### Ansible教程--马哥【王晓春】

**自动化运维应用场景**

文件传输

命令执行：应用部署、配置管理、任务流编排

**企业实际应用场景分析**

发布环境：代码发布机(2台主备)--**堡垒机**(安全屏障--审计)

功能：发布代码到生产环境

生产环境：少数情况开放权限给核心开放人员。**运维**负责

功能：对用户提供公司产品的服务【机器数量多且应用重要】

灰度环境：生产环境的一部分---【**灰度部署**】

功能：在全量发布代码前将代码功能面向少量精准用户发布的环境，

基于**主机或用户**执行灰度发布。

过程：版本功能变更较大，让一部分用户**提前体验**该功能，没问题后在全量发布

**程序发布：**

预发布验证：新版本的代码先发布到服务器

【跟线上环境配置完全相同，只是未接入到调度器】

程序发布：不能导致系统故障或造成系统完全不可用

不能影响用户体验

灰度发布：

发布路径：/webapp/tuangou【客户请求--软链接】

/webapp/tuangou-1.1【旧版本】

/webapp/tuangou-1.2【新版本】

发布过程：在调度器**上下线**一批主机【标记为**maintanance**状态】--关闭服务--并部署新版本的应用程序--启动服务--在调度器上启用这一批服务器

自动化灰度发布：脚本、发布平台

**常用自动化运维工具**

**Ansible**：python，Agentless【适合中小型应用环境--**SSH+Playbooks**】

**Saltstack**：python，一般需部署agent，执行效率高【适合大型环境】

Puppet：Ruby，功能强大，配置复杂【重型，适合大型环境】

Fabric：python，agentless

Chef：Ruby，国内应用少

Cfengine

func

**Ansible工作架构和原理**

特性：

**模块化**：调用特定的模块，完成特定任务

有**Paramiko**(适配SSH)、**PyYAML**(playbook)、**Jinja2**(模块语言)三个关键模块

支持自定义模块

基于python语言实现

部署简单，基于**Python+SSH**【默认安装】agentless

安全，基于OpenSSH

支持**playbook**编排任务

**幂等性**：一个任务执行1遍和执行n遍效果一样，不因重复执行带来意外情况

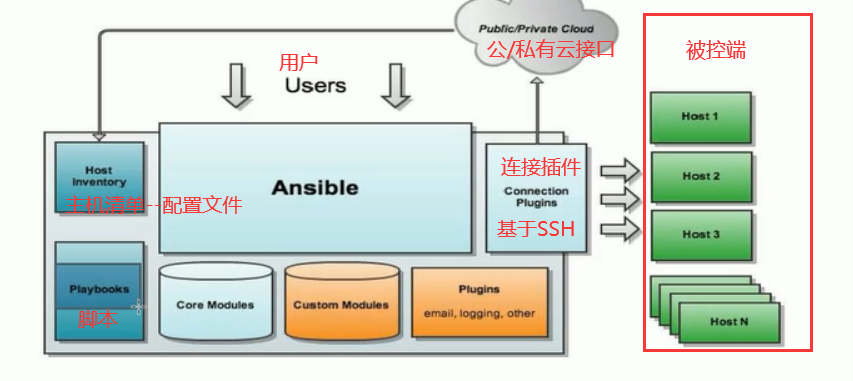
无需代理不依赖PKI(无需ssl)

可使用任何编程语言写模块

**Yaml**格式，编排任务，支持丰富的数据结构

较强大的多层解决方案

**Ansible架构**：

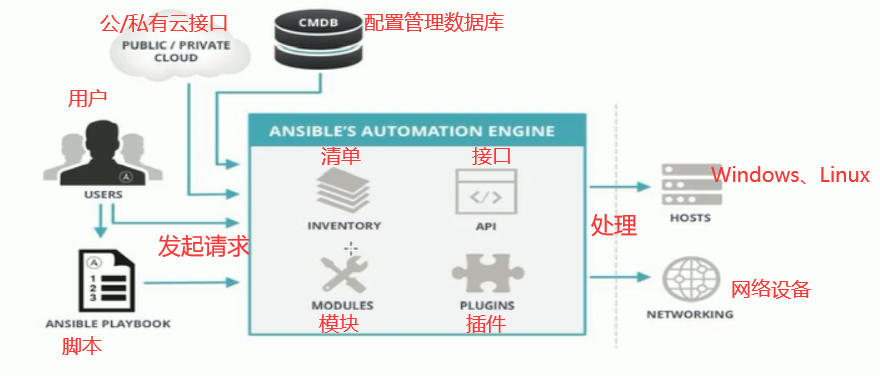


**Host inventory**：主机清单--配置文件

**Playbook**--脚本

**Connection Plugin**s连接插件--基于SSH或私/公有云接口

**Ansible工作原理**



CMDB：配置管理数据库--管理

Ansible组成部分：包含Inventory(清单)、Modules(模块)、API(接口)、Plugins(插件)

发起请求：**用户、Ansible Playbook**、CMDB、公/私有云接口

处理主机：Window、Linux Networking：网络设备

**Ansible主要组成部分**

Ansible **playbook**：**任务集**，编排定义Ansible任务集的配置文件

由Ansible**顺序**依次执行，JSON格式的YML文件

**Inventory**：Ansible管理主机的**清单**：**/etc/ansible/hosts**

**Modules**：执行命令的**功能模块**，【内置核心模块、可自定义】

**Plugins**：模块功能的补充，如连接类、循环、变量、过滤等**插件**【不常用】

**API**：供第三方程序调用的应用程序编程接口

**Ansible**：ansible命令工具，核心执行工具

**Ansible命令执行来源**：

User(普通用户)：即system administrator

CMDB(配置管理数据库)：API调用

Public/Private CLOUD：API调用

User--》Ansible Playbook--》Ansible

**利用ansible实现管理方式**：

**Ad-Hoc**： 即ansible命令，用于临时命令使用场景【单条命令】

**Ansible-playbook**：ansible任务集，用于长期规划、大型项目的场景【有前提规划】

Ansible-playboook(剧本)执行过程：

将已有编排好的任务集写入Ansible-Playbook

通过ansible-playbook命令分拆任务集至逐条ansible命令，按预定规则逐条执行

Ansible操作对象： Hosts主机、Networking网络设备

注意事项：

执行ansible的主机一般称为**主控端，中控，master或堡垒机**

主控端python版本>=2.6

被控端python版本<2.4时，需安装python-simplejson

被控端如开启SElinux需要安装libselinux-python

Windows不能作为主控端

官网：www.ansible.com

Ansible的安装

①**rpm包安装**：

EPEL源

wget -O /etc/yum.repos.d/epel.repo https://mirrors.aliyun.com/repo/epel-7.repo

yum -y install ansible

yum info ansible #查看ansible相关版本信息

rpm -ql ansible #ansible相关目录

ansible --version

②**编译安装**

yum -y install python-jinja2 PyYAML python-paramiko python-babel python-crypto

tar xf ansible-1.5.4.tar.gz

cd ansible-1.5.4

python setup.py build

python setup.py install

mkdir /etc/ansible

cp -r examples/\* /etc/ansible

③**Git安装**

git clone git://github.com/ansible/ansible.git --recursive

cd ./ansible

source ./hacking/env-setup

Pip安装：pip是python包的管理器【同yum】

yum -y install python-pip python-devel

yum -y install gcc glibc-devel zibl-devel rpm-bulid openssl-devel

pip install --upgrade pip

pip install ansibl --upgrade

确认安装：ansible --version

**相关文件**

**配置文件**：

/etc/ansible/**ansible.cfg**：主配置文件，配置ansible工作特性

/etc/ansible/**hosts**：主机清单

/etc/ansible/**roles**/：存放角色命令

**程序**：

/usr/bin/ansible：主程序，临时命令执行工具

/usr/bin/**ansible-doc**：查看配置文档，模块功能查看【同man】

/usr/bin/**ansible-galaxy**：上传/下载优秀代码或Roles模块的官网平台

/usr/bin**/ansible-playbook**：定制自动化任务，编排剧本工具

/usr/bin/ansible-pull：远程执行命令的工具

/usr/bin/ansible-vault：文件加密工具

/usr/bin/ansible-console：基于console界面与用户交互的执行工具

**主机清单：inventory**

默认的inventory file为：**/etc/ansible/hosts**

主要功能：**批量主机操作**，若便捷使用部分主机，用**[]分组命名**

inventory file可以有多个，可以通过Dynamic Inventory动态生成

inventory遵循INI文件风格，[xx]：xx为组名。同一个主机可归为多个不同的组中。 若目标主机使用非默认的SSH端口，在**主机名称+ :Port**标注

e.g ntp.magedu.com

[websrvs]

www1.magedu.com:222

www2.magedu.com

[dbsrvs]

db1.magedu.com

192.168.10.11

192.168.10.91:1111

若主机名称遵循相似的命名模式，使用列表方式标识各主机

[websrvs]

www**[01:100]**.example.com

[dbsrvs]

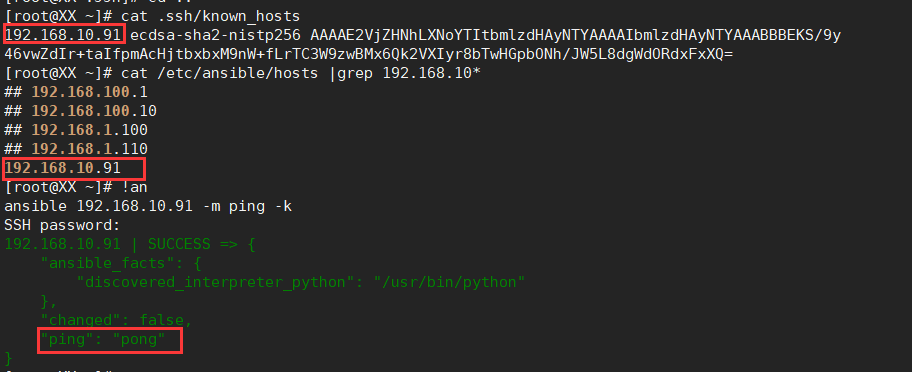
db-[a:f].example.com

ansible --help #查看命令帮助

ansible IP -m ping -k #m：模块 k：key 输入用户名及口令

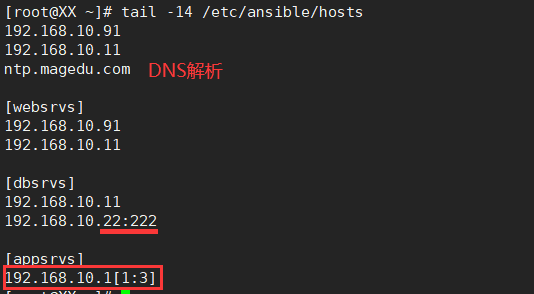
必须先加入到主机清单：**/etc/ansible/hosts 添加IP**

e.g ansible 192.168.10.91 -m ping -k #**走SSH协议**



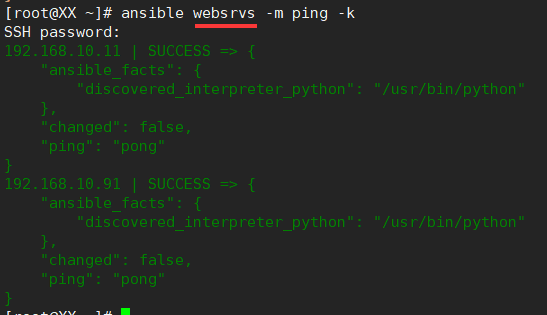
ansible 192.168.10.11,192.168.10.91 -m ping -k #两台主机密钥一样+可信主机

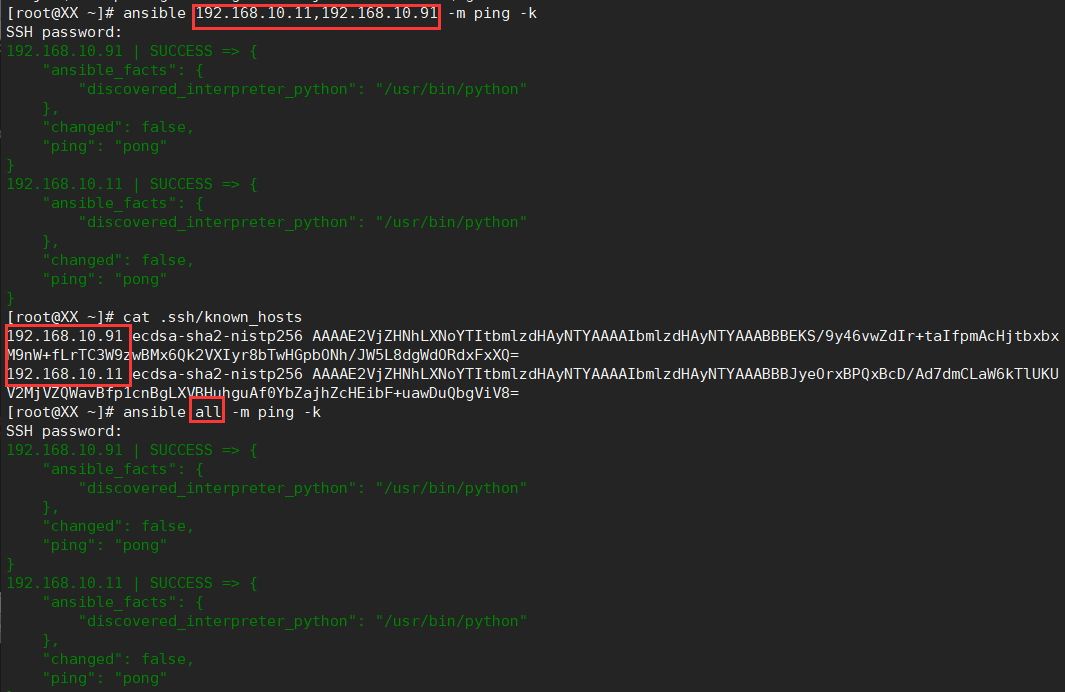




调优：/etc/ssh/sshd\_config文件中**Use DNS no**;

ansible websrvs -m ping -k





**ansible配置文件**

Ansible配置文件：/etc/ansible/ansible.cfg【一般保持默认】

[defaults]

#inventory = /etc/ansible/hosts #主机列表配置文件

#library = /usr/share/my\_modules #库文件存放目录

#remote\_tmp = $HOME/.ansible/tmp #临时py命令文件存放在远程主机目录

#local\_tmp = $HOME/.ansible/tmp #本机临时py命令的存放目录

#forks = 5 #默认并发数

#sudo\_user = root #**默认sudo用户**

#ask\_sudo\_pass = True #每次执行ansible命令是否询问sudo密码

#ask\_pass = True #

#remote\_port = 22

**host\_key\_checking = false** #不检查对应服务器的host\_key

不用在.ssh/known\_hosts留下key就可以直接连接

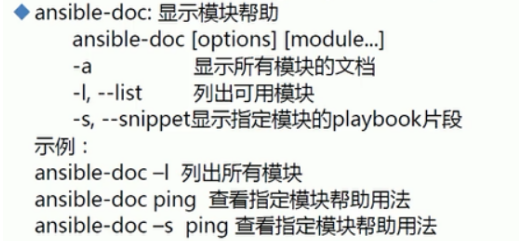
**log\_path=/var/log/ansible.log** #推荐记录日志

**Ansible系列命令**

ls /usr/bin/ansible\* #查看ansible命令

ansible、ansible-doc、ansible-playbook、ansible-vault、ansible-console、ansible-galaxy、ansible-pull

①ansible-doc：显示模块帮助



②ansible通过ssh实现配置管理、应用部署、任务执行等功能

建议配置ansible端**基于密钥认证方式**联系各被管理节点

**基于key验证【ansible机器：**192.168.10.11**】**

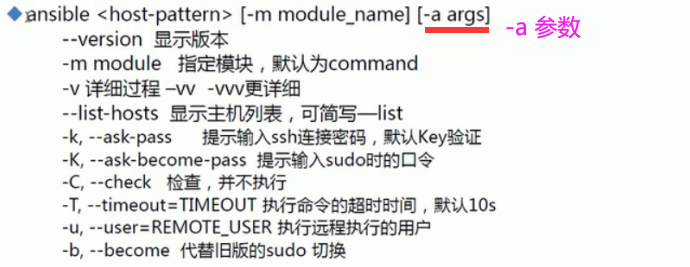
192.168.10.11>#ssh-keygen

192.168.10.11>#ssh-copy-id 目标主机IP

验证：ansible all -m ping

③**ansible <host-pattern> [-m module\_name] [-a args]**

**-a args : -a 参数**



ansible websrvs --list #查看websrvs的主机列表

ansible websrvs -m ping -vvv #查看执行ping模块的详细过程

ansible dbsrvs -m ping -k #输入root用户的ssh密钥

ansible 192.168.10.11 -m ping -u wang -k #以wang用户执行ping存活检测

ansible 192.168.10.11 -a 'cat /etc/fstab ' -u wang -k

ansible 192.168.10.11 -m command -a 'ls /root ' **-u wang -k -b -K**

#以wang登录11机器，再以sudo root执行ls /root命令

192.168.10.11必备条件：

①useradd wang;passwd wang #创建用户及密码

ansible websrvs -m shell -a 'useradd wang'

ansible websrvs -m shell -a 'echo 123|passwd --stdin wang'

②sudo授权visudo中 取消注释**%wheel ALL=(ALL) NOPASSWD: ALL**

③**usermod -aG wheel wang**; #将wang加入nopasswd组内

验证：

ansible dbsrvs -m command -a ‘ls /root’ -u wang -k -b

另一种方法：

ansible>#ssh-keygen

ansible>#ssh-copy-id 目标 IP

验证

ansible dbsrvs -m command -a 'ls /root'

a)ansible的Host-pattern【**匹配主机列表**】

**all**：表示所有inventory中的所有主机

e.g ansible all -m ping

**,**：多个不连续的主机IP

e.g anisble 192.168.10.12,192.168.10.13,192.168.10.14 -m ping

**\***：通配符

e.g ansible 192.168.1.\* -m ping

ansible "srvs" -m ping

**或关系:**

ansible 'websrvs:appsrvs' -m ping

ansible '192.168.1.10:192.168.1.20' -m ping

**逻辑与:&**

ansible 'websrvs:&dbsrvs' -m ping #在websrvs组且在dbservs组中的主机

**逻辑非:! 【注意：此处为单引号】**

ansible 'websrvs:!dbsrbs' -m ping #在websrvs组，但不在dbsrvs组中的主机

**综合逻辑**

ansible 'websrvs**:**dbsrbs**:&**appsrvs**:!**ftpsrvs' -m ping

**正则表达式**

ansible '~(web|db).\*\.magedu\.com' -m ping

**ansible命令执行过程**

①加载自己配置文件，【默认/etc/ansible/ansible.cfg]

②加载自己对应的模块文件，如command

③通过ansible将模块或命令生成对应的临时py文件，并将该文件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执行

⑤执行并返回结果

⑥删除临时py文件，sleep 0退出

**执行状态**：

**绿色**：执行成功且不需要做改变的操作【**无变更**】

**黄色**：执行成功且对目标主机做变更【**有变更**】

**红色**：执行失败

ansible all -m ping -vvv #查看执行权限

**Ansible常用模块**

【ansible端】/data/f1.sh脚本中 chmod +x /data/f1.sh

#!/bin/bash

hostname

①**command模块**：在远程主机执行命令【**默认**模块，可忽略 -m 选项】

注：此命令**不支持$varname、<、>、|、&**等命令，用shell模块实现

ansible all -a 'cat /etc/fstab'

ansible appsrvs -m command -a 'service nginx start'

ansible appsrvs -m command -a 'chdir=/boot ls' #切换到boot目录下执行ls命令

②**shell模块**：和command相似，【**推荐使用shell**】

ansible appsrvs -m shell -a 'echo magedu|passwd --stdin wang'

ansible appsrvs -m shell -a 'echo $URl >aa.txt'

注：类似 cat /tmp/aa.txt|awk -F'|' '{print $1,$2}' &>/tmp/bb.txt等复杂命令会失败。

解决：写到脚本中，再copy到远程，执行，再把结果拉回执行命令的机器

ansible all -m shell -a 'getenforce'

ansible all -m shell -a 'reboot'

③**script模块**：运行脚本

ansible websrvs -m script -a '/data/f1.sh'

ansible all -m script -a '/root/ansible/host.sh'

【ansible端】/root/ansible/host.sh脚本中 chmod +x /root/ansible/host.sh

#!/bin/bash

hostname

④**copy模块**：从服务器复制文件到客户端

ansible srv -m copy -a "content='hello\nworld\n' dest=/tmp/f1.sh"

利用内容，直接生成目标文件

ansible srv -m copy -a "src=/root/f1.sh dest=/tmp/f2.sh owner=wang mode=600 **backup=yes**"

若目标存在，默认覆盖，此处指定先备份

ansible all -m copy -a 'src=/etc/shadow dest=/data/ mode=000 owner=wang'

⑤**fetch模块**：从客户端取单个文件至服务器端，与copy相反，目录可先tar压缩

ansible srv -m fetch -a "src=/root/a.sh dest=/data/scripts"

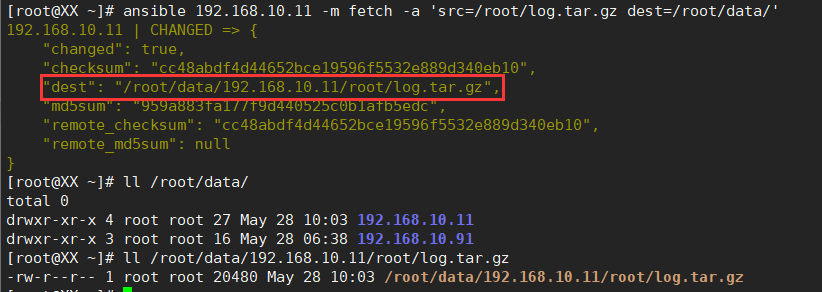
#单个文件

ansible all -m fetch -a 'src=/var/log/messages dest=/root/data'

#多个文件：**先打包压缩tar，再fetch**

ansible 192.168.10.11 -m shell -a '**tar cf** log.tar.gz /var/log/\*.log' #压缩文件

ansible 192.168.10.11 -m fetch -a 'src=/root/log.tar.gz dest=/root/data/'



⑥**file模块**：设置文件属性ansible-doc file

state=**absent**：删除文档/目录

**dire**ctory：创建目录

**touc**h：创建文件

**hard**：创建硬链接

**link：**创建软链接

**diff**： **stat：**

**file**

ansible srv -m file -a "path=/root/a.sh owner=wang mode=755"

文件：

ansible 192.168.10.11 -m file -a 'name=/data/f3 **state=touch**' #创建f3文件

ansible 192.168.10.11 -m shell -a 'ls -al /data/f3' #查看f3文件是否存在

ansible 192.168.10.11 -m file -a 'name=/data/f3 **state=absent**' #删除f3文件



链接：

ansible 192.168.10.11 -m file -a "src=/etc/fstab dest=/data/fstab.link state=**link**" #创建软链接

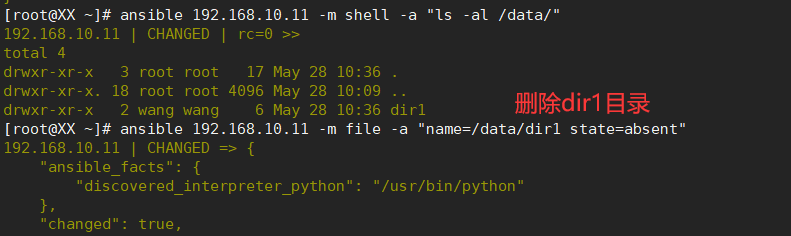


文件夹：

ansible 192.168.10.11 -m file -a "name=/data/dir1 state=**directory** owner=wang group=wang"

ansible 192.168.10.11 -m file -a "name=/data/dir1 **state=absent**" #删除dir1目录

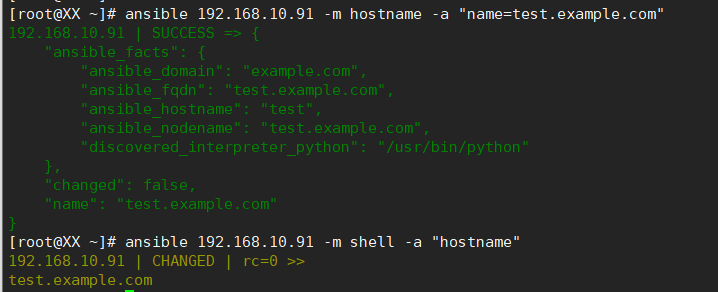
或：ansible 192.168.10.11 -m shell -a "rm -rf /data/dir1" #删除dir1目录



⑦**Hostname模块**：管理主机名ansible-doc hostname

ansible node1 -m hostname -a "name=websrv"

ansible 192.168.10.91 -m hostname -a "name=test.example.com"



⑧**Cron模块**：计划任务 ansible-doc cron

支持时间：**minute(分钟)、hour(小时)、day(日期)、month(月份)、weekday(周几)**

minute=**\*** #每分钟 weekday=**1,3,5** #每周1、3、5

hour=**\*/2** #每2个小时 day=**2-12** #每月的2-12号

ansible srv -m cron -a "minute=\*/5 job='/usr/sbin/ntpdate 172.16.0.1&>/dev/null name=Synctime'" #创建任务

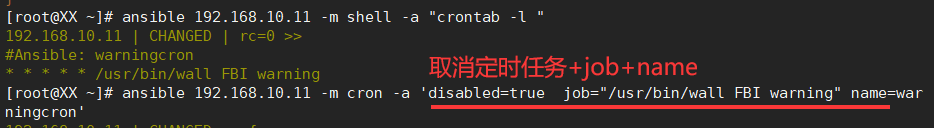
ansible srv -m cron -a "state=**absent** name=Synctime" #删除任务

创建任务

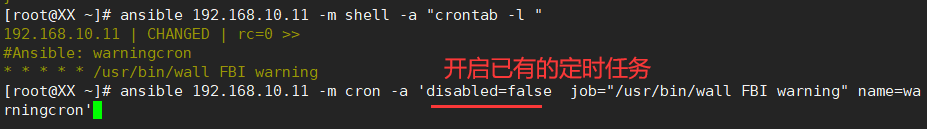
ansible srv -m cron -a 'minute=\* job="/usr/bin/wall FBI warning" name=warningcron'

ansible 192.168.10.11 -m cron -a '**disabled=false** job="/usr/bin/wall FBI" name=test'

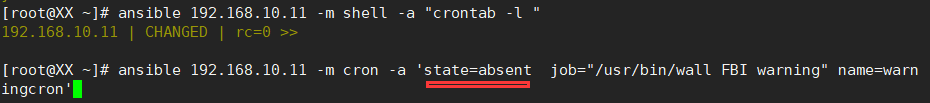
ansible 192.168.10.11 -m cron -a '**disabled=true** job="/usr/bin/wall FBI warning" name=warningcron' #注释定时任务



ansible 192.168.10.11 -m cron -a '**disabled=false** job="/usr/bin/wall FBI warning" name=warningcron' #开启定时任务



ansible 192.168.10.11 -m cron -a '**state=absent** job=**"**/usr/bin/wall FBI warning**"** name=warningcron' #删除定时任务



⑨**Yum模块**：管理包 ansible-doc yum

state=**install**【安装：**present、installed、latest**】--默认

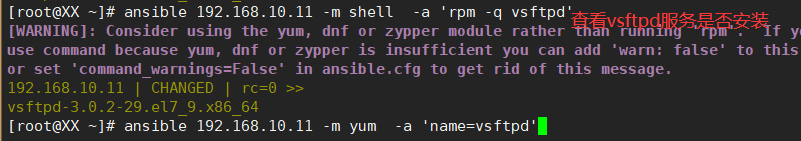
**remove**【删除：**absent、removed**】

ansible srv -m yum -a 'name=httpd state=**latest**' #安装最近版本的http

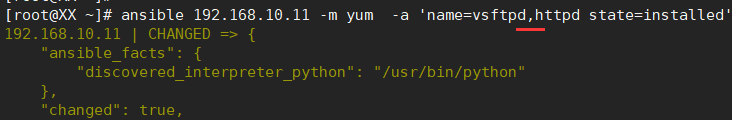
ansible srv -m yum -a 'name=httpd state=**absent**' #删除http

ansible 192.168.10.11 -m yum -a 'name=vsftpd' #-默认present安装

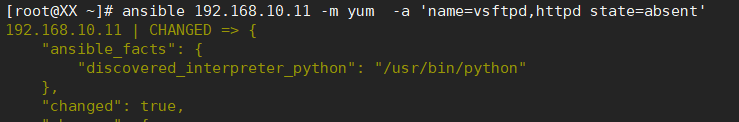
ansible 192.168.10.11 -m shell -a '**rpm -q** vsftpd' #查看vsftpd服务是否安装



ansible 192.168.10.11 -m yum -a 'name=**vsftpd,httpd** state=installed' #安装多个软件



ansible 192.168.10.11 -m yum -a 'name=vsftpd,httpd state=**absent**' #卸载多个软件



安装本地的软件包：先复制到远端机器

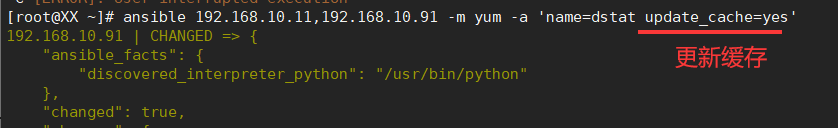
ansible 192.168.10.11,192.168.10.91 -m copy -a 'src=/data/vsftpd-3.0.2-28.el7.x86\_64.rpm dest=/root/' #先复制vsftpd



ansible 192.168.10.11,192.168.10.91 -m yum -a 'name=/root/vsftpd-3.0.2-28.el7.x86\_64.rpm'

ansible 192.168.10.11,192.168.10.91 -m yum -a 'name=/root/vsftpd-3.0.2-28.el7.x86\_64.rpm **disable\_gpg\_check=yes**' #忽略gpg\_check检查的安装

ansible 192.168.10.11,192.168.10.91 -m yum -a 'name=dstat **update\_cache=yes**' #更新缓存



⑩**Service模块**：管理服务

state=**stopped(停止)、started(打开)、reloaded(加载)、restarted(重新打开)、**

**enabled(开启启动):yes--开机自启动，false--关闭开机自启动**

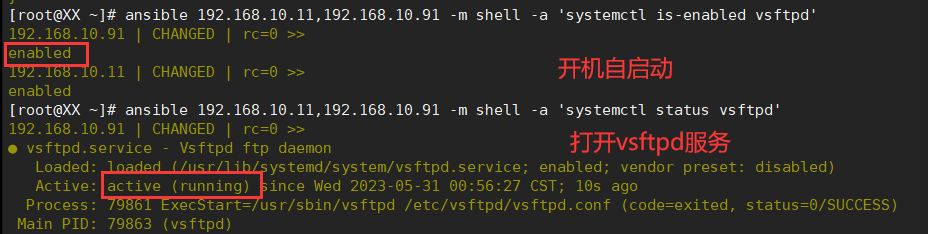
ansible srv -m service -a 'name=httpd state=stopped' #关闭http服务

ansible srv -m service -a 'name=httpd state=started' #打开http服务

ansible 192.168.10.11,192.168.10.91 -m service -a 'name=vsftpd state=started **enabled=yes**'

ansible 192.168.10.11,192.168.10.91 -m shell -a 'systemctl is-enabled vsftpd'

ansible 192.168.10.11,192.168.10.91 -m shell -a 'systemctl status vsftpd'



⑪**User模块**：管理用户

-r 等同--remove：连同家目录一起删除

system=yes|no ：是否为系统账号

ansible srv -m user -a 'name=user1 comment="test user" uid=88 home=/app/user1 group=root'

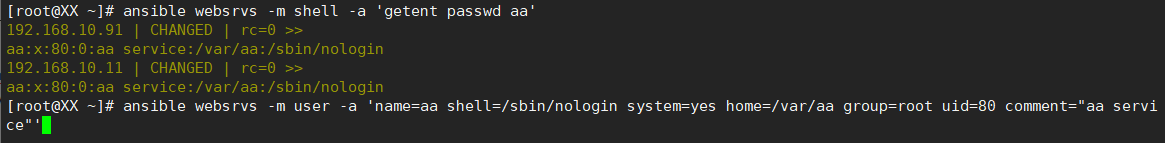
ansbile srv -m user -a 'name=sysuser1 system=yes home=/app/sysuser1'

ansible srv -m user -a 'name=user1 state=absent remove=yes'

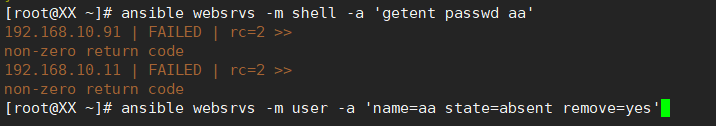
#删除用户及家目录等数据

创建账户

ansible websrvs -m user -a 'name=aa shell=/sbin/nologin system=yes home=/var/aa group=root uid=80 comment="aa service"'



ansible websrvs -m user -a 'name=aa state=absent **remove=yes**' #删除aa用户及其家目录



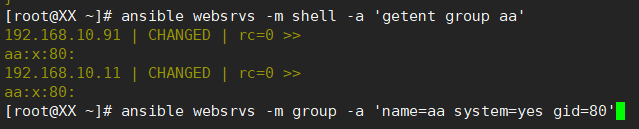
⑫**Group模块**：管理组

ansbile srv -m group -a 'name=testgroup system=yes'

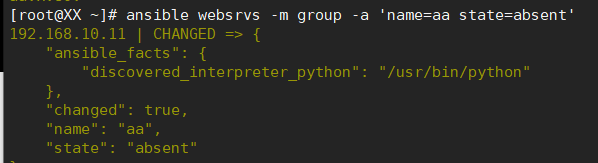
ansbile srv -m group -a 'name=testgroup state=absent' #删除testgroup用户组

ansible websrvs -m group -a 'name=aa system=yes gid=80' #创建aa系统组且gid为80

ansible websrvs -m shell -a 'getent group aa'



ansible websrvs -m group -a 'name=aa state=absent'



**Ansible系列命令**

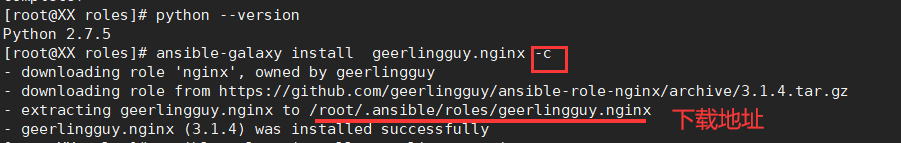
**① ansible-galaxy**

连接**https://galaxy.ansible.com**下载相关的roles

ansible-galaxy list #列出所有已安装的galaxy

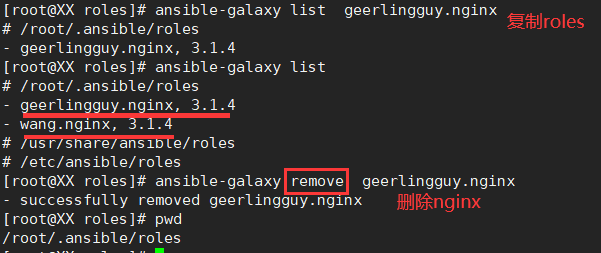
ansible-galaxy **install** geerlingguy.nginx **-c** #安装 【使用-c参数，忽略证书】

下载地址：**/root/.ansible/roles/geerlingguy.nginx**



ansible-galaxy list #列出所有已安装的galaxy

ansible-galaxy **remove** geerlingguy.nginx #卸载



或：删除 rm -rf /root/.ansible/roles/geerlingguy.nginx

②**ansible-pull**： 推送命令至远程，效率提升。【运维要求高】

③**ansible-vault**：功能：管理加/解密yml文件

**encrypt(加密)、decrypt(解密)、edit(编辑加密文件)、**

**rekey(修改口令)、view(查看)、create**(创建新文件)

ansible-vault **encrypt** hello.yml #加密文件

**decrypt** hello.yml #解密文件

**view** hello.yml #查看文件

**edit** hello.yml #编辑加密文件

**rekey** hello.yml #修改口令

**create** hello.yml #创建新文件

**④ansible-console**： 版本2.0+新增，可交互执行命令，支持tab

**ansible-console**

**root@test(2)[f:10]$**

格式：执行用户@当前操作的主机组(当前组的主机数)[f:并发数]$

设置并发数：forks n e.g forks 2

切换组： cd 主机组 e.g cd web

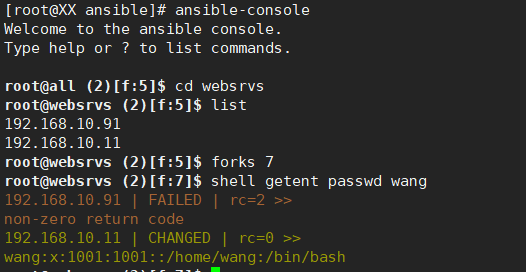
列出当前主机列表：list

列出所有的内置命令： ?或help

root@all(2)[f:5]$list

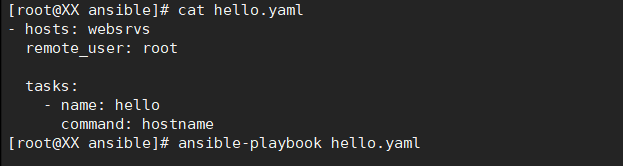
root@all(2)[f:5]$cd appsrvs

root@appsrvs(2)[f:5]$yum name=httpd state=present



**⑤Ansible-playbook**

ansible-playbook hello.yml



**Playbook**

由一个或多个**play**组成的列表

主要功能：事先将归为一组的主机装扮成事先通过ansible中**task定义好的角色**

task是调用ansible的module，将多个play组织在一个playbook中，**按序执行**

playbook采用**YAML**语言编写

YAML特性：

可读写性好、交互性好、易于实现、基于流处理、表达性强，扩展性好、

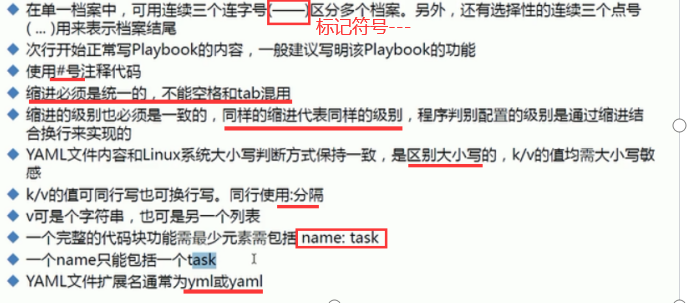
一致的信息模型、实现语言的数据类型

**YAML语言简介**

用三个连字符**---**：标记符号【区分多个档案】

**#号开头：为注释**

**一个name只能包括一个task，**若有多个task，则只执行**最后一个task**



**List：列表**，其所有元素均使用**-**打头

e.g # A list of fruits

- Apple

- Orange

- Mango

**Dictionary：字典**，通常由多个**key: value**构成

e.g # student info

name: lisi

age: 22

sex: man

或将key: value放置于**花括号{}**中进行表示，用**逗号,**分隔多个key: value

{name: lisi**,**age: 22**,**sex: man}

Yaml语法表达：

结构：通过**空格**来展示

序列：用**-**代表

Map的键值对用**:** 分隔



**playbook核心元素**： 参考：Ansible中文权威指南：www.ansible.com.cn

①**Hosts**：执行的远程**主机列表**

②**Tasks**：任务集【**操作**】

③**Varniables**：内置变量或自定义变量再playbook中调用【**变**量】

④**Templates**：模板，替换模板文件中的变量实现简单逻辑的文件【**模板**】

⑤**Handlers**：和notity结合，由特定条件触发的操作，满足条件方可执行，否则不执行

⑥**tags**：**标签** 指定某条任务执行，用于选择运行playbook中的部分代码。

ansible具有幂等性，会自动跳过没有变化的部分。通过tags跳过确实没有变化的代码

ansible-playbook **-t tagsname** useradd.yml

①**Hosts**：指定要执行指定任务的主机【**必须事先定义在主机清单中**】

playbook的每一个play目的是：让某个或某些主机以某个指定用户身份取执行任务

a)形式：

one.example.com

one.example.com**:**two.example.com

192.168.10.11

192.168.10.\*

192.168.10.12:333

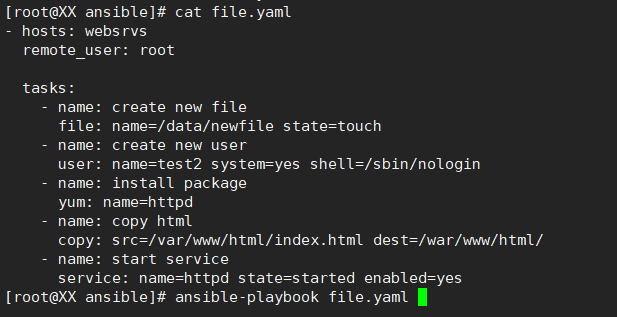
www**[01:100]**.example.com

b)websrvs**:**dbsrvs #两个组的并集

websrvs**:&**dbsrvs #两个组的交集

websrvs**:!**dbsrvs #在websrvs组，但不在dbsrvs组

e.g - hosts: websrvs:dbsrvs



echo "hello,world" >>/var/www/html/index.html

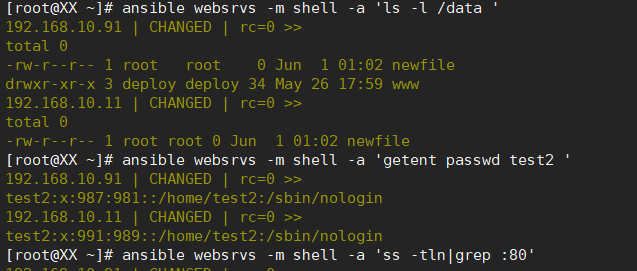
ansible-playbook file.yaml

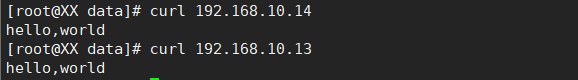
检查：

ansible websrvs -m shell -a 'ls -l /data ' #检查文件

ansible websrvs -m shell -a 'getent passwd test2 ' #检查用户

ansible websrvs -m shell -a 'ss -tln|grep :80' #检查服务端口





**remote\_user**：可用于Host和task中，或通过指定其通过sudo方式在远程主机执行任务。

用于play全局或某任务。甚至在sudo时使用sudo\_user指定sudo时切换的用户

- hosts: websrvs

remote\_user: root

tasks:

- name: test connection

ping

remote\_user: magedu

sudo: yes 默认sudo为root

sudo\_user: wang sudo为wang

task列表和action

**play的主体部分：task list**。

task list中的各任务**按次序**逐个在hosts中指定主机上执行。【自上而下执行playbook】

即在所有主机上完成第一个任务后再开始第二个。【**串行化**】

若中途发生错误，**所有已执行任务都将回滚**。即更正playbook后重新执行即可

**task**的目的：指定参数执行模块，而在模块参数中可用使用变量。

模块执行时**幂等**的，即多次执行是安全的，其结果均一致

每个task都应该有其name，用户playbook执行结果输出。建议详细描述执行任务步骤

若未提供name，则action的结果将用于输出

①tasks：任务列表

格式： a)action: module arguments

b)module: arguments【模块：参数 --建议使用】

注：shell和command模块后面**跟命令，而非key=value**

如：tasks:

- name: disable selinux

command: /sbin/setencorce 0

②某任务状态在运行后为changed时，可通过**notify通知给相应的handlers**

③任务可通过tag打标签，而后再ansible-playbook命令+ **-t tagname** 指定进行调用

④若命令或脚本的退出码不为0，则用如下方式替代【**忽略错误继续执行**】

task:

- name: run this command and ignore the result

shell: /usr/bin/command **||/bin/true**

或使用ignore\_errors来忽略错误信息：

tasks：

- name: run this command and ignore the result

shell: /usr/bin/command

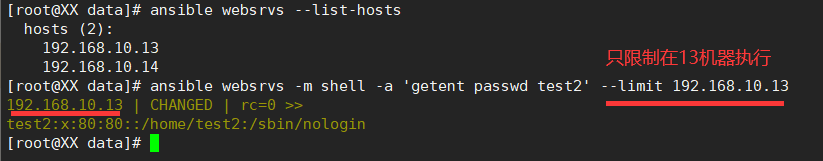
**ignore\_errors: True**

**运行playbook**

格式：ansible-playbook <filename.yml>... [options]



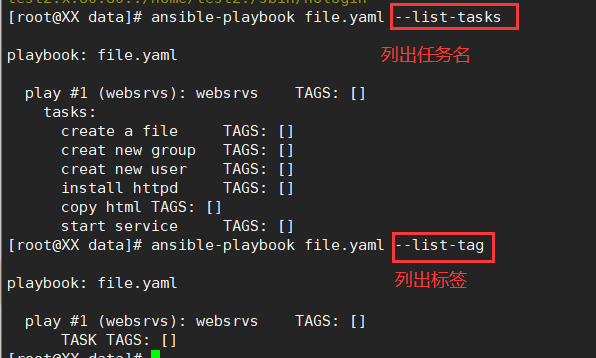
ansible websrvs -m shell -a 'getent passwd test2' **--limit** 192.168.10.13 #只在13机器执行

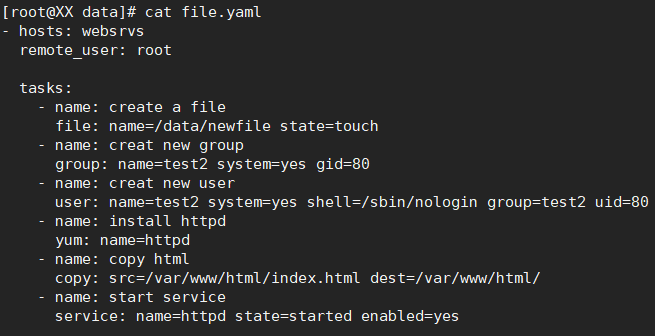


ansible-playbook file.yaml --list-hosts #列出file.yaml中的主机

ansible-playbook file.yaml --list-tasks #列出file.yaml中的任务名

ansible-playbook file.yaml --list-tags #列出file.yaml中的标签



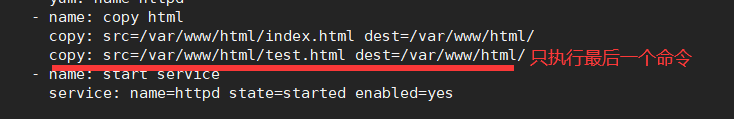


Playbook示例【httpd.yaml】

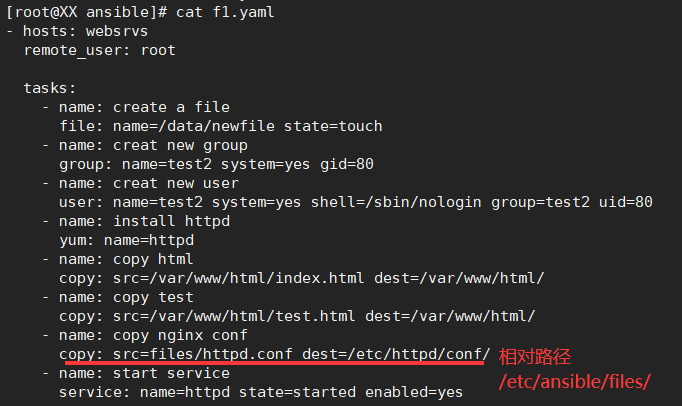
ansible-playbook -C file.yaml #检查file.yaml的语法错误

等同于：ansible-playbook file.yaml --check

**一个name只能包括一个task，**若有多个task，则只执行**最后一个task**



files/httpd.conf 等同于 cd /root/ansible/files/httpd.conf



**Handlers + notify**结合使用触发条件

Handlers：**触发器**，同task并列关系，**监控**某个或多个task的动作【task列表】

与task没有本质上的不同，用于当**关注的资源发生变化**时，才采取一定的操作

Notify此action可用于在每个play的**最后被触发**，避免多次有改变发生时每次都执行指定的操作，仅在**所有变化发生完成后**一次性的执行指定操作。

在notify中列出的操作成为handler，即notify中调用handler中定义的操作

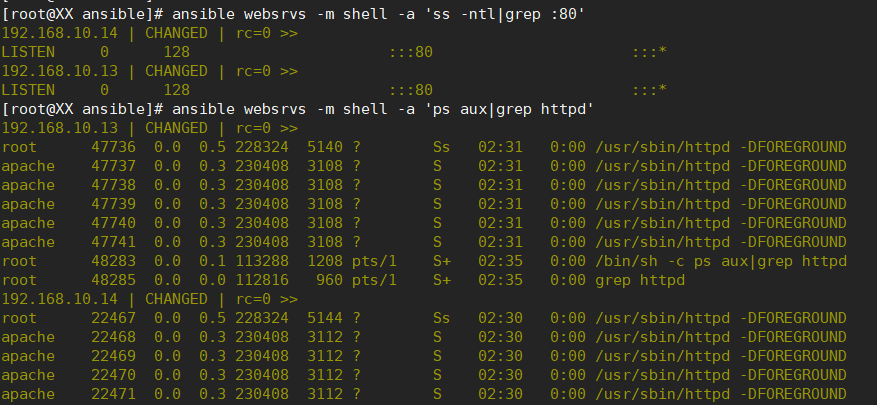
cp /etc/httpd/conf/httpd.conf ansible/files/ #复制httpd配置文件到files文件夹

ansible websrvs -m yum -a 'name=httpd state=absent' #卸载httpd



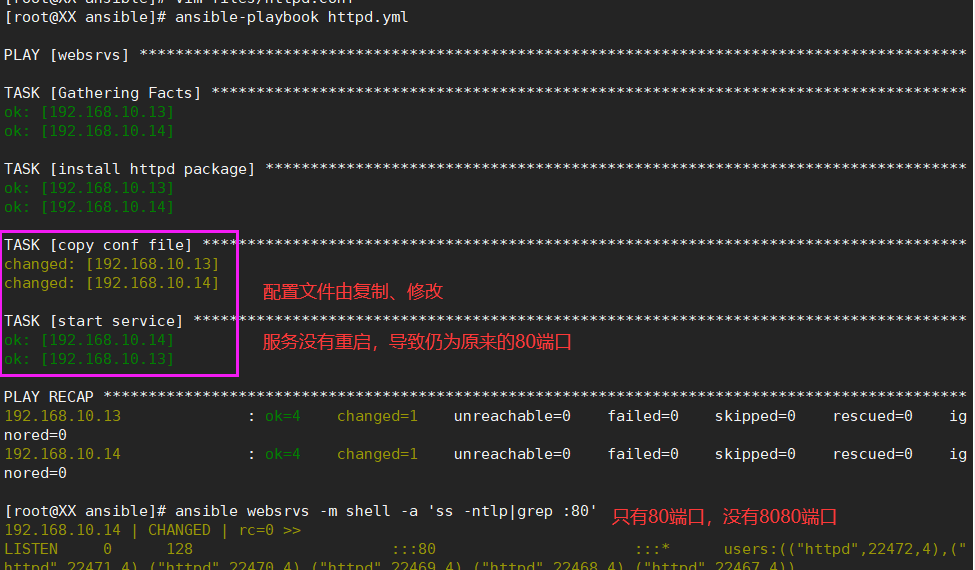
ansible websrvs -m shell -a 'ss -ntl|grep :80'

ansible websrvs -m shell -a 'ps aux|grep httpd'

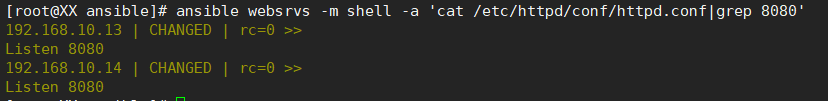


修改files/httpd.conf的监听端口为：Listen 8080

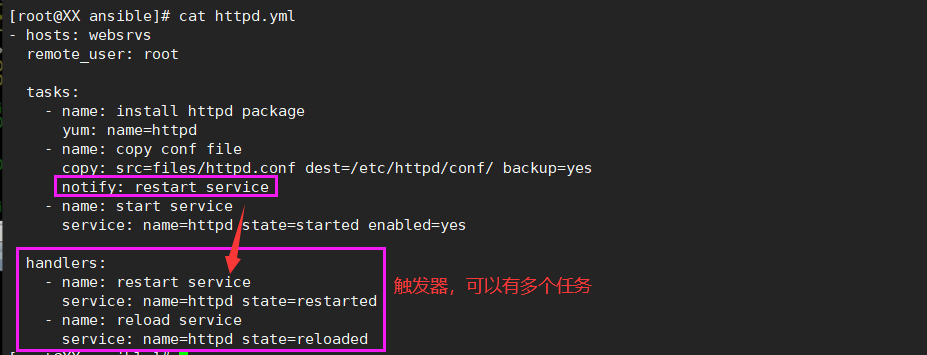
ansible-playbook httpd.yml

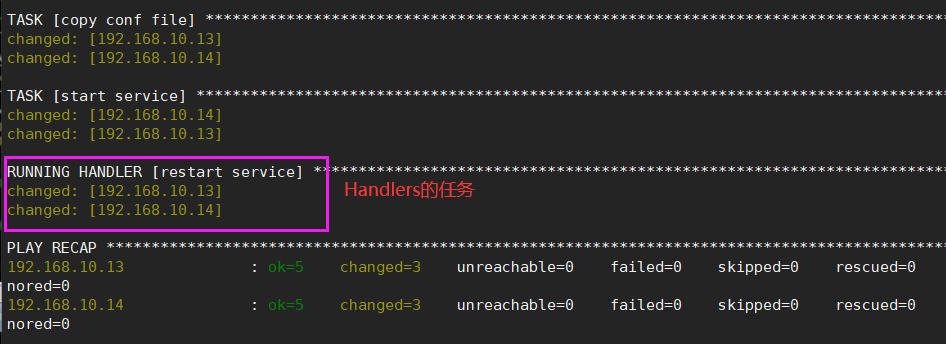


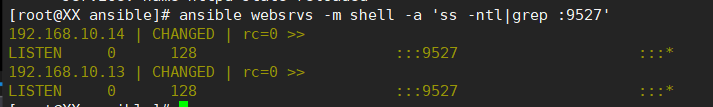
ansible websrvs -m shell -a 'cat /etc/httpd/conf/httpd.conf|grep 8080' #httpd已修改为8080

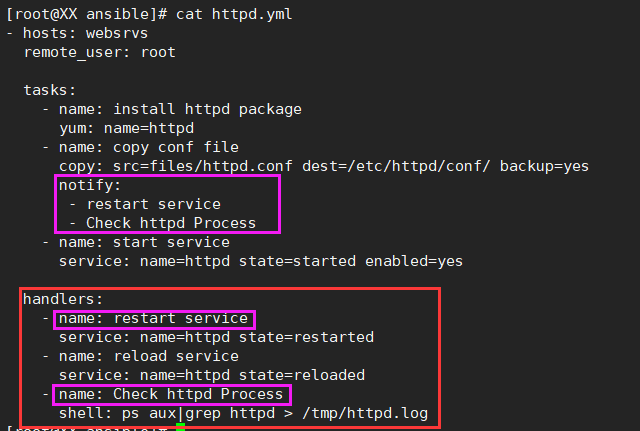


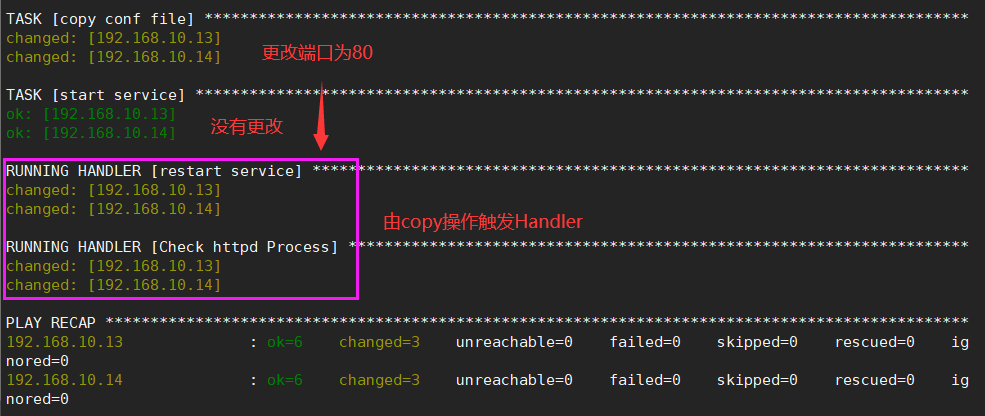
再次修改files/httpd.conf的监听端口为：Listen 9527

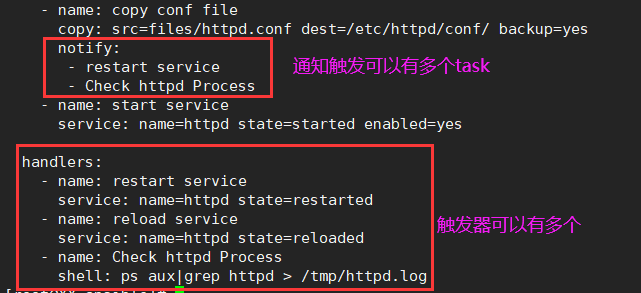












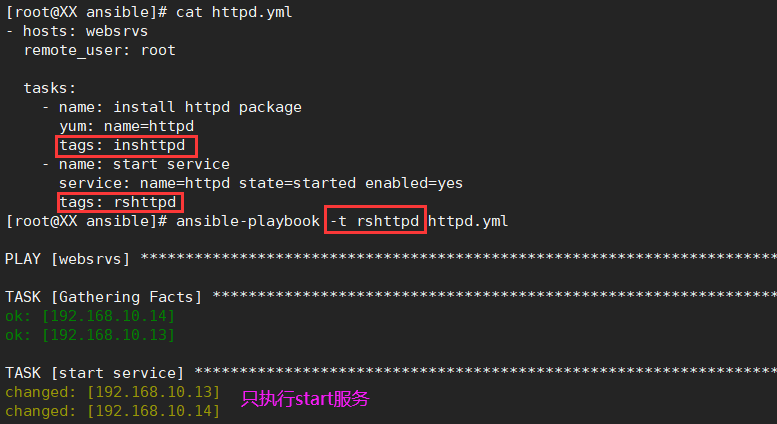
**playbook中tags使用**

格式：tags: tagsname

应用：ansible-playbook **-t tagsname** file.yaml

单独执行某个标签

ansible-playbook -t rshttpd httpd.yml



ansible websrvs -m yum -a 'name=httpd state=absent' #卸载http服务

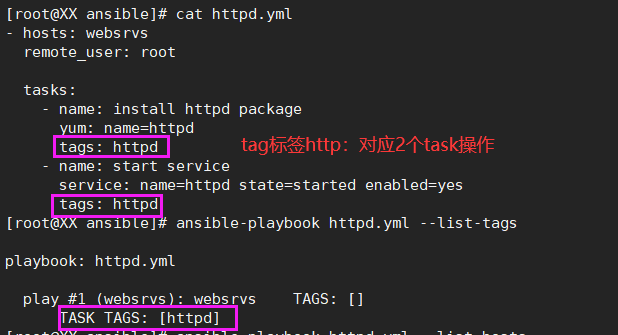
ansible websrvs -m shell -a 'rpm -q httpd' #验证httpd包是否存在

执行多个标签，用**,号**分隔

ansible-playbook **-t inshttpd,rshttpd** httpd.yml

多个动作可以共有同一个标签

ansible-playbook -t httpd httpd.yml



------06.27看完

**Playbook中变量使用**

变量名：仅能由字母、数字和下划线组成，且只能以字母开头

变量定义： key=value

e.g http\_port=80

变量调用方式：

a)通过**{{variable\_name}}**调用变量，且变量名前后必须有空格，

有时用"{{variable\_name}}"才生效

b) ansible-playbook **-e 选项**指定

ansible-playbook test.yml -e "hosts=www user=magedu"

优先级：**命令行的变量[-e ] >普通变量>公共(组)变量**

变量来源：

①**ansible setup facts** 远程主机的所有变量都可以直接调用【系统自带变量】

ansible websrvs -m setup |grep less #查看所有系统状态信息

ansible 192.168.10.14 **-m setup -a** '**filter=**\*address\*' #**过滤**address字段

ansible 192.168.10.14 -m setup -a 'filter=\*eno16777736\*'

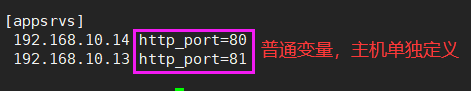
②在**/etc/ansible/hosts**中定义

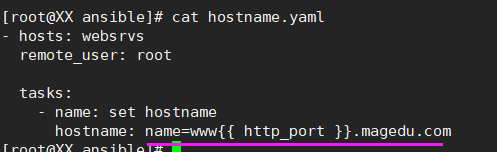
普通变量：主机组中主机单独定义，优先级高于公共变量

公共(组)变量：针对主机组中所有主机定义统一变量

cat **/etc/ansible/hosts**

ansible-playbook hostname.yaml



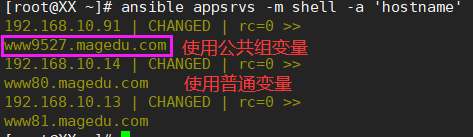




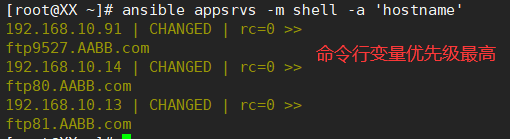
改进：

cat /etc/ansible/hosts





ansible-playbook **-e 'nodename=ftp domainname=AABB.com'** hostname.yaml

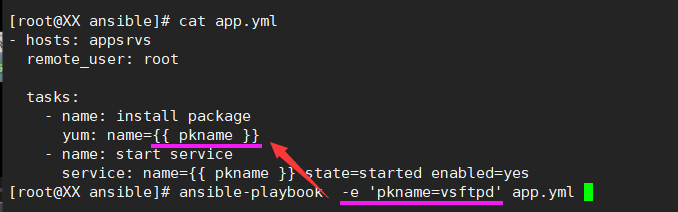


③通过命令行指定变量，优先级最高

ansible-playbook **-e varname=value**

**-e** 赋值操作 vsftpd的端口为21，**{{ pkname }}为变量名**

ansible-playbook **-e 'pkname=vsftpd'** app.yml



ansible appsrvs -m shell -a 'ss -ntl|grep :21'

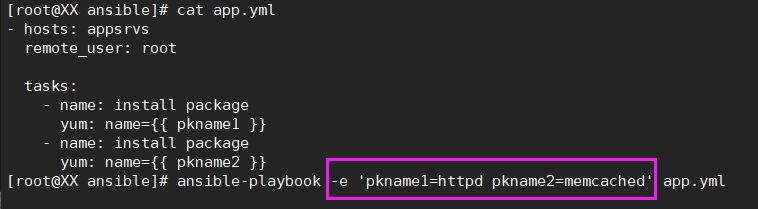
ansible appsrvs -m shell -a 'rpm -q vsftpd'

多个变量：

ansible-playbook **-e 'pkname1=httpd pkname2=memcached'** app.yml

ansible appsrvs -m shell -a 'rpm -q memcached httpd'

ansible appsrvs -m shell -a 'yum -y remove memcached httpd' #卸载memcached和httpd



④在playbook中定义变量

**vars:**

**- var1: value1**

**- var2: value2**

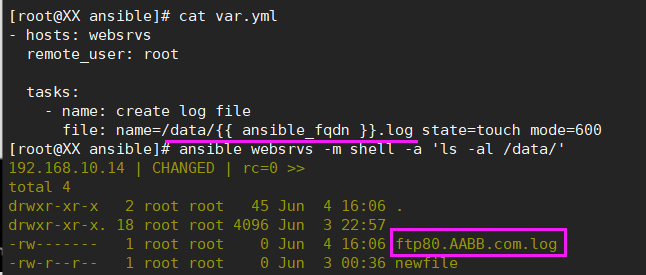


ansible appsrvs -m shell -a 'rpm -q 'httpd vsftpd'

⑤在role中定义

ansible all -m setup |grep fqdn 获得：**ansible\_fqdn**": "ftp80.AABB.com

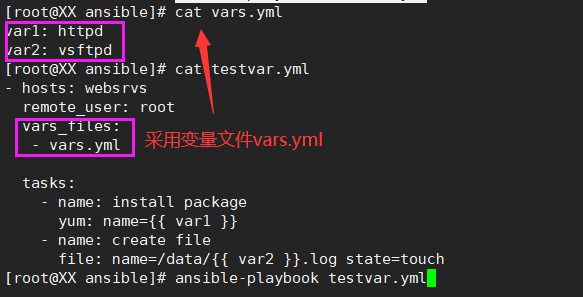
ansible-playbook var.yml

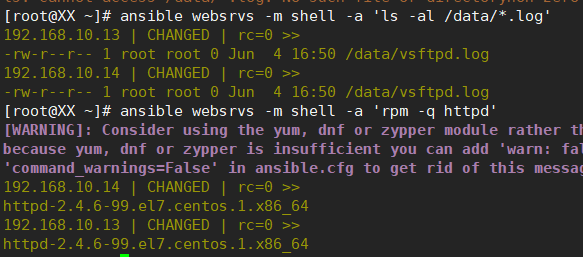


ansible all -m setup |grep mem #查看内存相关系统变量

**使用变量文件**：vars.yml

ansible-playbook testvar.yml





**模板templates**：ansible-doc template

文本文件，嵌套有脚本(使用模板编程语言编写)

**Jinja2**语言，使用字面量，形式：

字符串：使用**单引号或双引号** 布尔型：true/false

数字：整数，浮点数 列表：[item1,item2,...]

元组：(item1,item2,...) 字典：{key1:value1,key2:value2,...}

算术运算： +、-、\*、/、**//(整除**，取整)、%(取模)、**\*\*(幂次方**，指数)

比较操作：==、!=、>、>=、<、<=

逻辑运算：and、or、not 流表达式：For、If、When

注: template模块只能用于ansible-playbook，不能用于ansible命令

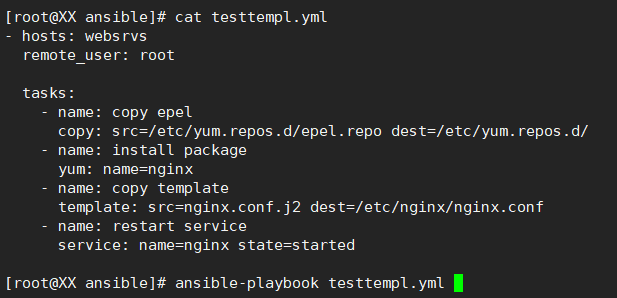
mkdir -pv /root/ansible/templates

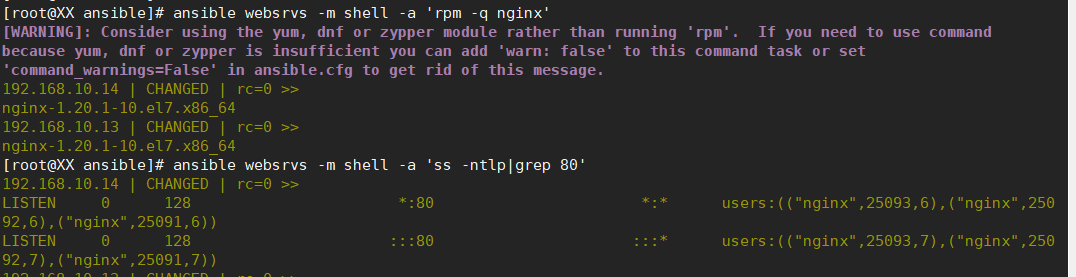
yum -y install nginx

cp /etc/nginx/nginx.conf templates/nginx.conf.j2

/etc/nginx/nginx.conf中worker\_processes auto;

testtempl.yml





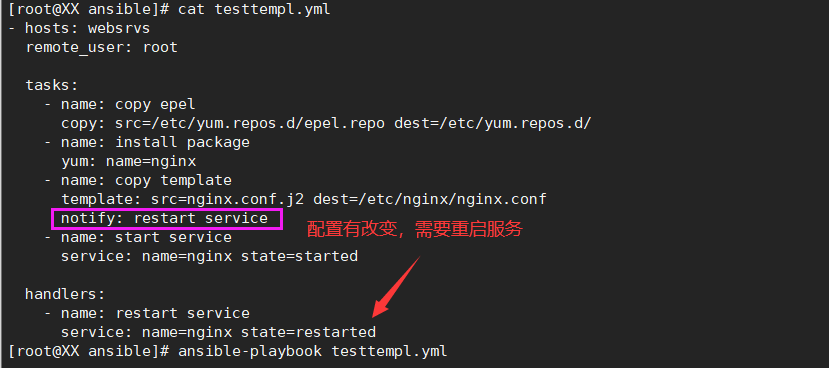
根据机器自适应nginx的进程：

ansible websrvs -m shell -a 'lscpu|grep "CPU(s):"' #获取CPU个数

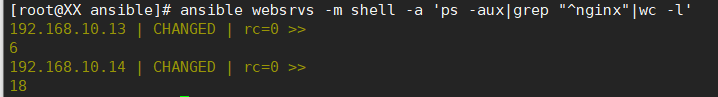
ansible websrvs -m setup|grep vcpu #获取CPU的个数 ansible\_processor\_vcpus: 2

templates/nginx.conf.j2文件中：

worker\_processes **{{ ansible\_processor\_vcpus\*\*2+2 }}**;

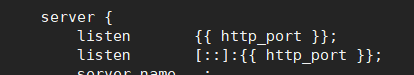


ansible websrvs -m shell -a 'ps -aux|grep "^nginx"|wc -l'

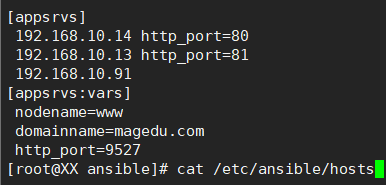


**更改端口变量**：

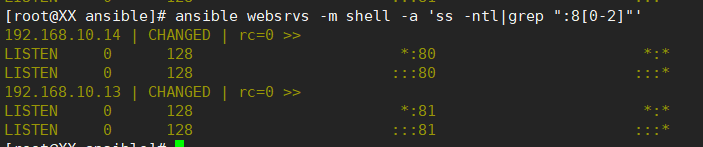
templates/nginx.conf.j2 listen  **{{ http\_port }}**;



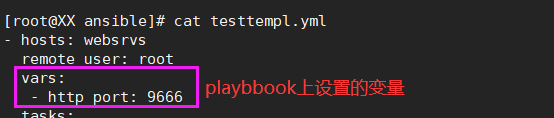
情况①在主机清单上设置变量【 cat /etc/ansible/hosts】

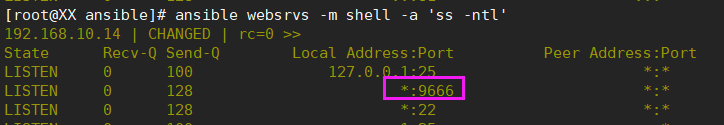


ansible-playbook testtempl.yml



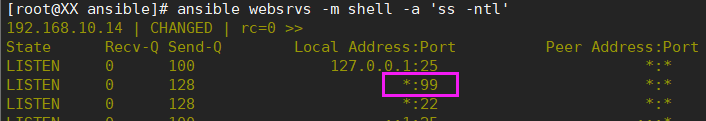
情况② 在playbook上设置变量【cat testtempl.yml】





情况③在命令行设置变量【 -e "http\_port=99"】

ansible-playbook -e "http\_port=99" testtempl.yml



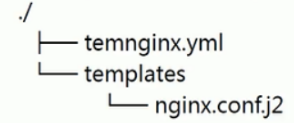
变量的优先级：

**命令行变量[-e 变量] > playbook中的变量[yaml] >主机清单的普通变量 >公共组变量**

Templates：根据**模块文件动态生成对应的配置文件**

**templates文件必须存放于templates目录下，且命名为\*.j2结尾**

**yaml/yml文件需和templates目录**平级，目录结构如下：



Playbook中template变更替换

修改文件nginx.conf.j2下面

worker\_processes {{ ansible\_processor\_vcpus\*\*2+2 }};

- name: copy template

template: src=nginx.conf.j2 dest=/etc/nginx/nginx.conf

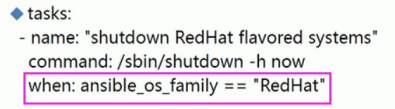
notify: restart service

When：【**条件测试**】

根据变量、facts或此前任务的执行结果作为某task执行与否的前提时要用到条件测试，

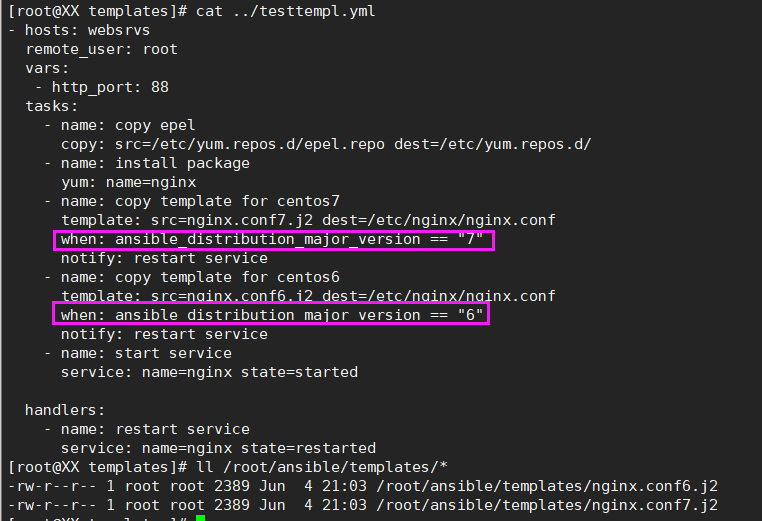
通过when语句实现，在task中使用，jinja2的语法格式

when语句：在task后添加when子句即可使用条件测试；when语句支持jinja2语法

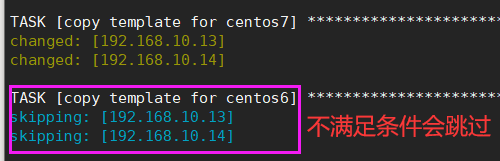


条件判断：

ansible all -m setup -a 'filter="\*distri\*"' 获取ansible\_distribution\_major\_version: "7"

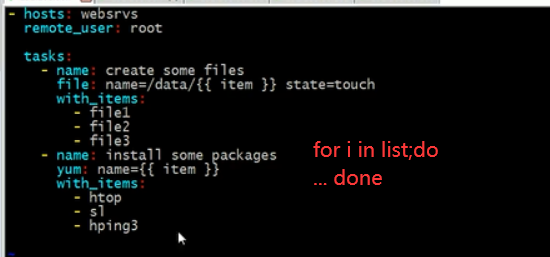


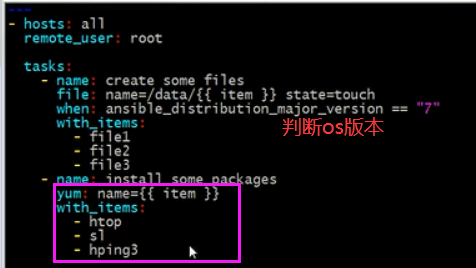
ansible-playbook testtempl.yml



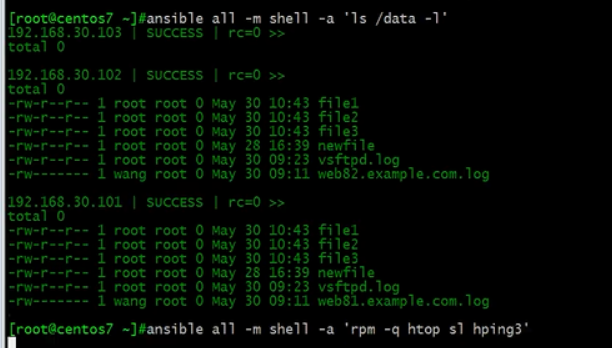


testitem.yml

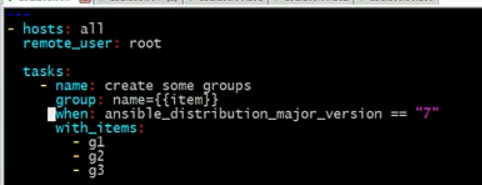




验证：



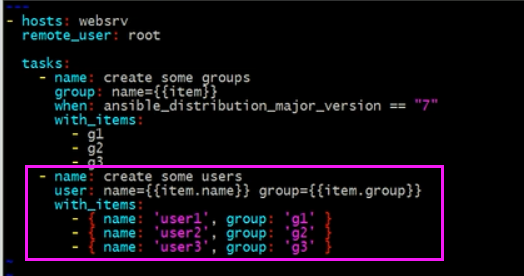
testitem2.yml

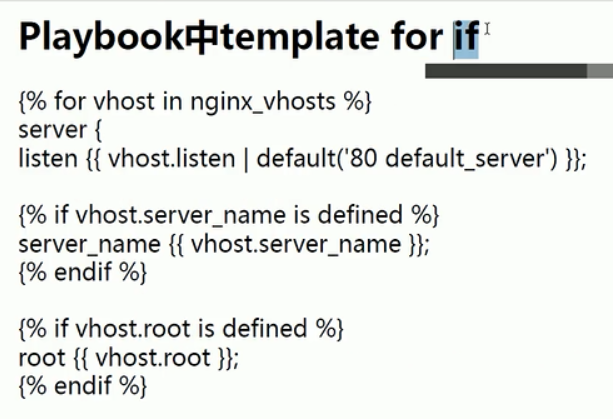


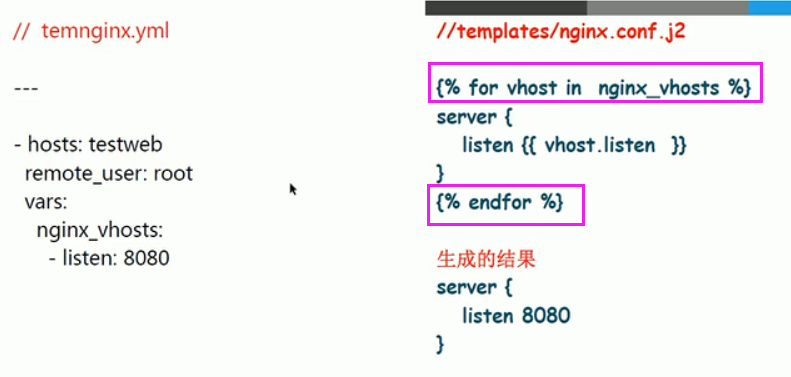


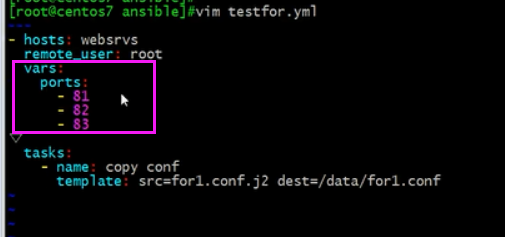
创建三个用户并分别添加到分别组中

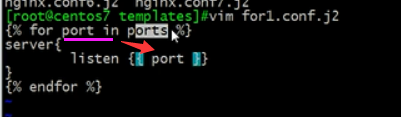
testitem3.yml

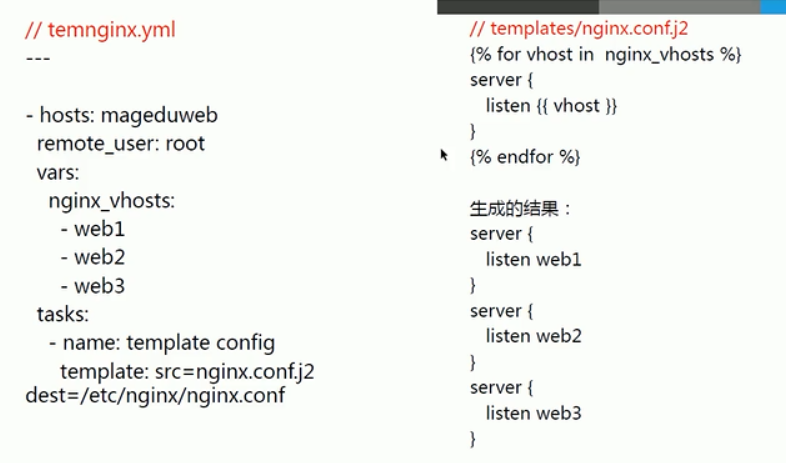






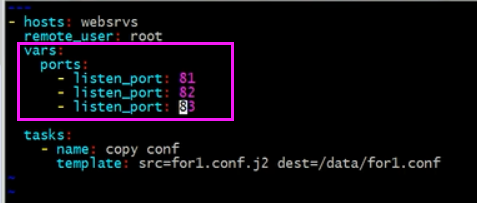


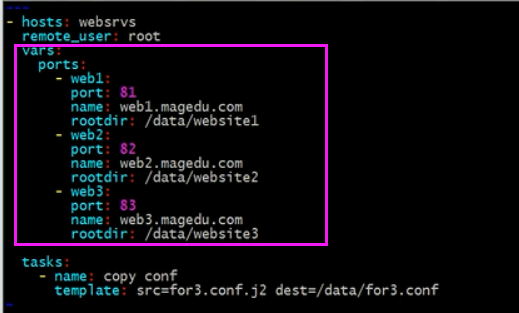




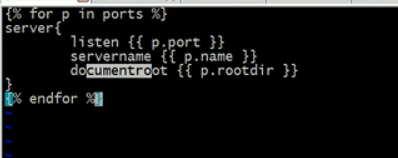
按照字典：

testfor2.yml

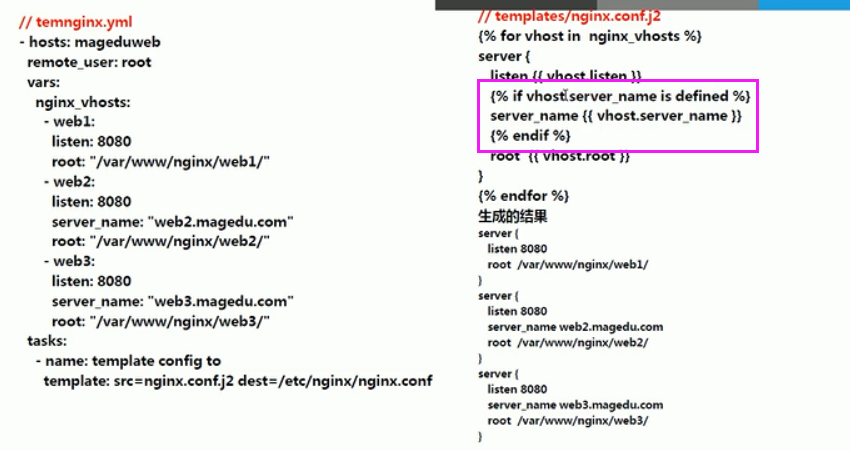




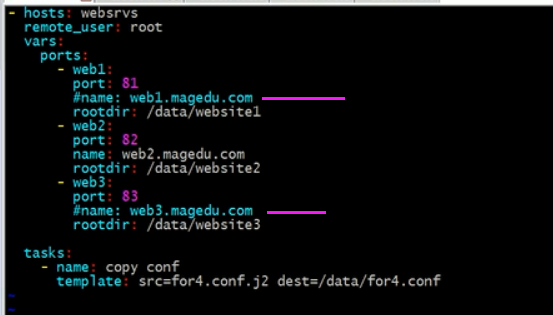
for3.conf.j2



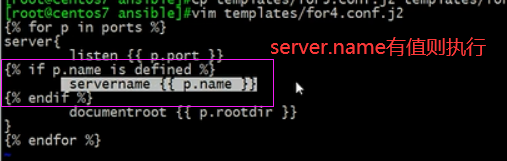
if条件

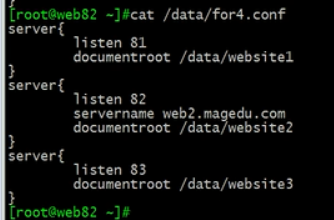


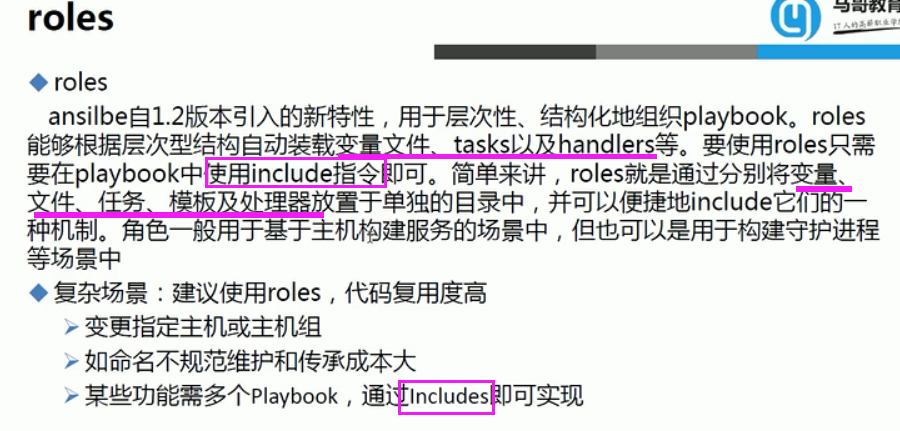
testfor4.yml



for4.conf.j2

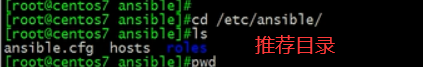




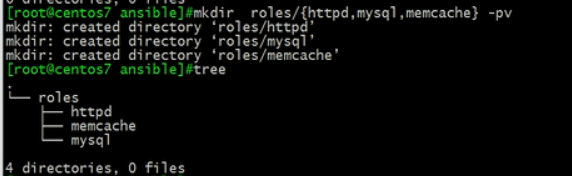


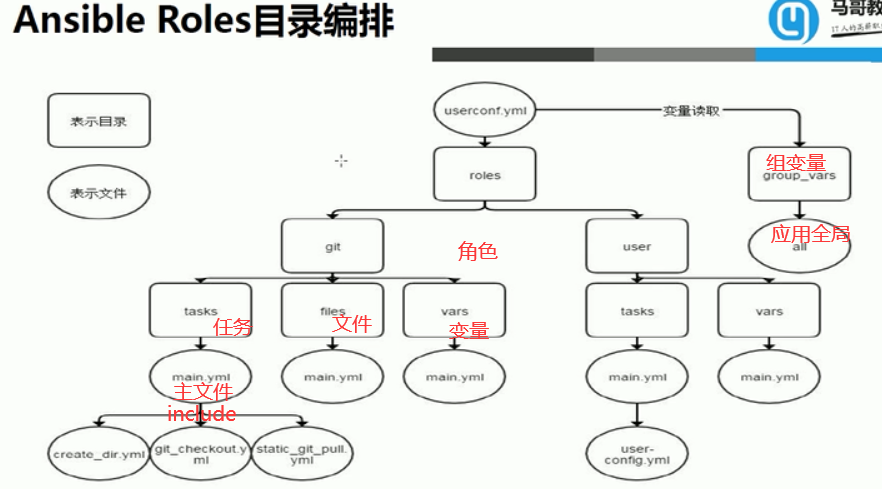
模块化





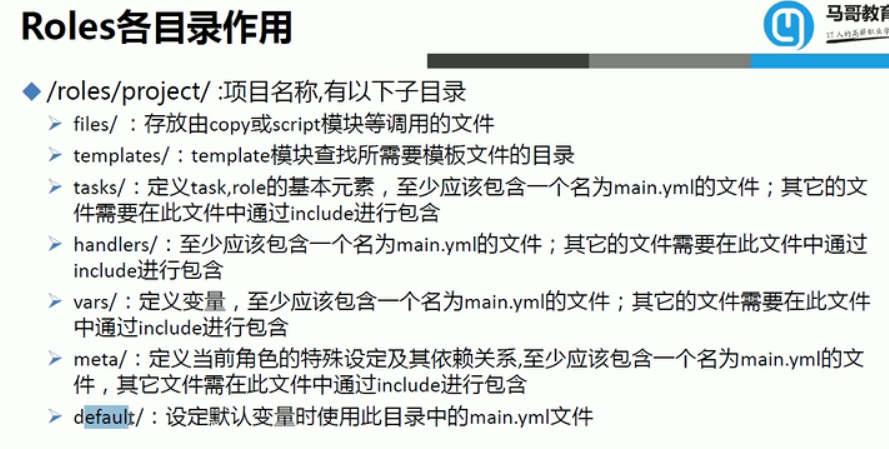
mkdir /root/ansible/roles/

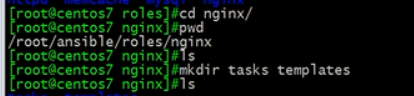


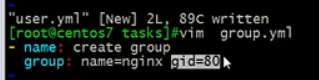


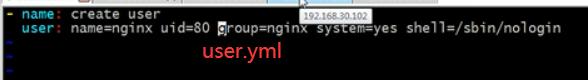
最基本的文件夹时tasks



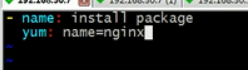




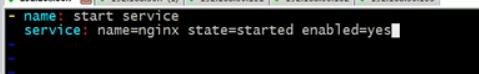




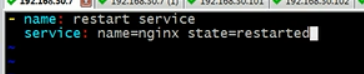
yum.yml



start.yml



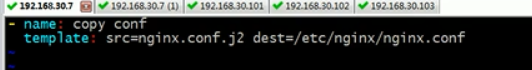
restart.yml



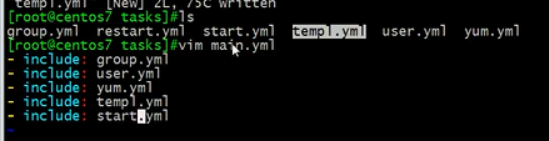
tmplates/nginx.conf.j2



templ.yml

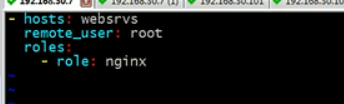


main.yml



调用角色--与roles是平级的 /etc/ansible/

nginx\_role.yml



ansible-playbook nginx\_role.yml

