DO407 综合练习

System	IP Address	Role
workstation.lab.example.com	172.25.250.254	管理节点
servera.lab.example.com	172.25.250.10	Ansible Client
serverb.lab.example.com	172.25.250.11	Ansible Client
serverc.lab.example.com	172.25.250.12	Ansible Client
serverd.lab.example.com	172.25.250.13	Ansible Client

每个节点的 root 用户密码为 redhat，student 用户密码为 student

需要在 workstation 上安装 ansible 软件，workstation 上的 student 和 root 用户的 ssh 公钥已经分发到每个节点

在 workstation 上，所有 playbook 必须在/home/student/playbooks 目录中以 student 用户运行

每个节点的防火墙都没有开启，SELinux 运行在 Permissive 模式

每个节点已预先配置好 yum 源

练习所需的各种文件在 workstation 的 /files/目录中，需要拷贝到各个节点的/files/

0.考前环境准备

```
[root@foundation0 ~]# scp Desktop/files.tar.gz root@workstation:/root/
[root@foundation0 ~]# ssh root@workstation
[root@workstation ~]# yum -y install ansible
vim /etc/sudoers
99 #注释
101 删除注释
这个文件发给所有的 servera-d
server-a 关闭防火墙
tar xf /files/1/opt_ansible_inventory_dynamic.tar.gz -C /
```

```
mkdir playbooks
cd playbooks
vim ansible.cfg
[defaults]
inventory=/opt/ansible/inventory/dynamic
timeout=60
roles_path=roles

[privilege_escalation]
become=True
become_method=sudo
become_user=root
become_ask_pass=False
:x

tar xf /files/1/
```

1.诊断并修复动态 inventory

在 workstation.lab.example.com 上修改 Ansible 配置文件，以调用一个动态的 inventory

/opt/ansible/inventory/dynamic 当加载此动态 inventory 后，确认可以访问其中的 marzipan，bunnies 和 newbunnies 主机中的主机。你必须修复此目录中所有错误。

你无需对此目录中的文件内容进行修改。不需要掌握 Python 知识。你可以重命名和重新排序动态文件和脚本。需要确保现在及将来的所有脚本均能使用此动态 inventory。

须在 ansible 配置文件的[defaults]段添加 timeout = 60

可以使用 playbook 或 ad-hoc 命令来访问此动态 inventory 中定义的 groups

```
cd /opt/ansible/inventory/dynamic/
chmod *.py
```

2.在 workstation 上创建名为/home.student/playbooks/httpd.yum 的 playbook，满足以下要求：

安装 httpd 软件包

使用 get_url 方式将/files/2/httpd.conf.template 文件拷贝到/etc/httpd/conf/httpd.conf

配置 httpd 服务在 4 个节点上启动并开机运行

启动 httpd 服务

在 4 个节点上使用 lineinfile 方式自定义/var/www/html/index.html 文件， 包含“Hello from XXX” XXX 为 servera，serverb，serverc 或 serverd

```
vim httpd.yml
---
- name: install and start httpd
  hosts: nodes
  tasks:
    - name: install httpd
      yum:
        name: httpd
        state: latest

    - name: copy conf
      get_url:
        url: file:///files/2/httpd.conf.template
        dest: /etc/httpd/conf/httpd.conf

    - name: start httpd
      service:
        name: httpd
        state: started
        enabled: true

    - name: edit index.html
      lineinfile:
        path: /var/www/html/index.html
        line: Hello from {{  ansible_hostname }}
        create: yes
```

3.在 workstation 上创建名为/home/student/playbooks/httpd_auth.yml 的 playbook，满足以下要求：

在 4 个受管理节点上运行

以适当权限创建目录/var/www/html/protected 允许被浏览器访问

以 get_url 方式将 /files/3/httpd.htaccess 拷贝到 /var/www/html/protected/.htaccess 并将 /file/3/httpd.htpasswd 拷贝到/var/www/html/protected/.htpasswd

为 /var/www/html/protected/.htaccess 和 /var/www/html/protected/.htpasswd 设置权限，可以被 apache 用户读取

修改/etc/httpd/conf/httpd.conf 使用 replace 模块搜索所有”AllowOverride None” 内容并替换为“AlloOverride Authconfig”

创建文件/var/www/html/protected/index.html，包含一行文字”This node server# is protected” server#表示 4 个节点中的一个

重启 httpd 服务

如果此 playbook 正确运行，在 4 个节点上访问/var/www/html/protevted 均会提示输入用户名/密码（ansi/ansible）

```
vim httpd_auth.yml
---
- name: config secure content
  hosts: nodes
  tasks:
    - name: create dir
      file:
        path: /var/www/html/protected
        state: directory
        owner: apache
        group: apache
        mode: 0700

    - name: copy file
      get_url:
        url: file:///files/3/httpd.htaccess
        dest: /var/www/html/protected/.htaccess
        owner: apache
        group: apache
        mode: 0600

    - name: copy another file
      get_url:
        url: file:///files/3/httpd.htpasswd
        dest: /var/www/html/protected/.htpasswd
        owner: apache
        group: apache
        mode: 0600

    - name: update conf
      replace:
        path: /etc/httpd/conf/httpd.conf
        regexp: AllowOverride None
        replace: AllowOverride AuthConfig

    - name: update index
      lineinfile:
        path: /var/www/html/protected/index.html
        line: This node {{ ansible_hostname }} is protected
        create: yes

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

4.使用自定义的 group

在 ansible 控制节点上定义 production 和 backup group

production group 包含 servera 和 serverb，backup group 包含 serverc 和 serverd

在 workstation.lab.example.com 上创建一个名为/home/student/playbooks/webproduction.yml 的 playbook，要求如下：

对 production group 的节点，创建一个名为/var/tmp/production.tar.gz 的压缩包，包含/var/www/html 的内容

对 backup group 的节点，创建一个名为/var/tmp/backup.tar.gz 的压缩包，包含/var/log/httpd 的内容

vim webproduction.yml

```
---
- name: create tar files
  hosts: production
  tasks:
    - name: tar html
      shell: "tar zcf /var/tmp/production.tar.gz /var/www/html"

- name: create tar files
  hosts: backup
  tasks:
    - name: tar log html
      shell: "tar zcf /var/tmp/backup.tar.gz /var/www/html"
```

5.创建运行 ansible 命令的脚本：

在 workstation.lab.example.com 上创建/home/student/playbooks/adhoc.sh 脚本，使用 ad-hoc Ansible 命令完成以下事务：

将/files/5/runme.sh 从 ansible 主机拷贝到 4 个受控节点的/usr/bin/runme.sh

设置/usr/bin/runme.sh 的权限为 0755

设置/usr/bin/runme.sh 的 ower 和 group 为 student

以 student 用户身份在四个受管节点上运行/usr/bin/runme.sh

如有需要，赐脚本可以进行 privilege escalation

adhoc.sh 的权限应该为 0755

```
vim adhoc.sh
#!/bin/bash
ansible nodes -m copy -a "src=/files/5/runme.sh dest=/usr/bin/runme.sh mode=
0755 owner=student group=student" -u student -b -become-user=root --become-method=sudoansible nodes -m
shell -a "/usr/bin/runme.sh"
chmod 755 adhoc.sh
```

6.有选择的运行 tasks：

在 workstation.lab.example.com 上创建名为/home/student/playbooks/ping.yml 的 playbook 完成以下任务：

/files/5/ans.txt 文件在 workstation.lab.example.com 上

运行/home/student/playbook/ping.yml --tag “nodegroup1” 将拷贝 ans.txt 到 servera 和 serverb 的/var/www/html/tag1.html

运行/home/student/playbook/ping.yml --tag “nodegroup2” 将拷贝 ans.txt 到 serverc 和 serverd 的/var/www/html/tag2.html

```
vim ping.yml
---
- name: copy txt to selected nodes
  hosts: server[a:b]
  tasks:
    - name: tag1
      copy:
```

```
        src: /files/6/ans.txt
        dest: /var/www/html/tag1.html
      tags: nodegroup1

- name: copy txt to selected nodes
  hosts: server[c:d]
  tasks:
    - name: tag2
      copy:
        src: /files/6/ans.txt
        dest: /var/www/html/tag2.html
      tags: nodegroup2

ansible-playbook --tags nodegroup1 ping.yml
ansible-playbook --tags nodegroup2 ping.yml
```

7.处理无法访问的报错:

在 workstation.la.example.com 上创建名为/home/student/playbooks/inaccess.yml 的 playbook 完成以下任务:

在所有受管主机上运行

使用 block group 将/files/7/inaccess.html 文件拷贝到/var/www/html/inaccess.html

如果拷贝失败，使用 rescue group 将文字”I did not have access to the url” 写入 /var/www/html/inaccess.html

```
vim inaccess.yml
---
- name: block and rescue
  hosts: nodes
  tasks:
    - block:
        - name: normal
          get_url:
            url: file:///files/7/inaccess.html
            dest: /var/www/html/inaccess.html
            force: yes
      rescue:
        - name: rescue
          lineinfile:
            path: /var/www/html/inaccess.html
            line: I did not have access to the url
            create: yes
```

8.创建 playbook 有选择的更新文件:

在 workstation.lab.example.com 上创建名为/home/student/playbooks/saveabort.yml 完成以下任务:

在所有受管主机上运行

创建/etc/ansible_abort.txt 的文件，包含”my node is also server#”，server#代表此 playbook 运行在 servera-d

如果文件/etc/ansible_abort.txt 已存在，则不要做任何修改

```
vim saveabort.yml
---
- name: saveabort
  hosts: nodes
  tasks:
    - name: create file
```

```
      lineinfile:
        path: /etc/ansible_abort.txt
        line: my node is also {{ ansible_hostname }}
        create: yes
```

注意：这样做有问题，如果 ansible_abort.txt 存在，还是会修改内容

9.有选择的控制一个服务

在 workstation.lab.example.com 上创建名为/home/student/playbooks/noshutdown.yml 的 playbook 完成以下任务：

在 4 个受管节点上运行

启动 httpd 服务

确保 httpd 服务已启动而且开机自启动

在 workstation.lab.example.com 上创建名为/home/student/playbooks/shudown.yml 的 playbook 完成以下任务：

在 servera 和 serverc 上停止 httpd 服务

```
vim shutdown.yml
---
- name: shutdown httpd
  hosts: server[a:b]
  tasks:
    - name: shutdown
      service:
        name: httpd
        state: stopped

vim noshutdown.yml
---
- name: start and enable
  hosts: nodes
  tasks:
    - name: start and enable
      service:
        name: httpd
        state: started
        enabled: true
```

10.使用 Ansible Galaxy 创建一个 role：

在 workstation.lab.example.com 上创建名为 /home/student/plaubooks/galaxy_install.yml 的 playbook 完成以下任务：

使用 ansible-galaxy 安装来自/files/10/examfun.tar.gz 的 role

将 role 安装到/home/student/playbooks/roles/examfun

在控制节点中的 Ansible 配置文件中将/home/student/playbook/roles 定义为 role 目录

在控制节点的 Ansible 配置文件中，在[defualts]段中设置 timeout = 60

```
mkdir roles
vim galaxy_install.yml
---
- name: examfun
  src: file:///files/10/examfun.tar.gz

ansible-galaxy install -r galaxy_install.yml
```

11. 创建用户账户

在 workstation.lab.example.com 上创建名为/home/student/playbooks/users.yml 的 playbook 完成以下任务：

在所有受管节点创建用户组 staff,guests 和 webclients

使用 examfun 角色定义的所有 usermames 变量

在所有受管节点中，创建 staff list 中的用户账户，并隶属于 staff 和 webclients 用户组

在所有受管节点中，创建 guests list 中的用户账户，并隶属于 guests 和 webclients 用户组

在所有受管节点中，删除 revoked list 中的账户

```
vim users.yml
---
- name: create and remove users
  hosts: nodes
  pre_tasks:
    - name: create groups
      group:
        name: "{{ item }}"
        state: present
      with_items:
        - staff
        - guests
        - webclients
  roles:
    - examfun

  post_tasks:
    - name: create user from staff group
      user:
        name: "{{ item }}"
        groups: staff,webclients
      with_items: "{{ staff }}"
    - name: create user from guests group
      user:
        name: "{{ item }}"
        groups: guests,webclients
      with_items: "{{ guests }}"
    - name: remove user from revoke group
      user:
        name: "{{ item }}"
        state: absent
        remove: yes
      with_items: "{{ revoked }}"
```

12.创建一个 playbook 显示主机的信息：

在 workstation.lab.example.com 上创建名为/home/student/playbook/ansible.yml 的 playbook 显示主机的信息：

在所有受管主机中运行此 playbook

此 playbook 在/var/www/html/ansible_details.html 文件中创建一行信息，包含以下内容：

受管主机的短主机名

受管主机的物理内核数量

受管主机默认网卡的 ipv4 地址

受管主机默认网卡的 MAC 地址

内容如下例

node7 4 123.123.123.123 DE:AD:EF:DE:AD:BE:EF

```
vim ansible.yml
```

```
---
- name: gather facts
  hosts: nodes
  tasks:
    - name: update ansible_details.html
      lineinfile:
        path: /var/www/html/ansible_details.html
        line: "{{ ansible_hostname }} {{ ansible_processor_count }} {{ ansible_eth0.ipv4.address }}
{{ ansible_eth0.macaddress }}"
        create: yes
```

13.创建一个 playbook 处理敏感信息：

在 workstation.lab.example.com 上创建名为/home/student/playbooks/vault.yml 的 playbook 完成以下任务：

使用 vault 文件 crypt.yml （描述如下）来访问 password 变量

使用 password 值解压缩/files/13/vault.zip 到 serverb 和 serverd 的/var/www/html/vault 目录

使用 unzip -o -P password 参数进行解压缩

运行 playbook 时需要验证/home/student/palybooks/.vaultpwd 密码文件。

vault 信息：

加密的 vault YML 文件名为 crypt.yml 放置在/home/student/playbooks

此 vault 被密码 ansiansi 加密。此密码保存在/home/student/playbooks/.vaultpwd 文件中

此 vault 应定义一个 ansible 变量，名为 password，值为明文：drone

```
vim .vaultpwd
ansiansi

ansible-vault create crypt.yml
输入 ansiansi 密码
password: drone

vim vault.yml
---
- name: unzip encryped vault.zip
  hosts: serverb serverd
  vars_files:
    - crypt.yml
  tasks:
    - name: create vault directory
      file:
        path: /var/www/html/vault
        state: directory
        owner: apache
        group: apache
        mode: 0755
    - name: unzip vault.zip file
      command: "unzip -o -P {{ password }} /files/13/vault.zip -d /var/www/html/vault"

ansible-playbook --vault-password-file=.vaultpwd vault.yml
```

14.进行报错的处理

在 workstation.lab.example.com 上创建名为/home/student/playbooks/failsamba.yml 的 playbook：

在 servera 上尝试安装并启动 samba 服务

如果软件包安装或服务启动失败，playbook 获取任务信息并显示

即使一个任务执行失败，其他所有任务也必须被执行

如果安装任务失败，为了便于排错，需要显示 debug 信息”Installation failed”

如果配置任务失败，为了便于排错，需要显示服务启动失败的 debug 信息”Starting failed”

如果任何任务失败，需要显示信息 ”playbook was not successful”

```
vim failsamba.yml
---
- name: install and start samba
  hosts: servera
  tasks:
    - name: install samba
      yum:
        name: samba
        state: latest
      register: install_result
      ignore_errors: true

    - name: debug install
      debug:
        msg: Installation failed
      when: install_result.rc != 0

    - name: start samba
      service:
        name: smb
        state: started
      register: start_result
      ignore_errors: true

    - name: debug start
      debug:
        msg: Starting faied
      when: start_result | failed

    - name: debug all err
      debug:
        msg: playbook was not successful
      when: start_result | failed or install_result.rc != 0
```