar 2 08:43 directory
```

- **d 表示它是一个目录。**
- **rwx 表示目录所有者有读、写和访问目录内容的权限。**

  作为目录所有者，您可以列出目录中的项目（文件、子目录），访问这些项目的内容并进行修改。
- **r-x 表示组有读目录内容的权限，但没有写权限 - 创建新条目或删除文件。x 权限意味着您也可以使用 cd 命令访问该目录。**
- **--- 表示其他用户没有权限读取、写入或者访问该目录的内容。**

  作为不是用户拥有者或该目录的组所有者的用户，您无法列出目录中的项目、关于这些项目的访问信息或修改它们。

- . 表示为该目录设定了 SELinux 安全性上下文。



注意

自动分配给某个文件或者目录的基本权限不是文件或目录最终的默认权限。当您创建文件或目录时，基本权限会被 *umask* 更改。基本权限和 *umask* 的组合会为文件和目录创建默认权限。

13.1.2. 用户文件创建模式掩码

用户文件创建模式掩码(*umask*)是一个变量，用于控制如何为新创建的文件和目录设置文件权限。*umask* 会自动从基本权限值中删除权限，以提高 Linux 系统的整体安全性。*umask* 可以用 *符号* 或 *八进制* 值表示。

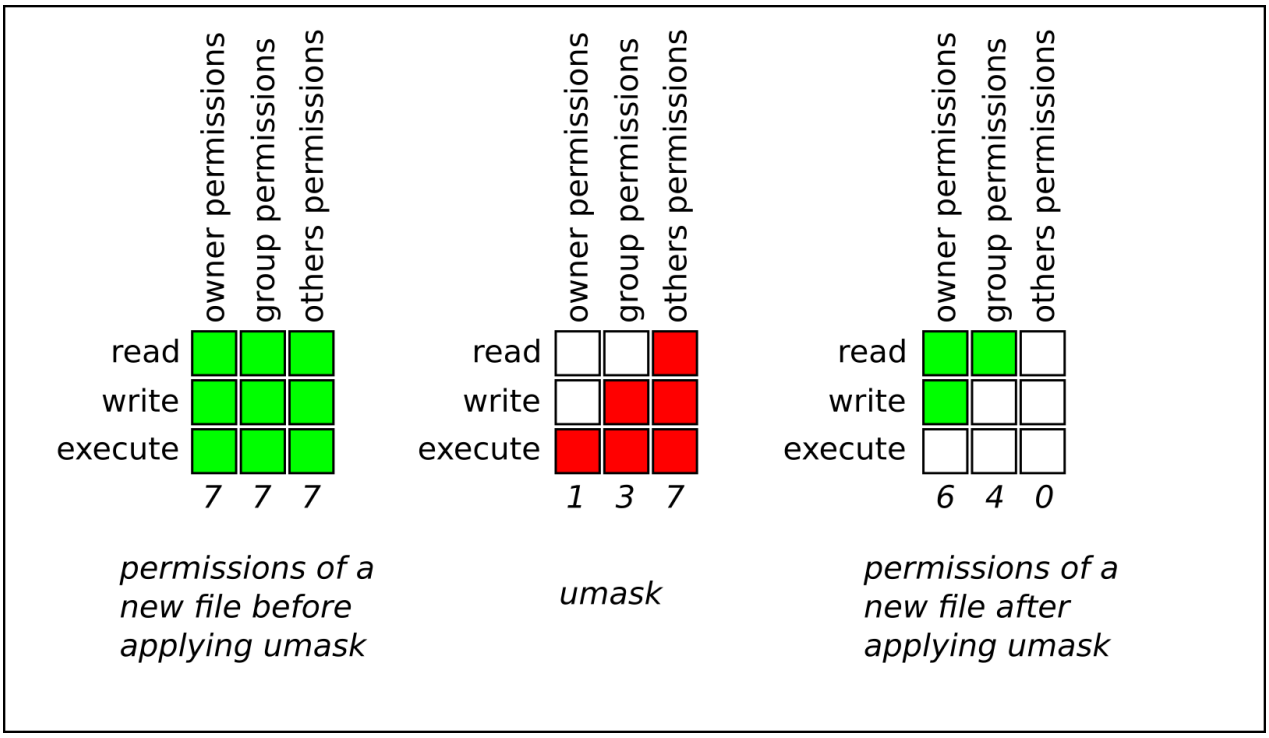
权限	符号	数值
读、写和执行	rwx	0
读写	rw-	1
读和执行	r-x	2
读	r--	3
写和执行	-wx	4
写	-w-	5
执行	--x	6
无权限	---	7

标准用户的默认 *umask* 是 0002。root 用户的默认 *umask* 为 0022。

umask 的第一个数字代表特殊权限（sticky 位）。*umask* 的最后三位数字分别代表从用户拥有者（u）、组群所有者（g）和其它（o）中删除的权限。

例 13.3. 在创建文件时应用 umask

下面的例子演示了，对一个基本权限为 777 的文件应用值为 0137 的 *umask*，使在创建该文件时其默认权限变为 640。



13.1.3. 默认的文件权限

为所有新创建的文件和目录自动设置默认权限。默认权限的值通过将 *umask* 应用到基本权限来确定。

例 13.4. 标准用户创建的目录的默认权限

当 标准用户 创建了一个新 目录 时，*umask* 被设为 002 (rwxrwxr-x)，目录的基本权限被设为 777 (rwxrwxrwx)。这会使默认权限为 775(drwxrwxr-x)。

	符号	数值
基本权限	rwxrwxrwx	777
Umask	rwxrwxr-x	002
默认权限	rwxrwxr-x	775

这意味着目录所有者、组和其它可以列出目录的内容，并可以在该目录下（以及其子目录）中创建、删除和编辑项。其他用户只能列出该目录的内容并将其下移到其中。

例 13.5. 由标准用户创建的文件的默认权限

当 标准用户 创建一个新 文件 时，*umask* 被设为 002 (rwxrwxr-x)，文件的基本权限被设为 666 (rw-rw-rw-)。这会使默认权限为 664(-rw-rw-r--)。

	符号	数值
基本权限	rw-rw-rw-	666
Umask	rwxrwxr-x	002
默认权限	rw-rw-r--	664

这意味着，文件拥有者和组群可以读取和编辑该文件，而其他用户只能读取该文件。

例 13.6. root 用户创建的目录的默认权限

当 root 用户 创建了一个新 目录 时，*umask* 被设为 022 (rwxr-xr-x)，目录的基本权限被设为 777 (rwxrwxrwx)。这会使默认权限为 755 (rwxr-xr-x)。

	符号	数值
基本权限	rwxrwxrwx	777
Umask	rwxr-xr-x	022
默认权限	rwxr-xr-x	755

这意味着目录所有者可以列出目录的内容，并可以在该目录下（以及其子目录）中创建、删除和编辑项。这个组群和其它只能列出该目录的内容并将其下移。

例 13.7. 由 root 用户创建的文件的默认权限

当 root 用户 创建了一个新 文件 时，*umask* 被设为 022 (rwxr-xr-x)，文件的基本权限被设为 666 (rw-rw-rw-)。这会使默认权限为 644(-rw-r-r--)。

	符号	数值

基本权限	rw-rw-rw-	666
Umask	rwxr-xr-x	022
默认权限	rw-r--r--	644

这意味着，文件所有者可以读取和编辑文件，而组和其它用户只能读取该文件。

注意

出于安全考虑，常规文件默认没有执行权限，即使 ***umask*** 设为 **000（rwxrwxrwx）**。但是，创建的目录可以具有执行权限。

13.1.4. 使用符号值更改文件权限

您可以使用带有符号值（字母和符号的组合）的 **chmod** 工具来更改文件或目录的文件权限。

您可以分配以下 *权限* ：

- 读(r)
- 写(w)
- 执行(x)

权限可分配给以下 *所有权级别* ：

- 用户所有者 (u)
- 组所有者(g)

- 其他 (o)
- 所有 (a)

要添加或删除权限，您可以使用以下 *符号* ：

- + 在现有权限之上添加权限
- - 从现有权限中删除权限
- = 删除现有权限，并明确定义新权限

流程

- 要更改文件或目录的权限，请使用：

```
$ chmod <level><operation><permission> file-name
```

将 *<level>* 替换为您要为其设置权限的 所有权级别。将 *<operation>* 替换为其中一个 符号。将 *<permission>* 替换为您要分配的 权限。用文件或目录的名称替换 *file-name*。例如，要为每个人授予读、写和执行(rwx) my-script.sh 的权限，请使用 chmod a=rwx my-script.sh 命令。

如需了解更多详细信息，请参阅 基本文件权限。

验证步骤

- 要查看特定文件的权限，请使用：

```
$ ls -l file-name
```

用文件名替换 *file-name*。

- 要查看特定目录的权限，请使用：

```
$ ls -dl directory-name
```

使用目录名替换 *directory-name*。

- 要查看特定目录中所有文件的权限，请使用：

```
$ ls -l directory-name
```

使用目录名替换 *directory-name*。

例 13.8. 更改文件和目录的权限

- 要将 my-file.txt 的文件权限从 -rw-rw-r-- 改为 -rw------ ，请使用：

1. 显示 my-file.txt 的当前权限：

```
$ ls -l my-file.txt
-rw-rw-r--. 1 username username 0 Feb 24 17:56 my-file.txt
```

2. 从组所有者(g)和其他用户(o)删除读、写和执行(rwx)文件的权限：

```
$ chmod go= my-file.txt
```

请注意，任何在等号（=）之后没有被指定的权限都会被自动禁止。

3. 验证 my-file.txt 的权限是否设置正确：

```
$ ls -l my-file.txt
-rw-------. 1 username username 0 Feb 24 17:56 my-file.txt
```

- 要将 my-directory 的文件权限从 drwxrwx--- 改为 drwxrwxr-x，请使用：

1. 显示 my-directory 的当前权限：

```
$ ls -dl my-directory
drwxrwx---. 2 username username 4096 Feb 24 18:12 my-directory
```

2. 为所有用户(a) 添加读和执行(r-x)权限：

```
$ chmod o+rx my-directory
```

3. 验证 my-directory 及其内容的权限是否设置正确：

```
$ ls -dl my-directory
drwxrwxr-x. 2 username username 4096 Feb 24 18:12 my-directory
```

13.1.5. 使用数值更改文件权限

您可以使用带有八进制（数字）的 chmod 工具来更改文件或目录的文件权限。

流程

- 要为现有文件或者目录更改文件权限，请使用：

```
$ chmod octal_value file-name
```

用文件或目录的名称替换 *file-name*。使用数值替换 *octal_value*。如需了解更多详细信息，请参阅 基本文件权限。

13.2. 管理访问控制列表

每个文件和目录同时只能有一个用户所有者和一个组所有者。如果您要授予用户权限访问属于不同用户或组的特定文件或目录，同时保持其他文件和目录私有，则您可以使用 Linux 访问控制列表(ACL)。

13.2.1. 显示当前的访问控制列表

您可以使用 getfacl 工具来显示当前 ACL。

流程

- 要显示特定文件或目录的当前 ACL，请使用：

```
$ getfacl file-name
```

用文件或目录的名称替换 *file-name*。

13.2.2. 设置访问控制列表

您可以使用 setfacl 工具为文件或目录设置 ACL。

先决条件

- root 访问权限。

流程

- 要为文件或目录设置 ACL，请使用：

```
# setfacl -m u:username:symbolic_value file-name
```

使用用户名替换 *username*，使用符号值替换 *symbolic_value*，使用文件或目录的名称替换 *file-name*。详情请查看 setfacl man page。

例 13.9. 修改组项目的权限

以下示例描述了如何修改属于 root 组的 root 用户拥有的 group-project 文件的权限，以便使该文件：

- 不能被任何人执行。
- 用户 andrew 有 rw- 权限。
- 用户 susan 有 --- 权限。

- 其他用户有 r-- 权限。

流程

```
# setfacl -m u:andrew:rw- group-project
# setfacl -m u:susan:--- group-project
```

验证步骤

- 要验证用户 andrew 有 rw- 权限，用户 susan 有 --- 权限，其他用户有 r-- 权限，使用：

```
$ getfacl group-project
```

输出会返回：

```
# file: group-project
# owner: root
# group: root
user:andrew:rw-
user:susan:---
group::r--
mask::rw-
other::r--
```

13.3. 管理 UMASK

您可以使用 umask 工具显示、设置或更改 *umask* 的当前或默认值。

13.3.1. 显示 umask 的当前值

您可以使用 umask 工具以符号或数值模式显示 *umask* 的当前值。

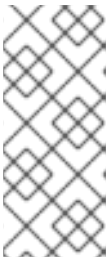
流程

- 要在符号模式下显示 *umask* 的当前值，请使用：

```
$ umask -S
```

- 要在八进制模式下显示 *umask* 的当前值，请使用：

```
$ umask
```



注意

以八进制模式显示 *umask* 时，您可以注意到它显示了四位数字（0002 或 0022）。*umask* 的第一个数字代表一个特殊的位（spicky 位、SGID 位或 SUID 位）。如果第一个数字设定为 0，则代表没有设置特殊位。

13.3.2. 显示默认 bash umask

您可以使用不同的 shell，如 bash、ksh、zsh 和 tcsh。这些 shell 可以是登录或非登录 shell。您可以通过打开一个原生或 GUI 终端来调用登录 shell。

要判断您是在登录 shell 还是非登录 shell 中执行某个命令，请使用 echo $0 命令。

例 13.10. 确定您在登录或非登录 bash shell 下工作

- 如果 echo $0 命令的输出返回 bash，则您在非登录 shell 下执行命令。

```
$ echo $0
bash
```

非登录 shell 的默认 *umask* 在 /etc/bashrc 配置文件中设置。

- 如果 echo $0 命令的输出返回 -bash，则您在登录 shell 下执行命令。

```
# echo $0
-bash
```

登录 shell 的默认 *umask* 在 /etc/profile 配置文件中设置。

流程

- 要显示非登录 shell 的默认 bash *umask*，请使用：

```
$ grep umask /etc/bashrc
```

输出会返回：

```
# By default, we want umask to get set. This sets it for non-login shell.
    umask 002
    umask 022
```

- 要显示登录 shell 的默认 bash *umask*，请使用：

```
$ grep umask /etc/profile
```

输出会返回：

```
# By default, we want umask to get set. This sets it for login shell
    umask 002
    umask 022
```

13.3.3. 使用符号值设置 umask

您可以使用 umask 工具及符号值（字母和符号组合）来为当前的 shell 会话设置 *umask*

您可以分配以下 *权限* ：

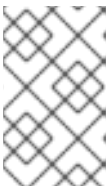
- 读(r)
- 写(w)
- 执行(x)

权限可分配给以下 *所有权级别* ：

- 用户所有者 (u)
- 组所有者(g)
- 其他 (o)
- 所有 (a)

要添加或删除权限，您可以使用以下 *符号* ：

- + 在现有权限之上添加权限
- - 从现有权限中删除权限
- = 删除现有权限，并明确定义新权限



注意

任何在等号(=)后未指定的权限都将被自动禁止。

流程

- 要为当前的 shell 会话设置 *umask*，请使用：

```
$ umask -S <level><operation><permission>
```

将 *<level>* 替换为您要为其设置 umask 的 所有权级别。将 *<operation>* 替换为其中一个 符号。将 *<permission>* 替换为您要分配的 权限。例如，要将 *umask* 设为 u=rwx,g=rwx,o=rwx，使用 umask -S a=rwx。

如需了解更多详细信息，请参阅 用户文件创建模式。

注意

umask 仅对当前 shell 会话有效。

13.3.4. 使用数值设置 umask

您可以使用 umask 工具和八进制值（数字）来为当前 shell 会话设置 *umask*。

流程

- 要为当前的 shell 会话设置 *umask*，请使用：

```
$ umask octal_value
```

使用数值替换 *octal_value*。如需了解更多详细信息，请参阅 用户文件创建模式掩码。

注意

umask 仅对当前 shell 会话有效。

13.3.5. 更改非登录 shell 的默认 umask

您可以通过修改 /etc/bashrc 文件来更改标准用户的默认 bash *umask*。

先决条件

- root 访问权限

流程

1. 以 root 用户身份，在编辑器中打开 /etc/bashrc 文件。

2. 修改以下部分以设置新的默认 bash *umask* ：

```
if [ $UID -gt 199 ] && [ “id -gn” = “id -un” ]; then
   umask 002
else
   umask 022
fi
```

将 *umask* 的默认值（002）替换为另一个数值。如需了解更多详细信息，请参阅 用户文件创建模式掩码。

3. 保存更改并退出编辑器。

13.3.6. 更改登录 shell 的默认 umask

您可以通过修改 /etc/profile 文件来更改 root 用户的默认 bash *umask*。

先决条件

- root 访问权限

流程

1. 以 root 用户身份，在编辑器中打开 /etc/profile 文件。

2. 修改以下部分以设置新的默认 bash *umask* ：

```
if [ $UID -gt 199 ] && [ “/usr/bin/id -gn” = “/usr/bin/id -un” ]; then
   umask 002
else
   umask 022
fi
```

将 *umask* 的数值（022）替换为另一个数值。如需了解更多详细信息，请参阅 用户文件创建模式掩码。

3. 保存更改并退出编辑器。

13.3.7. 更改特定用户的默认 umask

您可以通过修改用户的 .bashrc 来更改特定用户的默认 *umask*。

流程

- 将指定 *umask* 的八进制值的行追加到特定用户的 .bashrc 文件中。

```
$ echo 'umask octal_value' >> /home/username/.bashrc
```

使用数值替换 *octal_value*，并使用用户名替换 *username*。如需了解更多详细信息，请参阅 用户文件创建模式掩码。

13.3.8. 为新创建的主目录设置默认权限

您可以通过修改 /etc/login.defs 文件来更改新创建的用户的主目录的权限模式。

流程

1. 以 root 用户身份，在编辑器中打开 /etc/login.defs 文件。
2. 修改以下部分来设置新的默认 *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
```

将默认的八进制值(0700)替换为另一个八进制值。所选模式将用于为主目录创建权限。

3. 如果设置了 *HOME_MODE*，请保存更改并退出编辑器。
4. 如果没有设置 *HOME_MODE*，请修改 *UMASK* 来为新创建的主目录设置模式：

```
# 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
```

将默认的八进制值(022)替换为另一个八进制值。如需了解更多详细信息，请参阅 用户文件创建模式掩码。

5. **保存更改并退出编辑器。**

第 14 章 管理 SYSTEMD

作为系统管理员，您可以使用 systemd 管理系统的关键方面。充当 Linux 操作系统的系统和服务管理器的systemd 软件套件提供用于控制、报告和系统初始化的工具和服务。systemd 的主要功能包括：

- 在启动过程中并行启动系统服务
- 按需激活守护进程
- 基于依赖项的服务控制逻辑

systemd 管理的基本对象是 *systemd 单元*，一个系统资源和服务的表示。systemd 单元由一个名称、类型和配置文件组成，用来定义和管理特定的任务。您可以使用单元文件来配置系统行为。请参阅以下各种 systemd 单元类型示例：

服务

: 控制和管理单个系统服务。

目标

: 表示定义系统状态的一组单元。

设备

: 管理硬件设备及其可用性。

Mount

: 处理文件系统挂载。

定时器

: 调度任务以特定间隔运行。

注意

要显示所有可用的单元类型：

```
# systemctl -t help
```

14.1. SYSTEMD 单元文件位置

您可以在以下目录中找到单元配置文件：

表 14.1. systemd 单元文件位置

目录	描述
/usr/lib/systemd/system/	与安装的 RPM 软件包一起分发的 **systemd** 单元文件。
/run/systemd/system/	在运行时创建的 **systemd** 单元文件。该目录优先于安装了的服务单元文件的目录。
/etc/systemd/system/	使用 **systemctl enable** 命令创建的 **systemd** 单元文件，以及为扩展服务添加的单元文件。这个目录优先于带有运行时单元文件的目录。

systemd 的默认配置在编译过程中定义，您可以在 /etc/systemd/system.conf 文件中找到配置。通过编辑此文件，您可以通过全局覆盖 systemd 单元的值来修改默认配置。

例如，若要覆盖设为 90 秒的超时限制的默认值，可使用 DefaultTimeoutStartSec 参数输入所需的值（以秒为单位）。

```
DefaultTimeoutStartSec=required value
```

14.2. 使用 SYSTEMCTL 管理系统服务

作为系统管理员，您可以使用 systemctl 工具管理系统服务。您可以执行各种任务，如启动、停止、重启运行的服务、启用和禁用服务以在引导时启动、列出可用的服务以及显示系统服务状态。

14.2.1. 列出系统服务

您可以列出所有当前载入的服务单元，并显示所有可用服务单元的状态。

流程

使用 systemctl 命令执行以下任何一个任务：

- 列出所有当前载入的服务单元：

```
$ 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'
```

默认情况下，systemctl list-units 命令只显示活跃的单位。对于每个服务单元文件，命令提供以下参数的概述：

UNIT

服务单元的全名

LOAD

配置文件的载入状态

ACTIVE 或 SUB

当前高级别和低级别单元文件激活状态

DESCRIPTION

单元目的和功能的简短描述

- 使用以下带有 --all 或 -a 命令行选项的命令，列出 所有载入的单元，而不考虑其状态：

```
$ systemctl list-units --type service --all
```

- 列出所有可用服务单元的状态(enabled 或 disabled)：

```
$ 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.
```

对于每个服务单元，这个命令会显示：

UNIT FILE

服务单元的全名

STATE

服务单元是否已启用或禁用，以便在引导时自动启动的信息

其他资源

- 显示系统服务状态

14.2.2. 显示系统服务状态

您可以检查任何服务单元以获取详细信息，并验证该服务的状态，无论是否启用了以便在引导期间启动还是当前正在运行。您还可以查看在特定的服务单元之后或之前启动的服务。

流程

使用 systemctl 命令执行以下任何一个任务：

- 显示与系统服务对应的服务单元的详细信息：

```
$ systemctl status <name>.service
```

将 *<name>* 替换为您要检查的服务单元的名称（例如：gdm）。

这个命令显示以下信息：

- 所选服务单元的名称，后跟一个简短描述
- 可用服务单元信息中 描述的一个或多个字段
- 服务单元的执行：如果单元由 root 用户执行
- 最新的日志条目

表 14.2. 可用的服务单元信息

项	描述
Loaded	是否服务单元已载入的信息、到单元文件的绝对路径，以及是否已启用该单元以便在引导时启动。
Active	服务单元是否在运行的信息，后面有一个时间戳。
Main PID	进程 ID 和相应的系统服务的名称。
Status	相关系统服务的额外信息。
Process	有关相关进程的附加信息。
CGroup	有关相关控制组(**cgroups**)的更多信息。

例 14.1. 显示服务状态

GNOME 显示管理器的服务单元名为 gdm.service。要确定这个服务单元的当前状态，

在 shell 提示下键入以下内容：

```
# 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.
```

- 验证特定的服务单元是否正在运行：

```
$ systemctl is-active <name>.service
```

- 确定是否已启用了特定的服务单元以便在引导时启动：

```
$ systemctl is-enabled <name>.service
```

注意

如果指定的服务单元正在运行或已启用，则 systemctl is-active 和 systemctl is-enabled 命令都会返回退出状态 0。

- 检查在指定的服务单元之前，systemd 命令哪些服务启动

```
# systemctl list-dependencies --after <name>.service
```

例如，要查看在 gdm 之前启动的服务的列表，请输入：

```
# 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]
```

- 检查在指定的服务单元之后，systemd 命令哪些服务启动：

```
# systemctl list-dependencies --before <name>.service
```

例如，要查看在 gdm 后 systemd 要启动的服务的列表，请输入：

```
# 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
```

其他资源

- 列出系统服务

14.2.3. 启动一个系统服务

您可以使用 start 命令在当前会话中启动系统服务。

先决条件

- 根访问权限

流程

- 在当前会话中启动一个系统服务：

```
# systemctl start <name>.service
```

将 *<name>* 替换为您要启动的服务单元的名称（例如 httpd.service）。

注意

在 systemd 中，服务之间存在正和负的依赖项。启动一个特定的服务可能需要启动一个或多个其他服务（正依赖项）或停止一个或多个服务（负依赖项）。

当您试图启动新服务时，systemd 会自动解析所有依赖项，而不会向用户明确通知。这意味着，如果您已运行了一个服务，并且您尝试使用负依赖项启动另一个服务，则第一个服务会自动停止。

例如，如果您正在运行 postfix 服务，并且您试图启动 sendmail 服务，systemd 首先自动停止 postfix，因为这两个服务会冲突且无法在同一端口上运行。

其他资源

- systemctl (1) 手册页
- 启用一个系统服务，以便在引导时启动
- 显示系统服务状态

14.2.4. 停止一个系统服务

如果要在当前会话中停止系统服务，请使用 stop 命令。

先决条件

- 根访问权限

流程

- 停止一个系统服务：

```
# systemctl stop <name>.service
```

将 *<name>* 替换为您要停止的服务单元的名称（例如：bluetooth）。

其他资源

- systemctl (1) 手册页
- 禁用一个系统服务在引导时启动
- 显示系统服务状态

14.2.5. 重启一个系统服务

您可以使用 restart 命令在当前会话中重启系统服务，以执行以下操作：

- 在当前会话中停止所选的服务单元，并立即再次启动它。
- 仅在对应的服务已在运行时才重启服务单元。
- 重新加载系统服务的配置，而不中断其执行。

先决条件

- 根访问权限

流程

- 重启一个系统服务：

```
# systemctl restart <name>.service
```

使用您要重启的服务单元的名称替换 *<name>*（例如 httpd）。

注意

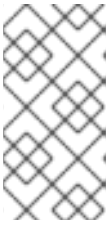
如果所选服务单元没有运行，这个命令也会启动它。

- 可选：只有在相应的服务已在运行时才重启服务单元：

```
# systemctl try-restart <name>.service
```

- 可选：重新载入配置，而不中断服务执行：

```
# systemctl reload <name>.service
```



注意

不支持此功能的系统服务忽略此命令。要重启这些服务，请使用 reload-or-restart 和 reload-or-try-restart 命令。

其他资源

- systemctl man page
- 显示系统服务状态

14.2.6. 启用一个系统服务，以便在引导时启动

您可以启用一个服务以便在引导时自动启动，这些更改将在下次重启时应用。

先决条件

- 根访问权限
- 您需要启用的服务不能被屏蔽。如果您有一个屏蔽的服务，请首先取消屏蔽：

```
# systemctl unmask <name>.service
```

流程

- 在引导时启用服务：

```
# systemctl enable <name>.service
```

使用您要启用的服务单元的名称替换 *<name>*（例如 httpd）。

- 可选：您也可以使用单个命令启用并启动一个服务：

```
# systemctl enable --now <name>.service
```

其他资源

- systemctl (1) 手册页
- 显示系统服务状态
- 启动一个系统服务

14.2.7. 禁用一个系统服务在引导时启动

您可以防止服务单元在引导时自动启动。如果您禁用某个服务，它不会在引导时启动，但可以手动启动。您还可以屏蔽服务，使其无法手动启动。屏蔽是一种禁用服务的方法，使该服务能够永久不可用，直到再次屏蔽该服务。

先决条件

- 根访问权限

流程

- 禁用要在引导时启动的服务：

```
# systemctl disable <name>.service
```

将 *<name>* 替换为您要禁用的服务单元的名称（例如：bluetooth）。

- 可选：如果要使服务永久不可用，请屏蔽该服务：

```
# systemctl mask <name>.service
```

这个命令将 /etc/systemd/system/*name*.service 文件替换为到 /dev/null 的符号链接，从而导致 systemd 无法访问实际的单元文件。

其他资源

- systemctl (1) 手册页
- 显示系统服务状态
- 停止一个系统服务

14.3. 引导至目标系统状态

作为系统管理员，您可以控制系统的引导过程，并定义您希望系统引导到的状态。这称为 systemd 目标，它是您的系统启动以达到某一特定级别功能的一组 systemd 单元。在使用 systemd 目标时，您可以查看默认目标，选择一个运行时的目标，更改默认引导目标，引导到紧急或救援目标。

14.3.1. 目标单元文件

systemd 中的目标是一组相关的单元，它们在系统启动期间充当同步点。目标单元文件以 .target 文件扩展名结尾，代表 systemd 目标。目标单元的目的是通过一组依赖项将各种 systemd 单元分组到一起。

请考虑以下示例：

- 用于启动图形会话的 graphical.target 单元启动系统服务，如 GNOME 显示管理器 (gdm.service)或 Accounts Service (accounts-daemon.service)，同时还激活 multi-user.target 单元。

- 同样，multi-user.target 单元启动其他基本系统服务，如 NetworkManager (NetworkManager.service) 或 D-Bus (dbus.service)，并激活另一个名为 basic.target 的目标单元。

您可以将以下 systemd 目标设置为默认或当前目标：

表 14.3. 常见 systemd 目标

rescue	在基本系统中拉取的单元目标，并生成一个救援 shell
多用户	用于设置多用户系统的单元目标
图形化	用于设置图形登录屏幕的单元目标
紧急	在主控制台上启动紧急 shell 的单元目标

其他资源

- systemd.special(7) man page
- systemd.target(5) 手册页

14.3.2. 更改引导到的默认目标

当系统启动时，systemd 激活 default.target 符号链接，该链接指向真正的目标单元。您可以在 /etc/systemd/system/default.target 文件中找到当前所选的默认目标单元。每个目标代表某个特定的功能级，用于对其他单元进行分组。另外，目标单元在引导过程中作为同步点提供服务。您可以更改系统引导到的默认目标。当您设置默认目标单元时，当前目标将保持不变，直到下次重启为止。

先决条件

- 根访问权限

流程

1. 确定当前用于启动系统的默认目标单元 systemd ：

```
# systemctl get-default
graphical.target
```

2. 列出当前载入的目标：

```
# systemctl list-units --type target
```

3. 将系统配置为默认使用不同的目标单元：

```
# systemctl set-default <name>.target
```

将 *<name>* 替换为您要默认使用的目标单元的名称。

```
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. 验证默认目标单元：

```
# systemctl get-default
multi-user.target
```

5. 通过重启来应用更改：

```
# reboot
```

其他资源

- systemctl (1) 手册页
- systemd.special(7) man page
- bootup (7) 手册页

14.3.3. 更改当前目标

在运行的系统中，您可以在不重启的情况下更改当前启动中的目标单元。如果您切换到不同的目

标，systemd 会启动这个目标需要的所有服务及其依赖项，并停止新目标没有启用的所有服务。隔离一个不同的目标只会影响当前引导。

流程

1. 可选：确定当前目标：

```
# systemctl get-default
graphical.target
```

2. 可选：显示您可以选择的目标列表：

```
# systemctl list-units --type target
```

注意

您只能隔离单元文件中设置了 AllowIsolate=yes 选项的目标。

3. 在当前引导中切换到不同的目标单元：

```
# systemctl isolate <name>.target
```

将 *<name>* 替换为您要在当前引导中使用的目标单元的名称。

```
Example:
# systemctl isolate multi-user.target
```

这个命令启动名为 multi-user 的目标单元和所有依赖的单元，并立即停止所有其他单元。

其他资源

- systemctl (1) 手册页

14.3.4. 引导至救援模式

您可以引导到提供单用户环境的 *救援模式*，以便在系统无法进入后续目标以及常规引导过程失败时进行故障排除或修复。在救援模式中，系统会尝试挂载所有本地文件系统，并启动某些重要的系统服务，但

不会激活网络接口。

先决条件

- 根访问权限

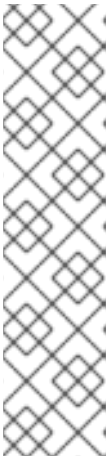
流程

- 要进入救援模式，在当前会话中更改当前目标：

```
# 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!
```



注意

这个命令与 systemctl isolate rescue.target 类似，但它也会向当前登录到该系统的所有用户发送信息性消息。

要防止 systemd 发送信息，输入带有以下命令行选项 --no-wall 的命令：

```
# systemctl --no-wall rescue
```

故障排除步骤

如果您的系统无法进入救援模式，您可以引导至 *紧急模式*，其提供尽可能小的环境。在紧急模式下，系统仅挂载用于读取的 root 文件系统，不会尝试挂载任何其他本地文件系统，不激活网络接口，并且仅启动几个必要的服务。

14.3.5. 引导过程故障排除

作为系统管理员，您可以在引导时选择非默认目标来对引导过程进行故障排除。在引导时更改目标仅会影响单个引导。您可以引导到 *紧急模式*，它提供尽可能小的环境。

流程

1. 重启系统，并通过按 Enter 键之外的任意键中断引导装载程序菜单倒计时，这将发起一个正

常启动。

2. 将光标移至要启动的内核条目。

3. 按 E 键编辑当前条目。

4. 移动到以 linux 开头的行的末尾，然后按 Ctrl+E 跳到行尾：

```
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. 要选择一个备用引导目标，请将 systemd.unit= 参数附加到以 linux 开头的行的末尾：

```
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
```

将 ***<name>*** 替换为您要使用的目标单元的名称。例如：systemd.unit=emergency.target

6. 按 Ctrl+X 使用这些设置进行引导。

14.4. 关闭、挂起和休眠系统

作为系统管理员，您可以使用不同的电源管理选项来管理功耗，执行合适的 shutdown 以确保保存所有数据，或者重启系统以应用更改和更新。

14.4.1. 系统关闭

要关闭系统，您可以直接使用 systemctl 工具，或者通过 shutdown 命令来调用这个工具。

使用 shutdown 有以下优点：

- 您可以使用 time 参数来安排一个 shutdown。这也会警告用户系统已计划 shutdown。

- 您可以取消 shutdown。

其他资源

- systemctl 的电源管理命令的概述

14.4.2. 安排一个系统 shutdown

作为系统管理员，您可以安排一个延迟 shutdown，给用户保存其工作及注销系统留出时间。使用 shutdown 命令执行以下操作：

- 关闭系统并在一定时间后关闭机器
- 在不关掉机器电源的情况下关闭和停止系统
- 取消待处理的 shutdown

先决条件

- 根访问权限

流程

使用 shutdown 命令执行以下任何一个任务：

- 指定您要关闭系统并关闭机器的时间：

```
# shutdown --poweroff hh:mm
```

其中 *hh:mm* 是 24 小时时间表示法的时间。为防止新的登录，在系统 shutdown 前 5 分钟会创建 /run/nologin 文件 。

当使用时间参数时，您可以通过指定可选的 *wall 消息* 来通知登录到计划关闭的系统的用户，如 shutdown --poweroff 13:59 "Attention。系统将于 13:59" 关闭。

- 在延迟后关闭和停止系统，而不关闭机器：

```
# shutdown --halt +m
```

其中 *+m* 是延迟时间（以分钟为单位）。您可以使用 now 关键字作为 +0 的别名。

- 取消一个待处理的 shutdown：

```
# shutdown -c
```

其他资源

- shutdown(8) 手册页
- 使用 systemctl 命令关闭系统

14.4.3. 使用 systemctl 命令来关闭系统

作为系统管理员，您可以关闭系统并关闭机器，或使用 systemctl 命令关闭和停止系统，而不关掉机器电源。

先决条件

- 根访问权限

流程

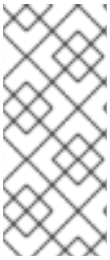
使用 systemctl 命令执行以下任何一个任务：

- 关闭系统并关掉机器电源：

```
# systemctl poweroff
```

- 关闭和停止系统，而不关掉机器电源：

```
# systemctl halt
```



注意

默认情况下，运行其中任何一个命令都可让 systemd 向当前登录到该系统的所有用户发送一条信息性消息。要防止 systemd 发送这条消息，请使用 --no-wall 命令行选项运行所选命令。

14.4.4. 重启系统

当您重启系统时，systemd 会停止所有正在运行的程序和服务，系统会关闭，然后立即启动。在以下情况下重启系统很有帮助：

- 安装新软件或更新后
- 更改系统设置后
- 故障排除系统问题时

先决条件

- 根访问权限

流程

- 重启系统：

```
# systemctl reboot
```

注意

默认情况下，当您使用此命令时，systemd 会向当前登录到该系统的所有用户发送一条信息性消息。要防止 systemd 发送这条消息，请使用 --no-wall 选项运行这个命令。

14.4.5. 通过挂起和休眠系统来优化功耗

作为系统管理员，您可以管理功耗，节省系统能源，并保留系统的当前状态。要做到这一点，请应用以下模式之一：

suspend

挂起会将系统状态保存在 RAM 中，但 RAM 模块除外，关闭机器中的大多数设备。当您重新打开机器时，系统会从内存中恢复其状态，而无需再次引导。由于系统状态保存在 RAM 中，而不是保存在硬盘上，因此，从挂起模式恢复系统比从休眠模式恢复要快得多。但是，挂起的系统状态也会受到断电的影响。

Hibernate

休眠会在硬盘上保存系统状态，并关闭机器。当您重新打开机器时，系统会从保存的数据中恢复其状态，而无需再次引导。由于系统状态保存在硬盘上，而不是保存在 RAM 中，因此机器不必保持对 RAM 模块的供电。但是，因此，从休眠模式恢复系统要比将其恢复为挂起模式恢复要慢得多。

Hybrid sleep

合并了休眠和挂起的元素。系统首先在硬盘上保存当前状态，并进入类似挂起的低电源状态，这允许系统更快地恢复。混合睡眠的好处是，如果系统在睡眠状态下断电，它仍然可以从硬盘上保存的镜像中恢复之前的状态，类似于休眠。

suspend-then-hibernate

此模式首先挂起系统，这会将当前系统状态保存到 RAM，并将系统置于低电源模式。如果系统保持挂起一段时间，则系统会休眠，您可以在 HibernateDelaySec 参数中定义。休眠将系统状态保存到硬盘上，并完全关闭系统。suspend-then-hibernate 模式提供了保留电池电源的好处，同时您仍能快速地恢复工作。另外，这个模式确保您的数据在出现电源故障时被保存。

先决条件

- 根访问权限

流程

选择适当的节能方法：

- 挂起系统：

```
# systemctl suspend
```

- 休眠系统：

```
# systemctl hibernate
```

- 休眠并挂起系统：

```
# systemctl hybrid-sleep
```

- 挂起然后休眠系统：

```
# systemctl suspend-then-hibernate
```

14.4.6. systemctl 的电源管理命令的概述

您可以使用以下 systemctl 命令列表来控制系统的电源管理。

表 14.4. systemctl 电源管理命令的概述

systemctl 命令	描述
systemctl halt	关闭系统。
systemctl poweroff	关闭系统。
systemctl reboot	重启该系统。
systemctl suspend	挂起系统。
systemctl hibernate	休眠系统。
systemctl hybrid-sleep	休眠并挂起系统。

14.4.7. 更改电源按钮行为

当您在计算机上按 power 按钮时，它会默认挂起或关闭系统。您可以根据您的偏好自定义此行为。

14.4.7.1. 更改 systemd 中的电源按钮行为

当您在非图形 systemd 目标中按 power 按钮时，它会默认关闭系统。您可以根据您的偏好自定义此行为。

先决条件

- 管理访问权限.

流程

1. 打开 /etc/systemd/logind.conf 配置文件。
2. 查找有 HandlePowerKey=poweroff 的行。
3. 如果行以 # 符号开头，请将其删除以启用设置。
4. 使用以下选项之一替换 poweroff ：

poweroff

关闭计算机。

reboot

重启系统。

halt

启动系统停止。

kexec

启动 kexec 重启。

suspend

挂起系统。

hibernate

启动系统休眠。

ignore

什么都不做。

例如，要在按下电源按钮时重启系统，请使用这个设置：

```
HandlePowerKey=reboot
```

5. 保存更改并关闭编辑器。

后续步骤

- 如果您使用图形会话，还要在 GNOME 中配置电源按钮。请参阅 第 14.4.7.2 节 “更改 GNOME 中的电源按钮行为”。

14.4.7.2. 更改 GNOME 中的电源按钮行为

在图形登录屏幕或在图形用户会话中，按 power 按钮默认挂起机器。当用户物理按下 power 按钮或从远程控制台按下虚拟 power 按钮时，才会出现这种情况。您可以选择不同的 power 按钮行为。

先决条件

- 您已在 systemd 中配置了电源按钮行为。请参阅 第 14.4.7.1 节 “更改 systemd 中的电源按钮行为”。

流程

1. 在 /etc/dconf/db/local.d/01-power 文件中为系统范围的设置创建一个本地数据库。输入以下内容：

```
[org/gnome/settings-daemon/plugins/power]
power-button-action='suspend'
```

使用以下 power 按钮操作之一替换 *suspend* ：

nothing

什么都不做。

suspend

挂起系统。

hibernate

休眠系统。

interactive

显示一个弹出窗口查询，询问用户要做什么。

使用交互模式时，在按下 power 按钮后，系统会在 60 秒后自动关闭。但是，您可以从弹出查询中选择不同的行为。

2. 可选：覆盖用户的设置，并阻止用户更改它。在 /etc/dconf/db/local.d/locks/01-power 文件中输入以下配置：

```
/org/gnome/settings-daemon/plugins/power/power-button-action
```

3. 更新系统数据库：

```
# dconf update
```

4. 注销并重新登录，使系统范围的设置生效。

第 15 章 配置时间同步

在 IT 环境中保持准确的时间非常重要。所有跨网络设备的一致的时间提高了日志文件的可追溯性，以及某些依赖同步时钟的协议。例如，Kerberos 使用时间戳来防止重播攻击。

15.1. 使用 CHRONY 套件配置 NTP

准确的计时在 IT 中很重要，原因有多个。例如在网络中，需要准确的数据包和日志的时间戳。在 Linux 系统中，NTP 协议是由在用户空间运行的守护进程实现的。

用户空间守护进程更新内核中运行的系统时钟。系统时钟可以通过使用不同的时钟源来维护系统的时间。通常, 使用 *时间戳计数器*（TSC）。TSC 是一个 CPU 寄存器，它计算从上次重置的循环数。它非常快，分辨率很高，且不会被中断。

从 Red Hat Enterprise Linux 8 开始，NTP 协议由 chronyd 守护进程实现，它可从 chrony 软件包中的存储库中获得。

以下章节描述了如何使用 chrony 套件来配置 NTP。

15.1.1. chrony 套件介绍

chrony 是 网络时间协议(NTP) 的一种实现。您可以使用 chrony:

- 将系统时钟与 NTP 服务器同步
- 将系统时钟与参考时钟同步，如 GPS 接收器
- 将系统时钟与手动时间输入同步
- 作为 NTPv4(RFC 5905) 服务器或对等服务器，为网络中的其他计算机提供时间服务

在多数条件下，chrony 都会表现良好，包括时断时续的网络连接、有大量网络数据的网络、温度不稳定（普通计算机时钟对温度敏感）以及不持续运行或在虚拟机上运行的系统。

通过互联网镜像同步的两天机器之间的准确性通常在几毫秒之内，而对于 LAN 中的机器则为几十微秒。硬件时间戳或硬件参考时钟可以将两台计算机之间的准确性提高到子微秒级。

chrony 包括 chronyd（一个在用户空间运行的守护进程）和 chronyc（可用来监控 chronyd 性能并在运行时更改各种操作参数的命令行程序）。

chrony 守护进程（chronyd） 可以由命令行工具 chronyc 监控和控制。这个工具提供了一个命令提示，允许输入大量命令来查询 chronyd 的当前状态并修改其配置。在默认情况下，chronyd 只接受来自本地 chronyc 实例的命令，但它也可以被配置为接受来自远程主机的监控命令。应该限制远程访问。

15.1.2. 使用 chronyc 来控制 chronyd

您可以使用 chronyc 命令行工具控制 chronyd。

流程

1. 要在互动模式中使用命令行工具 chronyc 来更改本地 chronyd 实例，以根用户身份输入以下命令：

```
# chronyc
```

如果要使用某些受限命令，chronyc 需要以 root 运行。

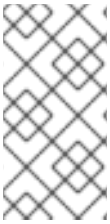
chronyc 命令提示符如下所示：

```
chronyc>
```

2. 要列出所有的命令，请输入 help。

3. 或者，如果与以下命令一同调用，该工具也可以在非交互命令模式下调用：

```
chronyc command
```



注意

使用 chronyc 所做的更改不具有持久性，它们会在 chronyd 重启后丢失。要使更改有持久性，修改 /etc/chrony.conf。

15.1.3. 迁移到 chrony

在 Red Hat Enterprise Linux 7 中，用户可以在 ntp 和 chrony 之间进行选择，以确保准确计时。有关 ntp 和 chrony、ntpd 和 chronyd 之间的区别，请参阅 ntpd 和 chronyd 之间的差别。

从 Red Hat Enterprise Linux 8 开始，不再支持 ntp。chrony 默认启用。因此，您可能需要从 ntp 迁移到 chrony。

在大多数情况下，从 ntp 迁移到 chrony 是非常直接的。程序、配置文件和服务的相应名称为：

表 15.1. 从 ntp 迁移到 chrony 时的程序、配置文件和服务对应的名称

ntp 名称	chrony 名称
/etc/ntp.conf	/etc/chrony.conf
/etc/ntp/keys	/etc/chrony.keys
ntpd	chronyd
ntpq	chronyc
ntpd.service	chronyd.service
ntp-wait.service	chrony-wait.service

通过使用 -q 选项或 -t 选项，chronyd 可以替代 ntpdate 和 sntp 程序（包含在 ntp 发布中）。可在命令行中指定配置以避免读取 /etc/chrony.conf。例如：如下所示运行 chronyd 可以替代运行 ntpdate ntp.example.com：

```
# 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
```

ntpstat 工具程序之前包含在 ntp 软件包中，且只支持 ntpd。现在它支持 ntpd 和 chronyd。它现在包括在 ntpstat 软件包中。

15.1.3.1. 迁移脚本

名为 ntp2chrony.py 的 Python 脚本包含在 chrony 软件包文档中（/usr/share/doc/chrony）。这个脚本会自动将现有的 ntp 配置转换为 chrony。它支持 ntp.conf 文件中最常用的指令和选项。所有在转换中忽略的行都会作为注释包含在生成的 chrony.conf 文件中以便用户进行核查。在 ntp 密钥文件中指定但未在 ntp.conf 中被标记为可信密钥的密钥会作为注释出现在生成的 chrony.keys 文件中。

默认情况下，该脚本不会覆盖任何文件。如果 /etc/chrony.conf 或 /etc/chrony.keys 已经存在，使用 -b 选项可以重新命名文件以作为备份。这个脚本支持其他选项。--help 选项输出所有支持选项。

在 ntp 软件包中提供了一个默认 ntp.conf 调用脚本示例：

```
# 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
```

本例中唯一忽略的指令是 disable monitor，它在 noclientlog 指令中有一个等同的 chrony 项。它包括在默认 ntp.conf 中只是用于缓解一个安全工具。

生成的 chrony.conf 文件通常包含大量与 ntp.conf 中限制行对应的 allow 指令。如果您不想使用 chronyd 作为 NTP 服务器，从 chrony.conf 中删除所有 allow 指令。

15.2. 使用 CHRONY

以下章节介绍了如何安装、启动和停止 chronyd，以及如何检查 chrony 是否同步。这些章节还介绍了如何手动调整系统时钟。

15.2.1. 管理 chrony

以下流程描述了如何安装、启动、停止和检查 chronyd 的状态。

流程

1. 默认在 Red Hat Enterprise Linux 上安装 chrony 套件。以 root 用户运行以下命令进行验证：

```
# yum install chrony
```

chrony 守护进程的默认位置为 /usr/sbin/chronyd。命令行工具将安装到 /usr/bin/chronyc。

2. 运行以下命令检查 chronyd 的状态：

```
$  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. 要启动 chronyd，使用 root 用户身份运行以下命令：

```
# systemctl start chronyd
```

要确保 chronyd 在系统启动时自动启动，以 root 身份运行以下命令：

```
# systemctl enable chronyd
```

4. 要停止 chronyd，以 root 身份运行以下命令：

```
# systemctl stop chronyd
```

要防止 chronyd 在系统启动时自动启动，以 root 身份运行以下命令：

```
# systemctl disable chronyd
```

15.2.2. 检查是否同步 chrony

以下流程描述了如何检查 chrony 是否与 tracking、sources 和 sourcestats 命令的使用同步。

流程

1. 运行以下命令检查 chrony 跟踪：

```
$  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 NP 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. sources 命令显示 chronyd 正在访问的当前时间源的信息。要检查 chrony 源，请运行以下命令：

```
$ 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
```

您可以指定可选的 -v 参数来打印更详细的信息。在这种情况下，会输出额外的标头行显示字段含义的信息。

3. sourcestats 命令显示目前被 chronyd 检查的每个源的偏移率和误差估算过程的信息。要检查 chrony 源的统计信息，请运行以下命令：

```
$  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
```

可以使用可选参数 -v 来包括详细信息。在这种情况下，会输出额外的标头行显示字段含义的信息。

其他资源

- chronyc(1) 手册页

15.2.3. 手动调整系统时钟

下面的流程描述了如何手动调整系统时钟。

流程

1. 要立即调整系统时钟，绕过单机进行的任何调整，以 root 身份运行以下命令：

```
# chronyc makestep
```

如果使用了 rtcfile 指令，则不应该手动调整实时时钟。随机调整会影响 chrony 测量实时时钟漂移速率的需要。

15.2.4. 禁用 chrony 分配程序脚本

chrony 分配程序脚本管理 NTP 服务器的在线和离线状态。作为系统管理员，您可以禁用分配程序脚本，以使 chronyd 持续轮询服务器。

如果在系统中启用 NetworkManager 来管理网络配置，NetworkManager 会在接口重新配置，停止或启动操作过程中执行 chrony 分配程序脚本。但是，如果您在 NetworkManager 之外配置某些接口或路由，您可能会遇到以下情况：

1. 当没有到 NTP 服务器的路由存在时，分配程序脚本可能会运行，从而导致 NTP 服务器切换到离线状态。

2. 如果您稍后建立路由，脚本默认不会再次运行，NTP 服务器保持在离线状态。

要确保 chronyd 可以与您的 NTP 服务器同步（后者有单独的受管接口），请禁用分配程序脚本。

先决条件

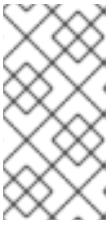
- 您在系统中安装了 NetworkManager 并启用了它。

- 根访问权限

流程

1. 要禁用 chrony 分配程序脚本，请编辑 /etc/NetworkManager/dispatcher.d/20-chrony-onoffline 文件，如下所示：

```
#!/bin/sh
exit 0
```



注意

当您升级或重新安装 chrony 软件包时，分配程序脚本的打包版本会替换您修改的分配程序脚本。

15.2.5. 在隔离的网络中为系统设定 chrony

对于从未连接到互联网的网络，一台计算机被选为主计时服务器。其他计算机是服务器的直接客户端，也可以是客户端的客户端。在服务器上，必须使用系统时钟的平均偏移率手动设置 drift 文件。如果服务器被重启，它将从周围的系统获得时间，并计算设置系统时钟的平均值。之后它会恢复基于 drift 文件的调整。当使用 settime 命令时会自动更新 drift 文件。

以下流程描述了如何为隔离的网络中的系统设置 chrony。

流程

1. 在选择为服务器的系统中，以 root 用户身份运行一个文本编辑器，编辑 /etc/chrony.conf，如下所示：

```
driftfile /var/lib/chrony/drift
commandkey 1
keyfile /etc/chrony.keys
initstepslew 10 client1 client3 client6
local stratum 8
manual
allow 192.0.2.0/24
```

其中 192.0.2.0/24 是允许客户端连接的网络或子网地址。详情请查看 chrony.conf (7) man page

2.

在选择成为服务器客户端的系统上，以 root 用户身份运行一个文本编辑器来编辑 /etc/chrony.conf，如下所示：

```
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
```

其中 192.0.2.123 是服务器的地址，ntp1.example.net 是服务器的主机名。带有此配置的客户端如果服务器重启，则与服务器重新同步。

在不是服务器直接客户端的客户端系统中，/etc/chrony.conf 文件应该相同，除了应该省略 local 和 allow 指令。

在隔离的网络中，您还可以使用 local 指令来启用本地参考模式。该模式可允许 chronyd 作为 NTP 服务器实时显示同步，即使它从未同步或者最后一次更新时钟早前发生。

要允许网络中的多个服务器使用相同的本地配置并相互同步，而不让客户端轮询多个服务器，请使用 local 指令的 orphan 选项启用孤立模式。每一个服务器都需要配置为使用 local 轮询所有其他服务器。这样可确保只有最小参考 ID 的服务器具有本地参考活跃状态，其他服务器与之同步。当服务器出现故障时，另一台服务器将接管。

15.2.6. 配置远程监控访问

chronyc 可以通过两种方式访问 chronyd:

- 互联网协议、IPv4 或者 IPv6。
- UNIX 域套接字，由 root 用户或 chrony 用户从本地进行访问。

默认情况下，chronyc 连接到 Unix 域套接字。默认路径为 /var/run/chrony/chronyd.sock。如果这个连接失败，比如，当 chronyc 在非 root 用户下运行时会发生，chronyc 会尝试连接到 127.0.0.1，然后 ::1。

网络中只允许以下监控命令，它们不会影响 chronyd 的行为：

- activity
- manual list
- rtcdata
- smoothing
- sources
- sourcestats
- tracking
- waitsync

chronyd 接受这些命令的主机集合可以使用 chronyd 配置文件中的 cmdallow 指令，或者在 chronyc 中使用 cmdallow 命令配置。默认情况下，仅接受来自 localhost（127.0.0.1 或 ::1）的命令。

所有其他命令只能通过 Unix 域套接字进行。当通过网络发送时，chronyd 会返回 Notauthorized 错误，即使它来自 localhost。

以下流程描述了如何使用 chronyc 远程访问 chronyd。

流程

1. 在 /etc/chrony.conf 文件中添加以下内容来允许 IPv4 和 IPv6 地址的访问：

```
bindcmdaddress 0.0.0.0
```

或者

```
bindcmdaddress ::
```

2. 使用 cmdallow 指令允许来自远程 IP 地址、网络或者子网的命令。

在 /etc/chrony.conf 文件中添加以下内容：

```
cmdallow 192.168.1.0/24
```

3. 在防火墙中打开端口 323 以从远程系统连接：

```
# firewall-cmd --zone=public --add-port=323/udp
```

另外，您可以使用 --permanent 选项永久打开端口 323：

```
# firewall-cmd --permanent --zone=public --add-port=323/udp
```

4. 如果您永久打开了端口 323，请重新载入防火墙配置：

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

其他资源

- chrony.conf(5) 手册页

15.2.7. 使用 RHEL 系统角色管理时间同步

您可以使用 timesync 角色在多个目标机器上管理时间同步。timesync 角色安装和配置 NTP 或 PTP 实现，作为 NTP 或 PTP 客户端来同步系统时钟。

请注意，使用 timesync 角色还可帮助 迁移到 chrony，因为您可以在从 RHEL 6 开始的所有 Red Hat Enterprise Linux 版本上使用相同的 playbook，而无论系统是否使用了 ntp 或 chrony 来实现 NTP 协议。

警告

timesync 角色替换了受管主机上给定或检测到的供应商服务的配置。之前的设置即使没有在角色变量中指定，也会丢失。如果没有定义 timesync_ntp_provider 变量，唯一保留的设置就是供应商选择。

以下示例演示了如何在只有一个服务器池的情况下应用 timesync 角色。

例 15.1. 为单一服务器池应用 timesync 角色的 playbook 示例

```
---
- hosts: timesync-test
  vars:
    timesync_ntp_servers:
      - hostname: 2.rhel.pool.ntp.org
        pool: yes
        iburst: yes
  roles:
    - rhel-system-roles.timesync
```

有关 timesync 角色变量的详细参考，请安装 rhel-system-roles 软件包，并参阅 /usr/share/doc/rhel-system-roles/timesync 目录中的 README.md 或 README.html 文件。

其他资源

- 准备一个控制节点和受管节点以使用 RHEL 系统角色

15.2.8. 其他资源

- chronyc(1) 手册页

- chronyd(8) 手册页
- 常见问题解答

15.3. 带有 HW 时间戳的 CHRONY

硬件时间戳是在一些网络接口控制器(NIC)中支持的一种功能，它提供传入和传出数据包的准确的时间戳。NTP 时间戳通常由内核及使用系统时钟的 chronyd 创建。但是，当启用了 HW 时间戳时，NIC 使用自己的时钟在数据包进入或离开链路层或物理层时生成时间戳。与 NTP 一起使用时，硬件时间戳可以显著提高同步的准确性。为了获得最佳准确性，NTP 服务器和 NTP 客户端都需要使用硬件时间戳。在理想条件下，可达到次微秒级的准确性。

另一个用于使用硬件时间戳进行时间同步的协议是 PTP

与 NTP 不同，PTP 依赖于网络交换机和路由器。如果您想要达到同步的最佳准确性，请在带有 PTP 支持的网络中使用 PTP，在使用不支持这个协议的交换机和路由器的网络上选择 NTP。

以下小节描述了如何进行：

- 验证是否支持硬件时间戳
- 启用硬件时间戳
- 配置客户端轮询间隔
- 启用交错模式
- 为大量客户端配置服务器
- 验证硬件时间戳

- 配置 PTP-NTP 网桥

15.3.1. 验证硬件时间戳支持

要验证接口是否支持使用 NTP 的硬件时间戳，请使用 ethtool -T 命令。如果 ethtool 列出了 SOF_TIMESTAMPING_TX_HARDWARE 和 SOF_TIMESTAMPING_TX_SOFTWARE 模式，以及 HWTSTAMP_FILTER_ ALL 过滤器模式，则可以使用硬件时间戳的 NTP。

例 15.2. 在特定接口中验证硬件时间戳支持

```
# ethtool -T eth0
```

输出：

```
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. 启用硬件时间戳

要启用硬件时间戳，请使用 /etc/chrony.conf 文件中的 hwtimestamp 指令。该指令可指定单一接口，也可以指定通配符字符来启用所有支持接口的硬件时间戳。在没有其他应用程序（如 linuxptp 软件包中的 ptp4l 在接口上使用硬件时间戳）的情况下，请使用通配符规范。在 chrony 配置文件中允许使用多个 hwtimestamp 指令。

例 15.3. 使用 hwtimestamp 指令启用硬件时间戳

```
hwtimestamp eth0
hwtimestamp eth1
hwtimestamp *
```

15.3.3. 配置客户端轮询间隔

建议为互联网中的服务器使用默认的轮询间隔范围（64-1024秒）。对于本地服务器和硬件时间戳，需要配置一个较短的轮询间隔，以便最小化系统时钟偏差。

/etc/chrony.conf 中的以下指令指定了使用一秒轮询间隔的本地 NTP 服务器：

```
server ntp.local minpoll 0 maxpoll 0
```

15.3.4. 启用交错模式

NTP 服务器不是硬件的 NTP 设备，而是运行软件 NTP 实现的通用计算机，如 chrony，将在发送数据包后才会获得硬件传输时间戳。此行为可防止服务器在其对应的数据包中保存时间戳。为了使 NTP 客户端接收传输后生成的传输时间戳，请将客户端配置为使用 NTP 交错模式，方法是在 /etc/chrony.conf 的 server 指令中添加 xleave 选项：

```
server ntp.local minpoll 0 maxpoll 0 xleave
```

15.3.5. 为大量客户端配置服务器

默认服务器配置允许最多几千个客户端同时使用交错模式。要为更多的客户端配置服务器，增大 /etc/chrony.conf 中的 clientloglimit 指令。这个指令指定了为服务器上客户端访问的日志分配的最大内存大小：

```
clientloglimit 100000000
```

15.3.6. 验证硬件时间戳

要校验该接口是否已成功启用了硬件时间戳，请检查系统日志。这个日志应该包含来自 chronyd 的每个接口的消息，并成功启用硬件时间戳。

例 15.4. 为启用硬件时间戳的接口记录日志信息

```
chronyd[4081]: Enabled HW timestamping on eth0
chronyd[4081]: Enabled HW timestamping on eth1
```

当 chronyd 被配置为 NTP 客户端或对等的客户端时，您可以使用 chronyc ntpdata 命令为每个 NTP 源报告传输和接收时间戳模式以及交错模式：

例 15.5. 报告每个 NTP 源的传输、接收时间戳以及交集模式

```
# chronyc ntpdata
```

输出：

```
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
```

例 15.6. 报告 NTP 测量的稳定性

```
# chronyc sourcestats
```

启用硬件时间戳后，正常负载下，NTP 测量的稳定性应该以十秒或数百纳秒为单位。此稳定性会在 chronyc sourcestats 命令的输出结果中的 Std Dev 列中报告：

输出：

```
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. 配置 PTP-NTP 桥接

如果网络中有一个高度准确的 Precision Time Protocol (PTP)主时间服务器，但没有支持 PTP 支持的交换机或路由器，则计算机可能专用于作为 PTP 客户端和 stratum-1 NTP 服务器。此类计算机需要具有两个或更多个网络接口，并且接近主时间服务器或与它直接连接。这样可保证高度准确的网络同步。

从 linuxptp 软件包中配置 ptp4l 和 phc2sys 程序，以使用 PTP 来同步系统时钟。

将 chronyd 配置为使用其他接口提供系统时间：

例 15.7. 将 chronyd 配置为使用其他接口提供系统时间

```
bindaddress 203.0.113.74
hwtimestamp eth1
local stratum 1
```

15.4. 在 CHRONY 中启用一些之前由 NTP 支持的设置

chrony 不支持在以前由 ntp 支持的 Red Hat Enterprise Linux 主版本中的某些设置。以下小节列出了此类设置，并描述了在具有 chrony 的系统中实现它们的方法。

15.4.1. 使用 ntpq 和 ntpdc 进行监控

chronyd 无法被由 ntp 提供的 ntpq 和 ntpdc 监控，因为 chrony 不支持 NTP 模式 6 和 7。它支持不同的协议，chronyc 是一个客户端的实现。如需更多信息，请参阅 chronyc (1) man page。

要监控使用 chronyd 的系统时钟的状态，您可以：

- 使用 tracking 命令

- 使用支持 chrony 的 ntpstat 工具，它提供和 ntpd类似的输出。

例 15.8. 使用跟踪命令

```
$ 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
```

例 15.9. 使用 ntpstat 程序

```
$ 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. 使用基于公钥加密的认证机制

在 Red Hat Enterprise Linux 7 中，ntp 支持的 Autokey，它是一个基于公钥加密的验证机制。

在 Red Hat Enterprise Linux 8 中，chronyd 支持 Network Time Security (NTS)，它是一个现代安全身份验证机制，而不是 Autokey。如需更多信息，请参阅 chrony 中网络时间协议(NTS)的概述。

15.4.3. 使用临时对称关联

在 Red Hat Enterprise Linux 7 中，ntpd 支持的临时对称关联（可以通过未在 ntp.conf 配置文件中指定的 peer）来移动化。在 Red Hat Enterprise Linux 8 中，chronyd 需要在 chrony.conf 中指定所有对等点。不支持临时对称关联。

请注意，使用通过 server 或 pool 指令启用的客户端/服务器模式和通过 peer 指令启用的对称模式相比，更为安全。

15.4.4. 多播/广播客户端

Red Hat Enterprise Linux 7 支持广播/多播 NTP 模式，该模式简化了客户端配置。使用这个模式，客户端可以配置为仅侦听发送到多播/广播地址的数据包，而不是侦听各个服务器的特定名称或地址，这可能会随时间变化。

在 Red Hat Enterprise Linux 8 中，chronyd 不支持广播/多播模式。主要的原因是它比一般的客户端/服务器以及对称模式的准确性较低且安全性较低。

从 NTP 广播/多播设置中迁移有几个选项：

- 将 DNS 配置为将单个名称（如 ntp.example.com）转换为不同服务器的多个地址。

  客户端只能使用单一池指令来与多个服务器同步进行静态配置。如果池中的服务器变得不可访问，或者不适合同步，客户端会自动将其替换为池中的另一台服务器。

- 通过 DHCP 分配 NTP 服务器列表

  当 NetworkManager 从 DHCP 服务器获得 NTP 服务器列表时，chronyd 会自动配置来使用它们。把 PEERNTP=no 添加到 /etc/sysconfig/network 文件可以禁用这个功能。

- 使用 精确时间协议 (PTP)

  这个选项主要适用于服务器经常更改的环境，或者大量的客户端需要在没有指定服务器的情况下相互同步。

  PTP 是为多播消息设计的，它的工作方式与 NTP 广播模式相似。linuxptp 软件包中包括了一个 PTP 实现。

  PTP 通常需要硬件时间戳并支持网络开关执行正常。但是，即使软件时间戳没有支持网络交换机，PTP 也应该比广播模式中的 NTP 更好地工作。

  在一个通信路径中有大量 PTP 客户端的网络中，建议使用 hybrid_e2e 选项配置 PTP 客户端，以减少客户端生成的网络流量。您可以将运行 chronyd 的计算机配置为 NTP 客户端，也可

配置为 NTP 服务器，来作为主 PTP 时间服务器运作，以使用多播消息传递将同步的时间分发到大量的计算机上。

15.5. CHRONY 中的网络时间安全概述(NTS)

Network Time Security(NTS)是用于网络时间协议(NTP)的身份验证机制，旨在扩展大量客户端。它将验证从服务器计算机接收的数据包在移到客户端机器时是否被取消处理。Network Time Security(NTS)包含 Key Establishment(NTS-KE)协议，该协议会自动创建在服务器及其客户端中使用的加密密钥。

警告

NTS 与 FIPS 和 OSPP 配置文件不兼容。当您启用 FIPS 和 OSPP 配置文件时，使用 NTS 配置的 chronyd 可能中止，并显示一条致命消息。您可以通过将 GNUTLS_FORCE_FIPS_MODE=0 添加到 /etc/sysconfig/chronyd 文件中，来禁用 chronyd 服务的 OSPP 配置文件和 FIPS 模式。

15.5.1. 在客户端配置文件中启用网络时间协议(NTS)

默认情况下不启用 Network Time Security(NTS)。您可以在 /etc/chrony.conf 中启用 NTS。为此，请执行以下步骤：

先决条件

- 带有 NTS 支持的服务器

流程

在客户端配置文件中：

1. 除推荐的 iburst 选项外，使用 nts 选项指定服务器。

```
For example:
server time.example.com iburst nts
server nts.netnod.se iburst nts
server ptbtime1.ptb.de iburst nts
```

2.

要避免在系统引导时重复 Network Time Security-Key Establishment(NTS-KE)会话，请在 chrony.conf 中添加以下行（如果不存在）：

```
ntsdumpdir /var/lib/chrony
```

3. 将以下行添加到 /etc/sysconfig/network 以禁用与 DHCP 提供的网络时间协议(NTP)服务器的同步：

```
PEERNTP=no
```

4. 保存您的更改。

5. 重启 chronyd 服务：

```
systemctl restart chronyd
```

验证

- 验证 NTS 密钥是否已成功建立：

```
# 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
```

KeyID、Type 和 KLen 应带有非零值。如果该值为零，请检查系统日志中来自 chronyd 的错误消息。

- 验证客户端是否正在进行 NTP 测量：

```
# 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
```

Reach 列中应具有非零值；理想情况是 377。如果值很少为 377 或永远不是 377，这表示 NTP 请求或响应在网络中丢失。

其他资源

- chrony.conf(5) 手册页

15.5.2. 在服务器上启用网络时间安全性(NTS)

如果您运行自己的网络时间协议(NTP)服务器，您可以启用服务器网络时间协议(NTS)支持来促进其客户端安全地同步。

如果 NTP 服务器是其它服务器的客户端，即它不是 Stratum 1 服务器，它应使用 NTS 或对称密钥进行同步。

先决条件

- 以 PEM 格式的服务器私钥
- 带有 PEM 格式的所需中间证书的服务器证书

流程

1. 在 chrony.conf 中指定私钥和证书文件。例如：

```
ntsserverkey /etc/pki/tls/private/<ntp-server.example.net>.key
ntsservercert /etc/pki/tls/certs/<ntp-server.example.net>.crt
```

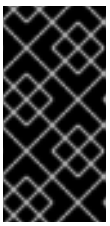
2. 通过设置组所有权，确保 chrony 系统用户可读密钥和证书文件。例如：

```
# chown :chrony /etc/pki/tls//<ntp-server.example.net>.
```

3. 确保 chrony.conf 中存在 ntsdumpdir /var/lib/chrony 指令。

4. 重启 chronyd 服务：

```
# systemctl restart chronyd
```



重要

如果服务器具有防火墙，则需要允许 NTP 和 Network Time Security-Key Establishment(NTS-KE)的 UDP 123 和 TCP 4460 端口。

验证

- 使用以下命令从客户端机器执行快速测试：

```
$ 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
```

System clock wrong 消息指示 NTP 服务器接受 NTS-KE 连接并使用 NTS 保护的 NTP 消息进行响应。

- 验证 NTS-KE 连接并验证服务器中观察的 NTP 数据包：

```
# 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
```

如果 NTS-KE connections accepted 和 Authenticated NTP packets 项带有一个非零值，这意味着至少有一个客户端能够连接到 NTS-KE 端口并发送经过身份验证的 NTP 请求。

第 16 章 使用 LANGPACKS

为系统的每个软件包安装额外附加软件包（包含翻译、字典和本地化内容）的元数据软件包被称之为 Langpacks。

在 Red Hat Enterprise Linux 8 系统中，langpacks 安装是基于langpacks-<langcode> 语言元软件包和 RPM 弱依赖项(Supplements 标签)。

对于所选语言，有两个先决条件可以使用 langpacks。如果满足这些先决条件，则语言 meta-packages 会在事务集中自动拉取所选语言的 langpack。

先决条件

- 系统中已安装了所选语言的 langpacks-<langcode> 语言 meta-package。

  在 Red Hat Enterprise Linux 8 中，langpacks 元软件包使用 Anaconda 安装程序，通过操作系统的初始安装自动安装的，因为这些软件包在 Application Stream 存储库中提供。

  如需更多信息，请参阅检查提供 langpacks 的语言。

- 您用来搜索区域设置软件包的基本软件包已安装在系统上。

16.1. 检查提供 LANGPACKS 的语言

按照以下步骤检查哪些语言提供语言包。

流程

- 执行以下命令：

```
# yum list langpacks-*
```

16.2. 使用 RPM 较弱依赖项的 LANGPACKS

本节介绍了在查询基于 RPM 弱依赖项的语言、安装或删除语言支持时可能需要执行的多个操作。

16.2.1. 列出已安装的语言支持

要列出已安装的语言支持，请使用此流程。

流程

- 执行以下命令：

```
# yum list installed langpacks*
```

16.2.2. 检查语言支持的可用性

要检查语言支持是否有适用于任何语言，请使用以下步骤。

流程

- 执行以下命令：

```
# yum list available langpacks*
```

16.2.3. 列出为语言安装的软件包

要列出为任何语言安装的软件包，请使用以下步骤：

流程

- 执行以下命令：

```
# yum repoquery --whatsupplements langpacks-<locale_code>
```

16.2.4. 安装语言支持

要添加新的语言支持，请使用以下步骤。

流程

- 执行以下命令：

```
# yum install langpacks-<locale_code>
```

16.2.5. 删除语言支持

要删除任何安装的语言支持，请使用以下步骤。

流程

- 执行以下命令：

```
# yum remove langpacks-<locale_code>
```

16.3. 使用 GLIBC-LANGPACK-<LOCALE_CODE> 保存磁盘空间

目前，所有区域都存储在 /usr/lib/locale/locale-archive 文件中，该文件需要很多磁盘空间。

在磁盘空间是一个关键问题的系统中，如容器和云镜像，或者只需要几个区域，您可以使用 glibc 语言 langpack 软件包 (glibc-langpack-<locale_code>)。

要单独安装区域设置，从而获得较小的软件包安装空间，请使用以下步骤。

流程

- 执行以下命令：

```
# yum install glibc-langpack-<locale_code>
```

当使用 Anaconda 安装操作系统时，会在安装过程中使用的语言安装 glibc-langpack-<locale_code>，并用于您选择的语言。请注意, glibc-all-langpacks （包含所有区域设置）会被默认安装，因此某些区域设置会被重复。如果您为一个或多个选择的语言安装 glibc-langpack-<locale_code>，您可以在安装后删除 glibc-all-langpacks 来保存磁盘空间。

请注意，只安装所选 glibc-langpack-<locale_code> 软件包而不是 glibc-all-langpacks 会对运行时性能造成影响。

注意

如果磁盘空间不是问题，请使用 glibc-all-langpacks 软件包安装所有区域。

第 17 章 转储崩溃的内核以便稍后进行分析

要分析系统崩溃的原因，可以使用 kdump 服务保存系统内存内容，以便稍后进行分析。本节概述了 kdump 以及使用 RHEL web 控制台或使用对应的 RHEL 系统角色配置 kdump 的信息。

17.1. KDUMP

kdump 是一个提供崩溃转储机制，并生成转储文件（称为崩溃转储或 vmcore 文件）的服务。vmcore 文件包含系统内存的内容，帮助分析和故障排除。kdump 使用 kexec 系统调用引导到第二个内核，这是一个没有重启的 *捕获内核*，然后捕获崩溃内核内存的内容，并将其保存到文件中。第二个内核在系统内存的保留部分中提供。

重要

当系统出现故障时，内核崩溃转储可能是唯一可用的信息。因此，在关键任务环境中操作 kdump 是非常重要的。红帽建议在常规内核更新周期中定期更新和测试 kexec-tools。这在安装新内核功能时尤为重要。

您可以为机器上所有安装的内核或者只为指定的内核启用 kdump。当在机器上使用多个内核时，这非常有用，有些内核足够稳定，不必担心它们会崩溃。安装 kdump 时，会创建一个默认的 /etc/kdump.conf 文件。/etc/kdump.conf 文件包含默认的最小 kdump 配置，您可以编辑该文件来自定义 kdump 配置。

17.2. 在 WEB 控制台中配置 KDUMP 内存使用率和目标位置

您可以为 kdump 内核配置内存保留，也可以指定目标位置来使用 RHEL web 控制台界面捕获 vmcore 转储文件。

先决条件

- 必须安装并可以访问 Web 控制台。

  详情请参阅安装 Web 控制台。

步骤

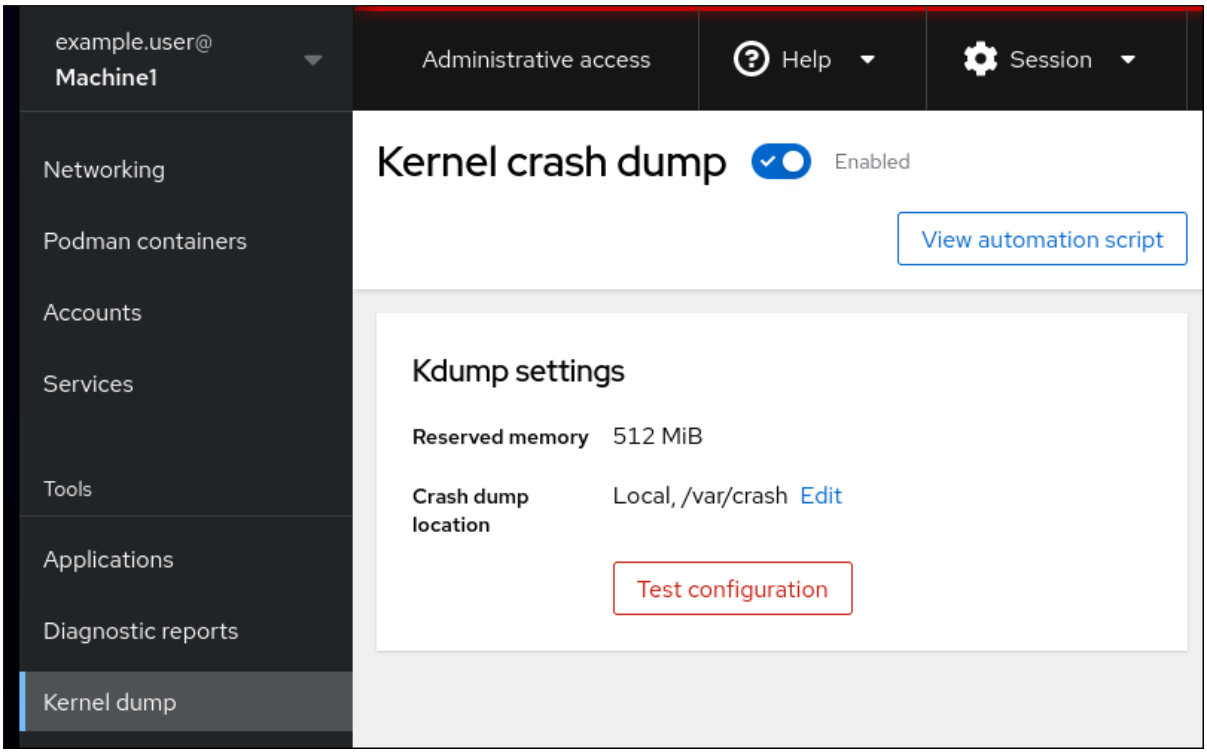
1. 在 web 控制台中，打开 Kernel dump 选项卡，并通过将 Kernel crash dump 开关设置为 on 来启动 kdump 服务。

2. **在终端中配置 kdump 内存用量，例如：**

```
$ sudo grubby --update-kernel ALL --args crashkernel=512M
```

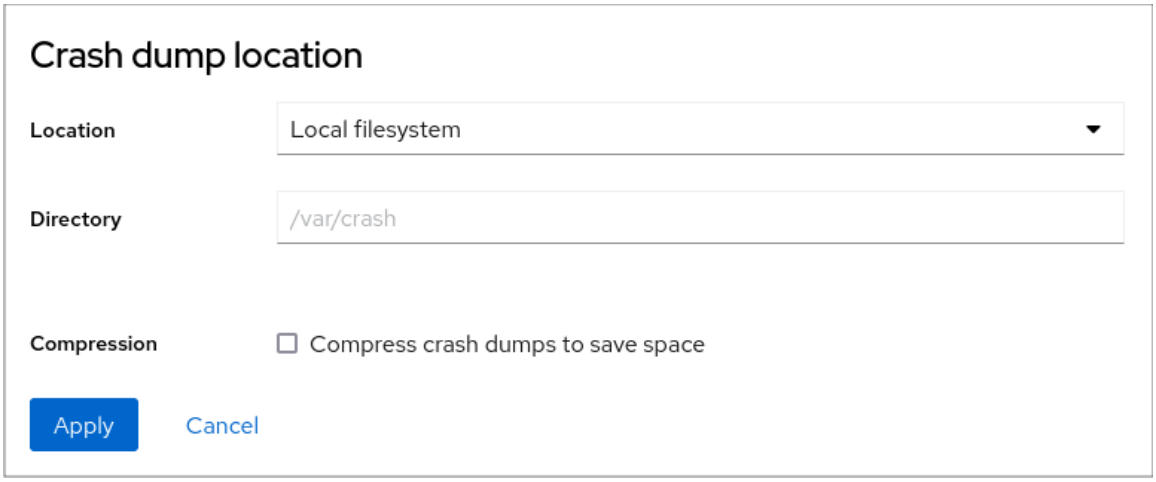
重启系统以应用更改。

3. **在 Kernel dump 选项卡中，点 Crash dump location 字段末尾的 Edit。**

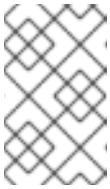


4. **指定保存 vmcore 转储文件的目标目录：**

- **对于本地文件系统，从下拉菜单中选择 Local Filesystem。**



- 对于使用 SSH 协议的远程系统，从下拉菜单中选择 Remote over SSH，并指定以下字段：
  - 在 Server 字段中，输入远程服务器地址。
  - 在 SSH key 字段中，输入 SSH 密钥位置。
  - 在 Directory 字段中，输入目标目录。
- 对于使用 NFS 协议的远程系统，从下拉菜单中选择 Remote over NFS，并指定以下字段：
  - 在 Server 字段中，输入远程服务器地址。
  - 在 Export 字段中，输入 NFS 服务器的共享文件夹的位置。
  - 在 Directory 字段中，输入目标目录。



注意

您可以选择 Compression 复选框来减小 vmcore 文件的大小。

5. 可选：点 View automation script 显示自动化脚本。

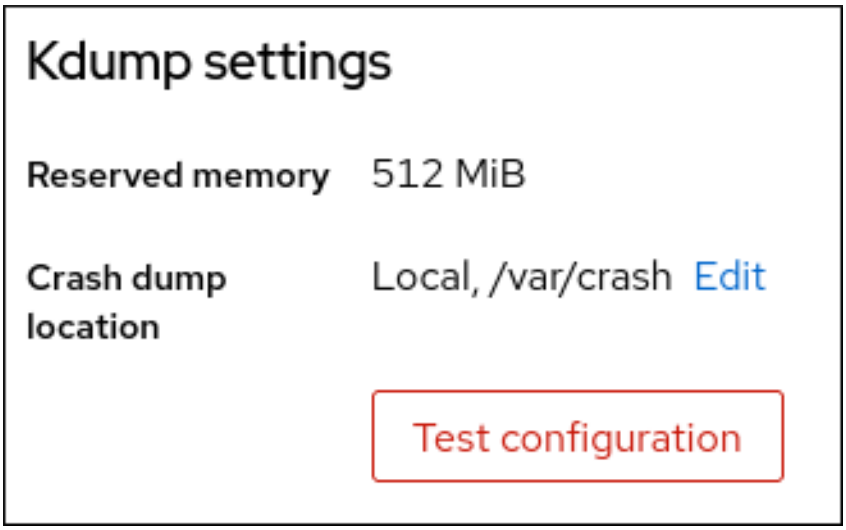
此时会打开一个带有生成的脚本的窗口。您可以在 shell 脚本页和 Ansible playbook 生成选项页之间转换。

6. 可选：点 Copy to clipboard 复制脚本。

您可以使用此脚本在多台机器上应用相同的配置。

验证

1. 单击 **Test configuration**。



2. 在 **Test kdump settings** 下点 **Crash system**。

警告

当您启动系统崩溃时，内核操作会停止，并导致系统因数据丢失而崩溃。

其他资源

- 支持的 kdump 目标

17.3. 使用 RHEL 系统角色进行 KDUMP

RHEL 系统角色是 Ansible 角色和模块的一个集合，其提供了一个一致的配置接口，来远程管理多个 RHEL 系统。kdump 角色可让您在多个系统中设置基本内核转储参数。

警告

通过替换 /etc/kdump.conf 文件，kdump 角色完全取代了受管主机的 kdump 配置。另外,如果应用了 kdump 角色，则之前的所有 kdump 设置也会被替换，即使它们没有被角色变量指定，也可以替换 /etc/sysconfig/kdump 文件。

以下示例 playbook 演示了如何应用 kdump 系统角色来设置崩溃转储文件的位置：

```
---
- hosts: kdump-test
  vars:
    kdump_path: /var/crash
  roles:
    - rhel-system-roles.kdump
```

有关 kdump 角色变量的详情，请安装 rhel-system-roles 软件包，并参阅 /usr/share/doc/rhel-system-roles/kdump 目录中的 README.md 或者README.html 文件。

其他资源

- RHEL 系统角色简介

17.4. 其他资源

- 安装 kdump
- 在命令行中配置 kdump
- 在 web 控制台中配置 kdump

第 18 章 恢复系统

要使用现有备份恢复系统，Red Hat Enterprise Linux 提供了 Relax-and-Recover(ReaR)工具。

您可以使用这个工具作为灾难恢复解决方案，也用于系统迁移。

该工具可让您执行以下任务：

- 生成可引导镜像，并使用镜像从现有备份中恢复系统。
- 复制原始存储布局。
- 恢复用户和系统文件。
- 将系统还原到不同的硬件中。

另外，对于灾难恢复，您还可以将某些备份软件与 ReaR 集成。

设置 ReaR 涉及以下高级别的操作：

1. 安装 ReaR。
2. 修改 ReaR 配置文件以添加备份方法详情。
3. 创建救援系统。
4. 生成备份文件。

18.1. 设置 REAR

使用以下步骤，使用 Relax-and-Recover(ReaR)工具安装软件包，来创建一个救援系统，配置并生成一个备份。

先决条件

- 根据备份恢复计划完成必要的配置。

  请注意： 您可以使用 NETFS 备份方法，该方法是 ReaR 完全整合的、内置的方法。

流程

1. 运行以下命令来安装 ReaR 工具：

```
# yum install rear
```

2. 在您选择的编辑器中修改 ReaR 配置文件，例如：

```
# vi /etc/rear/local.conf
```

3. 在 /etc/rear/local.conf 中添加备份设置详情。例如，在使用 NETFS 备份方法时添加以下行：

```
BACKUP=NETFS
BACKUP_URL=backup.location
```

   使用备份位置的 URL 替换 ***backup.location***。

4. 要将 ReaR 配置为在创建新归档时保留之前的备份归档，还需要将以下行添加到配置文件中：

```
NETFS_KEEP_OLD_BACKUP_COPY=y
```

5. 要让递增形式进行备份，在每个运行中只备份修改了的文件，添加以下行：

```
BACKUP_TYPE=incremental
```

6. 创建一个救援系统：

```
# rear mkrescue
```

7. 根据恢复计划进行备份。例如，在使用 NETFS 备份方法时运行以下命令：

```
# rear mkbackuponly
```

另外，您可以通过运行以下命令来在一个步骤中创建救援系统和备份：

```
# rear mkbackup
```

该命令将 rear mkrescue 和 rear mkbackuponly 的功能组合在一起。

18.2. 在 64 位 IBM Z 构架上使用 REAR 救援镜像

现在，基本的 Relax 和 Recover(ReaR)功能在 64 位 IBM Z 构架上作为技术预览提供。您只能在 z/VM 环境中的 IBM Z 上创建 ReaR 救援镜像。备份和恢复逻辑分区(LPAR)还没有进行测试。

重要

rear 软件包版本 2.6-9.el8 或更高版本仅支持 64 位 IBM Z 架构上的 ReaR。早期版本仅作为技术预览功能提供。有关红帽技术预览功能支持范围的更多信息，请参阅 https://access.redhat.com/support/offerings/techpreview。

当前唯一可用的输出方法是 Initial Program Load(IPL)。IPL 生成一个内核和一个初始 RAM 磁盘(initrd)，可与 zIPL 引导装载程序一起使用。

先决条件

- ReaR 已安装。
  - 要安装 ReaR，请运行 yum install rear 命令

流程

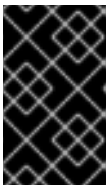
将以下变量添加到 /etc/rear/local.conf 中来配置 ReaR，以便在 64 位 IBM Z 构架上生成救援镜像：

1. 要配置 IPL 输出方法，请添加 OUTPUT=IPL。

2. 要配置备份方法和目的地，请添加 BACKUP 和 BACKUP_URL 变量。例如：

```
BACKUP=NETFS

BACKUP_URL=nfs://<nfsserver name>/<share path>
```



重要

目前在 64 位 IBM Z 构架上不支持本地备份存储。

3. 另外，您还可以配置 OUTPUT_URL 变量来保存内核和 initrd 文件。默认情况下，OUTPUT_URL 与 BACKUP_URL 保持一致。

4. 要执行备份和救援镜像创建：

```
# rear mkbackup
```

5. 这会在 BACKUP_URL 或 OUTPUT_URL（如果设置了）变量指定的位置创建内核和 initrd 文件，并使用指定的备份方法进行备份。

6. 要恢复系统，请使用第 3 步中创建的 ReaR 内核和 initrd 文件，并从与 zipl 引导装载程序、内核和 initrd 一起准备的直接附加存储设备(DASD)或者附加光纤通道协议(FCP)的 SCSI 设备启动。如需更多信息，请参阅使用准备的 DASD。

7. 当救援内核和 initrd 引导时，它会启动 ReaR 救援环境。继续系统恢复。

警告

目前，救援过程会重新格式化连接到系统的所有 DASD（直接附加存储设备）。如果系统存储设备中存在任何有价值的数据，则不要尝试系统恢复。这还包括用 zipl 引导装载程序、ReaR 内核和 initrd 准备的设备，用来引导到救援环境。确保保留一份副本。

其他资源

- 在 z/VM 中安装
- 使用一个准备的 DASD

第 19 章 安装和使用动态编程语言

红帽提供了不同的编程语言，如 Python、PHP 和 Tcl/TK。使用他们来开发自己的应用程序和服务。

19.1. PYTHON 简介

Python 是一种高级编程语言，支持多种编程模式，如面向对象、所需功能以及程序式范式。Python 具有动态语义，可用于通用编程。

使用 Red Hat Enterprise Linux 时，系统中安装的许多软件包（如提供系统工具、数据分析工具或 Web 应用程序的软件包）会使用 Python 编写。要使用这些软件包，您必须安装 python* 软件包。

19.1.1. Python 版本

两个不兼容的 Python 版本被广泛使用，即 Python 2.x 和 Python 3.x。RHEL 8 提供以下 Python 版本。

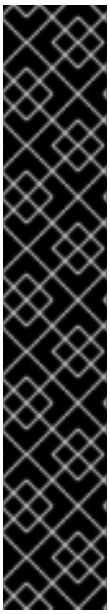
表 19.1. RHEL 8 中的 Python 版本

版本	要安装的软件包	命令示例	此后提供	生命周期
Python 3.6	**python3**, **python36**	**python3**, **python3.6**, **pip3**, **pip3.6**	RHEL 8.0	完整的 RHEL 8
Python 2.7	**python2**	**python2**, **pip2**	RHEL 8.0	较短
Python 3.8	**python38**	**python3.8**, **pip3.8**	RHEL 8.2	较短
Python 3.9	**python39**	**python3.9**,**pip3.9**	RHEL 8.4	较短
Python 3.11	**python3.11**	**python3.11**, **pip3.11**	RHEL 8.8	较短
Python 3.12	**python3.12**	**python3.12**, **pip3.12**	RHEL 8.10	较短

有关支持长度的详情，请查看 Red Hat Enterprise Linux 生命周期 和 Red Hat Enterprise Linux 应用程序流生命周期。

3.9 之前的每个 Python 版本都在一个单独的模块中分发。Python 3.11 和 Python 3.12 作为非模块化 RPM 软件包的套件发布，包括 python3.11 和 python3.12 软件包。

您可以在同一 RHEL 8 系统上并行安装多个 Python 版本。



重要

在安装、调用或与之交互时，始终指定 Python 版本。例如，在软件包和命令名称中使用 python3 而不是 python。所有与 Python 相关的命令还必须包括版本，例如 pip3,pip2,pip3.8,pip3.9,pip3.11, pip3.12。

RHEL 8 中默认不提供未指定版本的 python 命令(/usr/bin/python)。您可以使用 alternatives 命令来配置它；有关说明，请参阅 配置未版本化的 Python

任何对 /usr/bin/python 的手动更改（除使用 alternatives 命令所做的更改外），在更新时可能会覆盖。

作为系统管理员，使用 Python 3 的原因如下：

- Python 3 代表 Python 项目的主要开发方向。
- 2020 年终止了对上游社区中的 Python 2 的支持。
- 流行的 Python 库在上游社区支持 Python 2 支持。
- Red Hat Enterprise Linux 8 中的 Python 2 的生命周期比较短，旨在方便客户过渡到 Python 3。

对于开发人员，Python 3 与 Python 2 相比有以下优点：

- Python 3 可让您更轻松地编写表达、可维护且正确的代码。
- 使用 Python 3 编写的代码将具有更大的灵活性。

- Python 3 具有新功能，包括 asyncio、f-strings、高级解包、仅关键字的参数和链的例外。

但是，传统的软件可能需要将 /usr/bin/python 配置为 Python 2。因此，没有与 Red Hat Enterprise Linux 8 一起分发的默认的 python 软件包，您可以选择使用 Python 2 和 3 来作为 /usr/bin/python，如 配置未版本化的 Python 中所述。

重要

Red Hat Enterprise Linux 8 中的系统工具使用由内部 platform-python 软件包提供的 Python 版本 3.6，它不适用于客户直接使用。对于 Python 3.6，建议您使用 python36 软件包中的 python3 或 python3.6 命令，或使用后续的 Python 版本。

不要从 RHEL 8 中删除 platform-python 软件包，因为其他软件包需要它。

19.1.2. Python 版本之间的显著区别

RHEL 8 中包含的 Python 版本在许多方面有所不同。

Python 绑定

python38 和 python39 模块以及 python3.11 和 python3.12 软件包套件不包括与为 python36 模块提供的系统工具(RPM、DNF、SELinux 等)相同的绑定。因此，在需要与基础操作系统或二进制兼容性的情况下，使用 python36。在将系统绑定与各种 Python 模块的更新版本一起需要绑定的唯一实例中，请使用 python36 模块和通过 pip 安装到 Python 的 venv 或 virtualenv 环境中安装的第三方上游 Python 模块。

Python 3.11 和 Python 3.12 虚拟环境必须使用 venv 而不是 virtualenv创建

RHEL 8 中的 virtualenv 工具由 python3-virtualenv 软件包提供，与 Python 3.11 和 Python 3.12 不兼容。尝试使用 virtualenv 创建虚拟环境将失败，并显示出错信息，例如：

```
$ 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.
```

要创建 Python 3.11 或 Python 3.12 虚拟环境，请使用 python3.11 -m venv 或 python3.12 -m venv 命令，该命令使用标准库的 venv 模块。

19.2. 安装和使用 PYTHON

在 Red Hat Enterprise Linux 8 中，Python 3 在 3.6、3.8 和 3.9 版本中分发，由 python36、python38 和 python39 模块提供，以及 AppStream 存储库中的 python3.11 和 python3.12 软件包套件。

警告

默认情况下，使用未指定版本的 python 命令安装或运行 Python 无法正常工作，因为不确定。始终指定 Python 的版本，或使用 alternatives 命令配置系统默认版本。

19.2.1. 安装 Python 3

按照设计，您可以并行安装 RHEL 8 模块，包括 python27、python36、python38 和 python39 模块，以及 python3.11 和 python3.12 软件包套件。

您可以在同一个系统上与 Python 3.6 并行安装 Python 3.8、Python 3.9、Python 3.11 和 Python 3.12，包括为每个版本构建的软件包，与 Python 3.6 并行安装，但 mod_wsgi 模块除外。由于 Apache HTTP 服务器的限制，只能安装 python3-mod_wsgi、python38-mod_wsgi、python39-mod_wsgi、python3.11-mod_wsgi 或 python3.12-mod_wsgi 软件包。

流程

- 要从 python36 模块安装 Python 3.6，请使用：

```
# yum install python3
```

python36:3.6 模块流会自动启用。

- 要从 python38 模块安装 Python 3.8，请使用：

```
# yum install python38
```

python38:3.8 模块流会自动启用。

- 要从 python39 模块安装 Python 3.9，请使用：

```
# yum install python39
```

python39:3.9 模块流会自动启用。

- 要从 python3.11 RPM 软件包安装 Python 3.11，请使用：

```
# yum install python3.11
```

- 要从 python3.12 RPM 软件包安装 Python 3.12，请使用：

```
# yum install python3.12
```

验证步骤

- 要验证您系统中安装的 Python 版本，请使用 --version 选项以及特定于您所需版本的 Python 命令的 python 命令。
  - 对于 Python 3.6：

```
$ python3 --version
```

  - 对于 Python 3.8：

```
$ python3.8 --version
```

  - 对于 Python 3.9:

```
$ python3.9 --version
```

  - 对于 Python 3.11：

```
$ python3.11 --version
```

- 对于 Python 3.12：

```
$ python3.12 --version
```

其他资源

- 安装、管理和删除用户空间组件

19.2.2. 安装其他 Python 3 软件包

Python 3.6 的附加模块的软件包通常使用 python3- 前缀，Python 3.8 的软件包包括 python38- 前缀，Python 3.9 的软件包包括 python39- 前缀，Python 3.11 的软件包包括 python3.11- 前缀，以及 Python 3.12 的软件包包括 python3.12- 前缀。安装其他 Python 软件包时始终包含前缀，如下例所示。

流程

- 要为 Python 3.6 安装 Requests 模块，请使用：

```
# yum install python3-requests
```

- 要为 Python 3.8 安装 Cython 扩展，请使用：

```
# yum install python38-Cython
```

- 要从 Python 3.9 安装 pip 软件包安装程序，请使用：

```
# yum install python39-pip
```

- 要从 Python 3.11 安装 pip 软件包安装程序，请使用：

```
# yum install python3.11-pip
```

- 要从 Python 3.12 安装 pip 软件包安装程序，请使用：

```
# yum install python3.12-pip
```

其他资源

- 有关 Python 附加组件模块的上游文档

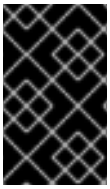
19.2.3. 为开发人员安装其他 Python 3 工具

开发人员的额外 Python 工具主要是通过相应 python38-devel 或 python39-devel 模块中的 CodeReady Linux Builder (CRB)存储库分发，或者 python3.11114 或 python3.12jpeg 软件包。

python3-pytest 软件包（用于 Python 3.6）及其依赖项在 AppStream 存储库中提供。

CRB 存储库提供：

- python38-devel 模块，其包含 python38-pytest 软件包及其依赖项。
- python39-devel 模块，其包含 python39-pytest 软件包及其依赖项，以及 python39-debug 和 python39-Cython 软件包。
- python3.11-* 软件包，其包括：
  - python3.11-pytest 及其依赖项
  - python3.11-idle
  - python3.11-debug
  - python3.11-Cython
- python3.12jpeg 软件包，其中包括一组类似的软件包，作为 python3.11jpeg。



重要

红帽不支持 CodeReady Linux Builder 存储库中的内容。

注意

并非所有与上游 Python 相关的软件包都包括在 RHEL 中。

要安装 ppython3*-pytest 软件包，请使用以下流程。

流程

1. 对于 Python 3.8 及之后的版本，请启用 CodeReady Linux Builder 存储库：

```
# subscription-manager repos --enable codeready-builder-for-rhel-8-x86_64-rpms
```

2. 对于 Python 3.8 或 3.9，启用相应的 python3*-devel 模块，例如：

```
# yum module enable python39-devel
```

3. 安装 python3*-pytest 软件包：

- 对于 Python 3.6：

```
# yum install python3-pytest
```

- 对于 Python 3.8：

```
# yum install python38-pytest
```

- 对于 Python 3.9:

```
# yum install python39-pytest
```

- 对于 Python 3.11：

```
# yum install python3.11-pytest
```

- 对于 Python 3.12：

```
# yum install python3.12-pytest
```

其他资源

- 如何在 CodeReady Linux Builder 中启用和使用内容
- 软件包清单

19.2.4. 安装 Python 2

有些应用程序和脚本还没有完全移植到 Python 3，并需要 Python 2 运行。Red Hat Enterprise Linux 8 允许并行安装 Python 3 和 Python 2。如果您需要 Python 2 功能，请安装 python27 模块，该模块可在 AppStream 存储库中获得。

警告

请注意，Python 3 是 Python 项目的主要开发方向。对 Python 2 的支持正在分阶段阶段。python27 模块的支持周期比 Red Hat Enterprise Linux 8 短。

流程

- 要从 python27 模块安装 Python 2.7，请使用：

```
# yum install python2
```

python27:2.7 模块流会自动启用。

Python 2 附加模块的软件包通常使用 python2- 前缀。安装其他 Python 软件包时始终包含前缀，如下例所示。

- 要为 Python 2 安装 Requests 模块，请使用：

```
# yum install python2-requests
```

- 要为 Python 2 安装 Cython 扩展，请使用：

```
# yum install python2-Cython
```

验证步骤

- 要验证安装在您的系统中的 Python 版本，请使用：

```
$ python2 --version
```

注意

按照设计，您可以并行安装 RHEL 8 模块，包括 python27、python36、python38 和 python39 模块。

其他资源

- 在 RHEL 8 中安装、管理和删除用户空间组件

19.2.5. 从 Python 2 迁移到 Python 3

作为开发者，您可能想要将之前使用 Python 2 编写的代码迁移到 Python 3。

有关如何将大型代码库迁移到 Python 3 的更多信息，请参阅 保守 Python 3 移植指南。

请注意,在迁移后,原始 Python 2 代码就可以被 Python 3 解释器解释，也可以被 Python 2 解释器解析。

19.2.6. 使用 Python

在运行 Python 解释器或 Python 相关的命令时，请始终指定版本。

先决条件

- 确定安装了所需的 Python 版本。
- 如果要为 Python 3.11 或 Python 3.12 下载并安装第三方应用程序，请安装 python3.11-pip 或 python3.12-pip 软件包。

流程

- 要运行 Python 3.6 解释器或相关命令，请使用：

```
$ python3
$ python3 -m venv --help
$ python3 -m pip install package
$ pip3 install package
```

- 要运行 Python 3.8 解释器或相关命令，请使用：

```
$ python3.8
$ python3.8 -m venv --help
$ python3.8 -m pip install package
$ pip3.8 install package
```

- 要运行 Python 3.9 解释器或相关命令，请使用：

```
$ python3.9
$ python3.9 -m venv --help
$ python3.9 -m pip install package
$ pip3.9 install package
```

- 要运行 Python 3.11 解释器或相关命令，请使用：

```
$ python3.11
$ python3.11 -m venv --help
$ python3.11 -m pip install package
$ pip3.11 install package
```

- 要运行 Python 3.12 解释器或相关命令，请使用：

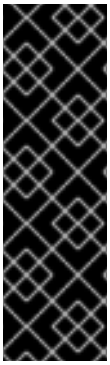
```
$ python3.12
$ python3.12 -m venv --help
$ python3.12 -m pip install package
$ pip3.12 install package
```

- 要运行 Python 2 解释器或相关命令，请使用：

```
$ python2
$ python2 -m pip install package
$ pip2 install package
```

19.3. 配置未指定版本的 PYTHON

系统管理员可以使用 alternatives 命令配置位于 /usr/bin/python 的被指定版本的 python 命令。请注意，在将未指定版本的命令配置为对应的版本之前，必须安装所需的软件包 python3、python38、python39、python3.11、python3.12 或 python2。



重要

/usr/bin/python 执行文件由 alternatives 系统控制。更新时可能会覆盖任何手动更改。

其他 Python 相关命令（如 pip3） 没有可配置的未指定版本的变体。

19.3.1. 直接配置未指定版本的 python 命令

您可以将未指定版本的 python 命令直接配置为所选 Python 版本。

先决条件

- 确定安装了所需的 Python 版本。

流程

- 要将未指定版本的 python 命令配置为 Python 3.6，请使用：

-

```
# alternatives --set python /usr/bin/python3
```

- 要将未指定版本的 python 命令配置为 Python 3.8，请使用：

```
# alternatives --set python /usr/bin/python3.8
```

- 要将未指定版本的 python 命令配置为 Python 3.9，请使用：

```
# alternatives --set python /usr/bin/python3.9
```

- 要将未指定版本的 python 命令配置为 Python 3.11，请使用：

```
# alternatives --set python /usr/bin/python3.11
```

- 要将未版本化的 python 命令配置为 Python 3.12，请使用：

```
# alternatives --set python /usr/bin/python3.12
```

- 要将未指定版本的 python 命令配置为 Python 2，请使用：

```
# alternatives --set python /usr/bin/python2
```

19.3.2. 以互动方式将未指定版本的 python 命令配置为所需的 Python 版本

您可以以互动方式将未指定版本的 python 命令配置为所需的 Python 版本。

先决条件

- 确定安装了所需的 Python 版本。

流程

1. 要以互动方式配置未指定版本的 python 命令，请使用：

```
# alternatives --config python
```

2. 从提供的列表中选择所需版本。

3. 要重置此配置并删除未指定版本的 python 命令，请使用：

```
# alternatives --auto python
```

19.3.3. 其他资源

- alternatives (8) 和 unversioned-python (1) man page

19.4. 打包 PYTHON 3 RPM

大多数 Python 项目使用 Setuptools 进行打包，并在 setup.py 文件中定义软件包信息。有关 Setuptools 打包的更多信息，请参阅 Setuptools 文档。

您还可以将 Python 项目打包成 RPM 软件包，与 Setuptools 打包相比有以下优点：

- 在其他 RPM （即使非 Python）上依赖关系软件包的规格
- 加密签名

  通过加密签名，可以使用其余操作系统验证、集成和测试 RPM 软件包的内容。

19.4.1. Python 软件包的 spec 文件描述

spec 文件包含 rpmbuild 实用程序用于构建 RPM 的说明。这些指令包含在不同的部分。spec 文件有两个主要部分来定义：

- Preamble（包含一系列在 Body 中使用的元数据项）
- Body（包含指令的主要部分）

与非 Python RPM SPEC 文件相比，Python 项目的 RPM SPEC 文件有一些特定信息。最值得注意的是，Python 库的任何 RPM 软件包的名称必须始终包含确定版本的前缀，例如： Python 3.6 的 python 3、Python 3.8 的 python38、Python 3.9 的 python39、Python 3.11 的 python3.11 或 Python 3.12 的 python3.12。

其他具体信息显示在 python3-detox 软件包的以下 spec 文件示例 中。有关此类特定描述，请查看示例中的备注。

```
%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

modname 宏包含 Python 项目的名称。在这个示例中，它是 detox。

2

将 Python 项目打包成 RPM 时，python3 前缀始终需要添加到项目的原始名称中。此处的原始名称为 detox，RPM 名称为 python3-detox。

3

BuildRequires 指定了构建和测试此软件包所需的软件包。在 BuildRequires 中，始终包括提供构建 Python 软件包所需的工具： python36-devel 和 python3-setuptools。需要 python36-rpm-macros 软件包，以便具有 /usr/bin/python3 解释器指令的文件会自动改为 /usr/bin/python3.6。

4

每个 Python 软件包都需要其他软件包才能正常工作。这些软件包还需要在 spec 文件中指定。要指定依赖项，您可以使用 %python_enable_dependency_generator 宏自动使用 setup.py 文件中定义的依赖项。如果软件包的依赖软件包没有使用 Setuptools 指定，请在附加 Requires 指令中指定它们。

5

%py3_build 和 %py3_install 宏会分别运行 setup.py build 和 setup.py install 命令，使用附加参数来指定安装位置、要使用的解释器以及其他详情。

6

check 部分提供了一个宏，它运行正确的 Python 版本。%{__python3} 宏包含 Python 3 解释器的路径，例如 /usr/bin/python3。我们建议您使用宏而不是字面路径。

19.4.2. Python 3 RPM 的常见宏

在 spec 文件中，始终使用用于 Python 3 RPM 的 Macros 的 Macros 表中的宏，而不是硬编码其值。

在宏名称中，始终使用 python3 或 python2，而不是未指定版本的 python。在 SPEC 文件的 BuildRequires 部分中，将特定的 Python 3 版本配置为 python36-rpm-macros,python38-rpm-macros,python39-rpm-macros,python3.11-rpm-macros, 或 python3.12-rpm-macros。

表 19.2. Python 3 RPM 宏

Macro	常规定义	描述
%{__python3}	/usr/bin/python3	Python 3 解释器
%{python3_version}	3.6	Python 3 解释器的完整版本。
%{python3_sitelib}	/usr/lib/python3.6/site-packages	其中安装了纯 Python 模块。
%{python3_sitearch}	/usr/lib64/python3.6/site-packages	安装了包含特定于架构扩展的模块。
%py3_build		使用适合系统软件包的参数运行 **setup.py build** 命令。
%py3_install		使用适合系统软件包的参数运行 **setup.py install** 命令。

19.4.3. 自动为 Python RPM 提供

在打包 Python 项目时，请确保以下目录包含在生成的 RPM 中（如果这些目录存在）：

- **.dist-info**
- **.egg-info**
- **.egg-link**

从这些目录中，RPM 构建流程自动生成虚拟 pythonX.Ydist 提供，如 python3.6dist (detox)。这些虚拟提供由 %python_enable_dependency_generator 宏指定的软件包使用。

19.5. 在 PYTHON 脚本中处理解释器指令

在 Red Hat Enterprise Linux 8 中，可执行 Python 脚本应该使用解析程序指令（也称为 hashbangs

或 shebangs），至少指定主 Python 版本。例如：

```
#!/usr/bin/python3
#!/usr/bin/python3.6
#!/usr/bin/python3.8
#!/usr/bin/python3.9
#!/usr/bin/python3.11
#!/usr/bin/python3.12
#!/usr/bin/python2
```

在构建任何 RPM 软件包时，/usr/lib/rpm/redhat/brp-mangle-shebangs buildroot 策略 (BRP) 脚本会自动运行，并尝试在所有可执行文件中更正解释器指令。

当遇到带有模糊的解释解释器指令的 Python 脚本时，BRP 脚本会生成错误，例如：

```
#!/usr/bin/python
```

或者

```
#!/usr/bin/env python
```

19.5.1. 修改 Python 脚本中的解释器指令

修改 Python 脚本中的解释器指令，该指令在 RPM 构建时导致构建错误。

先决条件

- Python 脚本中的一些解释器指令会导致构建错误。

流程

要修改解释器指令，请完成以下任务之一：

- 从 platform-python-devel 软件包中应用 pathfix.py 脚本：

```
# pathfix.py -pn -i %{__python3} PATH ...
```

请注意，可以指定多个 *PATH*。如果 *PATH* 是一个目录，则 pathfix.py 会递归扫描与模式

^[a-zA-Z0-9_]+\.py$ 匹配的 Python 脚本，而不仅仅是具有模糊的解释器指令。在 %prep 部分或 %install 部分的末尾添加这个命令。

- 修改打包的 Python 脚本，以便它们符合预期格式。因此，pathfix.py 也可以在 RPM 构建过程之外使用。当在 RPM 构建之外运行 pathfix.py 时，请使用解释器指令的路径替换 %*{__python3}*，如 /usr/bin/python3。

如果打包的 Python 脚本需要 Python 3.6 以外的版本，请调整前面的命令以包含所需的版本。

19.5.2. 更改自定义软件包中的 /usr/bin/python3 解释器指令

默认情况下，/usr/bin/python3 格式的解释器指令被替换为指向 platform-python 软件包中的 Python 解释器指令，该指令用于 Red Hat Enterprise Linux 的系统工具。您可以更改自定义软件包中的 /usr/bin/python3 解释器指令，以指向从 AppStream 存储库安装的 Python 的特定版本。

流程

- 要为特定版本的 Python 构建软件包，请将相应 python 软件包的*python*-rpm-macros 子软件包添加到 spec 文件的 BuildRequires 部分。例如，对于 Python 3.6，请包含以下行：

```
BuildRequires: python36-rpm-macros
```

因此，自定义软件包中的 /usr/bin/python3 解释器指令会自动转换为 /usr/bin/python3.6。

注意

要防止 BRP 脚本检查和修改解释器指令，请使用以下 RPM 指令：

```
%undefine __brp_mangle_shebangs
```

19.6. 使用 PHP 脚本语言

超文本 Preprocessor(PHP)是主要用于服务器端脚本的通用脚本语言，可让您使用 Web 服务器运行 PHP 代码。

在 RHEL 8 中，PHP 脚本语言由 php 模块提供，该模块可在多个流（版本）中可用。

根据您的用例，您可以安装所选模块流的特定配置集：

- common - 使用 Web 服务器进行服务器端脚本的默认配置文件。它包括多个广泛使用的扩展。
- minimal - 此配置集只安装命令行界面以用于使用 PHP 编写脚本，而无需使用 Web 服务器。
- devel - 此配置集包含来自 common 配置集的软件包以及用于开发用途的其他软件包。

19.6.1. 安装 PHP 脚本语言

您可以安装 php 模块的所选版本。

流程

- 要使用默认配置集安装 php 模块流，请使用：

```
# yum module install php:stream
```

使用您要安装的 PHP 版本替换 *stream*。

例如，要安装 PHP 8.0：

```
# yum module install php:8.0
```

默认 common 配置集安装 php-fpm 软件包，并预配置 PHP 以用于 Apache HTTP 服务器或 nginx。

- 要安装 php 模块流的特定配置集，请使用：

```
# yum module install php:stream/profile
```

使用您要安装的 *配置集* 的名称替换 *stream*。

例如，要安装 PHP 8.0 以供不使用 Web 服务器：

```
# yum module install php:8.0/minimal
```

其他资源

- 如果要从 RHEL 8 中可用的 PHP 版本升级，请参阅切换到更新的流。
- 有关管理 RHEL 8 模块和流的更多信息，请参阅 安装、管理和删除用户空间组件。

19.6.2. 通过 Web 服务器使用 PHP 脚本语言

19.6.2.1. 在 Apache HTTP 服务器中使用 PHP

在 Red Hat Enterprise Linux 8 中，Apache HTTP 服务器 允许您将 PHP 作为 FastCGI 进程服务器运行。FastCGI Process Manager(FPM)是一种替代 PHP FastCGI 守护进程，它允许网站管理高负载。PHP 在 RHEL 8 中使用 FastCGI 流程管理器。

您可以使用 FastCGI 进程服务器运行 PHP 代码。

先决条件

- 在您的系统上安装 PHP 脚本语言。

  请参阅 安装 PHP 脚本语言。

流程

1. 安装 httpd 模块：

   ```
   # yum module install httpd:2.4
   ```

2. 启动 Apache HTTP 服务器 ：

   ```
   # systemctl start httpd
   ```

或者，如果 Apache HTTP 服务器已在您的系统中运行，请在安装 PHP 后重启 httpd 服务：

```
# systemctl restart httpd
```

3. 启动 php-fpm 服务：

```
# systemctl start php-fpm
```

4. 可选： 在引导时启用这两个服务：

```
# systemctl enable php-fpm httpd
```

5. 要获取有关 PHP 设置的信息，请在 /var/www/html/ 目录中创建带有以下内容的 index.php 文件：

```
# echo '<?php phpinfo(); ?>' > /var/www/html/index.php
```

6. 要运行 index.php 文件，请将浏览器指向：

```
http://<hostname>/
```

7. 可选：如果您有具体要求，请调整配置：

- /etc/httpd/conf/httpd.conf - 一般的 httpd 配置
- /etc/httpd/conf.d/php.conf - httpd特定 PHP 配置
- /usr/lib/systemd/system/httpd.service.d/php-fpm.conf - 默认情况下， php-fpm 服务会与 httpd 一起启动
- /etc/php-fpm.conf - FPM 主配置

- /etc/php-fpm.d/www.conf - 默认 www 池配置

例 19.1. 运行"Hello, World!"使用 Apache HTTP 服务器的 PHP 脚本

1. 在 /var/www/html/ 目录中为您的项目创建一个 hello 目录：

```
# mkdir hello
```

2. 在 /var/www/html/hello/ 目录中创建 hello.php 文件，其内容如下：

```
# <!DOCTYPE html>
<html>
<head>
<title>Hello, World! Page</title>
</head>
<body>
<?php
   echo 'Hello, World!';
?>
</body>
</html>
```

3. 启动 Apache HTTP 服务器 ：

```
# systemctl start httpd
```

4. 要运行 hello.php 文件，请将浏览器指向：

```
http://<hostname>/hello/hello.php
```

因此，会显示带有 "Hello, World!" 文本的网页。

其他资源

- 设置 Apache HTTP web 服务器

19.6.2.2. 使用带有 nginx web 服务器的 PHP

您可以通过 nginx web 服务器运行 PHP 代码。

先决条件

- 在您的系统上安装 PHP 脚本语言。

  请参阅 安装 PHP 脚本语言。

流程

1. 安装 nginx 模块流：

```
# yum module install nginx:stream
```

   使用要安装的 nginx 版本替换 *stream*。

   例如，要安装 nginx 版本 1.18: :

```
# yum module install nginx:1.18
```

2. 启动 nginx 服务器：

```
# systemctl start nginx
```

   或者，如果 nginx 服务器已在您的系统中运行，请在安装 PHP 后重启 nginx 服务：

```
# systemctl restart nginx
```

3. 启动 php-fpm 服务：

```
# systemctl start php-fpm
```

4. 可选： 在引导时启用这两个服务：

```
# systemctl enable php-fpm nginx
```

5. 要获取 PHP 设置的信息，请在 /usr/share/nginx/html/ 目录中使用以下内容创建 index.php 文件：

```
# echo '<?php phpinfo(); ?>' > /usr/share/nginx/html/index.php
```

6. 要运行 index.php 文件，请将浏览器指向：

```
http://<hostname>/
```

7. 可选：如果您有具体要求，请调整配置：

- /etc/nginx/nginx.conf - nginx 主配置
- /etc/nginx/conf.d/php-fpm.conf - FPM 配置 nginx
- /etc/php-fpm.conf - FPM 主配置
- /etc/php-fpm.d/www.conf - 默认 www 池配置

例 19.2. 运行"Hello, World!"使用 nginx 服务器的 PHP 脚本

1. 在 /usr/share/nginx/html/ 目录中为您的项目创建一个 hello 目录：

```
# mkdir hello
```

2. 在 /usr/share/nginx/html/hello/ 目录中创建一个包含以下内容的 hello.php 文件：

```
# <!DOCTYPE html>
<html>
<head>
<title>Hello, World! Page</title>
</head>
<body>
<?php
   echo 'Hello, World!';
```

```
?>
</body>
</html>
```

3. 启动 nginx 服务器：

```
# systemctl start nginx
```

4. 要运行 hello.php 文件，请将浏览器指向：

```
http://<hostname>/hello/hello.php
```

因此，会显示带有 "Hello, World!" 文本的网页。

其他资源

- 设置和配置 NGINX

19.6.3. 使用命令行界面运行 PHP 脚本

PHP 脚本通常使用 Web 服务器运行，但也可以使用 命令行界面来运行。

如果只使用命令行运行 php 脚本，请安装 php 模块流的 minimal 配置集。

请参阅 安装 PHP 脚本语言。

先决条件

- 在您的系统上安装 PHP 脚本语言。

  请参阅 安装 PHP 脚本语言。

流程

1. 在文本编辑器中，创建一个 *filename*.php 文件

   将 *filename* 替换为您的文件名称。

2. 从命令行执行创建 *filename*.php 文件：

```
# php filename.php
```

例 19.3. 运行"Hello, World!"使用命令行界面 PHP 脚本

1. 使用文本编辑器，创建包含以下内容的 hello.php 文件：

```
<?php
  echo 'Hello, World!';
?>
```

2. 从命令行执行 hello.php 文件：

```
# php hello.php
```

   结果会输出 "Hello, World!"。

19.6.4. 其他资源

- httpd (8) - httpd 服务的手册页，包含其命令行选项的完整列表。
- httpd.conf(5) - httpd 配置的 man page，描述 httpd 配置文件的结构和位置。
- nginx(8) - nginx web 服务器的 man page，其中包含其命令行选项的完整列表和信号列表。
- php-fpm(8) - PHP FPM 的 man page 描述其命令行选项和配置文件的完整列表。

19.7. TCL/TK 入门

19.7.1. Tcl/Tk 简介

工具命令语言(Tcl) 是一种动态编程语言。这个语言的解释器和 C 库一同由 tcl 软件包提供。

一起使用 Tcl 和 Tk (Tcl/Tk) 启用创建跨平台 GUI 应用程序。TK 由 tk 软件包提供。

请注意，Tk 可以引用以下任意一种：

- 用于多种语言的编程工具包
- Tk C 库绑定可用于多种语言，如 C、Ruby、Perl 和 Python
- 一个需要解释器来实例化 Tk 控制台
- 为特定 Tcl 解释器添加多个新命令的 Tk 扩展

有关 Tcl/Tk 的更多信息，请参阅 Tcl/Tk manual 或 Tcl/Tk 文档网页。

19.7.2. Tcl/Tk 8.6 中的显著变化

Red Hat Enterprise Linux 7 使用 Tcl/Tk 8.5。在 Red Hat Enterprise Linux 8 中，Tcl/Tk 版本 8.6 在 Base OS 仓库中提供。

与 Tcl/Tk 8.5 相比，Tcl/Tk 8.6 的主要更改 有：

- 基于对象的编程支持
- 无堆栈评估实施
- 增强的例外处理

- 使用 Tcl 构建并安装的第三方软件包集合
- 启用多线程操作
- 对 SQL 数据库增强脚本的支持
- IPv6 网络支持
- 内置 Zlib 压缩
- 列表处理

  提供了两个新命令，即 lmap 和 dict map，它允许通过 Tcl 容器进行转换。
- 按脚本堆叠频道

  有两个新命令 chan push 和 chan pop 可用，允许向 I/O 频道添加或删除转换。

Tk 的主要更改包括：

- 内置 PNG 镜像支持
- 忙碌窗口

  新的命令 tk busy 可用，它禁用对窗口或小部件的用户交互，并显示忙碌的光标。
- 新的字体选择对话框接口
- Angled 文本支持

- 在 canvas 上移动支持

有关 Tcl 8.5 和 Tcl 8.6 之间的更改的详细列表，请参阅 Tcl/Tk 8.6 中的更改。

19.7.3. 迁移到 Tcl/Tk 8.6

Red Hat Enterprise Linux 7 使用 Tcl/Tk 8.5。在 Red Hat Enterprise Linux 8 中，Tcl/Tk 版本 8.6 在 Base OS 仓库中提供。

本节论述了到 Tcl/Tk 8.6 的迁移路径：

- 开发人员编写 Tcl 扩展或将 Tcl 解释器嵌入到其应用程序中
- 用户使用 Tcl/Tk编写任务

19.7.3.1. Tcl 扩展开发人员迁移路径

要使您的代码与 Tcl 8.6 兼容，请使用以下步骤。

流程

1. 重写代码以使用 interp 结构。例如，如果您的代码读取 interp→errorline，将其重写为使用以下功能：

```
Tcl_GetErrorLine(interp)
```

这是必要的，因为 Tcl 8.6 限制对 interp 结构成员的直接访问。

2. 要使您的代码与 Tcl 8.5 和 Tcl 8.6 兼容，请在包含 Tcl 库的头文件中使用以下代码片段：

```
# include <tcl.h>
# if !defined(Tcl_GetErrorLine)
# define Tcl_GetErrorLine(interp) (interp→errorLine)
# endif
```

19.7.3.2. 用户通过 Tcl/Tk 编写任务的迁移路径

在 Tcl 8.6 中，大多数脚本的工作方式与之前的 Tcl 版本相同。

要将代码迁移到 Tcl 8.6，请使用此流程。

流程

- 在编写可移植代码时，请确保不使用 Tk 8.6 中不再支持的命令：

```
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
```

  请注意，您还可以检查 /usr/share/tk8.6/unsupported.tcl 文件中不支持的命令列表。