运行前检查

1. 当我们在运行ansible-playbook时，使用--check选项时，将不会对受控主机作出任何更改，而是通过模拟运行的方式执行所有task，以用于检查playbook在运行时的状态：

```
ansible-playbook foo.yml --check
```

2. 在运行ansible-playbook时，如果使用--diff选项配合--check选项，可以用于检查本次执行play时，相较上一次产生了哪些改变：

```
ansible-playbook foo.yml  --check --diff --limit foo.example.com
```

3. 有些时候，我们在检测模式下运行play时，我们会希望某个play总是运行，我们可以使用always_run子句：

```
tasks:
  - name: this task is run even in check mode
    command: /something/to/run --even-in-check-mode
    always_run: yes
```

> 需要说明的是，如果一个task中同时包含when和always_run，如果when返回了false，即使alwys_run为true，任务依然会被跳过。

打印详细输出信息

参考《6. Ansible Playbook基本使用》

使用debug模块

在前面debug模块使用的比较多了，这里直接再给个示例：

```
tasks:
  - debug:
      var: myvariable
  - debug:
      msg: "The value of myvariable is {{ var }}"
```

使用assert模块

assert模块会在指定的条件不符合的时候返回错误并失败退出。

```
# 当目标机没有eth1网卡时则playbook会返回失败
- name: assert that eth1 interface exists
  assert:
    that: ansible_eth1 is defined
```

当调试playbook的时候，插入assert模块在我们设定的某些条件不成立时立刻失败，对调试很有用。

下面示例用于检查目标文件是否是一个目录，如果不是，则失败退出：

```
- name: stat /opt/foo
  stat: path=/opt/foo
  register: st

- name: assert that /opt/foo is a directory
  assert:
    that: st.stat.isdir
```

限制特定的task运行

指定任务执行

可以通过--start-at-task参数告诉Ansible从指定的task开始运行playbook，而不是从头开始运行。如果你的playbook因为某一个task中有bug而失败了，在你修复了这个bug后希望从被修复的这个task开始再次执行playbook的时候，就可以使用这个参数。

```
# 以下命令会从名为"install packages"的任务开始执行playbook

ansible-playbook playbook.yml --start-at-task="install packages"
```

分步执行

可以通过 `--step` 选项来交互式的执行playbook：

```
ansible-playbook playbook.yml --step
```

这样ansible在每个任务前会自动停止,并询问是否应该执行该任务。 假如有一个名为"configure ssh"的任务,playbook执行到这里会停止并询问:

```
Perform task: configure ssh (y/n/c):
```

- "y"会执行该任务
- "n"会跳过该任务
- "c"则会继续执行剩余的所有任务而不再询问

tags

参考《17. Ansible Playbook之tags》

智慧能力中心/运维部 AMT332黄佳伟