# 1.基本操作

|  |
| --- |
| #-k:提示输入密码，对所有主机都是输入这个密码的,默认是根据当前用户执行的  ansible 192.168.230.132,192.168.230.132 -m ping -k  #-u指定用户  ansible all -m ping -k -u lw  #-v,-vv,-vvv查看详细过程  ansible all -m ping -vvv  #sudo, -K表示sudo时需要输入自己的密码,command是默认模块  ansible all -m command -a 'ls /root' -u lw -k -b -K  #或关系:在d1或者d2里的都匹配  ansible "d1:d2" -m ping  #与关系:既在d1，也在d2组里的匹配  ansible 'd1:&d2' -m ping  #查看某个模块的文档  ansible-doc command  #command 选项  ansible all -a "chdir=/tmp ls" |

# 2.ansible命令执行过程

1.加载配置文件/etc/ansible/ ansible.cfg

2.加载自己对应的模块文件,如command

3.通过ansible将模块或者命令生成对应的临时py文件，并将该问紧啊传输至远程服务器的对应执行用户$HOME/.ansible/tmp/ansible-tmp-数字/XXX.PY文件

4.给文件+x权限

5.执行并返回结果

6.删除临时py文件,sleep 0退出

可以查看执行过程

ansible all -m ping -vvv

# 3.常用模块

## 3.1copy

从服务器复制文件到客户端

ansible all -m copy -a "src=/root/01.sh dest=/tmp/01.sh owner=lw mode=600 backup=yes"

ansible all -m copy -a "content='hello\n world\n' dest=/tmp/01.sh owner=lw mode=600 backup=yes"

## 3.2fetch

和Copy相反:从客户端拉取单个文件(不是目录)到服务器端.

ansible all -m fetch -a "src=/root/a.txt dest=/tmp"

此时:/tmp对应许多匹配到的以ip地址为目录名的目录，此目录下有root/a.txt

例如:

|  |
| --- |
| [root@node1 192.168.230.133]# pwd  /tmp/192.168.230.133  [root@node1 192.168.230.133]# tree .  .  └── root  └── a.txt  1 directory, 1 file  [root@node1 192.168.230.133]# |

## 3.3file

1.创建新文件,path name dest都是一样的

ansible all -m file -a 'path=/data/haha.txt state=touch mode=0777'

2.删除文件,删除任何东西都是absent

ansible all -m file -a 'path=/data/haha.txt state=absent'

3.创建文件夹

ansible all -m file -a 'path=/data/dir1 state=directory mode=0777'

4.创建软链接

ansible all -m file -a "src=/etc/fstab path=/root/fstab.link state=link"

## 3.4hostname

修改主机名

ansible 192.168.230.133 -m hostname -a "name=www1"

## 3.5cron

1.添加一个计划任务(如果一样就修改为这个)

|  |
| --- |
| **ansible all -m cron -a "minute=\*/5 job='/usr/sbin/ntpdate ntp1.aliyun.com' name=ntp"**  [root@node1 ~]# crontab -l  \*/5 \* \* \* \* /usr/sbin/ntpdate master1  #Ansible: ntp  \*/5 \* \* \* \* /usr/sbin/ntpdate ntp1.aliyun.com |

2.禁用某个任务

|  |
| --- |
| ansible all -m cron -a "disabled=true minute=\*/5 job='/usr/sbin/ntpdate ntp1.aliyun.com' name=ntp" |

3.取消禁用(开启)

|  |
| --- |
| ansible all -m cron -a "disabled=false minute=\*/5 job='/usr/sbin/ntpdate ntp1.aliyun.com' name=ntp" |

4.删除某个job

|  |
| --- |
| ansible all -m cron -a "job='/usr/sbin/ntpdate ntp1.aliyun.com' name=ntp state=absent" |

## 3.6yum

1.安装软件,如果想指定版本，可以把name的值写全，也可以传递一个url或者一个本地的rpm包的路径

|  |
| --- |
| ansible all -m yum -a "name=vsftpd" #安装单个软件  ansible all -m yum -a "name=vsftpd,xclock" |

2.列出已安装的软件包

|  |
| --- |
| ansible all -m yum -a "list=installed" |

3.卸载软件

|  |
| --- |
| ansible all -m yum -a "name=vsftpd state=absent" |

## 3.7service

1.启动某个服务，并开机自启

|  |
| --- |
| ansible all -m service -a "name=httpd state=started enabled=yes" |

2.停止某个服务，并停止开机自启

|  |
| --- |
| ansible all -m service -a "name=httpd state=stopped enabled=no" |

3.重启某个服务

|  |
| --- |
| ansible all -m service -a "name=httpd state=restarted" |

## 3.8user

1.创建用户

|  |
| --- |
| ansible all -m user -a "name=nginx shell=/sbin/nologin system=yes home=/var/nginx groups=bin,root group=wheel comment='nginx use'" |

2.删除用户及其家目录

|  |
| --- |
| ansible all -m user -a "name=nginx state=absent remove=yes" |

## 3.9group

1.创建用户组

|  |
| --- |
| ansible all -m group -a "name=nginx2 system=yes" |

2.删除用户组

|  |
| --- |
| ansible all -m group -a "name=nginx2 state=absent" |

# 4.galaxy

ansible-galaxy：连接https://galaxy.ansible.com下载相应的roles

列出所有已安装的galaxy

ansible-galaxy list

安装galaxy

ansible-galaxy install geerlingguy.nginx

删除galaxy

ansible-galaxy remove geerlingguy.nginx

# 5.playbook

## 5.1example

|  |
| --- |
| [root@node1 ansible]# cat hello.yaml  ---  - hosts: all  remote\_user: root    tasks:  - name: task1  command: hostname  #运行  [root@node1 ansible]# ansible-playbook hello.yaml |

加密，加密之后hello.yaml文件内容乱码了

ansible-vault encrypt hello.yaml

解密,解密之后，恢复加密前的原状

ansible-vault decrypt hello.yaml

加密后查看内容:

ansible-vault view hello.yaml

加密后编辑内容:

ansible-vault edit hello.yaml

加密后修改密码:

ansible-vault rekey hello.yaml

加密创建新的playbook

ansible-vault create hello2.yaml

复杂点的例子:

|  |
| --- |
| ---  - hosts: all  remote\_user: root    tasks:  - name: craete a new file  file: name=/data/file1 state=touch  - name: echo hello world > /data/file1  shell: echo hello,world > /data/file1  - name: create a user  user: name=test1  - name: install httpd  yum: name=httpd  - name: copy httpd.conf  copy: src=/var/www/html/index.html dest=/var/www/html/index1.html  - name: start service  service: name=httpd state=started enabled=yes |

## 5.2出错继续执行

对于shell来说:

tasks:

- name: xxx

shell: /usr/bin/somecommand || /bin/true

或者使用ignore\_errors来忽略错误信息

tasks:

- name: xxx

shell: /usr/bin/somecommand

ignore\_errors: True

## 5.3playbook常见选项

--check 只是检测可能会发生的改变，但是不真实执行

--list-hosts 列出运行任务的主机

--list-tasks 列出task

--limit ,--list-hosts列出主机，limit可以只运行列出主机的一部分

-v 显示过程,-vv,-vvv

## 5.4handlers

handlers也是tasks

|  |
| --- |
| ---  - hosts: all  remote\_user: root    tasks:  - name: copy httpd.conf  copy: src=/var/www/html/index.html dest=/var/www/html/index1.html  - name: start service  service: name=httpd state=started enabled=yes |

问题：  
如果先执行了一遍这个yaml文件，然后修改src里的index.html,那么再次执行这个yaml文件时，不会执行start service 这个task,handlers则可以解决这个问题

例子:

|  |
| --- |
| ---  - hosts: all  remote\_user: root  tasks:  - name: install httpd package  yum: name=httpd  - name: copy conf file  copy: src=files/httpd.conf dest=/etc/httpd/conf/ backup=yes  notify: restart service  - name: start service  service: name=httpd state=started enabled=yes  handlers:  - name : restart service  service: name=httpd state=restarted |

notify可以出发多个handlers task,写法如下:

notify:

- xxxx1

- xxxx2

## 5.5tags

查看tags

ansible-playbook http.yaml --list-tags

如果只想要执行某个tasks里的task,怎么办？用tags

|  |
| --- |
| ---  - hosts: all  remote\_user: root  tasks:  - name: install httpd package  yum: name=httpd  tags: inshttpd  - name: copy conf file  copy: src=files/httpd.conf dest=/etc/httpd/conf/ backup=yes  notify: restart service  - name: start service  service: name=httpd state=started enabled=yes  tags: starthttpd  handlers:  - name : restart service  service: name=httpd state=restarted |

#指定tags进行执行

ansible-playbook -t inshttpd,starthttpd http.yaml

当然多个task也可以使用一个标签

## 5.6变量

变量名:字母，数字，下划线组成，必须字母开头

变量分类:

1.公共变量,优先级最低

2.普通变量,优先级其次

3.命令行指定变量,优先级最高

变量来源:

1.ansible 的setup 模块，所包含的变量

ansible all -m setup 可以查看setup模块所有变量

2./etc/ansible/hosts中定义

包含普通变量:主机组中主机单独定义

公共变量:针对主机组中所有主机定义统一变量

3.通过命令行指定变量，优先级最高

ansible-playbook -e k1=v1

4.在Playbook中定义

vars:

- var1: v1

- var2: v2

5.在role中定义

### 5.6.1命令行方式例子

|  |
| --- |
| ---  - hosts: all  remote\_user: root  tasks:  - name: install package  yum: name={{ pkname1 }}  - name: install package  yum: name={{ pkname2 }} |

[root@node1 ansible]# ansible-playbook -e 'pkname1=vsftpd pkname2=httpd' app.yaml

### 5.6.2playbook里定义变量

|  |
| --- |
| ---  - hosts: all  remote\_user: root  vars:  - pkname1: httpd  - pkname2: vsftpd  tasks:  - name: install package  yum: name={{ pkname1 }}  - name: install package  yum: name={{ pkname2 }} |

### 5.6.3hosts变量-普通变量

vi /etc/ansible/hosts

|  |
| --- |
| [d2]  192.168.230.133 http\_port=8001  192.168.230.132 http\_port=8002 |

|  |
| --- |
| ---  - hosts: d2  remote\_user: root  tasks:  - name: set hostname  hostname: name=www{{http\_port}}.com |

ansible-playbook host1.yaml

### 5.6.4hosts变量-公共变量

vi /etc/ansible/hosts

|  |
| --- |
| [d2]  192.168.230.133 http\_port=8001  192.168.230.132 http\_port=8002  [d2:vars]  nodename=www |

|  |
| --- |
| ---  - hosts: d2  remote\_user: root  tasks:  - name: set hostname  hostname: name={{nodename}}{{http\_port}}.com |

ansible-playbook host2.yaml

### 5.6.5把变量定义在单独的文件中

cat vars.yaml

|  |
| --- |
| [root@node1 ansible]# cat vars.yml  var1: httpd  var2: aaa |

testvar.yaml

|  |
| --- |
| ---  - hosts: all  remote\_user: root  vars\_files:  - vars.yml  tasks:  - name: install package  yum: name={{ var1 }}  - name: create file  file: name=/root/{{ var2 }}.log state=touch |

## 5.7.template

例子:

首先在指定目录创建templates目录,然后创建nginx.conf.j2,修改worker\_processes的值

|  |
| --- |
| worker\_processes {{ ansible\_processor\_vcpus\*2 }}; |

在templates同级目录下建立测试yaml文件.

|  |
| --- |
| ---  - hosts: all  remote\_user: root    tasks:  - name: install package  yum: name=nginx  - name: copy template  template: src=nginx.conf.j2 dest=/etc/nginx/nginx.conf backup=yes  notify: restart service  - name: start service  service: name=nginx state=started enabled=yes  handlers:  - name: restart service  service: name=nginx state=restarted |

## 5.8.when

task，要满足条件才进行执行

example

|  |
| --- |
| ---  - hosts: all  remote\_user: root    tasks:  - name: install package  yum: name=nginx  - name: copy template for centos7  template: src=nginx.conf7.j2 dest=/etc/nginx/nginx.conf backup=yes  notify: restart service  when: ansible\_distribution\_major\_version == "7"  - name: copy template for centos6  template: src=nginx.conf6.j2 dest=/etc/nginx/nginx.conf backup=yes  notify: restart service  when: ansible\_distribution\_major\_version == "6"  - name: start service  service: name=nginx state=started enabled=yes  handlers:  - name: restart service  service: name=nginx state=restarted |

## 5.9with\_items

### 5.9.1简单示例

例子:

|  |
| --- |
| ---  - hosts: all  remote\_user: root    tasks:  - name: create some files  file: name=/data/{{ item }} state=touch  with\_items:  - file1  - file2  - file3  - name: install some packages  yum: name={{ item }}  with\_items:  - htop  - sl  - hping3 |

**对于yum来说最好以下面这种写法为主:**

|  |
| --- |
| ---  - hosts: all  remote\_user: root    tasks:  - name: install some packages  yum:  name:  - htop  - sl  - httpd |

### 5.9.2字典

|  |
| --- |
| ---  - hosts: all  remote\_user: root    tasks:  - name: add some groups  group: name={{item}} state=present  with\_items:  - g1  - g2  - g3  - name: add some users  user: name={{item.name}} group={{item.group}} state=present  with\_items:  - {name: user1,group: g1}  - {name: user2,group: g2}  - {name: user3,group: g3} |

## 5.10for

cat templates/for1.conf.j2:

|  |
| --- |
| {%for port in ports %}  server {  listen {{port}}  }  {%endfor%} |

cat for1.yaml:

|  |
| --- |
| [root@node1 ansible]# cat for.yaml  ---  - hosts: all  remote\_user: root  vars:  ports:  - 81  - 82  - 83  tasks:  - name: copy conf  template: src=templates/for1.conf.j2 dest=/data/for1.conf |

cat /data/for1.conf

|  |
| --- |
| server {  listen 81  }  server {  listen 82  }  server {  listen 83  } |

## 5.11if

[root@node1 ansible]# cat templates/for2.conf.j2

|  |
| --- |
| {%for p in ports %}  server {  listen {{p.port}}  {%if p.name is defined%}  servername {{p.name}}  {%endif%}  documentroot {{p.rootdir}}  }  {%endfor%} |

[root@node1 ansible]# cat for2.yaml

|  |
| --- |
| [root@node1 ansible]# cat for2.yaml  #ports的每个元素都是字典  ---  - hosts: all  remote\_user: root  vars:  ports:  - web1:  port: 81  name: web1  rootdir: /data/web1  - web2:  port: 82  #name: web2  rootdir: /data/web2  - web3:  port: 83  name: web3  rootdir: /data/web3    tasks:  - name: copy conf  template: src=templates/for2.conf.j2 dest=/data/for2.conf |

成成的文件:

|  |
| --- |
| server {  listen 81  servername web1  documentroot /data/web1  }  server {  listen 82  documentroot /data/web2  }  server {  listen 83  servername web3  documentroot /data/web3  } |

# 6.roles

## 6.1简介

用于层次性，结构化的组织playbook.

roles能够根据层次结构自动装载变量文件，tasks,handlers.