**一、saltstack安装**

设置hostname以及hosts，jzlinux-01,jzlinux-02 //两台设备上都要设置

vim /etc/hosts

192.168.x.x jzlinux-01

192.168.x.x jzlinux-02

两台机器全部安装saltstack yum源

yum install -y https://repo.saltstack.com/yum/redhat/salt-repo-latest-2.el7.noarch.rpm

11上执行 yum install -y salt-master salt-minion //服务端

19上执行 yum install -y salt-minion //客户端

**二、启动salt相关服务**

1、vi /etc/salt/minion //在服务端上编辑更改配置文件



systemctl start salt-master; systemctl start salt-minion //在服务端启动服务

2、vi /etc/salt/minion //在客户端上编辑更改配置文件



systemctl start salt-minion //在客户机上启动

3、别人启动服务时报错

分析错误信息：从查看服务的报错信息中可以看到这主要的两句：

import psutil.\_pslinux as \_psplatform

AttributeError: 'module' object has no attribute '\_pslinux'

这个错误是因为导入psutil包中的\_pslinux时发现\_pslinux不存在，解决办法就是重新下载安装psutil包即可，步骤如下：

[root@server ~]# wget https://pypi.python.org/packages/source/p/psutil/psutil-3.2.2.tar.gz

[root@server ~]# tar zxvf psutil-3.2.2.tar.gz

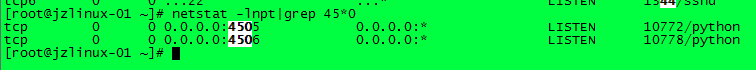
[root@server ~]# cd psutil-3.2.2/

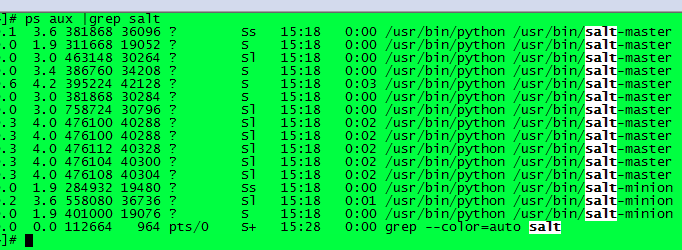
[root@server ~/psutil-3.2.2]# python setup.py install

[root@server ~/psutil-3.2.2]# echo $?

0

4、服务端监听4505和4506两个端口，4505为消息发布的端口，4506为和客户端通信的端口





**三、saltstack配置认证**

1、在客户端可以看到，pem为私钥，pub为公钥

[root@jzlinux-02 ~]# ls /etc/salt/pki/minion/

minion.pem minion.pub

2、服务端也可以看到私钥和公钥

[root@jzlinux-01 ~]# ls /etc/salt/pki/master/

master.pem minions minions\_denied minions\_rejected

master.pub minions\_autosign minions\_pre

3、在服务端master执行

salt-key list //显示下面内容

Accepted Keys:

jzlinux-01

Denied Keys:

Unaccepted Keys:

jzlinux-02

Rejected Keys:

[root@jzlinux-01 ~]# salt-key -a jzlinux-02 //显示下面内容

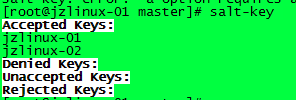
The following keys are going to be accepted:

Unaccepted Keys:

jzlinux-02

Proceed? [n/Y] y

Key for minion jzlinux-02 accepted.



4、查看接受设备的公钥



5、相关目录选项

* -a 后面跟主机名，认证指定主机
* -A 认证所有主机
* -r 跟主机名，拒绝指定主机
* -R 拒绝所有主机
* -d 跟主机名，删除指定主机认证
* -D 删除全部主机认证
* -y 省略掉交互，相当于直接按了y

6、删除所有主机

[root@jzlinux-01 master]# salt-key -D

The following keys are going to be deleted:

Accepted Keys:

jzlinux-01

jzlinux-02

Proceed? [N/y] Y

Key for minion jzlinux-01 deleted.

Key for minion jzlinux-02 deleted.

7、删除后识别的密钥也没有了



8、这时候无法添加主机，需要重启服务端和客户端上的服务

[root@jzlinux-01 ~]# salt-key -A

The key glob '\*' does not match any unaccepted keys.

[root@jzlinux-01 master]# salt-key //这时候都是空的

Accepted Keys:

Denied Keys:

Unaccepted Keys:

Rejected Keys:

9、systemctl restart salt-minion //服务端和客户端上都要执行重启

[root@jzlinux-01 master]# salt-key //这时候里面有东西了

Accepted Keys:

Denied Keys:

Unaccepted Keys:

jzlinux-01

jzlinux-02

Rejected Keys:

10、拒绝主机，只要当主机在Unaccepted Keys:下面的时候才能用“-r”

[root@jzlinux-01 ~]# salt-key -r jzlinux-02 -y

The following keys are going to be rejected:

Unaccepted Keys:

jzlinux-02

Key for minion jzlinux-02 rejected.

[root@jzlinux-01 ~]# salt-key

Accepted Keys:

Denied Keys:

Unaccepted Keys:

jzlinux-01

Rejected Keys:

jzlinux-02

11、想要将拒绝的搞到接受里面去，需要先删除，再配置，注意删除后在被删除主机上重启服务。

[root@jzlinux-01 ~]# salt-key -d jzlinux-02 -y

The following keys are going to be deleted:

Rejected Keys:

jzlinux-02

Key for minion jzlinux-02 deleted.

[root@jzlinux-01 ~]# salt-key -a jzlinux-02

The following keys are going to be accepted:

Unaccepted Keys:

jzlinux-02

Proceed? [n/Y] y

Key for minion jzlinux-02 accepted.

**四、saltstack远程执行命令**

1、测试设备是否在线

salt '\*' test.ping

jzlinux-02:

True

jzlinux-01:

True

2、运行系统命令

[root@jzlinux-01 ~]# salt '\*' cmd.run "ip add|grep 192"

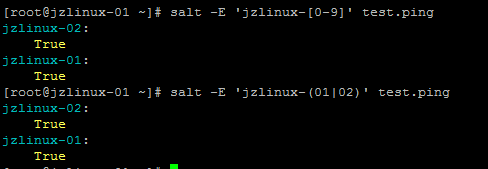
jzlinux-01:

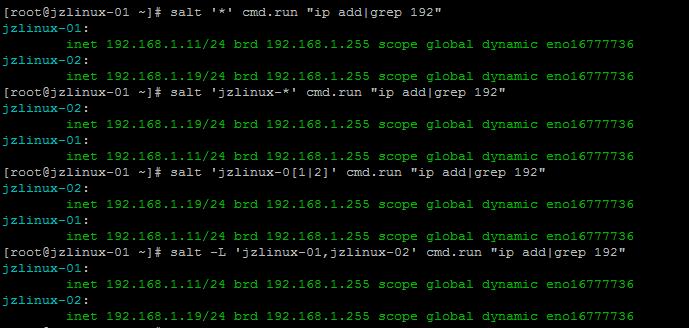
inet 192.168.1.11/24 brd 192.168.1.255 scope global dynamic eno16777736

jzlinux-02:

inet 192.168.1.19/24 brd 192.168.1.255 scope global dynamic eno16777736

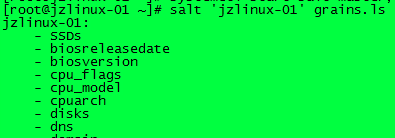
相关命令使用截图



****

**四、grains**

1、salt 'jzlinux-01' grains.ls //列出所有项目的名字



2 、salt 'jzlinux-03' grains.items|grep -A1 ip4\_ //显示项目的值

ip4\_nameservers:

- 192.168.1.1

--

ip4\_interfaces:

----------

3、

minion上：

vim /etc/salt/grains //添加：

role: nginx

env: test

重启minion服务

systemctl restart salt-minion

master上：

获取grains：

salt '\*' grains.item role env

4、在master获取

[root@jzlinux-01 etc]# salt 'jzlinux-03' grains.item role env

jzlinux-03:

----------

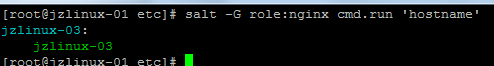
env:

test

role:

nginx

5、salt -G role:nginx cmd.run 'hostname' //根据定义的东西来获取设备信息



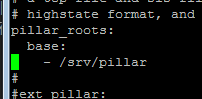
**五、pillar //在master定义**

1、vim /etc/salt/master //编辑配置文件

pillar\_roots: //项目名称

base: #此行前面有两个空格 //这个我理解为key

- /srv/pillar #此行前面有4个空格 //这个是项目的值



2、systemctl restart salt-master //重启服务

3、mkdir /srv/pillar //在master创建目录

4、在创建的目录下，创建配置文件

vi test.sls //添加下面的内容

conf: /etc/123.conf //key: value的形式

vi test1.sls

dir: /data/123

5、vi /srv/pillar/top.sls //添加下面的内容

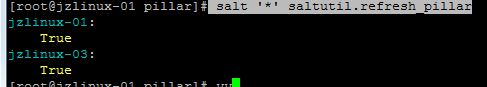
base:

'jzlinux-03': //这里是客户机设备的主机名。

- test //test就是要加载的配置文件名称，上面定义过的test.sls

- test1

6、salt '\*' saltutil.refresh\_pillar //刷新配置文件



7、验证

[root@jzlinux-01 pillar]# salt '\*' pillar.item conf dir

jzlinux-01:

----------

conf:

dir:

jzlinux-03:

----------

conf:

/etc/123.conf

dir:

/data/123

8、每个主机定义一个配置，改top配置后，需要刷新

[root@jzlinux-01 pillar]# cat top.sls

base:

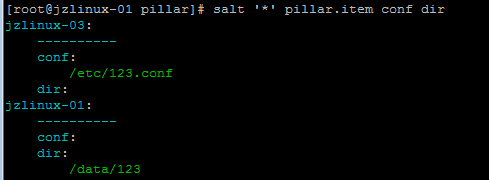
'jzlinux-03':

- test

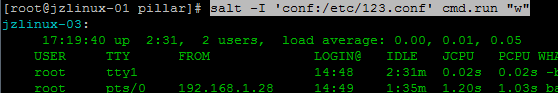
'jzlinux-01':

- test1

salt '\*' saltutil.refresh\_pillar //刷新配置文件

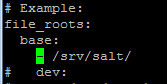


9、salt -I 'conf:/etc/123.conf' cmd.run "w" //通过值来匹配主机



**六、安装配置httpd**

vi /etc/salt/master //编辑master



2、systemctl restart salt-master //重启服务

3、mkdir /srv/salt //创建目录

vi top.sls //在创建的目录下，编辑配置文件

base:

'\*': //\*代表所有主机，也可以指定知己：（jzlinux-02）

- httpd

4、vim httpd.sls //在创建的目录下编辑文件

httpd-service:

pkg.installed: //安装包的函数

- names:

- httpd //安装包的名称

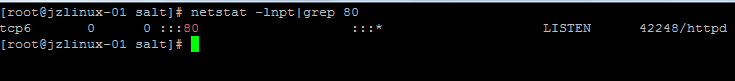
- httpd-devel //安装包的名称

service.running: //运行服务的函数

- name: httpd //运行服务的名称

- enable: True //是否需要开机启动，True表示开机启动，false：开机不启动

3、salt 'jzlinux-01' state.highstate //安装测试



**七、配置管理文件**

vi /srv/salt/test.sls //在master上编辑文件，增加下面的内容

file\_test:

file.managed:

- name: /tmp/jzlinux.com

- source: salt://test/123/1.txt

- user: root

- group: root

- mode: 600

* 说明：第一行的file\_test为自定的名字，表示该配置段的名字，可以在别的配置段中引用它，source指定文件从哪里拷贝，这里的salt://test/123/1.txt相当于是/srv/salt/test/123/1.txt

vi /etc/salt/master 定义了 /srv/salt

* 

2、mkdir -p /srv/salt/test/123

cp /etc/inittab test/123/1.txt //随便搞个文件测试

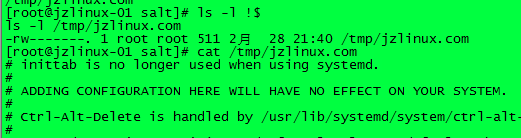
3、vi /srv/salt/top.sls

base:

'\*':

- test

4、salt 'jzlinux-03' state.highstate //执行，然后去执行设备上查看是否有相关内容产生



**八、配置管理目录**

1、master上vi /srv/salt/test\_dir.sls //加入如下内容

file\_dir:

file.recurse:

- name: /tmp/testdir

- source: salt://test/123

- user: root

- file\_mode: 640

- dir\_mode: 750

- mkdir: True

- clean: True True //加上它之后，源删除文件或目录，目标也会跟着删除，否则不会删除

2、修改top.sls, vi /srv/salt/top.sls //改为如下内容

cat /srv/salt/top.sls

base:

'\*':

- test //这个是上个实验做的，与本次无关

- test\_dir

3、salt 'jzlinux-02' state.highstate

4、去jzlinux-02主机上查看

[root@jzlinux-02 testdir]# pwd

/tmp/testdir

5、测试

/srv/salt/test/123 //在master的该目录下创建如下文件

touch.22.txt

mkdir 456

cd 456 //salt 无法同步空目录

touch 22.txt

vi top.sls //更改top的配置

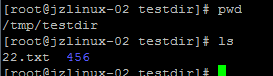
base:

'\*':

- test\_dir

salt 'jzlinux-02' state.highstate

去jzlinux-02主机上查看否有相同的目录和文件创建



**九、配置管理远程命令**

1、master 上 vi /srv/salt/shell\_test.sls //加入如下内容

shell\_test:

cmd.script:

- source: salt://test/1.sh

- user: root

2、vi /srv/salt/test/1.sh

#!/bin/bash

touch /tmp/111.txt

if [ ! -d /tmp/1233 ]

then

mkdir /tmp/1233

fi

3、

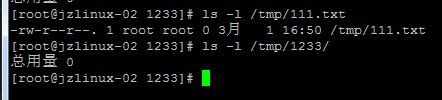
vim /srv/salt/top.sls

base:

'\*':

- shell\_test

3、去“jzlinux-02”主机上验证结果。



**十、配置管理计划任务**

1、master上vi /srv/salt/cron\_test.sls //加入如下内容

cron\_test:

cron.present:

- name: /bin/touch /tmp/111.txt

- user: root

- minute: '\*'

- hour: 20

- daymonth: '\*'

- month: '\*'

- dayweek: '\*'

2、vim /srv/salt/top.sls //master操作

base:

'\*':

- cron\_test

3、salt 'jzlinux-02' state.highstate

4、去minion也就是客户机上测试，看是否有该任务计划生成

crontab -e



5、如果想变更任何计划的时间，就变更下面的内容

master上vi /srv/salt/cron\_test.sls //加入如下内容

cron\_test:

cron.present:

- name: /bin/touch /tmp/111.txt

- user: root

- minute: '\*'

- hour: 20

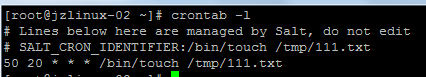
- daymonth: '\*'

- month: '\*'

- dayweek: '\*'

6、salt 'jzlinux-02' state.highstate

7、在minion查看，里面的内容不要手动更改



8、删除任务计划

master上vi /srv/salt/cron\_test.sls //加入如下内容

cron\_test:

cron.absent:

- name: /bin/touch /tmp/111.txt

9、salt 'jzlinux-02' state.highstate //master执行

10、去客户机上测试，看是否删除任务计划

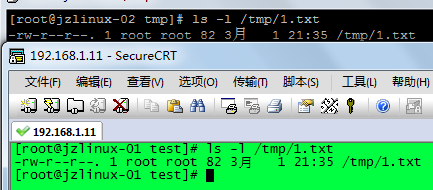


**十一、saltstack – 其它可能会用到的命令**

1、cp.get\_file 拷贝master上的文件到客户端

salt '\*' cp.get\_file salt://test/1.sh /tmp/1.txt

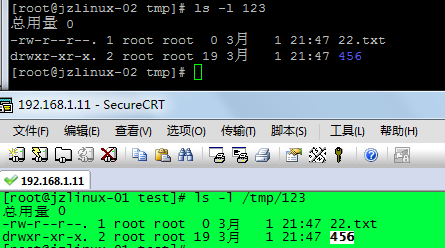
2、查看设备的/tmp/目录下是否有 1.txt



3、拷贝目录到客户端

salt '\*' cp.get\_dir salt://test/123 /tmp/

4、去客户端查看，是否有目录被创建



5、salt-run manage.up //显示存活的minion



6、salt '\*' cmd.script salt://test/1.sh 命令行下执行master上的shell脚本

**十二、salt-ssh使用**

1、yum install -y https://repo.saltstack.com/yum/redhat/salt-repo-latest-2.el7.noarch.rpm

2、yum install -y salt-ssh //都在master上操作

3、vi /etc/salt/roster //在master操作

jzlinux-01:

host: 192.168.1.11

user: root

passwd: 123456

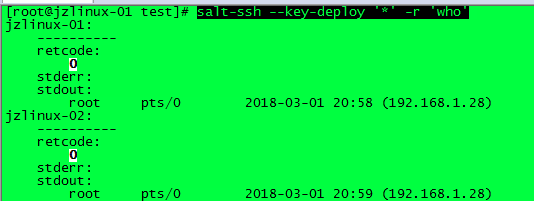
jzlinux-02:

host: 192.168.1.19

user: root

passwd: 123456

4、salt-ssh --key-deploy '\*' -r 'who' //第一次执行的时候会自动把本机的公钥放到对方机器上，然后就可以把roster里面的密码去掉



**十三、ansible安装**

1、yum install -y ansible.noarch ansible-doc.noarch //安装包

ansible是不需要安装客户端的。

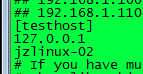
2、ssh-keygen -t rsa //在总分发设备上生成密钥对，直接回车就行



3、vi .ssh/authorized\_keys //在被分发设备上，将总分发设备生成的公钥及id\_rsa.pub里面的内容添加到这里去，

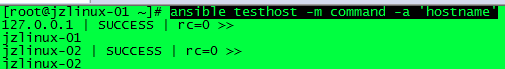
4、ssh jzlinux-02 //在总分发设备上测试，看能否登录到被分发设备上去。

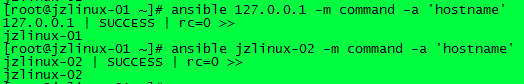
5、vi /etc/ansible/hosts //在总上添加如下内容，testhost是自定义的组名，下面是这个组里面的主机，127.0.0.1是自己，所以在自己设备上也要执行第3部，这里可以写IP，也可以写主机名，如果写主机名，请到vi /etc/hosts里面定义



**十四、ansible远程执行命令**

1、ansible testhost -m command -a 'hostname'

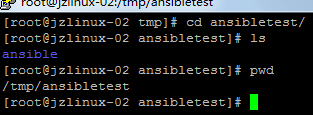




**十五、 ansible拷贝文件或目录**

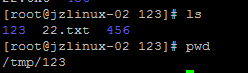
1、ansible jzlinux-02 -m copy -a "src=/etc/ansible dest=/tmp/ansibletest owner=root group=root mode=755”

拷贝目录到指定的设备，这时候jzlinux-02设备上是没有/tmp/ansibletest，但是拷贝的时候会自动创建，而且会把源目录/etc/ansible直接放到创建的/tmp/ansibletest/里面去。



如果指定的源目录和目标目录一样且存在，那么会把源目录放在目标目录里面去

2、ansible jzlinux-02 -m copy -a "src=/tmp/123 dest=/tmp/123 owner=root group=root mode=755"



3、同步文件，这时候源上的1.txt会放在目标/tmp/123/里面去，如果指定的目标目录不存在，会先创建目录，再把copy的文件放在指定的目录下

ansible jzlinux-02 -m copy -a "src=/tmp/1.txt dest=/tmp/123 owner=root group=root mode=755"

如果源文件及目录与目标文件及目录一样且内容一致，那么源文件的内容会覆盖目标文件的内容，同步的目录相同的话，源新增的东西会同步到目标，而且源不会覆盖目标之前已经存在的东西。

**十六、ansible远程执行脚本**

1、ansible需要把脚本先拷贝到远程设备上去，然后在总分发设备上执行。

ansible testhost -m copy -a "src=/tmp/1.sh dest=/tmp/test.sh mode=0755"

2、ansible testhost -m shell -a "/tmp/test.sh" //执行脚本

cat /tmp/test.sh

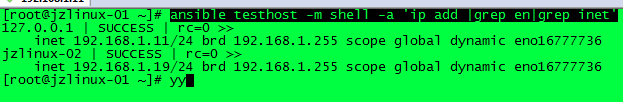
#!/bin/bash

echo `date` > /tmp/ansible\_test.txt

3、查看结果：



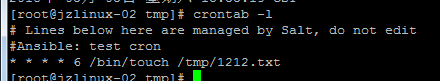
3、ansible testhost -m shell -a 'ip add |grep en|grep inet'



shell:模块在执行命令的时候，可以待管道。

**十七、ansible管理任务计划**

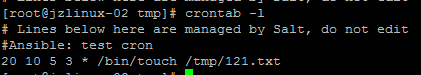
ansible testhost -m cron -a "name='test cron' job='/bin/touch /tmp/1212.txt' weekday=6" //在总分发设备上定义



删除

ansible testhost -m cron -a "name='test cron' state=absent"

ansible jzlinux-02 -m cron -a "name='test cron' job='/bin/touch /tmp/121.txt'  minute=20 hour=10 day=5 month=3"



**十八、ansible安装包和管理服务**

1、ansible testhost -m yum -a "name=vim" //安装包

2、ansible jzlinux-02 -m yum -a "name=httpd state=removed" //卸载包

3、ansible jzlinux-02 -m service -a "name=httpd state=started enabled=no"

启动程序。enabled=no（开机不启动）|yes(开机启动)

4、ansible-doc -l //列出所有的模块

5、ansible-doc shell //列出指定模块

**十九、使用ansible playbook**

1、vi  /etc/ansible/test.yml

---

- hosts: jzlinux-02

remote\_user: root

tasks:

- name: test\_playbook

shell: touch /tmp/lishiming.txt

2、ansible-playbook test.yml //执行

3、在“jzlinux-02”测试



**二十、playbook里的变量**

1、vi /etc/ansible/create\_user.yml //就是在指定主机上创建一个test用户

---

- name: create\_user

hosts: jzlinux-02

user: root

gather\_facts: false

vars: //定义一个名为user的变量，其值为test

- user: "test"

tasks:

- name: create user

user: name="{{ user }}" //{{就是上面的user变量}}

**二十一、 playbook循环**

vim /etc/ansible/while.yml //批量更改文件的权限

---

- hosts: jzlinux-02

user: root

tasks:

- name: change mode for files

file: path=/tmp/{{ item }} mode=600

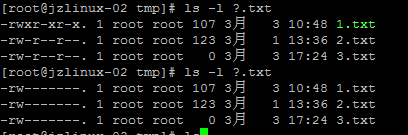
with\_items:

- 1.txt

- 2.txt

- 3.txt

ansible-playbook while.yml



cat while.yml //现创建文件，然后改权限

---

- hosts: jzlinux-02

user: root

tasks:

- name: change mode for files

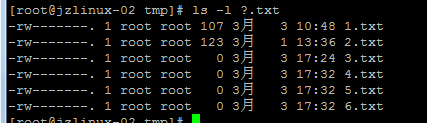
file: path=/tmp/{{ item }} state=touch mode=600

with\_items:

- 4.txt

- 5.txt

- 6.txt



**二十二、playbook中的条件判断**

1、vi /etc/ansible/when.yml

---

- hosts: testhost

user: root

gather\_facts: True

tasks:

- name: use when

shell: touch /tmp/when.txt

when: ansible\_ eno16777736.ipv4.address == "192.168.1.19"

这个条件“ansible\_ eno16777736.ipv4.address”，是匹配的ansible jzlinux-02 -m setup

现在找到ansible\_ eno16777736------ipv4----- "address": "192.168.1.19"。

2、ansible-playbook when.yml //运行



ansible\_ eno16777736.ipv4.address：

从ansible jzlinux-02 -m setup去找



|



**二十三、playbook中的handlers**

1、vi /etc/ansible/handlers.yml

---

- name: handlers test

hosts: jzlinux-02

user: root

tasks:

- name: copy file

copy: src=/etc/passwd dest=/tmp/aaa.txt

notify: test handlers

handlers:

- name: test handlers

shell: echo "111111" >> /tmp/aaa.txt

说明：就是tasks里面的东西执行成功后，继续执行handlers里面的内容，如果copy里面前后2个文件里面的内容一致，并不会去执行handlers里面的shell相关命令。 这种比较适合配置文件发生更改后，重启服务的操作。

2、ansible-playbook handlers.yml //执行

**二十四、用playbook安装nginx（源码安装）**

思路：先在一台机器上编译安装好nginx、打包，然后再用ansible去下发

1、cd /etc/ansible 进入ansible配置文件目录

2、mkdir nginx\_install 创建一个nginx\_install的目录，方便管理

3、cd nginx\_install

4、mkdir -p roles/{common,install}/{handlers,files,meta,tasks,templates,vars}

说明，先创建roles目录，然后在roles目录下创建common、install目录，然后在common和install目录下创建handlers,files,meta,tasks,templates,vars目录。

5、在设备上编译安装好nginx

6、打包安装好的nginx目录 //--exclude不要打包的内容

tar czvf nginx.tar.gz --exclude "nginx.conf" --exclude "vhost" /usr/local/nginx/

7、打包的文件移到指定目录

mv nginx.tar.gz /etc/ansible/nginx\_install/roles/install/files/

8、拷贝相应的文件到指定目录

cp /usr/local/nginx/conf/nginx.conf /etc/ansible/nginx\_install/roles/install/templates/

cp /etc/init.d/nginx /etc/ansible/nginx\_install/roles/install/templates/

9、

cd /etc/ansible/nginx\_install/roles/common/

vim tasks/main.yml

- name: Install initializtion require software

yum: name={{ item }} state=installed

with\_items:

- zlib-devel

- pcre-devel

9、vim /etc/ansible/nginx\_install/roles/install/vars/main.yml

nginx\_user: www

nginx\_port: 80

nginx\_basedir: /usr/local/nginx

10、vim /etc/ansible/nginx\_install/roles/install/tasks/copy.yml

- name: Copy Nginx Software

copy: src=nginx.tar.gz dest=/tmp/nginx.tar.gz owner=root group=root

//src没有定义路径，原因是默认会去/etc/ansible/nginx\_install/roles/install/files找内容

- name: Uncompression Nginx Software

shell: tar zxf /tmp/nginx.tar.gz -C /usr/local/

- name: Copy Nginx Start Script

template: src=nginx dest=/etc/init.d/nginx owner=root group=root mode=0755

- name: Copy Nginx Config

template: src=nginx.conf dest={{ nginx\_basedir }}/conf/ owner=root group=root mode=0644

11、vim /etc/ansible/nginx\_install/roles/install/tasks/install.yml

- name: Create Nginx User

user: name={{ nginx\_user }} state=present createhome=no shell=/sbin/nologin

- name: Start Nginx Service

shell: /etc/init.d/nginx start

- name: Add Boot Start Nginx Service

shell: chkconfig --level 345 nginx on

- name: Delete Nginx compression files

shell: rm -rf /tmp/nginx.tar.gz

11、再创建main.yml并且把copy和install调用

- include: copy.yml

- include: install.yml

12、vim /etc/ansible/nginx\_install/install.yml

---

- hosts: jzlinux-02

remote\_user: root

gather\_facts: True

roles:

- common

- install

12、ansible-playbook /etc/ansible/nginx\_install/install.yml //执行

**二十五、playbook管理配置文件**

1、创建相关目录，在总分发设备上执行

mkdir -p /etc/ansible/nginx\_config/roles/{new,old}/{files,handlers,vars,tasks}

2、cd nginx.conf

cp -r nginx.conf vhost /etc/ansible/nginx\_config/roles/new/files/

3、vim /etc/ansible/nginx\_config/roles/new/vars/main.yml

nginx\_basedir: /usr/local/nginx

4、vim /etc/ansible/nginx\_config/roles/new/handlers/main.yml

- name: restart nginx

shell: /etc/init.d/nginx reload

5、 vim /etc/ansible/nginx\_config/roles/new/tasks/main.yml

- name: copy conf file

copy: src={{ item.src }} dest={{ nginx\_basedir }}/{{ item.dest }} backup=yes owner=root group=root mode=0644

with\_items:

- { src: nginx.conf, dest: conf/nginx.conf }

- { src: vhost, dest: conf/ }

notify: restart nginx

6、vim /etc/ansible/nginx\_config/update.yml

---

- hosts: jzlinux-02

user: root

roles:

- new

ansible-playbook /etc/ansible/nginx\_config/update.yml //执行