> Playbook是翻译为“剧本”，即按Playbook中编写的任务在远程服务器执行。Playbook用YAML语法书写。YAML最初是Yet Another Markup Language(另一种标记语言)的缩写，后来为了强调yaml是以数据为中心重新命名为递归写法YAML Ain't Markup Language(YAML 不是一种标记语言)。

YAML语法

散列表：(又叫哈希表)，由键值对组成。

```
name: xuejinwei
age: 12
```

列表

```
- list1
- list2
- list3
```

散列表中引用列表

```
name:
    - list1
    - list2
```

列表中使用散列表

```
- name: xuejinwei
  age: 18
```

列表中使用列表项

```
-
    - list1
    - list2
```

散列表中使用散列表

```
name1:
    name2: xue
    name3: jin
    name4: wei
```

Playbook格式及结构

1. 所有主机执行完一个模块的指令后，然后所有主机执行另一个模块的指令。
2. 如果指令执行过程出错，则所有执行过程有可能回滚，但因为ansible的执行过程具有幂等性，所以重新执行即可。
3. command模块和shell模块不需要键值对书写。例如：`command: systemctl status httpd`。
4. notify可定义在任意一个paly

```
- hosts: HOST_GROUP1     # 主机Inventory，意思是一下的任务将在这些主机中执行。主机组可以是由冒号分割的多个
  vars:                  # 定义的变量
  - VAR_NAME: VAR1
  - VAR_NAME: VAR2
  remote_user: root      # 连接到远程主机时，以什么身份执行。此指令也用于每个task中
  tasks:                 # 任务列表tasks，一个tasks中可以使用多个模块，进行执行指令
  - name: Task1 Description # 用于执行结果的输出提示内容
    MODULES_NAME:
        COMMAND1: {{ VAR_NAME }}     # 引用变量
        COMMAND2: ...
        remote_user: xuekaixin
        sudo: yes        # 以sudo的方式在远程主机执行指令
        notify:          # 指定触发时要引用执行的handlers
            - Handlers1
            - Handlers2
```

```
        ignore_errors: True # 忽略错误信息，继续执行
  - name: Task2 Description
    MODULES_NAME:
        COMMAND1: ...
        COMMAND2: ...
  handlers:              # 定义处理器操作，由某个事件触发执行的操作
  - name: Handlers1      # 处理器名称
    MODULES_NAME:        # 处理器调用模块
        COMMAND1: ...
        COMMAND2: ...
  roles:
#**********************************************************************
# 可同时书写两个主机组
- hosts: HOST_GROUP2
  vars:
  - VAR_NAME: VAR1
  - VAR_NAME: VAR2
  remote_user: root
  tasks:
  - name: Task1 Description
    MODULES_NAME:
        COMMAND1: ...
        COMMAND2: ...
        remote_user: xuekaixin
        sudo: yes
  - name: Task2 Description
    MODULES_NAME:
        COMMAND1: ...
        COMMAND2: ...
  handlers:
```

Playbook-handlers

> notify可定义在任意一个paly，当notify所在play执行资源有改变，则触发handlers。

```
# 示例
- name: config
  copy: src=/root/nginx.conf
dest=/etc/nginx/nginx.conf
  notify: # 当配置文件改变时，触发handlers，执行重启操作
    - restart nginx
handlers:
  - name: restart nginx
  service: name=nginx state=restarted
```

Palybook条件测试

> 某任务执行的条件。when语句支持Jinja2表达式语法。

```
tasks:
  - name: Description
    command: ....
    when: ...
```

Playbook迭代

> 为重复执行的任务定义为迭代。参考

```
# 一次性创建多个用户
- name: add user
  user:
    name: {{ item }}
    state: present
    groups: test
  with_items:
    - test1
    - test2
```

Playbook中使用变量

facts变量

> 执行命令`ansible all -m setup`，返回远程服务器的信息，这些信息就保存在facts变量中。所有的facts变量都可以在facts中引用。

```
# 部分facts变量及其值
192.168.1.151 | SUCCESS => {
    "ansible_facts": {
        "ansible_all_ipv4_addresses": [
            "192.168.1.151"
        ],
        "ansible_all_ipv6_addresses": [
            "fe80::20c:29ff:fe8c:76d5"
        ],
        "ansible_architecture": "x86_64",
        "ansible_bios_date": "07/02/2015",
        "ansible_bios_version": "6.00",
        "ansible_cmdline": {
            "BOOT_IMAGE": "/vmlinuz-3.10.0-327.28.3.el7.x86_64",
            "LANG": "en_US.UTF-8",
            "crashkernel": "auto",
            "quiet": true,
            "rhgb": true,
            "ro": true,
            "root": "UUID=bcf52617-e5d3-473a-acb9-e409842e5763"
        },
```

命令行传递变量

> 在执行playbook时，在命令行将变量传递给playbook，playbook中直接引用变量。

```
# 示例
ansible-playbook file.yml --extra-vars "file1=test1 file2=test2"
```

Inventory变量

> 默认的inventory文件的/etc/ansible/hosts文件。hosts文件的定义批量管理的主机组，在定义主机组时可以定义变量以传递给playbook使用。

```
# 单个主机示例
[web]
192.168.1.151 port=8080 ServerName=localhost
192.168.1.152
192.168.1.153
```

```
# 组变量,组变量在定义的主机组中生效
[web:vars]
var=Require
```

在文件模板中使用变量

> ansible中template模板中可以使用变量，可以facts变量，也可以使用inventory里面定义的变量。可以在httpd.conf中使用实现定义好或者内置变量。

```
# 示例
- name: Write the configuration file
  template: src=templates/httpd.conf dest=/etc/httpd/conf/httpd.conf
```

Playbook-tags

> 可以有选择的执行Playbook中的部分任务。

```
# 示例
- name: Write the configuration file
  template: src=templates/httpd.conf
dest=/etc/httpd/conf/httpd.conf
  tags:
  - conf
# - always 特殊tags，表示无论调用哪一个tags，都执行此任务
```

```
# 调用
ansible-playbook test.yml --tags="conf"
```

Playbook-roles

> roles是ansible1.2版本之后引入的新特性。可以层次性、结构性的书写Playbook。roles通过将变量、任务、模块、处理器文件防止于单独的目录文件内，然后在Playbook中使用加载上述元素。roles的好处是可以按功能书写Playbook，例如web角色的，只执行安装web服务，mysql角色，只执行安装mysql服务，而么一个角色都可以重复调用。
>
> 1. 在roles目录下创建以角色(功能)命名的目录。如apache_server、php_server
> 2. 在每个角色命名的目录中分别创建files、handlers、meta、tasks、templates和vars目录
> 3. 参考roles:https://galaxy.ansible.com/list#/roles?page=1&page_size=10
> 4. 安装下载`roles：ansible-galaxy install geerlingguy.apache`

```
# roules目录结构
bennojoy.nginx/
├── defaults          # 当前角色设定默认变量时使用此目录
|   └── main.yml
├── files             # 存放模块使用到的文件，如copy、file模块使用到的文件
```

```
|   └── epel.repo
├── handlers            # 用于定义此角色用到的各handler
|   └── main.yml
├── meta                # 于定义此角色的特殊设定及其依赖关系
|   └── main.yml
├── README.md
├── tasks               # 至少应该包含一个名为main.yml的文件，其定义了此角色的任务列表
|   └── main.yml
├── templates           # template模块会自动在此目录中寻找Jinja2模板文件
|   ├── default.conf.j2
|   ├── default.j2
|   ├── nginx.conf.j2
|   └── site.j2
└── vars                # 于定义此角色用到的变量
    └── main.yml
```

> 在角色目录外定义一个`site.yml`的文件，进行执行角色任务。

```
- hosts: web
  remote_user: root
  roles:
  - bennojoy.nginx
```