

色是否安装在了其中一个目录中，将使用第一个匹配项。以下 **use-role.yml** playbook 引用了 **redis_prod** 和 **geerlingguy.redis** 角色：

```
[user@host project]$ cat use-role.yml
---
- name: use redis_prod for Prod machines
  hosts: redis_prod_servers
  remote_user: devops
  become: true
  roles:
    - redis_prod

- name: use geerlingguy.redis for Dev machines
  hosts: redis_dev_servers
  remote_user: devops
  become: true
  roles:
    - geerlingguy.redis
```

此 playbook 使不同版本的 **geerlingguy.redis** 角色应用到生产和开发服务器。借助这种方式，可以对角色更改进行系统化测试和集成，然后再部署到生产服务器上。如果角色的近期更改造成了问题，则借助版本控制来开发角色，您就能回滚到过去某一个稳定的角色版本。



参考文献

Ansible Galaxy —Ansible 文档

https://docs.ansible.com/ansible/latest/reference_appendices/galaxy.html

► 指导练习

使用 ANSIBLE GALAXY 部署角色

在本练习中，您将使用 Ansible Galaxy 下载和安装 Ansible 角色。

成果

您应能够：

- 通过创建角色文件来指定 playbook 的角色依赖项
- 安装角色文件中指定的角色
- 使用 **ansible-galaxy** 命令列出角色

场景概述

您的组织将自定义文件放到所有主机上的 **/etc/skel** 目录中。因此，新的用户帐户将配置有标准化的组织专用 Bash 环境。

您将对负责部署 Bash 环境框架文件的 Ansible 角色的开发版本进行测试。

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 role-galaxy start** 命令。这会创建工作目录 **/home/student/role-galaxy**，并为它填充 Ansible 配置文件和主机清单。

```
[student@workstation ~]$ lab role-galaxy start
```

► 1. 更改到 **role-galaxy** 工作目录。

```
[student@workstation ~]$ cd ~/role-galaxy  
[student@workstation role-galaxy]$
```

► 2. 要测试配置框架文件的 Ansible 角色，请将角色规格添加到角色文件中。

启动您首选的文本编辑器，并在 **roles** 子目录中创建名为 **requirements.yml** 的文件。角色的 Git 存储库的 URL 是 **git@workstation.lab.example.com:student/bash_env**。要查看角色对生产主机行为的影响，可使用存储库的 **master** 分支。将角色的本地名称设为 **student.bash_env**。

roles/requirements.yml 现在应包含以下内容：

```
---
# requirements.yml

- src: git@workstation.lab.example.com:student/bash_env
  scm: git
  version: master
  name: student.bash_env
```

- 3. 使用 **ansible-galaxy** 命令，以处理您刚才创建的角色文件并安装 **student.bash_env** 角色。

3.1. 在安装该角色前，显示 **roles** 子目录的内容以进行比较。

```
[student@workstation role-galaxy]$ ls roles/
requirements.yml
```

3.2. 使用 Ansible Galaxy 下载并安装 **roles/requirements.yml** 文件中列出的角色。确保任何下载的所有角色存储在 **roles** 子目录中。

```
[student@workstation role-galaxy]$ ansible-galaxy install -r \
> roles/requirements.yml -p roles
- extracting student.bash_env to /home/student/role-galaxy/roles/student.bash_env
- student.bash_env (master) was installed successfully
```

3.3. 在角色安装后，显示 **roles** 子目录。确认它具有一个名为 **student.bash_env** 的新子目录，与 YAML 文件中指定的 **name** 值相符。

```
[student@workstation role-galaxy]$ ls roles/
requirements.yml  student.bash_env
```

3.4. 尝试不带选项使用 **ansible-galaxy** 命令来列出项目角色：

```
[student@workstation role-galaxy]$ 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.
```

由于您之前将 **-p** 选项与 **ansible-galaxy install** 命令搭配使用，**student.bash_env** 角色没有安装到默认的位置。使用 **-p** 选项和 **ansible-galaxy list** 命令来列出下载的角色：

```
[student@workstation role-galaxy]$ ansible-galaxy list -p roles
# /home/student/role-galaxy/roles
- student.bash_env, master
...output omitted...
[WARNING]: - the configured path /home/student/.ansible/roles does not exist.
```

 **注意**

/home/student/.ansible/roles 目录在您的默认 **roles_path** 中，但由于您没有尝试不带 **-p** 选项来安装角色，因此 **ansible-galaxy** 还没有创建该目录。

- 4. 创建名为 **use-bash_env-role.yml** 的 playbook，它将使用 **student.bash_env** 角色。playbook 的内容应当与下方所示相符：

```
---
- name: use student.bash_env role playbook
  hosts: devservers
  vars:
    default_prompt: '[\u on \h in \W dir]\$ '
  pre_tasks:
    - name: Ensure test user does not exist
      user:
        name: student2
        state: absent
        force: yes
        remove: yes

  roles:
    - student.bash_env

  post_tasks:
    - name: Create the test user
      user:
        name: student2
        state: present
        password: "{{ 'redhat' | password_hash('sha512', 'mysecretsalt') }}"
```

为了查看配置更改的效果，必须创建一个新用户帐户。Playbook 的 **pre_tasks** 和 **post_tasks** 部分将确保每次执行该 playbook 时创建 **student2** 用户帐户。在 playbook 执行后，通过密码 **redhat** 来访问 **student2** 帐户。

 **注意**

user2 密码是使用过滤器生成的。过滤器获取数据并进行修改；此处，通过将 **redhat** 传递给 **password_hash** 模块来修改该字符串。过滤器是本课程未予阐述的高级主题。

- 5. 运行 playbook。`student.bash_env` 角色在受管主机上的 `/etc/skel` 中创建标准的模板配置文件。它创建的文件包括 `.bashrc`、`.bash_profile` 和 `.vimrc`。

```
[student@workstation role-galaxy]$ ansible-playbook use-bash_env-role.yml

PLAY [use student.bash_env role playbook] *****

TASK [Gathering Facts] *****
ok: [servera.lab.example.com]

TASK [Ensure test user does not exist] *****
ok: [servera.lab.example.com]

TASK [student.bash_env : put away .bashrc] *****
changed: [servera.lab.example.com]

TASK [student.bash_env : put away .bash_profile] *****
ok: [servera.lab.example.com]

TASK [student.bash_env : put away .vimrc] *****
changed: [servera.lab.example.com]

TASK [Create the test user] *****
changed: [servera.lab.example.com]

PLAY RECAP *****
servera.lab.example.com      : ok=6      changed=4      unreachable=0      failed=0
```

- 6. 以 `student2` 用户身份，使用 SSH 连接到 `servera`。观察 `student2` 用户的自定义提示符，然后断开与 `servera` 的连接。

```
[student@workstation role-galaxy]$ ssh student2@servera
Activate the web console with: systemctl enable --now cockpit.socket

[student2 on servera in ~ dir]$ exit
logout
Connection to servera closed.
[student@workstation role-galaxy]$
```

- 7. 使用 `student.bash_env` 角色的开发版本执行 playbook。

该角色的开发版本位于 Git 存储库的 `dev` 分支中。该角色的开发版本使用一个新变量 `prompt_color`。在执行 playbook 前，将 `prompt_color` 变量添加到 playbook 的 `vars` 部分，并将其值设为 `blue`。

- 7.1. 更新 `roles/requirements.yml` 文件，并将 `version` 值设为 `dev`。`roles/requirements.yml` 文件现在包含：

```
---  
# requirements.yml  
  
- src: git@workstation.lab.example.com:student/bash_env  
  scm: git  
  version: dev  
  name: student.bash_env
```

7.2. 运行 **ansible-galaxy install** 命令，使用更新的角色文件来安装角色。使用 **--force** 选项，将角色的现有 **master** 版本覆盖为角色的 **dev** 版本。

```
[student@workstation role-galaxy]$ ansible-galaxy install \  
> -r roles/requirements.yml --force -p roles  
- changing role student.bash_env from master to dev  
- extracting student.bash_env to /home/student/role-galaxy/roles/student.bash_env  
- student.bash_env (dev) was installed successfully
```

7.3. 编辑 **use-bash_env-role.yml** 文件。将 **prompt_color** 变量添加到 playbook 的 **vars** 部分，并将值设为 **blue**。该文件现在包含：

```
---  
- name: use student.bash_env role playbook  
  hosts: devservers  
  vars:  
    prompt_color: blue  
    default_prompt: '[\u on \h in \W dir]\$ '  
  pre_tasks:  
...output omitted...
```

7.4. 执行 **use-bash_env-role.yml** playbook。

```
[student@workstation role-galaxy]$ ansible-playbook use-bash_env-role.yml  
  
PLAY [use student.bash_env role playbook] *****  
  
TASK [Gathering Facts] *****  
ok: [servera.lab.example.com]  
  
TASK [Ensure test user does not exist] *****  
changed: [servera.lab.example.com]  
  
TASK [student.bash_env : put away .bashrc] *****  
changed: [servera.lab.example.com]  
  
TASK [student.bash_env : put away .bash_profile] *****  
changed: [servera.lab.example.com]  
  
TASK [student.bash_env : put away .vimrc] *****  
okay: [servera.lab.example.com]  
  
TASK [Create the test user] *****
```

```
changed: [servera.lab.example.com]

PLAY RECAP ****
servera.lab.example.com : ok=6    changed=4    unreachable=0    failed=0
```

- 8. 再次以 student2 用户身份，使用 SSH 连接到 servera。观察 student2 用户的错误，然后断开与 servera 的连接。

```
[student@workstation role-galaxy]$ ssh student2@servera
Activate the web console with: systemctl enable --now cockpit.socket

-bash: [: missing `]'
[student2@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation role-galaxy]$
```

解析 student2 用户的 .bash_profile 文件时发生了错误。

- 9. 更正 student.bash_env 角色的开发版本中的错误，再重新执行 playbook。

9.1. 编辑 roles/student.bash_env/templates/_bash_profile.j2 文件。将缺失的] 字符添加到第 4 行，再保存文件。文件的开头现在为：

```
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi

# User specific environment and startup programs

PATH=$PATH:$HOME/.local/bin:$HOME/bin

export PATH
```

保存该文件。

9.2. 执行 use-bash_env-role.yml playbook。

```
[student@workstation role-galaxy]$ ansible-playbook use-bash_env-role.yml

PLAY [use student.bash_env role playbook] ****
TASK [Gathering Facts] ****
ok: [servera.lab.example.com]

TASK [Ensure test user does not exist] ****
changed: [servera.lab.example.com]

TASK [student.bash_env : put away .bashrc] ****
ok: [servera.lab.example.com]
```

```
TASK [student.bash_env : put away .bash_profile] ****
changed: [servera.lab.example.com]

TASK [student.bash_env : put away .vimrc] ****
ok: [servera.lab.example.com]

TASK [Create the test user] ****
changed: [servera.lab.example.com]

PLAY RECAP ****
servera.lab.example.com    : ok=6      changed=3      unreachable=0      failed=0
```

9.3. 再次以 `student2` 用户身份，使用 SSH 连接到 `servera`。

```
[student@workstation role-galaxy]$ ssh student2@servera
Activate the web console with: systemctl enable --now cockpit.socket

[student2 on servera in ~ dir]$ exit
logout
Connection to servera closed.
[student@workstation role-galaxy]$
```

错误消息不再存在。`student2` 用户的自定义提示符现在将以蓝色字符显示。

上述步骤演示了 `student.bash_env` 角色的开发版本存在缺陷。根据测试结果，开发人员会将必要的修复重新提交到角色的开发分支。当开发分支通过必要的质量检查后，开发人员将开发分支中的功能合并到 `master` 分支中。

提交角色更改到 Git 存储库不在本课程范畴内。



重要

在跟踪项目中某一角色的最新版本时，请定期重新安装该角色来进行更新。这样能确保您的本地副本在错误修复、补丁和其他功能方面保持最新的状态。

不过，如果在生产中使用第三方角色，请指定要使用的版本，从而避免因为意外更改而造成服务中断。定期在测试环境中更新到角色的最新版本，以便以受控的方式采用功能增强和更改。

完成

运行 `lab role-galaxy finish` 命令，以清理受管主机。

```
[student@workstation ~]$ lab role-galaxy finish
```

本引导式练习到此结束。

▶ 开放研究实验

利用角色简化 PLAYBOOK

任务执行清单

在本实验中，您将创建使用变量、文件、模板、任务和处理程序的 Ansible 角色。

成果

您应能够：

- 创建使用变量、文件、模板、任务和处理程序的 Ansible 角色，以配置开发用 Web 服务器。
- 在 playbook 中使用托管于远程存储库的角色。
- 在 playbook 中使用红帽企业 Linux 系统角色。

场景概述

您的组织必须提供一台 Web 服务器来托管所有 Web 开发人员的开发代码。您的任务是编写一个 playbook 来配置此开发用 Web 服务器。

开发用 Web 服务器必须满足几个要求：

- 开发服务器配置与生产服务器配置相匹配。生产服务器配置为使用由组织的基础架构团队开发的 Ansible 角色。
- 每位开发人员在开发服务器上都拥有一个目录，用于托管代码和内容。每位开发人员的内容都使用分配的非标准端口进行访问。
- SELinux 设置为 enforcing 和 targeted。

您的 playbook 将：

- 使用一个角色在 Web 服务器上为每位开发人员配置目录和端口。您必须编写该角色。
此角色依赖于组织编写用来配置 Apache 的角色。您应该使用该组织角色的版本 **v1.4** 来定义依赖项。依赖项的存储库的 URL 为：`git@workstation.lab.example.com:infra/apache`
- 使用 `rhel-system-roles.selinux` 角色，为 Web 服务器使用的非标准 HTTP 端口配置 SELinux。您将获得一个 `selinux.yml` 变量文件，该文件可作为 `group_vars` 文件安装，以将正确的设置传递给角色。

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 role-review start` 命令。该脚本将创建工作目录 `/home/student/role-review`，并为它填充 Ansible 配置文件、主机清单和其他实验文件。

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

1. 更改到 `/home/student/role-review` 工作目录。
2. 创建一个名为 `web_dev_server.yml` 的 playbook，其中含有一个名为 **Configure Dev Web Server** 的 play。将该 play 配置为以主机组 `dev_webserver` 为目标。暂时不要在 play 中添加任何角色或任务。

确保 play 强制执行处理程序，因为您可能会在开发 playbook 时遇到错误。
3. 检查 playbook 的语法。运行 playbook。语法检查应该会通过，并且 playbook 应该成功运行。
4. 确保已安装 playbook 的角色依赖项。

您要创建的 `apache.developer_configs` 角色将依赖于 `infra.apache` 角色。创建 `roles/requirements.yml` 文件。它应当从位于 `git@workstation.lab.example.com:infra/apache` 的 Git 存储库安装角色，使用版本 `v1.4`，并在本地将它命名为 `infra.apache`。您可以假定已经配置了 SSH 密钥，可以自动从该存储库获取角色。使用 `ansible-galaxy` 命令安装角色。
此外，如果 `rhel-system-roles` 软件包不存在，则还要安装此软件包。
5. 在 `roles` 子目录中，初始化一个名为 `apache.developer_configs` 的新角色。

添加 `infra.apache` 角色作为新角色的依赖项，为名称、来源、版本和版本控制系统使用与 `roles/requirements.yml` 文件相同的信息。
项目目录中的 `developer_tasks.yml` 文件包含该角色的任务。将此文件移到正确的位置，使其成为此角色的任务文件，并替换该位置中的现有文件。
项目目录中的 `developer.conf.j2` 文件是供任务文件使用的 Jinja2 模板。将它移到此角色使用的模板文件对应的正确位置。
6. `apache.developer_configs` 角色将处理变量 `web_developers` 中定义的用户列表。项目目录中的 `web_developers.yml` 文件定义 `web_developers` 用户列表变量。检查该文件，并使用它来定义开发用 Web 服务器主机组的 `web_developers` 变量。
7. 将角色 `apache.developer_configs` 添加到 `web_dev_server.yml` playbook 中的 play 中。
8. 检查 playbook 的语法。运行 playbook。语法检查应该会通过，但 playbook 应该会在 `infra.apache` 角色试图重启 Apache HTTPD 时失败。
9. Apache HTTPD 在上一步中重启失败，因为供开发人员使用的网络端口标记了错误的 SELinux 上下文。已为您提供了一个变量文件 `selinux.yml`，它可与 `rhel-system-roles.selinux` 角色一起用来修复此问题。

在 `web_dev_server.yml` playbook 中，为您的 play 创建 `pre_tasks` 部分。在该部分中，使用一个任务将 `rhel-system-roles.selinux` 角色包含到 `block/rescue` 结构中，使它能被正确应用。查看此角色的相关讲座或文档，以了解具体的操作方法。
检查 `selinux.yml` 文件。将它移到正确的位置，以便针对 `dev_webserver` 主机组设置其变量。
10. 检查最终 playbook 的语法。语法检查应该会通过。
11. 运行 playbook。应该会成功。
12. 测试开发用 Web 服务器的配置。验证所有端点都可访问，并且能提供每位开发人员的内容。

评估

从 `workstation` 计算机运行 `lab role-review grade` 命令，以评测您的工作。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 role-review grade
```

完成

在 workstation 上，运行 **lab role-review finish** 脚本来清理本练习。

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

本实验到此结束。

► 解决方案

利用角色简化 PLAYBOOK

任务执行清单

在本实验中，您将创建使用变量、文件、模板、任务和处理程序的 Ansible 角色。

成果

您应能够：

- 创建使用变量、文件、模板、任务和处理程序的 Ansible 角色，以配置开发用 Web 服务器。
- 在 playbook 中使用托管于远程存储库的角色。
- 在 playbook 中使用红帽企业 Linux 系统角色。

场景概述

您的组织必须提供一台 Web 服务器来托管所有 Web 开发人员的开发代码。您的任务是编写一个 playbook 来配置此开发用 Web 服务器。

开发用 Web 服务器必须满足几个要求：

- 开发服务器配置与生产服务器配置相匹配。生产服务器配置为使用由组织的基础架构团队开发的 Ansible 角色。
- 每位开发人员在开发服务器上都拥有一个目录，用于托管代码和内容。每位开发人员的内容都使用分配的非标准端口进行访问。
- SELinux 设置为 enforcing 和 targeted。

您的 playbook 将：

- 使用一个角色在 Web 服务器上为每位开发人员配置目录和端口。您必须编写该角色。
此角色依赖于组织编写用来配置 Apache 的角色。您应该使用该组织角色的版本 **v1.4** 来定义依赖项。依赖项的存储库的 URL 为：`git@workstation.lab.example.com:infra/apache`
- 使用 `rhel-system-roles.selinux` 角色，为 Web 服务器使用的非标准 HTTP 端口配置 SELinux。您将获得一个 `selinux.yml` 变量文件，该文件可作为 `group_vars` 文件安装，以将正确的设置传递给角色。

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 role-review start` 命令。该脚本将创建工作目录 /`home/student/role-review`，并为它填充 Ansible 配置文件、主机清单和其他实验文件。

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

- 更改到 `/home/student/role-review` 工作目录。

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

- 创建一个名为 `web_dev_server.yml` 的 playbook，其中含有一个名为 **Configure Dev Web Server** 的 play。将该 play 配置为以主机组 `dev_webserver` 为目标。暂时不要在 play 中添加任何角色或任务。

确保 play 强制执行处理程序，因为您可能会在开发 playbook 时遇到错误。

完成后，`/home/student/role-review/web_dev_server.yml` playbook 将包含：

```
---
- name: Configure Dev Web Server
  hosts: dev_webserver
  force_handlers: yes
```

- 检查 playbook 的语法。运行 playbook。语法检查应该会通过，并且 playbook 应该成功运行。

```
[student@workstation role-review]$ ansible-playbook \
> --syntax-check web_dev_server.yml

playbook: web_dev_server.yml
[student@workstation role-review]$ ansible-playbook web_dev_server.yml
PLAY [Configure Dev Web Server] ****
TASK [Gathering Facts] ****
ok: [servera.lab.example.com]

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

- 确保已安装 playbook 的角色依赖项。

您要创建的 `apache.developer_configs` 角色将依赖于 `infra.apache` 角色。创建 `roles/requirements.yml` 文件。它应当从位于 `git@workstation.lab.example.com:infra/apache` 的 Git 存储库安装角色，使用版本 **v1.4**，并在本地将它命名为 `infra.apache`。您可以假定已经配置了 SSH 密钥，可以自动从该存储库获取角色。使用 `ansible-galaxy` 命令安装角色。

此外，如果 `rhel-system-roles` 软件包不存在，则还要安装此软件包。

- 为 playbook 项目创建 `roles` 子目录。

```
[student@workstation role-review]$ mkdir -v roles
mkdir: created directory 'roles'
```

- 创建 `roles/requirements.yml` 文件，并为 `infra.apache` 角色添加一个条目。使用角色的 git 存储库中的版本 **v1.4**。

完成后，`roles/requirements.yml` 文件将包含：

```
- name: infra.apache
  src: git@workstation.lab.example.com:infra/apache
  scm: git
  version: v1.4
```

4.3. 安装项目依赖项。

```
[student@workstation role-review]$ ansible-galaxy install \
> -r roles/requirements.yml -p roles
- extracting infra.apache to /home/student/role-review/roles/infra.apache
- infra.apache (v1.4) was installed successfully
```

4.4. 如果 RHEL 系统角色软件包不存在，则安装此软件包。这是在前面的练习中安装的。

```
[student@workstation role-review]$ sudo yum install rhel-system-roles
```

5. 在 **roles** 子目录中，初始化一个名为 **apache.developer_configs** 的新角色。

添加 **infra.apache** 角色作为新角色的依赖项，为名称、来源、版本和版本控制系统使用与 **roles/requirements.yml** 文件相同的信息。

项目目录中的 **developer_tasks.yml** 文件包含该角色的任务。将此文件移到正确的位置，使其成为此角色的任务文件，并替换该位置中的现有文件。

项目目录中的 **developer.conf.j2** 文件是供任务文件使用的 Jinja2 模板。将它移到此角色使用的模板文件对应的正确位置。

5.1. 使用 **ansible-galaxy init**，为 **apache.developer_configs** 角色创建角色框架。

```
[student@workstation role-review]$ cd roles
[student@workstation roles]$ ansible-galaxy init apache.developer_configs
- apache.developer_configs was created successfully
[student@workstation roles]$ cd ..
[student@workstation role-review]$
```

5.2. 更新 **apache.developer_configs** 角色的 **roles/apache.developer_configs/meta/main.yml** 文件，以反映对 **infra.apache** 角色的依赖。

编辑后，**dependencies** 变量将按如下所示进行定义：

```
dependencies:
- name: infra.apache
  src: git@workstation.lab.example.com:infra/apache
  scm: git
  version: v1.4
```

5.3. 使用 **developer_tasks.yml** 文件来覆盖角色的 **tasks/main.yml** 文件。

```
[student@workstation role-review]$ mv -v developer_tasks.yml \
> roles/apache.developer_configs/tasks/main.yml
renamed 'developer_tasks.yml' -> 'roles/apache.developer_configs/tasks/main.yml'
```

5.4. 将 `developer.conf.j2` 文件放入角色的 `templates` 目录中。

```
[student@workstation role-review]$ mv -v developer.conf.j2 \
> roles/apache.developer_configs/templates/
renamed 'developer.conf.j2' -> 'roles/apache.developer_configs/templates/
developer.conf.j2'
```

6. `apache.developer_configs` 角色将处理变量 `web_developers` 中定义的用户列表。项目目录中的 `web_developers.yml` 文件定义 `web_developers` 用户列表变量。检查该文件，并使用它来定义开发用 Web 服务器主机组的 `web_developers` 变量。

6.1. 检查 `web_developers.yml` 文件。

```
---
web_developers:
  - username: jdoe
    name: John Doe
    user_port: 9081
  - username: jdoe2
    name: Jane Doe
    user_port: 9082
```

会为每位 Web 开发人员定义一个 `name`、`username`、`user_port`。

6.2. 将 `web_developers.yml` 放到 `group_vars/dev_webserver` 子目录中。

```
[student@workstation role-review]$ mkdir -pv group_vars/dev_webserver
mkdir: created directory 'group_vars'
mkdir: created directory 'group_vars/dev_webserver'
[student@workstation role-review]$ mv -v web_developers.yml \
> group_vars/dev_webserver/
renamed 'web_developers.yml' -> 'group_vars/dev_webserver/web_developers.yml'
```

7. 将角色 `apache.developer_configs` 添加到 `web_dev_server.yml` playbook 中的 play 中。

编辑后的 playbook:

```
---
- name: Configure Dev Web Server
  hosts: dev_webserver
  force_handlers: yes
  roles:
    - apache.developer_configs
```

8. 检查 playbook 的语法。运行 playbook。语法检查应该会通过，但 playbook 应该会在 `infra.apache` 角色试图重启 Apache HTTPD 时失败。

```
[student@workstation role-review]$ ansible-playbook \
> --syntax-check web_dev_server.yml

playbook: web_dev_server.yml
[student@workstation role-review]$ ansible-playbook web_dev_server.yml
```

```

PLAY [Configure Dev Web Server] ****
TASK [Gathering Facts] ****
ok: [servera.lab.example.com]

...output omitted...

TASK [infra.apache : Install a skeleton index.html] ****
skipping: [servera.lab.example.com]

TASK [apache.developer_configs : Create user accounts] ****
changed: [servera.lab.example.com] => (item={u'username': u'jdoe', u'user_port': 9081, u'name': u'John Doe'})
changed: [servera.lab.example.com] => (item={u'username': u'jdoe2', u'user_port': 9082, u'name': u'Jane Doe'})

...output omitted...

RUNNING HANDLER [infra.apache : restart firewalld] ****
changed: [servera.lab.example.com]

RUNNING HANDLER [infra.apache : restart apache] ****
fatal: [servera.lab.example.com]: FAILED! => {"changed": false, "msg": "Unable to restart service httpd: Job for httpd.service failed because the control process exited with error code. See \\"systemctl status httpd.service\\" and \\"journalctl -xe\\" for details.\n"}

NO MORE HOSTS LEFT ****
to retry, use: --limit @/home/student/role-review/web_dev_server.retry

PLAY RECAP ****
servera.lab.example.com    : ok=13    changed=11    unreachable=0    failed=1
skipped=1      rescued=0      ignored=0

```

httpd 服务重启时出现了错误。由于非标准 HTTP 端口上的 SELinux 上下文的原因，**httpd** 服务守护进程无法绑定到这些端口。

9. Apache HTTPD 在上一步中重启失败，因为供开发人员使用的网络端口标记了错误的 SELinux 上下文。已为您提供了一个变量文件 **selinux.yml**，它可与 **rhel-system-roles.selinux** 角色一起用来修复此问题。

在 **web_dev_server.yml** playbook 中，为您的 play 创建 **pre_tasks** 部分。在该部分中，使用一个任务将 **rhel-system-roles.selinux** 角色包含到 **block/rescue** 结构中，使它能被正确应用。查看此角色的相关讲座或文档，以了解具体的操作方法。

检查 **selinux.yml** 文件。将它移到正确的位置，以便针对 **dev_webserver** 主机组设置其变量。

9.1. **pre_tasks** 部分可以添加到 **web_dev_server.yml** playbook 中相应 play 的末尾。

您可以查看 **/usr/share/doc/rhel-system-roles-1.0/selinux/examples/selinux-playbook.yml** 中的块，以了解有关如何应用该角色的基本概要，但红帽 Ansible 引擎 2.7 允许您将复杂的 **shell** 和 **wait_for** 逻辑替换为 **reboot** 模块。

pre_tasks 部分应包含：

```

pre_tasks:
  - name: Check SELinux configuration
    block:
      - include_role:
          name: rhel-system-roles.selinux
    rescue:
      # Fail if failed for a different reason than selinux_reboot_required.
      - name: Check for general failure
        fail:
          msg: "SELinux role failed."
        when: not selinux_reboot_required

      - name: Restart managed host
        reboot:
          msg: "Ansible rebooting system for updates."

    - name: Reapply SELinux role to complete changes
      include_role:
        name: rhel-system-roles.selinux

```

9.2. **selinux.yml** 文件包含 `rhel-system-roles.selinux` 角色的变量定义。使用该文件来定义 play 的主机组的变量。

```

[student@workstation role-review]$ cat selinux.yml
---
# variables used by rhel-system-roles.selinux

selinux_policy: targeted
selinux_state: enforcing

selinux_ports:
  - ports:
      - "9081"
      - "9082"
    proto: 'tcp'
    setype: 'http_port_t'
    state: 'present'

[student@workstation role-review]$ mv -v selinux.yml \
> group_vars/dev_webserver/
renamed 'selinux.yml' -> 'group_vars/dev_webserver/selinux.yml'

```

10. 检查最终 playbook 的语法。语法检查应该会通过。

```

[student@workstation role-review]$ ansible-playbook \
> --syntax-check web_dev_server.yml

playbook: web_dev_server.yml

```

最终的 `web_dev_server.yml` playbook 应当如下所示：

```

---
- name: Configure Dev Web Server
  hosts: dev_webserver
  force_handlers: yes
  roles:
    - apache.developer_configs
  pre_tasks:
    - name: Check SELinux configuration
      block:
        - include_role:
            name: rhel-system-roles.selinux
  rescue:
    # Fail if failed for a different reason than selinux_reboot_required.
    - name: Check for general failure
      fail:
        msg: "SELinux role failed."
      when: not selinux_reboot_required

    - name: Restart managed host
      reboot:
        msg: "Ansible rebooting system for updates."

    - name: Reapply SELinux role to complete changes
      include_role:
        name: rhel-system-roles.selinux

```



注意

不论 **pre_tasks** 是在 play 的末尾还是从 playbook 文件中的执行顺序看来的“正确”位置，对 **ansible-playbook** 来说都不重要。它仍然会按照正确的顺序执行 play 的任务。

11. 运行 playbook。应该会成功。

```
[student@workstation role-review]$ ansible-playbook web_dev_server.yml

PLAY [Configure Dev Web Server] ****
TASK [Gathering Facts] ****
ok: [servera.lab.example.com]

TASK [include_role : rhel-system-roles.selinux] ****
TASK [rhel-system-roles.selinux : Install SELinux python3 tools] ****
ok: [servera.lab.example.com]

...output omitted...

TASK [infra.apache : Apache Service is started] ****
changed: [servera.lab.example.com]

...output omitted...
```

```
TASK [apache.developer_configs : Copy Per-Developer Config files] ****
ok: [servera.lab.example.com] => (item={'username': u'jdoe', 'user_port': 9081,
  'name': u'John Doe'})
ok: [servera.lab.example.com] => (item={'username': u'jdoe2', 'user_port': 9082,
  'name': u'Jane Doe'})

PLAY RECAP ****
servera.lab.example.com    : ok=19    changed=3    unreachable=0    failed=0
skipped=14    rescued=0    ignored=0
```

12. 测试开发用 Web 服务器的配置。验证所有端点都可访问，并且能提供每位开发人员的内容。

```
[student@workstation role-review]$ curl servera
servera.lab.example.com has been customized using Ansible.
[student@workstation role-review]$ curl servera:9081
This is index.html for user: John Doe (jdoe)
[student@workstation role-review]$ curl servera:9082
This is index.html for user: Jane Doe (jdoe2)
[student@workstation role-review]$
```

评估

从 workstation 计算机运行 **lab role-review grade** 命令，以评测您的工作。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 role-review grade
```

完成

在 workstation 上，运行 **lab role-review finish** 脚本来清理本练习。

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

总结

在本章中，您学到了：

Ansible 角色

- 角色以一定的方式组织 Ansible 代码，从而实现重复使用和共享。
- 红帽企业 Linux 系统角色是一系列经过测试并受到支持的角色，旨在帮助您在不同版本的红帽企业 Linux 中配置主机子系统。
- Ansible Galaxy [<https://galaxy.ansible.com>] 是一个 Ansible 角色公共资源库，这些角色由 Ansible 用户编写。**ansible-galaxy** 命令可以搜索角色，显示角色相关的信息，以及安装、列举、删除或初始化角色。
- Playbook 需要的外部角色可以在 **roles/requirements.yml** 文件中定义。**ansible-galaxy install -r roles/requirements.yml** 命令使用此文件将角色安装到控制节点上。

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

章 9

对 ANSIBLE 进行故障排除

目标

对 playbook 和受管主机进行故障排除。

培训目标

- 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。

章节

- 对 playbook 进行故障排除（及引导式练习）
- 对 Ansible 受管主机进行故障排除（及引导式练习）

实验

- 对 Ansible 进行故障排除

对 PLAYBOOK 进行故障排除

培训目标

学完本章节后，您应该能够对新 playbook 的一般问题进行故障排除并修复这些问题。

ANSIBLE 日志文件

默认情况下，红帽 Ansible 引擎配置为不将其输出记录到任何日志文件。它提供了一个内置日志基础架构，可以通过 `ansible.cfg` 配置文件的 `default` 部分中的 `log_path` 参数进行配置，或通过 `$ANSIBLE_LOG_PATH` 环境变量来配置。如果进行了其中任一/全部配置，Ansible 会把来自 `ansible` 和 `ansible-playbook` 命令的输出存储到通过 `ansible.cfg` 配置文件或 `$ANSIBLE_LOG_PATH` 环境变量配置的日志文件中。

如果您将 Ansible 配置为将日志文件写入 `/var/log`，则红帽建议您配置 `logrotate` 来管理 Ansible 日志文件。

调试模块

通过 `debug` 模块可以了解 play 中发生的情况。此模块可以显示 play 中某个点上某个变量的值。在对使用变量互相通信的任务（例如，将一项任务的输出用作后续任务的输入）进行调试时，此功能可以发挥关键作用。

以下示例使用 `debug` 任务中的 `msg` 和 `var` 设置。第一个示例显示 `ansible_facts['memfree_mb']` 事实的运行时值，作为 `ansible-playbook` 输出中显示的消息的一部分。第二个示例显示 `output` 变量的值。

```
- name: Display free memory
  debug:
    msg: "Free memory for this system is {{ ansible_facts['memfree_mb'] }}"

- name: Display the "output" variable
  debug:
    var: output
    verbosity: 2
```

管理错误

playbook 运行期间可能会发生多种问题，它们主要与 playbook 或它使用的任何模板的语法相关，或者源自与受管主机的连接问题（例如，清单文件中受管主机的主机名称存在错误）。这些错误在执行时由 `ansible-playbook` 命令发出。

在本课程的前面部分，您了解了 `--syntax-check` 选项，它可检查 playbook 的 YAML 语法。在使用 playbook 之前，或者当您遇到了相关问题时，最好对其进行语法检查。

```
[student@demo ~]$ ansible-playbook play.yml --syntax-check
```

您也可以使用 **--step** 选项来逐步调试 playbook，一次一个任务。**ansible-playbook --step** 命令以交互方式提示您确认希望运行的每个任务。

```
[student@demo ~]$ ansible-playbook play.yml --step
```

--start-at-task 选项允许您从特定的任务开始执行 playbook。它取要作为开始的任务名称作为参数。

```
[student@demo ~]$ ansible-playbook play.yml --start-at-task="start httpd service"
```

调试

以 **ansible-playbook** 命令运行 playbook 所提供的输出作为起点，是对 Ansible 受管主机相关问题进行故障排除的良好开端。参考 playbook 执行的以下输出：

```
PLAY [Service Deployment] *****
...output omitted...
TASK: [Install a service] *****
ok: [demoservera]
ok: [demoserverb]

PLAY RECAP *****
demoservera      : ok=2    changed=0    unreachable=0    failed=0
demoserverb      : ok=2    changed=0    unreachable=0    failed=0
```

在上面的输出中，显示了一个 **PLAY** 标头（带有要执行的 play 的名称），后跟一个或多个 **TASK** 标头。这些标头各自代表其在 playbook 中相关的任务，它会在属于 playbook 中 hosts 参数包含的组的所有受管主机上执行。

当每一受管主机执行各个 play 的任务时，受管主机的名称显示在对应的 **TASK** 标头下，同时显示有该受管主机上的任务状态。任务状态可以显示为 **ok**、**fatal**、**changed** 或 **skipping**。

在各个 play 的输出的底部，**PLAY RECAP** 部分显示对每一受管主机执行的任务的数量。

正如本课程前面部分所讨论的，您可以通过添加一个或多个 **-v** 选项来提高 **ansible-playbook** 输出的详细程度。**ansible-playbook -v** 命令提供了额外的调试信息，总共有四个级别。

详细程度配置

选项	描述
-v	显示输出数据。
-vv	显示输出和输入数据。
-vvv	包含关于和受管主机连接的信息。
-vvvv	包括其他信息，如每一远程主机上执行的脚本，以及执行各个脚本的用户。

PLAYBOOK 管理的推荐做法

尽管上文讨论的工具或可帮助识别和改正 playbook 中的问题，在开发这些 playbook 时，务必要牢记一些推荐做法，它们有助于简化故障排除的流程。下方列出了一些 playbook 开发推荐做法：

- 使用 play 或任务的用途的简要描述来命名 play 和任务。在执行 playbook 时，play 名称或任务名称会显示出来。这也有助于记录每个 play 或任务应该达到的目标，以及可能需要它的原因。
- 包含注释，以添加与任务相关的其他内嵌文档。
- 有效利用垂直空白。通常，垂直组织任务属性可以使它们更易于阅读。
- 一致的水平缩进至关重要。使用空格而不是制表符，以避免缩进错误。将文本编辑器设置为当您按下 **Tab** 键时插入空格，以简化操作。
- 尽可能使 playbook 简单。仅使用您需要的功能。



参考文献

配置 Ansible —— Ansible 文档

https://docs.ansible.com/ansible/latest/installation_guide/intro_configuration.html

debug —— 执行期间显示语句 —— Ansible 文档

https://docs.ansible.com/ansible/latest/modules/debug_module.html

最佳做法 —— Ansible 文档

https://docs.ansible.com/ansible/latest/user_guide/playbooks_best_practices.html