第3天-自动化运维利器 Ansible-变量

一、 Ansible 变量介绍

我们在PlayBook一节中，将PlayBook类比成了Linux中的shell。

那么它作为一门Ansible特殊的语言，肯定要涉及到变量定义、控制结构的使用等特性。

在这一节中主要讨论变量的定义和使用。

二、变量命名规则

变量的名字由字母、下划线和数字组成，必须以字母开头。

如下变量命名为正确:

```
good_a
ok_b
```

如下变量命名为错误:

```
_aaa
2_bb
```

保留关键字不能作为变量名称

```
add, append, as_integer_ratio, bit_length,
capitalize, center, clear,
```

```
conjugate, copy, count, decode, denominator, difference,
difference_update, discard, encode, endswith, expandtabs,
extend, find, format, fromhex, fromkeys, get, has_key,
hex, imag, index, insert, isalnum, intersection,
intersection_update, isalpha, isdecimal, isdigit, isdisjoint, is_integer, islower,
isnumeric, isspace, issubset, issuperset, istitle, isupper,
items, iteritems, iterkeys, itervalues, join, keys, ljust, lower,
 lstrip, numerator, partition, pop, popitem, real, remove,
 replace, reverse, rfind, rindex, rjust, rpartition, rsplit, rstrip,
 setdefault, sort, split, splitlines, startswith, strip, swapcase,
symmetric_difference, symmetric_difference_update, title,
translate, union, update, upper, values, viewitems, viewkeys,
viewvalues, zfill
```

三、变量类型

根据变量的作用范围大体的将变量分为:

- 全局变量
- 剧本变量
- 资产变量

但只是一个比较粗糙的划分，不能囊括Ansible 中的所有变量。下面将分别从这三种变量入手，去介绍变量的使用

1. 全局变量

全局变量，是我们使用ansible 或使用ansible-playbook 时，手动通过 -e 参数传递给Ansible 的变量。

通过ansible 或 ansible-playbook 的 help 帮助, 可以获取具体格式使用方式:

```
# ansible -h |grep var
  -e EXTRA_VARS, --extra-vars=EXTRA_VARS
                        set additional variables as
key=value or YAML/JSON

# ansible-playbook  -h |grep var
  -e EXTRA_VARS, --extra-vars=EXTRA_VARS
                        set additional variables as
key=value or YAML/JSON
```


Example

传递普通的key=value 的形式

```
# ansible all -i localhost, -m debug -a "msg='my key
is {{ key }}'" -e "key=value"
```

传递一个YAML/JSON 的形式(注意不管是YAML还是JSON，它们的最终格式一定要是一个字典)

```
# cat a.json
{"name":"qfedu","type":"school"}
```

```
# ansible all -i localhost, -m debug -a "msg='name is {{ name  }}, type is {{ type }}'" -e @a.json
```

```
# cat a.yml
---
name: qfedu
type: school
...
```

```
# ansible all -i localhost, -m debug -a "msg='name is {{ name  }}, type is {{ type }}'" -e @a.yml
```

2. 剧本变量

此种变量和PlayBook 有关，定义在PlayBook中的。它的定义方式有多种，我们这里介绍两种最常用的定义方式。

通过PLAY 属性 vars 定义

```
---
- name: test play vars
  hosts: all
  vars:
    user: lilei
    home: /home/lilei
```

通过PLAY 属性 vars_files 定义

```yaml
# 当通过vars属性定义的变量很多时，这个Play就会感觉特别臃肿。此时我们可以将变量单独从Play中抽离出来，
# 形成单独的YAML 文件。
---
- name: test play vars
  hosts: all
  vars_files:
    - vars/users.yml
```

```yaml
# cat vars/users.yml
---
user: lilei
home: /home/lilei
```

如何在PlayBook中使用这些变量

在PlayBook中使用变量时，使用 **{{ 变量名 }}** 来使用变量

```yaml
---
- name: test play vars
  hosts: all
  vars:
    user: lilei
    home: /home/lilei
  tasks:
    - name: create the user {{ user }}
      user:
        name: "{{ user }}"
        home: "{{ home }}"
```

在PlayBook中使用变量的注意点

```yaml
---
# 这里我们将上面的Playbook中引用变量的部分进行修改，去掉了双引号。
- name: test play vars
  hosts: all
  vars:
    user: lilei
    home: /home/lilei
  tasks:
    - name: create the user {{ user }}
      user:
        # 注意这里将 "{{ user }}" 改成了 {{ user }}
        name: {{ user }}
        home: "{{ home }}"
```

执行以上的PlayBook 时，会出现以下错误

```
The offending line appears to be:

      user:
        name: {{ user }}
                ^ here
We could be wrong, but this one looks like it might be an issue with
missing quotes.  Always quote template expression brackets when they
start a value. For instance:

    with_items:
      - {{ foo }}

Should be written as:

    with_items:
```

```
16          - "{{ foo }}"
```

```
1  这样错误的主要原因是PlayBook 是YAML 的文件格式， 当Ansible
   分析YAML 文件时，有可能会误认为字典。
2  name: {{ user }} 是一个字典的开始。因此加针对变量的使用，加
   上了双引号，避免Ansible错误解析。
```

3. 资产变量

在之前的课程中学习了资产。资产共分为静态资产和动态资产。

这一节中学习的资产变量，就是和资产紧密相关的一种变量。

资产变量分为主机变量和主机组变量，分别针对资产中的单个主机和主机组。

3.1 主机变量

以下资产中，定义了一个主机变量 lilei ， 此变量只针对 172.18.0.3 这台服务器有效。


```
1  # cat hostsandhostvars
2  [webservers]
3  172.18.0.3 user=lilei port=3309
4  172.18.0.4
```

验证

```
// 获取定义的变量值
# ansible 172.18.0.3  -i hostsandhostvars  -m debug -a "msg='{{user}} {{port}}'"
172.18.0.3 | SUCCESS => {
    "user": "lilei"
}

// 未获取到定义的变量值，因为 user 这个变量针对 172.18.0.4 主机无效。
# ansible 172.18.0.4  -i hostsandhostvars  -m debug -a "var=user"
172.18.0.4 | SUCCESS => {
    "user": "VARIABLE IS NOT DEFINED!"
}
```

3.2 主机组变量

以下资产中，定义了一个组变量home，此变量将针对webservers 这个主机组中的所有服务器有效

```
# cat hostsandgroupvars
[webservers]
172.18.0.3 user=lilei
172.18.0.4

[webservers:vars]
home="/home/lilei"
```

验证

```
// home 是 web_servers 的组变量，会针对这个组内的所有服务器生效。
# ansible webservers  -i hostsandgroupvars  -m debug -a "var=home"
172.18.0.3 | SUCCESS => {
    "home": "/home/lilei"
}
172.18.0.4 | SUCCESS => {
    "home": "/home/lilei"
}
```

3.3 主机变量 VS 主机组变量

当主机变量和组变量在同一个资产中发生重名的情况，会有什么效果呢?

```
# cat hosts_v2
[webservers]
172.18.0.3 user=lilei
172.18.0.4

[webservers:vars]
user=tom
```


验证

```
// 在资产中定义了主机变量和组变量 user，此时发现 172.18.0.3 这台机器的主机变量 user 的优先级更高。
# ansible webservers  -i hosts_v2  -m debug -a "var=user"
172.18.0.3 | SUCCESS => {
    "user": "lilei"
}
172.18.0.4 | SUCCESS => {
    "user": "tom"
}
```

3.4 变量的继承

在介绍资产时说过资产的继承，那么变量是否也存在继承关系呢?

```
# cat hosts_v3
[webservers]
172.18.0.3

[dbservers]
172.18.0.4

[allservers]
[allservers:children]
dbservers
webservers

[allservers:vars]
user=lilei
```


验证

```
// 在资产继承的同时，对应的变量也发生了继承
```

```
# ansible allservers  -i hosts_v3  -m debug -a "var=user"
172.18.0.4 | SUCCESS => {
    "user": "lilei"
}
172.18.0.3 | SUCCESS => {
    "user": "lilei"
}
# ansible dbservers  -i hosts_v3  -m debug -a "var=user"
172.18.0.4 | SUCCESS => {
    "user": "lilei"
}
# ansible webservers  -i hosts_v3  -m debug -a "var=user"
172.18.0.3 | SUCCESS => {
    "user": "lilei"
}
```


3.5 Inventory 内置变量的说明

内置变量几乎都是以 `ansible_` 为前缀。

```
ansible_ssh_host
      将要连接的远程主机名与你想要设定的主机的别名不同的话,可通过此变量设置.

ansible_ssh_port
      ssh端口号.如果不是默认的端口号,通过此变量设置.

ansible_ssh_user
      默认的 ssh 用户名

ansible_ssh_pass
```

```
11        ssh 密码(这种方式并不安全,官方强烈建议使用 --ask-
    pass 或 SSH 密钥)
12
13  ansible_sudo_pass
14        sudo 密码(这种方式并不安全,官方强烈建议使用 --ask-
    sudo-pass)
15
16  ansible_sudo_exe (new in version 1.8)
17        sudo 命令路径(适用于1.8及以上版本)
18
19  ansible_ssh_private_key_file
20        ssh 使用的私钥文件.适用于有多个密钥,而你不想使用 SSH
    代理的情况.
21
22  ansible_python_interpreter
23        目标主机的 python 路径.适用于的情况: 系统中有多个
    Python, 或者命令路径不是"/usr/bin/python",比如
    /usr/local/bin/python3
```


4. Facts变量

Facts变量不包含在前文中介绍的全局变量、剧本变量及资产变量之内。

Facts变量不需要我们人为去声明变量名及赋值。

它的声明和赋值完全有Ansible 中的 setup 模块帮我们完成。

它收集了有关被管理服务器的操作系统版本、服务器IP地址、主机名，磁盘的使用情况、CPU个数、内存大小等等有关被管理服务器的私有信息。

在每次PlayBook运行的时候都会发现在PlayBook执行前都会有一个Gathering Facts的过程。这个过程就是收集被管理服务器的Facts信息过程。

4.1 手动收集Facts 变量

```
# ansible all -i localhost, -c local -m setup
localhost | SUCCESS => {
    "ansible_facts": {
        "ansible_all_ipv4_addresses": [
            "192.168.122.130"
        ],
        "ansible_all_ipv6_addresses": [
            "fe80::20c:29ff:fede:b5b"
        ],
        "ansible_apparmor": {
            "status": "disabled"
        },
        "ansible_architecture": "x86_64",
        "ansible_bios_date": "07/02/2015",
        "ansible_bios_version": "6.00",
        "ansible_cmdline": {
            "KEYBOARDTYPE": "pc",
            "KEYTABLE": "us",
            "LANG": "en_US.UTF-8",
            "SYSFONT": "latarcyrheb-sun16",
            "nomodeset": true,
            "quiet": true,
            "rd_LVM_LV": "vg_mouse00/lv_root",
            "rd_NO_DM": true,
            "rd_NO_LUKS": true,
            "rd_NO_MD": true,
            "rhgb": true,
            "ro": true,
            "root": "/dev/mapper/vg_mouse00-lv_root"
        },
    ...
    ...
```

```
    ...
```

4.2 过滤Facts

通过刚刚的手动收集Facts，我们发现facts 信息量很大。 能不能有针对性的显示我们想要的信息呢?

可以通过使用Facts 模块中的filter参数去过滤我们想要的信息。

- 仅获取服务器的内存情况信息

```
# ansible all -i localhost, -m setup -a "filter=*memory*" -c local
localhost | SUCCESS => {
    "ansible_facts": {
        "ansible_memory_mb": {
            "nocache": {
                "free": 508,
                "used": 473
            },
            "real": {
                "free": 59,
                "total": 981,
                "used": 922
            },
            "swap": {
                "cached": 0,
                "free": 1981,
                "total": 1983,
                "used": 2
            }
        }
    },
    "changed": false
}
```

- 仅获取服务器的磁盘挂载情况

```
# ansible all -i localhost, -m setup -a "filter=*mount*" -c local
localhost | SUCCESS => {
    "ansible_facts": {
        "ansible_mounts": [
            {
                "device": "/dev/mapper/vg_mouse00-lv_root",
                "fstype": "ext4",
                "mount": "/",
                "options": "rw",
                "size_available": 5795786752,
                "size_total": 18435350528,
                "uuid": "N/A"
            },
            {
                "device": "/dev/sda1",
                "fstype": "ext4",
                "mount": "/boot",
                "options": "rw",
                "size_available": 442216448,
                "size_total": 499355648,
                "uuid": "N/A"
            }
        ]
    },
    "changed": false
}
```


4.3 在PlayBook中去使用Facts 变量

默认情况下，在执行PlayBook的时候，它会去自动的获取每台被管理服务器的facts信息。

```
---
- name: a play example
  hosts: all
  remote_user: root
  tasks:
    - name: install nginx package
      yum: name=nginx state=present
    - name: copy nginx.conf to remote server
      copy: src=nginx.conf
dest=/etc/nginx/nginx.conf
    - name: start nginx server
      service:
        name: nginx
        enabled: true
        state: started
```

执行

```
# ansible-playbook   myplaybook.yml

PLAY [a play example] *********************************************************
*********************************************************
*********************************************************
***
# 执行PLAYBOOK时，自动收集facts 信息
TASK [Gathering Facts] *********************************************************
*********************************************************
*********************************************************
**
ok: [172.18.0.4]
ok: [172.18.0.3]

TASK [install nginx package] *********************************************************
*********************************************************
*********************************************
ok: [172.18.0.3]
ok: [172.18.0.4]
......
......
```


可以像使用其他变量一样，去使用facts 变量

```
---
- name: print facts variable
  hosts: all
  tasks:
   - name: print facts variable
     debug:
       msg: "The default IPV4 address is {{
ansible_default_ipv4.address }}"
```

在PlayBook中去关闭Facts 变量的获取

若在整个PlayBook 的执行过程中，完全未使用过 Facts 变量，此时我们可以将其关闭，以加快PlayBook的执行速度。

```
---
- name: a play example
  hosts: webservers
  # 关闭 facts 变量收集功能
  gather_facts: no
  remote_user: root
  tasks:
    - name: install nginx package
      yum: name=nginx state=present
    - name: copy nginx.conf to remote server
      copy: src=nginx.conf
dest=/etc/nginx/nginx.conf
    - name: start nginx server
      service:
        name: nginx
        enabled: true
        state: started
```


执行

```
# ansible-playbook -i hosts  myplaybook2.yml
```

```
2
3  PLAY [a play example]
   ****************************************************
   ******
4
5  TASK [install nginx package]
   ****************************************************
6  ok: [172.18.0.4]
7
8  TASK [copy nginx.conf to remote server]
   ****************************************
9  ok: [172.18.0.4]
10
11 TASK [start nginx server]
   ****************************************************
   **
12 ok: [172.18.0.4]
13
14 PLAY RECAP
   ****************************************************
   ****************
15 172.18.0.4                 : ok=3    changed=0
    unreachable=0    failed=0    skipped=0    rescued=0
      ignored=0
```


5. 注册变量

往往用于保存一个task任务的执行结果, 以便于debug时使用。

或者将此次task任务的结果作为条件，去判断是否去执行其他task任务。

注册变量在PlayBook中通过`register`关键字去实现。

```yaml
---
- name: install a package and print the result
  hosts: webservers
  remote_user: root
  tasks:
    - name: install nginx package
      yum: name=nginx state=present
      register: install_result
    - name: print result
      debug: var=install_result
```

执行

```
# ansible-playbook  myplaybook3.yml

PLAY [install a package and print the result] ***********************************

TASK [Gathering Facts] ***************************************************************
ok: [172.18.0.4]

TASK [install nginx package] *********************************************************
ok: [172.18.0.4]

TASK [print result] ******************************************************************
ok: [172.18.0.4] => {
    "install_result": {
        "changed": false,
        "failed": false,
```

```
        "msg": "",
        "rc": 0,
        "results": [
            "1:nginx-1.16.1-1.el7.ngx.x86_64 providing nginx is already installed"
        ]
    }
}

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

6. 变量优先级

目前介绍了全局变量、剧本变量、资产变量、Facts变量及注册变量。

其中Facts变量不需要人为去声明、赋值；注册变量只需通过关键字register去声明，而不需要赋值。

而全局变量、剧本变量及资产变量则完全需要人为的去声明、赋值。

变量的优先权讨论，也将着重从这三类变量去分析。

假如在使用过程中，我们同时在全局变量、剧本变量及资产变量声明了同一个变量名，那么哪一个优先级最高呢? 下面我们将以实验的形式去验证变量的优先级

环境准备

1、定义一份资产、且定义了资产变量user

```
[dbservers]
172.18.0.3

[webservers]
172.18.0.4  ansible_ssh_port=2222

[allservers:children]
dbservers
webservers

[allservers:vars]
user=tomcat
```

2、编写一份PlayBook、同样定义剧本变量user

```
---
- name: test variable priority
  hosts: all
  remote_user: root
  vars:
    user: mysql
  tasks:
    - name: print the user value
      debug: msg='the user value is {{ user }}'
```


验证测试

同时使用全局变量、剧本变量、资产变量

当变量user同时定义在全局变量、剧本变量及资产变量中时，全局变量的优先级最高。

```
# ansible-playbook -i hosts priority.yml -e "user=www"

```

```
PLAY [a play example] *********************************************************
*********************************************************************
*********************************************************************
***

TASK [Gathering Facts] ********************************************************
*********************************************************************
*********************************************************************
**
ok: [172.18.0.4]
ok: [172.18.0.3]

// 打印的 user 值 为www , 全局变量生效
TASK [print the user value] ***************************************************
*********************************************************************
*********************************************************************
*************************************************
ok: [172.18.0.3] => {
    "msg": "the user value is www"
}
ok: [172.18.0.4] => {
    "msg": "the user value is www"
}

PLAY RECAP ********************************************************************
*********************************************************************
*********************************************************************
**************
172.18.0.3                 : ok=2    changed=0
 unreachable=0    failed=0
172.18.0.4                 : ok=2    changed=0
 unreachable=0    failed=0
```

同时使用剧本变量和资产变量

取消全局变量，发现剧本变量的优先级要高于资产变量的优先级。

```
# ansible-playbook -i hosts priority.yml

PLAY [a play example] ***********************************************************************************************************************************

TASK [Gathering Facts] **********************************************************************************************************************************
ok: [172.18.0.3]
ok: [172.18.0.4]
// 打印的 user 值 为mysql , 剧本变量生效
TASK [print the user value] *****************************************************************************************************************************
ok: [172.18.0.3] => {
    "msg": "the user value is mysql"
}
ok: [172.18.0.4] => {
    "msg": "the user value is mysql"
}

```

```
PLAY RECAP *********************************************************************
172.18.0.3                 : ok=2    changed=0    unreachable=0    failed=0
172.18.0.4                 : ok=2    changed=0    unreachable=0    failed=0
```

只是用资产变量的情况

不使用全局变量、且注释掉剧本变量后，资产变量才最终生效。

```
---
- name: test variable priority
  hosts: all
  remote_user: root
  #vars:
  #  user: mysql
  tasks:
    - name: print the user value
      debug: msg='the user value is {{ user }}'
```

```
# ansible-playbook -i hosts priority.yml

PLAY [a play example] **********************************************************

```

```
TASK [Gathering Facts]
*********************************************************
*********************************************************
*********************************************************
**
ok: [172.18.0.3]
ok: [172.18.0.4]

// 打印的 user 值 为tomcat , 资产变量生效。
TASK [print the user value]
*********************************************************
*********************************************************
*************************************************
ok: [172.18.0.3] => {
    "msg": "the user value is tomcat"
}
ok: [172.18.0.4] => {
    "msg": "the user value is tomcat"
}

PLAY RECAP
*********************************************************
*********************************************************
*********************************************************
**************
172.18.0.3                 : ok=2    changed=0
 unreachable=0    failed=0
172.18.0.4                 : ok=2    changed=0
 unreachable=0    failed=0
```


变量优先级结论

当一个变量同时在全局变量、剧本变量和资产变量中定义时，优先级最高的是全局变量；其次是剧本变量；最后才是资产变量。