day07

任务块

- 可以通过block关键字，将多个任务组合到一起
- 可以将整个block任务组，一起控制是否要执行

```
# ansible的playbook默认有一个名为Gathering facts的任务，用于收集远程主机的facts变量。如果不需要，可以把它关闭
[root@control ansible]# vim ansible.cfg
```

```
[defaults]
inventory = hosts
gathering = explicit


# 如果test组中的主机系统发行版是RedHat，则安装并启动
httpd
# 由于该playbook需要用到facts变量，所以需要明确收集
facts变量
[root@control ansible]# vim block1.yml
---
- name: block tasks
  hosts: test
  tasks:
    - name: gathering facts
      setup:

    - name: define a group of tasks
      block:
        - name: install httpd
          yum:
            name: httpd
            state: present

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

      when: ansible_distribution=="RedHat"
```

```
[root@control ansible]# ansible-playbook block1.yml
```

rescue和always

- block和rescue、always联合使用：
  - block中的任务都成功，rescue中的任务不执行
  - block中的任务出现失败（failed），rescue中的任务执行
  - block中的任务不管怎么样，always中的任务总是执行

```
[root@control ansible]# vim block2.yml
---
- name: block test
  hosts: test
  tasks:
    - name: block / rescue / always test1
      block:
        - name: touch a file
          file:
            path: /tmp/test1.txt
            state: touch
      rescue:
        - name: touch file test2.txt
          file:
            path: /tmp/test2.txt
            state: touch
```

```
      always:
        - name: touch file test3.txt
          file:
            path: /tmp/test3.txt
            state: touch

# 执行playbook node1上将会出现/tmp/test1.txt
和/tmp/test3.txt
[root@control ansible]# ansible-playbook block2.yml
[root@node1 ~]# ls /tmp/test*.txt
/tmp/test1.txt  /tmp/test3.txt

# 修改上面的playbook，使block中的任务出错
[root@node1 ~]# rm -f /tmp/test*.txt
[root@control ansible]# vim block2.yml
---
- name: block test
  hosts: test
  tasks:
    - name: block / rescue / always test1
      block:
        - name: touch a file
          file:
            path: /tmp/abcd/test11.txt
            state: touch
      rescue:
        - name: touch file test22.txt
          file:
            path: /tmp/test22.txt
            state: touch
      always:
```

```
        - name: touch file test33.txt
          file:
            path: /tmp/test33.txt
            state: touch
# 因为node1上没有/tmp/abcd目录，所以block中的任务失败。但是playbook不再崩溃，而是执行rescue中的任务。always中的任务总是执行
[root@control ansible]# ansible-playbook block2.yml
[root@node1 ~]# ls /tmp/test*.txt
/tmp/test2.txt  /tmp/test3.txt
```

loop循环

- 相当于shell中for循环
- ansible中循环用到的变量名是固定的，叫item

```
# 在test组中的主机上创建5个目录/tmp/{aaa,bbb,ccc,ddd,eee}
[root@control ansible]# vim loop1.yml
---
- name: use loop
  hosts: test
  tasks:
    - name: create directory
      file:
        path: /tmp/{{item}}
        state: directory
      loop: [aaa,bbb,ccc,ddd,eee]
```

```
# 上面写法，也可以改为：
---
- name: use loop
  hosts: test
  tasks:
    - name: create directory
      file:
        path: /tmp/{{item}}
        state: directory
      loop:
        - aaa
        - bbb
        - ccc
        - ddd
        - eee

[root@control ansible]# ansible-playbook loop1.yml


# 使用复杂变量。创建zhangsan用户，密码是123；创建lisi用户，密码是456
# item是固定的，用于表示循环中的变量
# 循环时，loop中每个-后面的内容作为一个整体赋值给item。
# loop中{}中的内容是自己定义的，写法为key:val
# 取值时使用句点表示。如下例中取出用户名就是{{item.uname}}
[root@control ansible]# vim loop_user.yml
---
- name: create users
  hosts: test
```

```
  tasks:
    - name: create multiple users
      user:
        name: "{{item.uname}}"
        password: "
{{item.upass|password_hash('sha512')}}"
      loop:
        - {"uname": "zhangsan", "upass": "123"}
        - {"uname": "lisi", "upass": "456"}
[root@control ansible]# ansible-playbook  loop_user.yml
```

role角色

- 为了实现playbook重用，可以使用role角色
- 角色role相当于把任务打散，放到不同的目录中
- 再把一些固定的值，如用户名、软件包、服务等，用变量来表示
- role角色定义好之后，可以在其他playbook中直接调用

```
# 使用常规playbook，修改/etc/motd的内容
# 1. 修改默认配置
[root@control ansible]# vim ansible.cfg
[defaults]
inventory = hosts
# gathering = explicit    # 注释该行，使得playbook默认收集facts

# 2. 创建motd模板文件
```

```
[root@control ansible]# vim motd.j2
Hostname: {{ansible_hostname}}     # facts变量，主机名
Date: {{ansible_date_time.date}}   #  facts变量，日期
Contact to: {{admin}}              # 自定义变量

# 3. 编写playbook
[root@control ansible]# vim motd.yml
---
- name: modifty /etc/motd
  hosts: test
  vars:
    admin: root@tedu.cn            # 自定义名为admin的
变量
  tasks:
    - name: modify motd
      template:
        src: motd.j2
        dest: /etc/motd

[root@control ansible]# ansible-playbook motd.yml
[root@node1 ~]# cat /etc/motd
Hostname: node1
Date: 2021-11-01
Contact to: root@tedu.cn


# 创建角色
# 1. 声明角色存放的位置
[root@control ansible]# vim ansible.cfg
[defaults]
inventory = hosts
```

```
# gathering = explicit
roles_path = roles    # 定义角色存在当前目录的roles子目录中

# 2. 创建角色目录
[root@control ansible]# mkdir roles

# 3. 创建名为motd的角色
[root@control ansible]# ansible-galaxy init roles/motd
[root@control ansible]# ls roles/
motd     # 生成了motd角色目录
[root@control ansible]# yum install -y tree
[root@control ansible]# tree roles/motd/
roles/motd/
├── defaults          # 定义变量的目录，一般不用，因为优先级太低
│   └── main.yml
├── files             # 保存上传的文件（如copy模块用到的文件）
├── handlers          # handlers任务写到这个目录的main.yml中
│   └── main.yml
├── meta              # 保存说明数据，如角色作者、版本等
│   └── main.yml
├── README.md         # 保存角色如何使用之类的说明
├── tasks             # 保存任务
│   └── main.yml
├── templates         # 保存template模块上传的模板文件
├── tests             # 保存测试用的playbook。可选
```

```
│   ├── inventory
│   └── test.yml
└── vars                # 定义变量的位置，推荐使用的位置
    └── main.yml

# 4. 将不同的内容分别写到对应目录的main.yml中
# 4.1 创建motd.j2模板文件
[root@control ansible]# vim roles/motd/templates/motd.j2
Hostname: {{ansible_hostname}}
Date: {{ansible_date_time.date}}
Contact to: {{admin}}

# 4.2 创建变量
[root@control ansible]# vim roles/motd/vars/main.yml  # 追加一行
admin: zzg@tedu.cn

# 4.3 创建任务
[root@control ansible]# vim roles/motd/tasks/main.yml
# 追加
- name: modify motd
  template:
    src: motd.j2      # 这里的文件，自动到templates目录下查找
    dest: /etc/motd

# 5. 创建playbook，调用motd角色
[root@control ansible]# vim role_motd.yml
---
```

```
- name: modify motd with role
  hosts: test
  roles:
    - motd

# 6. 执行playbook
[root@control ansible]# ansible-playbook role_motd.yml
```

- ansible的公共角色仓库：https://galaxy.ansible.com/

```
# 在公共仓库中搜索与httpd相关的角色
[root@zzgrhel8 ~]# ansible-galaxy search httpd
# 如果找到相应的角色，如名字为myhttpd，可以下载它到roles目录
[root@zzgrhel8 ~]# ansible-galaxy install myhttpd -p roles/
```

role练习

1. 创建名为pkgs的角色。用于装包。包名使用变量pkg代表
2. 创建inst_http.yml，调用pkgs角色，安装httpd
3. 创建inst_php.yml，调用pkgs角色，安装php

```
# 1. 创建名为pkgs的角色。
# 1.1 创建角色目录
[root@control ansible]# ansible-galaxy init roles/pkgs
# 1.2 创建装包的任务，包名使用变量pkg代表
```

```
[root@control ansible]# vim roles/pkgs/tasks/main.yml
---
# tasks file for roles/pkgs
- name: install rpm pkg
  yum:
    name: "{{pkg}}"
    state: present
# 1.3 定义变量
[root@control ansible]# vim
roles/pkgs/defaults/main.yml
---
# defaults file for roles/pkgs
pkg: httpd

# 2. 创建inst_http.yml，调用pkgs角色，安装httpd
[root@control ansible]# vim inst_httpd.yml
---
- name: install httpd pkg
  hosts: test
  roles:
    - pkgs
[root@control ansible]# ansible-playbook inst_httpd.yml

# 3. 创建inst_php.yml，调用pkgs角色，安装php
[root@control ansible]# vim inst_php.yml
---
- name: install php pkg
  hosts: test
  vars:
    pkg: php
  roles:
```

```
    - pkgs
[root@control ansible]# ansible-playbook inst_php.yml
```

ansible加解密文件

- ansible加解密文件使用ansible-vault命令

```
[root@control ansible]# echo "Hi ni hao" > hello.txt
[root@control ansible]# cat hello.txt
Hi ni hao

# 加密文件
[root@control ansible]# ansible-vault encrypt hello.txt
New Vault password: 123456
Confirm New Vault password: 123456
Encryption successful
[root@control ansible]# cat hello.txt
$ANSIBLE_VAULT;1.1;AES256
37373366353566346235613731396566646533393361386131313633
23065636333363339633734653
61643234613561303038636339643933393637386530363 10a66656
43138323162633930616163 30
32373133323162353864316435366439386266616661373936363563
33736343563653266373361 65
63366362303665646 50a383239636230623633356565623461326433
13936346566663063306635 33
6235

# 解密
```

```
[root@control ansible]# ansible-vault decrypt hello.txt
Vault password: 123456
Decryption successful
[root@control ansible]# cat hello.txt
Hi ni hao


# 加密后更改密码
[root@control ansible]# ansible-vault encrypt hello.txt
New Vault password: 123456
Confirm New Vault password: 123456
Encryption successful

[root@control ansible]# ansible-vault rekey hello.txt
# 改密码
Vault password: 123456    # 旧密码
New Vault password: abcd  # 新密码
Confirm New Vault password: abcd
Rekey successful

# 不解密文件，查看内容
[root@control ansible]# ansible-vault view hello.txt
Vault password: abcd
Hi ni hao


# 使用密码文件进行加解密
# 1. 将密码写入文件
[root@control ansible]# echo 'tedu.cn' > pass.txt
# 2. 创建明文文件
[root@control ansible]# echo 'hello world' > data.txt
```

```
# 3. 使用pass.txt中的内容作为密码加密文件
[root@control ansible]# ansible-vault encrypt --vault-id=pass.txt data.txt
Encryption successful
[root@control ansible]# cat data.txt    # 文件已加密
# 4. 使用pass.txt中的内容作为密码解密文件
[root@control ansible]# ansible-vault decrypt --vault-id=pass.txt data.txt
Decryption successful
[root@control ansible]# cat data.txt
hello world
```

sudo命令

- 一般用于普通用户执行需要root权限的命令
- 在node1上配置zhangsan拥有sudo权限

```
# 如果没有zhangsan，手工创建
[root@node1 ~]# visudo   # 将会打开vi，在尾部追加以下一行
zhangsan        ALL=(ALL)       ALL
# 中间的ALL=(ALL)在集中认证的域环境中才有效，单机忽略即可
# zhangsan是用户名，最后的ALL表示zhangsan可以以管理员的身份执行所有命令

# 切换成zhangsan用户，执行命令
[root@node1 ~]# su - zhangsan
```

```
[zhangsan@node1 ~]$ useradd wangwu   # 失败，因为还是张三身份
[zhangsan@node1 ~]$ sudo useradd wangwu  # 以管理员身份执行
... ...
[sudo] password for zhangsan: # 输入zhangsan的密码，不是root


# 配置lisi不输入密码可以直接运行sudo
[root@node1 ~]# visudo    # 在最后追加一行
lisi    ALL=(ALL)       NOPASSWD: ALL

# 切换成lisi运行
[root@node1 ~]# su - lisi
[lisi@node1 ~]$ ls /root/   # 没权限
ls: cannot open directory '/root/': Permission denied
[lisi@node1 ~]$ sudo ls /root/    # 成功运行，无需输入密码
a3.txt  anaconda-ks.cfg
```

特殊的主机清单变量

- 如果远程主机没有使用免密登陆，如果远程主机ssh不是标准的22端口，可以设置特殊的主机清单变量
- ansible_ssh_user：指定登陆远程主机的用户名
- ansible_ssh_pass：指定登陆远程主机的密码
- ansible_ssh_port：指定登陆远程主机的端口号

```
# 删除远程主机的/root/.ssh/authorized_keys，以便恢复通过密码登陆
[root@control ansible]# ansible all -m file -a "path=/root/.ssh/authorized_keys state=absent"

# 创建新的工作目录
[root@control ~]# mkdir myansible
[root@control ~]# cd myansible
[root@control myansible]# vim ansible.cfg
[defaults]
inventory = hosts
[root@control myansible]# vim hosts
[group1]
node1
node2
node3
[root@control myansible]# ansible all -m ping  # 报错，因为无法免密执行

# 修改node1 ssh服务的端口为220
[root@node1 ~]# systemctl stop firewalld
[root@node1 ~]# vim +17 /etc/ssh/sshd_config
Port 220
[root@node1 ~]# systemctl restart sshd
# 退出再登陆时，需要指定端口号
[root@zzgrhel8 ~]# ssh -p220 192.168.4.11



# 配置ssh通过用户名、密码管理远程主机，通过220端口连接node1
```

```
[root@control myansible]# vim hosts
[group1]
node1 ansible_ssh_user=root ansible_ssh_pass=a
ansible_ssh_port=220
node2 ansible_ssh_user=root ansible_ssh_pass=a
node3 ansible_ssh_user=root ansible_ssh_pass=a

[root@control myansible]# ansible all -m ping
```