目 录

[目 录 I](#_Toc493334309)

[第1章 ansible 1](#_Toc493334310)

[第一章    ansible的简介 2](#_Toc493334311)

[1、ansible的软件结构 3](#_Toc493334312)

[2、ansible的特性 3](#_Toc493334313)

[第二章    ansible的基础应用 4](#_Toc493334314)

[1、ansible管理端的安装 4](#_Toc493334315)

[2、ansible被管控主机的定义 4](#_Toc493334316)

[3、配置管理节点基于ssh秘钥登录被管理节点 5](#_Toc493334317)

[4、ansible命令的用法介绍 5](#_Toc493334318)

[第三章    ansible常用模块介绍 7](#_Toc493334319)

[1、获取常用模块的列表和对应模块的使用帮助信息 7](#_Toc493334320)

[command模块 8](#_Toc493334321)

[shell模块 8](#_Toc493334322)

[copy模块 9](#_Toc493334323)

[cron模块 11](#_Toc493334324)

[service模块 13](#_Toc493334325)

[user模块 14](#_Toc493334326)

[group模块 16](#_Toc493334327)

[file模块 17](#_Toc493334328)

[fetch模块 21](#_Toc493334329)

[hostname模块 22](#_Toc493334330)

[yum模块 23](#_Toc493334331)

[uri模块 24](#_Toc493334332)

[script模块 25](#_Toc493334333)

[setup模块 26](#_Toc493334334)

[template模块的使用 27](#_Toc493334335)

[unarchive模块 28](#_Toc493334336)

[1.1.1 get\_url 模块 29](#_Toc493334337)

[1.1.2 mount模块 29](#_Toc493334338)

[1.1.3 filesystem模块 30](#_Toc493334339)

[1.1.4 synchronize模块 31](#_Toc493334340)

[1.1.5 mysql\_db - 从远程主机添加或删除MySQL数据库。 32](#_Toc493334341)

[第四章    ansible的playbook基础应用介绍 32](#_Toc493334342)

[1、YAML文件的语法 32](#_Toc493334343)

[2、playbook的核心元素 33](#_Toc493334344)

[3、playbook的基础组件 34](#_Toc493334345)

[4、playbook文件的执行 35](#_Toc493334346)

[第五章    playbook中的handlers(触发器)的介绍 37](#_Toc493334347)

[1、handlers的作用 37](#_Toc493334348)

[2、handlers触发器的使用示例： 38](#_Toc493334349)

[第六章    playbook中的tags(标签)的介绍 39](#_Toc493334350)

[1、tags标签的作用 39](#_Toc493334351)

[2、tags标签的示例 40](#_Toc493334352)

[第七章    playbook中的variables(变量)的介绍 41](#_Toc493334353)

[1、facts类型的变量： 41](#_Toc493334354)

[2、ansible-playbook命令的命令行中的自定义变量： 43](#_Toc493334355)

[3、在定义主机的hosts中(也就是/etc/ansible/hosts文件中)定义变量 43](#_Toc493334356)

[4、在playbook的yaml文件中定义变量 46](#_Toc493334357)

[第八章    playbook中的templates(模板)的介绍 47](#_Toc493334358)

[1、templates模板文件的说明 47](#_Toc493334359)

[2、jinja2常用的语法 48](#_Toc493334360)

[3、template模块 49](#_Toc493334361)

[4、template模板使用配置示例 49](#_Toc493334362)

[第九章    playbook中的条件判断机制的介绍 50](#_Toc493334363)

[第十章    playbook中的循环(迭代)机制的介绍 51](#_Toc493334364)

[1、循环的相关概念 51](#_Toc493334365)

[2、循环的示例一：列表形式的迭代项的循环引用 52](#_Toc493334366)

[3、循环示例二：字典形式的迭代项的循环引用 54](#_Toc493334367)

[第十一章    ansible的roles(角色)功能的介绍 54](#_Toc493334368)

[1、角色的相关概念 54](#_Toc493334369)

[2、角色的目录结构 55](#_Toc493334370)

[3、在playbook中调用角色方法一： 56](#_Toc493334371)

[4、在playbook中调用角色方法二：(在角色调用时传递变量) 57](#_Toc493334372)

[5、在playbook中调用角色时，实现条件判断： 57](#_Toc493334373)

[实战一：利用ansible配置主备模型的keepalived+nginx 58](#_Toc493334374)

[1、实验环境 58](#_Toc493334375)

[2、实验前准备工作 58](#_Toc493334376)

[3、在ansible主机上利用ansible的roles功能，在两台被管理主机上安装和配置nginx 64](#_Toc493334377)

[5、验证被管理主机上nginx是否运行正常 71](#_Toc493334378)

[6、编辑生成keepalived的roles角色，和相关配置文件 72](#_Toc493334379)

[7、编辑playbook剧本文件，运行剧本 80](#_Toc493334380)

[8、验证keepalived对nginx的高可用是否成功 81](#_Toc493334381)

[实战二：实战一的基础上在nginx后端提供httpd+php+php-mysql 82](#_Toc493334382)

[1、实验环境 82](#_Toc493334383)

[2、利用ansible的roles，编辑roles相关配置 82](#_Toc493334384)

[3、编辑playbook文件，引用角色，测试运行，检测有无错误信息 91](#_Toc493334385)

[4、运行剧本，验证反代是否成功 91](#_Toc493334386)

[实战三：在此前实验基础上配置mysql服务 92](#_Toc493334387)

[1、实验环境 92](#_Toc493334388)

[2、编写ansible的roles角色的相关内容 92](#_Toc493334389)

[3、编辑生成playbook文件，引用角色，测试执行剧本，查看是否有报错 96](#_Toc493334390)

[4、执行剧本，验证配置是否正确 97](#_Toc493334391)

# ansible

ansible是一款无需在被管理主机上安装客户端，基于SSH对多台目标主机进行同时操作的轻量级的管理软件，借助各个内部的功能模块，实现了批量系统配置、批量程序部署、批量运行命令等功能。本篇就介绍一些关于ansible的基础配置相关的内容，具体包括：

1、ansible的简介

2、ansible的基础应用

3、ansible常用模块介绍

4、ansible的playbook基础应用介绍

5、playbook中的handlers(触发器)的介绍

6、playbook中的tags(标签)的介绍

7、playbook中的variables(变量)的介绍

8、playbook中的templates(模板)的介绍

9、playbook中的条件判断机制的介绍

10、playbook中的循环(迭代)机制的介绍

11、ansible的roles(角色)功能的介绍

12、ansible实战一：利用ansible配置主备模型的keepalived+nginx

13、ansible实战二：实战一的基础上在nginx后端提供httpd+php+php-mysql

14、ansible实战三：在此前实验基础上配置mysql服务

## 第一章    ansible的简介

ansible是新出现的自动化运维工具，基于Python开发，集合了众多运维工具（puppet、cfengine、chef、func、fabric）的优点，实现了批量系统配置、批量程序部署、批量运行命令等功能。

ansible是基于模块工作的，本身没有批量部署的能力。真正具有批量部署的是ansible所运行的模块，ansible只是提供一种框架。主要包括：

(1)、连接插件connection plugins：负责和被监控端实现通信；

(2)、host inventory：指定操作的主机，是一个配置文件里面定义监控的主机；

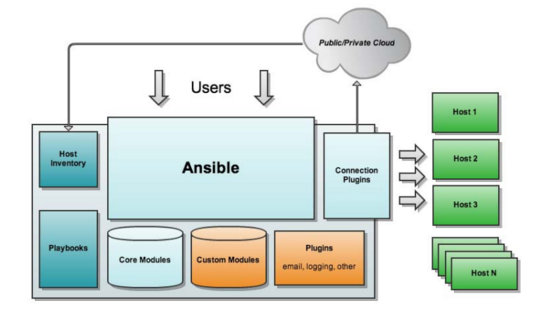
(3)、各种模块核心模块、command模块、自定义模块；

(4)、借助于插件完成记录日志邮件等功能；

(5)、playbook：剧本执行多个任务时，非必需可以让节点一次性运行多个任务

简单的说

### 1、ansible的软件结构



Host Inventory：主机清单，也就是被管理的主机列表

Playbooks：ansible的剧本，可想象为将多个任务放置在一起，一块执行

Core Modules：ansible的核心模块

Custom Modules：自定义模块

Connection Plugins：连接插件，用于与被管控主机之间基于SSH建立连接关系

Plugins：其他插件，包括记录日志等

### 2、ansible的特性

<1>模块化：调用特定的模块，完成特定任务

<2>基于python语言实现，由Paramiko(完成基于ssh的连接)，PyYAML(对YAML文件的支持)，jinja2(python的模板库)三个关键的模块

<3>部署简单：是没有客户端的

<4>支持自定义模块，使用任意编程语言

<5>支持强大的playbook

<6>具有幂等性：一个操作在一个主机上执行一遍和执行N遍的结果是一样的

## 第二章    ansible的基础应用

### 1、ansible管理端的安装

在EPEL源中，有包含ansible的软件包安装只需要配置好EPEL的yum源，yum安装即可

yum -y install ansible 安装ansible

yum安装介意解决软件依赖关系

rpm -qa ansible 检查安装是否成功

### 2、ansible被管控主机的定义

对希望被管控的主机的定义需要实现在ansible管理端的针对被管理主机的配置文件(/etc/ansible/hosts)中进行定义

[root@ m01 ~]# vi /etc/ansible/hosts

## 192.168.1.100

## 192.168.1.110

# If you have multiple hosts following a pattern you can specify

# them like this:

## www[001:006].example.com 一个主机组范围

# Ex 3: A collection of database servers in the 'dbservers' group

## [dbservers]

##

## db01.intranet.mydomain.net 可以是主机名也可以是ip

## db02.intranet.mydomain.net

## 10.25.1.56

## 10.25.1.57

# Here's another example of host ranges, this time there are no

# leading 0s:

## db-[99:101]-node.example.com

[boy]

172.16.1.31

172.16.1.7

172.16.1.41

172.16.1.8

### 3、配置管理节点基于ssh秘钥登录被管理节点

ssh-keygen -t dsa -f ~/.ssh/id\_dsa -P "" 非交互创建密钥

sshpass -p "123456" ssh-copy-id -i ~/.ssh/id\_dsa.pub "-o StrictHostKeyChecking=no [root@172.16.1.7](mailto:root@172.16.1.7)" 非交互发送公钥

### 4、ansible命令的用法介绍

ansible HOST-PATTERN [-f FORKS] [-m MOD\_NAME] [-a MOD\_ARGS]

HOST\_PATTERN：指明对哪些被管控主机进行操作

-f FORKS：表示一批处理几台主机，也就是当被管控主机很多时，ansible不是对所有主机同时发起管理操作，而是一批处理几台，然后再换一批，直到所有主机被处理完成，如果不指定，则默认是5台

-m MOD\_NAME：指明调用哪个模块执行操作，各个模块所能实现的功能不同，如果不指定，默认是用-m command模块

-a MOD\_ARGS：指明使用该模块的执行操作时的参数

[root@ m01 ~]# ansible boy -m ping 利用ping模块检测主机是否存在

172.16.1.41 | SUCCESS => {

"changed": false,

"ping": "pong"

}

172.16.1.7 | SUCCESS => {

"changed": false,

"ping": "pong"

}

172.16.1.31 | SUCCESS => {

"changed": false,

"ping": "pong"

}

172.16.1.8 | UNREACHABLE! => {

"changed": false,

"msg": "Failed to connect to the host via ssh: ssh: connect to host 172.16.1.8 port 22: No route to host\r\n",

"unreachable": true

} 为了测试172.16.1.8没有开启

[root@ m01 ~]# ansible boy -m command -a 'ls -l /root/.ssh/'

172.16.1.7 | SUCCESS | rc=0 >>

total 4

-rw------- 1 root root 598 May 16 10:20 authorized\_keys

172.16.1.41 | SUCCESS | rc=0 >>

total 4

-rw------- 1 root root 598 May 16 10:20 authorized\_keys

172.16.1.31 | SUCCESS | rc=0 >>

total 4

-rw------- 1 root root 1196 May 16 10:19 authorized\_keys

172.16.1.8 | UNREACHABLE! => {

"changed": false,

"msg": "Failed to connect to the host via ssh: ssh: connect to host 172.16.1.8 port 22: No route to host\r\n",

"unreachable": true

}

## 第三章    ansible常用模块介绍

### 1、获取常用模块的列表和对应模块的使用帮助信息

可用ansible-doc -l  来查看所有可用的模块列表

可用ansible-doc -s MOD\_NAME  来查看对应模块的帮助信息

[root@ m01 ~]# ansible-doc -s command 查看command模块帮助信息

- name: Executes a command on a remote node

action: command

chdir # cd into this directory before running the command

creates # a filename or (since 2.0) glob pattern, when it already exists, this step will \*not\* be run.

executable # change the shell used to execute the command. Should be an absolute path to the executable.

free\_form= # the command module takes a free form command to run. There is no parameter actually named

'free form'. See the examples!

removes # a filename or (since 2.0) glob pattern, when it does not exist, this step will \*not\* be run.

warn # if command warnings are on in ansible.cfg, do not warn about this particular line if set to

no/false.

### command模块

在远程主机执行命令,**不支持管道，重定向等shell的特性**

常用参数有：

chdir=   表示指明命令在远程主机上哪个目录下运行，也就是在命令执行前切换到哪个目录下

creates=   在命令运行时创建一个文件，如果文件已存在，则不会执行创建任务

removes=   在命令运行时移除一个文件，如果文件不存在，则不会执行移除任务

executeble=   指明运行命令的shell程序

[root@ m01 ~]# ansible boy -m command -a 'chdir=/tmp ls -l'

172.16.1.7 | SUCCESS | rc=0 >>

total 8

drwx------ 2 root root 4096 May 16 14:36 ansible\_pqYSL0

-rw-r--r-- 1 root root 7 May 16 11:02 txt

172.16.1.31 | SUCCESS | rc=0 >>

total 8

drwx------ 2 root root 4096 May 16 14:36 ansible\_4ydkYw

-rw-r--r-- 1 root root 1909 May 7 16:18 mail.rc.bak

### shell模块

在远程主机执行命令，相当于调用远程主机的shell进程，然后在该shell下打开一个子shell运行命令

**支持shell特性，如管道，重定向等**

注意：command和shell模块的核心参数直接为命令本身，而其他模块的核心参数一般是"key=value"格式

常见参数有：

chdir=   表示指明命令在远程主机上哪个目录下运行

creates=   在命令运行时创建一个文件，如果文件已存在，则不会执行创建任务

removes=  在命令运行时移除一个文件，如果文件不存在，则不会执行移除任务

executeble=   指明运行命令的shell程序

[root@ m01 ~]# ansible boy -m shell -a 'echo "hello">>/tmp

172.16.1.7 | SUCCESS | rc=0 >>

172.16.1.31 | SUCCESS | rc=0 >>

[root@ m01 ~]# ansible boy -m shell -a 'cat /tmp/test.txt'

172.16.1.7 | SUCCESS | rc=0 >>

hello

172.16.1.31 | SUCCESS | rc=0 >>

hello

### copy模块

拷贝ansible管理端的文件到远程主机的指定位置(**测试结果copy模块拷贝空目录)**

常见参数有：

dest=   指明拷贝文件的目标目录位置，使用绝对路径，如果源是目录，则目标也要是目录,如果目标文件已存在，会覆盖原有内容

src=   指明本地路径下的某个文件，可以使用相对路径和绝对路径，支持直接指定目录，如果源是目录，则目标也要是目录

mode=   指明复制时，目标文件的权限 例如mode=0777

owner=   指明复制时，目标文件的属主

group=   指明复制时，目标文件的属组

content=  指明复制到目标主机上的内容，不能与src一起使用，相当于复制content指明的数据，到目标文件中

[root@ m01 ~]# ansible boy -m copy -a 'src=/root/.ssh/id\_dsa.pub dest=/tmp mode=0700 owner=root group=root'   
172.16.1.31 | SUCCESS => {  
 "changed": true,   
 "checksum": "3ca1cffc40db6ffa1b3f9a55132ba81a300c47c1",   
 "dest": "/tmp/id\_dsa.pub",   
 "gid": 0,   
 "group": "root",   
 "md5sum": "0705ba19ecea04cb515be8a3947a5d72",   
 "mode": "0700",   
 "owner": "root",   
 "size": 598,   
 "src": "/root/.ansible/tmp/ansible-tmp-1494917300.28-258379254666145/source",   
 "state": "file",   
 "uid": 0  
}  
172.16.1.7 | SUCCESS => {  
 "changed": true,   
 "checksum": "3ca1cffc40db6ffa1b3f9a55132ba81a300c47c1",   
 "dest": "/tmp/id\_dsa.pub",   
 "gid": 0,   
 "group": "root",   
 "md5sum": "0705ba19ecea04cb515be8a3947a5d72",   
 "mode": "0700",   
 "owner": "root",   
 "size": 598,   
 "src": "/root/.ansible/tmp/ansible-tmp-1494917300.25-194887707025470/source",   
 "state": "file",   
 "uid": 0  
}

[root@ m01 ~]# ansible boy -m shell -a 'ls -l /tmp/'

172.16.1.7 | SUCCESS | rc=0 >>

total 16

drwx------ 2 root root 4096 May 16 14:49 ansible\_bkYIYB

-rwx------ 1 root root 598 May 16 14:48 id\_dsa.pub

-rw-r--r-- 1 root root 6 May 16 14:39 test.txt

-rw-r--r-- 1 root root 7 May 16 11:02 txt

172.16.1.31 | SUCCESS | rc=0 >>

total 16

drwx------ 2 root root 4096 May 16 14:49 ansible\_TFMQ6I

-rwx------ 1 root root 598 May 16 14:48 id\_dsa.pub

-rw-r--r-- 1 root root 1909 May 7 16:18 mail.rc.bak

-rw-r--r-- 1 root root 6 May 16 14:39 test.txt

### cron模块

管理计划任务的模块

常见参数有：

minute=  指明计划任务的分钟，支持格式：0-59，\*，\*/2等，与正常cron任务定义的一样的语法,省略时，默认为\*，也就是每分钟都执行

hour=  指明计划任务的小时，支持的语法：0-23，\*，\*/2等，省略时，默认为\*，也就是每小时都执行

day=  指明计划任务的天，支持的语法：1-31，\*，\*/2等，省略时，默认为\*，也就是每天都执行

month=  指明计划任务的月，支持的语法为：1-12，\*，\*/2等，省略时，默认为\*，也就是每月都执行

weekday=  指明计划任务的星期几，支持的语法为：0-6，\*等，省略时，默认为\*，也就是每星期几都执行

reboot  指明计划任务执行的时间为每次重启之后

name=   给该计划任务取个名称,必须要给明。每个任务的名称不能一样。删除任务时，只需要给明任务的名称即可

job=  执行的任务是什么，当state=present时才有意义

state=present|absent   表示这个任务是创建还是删除，present表示创建，absent表示删除，默认是present

[root@ m01 ~]# ansible boy -m cron -a 'minute=\*/5 hour=7-13 day=\* name="test cron" job="/usr/sbin/ntpdate time.nist.gov"'

172.16.1.7 | SUCCESS => { **成功一般都是金黄色**

"changed": true,

"envs": [],

"jobs": [

"test cron"

]

}

172.16.1.31 | SUCCESS => {

"changed": true,

"envs": [],

"jobs": [

"test cron"

]

}

[root@ m01 ~]# ansible boy -m shell -a 'crontab -l'

172.16.1.7 | SUCCESS | rc=0 >>

#crond-id-001:time sync by boy

\*/5 \* \* \* \* /usr/sbin/ntpdate time.nist.gov >/dev/null 2>&1

#Ansible: test cron

\*/5 7-13 \* \* \* /usr/sbin/ntpdate time.nist.gov

172.16.1.31 | SUCCESS | rc=0 >>

#crond-id-001:time sync by boy

\*/5 \* \* \* \* /usr/sbin/ntpdate time.nist.gov >/dev/null 2>&1

#Ansible: test cron

\*/5 7-13 \* \* \* /usr/sbin/ntpdate time.nist.gov

[root@ m01 ~]# ansible boy -m cron -a 'name="test cron" state=absent'

172.16.1.31 | SUCCESS => {

"changed": true,

"envs": [],

"jobs": []

}

172.16.1.7 | SUCCESS => {

"changed": true,

"envs": [],

"jobs": []

}

[root@ m01 ~]# ansible boy -m shell -a 'crontab -l'

172.16.1.7 | SUCCESS | rc=0 >>

#crond-id-001:time sync by boy

\*/5 \* \* \* \* /usr/sbin/ntpdate time.nist.gov >/dev/null 2>&1

172.16.1.31 | SUCCESS | rc=0 >>

#crond-id-001:time sync by boy

\*/5 \* \* \* \* /usr/sbin/ntpdate time.nist.gov >/dev/null 2>&1

### service模块

用来管理远程主机上的服务的模块

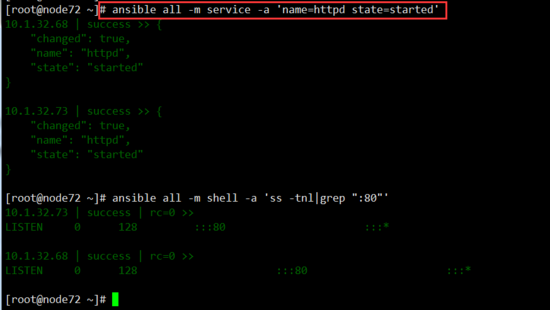
常见参数有：

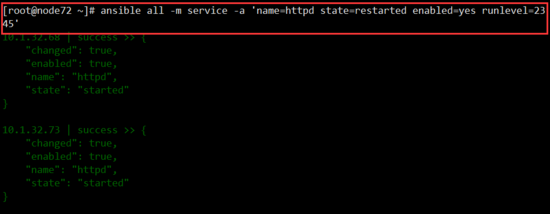
name=   被管理的服务名称

state=started|stopped|restarted   表示启动或关闭或重启

enabled=yes|no  表示要不要设定该服务开机自启动

runlevel=   如果设定了enabled开机自动启动，则要定义在哪些运行级别下自动启动







### user模块

管理远程主机上的用户的账号

常见参数有：

name=   指明要管理的账号名称

state=present|absent   指明是创建账号还是删除账号，present表示创建，absent表示删除

system=yes|no   指明是否为系统账号

uid=   指明用户UID

group=   指明用户的基本组

groups=   指明用户的附加组

shell=   指明默认的shell

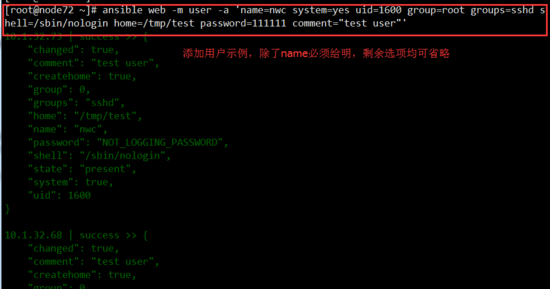
home=   指明用户的家目录

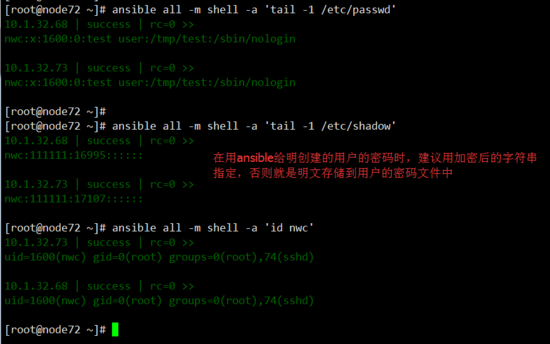
move\_home=yes|no   当home设定了家目录，如果要创建的家目录已存在，是否将已存在的家目录进行移动

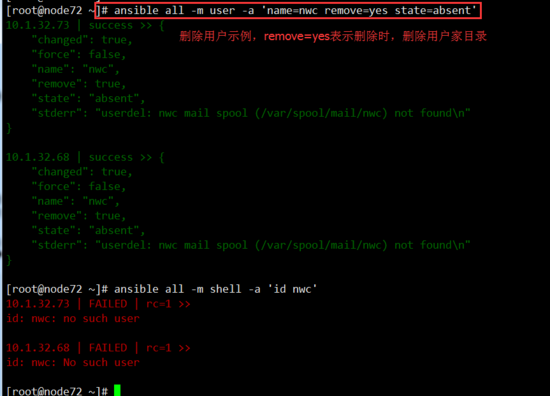
password=   指明用户的密码，最好使用加密好的字符串

comment=   指明用户的注释信息

remove=yes|no   当state=absent时，也就是删除用户时，是否要删除用户的而家目录







### group模块

用来添加或删除远端主机的用户组

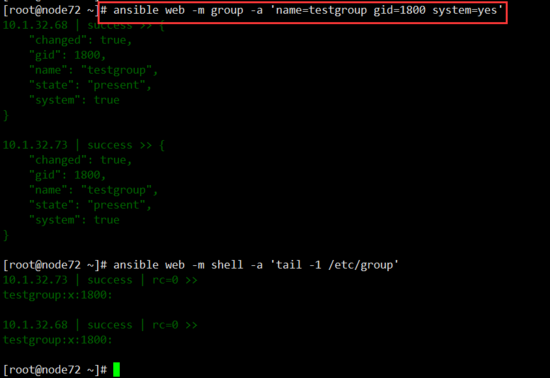
常见参数有：

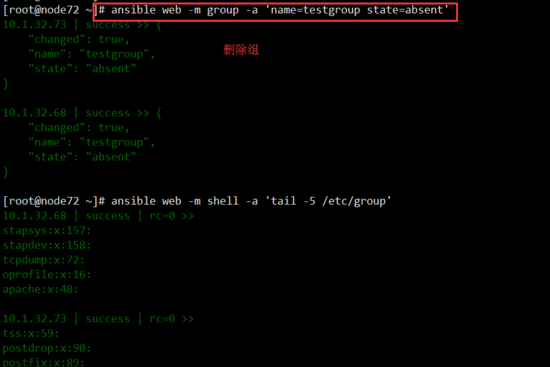
name=   被管理的组名

state=present|absent   是添加还是删除,不指名默认为添加

gid=   指明GID

system=yes|no   是否为系统组





### file模块

file模块它包含了文件、文件夹、超级链接类的创立、拷贝、移动、删除操作。

用于设定远程主机上的文件属性

常见参数有：

path=   指明对哪个文件修改其属性

src=   指明path=指明的文件是软链接文件，其对应的源文件是谁，必须要在state=link时才有用

state= file/link/directory/hard/touch/absent #file代表拷贝后是文件；link代表最终是个软链接；directory代表文件夹；hard代表硬链接；touch代表生成一个空文件；absent代表删除

owner=   指明文件的属主

group=   指明文件的属组

mode=   指明文件的权限

创建软链接的用法：src=  path=  state=link

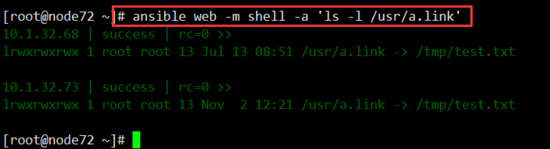
修改文件属性的用法：path=   owner=  mode=  group=

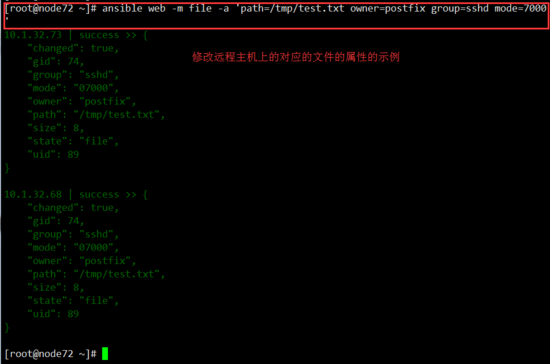
创建目录的用法：path=   state=directory

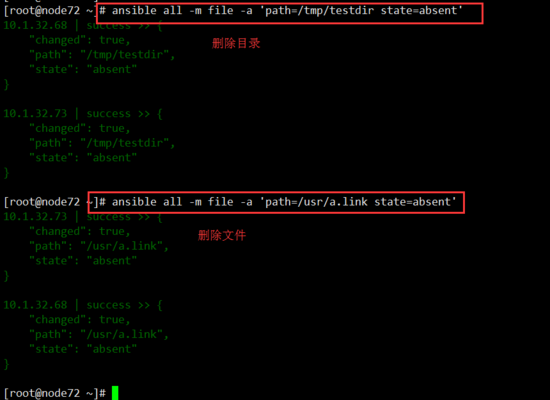
例子： ansible test -m file -a 'path=/home/boy/tools state=directory'

删除文件：path= state=absent









file模块

在之前ansible命令行的时候有copy模块，在playbook的时代自然也有一个模块专门负责文件的拷贝，当然这个时代它不仅仅是文件拷贝那么简单。

来自官方的解释：

file模块它包含了文件、文件夹、超级链接类的创立、拷贝、移动、删除操作。

常用参数

| **参数名** | **是否必须** | **默认值** | **选项** | **说明** |
| --- | --- | --- | --- | --- |
| follow | no | no | yes/no | 如果原来的文件是link，拷贝后依旧是link |
| force | no | no | yes/no | 强制执行，没说的 |
| group | no |  |  | 设定一个群组拥有拷贝到远程节点的文件权限 |
| mode | no |  |  | 等同于chmod，参数可以为“u+rwx or u=rw,g=r,o=r” |
| owner | no |  |  | 设定一个用户拥有拷贝到远程节点的文件权限 |
| path | yes |  |  | 目标路径，也可以用dest,name代替 |
| src | yes |  |  | 待拷贝文件/文件夹的原始位置。 |
| state | no | file | file/link/directory/hard/touch/absent | file代表拷贝后是文件；link代表最终是个软链接；directory代表文件夹；hard代表硬链接；touch代表生成一个空文件；absent代表删除 |

案例

# 修改文件的所有组、人、权限。

- file: path=/etc/foo.conf owner=foo group=foo mode=0644

# 操作链接的案例

- file: src=/file/to/link/to dest=/path/to/symlink owner=foo group=foo state=link

#参数化案例

- file: src=/tmp/{{ item.path }} dest={{ item.dest }} state=link

with\_items:

- { path: 'x', dest: 'y' }

- { path: 'z', dest: 'k' }

# 使用touch来创建一个空文件并定义权限

- file: path=/etc/foo.conf state=touch mode="u=rw,g=r,o=r"

# touch一个空文件，并且修改权限

- file: path=/etc/foo.conf state=touch mode="u+rw,g-wx,o-rwx"

### fetch模块

从远程主机拉取文件到本地

一般情况下，只会从一个远程节点拉取数据

常见参数有：

dest=  从远程主机上拉取的文件存放在本地的位置，一般只能是目录

src=   指明远程主机上要拉取的文件，只能是文件，不能是目录

[root@ m01 ~]# ansible boy -m fetch -a 'src=/tmp/test.txt dest=/tmp'

[root@ m01 tmp]# tree

.

├── 172.16.1.31

│   └── tmp

│   └── test.txt

├── 172.16.1.7

│   └── tmp

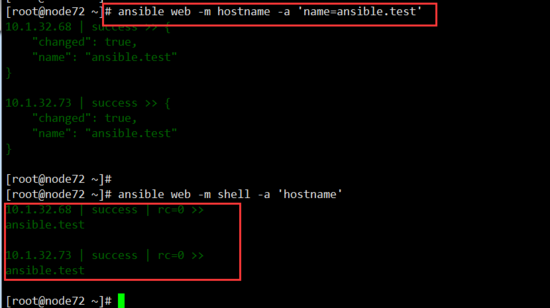
│   └── test.txt

### hostname模块

管理远程主机上的主机名

常用参数有

name=  指明主机名



### yum模块

基于yum机制，对远程主机管理程序包

常用参数有：

name=   指明程序包的名称，可以带上版本号，不指明版本，就是默认最新版本。

state=present|latest|absent   指明对程序包执行的操作，present表示安装程序包，latest表示安装最新版本的程序包，absent表示卸载程序包

disablerepo=    在用yum安装时，临时禁用某个仓库，仓库的ID

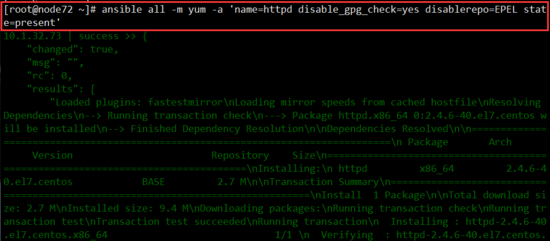
enablerepo=    在用yum安装时，临时启用某个仓库,仓库的ID

conf\_file=   指明yum运行时采用哪个配置文件，而不是使用默认的配置文件

diable\_gpg\_check=yes|no  是否启用gpg-check







### uri模块

如果远端是web服务器，可以利用ansible直接请求某个网页

常见参数有：

url=  指明请求的url的路径，如：http://10.1.32.68/test.jpg

user=  如果请求的url需要认证，则认证的用户名是什么

password=  如果请求的url需要认证，则认证的密码是什么

method=  指明请求的方法，如GET、POST…

body=   指明报文中实体部分的内容，一般是POST方法或PUT方法时用到

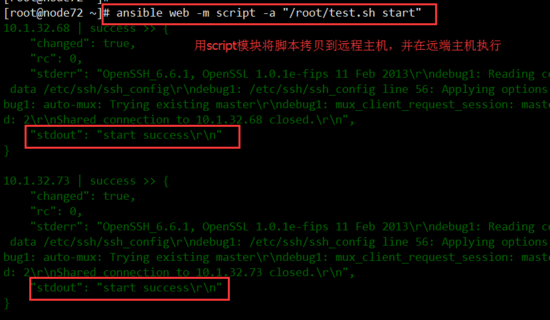
HEADER\_   自定义请求报文中的添加的首部

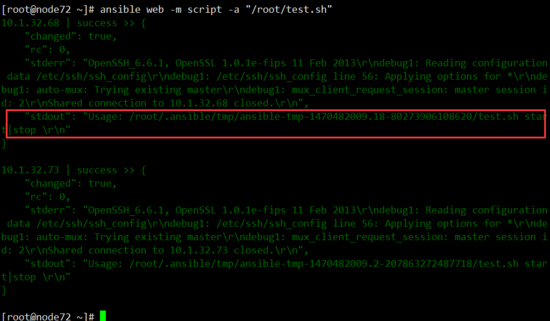
### script模块

将管理端的某个脚本，移动到远端主机(不需要指明传递到远端主机的哪个路径下，系统会自动移动)，然后执行

一般是自动移动到远端主机的/root/.ansible/tmp目录下，然后自动给予其权限，然后再开个子shell然后运行脚本，**运行完成后删除脚本**







### setup模块

可收集远程主机的facts变量的信息，相当于收集了目标主机的相关信息(如内核版本、操作系统信息、cpu、…)，保存在ansible的内置变量中，之后我们有需要用到时，直接调用变量即可



### template模块的使用

基于模板方式，生成一个模板文件，复制到远程主机，让远程主机基于模板，生成符合远程主机自身的文件

**注意：此模块不能在命令行使用，只能用在playbook中**

常见的参数有：

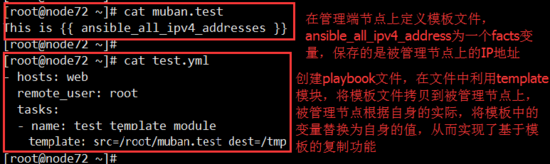
src=  指明管理端本地的模板文件的目录

dest=   指明将模板文件拷贝到远程主机的哪个目录下

owner=  指明拷贝到远程主机的文件的属主

group=  指明拷贝到远程主机的文件的属组

mode=   指明拷贝到远程主机的文件的权限





### unarchive模块

用于解压文件，模块包含如下选项：

copy：在解压文件之前，是否先将文件复制到远程主机，默认为yes。若为no，则要求目标主机上压缩包必须存在。

creates：指定一个文件名，当该文件存在时，则解压指令不执行

dest：远程主机上的一个路径，即文件解压的路径

grop：解压后的目录或文件的属组

list\_files：如果为yes，则会列出压缩包里的文件，默认为no，2.0版本新增的选项

mode：解决后文件的权限

src：如果copy为yes，则需要指定压缩文件的源路径

owner：解压后文件或目录的属主

示例如下：

- unarchive: src=foo.tgz dest=/var/lib/foo

- unarchive: src=/tmp/foo.zip dest=/usr/local/bin copy=no

- unarchive: src=https://example.com/example.zip dest=/usr/local/bin copy=no

[root@ m01 tools]# ansible lb01 -m unarchive -a 'src=/home/boy/tools/mysql-5.6.34-linux-glibc2.5-x86\_64.tar.gz dest=/home/boy/tools'

### get\_url 模块

该模块主要用于从http、ftp、https服务器上下载文件（类似于wget），主要有如下选项：

sha256sum：下载完成后进行sha256 check；

timeout：下载超时时间，默认10s

url：下载的URL

url\_password、url\_username：主要用于需要用户名密码进行验证的情况

use\_proxy：是事使用代理，代理需事先在环境变更中定义

示例：

get\_url: url=http://example.com/path/file.conf dest=/etc/foo.conf mode=0440

get\_url: url=http://example.com/path/file.conf dest=/etc/foo.conf sha256sum=b5bb9d8014a0f9b1d61e21e7

### mount模块

配置挂载点

选项：

dump

fstype：必选项，挂载文件的类型

name：必选项，挂载点

opts：传递给mount命令的参数

src：必选项，要挂载的文件

state：必选项

present：只处理fstab中的配置

absent：删除挂载点

mounted：自动创建挂载点并挂载之

umounted：卸载

示例：

name=/mnt/dvd src=/dev/sr0 fstype=iso9660 opts=ro state=present

name=/srv/disk src='LABEL=SOME\_LABEL' state=present

name=/home src='UUID=b3e48f45-f933-4c8e-a700-22a159ec9077' opts=noatime state=present

ansible test -a 'dd if=/dev/zero of=/disk.img bs=4k count=1024'

ansible test -a 'losetup /dev/loop0 /disk.img'

ansible test -m filesystem 'fstype=ext4 force=yes opts=-F dev=/dev/loop0'

ansible test -m mount 'name=/mnt src=/dev/loop0 fstype=ext4 state=mounted opts=rw

### filesystem模块

在块设备上创建文件系统

选项：

dev：目标块设备

force：在一个已有文件系统 的设备上强制创建

fstype：文件系统的类型

opts：传递给mkfs命令的选项

示例：

ansible test -m filesystem -a 'fstype=ext2 dev=/dev/sdb1 force=yes'

ansible test -m filesystem -a 'fstype=ext4 dev=/dev/sdb1 opts="-cc"

### synchronize模块

使用rsync同步文件，其参数如下：

archive: 归档，相当于同时开启recursive(递归)、links、perms、times、owner、group、-D选项都为yes ，默认该项为开启

checksum: 跳过检测sum值，默认关闭

compress:是否开启压缩

copy\_links：复制链接文件，默认为no ，注意后面还有一个links参数

delete: 删除不存在的文件，默认no

dest：目录路径

dest\_port：默认目录主机上的端口 ，默认是22，走的ssh协议

dirs：传速目录不进行递归，默认为no，即进行目录递归

rsync\_opts：rsync参数部分

set\_remote\_user：主要用于/etc/ansible/hosts中定义或默认使用的用户与rsync使用的用户不同的情况

mode: push或pull 模块，push模的话，一般用于从本机向远程主机上传文件，pull 模式用于从远程主机上取文件

使用示例：

src=some/relative/path dest=/some/absolute/path rsync\_path="sudo rsync"

src=some/relative/path dest=/some/absolute/path archive=no links=yes

src=some/relative/path dest=/some/absolute/path checksum=yes times=no

src=/tmp/helloworld dest=/var/www/helloword rsync\_opts=--no-motd,--exclude=.git mode=pull

### mysql\_db - 从远程主机添加或删除MySQL数据库。

## 第四章    ansible的playbook基础应用介绍

当需要执行的任务有多个时，需要一条一条编辑ansible命令，然后执行，而且当需要重复执行时，又要重新编辑执行，这样效率不高，因此ansible就可以利用playbook来完成将任务写到一个YAML格式的文件中，然后利用ansible-playbook进行调用该文件，从而实现了多条语句，可重复执行的效果，类似shell脚本的效果，ansible的playbook要借助YAML文件来实现，YAML文件扩展名通常为.yaml或.yml

### 1、YAML文件的语法

YAML语法和其他高阶语言类似，并可以简单表达清单，散列表、标量等数据结构。其结构通过空格来展示，序列里的项用“-”来代表，Map你的键值用“ : ”分隔

YAML文件中列表的表示：列表中的所有元素均使用"-" 开头，例如：

– apple

– orange

– mango

YAML文件中字典的表示：字典通过key与value进行标识，如：

name: nwc

job: manager

sex: M

也可以将key:value放置于{}中进行表示，如

{name: nwc,job: manager,sex: M}

### 2、playbook的核心元素

Hosts：运行在哪些主机之上

Users：远程主机上，运行此任务的身份，不指名默认为root

Tasks：任务，也就是定义的具体任务，由模块定义的操作的列表

Variables：变量

Templates：模板，包含了模板语法编写的模板的文本文件

Handlers：处理器，类似Tasks，只是在特定的条件下才会触发的任务

某任务的状态在运行后为changed时，可通过"notify"通知给相应的handlers进行触发执行

Roles：角色，将Hosts剥离出去，由Tasks、Variables、Templates、Handlers所组成的一种特定的结构的集合

### 3、playbook的基础组件

hosts：运行指定任务的而目标主机，多个主机用:冒号分隔

remote\_user：在远程主机上执行任务的用户;可以全局指定，也可以单个任务指定

sudo\_user：表示以sudo方式运行任务时，切换为哪个用户身份运行

tasks：

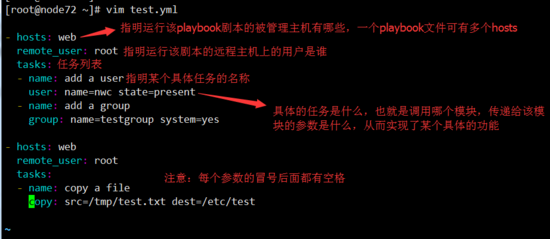
任务列表，ansible运行任务的方式为，将第一个任务在所有主机上运行完成，然后再将第二个任务在所有主机上运行…，当某个任务在某个主机上运行出现故障，会造成任务终止，再次执行任务只需直接执行即可

定义任务列表，实际就是指明使用的模块和对应的模块参数来完成的任务的列表，其格式有两种：

(1)action:MODULE  ARGUMENTS

(2)MODULE:ARGUMENTS

注意：shell和command模块后面直接跟命令，而不是key=value的参数列表



### 4、playbook文件的执行

playbook文件定义的任务要向执行，需要利用ansible-playbook命令进行调用

ansible-playbook命令用法：

<1> 检测语法

ansible-playbook –syntax-check /PATH/TO/PLAYBOOK.yaml

<2> 测试运行

ansible-playbook -C|–check /PATH/TO/PLAYBOOK.yaml

只检测执行指定的YAML文件可能会发生改变，但不真正执行操作，相当于测试运行

–list-hosts  检测YAML文件可能影响到的主机列表

–list-tasks   列出YAML文件的任务列表

–list-tags    列出YAML文件中的标签

<3> 运行

ansible-playbook /PATH/TO/PLAYBOOK.yml

可用选项：

不加任何选项表示完整运行整个playbook文件

-t TAGS,–tags=TAGS  表示只执行那个标签的任务

–skip-tags=SKIP\_TAGS  表示除了指明的标签的任务，其他任务都执行

–start-at-task=START\_AT   从指明的任务开始往下运行

<4> 通常情况下剧本的执行过程

先要利用 ansible-playbook -C|–check /PATH/TO/PLAYBOOK.yaml进行测试，测试没问题后

再利用 ansible-playbook /PATH/TO/PLAYBOOK.yml正式执行





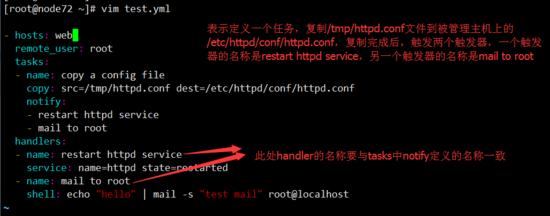
## 第五章    playbook中的handlers(触发器)的介绍

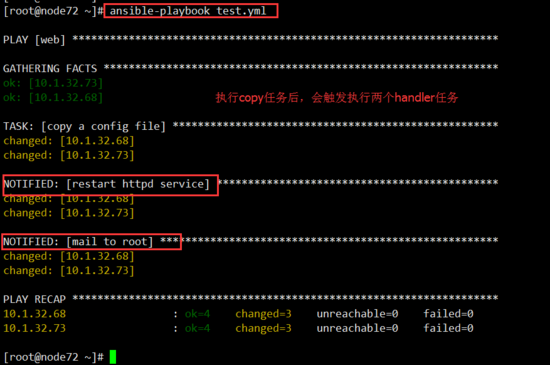
### 1、handlers的作用

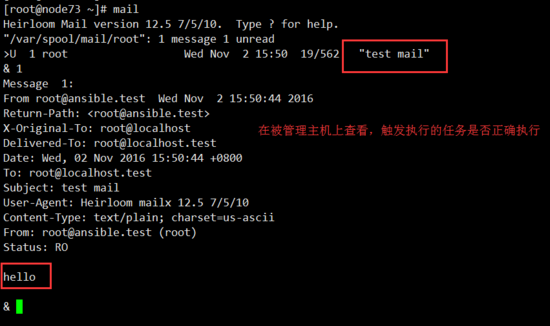
用于当关注资源发生变化时采取一定的操作，可理解为：当之前定义在tasks中的任务，如果执行成功后，我们希望在此基础上触发某个别的任务，这时就需要定义handlers。

要想handlers生效，首先需要在tasks的任务中定义一个notify，表示执行成功后，通知执行哪个handler，然后再定义handlers中，定义handler任务，handler任务的name要与notify中定义通知给哪个handler的名称一致

### 2、handlers触发器的使用示例：







## 第六章    playbook中的tags(标签)的介绍

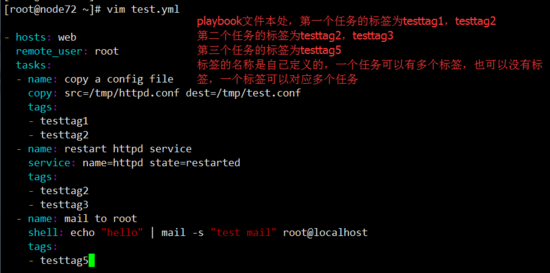
### 1、tags标签的作用

当我们定义了一个playbook文件，文件有很多任务要执行，如果我们只是希望执行其中的某一个任务，则可以在编写该任务时，为该任务加上标签，然后利用ansible-playbook调用时，指明只执行那个tags标签的任务(ansible-playbook -t TAG\_NANE YAML文件)

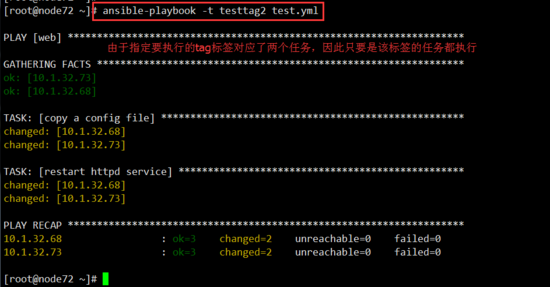
可以将多个任务提供一样的标签，这样，就可以实现指定运行某标签的任务时，同时运行多个任务；也支持一个任务定义多个标签

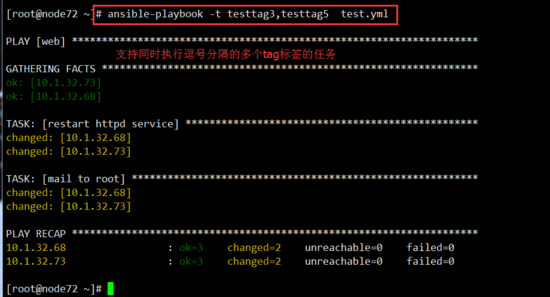
可以在用ansible-playbook利用-t指明执行的标签的任务时，支持用逗号,隔开的多个标签，则也是多个标签的任务都执行

### 2、tags标签的示例









## 第七章    playbook中的variables(变量)的介绍

在playbook中可在各个任意地方使用变量，引用变量的格式为：{{ VAR\_NAME }}，变量名与大括号之间有空格

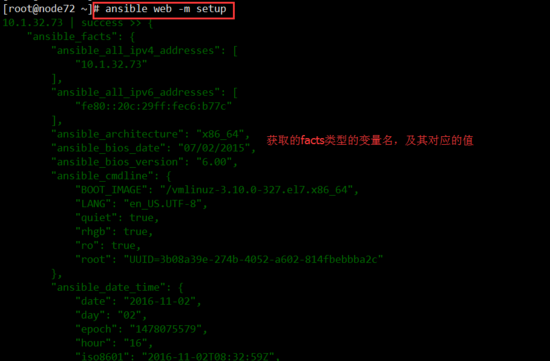
定义变量的方式分别为：

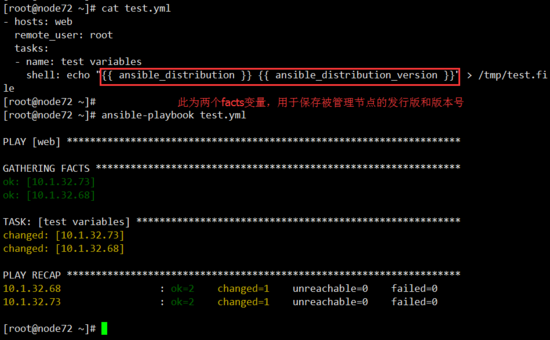
### 1、facts类型的变量：

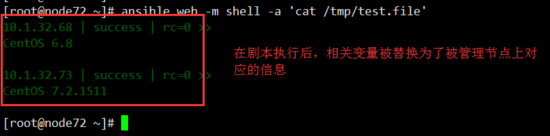
可直接调用，是ansible收集的关于被管理主机的相关信息，其被保存在ansible的一些变量中，如果要查看某个被管理主机有哪些facts变量可用，则可以执行：

ansible 10.1.32.72 -m setup

可列出10.1.32.72主机上可用的所有的facts变量及其值



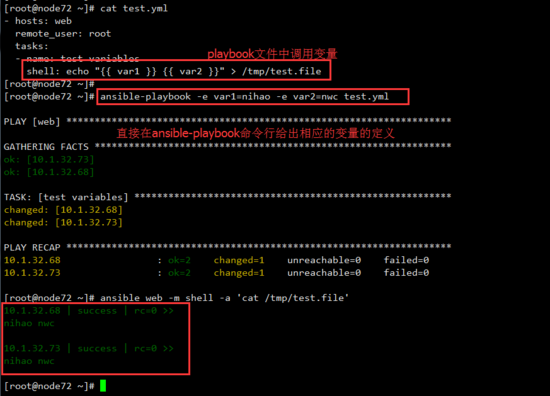




### 2、ansible-playbook命令的命令行中的自定义变量：

ansible-playbook -e VARS=VALUE

如果要指定多个变量，则用多个-e引导即可



### 3、在定义主机的hosts中(也就是/etc/ansible/hosts文件中)定义变量

<1>实现向不同的主机传递不同的变量

如：vim /etc/ansible/hosts

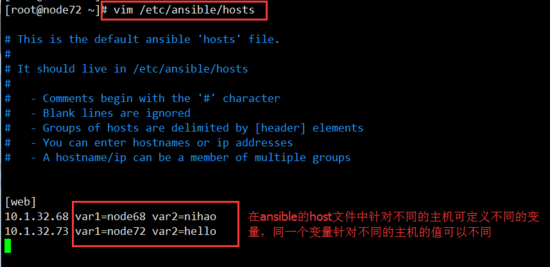
[webserver]

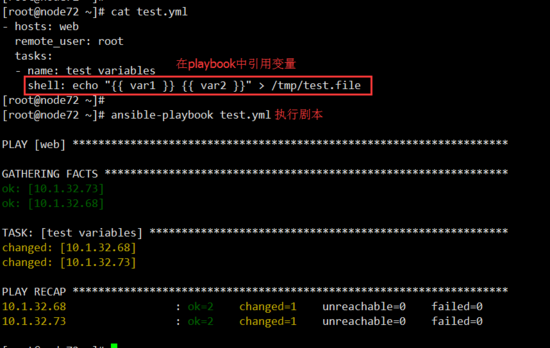
10.1.32.72 hname=web1 aaa=111 bbb=test

10.1.32.73 hname=web2 aaa=222

表示针对10.1.32.72这台主机，hname这个变量的值为8080，aaa变量的值为111，bbb变量的值为test

针对10.1.32.73这台主机，hname这个变量的值为8090，aaa变量的值为222







<2>实现向某个组内的主机，传递相同的变量

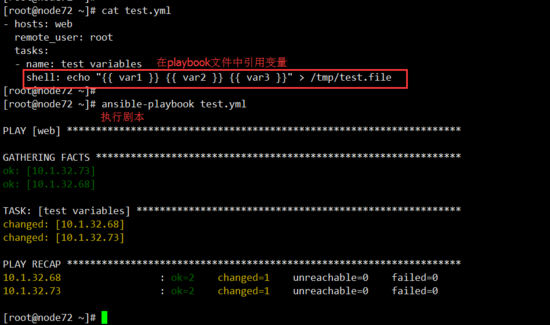
如：vim /etc/ansible/hosts

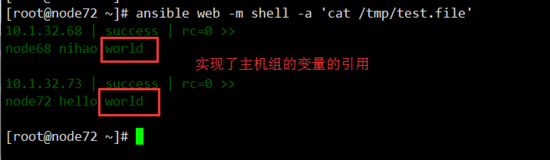
[webserver:vars]

http\_port=8080

表示向webserver组内的主机定义相同的变量http\_port,其值都为8080







<3>hosts文件中在每个主机后面可以用ansible\_ssh\_user和ansible\_ssh\_pass来指明ansible连接该主机时，不是采用我们之前自己手动执行的利用ssh秘钥登录，而是直接将用户名密码写入到/etc/ansible/hosts文件中，每次连接都是基于用户名密码的登录

如：vim /etc/ansible/hosts

[dbserver]

10.1.32.68 ansible\_ssh\_user=root ansible\_ssh\_pass=123456

10.1.32.73

注意，此类参数不能传递给playbook，也就是无法进行调用，只是用来ansible连接远程主机时的定义

除了上面的两个参数，还有：

ansible\_ssh\_host  连接的远程主机

ansible\_ssh\_port  连接的远程主机的端口

ansible\_sudo\_pass   以sudo方式运行任务时的sudo用户的密码

### 4、在playbook的yaml文件中定义变量

如:有个YAML文件为/root/test.yaml，内容为

– hosts: webserver

remote\_user: root

vars:

– pkname: httpd

– yname: php

tasks:

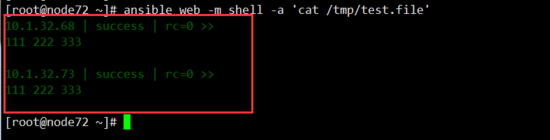
– name: install packages

yum: name={{ pkname }} state=present

– name: install packages2

yum: name={{ yname }} state=present





## 第八章    playbook中的templates(模板)的介绍

### 1、templates模板文件的说明

templates是模板文本文件，但是此文本文件内部嵌套有脚本(脚本是使用模板编程语言编写)，主要用来提供文件模板，让被管理主机根据模板中定义的脚本，生成符合远程主机环境的文件，也就是相当于提供一个模板文件，拷贝到被管理主机上，被管理主机根据自身的情况，生成符合自身实际的文件，模板文件中的语法，是语句模板编程语言所定义的，在ansible中，是jinja2的语法格式(因为ansible是python语言开发，而python嵌入文本中的语言是jinja2)

### 2、jinja2常用的语法

数据类型(字面量)：

字符串：使用单引号或双引号引用起来的都被认为是字符串

数字：整数、浮点数，不能用引号

列表：[item1,item2,…]

元组：(item1,item2,…)

字典：{key1:value1,key2:value2,…}

字典的key一般是字符串，所以要用引号引起来

布尔型：true/false

算数运算：

+、-、\*、/、//(除完以后只保留商)、%(取模，除完以后只留余数)、\*\*(次方)

比较操作：

==、!=、>、>=、<、<=

逻辑运算：

and、or、not

变量引用：与YAML语法一样

{{ VAR\_NAME }}

迭代(循环)、条件判断

### 3、template模块

当模板文件生成后，就可以借助template模块，将模板文件拷贝到被管控主机上，生成符合远端主机环境的文件，注意不能用copy模块进行拷贝，因为copy模块拷贝时，模板文件中定义的一些jinja2的语法结构会被当做纯文本信息进行拷贝，而用template模块进行拷贝时，则会识别jinja2的语法，将对应的语法替换为符合远端主机的具体的值，生成符合远端主机环境的文件

template模块的参数有：

src=  指明管理端本地的模板文件的目录

dest=   指明将模板文件拷贝到远程主机的哪个目录下

owner=  指明拷贝到远程主机的文件的属主

group=  指明拷贝到远程主机的文件的属组

mode=   指明拷贝到远程主机的文件的权限

### 4、template模板使用配置示例





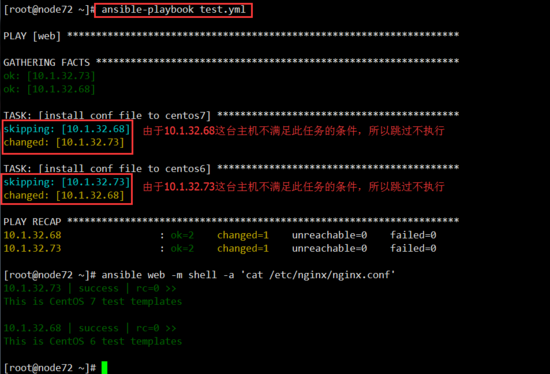
## 第九章    playbook中的条件判断机制的介绍

当我们希望在playbook文件中，完成诸如在某条件满足时，才执行指定的任务时，就需要借助条件判断机制。

要想使用条件判断，可以在tasks中使用when语句，标明在什么情况下，才执行该任务，when语句支持jinja2的语法格式

示例：





## 第十章    playbook中的循环(迭代)机制的介绍

### 1、循环的相关概念

**当需要重复执行同一类任务时，可以用到循环**

循环实际就是对迭代项的引用，迭代项的固定变量名为item，而后在tasks使用with\_items给定要迭代的元素列表

with\_items在表示机制可以使用

列表：如

with\_items:

– aa

– bb

…

该种方式定义的迭代项直接用：{{ item }}进行引用

也可以使用字典，如：(如果值为字符串，需要用引号引起来，变量与值之间要用空格隔开)

with\_items:

– {var1: value1,var2: value2,…}

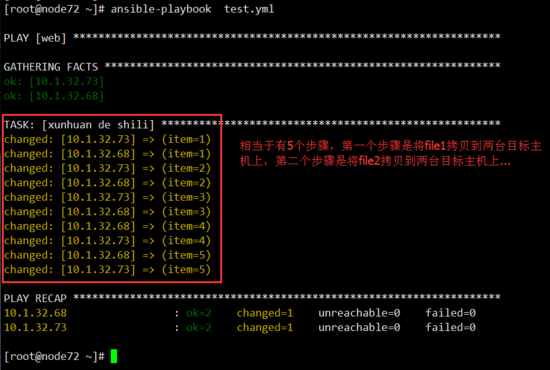
– {var1: value3,var2: value4,…}

…

该种方式定义的迭代项，应用要用 {{ item.var1 }}引用第一个变量，{{ item.var2 }}引用第二个变量…

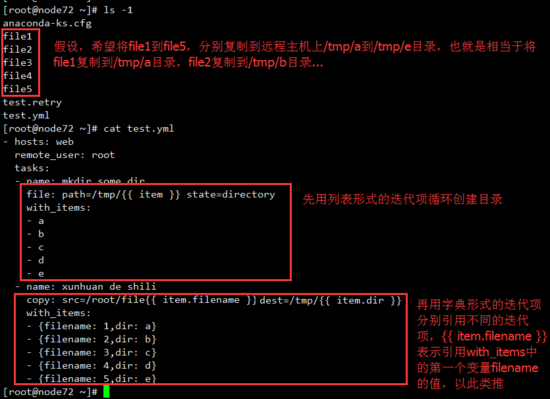
### 2、循环的示例一：列表形式的迭代项的循环引用







### 3、循环示例二：字典形式的迭代项的循环引用



## 第十一章    ansible的roles(角色)功能的介绍

### 1、角色的相关概念

**角色集合，实际是相当于多种不同的tasks的文件的集中存储在某个目录下**，该目录就是角色集合就是roles(默认是/etc/ansible/roles/目录，可通过ansible的配置文件来调整默认的角色目录)，在该目录下有很多子目录，就是一个一个的不同角色目录，而在每个角色目录下就会有分别有具体的功能的实现

如：/etc/ansible/roles/  此为角色集合，目录下有自定义的各个子目录，如

mysql/子目录，也就是mysql角色

httpd/子目录，也就是httpd角色

nginx/子目录，也就是nginx角色

### 2、角色的目录结构

每个角色的定义，以特定的层级目录结构进行组织：以mysql/子目录(mysql角色)为例：(每种角色的目录结构都一样)

files/子目录

存放由copy或script等模块调用的文件

templates/子目录

存放template模块查找所需要的模板文件的目录，如之前示例中用于给被管理主机提供nginx的模板配置文件

tasks/子目录

任务存放的目录，至少应该包含一个main.yml的文件，文件中定义了需要执行的任务清单，该目录下也可以有

其他.yml文件，但是需要在main.yml文件中用**include**指令将其他.yml文件包含进来

handlers/子目录

存放相关触发执行器的目录，至少应该包含一个main.yml的文件，文件中定义了触发器的任务清单，该目录下

也可以有其他.yml文件，但是需要在main.yml文件中用**include**指令将其他.yml文件包含进来

vars/子目录

变量存放的目录，至少应该包含一个main.yml的文件，文件中定义了相关的变量及其值，该目录下也可以有其

他.yml文件，但是需要在main.yml文件中用**include**指令将其他.yml文件包含进来

meta/

用于存放此角色元数据，至少应该包含一个main.yml的文件，文件中定义当前角色的特殊设定及其依赖关系，

该目录下也可以有其他.yml文件，但是需要在main.yml文件中用**include**指令将其他.yml文件包含进来

**default**/

默认变量存放的目录，至少应该包含一个main.yml的文件，文件中定义了此角色使用的默认变量，该目录下也

可以有其他.yml文件，但是需要在main.yml文件中用**include**指令将其他.yml文件包含进来

####### 除了tasks目录，上述目录结构并非每个都必须，而是根据实际需要进行创建 #######

### 3、在playbook中调用角色方法一：

如何利用定义的角色，在某些主机上完成某些任务：此时就需要调用相关的角色了

如：(相关示例详见实战部分)

- hosts: webserver

remote\_user: root

roles:

- mysql

- httpd

可以只调用一个角色，也可以调用多个角色，当定义了角色后，用ansible-playbook PLAYBBOOK文件 执行即可

此时ansible会到角色集合的目录(默认/etc/ansible/roles/ 目录)去找 roles:调用的角色，也就是在角色集合

目录下找同名的子目录，将子目录下的所有代码运行一遍

### 4、在playbook中调用角色方法二：(在角色调用时传递变量)

如：(相关示例详见实战部分)

- hosts: webserver

remote\_user: root

roles:

- {role: mysql,var1: value1,var2: value2,...}

- {role: httpd,var3: value3,var4: value4,...}

表示调用两个角色，(role键用于指定调用的角色名称，后续的key/value用于传递变量给角色，每个键后面对应的值之间有空格)

一个角色是mysql，向该角色传递变量var1，其值为value1，...

调用另一个角色httpd，向该角色传递变量var3，其值为value3，...

### 5、在playbook中调用角色时，实现条件判断：

如：(相关示例详见实战部分)

**-** **hosts**: **webserver**

**remote\_user**: **root**

**roles**:

**-** {**role**: mysql,var1: value1,when:ansible\_distribution\_major\_version=='7'}

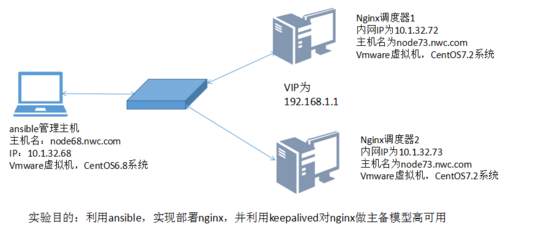
**-** {**role**: httpd,when:ansible\_distribution\_major\_version=='6'}

表示当**ansible\_distribution\_major\_version**的值为7时，调用**mysql**角色,传递变量**var1**，变量值为**value1**

表示当**ansible\_distribution\_major\_version**的值为6时，调用**httpd**角色

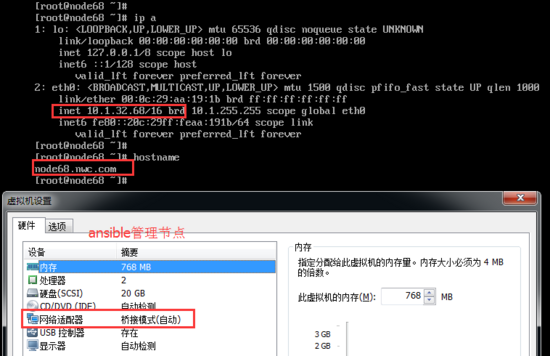
## 实战一：利用ansible配置主备模型的keepalived+nginx

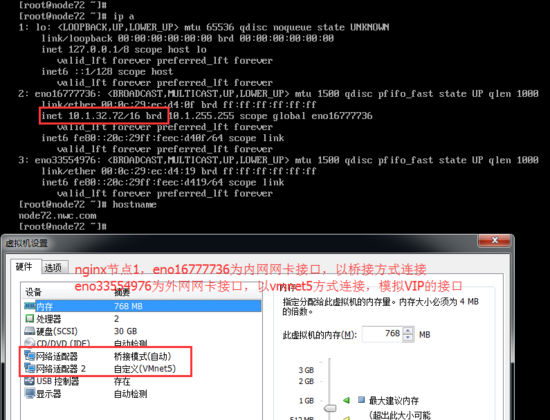
### 1、实验环境

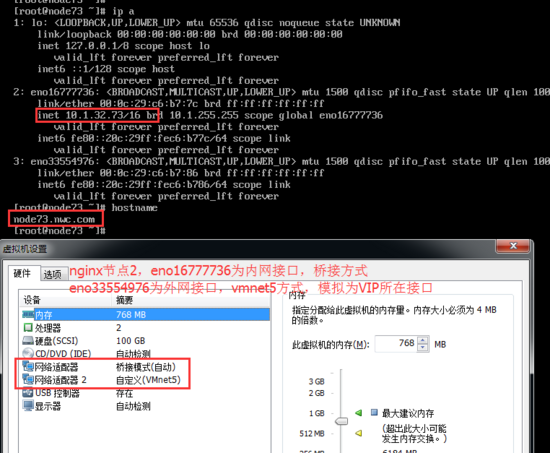


### 2、实验前准备工作

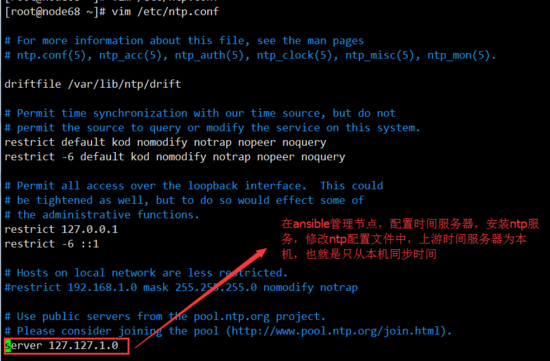
<1> 配置好各个节点之间的网络环境







<2> 各个节点之间时间同步

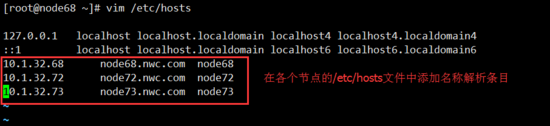


http://img1.tuicool.com/2uMRne7.png!web





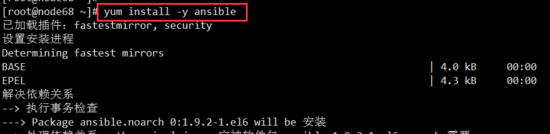
<3> 配置各个节点之间，可基于主机名解析，且解析结果与实际主机名一致



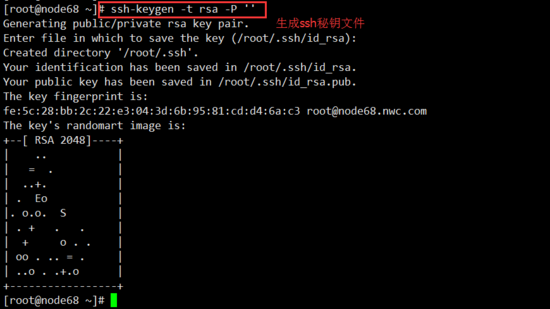
<4> 确保iptables和selinux不会影响实验正常进行

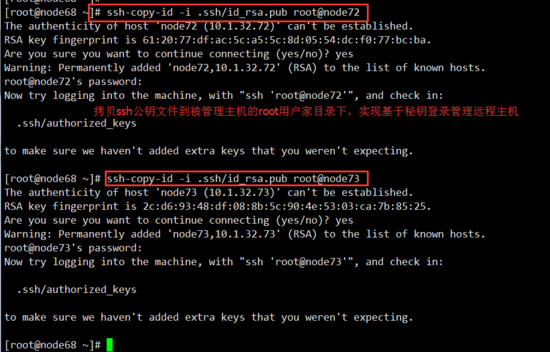


<5> 在ansible管理节点上部署ansible

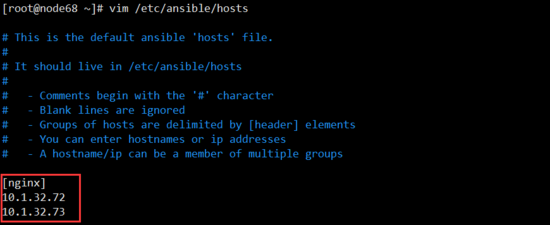


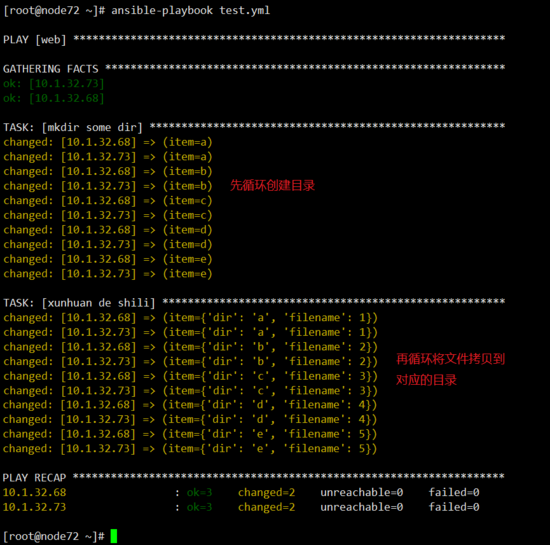
<6> 配置ansible主机可基于ssh秘钥登录被管理主机的root用户

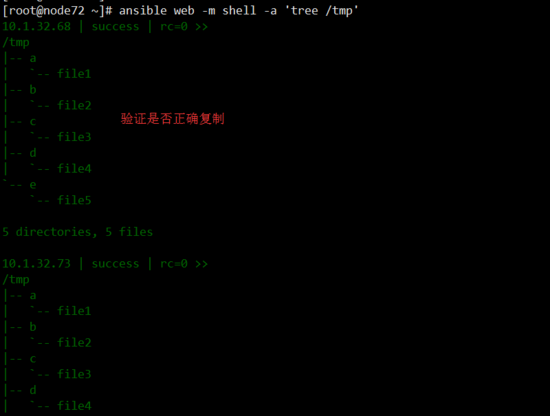




<7> 为ansible配置被管理主机







### 3、在ansible主机上利用ansible的roles功能，在两台被管理主机上安装和配置nginx

[root@node68 ~]# \*\*\*\*\*\* 创建角色的目录结构 \*\*\*\*\*\*

[root@node68 ~]# mkdir /etc/ansible/roles/nginx/{files,templates,tasks,handlers,vars,default,meta} -pv

mkdir: 已创建目录 "/etc/ansible/roles/nginx"

mkdir: 已创建目录 "/etc/ansible/roles/nginx/files"

mkdir: 已创建目录 "/etc/ansible/roles/nginx/templates"

mkdir: 已创建目录 "/etc/ansible/roles/nginx/tasks"

mkdir: 已创建目录 "/etc/ansible/roles/nginx/handlers"

mkdir: 已创建目录 "/etc/ansible/roles/nginx/vars"

mkdir: 已创建目录 "/etc/ansible/roles/nginx/default"

mkdir: 已创建目录 "/etc/ansible/roles/nginx/meta"

[root@node68 ~]#

[root@node68 ~]#

[root@node68 ~]# \*\*\*\*\*\* 编辑tasks文件 \*\*\*\*\*\*

[root@node68 ~]# cat /etc/ansible/roles/nginx/tasks/main.yml

- name: install a wget tool

yum: name=wget state=present

tags:

- anzhuang wget

- name: download nginx rpm package

shell: chdir=/root wget ftp://10.1.0.1/pub/Sources/7.x86\_64/nginx/nginx-1.10.0-1.el7.ngx.x86\_64.rpm

- name: install nginx

shell: chdir=/root rpm -i nginx-1.10.0-1.el7.ngx.x86\_64.rpm

tags:

- anzhuang nginx

- name: provide a config file

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

notify:

- restart nginx

- mail **to** root

tags:

- config **and** restart

- name: move the **default** index page

shell: mv /usr/share/nginx/html/index.html /usr/share/nginx/html/index.html.bak

tags:

- move **default** page **and** provide a **new** page

- name: provied a index page

template: src=index.html.j2 dest=/usr/share/nginx/html/index.html

tags:

- move **default** page **and** provide a **new** page

[root@node68 ~]#

[root@node68 ~]# \*\*\*\*\*\* 因为tasks中定义了通知机制，故要编辑handler文件 \*\*\*\*\*\*

[root@node68 ~]# cat /etc/ansible/roles/nginx/handlers/main.yml

- name: restart nginx

shell: service nginx restart

- name: mail **to** root

shell: echo "nginx config file has been changed" | mail -s "nginx config file changed" root@localhost

[root@node68 ~]#

[root@node68 ~]# \*\*\*\*\*\* 因为tasks中定义了template模块相关任务，故要编辑生成template模板文件 \*\*\*\*\*\*

[root@node68 ~]# ls /etc/ansible/roles/nginx/templates/

index.html.j2 nginx.conf.j2

[root@node68 ~]# \*\*\*\*\*\* nginx的模板配置文件 \*\*\*\*\*\*

[root@node68 ~]# cat /etc/ansible/roles/nginx/templates/nginx.conf.j2

user nginx;

worker\_processes {{ ansible\_processor\_vcpus }};

###### worker进程的个数为ansible\_processor\_vcpus变量的值，表示与被管理主机上的cpu个数相等 ######

error\_log /**var**/log/nginx/error.log warn;

pid /**var**/run/nginx.pid;

events {

worker\_connections 1024;

}

http {

include /etc/nginx/mime.types;

default\_type application/octet-stream;

log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

'$status $body\_bytes\_sent "$http\_referer" '

'"$http\_user\_agent" "$http\_x\_forwarded\_for"';

access\_log /**var**/log/nginx/access.log main;

sendfile on;

#tcp\_nopush on;

keepalive\_timeout 65;

#gzip on;

server {

listen 80;

server\_name {{ ansible\_fqdn }};

###### server\_name为ansible\_fqdn变量的值 ######

#charset koi8-r;

#access\_log /var/log/nginx/log/host.access.log main;

location / {

root /usr/share/nginx/html;

index index.html index.htm;

}

#error\_page 404 /404.html;

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root /usr/share/nginx/html;

}

}

}

[root@node68 ~]#

[root@node68 ~]#

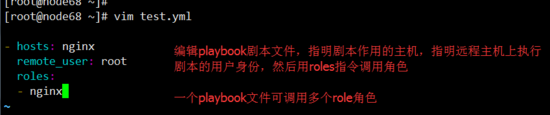
[root@node68 ~]# \*\*\*\*\*\* 提供的默认主页的文件模板 \*\*\*\*\*\*

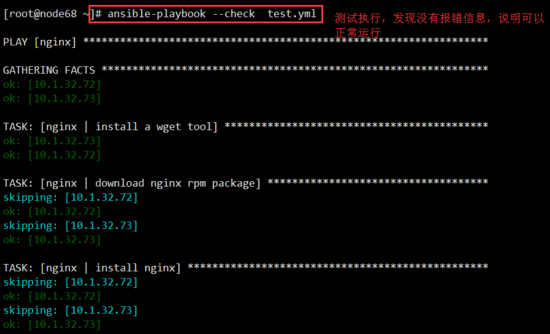
[root@node68 ~]# cat /etc/ansible/roles/nginx/templates/index.html.j2

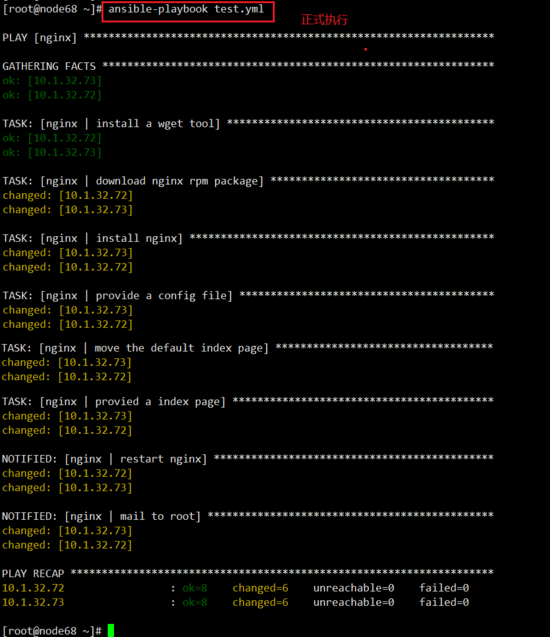
<h1>This **is** {{ ansible\_fqdn }} index page IP **is** {{ ansible\_all\_ipv4\_addresses }}</h1>

[root@node68 ~]#

4、编辑playbook文件，调用角色，执行剧本，让被管理主机通过roles定义的方式，完成nginx安装和配置

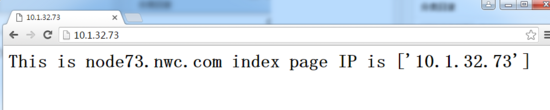






### 5、验证被管理主机上nginx是否运行正常







### 6、编辑生成keepalived的roles角色，和相关配置文件

[root@node68 ~]# \*\*\*\*\* 创建角色工作目录 \*\*\*\*\*\*

[root@node68 ~]# mkdir -pv /etc/ansible/roles/keepalived/{files,templates,tasks,handlers,vars}

**mkdir**: 已创建目录 "/etc/ansible/roles/keepalived"

**mkdir**: 已创建目录 "/etc/ansible/roles/keepalived/files"

**mkdir**: 已创建目录 "/etc/ansible/roles/keepalived/templates"

**mkdir**: 已创建目录 "/etc/ansible/roles/keepalived/tasks"

**mkdir**: 已创建目录 "/etc/ansible/roles/keepalived/handlers"

**mkdir**: 已创建目录 "/etc/ansible/roles/keepalived/vars"

[root@node68 ~]#

[root@node68 ~]# \*\*\*\*\*\* 创建tasks任务列表 \*\*\*\*\*\*

[root@node68 ~]# cat /etc/ansible/roles/keepalived/tasks/main.yml

- name: install keepalived **package**

yum: name=keepalived **state**=present

- name: move default config file

shell: mv /etc/keepalived/keepalived.conf /etc/keepalived/keepalived.conf.bak

tags:

- move old config file

- name: provide a config file **for** master

template: src=keepalived.conf.master.j2 dest=/etc/keepalived/keepalived.conf

when: {{ is\_master }} == "yes"

notify:

- mail to root

tags:

- provide a config file

- name: provide a config file for backup

template: src=keepalived.conf.backup.j2 dest=/etc/keepalived/keepalived.conf

**when**: {{ is\_master }} == "no"

notify:

- mail to root

tags:

- provide a config file

- name: restart keepalived

shell: systemctl restart keepalived.service

tags:

- restart keepalived

[root@node68 ~]#

[root@node68 ~]# \*\*\*\*\*\* 由于在tasks中定义了 notify，故定义相应的handlers \*\*\*\*\*\*

[root@node68 ~]# cat /etc/ansible/roles/keepalived/handlers/main.yml

- name: mail to root

shell: echo "keepalived on {{ ansible\_all\_ipv4\_addresses }} config file has been changed" | mail -**s** "keepalived changed" root@localhost

[root@node68 ~]#

[root@node68 ~]# \*\*\*\*\*\* 由于在task中定义了template模块，故提供模板文件 \*\*\*\*\*\*\*

[root@node68 ~]# ls /etc/ansible/roles/keepalived/templates/

keepalived.conf.backup.j2 keepalived.conf.master.j2

[root@node68 ~]#

[root@node68 ~]#

[root@node68 ~]# \*\*\*\*\* 为keepalived的master节点提供的模板配置文件 \*\*\*\*\*\*

[root@node68 ~]# cat /etc/ansible/roles/keepalived/templates/keepalived.conf.master.j2

! Configuration File **for** keepalived

global\_defs {

notification\_email {

root@localhost

}

notification\_email\_from keepalivedadmin@nwc.com

smtp\_server 127.0.0.1

smtp\_connect\_timeout 30

router\_id {{ ansible\_hostname }}

vrrp\_mcast\_group4 224.0.32.18

}

vrrp\_script chk\_nginx {

script "killall -0 nginx"

interval 2

weight -5

}

vrrp\_instance VI\_1 {

**state** MASTER

interface eno16777736 #####此为心跳信息传递的接口，可以与VIP的接口不一样#####

virtual\_router\_id 32

priority 100

advert\_int 2

authentication {

auth\_type PASS

auth\_pass 123456

}

track\_script {

chk\_nginx

}

virtual\_ipaddress {

192.168.1.1/24 dev eno33554976

}

track\_interface {

eno16777736

eno33554976

}

}

[root@node68 ~]#

[root@node68 ~]#

[root@node68 ~]# \*\*\*\*\* 为keepalived的backup节点提供的模板配置文件 \*\*\*\*\*\*

[root@node68 ~]# cat /etc/ansible/roles/keepalived/templates/keepalived.conf.backup.j2

! Configuration File **for** keepalived

global\_defs {

notification\_email {

root@localhost

}

notification\_email\_from keepalivedadmin@nwc.com

smtp\_server 127.0.0.1

smtp\_connect\_timeout 30

router\_id {{ ansible\_hostname }}

vrrp\_mcast\_group4 224.0.32.18

}

vrrp\_script chk\_nginx {

script "killall -0 nginx"

interval 2

weight -5

}

vrrp\_instance VI\_1 {

**state** BACKUP

interface eno16777736 #####此为心跳信息传递的接口，可以与VIP的接口不一样#####

virtual\_router\_id 32

priority 98

advert\_int 2

authentication {

auth\_type PASS

auth\_pass 123456

}

track\_script {

chk\_nginx

}

virtual\_ipaddress {

192.168.1.1/24 dev eno33554976

}

track\_interface {

eno16777736

eno33554976

}

}

[root@node68 ~]#

[root@node68 ~]# \*\*\*\*\* 由于在task任务列表文件中定义了变量，判断当前节点是否是主节点的操作 \*\*\*\*\*\*

[root@node68 ~]# \*\*\*\*\* 故针对每个节点定义其是否为主节点的变量 is\_master \*\*\*\*\*\*

[root@node68 ~]#

[root@node68 ~]# cat /etc/ansible/hosts

# This is the default ansible 'hosts' file.

#

# It should live in /etc/ansible/hosts

#

# - Comments begin with the '#' character

# - Blank lines are ignored

# - Groups of hosts are delimited by [header] elements

# - You can enter hostnames or ip addresses

# - A hostname/ip can be a member of multiple groups

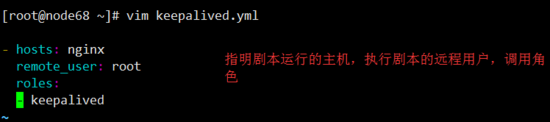
[nginx]

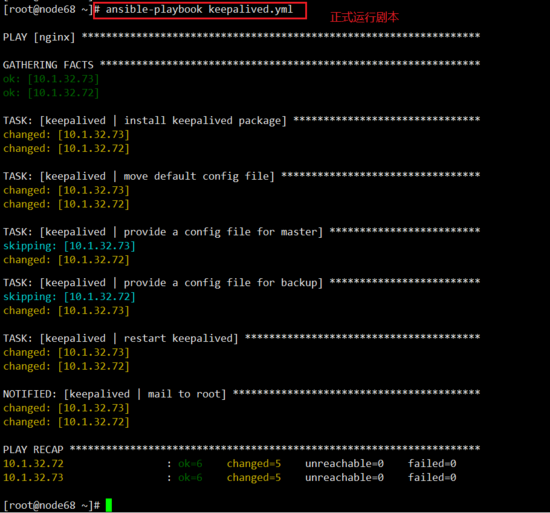
10.1.32.72 is\_master=yes

10.1.32.73 is\_master=**no**

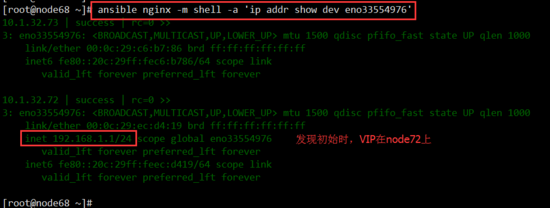
[root@node68 ~]#

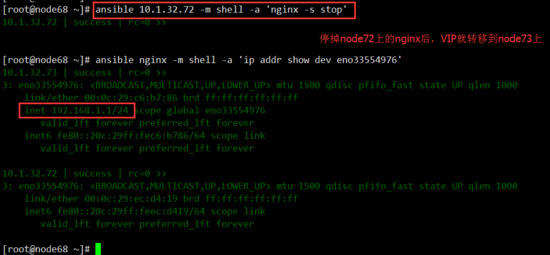
### 7、编辑playbook剧本文件，运行剧本





### 8、验证keepalived对nginx的高可用是否成功





## 实战二：实战一的基础上在nginx后端提供httpd+php+php-mysql

### 1、实验环境

在实战一的基础上，为nginx提供后端提供httpd+php+php-mysql，本实验中，将httpd，php，php-mysql均部署在

原有的nginx两个节点上，将httpd的监听端口改为8080，nginx继续监听80端口，修改nginx的配置文件，让nginx

接受到的请求均反代到httpd服务上进行处理

用的实验环境是实验一的环境，故相关准备工作参照实验一

### 2、利用ansible的roles，编辑roles相关配置

[root@node68 ~]# \*\*\*\*\*\* 创建lap角色目录 \*\*\*\*\*\*

[root@node68 ~]# mkdir -pv /etc/ansible/roles/lap/{files,templates,tasks,handlers,vars}

**mkdir**: 已创建目录 "/etc/ansible/roles/lap"

**mkdir**: 已创建目录 "/etc/ansible/roles/lap/files"

**mkdir**: 已创建目录 "/etc/ansible/roles/lap/templates"

**mkdir**: 已创建目录 "/etc/ansible/roles/lap/tasks"

**mkdir**: 已创建目录 "/etc/ansible/roles/lap/handlers"

**mkdir**: 已创建目录 "/etc/ansible/roles/lap/vars"

[root@node68 ~]# \*\*\*\*\* 编辑生成tasks任务列表文件 \*\*\*\*\*\*

[root@node68 ~]# cat /etc/ansible/roles/lap/tasks/main.yml

- name: install httpd

yum: name=httpd **state**=present

- name: install php

yum: name=php **state**=present

- name: install php-mysql

yum: name=php-mysql **state**=present

- name: make sure nginx proxy the request to httpd

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

notify:

- reload nginx

tags:

- nginx proxy

- name: move old httpd config file

shell: mv /etc/httpd/conf/httpd.conf /etc/httpd/conf/httpd.conf.bak

tags:

- move old config

- name: provide a httpd config file

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

notify:

- restart http

- name: provide index page

template: src=index.html.j2 dest=/var/www/html/index.html

- name: restart httpd service

shell: systemctl restart httpd

[root@node68 ~]#

[root@node68 ~]# \*\*\*\*\* 创建handlers触发器文件 \*\*\*\*\*

[root@node68 ~]# cat /etc/ansible/roles/lap/handlers/main.yml

- name: reload nginx

shell: systemctl restart nginx

- name: restart http

shell: systemctl restart httpd

[root@node68 ~]# \*\*\*\*\*\* 根据tasks中定义的模板文件，提供对应的template模板文件 \*\*\*\*\*

[root@node68 ~]# ls /etc/ansible/roles/lap/templates/

httpd.conf.j2 index.html.j2 nginx.new.conf.j2

[root@node68 ~]#

[root@node68 ~]# \*\*\*\*\*\* 提供给远程主机的httpd的默认主页面的文件 \*\*\*\*\*\*

[root@node68 ~]# cat /etc/ansible/roles/lap/templates/index.html.j2

<h1> This is {{ ansible\_nodename }} index page </h1>

[root@node68 ~]#

[root@node68 ~]#

[root@node68 ~]# \*\*\*\*\*\*\* 修改的nginx配置文件的模板文件 \*\*\*\*\*\*

[root@node68 ~]# cat /etc/ansible/roles/lap/templates/nginx.new.conf.j2

user nginx;

worker\_processes {{ ansible\_processor\_vcpus }};

error\_log /var/**log**/nginx/error.log **warn**;

pid /var/run/nginx.pid;

events {

worker\_connections 1024;

}

http {

include /etc/nginx/mime.types;

default\_type application/octet-stream;

log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

'$status $body\_bytes\_sent "$http\_referer" '

'"$http\_user\_agent" "$http\_x\_forwarded\_for"';

access\_log /var/**log**/nginx/access.log main;

sendfile on;

#tcp\_nopush on;

keepalive\_timeout 65;

#gzip on;

##### 定义后端主机 #####

upstream web {

server 10.1.32.72:8080 max\_fails=2;

server 10.1.32.73:8080 max\_fails=2;

}

server {

**listen** 80;

server\_name {{ ansible\_fqdn }};

##### 定义将所有请求反代到后端主机的8080端口 #####

location / {

root /usr/share/nginx/html;

**index** index.html index.htm;

proxy\_pass http://web;

}

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root /usr/share/nginx/html;

}

}

}

[root@node68 ~]#

[root@node68 ~]#

[root@node68 ~]# \*\*\*\*\* 提供httpd的配置文件的模板文件 \*\*\*\*\*\*

[root@node68 ~]# cat /etc/ansible/roles/lap/templates/httpd.conf.j2

ServerRoot "/etc/httpd"

Listen 8080 ##### 修改监听端口为8080 #####

Include conf.modules.d/\*.conf

User apache

Group apache

ServerAdmin root@localhost

ServerName {{ ansible\_nodename }} ##### ServerName的值修改为远程主机的主机名 ######

<Directory />

AllowOverride none

Require all denied

</Directory>

DocumentRoot "/var/www/html"

<Directory "/var/www">

AllowOverride None

# Allow open access:

Require all granted

</Directory>

<Directory "/var/www/html">

Options Indexes FollowSymLinks

AllowOverride None

Require all granted

</Directory>

<IfModule dir\_module>

DirectoryIndex index.html

</IfModule>

<Files ".ht\*">

Require all denied

</Files>

ErrorLog "logs/error\_log"

LogLevel warn

<IfModule log\_config\_module>

LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined

LogFormat "%h %l %u %t \"%r\" %>s %b" common

<IfModule logio\_module>

# You need to enable mod\_logio.c to use %I and %O

LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" %I %O" combinedio

</IfModule>

CustomLog "logs/access\_log" combined

</IfModule>

<IfModule alias\_module>

ScriptAlias /cgi-bin/ "/var/www/cgi-bin/"

</IfModule>

<Directory "/var/www/cgi-bin">

AllowOverride None

Options None

Require all granted

</Directory>

<IfModule mime\_module>

AddType application/x-compress .Z

AddType application/x-gzip .gz .tgz

AddType text/html .shtml

AddOutputFilter INCLUDES .shtml

</IfModule>

AddDefaultCharset UTF-8

<IfModule mime\_magic\_module>

MIMEMagicFile conf/magic

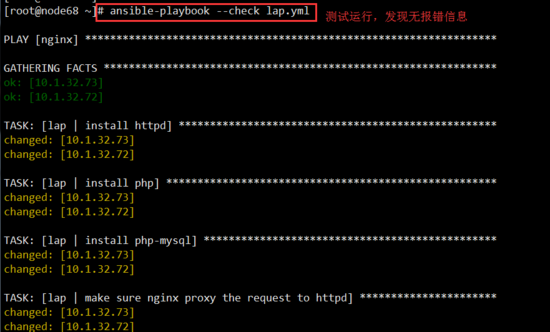
</IfModule>

EnableSendfile on

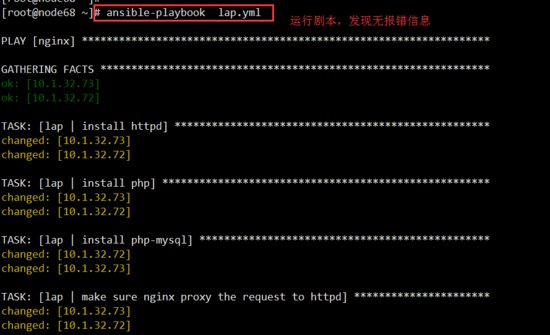
IncludeOptional conf.d/\*.conf

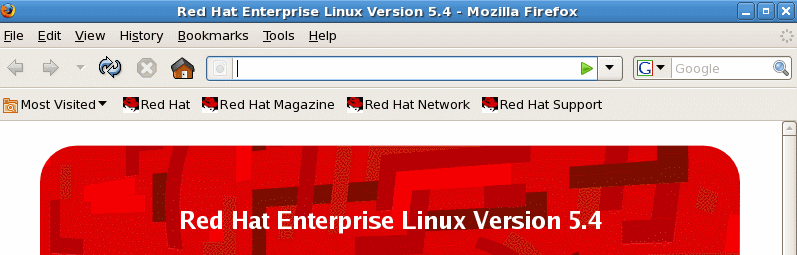
### 3、编辑playbook文件，引用角色，测试运行，检测有无错误信息

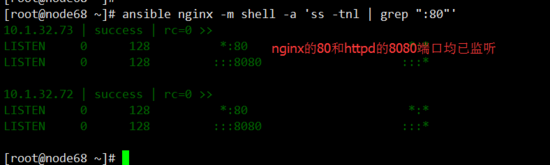




### 4、运行剧本，验证反代是否成功







## 实战三：在此前实验基础上配置mysql服务

### 1、实验环境

在实验一和实验二的基础上，部署一个后端**mysql**服务器，并启动

配置**mysql**服务器拥有**testdb**库，并允许**testuser**对其拥有所有权限

本实验继续利用实验一的环境，在**node73**，也就是10.1.32.73这台主机上安装**mariadb**服务

相关准备工作的流程，详见实验一的准备工作部分

### 2、编写ansible的roles角色的相关内容

[root@node68 ~]# \*\*\*\*\* 为ansible管理端添加db主机组 \*\*\*\*\*

[root@node68 ~]# cat /etc/ansible/hosts

# This is the default ansible 'hosts' file.

#

# It should live in /etc/ansible/hosts

#

# - Comments begin with the '#' character

# - Blank lines are ignored

# - Groups of hosts are delimited by [header] elements

# - You can enter hostnames or ip addresses

# - A hostname/ip can be a member of multiple groups

[nginx]

10.1.32.72 is\_master=yes

10.1.32.73 is\_master=**no**

[db]

10.1.32.73

[root@node68 ~]#

[root@node68 ~]#

[root@node68 ~]# \*\*\*\*\* 创建mariadb角色的相关目录结构 \*\*\*\*\*

[root@node68 ~]# mkdir -pv /etc/ansible/roles/mariadb/{files,templates,tasks,handlers,vars}

**mkdir**: 已创建目录 "/etc/ansible/roles/mariadb"

**mkdir**: 已创建目录 "/etc/ansible/roles/mariadb/files"

**mkdir**: 已创建目录 "/etc/ansible/roles/mariadb/templates"

**mkdir**: 已创建目录 "/etc/ansible/roles/mariadb/tasks"

**mkdir**: 已创建目录 "/etc/ansible/roles/mariadb/handlers"

**mkdir**: 已创建目录 "/etc/ansible/roles/mariadb/vars"

[root@node68 ~]#

[root@node68 ~]#

[root@node68 ~]# \*\*\*\*\* 提供mariadb角色的tasks任务文件 \*\*\*\*\*

[root@node68 ~]# cat /etc/ansible/roles/mariadb/tasks/main.yml

- name: install mariadb **package**

yum: name=mariadb-server **state**=present

- name: move old config file

shell: mv /etc/my.cnf /etc/my.cnf.bak

- name: provide a config file

copy: src=my.cnf dest=/etc/my.cnf

notify:

- restart mariadb

- name: create a testdb

shell: mysql -uroot -e "CREATE DATABASE testdb;GRANT ALL ON testdb.\* TO 'testuser'@'10.1.%.%' IDENTIFIED BY '111111';FLUSH PRIVILEGES;"

[root@node68 ~]#

[root@node68 ~]#

[root@node68 ~]# \*\*\*\*\*\* 由于tasks定义了notify，故提供handlers触发器的文件 \*\*\*\*\*\*

[root@node68 ~]# cat /etc/ansible/roles/mariadb/handlers/main.yml

- name: restart mariadb

shell: systemctl restart mariadb

[root@node68 ~]#

[root@node68 ~]#

[root@node68 ~]# #### 提供mariadb的样例配置文件 #####

[root@node68 ~]# cat /etc/ansible/roles/mariadb/files/my.cnf

[mysqld]

datadir=/var/lib/mysql

socket=/var/lib/mysql/mysql.sock

symbolic-links=0

#### 修改的配置文件 ######

skip\_name\_resolve = ON

innodb\_file\_per\_table = ON

[mysqld\_safe]

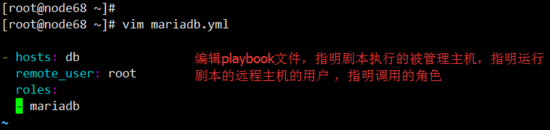
log-error=/var/log/mariadb/mariadb.log

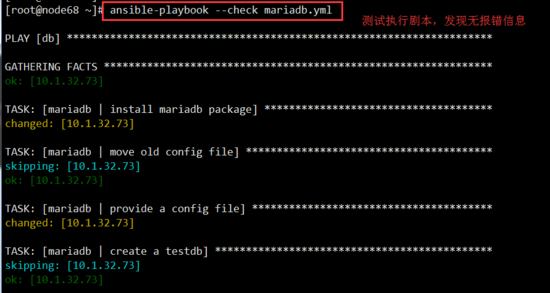
pid-file=/var/run/mariadb/mariadb.pid

!includedir /etc/my.cnf.d

[root@node68 ~]#

### 3、编辑生成playbook文件，引用角色，测试执行剧本，查看是否有报错





### 4、执行剧本，验证配置是否正确

