

控制系统服务

培训目标

学完本节后，您应该能够通过 **systemctl** 控制系统守护进程和网络服务。

启动和停止服务

需要手动停止或启动服务的原因有很多：可能需要更新服务；可能需要更改配置文件；可能需要卸载服务；或者，管理员可能会手动启动不经常使用的服务。

要启动服务，首先通过 **systemctl status** 验证它是否未在运行。然后，以 **root** 用户身份（必要时使用 **sudo**）使用 **systemctl start** 命令。以下示例显示了如何启动 **sshd.service** 服务：

```
[root@host ~]# systemctl start sshd.service
```

如果命令中的服务名称缺少服务类型，**systemd** 服务会查找用于服务管理的 **.service** 文件。因此，上述命令可以执行为：

```
[root@host ~]# systemctl start sshd
```

要停止当前正在运行的服务，请使用 **stop** 参数来运行 **systemctl** 命令。以下示例显示了如何停止 **sshd.service** 服务：

```
[root@host ~]# systemctl stop sshd.service
```

重新启动和重新加载服务

在重新启动正在运行的服务期间，服务将停止然后启动。在重新启动服务时，进程 ID 会改变，并且在启动期间会关联新的进程 ID。要重新启动正在运行的服务，请使用 **restart** 参数来运行 **systemctl** 命令。以下示例演示了如何重新启动 **sshd.service** 服务：

```
[root@host ~]# systemctl restart sshd.service
```

某些服务可以重新加载其配置文件，而无需重新启动。这个过程称为服务重新加载。重新加载服务不会更改与各种服务进程关联的进程 ID。要重新加载正在运行的服务，请使用 **reload** 参数来运行 **systemctl** 命令。以下示例演示了如何在配置更改后重新加载 **sshd.service** 服务：

```
[root@host ~]# systemctl reload sshd.service
```

如果您不确定服务是否具有重新加载配置文件更改的功能，请使用 **reload-or-restart** 参数来运行 **systemctl** 命令。如果重新加载功能可用，该命令将重新加载配置更改。否则，该命令将重新启动服务以实施新的配置更改：

```
[root@host ~]# systemctl reload-or-restart sshd.service
```

列出单元依赖项

某些服务要求首先运行其他服务，从而创建对其他服务的依赖项。其他服务并不在系统引导时启动，而是仅在需要时启动。在这两种情况下，**systemd** 和 **systemctl** 根据需要启动服务，不论是解决依赖项，还是启动不经常使用的服务。例如，如果 CUPS 打印服务未在运行，并有文件被放入打印假脱机目录，则系统将启动 CUPS 相关的守护进程或命令来满足打印请求。

```
[root@host ~]# systemctl stop cups.service
Warning: Stopping cups, but it can still be activated by:
cups.path
cups.socket
```

要在系统上彻底停止打印服务，请停止所有三个单元。禁用服务将禁用其依赖项。

systemctl list-dependencies UNIT 命令显示启动服务单元所需的依赖项的层次结构映射。要列出反向依赖项（依赖于指定单元的单元），请使用 **--reverse** 选项来运行此命令。

```
[root@host ~]# systemctl list-dependencies sshd.service
sshd.service
● ├─system.slice
● ├─sshd-keygen.target
● | ├─sshd-keygen@ecdsa.service
● | ├─sshd-keygen@ed25519.service
● | └─sshd-keygen@rsa.service
● └─sysinit.target
...output omitted...
```

屏蔽未屏蔽的服务

有时，系统中安装的不同服务之间可能彼此冲突。例如，有多种方法可以管理邮件服务器（如 **postfix** 和 **sendmail** 等）。屏蔽服务可防止管理员意外启动与其他服务冲突的服务。屏蔽操作会在配置目录中创建指向 **/dev/null** 文件的链接，该文件可阻止服务启动。

```
[root@host ~]# systemctl mask sendmail.service
Created symlink /etc/systemd/system/sendmail.service → /dev/null.
```

```
[root@host ~]# systemctl list-unit-files --type=service
UNIT FILE                                     STATE
...output omitted...
sendmail.service                                masked
...output omitted...
```

尝试启动已屏蔽的服务单元会失败，并显示如下输出：

```
[root@host ~]# systemctl start sendmail.service
Failed to start sendmail.service: Unit sendmail.service is masked.
```

使用 **systemctl unmask** 命令可取消屏蔽服务单元。

```
[root@host ~]# systemctl unmask sendmail
Removed /etc/systemd/system/sendmail.service.
```

**重要**

禁用的服务可以手动启动，或通过其他单元文件启动，但不会在系统引导时自动启动。屏蔽的服务无法手动启动，也不会自动启动。

使服务在系统引导时启动或停止

在运行中的系统上启动一项服务不能确保该服务在系统重启时自动启动。类似地，在运行中的系统上停止一项服务也不能防止它在系统重启时再次启动。在 **systemd** 配置目录中创建链接，使服务在系统引导时启动。**systemctl** 命令可以创建和删除这些链接。

要在系统引导时启动服务，请使用 **systemctl enable** 命令。

```
[root@root ~]# systemctl enable sshd.service
Created symlink /etc/systemd/system/multi-user.target.wants/sshd.service → /usr/
lib/systemd/system/sshd.service.
```

以上命令会从服务单元文件（通常位于 **/usr/lib/systemd/system** 目录）创建一个符号链接，指向磁盘上供 **systemd** 寻找文件的位置，即 **/etc/systemd/
system/TARGETNAME.target.wants** 目录。启用服务不会在当前会话中启动该服务。要启动服务并使其在引导期间自动启动，请执行 **systemctl start** 和 **systemctl enable** 命令。

要禁用服务自动启动，请使用以下命令，该命令将删除在启用服务时创建的符号链接。请注意禁用服务不会停止该服务。

```
[root@host ~]# systemctl disable sshd.service
Removed /etc/systemd/system/multi-user.target.wants/sshd.service.
```

要验证服务是启用还是禁用状态，请使用 **systemctl is-enabled** 命令。

systemctl 命令摘要

可以在运行中的系统上启动和停止服务，也可启用或禁用服务在系统引导时自动启动。

服务管理实用命令

任务	命令
查看有关单元状态的详细信息。	systemctl status UNIT
在运行中的系统上停止一项服务。	systemctl stop UNIT
在运行中的系统上启动一项服务。	systemctl start UNIT
在运行中的系统上重新启动一项服务。	systemctl restart UNIT
重新加载运行中服务的配置文件。	systemctl reload UNIT

任务	命令
彻底禁用服务，使其无法手动启动或在系统引导时启动。	systemctl mask UNIT
使屏蔽的服务变为可用。	systemctl unmask UNIT
将服务配置为在系统引导时启动。	systemctl enable UNIT
禁止服务在系统引导时启动。	systemctl disable UNIT
列出指定单元需要的单元。	systemctl list-dependencies UNIT



参考文献

`systemd(1)`、`systemd.unit(5)`、`systemd.service(5)`、`systemd.socket(5)` 和 `systemctl(1)` man page

如需更多信息，请参阅《红帽企业 Linux 8.0 基本系统设置配置指南》中的使用 `systemd` 管理服务章节，网址为：

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_basic_system_settings/managing-services-with-systemd_configure-basic-system-settings#managing-system-services_managing-services-with-systemd

► 指导练习

控制系统服务

在本练习中，您将使用 **systemctl** 来停止、启动、重新启动、重新加载、启用和禁用受 systemd 管理的服务。

成果

您应能够使用 **systemctl** 命令来控制受 **systemd** 管理的服务。

在你开始之前

在 **workstation** 上，以 **student** 用户身份并使用密码 **student** 进行登录。

从 **workstation**，运行 **lab services-control start** 命令。该命令将运行一个起始脚本，它将确定主机 **servera** 是否可从网络访问。该脚本还确保 **sshd** 和 **chronyd** 服务正在 **servera** 上运行。

```
[student@workstation ~]$ lab services-control start
```

- 1. 使用 **ssh** 命令，以 **student** 用户身份登录 **servera**。系统已配置为使用 SSH 密钥来进行身份验证，因此不需要提供密码。

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- 2. 对 **sshd** 服务执行 **systemctl restart** 和 **systemctl reload** 命令。观察执行这些命令的不同结果。

2.1. 显示 **sshd** 服务的状态。记下 **sshd** 守护进程的进程 ID。

```
[student@servera ~]$ systemctl status sshd
● sshd.service - OpenSSH server daemon
  Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
  Active: active (running) since Wed 2019-02-06 23:50:42 EST; 9min ago
    Docs: man:sshd(8)
          man:sshd_config(5)
  Main PID: 759 (sshd)
     Tasks: 1 (limit: 11407)
    Memory: 5.9M
  ...output omitted...
```

按 **q** 键退出该命令。

2.2. 重新启动 **sshd** 服务，再查看其状态。守护进程的进程 ID 必定会改变。

```
[student@servera ~]$ sudo systemctl restart sshd
[sudo] password for student: student
[student@servera ~]$ systemctl status sshd
● sshd.service - OpenSSH server daemon
  Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
  Active: active (running) since Wed 2019-02-06 23:50:42 EST; 9min ago
    Docs: man:sshd(8)
          man:sshd_config(5)
  Main PID: 1132 (sshd)
    Tasks: 1 (limit: 11407)
   Memory: 5.9M
...output omitted...
```

在上面的输出中，请注意进程 ID 从 759 变为 1132（您系统上的数字可能会不同）。按 **q** 键退出该命令。

2.3. 重新加载 **sshd** 服务，再查看其状态。守护进程的进程 ID 必定不会改变，连接也不会中断。

```
[student@servera ~]$ sudo systemctl reload sshd
[student@servera ~]$ systemctl status sshd
● sshd.service - OpenSSH server daemon
  Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
  Active: active (running) since Wed 2019-02-06 23:50:42 EST; 9min ago
    Docs: man:sshd(8)
          man:sshd_config(5)
  Main PID: 1132 (sshd)
    Tasks: 1 (limit: 11407)
   Memory: 5.9M
...output omitted...
```

按 **q** 键退出该命令。

► 3. 验证 **chronyd** 服务正在运行。

```
[student@servera ~]$ systemctl status chronyd
● chronyd.service - NTP client/server
  Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled; vendor
  preset: enabled)
  Active: active (running) since Wed 2019-02-06 23:50:38 EST; 1h 25min ago
...output omitted...
```

按 **q** 键退出该命令。

► 4. 停止 **chronyd** 服务，再查看其状态。

```
[student@servera ~]$ sudo systemctl stop chronyd
[student@servera ~]$ systemctl status chronyd
● chronyd.service - NTP client/server
```

```
Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled; vendor
preset: enabled)
  Active: inactive (dead) since Thu 2019-02-07 01:20:34 EST; 44s ago
    ...output omitted...
... servera.lab.example.com chronyd[710]: System clock wrong by 1.349113 seconds,
adjustment started
... servera.lab.example.com systemd[1]: Stopping NTP client/server...
... servera.lab.example.com systemd[1]: Stopped NTP client/server.
```

按 **q** 键退出该命令。

► 5. 确定 chronyd 服务是否已启用为在系统引导时启动。

```
[student@server ~]$ systemctl is-enabled chronyd
enabled
```

► 6. 重新引导 servera，然后查看 chronyd 服务的状态。

```
[student@servera ~]$ sudo systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
[student@workstation ~]$
```

以 student 用户身份登录 servera，并查看 chronyd 服务的状态。

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ systemctl status chronyd
● chronyd.service - NTP client/server
  Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled; vendor
  preset: enabled)
    Active: active (running) since Thu 2019-02-07 01:48:26 EST; 5min ago
      ...output omitted...
```

按 **q** 键退出该命令。

► 7. 禁用 chronyd 服务，使其不在系统引导时启动，然后查看该服务的状态。

```
[student@servera ~]$ sudo systemctl disable chronyd
[sudo] password for student: student
Removed /etc/systemd/system/multi-user.target.wants/chronyd.service.
[student@servera ~]$ systemctl status chronyd
● chronyd.service - NTP client/server
  Loaded: loaded (/usr/lib/systemd/system/chronyd.service; disabled; vendor
  preset: enabled)
    Active: active (running) since Thu 2019-02-07 01:48:26 EST; 5min ago
      ...output omitted...
```

按 **q** 键退出该命令。

- 8. 重新引导 servera, 然后查看 chronyd 服务的状态。

```
[student@servera ~]$ sudo systemctl reboot  
Connection to servera closed by remote host.  
Connection to servera closed.  
[student@workstation ~]$
```

以 student 用户身份登录 servera, 并查看 chronyd 服务的状态。

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$ systemctl status chronyd  
● chronyd.service - NTP client/server  
  Loaded: loaded (/usr/lib/systemd/system/chronyd.service; disabled; vendor  
    preset: enabled)  
  Active: inactive (dead)  
    Docs: man:chronyd(8)  
          man:chrony.conf(5)
```

- 9. 从 servera 退出。

```
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation]$
```

完成

在 workstation 上, 运行 **lab services-control finish** 脚本来完成本练习。

```
[student@workstation ~]$ lab services-control finish
```

本引导式练习到此结束。

▶ 开放研究实验

控制服务和守护进程

任务执行清单

在本实验中，您将根据提供给您的规范配置要启用或禁用、运行或停止的几项服务。

成果

您应能够启用、禁用、启动和停止服务。

在你开始之前

在 workstation 上，以 student 用户身份并使用密码 student 进行登录。

从 workstation，运行 **lab services-review start** 命令。该命令将运行一个起始脚本，它将确定主机 serverb 是否可从网络访问。该脚本还将确保 serverb 上已适当配置了 psacct 和 rsyslog 服务。

```
[student@workstation ~]$ lab services-review start
```

1. 在 serverb 上，启动 psacct 服务。
2. 将 psacct 服务配置为在系统引导时启动。
3. 停止 rsyslog 服务。
4. 配置 rsyslog 服务，使其不在系统引导时启动。
5. 重启 serverb，再评估实验结果。

评估

在 workstation 上，运行 **lab services-review grade** 脚本来确认是否成功完成本实验。

```
[student@workstation ~]$ lab services-review grade
```

完成

在 workstation 上，运行 **lab services-review finish** 脚本来完成本实验。

```
[student@workstation ~]$ lab services-review finish
```

本实验到此结束。

► 解决方案

控制服务和守护进程

任务执行清单

在本实验中，您将根据提供给您的规范配置要启用或禁用、运行或停止的几项服务。

成果

您应能够启用、禁用、启动和停止服务。

在你开始之前

在 workstation 上，以 student 用户身份并使用密码 student 进行登录。

从 workstation，运行 **lab services-review start** 命令。该命令将运行一个起始脚本，它将确定主机 serverb 是否可从网络访问。该脚本还将确保 serverb 上已适当配置了 psacct 和 rsyslog 服务。

```
[student@workstation ~]$ lab services-review start
```

1. 在 serverb 上，启动 psacct 服务。

- 1.1. 使用 ssh 命令，以 student 用户身份登录 serverb。

```
[student@workstation ~]$ ssh student@serverb  
...output omitted...  
[student@serverb ~]$
```

- 1.2. 使用 systemctl 命令来验证 psacct 服务的状态。注意到 psacct 已被停止并已禁用在系统引导时启动。

```
[student@serverb ~]$ systemctl status psacct  
● psacct.service - Kernel process accounting  
  Loaded: loaded (/usr/lib/systemd/system/psacct.service; disabled; vendor  
  preset: disabled)  
  Active: inactive (dead)
```

- 1.3. 启动 psacct 服务。

```
[student@serverb ~]$ sudo systemctl start psacct  
[sudo] password for student: student  
[student@serverb ~]$
```

- 1.4. 验证 psacct 服务正在运行。

```
[student@serverb ~]$ systemctl is-active psacct  
active
```

2. 将 **psacct** 服务配置为在系统引导时启动。

2.1. 为 **psacct** 服务启用在系统引导时启动。

```
[student@serverb ~]$ sudo systemctl enable psacct  
Created symlink /etc/systemd/system/multi-user.target.wants/psacct.service → /usr/  
lib/systemd/system/psacct.service.
```

2.2. 验证 **psacct** 服务已启用在系统引导时启动。

```
[student@serverb ~]$ systemctl is-enabled psacct  
enabled
```

3. 停止 **rsyslog** 服务。

3.1. 使用 **systemctl** 命令来验证 **rsyslog** 服务的状态。注意到 **rsyslog** 服务正在运行，并已启用在系统引导时启动。

```
[student@serverb ~]$ systemctl status rsyslog  
● rsyslog.service - System Logging Service  
  Loaded: loaded (/usr/lib/systemd/system/rsyslog.service; enabled; vendor  
  preset: enabled)  
  Active: active (running) since Fri 2019-02-08 10:16:00 IST; 2h 34min ago  
    ...output omitted...  
按 q 键退出该命令。
```

3.2. 停止 **rsyslog** 服务。

```
[student@serverb ~]$ sudo systemctl stop rsyslog  
[sudo] password for student: student  
[student@serverb ~]$
```

3.3. 验证 **rsyslog** 服务已被停止。

```
[student@serverb ~]$ systemctl is-active rsyslog  
inactive
```

4. 配置 **rsylog** 服务，使其不在系统引导时启动。

4.1. 禁用 **rsylog** 服务，使其不在系统引导时启动。

```
[student@serverb ~]$ sudo systemctl disable rsyslog  
Removed /etc/systemd/system/syslog.service.  
Removed /etc/systemd/system/multi-user.target.wants/rsyslog.service.
```

4.2. 验证 **rsyslog** 已禁用在系统引导时启动。

```
[student@serverb ~]$ systemctl is-enabled rsyslog  
disabled
```

5. 重启 serverb，再评估实验结果。

```
[student@serverb ~]$ sudo systemctl reboot  
Connection to serverb closed by remote host.  
Connection to serverb closed.  
[student@workstation ~]$
```

评估

在 workstation 上，运行 **lab services-review grade** 脚本来确认是否成功完成本实验。

```
[student@workstation ~]$ lab services-review grade
```

完成

在 workstation 上，运行 **lab services-review finish** 脚本来完成本实验。

```
[student@workstation ~]$ lab services-review finish
```

总结

在本章中，您学到了：

- **systemd** 提供了一种方式，可在系统引导时以及运行中的系统上激活系统资源、服务器守护进程和其他进程。
- 使用 **systemctl** 可以启动、停止、重新加载、启用和禁用服务。
- 使用 **systemctl status** 命令可以确定 **systemd** 启动的系统守护进程和网络服务的状态。
- **systemctl list-dependencies** 命令可列出特定服务单元依赖的所有服务单元。
- **systemd** 可屏蔽服务单元，使其即便是为了满足依赖关系的需要也不会运行。

海量视频题库 myitpub.com QQ:5565462
www.52myit.com

章 10

配置和保护 SSH

目标

使用 OpenSSH 配置远程系统上的安全命令行服务。

培训目标

- 使用 **ssh** 登录远程系统并运行命令。
- 为用户帐户配置基于密钥的身份验证，以便其无需密码就能安全登录远程系统。
- 限制直接以 root 身份登录，并为 OpenSSH 服务禁用基于密码的身份验证。

章节

- 使用 SSH 访问远程命令行（及引导式练习）
- 配置基于 SSH 密钥的身份验证（及引导式练习）
- 自定义 OpenSSH 服务配置（及引导式练习）

实验

配置和保护 SSH

使用 SSH 访问远程命令行

培训目标

学完本节后，您应能够使用 **ssh** 登录 Linux 系统并运行命令。

什么是 OPENSSH？

OpenSSH 在红帽企业 Linux 系统上实施 Secure Shell 或 SSH 协议。SSH 协议使系统能够通过不安全的网络以加密和安全的方式进行通信。

您可以使用 **ssh** 命令来创建与远程系统的安全连接、以特定用户身份进行身份验证，并以该用户身份在远程系统上获取交互式 shell 会话。您也可以使用 **ssh** 命令在远程系统上运行单个命令，而不运行交互式 shell。

安全 SHELL 示例

以下的 **ssh** 命令将使用与当前本地用户相同的用户名登录远程服务器 **remotehost**。在本例中，远程系统会提示您使用该用户的密码进行身份验证。

```
[user01@host ~]$ ssh remotehost  
user01@remotehost's password: redhat  
...output omitted...  
[user01@remotehost ~]$
```

您可以使用 **exit** 命令来注销远程系统。

```
[user01@remotehost ~]$ exit  
logout  
Connection to remotehost closed.  
[user01@host ~]$
```

以下的 **ssh** 命令将使用用户名 **user02** 登录远程服务器 **remotehost**。同样，远程系统会提示您使用该用户的密码进行身份验证。

```
[user01@host ~]$ ssh user02@remotehost  
user02@remotehost's password: shadowman  
...output omitted...  
[user02@remotehost ~]$
```

此 **ssh** 命令将在 **remotehost** 远程系统上以 **user02** 用户身份运行 **hostname** 命令，而不访问远程交互式 shell。

```
[user01@host ~]$ ssh user02@remotehost hostname  
user02@remotehost's password: shadowman  
remotehost.lab.example.com  
[user01@host ~]$
```

请注意，上述命令在本地系统的终端中显示输出。

识别远程用户

w 命令可显示当前登录到计算机的用户列表。这对于显示哪些用户使用 **ssh** 从哪些远程位置进行了登录以及执行了何种操作等内容特别有用。

```
[user01@host ~]$ ssh user01@remotehost
user01@remotehost's password: redhat
[user01@remotehost ~]$ w
 16:13:38 up 36 min,  1 user,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM          LOGIN@    IDLE   JCPU   PCPU WHAT
user02    pts/0    172.25.250.10  16:13     7:30   0.01s  0.01s -bash
user01    pts/1    172.25.250.10  16:24     3.00s  0.01s  0.00s w
[user02@remotehost ~]$
```

前面的输出显示，**user02** 用户于今天 **16:13** 从 IP 地址为 **172.25.250.10** 的主机登录了伪终端 **0** 上的系统，并且在 shell 提示符下闲置了 7 分 30 秒。前面的输出也显示，**user01** 用户登录了伪终端 **1** 上的系统，并且自执行了 **w** 命令后三秒一直处于闲置状态。

SSH 主机密匙

SSH 通过公钥加密的方式保持通信安全。当某一 SSH 客户端连接到 SSH 服务器时，在该客户端登录之前，服务器会向其发送公钥副本。这可用于设置通信渠道的安全加密，并可验证客户端的服务器。

当用户使用 **ssh** 命令连接到 SSH 服务器时，该命令会检查它在本地已知主机文件中是否有该服务器的公钥副本。系统管理员可能在 **/etc/ssh/ssh_known_hosts** 中已进行预配置，或者用户的主目录中可能有一个包含公钥的 **~/.ssh/known_hosts** 文件。

如果客户端有公钥的副本，**ssh** 就会将该服务器已知主机文件中的公钥与它收到的公钥进行比较。如果公钥不匹配，客户端会假定服务器的网络流量已遭劫持或服务器已被入侵，并且请求用户确认是否要继续连接。



注意

在特定于用户的 **~/.ssh/config** 文件或系统范围的 **/etc/ssh/ssh_config** 中将 **StrictHostKeyChecking** 参数设为 **yes**，使得 **ssh** 命令在公钥不匹配时始终中断 SSH 连接。

如果客户端的已知主机文件中没有公钥的副本，**ssh** 命令会询问您是否仍要登录。如果仍进行登录，公钥的副本就会保存到您的 **~/.ssh/known_hosts** 文件中，以便将来能自动确认服务器的身份。

```
[user01@host ~]$ ssh newhost
The authenticity of host 'remotehost (172.25.250.12)' can't be established.
ECDSA key fingerprint is SHA256:qaS0PToLrqlC02XGk1A0iY7CaP7aPKimerDoaUkv720.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'newhost,172.25.250.12' (ECDSA) to the list of known
hosts.
user01@newhost's password: redhat
...output omitted...
[user01@newhost ~]$
```

SSH 已知主机公钥管理

如果由于硬盘驱动器故障而导致公钥丢失或由于某些正当理由而导致公钥被更换，并由此更改了服务器的公钥，您需要编辑已知的主机文件以确保将旧公钥条目替换为新公钥条目，从而确保登录时不会产生错误。

公钥存储在 **/etc/ssh/ssh_known_hosts** 及 SSH 客户端上每个用户的 **~/.ssh/known_hosts** 文件中。每个公钥各占一行。第一个字段是共享该公钥的主机名和 IP 地址的列表。第二个字段是公钥的加密算法。最后一个字段是公钥本身。

```
[user01@host ~]$ cat ~/.ssh/known_hosts
remotehost,172.25.250.11 ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmzdHAyNTYAAQABBOsEi0e+FlaNT6jul8Ag5Nj
+RViZ10yE2w6iYUr+1fPt0IF0EaOgFZ1LXM37VFTxdgFxHS3D5WhnIfb+68zf8+w=
```

您所连接的每个远程 SSH 服务器都将其公钥存储在 **/etc/ssh** 目录下扩展名为 **.pub** 的文件中。

```
[user01@remotehost ~]$ ls /etc/ssh/*key.pub
/etc/ssh/ssh_host_ecdsa_key.pub  /etc/ssh/ssh_host_ed25519_key.pub  /etc/ssh/
ssh_host_rsa_key.pub
```



注意

最好是添加一些将服务器的 **ssh_host_*key.pub** 文件与您的 **~/.ssh/known_hosts** 文件或系统范围内的 **/etc/ssh/ssh_known_hosts** 文件相匹配的条目。



参考文献

ssh(1)、**w(1)** 和 **hostname(1)** man page

如需更多信息，请参阅《红帽企业 Linux 8.0 配置和管理安全指南》中的 OpenSSH 章节，了解如何在两个系统之间使用安全通信，网址为：

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/securing_networks/assembly_using-secure-communications-with-openssh-securing-networks#The_SSH_protocol_configuring-and-managing-security

► 指导练习

访问远程命令行

在本练习中，您将作为不同的用户登录远程系统并执行命令。

成果

您应能够：

- 登录远程系统。
- 使用 OpenSSH 安全 shell 执行命令。

在你开始之前

以 student 用户身份并使用 student 作为密码登录 workstation。

在 workstation 上，运行 lab ssh-access start 脚本来开始本练习。此脚本会确保环境设置正确无误。

```
[student@workstation ~]$ lab ssh-access start
```

- 1. 从 workstation，以 student 用户身份打开连接 servera 的 SSH 会话。

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- 2. 以 student 身份打开连接到 serverb 的 SSH 会话。接受主机密钥。当提示您输入 serverb 上 student 用户的密码时，将 student 用作密码。

```
[student@servera ~]$ ssh student@serverb
The authenticity of host 'serverb (172.25.250.11)' can't be established.
ECDSA key fingerprint is SHA256:ERTdjoo0IrIwVSZQnqD5or+JbXfidg0udb3DXBuHWzA.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'serverb,172.25.250.11' (ECDSA) to the list of known
hosts.
student@serverb's password: student
...output omitted...
[student@serverb ~]$
```

主机密钥记录在 servera 上的 /home/student/.ssh/known_hosts 文件中，用于标识 serverb，因为 student 用户从 servera 发起了 SSH 连接。如果 /home/student/.ssh/known_hosts 文件尚未存在，则会以新文件的形式创建，并且其包含相应的新条目。如果远程主机的密钥与记录的密钥不同，ssh 命令将无法正确执行。

- 3. 运行 w 命令，以显示当前登录到 serverb 的用户。

```
[student@serverb ~]$ w
18:49:29 up 2:55, 1 user, load average: 0.00, 0.00, 0.00
USER      TTY      FROM          LOGIN@    IDLE     JCPU      PCPU WHAT
student   pts/0    172.25.250.10    18:33    0.00s  0.01s  0.00s w
```

前面的输出显示，`student` 用户已从 IP 地址为 `172.25.250.10` 的主机登录了系统，该主机就是课堂网络中的 `servera`。



注意

系统的 IP 地址在网络上标识系统。您将在后续章节中了解 IP 地址。

- 4. 退出 `serverb` 上 `student` 用户的 shell。

```
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@servera ~]$
```

- 5. 以 `root` 身份打开连接到 `serverb` 的 SSH 会话。使用 `redhat` 作为 `root` 用户的密码。

```
[student@servera ~]$ ssh root@serverb
root@serverb's password: redhat
...output omitted...
[root@serverb ~]#
```

注意前面的 `ssh` 命令没有要求您接受主机密钥，因为在已知主机中找到了该密钥。如果在任何时候 `serverb` 的身份出现变化，OpenSSH 会提示您验证和接受新的主机密钥。

- 6. 运行 `w` 命令，以显示当前登录到 `serverb` 的用户。

```
[root@serverb ~]# w
19:10:28 up 3:16, 1 user, load average: 0.00, 0.00, 0.00
USER      TTY      FROM          LOGIN@    IDLE     JCPU      PCPU WHAT
root      pts/0    172.25.250.10    19:09    1.00s  0.01s  0.00s w
```

前面的输出显示，`root` 用户已从 IP 地址为 `172.25.250.10` 的主机登录了系统，该主机就是课堂网络中的 `servera`。

- 7. 退出 `serverb` 上 `root` 用户的 shell。

```
[root@serverb ~]# exit
logout
Connection to serverb closed.
[student@servera ~]$
```

- 8. 从 `servera` 删除 `/home/student/.ssh/known_hosts` 文件。这会导致 `ssh` 丢失已记录的远程系统身份。

```
[student@servera ~]$ rm /home/student/.ssh/known_hosts
```

主机密钥可能由于合法原因而更改：可能由于硬件故障而更换了远程计算机，或者可能重新安装了远程计算机。通常，建议您仅删除特定主机在 `known_hosts` 文件中的密钥条目。由于该特定 `known_hosts` 文件中只有一个条目，因此您可以删除整个文件。

- 9. 以 `student` 身份打开连接到 `serverb` 的 SSH 会话。如果系统询问，请接受主机密钥。当提示您输入 `serverb` 上 `student` 用户的密码时，将 `student` 用作密码。

```
[student@servera ~]$ ssh student@serverb
The authenticity of host 'serverb (172.25.250.11)' can't be established.
ECDSA key fingerprint is SHA256:ERTdjoo0IrIwVSZQnqD5or+JbXfidg0udb3DXBuHWzA.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'serverb,172.25.250.11' (ECDSA) to the list of known
hosts.
student@serverb's password: student
...output omitted...
[student@serverb ~]$
```

注意 `ssh` 命令要求您确认接受或拒绝主机密钥，因为它找不到远程主机的密钥。

- 10. 退出 `serverb` 上 `student` 用户的 shell，再确认 `servera` 上存在 `known_hosts` 的新实例。

```
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@servera ~]$ ls -l /home/student/.ssh/known_hosts
-rw-r--r--. 1 student student 183 Feb 1 20:26 /home/student/.ssh/known_hosts
```

- 11. 确认 `known_hosts` 文件的新实例包含 `serverb` 的主机密钥。

```
[student@servera ~]$ cat /home/student/.ssh/known_hosts
serverb,172.25.250.11 ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBI9LEYEhwmu1rnqnbBPukH2Ba0/
QBAu9WbS4m03B3MIhhXWKFFNa/U1NjY8NDpEM+hkJe/GmnkcEYMLbCfd9nMA=
```

实际输出会有所不同。

- 12. 在 `serverb` 上远程运行 `hostname`，而不访问交互式 shell。

```
[student@servera ~]$ ssh student@serverb hostname
student@serverb's password: student
serverb.lab.example.com
```

上面的命令显示了远程系统 `serverb` 的完整主机名。

- 13. 退出 servera 上 student 用户的 shell。

```
[student@servera ~]$ exit  
logout  
Connection to servera closed.
```

完成

在 workstation 上，运行 **lab ssh-access finish** 来完成本练习。

```
[student@workstation ~]$ lab ssh-access finish
```

本引导式练习到此结束。

配置基于 SSH 密钥的身份验证

培训目标

学完本节后，您应能够将用户帐户配置为使用基于密钥的身份验证来安全地登录远程系统，并且无需输入密码。

基于 SSH 密钥的身份验证

您可以配置 SSH 服务器，以便能通过基于密钥的身份验证在不使用密码的情况下进行身份验证。这种身份验证基于私钥-公钥方案。

为此，您需要生成加密密钥文件的一个匹配对。一个是私钥，另一个是匹配的公钥。私钥文件用作身份验证凭据，像密码一样，必须妥善保管。公钥复制到用户希望连接到的系统，用于验证私钥。公钥并不需要保密。

请将公钥的副本放在服务器上的帐户中。在尝试登录时，SSH 服务器可使用公钥发出一个只能用私钥正确解答的难题。最终，您的 **ssh** 客户端可利用私钥的唯一副本自动验证您的服务器登录。这样，您就不必在每次访问系统时以交互方式键入密码，但安全性仍能得到保证。

生成 SSH 密钥

要创建用于进行身份验证的私钥和匹配的公钥，请使用 **ssh-keygen** 命令。默认情况下，私钥和公钥分别保存在 `~/.ssh/id_rsa` 和 `~/.ssh/id_rsa.pub` 文件中。

```
[user@host ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_rsa): Enter
Created directory '/home/user/.ssh'.
Enter passphrase (empty for no passphrase): Enter
Enter same passphrase again: Enter
Your identification has been saved in /home/user/.ssh/id_rsa.
Your public key has been saved in /home/user/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:vxutUNPi03QDCyvkYm1oIx35hmMrHpPKWFdIYu3HV+w user@host.lab.example.com
The key's randomart image is:
+---[RSA 2048]---+
|           |
| .         |
| o o       o |
| . = o     o . |
| o + = S E . |
| ..o o + * + |
| .+% 0 . + B . |
| =*oO . . + * |
| ++. . +.    |
+---[SHA256]---+
```

如果您未在 **ssh-keygen** 提示时指定密语，则生成的私钥不受保护。在这种情况下，任何拥有您私钥文件的人都可以使用它进行身份验证。如果您设置了密码，则在使用私钥进行身份验证时需要输入此密语。（因此，您将使用私钥密语而非远程主机上的密码进行身份验证。）

您可以运行一个名为 **ssh-agent** 的帮助程序，它可以在会话开始时临时将您的私钥密语缓存到内存中，以实现真正的无密码身份验证。本节稍后将对此进行探讨。

下例中的 **ssh-keygen** 命令显示创建受密语保护的私钥及其公钥。

```
[user@host ~]$ ssh-keygen -f .ssh/key-with-pass
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in .ssh/key-with-pass.
Your public key has been saved in .ssh/key-with-pass.pub.
The key fingerprint is:
SHA256:w3GGB7EyHUr4a0cNPKmhNKS7d11YsMVLvFZJ77VxAo user@host.lab.example.com
The key's randomart image is:
+---[RSA 2048]---+
|   . + =.o ...
|   = B XEo o.
| . o O X =....
| == = B = o.
|= + * * S .
|. + = o + .
| +
|
| +---[SHA256]---
```

-f 选项与 **ssh-keygen** 命令配合可用来确定保存密钥的文件。在前面的示例中，私钥和公钥分别保存在 **/home/user/.ssh/key-with-pass** /**/home/user/.ssh/key-with-pass.pub** 文件中。



警告

在进一步生成 SSH 密钥对期间，除非您指定唯一的文件名，否则系统会询问您是否允许覆盖现有的 **id_rsa** 和 **id_rsa.pub** 文件。如果覆盖现有的 **id_rsa** 和 **id_rsa.pub** 文件，那么您必须在具有旧公钥的所有 SSH 服务器上使用新公钥替换旧公钥。

生成 SSH 密钥后，密钥将默认存储在用户主目录下的 **.ssh/** 目录中。私钥和公钥的权限模式必须分别为 600 和 644。

共享公钥

在可以使用基于密钥的身份验证之前，需要将公钥复制到目标系统上。**ssh-copy-id** 命令可将 SSH 密钥对的公钥复制到目标系统。如果在运行 **ssh-copy-id** 时省略了公钥文件的路径，它会使用默认的 **/home/user/.ssh/id_rsa.pub** 文件。

```
[user@host ~]$ ssh-copy-id -i .ssh/key-with-pass.pub user@remotehost
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/user/.ssh/
id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted
now it is to install the new keys
user@remotehost's password: redhat
Number of key(s) added: 1
```

Now try logging into the machine, with: "ssh 'user@remotehost'"
and check to make sure that only the key(s) you wanted were added.

将公钥成功传输到远程系统后，您可以使用对应的私钥对远程系统进行身份验证，同时通过 SSH 登录远程系统。如果在运行 **ssh** 命令时省略了私钥文件的路径，它会使用默认的 **/home/user/.ssh/id_rsa** 文件。

```
[user@host ~]$ ssh -i .ssh/key-with-pass user@remotehost
Enter passphrase for key '.ssh/key-with-pass': redhatpass
...output omitted...
[user@remotehost ~]$ exit
logout
Connection to remotehost closed.
[user@host ~]$
```

使用 **ssh-agent** 进行非交互式身份验证

如果您的 SSH 私钥受密语保护，通常必须输入密语才能使用私钥进行身份验证。但是，您可以使用名为 **ssh-agent** 的程序临时将密语缓存到内存中。之后，当您使用 SSH 通过私钥登录另一个系统时，**ssh-agent** 会自动为您提供密码。这样做不仅方便，而且能减少他人“肩窥”您密码输入的机会，从而提高安全性。

如果是初始登录 GNOME 图形桌面环境，可能会自动启动并为您配置 **ssh-agent** 程序，具体取决于您本地系统的配置。

如果是在文本控制台上进行登录，使用 **ssh** 进行登录，或者使用 **sudo** 或 **su**，可能需要为该会话手动启动 **ssh-agent**。为此，可以使用以下命令：

```
[user@host ~]$ eval $(ssh-agent)
Agent pid 10155
[user@host ~]$
```



注意

当运行 **ssh-agent** 时，它会显示出一些 shell 命令。您需要运行这些命令来设置程序（如 **ssh-add**）所用的环境变量，以便与它进行通信。**eval \$(ssh-agent)** 命令将启动 **ssh-agent** 并运行这些命令以自动为该 shell 会话设置这些环境变量。此外，它还显示 **ssh-agent** 进程的 PID。

一旦 **ssh-agent** 开始运行，您需要告诉它您的私钥密语或密钥。为此，可以使用 **ssh-add** 命令。