```
tasks:
  - name: copy demo.example.conf configuration template
    template:
      src: /var/lib/templates/demo.example.conf.template
      dest: /etc/httpd/conf.d/demo.example.conf
    notify:
      - restart mysql
      - restart apache

handlers:
  - name: restart mysql
    service:
      name: mariadb
      state: restarted

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

描述使用处理程序的好处

如 Ansible 文档中所述，使用处理程序时需要牢记几个重要事项：

- 处理程序始终按照 play 的 **handlers** 部分指定的顺序运行。它们不按在任务中由 **notify** 语句列出的顺序运行，或按任务通知它们的顺序运行。
- 处理程序通常在相关 play 中的所有其他任务完成后运行。playbook 的 **tasks** 部分中某一任务调用的处理程序，将等到 **tasks** 下的所有任务都已处理后才会运行。（这有一些小的例外。）
- 处理程序名称存在于各 play 命名空间中。如果两个处理程序被错误地给予相同的名称，则仅会运行一个。
- 即使有多个任务通知处理程序，该处理程序依然仅运行一次。如果没有任务通知处理程序，它就不会运行。
- 如果包含 **notify** 语句的任务没有报告 **changed** 结果（例如，软件包已安装并且任务报告 **ok**），则处理程序不会获得通知。处理程序将被跳过，直到有其他任务通知它。只有相关任务报告了 **changed** 状态，Ansible 才会通知处理程序。

重要

处理程序用于在任务对受管主机进行更改时执行额外操作。它们不应用作正常任务的替代。

参考文献

Playbook 简介 — Ansible 文档

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

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

指导练习

实施处理程序

在本练习中，您将在 playbook 中实施处理程序。

成果

您应能够在 playbook 中定义处理程序，并向它们通知配置更改。

在你开始之前

在 workstation 上运行 **lab control-handlers start**，以配置本练习的环境。该脚本将创建 **/home/student/control-handlers** 项目目录，并下载练习所需的 Ansible 配置文件和主机清单文件。项目目录还包含部分完成的 playbook **configure_db.yml**。

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

1. 在 workstation.lab.example.com 上，打开一个新终端并更改到 **/home/student/control-handlers** 项目目录。

```
[student@workstation ~]$ cd ~/control-handlers
[student@workstation control-handlers]$
```

2. 在该目录中，使用文本编辑器编辑 **configure_db.yml** playbook 文件。此 playbook 将安装和配置一个数据库服务器。当数据库服务器配置更改时，playbook 将触发数据库服务重新启动并配置数据库管理密码。

   2.1. 使用文本编辑器，查看 **configure_db.yml** playbook。从一些变量的初始化开始：

```
---
- name: MariaDB server is installed
  hosts: databases
  vars:
    db_packages: ❶
      - mariadb-server
      - python3-PyMySQL
    db_service: mariadb ❷
    resources_url: http://materials.example.com/labs/control-handlers ❸
    config_file_url: "{{ resources_url }}/my.cnf.standard" ❹
    config_file_dst: /etc/my.cnf ❺
  tasks:
```

❶ **db_packages** 定义要为数据库服务安装的软件包的名称。
❷ **db_service** 定义数据库服务的名称。
❸ **resources_url** 表示远程配置文件所在的资源目录的 URL。
❹ **config_file_url** 表示要安装的数据库配置文件的 URL。
❺ **config_file_dst** 是受管主机上已安装配置文件的位置。

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

2.2. 在 **configure_db.yml** 文件中定义一个任务，它使用 yum 模块安装由 **db_packages** 变量定义的必要数据库软件包。如果该任务更改了系统，则表示未安装数据库，您需要通知 **set db password** 处理程序，设置初始数据库用户和密码。请记住，处理程序任务（如果获得通知）在 **tasks** 部分中的每个任务都已运行之前将不会运行。

该任务应当如下所示：

```
tasks:
  - name: "{{ db_packages }} packages are installed"
    yum:
      name: "{{ db_packages }}"
      state: present
    notify:
      - set db password
```

2.3. 添加一个任务，以启动并启用数据库服务。该任务应当如下所示：

```
  - name: Make sure the database service is running
    service:
      name: "{{ db_service }}"
      state: started
      enabled: true
```

2.4. 添加一项任务，它使用 get_url 模块将 **my.cnf.standard** 下载到受管主机上的 **/etc/my.cnf**。添加通知 **restart db service** 处理程序的条件，以在配置文件更改后重新启动数据库服务。该任务应当如下所示：

```
  - name: The {{ config_file_dst }} file has been installed
    get_url:
      url: "{{ config_file_url }}"
      dest: "{{ config_file_dst }}"
      owner: mysql
      group: mysql
      force: yes
    notify:
      - restart db service
```

2.5. 添加 **handlers** 关键字以定义处理程序任务的开头。定义第一个处理程序 **restart db service**，它重新启动 mariadb 服务。它应显示如下：

```
handlers:
  - name: restart db service
    service:
      name: "{{ db_service }}"
      state: restarted
```

2.6. 定义第二个处理程序 **set db password**，它设置数据库服务的管理密码。该处理程序使用 mysql_user 模块来执行命令。该处理程序应当如下所示：

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

```
    - name: set db password
      mysql_user:
        name: root
        password: redhat
```

完成时，该 playbook 应当如下所示：

```
---
- name: MariaDB server is installed
  hosts: databases
  vars:
    db_packages:
      - mariadb-server
      - python3-PyMySQL
    db_service: mariadb
    resources_url: http://materials.example.com/labs/control-handlers
    config_file_url: "{{ resources_url }}/my.cnf.standard"
    config_file_dst: /etc/my.cnf
  tasks:
    - name: "{{ db_packages }} packages are installed"
      yum:
        name: "{{ db_packages }}"
        state: present
      notify:
        - set db password

    - name: Make sure the database service is running
      service:
        name: "{{ db_service }}"
        state: started
        enabled: true

    - name: The {{ config_file_dst }} file has been installed
      get_url:
        url: "{{ config_file_url }}"
        dest: "{{ config_file_dst }}"
        owner: mysql
        group: mysql
        force: yes
      notify:
        - restart db service

  handlers:
    - name: restart db service
      service:
        name: "{{ db_service }}"
        state: restarted

    - name: set db password
      mysql_user:
        name: root
        password: redhat
```

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

3. 在运行该 playbook 前，使用 **--syntax-check** 选项通过运行 **ansible-playbook** 来验证其语法是否正确。如果报告任何错误，请更正后再继续下一步。您应看到类似于下文的输出：

```
[student@workstation control-handlers]$ ansible-playbook configure_db.yml \
> --syntax-check

playbook: configure_db.yml
```

4. 运行 **configure_db.yml** playbook。输出中显示处理程序正在执行。

```
[student@workstation control-handlers]$ ansible-playbook configure_db.yml

PLAY [Installing MariaDB server] ******************************************

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

TASK [['mariadb-server', 'python3-PyMySQL'] packages are installed] **********
changed: [servera.lab.example.com]

TASK [Make sure the database service is running] ******************************
changed: [servera.lab.example.com]

TASK [The /etc/my.cnf file has been installed] ********************************
changed: [servera.lab.example.com]

RUNNING HANDLER [restart db service] ******************************************
changed: [servera.lab.example.com]

RUNNING HANDLER [set db password] *********************************************
changed: [servera.lab.example.com]

PLAY RECAP ****************************************************************
servera.lab.example.com    : ok=6    changed=5    unreachable=0    failed=0
```

5. 再次运行该 playbook。

```
[student@workstation control-handlers]$ ansible-playbook configure_db.yml

PLAY [Installing MariaDB server] ******************************************

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

TASK [['mariadb-server', 'python3-PyMySQL'] packages are installed] **********
ok: [servera.lab.example.com]

TASK [Make sure the database service is running] ******************************
ok: [servera.lab.example.com]

TASK [The /etc/my.cnf file has been installed] ********************************
```

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

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

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

这一次处理程序被跳过。如果将来更改远程配置文件，执行 playbook 将触发 **restart db service** 处理程序，而不是 **set db password** 处理程序。

完成

在 workstation 上，运行 **lab control-handlers finish** 脚本来清理本练习中创建的资源。

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

本引导式练习到此结束。

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

处理任务失败

培训目标

学完本节后，您应能够控制任务失败时发生的情况，以及导致任务失败的条件。

管理 PLAY 中的任务错误

Ansible 评估各任务的返回代码，从而确定任务是成功还是失败。通常而言，当任务失败时，Ansible 将立即在该主机上中止 play 的其余部分并且跳过所有后续任务。

但有些时候，您可能希望即使在任务失败时也继续执行 play。例如，您或许预期特定任务有可能会失败，并且希望通过有条件地运行某项其他任务来恢复。有多种 Ansible 功能可用于管理任务错误。

忽略任务失败

默认情况下，任务失败时 play 会中止。不过，可以通过忽略失败的任务来覆盖此行为。您可以在任务中使用 **ignore_errors** 关键字来实现此目的。

下列代码片段演示了如何在任务中使用 **ignore_errors**，以便在任务失败时也继续在主机上执行 playbook。例如，如果 notapkg 软件包不存在，则 yum 模块将失败，但若将 ignore_errors 设为 **yes**，则执行将继续。

```
- name: Latest version of notapkg is installed
  yum:
    name: notapkg
    state: latest
  ignore_errors: yes
```

任务失败后强制执行处理程序

通常而言，如果任务失败并且 play 在该主机上中止，则收到 play 中早前任务通知的处理程序将不会运行。如果您在 play 中设置 **force_handlers: yes** 关键字，则即使 play 因为后续任务失败而中止也会调用被通知的处理程序。

下列代码片段演示了如何在 play 中使用 **force_handlers** 关键字，以便在任务失败时也强制执行相应的处理程序：

```
---
- hosts: all
  force_handlers: yes
  tasks:
    - name: a task which always notifies its handler
      command: /bin/true
      notify: restart the database

    - name: a task which fails because the package doesn't exist
      yum:
```

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

```
        name: notapkg
        state: latest

    handlers:
      - name: restart the database
        service:
          name: mariadb
          state: restarted
```

注意

请记住，处理程序会在任务报告 **changed** 结果时获得通知，而在任务报告 **ok** 或 **failed** 结果时不会获得通知。

指定任务失败条件

您可以在任务中使用 **failed_when** 关键字来指定表示任务已失败的条件。这通常与命令模块搭配使用，这些模块可能成功执行了某一命令，但命令的输出可能指示了失败。

例如，您可以运行输出错误消息的脚本，并使用该消息定义任务的失败状态。下列代码片段演示了如何在任务中使用 **failed_when** 关键字：

```
tasks:
  - name: Run user creation script
    shell: /usr/local/bin/create_users.sh
    register: command_result
    failed_when: "'Password missing' in command_result.stdout"
```

`fail` 模块也可用于强制任务失败。上面的场景也可以编写为两个任务：

```
tasks:
  - name: Run user creation script
    shell: /usr/local/bin/create_users.sh
    register: command_result
    ignore_errors: yes

  - name: Report script failure
    fail:
      msg: "The password is missing in the output"
    when: "'Password missing' in command_result.stdout"
```

您可以使用 `fail` 模块为任务提供明确的失败消息。此方法还支持延迟失败，允许您运行中间任务以完成或回滚其他更改。

指定何时任务报告 “Changed” 结果

当任务对托管主机进行了更改时，会报告 **changed** 状态并通知处理程序。如果任务不需要进行更改，则会报告 **ok** 并且不通知处理程序。

changed_when 关键字可用于控制任务在何时报告它已进行了更改。例如，下一个示例中的 **shell** 模块将用于获取供后续任务使用的 Kerberos 凭据。它通常会在运行时始终报告 **changed**。为抑制这种更改，应设置 **changed_when: false**，以便它仅报告 **ok** 或 **failed**。

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

```
  - name: get Kerberos credentials as "admin"
    shell: echo "{{ krb_admin_pass }}" | kinit -f admin
    changed_when: false
```

以下示例使用 **shell** 模块，根据通过已注册变量收集的模块的输出来报告 **changed**：

```
tasks:
  - shell:
      cmd: /usr/local/bin/upgrade-database
    register: command_result
    changed_when: "'Success' in command_result.stdout"
    notify:
      - restart_database

handlers:
  - name: restart_database
     service:
       name: mariadb
       state: restarted
```

Ansible 块和错误处理

在 playbook 中，块是对任务进行逻辑分组的子句，可用于控制任务的执行方式。例如，任务块可以含有 **when** 关键字，以将某一条件应用到多个任务：

```
- name: block example
  hosts: all
  tasks:
    - name: installing and configuring Yum versionlock plugin
      block:
      - name: package needed by yum
        yum:
          name: yum-plugin-versionlock
          state: present
      - name: lock version of tzdata
        lineinfile:
          dest: /etc/yum/pluginconf.d/versionlock.list
          line: tzdata-2016j-1
          state: present
      when: ansible_distribution == "RedHat"
```

通过块，也可结合 **rescue** 和 **always** 语句来处理错误。如果块中的任何任务失败，则执行其 **rescue** 块中的任务来进行恢复。在 block 子句中的任务以及 rescue 子句中的任务（如果出现故障）运行之后，**always** 子句中的任务运行。总结：

- **block**：定义要运行的主要任务。
- **rescue**：定义要在 **block** 子句中定义的任务失败时运行的任务。
- **always**：定义始终都独立运行的任务，不论 **block** 和 **rescue** 子句中定义的任务是成功还是失败。

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

以下示例演示了如何在 playbook 中实施块。即使 **block** 子句中定义的任务失败，**rescue** 和 **always** 子句中定义的任务也会执行。

```
tasks:
  - name: Upgrade DB
    block:
      - name: upgrade the database
        shell:
          cmd: /usr/local/lib/upgrade-database
    rescue:
      - name: revert the database upgrade
        shell:
          cmd: /usr/local/lib/revert-database
    always:
      - name: always restart the database
        service:
          name: mariadb
          state: restarted
```

block 中的 **when** 条件也会应用到其 **rescue** 和 **always** 子句（若存在）。

参考文献

Playbook 中的错误处理 — Ansible 文档
https://docs.ansible.com/ansible/latest/user_guide/playbooks_error_handling.html

错误处理 — 块 — Ansible 文档
https://docs.ansible.com/ansible/latest/user_guide/playbooks_blocks.html#error-handling

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

指导练习

处理任务失败

在本练习中，您将探索在 Ansible Playbook 中处理任务失败的不同方法。

成果

您应能够：

- 在执行 playbook 期间忽略失败的命令。
- 强制执行处理程序。
- 覆盖造成任务失败的条目。
- 覆盖任务的 **changed** 状态。
- 在 playbook 中实施 **block**、**rescue** 和 **always**。

在你开始之前

在 `workstation` 上，运行实验起始脚本，以确认环境已就绪，可开始实验。该脚本将创建工作目录 **/home/student/control-errors**。

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

1. 在 `workstation.lab.example.com` 上，切换到 **/home/student/control-errors** 项目目录。

```
[student@workstation ~]$ cd ~/control-errors
[student@workstation control-errors]$
```

2. 实验脚本创建了一个 Ansible 配置文件，以及一个将服务器 `servera.lab.example.com` 包含在 **databases** 组中的清单文件。检查文件后再继续。

3. 创建 **playbook.yml** playbook，它将含有一个包含两个任务的 play。编写第一个任务，有意包含一个将导致失败的错误。

3.1. 在文本编辑器中打开 playbook。定义三个变量：`web_package`（值为 **http**）、`db_package`（值为 **mariadb-server**）和 `db_service`（值为 **mariadb**）。这些变量将用于安装需要的软件包并启动服务器。

http 值是软件包名称中一个有意的错误。该文件应当如下所示：

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

```
---
- name: Task Failure Exercise
  hosts: databases
  vars:
    web_package: http
    db_package: mariadb-server
    db_service: mariadb
```

3.2. 定义两个任务，它们使用 **yum** 模块以及 web_package 和 db_package 这两个变量。任务将安装所需的软件包。这些任务应当如下所示：

```
  tasks:
    - name: Install {{ web_package }} package
      yum:
        name: "{{ web_package }}"
        state: present

    - name: Install {{ db_package }} package
      yum:
        name: "{{ db_package }}"
        state: present
```

4. 运行 playbook，并观察 play 的输出。

```
[student@workstation control-errors]$ ansible-playbook playbook.yml

PLAY [Task Failure Exercise] ***************************************************

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

TASK [Install http package] ****************************************************
fatal: [servera.lab.example.com]: FAILED! => {"changed": false, "failures":
 ["No package http available."], "msg": "Failed to install some of the specified
 packages", "rc": 1, "results": []}

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

任务将失败，因为不存在名为 **http** 的现有软件包。由于第一个任务失败，因此未运行第二个任务。

5. 通过添加 **ignore_errors** 关键字更新第一个任务，以忽略任何错误。这些任务应当如下所示：

```
  tasks:
    - name: Install {{ web_package }} package
      yum:
        name: "{{ web_package }}"
        state: present
      ignore_errors: yes
```

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

```
  - name: Install {{ db_package }} package
    yum:
      name: "{{ db_package }}"
      state: present
```

6. 再次运行 playbook，并观察 play 的输出。

```
[student@workstation control-errors]$ ansible-playbook playbook.yml

PLAY [Task Failure Exercise] ***************************************************

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

TASK [Install http package] ****************************************************
fatal: [servera.lab.example.com]: FAILED! => {"changed": false, "failures":
 ["No package http available."], "msg": "Failed to install some of the specified
 packages", "rc": 1, "results": []}
...ignoring

TASK [Install mariadb-server package] ******************************************
changed: [servera.lab.example.com]

PLAY RECAP *********************************************************************
servera.lab.example.com    : ok=3    changed=1    unreachable=0    failed=0
```

尽管第一个任务失败，但 Ansible 仍然执行了第二个任务。

7. 在这一步中，您将设置 **block** 关键字，以便您可以试验它的运作方式。

7.1. 更新 playbook，将第一个任务嵌套在 **block** 子句中。删除设置 **ignore_errors: yes** 的行。这个块应当如下所示：

```
- name: Attempt to set up a webserver
  block:
    - name: Install {{ web_package }} package
      yum:
        name: "{{ web_package }}"
        state: present
```

7.2. 将安装 mariadb-server 软件包的任务嵌套在 **rescue** 子句中。如果 **block** 子句中所列的任务失败，此任务将执行。该 block 子句应当如下所示：

```
  rescue:
    - name: Install {{ db_package }} package
      yum:
        name: "{{ db_package }}"
        state: present
```

7.3. 最后，添加一个 **always** 子句，以在安装后使用 **service** 模块启动数据库服务器。该子句应当如下所示：

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

```
    always:
      - name: Start {{ db_service }} service
        service:
          name: "{{ db_service }}"
          state: started
```

7.4. 完成的任务部分应当如下所示：

```
tasks:
  - name: Attempt to set up a webserver
    block:
      - name: Install {{ web_package }} package
        yum:
          name: "{{ web_package }}"
          state: present
    rescue:
      - name: Install {{ db_package }} package
        yum:
          name: "{{ db_package }}"
          state: present
    always:
      - name: Start {{ db_service }} service
        service:
          name: "{{ db_service }}"
          state: started
```

8. 现在再次运行 playbook 并观察输出。

8.1. 运行 playbook。块中确保已安装 **web_package** 的任务失败，这将导致 **rescue** 块中的任务运行。然后，**always** 块中的任务将运行。

```
[student@workstation control-errors]$ ansible-playbook playbook.yml

PLAY [Task Failure Exercise] ***************************************************

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

TASK [Install http package] ****************************************************
fatal: [servera.lab.example.com]: FAILED! => {"changed": false, "failures":
 ["No package http available."], "msg": "Failed to install some of the specified
 packages", "rc": 1, "results": []}

TASK [Install mariadb-server package] ******************************************
ok: [servera.lab.example.com]

TASK [Start mariadb service] ***************************************************
changed: [servera.lab.example.com]

PLAY RECAP *********************************************************************
servera.lab.example.com    : ok=3    changed=1    unreachable=0    failed=1
```

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

8.2. 编辑 playbook，更正 **web_package** 变量的值，使其显示为 **httpd**。这样，下次运行该 playbook 时块中的任务会成功。

```
vars:
  web_package: httpd
  db_package: mariadb-server
  db_service: mariadb
```

8.3. 再次运行该 playbook。这一次，块中的任务不会失败。这将导致 **rescue** 部分中的任务被忽略。**always** 中的任务仍然会运行。

```
[student@workstation control-errors]$ ansible-playbook playbook.yml

PLAY [Task Failure Exercise] ***************************************************

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

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

TASK [Start mariadb service] ***************************************************
ok: [servera.lab.example.com]

PLAY RECAP *********************************************************************
servera.lab.example.com    : ok=3    changed=1    unreachable=0    failed=0
```

9. 这一步将探索如何控制使任务对托管主机报告为 “changed” 的条件。

9.1. 编辑 playbook，在 play 的开头、**block** 的前面添加两个任务。第一个任务使用 **command** 模块运行 **date** 命令，并将结果注册在 **command_result** 变量中。第二个任务使用 **debug** 模块显示第一个任务的命令的标准输出。

```
tasks:
  - name: Check local time
    command: date
    register: command_result

  - name: Print local time
    debug:
      var: command_result.stdout
```

9.2. 运行 playbook。您应当会看到运行 **command** 模块的第一个任务报告 **changed**，尽管它没有更改远程系统；它只是收集了有关时间的信息。这是因为 **command** 模块无法区分收集数据的命令和更改状态的命令。

```
[student@workstation control-errors]$ ansible-playbook playbook.yml

PLAY [Task Failure Exercise] ***************************************************

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

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

```
TASK [Check local time] ********************************************************
changed: [servera.lab.example.com]

TASK [Print local time] ********************************************************
ok: [servera.lab.example.com] => {
    "command_result.stdout": "mié mar 27 08:07:08 EDT 2019"
}

TASK [Install httpd package] ***************************************************
ok: [servera.lab.example.com]

TASK [Start mariadb service] ***************************************************
ok: [servera.lab.example.com]

PLAY RECAP *********************************************************************
servera.lab.example.com    : ok=5    changed=1    unreachable=0    failed=0
```

如果您再次运行该 playbook，**Check local time** 任务将再次返回 **changed**。

9.3. 此 **command** 任务没有更改托管主机，所以不应每次运行时都报告 **hanged**。由于您知道该任务绝不会更改托管主机，因此在任务中添加 **changed_when: false** 行以抑制更改。

```
tasks:
  - name: Check local time
    command: date
    register: command_result
    changed_when: false

  - name: Print local time
    debug:
      var: command_result.stdout
```

9.4. 再次运行 playbook，并注意任务现在报告 **ok**，但该任务仍会运行，并且仍然会在变量中保存时间。

```
[student@workstation control-errors]$ ansible-playbook playbook.yml

PLAY [Task Failure Exercise] ***************************************************

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

TASK [Check local time] ********************************************************
ok: [servera.lab.example.com]

TASK [Print local time] ********************************************************
ok: [servera.lab.example.com] => {
    "command_result.stdout": "mié mar 27 08:08:36 EDT 2019"
}

TASK [Install httpd package] ***************************************************
ok: [servera.lab.example.com]
```

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

```
TASK [Start mariadb service] ***************************************************
ok: [servera.lab.example.com]

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

10. 作为最后一个练习，编辑 playbook 来探索 **failed_when** 关键字与任务的交互方式。

10.1.编辑 **Install {{ web_package }} package** 任务，使得它在 web_package 的值为 **httpd** 时报告为已失败。由于情况就是如此，因此在运行这个 play 时任务会报告失败。

请注意缩进，以确保在任务中正确设置了该关键字。

```
- block:
    - name: Install {{ web_package }} package
      yum:
        name: "{{ web_package }}"
        state: present
      failed_when: web_package == "httpd"
```

10.2.运行 playbook。

```
[student@workstation control-errors]$ ansible-playbook playbook.yml

PLAY [Task Failure Exercise] ***************************************************

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

TASK [Check local time] ********************************************************
ok: [servera.lab.example.com]

TASK [Print local time] ********************************************************
ok: [servera.lab.example.com] => {
    "command_result.stdout": "mié mar 27 08:09:35 EDT 2019"
}

TASK [Install httpd package] ***************************************************
fatal: [servera.lab.example.com]: FAILED! => {"changed": false,
 "failed_when_result": true, "msg": "Nothing to do", "rc": 0, "results":
 ["Installed: httpd"]}

TASK [Install mariadb-server package] ******************************************
ok: [servera.lab.example.com]

TASK [Start mariadb service] ***************************************************
ok: [servera.lab.example.com]

PLAY RECAP *********************************************************************
servera.lab.example.com    : ok=5    changed=0    unreachable=0    failed=1
```

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

仔细查看输出内容。**Install httpd package** 任务报告它已失败，但实际上此任务已运行，并确保首先安装了软件包。**failed_when** 关键字更改了任务运行之后任务报告的状态；并没有更改任务本身的行为。

不过，报告的失败状态可能会改变 play 中其余部分的行为。由于该任务在块中并且报告了失败，因此运行了这个块的 **rescue** 部分中的 **Install mariadb-server package** 任务。

完成

在 workstation 上，运行 **lab control-errors finish** 脚本来清理本练习中创建的资源。

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

本引导式练习到此结束。

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

开放研究实验

实施任务控制

任务执行清单

在本实验中，您将安装 Apache Web 服务器并使用 `mod_ssl` 对其进行保护。您将在 playbook 中使用条件、处理程序和任务失败处理来部署环境。

成果

您应该能够在 Ansible Playbook 中定义条件，设置迭代元素的循环，在 playbook 中定义处理程序，以及处理任务错误。

在你开始之前

在 `workstation` 上以 `student` 用户身份登录，再运行 **`lab control-reviewstart`**。此脚本将确保受管主机 `serverb` 可在网络上访问。它也将确保控制节点上安装了正确的 Ansible 配置文件和清单。

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

1. 在 `workstation` 上，更改到 **`/home/student/control-review`** 项目目录。
2. 项目目录包含部分完成的 playbook **`playbook.yml`**。使用文本编辑器，在 **`#Fail Fast Message`** 注释下添加使用 `fail` 模块的任务。为任务定义一个合适的名称。只有在远程系统不满足最低要求时才应执行此任务。

   下面列出了远程主机的最低要求：

   - 至少具有 `min_ram_mb` 变量指定的 RAM 内存量。`min_ram_mb` 变量在 **`vars.yml`** 文件中定义，且值为 **`256`**。
   - 运行的是红帽企业 Linux。
3. 在 **`#Install all Packages`** 注释下向 playbook 添加一个任务以安装任何缺失的软件包的最新版本。所需的软件包由 `packages` 变量指定，该变量在 **`vars.yml`** 文件中定义。

   任务名称应为 **`Ensure required packages are present`**。
4. 在 **`#Enable and start services`** 注释下向 playbook 添加一个任务以启动所有服务。在 **`vars.yml`** 文件中定义的 `services` 变量指定的所有服务应启动并启用。务必为该任务提供适当的名称。
5. 在 **`#Block of config tasks`** 注释下向 playbook 添加一个任务块。该块包含两个任务：
   - 确保远程主机上存在 `ssl_cert_dir` 变量指定的目录的任务。此目录存储 Web 服务器的证书。
   - 将 `web_config_files` 变量指定的所有文件复制到远程主机的任务。检查 **`vars.yml`** 文件中的 `web_config_files` 变量的结构。配置此任务以将每个文件复制到远程主机上的正确目标。

海量视频题库 myitpub.com QQ:565462
www.52myit.com

如果在远程服务器上更改了其中任何文件，此任务应触发 **restart web service** 处理程序。

此外，如果上述两个任务中的任何一个失败，将执行调试任务。在这种情况下，该任务显示消息：**One or more of the configuration changes failed, but the web service is still active.**。

务必为所有任务提供适当的名称。

6. 该 playbook 配置远程主机以侦听标准 HTTPS 请求。在 **#Configure the firewall** 注释下向 playbook 添加一个任务以配置 `firewalld`。

   此任务应确保远程主机允许标准 HTTP 和 HTTPS 连接。这些配置更改应立即生效，并在系统重新引导后保持不变。务必为该任务提供适当的名称。

7. 定义 **restart web service** 处理程序。

   触发时，此任务应重新启动由 **vars.yml** 文件中定义的 `web_service` 变量定义的 Web 服务。

8. 从项目目录 **~/control-review**，运行 **playbook.yml** playbook。该 playbook 应正确无误地执行，并触发处理程序任务执行。

9. 验证 Web 服务器现在是否响应 HTTPS 请求，使用自签名自定义证书加密连接。Web 服务器响应应与字符串 **Configured for both HTTP and HTTPS** 匹配。

评估

在 `workstation` 上运行 **lab control-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 control-review grade
```

完成

运行 **lab control-review finish** 命令，进行实验后清理。

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

实施任务控制

任务执行清单

在本实验中，您将安装 Apache Web 服务器并使用 `mod_ssl` 对其进行保护。您将在 playbook 中使用条件、处理程序和任务失败处理来部署环境。

成果

您应该能够在 Ansible Playbook 中定义条件，设置迭代元素的循环，在 playbook 中定义处理程序，以及处理任务错误。

在你开始之前

在 `workstation` 上以 `student` 用户身份登录，再运行 **`lab control-reviewstart`**。此脚本将确保受管主机 `serverb` 可在网络上访问。它也将确保控制节点上安装了正确的 Ansible 配置文件和清单。

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

1. 在 `workstation` 上，更改到 **`/home/student/control-review`** 项目目录。

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

2. 项目目录包含部分完成的 playbook **`playbook.yml`**。使用文本编辑器，在 **`#Fail Fast Message`** 注释下添加使用 `fail` 模块的任务。为任务定义一个合适的名称。只有在远程系统不满足最低要求时才应执行此任务。

   下面列出了远程主机的最低要求：

   - 至少具有 `min_ram_mb` 变量指定的 RAM 内存量。`min_ram_mb` 变量在 **`vars.yml`** 文件中定义，且值为 **256**。
   - 运行的是红帽企业 Linux。

   完成的任务与以下内容一致：

```
tasks:
  #Fail Fast Message
  - name: Show Failed System Requirements Message
    fail:
      msg: "The {{ inventory_hostname }} did not meet minimum reqs."
    when: >
      ansible_memtotal_mb < min_ram_mb or
      ansible_distribution != "RedHat"
```

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

3. 在 **#Install all Packages** 注释下向 playbook 添加一个任务以安装任何缺失的软件包的最新版本。所需的软件包由 packages 变量指定，该变量在 **vars.yml** 文件中定义。

   任务名称应为 **Ensure required packages are present**。

   完成的任务与以下内容一致：

```
#Install all Packages
- name: Ensure required packages are present
  yum:
    name: "{{ packages }}"
    state: latest
```

4. 在 **#Enable and start services** 注释下向 playbook 添加一个任务以启动所有服务。在 **vars.yml** 文件中定义的 services 变量指定的所有服务应启动并启用。务必为该任务提供适当的名称。

   完成的任务与以下内容一致：

```
#Enable and start services
- name: Ensure services are started and enabled
  service:
    name: "{{ item }}"
    state: started
    enabled: yes
  loop: "{{ services }}"
```

5. 在 **#Block of config tasks** 注释下向 playbook 添加一个任务块。该块包含两个任务：

   - 确保远程主机上存在 ssl_cert_dir 变量指定的目录的任务。此目录存储 Web 服务器的证书。

   - 将 web_config_files 变量指定的所有文件复制到远程主机的任务。检查 **vars.yml** 文件中的 web_config_files 变量的结构。配置此任务以将每个文件复制到远程主机上的正确目标。

     如果在远程服务器上更改了其中任何文件，此任务应触发 **restart web service** 处理程序。

   此外，如果上述两个任务中的任何一个失败，将执行调试任务。在这种情况下，该任务显示消息：**One or more of the configuration changes failed, but the web service is still active.**。

   务必为所有任务提供适当的名称。

   完成的任务块与以下内容一致：

```
#Block of config tasks
- name: Setting up the SSL cert directory and config files
  block:
    - name: Create SSL cert directory
      file:
        path: "{{ ssl_cert_dir }}"
        state: directory

    - name: Copy Config Files
```

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

```
        copy:
          src: "{{ item.src }}"
          dest: "{{ item.dest }}"
        loop: "{{ web_config_files }}"
        notify: restart web service

    rescue:
      - name: Configuration Error Message
        debug:
          msg: >
            One or more of the configuration
            changes failed, but the web service
            is still active.
```

6. 该 playbook 配置远程主机以侦听标准 HTTPS 请求。在 **#Configure the firewall** 注释下向 playbook 添加一个任务以配置 **firewalld**。

   此任务应确保远程主机允许标准 HTTP 和 HTTPS 连接。这些配置更改应立即生效，并在系统重新引导后保持不变。务必为该任务提供适当的名称。

   完成的任务与以下内容一致：

```
  #Configure the firewall
  - name: ensure web server ports are open
    firewalld:
      service: "{{ item }}"
      immediate: true
      permanent: true
      state: enabled
    loop:
      - http
      - https
```

7. 定义 **restart web service** 处理程序。

   触发时，此任务应重新启动由 **vars.yml** 文件中定义的 **web_service** 变量定义的 Web 服务。

   Playbook 的末尾将添加一个 **handlers** 部分。

```
handlers:
  - name: restart web service
    service:
      name: "{{ web_service }}"
      state: restarted
```

   完成的 playbook 包含：

```
---
- name: Playbook Control Lab
  hosts: webservers
  vars_files: vars.yml
  tasks:
    #Fail Fast Message
    - name: Show Failed System Requirements Message
      fail:
```

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

```
        msg: "The {{ inventory_hostname }} did not meet minimum reqs."
      when: >
        ansible_memtotal_mb < min_ram_mb or
        ansible_distribution != "RedHat"

    #Install all Packages
    - name: Ensure required packages are present
      yum:
        name: "{{ packages }}"
        state: latest

    #Enable and start services
    - name: Ensure services are started and enabled
      service:
        name: "{{ item }}"
        state: started
        enabled: yes
      loop: "{{ services }}"

    #Block of config tasks
    - name: Setting up the SSL cert directory and config files
      block:
        - name: Create SSL cert directory
          file:
            path: "{{ ssl_cert_dir }}"
            state: directory

        - name: Copy Config Files
          copy:
            src: "{{ item.src }}"
            dest: "{{ item.dest }}"
          loop: "{{ web_config_files }}"
          notify: restart web service

      rescue:
        - name: Configuration Error Message
          debug:
            msg: >
              One or more of the configuration
              changes failed, but the web service
              is still active.

    #Configure the firewall
    - name: ensure web server ports are open
      firewalld:
        service: "{{ item }}"
        immediate: true
        permanent: true
        state: enabled
      loop:
        - http
        - https

  #Add handlers
  handlers:
```

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

```
  - name: restart web service
    service:
      name: "{{ web_service }}"
      state: restarted
```

8. 从项目目录 **~/control-review**，运行 **playbook.yml** playbook。该 playbook 应正确无误地执行，并触发处理程序任务执行。

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

PLAY [Playbook Control Lab] ****************************************************

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

TASK [Show Failed System Requirements Message] *******************************
skipping: [serverb.lab.example.com]

TASK [Ensure required packages are present] **********************************
changed: [serverb.lab.example.com]

TASK [Ensure services are started and enabled] *******************************
changed: [serverb.lab.example.com] => (item=httpd)
ok: [serverb.lab.example.com] => (item=firewalld)

TASK [Create SSL cert directory] ***********************************************
changed: [serverb.lab.example.com]

TASK [Copy Config Files] *******************************************************
changed: [serverb.lab.example.com] => (item={'src': 'server.key', 'dest': '/etc/
httpd/conf.d/ssl'})
changed: [serverb.lab.example.com] => (item={'src': 'server.crt', 'dest': '/etc/
httpd/conf.d/ssl'})
changed: [serverb.lab.example.com] => (item={'src': 'ssl.conf', 'dest': '/etc/
httpd/conf.d'})
changed: [serverb.lab.example.com] => (item={'src': 'index.html', 'dest': '/var/
www/html'})

TASK [ensure web server ports are open] ****************************************
changed: [serverb.lab.example.com] => (item=http)
changed: [serverb.lab.example.com] => (item=https)

RUNNING HANDLER [restart web service] ******************************************
changed: [serverb.lab.example.com]

PLAY RECAP *********************************************************************
serverb.lab.example.com    : ok=7    changed=6    unreachable=0    failed=0
```

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