

1 案例1：环境准备

1.1 问题

本案例要求准备ansible的基础环境：

- 启动6台虚拟机
- 禁用selinux和firewalld
- 编辑/etc/hosts
- 配置yum扩展源并在管理节点安装ansible

1.2 方案

此方案需要准备六台主机，1台管理主机，5台托管主机，以实现批量程序部署，批量运行命令等功能，具体要求如表-1所示：

表-1

主机名	Ip 地址	角色
ansible	192.168.1.51	管理主机
web1	192.168.1.52	托管主机
web2	192.168.1.53	托管主机
db1	192.168.1.54	托管主机
db2	192.168.1.55	托管主机
cache	192.168.1.56	托管主机

1.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：基础环境准备

- 1) 启动6台虚拟机，由于已经讲过怎么创建，这里不再在案例里体现
- 2) 真机配置yum仓库

01. [root@room9pc01 ~]# tar -xf ansible_soft.tar.xz
02. [root@room9pc01 ~]# cd ansible_soft/
03. [root@room9pc01 ansible_soft]# mkdir /var/ftp/ansible
04. [root@room9pc01 ansible_soft]# cp * /var/ftp/ansible
05. [root@room9pc01 ansible_soft]# createrepo /var/ftp/ansible
06. Spawning worker 0 with 1 pkgs
07. Spawning worker 1 with 1 pkgs
08. Spawning worker 2 with 1 pkgs
09. Spawning worker 3 with 1 pkgs
10. Spawning worker 4 with 1 pkgs

[Top](#)

11. Spawning worker 5 with 1 pkgs
12. Workers Finished
13. Saving Primary metadata
14. Saving file lists metadata
15. Saving other metadata
16. Generating sqlite DBs
17. Sqlite DBs complete

3) 修改主机名 (容易区分 , 6台机器都需要修改) 这里以ansible主机为例子

01. [root@localhost ~] # echo ansible > /etc/hostname
02. [root@localhost ~] # hostname ansible

4) 配置ip (6台机器都需要配置) , 这里以ansible主机为例子

01. [root@localhost ~] # vim /etc/sysconfig/network-scripts/ifcfg-eth0
02. # Generated by dracut initrd
03. DEVICE="eth0"
04. ONBOOT="yes"
05. IPV6INIT="no"
06. IPV4_FAILURE_FATAL="no"
07. NM_CONTROLLED="no"
08. TYPE="Ethernet"
09. BOOTPROTO="static"
10. IPADDR=192.168.1.51
11. PREFIX=24
12. GATEWAY=192.168.1.254
13. [root@localhost ~] # systemctl restart network
14. [root@localhost ~] # ifconfig
15. eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
16. inet 192.168.1.51 netmask 255.255.255.0 broadcast 192.168.1.255
17. ether 52:54:00:b2:69:9e txqueuelen 1000 (Ethernet)
18. RX packets 234 bytes 16379 (15.9 KiB)
19. RX errors 0 dropped 36 overruns 0 frame 0
20. TX packets 31 bytes 2618 (2.5 KiB)
21. TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[Top](#)

5) 配置yum客户端 , 在管理节点ansible上面配置

```

01. [ root@ansible ~] # vim /etc/yum.repos.d/local.repo
02. [ local_repo]
03. name=CentOS-$releasever - Base
04. baseurl="ftp://192.168.1.254/system"
05. enabled=1
06. gpgcheck=1
07.
08. [ local]
09. name=local
10. baseurl="ftp://192.168.1.254/ansible"
11. enabled=1
12. gpgcheck=0
13. [ root@ansible ~] # yum clean all
14. [ root@ansible ~] # yum repolist
15. [ root@ansible ~] # yum -y install ansible
16. [ root@ansible ~] # ansible --version
17. ansible 2.4.2.0 //显示版本说明安装成功
18. config file = /etc/ansible/ansible.cfg
19. configured module search path = [ u'/root/.ansible/plugins/modules', u'/usr/share/ansib
20. ansible python module location = /usr/lib/python2.7/site-packages/ansible
21. executable location = /usr/bin/ansible
22. python version = 2.7.5 ( default, Aug 4 2017, 00:39:18) [ GCC 4.8.5 20150623 ( Red Hat

```

6) 请在6台主机上面配置/etc/hosts , 这里以ansible主机为例子

```

01. [ root@ansible ansible] # cat /etc/hosts
02. 192.168.1.51 ansible
03. 192.168.1.52 web1
04. 192.168.1.53 web2
05. 192.168.1.54 db1
06. 192.168.1.55 db2
07. 192.168.1.56 cache

```

2 案例2：主机定义与分组：

2.1 问题

[Top](#)

本案例要求：

- 熟悉ansible配置文件
- 定义主机，分组和子组练习
- 自定义文件，多配置路径练习

2.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：ansible.cfg配置文件

```

01. [ root@ansible ~] # cd /etc/ansible/
02. [ root@ansible ansible] # ls
03. ansible.cfg hosts roles
04. [ root@ansible ansible] # vim ansible.cfg
05. #inventory      = /etc/ansible/hosts    //指定分组文件路径，主机的分组文件hosts
06. [ selinux]       //组名称，selinux的相关选项在这个下面配置
07. ...
08. [ colors]       //组名称，colors的相关选项在这个下面配置
09. ...

```

步骤二：定义主机，分组和子组练习

1) 静态主机的定义

```

01. [ root@ansible ansible] # vim hosts
02. [ web]
03. web1
04. web2
05.
06. [ db]
07. db[ 1: 2]           //1: 2为db1到db2两台主机，1: 20为db1到db20多台主机
08.
09. [ other]
10. cache
11.
12. [ root@ansible ansible] # ansible web -- list- host //显示web组的主机
13. hosts ( 2):
14. web1
15. web2
16. [ root@ansible ansible] # ansible db -- list- host
17. hosts ( 2):
18. db1
19. db2

```

[Top](#)

```

20. [root@ansible ansible] # ansible other --list-host
21.   hosts ( 1 ):
22.     cache
23. [root@ansible ansible] # ansible all --list-host //显示所有组的主机
24.   hosts ( 5 ):
25.     web1
26.     web2
27.     cache
28.     db1
29.     db2

```

2) 直接测试

```

01. [root@ansible ansible] # ansible cache -m ping
02. //测试是否可以连接，若失败颜色为红色
03. cache | UNREACHABLE! => {
04.     "changed": false,
05.     "msg": "Failed to connect to the host via ssh: ssh: Could not resolve hostname cache:
06.     "unreachable": true
07. }

```

3) 修改后测试

```

01. [root@ansible ansible] # vi hosts
02. [other]
03. cache ansible_ssh_user="root" ansible_ssh_pass="a"
04.
05. [root@ansible ansible] # ansible other -m ping //测试成功，颜色为绿色
06. cache | SUCCESS => {
07.     "changed": false,
08.     "ping": "pong"
09. }

```

4) 不检测主机的sshkey，在第一次连接的时候不用输入yes

[Top](#)

```

01. [root@ansible ansible] # vim ansible.cfg
02. 61 host_key_checking = False

```

```

03. [ root@ansible ansible] # vim hosts
04. [ web]
05. web1
06. web2
07.
08. [ web:vars] //web组:变量( vars不改) , web组的多台机器共用一个用户名和密码
09. ansible_ssh_user="root"
10. ansible_ssh_pass="a"
11. [ root@ansible ansible] # ansible web -m ping
12. web2 | SUCCESS => {
13.     "changed": false,
14.     "ping": "pong"
15. }
16. web1 | SUCCESS => {
17.     "changed": false,
18.     "ping": "pong"
19. }

```

步骤三：定义子组

```

01. [ root@ansible ansible] # vi hosts
02. [ app:children] //指定子分组( app可改: children不改) , web , db是提前分好的组
03. web
04. db
05.
06. [ app:vars]
07. ansible_ssh_user="root"
08. ansible_ssh_pass="a"
09. [ root@ansible ansible] # ansible app --list-host //查看
10. hosts ( 4 ):
11. web1
12. web2
13. db1
14. db2
15. [ root@ansible ansible] # ansible app -m ping //测试
16. web1 | SUCCESS => {
17.     "changed": false,
18.     "ping": "pong"
19. }
20. web2 | SUCCESS => {

```

[Top](#)

```

21.     "changed": false,
22.     "ping": "pong"
23. }
24. db1 | SUCCESS => {
25.     "changed": false,
26.     "ping": "pong"
27. }
28. db2 | SUCCESS => {
29.     "changed": false,
30.     "ping": "pong"
31. }

```

步骤四：多路径练习

自定义的ansible文件只在当前路径生效

1) 多路径

```

01. [ root@ansible ~] # mkdir aaa
02. [ root@ansible ~] # cd aaa/
03. [ root@ansible aaa] # vim my host
04. [ app1]
05. web1
06. db1
07.
08. [ app2]
09. web2
10. db2
11.
12. [ app:children]
13. app1
14. app2
15.
16. [ other]
17. cache
18.
19. [ app:vars]
20. ansible_ssh_user="root"
21. ansible_ssh_pass="a"
22. [ root@ansible aaa] # touch ansible.cfg
23. [ root@ansible aaa] # grep -Ev "^#|^$" /etc/ansible/ansible.cfg
24. [ defaults]

```

[Top](#)

```
25.  roles_path  = /etc/ansible/roles: /usr/share/ansible/roles
26.  host_key_checking = False
27.  [ inventory ]
28.  [ privilege_escalation ]
29.  [ paramiko_connection ]
30.  [ ssh_connection ]
31.  [ persistent_connection ]
32.  [ accelerate ]
33.  [ selinux ]
34.  [ colors ]
35.  [ diff ]
36.
37.  [ root@ansible aaa ] # vim ansible.cfg
38.  [ defaults ]
39.  inventory = my host
40.  host_key_checking = False
```

2) 测试结果

```
01.  [ root@ansible aaa ] # ansible app1 - m ping
02.  web1 | SUCCESS => {
03.      "changed": false,
04.      "ping": "pong"
05.  }
06.  db1 | SUCCESS => {
07.      "changed": false,
08.      "ping": "pong"
09.  }
10.  [ root@ansible aaa ] # ansible app - m ping
11.  web1 | SUCCESS => {
12.      "changed": false,
13.      "ping": "pong"
14.  }
15.  db1 | SUCCESS => {
16.      "changed": false,
17.      "ping": "pong"
18.  }
19.  db2 | SUCCESS => {
20.      "changed": false,
21.      "ping": "pong"
```

[Top](#)


```

22.     }
23.     web2 | SUCCESS => {
24.         "changed": false,
25.         "ping": "pong"
26.     }
27. [root@ansible aaa] # ansible app --list-host
28.     hosts ( 4 ):
29.         web1
30.         db1
31.         web2
32.         db2
33. [root@ansible aaa] # cd
34. [root@ansible ~] # ansible app1 --list-host //切换到别的目录,测试失败
35. [WARNING]: Could not match supplied host pattern, ignoring: app1
36.
37. [WARNING]: No hosts matched, nothing to do
38.
39.     hosts ( 0 ):

```

3 案例3：动态主机

3.1 问题

本案例要求：

- 脚本输出主机列表

3.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：脚本输出主机列表

```

01. [root@ansible ~] # cd aaa
02. [root@ansible aaa] # vim host.py
03. #!/usr/bin/python
04. import json
05. hostlist = {}
06. hostlist["bb"] = ["192.168.1.52", "192.168.1.53"]
07. hostlist["192.168.1.54"] = {
08.     "ansible_ssh_user": "root", "ansible_ssh_pass": "pwd"
09. }
10. hostlist["aa"] = {
11.     "hosts": ["192.168.1.55", "192.168.1.56"],

```

[Top](#)

```

12.         "vars" : {
13.             "ansible_ssh_user": "root", "ansible_ssh_pass": "pwd"
14.         }
15.     }
16.     print(json.dumps( hostlist))
17. [ root@ansible aaa] # chmod 755 ./host.py

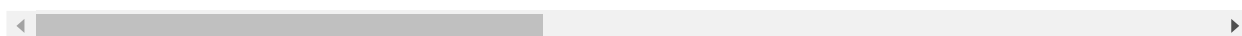
```

步骤二：脚本输出样例（这样写输出的结果有些乱）

```

01. [ root@ansible aaa] # ./host.py
02. { "aa": { "hosts": [ "192.168.1.55", "192.168.1.56"], "vars": { "ansible_ssh_user": "root",

```



步骤三：可以用shell脚本输出

```

01. [ root@ansible aaa] # vim my.sh
02. #!/bin/bash
03. echo '
04. { "aa": {
05.     "hosts":
06.         [ "192.168.1.55", "192.168.1.56"],
07.     "vars": {
08.         "ansible_ssh_user": "root",
09.         "ansible_ssh_pass": "a"}
10.     },
11. } '
12. [ root@ansible aaa] # chmod 755 my.sh
13. [ root@ansible aaa] # ./my.sh
14.
15. { "aa": {
16.     "hosts":
17.         [ "192.168.1.55", "192.168.1.56"],
18.     "vars": {
19.         "ansible_ssh_user": "root",
20.         "ansible_ssh_pass": "a"}
21.     },
22. }
23. [ root@ansible aaa] # vim ansible.cfg
24. [ defaults]

```

[Top](#)

```

25.  inventory = my .sh
26.  host_key_checking = False
27.  [ root@ansible aaa] # ansible aa - m ping
28.  192.168.1.55 | SUCCESS => {
29.      "changed": false,
30.      "ping": "pong"
31.  }
32.  192.168.1.56 | SUCCESS => {
33.      "changed": false,
34.      "ping": "pong"
35.  }

```

步骤二：批量执行

1) 查看负载

```

01.  [ root@ansible aaa] # ansible app - m command - a 'uptime'
02.  db1 | SUCCESS | rc=0 >>
03.  11:35:52 up 1:59, 2 users, load average: 0.00, 0.01, 0.01
04.
05.  web1 | SUCCESS | rc=0 >>
06.  11:35:52 up 2:00, 2 users, load average: 0.00, 0.01, 0.02
07.
08.  db2 | SUCCESS | rc=0 >>
09.  11:35:53 up 1:59, 2 users, load average: 0.00, 0.01, 0.03
10.
11.  web2 | SUCCESS | rc=0 >>
12.  11:35:52 up 1:59, 2 users, load average: 0.00, 0.01, 0.02

```

2) 查看时间

```

01.  [ root@ansible aaa] # ansible app - m command - a 'date +%F\ %T'
02.  db1 | SUCCESS | rc=0 >>
03.  2018-09-06 11:42:18
04.
05.  web1 | SUCCESS | rc=0 >>
06.  2018-09-06 11:42:18
07.
08.  web2 | SUCCESS | rc=0 >>

```

[Top](#)

09. 2018-09-06 11:42:18
- 10.
11. db2 | SUCCESS | rc=0 >>
12. 2018-09-06 11:42:19

4 案例4：批量部署证书文件

4.1 问题

本案例要求：

- 创建一对密钥
- 给所有主机部署密钥

4.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：批量部署证书文件，给所有主机部署密钥

1) 创建密钥

01. [root@ansible aaa] # cd /root/.ssh/
02. [root@ansible .ssh] # vi /etc/ansible/hosts
03. [web]
04. web1
05. web2
- 06.
- 07.
08. [db]
09. db[1 2]
- 10.
11. [other]
12. cache
13. [root@ansible .ssh] # ansible all - m ping //直接ping会报错
14. [root@ansible .ssh] # ssh-keygen -t rsa -b 2048 -N '' //创建密钥

2) 给所有主机部署密钥

01. [root@ansible .ssh] # ansible all - m authorized_key - a "user=root exclusive=true manage_c
02. SSH password: //输入密码
03. [root@ansible .ssh] # ansible all - m ping //成功
04. web2 | SUCCESS => {

[Top](#)

```

05.     "changed": false,
06.     "ping": "pong"
07. }
08. db2 | SUCCESS => {
09.     "changed": false,
10.     "ping": "pong"
11. }
12. web1 | SUCCESS => {
13.     "changed": false,
14.     "ping": "pong"
15. }
16. cache | SUCCESS => {
17.     "changed": false,
18.     "ping": "pong"
19. }
20. db1 | SUCCESS => {
21.     "changed": false,
22.     "ping": "pong"
23. }
24. [ root@ansible .ssh] # ssh web1      //不需要输入密码,可以直接登陆
25. Last login: Thu Sep  6 11:49:00 2018 from 192.168.1.51
26. [ root@web1 ~] #

```

5 案例5：练习模块

5.1 问题

本案例要求：

- 练习使用command, shell, raw, script模块

5.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：练习模块

ansible-doc //模块的手册，相当于man

ansible-doc -l //列出所有模块

ansible-doc 模块名 //查看指定模块的帮助信息

1) ping模块

[Top](#)

```
01. [ root@ansible .ssh] # ansible web1 -m ping
```

```
02. web1 | SUCCESS => {
03.     "changed": false,
04.     "ping": "pong"
05. }
```

2) command模块

```
01. [ root@ansible .ssh ] # ansible web1 - m command - a 'chdir=/tmp touch f1' //创建成功
02. [ root@web1 ~ ] # cd /tmp/
03. [ root@web1 tmp ] # ls //在web1上面查看
04. f1
```

3) shell模块

```
01. [ root@ansible .ssh ] # ansible web1 - m shell - a 'chdir=/tmp touch f2' //创建成功
02. [ root@web1 ~ ] # cd /tmp/
03. [ root@web1 tmp ] # ls //在web1上面查看
04. f2
```

4) raw模块

```
01. [ root@ansible .ssh ] # ansible web1 - m raw - a 'chdir=/tmp touch f3'
02. //文件可以创建，但无法切换目录，文件在用户家目录下生成
03. web1 | SUCCESS | rc=0 >>
04. Shared connection to web1 closed.
05. [ root@web1 tmp ] # cd /root/
06. [ root@web1 ~ ] # ls //在web1上面查看
07. f3
```

5) script模块

对于太复杂的命令，可以写个脚本，然后用script模块执行

在web1主机上创建zhangsan3用户，修改zhangsan3的密码为123456，设置zhangsan3第一次登陆必须修改密码

用命令写：

[Top](#)

```
01. [ root@ansible .ssh ] # ansible web1 - m shell - a 'useradd zhangsan3'
```

```

02. [root@ansible .ssh] # ansible web1 - m shell - a 'echo 123456 | passwd -- stdin zhangsan3'
03. [root@ansible .ssh] # ssh -l zhangsan3 web1
04. zhangsan3@web1's password: //输入zhangsan3的密码
05. [root@ansible .ssh] # ansible web1 - m shell - a 'chage - d 0 zhangsan3'
06. [root@ansible .ssh] # ssh -l zhangsan3 web1

```

用脚本写，script模块执行：

```

01. [root@ansible .ssh] # vim user.sh
02. #!/bin/bash
03. useradd zhangsan3
04. echo 123456 | passwd -- stdin zhangsan3
05. chage - d 0 zhangsan3
06. echo
07. [root@ansible .ssh] # ansible web1 - m script - a './user.sh'
08. web1 | SUCCESS => {
09.     "changed": true,
10.     "rc": 0,
11.     "stderr": "Shared connection to web1 closed. \r\n",
12.     "stdout": "Changing password for user zhangsan3. \r\npasswd: all authentication token
13.     "stdout_lines": [
14.         "Changing password for user zhangsan3.",
15.         "passwd: all authentication tokens updated successfully.",
16.         ""
17.     ]
18. }
19. [root@ansible .ssh] # ssh -l lisi web1
20. lisi@web1's password:
21. You are required to change your password immediately ( root enforced)
22. Last login: Thu Sep  6 14: 51: 33 2018 from 192.168.1.51
23. WARNING: Your password has expired.
24. You must change your password now and login again!
25. Changing password for user lisi.
26. Changing password for lisi.
27. ( current) UNIX password:

```

6 案例6：模块练习

[Top](#)

6.1 问题

本案例要求：

- 使用copy模块同步数据
- 使用lineinfile模块编辑文件
- 使用replace模块修改文件

6.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：模块练习

1) 使用copy模块同步数据

src：要复制到进程主机的文件在本地的地址,可以是绝对路径,也可以是相对路径。如果路径是一个目录,它将递归复制。在这种情况下,如果路径使用"/"来结尾,则只复制目录里的内容,如果没有使用"/"来结尾,则包含目录在内的整个内容全部复制,类似于rsync

dest：必选项。进程主机的绝对路径,如果源文件是一个目录,那么该路径也必须是个目录

backup：在覆盖之前将原文件备份,备份文件包含时间信息。有两个选项:yes|no

force：如果目标主机包含该文件,但内容不同,如果设置为yes,则强制覆盖,如果为no,则只有当目标主机的目标位置不存在该文件时,才复制。默认为yes

```

01. [root@ansible .ssh] # ansible all - m shell - a 'cat /etc/resolv.conf'
02. //查看/etc/resolv.conf
03. cache | SUCCESS | rc=0 >>
04. ; generated by /usr/sbin/dhclient-script
05. nameserver 192.168.1.254
06. search localhost
07.
08. db2 | SUCCESS | rc=0 >>
09. ; generated by /usr/sbin/dhclient-script
10. nameserver 192.168.1.254
11. search localhost
12.
13. web1 | SUCCESS | rc=0 >>
14. ; generated by /usr/sbin/dhclient-script
15. nameserver 192.168.1.254
16. search localhost
17.
18. web2 | SUCCESS | rc=0 >>
19. ; generated by /usr/sbin/dhclient-script
20. nameserver 192.168.1.254
21. search localhost
22.
23. db1 | SUCCESS | rc=0 >>
24. ; generated by /usr/sbin/dhclient-script

```

[Top](#)


```

25. nameserver 192.168.1.254
26. search localhost
27.
28. [ root@ansible .ssh] # vi /etc/resolv.conf
29. nameserver 172.40.1.10
30. [ root@ansible .ssh] # ansible all - m copy - a 'src=/etc/resolv.conf dest=/etc/resolv.conf'
31. [ root@ansible .ssh] # ansible all - m shell - a 'cat /etc/resolv.conf'
32. //查看有nameserver 172.40.1.10
33. [ root@ansible ~] # mkdir aa
34. [ root@ansible ~] # ansible all - m copy - a 'src=/root/aa dest=/root/a.log'
35. //复制本机的目录/root/aa到其他机器的/root/a.log，复制目录只能少数批量执行同步
36. [ root@ansible ~] # ansible all - m shell - a 'ls -ld /root'
37. db2 | SUCCESS | rc=0 >>
38. dr- xr- x- - - . 4 root root 167 Sep 6 11: 48 /root
39.
40. web2 | SUCCESS | rc=0 >>
41. dr- xr- x- - - . 4 root root 167 Sep 6 11: 48 /root
42.
43. cache | SUCCESS | rc=0 >>
44. dr- xr- x- - - . 4 root root 177 Sep 6 14: 35 /root
45.
46. db1 | SUCCESS | rc=0 >>
47. dr- xr- x- - - . 4 root root 167 Sep 6 11: 48 /root
48.
49. web1 | SUCCESS | rc=0 >>
50. dr- xr- x- - - . 4 root root 177 Sep 6 14: 35 /root

```

2) 使用lineinfile模块编辑文件

以行为基础，整行修改(整行被替换掉)

```

01. [ root@ansible ~] # ansible cache - m lineinfile \
02. - a 'path=/etc/sysconfig/network-scripts/ifcfg-eth0 \
03. regexp="^ONBOOT=" line="ONBOOT=\\"no\\""'
04.
05. cache | SUCCESS => {
06.     "backup": "",
07.     "changed": true,
08.     "msg": "line replaced"
09. }

```

[Top](#)

3) 使用replace模块修改文件

修改文件的某一部分(替换一行中匹配的内容)，以正则表达式匹配为基础修改

```
01. [root@ansible ~] # ansible cache - m replace - a \
02. 'path=/etc/sysconfig/network-scripts/ifcfg-eth0 \
03. regexp="^(ONBOOT=) .*" replace="\1"yes\ ""
04.
05. cache | SUCCESS => {
06.     "changed": true,
07.     "msg": "1 replacements made"
08. }
```

7 案例7：综合练习

7.1 问题

本案例要求：

- 安装Apache并修改监听端口为8080
- 修改ServerName配置，执行apachectl -t命令不报错
- 设置默认主页hello world
- 启动服务并设开机自启

7.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：熟悉模块

1) yum模块

```
01. [root@ansible ~] # ansible other - m yum - a 'name="lrzsz" state=removed'
02. //lrzsz软件包名，removed=absent删除
03. [root@ansible ~] # ansible other - m yum - a 'name="lrzsz,lftp" state=installed'
04. //安装多个软件包，不写state默认为安装
```

2) service模块

```
01. [root@ansible ~] # ansible other - m service - a 'name="sshd" enabled="yes" state="starte
```

[Top](#)

3) setup模块

filter 过滤指定的关键字（可以过滤到我们需要的信息）

```

01. [root@ansible ~] # ansible cache - m setup - a 'filter=os'
02. cache | SUCCESS => {
03.     "ansible_facts": {},
04.     "changed": false
05. }
06. [root@ansible ~] # ansible cache - m setup - a 'filter=ansible_distribution'
07. cache | SUCCESS => {
08.     "ansible_facts": {
09.         "ansible_distribution": "CentOS"
10.     },
11.     "changed": false
12. }

```

步骤二：安装Apache

1) 安装Apache服务设置开机自启

```

01. [root@ansible ~] # ansible cache - m yum - a 'name=httpd state=installed'
02. [root@ansible ~] # ansible cache - m service - a 'name=httpd enabled=yes state=started'

```

2) 修改端口号为8080

```

01. [root@ansible ~] # ssh cache
02. Last login: Thu Sep  6 15:30:33 2018 from 192.168.1.51
03. [root@cache ~] # cat /etc/httpd/conf/httpd.conf | grep Listen
04. Listen 80
05. [root@ansible ~] # ansible cache - m lineinfile - a 'path="/etc/httpd/conf/httpd.conf" regexp=
06.     "backup": "",
07.     "changed": true,
08.     "msg": "line replaced"
09. }
10. [root@ansible ~] # ssh cache
11. Listen 8080

```

[Top](#)

步骤三：修改ServerName配置，执行apachectl -t命令不报错

1) 没有修改之前

- ```

01. [root@cache ~]# apachectl -t //有报错
02. AH00558: httpd: Could not reliably determine the server's fully qualified domain name, usi
03. Syntax OK

```

## 2) 修改之后

- ```

01. [root@ansible ~]# ansible cache - m lineinfile - a 'path="/etc/httpd/conf/httpd.conf" rege
02. cache | SUCCESS => {
03.     "backup": "",
04.     "changed": true,
05.     "msg": "line added"
06. }
07. [root@ansible ~]# ssh cache
08. Last login: Thu Sep  6 15:36:08 2018 from 192.168.1.51
09. [root@cache ~]# apachectl -t
10. Syntax OK

```

步骤四：设置默认主页为hello world

- ```

01. [root@ansible ~]# ansible cache - m copy - a 'src=/root/index.html dest=/var/www/html/
02. cache | SUCCESS => {
03. "changed": true,
04. "checksum": "22596363b3de40b06f981fb85d82312e8c0ed511",
05. "dest": "/var/www/html/index.html",
06. "gid": 0,
07. "group": "root",
08. "md5sum": "6f5902ac237024bdd0c176cb93063dc4",
09. "mode": "0644",
10. "owner": "root",
11. "size": 12,
12. "src": "/root/.ansible/tmp/ansible-tmp-1536219767.29-30682157793478/source",
13. "state": "file",
14. "uid": 0
15. }

```

[Top](#)

[Top](#)