

```
Raw Audit Messages
type=AVC msg=audit(1554726569.188:852): avc: denied { name_bind } for
pid=28393 comm="httpd" src=82 scontext=system_u:system_r:httpd_t:s0
tcontext=system_u:object_r:reserved_port_t:s0 tclass=tcp_socket permissive=0
...output omitted...
```

- 4. 将 SELinux 配置为允许 **httpd** 绑定到端口 **82/TCP**，然后重新启动 **httpd.service** 服务。

4.1. 使用 **semanage** 命令查找适合端口 **82/TCP** 的端口类型。

```
[root@servera ~]# semanage port -l | grep http
http_cache_port_t          tcp     8080, 8118, 8123, 10001-10010
http_cache_port_t          udp     3130
http_port_t                tcp     80, 81, 443, 488, 8008, 8009, 8443, 9000
pegasus_http_port_t        tcp     5988
pegasus_https_port_t       tcp     5989
```

http_port_t 中包含默认 HTTP 端口 **80/TCP** 和 **443/TCP**。这是 Web 服务器的正确端口类型。

4.2. 使用 **semanage** 命令为端口 **82/TCP** 分配 **http_port_t** 类型。

```
[root@servera ~]# semanage port -a -t http_port_t -p tcp 82
```

4.3. 使用 **systemctl** 命令重启 **httpd.service** 服务。此命令应该会成功。

```
[root@servera ~]# systemctl restart httpd.service
```

- 5. 检查您现在是否可以访问在端口 **82/TCP** 上运行的 Web 服务器。使用 **curl** 命令从 **servera** 访问 Web 服务。

```
[root@servera ~]# curl http://servera.lab.example.com:82
Hello
```

- 6. 在另一个终端窗口中，检查是否可以从 **workstation** 访问新的 Web 服务。使用 **curl** 命令从 **workstation** 访问 Web 服务。

```
[student@workstation ~]$ curl http://servera.lab.example.com:82
curl: (7) Failed to connect to servera.example.com:82; No route to host
```

该错误意味着您仍无法从 **workstation** 连接到 Web 服务。

- 7. 在 **servera** 上，打开防火墙上的端口 **82/TCP**。

7.1. 使用 **firewall-cmd** 命令，在 **servera** 上的防火墙默认区域的永久配置中打开端口 **82/TCP**。

```
[root@servera ~]# firewall-cmd --permanent --add-port=82/tcp
success
```

7.2. 在 servera 上激活防火墙更改。

```
[root@servera ~]# firewall-cmd --reload  
success
```

► 8. 使用 curl 命令从 workstation 访问 Web 服务。

```
[student@workstation ~]$ curl http://servera.lab.example.com:82  
Hello
```

► 9. 从 servera 退出。

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

完成

在 workstation 上，运行 lab netsecurity-ports finish 脚本来完成本练习。

```
[student@workstation ~]$ lab netsecurity-ports finish
```

本引导式练习到此结束。

► 开放研究实验

管理网络安全

任务执行清单

在本实验中，您将配置防火墙和 SELinux 设置，以允许访问 `serverb` 上运行的多个 Web 服务器。

成果

您应能够在 Web 服务器主机上配置防火墙和 SELinux 设置。

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 netsecurity-review start` 命令。该命令将运行一个起始脚本，以确定 `serverb` 主机是否可从网络访问。

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

您的公司已决定运行一个新 Web 应用。此应用侦听端口 **80/TCP** 和 **1001/TCP**。此外，用于访问 `ssh` 的端口 **22/TCP** 也必须可用。您进行的所有更改都应在重新启动后仍然有效。

收到 `sudo` 提示时，使用 `student` 作为密码。

1. 从 `workstation`，测试对 `http://serverb.lab.example.com` 默认 Web 服务器的访问权限，以及对 `http://serverb.lab.example.com:1001` 虚拟主机的访问权限。
2. 登录 `serverb`，确定阻止访问 Web 服务器的原因。
3. 将 SELinux 配置为允许 `httpd` 服务侦听端口 **1001/TCP**。
4. 从 `workstation`，测试对 `http://serverb.lab.example.com` 默认 Web 服务器的访问权限，以及对 `http://serverb.lab.example.com:1001` 虚拟主机的访问权限。
5. 登录 `serverb`，确定是否为防火墙分配了正确的端口。
6. 向 `public` 网络区域的永久配置中添加端口 **1001/TCP**。确认配置。
7. 从 `workstation`，确认 `serverb.lab.example.com` 的默认 Web 服务器是否返回 **SERVer B**，而 `serverb.lab.example.com:1001` 的虚拟主机是否返回 **VHOST 1**。

评估

在 `workstation` 上，运行 `lab netsecurity-review grade` 命令以确认本实验练习是否成功。

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

完成

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

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

本实验到此结束。

► 解决方案

管理网络安全

任务执行清单

在本实验中，您将配置防火墙和 SELinux 设置，以允许访问 `serverb` 上运行的多个 Web 服务器。

成果

您应能够在 Web 服务器主机上配置防火墙和 SELinux 设置。

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 netsecurity-review start` 命令。该命令将运行一个起始脚本，以确定 `serverb` 主机是否可从网络访问。

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

您的公司已决定运行一个新 Web 应用。此应用侦听端口 **80/TCP** 和 **1001/TCP**。此外，用于访问 `ssh` 的端口 **22/TCP** 也必须可用。您进行的所有更改都应在重新启动后仍然有效。

收到 `sudo` 提示时，使用 `student` 作为密码。

1. 从 `workstation`，测试对 `http://serverb.lab.example.com` 默认 Web 服务器的访问权限，以及对 `http://serverb.lab.example.com:1001` 虚拟主机的访问权限。

1.1. 测试对 `http://serverb.lab.example.com` Web 服务器的访问权限。测试目前失败。最终，Web 服务器应返回 **SERVer B**。

```
[student@workstation ~]$ curl http://serverb.lab.example.com  
curl: (7) Failed to connect to serverb.lab.example.com port 80: Connection refused
```

1.2. 测试对 `http://serverb.lab.example.com:1001` 虚拟主机的访问权限。测试目前失败。最终，虚拟主机应返回 **VHOST 1**。

```
[student@workstation ~]$ curl http://serverb.lab.example.com:1001  
curl: (7) Failed to connect to serverb.lab.example.com port 1001: No route to host
```

2. 登录 `serverb`，确定阻止访问 Web 服务器的原因。

2.1. 从 `workstation`，以 `student` 用户身份打开连接 `serverb` 的 SSH 会话。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@serverb  
...output omitted...  
[student@serverb ~]$
```

2.2. 确定 httpd 服务是否处于活动状态。

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

2.3. 启用并启动 httpd 服务。httpd 服务无法启动。

```
[student@serverb ~]$ sudo systemctl enable --now httpd  
[sudo] password for student: student  
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/  
lib/systemd/system/httpd.service.  
Job for httpd.service failed because the control process exited with error code.  
See "systemctl status httpd.service" and "journalctl -xe" for details.
```

2.4. 调查 httpd.service 服务启动失败的原因。

```
[student@serverb ~]$ systemctl status httpd.service  
● httpd.service - The Apache HTTP Server  
  Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset:  
disabled)  
  Active: failed (Result: exit-code) since Thu 2019-04-11 19:25:36 CDT; 19s ago  
    Docs: man:httpd.service(8)  
   Process: 9615 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND (code=exited,  
status=1/FAILURE)  
 Main PID: 9615 (code=exited, status=1/FAILURE)  
    Status: "Reading configuration..."  
  
Apr 11 19:25:36 serverb.lab.example.com systemd[1]: Starting The Apache HTTP  
Server...  
Apr 11 19:25:36 serverb.lab.example.com httpd[9615]: (13)Permission denied:  
AH00072: make_sock: could not bind to address [::]:1001  
Apr 11 19:25:36 serverb.lab.example.com httpd[9615]: (13)Permission denied:  
AH00072: make_sock: could not bind to address 0.0.0.0:1001  
Apr 11 19:25:36 serverb.lab.example.com httpd[9615]: no listening sockets  
available, shutting down  
Apr 11 19:25:36 serverb.lab.example.com httpd[9615]: AH00015: Unable to open logs  
Apr 11 19:25:36 serverb.lab.example.com systemd[1]: httpd.service: Main process  
exited, code=exited, status=1/FAILURE  
Apr 11 19:25:36 serverb.lab.example.com systemd[1]: httpd.service: Failed with  
result 'exit-code'.  
Apr 11 19:25:36 serverb.lab.example.com systemd[1]: Failed to start The Apache  
HTTP Server.
```

2.5. 使用 **sealert** 命令检查 SELinux 是否在阻止 httpd 服务绑定到端口 1001/TCP。

```
[student@serverb ~]$ sudo sealert -a /var/log/audit/audit.log
100% done
found 1 alerts in /var/log/audit/audit.log
-----
SELinux is preventing /usr/sbin/httpd from name_bind access on the tcp_socket port
1001.

***** Plugin bind_ports (99.5 confidence) suggests *****

If you want to allow /usr/sbin/httpd to bind to network port 1001
Then you need to modify the port type.
Do
# semanage port -a -t PORT_TYPE -p tcp 1001
    where PORT_TYPE is one of the following: http_cache_port_t, http_port_t,
jboss_management_port_t, jboss.messaging_port_t, ntop_port_t, puppet_port_t.

***** Plugin catchall (1.49 confidence) suggests *****

...output omitted...
```

3. 将 SELinux 配置为允许 **httpd** 服务侦听端口 **1001/TCP**。

3.1. 使用 **semanage** 命令查找正确的端口类型。

```
[student@serverb ~]$ sudo semanage port -l | grep 'http'
http_cache_port_t      tcp  8080, 8118, 8123, 10001-10010
http_cache_port_t      udp  3130
http_port_t           tcp  80, 81, 443, 488, 8008, 8009, 8443, 9000
pegasus_http_port_t    tcp  5988
pegasus_https_port_t   tcp  5989
```

3.2. 使用 **semanage** 命令将端口 **1001/TCP** 绑定到 **http_port_t** 类型。

```
[student@serverb ~]$ sudo semanage port -a -t http_port_t -p tcp 1001
[student@serverb ~]$
```

3.3. 确认端口 **1001/TCP** 是否已绑定到 **http_port_t** 端口类型。

```
[student@serverb ~]$ sudo semanage port -l | grep '^http_port_t'
http_port_t           tcp  1001, 80, 81, 443, 488, 8008, 8009, 8443, 9000
```

3.4. 启用并启动 **httpd** 服务。

```
[student@serverb ~]$ sudo systemctl enable --now httpd
```

3.5. 验证 **httpd** 服务的运行状态。

```
[student@serverb ~]$ systemctl is-active httpd; systemctl is-enabled httpd
active
enabled
```

3.6. 从 serverb 退出。

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

4. 从 workstation, 测试对 `http://serverb.lab.example.com` 默认 Web 服务器的访问权限, 以及对 `http://serverb.lab.example.com:1001` 虚拟主机的访问权限。

4.1. 测试对 `http://serverb.lab.example.com` Web 服务器的访问权限。Web 服务器应返回 SERVER B。

```
[student@workstation ~]$ curl http://serverb.lab.example.com
SERVER B
```

4.2. 测试对 `http://serverb.lab.example.com:1001` 虚拟主机的访问权限。测试仍旧失败。

```
[student@workstation ~]$ curl http://serverb.lab.example.com:1001
curl: (7) Failed to connect to serverb.lab.example.com port 1001: No route to host
```

5. 登录 serverb, 确定是否为防火墙分配了正确的端口。

5.1. 从 workstation, 以 student 用户身份登录 serverb。

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

5.2. 验证默认防火墙区域是否已设置为 public。

```
[student@serverb ~]$ firewall-cmd --get-default-zone
public
```

5.3. 如果上一步未返回 public 作为默认区域, 则使用以下命令进行修正:

```
[student@serverb ~]$ sudo firewall-cmd --set-default-zone public
```

5.4. 确定 public 网络区域中列出的开放端口。

```
[student@serverb ~]$ sudo firewall-cmd --permanent --zone=public --list-all
[sudo] password for student: student
public
```

```
target: default
icmp-block-inversion: no
interfaces:
sources:
services: cockpit dhcpcv6-client http ssh
ports:
protocols:
masquerade: no
forward-ports:
source-ports:
icmp-blocks:
rich rules:
```

6. 向 public 网络区域的永久配置中添加端口 **1001/TCP**。确认配置。

- 6.1. 向 public 网络区域中添加端口 **1001/TCP**。

```
[student@serverb ~]$ sudo firewall-cmd --permanent --zone=public \
--add-port=1001/tcp
success
```

- 6.2. 重新加载防火墙配置。

```
[student@serverb ~]$ sudo firewall-cmd --reload
success
```

- 6.3. 确认配置。

```
[student@serverb ~]$ sudo firewall-cmd --permanent --zone=public --list-all
public
target: default
icmp-block-inversion: no
interfaces:
sources:
services: cockpit dhcpcv6-client http ssh
ports: 1001/tcp
protocols:
masquerade: no
forward-ports:
source-ports:
icmp-blocks:
rich rules:
```

- 6.4. 从 serverb 退出。

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

7. 从 workstation，确认 serverb.lab.example.com 的默认 Web 服务器是否返回 **SERVER B**，而 serverb.lab.example.com:1001 的虚拟主机是否返回 **VHOST 1**。

7.1. 测试对 `http://serverb.lab.example.com` Web 服务器的访问权限。

```
[student@workstation ~]$ curl http://serverb.lab.example.com  
SERVER B
```

7.2. 测试对 `http://serverb.lab.example.com:1001` 虚拟主机的访问权限。

```
[student@workstation ~]$ curl http://serverb.lab.example.com:1001  
VHOST 1
```

评估

在 workstation 上，运行 `lab netsecurity-review grade` 命令以确认本实验练习是否成功。

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

完成

在 workstation 上，运行 `lab netsecurity-review finish` 脚本来完成本练习。

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

本实验到此结束。

总结

在本章中，您学到了：

- **netfilter** 子系统允许内核模块对遍历系统的每个数据包进行检查。检查所有传入、传出或转发的网络数据包。
- 使用 **firewalld** 可将所有网络流量分为多个区域，从而简化了管理。每个区域都有自己的端口和服务列表。**public** 区域被设置为默认区域。
- **firewalld** 服务随附有一些预定义服务。使用 **firewall-cmd --get-services** 命令可以列出这些服务。
- SELinux 策略严格控制网络流量。网络端口已予以标记。例如，端口 **22/TCP** 具有标签 **ssh_port_t** 与其相关联。当某个进程希望侦听端口时，SELinux 将检查是否允许与其相关联的标签绑定该端口标签。
- **semanage** 命令用于添加、删除和修改标签。

章 12

安装红帽企业 LINUX

目标

在服务器和虚拟机上安装红帽企业 Linux。

培训目标

- 在服务器上安装红帽企业 Linux。
- 使用 Kickstart 自动完成安装过程。
- 使用 Cockpit 在红帽企业 Linux 服务器上安装虚拟机。

章节

- 安装红帽企业 Linux (及引导式练习)
- 使用 Kickstart 自动安装 (及引导式练习)
- 安装和配置虚拟机 (及测验)

实验

安装红帽企业 Linux

安装红帽企业 LINUX

培训目标

学完本节后，您应能够在服务器上安装红帽企业 Linux。

选择安装介质

红帽提供了几种安装介质；您可以使用自己的有效订阅从客户门户网站中下载这些介质。

- 含有 Anaconda 的二进制 DVD，红帽企业 Linux 安装程序，以及 BaseOS 和 AppStream 软件包存储库。这些存储库中包含有完成安装所需的软件包，无需其他材料。
- 一个含有 Anaconda 的启动 ISO，但它需要配置网络，以便访问通过 HTTP、FTP 或 NFS 提供的软件包存储库。
- 一个含有预构建系统磁盘的 QCOW2 映像，它可以在云或企业虚拟环境中部署为虚拟机。QCOW2 (QEMU Copy On Write) 是红帽使用的标准镜像格式。

红帽为所支持的四种处理器架构提供了安装介质：x86 64 位（AMD 和 Intel）、IBM Power Systems (Little Endian)、IBM Z 及 ARM 64 位。

下载后，刻录 DVD 或将 ISO 启动到物理介质，将每个都复制到 USB 闪存驱动器或类似设备，或者从网络服务器进行发布，以供自动 Kickstart 使用。

使用 Composer 构建镜像

Composer 是 RHEL 8 中提供的新工具。针对特殊用例，Composer 允许管理员构建自定义系统镜像，以便在云平台或虚拟环境中进行部署。

Composer 使用 Cockpit 图形 Web 控制台。也可以使用 **composer-cli** 命令，从命令行调用 Composer。

通过 ANACONDA 进行手动安装

借助二进制 DVD 或启动 ISO，管理员可以在一台裸机服务器或虚拟机上安装新的 RHEL 系统。Anaconda 程序支持两种安装方法：

- 手动安装将与用户进行交互，以查询 Anaconda 应如何安装和配置系统。
- 自动安装将使用 Kickstart 文件，它会告诉 Anaconda 如何安装系统。后面的部分将更详细地讨论 Kickstart 安装。

使用图形界面安装 RHEL

通过二进制 DVD 或启动 ISO 来启动系统时，Anaconda 将作为图形应用启动。

在 Welcome to Red Hat Enterprise Linux 8 屏幕中，选择安装期间要使用的语言。这也会设置系统安装后的默认语言。每个用户都可以在安装后选择自己帐户的首选语言。

Anaconda 中显示 Installation Summary 窗口，这是在开始安装之前自定义参数的中心位置。

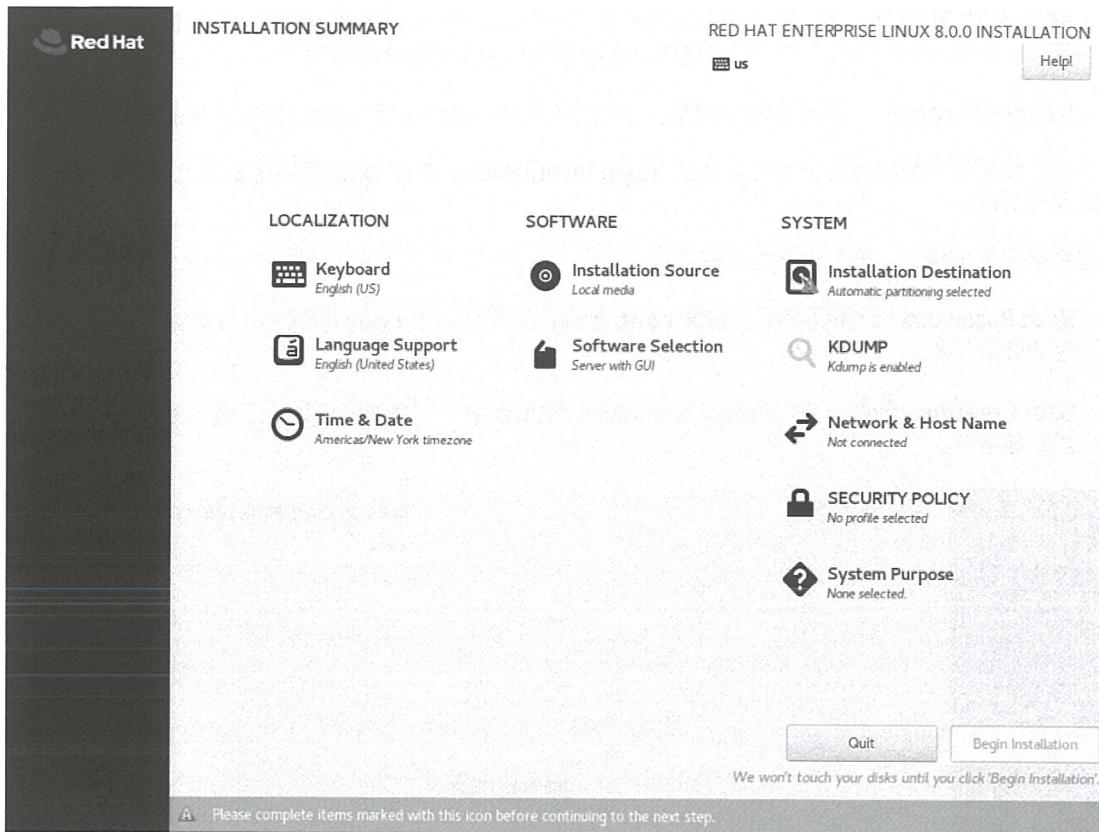


图 12.1: Installation Summary 窗口

在此窗口中，通过按任意顺序选择图标来配置安装参数。选择项目进行查看或编辑。在任一项目中，单击 **Done** 可返回此中心屏幕。

Anaconda 使用三角形警告符号和消息来标记必填的项目。屏幕底部显示橙色状态栏是提醒您必须完成这些必填的项目后才能开始安装。

根据需要填写以下项目：

- **Keyboard** - 添加其他键盘布局。
- **Language Support** - 选择要安装的其他语言。
- **Time & Date** - 通过在交互式地图中单击或者从下拉列表中选择，选择系统所在的城市。指定本地时区，即便是使用了网络时间协议 (NTP)。
- **Installation Source** - 提供 Anaconda 安装所需的源软件包位置。如果使用二进制 DVD，则安装源字段已指向 DVD。
- **Software Selection** - 选择要安装的基本环境以及其他任何附加组件。Minimal Install 环境将仅安装运行红帽企业 Linux 所需的基本软件包。
- **Installation Destination** - 选择磁盘并为其分区，以便将红帽企业 Linux 安装到其中。此项要求管理员掌握分区方案和文件系统选择标准。自动分区的默认单选按钮将使用所有可用的空间来分配所选的存储设备。
- **KDUMP** - Kdump 是一项内核功能，它可在内核崩溃时收集系统内存内容。红帽工程师可以分析 kdump，以确定崩溃的原因。使用此 Anaconda 项目可以启用或禁用 Kdump。
- **Network & Host Name** - 检测到的网络连接列在左侧。选择一个连接以显示其详细信息。要配置所选的网络连接，请单击 **Configure**。

- **SECURITY POLICY** - 通过激活安全策略配置文件（例如支付卡行业数据安全标准 (PCI DSS) 配置文件），Anaconda 可在安装期间实施由所选配置文件定义的限制和建议。
- **System Purpose** - 一个新的安装功能，可用于分配与预期系统用途吻合的有效系统权利。

完成安装配置并解除所有警告后，单击 **Begin Installation**。单击 **Quit** 将中止安装且不应用对系统的任何更改。

在系统安装过程中，填写以下显示的项目：

- **Root Password** - 安装程序提示设置 **root** 密码。只有定义了 **root** 密码后，才会继续执行安装过程的最终阶段。
- **User Creation** - 创建一个可选的非 **root** 帐户。建议维护一个本地通用帐户。也可以在安装完成后创建帐户。

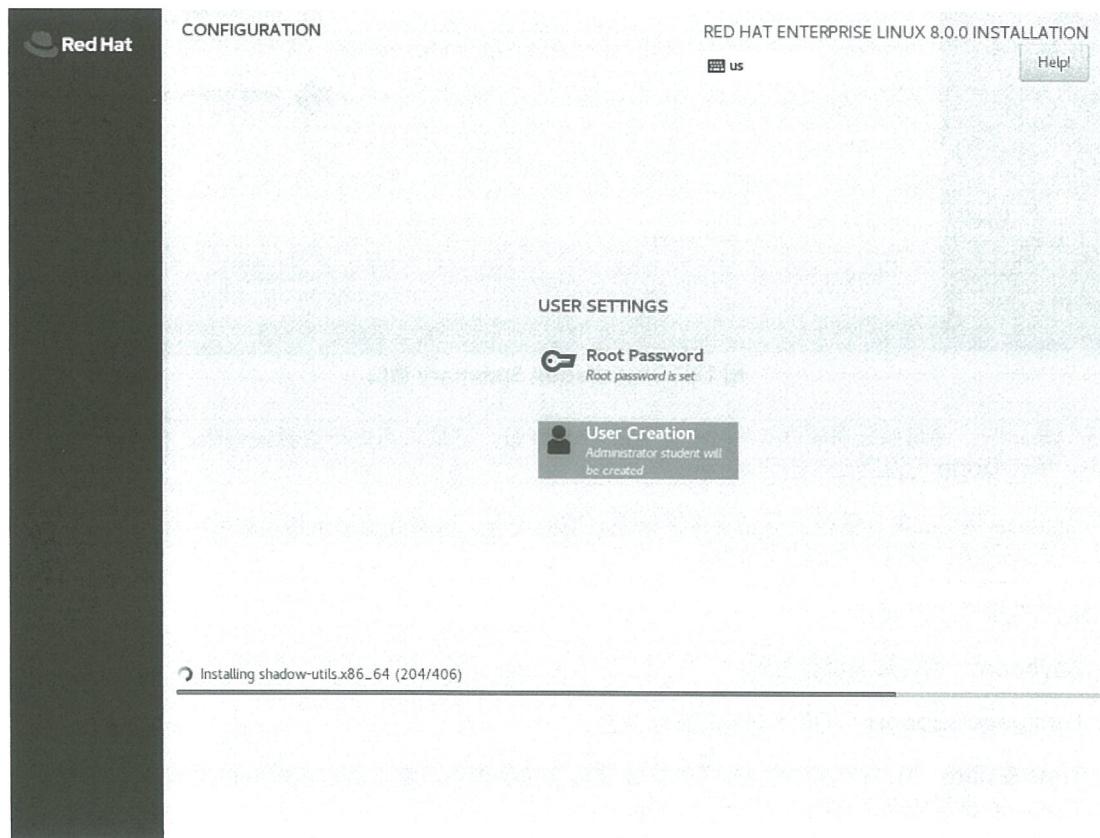


图 12.2: 设置 root 密码并创建用户

安装完成后，单击 **Reboot**。如果安装了图形桌面，Anaconda 就会显示 **Initial Setup** 屏幕。接受许可证信息，并可选择向订阅管理器注册系统。您可以跳过系统注册，稍后再执行该操作。

对安装进行故障排除

在红帽企业 Linux 8 安装期间，Anaconda 提供了两个虚拟控制台。第一个虚拟控制台有五个窗口，它们由 **tmux** 软件终端多路复用器提供。您可以通过 **Ctrl+Alt+F1** 访问该控制台。第二个虚拟控制台（默认会显示此控制台）将显示 Anaconda 图形界面。您可以通过 **Ctrl+Alt+F6** 来访问该控制台。

在第一个虚拟控制台中，**tmux** 在第二个窗口中提供了 shell 提示符。您可以使用它来输入命令，以在安装过程中检查系统并进行故障排除。另外的窗口中提供日志消息、日志及其他信息。

下表列出了用于访问虚拟控制台和 **tmux** 窗口的按键组合。对于 **tmux**，键盘快捷键通过两个操作执行：按下和释放 **Ctrl+b**，然后按下您要访问的窗口所对应的数字键。对于 **tmux**，您也可以使用 **Alt+Tab** 在窗口之间轮换当前焦点。

按键顺序	内容
Ctrl+Alt+F1	访问 tmux 终端多路复用器。
Ctrl+b 1	在 tmux 中时，访问安装过程的主要信息页面。
Ctrl+b 2	在 tmux 中时，提供 root shell。Anaconda 会将安装日志文件存储到 /tmp 文件中。
Ctrl+b 3	在 tmux 中时，显示 /tmp/anaconda.log 文件的内容。
Ctrl+b 4	在 tmux 中时，显示 /tmp/storage.log 文件的内容。
Ctrl+b 5	在 tmux 中时，显示 /tmp/program.log 文件的内容。
Ctrl+Alt+F6	访问 Anaconda 图形界面。



注意

为了与早期的红帽企业 Linux 版本兼容，通过 **Ctrl+Alt+F2** 到 **Ctrl+Alt+F5** 访问的虚拟控制台还会在安装过程中显示 root shell。



参考文献

如需更多信息，请参阅《RHEL 安装和部署指南》，网址为：

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/performing_a_standard_rhel_installation/index

► 指导练习

安装红帽企业 LINUX

在本练习中，您将以红帽企业 Linux 的最小安装模式重新安装其中一个服务器。

成果

您应能够手动安装红帽企业 Linux 8。

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。

在 workstation 上，运行 **lab installing-install start** 命令。此命令将运行一个起始脚本，它将确定 servera 计算机是否可从网络访问。此外，它还将在 GRUB2 菜单中添加一个新条目，用于将 servera 启动到安装介质。

```
[student@workstation ~]$ lab installing-install start
```

► 1. 访问 servera 控制台并将系统重启到安装介质。

- 1.1. 根据您的课堂环境，找到 servera 控制台的图标。打开控制台。
- 1.2. 要想重启，请使用相关键盘、虚拟设备或菜单项向系统发送 **Ctrl+Alt+Del**。
- 1.3. 当显示启动加载器菜单时，选择 Install Red Hat Enterprise Linux 8。
- 1.4. 等待语言选择窗口出现。

► 2. 保持默认选择的语言并单击 Continue。

► 3. 对 /dev/vda 磁盘使用自动分区。

- 3.1. 单击 Installation Destination。
- 3.2. 单击第一个磁盘 vda 以将其选定。单击 Done，以使用自动分区的默认选项。
- 3.3. 在 Installation Options 窗口中，单击 Reclaim space。由于 /dev/vda 磁盘已经有之前安装的分区和文件系统，因此该选项允许您擦除磁盘以进行新的安装。在 Reclaim Disk Space 窗口中，单击 Delete all，然后单击 Reclaim space。

► 4. 将服务器主机名设为 servera.lab.example.com，然后验证网络接口的配置。

- 4.1. 单击 Network & Host Name。
- 4.2. 在 Host Name 字段中，输入 **servera.lab.example.com**，然后单击 Apply。
- 4.3. 单击 Configure，然后单击 IPv4 Settings 选项卡。
- 4.4. 确认网络参数是否正确。IP 地址为 172.25.250.10，子网掩码为 24，网关和名称服务器均设置为 172.25.250.254。单击 Save。

- 4.5. 确认是否已通过将 ON/OFF 设为 **ON** 而启用了网络接口。
- 4.6. 单击 **Done**。
- ▶ 5. 将 Installation Source 字段设置为 `http://content.example.com/rhel8.0/x86_64/dvd`。
 - 5.1. 单击 Installation Source。
 - 5.2. 在 http:// 字段中, 键入 `content.example.com/rhel8.0/x86_64/dvd`
 - 5.3. 单击 **Done**。
- ▶ 6. 选择运行最小安装所需的软件。
 - 6.1. 单击 Software Selection。
 - 6.2. 从 Base Environment 列表中, 选择 Minimal Install。
 - 6.3. 单击 **Done**。
- ▶ 7. 配置系统的用途。
 - 7.1. 单击 System Purpose。
 - 7.2. 选择 Red Hat Enterprise Linux Server 角色。
 - 7.3. SLA 级别选择为 Self-Support。
 - 7.4. 用途选择为 Development/Test。
 - 7.5. 单击 **Done**。
- ▶ 8. 单击 Begin Installation。
- ▶ 9. 在安装过程中, 将 root 用户的密码设为 redhat。
 - 9.1. 单击 Root Password。
 - 9.2. 在 Root Password 字段中, 输入 `redhat`。
 - 9.3. 在 Confirm 字段中, 输入 `redhat`。
 - 9.4. 此密码安全性很弱, 因此您需要单击 Done 两次。
- ▶ 10. 在安装过程中, 添加 student 用户。
 - 10.1. 单击 User Creation。
 - 10.2. 在 Full Name 字段中, 输入 `student`。
 - 10.3. 选中 Make this user administrator, 这样 student 即可使用 sudo 以 root 用户身份运行命令。
 - 10.4. 在 Password 字段中, 输入 `student`。
 - 10.5. 在 Confirm password 字段中, 输入 `student`。

10.6.此密码安全性很弱，因此您需要单击 Done 两次。

► 11. 安装完成后，单击 Reboot。

► 12. 系统显示登录提示符时，以 student 用户身份并使用密码 student 进行登录。

完成

使用适合您课堂环境的方法重置 servera 计算机。

本引导式练习到此结束。

使用 KICKSTART 自动安装

培训目标

学完本节后，您应能够：

- 讲解 Kickstart 概念和架构。
- 使用 Kickstart Generator 网站创建 Kickstart 文件。
- 使用文本编辑器修改现有 Kickstart 文件，并使用 **ksvalidator** 检查其语法。
- 将 Kickstart 文件发布到安装程序。
- 执行网络 Kickstart 安装。

创建 KICKSTART 配置文件

您可以使用名为 Kickstart 的功能自动安装红帽企业 Linux。借助 Kickstart，您可以在 Kickstart 文本文件中指定 Anaconda 完成安装所需的所有内容，包括磁盘分区、网络接口配置、程序包选择及其他参数。通过引用此文本文件，Anaconda 无需进一步的用户交互即可执行安装过程。



注意

红帽企业 Linux 中的 Kickstart 与 Oracle Solaris 中的 Jumpstart 功能或 Microsoft Windows 使用无人值守安装应答文件相类似。

Kickstart 文件以一个命令列表开头，这些命令定义如何安装目标计算机。以 # 字符开头的行是安装程序将会忽略的注释。其他部分以指令开头（由第一个字符 % 识别），并以带有 %end 指令的行结束。

%packages 部分指定要在目标系统上安装的软件。请根据名称（不带版本）指定单个软件包。软件包组（根据名称或 ID 指定）的识别方式是以 @ 字符开头。环境组（软件包组的组）的识别方式是以 @^ 字符开头。请使用 @module:stream/profile 语法来指定模块、流和配置文件。

组具有必需、默认和可选组件。通常，Kickstart 将安装必需组件和默认组件。要从安装中排除某个软件包或软件包组，请在其前面加上 - 字符。但是，如果排除的软件包或软件包组是其他所请求软件包的强制依赖项，则仍可能会安装这些软件包或软件包组。

Kickstart 配置通常使用两个附加部分：**%pre** 和 **%post**，它们含有能进一步配置系统的 shell 脚本命令。**%pre** 脚本在进行任何磁盘分区之前执行。通常，只有在磁盘分区之前需要采取操作来识别或初始化设备时，才会使用此部分。**%post** 脚本在以其他方式完成安装之后执行。

您必须在 **%pre**、**%post** 和 **%packages** 部分之前指定主要 Kickstart 命令，但除此之外，您可以在文件中以任意顺序放置这些部分。

KICKSTART 文件命令

安装命令

定义安装源以及如何执行安装。每个后面都有一个示例。

- **url**: 指定指向安装介质的 URL。

```
url --url="http://classroom.example.com/content/rhel8.0/x86_64/dvd/"
```

- **repo**: 指定到哪里查找要安装的其他软件包。此选项必须指向有效的 yum 存储库。

```
repo --name="appstream" --baseurl=http://classroom.example.com/content/rhel8.0/x86_64/dvd/AppStream/
```

- **text**: 强制进行文本模式安装。

- **vnc**: 允许通过 VNC 远程查看图形安装。

```
vnc --password=redhat
```

分区命令

定义要使用的设备和分区方案。

- **clearpart**: 在创建新分区之前从系统中删除分区。默认情况下，不会删除任何分区。

```
clearpart --all --drives=sda,sdb --initlabel
```

- **part**: 指定分区的大小、格式和名称。

```
part /home --fstype=ext4 --label=homes --size=4096 --maxsize=8192 --grow
```

- **autopart**: 自动为架构创建 root 分区、交换分区和适当的启动分区。在足够大的驱动器上，此时还会创建 /home 分区。

- **ignoredisk**: 控制 Anaconda 对系统所连接的磁盘的访问。

```
ignoredisk --drives=sdc
```

- **bootloader**: 定义在何处安装启动加载器。

```
bootloader --location=mbr --boot-drive=sda
```

- **volgroup**、**logvol**: 创建 LVM 卷组和逻辑卷。

```
part pv.01 --size=8192
volgroup myvg pv.01
logvol / --vname=myvg --fstype=xfs --size=2048 --name=rootvol --grow
logvol /var --vname=myvg --fstype=xfs --size=4096 --name=varvol
```

- **zerombr**: 对格式未被识别的磁盘执行初始化。

网络命令

定义主机所使用的网络功能。

- **network**: 配置目标系统的网络信息。激活安装程序环境中的网络设备。

```
network --device=eth0 --bootproto=dhcp
```

- **firewall**: 定义目标系统的防火墙配置。

```
firewall --enabled --service=ssh,http
```

位置和安全命令

配置与安全性、语言和区域相关的设置。

- **lang**: 设置安装时要使用的语言和已安装系统的默认语言。必填。

```
lang en_US.UTF-8
```

- **keyboard**: 设置系统键盘类型。必填。

```
keyboard --vckeymap=us --xlayouts=''
```

- **timezone**: 定义时区、NTP 服务器以及硬件时钟是否使用 UTC。

```
timezone --utc --ntpserver=time.example.com Europe/Amsterdam
```

- **authselect**: 设置身份验证选项。能被 **authselect** 识别的选项对此命令有效。请参阅 authselect(8)。

- **rootpw**: 定义初始 **root** 用户密码。

```
rootpw --plaintext redhat
```

or

```
rootpw --iscrypted $6$KUnFfrTz08jv.PiH$YlBb0tXBkWzoMuRfb0.SpbQ....XDR1Uuch0MG1
```

- **selinux**: 设置已安装系统的 SELinux 模式。

```
selinux --enforcing
```

- **services**: 修改默认的 **systemd** 目标下运行的默认服务集合。

```
services --disabled=network,iptables,ip6tables --enabled=NetworkManager,firewalld
```

- **group**、**user**: 在系统上创建本地组或用户。

```
group --name=admins --gid=10001
```

```
user --name=jdoe --gecos="John Doe" --groups=admins --password=changeme --  
plaintext
```

杂项命令

配置与安装期间的日志记录和完成时的主机电源状态相关的杂项。

- **logging**: 此命令定义安装期间 Anaconda 将如何执行日志记录。

- ```
logging --host=loghost.example.com --level=info
```
- **firstboot**: 如果启用，则在系统首次启动时将启动安装代理。必须安装 initial-setup 软件包。
- ```
firstboot --disabled
```
- **reboot**、**poweroff**、**halt**: 指定安装完成时要执行的最终操作。

**注意**

对于识别红帽企业 Linux 或 Fedora 的两个版本间 Kickstart 文件语法中的区别，pykickstart 软件包中的 **ksverdiff** 实用程序很有用。

例如，**ksverdiff -f RHEL7 -t RHEL8** 将识别从 RHEL 7 到 RHEL 8 的语法更改。可用版本在 **/usr/lib/python3.6/site-packages/pykickstart/version.py** 文件顶部列出。

示例 KICKSTART 文件

该文件的第一部分由安装命令组成，如磁盘分区和安装源等。

```
#version=RHEL8
ignoredisk --only-use=vda
# System bootloader configuration
bootloader --append="console=ttyS0 console=ttyS0,115200n8 no_timer_check
net.ifnames=0 crashkernel=auto" --location=mbr --timeout=1 --boot-drive=vda
# Clear the Master Boot Record
zerombr
# Partition clearing information
clearpart --all --initlabel
# Use text mode install
text
repo --name="appstream" --baseurl=http://classroom.example.com/content/rhel8.0/
x86_64/dvd/AppStream/
# Use network installation
url --url="http://classroom.example.com/content/rhel8.0/x86_64/dvd/"
# Keyboard layouts
# old format: keyboard us
# new format:
keyboard --vckeymap=us --xlayouts=''
# System language
lang en_US.UTF-8
# Root password
rootpw --plaintext redhat
# System authorization information
auth --enableshadow --passalgo=sha512
# SELinux configuration
selinux --enforcing
firstboot --disable
# Do not configure the X Window System
skipx
# System services
```

```
services --disabled="kdump, rhsmcertd" --enabled="sshd, rngd, chronyd"
# System timezone
timezone America/New_York --isUtc
# Disk partitioning information
part / --fstype="xfs" --ondisk=vda --size=10000
```

第二部分包含 **%packages** 部分，详细说明应当安装的软件包和软件包组，以及不应安装的软件包。

```
%packages
@core
chrony
cloud-init
dracut-config-generic
dracut-norescue
firewalld
grub2
kernel
rsync
tar
-plymouth

%end
```

最后一个部分包含所有 **%pre** 和 **%post** 安装脚本。

```
%post --erroronfail

# For cloud images, 'eth0' _is_ the predictable device name, since
# we don't want to be tied to specific virtual (!) hardware
rm -f /etc/udev/rules.d/70*
ln -s /dev/null /etc/udev/rules.d/80-net-name-slot.rules

# simple eth0 config, again not hard-coded to the build hardware
cat > /etc/sysconfig/network-scripts/ifcfg-eth0 << EOF
DEVICE="eth0"
BOOTPROTO="dhcp"
ONBOOT="yes"
TYPE="Ethernet"
USERCTL="yes"
PEERDNS="yes"
IPV6INIT="no"
EOF

%end
```



注意

在 Kickstart 文件中，缺少必需值会导致安装程序中断并以交互方式提示用户输入答案或完全中止安装。

KICKSTART 安装步骤

要想成功地自动安装红帽企业 Linux，请按照以下步骤操作：

1. 创建 Kickstart 文件。
2. 将 Kickstart 文件发布到安装程序。
3. 启动 Anaconda 并将其指向 Kickstart 文件。

创建 KICKSTART 文件

使用以下任一方法创建 Kickstart 文件：

- 使用 Kickstart Generator 网站。
- 使用文本编辑器。

Kickstart Generator 网站（网址为 <https://access.redhat.com/labs/kickstartconfig/>）会显示用户输入对话框，并按照用户的选择创建 Kickstart 指令文本文件。每个对话框都对应 Anaconda 安装程序中的可配置项。

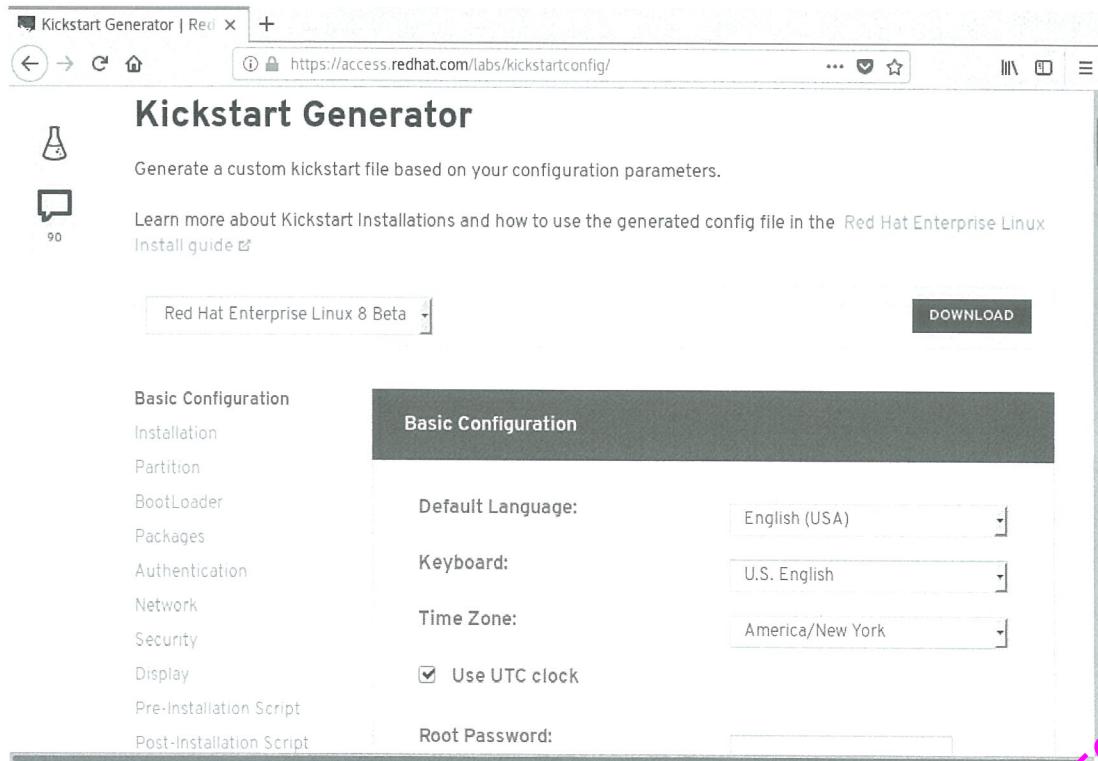


图 12.3: 使用 Kickstart Generator 进行基本配置



注意

在撰写本文时，Kickstart Generator 网站没有提供 红帽企业 Linux 8 作为菜单选项。红帽企业 Linux 8 Beta 版曾经是一个有效的选择。

从头开始创建 Kickstart 文件通常过于复杂，但编辑现有的 Kickstart 文件会很常见且很有用。每个安装都会创建一个 `/root/anaconda-ks.cfg` 文件，其中包含了该安装中使用的 Kickstart 指令。在手动创建 Kickstart 文件时，此文件是一个很好的起点。