简单说明

ansible支持变量，用于存储会在整个项目中重复使用到的一些值。以简化项目的创建与维护，降低出错的机率。

变量的定义：

- 变量名应该由字母、数字下划数组成
- 变量名必须以字母开头
- ansible内置关键字不能作为变量名

1. 在Inventory中定义变量

1.1. 定义主机变量

1.1.1. 内置主机变量

所谓内置变量其实就是ansible.cfg配置文件中的选项，在其前加上 `ansible_` 即成为内置变量。当然内置变拥有比ansible.cfg中选项更高的优先级，而且针对不同的主机，可以定义不同的值。

```
# 一般连接

ansible_host      #用于指定被管理的主机的真实IP
```

```
ansible_port      #用于指定连接到被管理主机的ssh端口号，默认是22
ansible_user      #ssh连接时默认使用的用户名


# 特定ssh连接
ansible_connection      #SSH连接的类型：local，ssh，paramiko，在ansible 1.2 之前默认是
paramiko，后来智能选择，优先使用基于ControlPersist的ssh（如果支持的话）

ansible_ssh_pass      #ssh连接时的密码
ansible_ssh_private_key_file  #秘钥文件路径，如果不想使用ssh-agent管理秘钥文件时可以使用此选项

ansible_ssh_executable  #如果ssh指令不在默认路径当中，可以使用该变量来定义其路径


# 特权升级
ansible_become  #相当于ansible_sudo或者ansible_su，允许强制特权升级
ansible_become_user #通过特权升级到的用户，相当于ansible_sudo_user或者ansible_su_user
ansible_become_pass  # 提升特权时，如果需要密码的话，可以通过该变量指定，相当于
ansible_sudo_pass或者ansible_su_pass

ansible_sudo_exec      #如果sudo命令不在默认路径，需要指定sudo命令路径


# 远程主机环境参数

ansible_shell_executable # 设置目标机上使用的shell，默认为/bin/sh

ansible_python_interpreter      #用来指定python解释器的路径，默认为/usr/bin/python 同样可以指
定ruby 、perl 的路径
ansible_*_interpreter      #其他解释器路径，用法与ansible_python_interpreter类似，这里"*"可以
是ruby或才perl等其他语言
```

下面是一个简单的示例：

```
# 指定了三台主机，三台主机的用密码分别是P@ssw0rd、123456、45789，指定的ssh连接的用户名分别为
root、breeze、bernie，ssh 端口分别为22、22、3055 ，这样在ansible命令执行的时候就不用再指令用户
和密码等了

[test]
192.168.1.1 ansible_ssh_user=root ansible_ssh_pass='P@ssw0rd'
192.168.1.2 ansible_ssh_user=breeze ansible_ssh_pass='123456'
192.168.1.3 ansible_ssh_user=bernie ansible_ssh_port=3055 ansible_ssh_pass='456789'
```

1.2. 定义主机组变量

变量也可以通过组名，应用到组内的所有成员：

```
# test组中包含两台主机，通过对test组指定vars变更，相应的host1和host2相当于相应的指定了
ntp_server和proxy变量参数值
```

```
[test]
host1
host2
[test:vars]
ntp_server=192.168.1.10
proxy=192.168.1.20
```

主机组变量示例：

```
# 下面是一个示例，指定了一个武汉组有web1、web2；随州组有web3、web4主机；又指定了一个湖北组，同时包含武汉和随州；同时为该组内的所有主机指定了2个vars变量。设定了一个组中国组，包含湖北、湖南。

[wuhan]
web1
web2

[suizhou]
web4
web3

[hubei:children]
wuhan
suizhou

[hubei:vars]
ntp_server=192.168.1.10
zabbix_server=192.168.1.10
```

2. 在Playbook中定义变量

2.1. 变量的定义方式

2.1.1. 通过vars关键字定义

下面是一个简单示例：

```
- name:  use vars define invrionmemnt
  hosts: test
  user: ansible
  vars:
    http_port: 80
    server_name: localhost
    conf_file: /etc/nginx/conf/default.conf
```

2.1.2. 通过vars_files关键字引入变量文件

下面是一个简单示例：

```
- hosts: all
  remote_user: root
  vars:
    favcolor: blue
  vars_files:
    - vars/external_vars.yml
    - vars/user_vars.yml

# vars/user_vars.yml示例:

users:
  bjones:
    first_name: Bob
    last_name: Jones
    home_dirs: /users/bjones
  acook:
    first_name: Anne
    last_name: Cook
    home_dirs: /users/acook
```

> 变量的定义格式是成键值对出现的，键值对之间可以嵌套，最终形成一个大字典

2.1.3. 在playbook中通过host_vars和group_vars目录定义变量

下面这是一个项目的playbook目录结构。这个项目中，包含一个ansible.cfg文件，一个inventory文件，一个playbook.yml文件，一个 `group_vars` 目录和一个 `host_vars` 目录:

```
# tree /etc/ansible/playbooks/project
/etc/ansible/playbooks/project
├── ansible.cfg
├── group_vars
│   ├── datacenter1
│   ├── datacenter2
│   └── datacenters
├── host_vars
│   ├── demo1.example.com
│   ├── demo2.example.com
│   ├── demo3.example.com
│   └── demo4.example.com
├── inventory
└── playbook.yml
```

其中inventory文件的示例如下：

```
[datacenter1]
demo1.example.com
demo2.example.com
```

```
[datacenter2]
demo3.example.com
demo4.example.com

[datacenters:children]
datacenter1
datacenter2
```

可以看到group_vars目录中，定义了三个文件，分别以inventory文件中的三个主机组命名，所以这三个文件中定义的就分别是这三个组可以使用的变量。

```
# cat datacenter1
package: httpd

# cat datacenter2
package: apache

# cat datacenters
package: httpd
```

在host_vars目录中，定义了三个文件，分别以inventory文件中的四个主机命名，所以这四个文件中定义的就分别是这四个主机可以使用的变量。

```
# cat demo1.example.com
package: httpd

# cat demo2.example.com
package: apache

# cat demo3.example.com
package: mariadb-server

# cat demo4.example.com
package: mysql-server
```

> 需要说明的是，如果主机组定义的变量与主机冲突，主机变量优先级最高

2.1.4. 注册变量

在有些时候，可能需要将某一条任务执行的结果保存下来，以便在接下的任务中调用或者做些判断。可以通过register关键字来实现将某一任务结果保存为一个变量。

下面是个简单的例子，将whoami命令执行的结果注册到变量login：

```
- name: register variables
  hosts: test
  tasks:
    - name: capture output of whoami command
      command: whoami
      register: login
```

注册变量的应用场景：

- 在一台远端的服务器获取一个目录下的一列表的文件,然后下载这些文件
- 在handler执行之前,发现前面一个task发生了changed,然后执行一个指定的task
- 获取远端服务器的ssh key的内容,构建出known_hosts文件

2.1.5. 通过命令行设置变量

示例如下：

```
---
- hosts: '{{ hosts }}'
  remote_user: '{{ user }}'
  tasks:
     - ...

ansible-playbook release.yml --extra-vars "hosts=vipers user=starbuck"
```

也可以写成类似如下方式：

```
--extra-vars '{"hosts":"vipers","user":"starbuck"}'
```

2.2 使用与调试变量

我们通过以上5种方式在playbook中定义了各种变量。说到底，最终的目的，还是为了方便使用。下面我们就看一看具体如何使用这些变量。

2.2.1. 变量的引用

下面是一个变量的基本使用示例，前面的变量定义部分，直接使用的2.1.1中的变量示例：

```
- name: use vars define variables
  hosts: test
  vars:
    http_port: 80
    server_name: localhost
    conf_file: /etc/nginx/conf/default.conf

  tasks:
    - name: print variables
      shell: echo "{{ http_port }} {{ server_name }} {{ conf_file }}"  > /tmp/text.txt
```

在2.1.2中，我们通过 `vars_files` 引用了一个文件 `user_vars.yml` ，在该文件中定义了一个稍微复杂的字典变量，现在我想要获取 `users` 中 `bjones` 用户的 `first_name` 和 `acook` 用户的 `home_dirs` ，可以

使用如下方法：

```
{{ users.bjones.first_name }}
{{ users.acook.home_dirs }}
```

或者如下写法：

```
{{ users['bjones']['first_name'] }}
{{ users['acook']['home_dirs'] }}
```

2.2.2. 变量的调试输出

有些时候，我们在引用变量的时候，可能需要知道变量中包含哪些信息，以方便在执行过程中，对变量做些处理。ansible提供一个debug模块用于调试变量输出：

```
- name: register variables
  hosts: test
  tasks:
    - name: capture output of whoami command
      command: whoami
      register: login
    - debug: var=login
```

执行后输出如下：

```
# ansible-playbook register.yml

PLAY [register variables]
*********************************************************************************************
*************************************************************

TASK [Gathering Facts]
*********************************************************************************************
*************************************************************
ok: [10.1.61.187]

TASK [capture output of whoami command]
*********************************************************************************************
*****************************************
changed: [10.1.61.187]

TASK [debug]
*********************************************************************************************
*********************************************************************
ok: [10.1.61.187] => {
    "login": {
        "changed": true,
        "cmd": [
```

```
            "whoami"
        ],
        "delta": "0:00:00.004279",
        "end": "2019-05-24 00:41:43.710398",
        "failed": false,
        "rc": 0,
        "start": "2019-05-24 00:41:43.706119",
        "stderr": "",
        "stderr_lines": [],
        "stdout": "root",
        "stdout_lines": [
            "root"
        ]
    }
}

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

关于输出的debug部分重点说明如下：

- login： 变量名，其值为一个字典
- changed：ansible基于此来判断是否发生了状态改变
- cmd：被调用的命令
- failed：是否运行失败
- rc：返回值，0代表正常，非0代表异常
- stderr：如果出现异常，会在这里显示错误输出
- stderr_lines：按行分割的错误输出
- stdout：如果指令正常运行，则在这里输出返回结果
- stdout：按行分割的返回结果

> 需要说明的是，通过register注册的变量的结果并不是一成不变的，在不确定返回值的情况下，尽量调试看看输出结果。

关于debug的更多用法说明：

调试模块，用于在调试中输出信息
常用参数：

- msg：调试输出的消息
- var：将某个变量传递给debug模块，debug会直接将其打印输出
- verbosity：debug的级别

示例：

```
# Example that prints the loopback address and gateway for each host
- debug: msg="System {{ inventory_hostname }} has uuid {{ ansible_product_uuid }}"
```

```
- debug: msg="System {{ inventory_hostname }} has gateway {{ ansible_default_ipv4.gateway }}"
  when: ansible_default_ipv4.gateway is defined

- shell: /usr/bin/uptime
  register: result

- debug: var=result verbosity=2    #直接将上一条指令的结果作为变量传递给var，由debug打印出result的值

- name: Display all variables/facts known for a host
  debug: var=hostvars[inventory_hostname] verbosity=4
```