04.Ansible变量解析

1.变量概述

变量提供了便捷的方式来管理ansible项目中的动态值。 比如zabbix-3.4.15，可能后期会反复的使用到这个版本的值，那么如果将此值设置为变量，后续使用和修改都将变得非常方便。这样可以简化项目的创建和维护

定义变量分为如下三种方式
*1) 通过**命令行**进行变量定义*
*2) 在**play文件**中进行定义变量*
*3) 通过**inventory**在主机组或单个主机中设置变量*

如果定义的变量出现重复，且造成冲突，优先级如下:
1.命令行定义的变量-高于->play文件定义的变量-高于->inventory文件定义的变量。

2.变量定义

ansible变量使用权威指南 传送门

key: value 形式

键(变量): 值(内容)对形式

1、playbook变量可以通过多种方式进行定义，最简单的方式就是在playbook的开头通过vars进行定义

```
[root@m01 /server/playbook]# cat 04_var.yml
- hosts: lb
  vars:
    - soft01: nc
    - soft02: nmap
  tasks:
    - name: Install Software nc  nmap
      yum:
        name:
        - "{{ soft01 }}"
        - "{{ soft02 }}"
        state: present
```

```
#安装两个软件包使用变量方式

[root@m01 project1]# cat  p2.yml
- hosts: webservers
  vars:
    - web_package: httpd
    - ftp_package: vsftpd
  tasks:
    - name: Installed Packages
      yum:
        name:
          - "{{ web_package }}"
          - "{{ ftp_package }}"
        state: present
```

*2.也可以在playbook中使用**vars_files**指定文件作为变量文件，好处就是其他的playbook也可以调用*

```
[root@m01 /server/playbook]# cat vars.yml
soft01: nc
soft02: nmap
[root@m01 /server/playbook]# cat 04_var.yml
```

```
- hosts: lb
  vars_files: ./vars.yml
  tasks:
    - name: Install Software nc  nmap
      yum:
        name:
        - "{{ soft01 }}"
        - "{{ soft02 }}"
        state: present
```

```
[root@m01 project1]# cat vars.yml
web_package: httpd
ftp_package: vsftpd

[root@m01 project1]# cat p2.yml
- hosts: webservers
  vars_files: ./vars.yml
  tasks:
    - name: Installed Packages
      yum:
        name:
          - "{{ web_package }}"
          - "{{ ftp_package }}"
        state: present
```

- ==注意:== 什么时候变量加上双引号 {{ ansible_facts }} 如果变量是开头的时候 加上双引号 "{{ oldboy }}/lidao/a/b/c/cd/"

3.在inventory中定义变量，主机变量优先级高于主机组变量(不推荐，容易将环境弄的特别乱)

```
[root@m01 /server/playbook]# cat hosts
[lb]
172.16.1.5
172.16.1.6
```

```
[web]
172.16.1.7
172.16.1.8
172.16.1.9
172.16.1.10
[data]
172.16.1.31
172.16.1.41
172.16.1.51
[servers:children]
lb
web
[lb:vars]
soft01=nc
soft02=nmap
[root@m01 /server/playbook]# cat 04_var.yml
[root@m01 /server/playbook]# cat 04_var.yml
- hosts: lb
  #vars_files: ./vars.yml
  tasks:
    - name: Install Software nc  nmap
      yum:
        name:
        - "{{ soft01 }}"
        - "{{ soft02 }}"
        state: present
```

```
#在inventory主机清单定义变量
[root@m01 project1]# vim /etc/ansible/hosts
[webservers]
web01 ansible_ssh_host=172.16.1.7
web02 ansible_ssh_host=172.16.1.8
[webservers:vars]
filename=group_vars

#playbook执行，在/tmp目录创建group_vars文件
[root@m01 project1]# cat p3.yml
- hosts: webservers
  tasks:
    - name: Create File
      file: path=/tmp/{{ filename }} state=touch
```

4.更好的方式是在ansible的项目目录中创建额外的两个变量目录，分别是host_vars和group_vars，需要主机host_vars优先级高于group_vars

```
group_vars/
     lb/vars.yml
     web/vars.yml
     data/vars.yml

ansible   -i hosts  lb     -a 'rpm -e  nmap-ncat nmap'
ansible   -i hosts  lb     -a 'rpm -qa  nmap-ncat nmap'
ansible-playbook -i hosts    04_var.yml  -C
ansible-playbook -i hosts    04_var.yml
ansible   -i hosts  lb     -a 'rpm -qa  nmap-ncat nmap'


#group_vars目录下必须存放和inventory清单文件中定义的组名一致，如下
[root@m01 project1]# cat /etc/ansible/hosts
[webservers]
web01 ansible_ssh_host=172.16.1.7
```

```
web02 ansible_ssh_host=172.16.1.8

[root@m01 project1]# cat group_vars/webservers
web_package: httpd
ftp_package: vsftpd
#注意：系统提供了特殊的all组，也就说在group_vars目录下创建一个all文件，定义变量对所有的主机都生效

#测试group_vars和hosts_vars变量优先级(web01文件对应inventory文件中定义主机名)
[root@m01 project1]# cat host_vars/web01
web_package: zlib-static
ftp_package: zmap
[root@m01 project1]# cat group_vars/webservers
web_package: httpd
ftp_package: vsftpd


[root@m01 project1]#  cat p4.yml
- hosts: webservers
#- hosts: otherservers
  tasks:
    - name: Installed Packages
      yum:
        name:
          - "{{ web_package }}"
          - "{{ ftp_package }}"
        state: present


[root@m01 project1]# ansible-playbook p4.yml

PLAY [webservers]
*****************************************************************
*****************************************

TASK [Gathering Facts]
*****************************************************************
*****************************************
ok: [web02]
ok: [web01]
```

```
TASK [Installed Packages]
*********************************************************************
*************************************
ok: [web02]
changed: [web01]

PLAY RECAP
*********************************************************************
*************************************************
web01                      : ok=2    changed=1
unreachable=0    failed=0
web02                      : ok=2    changed=0
unreachable=0    failed=0
```

5.通过命令行覆盖变量，inventory的变量会被playbook文件中覆盖，这两种方式的变量都会被命令行直接指定变量所覆盖。使用--extra-vars或-e设定变量。

```
#playbook中引用变量
[root@manager ~]# cat f3.yml
- hosts: all

  tasks:
  - name: Create New File
    file: path=/tmp/{{ file_name }} state=touch

#playbook执行时传入file_name变量的参数，在/tmp目录创建
oldboy_extra-vars文件
[root@manager ~]# ansible-playbook f2.yml -e
"file_name=oldboy_extra-vars"
```

6.变量优先级测试

```
#1.创建目录使用变量，测试优先级
[root@m01 project1]# cat p5.yml
- hosts: webservers
  vars:
    filename: play_vars
  vars_files: ./vars.yml
  tasks:
```

```
    - name: Create
      shell: mkdir -pv /tmp/{{ filename }}
      register: mk_test

    - name: debug
      debug: msg={{ mk_test }}

#2.测试优先级结果如下:
#命令行变量--->play中的vars_files--->play中的vars变量-->host_vars中定义的变量--->group_vars/组--->group_vars/all
```

7.变量也支持层级定义，使用"."可能会有问题，建议使用"[]"代替。

```
[root@m01 ~]# cat vars_file.yml
lamp:
  web:
    web_package: httpd
    db_package: mariadb-server
    php_package: php

[root@m01 ~]# cat test.yml
---
- hosts: all
  vars_files:
    - vars_file.yml
  tasks:
    - name: Installed httpd
      yum: name={{ lamp['web']['web_package'] }}

    - name: Install Mariadb
      yum: name={{ lamp.web.db_package }}
```

变量总结:

- 剧本中变量 play　　vars
- 剧本中的变量 指定变量文件 vars_files
- 指定主机组共享的变量文件

```
#- 剧本中变量 play　　vars
- hosts: lb
```

```
  vars:
    - soft01: nc
    - soft02: nmap
  tasks:
    - name: Install Software nc  nmap
      yum:
        name:
        - "{{ soft01 }}"
        - "{{ soft02 }}"
        state: present
#- 剧本中变量 play      vars_files
cat   vars.yml
soft01: nc
soft02: nmap

剧本中调用
- hosts: lb
  vars_files: ./vars.yml
  tasks:
    - name: Install Software nc  nmap
      yum:
        name:
        - "{{ soft01 }}"
        - "{{ soft02 }}"
        state: present

#指定主机组共享的变量文件
group_vars/
     lb/vars.yml
     web/vars.yml
     data/vars.yml
##lb  web data是 主机清单中的 主机组


cat   group_vars/lb/vars.yml
soft01: nc
soft02: nmap

- hosts: lb
  tasks:
    - name: Install Software nc  nmap
```

```
    yum:
      name:
      - "{{ soft01 }}"
      - "{{ soft02 }}"
      state: present
```

3.变量注册

*register关键字可以将某个task任务结果存储至变量中，最后使用debug输出变量内容，可以用于后**续排障***

```
#应用场景: register   类似于命令行的 $?, 取出本模块的执行情况.

[root@m01 /server/playbook]# ls
01_hostname.yml  02_nfs.yml  03_nginx.yml  04_var.yml
ansible_facts  exports.j2  group_vars  hosts  vars.yml
www.conf
[root@m01 /server/playbook]# echo $?
0

#
```

```
[root@m01 /server/playbook]# cat 05_register.yml
- hosts: lb
  tasks:
```

```
    - name: print ip addr
      shell: hostname -I
      register: ip_addr
    - name:  echo
      debug:
        msg: "you ip address is {{ ip_addr }}"
```

key value

```
{
'stderr_lines': [],
u'changed': True,
u'end': u'2021-06-07 10:04:25.161236',
'failed': False,
u'stdout': u'10.0.0.5 10.0.0.3 172.16.1.5 ',
u'cmd': u'hostname -I',
u'rc': 0,
u'start': u'2021-06-07 10:04:25.153430',
u'stderr': u'',
u'delta': u'0:00:00.007806',
'stdout_lines': [u'10.0.0.5 10.0.0.3 172.16.1.5 ']
}
```

变量叫做ip_addr

```
ip_addr.stdout_lines  #取出ip
ip_addr.rc            #返回值 return code  $?
ip_addr.stdout        #标准输出, 屏幕上面的输出
ip_addr.stderr        #标准错误输出,错误信息.
```

```
[root@manager ~]# cat f5.yml
---
```

```
- hosts: all
  tasks:
    - name:
      shell: netstat -lntp
      register: System_Status

    - name: Get System Status
      debug: msg={{System_Status.stdout_lines}}

#playbook执行结果
[root@manager ~]# ansible-playbook f5.yml
PLAY [all]
**********************************************************************
**********************************************************************
***********

TASK [Gathering Facts]
**********************************************************************
**********************************************************************
ok: [10.0.0.30]

TASK [shell]
**********************************************************************
**********************************************************************
**********
changed: [10.0.0.30]

TASK [Get System Status]
**********************************************************************
**********************************************************************
ok: [10.0.0.30] => {
    "msg": [
        "tcp        0      0 0.0.0.0:22
0.0.0.0:*               LISTEN      925/sshd            ",
        "tcp6       0      0 :::22                   :::*
                    LISTEN      925/sshd            "
    ]
}
```

```
PLAY RECAP
*********************************************************************
*********************************************************************
************
10.0.0.30                  : ok=3    changed=1
unreachable=0    failed=0
```

- **register应用场景小结**
  - **用于调试 配合debug 模块 msg 输出**
  - 用于与when一起使用实现判断功能.

4.facts变量

Ansible facts是在被管理主机上通过ansible自动采集发现的变量。facts包含每台特定的主机信息。比如:被控端主机的主机名、IP地址、系统版本、CPU数量、内存状态、磁盘状态等等。

facts使用场景
1.通过facts检查CPU，来生成对应的Nginx配置文件
2.通过facts检查主机名信息，来生成不同的Zabbix配置文件
3.通过act检索的内存情况来自定义mysql的配置文件

4. 通过facts收集内存大小,设置tomcat最多可以用多少内存

```
#显示所有 ansible facts

[root@m01 /server/playbook]# ansible 172.16.1.51  -i hosts
-m setup
```

1.facts基本用法，比如获取被控端的主机名与IP地址

```
[root@m01 /server/playbook]# cat   06_facts.yml
- hosts: lb
  tasks:
    - name:  echo
      debug:
        msg:
        - "所有的ip地址 {{ ansible_all_ipv4_addresses }}"
        - "默认的ip地址 {{ ansible_default_ipv4.address }}"
        - "you interface is {{
ansible_default_ipv4.interface }}"
        - "time is  {{ ansible_date_time.date  }}"
        - "mem total is  {{ ansible_memtotal_mb  }}"
        - "you system is {{ ansible_distribution  }}
version is {{ ansible_distribution_version }}"
[root@m01 /server/playbook]# ansible-playbook -i hosts
06_facts.yml

PLAY [lb]
**********************************************************************
**********************************************************************
*****

TASK [Gathering Facts]
**********************************************************************
**********************************************************
ok: [172.16.1.5]
ok: [172.16.1.6]

TASK [echo]
**********************************************************************
**********************************************************************
***
ok: [172.16.1.5] => {
    "msg": [
        "所有的ip地址 [u'10.0.0.5', u'10.0.0.3',
u'172.16.1.5']",
        "默认的ip地址 10.0.0.5",
        "you interface is eth0",
        "time is  2021-06-07",
        "mem total is  3770",
        "you system is CentOS  version is 7.8"
```

```
    ]
}
ok: [172.16.1.6] => {
    "msg": [
        "所有的ip地址 [u'172.16.1.6', u'10.0.0.6']",
        "默认的ip地址 10.0.0.6",
        "you interface is eth0",
        "time is  2021-06-07",
        "mem total is  3770",
        "you system is CentOS  version is 7.8"
    ]
}

PLAY RECAP
*********************************************************************
*********************************************************************
****
172.16.1.5                 : ok=2    changed=0
unreachable=0    failed=0    skipped=0    rescued=0
ignored=0
172.16.1.6                 : ok=2    changed=0
unreachable=0    failed=0    skipped=0    rescued=0
ignored=0

[root@m01 /server/playb
```

```
[root@m01 ~]# cat facts.yml
- hosts: web
  tasks:
    - name: Output variables ansible facts
      debug:
        msg:
        -  this default IPv4 address "{{ ansible_fqdn}}"
is "{{ ansible_default_ipv4.address}}"
```

```
[root@m01 /server/playbook]# cat  .tmp/04-facts.yml
- hosts: lb
  tasks:
    - name: show buildin variables
      debug:
        msg:
        - "date: {{ ansible_date_time.date }}    "
        - "ip: {{ ansible_default_ipv4.address }} "
```

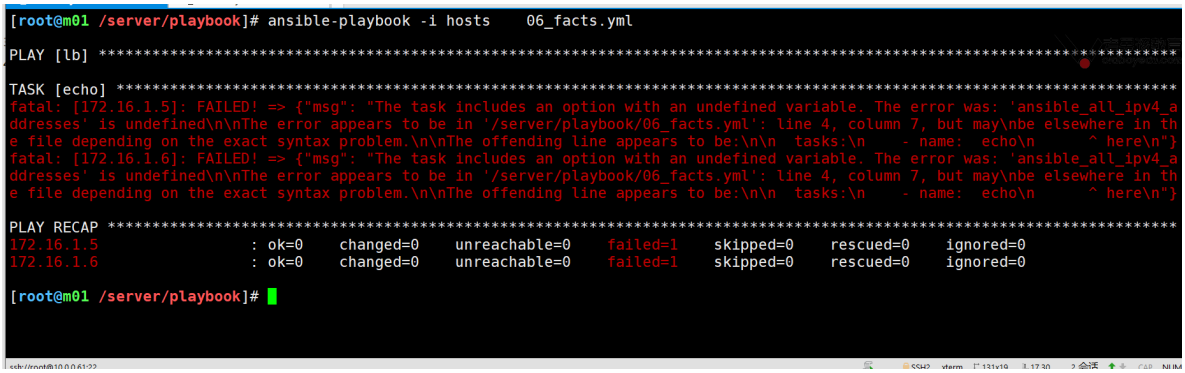
2.facts开启后会影响Ansible主机的性能，如果没有采集被控端主机需求可选择关闭

```
[root@m01 ~]# cat facts.yml
- hosts: web
  gather_facts: no  #关闭信息采集
  tasks:

  然后调用facts变量.
```

3.如何获取facts的变量，需要使用filter进行过滤

Parameter	Choices/Defaults	Comments
fact_path path *added in 1.3 of ansible.builtin*	**Default:** "/etc/ansible /facts.d"	Path used for local ansible facts (`*.fact`) - files in this dir will be run (if executable) and their results be added to `ansible_local` facts. If a file is not executable it is read instead. File/results format can be JSON or INI-format. The default `fact_path` can be specified in `ansible.cfg` for when setup is automatically called as part of `gather_facts` . NOTE - For windows clients, the results will be added to a variable named after the local file (without extension suffix), rather than `ansible_local` . Since Ansible 2.1, Windows hosts can use `fact_path` . Make sure that this path exists on the target host. Files in this path MUST be PowerShell scripts `.ps1` which outputs an object. This object will be formatted by Ansible as json so the script should be outputting a raw hashtable, array, or other primitive object.
filter list / elements=string *added in 1.1 of ansible.builtin*	**Default:** []	If supplied, only return facts that match one of the shell-style (fnmatch) pattern. An empty list basically means 'no filter'. As of Ansible 2.11, the type has changed from string to list and the default has became an empty list. A simple string is still accepted and works as a single pattern. The behaviour prior to Ansible 2.11 remains.
gather_subset list / elements=string *added in 2.1 of ansible.builtin*	**Default:** "all"	If supplied, restrict the additional facts collected to the given subset. Possible values: `all` , `min` , `hardware` , `network` , `virtual` , `ohai` , and `facter` . Can specify a list of values to specify a larger subset. Values can also be used with an initial `!` to specify that that specific subset should not be collected. For instance: `!hardware,!network,!virtual,!ohai,!facter` . If `!all` is specified then only the min subset is collected. To avoid collecting even the min subset, specify `!all,!min` . To collect only specific facts, use `!all,!min` , and specify the particular fact subsets. Use the filter parameter if you do not want to display some collected facts.




```
#通过filter进行过滤  显示我们索要的facts
[root@m01 /server/playbook]# ansible  172.16.1.5  -i hosts
-m setup  -a 'filter="*eth1*"'
172.16.1.5 | SUCCESS => {
    "ansible_facts": {
        "ansible_eth1": {
            "active": true,
            "device": "eth1",
            "features": {
                "busy_poll": "off [fixed]",
                "fcoe_mtu": "off [fixed]",
                "generic_receive_offload": "on",
                "generic_segmentation_offload": "on",
                "highdma": "off [fixed]",
                "hw_tc_offload": "off [fixed]",
                "l2_fwd_offload": "off [fixed]",
                "large_receive_offload": "off [fixed]",
                "loopback": "off [fixed]",
                "netns_local": "off [fixed]",
                "ntuple_filters": "off [fixed]",
                "receive_hashing": "off [fixed]",
                "rx_all": "off",
                "rx_checksumming": "off",
                "rx_fcs": "off",
                "rx_gro_hw": "off [fixed]",
                "rx_udp_tunnel_port_offload": "off
[fixed]",
                "rx_vlan_filter": "on [fixed]",
                "rx_vlan_offload": "on",
                "rx_vlan_stag_filter": "off [fixed]",
                "rx_vlan_stag_hw_parse": "off [fixed]",
                "scatter_gather": "on",
```

```
                "tcp_segmentation_offload": "on",
                "tx_checksum_fcoe_crc": "off [fixed]",
                "tx_checksum_ip_generic": "on",
                "tx_checksum_ipv4": "off [fixed]",
                "tx_checksum_ipv6": "off [fixed]",
                "tx_checksum_sctp": "off [fixed]",
                "tx_checksumming": "on",
                "tx_fcoe_segmentation": "off [fixed]",
                "tx_gre_csum_segmentation": "off [fixed]",
                "tx_gre_segmentation": "off [fixed]",
                "tx_gso_partial": "off [fixed]",
                "tx_gso_robust": "off [fixed]",
                "tx_ipip_segmentation": "off [fixed]",
                "tx_lockless": "off [fixed]",
                "tx_nocache_copy": "off",
                "tx_scatter_gather": "on",
                "tx_scatter_gather_fraglist": "off
[fixed]",
                "tx_sctp_segmentation": "off [fixed]",
                "tx_sit_segmentation": "off [fixed]",
                "tx_tcp6_segmentation": "off [fixed]",
                "tx_tcp_ecn_segmentation": "off [fixed]",
                "tx_tcp_mangleid_segmentation": "off",
                "tx_tcp_segmentation": "on",
                "tx_udp_tnl_csum_segmentation": "off
[fixed]",
                "tx_udp_tnl_segmentation": "off [fixed]",
                "tx_vlan_offload": "on [fixed]",
                "tx_vlan_stag_hw_insert": "off [fixed]",
                "udp_fragmentation_offload": "off
[fixed]",
                "vlan_challenged": "off [fixed]"
            },
            "hw_timestamp_filters": [],
            "ipv4": {
                "address": "172.16.1.5",
                "broadcast": "172.16.1.255",
                "netmask": "255.255.255.0",
                "network": "172.16.1.0"
            },
            "ipv6": [
```

```
                {
                    "address": "fe80::250:56ff:fe38:166c",
                    "prefix": "64",
                    "scope": "link"
                }
            ],
            "macaddress": "00:50:56:38:16:6c",
            "module": "e1000",
            "mtu": 1500,
            "pciid": "0000:02:02.0",
            "promisc": false,
            "speed": 1000,
            "timestamping": [
                "tx_software",
                "rx_software",
                "software"
            ],
            "type": "ether"
        },
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false
}
```


4.使用facts根据不同的内存生成不同Memcached配置文件

```
cat    web



#memcached配置文件如下
[root@m01 project1]# cat memcached.j2
PORT="11211"
USER="memcached"
```

```
MAXCONN="1024"
CACHESIZE="{{ ansible_memtotal_mb //2 }}"    #根据内存状态生成不同的配置(支持+-*/运算)
OPTIONS=""

[root@m01 /server/playbook]# cat memcached.j2
PORT="11211"
USER="memcached"
MAXCONN="1024"
CACHESIZE="{{ ansible_memtotal_mb //2 }}"
OPTIONS=""
#{{ ansible_default_ipv4.address }}
#{{ ansible_date_time.date }}



#playbook如下
[root@m01 project1]# cat p11.yml
- hosts: webservers
  tasks:
    - name: Installed Memcached
      yum: name=memcached state=present

    - name: Configure Memcached
      template: src=./memcached.j2
dest=/etc/sysconfig/memcached

    - name: Start Memcached
      service: name=memcached state=started enabled=yes
```

- 几个常用的facts

```
ansible_default_ipv4.address    #默认的网卡ip eth0

ansible_distribution            #系统发行版本名字 CentOS
Ubuntu  Debian ...

ansible_memtotal_mb             #内存大小

ansible_processor_vcpus
ansible_processor_cores
ansible_date_time.date
-
```

- facts 小结:
  - 通过facts取出,服务器相关信息,ip,时间,磁盘分区,.....
  - setup+filter进行过滤出

5. 变量总结

- [x] 通过剧本paly部分中vars定义变量, 剧本中play部分**vars_files**指定变量文件,通过**group_vars**指定主机组共享变量文件.
- [x] 实现echo $?. 返回值/返回信息,命令执行结果/显示. register 模块下面加上register 把模块输出存放在一个变量中. 通过debug msg输出变量内容 变量.rc 变量.stdout 变量.stderr
- [x] facts变量,比较容易的快速的取出被管理端,ip,主机名,发行版本,系统版本,内存,网卡,磁盘.....