

2.1. 创建 **vars/users_vars.yml** 变量文件来定义用户 ops1 和 ops2，这两个用户从属于 webadmin 用户组。您可能需要创建 **vars** 子目录。

```
[student@workstation system-review]$ mkdir vars
[student@workstation system-review]$ vi vars/users_vars.yml
---
users:
  - username: ops1
    groups: webadmin
  - username: ops2
    groups: webadmin
```

2.2. 创建 **users.yml** playbook。在 playbook 中定义一个以 **webservers** 主机组为目标的 play。添加 **vars_files** 子句，以定义 **vars/users_vars.yml** 文件名的位置。添加一个任务，以使用 **group** 模块在远程主机上创建 **webadmin** 用户组。

```
---
- name: Create multiple local users
  hosts: webservers
  vars_files:
    - vars/users_vars.yml
  tasks:
    - name: Add webadmin group
      group:
        name: webadmin
        state: present
```

2.3. 在 playbook 中添加第二个任务，以使用 **user** 模块来创建用户。在任务中添加 **loop: "{{ users }}"** 子句，在变量文件中循环遍历 **vars/users_vars.yml** 文件中找到的每个用户名。对于用户的 **name:**，使用 **item.username** 变量名称。这样，变量文件可以包含可能对创建用户有用的其他信息，例如用户应属于的组。第二个任务包含下列内容：

```
- name: Create user accounts
  user:
    name: "{{ item.username }}"
    groups: webadmin
  loop: "{{ users }}"
```

2.4. 执行 playbook：

```
[student@workstation system-review]$ ansible-playbook users.yml
PLAY [Create multiple local users]
*****
TASK [Gathering Facts]
*****
ok: [serverb.lab.example.com]

TASK [Add webadmin group]
*****
```

```

changed: [serverb.lab.example.com]

TASK [Create user accounts]
*****
changed: [serverb.lab.example.com] => (item={'username': 'ops1', 'groups': 'webadmin'})
changed: [serverb.lab.example.com] => (item={'username': 'ops2', 'groups': 'webadmin'})

PLAY RECAP
*****
serverb.lab.example.com      : ok=3    changed=2    unreachable=0    failed=0

```

3. 创建一个 playbook 并在 `webservers` 主机组中执行，以使用 `/dev/vdb` 设备创建名为 `apache-vg` 的卷组。此 playbook 还将创建两个逻辑卷，取名为 `content-lv` 和 `logs-lv`，它们都由 `apache-vg` 卷组支持。最后，它在每个逻辑卷上创建 XFS 文件系统，并将 `content-lv` 和 `logs-lv` 逻辑卷分别挂载到 `/var/www` 和 `/var/log/httpd`。实验室脚本在 `~/system-review` 中填充两个文件，`storage.yml` 提供 playbook 的初始框架，`storage_vars.xml` 则为不同模块所需的所有变量提供值。

3.1. 检查 `storage_vars.yml` 变量文件。

```
[student@workstation system-review]$ cat storage_vars.yml
---

partitions:
  - number: 1
    start: 1MiB
    end: 257MiB

volume_groups:
  - name: apache-vg
    devices: /dev/vdb1

logical_volumes:
  - name: content-lv
    size: 64M
    vgroup: apache-vg
    mount_path: /var/www

  - name: logs-lv
    size: 128M
    vgroup: apache-vg
    mount_path: /var/log/httpd
```

此文件描述各 Web 服务器上的预期分区结构、卷组和逻辑卷。第一个分区始于 `/dev/vdb` 设备开头偏移 1 MiB 处，终于偏移 257 MiB 处，总大小为 256 MiB。

每台 Web 服务器上具有一个卷组 `apache-vg`，其包含 `/dev/vdb1` 设备的第一个分区。

每台 Web 服务器具有两个逻辑卷。第一逻辑卷名为 `content-lv`，大小为 64 MiB，连至 `apache-vg` 卷组，并挂载于 `/var/www`。第二逻辑卷名为 `logs-lv`，大小为 128 MiB，连至 `apache-vg` 卷组，并挂载于 `/var/log/httpd`。

**注意**

apache-vg 卷组的容量为 256 MiB，因为它由 `/dev/vdb1` 分区提供支持。它为两个逻辑卷提供充足的容量。

- 3.2. 更改 **storage.yml** playbook 中的第一个任务，以使用 **parted** 模块来配置各个循环项的分区。各个项目描述每台 Web 服务器上 `/dev/vdb` 设备的预期分区：

number

分区编号。将此用作 **parted** 模块 **number** 关键字的值。

start

分区的起点，表示为块设备开头的偏移量。将此用作 **parted** 模块 **part_start** 关键字的值。

end

分区的终点，表示为块设备开头的偏移量。将此用作 **parted** 模块 **part_end** 关键字的值。

第一个任务的内容应该是：

```
- name: Correct partitions exist on /dev/vdb
  parted:
    device: /dev/vdb
    state: present
    number: "{{ item.number }}"
    part_start: "{{ item.start }}"
    part_end: "{{ item.end }}"
    loop: "{{ partitions }}"
```

- 3.3. 更改 play 的第二个任务，以使用 **lvg** 模块为各个循环项配置卷组。**volume_groups** 变量的每个项目描述各 Web 服务器上应存在的卷组：

name

卷组的名称。将此用作 **lvg** 模块 **vg** 关键字的值。

devices

组成卷组的设备或分区的逗号分隔列表。将此用作 **lvg** 模块 **pvs** 关键字的值。

第二个任务的内容应该是：

```
- name: Ensure Volume Groups Exist
  lvg:
    vg: "{{ item.name }}"
    pvs: "{{ item.devices }}"
    loop: "{{ volume_groups }}"
```

- 3.4. 更改第三个任务，以使用 **lvol** 模块。利用每个项目的关键字，设置卷组名称、逻辑卷名称和逻辑卷大小。第三个任务的内容现在是：

```
- name: Create each Logical Volume (LV) if needed
lvol:
  vg: "{{ item.vgroup }}"
  lv: "{{ item.name }}"
  size: "{{ item.size }}"
loop: "{{ logical_volumes }}"
```

- 3.5. 更改第四个任务以使用 `filesystem` 模块。配置该任务，以确保每个逻辑卷格式化为 XFS 文件系统。回想一下，逻辑卷与逻辑设备 `/dev/<volume group name>/<logical volume name>` 相关联。

第四个任务的内容应该是：

```
- name: Ensure XFS Filesystem exists on each LV
filesystem:
  dev: "/dev/{{ item.vgroup }}/{{ item.name }}"
  fstype: xfs
loop: "{{ logical_volumes }}"
```

- 3.6. 配置第五个任务以确保每个逻辑卷具有正确的存储容量。如果逻辑卷的容量增大，请务必强制扩展卷的文件系统。



警告

如果逻辑卷需要减小容量，则此任务将失败，因为 XFS 文件系统不支持收缩容量。

第五个任务的内容应该是：

```
- name: Ensure the correct capacity for each LV
lvol:
  vg: "{{ item.vgroup }}"
  lv: "{{ item.name }}"
  size: "{{ item.size }}"
  resizefs: yes
  force: yes
loop: "{{ logical_volumes }}"
```

- 3.7. 在第六个任务中，使用 `mount` 模块来确保每个逻辑卷挂载到对应的挂载路径，并在重新引导后保留。

第六个任务的内容应该是：

```
- name: Each Logical Volume is mounted
mount:
  path: "{{ item.mount_path }}"
  src: "/dev/{{ item.vgroup }}/{{ item.name }}"
  fstype: xfs
  state: mounted
loop: "{{ logical_volumes }}"
```

- 3.8. 执行 playbook 以在远程主机上创建逻辑卷。

```
[student@workstation system-review]$ ansible-playbook storage.yml
PLAY [Ensure Apache Storage Configuration]
*****
TASK [Gathering Facts]
*****
ok: [serverb.lab.example.com]

TASK [Correct partitions exist on /dev/vdb]
*****
changed: [serverb.lab.example.com] => (item={'number': 1, 'start': '1MiB', 'end': '257MiB'})

TASK [Ensure Volume Groups Exist]
*****
changed: [serverb.lab.example.com] => (item={'name': 'apache-vg', 'devices': '/dev/vdb1'})
...output omitted...

TASK [Create each Logical Volume (LV) if needed]
*****
changed: [serverb.lab.example.com] => (item={'name': 'content-lv', 'size': '64M',
'vgroup': 'apache-vg', 'mount_path': '/var/www'})
changed: [serverb.lab.example.com] => (item={'name': 'logs-lv', 'size': '128M',
'vgroup': 'apache-vg', 'mount_path': '/var/log/httpd'})

TASK [Ensure XFS Filesystem exists on each LV]
*****
changed: [serverb.lab.example.com] => (item={'name': 'content-lv', 'size': '64M',
'vgroup': 'apache-vg', 'mount_path': '/var/www'})
changed: [serverb.lab.example.com] => (item={'name': 'logs-lv', 'size': '128M',
'vgroup': 'apache-vg', 'mount_path': '/var/log/httpd'})

TASK [Ensure the correct capacity for each LV]
*****
ok: [serverb.lab.example.com] => (item={'name': 'content-lv', 'size': '64M',
'vgroup': 'apache-vg', 'mount_path': '/var/www'})
ok: [serverb.lab.example.com] => (item={'name': 'logs-lv', 'size': '128M',
'vgroup': 'apache-vg', 'mount_path': '/var/log/httpd'})

TASK [Each Logical Volume is mounted]
*****
changed: [serverb.lab.example.com] => (item={'name': 'content-lv', 'size': '64M',
'vgroup': 'apache-vg', 'mount_path': '/var/www'})
changed: [serverb.lab.example.com] => (item={'name': 'logs-lv', 'size': '128M',
'vgroup': 'apache-vg', 'mount_path': '/var/log/httpd'})

PLAY RECAP
*****
serverb.lab.example.com    : ok=7      changed=5      unreachable=0      failed=0
```

4. 创建一个 playbook 并在 web servers 主机组中执行，以使用 cron 模块来创建调度周期性 cron 作业的 /etc/cron.d/disk_usage crontab 文件。该作业应当以 devops 用户身份运

行，在 **Monday** 到 **Friday** 的 **09:00** 至 **16:59** 之间每隔两分钟运行一次。该作业应将当前磁盘使用量附加到 **/home/devops/disk_usage** 文件中。

- 4.1. 创建新 playbook **create_crontab_file.yml**，再添加开始 play 所需的行。它应当以 **webservers** 组中的受管主机为目标，并启用特权升级。

```
---
```

- name: Recurring cron job
 hosts: webservers
 become: true

- 4.2. 定义一项任务，以使用 **cron** 模块调度周期性 cron 作业。



注意

cron 模块提供了一个 **name** 选项，可以唯一地描述 crontab 文件条目并确保结果与预期相符。该描述将添加到 crontab 文件中。例如，如果您要使用 **state=absent** 删除 crontab 条目，则需要 **name** 选项。此外，如果设置了默认状态 **state=present**，**name** 选项会防止不顾现有条目而始终创建新的 crontab 条目。

```
tasks:
- name: Crontab file exists
  cron:
    name: Add date and time to a file
```

- 4.3. 将作业配置为在 **Monday** 到 **Friday** 的 **09:00** 至 **16:59** 之间每隔两分钟运行一次。

```
minute: "*/2"
hour: 9-16
weekday: 1-5
```

- 4.4. 使用 **cron_file** 参数来使用 **/etc/cron.d/disk_usage** crontab 文件，而不使用 **/var/spool/cron/** 中个别用户的 crontab。相对路径会将文件放到 **/etc/cron.d** 目录中。如果使用了 **cron_file** 参数，您还必须指定 **user** 参数。

```
user: devops
job: df >> /home/devops/disk_usage
cron_file: disk_usage
state: present
```

- 4.5. 完成时，该 playbook 应当如下所示。检查 playbook 的准确性。

```
---
```

- name: Recurring cron job
 hosts: webservers
 become: true

```
tasks:
- name: Crontab file exists
  cron:
```

```

name: Add date and time to a file
minute: "*/2"
hour: 9-16
weekday: 1-5
user: devops
job: df >> /home/devops/disk_usage
cron_file: disk_usage
state: present

```

4.6. 运行 playbook。

```
[student@workstation system-review]$ ansible-playbook create_crontab_file.yml
PLAY [Recurring cron job]
*****
TASK [Gathering Facts]
*****
ok: [serverb.lab.example.com]

TASK [Crontab file exists]
*****
changed: [serverb.lab.example.com]

PLAY RECAP
*****
serverb.lab.example.com : ok=2    changed=1    unreachable=0    failed=0
```

5. 创建一个 playbook 并在 `webservers` 主机组中执行，以使用 `linux-system-roles.network` 角色将 **172.25.250.40/24** IP 地址配置给备用网络接口 `ens4`。

- 5.1. 使用 `ansible-galaxy` 验证系统角色可用。若不可用，您需要安装 `rhel-system-roles` 软件包。

```
[student@workstation system-review]$ ansible-galaxy list
# /usr/share/ansible/roles
- linux-system-roles.kdump, (unknown version)
- linux-system-roles.network, (unknown version)
- linux-system-roles.postfix, (unknown version)
- linux-system-roles.selinux, (unknown version)
- linux-system-roles.timesync, (unknown version)
- rhel-system-roles.kdump, (unknown version)
- rhel-system-roles.network, (unknown version)
- rhel-system-roles.postfix, (unknown version)
- rhel-system-roles.selinux, (unknown version)
- rhel-system-roles.timesync, (unknown version)
# /etc/ansible/roles
[WARNING]: - the configured path /home/student/.ansible/roles does not exist.
```

- 5.2. 创建一个 playbook `network_playbook.yml`，其中包含一个以 `webservers` 主机组为目标的 play。将 `rhel-system-roles.network` 角色包含到 play 的 `roles` 部分。

```

---
- name: NIC Configuration
  hosts: webservers

  roles:
    - rhel-system-roles.network

```

5.3. 创建 **group_vars/webservers** 子目录。

```
[student@workstation system-review]$ mkdir -pv group_vars/webservers
mkdir: created directory 'group_vars'
mkdir: created directory 'group_vars/webservers'
```

5.4. 创建一个新文件 **network.yml** 来定义角色变量。由于这些变量值将应用到 **webservers** 主机组中的主机，您需要在 **group_vars/webservers** 目录中创建该文件。添加变量定义以支持配置 **ens4** 网络接口。该文件现在包含：

```

[student@workstation system-review]$ vi group_vars/webservers/network.yml
---
network_connections:
  - name: ens4
    type: ethernet
    ip:
      address:
        - 172.25.250.40/24

```

5.5. 运行 playbook，以配置第二网络接口。

```
[student@workstation system-review]$ ansible-playbook network_playbook.yml

PLAY [NIC Configuration]
*****
TASK [Gathering Facts]
*****
ok: [serverb.lab.example.com]

TASK [rhel-system-roles.network : Check which services are running]
*****
ok: [serverb.lab.example.com]

TASK [rhel-system-roles.network : Check which packages are installed]
*****
ok: [serverb.lab.example.com]

TASK [rhel-system-roles.network : Print network provider]
*****
ok: [serverb.lab.example.com] => {
    "msg": "Using network provider: nm"
}
```

```
TASK [rhel-system-roles.network : Install packages]
*****
skipping: [serverb.lab.example.com]

TASK [rhel-system-roles.network : Enable network service]
*****
ok: [serverb.lab.example.com]

TASK [rhel-system-roles.network : Configure networking connection profiles]
*****
[WARNING]: [002] <info> #0, state:None persistent_state:present, 'ens4': add
connection
ens4, 38d63afed-e610-4929-ba1b-1d38413219fb

changed: [serverb.lab.example.com]

TASK [rhel-system-roles.network : Re-test connectivity]
*****
ok: [serverb.lab.example.com]

PLAY RECAP
*****
serverb.lab.example.com      : ok=7      changed=1      unreachable=0      failed=0
```

5.6. 验证 ens4 网络接口是否使用了 **172.25.250.40** IP 地址。配置 IP 地址可能需要最多一分钟。

```
[student@workstation system-review]$ ansible webservers -m setup \
> -a 'filter=ansible_ens4'
serverb.lab.example.com | SUCCESS => {
    "ansible_facts": {
        "ansible_ens4": {
            ...output omitted...
            "ipv4": {
                "address": "172.25.250.40",
                "broadcast": "172.25.250.255",
                "netmask": "255.255.255.0",
                "network": "172.25.250.0"
            },
            ...output omitted...
        }
    }
}
```

6. 在 workstation 上运行 **lab system-review grade** 以评测您的工作。

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

完成

从 workstation，运行 **lab system-review finish** 脚本来清理本实验中创建的资源。

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

总结

在本章中，您学到了：

- 该**yum_repository** module 在托管主机上配置 Yum 存储库。对于使用公钥的存储库，您可以通
过 **rpm_key** 模块验证密钥是否可用。
- **user** 和 **group** 模块分别在受管主机上创建用户和组。您可以使用 **authorized_key** 模块配置
用户的授权密钥。
- 可以通过 **cron** 模块在受管主机上配置 cron 作业。
- Ansible 通过 **lvg** 和 **lvol** 模块支持逻辑卷的配置。**parted** 和 **filesystem** 模块分别支持设备
分区和文件系统创建。
- 红帽企业 Linux 8 包含网络系统角色，支持在受管主机上配置网络接口。

章 11

总复习：使用 ANSIBLE 实现自动化

目标

通过安装、优化和配置 Ansible 来管理受管主机，展示本课程中所学到的技能。

章节

- 总复习

实验

- 实验：部署 Ansible
- 实验：创建 Playbook
- 实验：创建角色和使用动态清单

总复习

培训目标

学完本章节后，您应该能够展示在 红帽系统管理 III：Linux 自动化 中所学知识和技能的熟练程度。

复习红帽系统管理 III：Linux 自动化

在开始进行本课程的总复习之前，您应该熟悉各章节所涵盖的主题。

请尽管向讲师寻求有关这些主题的额外指导或解释。

第 1 章 介绍 Ansible

描述 Ansible 基本概念和使用方法，并安装红帽 Ansible 引擎。

- 描述使用 Ansible 自动化 Linux 管理任务的动机、基本的 Ansible 概念，以及 Ansible 的基本架构。
- 在控制节点上安装 Ansible 并描述社区 Ansible 和红帽 Ansible 引擎之间的区别。

第 2 章 部署 Ansible

配置 Ansible 以管理主机和运行临时 Ansible 命令。

- 描述 Ansible 清单概念并管理静态清单文件。
- 描述 Ansible 配置文件的位置、Ansible 如何选择这些文件，并编辑它们以对默认设置应用更改。
- 使用临时命令运行单个 Ansible 自动化任务，并解释临时命令的一些用例。

第 3 章 实施 Playbook

编写简单的 Ansible Playbook 并运行它，以在多个主机上自动执行任务。

- 编写基本的 Ansible Playbook，再使用 **ansible-playbook** 命令运行它。
- 编写一个使用多个 play 和各 play 特权升级的 playbook。
- 有效使用 **ansible-doc** 学习如何使用新模块来实施 play 任务。

第 4 章 管理变量和事实

编写使用变量的 playbook 来简化 playbook 的管理，以引用有关受管主机的信息。

- 创建和引用影响特定主机或主机组、play 或全局环境的变量，并描述变量优先级的工作方式。
- 使用 Ansible Vault 加密敏感变量，并运行引用 Vault 加密变量文件的 playbook。
- 使用 Ansible 事实引用有关受管主机的数据，并在受管主机上配置自定义事实。

第 5 章 实施任务控制

在 Ansible Playbook 中管理任务控制、处理程序和任务错误。

- 实施循环来编写高效的任务，从而控制运行任务的时间。
- 实施仅在另一个任务更改托管主机时运行的任务。
- 控制任务失败时发生的情况，以及导致任务失败的条件。

第 6 章 在被管理节点上创建文件或目录

在 Ansible 管理的主机上部署、管理和调整文件。

- 在受管主机上创建、安装、编辑和删除文件，以及管理这些文件的权限、所有权、SELinux 上下文和其他特征。
- 使用 Jinja2 模板自定义文件，并将它们部署到受管主机。

第 7 章 管理大项目

编写针对更大、更复杂的项目进行优化的 playbook。

- 编写复杂的主机模式，以便高效地为 play 或临时命令选择主机。
- 描述动态清单的性质和用途，然后安装和使用现有脚本作为 Ansible 动态清单来源。
- 调整 Ansible 向受管主机开放的同时连接数量，以及 Ansible 如何通过 play 任务处理受管主机组。
- 通过无条件或基于条件测试导入或包含来自外部文件的 playbook 和任务来管理大型 playbook。

第 8 章 利用角色简化 Playbook

使用 Ansible 角色以更快地开发 playbook 并重用 Ansible 代码。

- 描述什么是角色、如何构建角色，以及如何在 playbook 中使用角色。
- 在 playbook 的项目目录中创建一个角色，并将其作为 playbook 中某个 play 的一部分来运行。
- 从 Ansible Galaxy 或其他来源（例如 Git 存储库）中选择和检索角色，并在您的 playbook 中使用它们。
- 编写利用红帽企业 Linux 系统角色执行标准操作的 playbook。

第 9 章 对 Ansible 进行故障排除

对 playbook 和受管主机进行故障排除。

- 对新 playbook 的一般问题进行故障排除，并修复这些问题。
- 对运行 playbook 时受管主机上的故障进行故障排除。

第 10 章 自动执行 Linux 管理任务

利用 Ansible，实现常见 Linux 系统管理任务自动化。

- 订阅系统，配置软件通道和存储库，以及管理受管主机上的 RPM 软件包。
- 管理 Linux 用户和组，配置 SSH，以及修改受管主机上的 Sudo 配置。
- 管理服务启动，使用 at、cron 和 systemd 调度进程，重新引导，以及控制受管主机上的默认启动目标。

- 对存储设备进行分区，配置 LVM，格式化分区或逻辑卷，挂载文件系统，以及添加交换文件或空间。

▶ 开放研究实验

部署 ANSIBLE

在本复习中，您将在 **workstation** 上安装 Ansible，将它用作控制节点，再对它进行配置以用于连接受管主机 **servera** 和 **serverb**。使用临时命令对受管主机执行操作。

成果

您应能够：

- 安装 Ansible。
- 使用临时命令对受管主机执行操作。

说明

在 **workstation** 上安装和配置 Ansible。演示您能够构建条件列表中指定的临时命令，以修改受管主机并验证修改可以正常起作用：

- 在 **workstation** 上安装 Ansible，使它可以充当控制节点。
- 在控制节点上，创建清单文件 **/home/student/review-deploy/inventory**，其含有一个名为 **dev** 的组。此组应当由受管主机 **servera.lab.example.com** 和 **serverb.lab.example.com** 组成。
- 在 **/home/student/review-deploy/ansible.cfg** 中创建 Ansible 配置文件。该配置文件应指向清单文件 **/home/student/review-deploy/inventory**。
- 利用特权升级执行临时命令，以修改 **servera** 和 **serverb** 上 **/etc/motd** 文件的内容，使其包含字符串 **Managed by Ansible\n**。使用 **devops** 作为远程用户。
- 执行临时命令，以验证 **servera** 和 **serverb** 上 **/etc/motd** 文件的内容相同。

评估

在 **workstation** 上，运行 **lab review-deploy grade** 命令以确认本练习是否成功。更正报告的所有错误并重新运行命令，直到成功为止。

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

完成

在 **workstation** 上，运行 **lab review-deploy finish** 命令来清理本练习。

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

本实验到此结束。

► 解决方案

部署 ANSIBLE

在本复习中，您将在 `workstation` 上安装 Ansible，将它用作控制节点，再对它进行配置以用于连接受管主机 `servera` 和 `serverb`。使用临时命令对受管主机执行操作。

成果

您应能够：

- 安装 Ansible。
- 使用临时命令对受管主机执行操作。

说明

在 `workstation` 上安装和配置 Ansible。演示您能够构建条件列表中指定的临时命令，以修改受管主机并验证修改可以正常起作用：

- 在 `workstation` 上安装 Ansible，使它可以充当控制节点。
- 在控制节点上，创建清单文件 `/home/student/review-deploy/inventory`，其含有一个名为 `dev` 的组。此组应当由受管主机 `servera.lab.example.com` 和 `serverb.lab.example.com` 组成。
- 在 `/home/student/review-deploy/ansible.cfg` 中创建 Ansible 配置文件。该配置文件应指向清单文件 `/home/student/review-deploy/inventory`。
- 利用特权升级执行临时命令，以修改 `servera` 和 `serverb` 上 `/etc/motd` 文件的内容，使其包含字符串 `Managed by Ansible\n`。使用 `devops` 作为远程用户。
- 执行临时命令，以验证 `servera` 和 `serverb` 上 `/etc/motd` 文件的内容相同。

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

在 `workstation` 上，运行 `lab review-deploy start` 命令。此脚本将确保受管主机 `servera` 和 `serverb` 可在网络上访问。该脚本在 `student` 用户的主目录中创建名为 `review-deploy` 的实验子目录。

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

2. 在 `workstation` 上安装 Ansible，使它可以充当控制节点。

```
[student@workstation ~]$ sudo yum install ansible
[sudo] password for student:
Loaded plugins: langpacks, search-disabled-repos
Resolving Dependencies
--> Running transaction check
...output omitted...
Is this ok [y/d/N]: y
...output omitted...
```

3. 在控制节点上，创建清单文件 `/home/student/review-deploy/inventory`，其含有一个名为 `dev` 的组。此组应当由受管主机 `servera.lab.example.com` 和 `serverb.lab.example.com` 组成。

```
[dev]
servera.lab.example.com
serverb.lab.example.com
```

4. 在 `/home/student/review-deploy/ansible.cfg` 中创建 Ansible 配置文件。该配置文件应引用 `/home/student/review-deploy/inventory` 清单文件。

添加以下条目以将清单文件 `./inventory` 配置为清单来源。保存更改并退出文本编辑器。

```
[defaults]
inventory=./inventory
```

5. 利用特权升级执行临时命令，以修改 `servera` 和 `serverb` 上 `/etc/motd` 文件的内容，使其包含字符串 `Managed by Ansible\n`。使用 `devops` 作为远程用户。

```
[student@workstation review-deploy]$ ansible dev -m copy \
> -a 'content="Managed by Ansible\n" dest=/etc/motd' -b -u devops
servera.lab.example.com | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": true,
    "checksum": "4458b979ede3c332f8f2128385df4ba305e58c27",
    "dest": "/etc/motd",
    "gid": 0,
    "group": "root",
    "md5sum": "65a4290ee5559756ad04e558b0e0c4e3",
    "mode": "0644",
    "owner": "root",
    "secontext": "unconfined_u:object_r:etc_t:s0",
    "size": 19,
    "src": "/home/devops/.ansible/tmp/...output omitted...",
    "state": "file",
    "uid": 0
}
serverb.lab.example.com | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": true,
    "checksum": "4458b979ede3c332f8f2128385df4ba305e58c27",
    "dest": "/etc/motd",
    "gid": 0,
    "group": "root",
    "md5sum": "65a4290ee5559756ad04e558b0e0c4e3",
    "mode": "0644",
    "owner": "root",
    "secontext": "system_u:object_r:etc_t:s0",
    "size": 19,
```

```
"src": "/home/devops/.ansible/tmp/...output omitted...",  
"state": "file",  
"uid": 0  
}
```

6. 执行临时命令，以验证 servera 和 serverb 上 /etc/motd 文件的内容相同。

```
[student@workstation review-deploy]$ ansible dev -m command -a "cat /etc/motd"  
servera.lab.example.com | CHANGED | rc=0 >>  
Managed by Ansible  
  
serverb.lab.example.com | CHANGED | rc=0 >>  
Managed by Ansible
```

评估

在 workstation 上，运行 **lab review-deploy grade** 命令以确认本练习是否成功。更正报告的所有错误并重新运行命令，直到成功为止。

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

完成

在 workstation 上，运行 **lab review-deploy finish** 命令来清理本练习。

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

▶ 开放研究实验

创建 PLAYBOOK

在本复习中，您将在 Ansible 项目目录 `/home/student/review-playbooks` 中创建三个 playbook。一个 playbook 将确保在应成为 FTP 客户端的系统上安装 lftp，另一个 playbook 将确保在应成为 FTP 服务器的系统上安装和配置 vsftpd，最后一个 playbook (`site.yml`) 将用于运行另外两个 playbook。

成果

您应能够：

- 创建和执行用于在受管主机上执行任务的 playbook。
- 在 playbook 中使用 Jinja2 模板、变量和处理程序。

说明

在 `review-playbooks/inventory` 中创建静态清单，使得 `serverc.lab.example.com` 位于组 `ftpclients` 中，并且 `serverb.lab.example.com` 和 `serverd.lab.example.com` 位于组 `ftpservers` 中。创建 `review-playbooks/ansible.cfg` 文件，以将您的 Ansible 项目配置为使用该清单。您可能会发现查看系统的 `/etc/ansible/ansible.cfg` 文件以获取语法相关的帮助很有用。

配置您的 Ansible 项目以使用远程用户 `devops` 及 `sudo` 特权升级方法连接清单中的主机。您已配置了用于以 `devops` 用户身份登录的 SSH 密钥。`devops` 用户使用 `sudo` 进行特权升级时不需要输入密码。

在 `review-playbooks` 目录中创建名为 `ftpclients.yml` 的 playbook，它将包含一个以清单组 `ftpclients` 中的主机为目标的 play。该 playbook 必须确保 lftp 软件包已经安装。

在 `review-playbooks` 目录中创建名为 `ansible-vsftpd.yml` 的第二个 playbook，它将包含一个以清单组 `ftpservers` 中的主机为目标的 play。它必须满足下列要求：

- 您已有从 Jinja2 模板生成的 `vsftpd` 配置文件。为模板创建目录 `review-playbooks/templates`，并将提供的 `vsftpd.conf.j2` 文件复制到其中。创建目录 `review-playbooks/vars`。将提供的 `defaults-template.yml` 文件复制到 `review-playbooks/vars` 目录，该文件中含有在部署时用于完成该模板的默认变量设置。
- 创建变量文件 `review-playbooks/vars/vars.yml`，它将设置三个变量：

变量	值
<code>vsftpd_package</code>	<code>vsftpd</code>
<code>vsftpd_service</code>	<code>vsftpd</code>
<code>vsftpd_config_file</code>	<code>/etc/vsftpd/vsftpd.conf</code>

- 在 `ansible-vsftpd.yml` playbook 中，使用 `vars_files` 在您的 play 中包含 `review-playbooks/vars` 目录中的变量文件。

· 在 **ansible-vsftpd.yml** play 中，创建以下任务：

1. 确保安装了变量 `vsftpd_package` 所列出的软件包。
2. 确保变量 `vsftpd_service` 列出的服务已启动并且已启用在引导时启动。
3. 使用 `template` 模块，将 `templates/vsftpd.conf.j2` 模板部署到 `vsftpd_config_file` 变量定义的位置。该文件应当归用户 `root`、组 `root` 所有，且具有八进制文件权限 0600 以及 SELinux 类型 `etc_t`。通知处理程序在此任务造成更改时重启 `vsftpd`。
4. 确保已安装 `firewalld` 软件包，并且已启动并启用其服务。确保已将 `firewalld` 配置为立即且永久允许连接 `ftp` 服务。被动 `FTP` 数据连接的临时端口范围已设置为 21000-21020 TCP。您还需要通过防火墙允许此范围。

· 在您的 **ansible-vsftpd.yml** playbook 中，创建一个处理程序以在获得通知时重启变量 `vsftpd_service` 所列出的服务。

在 **review-playbooks** 目录中，创建第三个 playbook **site.yml**。此 playbook 应当仅导入其他两个 playbook。

通过命名所有 play 和任务来遵循推荐的 playbook 实践。使用适当的模块编写 playbook，并确保它们可以安全地重新运行。Playbook 不应对系统造成不必要的更改。

使用 **ansible-doc** 命令帮助您查找模块及其使用方法的信息。

完成后，请使用 **ansible-playbook site.yml** 检查您的作业，然后再运行评测脚本。您也可以单独运行各个 playbook，确保它们功能正常。



重要

如果遇到与 `site.yml` playbook 相关的问题，请确保 `ansible-vsftpd.yml` 和 `ftpclients.yml` 使用一致的缩进。

评估

在 `workstation` 上，以 `student` 用户身份运行 `lab review-playbooks grade` 命令，以确认是否成功完成本练习。更正报告的所有错误并重新运行脚本，直到成功为止。

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

完成

运行 `lab review-playbooks finish` 命令，以清理 `serverb`、`serverc` 和 `serverd` 上的实验任务。

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

本实验到此结束。

► 解决方案

创建 PLAYBOOK

在本复习中，您将在 Ansible 项目目录 `/home/student/review-playbooks` 中创建三个 playbook。一个 playbook 将确保在应成为 FTP 客户端的系统上安装 lftp，另一个 playbook 将确保在应成为 FTP 服务器的系统上安装和配置 vsftpd，最后一个 playbook (`site.yml`) 将用于运行另外两个 playbook。

成果

您应能够：

- 创建和执行用于在受管主机上执行任务的 playbook。
- 在 playbook 中使用 Jinja2 模板、变量和处理程序。

说明

在 `review-playbooks/inventory` 中创建静态清单，使得 `serverc.lab.example.com` 位于组 `ftpclients` 中，并且 `serverb.lab.example.com` 和 `serverd.lab.example.com` 位于组 `ftpservers` 中。创建 `review-playbooks/ansible.cfg` 文件，以将您的 Ansible 项目配置为使用该清单。您可能会发现查看系统的 `/etc/ansible/ansible.cfg` 文件以获取语法相关的帮助很有用。

配置您的 Ansible 项目以使用远程用户 `devops` 及 `sudo` 特权升级方法连接清单中的主机。您已配置了用于以 `devops` 用户身份登录的 SSH 密钥。`devops` 用户使用 `sudo` 进行特权升级时不需要输入密码。

在 `review-playbooks` 目录中创建名为 `ftpclients.yml` 的 playbook，它将包含一个以清单组 `ftpclients` 中的主机为目标的 play。该 playbook 必须确保 lftp 软件包已经安装。

在 `review-playbooks` 目录中创建名为 `ansible-vsftpd.yml` 的第二个 playbook，它将包含一个以清单组 `ftpservers` 中的主机为目标的 play。它必须满足下列要求：

- 您已有从 Jinja2 模板生成的 `vsftpd` 配置文件。为模板创建目录 `review-playbooks/templates`，并将提供的 `vsftpd.conf.j2` 文件复制到其中。创建目录 `review-playbooks/vars`。将提供的 `defaults-template.yml` 文件复制到 `review-playbooks/vars` 目录，该文件中含有在部署时用于完成该模板的默认变量设置。
- 创建变量文件 `review-playbooks/vars/vars.yml`，它将设置三个变量：

变量	值
<code>vsftpd_package</code>	<code>vsftpd</code>
<code>vsftpd_service</code>	<code>vsftpd</code>
<code>vsftpd_config_file</code>	<code>/etc/vsftpd/vsftpd.conf</code>

- 在 `ansible-vsftpd.yml` playbook 中，使用 `vars_files` 在您的 play 中包含 `review-playbooks/vars` 目录中的变量文件。

· 在 **ansible-vsftpd.yml** play 中，创建以下任务：

1. 确保安装了变量 `vsftpd_package` 所列出的软件包。
 2. 确保变量 `vsftpd_service` 列出的服务已启动并且已启用在引导时启动。
 3. 使用 `template` 模块，将 `templates/vsftpd.conf.j2` 模板部署到 `vsftpd_config_file` 变量定义的位置。该文件应当归用户 `root`、组 `root` 所有，且具有八进制文件权限 0600 以及 SELinux 类型 `etc_t`。通知处理程序在此任务造成更改时重启 `vsftpd`。
 4. 确保已安装 `firewalld` 软件包，并且已启动并启用其服务。确保已将 `firewalld` 配置为立即且永久允许连接 `ftp` 服务。被动 `FTP` 数据连接的临时端口范围已设置为 21000-21020 TCP。您还需要通过防火墙允许此范围。
- 在您的 **ansible-vsftpd.yml** playbook 中，创建一个处理程序以在获得通知时重启变量 `vsftpd_service` 所列出的服务。

在 **review-playbooks** 目录中，创建第三个 playbook **site.yml**。此 playbook 应当仅导入其他两个 playbook。

通过命名所有 play 和任务来遵循推荐的 playbook 实践。使用适当的模块编写 playbook，并确保它们可以安全地重新运行。Playbook 不应对系统造成不必要的更改。

使用 **ansible-doc** 命令帮助您查找模块及其使用方法的信息。

完成后，请使用 **ansible-playbook site.yml** 检查您的作业，然后再运行评测脚本。您也可以单独运行各个 playbook，确保它们功能正常。



重要

如果遇到与 `site.yml` playbook 相关的问题，请确保 `ansible-vsftpd.yml` 和 `ftpclients.yml` 使用一致的缩进。

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

在 `workstation` 上，运行 `lab review-playbooks start` 命令。

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

2. 在 `workstation` 上，以 `student` 用户身份创建清单文件 `/home/student/review-playbooks/inventory`，使 `serverc.lab.example.com` 包含在 `ftpclients` 组中，并且 `serverb.lab.example.com` 和 `serverd.lab.example.com` 包含在 `ftpservers` 组中。

- 2.1. 将目录更改到由设置脚本创建的 Ansible 项目目录 `/home/student/review-playbooks`。

```
[student@workstation ~]$ cd ~/review-playbooks
```

- 2.2. 使用以下条目填充 `inventory` 文件，然后保存并退出。

```
[ftpservers]
serverb.lab.example.com
serverd.lab.example.com
```

```
[ftpclients]
serverc.lab.example.com
```

3. 创建 Ansible 配置文件 **/home/student/review-playbooks/ansible.cfg**, 并在文件中填充必要的条目以满足下列要求：

- 将 Ansible 项目配置为使用新创建的清单
- 以 **devops** 用户身份连接受管主机
- 以 **root** 用户身份通过 **sudo** 来利用特权升级
- 默认为每个任务升级特权

```
[defaults]
remote_user = devops
inventory = ./inventory

[privilegeEscalation]
become_user = root
become_method = sudo
become = true
```

4. 创建 playbook **/home/student/review-playbooks/ftpclients.yml**, 使其包含以 **ftpclients** 清单组为目标的 play, 并且确保安装了 lftp 软件包。

```
---
- name: Ensure FTP Client Configuration
  hosts: ftpclients

  tasks:
    - name: latest version of lftp is installed
      yum:
        name: lftp
        state: latest
```

5. 将提供的 vsftpd 配置文件 **vsftpd.conf.j2** 放到 **templates** 子目录中。

- 5.1. 创建 **templates** 子目录。

```
[student@workstation review-playbooks]$ mkdir -v templates
mkdir: created directory 'templates'
```

- 5.2. 将 **vsftpd.conf.j2** 文件移到新建的 **templates** 子目录。

```
[student@workstation review-playbooks]$ mv -v vsftpd.conf.j2 templates
renamed 'vsftpd.conf.j2' -> 'templates/vsftpd.conf.j2'
```

6. 将提供的 `defaults-template.yml` 配置文件放到 `vars` 子目录中。

6.1. 创建 `vars` 子目录。

```
[student@workstation review-playbooks]$ mkdir -v vars
mkdir: created directory 'vars'
```

6.2. 将 `defaults-template.yml` 文件移到新建的 `vars` 子目录。

```
[student@workstation review-playbooks]$ mv -v defaults-template.yml vars/
renamed 'defaults-template.yml' -> 'vars/defaults-template.yml'
```

7. 在 `vars` 子目录中创建 `vars.yml` 变量定义文件，以定义下列三个变量和对应的值。

变量	值
<code>vsftpd_package</code>	<code>vsftpd</code>
<code>vsftpd_service</code>	<code>vsftpd</code>
<code>vsftpd_config_file</code>	<code>/etc/vsftpd/vsftpd.conf</code>

```
vsftpd_package: vsftpd
vsftpd_service: vsftpd
vsftpd_config_file: /etc/vsftpd/vsftpd.conf
```

8. 使用之前创建的 Jinja2 模板和变量定义文件，创建第二个 playbook `/home/student/review-playbooks/ansible-vsftpd.yml`，以在 `ftpservers` 清单组中的主机上配置 vsftpd 服务。

```
---
- name: FTP server is installed
  hosts:
    - ftptests
  vars_files:
    - vars/defaults-template.yml
    - vars/vars.yml

  tasks:
    - name: Packages are installed
      yum:
        name: "{{ vsftpd_package }}"
        state: present

    - name: Ensure service is started
      service:
        name: "{{ vsftpd_service }}"
        state: started
        enabled: true

    - name: Configuration file is installed
      template:
```

```

src: templates/vsftpd.conf.j2
dest: "{{ vsftpd_config_file }}"
owner: root
group: root
mode: 0600
setype: etc_t
notify: restart vsftpd

- name: firewalld is installed
  yum:
    name: firewalld
    state: present

- name: firewalld is started and enabled
  service:
    name: firewalld
    state: started
    enabled: yes

- name: FTP port is open
  firewalld:
    service: ftp
    permanent: true
    state: enabled
    immediate: yes

- name: FTP passive data ports are open
  firewalld:
    port: 21000-21020/tcp
    permanent: yes
    state: enabled
    immediate: yes

handlers:
- name: restart vsftpd
  service:
    name: "{{ vsftpd_service }}"
    state: restarted

```

9. 创建第三个 playbook `/home/student/review-playbooks/site.yml`, 在其中包含前面创建的两个 playbook `ftpclients.yml` 和 `ansible-vsftpd.yml` 中的 play。

```

---
# FTP Servers playbook
- import_playbook: ansible-vsftpd.yml

# FTP Clients playbook
- import_playbook: ftpclients.yml

```

10. 执行 `/home/student/review-playbooks/site.yml` playbook, 以验证它是否会向受管的主机执行所需的任务。

```
[student@workstation review-playbooks]$ ansible-playbook site.yml
```