1. [案例1：ansible应用案例](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/AUTOMATION/DAY03/CASE/01/index.html" \l "case1)
2. [案例2：加密敏感数据](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/AUTOMATION/DAY03/CASE/01/index.html#case2)
3. [案例3：Ansible Roles](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/AUTOMATION/DAY03/CASE/01/index.html#case3)
4. [案例4：综合练习（自动化部署Web集群）](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/AUTOMATION/DAY03/CASE/01/index.html#case4)

**1 案例1：ansible应用案例**

**1.1 问题**

本案例要求掌握Ansible更多高级语法知识，具体要求如下：

* 熟悉firewalld和template模块的使用
* 熟悉error处理机制
* 熟悉handlers任务
* 熟悉when条件判断
* 熟悉block任务块
* 熟悉loop循环的使用方法

**1.2 步骤**

实现此案例需要按照如下步骤进行。

**步骤一：firewalld模块**

使用firewalld模块可以配置防火墙策略。

1. **[**root@control **~]**# vim **~**/ansible/firewall**.**yml
2. **---**
3. **-** hosts**:** test #hosts定义需要远程的主机
4. tasks**:** #tasks定义需要执行哪些任务
5. **-** name**:** install firewalld**.** #name为第一个任务定义描述信息
6. yum**:** #第一个任务调用yum模块安装软件
7. name**:** firewalld #需要安装的软件名称为firewalld
8. state**:** present #state等于present代表安装软件
9. **-** name**:** run firewalld**.** #定义第二个任务的描述信息
10. service**:** #第二个任务调用service模块启动服务
11. name**:** firewalld #启动的服务名称为firewalld
12. state**:** started #state等于started代表启动服务
13. enabled**:** yes #enabled等于yes是设置服务为开机自启动
14. **-** name**:** set firewalld rule**.** #第三个任务的描述信息
15. firewalld**:** #第三个任务调用firewalld模块设置防火墙规则
16. port**:** **80/**tcp #在防火墙规则中添加一个放行tcp，**80**端口的规则
17. permanent**:** yes #permaenent 是设置永久规则
18. immediate**:** yes #immediate 是让规则立刻生效
19. state**:** enabled #state等于enabled是添加防火墙规则
20. #最终：在默认zone中添加一条放行**80**端口的规则

**步骤二：template模块**

copy模块可以将一个文件拷贝给远程主机，但是如果希望每个拷贝的文件内容都不一样呢？如何给所有web主机拷贝index.html内容是各自的IP地址？

Ansible可以利用Jinja2模板引擎读取变量，之前在playbook中调用变量，也是Jinja2的功能，Jinja2模块的表达式包含在分隔符"{{ }}"内。

这里，我们给webserver主机拷贝首页，要求每个主机内容不同。

1. **[**root@control ansible**]**# mkdir **~**/ansible/template
2. **[**root@control ansible**]**# vim **~**/ansible/template**/**index**.**html
3. Welcome to **{{**ansible\_hostname**}}** on **{{** ansible\_eth0**.**ipv4**.**address **}}.**
4. #注意网卡名称根据实际情况填写，不可以完全照抄，不知道网卡名可以通过ip a s查询！
5. #**{{**ansible\_hostname**}}**和**{{** ansible\_eth0**.**ipv4**.**address **}}**是ansible自动的facts变量。

2）编写Playbook将网页模板文件拷贝到远程主机。

1. **[**root@control ansible**]**# vim **~**/ansible/template**.**yml
2. **---**
3. **-** hosts**:** webserver
4. tasks**:**
5. **-** name**:** use template copy index**.**html to webserver**.**
6. template**:**
7. src**:** **~**/ansible/template**/**index**.**html
8. dest**:** /tmp/index**.**html
9. #hosts定义需要远程的目标主机是谁；tasks定义需要执行的任务是什么
10. #**-** name定义任务的描述信息；任务需要调用的模块是template模块
11. #template模块需要两个参数，src指定需要拷贝的源文件，dest指定需要拷贝的目标位置
12. #src**:** **~**/ansible/template**/**index**.**html是上面创建的文件**,**文件中包含变量
13. #dest**:** /tmp/index**.**html拷贝到目标主机放在**/**tmp目录下

**步骤三：Ansible高级语法应用**

1）error错误处理

默认ansible在遇到error会立刻停止playbook，使用ignore\_errors可以忽略错误，继续后续的任务。

如果一个剧本里面有20个任务，执行到第3个时失败，则不再往下执行。

下面这个这个Playbook在执行时会意外中断。

1. **[**root@control ansible**]**# vim **~**/ansible/error**.**yml
2. **---**
3. **-** hosts**:** test
4. tasks**:**
5. **-** name**:** start a service that does not exist**.**
6. service**:**
7. name**:** hehe #注意：没有这个服务（启动一个不存在的服务）
8. state**:** started
9. **-** name**:** touch a file**.**
10. file**:**
11. path**:** /tmp/service**.**txt
12. state**:** touch

下面这个Playbook在执行时因为忽略了错误（针对某一个任务），不会被中断。

1. **[**root@control ansible**]**# vim **~**/ansible/error**.**yml
2. **---**
3. **-** hosts**:** test
4. tasks**:**
5. **-** name**:** start a service that does not exist**.**
6. service**:**
7. name**:** hehe
8. state**:** started
9. ignore\_errors**:** **true** #针对某一个任务忽略错误**(**ignore\_errors是关键词**)**
10. **-** name**:** touch a file**.**
11. file**:**
12. path**:** /tmp/service**.**txt
13. state**:** touch

下面这个Playbook在执行时因为忽略了错误，不会被中断。

1. **[**root@control ansible**]**# cat **~**/ansible/error**.**yml
2. **---**
3. **-** hosts**:** test
4. ignore\_errors**:** **true** #针对playbook全局忽略错误
5. tasks**:**
6. **-** name**:** start a service that does not exist**.**
7. service**:**
8. name**:** hehe
9. state**:** started
10. **-** name**:** touch a file**.**
11. file**:**
12. path**:** /tmp/service**.**txt
13. state**:** touch

2）handlers

在剧本中tasks用来定义任务（一定会执行），handlers也可以定义任务（不一定执行），handlers任务要想执行必须要被别人触发才能执行。

1. 实例草稿：
2. **---**
3. **-** hosts**:** test
4. tasks**:**
5. **-** 任务**1**
6. nofify**:**任务**5**
7. **-** 任务**2**
8. handlers**:**
9. **-** 任务**5**
10. **-** 任务**6**

可以通过handlers定义一组任务，仅当某个任务触发(notify)handlers时才执行相应的任务，如果有多个notify触发执行handlers任务，也仅执行一次。

仅当任务的执行状态为changed时handlers任务才执行，handlers任务在所有其他任务都执行后才执行。

下面编写一个通过nofify触发执行handlers任务的案例。

1. **[**root@control ansible**]**# vim **~**/ansible/handlers**.**yml
2. **---**
3. **-** hosts**:** test
4. tasks**:**
5. **-** name**:** create directory**.** #多次执行playbook该任务状态不再是changed
6. file**:** #调用file模块创建目录
7. path**:** /tmp/parents**/**subdir**/** #需要创建的具体目录名称
8. state**:** directory #state等于directory代表创建目录
9. notify**:** touch file #notify后面名称必须和handlers中的任务名称一致
10. handlers**:** #通过handlers再定义一组任务
11. **-** name**:** touch file #给任务写描述信息（任务的名字，名字可以任意）
12. file**:** #调用file模块创建文件
13. path**:** /tmp/parents**/**subdir**/new.**txt #需要创建的文件名
14. state**:** touch #state等于touch代表创建文件
15. #备注：仅当file模块执行成功，
16. #并且状态为changed时才会通过notify触发执行handlers下面的任务，
17. #所以多次执行该剧本时，handlers任务不会被重复执行**,**
18. #notity后面的名称必须和handlers下面name定义的任务名称一致（名称可以任意）。

3）when条件判断

when可以定义判断条件，条件为真时才执行某个任务。

常见条件操作符有：==、!=、>、>=、<、<=。

多个条件可以使用and(并且)或or（或者）分割，when表达式中调用变量不要使用{{ }}。

下面编写Playbook，远程主机剩余内存不足700M则关闭NetworkManager服务

1. **[**root@control ansible**]**# vim **~**/ansible/when\_1**.**yml
2. **---**
3. **-** hosts**:** test
4. tasks**:**
5. **-** name**:** check memory size**.**
6. service**:**
7. name**:** NetworkManager
8. state**:** stopped
9. when**:** ansible\_memfree\_mb **<** **700**
10. #被管理端主机剩余内存不足700M则关闭NetworkManager服务**(**也可以关闭别的不需要的服务**)**
11. #ansible\_memfree\_mb这个是ansible自带的facts变量**,**代表剩余内存的容量。

下面再编写一个Playbook，判断操作系统是RedHat8则创建测试文件。YAML的语法格式中>支持多行输入，但不保留换行符。

1. **[**root@control ansible**]**# vim **~**/ansible/when\_2**.**yml
2. **---**
3. **-** hosts**:** test
4. tasks**:**
5. **-** name**:** touch a file
6. file**:**
7. path**:** /tmp/when**.**txt
8. state**:** touch
9. when**:** **>**
10. ansible\_distribution **==** "RedHat"
11. and
12. ansible\_distribution\_major\_version **==** "8"
13. #判断操作系统是RedHat8则创建测试文件
14. #YAML的语法格式中**>**支持多行输入，但不保留换行符（计算机会认为实际是一行内容）
15. #ansible\_distribution和ansible\_distribution\_major\_version都是自带的facts变量
16. #可以使用setup模块查看这些变量

4）block任务块

如果我们需要当条件满足时执行N个任务,我们可以给N个任务后面都加when判断(但是很麻烦),此时可以使用block定义一个任务块,当条件满足时执行整个任务块.

任务块就是把一组任务合并为一个任务组，使用block语句可以将多个任务合并为一个任务组。

1. **[**root@control ansible**]**# vim **~**/ansible/block\_1**.**yml
2. **---**
3. **-** hosts**:** test
4. tasks**:**
5. **-** name**:** define a group of tasks**.**
6. block**:** #block是关键词，定义任务组
7. **-** name**:** install httpd #任务组中的第一个任务
8. yum**:** #调用yum模块安装httpd软件包
9. name**:** httpd
10. state**:** present
11. **-** name**:** start httpd #任务组中的第二个任务
12. service**:** #调用service模块启动httpd服务
13. name**:** httpd
14. state**:** started
15. when**:** ansible\_distribution **==** "RedHat" #仅当条件满足再执行任务组
16. #注意**:**when和block是对齐的**,**他们在一个级别**,**当条件满足时要执行的是任务组（不是某一个任务）
17. #判断条件是看远程的目标主机使用的Linux发行版本是否是RedHat**.**

对于block任务块，我们可以使用rescue语句定义在block任务执行失败时要执行的其他任务，还可以使用always语句定义无论block任务是否成功，都要执行的任务。

下面编写一个包含rescue和always的示例。

1. **[**root@control ansible**]**# vim **~**/ansible/block\_2**.**yml
2. **---**
3. **-** hosts**:** test
4. tasks**:**
5. **-** block**:**
6. **-** name**:** touch a file test1**.**txt
7. file**:**
8. name**:** /tmp/test1**.**txt #如果修改为/tmp/xyz**/**test1**.**txt就无法创建成功
9. state**:** touch
10. rescue**:**
11. **-** name**:** touch a file test2**.**txt
12. file**:**
13. name**:** /tmp/test2**.**txt
14. state**:** touch
15. always**:**
16. **-** name**:** touch a file test3**.**txt
17. file**:**
18. name**:** /tmp/test3**.**txt
19. state**:** touch
20. #默认在/tmp/目录下创建test1**.**txt会成功，所以不执行rescue**(**创建test2**.**txt**)**
21. #如果我们把block中的任务改为创建/tmp/xyz**/**test1**.**txt（因为没有xyz目录所以会失败**)**
22. #当block默认任务失败时就执行rescue任务**(**创建test2**.**txt**)**
23. #但是不管block任务是否成功都会执行always任务**(**创建test3**.**txt**)**

5）loop循环

相同模块需要反复被执行怎么处理？使用loop循环可以避免重复。

编写Playbook，循环创建目录。

1. **[**root@control ansible**]**# vim **~**/ansible/simple\_loop**.**yml
2. **---**
3. **-** hosts**:** test
4. tasks**:**
5. **-** name**:** mkdir multi directory**.**
6. file**:**
7. path**=**/tmp/**{{**item**}}** #注意，item是关键字，调用loop循环的值
8. state**=**directory
9. loop**:** #loop是关键词**,**定义循环的值**,**下面是具体的值
10. **-** School
11. **-** Legend
12. **-** Life
13. #最终在**/**tmp目录下创建三个子目录**.**file模块被反复执行了三次。
14. #mkdir **/**tmp**/**School**;** mkdir **/**tmp**/**Legend**;** mkdir **/**tmp**/**Life。

编写Playbook，循环创建用户并设置密码。

1. **[**root@control ansible**]**# vim **~**/ansible/complex\_loop**.**yml
2. **---**
3. **-** hosts**:** test
4. tasks**:**
5. **-** name**:** create multi user**.**
6. user**:**
7. name**:** "{{item.iname}}"
8. password**:** "{{item.ipass | password\_hash('sha512')}}"
9. loop**:**
10. **-** **{** iname**:** 'term'**,** ipass**:** '123456' **}**
11. **-** **{** iname**:** 'amy' **,** ipass**:** '654321' **}**
12. #loop循环第一次调用user模块创建用户**,**user模块创建用户会读取loop里面的第一个值**.**
13. #loop第一个值里面有两个子值**,**iname和ipass
14. #创建用户item**.**iname就是loop第一个值里面的iname**=**term
15. #修改密码item**.**ipass就是loop第一个值里面的ipass**=123456**
16. #loop循环第二次调用user模块创建用户**,**user模块创建用户会读取loop里面的第二个值**.**
17. #loop第二个值里面有两个子值**,**iname和ipass
18. #创建用户item**.**iname就是loop第二个值里面的iname**=**amy
19. #修改密码item**.**ipass就是loop第二个值里面的ipass**=654321**

**2 案例2：加密敏感数据**

**2.1 问题**

本案例要求，使用ansible-vault对敏感数据进行加密处理，具体要求如下：

* 使用ansible-vault管理敏感数据

**2.2 步骤**

实现此案例需要按照如下步骤进行。

**步骤一：使用ansible-vault处理敏感数据**

1）加密敏感数据。

encrypt（加密）、decrypt（解密）、view（查看）。

1. **[**root@control ansible**]**# echo **123456** **>** data**.**txt #新建测试文件
2. **[**root@control ansible**]**# ansible**-**vault encrypt data**.**txt #加密文件
3. **[**root@control ansible**]**# cat data**.**txt
4. **[**root@control ansible**]**# ansible**-**vault view data**.**txt #查看加密文件

2）修改密码（rekey）

1. **[**root@control ansible**]**# ansible**-**vault rekey data**.**txt #修改密码
2. Vault password**:** **<**旧密码**>**
3. New Vault password**:** **<**新密码**>**
4. Confirm New Vault password**:<**确认新密码**>**

3）解密文件

1. **[**root@control ansible**]**# ansible**-**vault decrypt data**.**txt #解密文件
2. **[**root@control ansible**]**# cat data**.**txt

4）使用密码文件

加密、解密每次都输入密码很麻烦，可以将密码写入文件。

1. **[**root@control ansible**]**# echo "I'm secret data" **>** data**.**txt #需要加密的敏感数据
2. **[**root@control ansible**]**# echo **123456** **>** pass**.**txt #加密的密码
3. **[**root@control ansible**]**# ansible**-**vault encrypt **--**vault**-**id**=**pass**.**txt data**.**txt
4. **[**root@control ansible**]**# cat data**.**txt
5. **[**root@control ansible**]**# ansible**-**vault decrypt **--**vault**-**id**=**pass**.**txt data**.**txt
6. **[**root@control ansible**]**# cat data**.**txt

**3 案例3：Ansible Roles**

**3.1 问题**

学习Ansible Roles基本概念，掌握Roles应用案例，具体要求如下：

* 自定义Ansible Role
* 编写playbook调用role
* 使用ansible-galaxy管理Roles

**3.2 方案**

在实际生产环境中，为了实现不同的功能，我们会编写大量的playbook文件。而且，每个playbook还可能会调用其他文件（如变量文件），对于海量的、无规律的文件，管理起来非常痛苦！

Ansible从1.2版本开始支持Role（角色），Role（角色）是管理ansible文件的一种规范（目录结构），Role（角色）会按照标准的规范，自动到特定的目录和文件中读取数据。

如果我们创建了一个名称为user.example的Role（角色），则其标准的目录结构如下图-1所示。

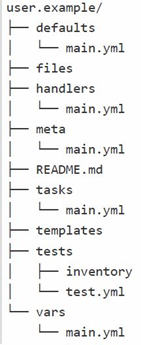


图-1

Roles目录结构中主要文件的作用是什么呢？

* defualts/main.yml：定义变量的缺省值，优先级较低
* files目录：存储静态文件的目录，如tar包、音乐、视频等
* handlers/main.yml:定义handlers
* meta/main.yml:写作者、版本等描述信息
* README.md:整个角色(role)的描述信息
* tasks/main.yml:定义任务的地方
* templates目录：存放动态数据文件的地方（文件中包含了变量的模板文件）
* vars/main.yml:定义变量，优先级高

**步骤一：Role应用案例**

1）创建Roles

下面这个案例目的：编写一个包含变量的模板文件，编写任务调用template模块，将模板文件拷贝给被管理端主机。

ansible-galaxy命令可以创建、管理自己的roles。

1. **[**root@control ansible**]**# mkdir **~**/ansible/roles
2. **[**root@control ansible**]**# ansible**-**galaxy init **~**/ansible/roles**/**issue
3. #创建一个Role，该Role的目的是拷贝自己新建的一个模板文件到远程主机的/etc/issue
4. **[**root@control ansible**]**# tree **~**/ansible/roles**/**issue**/**
5. #查看目录结构，如果没有tree命令则需要使用yum安装该软件

2）修改Role文件

定义名称为myfile.txt的模板文件（该文件包含变量,因此必须放置templates目录）

1. **[**root@control ansible**]**# vim **~**/ansible/roles**/**issue**/**templates**/**myfile**.**txt
2. This is the system **{{**ansible\_hostname**}}**
3. Today's date is:{{ansible\_date\_time.date}}
4. Contact to {{ admin }}

自定义变量文件（前面调用了admin这个变量，这里需要定义admin变量并赋值）

1. **[**root@control ansible**]**# vim **~**/ansible/roles**/**issue**/**vars**/**main**.**yml
2. **---**
3. # vars file **for** /root/ansible**/**roles**/**issue
4. admin**:** yoyo@tedu**.**cn
5. #变量名为admin，变量的值为yoyo@tedu**.**cn

文件准备好了，计算机不会自动将文件拷贝给被管理端主机！需要编写任务调用模块实现拷贝的功能。

修改任务文件，任务文件中不需要tasks关键词，Role的各个文件之间相互调用不需要写文件的路径。

1. **[**root@control ansible**]**# vim **~**/ansible/roles**/**issue**/**tasks**/**main**.**yml
2. **---**
3. # tasks file **for** /root/ansible**/**roles**/**issue
4. **-** name**:** delever issue file
5. template**:**
6. src**:** myfile**.**txt
7. dest**:** /etc/issue
8. #调用template模块将myfile**.**txt文件拷贝给被管理端主机**.**

3）在Playbook中调用Role

Role创建好了，role不会自己运行，需要编写一个剧本调用上面的role。

编写playbook剧本文件,通过roles关键词调用role。

1. **[**root@control ansible**]**# vim **~**/ansible/issue**.**yml
2. **---**
3. **-** hosts**:** test
4. roles**:**
5. **-** issue
6. # **-** role2 #支持加载多个role

修改ansible.cfg配置文件，定义roles目录。

1. **[**root@control ansible**]**# vim **~**/ansible/ansible**.**cfg
2. **[**defaults**]**
3. inventory **=** **./**inventory
4. roles\_path **=** **./**roles #指定到哪个目录下找role
5. remote\_user **=** alice
6. **[**privilege\_escalation**]**
7. become**=**True
8. become\_method**=**sudo
9. become\_user**=**root
10. become\_ask\_pass**=**False

**步骤二：ansible-galaxy命令**

公共Roles仓库(https://galaxy.ansible.com)管理。

1. **[**root@control ansible**]**# ansible**-**galaxy search 'httpd'
2. #联网搜索roles
3. **[**root@control ansible**]**# ansible**-**galaxy info acandid**.**httpd
4. #查看roles基本信息
5. **[**root@control ansible**]**# ansible**-**galaxy install acandid**.**httpd **-**p **~**/ansible/roles**/**
6. #下载roles到特定的目录，**-**p可以指定下载到哪个目录

使用ansible-galaxy install可以直接下载Role，也可以编写requirements.yml文件下载Role。

1. **[**root@control ansible**]**# vim **~**/ansible/roles**/**requirements**.**yml
2. #格式一：可以直接从Ansible Galaxy官网下载
3. **-** src**:** acandid**.**httpd
4. #格式二：可以从某个git服务器下载
5. **-** src**:** http**:**//gitlab.com/xxx/xxx.git
6. scm**:** git
7. version**:** 56e00a54
8. name**:** nginx**-**acme
9. #格式三：可以指定位置下载tar包，支持http、https、file
10. **-** src**:** http**:**//example.com/myrole.tar
11. name**:** myrole
12. **[**root@control ansible**]**# ansible**-**galaxy install **\**
13. **-**r **~**/ansible/roles**/**requirements**.**yml **\**
14. **-**p roles
15. # **-**r后面跟文件名**,**该文件中包含了需要下载哪些role以及他们的链接位置
16. # **-**p 指定将下载的role保存到哪个目录

**4 案例4：综合练习（自动化部署Web集群）**

**4.1 问题**

晚自习课外综合练习题，创建一个名为cluster的role，完成一个综合项目，具体要求如下：

* 创建Role，通过Role完成项目
* 部署Nginx调度器
* 部署2台http服务器

**4.2 方案**

晚自习综合练习题实验所需主机清单如表-1所示。

表-1



**步骤一：部署两台后端LNP服务器（没有mariadb）**

1）创建role角色

1. **[**root@control ansible**]**# ansible**-**galaxy init **~**/ansible/roles**/**http

2）修改role配置文件，准备2台http网站的素材

安装httpd，拷贝一个网页文件。

1. **[**root@control ansible**]**# vim roles**/**http**/**tasks**/**main**.**yml
2. **---**
3. **-** name**:** install httpd
4. yum**:**
5. name**:** httpd
6. state**:** present
7. **-** name**:** create index**.**html
8. copy**:**
9. content**:** "{{ansible\_hostname}}"
10. dest**:** /var/www**/**html**/**index**.**html
11. **-** name**:** set firewalld
12. firewalld**:**
13. service**:** http
14. state**:** enabled
15. permanent**:** yes
16. immediate**:** yes
17. **-** name**:** start httpd
18. service**:**
19. name**:** httpd
20. state**:** started
21. enabled**:** yes
22. #文件中包含多个任务，每个任务可以设置一个name名字（也可以没有name）
23. #第一个任务调用yum模块安装httpd软件包
24. #第二个任务调用copy模块创建一个新的网页文件**(**index**.**html**)**
25. #调用copy模块时可以在没有源文件的情况下，直接使用content指定文件的内容
26. #将该内容直接拷贝到被管理主机的某个文件中**(**/var/www**/**html**/**index**.**html**)**
27. #第三个任务调用firewalld模块，设置防火墙规则，允许访问http服务
28. #第四个任务调用service模块将httpd服务启动，并设置开机自启。

3）编写Playbook调用role，并执行Playbook。

1. **[**root@control ansible**]**# vim web**.**yml
2. **---**
3. **-** hosts**:** webserver
4. roles**:**
5. **-** http
6. **[**root@control ansible**]**# ansible**-**playbook web**.**yml

**步骤二：部署nginx代理服务器**

1）创建role角色

1. **[**root@control ansible**]**# ansible**-**galaxy init **~**/ansible/roles**/**proxy

2）准备代理服务器需要的素材

拷贝Nginx源码包，编写一个源码编译安装nginx的shell脚本。

1. **[**root@control ansible**]**# cp lnmp\_soft**/**nginx**-1.17.6.**tar**.**gz **\**
2. **~**/ansible/roles**/**proxy**/**files**/**
3. **[**root@control ansible**]**# vim **~**/ansible/roles**/**proxy**/**files**/**nginx\_install**.**sh
4. #**!**/bin/bash
5. yum **-**y install gcc pcre**-**devel openssl**-**devel make
6. cd **/**tmp
7. tar **-**xf **/**tmp**/**nginx**-1.17.6.**tar**.**gz
8. cd nginx**-1.17.6**
9. **./**configure **--with-**http\_ssl\_module
10. make
11. make install

新建一个Nginx代理服务器的配置文件模板。

1. **[**root@control ansible**]**# vim **~**/ansible/roles**/**proxy**/**files**/**nginx**.**conf
2. worker\_processes **2;**
3. #error\_log logs**/**error**.**log**;**
4. events **{**
5. worker\_connections **65535;**
6. **}**
7. http **{**
8. include mime**.**types**;**
9. default\_type application**/**octet**-**stream**;**
10. sendfile on**;**
11. tcp\_nopush on**;**
12. keepalive\_timeout **65;**
13. #gzip on**;**
14. upstream webs **{**
15. server **192.168.4.13;**
16. server **192.168.4.14;**
17. **}**
18. server **{**
19. listen **80;**
20. server\_name localhost**;**
21. location **/** **{**
22. proxy\_pass http**:**//webs;
23. root html**;**
24. index index**.**html index**.**htm**;**
25. **}**
26. error\_page **404** **/404.**html**;**
27. error\_page **500** **502** **503** **504** **/**50x**.**html**;**
28. location **=** **/**50x**.**html **{**
29. root html**;**
30. **}**
31. **}**
32. **}**

3）修改role配置文件。

1. **[**root@control ansible**]**# vim roles**/**proxy**/**tasks**/**main**.**yml
2. **---**
3. **-** name**:** copy nginx**-1.17.6.**tar**.**gz to proxy**.**
4. copy**:**
5. src**:** nginx**-1.17.6.**tar**.**gz
6. dest**:** /tmp/
7. #拷贝源码包软件
8. **-** name**:** install nginx through shell script**.**
9. script**:** nginx\_install**.**sh
10. args**:**
11. creates**:** /usr/local**/**nginx**/**sbin**/**nginx
12. #执行源码编译安装脚本，如果已经安装nginx，则不再执行安装脚本**.**
13. #args是关键词，设置script模块的参数，通过creates参数做判断，creates也是关键词
14. #creates后面跟文件名，如果creates判断文件存在的话就不再执行script模块对应的命令。
15. **-** name**:** copy nginx**.**conf to destination host**.**
16. copy**:**
17. src**:** nginx**.**conf
18. dest**:** /usr/local**/**nginx**/**conf**/**nginx**.**conf
19. **-** name**:** run nginx service**.**
20. shell**:** /usr/local**/**nginx**/**sbin**/**nginx
21. args**:**
22. creates**:** /usr/local**/**nginx**/**logs**/**nginx**.**pid
23. #nginx**.**pid存在，说明nginx已经启动。如果该文件存在，则不再启动nginx。

4）编写Playbook调用role,并执行Playbook。

1. **[**root@control ansible**]**# vim proxy**.**yml
2. **---**
3. **-** hosts**:** proxy
4. roles**:**
5. **-** proxy
6. **[**root@control ansible**]**# ansible**-**playbook proxy**.**yml

