playbook的结构说明

playbook是由一个或多个"play"组成的列表。play的主要功能就是对一组主机应用play中定义好的task。从根本上来讲一个task就是对ansible一个module的调用。而将多个play按照一定的顺序组织到一个playbook中，我们称之为编排。

playbook主要有以下四部分构成:

- Target section： 用于定义将要执行playbook的远程主机组及远程主机组上的用户，还包括定义通过什么样的方式连接远程主机（默认ssh）
- Variable section： 定义playbook运行时需要使用的变量
- Task section： 定义将要在远程主机上执行的任务列表
- Handler section： 定义task执行完成以后需要调用的任务

Target section

playbook中的每一个play的目的都是为了让某个或某些主机以某个指定的用户身份执行任务。

Playbook中的远程用户

playbook中的远程用户和ad-hoc中的使用没有区别，默认不定义，则直接使用ansible.cfg配置中的用户相关的配置。也可在playbook中定义如下：

```
- name: /etc/hosts is up to date
  hosts: datacenter
  remote_user: automation
  become: yes
  become_mothod: sudo
  become_user: root

  tasks:
    - name: server.example.com in /etc/hosts
      lineinfile:
        path: /etc/hosts
        line: '192.168.0.200 server.exmaple.com server'
        state: present
```

Playbook中的hosts

playbook中的hosts即inentory中的定义主机与主机组，在《Ansible Inventory》中我们讲到了如何选择主机与主机组，在这里也完全适用。

```
- name: start mariadb
  hosts: db,&london
  tasks:
    - name: start mariadb
      service:
        name: mariadb
        state: started
```

Task section

play的主体部分是任务列表。

任务列表中的各任务按次序逐个在hosts中指定的所有主机上执行，在所有主机上完成第一个任务后再开始第二个。在自上而下运行某playbook时，如果中途发生错误，则整个playbook会停止执行，由于playbook的幂等性，playbook可以被反复执行，所以即使发生了错误，在修复错误后，再执行一次即可。

定义task可以使用 `action: module options` 或 `module: options` 的格式，推荐使用后者以实现向后兼容。

```
tasks:
  - name: make sure apache is running
    service:
      name: httpd
      state: started

  - name: disable selinux
    command: /sbin/setenforce 0
```

如果命令或脚本的退出码不为零可以使用如下方式替代：

```
tasks:
  - name: run this command and ignore the result
    shell: /usr/bin/somecommand || /bin/true
```

可以使用ignore_errors来忽略错误信息：

```
tasks:
  - name: run this command and ignore the result
    shell: /usr/bin/somecommand
    ignore_errors: True
```

Handler section

- 在Ansible Playbook中，handler事实上也是个task，只不过这个task默认并不执行，只有在被触发时才执行。
- handler通过notify来监视某个或者某几个task，一旦task执行结果发生变化，则触发handler，执行相应操作。
- handler会在所有的play都执行完毕之后才会执行，这样可以避免当handler监视的多个task执行结果都发生了变化之后而导致handler的重复执行（handler只需要在最后执行一次即可）。

```
tasks:
  - name: template configuration file
    template:
      src: template.j2
      dest: /etc/foo.conf
    notify:
      - restart memcached
      - restart apache
  - name: start memcached
    service:
      name: memcached
      state: started
  - name: start apache
    service
      name: httpd
      state: started
handlers:
  - name: restart memcached
    service:
      name: memcached
      state: restarted
  - name: restart apache
    service:
      name: httpd
      state: restarted
```

> 在notify中定义内容一定要和tasks中定义的 - name 内容一样，这样才能达到触发的效果，否则会不生效。

默认情况下，在一个play中，只要有task执行失败，则play终止，即使是与handler关联的task在失败的task之前运行成功了，handler也不会被执行。如果希望在这种情况下handler仍然能够执行，则需要使用如下配置：

```
- hosts: all
  force_handlers: yes
  tasks:
    - name: a task which always notifies its handler
      command: /bin/true
```

```
    notify: restart the database
  - name: a task which fails because the package doesn't exist
    yum:
      name: notapkg
      state: latest

handlers:
  - name: restart the database
    service:
      name: mariadb
      state: restarted
```

> 如果与handler关联的task还未执行，在其前的task已经失败，整个play终止，则handler未被触发，也不会执行。