**Morning：**

1. playbook基础知识

**-ansible七种武器**

ansible 命令——执行临时性工作

ansible-doc——模块文档说明

ansible-console——交互式工具

ansible-galaxy——从github下载Roles的工具

ansible-playbook——读取预先编写好的 playbook 文件实现批量管理

ansible-vault——配置文件加密

ansible-pull——由被托管机主动请求执行命令，一般用于大数量机器配置

**-json、yaml、jinja2**

参考： 阶段4 -> Day01 -> Page01

**-什么是playbook?**

1. YAML语言编写，开始行是“---”。
2. playbook构成：

hosts: 定义将要执行 playbook 的远程主机组

Variable: 定义 playbook 运行时需要使用的变量

Tasks: 定义将要在远程主机上执行的任务列表(并发执行)

Handler: 定义 task 执行完成以后需要调用的任务

3）执行结果

绿色代表执行成功；

黄色代表文件发生变更；

红色代表执行失败

**-playbook案例**

**案例1：**每台机远程连接测试2次

[root@ansible ~]# vim play1.yml

---

- hosts: all #主机名

remote\_user: root #远程登陆账户名

tasks: #任务列表

- ping: #模块: 参数

- ping:

[root@ansible ~]# ansible-playbook play1.yml -f 5

# -f并发进程数量，默认是5

**案例2：**给所有主机创建用户gjq，密码为123，首次登陆修改密码

[root@ansible ~]# vim play2.yml

---

- hosts: db

remote\_user: root

tasks:

- name: add user gjq #tasks下的name为注释用

user: name=gjq shell=/bin/bash

- shell: echo 123 | passwd --stdin gjq

- shell: chage -d 0 gjq

[root@ansible ~]# ansible-playbook play2.yml

[root@ansible ~]# ansible db -m shell -a 'id gjq'

**案例3：**安装apache、监听8080

修改ServerName配置，执行apachectl -t不报错

修改主页为hello world

启动服务并设开机自启

[root@ansible ~]# cat play3.yml

---

- hosts: web2

remote\_user: root

tasks:

- name: install apache

yum: name="httpd" state="installed"

- lineinfile:

path: /etc/httpd/conf/httpd.conf

regexp: ^Listen

line: Listen 8080

- replace:

path: /etc/httpd/conf/httpd.conf

regexp: ^#(ServerName).\*

replace: \1 www.gjq.com

- shell: apachectl -t

- shell: echo "Hi gjq !!" > /var/www/html/index.html

- service: name="httpd" state="restarted" enabled="yes"

**案例4：**批量修改配置文件(含个性信息)

[root@ansible ~]# cat play4.yml

---

- hosts: all

remote\_user: root

tasks:

- template: src=/root/test.html dest=/root/

[root@ansible ~]# cat /root/test.html

This is {{ ansible\_hostname }}

**Afternoon：**

1. playbook语法进阶

**-变量、过滤器**

参考文章：https://www.cnblogs.com/f-ck-need-u/p/7571974.html

1）定义内部变量、过滤器

[root@ansible ~]# cat var.yml

---

- hosts: web1

remote\_user: root

vars: #定义变量

username: gjq2

tasks:

- user:

name： “{{username}}”

password: "{{'123'|password\_hash('sha512')}}"

#变量过滤器的使用

2）使用外部变量

方法1：使用-e传json格式的外部变量 (优先于内部定义变量)

[root@ansible ~]# ansible-playbook var.yml \

> -e '{"username": "gjq10"}'

方法2：使用-e指定json格式的文本文件

[root@ansible ~]# cat var.json

{

"username": "gjq11"

}

[root@ansible ~]# ansible-playbook -e '@var.json' var.yml

1. 注册变量

[root@ansible ~]# cat var.yml

---

- hosts: web

remote\_user: root

tasks:

- shell: echo "I love you"

register: var\_lov

- debug: var=var\_lov

**-error错误处理**

ansible-playbook默认判断$?，如果不为0则停止执行

[root@ansible ~]# vim error.yml

---

- hosts: cache

remote\_user: root

tasks:

- shell: useradd z3 || true #方法1：或命题(不显示报错)

- shell: adduser li4

ignore\_errors: True #方法2：忽略错误(显示报错)

**-handlers触发器与tags标签**

1）handlers触发器

1.与tasks平级，书写时必须写name，其他与tasks格式类似。

2.触发顺序：

-先执行所有tasks

-再调用触发器，相同触发器只会执行1次。

2）tags标签

给指定的任务定义一个调用标识。

调用方式：

-t TAGS, --tags=TAGS #执行tags

--skip-tags=SKIP\_TAGS #跳过tags

--start-at-task=START\_AT #从tags开始执行

**-案例：handlers与tags应用**

[root@ansible ~]# cat handlers.yml

---

- hosts: cache

remote\_user: root

vars:

vport: 8080

tasks:

- yum: name="httpd" state="installed"

- shell: echo 'hello cache!' > /var/www/html/index.html

- lineinfile:

path: /etc/httpd/conf/httpd.conf

regexp: '^Listen'

line: 'Listen {{vport}}'

notify: #调用handlers标签

- reload apache

tags: chport #将上面lineinfile模块定义标签

handlers:

- name: reload apache

service:

name: httpd

state: restarted

enabled: yes

[root@ansible ~]# ansible-playbook handlers.yml --tags chport

#只执行chport标签对应的模块，注意对应的handler也会执行

**-register返回状态与when判断**

#register用于返回模块执行结果给某变量

register: result

#when可以判断setup模块返回的用户信息，如：

when: ansible\_os\_family == "Debian"

#when: result默认判断$? 也可以判断result的标准输出值，如：

when: result

when: result.stdout|float > 0.5

案例：系统负载超过0.7时，停止httpd服务

[root@ansible ~]# cat load.yml

---

- hosts: web1

remote\_user: root

tasks:

- shell: uptime | awk '{printf("%.2f\n",$(NF-2))}'

register: result

- service:

name: httpd

state: stopped

when: result.stdout | float > 0.5

[root@web1 ~] awk 'BEGIN{while(1){}}' &

[root@web1 ~] watch -n 3 uptime

[root@ansible ~]# ansible-playbook load.yml

**-with\_items循环**

每次先从with\_items中取值，再执行模块

用法1：[root@ansible ~]# cat item.yml

---

- hosts: cache

remote\_user: root

tasks:

- user: name={{item}}

with\_items: ["nb", "dd", "plj", "lx"]

用法2：[root@ansible ~]# cat item.yml

---

- hosts: cache

remote\_user: root

tasks:

- user:

name: “{{item.name}}“

group: “{{item.group}}”

password: “{{item.pwd|password\_hash('sha512')}}“

with\_items:

- {name: 'nb', group: 'root', pwd: "nb"}

- {name: 'dd', group: 'root', pwd: "dd"}

- {name: 'lx', group: 'gjq', pwd: "lx"}

**-with\_nested嵌套循环**

#暂未发掘应用场景

[root@ansible ~]# cat test.yml

---

- hosts: web1

remote\_user: root

vars:

un: [a, b, c]

id: [1, 2, 3]

tasks:

- name: add users

shell: echo {{item}}

with\_nested:

- "{{un}}"

- "{{id}}"

**-include and roles**

include用于引入整个项目的.yml文件

roles相当于加强版的include，可以引入一组文件夹

**-debug**

1. 排错调试方法：

-检测语法

ansible-playbook --syntax-check playbook.yaml

-测试运行

ansible-playbook -C playbook.yaml

-显示影响到主机 --list-hosts

-显示工作的 task --list-tasks

-显示将要运行的 tag --list-tags

2）debug 模块可以在运行时输出更为详细的信息，debug 使用样例:

---

- hosts: 192.168.1.16

remote\_user: root

tasks:

- shell: uptime |awk '{printf("%f\n",$(NF-2))}'

register: result

- shell: touch /tmp/isreboot

when: result.stdout|float > 0.5

- name: Show debug info

debug: var=result

debug: 'msg="主机名: {{ansible\_hostname}}"'

#var参数只能输出变量；

#msg可以输出自定义信息，但是引用变量需要{{ }}