[Top](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/ARCHITECTURE/DAY01/CASE/01/index.html" \l "page_top_case)

# NSD ARCHITECTURE DAY01

1. [案例1：环境准备](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/ARCHITECTURE/DAY01/CASE/01/index.html" \l "case1)
2. [案例2：批量部署证书文件](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/ARCHITECTURE/DAY01/CASE/01/index.html" \l "case2)
3. [案例3：主机定义与分组](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/ARCHITECTURE/DAY01/CASE/01/index.html" \l "case3)
4. [案例4：练习理解批量执行](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/ARCHITECTURE/DAY01/CASE/01/index.html" \l "case4)
5. [案例5：创建用户](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/ARCHITECTURE/DAY01/CASE/01/index.html" \l "case5)
6. [案例6：练习模块](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/ARCHITECTURE/DAY01/CASE/01/index.html" \l "case6)
7. [案例7：练习模块](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/ARCHITECTURE/DAY01/CASE/01/index.html" \l "case7)
8. [案例8：模块练习](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/ARCHITECTURE/DAY01/CASE/01/index.html" \l "case8)

## 1 案例1：环境准备

### 1.1 问题

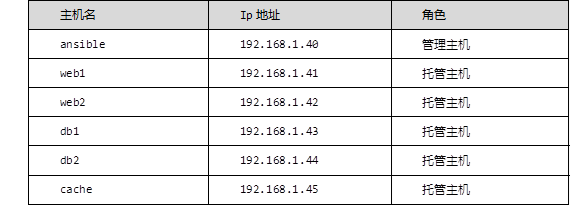
本案例要求准备ansible的基础环境：

* 启动6台虚拟机
* 2cpu，1.5G 以上内存
* 10G 以上硬盘，1块网卡

### 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



### 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

1. [student@room9pc01~]$ mkdir /var/ftp/ansible
2. [student@room9pc01 ~]$ cd /linux-soft/04
3. [student@room9pc01 ansible]$ ls
4. ansible-2.4.2.0-2.el7.noarch.rpm python-paramiko-2.1.1-4.el7.noarch.rpm
5. python2-jmespath-0.9.0-3.el7.noarch.rpm python-passlib-1.6.5-2.el7.noarch.rpm
6. python-httplib2-0.9.2-1.el7.noarch.rpm sshpass-1.06-2.el7.x86\_64.rpm
7. [student@room9pc01 ansible]$ cp \* /var/ftp/ansible/
8. [student@room9pc01 ansible]$ createrepo /var/ftp/ansible/
9. Spawning worker 0 with 2 pkgs
10. Spawning worker 1 with 2 pkgs
11. Spawning worker 2 with 1 pkgs
12. Spawning worker 3 with 1 pkgs
13. Workers Finished
14. Saving Primary metadata
15. Saving file lists metadata
16. Saving other metadata
17. Generating sqlite DBs
18. 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

1. [root@localhost ~]# echo ansible > /etc/hostname
2. [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

1. [root@localhost ~]# vim /etc/sysconfig/network-scripts/ifcfg-eth0
2. # Generated by dracut initrd
3. DEVICE="eth0"
4. ONBOOT="yes"
5. IPV6INIT="no"
6. IPV4\_FAILURE\_FATAL="no"
7. NM\_CONTROLLED="no"
8. TYPE="Ethernet"
9. BOOTPROTO="static"
10. IPADDR=192.168.1.40
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 192.168.1.40 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

5）配置yum客户端，在管理节点ansible上面配置

1. [root@ansible ~]# vim /etc/yum.repos.d/local.repo
2. [local\_repo]
3. name=CentOS-$releasever - Base
4. baseurl="ftp://192.168.1.254/system"
5. enabled=1
6. gpgcheck=1
7. [local]
8. name=local
9. baseurl="ftp://192.168.1.254/ansible"
10. enabled=1
11. gpgcheck=0
12. [root@ansible ~]# yum clean all
13. [root@ansible ~]# yum repolist
14. [root@ansible ~]# yum -y install ansible
15. [root@ansible ~]# ansible --version
16. ansible 2.4.2.0        //显示版本说明安装成功
17. config file = /etc/ansible/ansible.cfg
18. configured module search path = [u'/root/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']
19. ansible python module location = /usr/lib/python2.7/site-packages/ansible
20. executable location = /usr/bin/ansible
21. python version = 2.7.5 (default, Apr 11 2018, 07:36:10) [GCC 4.8.5 20150623 (Red Hat 4.8.5-28)]

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

1. [root@ansible ansible]# cat /etc/hosts
2. 192.168.1.40 ansible
3. 192.168.1.41 web1
4. 192.168.1.42 web2
5. 192.168.1.43 db1
6. 192.168.1.44 db2
7. 192.168.1.45 cache

## 2 案例2：批量部署证书文件

### 2.1 问题

本案例要求：

* 创建一对密钥
* cd /root/.ssh
* ssh-keygen -t rsa -b 2048 -N '' -f key
* 给所有主机部署密钥
* ssh-copy-id -i key.pub 主机名称

### 2.2 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

1. [root@ansible myansible]# cd /root/.ssh/
2. [root@ansible .ssh]# vim /etc/ansible/hosts
3. [web]
4. web1
5. web2
6. [db]
7. db[1:2]
8. [other]
9. cache
10. [root@ansible .ssh]# ansible all -m ping //直接ping会报错
11. [root@ansible .ssh]# ssh-keygen -t rsa -b 2048 -N '' -f key //创建密钥

2）给所有主机部署密钥

1. [root@ansible .ssh]# ssh-copy-id -i key.pub 主机名称
2. [root@ansible .ssh]# ansible all -m ping //失败

## 3 案例3：主机定义与分组

### 3.1 问题

本案例要求：

* 给所有主机部署 key
* 在 inventory 文件中指定 key 的位置
* 配置主机分组，自定义文件，在重新定义一个新的 ansible.cfg
* 在自定义的文件夹中完成之前的配置

### 3.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：给所有主机部署key，案例2已经做过，这里不再重复

步骤二：在 inventory 文件中指定 key 的位置

1. [all:vars]
2. ansible\_ssh\_private\_key\_file="/root/.ssh/key"
3. [root@ansible .ssh]# ansible all -m ping    //成功
4. [root@ansible .ssh]# ssh -i key cache        //不需要输入密码,可以直接登陆
5. Last login: Thu Sep 6 11:49:00 2018 from 192.168.1.40
6. ...
7. [root@web1 ~]#

步骤三：配置主机分组，自定义文件，在重新定义一个新的 ansible.cfg

自定义的ansible文件只在当前路径生效

1）自定义文件

1. [root@ansible ~]# mkdir myansible
2. [root@ansible ~]# cd myansible/
3. [root@ansible myansible]# vim myhost
4. [app1]
5. web1
6. db1
7. [app2]
8. web2
9. db2
10. [app:children]
11. app1
12. app2
13. [other]
14. cache
15. [all:vars]
16. ansible\_ssh\_private\_key\_file="/root/.ssh/key"
17. [root@ansible myansible]# touch ansible.cfg
18. [root@ansible myansible]# vim ansible.cfg
19. [defaults]
20. inventory = myhost
21. host\_key\_checking = False

2）测试结果

1. [root@ansible myansible]# ansible app1 -m ping
2. web1 | SUCCESS => {
3. "changed": false,
4. "ping": "pong"
5. }
6. db1 | SUCCESS => {
7. "changed": false,
8. "ping": "pong"
9. }
10. [root@ansible myansible]# 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"
22. }
23. web2 | SUCCESS => {
24. "changed": false,
25. "ping": "pong"
26. }
27. [root@ansible myansible]# ansible app --list-host
28. hosts (4):
29. web1
30. db1
31. web2
32. db2
33. [root@ansible myansible]# 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. [WARNING]: No hosts matched, nothing to do
37. hosts (0):

步骤五：在自定义的文件夹中完成之前的配置，由于步骤都一样，这里不再重复

## 4 案例4：练习理解批量执行

### 4.1 问题

本案例要求：

* shell
* 执行以下命令查看结果，并说明原因
* ansible web -m shell -a "echo ${HOSTNAME}"
* ansible web -m shell -a 'echo ${HOSTNAME}'
* testfile 文件在哪里
* ansible cache -m shell -a 'cd /tmp'
* ansible cache -m shell -a 'touch testfile'

### 4.2 步骤

实现此案例需要按照如下步骤进行。

1）shell

1. ansible web -m shell -a "echo ${HOSTNAME}"
2. ansible web -m shell -a 'echo ${HOSTNAME}'

2）创建的文件在哪查看

1. ansible cache -m shell -a 'cd /tmp'
2. ansible cache -m shell -a 'touch testfile'

注：

1）变量解析

ansible 执行命令是二次解析，第一次在本机解析, 第二次在执行机器解析，需要第二次解析的变量要转移（\）

2）创建的文件在哪里

文件在用户家目录，ansible 是使用 ssh 多次连接执行，连接退出以后之前的状态就全部失效了

解决方法：使用 chdir 代替 cd 命令

ansible cache -m shell -a 'chdir=/tmp touch testfile'

## 5 案例5：创建用户

### 5.1 问题

本案例要求：

* 添加用户
* 给 web1 db2 添加用户 nb
* 设置 nb 的密码为 123

### 5.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：添加用户

在web1和 db2主机上创建nb用户，修改nb的密码为123（以web1为例子）

1. [root@ansible .ssh]# ansible web1 -m shell -a 'useradd nb'
2. [root@ansible .ssh]# ansible web1 -m shell -a 'echo 123 | passwd --stdin nb'
3. [root@ansible .ssh]# ssh -l nb web1

## 6 案例6：练习模块

### 6.1 问题

本案例要求：

* 添加用户
* 给所有 web 主机添加用户 wk
* 要求 nb 用户与 wk 用户不能出现在同一台主机上
* 设置 wk 用户的 密码是 456

### 6.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：添加用户

对于太复杂的命令，可以写个脚本，然后用script模块执行

用脚本写，script模块执行

1. [root@ansible .ssh]# vim a.sh
2. #!/bin/bash
3. id nb
4. if [ $? != 0 ];then
5. useradd wk
6. echo 123 | passwd --stdin wk
7. fi
8. [root@ansible ~]# ansible web -m script -a '/root/.ssh/a.sh'
9. web1 | SUCCESS => {
10. "changed": true,
11. "rc": 0,
12. "stderr": "Shared connection to web1 closed.\r\n",
13. "stdout": "uid=1000(zhangsan3) gid=1000(zhangsan3) 组=1000(zhangsan3)\r\n",
14. "stdout\_lines": [
15. "uid=1000(zhangsan3) gid=1000(zhangsan3) 组=1000(zhangsan3)"
16. ]
17. }
18. web2 | SUCCESS => {
19. "changed": true,
20. "rc": 0,
21. "stderr": "Shared connection to web2 closed.\r\n",
22. "stdout": "id: zhangsan3: no such user\r\nuseradd：用户“wkl”已存在\r\n更改用户 wkl 的密码 。\r\npasswd：所有的身份验证令牌已经成功更新。\r\n",
23. "stdout\_lines": [
24. "id: zhangsan3: no such user",
25. "useradd：用户“wkl”已存在",
26. "更改用户 wkl 的密码 。",
27. "passwd：所有的身份验证令牌已经成功更新。"
28. ]
29. }

验证结果

1. [root@ansible .ssh]# ssh -i key web1 ls /home/ //web1上面只有nb用户
2. nb
3. [root@ansible .ssh]# ssh -i key web2 ls /home/ //web2上面wk用户
4. wk

## 7 案例7：练习模块

### 7.1 问题

本案例要求：

* 批量修改配置文件
* 批量修改所有机器的 dns 配置 /etc/resolv.conf
* 批量同步所有机器的 yum 配置文件
* 给所有 db 主机开启 binlog 日志
* log\_bin = mysql-bin
* binlog-format = mixed

### 7.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：练习模块

1）批量修改配置文件

批量修改所有机器的 dns 配置 /etc/resolv.conf

1. [root@ansible .ssh]# ansible all -m shell -a 'cat /etc/resolv.conf'
2. //查看/etc/resolv.conf
3. cache | SUCCESS | rc=0 >>
4. ; generated by /usr/sbin/dhclient-script
5. nameserver 192.168.1.254
6. search localhost
7. db2 | SUCCESS | rc=0 >>
8. ; generated by /usr/sbin/dhclient-script
9. nameserver 192.168.1.254
10. search localhost
11. web1 | SUCCESS | rc=0 >>
12. ; generated by /usr/sbin/dhclient-script
13. nameserver 192.168.1.254
14. search localhost
15. web2 | SUCCESS | rc=0 >>
16. ; generated by /usr/sbin/dhclient-script
17. nameserver 192.168.1.254
18. search localhost
19. db1 | SUCCESS | rc=0 >>
20. ; generated by /usr/sbin/dhclient-script
21. nameserver 192.168.1.254
22. [root@ansible .ssh]# vim /etc/resolv.conf
23. nameserver 172.40.92.6
24. [root@ansible .ssh]# ansible all -m copy -a 'src=/etc/resolv.conf dest=/etc/resolv.conf' //复制本机的resolv.conf到其他主机
25. [root@ansible .ssh]# ansible all -m shell -a 'cat /etc/resolv.conf'
26. //查看有nameserver 172.40.92.6

批量同步所有机器的 yum 配置文件

1. [root@ansible ~]# ansible all -m copy -a 'src=/etc/yum.repos.d/ \
2. dest=/etc/yum.repos.d/'

2） 给所有 db 主机开启 binlog 日志

log\_bin = mysql-bin

binlog-format = mixed

1. [root@ansible ansible]# ansible db -m yum -a 'name="mariadb-server" state=installed'
2. [root@ansible ansible]# ansible db -m service -a 'name="mariadb" enabled="yes" state="started"'

拷贝一个mariadb的配置文件到ansible的管理主机上面，修改其文件

1. [root@ansible ansible]# vim /root/my.cnf
2. [mysqld]
3. log-bin=mysql-bin
4. binlog-format=mixed
5. ...
6. [root@ansible ansible]# ansible db -m copy -a 'src=/root/my.cnf dest=/etc/my.cnf'
7. [root@ansible ansible]# ansible db -m service -a 'name="mariadb" enabled="yes" state="started"'

## 8 案例8：模块练习

### 8.1 问题

本案例要求：

* 使用copy模块同步 my.cnf 配置文件
* 使用 lineinfile 模块 修改 binlog 格式
* 使用 replace 模块修改 binlog 格式

### 8.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：综合练习

1）使用copy模块同步my.cnf配置文件

1. [root@ansible ansible]# ansible db -m copy -a 'src=/root/my.cnf dest=/etc/my.cnf'

2）使用 lineinfile 模块 修改 binlog 格式

类似sed的一种行编辑替换模块

path 目标文件文件

regexp 正则表达式，要修改的行

line 最终修改的结果

1. [root@ansible ~]# ansible db -m lineinfile -a 'path="/etc/my.cnf" \
2. regexp="^binlog-format" line="binlog-format=row"'

3）使用 replace 模块修改 binlog 格式

类似sed的一种行编辑替换模块

path 目的文件

regexp 正则表达式

replace 替换后的结果

1. [root@ansible ansible]# ansible other -m replace -a 'path="/etc/my.cnf" regexp="row" replace="mixed"'

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）

1. [root@ansible ~]# ansible cache -m setup -a 'filter=os'
2. cache | SUCCESS => {
3. "ansible\_facts": {},
4. "changed": false
5. }
6. [root@ansible ~]# ansible cache -m setup -a 'filter=ansible\_distribution'
7. cache | SUCCESS => {
8. "ansible\_facts": {
9. "ansible\_distribution": "CentOS"
10. },
11. "changed": false
12. }