30 Ansible Playbook

Ansible Playbook

一、YAML语法

YAML 是专门用来写配置文件的语言，非常简洁和强大。YAML 语言的设计目标，就是方便人类读写。它实质上是一种通用的数据串行化格式。

1.1 YAML语法规则

- 所有的 YAML 文件开始行都应该是 `---`
- 区分大小写。
- 使用缩进表示层级关系。
- 缩进时**不允许使用Tab键**，只允许使用空格。
- 缩进的空格数目不重要，只要相同层级的元素左侧对齐即可。
- `#` 表示行注释

1.2 YAML数据结构

YAML 支持的数据结构有三种。

- 对象：键值对的集合，又称为映射（mapping）/ 哈希（hashes） / **字典**（dictionary）
- 数组：一组按次序排列的值，又称为序列（sequence） / **列表**（list）
- 纯量（scalars）：单个的、不可再分的值

1.2.1 对象

对象的一组键值对，使用**冒号结构**表示。

`animal: pets` 对应的Python数据结构为 `{'animal': 'pets'}` 。

```yaml
hash:
  name: Steve
  foo: bar
```

上面对象对应的Python数据结构为 `{'hash': {'name': 'Steve', 'foo': 'bar'}}`

YAML也允许将所有键值对写成一个行内对象。上面对象可写为 `hash: { name: Steve, foo: bar }`

注意！`:`后要跟1个空格。

1.2.2 数组

一组连词线开头的行，构成一个数组。

```yaml
- Cat
- Dog
- Goldfish
```

上面数组对应的Python数据结构为 `[ 'Cat', 'Dog', 'Goldfish' ]` 。

数据结构的子成员是一个数组，则可以在该项下面缩进一个空格。

```yaml
-
  - Cat
  - Dog
  - Goldfish
```

上面数组对应的Python数据结构为 `[ [ 'Cat', 'Dog', 'Goldfish' ] ]`

数组也可以采用行内表示法。

`animal: [Cat, Dog]` 对应的Python数据结构为 `{'animal': ['Cat', 'Dog']}`

1.2.3 复合结构

对象和数组可以结合使用，形成复合结构。

```yaml
languages:
  - Ruby
  - Perl
  - Python
websites:
  YAML: yaml.org
  Ruby: ruby-lang.org
  Python: python.org
  Perl: use.perl.org
```

上面的复合对象对应的Python数据结构如下。

```python
{'languages': ['Ruby', 'Perl', 'Python'],
 'websites': {'Perl': 'use.perl.org',
              'Python': 'python.org',
              'Ruby': 'ruby-lang.org',
              'YAML': 'yaml.org'}}
```

1.2.4 纯量

纯量是最基本的、不可再分的值。纯量包括以下类型。

- 字符串
- 布尔值
- 整数
- 浮点数
- Null
- 时间
- 日期

数值直接以字面量的形式表示。例如：`number: 12.30`

布尔值用 `true` 和 `false` 表示。例如： `isSet: true`

null用 `~` 表示。例如： `parent: ~`

时间采用 ISO8601 格式。例如： `iso8601: 2001-12-14t21:59:43.10-05:00`

日期采用复合 ISO8601 格式的年、月、日表示。例如： `date: 1976-07-31`

1.2.5 字符串

字符串是最常见，也是最复杂的一种数据类型。

字符串默认不使用引号表示。

`str: 这是一行字符串` 对应的Python数据结构为 `{'str': '这是一行字符串'}`

如果字符串之中包含空格或特殊字符，需要放在引号之中。

`str: '内容： 字符串'` 对应的Python数据结构为 `{'str': '内容： 字符串'}`

单引号和双引号都可以使用，**双引号不会对特殊字符转义**。

`s1: '内容\n字符串'` 对应的Python数据结构为 `{'s1': '内容\\n字符串'}`

`s2: "内容\n字符串"` 对应的Python数据结构为 `{'s2': '内容\n字符串'}`

单引号之中如果还有单引号，必须连续使用两个单引号转义。

`str: 'labor''s day'` 对应的Python数据结构为 `{'str': "labor's day"}`

1.3 YAML案例

```yaml
---
# Employee records
-  martin:
     name: Martin D'vloper
     job: Developer
     skills:
       - python
       - perl
       - pascal
-  tabitha:
     name: Tabitha Bitumen
     job: Developer
     skills:
       - lisp
       - fortran
       - erlang
```

对应的Python数据结构为

```python
[{'martin': {'job': 'Developer',
             'name': "Martin D'vloper",
             'skills': ['python', 'perl', 'pascal']}},
 {'tabitha': {'job': 'Developer',
              'name': 'Tabitha Bitumen',
              'skills': ['lisp', 'fortran', 'erlang']}}]
```

二、Ansible Playbook基础

2.1 playbook语法

在很多情况下，仅仅执行单个命令或调用某一个模块，根本无法满足复杂工作的需要。Ansible允许用户根据需求，**依托于模块**，在类似于Shell脚本的模式下编写自动化运维脚本——playbook，然后由程序自动、重复地执行，从而大大提高了工作效率。

剧本文件会按照**从上到下的顺序自动运行**，其形式类似前面介绍的Shell脚本 。

Ansible官方的playbook文档：
https://docs.ansible.com/ansible/2.9/user_guide/playbooks_intro.html

Ansible官方提供了一些playbook的最佳实践可以作为参考：
https://github.com/ansible/ansible-examples

Ansible剧本（playbook）文件采用YAML语言编写，后缀名一般为 `.yml` 。

下面是官方的一个相对完整的剧本文件的实例。

YAML | 复制代码

```yaml
---
- name: 测试ping
# 主机范围
  hosts: webservers
# 变量
  vars:
    http_port: 80
    max_clients: 200
  remote_user: root
# 任务列表
  tasks:
  - name: ensure apache is at the latest version
    yum: pkg=httpd state=latest
  - name: write the apache config file
    template: src=/srv/httpd.j2 dest=/etc/httpd.conf
    notify:
    - restart apache
  - name: ensure apache is running
    service: name=httpd state=started
# 处理器
  handlers:
    - name: restart apache
      service: name=httpd state=restarted
```

Playbook可以称之为 `剧本` 。每一个剧本中都包含一系列任务，每个任务集在Ansible中又被称为“戏剧”（play）。一个剧本（Playbook）中包含多出戏剧（play）。

因此，从结构上看整个剧本（playbook）的结构是一个**数组**，每一个数组元素就是一个play，或者说**任务集**。

每个任务集由4部分组成，分别是`hosts`、`variable`、`task`、`handler`，其各自的作用如下。

- **hosts（主机范围）**：用于定义要执行任务集的主机范围。
- **variable（变量）**：用于定义任务集本执行时要用到的变量。**可选**。
- **task（任务列表）**：用于定义将在远程主机上执行的任务列表。核心部分，主要利用模块完成。
- **handler（处理器）**：用于定义执行完成后需要调用的后续任务，只有在使用**notify语句**显式调用时才会被触发。 **可选**。

`name`字段表示该任务集的名字，用于在执行过程中提示用户执行到了哪一步，以及帮助管理员在日后阅读时能想起这段代码的作用。`name`字段自行命名，没有任何限制。**可选。**

`hosts`字段表示要在哪些主机上执行任务集，多个主机组之间用逗号间隔；如果需要对全部主机进行操作，则使用all参数。

`tasks`字段用于定义要执行的任务集，每个任务都要有一个独立的`name`字段进行命名，并且每个任务的`name`字段和模块名称都要严格上下对齐，**参数要单独缩进**。

任务的核心功能都是由模块实现的，比如`yum`、`template`、`service`等。因此，在编写剧本文件时，用好`ansible-doc`帮助至关重要，帮助信息包含该模块的剧本样例。

```bash
[root@zabbix ~]# ansible-doc ping
EXAMPLES:

# Test we can logon to 'webservers' and execute python with json lib.
# ansible webservers -m ping

# Example from an Ansible Playbook
- ping:

# Induce an exception to see what happens
- ping:
    data: crash


```

了解 `ping` 模块的用法后，我们可以编写一个简单的剧本。

```yaml
[root@zabbix ~]# vi ping.yml
---
- name: 测试ping
  hosts: test
  tasks:
    - name: one
      ping:
```

运行剧本需要使用 `ansible-playbook` 命令，格式为 `ansible-playbook 剧本文件` 。

```Bash
[root@zabbix ~]# ansible-playbook ping.yml

PLAY [测试ping] ***********************************************************
*************************************************************

TASK [Gathering Facts] ****************************************************
*************************************************************
ok: [192.168.149.3]

TASK [one] ****************************************************************
*************************************************************
ok: [192.168.149.3]

PLAY RECAP ****************************************************************
*************************************************************
192.168.149.3              : ok=2    changed=0    unreachable=0    failed=0
    skipped=0    rescued=0    ignored=0
```

下面我们运行2个任务，只用在tasks对应的数组中增加一个任务即可。

```YAML
[root@zabbix ~]# vi ping.yml
---
- name: 测试ping
  hosts: test
  tasks:
    - name: one
      ping:
    - name: two
      service:
        name: zabbix-agent
        state: restarted
```

```
# 运行
[root@zabbix ~]# ansible-playbook ping.yml

PLAY [测试ping] ***********************************************************
*************************************************************

TASK [Gathering Facts] ****************************************************
*************************************************************
ok: [192.168.149.3]

TASK [one] ****************************************************************
*************************************************************
ok: [192.168.149.3]

TASK [two] ****************************************************************
*************************************************************
changed: [192.168.149.3]

PLAY RECAP ****************************************************************
*************************************************************
192.168.149.3              : ok=3    changed=1    unreachable=0    failed=
0    skipped=0    rescued=0    ignored=0
```

2.2 playbook案例

下面根据前面学习的内容完成了一个稍微复杂的playbook。

功能：执行两个任务集，第一个任务集针对test组执行两个任务：ping和重启zabbix-agent服务，第二个任务是复制文件。

准备工作：`[root@zabbix ~]# touch playbook`

编写剧本文件：

YAML | 复制代码

```yaml
[root@zabbix ~]# vi zabbix.yml
---
- name: 测试服务
  hosts: test
  tasks:
    - name: ping
      ping:
    - name: 重启zabbix-agent服务
      service:
        name: zabbix-agent
        state: restarted
- name: 文件处理
  hosts: test
  tasks:
    - name: 复制文件
      copy:
        src: /root/playbook
        dest: /tmp/playbook
        owner: root
        group: root
        mode: '0755'
```

由于YAML格式比较严格，该剧本比较复杂，因此，建议使用 `ansible-playbook` 相关选项以减少错误。

`ansible-playbook` 命令的常用选项如下。

- `--check|-C` ： 检测，**模拟剧本的执行过程，可以发现剧本执行中的问题**
- `--syntax-check` ： 语法检测，检测剧本文件的语法是否有错误
- `--list-hosts` ： 列出主机列表
- `--list-tasks` ： 列出任务
- `-v -vv` ： 显示详细执行过程

```bash
# 列出剧本的任务列表
[root@zabbix ~]# ansible-playbook  zabbix.yml --list-tasks

playbook: zabbix.yml

play #1 (test): 测试服务      TAGS: []
tasks:
ping      TAGS: []
重启zabbix-agent服务      TAGS: []

play #2 (test): 文件处理      TAGS: []
tasks:
复制文件  TAGS: []

# 列出剧本作用的主机列表
[root@zabbix ~]# ansible-playbook  zabbix.yml --list-hosts

playbook: zabbix.yml

play #1 (test): 测试服务      TAGS: []
pattern: [u'test']
hosts (1):
192.168.149.3

play #2 (test): 文件处理      TAGS: []
pattern: [u'test']
hosts (1):
192.168.149.3
# 语法检查
[root@zabbix ~]# ansible-playbook  zabbix.yml --syntax-check

playbook: zabbix.yml
# 预执行剧本，检测错误
[root@zabbix ~]# ansible-playbook  zabbix.yml -C

PLAY [测试服务] ***********************************************************
**********************************************************

TASK [Gathering Facts] ****************************************************
**********************************************************
ok: [192.168.149.3]

TASK [ping] ***************************************************************
**********************************************************
ok: [192.168.149.3]
```

```
TASK [重启zabbix-agent服务] ****************************************************
changed: [192.168.149.3]

PLAY [文件处理] ****************************************************************

TASK [Gathering Facts] *********************************************************
ok: [192.168.149.3]

TASK [服务文件] ****************************************************************
changed: [192.168.149.3]

PLAY RECAP *********************************************************************
192.168.149.3              : ok=5    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
# 执行剧本，结果与预执行结果相同
[root@zabbix ~]# ansible-playbook  zabbix.yml

PLAY [测试服务] ****************************************************************

TASK [Gathering Facts] *********************************************************
ok: [192.168.149.3]

TASK [ping] ********************************************************************
ok: [192.168.149.3]

TASK [重启zabbix-agent服务] ****************************************************
changed: [192.168.149.3]

PLAY [文件处理] ****************************************************************

TASK [Gathering Facts] *********************************************************
ok: [192.168.149.3]

TASK [服务文件] ****************************************************************
changed: [192.168.149.3]
```

```
78
79  PLAY RECAP *********************************************************************
    *********************************************************
80  192.168.149.3              : ok=5    changed=2    unreachable=0    failed=
    0    skipped=0    rescued=0    ignored=0
81
82  # 检测执行效果
83  [root@zabbix ~]# ansible test -a "ls -l /tmp/playbook"
84  192.168.149.3 | CHANGED | rc=0 >>
85  -rwxr-xr-x 1 root root 0 May 1 12:31 /tmp/playbook
```

小结

- YAML语法：对象（:后必须有一个空格）、数组
- playbook：剧本→列表，任务集→列表元素，任务集（hosts, tasks）

课程目标

- 知识目标：掌握YAML和Ansible playbook的语法。
- 技能目标：能够根据需求编写Ansible playbook完成任务。

课外拓展

- 了解更多Ansible模块的应用场景。

参考资料

- Ansible官方文档：https://docs.ansible.com/
- Ansible playbook文档：https://docs.ansible.com/ansible/2.9/user_guide/playbooks_intro.html
- 《DevOps和自动化运维实践》，余洪春，机械工业出版社
- 《Ansible权威指南》，李松涛，机械工业出版