

第1章 Ansible剧本介绍

1.什么是playbook剧本

- 1 电影剧本：
- 2 电影名
- 3 演员
- 4 场景
- 5 时间
- 6 事件
- 7 台词
- 8
- 9 **Ansible剧本：**
- 10 一系列的任务按照我们期望的结果编排在一起
- 11 **playbook组成：**
- 12 **hosts：** 定义主机角色
- 13 **tasks：** 具体执行的任务
- 14 简单理解：不同的模块去完成一件事

举例: 班长的阳光快乐时光

- 1 - 演员列表： 慧总
- 2 场景：
- 3 - 场景1： 慧总手里拿着肥皂走进浴室
- 4 动作场面： 手一滑，肥皂在空中托马斯360度回旋落地
- 5
- 6 - 场景2： 慧总捡肥皂
- 7 动作场面： 慧总弯腰的时候(马赛克),美女房东从后面出现了
- 8
- 9 - **hosts**需要执行的主机： **nfs**
- 10 **tasks**任务：
- 11 - 任务1： 创建用户
- 12 动作： 创建用户的命令
- 13 - 任务2： 创建目录
- 14 动作： 创建目录的命令

2.playbook剧本的优势

- 1 1.减少重复的书写的指令： `ansible backup -m file -a`
- 2 2.看起来简洁清晰
- 3 3.功能强大，可以控制流程，比如：判断，循环，变量，标签
- 4 4.其他剧本可以复用
- 5 5.提供语法检查以及模拟执行

第2章 剧本的格式书写要求

1.YAML格式特点

- 1 1.严格的缩进表示层级关系
- 2 2.不要使用tab缩进
- 3 3.: 后面一定要有空格
- 4 4.- 后面一定要有空格
- 5 5.文件后缀名需要改为yaml或yml, vim可以智能高亮提示

2.剧本的组成

- 1 hosts: 需要执行的主机
- 2 tasks: 需要执行的任务
- 3 name: 任务名称

第3章 编写Rsync剧本

0.官网举例

- 1 https://docs.ansible.com/ansible/latest/user_guide/playbooks_intro.html#playbook-language-example

1.命令行模式的编写

```
1 #1.创建www组和www用户
2 ansible backup -m group -a "name=www gid=666"
3 ansible backup -m user -a "name=www uid='666' group=www create_home=no
  shell=/sbin/nologin"
4
5 #2.创建数据目录并更改授权
6 ansible backup -m file -a "path=/data state=directory owner=www group=www
  mode='755'"
7 ansible backup -m file -a "path=/backup state=directory owner=www group=www
  mode='755'"
8
9 #3.安装rsync软件
10 ansible backup -m yum -a "name=rsync state=latest"
11
12 #4.复制配置文件和密码文件
13 ansible backup -m copy -a "src=/root/script/rsync/rsyncd.conf dest=/etc/"
14 ansible backup -m copy -a "src=/root/script/rsync/rsync.passwd dest=/etc/
  mode='600'"
15
16 #6.启动服务
17 ansible backup -m systemd -a "name=rsyncd state=started enabled=yes"
```

2.改写成剧本

```
1 - hosts: backup
2   tasks:
3     - name: 01创建www用户组
4       group:
5         name: www
6         gid: 666
7
```

```
8 - name: 02创建www用户
9   user:
10     name: www
11     uid: '666'
12     group: www
13     create_home: no
14     shell: /sbin/nologin
15
16 - name: 03创建数据目录并更改授权
17   file:
18     path: /data
19     state: directory
20     owner: www
21     group: www
22     mode: '755'
23
24 - name: 04安装rsync软件
25   yum:
26     name: rsync
27     state: latest
28
29 - name: 05复制配置文件和密码文件
30   copy:
31     src: /root/script/rsync/rsyncd.conf
32     dest: /etc/
33
34 - name: 06创建密码文件权限为600
35   copy:
36     src: /root/script/rsync/rsync.passwd
37     dest: /etc/
38     mode: 600
39
40 - name: 07启动服务
41   systemd:
42     name: rsyncd
43     state: started
44     enabled: yes
```

3.模拟执行

```
1 ansible-playbook -C rsync_install.yaml
```

4.执行

```
1 ansible-playbook rsync_install.yaml
```

第4章 编写NFS剧本

1.命令行模式的编写

NFS服务端：

```
1 [root@m-61 /scripts]# cat nfs_server_install.yaml
2 - hosts: nfs_server
```

```

3 tasks:
4   - name: 01-add group
5     group: name=www gid='666'
6   - name: 02-add user
7     user: name=www create_home=no shell=/sbin/nologin group=www uid=666
8   - name: 03-install nfs service
9     yum: name=nfs-utils state=latest
10  - name: 04-copy nfs exports
11    copy: src=/server/scripts/exports dest=/etc/
12  - name: 05-create data dir
13    file: path=/data state=directory owner=www group=www
14  - name: 06-start rpcbind
15    service: name=rpcbind state=started
16  - name: 07-start nfs
17    service: name=nfs state=started
18  - name: 08-enable rpcbind

19    systemd: name=rpcbind enabled=yes
20  - name: 09-enable nfs
21    systemd: name=nfs enabled=yes

```

NFS客户端:

```

1 [root@m-61 /scripts]# cat nfs_client_install.yaml
2 - hosts: nfs_client
3   tasks:
4     - name: 01-add group
5       group: name=www gid=666
6     - name: 02-add user
7       user: name=www create_home=no shell=/sbin/nologin group=www uid=666
8     - name: 03-install nfs service
9       yum: name=nfs-utils state=latest
10    - name: 04-create data dir
11      file: path=/data state=directory owner=www group=www
12    - name: 05-start rpcbind
13      service: name=rpcbind state=started
14    - name: 06-enable rpcbind
15      systemd: name=rpcbind enabled=yes
16    - name: 07-mount data
17      mount: path=/data src=172.16.1.31:/data fstype=nfs opts=defaults
          state=mounted

```

2.改写成剧本

1

第5章 剧本高级特性-循环

0.官方文档

1 https://docs.ansible.com/ansible/latest/user_guide/playbooks_loops.html?highlight=loop

1.应用场景

- 1 安装多个软件
- 2 创建多个目录
- 3 复制多个目录
- 4 复制多个文件到不同的目录
- 5 不同的文件权限不一样

2.循环书写风格1：单行模式

```
1 - name: create_data
2   file: path=/data state=directory owner=www group=www
3
4 - name: create_backup
5   file: path=/backup state=directory owner=www group=www
```

3.循环书写风格2：缩进模式

需求: 创建2个目录/data和/backup

以前的写法:

```
1 - name: create_data
2   file:
3     path: /data
4     state: directory
5     owner: www
6     group: www
7
8 - name: create_data
9   file:
10    path: /backup
11    state: directory
12    owner: www
13    group: www
```

循环实现:

```
1 - name: create_data
2   file:
3     path: "{{ item }}"
4     state: directory
5     owner: www
6     group: www
7   loop:
8     - /data
9     - /backup
```

4.循环书写风格3：混合风格

```
1 - name: create_data
2   file: path="{{ item }}" state=directory owner=www group=www
3   loop:
4     - /data
5     - /backup
```

5.循环书写风格3: 多参数循环模式

```
1 - hosts: backup
2   tasks:
3     - name: create_data
4       file:
5         path: "{{ item.path }}"
6         state: directory
7         owner: www
8         group: www
9         mode: "{{ item.mode }}"
10      loop:
11        - { path: '/data' , mode: '755' }
12        - { path: '/backup', mode: '777' }
```

第6章 剧本高级特性-变量

1.应用场景

1. 自定义某个变量，在任务中被多次引用
2. 从主机收集到系统信息里提取某个变量，比如IP地址，主机名

2.自定义变量并引用

```
1 - hosts: backup
2   vars:
3     data_path: /data/
4     dest_path: /etc/
5     file_path: /etc/rsync.passwd
6
7   tasks:
8     - name: 01mkdir
9       file:
10        path: "{{ data_path }}"
11        state: directory
12
13     - name: 02copy
14       copy:
15        src: "{{ file_path }}"
16        dest: "{{ dest_path }}"
```

3.使用变量获取主机的eth1地址和主机名

```

1 - hosts: all
2   tasks:
3     - name: 01get IP
4       shell: "echo {{ ansible_default_ipv4.address }} >> /tmp/ip.txt"
5     - name: 02get hostname
6       shell: "echo {{ ansible_hostname }} >> /tmp/hostname.txt"

```

4.在主机清单文件里定义变量

主机清单

```

1 [root@m01 ~/ansible_script]# cat /etc/ansible/hosts
2 [web]
3 10.0.0.7 port=8888
4 10.0.0.8 port=9999
5
6 [web:vars]
7 nginx_version='1.19'

```

引用变量

```

1 [root@m-61 /script/playbook]# cat web_vars.yaml
2 - hosts: web
3   tasks:
4     - name: 01get port
5       shell: "echo {{ port }} >> /tmp/port.txt"
6     - name: 02get version
7       shell: "echo {{ nginx_version }} >> /tmp/version.txt"

```

5.循环里引用变量-来自深圳6期班长同桌超总的灵魂发问

```

1 - name: test for
2   hosts: backup
3   vars:
4     rsyncd_conf: /script/rsyncd.conf
5     rsyncd_pass: /script/rsync.passwd
6   tasks:
7     - name: 01-copy
8       copy:
9         src: "{{ item.src }}"
10        dest: /etc/
11        mode: "{{ item.mode }}"
12      loop:
13        - { src: "{{ rsyncd_conf }}", mode: '0644' }
14        - { src: "{{ rsyncd_pass }}", mode: '0600' }

```

6.深圳4期夏xx同学整理ansible内置变量

```

1 其他ansible内置变量
2  ansible_facts.eth0.ipv4.address
3  ansible_facts.eth1.ipv4.address
4  ansible_nodename 节点名字
5  ansible_form_factor 服务器类型

```

```

6     ansible_virtualization_role 虚拟机角色（宿主机或者虚拟机）
7     ansible_virtualization_type 虚拟机类型（kvm）
8     ansible_system_vendor 供应商（Dell）
9     ansible_product_name 产品型号（PowerEdge R530）
10    ansible_product_serial 序列号（sn）
11    ansible_machine 计算机架构（x86_64）
12    ansible_bios_version BIOS版本
13    ansible_system 操作系统类型（linux）
14    ansible_os_family 操作系统家族（RedHat）
15    ansible_distribution 操作系统发行版（CentOS）
16    ansible_distribution_major_version 操作系统发行版主版本号（7）
17    ansible_distribution_release 操作系统发行版代号（core）
18    ansible_distribution_version 操作系统发行版本号（7.3.1611）
19    ansible_architecture 体系（x86_64）
20    ansible_kernel 操作系统内核版本号
21    ansible_userspace_architecture 用户模式体系（x86_64）
22    ansible_userspace_bits 用户模式位数
23    ansible_pkg_mgr 软件包管理器
24    ansible_selinux.status selinux状态
25    #-----
26    ansible_processor CPU产品名称
27    ansible_processor_count CPU数量
28    ansible_processor_cores 单颗CPU核心数量
29    ansible_processor_threads_per_core 每个核心线程数量
30    ansible_processor_vcpus CPU核心总数
31    ansible_memtotal_mb 内存空间
32    ansible_swaptotal_mb 交换空间
33    ansible_fqdn 主机的域名
34    ansible_default_ipv4.interface 默认网卡
35    ansible_default_ipv4.address 默认IP地址
36    ansible_default_ipv4.gateway 默认网关
37    ***** json 格式 *****
38    ansible_devices 硬盘设备名
39    ansible_devices.vendor 硬盘供应商
40    ansible_devices.model 硬盘整列卡型号
41    ansible_devices.host 硬盘整列卡控制器
42    ansible_devices.size 设备存储空间
43    ***** json 格式 *****
44    ansible_interfaces 网卡
45    ansible_{interfaces}.ipv4.address 网卡IP地址
46    ansible_{interfaces}.ipv6.0.address 网卡IPv6地址
47    ansible_{interfaces}.macaddress 网卡mac地址

```

第7章 剧本高级特性-注册变量

1.应用场景

- 1 调试，回显命令执行的内容
- 2 把状态保存成变量，其他任务可以进行判断或引用

2.使用内置变量将IP地址保存到文本里，并将文本内容显示出来

案例1:引用单个注册变量


```

1 - hosts: all
2   tasks:
3     - name: echo IP
4       shell: "echo {{ ansible_default_ipv4.address }} >> /tmp/ip.txt"
5
6     - name: cat IP
7       shell: "cat /tmp/ip.txt"
8       register: ip_txt
9
10    - debug:
11      msg: "{{ ip_txt.stdout_lines }}"

```

案例2:引用多个注册变量

```

1 [root@m-61 /script/playbook]# cat register.yml
2 - hosts: nfs
3   tasks:
4     - name: 01get IP
5       shell: "echo {{ ansible_default_ipv4.address }} > /tmp/ip.txt"
6     - name: 02get hostname
7       shell: "echo {{ ansible_hostname }} > /tmp/hostname.txt"
8
9     - name: 03get hostname
10      shell: "cat /tmp/hostname.txt"
11      register: hostname
12
13     - name: 04cat
14      shell: "showmount -e 172.16.1.31"
15      register: showmount
16
17     - debug:
18       msg: "{{ item }}"
19     loop:
20       - "{{ showmount.stdout_lines }}"
21       - "{{ hostname.stdout_lines }}"

```

3.如果配置文件发生了变化,就重启服务,否则不重启

```

1 - hosts: backup
2   tasks:
3     - name: 01-copy_conf
4       copy:
5         src: /opt/rsyncd.conf
6         dest: /etc/
7         register: conf_status
8
9     - name: 02-start
10      systemd:
11        name: rsyncd
12        state: started
13        enabled: yes
14
15     - name: 03-restart
16      systemd:
17        name: rsyncd

```

```
18 state: restarted
19 when: conf_status.changed
```

4.注册变量和判断场景

官方地址:

1 | https://docs.ansible.com/ansible/latest/user_guide/playbooks_conditionals.html

使用场景:

```
1 场景:
2 判断所有机器/tmp/下有没有ip.txt的文件
3
4 如果有, 打印出来内容并且格式为:
5 例如:
6
7 web01 has ip.txt
8 内容为:
9
10 如果不存在:
11 输出内容: nfs is nofile
```

5.深圳4期张xx同学解决方案

```
1 - hosts: all
2   vars:
3     path1: /tmp/ip
4   tasks:
5     - name: test1
6       shell: 'cat {{ path1 }}'
7       register: retval
8       ignore_errors: true
9
10    - name: test2
11      debug:
12        msg: '{{ansible_hostname}} has {{path1}} , content is:
13        {{retval.stdout}}'
14      when: retval is success
15
16    - name: test3
17      debug:
18        msg: '{{path1}} is nofile'
19      when: retval is failed
```

第8章 剧本高级特性-服务状态管理

0.官方文档

1 | https://docs.ansible.com/ansible/latest/user_guide/playbooks_handlers.html

1.应用场景

- 1 目前的情况：
- 2 配置文件发生变化也不会重启
- 3
- 4 理想中的情况：
- 5 1.如果配置文件不发生变化,就不执行重启
- 6 2.如果配置文件发生变化,就执行重启

2.命令实现

```
1 - hosts: backup
2   tasks:
3     - name: 01-copy_conf
4       copy:
5         src: /script/rsync/rsyncd.conf
6         dest: /etc/
7       notify:
8         - restart rsyncd
9
10    - name: 02-start
11      systemd:
12        name: rsyncd
13        state: started
14        enabled: yes
15
16    handlers:
17      - name: restart rsyncd
18        systemd:
19          name: rsyncd
20          state: restarted
```

3.错误总结

- 1 1.handlers位置要放在最后
- 2 2.handlers里任务定义的名字是什么,notify里就写什么,不能不一样

第9章 剧本高级特性-选择标签

1.应用场景

- 1 调试,选择性的执行任务

2.添加标签-深圳4期班长-杨x同学编写

```
1 - hosts: nfs
2   tasks:
3     - name: 01-add group
4       group: name=www gid=666
5       tags: 01-add-group
6
7     - name: 02-add user
8       user: name=www create_home=no shell=/sbin/nologin group=www uid=666
9       tags: 02-add-user
```

```
10
11 - name: 03-install nfs service
12   yum: name=nfs-utils state=latest
13   tags: 03-install nfs service
14
15 - name: 04-copy nfs exports
16   copy: src=/service/scripts/exports dest=/etc/
17   tags: 04-copy-nfs-exports
18
19 - name: 05-create data dir
20   file: path=/data state=directory owner=www group=www
21   tags: 05-create-data-dir
22
23 - name: 06-create passwd conf
24   copy: content='123' dest=/etc/rsync.passwd mode=600
25   tags: 06-create-passwd
26
27 - name: 07-start rpcbind
28   service: name=rpcbind state=started
29   tags: 07-start-rpcbind
30
31 - name: 08-start nfs
32   service: name=nfs state=started
33   tags: 08-start-nfs
34
35 - name: 09-enable rpcbind
36   systemd: name=rpcbind enabled=yes
37   tags: 09-enable-rpcbind
38
39 - name: 10-enable nfs
40   systemd: name=nfs enabled=yes
41   tags: 10-enable-nfs
```

3.打印出playbook里要执行的所有标签

```
1 ansible-playbook --list-tags rsync_install.yaml
```

4.指定运行某个标签

```
1 ansible-playbook -t '03-install nfs service' rsync_install_tag.yaml
```

5.指定运行多个标签

```
1 ansible-playbook -t 01-add-group,02-add-user,05-create-data-dir
  rsync_install_tag.yaml
```

6.指定不运行某个标签

```
1 ansible-playbook --skip-tags 01-add-group rsync_install_tag.yaml
```

7.指定不运行多个标签

```
1 | ansible-playbook --skip-tags 01-add-group,02-add-user,04-copy-nfs-exports  
rsync_install_tag.yaml
```

第10章 剧本高级特性-选择tasks

1.应用场景

- 1 | 调试的时候
- 2 | 从某个任务开始往下依次执行

2.查看task列表

```
1 | ansible-playbook --list-tasks rsync_install_tag.yaml
```

3.选择从哪一个task开始执行

```
1 | ansible-playbook --start-at-task '05-create data dir' rsync_install_tag.yaml
```

第11章 运行检查规范

1.检查剧本拼写规范

```
1 | ansible-playbook --syntax-check rsync_install.yaml
```

2.检查这个任务执行的主机对象

```
1 | ansible-playbook --list-hosts rsync_install.yaml
```

3.检查这个剧本需要执行哪些任务

```
1 | ansible-playbook --list-tasks rsync_install.yaml
```

4.模拟执行剧本

```
1 | ansible-playbook -C rsync_install.yaml
```

5.真正执行

```
1 | ansible-playbook rsync_install.yaml
```