

03.Ansible Playbook

03.Ansible Playbook

- 1.Ansible Playbook概述
- 3.Ansible Playbook书写格式
- 4.Ansible Playbook练习实验

1.Ansible Playbook概述

1.什么是playbook, playbook翻译过来就是“剧本”, 那playbook组成如下

playbook: 定义一个文本文件,以yaml为后缀结尾(翻译 我有一个剧本)
yaml格式

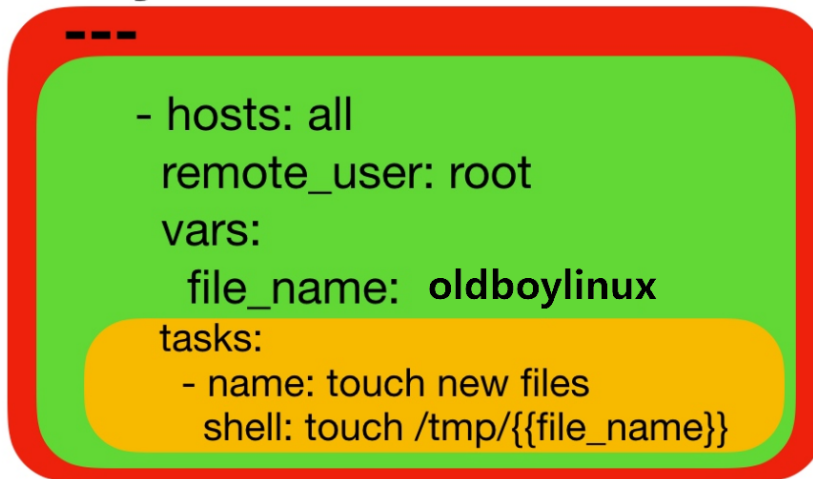
play: 定义的是主机的角色 (翻译 找哪个大腕明星)

task: 定义的是具体执行的任务 (翻译 大腕每一集拍什么)

总结: playbook是由一个或多个play组成, 一个play可以包含多个task任务。

可以理解为: 使用不同的模块来共同完成一件事情。

Playbook组成



playbook

play

task

2. Ansible playbook与AD-Hoc的关系

- 1) *playbook*是对AD-Hoc的一种编排方式。
- 2) *playbook*可以持久运行(重复), 而Ad-Hoc只能临时运行。
- 3) *playbook*适合复杂的任务, 而Ad-Hoc适合做快速简单的任务(检查, 查询, 巡检)。
- 4) *playbook*能控制任务执行的先后顺序。

- ad-hoc 用于检查, 测试, 临时获取数据
- *playbook*剧本适用于, 重复性操作 (部署环境、服务, 初始化操作 (优化))

3. Ansible Playbook书写格式

*playbook*是由yaml语法书写, 结构清晰, 可读性强, 所以必须掌握yaml基础语法

语法	描述
缩进	YAML使用固定的缩进风格表示层级结构,每个缩进由 两个空格 组成, 不能使用tabs
冒号	以冒号结尾的除外, 其他所有 冒号后面 所有必须有 空格 。
短横线	表示列表项, 使用一个短横杠加一个空格。多个项使用同样的缩进级别作为同一列表。

1. 下面我们一起来编写一个playbook文件, *playbook起步*

host: 对哪些主机进行操作

remote_user: 我要使用什么用户执行

tasks: 具体执行什么任务

```

---
- hosts: all
  tasks:
    - name: yum安装软件
      yum: xxxxxxxxxx
    - name: 服务启动
      systemd: xxxxxx

#人生中第1个剧本 查询所有主机的主机名
# ansible ad-hoc
ansible all -m command -a 'hostname' -i hosts

# ansible playbook

[root@m01 /server/playbook]# cat 01_hostname.yml
---
- hosts: all
  tasks:

```

```
- name: show hostname
  command: hostname
[root@m01 /server/playbook]# ansible-playbook
01_hostname.yml -i hosts

PLAY [all]
*****

*****

TASK [Gathering Facts]
*****

ok: [172.16.1.51]
ok: [172.16.1.5]
ok: [172.16.1.6]
ok: [172.16.1.41]
ok: [172.16.1.31]
ok: [172.16.1.7]
ok: [172.16.1.8]
ok: [172.16.1.9]
ok: [172.16.1.10]

TASK [show hostname]
*****

changed: [172.16.1.51]
changed: [172.16.1.41]
changed: [172.16.1.6]
changed: [172.16.1.31]
changed: [172.16.1.5]
changed: [172.16.1.8]
changed: [172.16.1.7]
changed: [172.16.1.10]
changed: [172.16.1.9]

PLAY RECAP
*****
*****
*****

172.16.1.10 : ok=2    changed=1
unreachable=0    failed=0    skipped=0    rescued=0
ignored=0
```

```
172.16.1.31      : ok=2    changed=1
unreachable=0    failed=0    skipped=0    rescued=0
ignored=0
172.16.1.41      : ok=2    changed=1
unreachable=0    failed=0    skipped=0    rescued=0
ignored=0
172.16.1.5       : ok=2    changed=1
unreachable=0    failed=0    skipped=0    rescued=0
ignored=0
172.16.1.51      : ok=2    changed=1
unreachable=0    failed=0    skipped=0    rescued=0
ignored=0
172.16.1.6       : ok=2    changed=1
unreachable=0    failed=0    skipped=0    rescued=0
ignored=0
172.16.1.7       : ok=2    changed=1
unreachable=0    failed=0    skipped=0    rescued=0
ignored=0
172.16.1.8       : ok=2    changed=1
unreachable=0    failed=0    skipped=0    rescued=0
ignored=0
172.16.1.9       : ok=2    changed=1
unreachable=0    failed=0    skipped=0    rescued=0
ignored=0
```

2.执行playbook，注意观察执行返回的状态颜色:

红色：表示有task执行失败，通常都会提示错误信息。

黄色：表示远程主机按照编排的任务执行且进行了改变。

绿色：表示该主机已经是描述后的状态，无需再次运行。

4.Ansible Playbook练习实验

案例一、使用ansible安装并配置nfs服务

#1.梳理流程 步骤

每一步使用模块

```
# db01 172.16.1.51
```

```
#在backup服务器上安装nfs
```

```
ansible 172.16.1.41 -i hosts -m yum -a 'name=nfs-utils  
state=present'
```

```
##配置
```

```
cat /etc/exports
```

```
/data-lidao/ 172.16.1.0/24(rw,all_squash) #默认压缩为  
nfsnobody用户
```

```
ansible 172.16.1.41 -i hosts -m copy -a  
'content="/data-lidao/ 172.16.1.0/24(rw,all_squash)"  
dest=/etc/exports backup=yes'
```

```
#创建目录 修改所有者
```

```
[root@m01 ~]# ansible 172.16.1.41 -m file -a  
'path=/data-lidao/ owner=nfsnobody group=nfsnobody  
state=directory ' -i hosts
```

```
#启动服务并开机自启动
```

```
ansible 172.16.1.41 -i hosts -m service -a  
'name=rpcbind state=started enabled=yes'  
ansible 172.16.1.41 -i hosts -m service -a 'name=nfs  
state=started enabled=yes'
```

```
#backup上面进行挂载(本地测试)
```

```
ansible 172.16.1.41 -i hosts -m mount -a  
'src=172.16.1.41:/data-lidao/ path=/mnt/ fstype=nfs  
state=mounted'
```

```
#web服务器进行挂载
```

```
挂载到web服务器的 /code/upload/img
```

```
[root@m01 ~]# ansible web -i hosts -m mount -a  
'src=172.16.1.41:/data-lidao path=/code/upload/img  
fstype=nfs state=mounted'
```

```
#
```

- 1. 书写playbook

```
[root@m01 /server/playbook]# cat 02_nfs.yml
---
- hosts: 172.16.1.51
  tasks:
    - name: install nfs
      yum: name=rpcbind,nfs-utils state=installed
    - name: nfs configure file
      copy: src=./exports.j2 dest=/etc/exports
            backup=yes
    - name: mkdir share dir
      file: path=/data-lidao996 state=directory
            owner=nfsnobody group=nfsnobody
    - name: start rpcbind
      systemd: name=rpcbind state=started enabled=yes
    - name: start nfs
      systemd: name=nfs state=started enabled=yes
    - name: mount local
      mount: src=172.16.1.51:/data-lidao996 path=/mnt
              fstype=nfs state=mounted
```

- 2. 根据playbook准备你的环境

```
[root@m01 /server/playbook]# echo '/data-lidao996
172.16.1.0/24(rw,all_squash)' >./exports.j2
[root@m01 /server/playbook]# cat ./exports.j2
/data-lidao996 172.16.1.0/24(rw,all_squash)
```

- 3. 检查语法并执行

```
[root@m01 /server/playbook]# ansible-playbook -i hosts 02_nfs.yml -C
```

- 4. 在其他机器上面进行挂载web服务器 进行挂载

```
[root@m01 /server/playbook]# cat 02_nfs.yml
---
- hosts: 172.16.1.51
  tasks:
    - name: install nfs
      yum: name=rpcbind,nfs-utils state=installed
    - name: nfs configure file
      copy: src=./exports.j2 dest=/etc/exports
      backup=yes
    - name: mkdir share dir
      file: path=/data-lidao996 state=directory
      owner=nfsnobody group=nfsnobody
    - name: start rpcbind
      systemd: name=rpcbind state=started enabled=yes
    - name: start nfs
      systemd: name=nfs state=started enabled=yes
    - name: mount local
      mount: src=172.16.1.51:/data-lidao996 path=/mnt
      fstype=nfs state=mounted
```



```
- hosts: web
  tasks:
    - name: web server mount  nfs
      mount: src=172.16.1.51:/data-lidao996
      path=/code/upload/dbserver  fstype=nfs  state=mounted
```

1) 编写安装配置nfs服务的playbook文件

```
[root@m01 ~]# cd /etc/ansible/playbook/
[root@m01 playbook]# cat nfs.yml
---
- hosts: web
  tasks:
    - name: Install NFS Server
      yum: name=nfs-utils state=latest

    - name: Configure NFS Server
      copy: src=./exports.j2 dest=/etc/exports

    - name: Create Data Directory
      file: path=/data state=directory owner=nfsnobody
      group=nfsnobody recurse=yes

    - name: Start NFS Server
      service: name=nfs state=started enabled=yes
```

2) 准备playbook依赖的exports.j2文件

```
[root@m01 playbook]# echo "/data 172.16.1.0/24(rw, sync)" >
exports.j2
```

3) 检查playbook语法

```
[root@m01 playbook]# ansible-playbook nfs.yml --syntax-check
```

```
playbook: nfs.yml
```

4) 执行playbook

5) 客户端执行命令测试

```
[root@m01 playbook]# showmount -e 172.16.1.8
Export list for 172.16.1.8:
/data 172.16.1.0/24
[root@m01 playbook]# showmount -e 172.16.1.7
Export list for 172.16.1.7:
/data 172.16.1.0/24
```

案例二、使用ansible安装并配置nginx服务

1. nginx

1. 安装nginx服务

#yum_repository

```
ansible 172.16.1.31 -i hosts -m yum_repository -a
'name=nginx description="nginx repo"
baseurl=http://nginx.org/packages/centos/7/x86_64/
enabled=yes gpgcheck=no state=present'
```

#yum

```
[root@m01 ~]# ansible 172.16.1.31 -i hosts -m yum -a
'name=nginx state=installed'
```

2. 编写简单网页测试内容

```
ansible 172.16.1.31 -i hosts -m copy -a
'content="backup.olldoby.com"
dest=/usr/share/nginx/html/index.html '
```

3. 启动服务不加入开机自启 #systemd/service

```
[root@m01 ~]# ansible 172.16.1.31 -i hosts -m systemd
-a 'name=nginx state=started enabled=yes'
```

4.放行对应的端口

#iptables

```
[root@m01 ~]# ansible 172.16.1.31 -i hosts -m iptables  
-a 'table=filter action=append chain=INPUT protocol=tcp  
destination_port=80 jump=ACCEPT'
```

#playbook 剧本

```
[root@m01 /server/playbook]# cat 03_nginx.yml
```

```
- hosts: 172.16.1.9
```

```
  tasks:
```

```
    - name: Add Nginx Yum Repo
```

```
      yum_repository:
```

```
        name: nginx
```

```
        description: nginx repo
```

```
        baseurl:
```

```
http://nginx.org/packages/centos/$releasever/$basearch/
```

```
        enabled: yes
```

```
        gpgcheck: yes
```

```
        gpgkey: https://nginx.org/keys/nginx_signing.key
```

```
    - name: Install Nginx
```

```
      yum:
```

```
        name: nginx
```

```
        state: installed
```

```
    - name: Index File
```

```
      copy:
```

```
        content: "This is ansible website  
ansible.oldboy.com"
```

```
        dest: /usr/share/nginx/html/index.html
```

```
    - name: Copy Nginx.d/conf File
```

```
      copy:
```

```
        src: ./www.conf
```

```
        dest: /etc/nginx/conf.d/default.conf
```

```
        backup: yes
```

```
    - name: Start Nginx
```

```
      systemd:
```

```
        name: nginx
```

```
        state: started
```

```
        enabled: yes
```

```
[root@m01 /server/playbook]# cat www.conf
server {
    listen 80;
    server_name ansible.oldboy.com;
    location / {
        root /usr/share/nginx/html;
        index index.html;
    }
}
```

2. 目前问题: 希望nginx配置如果发生变化,才重启nginx

notify 监控状态的变化(模块),如果变化 根据你指定的命令 去执行指令

handlers 实施具体的东西

```
[root@m01 /server/playbook]# cat 03_nginx.yml
---
- hosts: 172.16.1.9
  tasks:
    - name: Add Nginx Yum Repo
      yum_repository:
        name: nginx
        description: nginx repo
        baseurl:
http://nginx.org/packages/centos/$releasever/$basearch/
        enabled: yes
        gpgcheck: yes
        gpgkey: https://nginx.org/keys/nginx_signing.key
    - name: Install Nginx
      yum:
        name: nginx
        state: installed
    - name: Index File
      copy:
```

```

        content: "This is ansible website
ansible.oldboy.com"
        dest: /usr/share/nginx/html/index.html
- name: Copy Nginx.d/conf File
  copy:
    src: ./www.conf
    dest: /etc/nginx/conf.d/default.conf
    backup: yes
  notify: Restart Nginx
- name: Start Nginx
  systemd:
    name: nginx
    state: started
    enabled: yes
handlers:
- name: Restart Nginx
  systemd:
    name: nginx
    state: restarted

```

激活多个handlers

```

- name: install nfs
  yum: name=rpcbind,nfs-utils state=installed
- name: nfs configure file
  copy: src=./exports.j2 dest=/etc/exports backup=yes
  notify: 如果发生变化 激活 notify内容
    - Restart NFS
    - Remount Client
- name: mkdir share dir
- name: start rpcbind
- name: start nfs
- name: mount local
handlers:
- name: Restart NFS
  systemd:
    name: nfs
    state: restarted
- name: Remount Client
  mount: src=172.16.1.51:/data-lidao996 path=/mnt fstype=nfs state=remounted
hosts: web
tasks:
- name: web server mount nfs
  mount: src=172.16.1.51:/data-lidao996 path=/code/upload/dbserver fstype=nfs state=mounted

```

RUNNING HANDLER [Restart NFS]

```

*****
*****
changed: [172.16.1.51]

```

RUNNING HANDLER [Remount Client]

changed: [172.16.1.51]

```
[root@m01 /server/playbook]# cat 02_nfs.yml
```

- hosts: 172.16.1.51

tasks:

- name: install nfs

yum: name=rpcbind,nfs-utils state=installed

- name: nfs configure file

copy: src=./exports.j2 dest=/etc/exports

backup=yes

notify:

- Restart NFS

- Remount Client

- name: mkdir share dir

file: path=/data-lidao996 state=directory

owner=nfsnobody group=nfsnobody

- name: start rpcbind

systemd: name=rpcbind state=started enabled=yes

- name: start nfs

systemd: name=nfs state=started enabled=yes

- name: mount local

mount: src=172.16.1.51:/data-lidao996 path=/mnt

fstype=nfs state=mounted

handlers:

- name: Restart NFS

systemd:

name: nfs

state: restarted

- name: Remount Client

mount: src=172.16.1.51:/data-lidao996 path=/mnt

fstype=nfs state=remounted

- hosts: web

tasks:

- name: web server mount nfs

```
mount: src=172.16.1.51:/data-lidao996
path=/code/upload/dbserver fstype=nfs state=mounted
```

案例二、使用ansible安装并配置httpd服务

1) 编写安装配置httpd服务的playbook文件

```
[root@m01 playbook]# cat web.yml
---
- hosts: web
  tasks:
    - name: Installed Httpd Server
      yum: name=httpd state=latest

    - name: Started Httpd Server
      service: name=httpd state=started enabled=yes

    - name: Started Firewallld Server
      service: name=firewalld state=started enabled=yes

    - name: Copy Httpd Web Page
      copy: content='This is Web Page'
            dest=/var/www/html/index.html

    - name: Configure Firewallld Permit Http
      firewallld: service=http immediate=yes permanent=yes
                  state=enabled
```

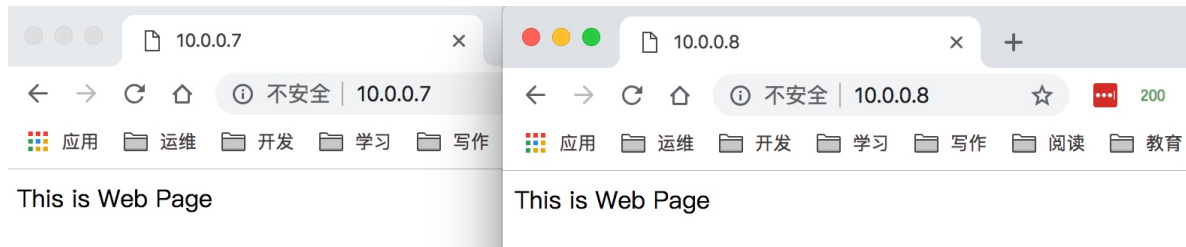
2) 检查playbook语法

```
[root@m01 playbook]# ansible-playbook web.yml --syntax-check
```

```
playbook: web.yml
```

3) 执行playbook

4. 访问服务器对应的web页面测试



案例三、ansible安装并配置httpd服务，根据不同的主机配置不同的网站。（多个play使用方式，但不是生产推荐了解即可），生产推荐使用循环方式）

1) 编写安装配置httpd服务的playbook文件

```
[root@m01 playbook]# cat web.yml
---
- hosts: web
  tasks:
    - name: Installed Httpd Server
      yum: name=httpd state=latest

    - name: Started Httpd Server
      service: name=httpd state=started enabled=yes

    - name: Started Firewall Server
      service: name=firewalld state=started enabled=yes

    - name: Configure Firewall Permit Http
      firewall: service=http immediate=yes permanent=yes
state=enabled

- hosts: 172.16.1.7 #单独针对7
  tasks:
    - name: Configure Httpd Web Page
```



```

    copy: content='This is Web-7 Page'
dest=/var/www/html/index.html

- hosts: 172.16.1.8 #单独针对7
  tasks:
    - name: Configure Httpd Web Page
      copy: content='This is Web-8 Page'
dest=/var/www/html/index.html

```

2) 检查playbook语法

```

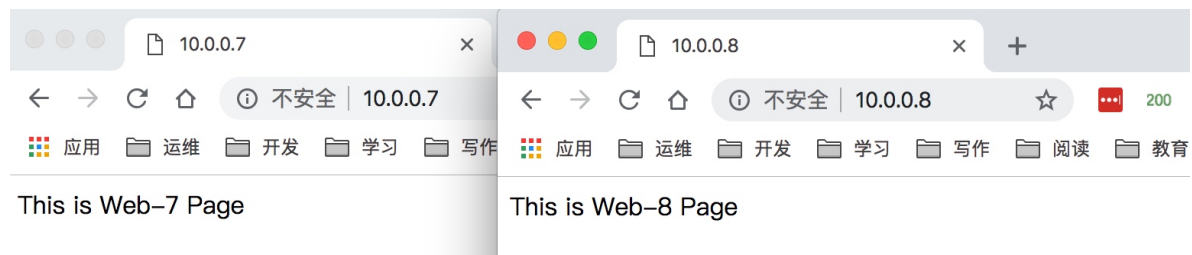
[root@m01 playbook]# ansible-playbook web.yml --syntax-check

playbook: web.yml

```

3) 执行playbook

4. 访问服务器对应的web页面测试



##

5. Ansible Playbook案例实践-LNMP 环境

使用AnsiblePlaybook方式构建LAMP架构, 具体操作步骤如下:

1. 使用yum安装 httpd、php、php-mysql、mariadb、firewalld 等
2. 启动httpd、firewalld、mariadb等服务
3. 添加防火墙规则, 放行http的流量, 并永久生效
4. 使用get_url 下载 www.01dboylinux.cn/index.html 文件

1) 将被管理主机进行分组, 分组名称定义为web

```

[root@m01 ~]# cat /etc/ansible/hosts
[web]
172.16.1.7
172.16.1.8

```

2) 编写对应的playbook文件

```
[root@m01 ~]# cd /etc/ansible/playbook/
[root@m01 playbook]# cat lamp.yml
---
- hosts: web
  tasks:
    - name: Installed LAMP Server
      yum: name=httpd,php,php-mysql,mariadb state=latest

    - name: Started Httpd Server
      service: name=httpd state=started enable=yes

    - name: Started Firewallld Server
      service: name=httpd state=started enable=yes

    - name: Get Url Index.php File
      get_url: url=http://www.oldboylinu.cn/index.php
      dest=/var/www/html/index.php

    - name: Configure Firewallld Permit Http
      firewallld: service=http immediate=yes permanent=yes
      state=enable
```

3) 检查playbook要执行的主机是否正确

```
[root@m01 playbook]# ansible-playbook lamp.yml --list-hosts
-i /etc/ansible/hosts

playbook: lamp.yml

play #1 (web): web    TAGS: []
  pattern: [u'web']
  hosts (2):
    172.16.1.7
    172.16.1.8
```

4) 检查playbook语法是否有错误, 并不会帮我们检查抒写的逻辑错误

```
[root@m01 playbook]# ansible-playbook --syntax-check  
lamp.yml
```

```
playbook: lamp.yml
```

5) 运行Playbook, 如果是生产环境记得使用C参数模拟执行

6) 打开浏览器检查