开放研究实验

管理大项目

任务执行清单

在本实验中，您将通过使用主机模式、包含、导入和动态清单来修改大型 playbook 以便于管理。您还将调整 Ansible 处理 playbook 的方式。

成果

您可以通过针对动态清单指定主机模式来简化 playbook 中的受管主机引用。您还应该能够重组 playbook，以便从外部任务文件导入任务并调整该 playbook 进行滚动更新。

说明

您继承了以前管理员的 playbook。该 playbook 用于在 `servera.lab.example.com`、`serverb.lab.example.com`、`serverc.lab.example.com` 和 `serverd.lab.example.com` 上配置 Web 服务。该 playbook 还在四个受管主机上配置防火墙，以便允许 Web 流量。

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 projects-review start`** 命令。此设置脚本将确保受管主机可在网络上访问。它也将确保控制节点上安装了正确的 Ansible 配置文件、清单文件和 playbook。

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

对 **`playbook.yml`** playbook 文件进行以下更改以方便进行管理，并对其进行调整，以便未来的执行使用滚动更新来防止所有四个 Web 服务器同时不可用。

1. 使用通配符主机模式简化 playbook 中受管主机的列表。使用 **`inventory/inventory.py`** 动态库存脚本验证通配符主机模式。
2. 重组该 playbook，以便其中的前三个任务保存在位于 **`tasks/web_tasks.yml`** 的外部任务文件中。使用 **`import_tasks`** 功能将此任务文件合并到 playbook 中。
3. 重组该 playbook，以便其中的第四、第五和第六个任务保存在位于 **`tasks/firewall_tasks.yml`** 的外部任务文件中。使用 **`import_tasks`** 功能将此任务文件合并到 playbook 中。
4. **`tasks/web_tasks.yml`** 和 **`tasks/firewall_tasks.yml`** 文件之间有一些重复的任务。将安装软件包和启用服务的任务移动到名为 **`tasks/install_and_enable.yml`** 的新文件中，并更新它们以使用变量。将原始任务替换为 **`import_tasks`** 语句，传入适当的变量值。
5. 如果将来会对 **`files/tune.conf`** 文件进行更改，则可触发重新启动 `httpd` 服务的处理程序，因此在 **`playbook.yml`** playbook 中实施滚动更新功能并将滚动更新批处理大小设置为两台主机。
6. 验证是否对 **`playbook.yml`** playbook 进行了正确的更改，然后执行此 playbook。

海量视频题库 mylpub.com QQ:5565462 www.52mylt.com

评估

从 workstation 运行 **lab projects-review grade** 命令，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 projects-review grade
```

完成

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

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

本实验到此结束。

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

解决方案

管理大项目

任务执行清单

在本实验中，您将通过使用主机模式、包含、导入和动态清单来修改大型 playbook 以便于管理。您还将调整 Ansible 处理 playbook 的方式。

成果

您可以通过针对动态清单指定主机模式来简化 playbook 中的受管主机引用。您还应该能够重组 playbook，以便从外部任务文件导入任务并调整该 playbook 进行滚动更新。

说明

您继承了以前管理员的 playbook。该 playbook 用于在 `servera.lab.example.com`、`serverb.lab.example.com`、`serverc.lab.example.com` 和 `serverd.lab.example.com` 上配置 Web 服务。该 playbook 还在四个受管主机上配置防火墙，以便允许 Web 流量。

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 projects-review start`** 命令。此设置脚本将确保受管主机可在网络上访问。它也将确保控制节点上安装了正确的 Ansible 配置文件、清单文件和 playbook。

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

对 **`playbook.yml`** playbook 文件进行以下更改以方便进行管理，并对其进行调整，以便未来的执行使用滚动更新来防止所有四个 Web 服务器同时不可用。

1. 使用通配符主机模式简化 playbook 中受管主机的列表。使用 **`inventory/inventory.py`** 动态库存脚本验证通配符主机模式。

   1.1. 将目录更改到 **`/home/student/projects-review`** 工作目录。查看 **`ansible.cfg`** 配置文件以确定清单文件的位置。该清单被定义为 **`inventory`** 子目录，且该子目录包含 **`inventory.py`** 动态清单脚本。

```
[student@workstation ~]$ cd ~/projects-review
[student@workstation projects-review]$ cat ansible.cfg
[defaults]
inventory = inventory
...output omitted...
[student@workstation projects-review]$ ll
total 16
-rw-rw-r--. 1 student student   33 Dec 19 00:48 ansible.cfg
drwxrwxr-x. 2 student student 4096 Dec 18 22:35 files
drwxrwxr-x. 2 student student 4096 Dec 19 01:18 inventory
```

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

```
-rw-rw-r--. 1 student student  959 Dec 18 23:48 playbook.yml
[student@workstation projects-review]$ ll inventory/
total 4
-rwxrwxr-x. 1 student student 612 Dec 19 01:18 inventory.py
```

1.2. 使 **inventory/inventory.py** 动态清单脚本可执行，然后使用 **--list** 选项执行此动态清单脚本以显示清单中的完整主机列表。

```
[student@workstation projects-review]$ chmod 755 inventory/inventory.py
[student@workstation projects-review]$ inventory/inventory.py --list
{"all": {"hosts": ["servera.lab.example.com", "serverb.lab.example.com",
 "serverc.lab.example.com", "serverd.lab.example.com",
 "workstation.lab.example.com"], "vars": { }}}
```

1.3. 验证主机模式 **server*.lab.example.com** 正确识别被 **playbook.yml** playbook 作为目标的四个受管主机。

```
[student@workstation projects-review]$ ansible server*.lab.example.com \
> --list-hosts
  hosts (4):
    serverb.lab.example.com
    serverd.lab.example.com
    servera.lab.example.com
    serverc.lab.example.com
```

1.4. 将 **playbook.yml** playbook 中的主机列表替换为 **server*.lab.example.com** 主机模式。

```
---
- name: Install and configure web service
  hosts: server*.lab.example.com
...output omitted...
```

2. 重组该 playbook，以便其中的前三个任务保存在位于 **tasks/web_tasks.yml** 的外部任务文件中。使用 **import_tasks** 功能将此任务文件合并到 playbook 中。

2.1. 创建 **tasks** 子目录。

```
[student@workstation projects-review]$ mkdir tasks
```

2.2. 将 **playbook.yml** playbook 中的前三个任务的内容放入 **tasks/web_tasks.yml** 文件。该任务文件应当包含以下内容：

```
---
- name: Install httpd
  yum:
    name: httpd
    state: latest

- name: Enable and start httpd
  service:
```

海量视频题库 myitpub.com QQ:556462
www.52myit.com

```
    name: httpd
    enabled: true
    state: started

- name: Tuning configuration installed
  copy:
    src: files/tune.conf
    dest: /etc/httpd/conf.d/tune.conf
    owner: root
    group: root
    mode: 0644
  notify:
    - restart httpd
```

2.3. 从 **playbook.yml** playbook 中删除前三个任务，并将以下行放在该位置以导入 **tasks/web_tasks.yml** 任务文件。

```
- name: Import the web_tasks.yml task file
  import_tasks: tasks/web_tasks.yml
```

3. 重组该 playbook，以便其中的第四、第五和第六个任务保存在位于 **tasks/firewall_tasks.yml** 的外部任务文件中。使用 **import_tasks** 功能将此任务文件合并到 playbook 中。

3.1. 将 **playbook.yml** playbook 中的剩余三个任务的内容放入 **tasks/firewall_tasks.yml** 文件。该任务文件应当包含以下内容。

```
---
- name: Install firewalld
  yum:
    name: firewalld
    state: latest

- name: Enable and start the firewall
  service:
    name: firewalld
    enabled: true
    state: started

- name: Open the port for http
  firewalld:
    service: http
    immediate: true
    permanent: true
    state: enabled
```

3.2. 从 **playbook.yml** playbook 中删除剩余的三个任务，并将以下行放在该位置以导入 **tasks/firewall_tasks.yml** 任务文件。

```
- name: Import the firewall_tasks.yml task file
  import_tasks: tasks/firewall_tasks.yml
```

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

4. **tasks/web_tasks.yml** 和 **tasks/firewall_tasks.yml** 文件之间有一些重复的任务。将安装软件包和启用服务的任务移动到名为 **tasks/install_and_enable.yml** 的新文件中，并更新它们以使用变量。将原始任务替换为 **import_tasks** 语句，传入适当的变量值。

   4.1. 将 **tasks/web_tasks.yml** 中的 yum 和服务任务复制到一个名为 **tasks/install_and_enable.yml** 的新文件中。

```
---
- name: Install httpd
  yum:
    name: httpd
    state: latest

- name: Enable and start httpd
  service:
    name: httpd
    enabled: true
    state: started
```

   4.2. 将 **tasks/install_and_enable.yml** 中的软件包和服务名称替换为变量 package 和 service。

```
---
- name: Install {{ package }}
  yum:
    name: "{{ package }}"
    state: latest

- name: Enable and start {{ service }}
  service:
    name: "{{ service }}"
    enabled: true
    state: started
```

   4.3. 将 **tasks/web_tasks.yml** 和 **tasks/firewall_tasks.yml** 中的 yum 和服务任务替换为 **import_tasks** 语句。

```
---
- name: Install and start httpd
  import_tasks: install_and_enable.yml
  vars:
    package: httpd
    service: httpd
```

```
---
- name: Install and start firewalld
  import_tasks: install_and_enable.yml
  vars:
    package: firewalld
    service: firewalld
```

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

5. 如果将来会对 **files/tune.conf** 文件进行更改，则可触发重新启动 httpd 服务的处理程序，因此在 **playbook.yml** playbook 中实施滚动更新功能并将滚动更新批处理大小设置为两台主机。

   5.1. 在 **playbook.yml** playbook 中添加 **serial** 参数。

```
---
- name: Install and configure web service
  hosts: server*.lab.example.com
  serial: 2
...output omitted...
```

6. 验证是否对 **playbook.yml** playbook 进行了正确的更改，然后执行此 playbook。

   6.1. 验证 **playbook.yml** playbook 是否包含以下内容。

```
---
- name: Install and configure web service
  hosts: server*.lab.example.com
  serial: 2

  tasks:
    - name: Import the web_tasks.yml task file
      import_tasks: tasks/web_tasks.yml

    - name: Import the firewall_tasks.yml task file
      import_tasks: tasks/firewall_tasks.yml

  handlers:
    - name: restart httpd
      service:
        name: httpd
        state: restarted
```

   6.2. 使用 **ansible-playbook --syntax-check** 执行该 playbook，以验证其不包含语法错误。如果存在错误，请先纠正它们再继续。

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

   6.3. 执行 playbook。该 playbook 应以批处理大小为两台受管主机的滚动更新针对主机执行。

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

PLAY [Install and configure web service] ********************************

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

TASK [Install httpd] *****************************************************
changed: [serverd.lab.example.com]
```

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

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

TASK [Enable and start httpd] ***********************************************
changed: [serverd.lab.example.com]
changed: [serverb.lab.example.com]

TASK [Tuning configuration installed] ***************************************
changed: [serverb.lab.example.com]
changed: [serverd.lab.example.com]

TASK [Install firewalld] ****************************************************
ok: [serverb.lab.example.com]
ok: [serverd.lab.example.com]

TASK [Enable and start firewalld] *******************************************
ok: [serverd.lab.example.com]
ok: [serverb.lab.example.com]

TASK [Open the port for http] ***********************************************
changed: [serverd.lab.example.com]
changed: [serverb.lab.example.com]

RUNNING HANDLER [restart httpd] *********************************************
changed: [serverb.lab.example.com]
changed: [serverd.lab.example.com]

PLAY [Install and configure web service] ************************************

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

TASK [Install httpd] ********************************************************
changed: [serverc.lab.example.com]
changed: [servera.lab.example.com]

TASK [Enable and start httpd] ***********************************************
changed: [serverc.lab.example.com]
changed: [servera.lab.example.com]

TASK [Tuning configuration installed] ***************************************
changed: [servera.lab.example.com]
changed: [serverc.lab.example.com]

TASK [Install firewalld] ****************************************************
ok: [servera.lab.example.com]
ok: [serverc.lab.example.com]

TASK [Enable and start firewalld] *******************************************
ok: [servera.lab.example.com]
ok: [serverc.lab.example.com]

TASK [Open the port for http] ***********************************************
changed: [servera.lab.example.com]
changed: [serverc.lab.example.com]
```

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

```
RUNNING HANDLER [restart httpd] ******************************************
changed: [serverc.lab.example.com]
changed: [servera.lab.example.com]

PLAY RECAP ***************************************************************
servera.lab.example.com    : ok=8    changed=2    unreachable=0    failed=0
serverb.lab.example.com    : ok=8    changed=5    unreachable=0    failed=0
serverc.lab.example.com    : ok=8    changed=5    unreachable=0    failed=0
serverd.lab.example.com    : ok=8    changed=5    unreachable=0    failed=0
```

评估

从 workstation 运行 **lab projects-review grade** 命令，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 projects-review grade
```

完成

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

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

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

总结

在本章中，您学到了：

- 主机模式用于指定应作为 play 或临时命令的目标的受管主机。
- 动态清单脚本可用于从目录服务或 Ansible 外部的其他来源生成受管主机的动态列表。
- Ansible 配置文件中的 `forks` 参数设置受管主机的最大并行连接数。
- 通过定义每个滚动更新批处理中的受管主机数，可以使用 `serial` 参数在受管主机上实施滚动更新。
- 您可以使用 **`import_playbook`** 功能将外部 play 文件合并到 playbook 中。
- 您可以使用 **`include_tasks`** 或 **`import_tasks`** 功能将外部任务文件合并到 playbook 中。

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

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

章 8

利用角色简化 PLAYBOOK

目标

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

培训目标

- 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。

章节

- 描述角色结构（及测验）
- 利用系统角色重用内容（及引导式练习）
- 创建角色（及引导式练习）
- 使用 Ansible Galaxy 部署角色（及引导式练习）

实验

- 利用角色简化 Playbook

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

描述角色结构

培训目标

学完本节后，您应能够描述角色的定义、它的结构，以及如何在 playbook 中使用角色。

利用角色构造 ANSIBLE PLAYBOOK

随着您开发更多的 playbook，您可能会发现有很多机会重复利用以前编写的 playbook 中的代码。或许，一个用于为某一应用配置 MySQL 数据库的 play 可以改变用途，通过利用不同的主机名、密码和用户来为另一个应用配置 MySQL 数据库。

但在现实世界中，这个 play 可能比较冗长且复杂，有许多包含或导入的文件，以及用于管理各种情况的任务和处理程序。将所有这些代码复制到另一 playbook 中可能并非易事。

Ansible 角色提供了一种方法，让您能以通用的方式更加轻松地重复利用 Ansible 代码。您可以在标准化目录结构中打包所有任务、变量、文件、模板，以及调配基础架构或部署应用所需的其他资源。只需通过复制相关的目录，将角色从一个项目复制到另一个项目。然后，您只需从一个 play 调用该角色就能执行它。

借助编写良好的角色，您可以从 playbook 中向角色传递调整其行为的变量，设置所有站点相关的主机名、IP 地址、用户名，或其他您在本地需要的具体详细信息。例如，部署数据库服务器的角色可能已编写为支持多个变量，这些变量用于设置主机名、数据库管理员用户和密码，以及需要为安装进行自定义的其他参数。角色的作者也可以确保在您选择不在 play 中设置变量值时，为这些变量设定合理的默认值。

Ansible 角色具有下列优点：

- 角色可以分组内容，从而与他人轻松共享代码
- 可以编写角色来定义系统类型的基本要素：Web 服务器、数据库服务器、Git 存储库，或满足其他用途
- 角色使得较大型项目更容易管理
- 角色可以由不同的管理员并行开发

除了自行编写、使用、重用和共享角色外，您也可以从其他来源获取角色。一些角色已包含在 rhel-system-roles 软件包中，作为红帽企业 Linux 的一部分。您也可以从 Ansible Galaxy 网站获取由社区提供支持的许多角色。您将在本章后文中进一步了解这些角色。

检查 ANSIBLE 角色结构

Ansible 角色由子目录和文件的标准化结构定义。顶级目录定义角色本身的名称。文件整理到子目录中，子目录按照各个文件在角色中的用途进行命名，如 **tasks** 和 **handlers**。**files** 和 **templates** 子目录中包含由其他 YAML 文件中的任务引用的文件。

以下 **tree** 命令显示了 **user.example** 角色的目录结构。

```
[user@host roles]$ tree user.example
user.example/
├── defaults
```

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

```
│   └── main.yml
├── files
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── README.md
├── tasks
│   └── main.yml
├── templates
├── tests
│   ├── inventory
│   └── test.yml
└── vars
    └── main.yml
```

Ansible 角色子目录

子目录	功能
defaults	此目录中的 **main.yml** 文件包含角色变量的默认值，使用角色时可以覆盖这些默认值。这些变量的优先级较低，应该在 play 中更改和自定义。
files	此目录包含由角色任务引用的静态文件。
handlers	此目录中的 **main.yml** 文件包含角色的处理程序定义。
meta	此目录中的 **main.yml** 文件包含与角色相关的信息，如作者、许可证、平台和可选的角色依赖项。
tasks	此目录中的 **main.yml** 文件包含角色的任务定义。
templates	此目录包含由角色任务引用的 Jinja2 模板。
tests	此目录可以包含清单和 **test.yml**playbook，可用于测试角色。
vars	此目录中的 **main.yml** 文件定义角色的变量值。这些变量通常用于角色内部用途。这些变量的优先级较高，在 playbook 中使用时不应更改。

并非每个角色都拥有所有这些目录。

定义变量和默认值

角色变量通过在角色目录层次结构中创建含有键值对的 **vars/main.yml** 文件来定义。与其他变量一样，这些角色变量在角色 YAML 文件中引用：**{{ VAR_NAME }}**。这些变量具有较高的优先级，无法被清单变量覆盖。这些变量旨在供角色的内部功能使用。

默认变量允许为可在 play 中使用的变量设置默认值，以配置角色或自定义其行为。它们通过在角色目录层次结构中创建含有键值对的 **defaults/main.yml** 文件来定义。默认变量具有任何可用变量中最低的优先级。它们很容易被包括清单变量在内的任何其他变量覆盖。这些变量旨在让你们在编写使用该角色的 play 时可以准确地自定义或控制它将要执行的操作。它们可用于向角色提供所需的信息，以正确地配置或部署某些对象。

海量视频题库 mytube.com QQ:556462
www.52mytube.com

在 **vars/main.yml** 或 **defaults/main.yml** 中定义具体的变量，但不要在两者中都定义。有意要覆盖变量的值时，应使用默认变量。

> **重要**
> 角色不应该包含特定于站点的数据。它们绝对不应包含任何机密，如密码或私钥。
>
> 这是因为角色应该是通用的、可以重复利用并自由共享。特定于站点的详细信息不应硬编码到角色中。
>
> 机密应当通过其他途径提供给角色。这是您可能要在调用角色时设置角色变量的一个原因。Play 中设置的角色变量可以提供机密，或指向含有该机密的 Ansible Vault 加密文件。

在 PLAYBOOK 中使用 ANSIBLE 角色

在 playbook 中使用角色非常简单。下例演示了调用 Ansible 角色的一种方式。

```
---
- hosts: remote.example.com
  roles:
    - role1
    - role2
```

对于每个指定的角色，角色任务、角色处理程序、角色变量和角色依赖项将按照顺序导入到 playbook 中。角色中的任何 `copy`、`script`、`template` 或 `include_tasks/import_tasks` 任务都可引用角色中相关的文件、模板或任务文件，且无需相对或绝对路径名称。Ansible 将分别在角色的 **files**、**templates** 或 **tasks** 子目录中寻找它们。

如果使用 **roles** 部分将角色导入到 play 中，这些角色会在您为该 play 定义的任何任务之前运行。

以下示例设置 **role2** 的两个角色变量 **var1** 和 **var2** 的值。使用 **role2** 时，任何 **defaults** 和 **vars** 变量都会被覆盖。

```
---
- hosts: remote.example.com
  roles:
    - role: role1
    - role: role2
      var1: val1
      var2: val2
```

以下是在此情形中您可能会看到的另一种等效的 YAML 语法：

```
---
- hosts: remote.example.com
  roles:
    - role: role1
    - { role: role2, var1: val1, var2: val2 }
```

尽管它更为精简，但在某些情况下它更加难以阅读。

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

重要

正如前面的示例中所示，内嵌设置的角色变量（角色参数）具有非常高的优先级。它们将覆盖大多数其他变量。

务必要谨慎，不要重复使用内嵌设置在 play 中任何其他位置的任何角色变量的名称，因为角色变量的值将覆盖清单变量和任何 play **vars**。

控制执行顺序

对于 playbook 中的每个 play，任务按照任务列表中的顺序来执行。执行完所有任务后，将执行任何通知的处理程序。

在角色添加到 play 中后，角色任务将添加到任务列表的开头。如果 play 中包含第二个角色，其任务列表添加到第一个角色之后。

角色处理程序添加到 play 中的方式与角色任务添加到 play 中相同。每个 play 定义一个处理程序列表。角色处理程序先添加到处理程序列表，后跟 play 的 **handlers** 部分中定义的任何处理程序。

在某些情形中，可能需要在角色之前执行一些 play 任务。若要支持这样的情形，可以为 play 配置 **pre_tasks** 部分。列在此部分中的所有任务将在执行任何角色之前执行。如果这些任务中有任何一个通知了处理程序，则这些处理程序任务也在角色或普通任务之前执行。

此外，play 也支持 **post_tasks** 关键字。这些任务在 play 的普通任务和它们通知的任何处理程序运行之后执行。

以下 play 演示了一个带有 **pre_tasks**、**roles**、**tasks**、**post_tasks** 和 **handlers** 的示例。一个 play 中通常不会同时包含所有这些部分。

```
- name: Play to illustrate order of execution
  hosts: remote.example.com
  pre_tasks:
    - debug:
        msg: 'pre-task'
      notify: my handler
  roles:
    - role1
  tasks:
    - debug:
        msg: 'first task'
      notify: my handler
  post_tasks:
    - debug:
        msg: 'post-task'
      notify: my handler
  handlers:
    - name: my handler
      debug:
        msg: Running my handler
```

在上例中，每个部分中都执行 debug 任务来通知 **my handler** 处理程序。**my handler** 任务执行了三次：

· 在执行了所有 **pre_tasks** 任务后

海量视频题库 myitpub.com QQ:556462 www.52myit.com

· 在执行了所有角色任务和 **tasks** 部分中的任务后

· 在执行了所有 **post_tasks** 后

除了将角色包含在 play 的 **roles** 部分中外，也可以使用普通任务将角色添加到 play 中。使用 `include_role` 模块可以动态包含角色，使用 `import_role` 模块则可静态导入角色。

以下 playbook 演示了如何通过 `include_role` 模块来利用任务包含角色。

```
- name: Execute a role as a task
  hosts: remote.example.com
  tasks:
    - name: A normal task
      debug:
        msg: 'first task'
    - name: A task to include role2 here
      include_role: role2
```

注意

`include_role` 模块是在 Ansible 2.3 中新增的，而 `import_role` 模块则是 Ansible 2.4 中新增的。

参考文献

角色 — Ansible 文档

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

海量视频题库 myitpub.com QQ:556546 2
www.52myit.com

小测验

描述角色结构

选择以下问题的正确答案：

1. 下列哪一句陈述最准确地描述了角色？

a. 让特定用户运行 Ansible playbook 的配置设置。
b. 用于数据中心的 playbook。
c. YAML 任务文件和支持项目的集合，它们整理在特定的结构中，以便能轻松共享、移植和重复使用。

2. 以下哪一项可以在角色中指定？

a. 处理程序
b. 任务
c. 模板
d. 变量
e. 上述全部

3. 哪一文件声明角色依赖项？

a. 使用该角色的 Ansible Playbook。
b. 角色层次结构中的 **`meta/main.yml`** 文件。
c. 项目目录中的 **`meta/main.yml`** 文件。
d. 角色依赖项无法在 Ansible 中定义。

4. 角色目录层次结构中的哪一文件应当包含可能用作角色参数的变量的初始值？

a. **`defaults/main.yml`**
b. **`meta/main.yml`**
c. **`vars/main.yml`**
d. 主机清单文件。

海量视频题库 myitpub.com QQ:556462
www.52myit.com

解决方案

描述角色结构

选择以下问题的正确答案：

1. 下列哪一句陈述最准确地描述了角色？

a. 让特定用户运行 Ansible playbook 的配置设置。

b. 用于数据中心的 playbook。

c. YAML 任务文件和支持项目的集合，它们整理在特定的结构中，以便能轻松共享、移植和重复使用。

2. 以下哪一项可以在角色中指定？

a. 处理程序

b. 任务

c. 模板

d. 变量

e. 上述全部

3. 哪一文件声明角色依赖项？

a. 使用该角色的 Ansible Playbook。

b. 角色层次结构中的 `meta/main.yml` 文件。

c. 项目目录中的 `meta/main.yml` 文件。

d. 角色依赖项无法在 Ansible 中定义。

4. 角色目录层次结构中的哪一文件应当包含可能用作角色参数的变量的初始值？

a. `defaults/main.yml`

b. `meta/main.yml`

c. `vars/main.yml`

d. 主机清单文件。

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

利用系统角色重用内容

培训目标

学完本节后，您应能够编写利用红帽企业 Linux 系统角色执行标准操作的 playbook。

红帽企业 LINUX 系统角色

自红帽企业 Linux 7.4 开始，操作系统随附了多个 Ansible 角色，作为 rhel-system-roles 软件包的一部分。在红帽企业 Linux 8 中，该软件包可从 AppStream 频道获取。以下是每个角色的简要描述：

RHEL 系统角色

名称	状态	角色描述
`rhel-system-roles.kdump`	全面支持	配置 `kdump` 崩溃恢复服务。
`rhel-system-roles.network`	全面支持	配置网络接口。
`rhel-system-roles.selinux`	全面支持	配置和管理 SELinux 自定义，包括 SELinux 模式、文件和端口上下文、布尔值设置，以及 SELinux 用户。
`rhel-system-roles.timesync`	全面支持	使用网络时间协议或精确时间协议配置时间同步。
`rhel-system-roles.postfix`	技术预览	使用 Postfix 服务将每个主机配置为邮件传输代理。
`rhel-system-roles.firewall`	开发中	配置主机的防火墙。
`rhel-system-roles.tuned`	开发中	配置 `tuned` 服务，以调优系统性能。

系统角色的目的是在多个版本之间标准化红帽企业 Linux 子系统的配置。使用系统角色来配置版本 6.10 及以上的任何红帽企业 Linux主机。

简化配置管理

举例而言，红帽企业 Linux 7 的建议时间同步服务为 `chronyd` 服务。但在红帽企业 Linux 6 中，建议的服务为 `ntpd` 服务。在混合了红帽企业 Linux 6 和 7 主机的环境中，管理员必须管理这两个服务的配置文件。

借助 RHEL 系统角色，管理员不再需要维护这两个服务的配置文件。管理员可以使用 `rhel-system-roles.timesync` 角色来配置红帽企业 Linux6 和 7 主机的时间同步。一个包含角色变量的简化 YAML 文件可以为这两种类型的主机定义时间同步配置。

海量视频题库 myitpub.com QQ:556462
www.52myit.com

支持 RHEL 系统角色

RHEL 系统角色衍生自 Ansible Galaxy 上的开源 Linux 系统角色项目。与 Linux 系统角色不同，RHEL 系统角色是作为标准红帽企业Linux 订阅的一部分而受到红帽的支持。RHEL 系统角色具有红帽企业 Linux 订阅附带的相同生命周期支持优势。

每个系统角色都经过测试，并且具有稳定性。全面支持的系统角色还具有稳定的接口。任何完全支持系统角色，红帽将努力确保角色变量在未来版本中保持不变。由于系统角色更改而导致的 playbook 重构应该会极少。

技术预览系统角色可能会在未来版本利用不同的角色变量。对于包含任何技术预览角色的 playbook，建议进行集成测试。如果角色的未来版本中出现角色变量更改，则 playbook 可能需要重构。

其他角色正在上游 Linux 系统角色项目中开发，但尚未通过 RHEL 订阅提供。这些角色可通过 Ansible Galaxy 获取。

安装 RHEL 系统角色

RHEL 系统角色由 rhel-system-roles 软件包提供，该软件包可从 `AppStream` 频道获取。在 Ansible 控制节点上安装该软件包。

采用下列程序来安装 rhel-system-roles 软件包。该程序假定控制节点已注册到红帽企业 Linux 订阅，并且安装了 Ansible 引擎。有关更多信息，请参见关于安装 Ansible 的章节。

1. 安装 RHEL 系统角色。

```
[root@host ~]# yum install rhel-system-roles
```

安装后，RHEL 系统角色位于 **/usr/share/ansible/roles** 目录中：

```
[root@host ~]# ls -l /usr/share/ansible/roles/
total 20
...output omitted... linux-system-roles.kdump -> rhel-system-roles.kdump
...output omitted... linux-system-roles.network -> rhel-system-roles.network
...output omitted... linux-system-roles.postfix -> rhel-system-roles.postfix
...output omitted... linux-system-roles.selinux -> rhel-system-roles.selinux
...output omitted... linux-system-roles.timesync -> rhel-system-roles.timesync
...output omitted... rhel-system-roles.kdump
...output omitted... rhel-system-roles.network
...output omitted... rhel-system-roles.postfix
...output omitted... rhel-system-roles.selinux
...output omitted... rhel-system-roles.timesync
```

每个角色的对应上游名称都链接到 RHEL 系统角色。这使角色可在 playbook 中通过任一名称来引用。

红帽企业 Linux 中的默认 `roles_path` 在路径中包含 **/usr/share/ansible/roles**，因此在 playbook 引用这些角色时 Ansible 应该会自动找到它们。

海量视频题库 myitpub.com Q:556462
www.52myit.com

> **注意**
> 如果在当前 Ansible 配置文件中覆盖了 `roles_path`，设置了环境变量 `ANSIBLE_ROLES_PATH`，或者 `roles_path` 中更早列出的目录下存在另一个同名的角色，则 Ansible 可能无法找到系统角色。

访问 RHEL 系统角色的文档

安装后，RHEL 系统角色的文档位于 **/usr/share/doc/rhel-system-roles-<version>/** 目录中。文档按照子系统整理到子目录中：

```
[root@host ~]# ls -l /usr/share/doc/rhel-system-roles/
total 4
drwxr-xr-x. ...output omitted... kdump
drwxr-xr-x. ...output omitted... network
drwxr-xr-x. ...output omitted... postfix
drwxr-xr-x. ...output omitted... selinux
drwxr-xr-x. ...output omitted... timesync
```

每个角色的文档目录均包含一个 **README.md** 文件。**README.md** 文件含有角色的说明，以及角色用法信息。

README.md 文件也会说明影响角色行为的角色变量。通常，**README.md** 文件中含有一个 playbook 代码片段，用于演示常见配置场景的变量设置。

部分角色文档目录中含有示例 playbook。首次使用某一角色时，请查看文档目录中的任何额外示例 playbook。

RHEL 系统角色的角色文档与 Linux 系统角色的文档相匹配。使用 Web 浏览器来访问位于 Ansible Galaxy 网站 `https://galaxy.ansible.com` 上的上游角色的角色文档。

时间同步角色示例

假设您需要在服务器上配置 NTP 时间同步。您可以自行编写自动化来执行每一个必要的任务。但是，RHEL 系统角色中有一个可以执行此操作的角色，那就是 `rhel-system-roles.timesync`。

该角色的详细记录位于 **/usr/share/doc/rhel-system-roles/timesync** 目录下的 **README.md** 中。此文件说明了影响角色行为的所有变量，还包含演示了不同时间同步配置的三个 playbook 代码片段。

为了手动配置 NTP 服务器，该角色具有一个名为 `timesync_ntp_servers` 的变量。此变量取一个要使用的 NTP 服务器的列表作为值。列表中的每一项均由一个或多个属性构成。两个关键属性如下：

`timesync_ntp_servers` 属性

属性	用途
`hostname`	要与其同步的 NTP 服务器的主机名。
`iburst`	一个布尔值，用于启用或禁用快速初始同步。在角色中默认为 **no**，但您通常应该将该属性设为 **yes**。

海量视频题库 myitpub.com QQ:556462 www.52myit.com

根据这一信息，以下示例 play 使用 `rhel-system-roles.timesync` 角色将受管主机配置为利用快速初始同步从三个 NTP 服务器获取时间。此外，还添加了一个任务，以使用 `timezone` 模块将主机的时区设为 UTC。

```
- name: Time Synchronization Play
  hosts: servers
  vars:
    timesync_ntp_servers:
      - hostname: 0.rhel.pool.ntp.org
        iburst: yes
      - hostname: 1.rhel.pool.ntp.org
        iburst: yes
      - hostname: 2.rhel.pool.ntp.org
        iburst: yes
    timezone: UTC

  roles:
    - rhel-system-roles.timesync

  tasks:
    - name: Set timezone
      timezone:
        name: "{{ timezone }}"
```

> **注意**
> 如果要设置不同的时区，您可以使用 **`tzselect`** 命令查询其他有效的值。也可以使用 **`timedatectl`** 命令来检查当前的时钟设置。

此示例在 play 的 **`vars`** 部分中设置角色变量，但更好的做法可能是将它们配置为主机或主机组的清单变量。

假如一个 playbook 项目具有以下结构：

```
[root@host playbook-project]# tree
.
├── ansible.cfg
├── group_vars
│   └── servers
│       └── timesync.yml❶
├── inventory
└── timesync_playbook.yml❷
```

❶ 定义时间同步变量，覆盖清单中 `servers` 组内主机的角色默认值。此文件看起来类似于：

海量视频题库 myitpub.com QQ:556546 2
www.52myit.com

```
timesync_ntp_servers:
  - hostname: 0.rhel.pool.ntp.org
    iburst: yes
  - hostname: 1.rhel.pool.ntp.org
    iburst: yes
  - hostname: 2.rhel.pool.ntp.org
    iburst: yes
timezone: UTC
```

❷ Playbook 的内容简化为：

```
- name: Time Synchronization Play
  hosts: servers
  roles:
    - rhel-system-roles.timesync
  tasks:
    - name: Set timezone
      timezone:
        name: "{{ timezone }}"
```

该结构可清楚地分隔角色、playbook 代码和配置设置。Playbook 代码简单易读，应该不需要复杂的重构。角色内容由红帽进行维护并提供支持。所有设置都以清单变量的形式进行处理。

该结构还支持动态的异构环境。具有新的时间同步要求的主机可能会放置到新的主机组中。相应的变量在 YAML 文件中定义，并放置到相应的 **group_vars**（或 **host_vars**）子目录中。

SELINUX 角色示例

再举一个例子，**rhel-system-roles.selinux** 角色可以简化 SELinux 配置设置的管理。它通过利用 SELinux 相关的 Ansible 模块来实施。与自行编写任务相比，使用此角色的优势是它能让您摆脱编写这些任务的职责。取而代之，您将为角色提供变量以对其进行配置，且角色中维护的代码将确保应用您需要的 SELinux 配置。

此角色可以执行的任务包括：

- 设置 enforcing 或 permissive 模式
- 对文件系统层次结构的各部分运行 **restorecon**
- 设置 SELinux 布尔值
- 永久设置 SELinux 文件上下文
- 设置 SELinux 用户映射

调用 SELinux 角色

有时候，SELinux 角色必须确保重新引导受管主机，以便能够完整应用其更改。但是，它本身从不会重新引导主机。如此一来，您便可以控制重新引导的处理方式。不过，这意味着在 play 中正确使用此角色要比平常复杂一些。

其工作方式为，该角色将一个布尔值变量 **selinux_reboot_required** 设为 **true**，如果需要重新引导，则失败。您可以使用 **block**/**rescue** 结构来从失败中恢复，具体操作为：如果该变量未设为 **true**，则让 play 失败，如果值是 **true**，则重新引导受管主机并重新运行该角色。Play 中的块看起来应该类似于：

海量视频题库 mypub.com QQ:5565462 www.52mpt.com

```
  - name: Apply SELinux role
    block:
      - include_role:
          name: rhel-system-roles.selinux
    rescue:
      - name: Check for failure for other reasons than required reboot
        fail:
        when: not selinux_reboot_required

      - name: Restart managed host
        reboot:

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

配置 SELinux 角色

用于配置 `rhel-system-roles.selinux` 角色的变量的详细记录位于其 **`README.md`** 文件中。以下示例演示了使用此角色的一些方法。

`selinux_state` 变量设置 SELinux 的运行模式。它可以设为 **`enforcing`**、**`permissive`** 或 **`disabled`**。如果未设置，则不更改模式。

```
selinux_state: enforcing
```

`selinux_booleans` 变量取一个要调整的 SELinux 布尔值的列表作为值。列表中的每一项是变量的散列/字典：布尔值的 `name`、`state`（它应是 **`on`** 还是 **`off`**），以及该设置是否应在重新引导后 `persistent`。

本例将 `httpd_enable_homedirs` 永久设为 **`on`**：

```
selinux_booleans:
  - name: 'httpd_enable_homedirs'
    state: 'on'
    persistent: 'yes'
```

`selinux_fcontext` 变量取一个要永久设置（或删除）的文件上下文的列表作为值。它的工作方式与 **`selinux fcontext`** 命令非常相似。

以下示例确保策略中包含一条规则，用于将 **`/srv/www`** 下所有文件的默认 SELinux 类型设为 `httpd_sys_content_t`。

```
selinux_fcontexts:
  - target: '/srv/www(/.*)?'
    setype: 'httpd_sys_content_t'
    state: 'present'
```

`selinux_restore_dirs` 变量指定要对其运行 **`restorecon`** 的目录的列表：

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