第3天-自动化运维利器 Ansible-Playbook

一、 Ad-Hoc 的问题

通过对 AD-HOC 的学习，我们发现 AD-HOC 每次只能在被管理节点上执行简单的命令。

而日常工作中，我们往往面临的是一系列的复杂操作，例如我们有可能需要安装软件、更新配置、启动服务等等一系列操作的结合。此时再通过 AD-HOC 去完成任务就有些力不从心了。

在这种场景下，Ansible引进了 PLAYBOOK 来帮忙我们解决这样复杂问题。

二、 PlayBook是什么

Playbook 也通常被大家翻译成剧本。

可以认为它是Ansible 自定义的一门语言(可以将 Playbook 比作 Linux 中的 shell，而 Ansible 中的 Module 可以比作为 Linux 中的各种命令。)

通过这样的类比，我们对PlayBook就有了一个更形象的认识了。

既然 Playbook 是一门语言，那么它遵循什么样的语法格式? 有哪些语言呢性? 我们将通过下面的学习逐步了解。

三、 YAML 学习

PlayBook遵循YAML 的语法格式。

因此在学习PlayBook之前，我们必须要先弄明白YAML 相关知识点。

1. YAML特点

YAML 文件

- 以 # 为注释符
- 以 .yml 或者.yaml 结尾
- 以 --- 开始 , 以 ... 结束, 但开始和结束标志都是可选的

2. 基本语法

- 大小写敏感
- 使用缩进表示层级关系
- 缩进时是使用Tab键还是使用空格一定要达到统一，建议使用空格。
- 相同层级的元素必须左侧对齐即可

YAML 支持的数据结构有三种

- 字符串
- 列表
- 字典

接下来分别介绍它们。

2.1 字符串

```
---
# YAML 中的字符串可以不使用引号，即使里面存在空格的时候，当然了使用单引号和双引号也没有错。
this is a string
```

```yaml
'this is a string'
"this is a string"
# YAML 中若一行写不完你要表述的内容的时候，可以进行折行。写法如下：
long_line: |
        Example 1
        Example 2
        Example 3
# 或者
long_line: >
        Example 1
        Example 2
        Example 3
...
```

2.2 列表

```yaml
---
# 若熟悉 Python 的话，可以认为它就是Python中的List ,若熟悉 C 语言的话， 可以认为它是 C 中的数组。
# 如何定义：以短横线开头 + 空格 + 具体的值
- red
- green
- blue

# 以上的值假如转换成 python 的 List 会是这样：
# ['red', 'green', 'blue']
...
```

2.3 字典

```yaml
---
# 若熟悉 Python 的话， 可以认为它就是 Python 中的 Dict
# 如何定义：key + 冒号(:) + 空格 + 值(value)，即 key: value

name: Using Ansible
code: D1234

# 转换为 python 的 Dict
# {'name': 'Using Ansibel', 'code': 'D1234'}
...
```

2.4 混合结构

以上,针对YAML 的所有基础知识点就介绍完了。但在日常生活中，往往需要的数据结构会特别复杂，有可能会是字符串、列表、字典的组合形式。 这里举一个小例子: 所有人都上过学，都知道到学校里是以班级为单位。我们去使用列表和字典的形式去描述一个班级的组成。


```yaml
---
class:
  - name: stu1
    num: 001
  - name: stu2
    num: 002
  - name: stu3
    num: 003
# {'class': [{'name': 'stu1', 'num': 1},{'name': 'stu2', 'num': 2},...]}
...
```

2.5 验证YAML 语法

```
// 将YAML文件，通过 Python 的YAML 模块验证， 若不正确则报错。若正确则会输出 YAML 里的内容。
// 注意使用时，一定确保安装了yaml 软件包。
python -c 'import yaml,sys; print yaml.load(sys.stdin)' < myyaml.yml
python3 -c 'import yaml,sys; print(yaml.load(sys.stdin))' < myyaml.yml
```

Example

```
// 正确的情况
# cat myyaml.yml
---
- red
- green
- blue
...
# python -c 'import yaml,sys; print yaml.safe_load(sys.stdin)' < myyaml.yml
['red', 'green', 'blue']

// 错误的情况， 将YAML文件写错
# cat myyaml.yml
---
- red
- green
-blue
...
# python -c 'import yaml,sys; print yaml.load(sys.stdin)' < myyaml.yml
Traceback (most recent call last):
  File "<string>", line 1, in <module>
  File "/usr/local/lib/python2.7/site-packages/yaml/__init__.py", line 71, in load
```

```
    return loader.get_single_data()
  File "/usr/local/lib/python2.7/site-packages/yaml/constructor.py", line 37, in get_single_data
    node = self.get_single_node()
...
...
```

四、 Playbook 的编写

1. Play 的定义

由于Playbook 是由一个或者多个Play组成,那么如果我们熟悉Play的写法，就自然掌握了我们这章的PlayBook。

那如何定义一个Play呢 1、每一个Play 都是以短横杠开始的 2、每一个Play 都是一个YAML 字典格式

根据上面两条Play 的规则，一个假想的 Play 应该是如下的样子

```
---
- key1: value1
  key2: value2
  key3: value3
...
```

由于一个Playbook 是由一个或者多个Play构成， 那么一个含有多个Play的Playbook 结构上应该是如下的样子

```yaml
---
# 一个含有3个Play 的伪PlayBook构成
- key1: value1
  key2: value2
  key3: value3
- key4: value1
  key5: value2
  key6: value3
- key1: value1
  key2: value2
  key3: value3
...
```

2. Play 属性

以上一小节中的Play为基础, Play中的每一个key，比如 key1, key2等；这些key在PlayBook中被定义为Play的属性。

这些属性都具有特殊的意义,我们不能随意的自定义Play 的属性。

接下来就来看看 Ansible 本身都支持哪些Play属性。

常用属性

- name 属性， 每个play的名字
- hosts 属性, 每个play 涉及的被管理服务器，同ad-hoc 中的资产选择器
- tasks 属性, 每个play 中具体要完成的任务，以列表的形式表达
- become 属性，如果需要提权，则加上become 相关属性
- become_user 属性, 若提权的话，提权到哪个用户上
- remote_user属性，指定连接到远程节点的用户，就是在远程服务器上执行具体操作的用户。若不指定，则默认使用当前执行 ansible Playbook 的用户

3. 一个完整剧本

根据上一小节中介绍的真实的属性，一个含有一个Play 的 Playbook 应该是如下的样子

```
---
- name: the first play example
  hosts: webservers
  remote_user: root
  tasks:
    - name: install nginx package
      yum: name=nginx state=present
    - name: copy nginx.conf to remote server
      copy: src=nginx.conf
dest=/etc/nginx/nginx.conf
    - name: start nginx server
      service:
        name: nginx
        enabled: true
        state: started
```


4. tasks 属性中任务的多种写法

```
# 以启动 nginx 服务，并增加开机启动为例
# 一行的形式：
service: name=nginx enabled=true state=started

# 多行的形式：
service: name=nginx
         enabled=true
         state=started

# 多行写成字典的形式：
service:
```

```
  name: nginx
  enabled: true
  state: started
```

5. 具有多个Play 的Playbook

```
---
- name: manage web servers
  hosts: webservers
  remote_user: root
  tasks:
    - name: install nginx package
      yum: name=nginx state=present
    - name: copy nginx.conf to remote server
      copy: src=nginx.conf
dest=/etc/nginx/nginx.conf
    - name: start nginx server
      service:
        name: nginx
        enabled: true
        state: started
- name: manager db servers
  hosts: db_servers
  tasks:
    - name: update database confg
      copy: src=my.cnf dest=/etc/my.cnf
```


6. 如何对Playbook 进行语法校验

下面校验的方法，只能校验PlayBook是否正确，而不能校验YAML文件是否语法正确。

```
# ansible-playbook -i hosts myplaybook.yml --syntax-check
```

因为PlayBook 属于YAML 格式， 我们同样可以使用检查YAML的语法格式的方法进行检查PlayBook的语法正确性。

```
# python -c 'import yaml,sys; print yaml.safe_load(sys.stdin)' < myplaybook.yml
```

7. 如何运行PlayBook

```
# ansible-playbook -i hosts myplaybook.yml
```

8. 如何单步跟从调试PlayBook

```
// 执行Task中的任务，需要手动确认是否往下执行。
# ansible-playbook -i hosts myplaybook.yml --step
```

9. 如何测试运行PlayBook

测试运行就是会执行完整个PlayBook ,但是所有Task中的行为都不会在远程服务器上执行，所有执行都是模拟行为。

```
# ansible-playbook -i hosts myplaybook.yml -C
// -C 为大写的字母 C
```