

# ansible配置详解

---

## 概述

---

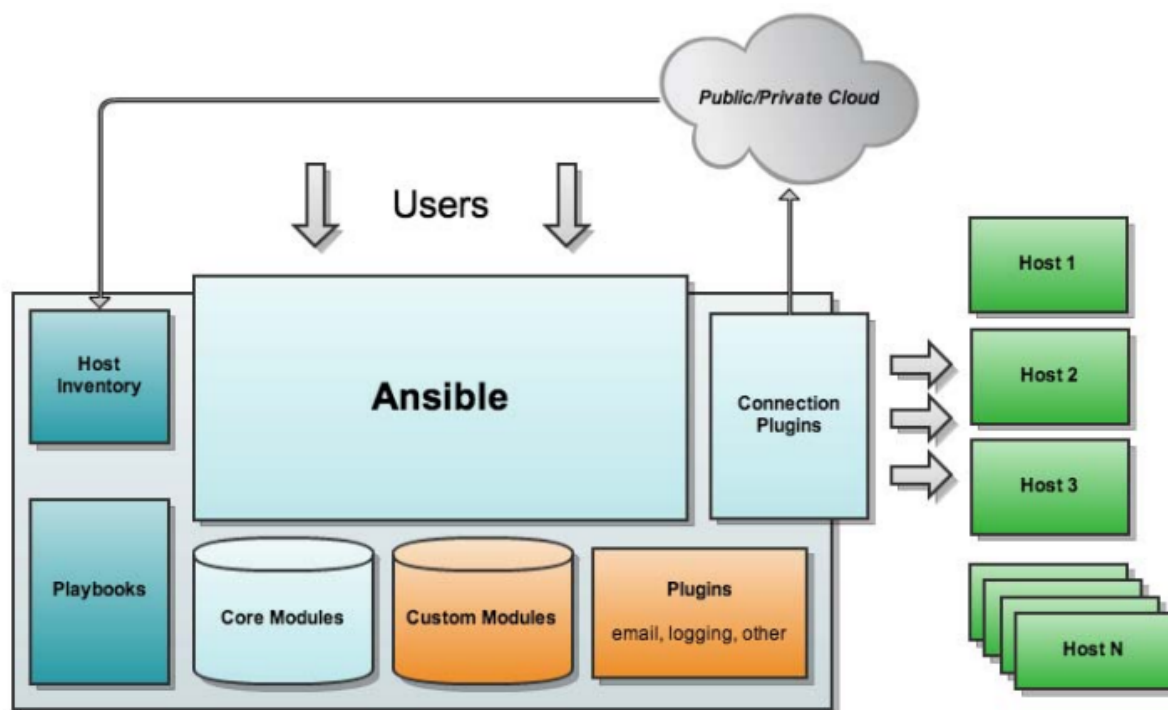
**ansible**是一款无需在被管理主机上安装客户端，基于**SSH**对多台目标主机进行同时操作的轻量级的管理软件，借助各个内部的功能模块，实现了批量系统配置、批量程序部署、批量运行命令等功能。本篇就介绍一些关于**ansible**的基础配置相关的内容，具体包括：

- 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

---

### 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

```
[root@node72 ~]# yum install -y ansible
已加载插件: fastestmirror, langpacks
Repodata is over 2 weeks old. Install yum-cron? Or run: yum makecache fast
BASE | 3.6 kB 00:00:00
EPEL | 4.3 kB 00:00:00
Loading mirror speeds from cached hostfile
正在解决依赖关系
--> 正在检查事务
```

安装生成的文件

```
[root@node72 ~]# rpm -ql ansible|less

/etc/ansible /etc/ansible/ansible.cfg ansible程序的配置文件
/etc/ansible/ansible.cfg /etc/ansible/hosts ansible定义的被管理主机的清单文件
/etc/ansible/hosts /etc/ansible/roles 角色的目录
/etc/ansible/roles /usr/bin/ansible 主程序，命令行工具
/usr/bin/ansible /usr/bin/ansible-doc 查看ansible的配置帮助文件的程序
/usr/bin/ansible-doc /usr/bin/ansible-playbook 用来运行ansible剧本的程序
/usr/bin/ansible-galaxy
/usr/bin/ansible-playbook 还有许多python的模块文件，因为ansible是基于python所研发的
/usr/bin/ansible-pull
/usr/bin/ansible-vault
/usr/lib/python2.7/site-packages/ansible
/usr/lib/python2.7/site-packages/ansible-1.9.2-py2.7.egg-info
/usr/lib/python2.7/site-packages/ansible-1.9.2-py2.7.egg-info/PKG-INFO
/usr/lib/python2.7/site-packages/ansible-1.9.2-py2.7.egg-info/SOURCES.txt
/usr/lib/python2.7/site-packages/ansible-1.9.2-py2.7.egg-info/dependency_links.txt
/usr/lib/python2.7/site-packages/ansible-1.9.2-py2.7.egg-info/requires.txt
/usr/lib/python2.7/site-packages/ansible-1.9.2-py2.7.egg-info/top_level.txt
/usr/lib/python2.7/site-packages/ansible/__init__.py
/usr/lib/python2.7/site-packages/ansible/__init__.pyc
/usr/lib/python2.7/site-packages/ansible/__init__.pyo
/usr/lib/python2.7/site-packages/ansible/cache
/usr/lib/python2.7/site-packages/ansible/cache/__init__.py
/usr/lib/python2.7/site-packages/ansible/cache/__init__.pyc
```

## 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

```

# Ex 1: Ungrouped hosts, specify before any group headers.
green.example.com
blue.example.com
192.168.100.1
192.168.100.10

# Ex 2: A collection of hosts belonging to the 'webservers' group
[webservers]
alpha.example.org
beta.example.org
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
# Ex 3: A collection of database servers in the 'dbservers' group
[dbservers]
db01.intranet.mydomain.net
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

```

定义的格式可以是主机名，或IP

可以将多个主机，定义为一个主机组，由[GROUP\_NAME]来进行标识

也可以是一个范围，此例表示www001.example.com到www006.example.com范围内的所有主机

此例表示db-990-node.example.com到db-101-node.example.com范围内的所有主机

```

# - 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

```

### 本例中的配置

```

[web]
10.1.32.68
10.1.32.73

[db]
10.1.32.73
10.1.32.168

```

## 3、配置管理节点可以基于ssh密钥登录被管理节点

```
[root@node72 ~]# ssh-keygen -t rsa -P ''
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
4c:9b:00:ca:1a:ae:1c:aa:29:c7:7c:55:49:66:e8:a7 root@node72.nwc.com
The key's randomart image is:
+--[ RSA 2048 ]-----+
|      .      |
|    . . . . + |
| . 0  . . + . |
| .0   . + + 0 |
| .0     + S   |
|+ .    E     |
| O+ .       |
| O.+ .      |
|+. .        |
+-----+
[root@node72 ~]#
```

在管理节点上生成SSH的秘钥文件

```
[root@node72 ~]# ssh-copy-id -i .ssh/id_rsa.pub root@10.1.32.73
The authenticity of host '10.1.32.73 (10.1.32.73)' can't be established.
ECDSA key fingerprint is a0:c4:9d:67:22:a1:14:d6:73:79:30:5f:8c:0d:d0:b4.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install t
root@10.1.32.73's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'root@10.1.32.73'"
and check to make sure that only the key(s) you wanted were added.

[root@node72 ~]# ssh 10.1.32.73
Last login: Wed Nov  2 10:57:11 2016 from 10.1.32.88
[root@node73 ~]#
[root@node73 ~]# cat .ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQAC91NU5bX+L8teP0TKIgr6cA7tUqoEwoRI5DBtXc0BnCBtEnaRs3c9aaxSR9PF+R/Pv9
Vj28iP6+Ms+nj/xd4aPNJwQQu2MGYIAAnS5wFnmic0ERZuszYFjJ35B/rBfcMebS25I0mT77+fNQAAJDEQw19sPJHLyRc/E589943q19yG
bWfNamM3Zri85B5EpAxziCoNwgVw+3ADybtIxrHrQHMFv0qB60n/PTDW9/rsAuU1Gh root@node72.nwc.com
[root@node73 ~]#
[root@node73 ~]# exit
登出
Connection to 10.1.32.73 closed.
[root@node72 ~]#
```

利用命令拷贝生成的公钥文件到被管理节点上  
需要免密码登录的用户的家目录下

测试基于ssh秘钥登录被管理主机

查看拷贝过来的公钥文件

## 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@node72 ~]#  
[root@node72 ~]# ansible web -m ping 对web组内的主机, 利用ping模块, 探测主机是否在线, 由于ping模块没有参数, 故直接给明模块即可  
10.1.32.73 | success >> {  
    "changed": false,  
    "ping": "pong"  
}  
  
10.1.32.68 | success >> {  
    "changed": false,  
    "ping": "pong"  
} 对web组内的主机, 利用command模块, -a指明传递给模块的参数, 此处相当于给明需要执行的命令  
  
[root@node72 ~]# ansible web -m command -a 'cat /root/.ssh/authorized_keys'  
10.1.32.73 | success | rc=0 >>  
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQAC91NU5bX+L8teP0TKIgr6cA7tUqoEwoRI5DBtXc0BnCBtErVj28iP6+Ms+nj/xd4aPNJwQQU2MGYIAnS5wFnmic0ERZuszYFjJ35B/rBfcMebsZ5I0mT77+fNQAADJDEQw19sbWfNamM3Zri85B5EpAxziCoNwgVw+3ADybtIxrHrQHMfv0qB60n/PTDW9/rsAuU1Gh root@node72.nwc.cc  
  
10.1.32.68 | success | rc=0 >>  
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQAC91NU5bX+L8teP0TKIgr6cA7tUqoEwoRI5DBtXc0BnCBtErVj28iP6+Ms+nj/xd4aPNJwQQU2MGYIAnS5wFnmic0ERZuszYFjJ35B/rBfcMebsZ5I0mT77+fNQAADJDEQw19sbWfNamM3Zri85B5EpAxziCoNwgVw+3ADybtIxrHrQHMfv0qB60n/PTDW9/rsAuU1Gh root@node72.nwc.cc  
  
[root@node72 ~]#
```

## 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@node72 ~]#
[root@node72 ~]# ansible-doc -s command 查看command模块的使用帮助信息
less 458 (POSIX regular expressions)
Copyright (C) 1984-2012 Mark Nudelman

less comes with NO WARRANTY, to the extent permitted by law.
For information about the terms of redistribution,
see the file named README in the less distribution.
Homepage: http://www.greenwoodsoftware.com/less
- name: Executes a command on a remote node
  action: command
    chdir          # cd into this directory before running the command
    creates        # a filename, when it already exists, this step will *not* be run.
    executable     # change the shell used to execute the command. Should be an absolute
    free_form=     # the command module takes a free form command to run. There is no
    removes        # a filename, 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
[root@node72 ~]#
[root@node72 ~]#

```

## 2、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

executable= 指明运行命令的shell程序

```

[root@node72 ~]# ansible all -m command -a 'chdir=/tmp ls ./'
10.1.32.73 | success | rc=0 >>
10.1.32.68 | success | rc=0 >>
yum.log
表示对all, 即所有被监控主机, 利用-m指明
利用command模块, -a指明参数, 也就是执
行命令

[root@node72 ~]# ansible all -m command -a 'ls /tmp'
10.1.32.68 | success | rc=0 >>
yum.log
10.1.32.73 | success | rc=0 >>

[root@node72 ~]#

```

```
[root@node72 ~]# ansible all -m command -a 'echo "hello" >> /tmp/a.txt'
```

10.1.32.68 | success | rc=0 >>  
hello >> /tmp/a.txt

10.1.32.73 | success | rc=0 >>  
hello >> /tmp/a.txt

command不支持管道，重定向等shell相关的特性

```
[root@node72 ~]# ansible all -m command -a 'cat /tmp/a.txt'
```

10.1.32.68 | FAILED | rc=1 >>  
cat: /tmp/a.txt: No such file or directory

10.1.32.73 | FAILED | rc=1 >>  
cat: /tmp/a.txt: No such file or directory

```
[root@node72 ~]#
```

### 3、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

executable= 指明运行命令的shell程序

```
[root@node72 ~]# ansible all -m shell -a 'echo "hello" >> /tmp/a.txt'
```

10.1.32.68 | success | rc=0 >>  
hello >> /tmp/a.txt

10.1.32.73 | success | rc=0 >>  
hello >> /tmp/a.txt

shell模块也是在被管控主机上执行命令，但是支持管道、重定向等特性

```
[root@node72 ~]# ansible all -m shell -a 'cat /tmp/a.txt'
```

10.1.32.68 | success | rc=0 >>  
hello

10.1.32.73 | success | rc=0 >>  
hello

```
[root@node72 ~]#
```



## 4、copy模块

拷贝ansible管理端的文件到远程主机的指定位置

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

**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@node72 ~]# ansible web -m copy -a 'src=test.txt dest=/tmp/ mode=777 owner=nobody group=root'
```

```
10.1.32.73 | success >> {
  "changed": true,
  "checksum": "5eba3b854df08c7125e55c593cc93519326c91a2",
  "dest": "/tmp/test.txt",
  "gid": 0,
  "group": "root",
  "md5sum": "f459c14ce4348b79d583fa679c1a14fb",
  "mode": "0777",
  "owner": "nobody",
  "size": 31,
  "src": "/root/.ansible/tmp/ansible-tmp-1470475871.69-167422691077704/source",
  "state": "file",
  "uid": 99
}
```

指明拷贝的源文件，指明拷贝到的目标目录  
**mode**为拷贝过去的权限，可选  
**owner**为拷贝过去的属主，可选  
**group**为拷贝过去的属组，可选

```
10.1.32.68 | success >> {
  "changed": true,
  "checksum": "5eba3b854df08c7125e55c593cc93519326c91a2",
  "dest": "/tmp/test.txt",
  "gid": 0,
  "group": "root",
  "md5sum": "f459c14ce4348b79d583fa679c1a14fb",
  "mode": "0777",
  "owner": "nobody"
```

```
[root@node72 ~]# ansible web -m shell -a 'ls -l /tmp'
10.1.32.68 | success | rc=0 >>
total 8
-rw-r--r--  1 root   root   6 Jul 13 08:14 a.txt
-rwxrwxrwx  1 nobody root  31 Jul 13 08:18 test.txt
-rw-----  1 root   root   0 Jul 13 00:16 yum.log

10.1.32.73 | success | rc=0 >>
total 8
-rw-r--r--  1 root   root   6 Nov  2 11:44 a.txt
-rwxrwxrwx  1 nobody root  31 Nov  2 11:49 test.txt

[root@node72 ~]#
```

```
[root@node72 ~]# ansible web -m copy -a 'content="I am NWC" dest=/tmp/test.txt'
10.1.32.73 | success >> {
  "changed": true,
  "checksum": "1ea7fa2df3126c29354ed0f72926dc71ed95fa84",
  "dest": "/tmp/test.txt",
  "gid": 0,
  "group": "root",
  "md5sum": "dc177bf7935393915a69d2c53384a7d4",
  "mode": "0777",
  "owner": "nobody",
  "size": 8,
  "src": "/root/.ansible/tmp/ansible-tmp-1470476214.42-154467459587699/source",
  "state": "file",
  "uid": 99
}

10.1.32.68 | success >> {
  "changed": true,
  "checksum": "1ea7fa2df3126c29354ed0f72926dc71ed95fa84",
  "dest": "/tmp/test.txt",
```

相当于复制数据到目标文件中

注意：会覆盖原有文件的内容

```
[root@node72 ~]# ansible web -m shell -a 'cat /tmp/test.txt'
10.1.32.73 | success | rc=0 >>
I am NWC

10.1.32.68 | success | rc=0 >>
I am NWC

[root@node72 ~]#
```

## 5、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@node72 ~]# ansible web -m cron -a 'minute=*/10 hour=9-17 day=20-30/2 name="test cron" job="ntpddate 10.1.0.1 &> /dev/null"'
10.1.32.68 | success >> {
  "changed": true,
  "jobs": [
    "test cron"
  ]
}
10.1.32.73 | success >> {
  "changed": true,
  "jobs": [
    "test cron"
  ]
}

[root@node72 ~]# ansible web -m shell -a 'crontab -l'
10.1.32.68 | success | rc=0 >>
#Ansible: test cron
*/10 9-17 20-30/2 * * ntpdate 10.1.0.1 &> /dev/null

10.1.32.73 | success | rc=0 >>
#Ansible: test cron
*/10 9-17 20-30/2 * * ntpdate 10.1.0.1 &> /dev/null

[root@node72 ~]#
```

```
[root@node72 ~]# ansible web -m cron -a 'name="test cron" state=absent'
```

10.1.32.68 | success >> {  
 "changed": true,  
 "jobs": []  
}

10.1.32.73 | success >> {  
 "changed": true,  
 "jobs": []  
}

```
[root@node72 ~]# ansible web -m shell -a 'crontab -l'
```

10.1.32.68 | success | rc=0 >>  
  
10.1.32.73 | success | rc=0 >>

```
[root@node72 ~]#
```

删除计划任务，要指明要删除的计划任务的名称  
这也是为什么在利用**ansible**定义计划任务时要加上**name**的意义  
**state**为**absent**表示删除任务

## 6、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@node72 ~]# ansible 10.1.32.73 -m fetch -a "src=/tmp/test.txt dest=/testdir"
```

10.1.32.73 | success >> {  
 "changed": true,  
 "checksum": "1ea7fa2df3126c29354ed0f72926dc71ed95fa84",  
 "dest": "/testdir/10.1.32.73/tmp/test.txt",  
 "md5sum": "dc177bf7935393915a69d2c53384a7d4",  
 "remote\_checksum": "1ea7fa2df3126c29354ed0f72926dc71ed95fa84",  
 "remote\_md5sum": null  
}

```
[root@node72 ~]# cat /testdir/10.1.32.73/tmp/test.txt
```

I am NWC  
[root@node72 ~]#  
[root@node72 ~]#

## 7、file模块

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=directory|link|absent 表示创建的文件是目录还是软链接

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

删除文件：

path= state=absent

```
[root@node72 ~]# ansible web -m file -a 'src=/tmp/test.txt path=/usr/a.link state=link'
```

10.1.32.73 | success >> {  
 "changed": true,  
 "dest": "/usr/a.link",  
 "gid": 0,  
 "group": "root",  
 "mode": "0777",  
 "owner": "root",  
 "size": 13,  
 "src": "/tmp/test.txt",  
 "state": "link",  
 "uid": 0  
}

10.1.32.68 | success >> {  
 "changed": true,  
 "dest": "/usr/a.link",  
 "gid": 0,  
 "group": "root",

为远程主机上指定文件或目录创建软链接的用法

```
[root@node72 ~]# ansible web -m shell -a 'ls -l /usr/a.link'
```

10.1.32.68 | success | rc=0 >>  
lrwxrwxrwx 1 root root 13 Jul 13 08:51 /usr/a.link -> /tmp/test.txt

10.1.32.73 | success | rc=0 >>  
lrwxrwxrwx 1 root root 13 Nov 2 12:21 /usr/a.link -> /tmp/test.txt

```
[root@node72 ~]#
```

```
[root@node72 ~]# ansible web -m file -a 'path=/tmp/test.txt owner=postfix group=sshd mode=7000'
```

```
10.1.32.73 | success >> {
  "changed": true,
  "gid": 74,
  "group": "sshd",
  "mode": "07000",
  "owner": "postfix",
  "path": "/tmp/test.txt",
  "size": 8,
  "state": "file",
  "uid": 89
}
```

修改远程主机上的对应的文件的属性的示例

```
10.1.32.68 | success >> {
  "changed": true,
  "gid": 74,
  "group": "sshd",
  "mode": "07000",
  "owner": "postfix",
  "path": "/tmp/test.txt",
  "size": 8,
  "state": "file",
  "uid": 89
}
```

```
[root@node72 ~]#
```

```
[root@node72 ~]#
```

```
[root@node72 ~]# ansible web -m shell -a 'ls -l /tmp/test.txt'
```

```
10.1.32.73 | success | rc=0 >>
```

```
---S--S--T 1 postfix sshd 8 Nov  2 11:55 /tmp/test.txt
```

```
10.1.32.68 | success | rc=0 >>
```

```
---S--S--T 1 postfix sshd 8 Jul 13 08:24 /tmp/test.txt
```

```
[root@node72 ~]#
```

```
[root@node72 ~]# ansible all -m file -a 'path=/tmp/testdir state=directory'
```

```
10.1.32.68 | success >> {  
    "changed": true,          创建目录的示例  
    "gid": 0,  
    "group": "root",  
    "mode": "0755",  
    "owner": "root",  
    "path": "/tmp/testdir",  
    "size": 4096,  
    "state": "directory",  
    "uid": 0  
}
```

```
10.1.32.73 | success >> {  
    "changed": true,  
    "gid": 0,  
    "group": "root",  
    "mode": "0755",  
    "owner": "root",  
    "path": "/tmp/testdir",  
    "size": 6,  
    "state": "directory",  
    "uid": 0  
}
```

```
[root@node72 ~]# ansible web -m shell -a 'ls -l /tmp'
```

```
10.1.32.68 | success | rc=0 >>  
total 12  
-rw-r--r--  1 root    root    6 Jul 13 08:14 a.txt  
---S--S--T  1 postfix sshd    8 Jul 13 08:24 test.txt  
drwxr-xr-x  2 root    root  4096 Jul 13 08:59 testdir  
-rw-----  1 root    root    0 Jul 13 00:16 yum.log
```

```
10.1.32.73 | success | rc=0 >>  
total 8  
-rw-r--r--  1 root    root  6 Nov  2 11:44 a.txt  
---S--S--T  1 postfix sshd  8 Nov  2 11:55 test.txt  
drwxr-xr-x  2 root    root  6 Nov  2 12:30 testdir
```

```
[root@node72 ~]#
```

```
[root@node72 ~]# ansible all -m file -a 'path=/tmp/testdir state=absent'
```

```
10.1.32.68 | success >> {  
    "changed": true,  
    "path": "/tmp/testdir",  
    "state": "absent"  
}
```

删除目录

```
10.1.32.73 | success >> {  
    "changed": true,  
    "path": "/tmp/testdir",  
    "state": "absent"  
}
```

```
[root@node72 ~]# ansible all -m file -a 'path=/usr/a.link state=absent'
```

```
10.1.32.73 | success >> {  
    "changed": true,  
    "path": "/usr/a.link",  
    "state": "absent"  
}
```

删除文件

```
10.1.32.68 | success >> {  
    "changed": true,  
    "path": "/usr/a.link",  
    "state": "absent"  
}
```

```
[root@node72 ~]#
```

```
[root@node72 ~]# ansible web -m shell -a 'ls -l /tmp'
```

```
10.1.32.68 | success | rc=0 >>  
total 8  
-rw-r--r-- 1 root    root 6 Jul 13 08:14 a.txt  
---S---S--T 1 postfix sshd 8 Jul 13 08:24 test.txt  
-rw----- 1 root    root 0 Jul 13 00:16 yum.log
```

```
10.1.32.73 | success | rc=0 >>  
total 8  
-rw-r--r-- 1 root    root 6 Nov  2 11:44 a.txt  
---S---S--T 1 postfix sshd 8 Nov  2 11:55 test.txt
```

```
[root@node72 ~]# ansible web -m shell -a 'ls -l /usr/a.link'
```

```
10.1.32.68 | FAILED | rc=2 >>  
ls: cannot access /usr/a.link: No such file or directory
```

```
10.1.32.73 | FAILED | rc=2 >>  
ls: cannot access /usr/a.link: No such file or directory
```

```
[root@node72 ~]#
```

## 8、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

```
[root@node72 ~]# ansible web -m hostname -a 'name=ansible.test'
10.1.32.68 | success >> {
  "changed": true,
  "name": "ansible.test"
}
10.1.32.73 | success >> {
  "changed": true,
  "name": "ansible.test"
}

[root@node72 ~]#
[root@node72 ~]# ansible web -m shell -a 'hostname'
10.1.32.68 | success | rc=0 >>
ansible.test
10.1.32.73 | success | rc=0 >>
ansible.test

[root@node72 ~]#
```

## 9、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

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

```
[root@node72 ~]# ansible all -m yum -a 'name=httpd state=present'
```

```
10.1.32.73 | success >> {
    "changed": true,
    "msg": "",
    "rc": 0,
    "results": [
        "Loaded plugins: fastestmirror\nLoading mirror speeds from cached hostfile\n--> Running transaction check\n--> Package httpd.x86_64 0:2.4.6-40.el7.centos will be installed\n--> Processing Dependency: httpd-tools = 2.4.6-40.el7.centos for package: httpd-2.4.6-40.el7.centos.x86_64\n--> Processing Dependency: /etc/mime.types for package: httpd-2.4.6-40.el7.centos.x86_64\n--> Processing Dependency: libaprutil-1.so.0()(64bit) for package: httpd-2.4.6-40.el7.centos.x86_64\n--> Processing Dependency: libapr-1.so.0()(64bit) for package: httpd-2.4.6-40.el7.centos.x86_64\n--> Running transaction check\n--> Package httpd.x86_64 0:2.4.6-40.el7.centos will be installed\n--> Package apr-util.x86_64 0:1.5.2-6.el7.centos will be installed\n--> Package httpd-tools.x86_64 0:2.4.6-40.el7.centos will be installed\n--> Package apr.x86_64 0:1.5.2-6.el7.centos will be installed\n--> Package httpd.x86_64 0:2.4.6-40.el7.centos will be installed\n--> Finished Dependency Resolution\n\nDependencies Resolved\n\n=====
```

安装软件包

```
[root@node72 ~]# ansible all -m shell -a 'rpm -q httpd'
```

```
10.1.32.68 | success | rc=0 >>
httpd-2.2.15-53.el6.centos.x86_64

10.1.32.73 | success | rc=0 >>
httpd-2.4.6-40.el7.centos.x86_64
```

```
[root@node72 ~]#
```

```
[root@node72 ~]# ansible all -m yum -a 'name=httpd state=absent'
```

```
10.1.32.73 | success >> {
    "changed": true,
    "msg": "",
    "rc": 0,
    "results": [
        "Loaded plugins: fastestmirror\nResolving Dependencies\n--> Running transaction check\n--> Package httpd.x86_64 0:2.4.6-40.el7.centos will be erased\n--> Finished Dependency Resolution\n\nDependencies Resolved\n\n=====
```

Package	Arch	Version	Release	Size
httpd	x86_64	2.4.6-40.el7.centos	@BASE	9.4 M

```
=====\nRemove 1 Package\n\nInstalled size: 9.4 M\nDownloading packages:\nRunning transaction check\nRunning transaction test\nTransaction test succeeded\nRunning transaction\n
```

卸载软件包

```
[root@node72 ~]# ansible all -m shell -a 'rpm -q httpd'
10.1.32.68 | FAILED | rc=1 >>
package httpd is not installed

10.1.32.73 | FAILED | rc=1 >>
package httpd is not installed

[root@node72 ~]#
```

```
[root@node72 ~]# ansible all -m yum -a 'name=httpd disable_gpg_check=yes disablerepo=EPEL state=present'
```

```
10.1.32.73 | success >> {
  "changed": true,
  "msg": "",
  "rc": 0,
  "results": [
    "Loaded plugins: fastestmirror\nLoading mirror speeds from cached hostfile\nResolving Dependencies\n--> Running transaction check\n--> Package httpd.x86_64 0:2.4.6-40.el7.centos will be installed\n--> Finished Dependency Resolution\n\nDependencies Resolved\n\n=====
=====
Package                               Arch
Version                               Repository      Size\n=====
Installing:\n httpd                      x86_64          2.4.6-40.el7.centos
BASE                                2.7 M\n\nTransaction Summary\n=====
Install 1 Package\n\nTotal download size: 2.7 M\nInstalled size: 9.4 M\nDownloading packages:\nRunning transaction check\nRunning transaction test\nTransaction test succeeded\nRunning transaction\n  Installing : httpd-2.4.6-40.el7.centos.x86_64
1/1 \n Verifying : httpd-2.4.6-40.el7.centos
```

## 10、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

```
[root@node72 ~]# ansible all -m service -a 'name=httpd state=started'
```

```
10.1.32.68 | success >> {  
  "changed": true,  
  "name": "httpd",  
  "state": "started"  
}
```

```
10.1.32.73 | success >> {  
  "changed": true,  
  "name": "httpd",  
  "state": "started"  
}
```

```
[root@node72 ~]# ansible all -m shell -a 'ss -tnl|grep ":80"'
```

```
10.1.32.73 | success | rc=0 >>  
LISTEN      0          128          :::80          :::*
```

```
10.1.32.68 | success | rc=0 >>  
LISTEN      0          128          :::80          :::*
```

```
[root@node72 ~]#
```

```
[root@node72 ~]# ansible all -m service -a 'name=httpd state=restarted enabled=yes runlevel=23 45'
```

```
10.1.32.68 | success >> {  
  "changed": true,  
  "enabled": true,  
  "name": "httpd",  
  "state": "started"  
}
```

```
10.1.32.73 | success >> {  
  "changed": true,  
  "enabled": true,  
  "name": "httpd",  
  "state": "started"  
}
```

```
[root@node72 ~]# ansible all -m shell -a 'chkconfig --list|grep "httpd"'
```

```
10.1.32.68 | success | rc=0 >>  
httpd      0:off  1:off  2:on   3:on   4:on   5:on   6:off
```

10.1.32.73 | FAILED | rc=1 >> 由于10.1.32.73节点是CentOS7系统，故显示出来此失败的内容

Note: This output shows SysV services only and does not include native systemd services. SysV configuration data might be overridden by native systemd configuration.

If you want to list systemd services use 'systemctl list-unit-files'.  
To see services enabled on particular target use  
'systemctl list-dependencies [target]'.

## 11、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

## 12、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，是否要删除用户的家目录

```
[root@node72 ~]# ansible web -m user -a 'name=nwc system=yes uid=1600 group=root groups=sshd shell=/sbin/nologin home=/tmp/test password=111111 comment="test user"'
```

```
10.1.32.73 | success >> {
  "changed": true,
  "comment": "test user",
  "createhome": true,
  "group": 0,
  "groups": "sshd",
  "home": "/tmp/test",
  "name": "nwc",
  "password": "NOT_LOGGING_PASSWORD",
  "shell": "/sbin/nologin",
  "state": "present",
  "system": true,
  "uid": 1600
}

10.1.32.68 | success >> {
  "changed": true,
  "comment": "test user",
  "createhome": true,
  "group": 0
```

添加用户示例，除了name必须给明，剩余选项均可省略

```
[root@node72 ~]# ansible all -m shell -a 'tail -1 /etc/passwd'
```

```
10.1.32.68 | success | rc=0 >>
nwc:x:1600:0:test user:/tmp/test:/sbin/nologin

10.1.32.73 | success | rc=0 >>
nwc:x:1600:0:test user:/tmp/test:/sbin/nologin
```

```
[root@node72 ~]#
```

```
[root@node72 ~]# ansible all -m shell -a 'tail -1 /etc/shadow'
```

```
10.1.32.68 | success | rc=0 >>
nwc:111111:16995::::::

10.1.32.73 | success | rc=0 >>
nwc:111111:17107::::::
```

在用ansible给明创建的用户密码时，建议用加密后的字符串指定，否则就是明文存储到用户的密码文件中

```
[root@node72 ~]# ansible all -m shell -a 'id nwc'
```

```
10.1.32.73 | success | rc=0 >>
uid=1600(nwc) gid=0(root) groups=0(root),74(sshd)

10.1.32.68 | success | rc=0 >>
uid=1600(nwc) gid=0(root) groups=0(root),74(sshd)
```

```
[root@node72 ~]#
```

```

[root@node72 ~]# ansible all -m user -a 'name=nwc remove=yes state=absent'
10.1.32.73 | success >> {
    "changed": true,
    "force": false,
    "name": "nwc",
    "remove": true,
    "state": "absent",
    "stderr": "userdel: nwc mail spool (/var/spool/mail/nwc) not found\n"
}

10.1.32.68 | success >> {
    "changed": true,
    "force": false,
    "name": "nwc",
    "remove": true,
    "state": "absent",
    "stderr": "userdel: nwc mail spool (/var/spool/mail/nwc) not found\n"
}

[root@node72 ~]# ansible all -m shell -a 'id nwc'
10.1.32.73 | FAILED | rc=1 >>
id: nwc: no such user

10.1.32.68 | FAILED | rc=1 >>
id: nwc: No such user

[root@node72 ~]#

```

## 13、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

```

[root@node72 ~]# ansible web -m group -a 'name=testgroup gid=1800 system=yes'
10.1.32.68 | success >> {
    "changed": true,
    "gid": 1800,
    "name": "testgroup",
    "state": "present",
    "system": true
}

10.1.32.73 | success >> {
    "changed": true,
    "gid": 1800,
    "name": "testgroup",
    "state": "present",
    "system": true
}

[root@node72 ~]# ansible web -m shell -a 'tail -1 /etc/group'
10.1.32.73 | success | rc=0 >>
testgroup:x:1800:

10.1.32.68 | success | rc=0 >>
testgroup:x:1800:

[root@node72 ~]# █

```

```

[root@node72 ~]# ansible web -m group -a 'name=testgroup state=absent'
10.1.32.73 | success >> {
    "changed": true,
    "name": "testgroup",
    "state": "absent"
}

10.1.32.68 | success >> {
    "changed": true,
    "name": "testgroup",
    "state": "absent"
}

[root@node72 ~]# ansible web -m shell -a 'tail -5 /etc/group'
10.1.32.68 | success | rc=0 >>
stapsys:x:157:
stapdev:x:158:
tcpdump:x:72:
oprofile:x:16:
apache:x:48:

10.1.32.73 | success | rc=0 >>
tss:x:59:
postdrop:x:90:
postfix:x:89:

```

删除组



## 14、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)，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

```
[root@node72 ~]# cat /root/test.sh
#!/bin/bash
#
# 在ansible管理端上创建一个脚本文件，假设脚本为此内容
if [[ $# -eq 1 ]] ;then
case $1 in
start)
touch /tmp/script.test && echo "start success"
;;
stop)
[[ -f /tmp/script.test ]] && rm -f /tmp/script.test && echo "stop success" || echo "Already Stopped"
;;
*)
echo "Wrong Argus"
exit
;;
esac
else
echo "Usage: $0 start|stop "
exit
fi
fi
[root@node72 ~]#
```

```
[root@node72 ~]#
[root@node72 ~]# ansible web -m script -a "/root/test.sh start"
10.1.32.68 | success >> {
  "changed": true,
  "rc": 0,
  "stderr": "OpenSSH_6.6.1, OpenSSL 1.0.1e-fips 11 Feb 2013\r\ndebug1: Reading config data /etc/ssh/ssh_config\r\ndebug1: /etc/ssh/ssh_config line 56: Applying options: bug1: auto-mux: Trying existing master\r\ndebug1: mux_client_request_session: master: 2\r\nShared connection to 10.1.32.68 closed.\r\n",
  "stdout": "start success\r\n"
}
10.1.32.73 | success >> {
  "changed": true,
  "rc": 0,
  "stderr": "OpenSSH_6.6.1, OpenSSL 1.0.1e-fips 11 Feb 2013\r\ndebug1: Reading config data /etc/ssh/ssh_config\r\ndebug1: /etc/ssh/ssh_config line 56: Applying options: bug1: auto-mux: Trying existing master\r\ndebug1: mux_client_request_session: master: 2\r\nShared connection to 10.1.32.73 closed.\r\n",
  "stdout": "start success\r\n"
}
```

```
[root@node72 ~]# ansible web -m script -a "/root/test.sh"
10.1.32.68 | success >> {
  "changed": true,
  "rc": 0,
  "stderr": "OpenSSH_6.6.1, OpenSSL 1.0.1e-fips 11 Feb 2013\r\ndebug1: Reading configuration data /etc/ssh/ssh_config\r\ndebug1: /etc/ssh/ssh_config line 56: Applying options for *\r\ndebug1: auto-mux: Trying existing master\r\ndebug1: mux_client_request_session: master session id: 2\r\n\r\nShared connection to 10.1.32.68 closed.\r\n",
  "stdout": "Usage: /root/.ansible/tmp/ansible-tmp-1470482009.18-80273906108620/test.sh start|stop \r\n"
}
10.1.32.73 | success >> {
  "changed": true,
  "rc": 0,
  "stderr": "OpenSSH_6.6.1, OpenSSL 1.0.1e-fips 11 Feb 2013\r\ndebug1: Reading configuration data /etc/ssh/ssh_config\r\ndebug1: /etc/ssh/ssh_config line 56: Applying options for *\r\ndebug1: auto-mux: Trying existing master\r\ndebug1: mux_client_request_session: master session id: 2\r\n\r\nShared connection to 10.1.32.73 closed.\r\n",
  "stdout": "Usage: /root/.ansible/tmp/ansible-tmp-1470482009.2-207863272487718/test.sh start|stop \r\n"
}
[root@node72 ~]#
```

## 15、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，之后我们需要用到时，直接调用变量即可

```
[root@node72 ~]# ansible all -m setup 收集指定主机上的facts变量的信息
10.1.32.73 | success >> {
  "ansible_facts": {
    "ansible_all_ipv4_addresses": [
      "10.1.32.73"
    ],
    "ansible_all_ipv6_addresses": [
      "fe80::20c:29ff:fec6:b77c"
    ],
    "ansible_architecture": "x86_64",
    "ansible_bios_date": "07/02/2015",
    "ansible_bios_version": "6.00",
    "ansible_cmdline": {
      "BOOT_IMAGE": "/vmlinuz-3.10.0-327.el7.x86_64",
      "LANG": "en_US.UTF-8",
      "quiet": true,
      "rhgb": true,
      "ro": true,
      "root": "UUID=3b08a39e-274b-4052-a602-814fbebba2c"
    },
    "ansible_date_time": {
      "date": "2016-11-02",
      "day": "02",
      "epoch": "1478064916",
      "hour": "13",
      "iso8601": "2016-11-02T05:35:16Z"
    }
  }
}
```

如本处，ansible\_all\_ipv4\_addresses为变量名，其值为10.1.32.73

## 16、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

```
[root@node72 ~]# cat muban.test
This is {{ ansible_all_ipv4_addresses }}
[root@node72 ~]#
[root@node72 ~]# cat test.yml
- hosts: web
  remote_user: root
  tasks:
    - name: test template module
      template: src=/root/muban.test dest=/tmp
[root@node72 ~]#
```

在管理端节点上定义模板文件，  
`ansible_all_ipv4_addresses`为一个facts变量，保存的是被管理节点上的IP地址

创建playbook文件，在文件中利用template模块，将模板文件拷贝到被管理节点上，被管理节点根据自身的实际，将模板中的变量替换为自身的值，从而实现了基于模板的复制功能

```
[root@node72 ~]# ansible-playbook test.yml
```

执行模板文件

```
PLAY [web] *****

GATHERING FACTS *****
ok: [10.1.32.73]
ok: [10.1.32.68]

TASK: [test template module] *****
changed: [10.1.32.68]
changed: [10.1.32.73]

PLAY RECAP *****
10.1.32.68      : ok=2    changed=1    unreachable=0    failed=0
10.1.32.73      : ok=2    changed=1    unreachable=0    failed=0
```

```
[root@node72 ~]# ansible web -m shell -a 'cat /tmp/muban.test'
10.1.32.68 | success | rc=0 >
This is [u'10.1.32.68']
10.1.32.73 | success | rc=0 >
This is [u'10.1.32.73']
```

发现对应的值被替换为了被管理客户端的上的值

```
[root@node72 ~]#
```

## 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

```
[root@node72 ~]# vim test.yml
- hosts: web
  remote_user: root
  tasks:
    - name: add a user
      user: name=nwc state=present
    - name: add a group
      group: name=testgroup system=yes
- hosts: web
  remote_user: root
  tasks:
    - name: copy a file
      copy: src=/tmp/test.txt dest=/etc/test
```

指明运行该playbook剧本的被管理主机有哪些，一个playbook文件可有多多个hosts

指明运行该剧本的远程主机上的用户是谁

任务列表

指明某个具体任务的名称

具体的任务是什么，也就是调用哪个模块，传递给该模块的参数是什么，从而实现了某个具体的功能

注意：每个参数的冒号后面都有空格

## 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.yml`进行测试, 测试没问题后

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

```
[root@node72 ~]#
[root@node72 ~]# ansible-playbook --check test.yml 测试执行playbook

PLAY [web] *****
                表示该剧本在哪些主机上执行

GATHERING FACTS *****
ok: [10.1.32.73]
ok: [10.1.32.68]
                默认每个剧本执行时都会执行该任务, 该任务是用来收集被管理主机上的各个facts变量的信息

TASK: [add a user] *****
changed: [10.1.32.68]
changed: [10.1.32.73]
                第一个任务, changed表示执行该任务时, 会改变远程主机现有的状态, OK表示执行该任务, 没有改变远程主机的现有状态

TASK: [add a group] *****
changed: [10.1.32.68]
changed: [10.1.32.73]
                执行该剧本的统计信息, ok表示可成功执行的任务的个数(加上了默认执行的收集facts变量的任务), changed表示执行该剧本, 会改变远程主机的任务的个数, unreachable表示任务不可达的个数, failed表示执行失败的任务的个数

PLAY RECAP *****
10.1.32.68      : ok=3    changed=2    unreachable=0    failed=0
10.1.32.73      : ok=3    changed=2    unreachable=0    failed=0

[root@node72 ~]#
```

```
[root@node72 ~]# ansible-playbook test.yml 真正执行剧本

PLAY [web] *****

GATHERING FACTS *****
ok: [10.1.32.73]      注意：执行剧本不会有任务的输出信息，故不建议将标准输出的任务定义
ok: [10.1.32.68]      在剧本文件中

TASK: [add a user] *****
changed: [10.1.32.73]
changed: [10.1.32.68]

TASK: [add a group] *****
changed: [10.1.32.73]
changed: [10.1.32.68]

PLAY RECAP *****
10.1.32.68      : ok=3    changed=2    unreachable=0    failed=0
10.1.32.73      : ok=3    changed=2    unreachable=0    failed=0

[root@node72 ~]#
```

## 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：

```
[root@node72 ~]# vim test.yml

- hosts: web
  remote_user: root
  tasks:
    - name: copy a config file
      copy: src=/tmp/httpd.conf dest=/etc/httpd/conf/httpd.conf
      notify:
        - restart httpd service
        - mail to root
  handlers:
    - name: restart httpd service
      service: name=httpd state=restarted
    - name: mail to root
      shell: echo "hello" | mail -s "test mail" root@localhost

~
```

表示定义一个任务，复制/tmp/httpd.conf文件到被管理主机上的/etc/httpd/conf/httpd.conf，复制完成后，触发两个触发器，一个触发器的名称是restart httpd service，另一个触发器的名称是mail to root

此处handler的名称要与tasks中notify定义的名称一致

```
[root@node72 ~]# ansible-playbook test.yml

PLAY [web] *****

GATHERING FACTS *****
ok: [10.1.32.73]
ok: [10.1.32.68]

TASK: [copy a config file] *****
changed: [10.1.32.68]
changed: [10.1.32.73]

NOTIFIED: [restart httpd service] *****
changed: [10.1.32.68]
changed: [10.1.32.73]

NOTIFIED: [mail to root] *****
changed: [10.1.32.68]
changed: [10.1.32.73]

PLAY RECAP *****
10.1.32.68      : ok=4    changed=3    unreachable=0    failed=0
10.1.32.73      : ok=4    changed=3    unreachable=0    failed=0

[root@node72 ~]#
```

执行copy任务后，会触发执行两个handler任务

在被管理主机上查看，触发执行的任务是否正确执行

```
[root@node73 ~]# mail
Heirloom Mail version 12.5 7/5/10.  Type ? for help.
"/var/spool/mail/root": 1 message 1 unread
>U 1 root      Wed Nov  2 15:50  19/562  "test mail"
& 1
Message 1:
From root@ansible.test  Wed Nov  2 15:50:44 2016
Return-Path: <root@ansible.test>
X-Original-To: root@localhost
Delivered-To: root@localhost.test
Date: Wed, 02 Nov 2016 15:50:44 +0800
To: root@localhost.test
Subject: test mail
User-Agent: Heirloom mailx 12.5 7/5/10
Content-Type: text/plain; charset=us-ascii
From: root@ansible.test (root)
Status: RO

hello

&
```

## 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\_NAME 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

```
[root@node72 ~]# vim test.yml
```

```
- hosts: web
  remote_user: root
  tasks:
    - name: copy a config file
      copy: src=/tmp/httpd.conf dest=/tmp/test.conf
      tags:
        - testtag1
        - testtag2
    - name: restart httpd service
      service: name=httpd state=restarted
      tags:
        - testtag2
        - testtag3
    - name: mail to root
      shell: echo "hello" | mail -s "test mail" root@localhost
      tags:
        - testtag5
```

playbook文件本处，第一个任务的标签为testtag1，testtag2  
第二个任务的标签为testtag2，testtag3  
第三个任务的标签为testtag5  
标签的名称是自己定义的，一个任务可以有多个标签，也可以没有标签，一个标签可以对应多个任务

```
[root@node72 ~]#
[root@node72 ~]# ansible-playbook -t testtag5 test.yml
```

指明只执行那个tags的任务

```
PLAY [web] *****

GATHERING FACTS *****
ok: [10.1.32.73]
ok: [10.1.32.68]

TASK: [mail to root] *****
changed: [10.1.32.68]
changed: [10.1.32.73]

PLAY RECAP *****
10.1.32.68      : ok=2    changed=1    unreachable=0    failed=0
10.1.32.73      : ok=2    changed=1    unreachable=0    failed=0

[root@node72 ~]#
```

```
[root@node72 ~]# ansible-playbook -t testtag2 test.yml

PLAY [web] *****
    由于指定要执行的tag标签对应了两个任务，因此只要是该标签的任务都执行
GATHERING FACTS *****
ok: [10.1.32.73]
ok: [10.1.32.68]

TASK: [copy a config file] *****
changed: [10.1.32.68]
changed: [10.1.32.73]

TASK: [restart httpd service] *****
changed: [10.1.32.68]
changed: [10.1.32.73]

PLAY RECAP *****
10.1.32.68      : ok=3    changed=2    unreachable=0    failed=0
10.1.32.73      : ok=3    changed=2    unreachable=0    failed=0

[root@node72 ~]#
```

```
[root@node72 ~]# ansible-playbook -t testtag3,testtag5 test.yml

PLAY [web] *****
    支持同时执行逗号分隔的多个tag标签的任务
GATHERING FACTS *****
ok: [10.1.32.73]
ok: [10.1.32.68]

TASK: [restart httpd service] *****
changed: [10.1.32.68]
changed: [10.1.32.73]

TASK: [mail to root] *****
changed: [10.1.32.68]
changed: [10.1.32.73]

PLAY RECAP *****
10.1.32.68      : ok=3    changed=2    unreachable=0    failed=0
10.1.32.73      : ok=3    changed=2    unreachable=0    failed=0

[root@node72 ~]#
```

## 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：

可列出10.1.32.72主机上可用的所有的facts变量及其值

获取的facts类型的变量名，及其对应的值

此为两个facts变量，用于保存被管理节点的发行版和版本号

```
[root@node72 ~]# ansible web -m shell -a 'cat /tmp/test.file'
```

```
10.1.32.68 | success | rc=0 >>
CentOS 6.8

10.1.32.73 | success | rc=0 >>
CentOS 7.2.1511
```

在剧本执行后，相关变量被替换为了被管理节点上对应的信息

```
[root@node72 ~]#
```

## 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

```
[root@node72 ~]# cat test.yml
```

```
- hosts: web
  remote_user: root
  tasks:
    - name: test variables
      shell: echo "{{ var1 }}" "{{ var2 }}" > /tmp/test.file
```

playbook文件中调用变量

```
[root@node72 ~]#
[root@node72 ~]# ansible-playbook -e var1=nihao -e var2=nwc test.yml
```

```
PLAY [web] *****
```

直接在ansible-playbook命令行给出相应的变量的定义

```
GATHERING FACTS *****
```

```
ok: [10.1.32.73]
```

```
ok: [10.1.32.68]
```

```
TASK: [test variables] *****
```

```
changed: [10.1.32.68]
```

```
changed: [10.1.32.73]
```

```
PLAY RECAP *****
```

```
10.1.32.68 : ok=2 changed=1 unreachable=0 failed=0
```

```
10.1.32.73 : ok=2 changed=1 unreachable=0 failed=0
```

```
[root@node72 ~]# ansible web -m shell -a 'cat /tmp/test.file'
```

```
10.1.32.68 | success | rc=0 >>
nihao nwc
```

```
10.1.32.73 | success | rc=0 >>
nihao nwc
```

```
[root@node72 ~]#
```

## 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

```
[root@node72 ~]# vim /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
```

```
[web]
10.1.32.68 var1=node68 var2=nihao
10.1.32.73 var1=node72 var2=hello
```

在ansible的host文件中针对不同的主机可定义不同的变量，同一个变量针对不同的主机的值可以不同

```
[root@node72 ~]# cat test.yml
```

```
- hosts: web
  remote_user: root
  tasks:
    - name: test variables
      shell: echo "{{ var1 }}" "{{ var2 }}" > /tmp/test.file
```

在playbook中引用变量

```
[root@node72 ~]#
```

```
[root@node72 ~]# ansible-playbook test.yml 执行剧本
```

```
PLAY [web] *****
```

```
GATHERING FACTS *****
```

```
ok: [10.1.32.73]
```

```
ok: [10.1.32.68]
```

```
TASK: [test variables] *****
```

```
changed: [10.1.32.68]
```

```
changed: [10.1.32.73]
```

```
PLAY RECAP *****
```

```
10.1.32.68 : ok=2 changed=1 unreachable=0 failed=0
```

```
10.1.32.73 : ok=2 changed=1 unreachable=0 failed=0
```

```
[root@node72 ~]# ansible web -m shell -a 'cat /tmp/test.file'
```

```
10.1.32.73 | success | rc=0 >>  
node72 hello
```

针对每个主机的变量实现了不同的引用

```
10.1.32.68 | success | rc=0 >>  
node68 nihao
```

```
[root@node72 ~]#
```

<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

```
[root@node72 ~]# vim /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
```

```
[web]
```

```
10.1.32.68 var1=node68 var2=nihao  
10.1.32.73 var1=node72 var2=hello
```

```
[web:vars]
```

```
var3=world
```

对某个组的主机，定义组内的通用变量

```

[root@node72 ~]# cat test.yml
- hosts: web
  remote_user: root
  tasks:
    - name: test variables 在playbook文件中引用变量
      shell: echo "{{ var1 }}" "{{ var2 }}" "{{ var3 }}" > /tmp/test.file
[root@node72 ~]#
[root@node72 ~]# ansible-playbook test.yml
执行脚本
PLAY [web] *****

GATHERING FACTS *****
ok: [10.1.32.73]
ok: [10.1.32.68]

TASK: [test variables] *****
changed: [10.1.32.68]
changed: [10.1.32.73]

PLAY RECAP *****
10.1.32.68      : ok=2    changed=1    unreachable=0    failed=0
10.1.32.73      : ok=2    changed=1    unreachable=0    failed=0

[root@node72 ~]# █

```

```

[root@node72 ~]# ansible web -m shell -a 'cat /tmp/test.file'
10.1.32.68 | success | rc=0 >>
node68 nihao world
10.1.32.73 | success | rc=0 >>
node72 hello world

[root@node72 ~]# █

```

实现了主机组的变量的引用

<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密钥登录，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
```

```
[root@node72 ~]# cat test.yml
```

```
- hosts: web
  remote_user: root
  vars:
    - a: 111
    - b: 222
    - c: 333
  tasks:
```

直接在playbook文件中定义变量, 注意变量定义的格式为VAR: VALUE, 而不是与其他地方定义变量一样用VAR=VALUE

```
  - name: test variables
    shell: echo "{{ a }}" "{{ b }}" "{{ c }}" > /tmp/test.file
```

引用变量

```
[root@node72 ~]#
```

```
[root@node72 ~]# ansible-playbook test.yml
```

执行剧本

```
PLAY [web] *****
```

```
GATHERING FACTS *****
```

```
ok: [10.1.32.73]
```

```
ok: [10.1.32.68]
```

```
TASK: [test variables] *****
```

```
changed: [10.1.32.73]
```

```
changed: [10.1.32.68]
```

```
PLAY RECAP *****
```

```
10.1.32.68      : ok=2    changed=1    unreachable=0    failed=0
```

```
10.1.32.73      : ok=2    changed=1    unreachable=0    failed=0
```

```
[root@node72 ~]#
```



```
[root@node72 ~]# ansible web -m shell -a 'cat /tmp/test.file'
10.1.32.68 | success | rc=0 >>
111 222 333

10.1.32.73 | success | rc=0 >>
111 222 333

[root@node72 ~]#
```

## 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

```
[root@node72 ~]# cat muban.test
My IP is {{ ansible_all_ipv4_addresses }}
My Mem is {{ ansible_memtotal_mb//2 }}MB
My Cpu count is {{ ansible_processor_vcpus+10 }}
[root@node72 ~]#
[root@node72 ~]# cat test.yml
- hosts: web
  remote_user: root
  tasks:
  - name: test templates
    template: src=/root/muban.test dest=/tmp
[root@node72 ~]#
[root@node72 ~]#
```

基于**jinja2**语言的风格定义模板文件

在**playbook**中利用**template**模块将模板文件复制到被管理主机，让远端主机根据自身的环境，生成符合自身环境的文件

```

[root@node72 ~]# ansible-playbook test.yml
PLAY [web] ***** 执行playbook剧本 *****

GATHERING FACTS *****
ok: [10.1.32.73]
ok: [10.1.32.68]

TASK: [test templates] *****
changed: [10.1.32.68]
changed: [10.1.32.73]

PLAY RECAP *****
10.1.32.68      : ok=2    changed=1    unreachable=0    failed=0
10.1.32.73      : ok=2    changed=1    unreachable=0    failed=0

[root@node72 ~]# ansible web -m shell -a 'cat /tmp/muban.test'
10.1.32.68 | success | rc=0 >>
My IP is [u'10.1.32.68']
My Mem is 363MB
My Cpu count is 12
10.1.32.73 | success | rc=0 >>
My IP is [u'10.1.32.73']
My Mem is 362MB
My Cpu count is 12

```

发现被管理主机上，根据自身的环境，生成了指定的文件

## 第九章 playbook中的条件判断机制的介绍

当我们在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：

```
[root@node72 ~]# cat test.yml
- hosts: web
  remote_user: root
  tasks:
    - name: install conf file to centos7
      template: src=/root/muban.c7.j2 dest=/etc/nginx/nginx.conf
      when: ansible_distribution == "CentOS" and ansible_distribution_major_version=="7"
    - name: install conf file to centos6
      template: src=/root/muban.c6.j2 dest=/etc/nginx/nginx.conf
      when: ansible_distribution_major_version=="6"
```

[root@node72 ~]#

[root@node72 ~]# 表示当ansible\_distribution变量的值为 "CentOS"并且ansible\_distribution\_major\_version变量的值为7时, 才执行 template: src=/root/muban.c7.j2 dest=/etc/nginx/nginx.conf这个任务

当ansible\_distribution\_major\_version变量的值为6时, 才执行 template:src=/root/muban.c6.j2 dest=/etc/nginx/nginx.conf这个任务

```
[root@node72 ~]# ansible-playbook test.yml

PLAY [web] *****

GATHERING FACTS *****
ok: [10.1.32.73]
ok: [10.1.32.68]

TASK: [install conf file to centos7] *****
skipping: [10.1.32.68] 由于10.1.32.68这台主机不满足此任务的条件, 所以跳过不执行
changed: [10.1.32.73]

TASK: [install conf file to centos6] *****
skipping: [10.1.32.73] 由于10.1.32.73这台主机不满足此任务的条件, 所以跳过不执行
changed: [10.1.32.68]

PLAY RECAP *****
10.1.32.68      : ok=2    changed=1    unreachable=0    failed=0
10.1.32.73      : ok=2    changed=1    unreachable=0    failed=0

[root@node72 ~]# ansible web -m shell -a 'cat /etc/nginx/nginx.conf'
10.1.32.73 | success | rc=0 >>
This is CentOS 7 test templates

10.1.32.68 | success | rc=0 >>
This is CentOS 6 test templates

[root@node72 ~]#
```

## 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

```
[root@node72 ~]# ls -l
```

```
anaconda-ks.cfg
```

```
file1
```

```
file2
```

```
file3
```

```
file4
```

```
file5
```

假设希望将此file1到file5复制到远端被管理主机上

```
test.retry
```

```
test.yml
```

```
[root@node72 ~]# cat test.yml
```

```
- hosts: web
```

```
  remote_user: root
```

```
  tasks:
```

```
    - name: xunhuan de shili
```

```
      copy: src=/root/file{{ item }} dest=/tmp
```

```
      with_items:
```

```
        - 1
```

```
        - 2
```

```
        - 3
```

```
        - 4
```

```
        - 5
```

此种模式下，只需要循环引用迭代项的值即可，{{ item }}将会被循环替换为1,2,3,4,5,从而实现了循环

```
[root@node72 ~]#
```

```
[root@node72 ~]# ansible-playbook test.yml
```

PLAY [web] \*\*\*\*\*

GATHERING FACTS \*\*\*\*\*

```
ok: [10.1.32.73]
ok: [10.1.32.68]
```

TASK: [xunhuan de shili] \*\*\*\*\*

```
changed: [10.1.32.73] => (item=1)
changed: [10.1.32.68] => (item=1)
changed: [10.1.32.73] => (item=2)
changed: [10.1.32.68] => (item=2)
changed: [10.1.32.73] => (item=3)
changed: [10.1.32.68] => (item=3)
changed: [10.1.32.68] => (item=4)
changed: [10.1.32.73] => (item=4)
changed: [10.1.32.68] => (item=5)
changed: [10.1.32.73] => (item=5)
```

PLAY RECAP \*\*\*\*\*

```
10.1.32.68      : ok=2    changed=1    unreachable=0    failed=0
10.1.32.73      : ok=2    changed=1    unreachable=0    failed=0
```

[root@node72 ~]#

相当于有5个步骤，第一个步骤是将file1拷贝到两台目标主机上，第二个步骤是将file2拷贝到两台目标主机上...

```
[root@node72 ~]# ansible web -m shell -a 'ls /tmp'
```

```
10.1.32.68 | success | rc=0 >>
file1
file2
file3
file4
file5
```

10.1.32.73 | success | rc=0 >>

```
file1
file2
file3
file4
file5
```

[root@node72 ~]#

已经成功拷贝

### 3、循环示例二：字典形式的迭代项的循环引用

```
[root@node72 ~]# ls -l
anaconda-ks.cfg
```

```
file1
file2
file3
file4
file5
```

假设，希望将file1到file5，分别复制到远程主机上/tmp/a到/tmp/e目录，也就是相当于将file1复制到/tmp/a目录，file2复制到/tmp/b目录...

```
test.retry
test.yml
```

```
[root@node72 ~]# cat test.yml
```

```
- hosts: web
  remote_user: root
  tasks:
```

```
  - name: mkdir some dir
```

```
    file: path=/tmp/{{ item }} state=directory
    with_items:
```

```
      - a
      - b
      - c
      - d
      - e
```

先用列表形式的迭代项循环创建目录

```
  - name: xunhuan de shili
```

```
    copy: src=/root/file{{ item.filename }} dest=/tmp/{{ item.dir }}
    with_items:
```

```
      - {filename: 1,dir: a}
      - {filename: 2,dir: b}
      - {filename: 3,dir: c}
      - {filename: 4,dir: d}
      - {filename: 5,dir: e}
```

再用字典形式的迭代项分别引用不同的迭代项，`{{ item.filename }}`表示引用with\_items中的第一个变量filename的值，以此类推

```
[root@node72 ~]#
```

```

[root@node72 ~]# ansible-playbook test.yml

PLAY [web] *****

GATHERING FACTS *****
ok: [10.1.32.73]
ok: [10.1.32.68]

TASK: [mkdir some dir] *****
changed: [10.1.32.68] => (item=a)
changed: [10.1.32.73] => (item=a)
changed: [10.1.32.68] => (item=b)
changed: [10.1.32.73] => (item=b) 先循环创建目录
changed: [10.1.32.68] => (item=c)
changed: [10.1.32.73] => (item=c)
changed: [10.1.32.68] => (item=d)
changed: [10.1.32.73] => (item=d)
changed: [10.1.32.68] => (item=e)
changed: [10.1.32.73] => (item=e)

TASK: [xunhuan de shili] *****
changed: [10.1.32.68] => (item={'dir': 'a', 'filename': 1})
changed: [10.1.32.73] => (item={'dir': 'a', 'filename': 1})
changed: [10.1.32.68] => (item={'dir': 'b', 'filename': 2})
changed: [10.1.32.73] => (item={'dir': 'b', 'filename': 2}) 再循环将文件拷贝到
changed: [10.1.32.68] => (item={'dir': 'c', 'filename': 3}) 对应的目录
changed: [10.1.32.73] => (item={'dir': 'c', 'filename': 3})
changed: [10.1.32.68] => (item={'dir': 'd', 'filename': 4})
changed: [10.1.32.73] => (item={'dir': 'd', 'filename': 4})
changed: [10.1.32.68] => (item={'dir': 'e', 'filename': 5})
changed: [10.1.32.73] => (item={'dir': 'e', 'filename': 5})

PLAY RECAP *****
10.1.32.68          : ok=3    changed=2    unreachable=0    failed=0
10.1.32.73          : ok=3    changed=2    unreachable=0    failed=0

[root@node72 ~]# █

```



```
[root@node72 ~]# ansible web -m shell -a 'tree /tmp'
10.1.32.68 | success | rc=0 >>
/tmp
|-- a
|   |-- file1
|-- b
|   |-- file2
|-- c
|   |-- file3
|-- d
|   |-- file4
|-- e
|   |-- file5

5 directories, 5 files

10.1.32.73 | success | rc=0 >>
/tmp
|-- a
|   |-- file1
|-- b
|   |-- file2
|-- c
|   |-- file3
|-- d
|   |-- file4
```

验证是否正确复制

## 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
```

即可

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 PLAYBOOK文件 执行

集合

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

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)

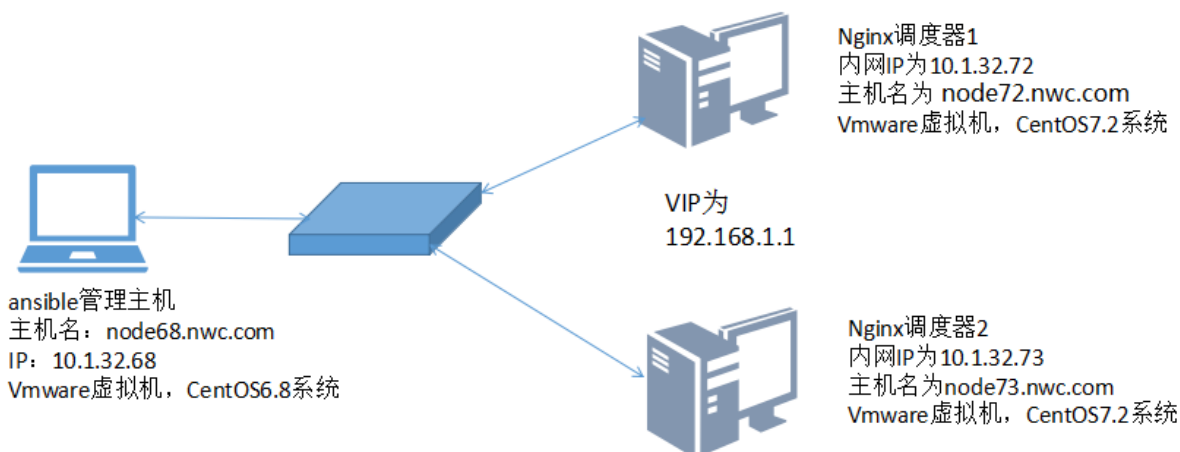
```
- hosts: webserver
  remote_user: root
  roles:
    - {role: mysql,var1: value1,when:ansible_distribution_major_version=='7'}
    - {role: httpd,when:ansible_distribution_major_version=='6'}
```

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

## 第十二章 ansible实战一：利用ansible配置主备模型的keepalived+nginx

### 1、实验环境

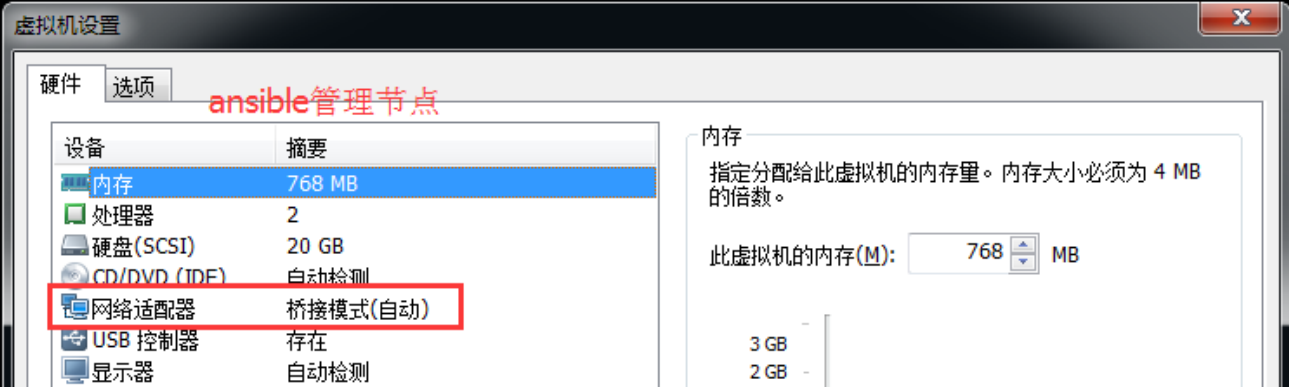


实验目的：利用ansible，实现部署nginx，并利用keepalived对nginx做主备模型高可用

## 2、实验前准备工作

<1> 配置好各个节点之间的网络环境

```
[root@node68 ~]#  
[root@node68 ~]# ip a  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN  
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  
    inet 127.0.0.1/8 scope host lo  
    inet6 ::1/128 scope host  
        valid_lft forever preferred_lft forever  
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000  
    link/ether 00:0c:29:aa:19:1b brd ff:ff:ff:ff:ff:ff  
    inet 10.1.32.68/16 brd 10.1.255.255 scope global eth0  
    inet6 fe80::20c:29ff:feaa:191b/64 scope link  
        valid_lft forever preferred_lft forever  
[root@node68 ~]#  
[root@node68 ~]# hostname  
node68.nwc.com  
[root@node68 ~]#
```



虚拟机设置

硬件 选项 **ansible管理节点**

设备	摘要
内存	768 MB
处理器	2
硬盘(SCSI)	20 GB
CD/DVD (IDE)	自动检测
网络适配器	桥接模式(自动)
USB 控制器	存在
显示器	自动检测

内存  
指定分配给此虚拟机的内存量。内存大小必须为 4 MB 的倍数。

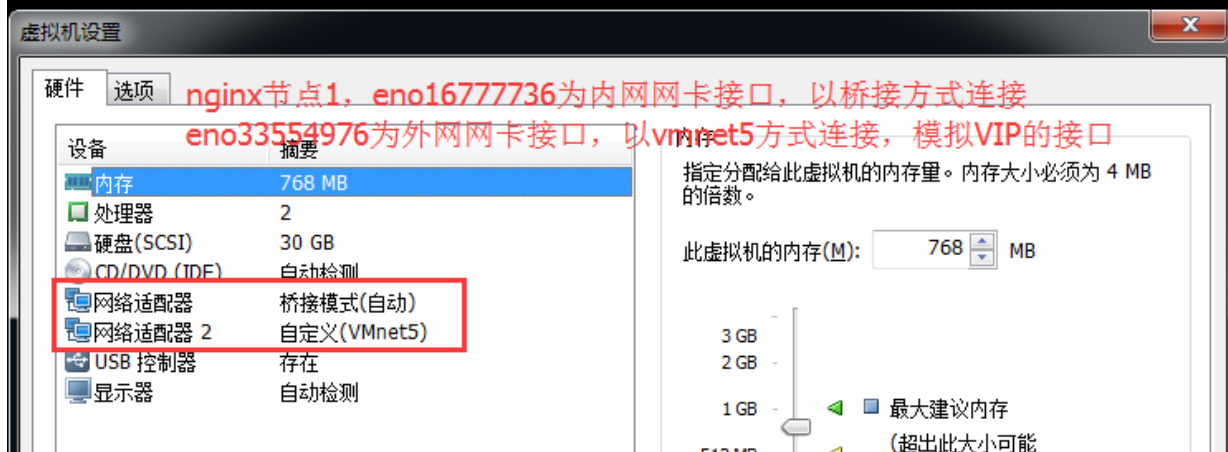
此虚拟机的内存(M): 768 MB

3 GB -  
2 GB -

```

[root@node72 ~]#
[root@node72 ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eno16777736: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:0c:29:ec:d4:0f brd ff:ff:ff:ff:ff:ff
    inet 10.1.32.72/16 brd 10.1.255.255 scope global eno16777736
        valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:feec:d40f/64 scope link
        valid_lft forever preferred_lft forever
3: eno33554976: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:0c:29:ec:d4:19 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::20c:29ff:feec:d419/64 scope link
        valid_lft forever preferred_lft forever
[root@node72 ~]# hostname
node72.nwc.com
[root@node72 ~]#

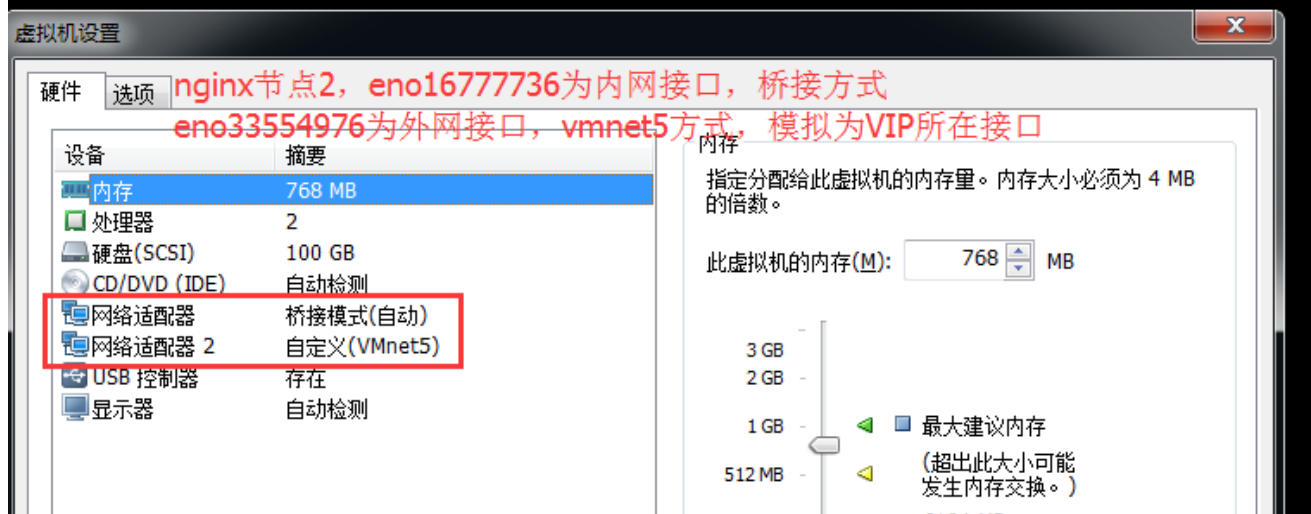
```



```

[root@node73 ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eno16777736: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:0c:29:c6:b7:7c brd ff:ff:ff:ff:ff:ff
    inet 10.1.32.73/16 brd 10.1.255.255 scope global eno16777736
        valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fec6:b77c/64 scope link
        valid_lft forever preferred_lft forever
3: eno33554976: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:0c:29:c6:b7:86 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::20c:29ff:fec6:b786/64 scope link
        valid_lft forever preferred_lft forever
[root@node73 ~]# hostname
node73.nwc.com
[root@node73 ~]#

```



<2> 各个节点之间时间同步

```
[root@node68 ~]# vim /etc/ntp.conf
```

```
# For more information about this file, see the man pages
# ntp.conf(5), ntp_acc(5), ntp_auth(5), ntp_clock(5), ntp_misc(5), ntp_mon(5).
```

```
driftfile /var/lib/ntp/drift
```

```
# Permit time synchronization with our time source, but do not
# permit the source to query or modify the service on this system.
restrict default kod nomodify notrap nopeer noquery
restrict -6 default kod nomodify notrap nopeer noquery
```

```
# Permit all access over the loopback interface. This could
# be tightened as well, but to do so would effect some of
# the administrative functions.
```

```
restrict 127.0.0.1
restrict -6 ::1
```

在ansible管理节点，配置时间服务器，安装ntp服务，修改ntp配置文件中，上游时间服务器为本机，也就是只从本机同步时间

```
# Hosts on local network are less restricted.
#restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap
```

```
# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).
```

```
server 127.127.1.0
```

```
[root@node68 ~]# service ntpd start
```

```
正在启动 ntpd:
```

启动ntp服务

[确定]

```
[root@node68 ~]#
```

```
# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).
#server 0.centos.pool.ntp.org iburst
#server 1.centos.pool.ntp.org iburst
#server 2.centos.pool.ntp.org iburst
#server 3.centos.pool.ntp.org iburst
server 10.1.32.68 iburst_
```

修改nginx主机上的/etc/chrony.conf文件的同步时间的服务器为node68的节点

```
[root@node72 ~]#
[root@node72 ~]# ntpdate 10.1.32.68
5 Nov 10:49:18 ntpdate[3143]: adjust time server 10.1.32.68 offset -0.002326 sec
[root@node72 ~]#
[root@node72 ~]# date
Sat Nov  5 10:49:20 CST 2016
[root@node72 ~]#
[root@node72 ~]# clock
Sat 05 Nov 2016 10:49:24 AM CST -0.185485 seconds
[root@node72 ~]#
```

在nginx节点上执行ntpdate同步时间

<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

```
[root@node68 ~]# vim /etc/hosts
```

127.0.0.1	localhost localhost.localdomain localhost4 localhost4.localdomain4
::1	localhost localhost.localdomain localhost6 localhost6.localdomain6
10.1.32.68	node68.nwc.com node68
10.1.32.72	node72.nwc.com node72
10.1.32.73	node73.nwc.com node73

在各个节点的/etc/hosts文件中添加名称解析条目

<4> 确保iptables和selinux不会影响实验正常进行

```
[root@node73 ~]# iptables -F
[root@node73 ~]#
[root@node73 ~]# getenforce
Disabled
[root@node73 ~]#
```

本例中关闭iptables和selinux

<5> 在ansible管理节点上部署ansible

```
[root@node68 ~]# yum install -y ansible
```

已加载插件: fastestmirror, security  
设置安装进程  
Determining fastest mirrors

BASE	4.0 kB	00:00
EPEL	4.3 kB	00:00

解决依赖关系  
--> 执行事务检查  
---> Package ansible.noarch 0:1.9.2-1.el6 will be 安装  
处理依赖关系: 安装 ansible 1:1.9.2-1.el6 需要: python2-six

<6> 配置ansible主机可基于ssh秘钥登录被管理主机的root用户



```
[root@node68 ~]# ssh-keygen -t rsa -P ''
Generating public/private rsa key pair. 生成ssh秘钥文件
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
fe:5c:28:bb:2c:22:e3:04:3d:6b:95:81:cd:d4:6a:c3 root@node68.nwc.com
The key's randomart image is:
+--[ RSA 2048 ]-----+
|      ..              |
|      = .             |
|    ..+.             |
|   . Eo               |
|. o.o. S              |
|. + . .               |
| +   o . .            |
| oo . . . = .         |
| ..o . .+.o           |
+-----+
[root@node68 ~]#
```

```
[root@node68 ~]# ssh-copy-id -i .ssh/id_rsa.pub root@node72
The authenticity of host 'node72 (10.1.32.72)' can't be established.
RSA key fingerprint is 61:20:77:df:ac:5c:a5:5c:8d:05:54:dc:f0:77:bc:ba.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'node72,10.1.32.72' (RSA) to the list of known hosts.
root@node72's password:
Now try logging into the machine, with "ssh 'root@node72'", and check in:
    拷贝ssh公钥文件到被管理主机的root用户家目录下，实现基于秘钥登录管理远程主机
    .ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expecting.

[root@node68 ~]# ssh-copy-id -i .ssh/id_rsa.pub root@node73
The authenticity of host 'node73 (10.1.32.73)' can't be established.
RSA key fingerprint is 2c:d6:93:48:df:08:8b:5c:90:4e:53:03:ca:7b:85:25.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'node73,10.1.32.73' (RSA) to the list of known hosts.
root@node73's password:
Now try logging into the machine, with "ssh 'root@node73'", and check in:
    .ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expecting.

[root@node68 ~]#
```

<7> 为ansible配置被管理主机

```
[root@node68 ~]# vim /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
10.1.32.73
```

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.el7ngx.x86_64.rpm
- name: install nginx
  shell: chdir=/root rpm -i nginx-1.10.0-1.el7ngx.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 ~]#
[root@node68 ~]#
[root@node68 ~]#
[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

```

[root@node68 ~]#
[root@node68 ~]# vim test.yml

- hosts: nginx
  remote_user: root
  roles:
    - nginx

```

编辑playbook剧本文件，指明剧本作用的主机，指明远程主机上执行剧本的用户身份，然后用roles指令调用角色

一个playbook文件可调用多个role角色

```

[root@node68 ~]# ansible-playbook --check test.yml
PLAY [nginx] *****

GATHERING FACTS *****
ok: [10.1.32.72]
ok: [10.1.32.73]

TASK: [nginx | install a wget tool] *****
ok: [10.1.32.73]
ok: [10.1.32.72]

TASK: [nginx | download nginx rpm package] *****
skipping: [10.1.32.72]
ok: [10.1.32.72]
skipping: [10.1.32.73]
ok: [10.1.32.73]

TASK: [nginx | install nginx] *****
skipping: [10.1.32.72]
ok: [10.1.32.72]
skipping: [10.1.32.73]
ok: [10.1.32.73]

```

测试执行，发现没有报错信息，说明可以正常运行

```

[root@node68 ~]# ansible-playbook test.yml
正式执行
PLAY [nginx] *****
GATHERING FACTS *****
ok: [10.1.32.73]
ok: [10.1.32.72]

TASK: [nginx | install a wget tool] *****
ok: [10.1.32.72]
ok: [10.1.32.73]

TASK: [nginx | download nginx rpm package] *****
changed: [10.1.32.72]
changed: [10.1.32.73]

TASK: [nginx | install nginx] *****
changed: [10.1.32.73]
changed: [10.1.32.72]

TASK: [nginx | provide a config file] *****
changed: [10.1.32.73]
changed: [10.1.32.72]

TASK: [nginx | move the default index page] *****
changed: [10.1.32.73]
changed: [10.1.32.72]

TASK: [nginx | provied a index page] *****
changed: [10.1.32.73]
changed: [10.1.32.72]

NOTIFIED: [nginx | restart nginx] *****
changed: [10.1.32.72]
changed: [10.1.32.73]

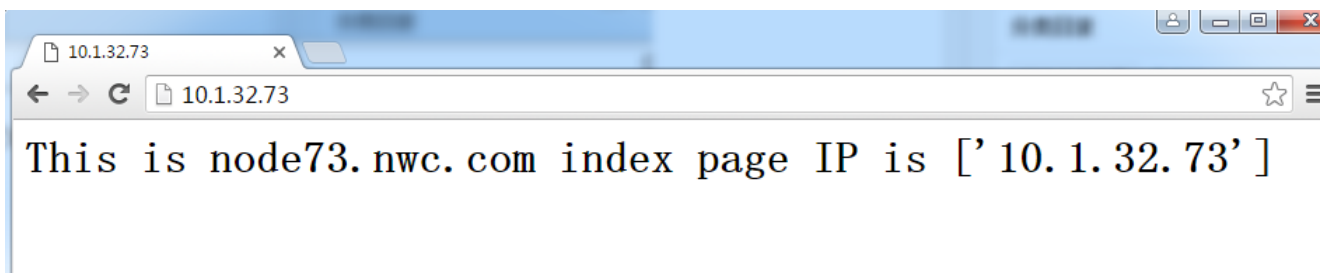
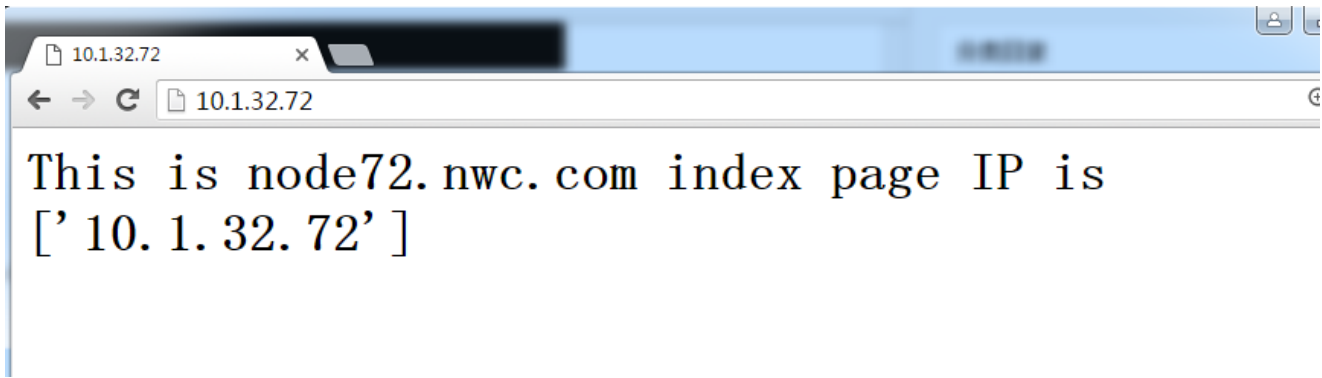
NOTIFIED: [nginx | mail to root] *****
changed: [10.1.32.73]
changed: [10.1.32.72]

PLAY RECAP *****
10.1.32.72      : ok=8    changed=6    unreachable=0    failed=0
10.1.32.73      : ok=8    changed=6    unreachable=0    failed=0

[root@node68 ~]#

```

## 5、验证被管理主机上nginx是否运行正常



```
[root@node68 ~]# ansible nginx -m shell -a 'grep -E "(worker_processes|server_name)" /etc/nginx/nginx.conf'
10.1.32.73 | success | rc=0 >>
worker_processes 2;
server_name node73.nwc.com;

10.1.32.72 | success | rc=0 >>
worker_processes 2;
server_name node72.nwc.com;

[root@node68 ~]# ansible nginx -m shell -a 'ls /usr/share/nginx/html'
10.1.32.72 | success | rc=0 >>
50x.html
index.html
index.html.bak

10.1.32.73 | success | rc=0 >>
50x.html
index.html
index.html.bak

[root@node68 ~]#
```

## 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 ~]#
[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 224.0.0.0:252
}

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

```
[root@node68 ~]# vim keepalived.yml
```

```
- hosts: nginx
  remote_user: root
  roles:
    - keepalived
```

指明剧本运行的主机，执行剧本的远程用户，调用角色

```
[root@node68 ~]# ansible-playbook keepalived.yml
```

正式运行剧本

```
PLAY [nginx] *****
```

```
GATHERING FACTS *****
```

```
ok: [10.1.32.73]
```

```
ok: [10.1.32.72]
```

```
TASK: [keepalived | install keepalived package] *****
```

```
changed: [10.1.32.73]
```

```
changed: [10.1.32.72]
```

```
TASK: [keepalived | move default config file] *****
```

```
changed: [10.1.32.73]
```

```
changed: [10.1.32.72]
```

```
TASK: [keepalived | provide a config file for master] *****
```

```
skipping: [10.1.32.73]
```

```
changed: [10.1.32.72]
```

```
TASK: [keepalived | provide a config file for backup] *****
```

```
skipping: [10.1.32.72]
```

```
changed: [10.1.32.73]
```

```
TASK: [keepalived | restart keepalived] *****
```

```
changed: [10.1.32.73]
```

```
changed: [10.1.32.72]
```

```
NOTIFIED: [keepalived | mail to root] *****
```

```
changed: [10.1.32.73]
```

```
changed: [10.1.32.72]
```

```
PLAY RECAP *****
```

```
10.1.32.72      : ok=6    changed=5    unreachable=0    failed=0
```

```
10.1.32.73      : ok=6    changed=5    unreachable=0    failed=0
```

```
[root@node68 ~]#
```

## 8、验证keepalived对nginx的高可用是否成功

```
[root@node68 ~]# ansible nginx -m shell -a 'ip addr show dev eno33554976'
10.1.32.73 | success | rc=0 >>
3: eno33554976: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:0c:29:c6:b7:86 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::20c:29ff:fec6:b786/64 scope link
        valid_lft forever preferred_lft forever

10.1.32.72 | success | rc=0 >>
3: eno33554976: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:0c:29:ec:d4:19 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.1/24 scope global eno33554976
        valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:feec:d419/64 scope link
        valid_lft forever preferred_lft forever

[root@node68 ~]#
```

发现初始时，VIP在node72上

```
[root@node68 ~]# ansible 10.1.32.72 -m shell -a 'nginx -s stop'
10.1.32.72 | success | rc=0 >>

[root@node68 ~]# ansible nginx -m shell -a 'ip addr show dev eno33554976'
10.1.32.73 | success | rc=0 >>
3: eno33554976: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:0c:29:c6:b7:86 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.1/24 scope global eno33554976
        valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fec6:b786/64 scope link
        valid_lft forever preferred_lft forever

10.1.32.72 | success | rc=0 >>
3: eno33554976: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:0c:29:ec:d4:19 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::20c:29ff:feec:d419/64 scope link
        valid_lft forever preferred_lft forever

[root@node68 ~]#
```

停掉node72上的nginx后，VIP就转移到node73上

## 第十三章 ansible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 ~]#
[root@node68 ~]#
[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 ~]#
[root@node68 ~]#
[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 ~]#
[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 .html
</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

```
[root@node68 ~]# vim lap.yml

- hosts: nginx
  remote_user: root
  roles:
    - lap
```

编辑生成playbook文件，指明角色运行在哪些主机上，运行剧本的远程用户，指明调用的角色

```
[root@node68 ~]# ansible-playbook --check lap.yml
```

测试运行，发现无报错信息

```
PLAY [nginx] *****

GATHERING FACTS *****
ok: [10.1.32.73]
ok: [10.1.32.72]

TASK: [lap | install httpd] *****
changed: [10.1.32.73]
changed: [10.1.32.72]

TASK: [lap | install php] *****
changed: [10.1.32.73]
changed: [10.1.32.72]

TASK: [lap | install php-mysql] *****
changed: [10.1.32.73]
changed: [10.1.32.72]

TASK: [lap | make sure nginx proxy the request to httpd] *****
changed: [10.1.32.73]
changed: [10.1.32.72]
```

### 4、运行剧本，验证反代是否成功



```
[root@node68 ~]# ansible-playbook lap.yml
```

运行剧本，发现无报错信息

```
PLAY [nginx] *****

GATHERING FACTS *****
ok: [10.1.32.73]
ok: [10.1.32.72]

TASK: [lap | install httpd] *****
changed: [10.1.32.73]
changed: [10.1.32.72]

TASK: [lap | install php] *****
changed: [10.1.32.73]
changed: [10.1.32.72]

TASK: [lap | install php-mysql] *****
changed: [10.1.32.72]
changed: [10.1.32.73]

TASK: [lap | make sure nginx proxy the request to httpd] *****
changed: [10.1.32.73]
changed: [10.1.32.72]
```



```
[root@node68 ~]# ansible nginx -m shell -a 'ss -tnl | grep ":80"'
10.1.32.73 | success | rc=0 >>
LISTEN 0 128 *:80 nginx的80和httpd的8080端口均已监听
LISTEN 0 128 :::8080 :::*

10.1.32.72 | success | rc=0 >>
LISTEN 0 128 *:80 *:
LISTEN 0 128 :::8080 :::*
```

[root@node68 ~]#

## 第十四章 ansible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

```

[root@node68 ~]#
[root@node68 ~]# vim mariadb.yml

```

编辑playbook文件，指明剧本执行的被管理主机，指明运行剧本的远程主机的用户，指明调用的角色

```

- hosts: db
  remote_user: root
  roles:
    - mariadb

```

```

[root@node68 ~]# ansible-playbook --check mariadb.yml

```

测试执行剧本，发现无报错信息

```

PLAY [db] *****

GATHERING FACTS *****
ok: [10.1.32.73]

TASK: [mariadb | install mariadb package] *****
changed: [10.1.32.73]

TASK: [mariadb | move old config file] *****
skipping: [10.1.32.73]
ok: [10.1.32.73]

TASK: [mariadb | provide a config file] *****
changed: [10.1.32.73]

TASK: [mariadb | create a testdb] *****
skipping: [10.1.32.73]
ok: [10.1.32.73]

```

### 4、执行剧本，验证配置是否正确

```
[root@node68 ~]#
[root@node68 ~]# ansible-playbook mariadb.yml 正式执行剧本

PLAY [db] *****

GATHERING FACTS *****
ok: [10.1.32.73]

TASK: [mariadb | install mariadb package] *****
changed: [10.1.32.73]

TASK: [mariadb | move old config file] *****
changed: [10.1.32.73]

TASK: [mariadb | provide a config file] *****
changed: [10.1.32.73]

TASK: [mariadb | restart mariadb] *****
changed: [10.1.32.73]

TASK: [mariadb | create a testdb] *****
changed: [10.1.32.73]

PLAY RECAP *****
10.1.32.73          : ok=6    changed=5    unreachable=0    failed=0

[root@node68 ~]#
```

```
[root@node68 ~]#
[root@node68 ~]# ansible db -m shell -a 'mysql -utestuser -h10.1.32.73 -p111111 -e "SHOW DATABASES ;"'
10.1.32.73 | success | rc=0 >> 验证, 以testuser能正常登陆数据库
Database
information_schema
test
testdb
[root@node68 ~]#
```