华为云微认证系列

Ansible自动化部署LNMP

实验指导手册

版本:1.0

华为云计算技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为云计算技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址： http://edu.huaweicloud.com

目录

背景介绍

Cloud2.0 时代，企业和个人开发者，更加关注应用上云，期待应用的高效迁移部署和快速迭代开发。面对市场高速变化且竞争激烈，产品需要根据市场变化不断更新迭代和升级；缺乏统一的持续交付工具，来确保产品随时可推向市场；缺乏工具保证客户快速反馈闭环。对此 Devops 自动化运维已日趋普及，利用 Devops 来提升资源利用率、优化算力、简化运维已成了企业云上开发与运维的共识。在电商、游戏、互联网等众多行业，Devops 已经成为趋势。

华为云 DevCloud 集成业界先进理念，是可操作可落地的端到端一站式开发方法论和工具链。

实验通过演示在弹性云服务器及 DevCloud 上运行 Ansible 部署 LNMP 环境，突出 DevCloud 软件开发平台在代码集成和快速部署上的极大优势，学习者可通过实验 1 充分了解 Ansible playbook 的编写方法，同时结合实验 2 体验 DevCloud 快速部署、生产的便捷。

实验目的

本实验指导包含两个实验任务，通过本实验，您将能够：

- 掌握基于弹性云服务器部署 Ansible 的基本流程
- 熟悉 Ansible playbook 的编写方法
- 利用 DevCloud 集成脚本代码和快速部署生产任务

注意事项

实验资源一旦购买就开始计费，请合理安排时间进行实验，并注意以下几点：

- 本实验预计 3 小时完成，实验结束后请一定释放资源，并确认资源彻底删除；
- 若实验中途离开或中断，建议释放实验资源，否则将会按照购买的资源继续计费；
- 资源释放步骤请根据实验指导进行。

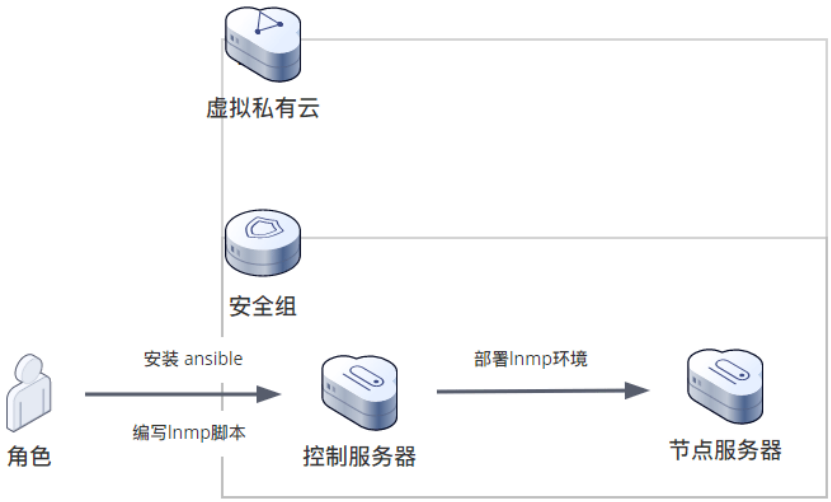
1 云主机环境 Ansible 部署 LNMP

1.1 实验介绍

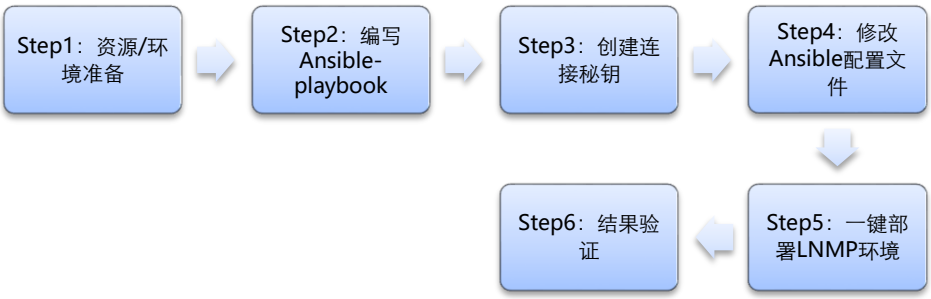
实验需两台华为弹性云服务器，一台管理节点，用于安装 Ansible 工具，并执行自动化脚本；一台被控节点服务器，用于部署 LNMP(Linux+Nginx+Mysql+Php)环境。被控节点服务器数量可根据实际情况添加，实验仅演示部署一台。

1.2 实验架构

在同一虚拟私有云环境下的两台弹性云服务器。其一为控制服务器，用户在该节点上安装自动化运维工具 Ansible，并编写运维脚本，在本实验中为 LNMP 脚本；通过控制服务器向远程节点服务器部署 LNMP，可为多台，本实验中为一台。



1.3 实验流程



1.4 资源准备

在部署 LNMP 环境之前，需提前配置好以下资源，实验及资源准备均在“**华北-北京四**”区域下进行：

- 使用 Chrome（86 以上）或版本相近的 Edge 浏览器
- 已成功注册华为云官网并通过认证。
- 创建虚拟私有云 VPC 及其子网，并创建安全组添加安全组规则。
- 购买两台按需计费弹性云服务器

1.4.1 实验环境准备

步骤 1 登录华为云

网站链接：https://auth.huaweicloud.com/authui/login.html#/login

步骤 2 进入控制台，选择“**华北-北京四**”区域

步骤 3 创建虚拟私有云 VPC

在控制台界面（上图），点击虚拟私有云 VPC，进入新页面。

点击创建虚拟私有云，在新页面内，仅修改 VPC 名称修改为：vpc-lnmp，其他选择默认，将自动生成子网，点击立即创建（如下图）。

步骤 4 新建安全组，并添加规则

在网络控制台中点击：访问控制-安全组，进入安全组界面。

点击创建安全组。

在弹出的窗口内，填入如下信息。

实验中需用到 mysql 数据库，因此需手动添加端口 3306；php，端口 9000

点击下图中的配置规则，进入手动配置界面。

点击添加规则

在弹出的页面输入图示信息，并确认。

同理，添加 php 需要使用的端口 9000。

步骤 5 购买弹性云服务器

回到控制台，点击弹性云服务器

进入新页面后，点击购买弹性云服务器

弹性云主机的具体参数如下表：

计费模式：按需计费

区域：北京四

可用区：随机分配

CPU 架构：x86 计算

规格：通用计算增强型 | c6s.2xlarge.2 | 8vCPUs | 16 GiB

镜像：公共镜像 centos 7.6

系统盘：通用性 SSD 40GB

网络：vpc-lnmp

安全组：sg-lnmp

弹性公网 IP：全动态 BGP/按流量计费/带宽大小 100M

云备份：暂不购买

在给弹性云服务器命名的页面，如下，输入 ecs-ansible-lnmp，购买成功后，系统将自动添加后缀 0001 和 0002。

在确认配置界面，重点确认购买费用是否一致。实验中用到的资源均为按需付费，实验结束后，删除资源，即可不再收费。

1.5 编写 Ansible-playbook

1.5.1 登录华为弹性云服务器

步骤 1 使用 Cloudshell 登录弹性云服务器

在弹性云服务器控制台，选择名称后缀为 001 的弹性云服务器，点击远程登录，进入 Cloudshell 登录界面。

在弹出的新标签页中，点击密码栏，输入购买云服务器时设置的密码。

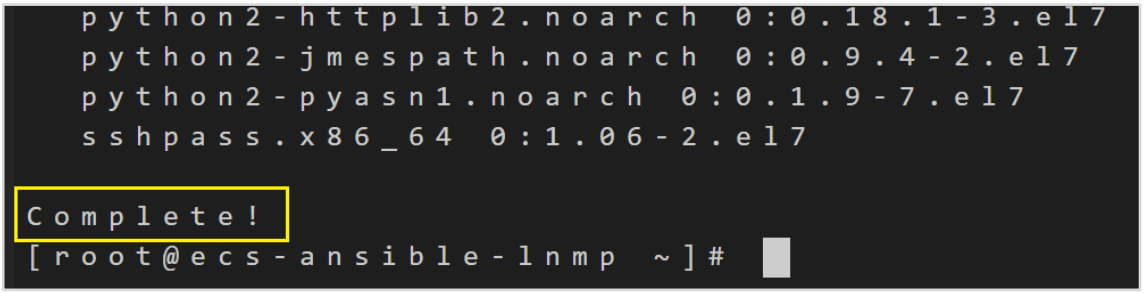
显示如下界面，即说明成功登录。

步骤 2 安装 Ansible 工具

在 Cloudshell 中输入命令，安装 Ansible.

```
yum -y install ansible
```

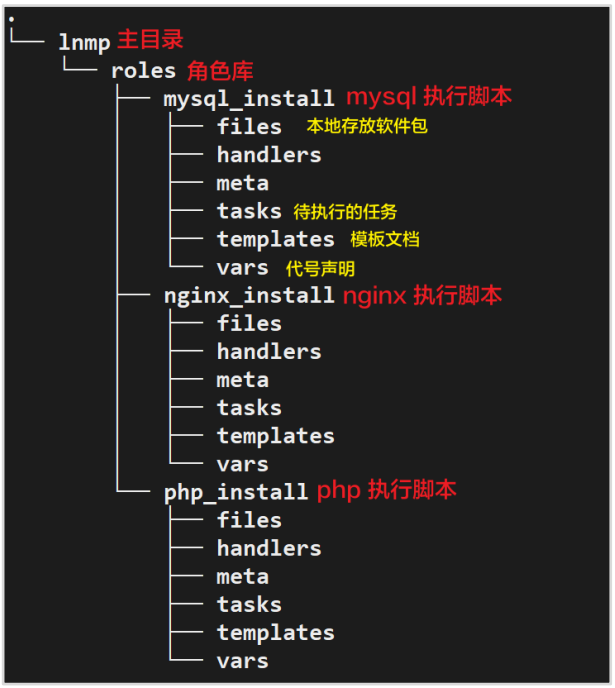
提示：Complete！如下图，说明安装成功。



1.5.2 搭建 playbook 框架

步骤 1 整体框架说明

Playbook 框架十分重要，能决定 playbook 在执行过程的逻辑完整性，细粒度的拆分和组合，能帮助运维人员精细化部署想要的目标环境。实验中 LNMP 环境的 playbook 框架如下图所示。



步骤 2 创建框架目录

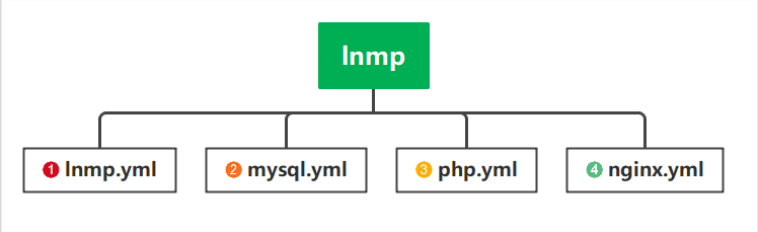
在命令行中输入命令，创建如上图 playbook 框架。

```
mkdir -
p lnmp/roles/{mysql_install,nginx_install,php_install}/{files,handlers,meta,tas
ks,templates,vars}
cd lnmp
yum -y install tree
tree .
```

回显的目录结构如上图所示。

1.5.3 编写具体脚本

步骤 1 脚本编写逻辑



主目录中包含上述 4 个脚本，其中

1. lnmp.yml 安装 LNMP 环境的总脚本
2. mysql.yml 仅安装 mysql 的分脚本
3. php.yml 仅安装 php 的分脚本

4. nginx.yml 仅安装 nginx 的分脚本

先写分脚本，总脚本是分脚本的集成。

roles 目录下是每个分脚本的框架，以 mysql 安装为例来说明：

- files 目录用于存放本地安装包，本次实验采取远程下载安装包的方式，故 files 目录为空；
- vars 目录下，有个 main.yml 文件，该文件的作用在于声明脚本流水线工作过程中需要用到的一些共性对象，比如说软件包的下载地址，具体安装目录等，详见后续脚本；
- tasks 目录是脚本具体任务的集合，本实验中将完成软件包的下载和安装，分别需要 download.yml 和 install.yml 两个执行脚本，同时还需要另一个 main.yml 脚本来规划任务的执行顺序；
- templates 目录用于存放部署过程需要用到的模板文件，方便自动化实施，如在 mysql 安装过程中需用到修改 root 用户密码，可提前写好修改密码的代码模板。

步骤 2 编写 mysql 分脚本

- 入口文件 mysql.yml

在命令行中输入：

```
vim mysql.yml
```

进入 vim 编辑器中，vim 在粘贴内容的时候，如果遇到以#开始的注释行，会自动将后续的所有行进行注释。这在本实验中并不方便，英文模式下输入：set paste，进入粘贴模式。点击 i 进入插入模式，拷贝如下代码：

为方便拷贝代码行，课程下载专区提供了 Word 版本命令行手册，请下载命令行手册拷贝实验。

```
# 用于批量安装 Mysql
---
- hosts: all
  remote_user: root
  gather_facts: True

  roles:
    - mysql_install
```

保存并退出 vim 编辑器，按 Esc 后，英文模式下 shift+:，输入 wq 退出。

- 变量声明 main.yml

在命令行中输入：

```
vim roles/mysql_install/vars/main.yml
```

进入 vim 编辑器中，点击 i 进入插入模式，拷贝如下代码：

```
# 定义安装的 mysql 的参数
```

```
MYSQL_VER: 5.7.33
MYSQL_VER_MAIN: "{{ MYSQL_VER.split('.')[0] }}.{{ MYSQL_VER.split('.')[1] }}"
DOWNLOAD_URL: https://weirenzhengnew.obs.cn-north-
4.myhuaweicloud.com/Ansible/mysql-5.7.33-linux-glibc2.12-x86_64.tar.gz
MYSQL_USER: mysql
MYSQL_PORT: 3306
MYSQL_PASSWD: Huawei@123
SOURCE_DIR: /software
BASE_DIR: /usr/local/mysql
DATA_DIR: /data/mysql
```

保存并退出 vim 编辑器，按 Esc 后，英文模式下 shift+:，输入 wq 退出。完成 main.yml 的编写

- 环境准备 prepare.yml

在命令行中输入：

```
vim roles/mysql_install/tasks/prepare.yml
```

进入 vim 编辑器中，点击 i 进入插入模式，拷贝如下代码：

```
#- name: 关闭 firewalld    # 云主机的防火墙一般默认都是关闭的
#  service: name=firewalld state=stopped enabled=no

#- name: 临时关闭 selinux    # 云主机一般默认关闭
#  shell: "setenforce 0"
#  failed_when: false

#- name: 永久关闭 selinux
#  lineinfile:
#    dest: /etc/selinux/config
#   regexp: "^SELINUX="
#    line: "SELINUX=disabled"

- name: 添加 EPEL 仓库      # 为"红帽系"的操作系统提供额外的软件包
  yum: name=epel-release state=latest

- name: 安装常用软件包
  yum:
    name:
      - vim
      - lrzsz
      - net-tools
      - wget
      - curl
      - bash-completion
      - rsync
      - gcc
      - unzip
```

```
      - git
      - perl-Data-Dumper
      - libaio-devel
      - autoconf
      - cmake
      - openssl
      - openssl-devel
      - pcre
      - pcre-devel
      - zlib
      - zlib-devel
      - gd-devel
      - libxml2-devel
      - bzip2-devel
      - gnutls-devel
      - ncurses-devel
      - bison
      - bison-devel
      - openldap
      - openldap-devel
      - libcurl-devel
      - libevent
      - libevent-devel
      - expat-devel
      - numactl
      - tree
      - unzip
    state: latest

- name: 更新系统
  shell: "yum update -y"
  args:
    warn: False
```

保存并退出 vim 编辑器，按 Esc 后，英文模式下 shift+:，输入 wq 退出。完成 prepare.yml 的编写

- 软件包下载 download.yml

在命令行中输入：

```
vim roles/mysql_install/tasks/download.yml
```

进入 vim 编辑器中，点击 i 进入插入模式，拷贝如下代码：

```
- name: 创建 mysql 用户组
  group: name={{ MYSQL_USER }}  state=present

- name: 创建 mysql 用户
```

```
  user: name={{ MYSQL_USER }}  group={{ MYSQL_USER }}  state=present create_hom
e=False shell=/sbin/nologin

- name: 创建所需目录
  file: name={{ item }} state=directory mode=0755 recurse=yes # 这里的 0755 指的
是：文件所有者对文件具有读、写和执行权限；组用户和其他用户对文件需有读和执行权限
  with_items:
  - "{{ SOURCE_DIR }}"
  - "{{ DATA_DIR }}"

- name: 更改目录属主属组
  file: name={{ DATA_DIR }} owner={{ MYSQL_USER }} group={{ MYSQL_USER }}

#当前主机下没有 mysql 包
- name: 下载 mysql 包
  get_url: url={{ DOWNLOAD_URL }} dest={{ SOURCE_DIR }} owner={{ MYSQL_USER }}
group={{ MYSQL_USER }}

#当前主机 file 目录下已有 mysql 包
#- name: 拷贝现有 mysql 包到所有主机
#  copy: src=mysql-{{ MYSQL_VER }}-linux-glibc2.12-
x86_64.tar.gz dest={{ SOURCE_DIR }} owner={{ MYSQL_USER }} group={{ MYSQL_USER
}}   # src 目录是主机的 file 目录

- name: 解压 mysql 包
  unarchive: src={{ SOURCE_DIR }}/mysql-{{ MYSQL_VER }}-linux-glibc2.12-
x86_64.tar.gz dest=/usr/local copy=no owner={{ MYSQL_USER }} group={{ MYSQL_USE
R }}  # copy=no，该参数可在远程机上解压

- name: 目录重命名
  shell: "mv /usr/local/mysql-{{ MYSQL_VER }}-linux-glibc2.12-
x86_64/ {{ BASE_DIR }} && chown -
R {{ MYSQL_USER }}:{{ MYSQL_USER }} {{ BASE_DIR }}"   # 留意 mv 指令还有移动文件位
置的功能

#复制 mysql 配置文件
- name: 拷贝 mysql 配置文件
  template: src=my.cnf dest=/etc/my.cnf owner=root group=root

#复制 mysql 服务文件
- name: 拷贝 mysql 服务文件
  template: src=mysqld.service dest=/usr/lib/systemd/system/mysqld.service owne
r=root group=root

#复制更改密码脚本
```

```
- name: 拷贝更改密码脚本
  template: src=change_passwd.sh dest={{ SOURCE_DIR }} owner=root group=root

- name: 创建日志目录
  file: name={{ item }} state=directory owner={{ MYSQL_USER }} group={{ MYSQL_U
SER }} mode=0755 recurse=yes
  with_items:
  - "/var/log/mysql/"
  - "/var/run/mysqld/"
  - "{{ BASE_DIR }}/tmp"
  - "{{ BASE_DIR }}/log"
  - "/etc/init.d/mysql/"

- name: 创建错误日志文件
  file:  name={{ item }} state=touch owner={{ MYSQL_USER }} group={{ MYSQL_USER
 }} mode=0644 # 文件格式新建是 touch
  with_items:
  - "{{ BASE_DIR }}/log/error.log"
```

保存并退出 vim 编辑器，完成 download.yml 的编写

- mysql 初始化安装 install.yml

在命令行中输入：

```
vim roles/mysql_install/tasks/install.yml
```

进入 vim 编辑器中，点击 i 进入插入模式，拷贝如下代码：

```
# 初始化安装 mysql
- name: mysql 初始化
  shell: "{{ BASE_DIR }}/bin/mysqld --initialize-insecure --
user={{ MYSQL_USER }} --basedir={{ BASE_DIR }}  --datadir={{ DATA_DIR }}"

- name: 拷贝启动脚本到/etc 下
# copy: src={{ BASE_DIR }}/support-
files/mysql.server dest=/etc/init.d/mysql  # copy 命令只能从管理主机上复制对象，自节
点上不可复制，所以使用 cp 命令
  shell: "cp {{ BASE_DIR }}/support-files/mysql.server /etc/init.d/mysql"

- name: 修改启动脚本_1
  lineinfile:
    dest: /etc/init.d/mysql/mysql.server
    regexp: "^basedir="
    insertbefore: "^# Default value, in seconds, afterwhich the script should t
imeout waiting"
    line: "basedir={{ BASE_DIR }}"

- name: 修改启动脚本_2
```

```
  lineinfile:
    dest: /etc/init.d/mysql/mysql.server
    regexp: "^datadir="
    insertbefore: "^# Default value, in seconds, afterwhich the script should t
imeout waiting"
    line: "datadir={{ DATA_DIR }}"

- name: 修改启动脚本_3
  file: dest=/etc/init.d/mysql/mysql.server state=file mode=0755

- name: 配置环境变量
  shell: " if [ `grep {{ BASE_DIR }}/bin /etc/profile |wc -l` -
eq 0 ]; then echo export PATH=$PATH:{{ BASE_DIR }}/bin >> /etc/profile && sourc
e /etc/profile; else source /etc/profile; fi"

- name: 启动 mysql 并开机启动
  shell: "systemctl daemon-
reload && systemctl enable mysqld && systemctl start mysqld"

- name: 设置数据库 root 密码    # 这一步的目的在于登录 mysql 需要重新设置 root 密码
  shell: "bash {{ SOURCE_DIR }}/change_passwd.sh"
```

保存并退出 vim 编辑器，完成 install.yml 的编写

- 引用文件 main.yml

在命令行中输入：

```
vim roles/mysql_install/tasks/main.yml
```

进入 vim 编辑器中，点击 i 进入插入模式，拷贝如下代码：

```
# 引用 prepare、download、install 模块
- include: prepare.yml
- include: download.yml
- include: install.yml
```

保存并退出 vim 编辑器，按 Esc 后，英文模式下 shift+:，输入 wq 退出。完成 main.yml 的编写

- mysql 配置文件-1 my.cnf

在命令行中输入：

```
vim roles/mysql_install/templates/my.cnf
```

进入 vim 编辑器中，点击 i 进入插入模式，拷贝如下代码：

```
[client]
port    = {{ MYSQL_PORT }}    # 主机 3306 端口
socket = {{ BASE_DIR }}/tmp/mysql.sock

[mysql]    # 默认编码格式
```

```
default-character-set=utf8

[mysqld]    # mysql 数据库的默认属性
default-storage-engine=INNODB
character_set_server=utf8
explicit_defaults_for_timestamp
basedir={{ BASE_DIR }}
datadir={{ DATA_DIR }}
socket={{ BASE_DIR }}/tmp/mysql.sock
log_error = {{ BASE_DIR }}/log/error.log
sql_mode=STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY
_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION
```

保存并退出 vim 编辑器，按 Esc 后，英文模式下 shift+:，输入 wq 退出。完成 my.cnf 的编写

- mysql 配置文件-2 mysqld.service

在命令行中输入：

```
vim roles/mysql_install/templates/mysqld.service
```

进入 vim 编辑器中，点击 i 进入插入模式，拷贝如下代码：

```
[Unit]
Description=MySQL Server
After=network.target
After=syslog.target

[Install]
WantedBy=multi-user.target

[Service]
User=mysql
Group=mysql
ExecStart={{ BASE_DIR }}/bin/mysqld --defaults-file=/etc/my.cnf

#连接数限制
LimitNOFILE=65535
LimitNPROC=65535

#Restart 配置可以在进程被 kill 掉之后，让 systemctl 产生新的进程，避免服务挂掉
#Restart=always
PrivateTmp=false
```

保存并退出 vim 编辑器，按 Esc 后，英文模式下 shift+:，输入 wq 退出。完成 mysqld.service 的编写

- 更改数据库 root 密码脚本 change_passwd.sh

在命令行中输入：

```
vim roles/mysql_install/templates/change_passwd.sh
```

进入 vim 编辑器中，点击 i 进入插入模式，拷贝如下代码：

```
# 该脚本用于更改数据库 root 密码
passwd={{ MYSQL_PASSWD }}
n=`grep "{{ BASE_DIR }}/bin" /etc/profile |wc -l`

if [ $n -eq 0 ]
then
        echo "export PATH=$PATH:{{ BASE_DIR }}/bin" >> /etc/profile
        source /etc/profile
else
        source /etc/profile
fi

{{ BASE_DIR }}/bin/mysql -uroot -D mysql -
e "UPDATE user SET authentication_string=PASSWORD('$passwd') WHERE user='root';
"

{{ BASE_DIR }}/bin/mysql -uroot -e "FLUSH PRIVILEGES;"

{{ BASE_DIR }}/bin/mysql -uroot -p$passwd -
e "grant all privileges on *.* to root@'%'  identified by '$passwd';"
```

保存并退出 vim 编辑器，按 Esc 后，英文模式下 shift+:，输入 wq 退出。完成 change_passwd.sh 的编写

步骤 3 编写 php 分脚本

- 入口文件 php.yml

在命令行中输入：

```
vim php.yml
```

进入 vim 编辑器中，点击 i 进入插入模式，拷贝如下代码：

```
# 用于批量安装 PHP
---
- hosts: all
  remote_user: Huawei
  gather_facts: True

  roles:
    - php_install
```

保存并退出 vim 编辑器，按 Esc 后，英文模式下 shift+:，输入 wq 退出。完成 php.yml 的编写

- 变量声明 main.yml

在命令行中输入：

```
vim roles/php_install/vars/main.yml
```

进入 vim 编辑器中，点击 i 进入插入模式，拷贝如下代码：

```
# 所安装的 PHP 版本信息
PHP_VER: 7.4.26
DOWNLOAD_URL: https://weirenzhengnew.obs.cn-north-
4.myhuaweicloud.com/Ansible/php-7.4.26.tar.gz
PHP_USER: php-fpm
PHP_PORT: 9000
SOURCE_DIR: /software
PHP_DIR: /usr/local/php7
MYSQL_DIR: /usr/local/mysql
```

保存并退出 vim 编辑器，按 Esc 后，英文模式下 shift+:，输入 wq 退出。完成 main.yml 的编写

- php 下载 download.yml

在命令行中输入：

```
vim roles/php_install/tasks/download.yml
```

进入 vim 编辑器中，点击 i 进入插入模式，拷贝如下代码：

```
- name: 创建 php 用户组
  group: name={{ PHP_USER }}  state=present

- name: 创建 php 用户
  user: name={{ PHP_USER }}  group={{ PHP_USER }}  state=present create_home=Fa
lse shell=/sbin/nologin

- name: 创建 software 目录
  file: name={{ SOURCE_DIR }} state=directory mode=0755 recurse=yes

#当前主机下没有 libmcrypt 依赖包

#- name: 下载依赖包 libmcrypt
#  get_url: url=http://nchc.dl.sourceforge.net/project/mcrypt/Libmcrypt/2.5.8/l
ibmcrypt-2.5.8.tar.gz dest={{ SOURCE_DIR }}

#当前主机 file 目录下已有 libmcrypt 依赖包
#- name: 拷贝现有 libmcrypt 依赖包到所有主机
#  copy: src=libmcrypt-2.5.8.tar.gz dest={{ SOURCE_DIR }}

#当前主机下没有 php 包
- name: 下载 php 包
  get_url: url={{ DOWNLOAD_URL }} dest={{ SOURCE_DIR }} owner={{ PHP_USER }} gr
oup={{ PHP_USER }}
```

```
# 当前主机 file 目录下已有 php 包
#- name: 拷贝现有 php 包到所有主机
#  copy: src=php-
{{ PHP_VER }}.tar.gz dest={{ SOURCE_DIR }} owner={{ PHP_USER }} group={{ PHP_US
ER }}

#- name: 解压依赖包 libmcrypt
#  unarchive: src={{ SOURCE_DIR }}/libmcrypt-
2.5.8.tar.gz dest={{ SOURCE_DIR }} copy=no

- name: 安装 libmcrypt
#  shell: "cd {{ SOURCE_DIR }}/libmcrypt-2.5.8 && yum -y install gcc-
c++ && ./configure && make && make install"
  shell: "yum -y install libmcrypt-2.5.8"

- name: 解压 php 包
  unarchive: src={{ SOURCE_DIR }}/php-
{{ PHP_VER }}.tar.gz dest={{ SOURCE_DIR }} copy=no owner={{ PHP_USER }} group={
{ PHP_USER }}
```

保存并退出 vim 编辑器，按 Esc 后，英文模式下 shift+:，输入 wq 退出。完成 download.yml 的编写

- php 安装 install.yml

在命令行中输入：

```
vim roles/php_install/tasks/install.yml
```

进入 vim 编辑器中，点击 i 进入插入模式，拷贝如下代码：

```
# 编译 PHP
- name: 编译 php     # 一大串编译选项和具体的部署环境有关，在 HW 云主机上测试，很多预置项是找
不到的
  shell: "cd {{ SOURCE_DIR }}/php-{{ PHP_VER }} && yum -y install sqlite-
devel && yum -y install oniguruma-devel && yum -y install libxml2-
devel && ./configure --prefix={{ PHP_DIR }} --with-config-file-
path={{ PHP_DIR }}/etc --with-mysqli=mysqlnd --with-pdo-mysql=mysqlnd --with-
mysql={{ MYSQL_DIR }} --with-mysql-sock={{ MYSQL_DIR }}/tmp/mysql.sock --with-
iconv-dir --with-freetype-dir --with-jpeg-dir --with-png-dir --with-zlib --
with-bz2 --with-libxml-dir --with-curl --with-gd --with-openssl --with-mhash  -
-with-xmlrpc --with-pdo-mysql --with-libmbfl --with-onig --with-pear --enable-
xml --enable-bcmath --enable-shmop --enable-sysvsem --enable-inline-
optimization --enable-mbregex --enable-fpm --enable-mbstring --enable-pcntl --
enable-sockets --enable-zip --enable-soap --enable-opcache --enable-pdo --
enable-mysqlnd-compression-support --enable-maintainer-zts  --enable-session --
with-fpm-user={{ PHP_USER }} --with-fpm-group={{ PHP_USER }}"

# 安装 PHP
- name: 安装 php    # j 后的数字代表进程数，一般来说和主机的核数有关系 8 核云主机，修改为 8
  shell: "cd {{ SOURCE_DIR }}/php-{{ PHP_VER }} && make -j 8 && make -
j 8 install"
```

```
- name: 创建 php-fpm 配置目录
  file: name={{ PHP_DIR }}/etc state=directory owner={{ PHP_USER }} group={{ PH
P_USER }} mode=0755 recurse=yes

- name: 修改 php-fpm 配置_1
  shell: "cd {{ SOURCE_DIR }}/php-{{ PHP_VER }} && cp php.ini-
production  {{ PHP_DIR }}/etc/php.ini"

- name: 修改 php-fpm 配置_2  # 设置单次推送内容最大体积
  lineinfile:
    dest: "{{ PHP_DIR }}/etc/php.ini"
    regexp: "post_max_size = 8M"
    line: "post_max_size = 16M"

- name: 修改 php-fpm 配置_3  # 配置缺省的最长执行时间
  lineinfile:
    dest: "{{ PHP_DIR }}/etc/php.ini"
    regexp: "max_execution_time = 30"
    line: "max_execution_time = 300"

- name: 修改 php-fpm 配置_4   # 设置了在强制终止脚本前 PHP 等待脚本执行完毕的时间，此时间以
秒计算。当脚本进入了一个无限循环状态 时此变量非常有用
  lineinfile:
    dest: "{{ PHP_DIR }}/etc/php.ini"
    regexp: "max_input_time = 60"
    line: "max_input_time = 300"

- name: 修改 php-fpm 配置_5  # 配置时区
  lineinfile:
    dest: "{{ PHP_DIR }}/etc/php.ini"
    regexp: ";date.timezone ="
    line: "date.timezone = Asia/Shanghai"

# 复制启动配置文件
- name: 拷贝启动配置文件
  shell: "cd {{ SOURCE_DIR }}/php-{{ PHP_VER }} && cp sapi/fpm/init.d.php-
fpm  /etc/init.d/php-fpm && chmod +x /etc/init.d/php-fpm"

# 复制 PHP 主配置文件
- name: 拷贝 php 主配置文件
  template: src=php-fpm.conf dest={{ PHP_DIR }}/etc/php-
fpm.conf owner={{ PHP_USER }} group={{ PHP_USER }}

# 编译安装 ldap 模块
- name: 编译安装 ldap 模块
```

```
  shell: "cd {{ SOURCE_DIR }}/php-{{ PHP_VER }}/ext/ldap && cp -
af /usr/lib64/libldap* /usr/lib/ && {{ PHP_DIR }}/bin/phpize && ./configure --
with-php-config={{ PHP_DIR }}/bin/php-config && make && make install"

- name: 修改 php-fpm 配置_6
  lineinfile:
    dest: "{{ PHP_DIR }}/etc/php.ini"
    regexp: ";extension=bz2"
    line: "aextension=ldap.so"

# 编译安装 getext 模块
- name: 编译安装 gettext 模块
  shell: "cd {{ SOURCE_DIR }}/php-{{ PHP_VER }}/ext/gettext && cp -
af /usr/lib64/libldap* /usr/lib/ && {{ PHP_DIR }}/bin/phpize && ./configure --
with-php-config={{ PHP_DIR }}/bin/php-config && make && make install"

- name: 修改 php-fpm 配置_7
  lineinfile:
    dest: "{{ PHP_DIR }}/etc/php.ini"
    regexp: ";extension=bz2"
    line: "aextension=gettext.so"

- name: 修改 php-fpm 配置_8
  lineinfile:
    dest: "{{ PHP_DIR }}/etc/php.ini"
    regexp: "pdo_mysql.default_socket="
    line: "pdo_mysql.default_socket=/usr/local/mysql/tmp/mysql.sock"

- name: 修改 php-fpm 配置_9
  lineinfile:
    dest: "{{ PHP_DIR }}/etc/php.ini"
    regexp: "mysqli.default_socket ="
    line: "mysqli.default_socket =/usr/local/mysql/tmp/mysql.sock"

- name: 启动 php 并开机启动
  shell: "chkconfig --add php-fpm && chkconfig php-fpm on && /etc/init.d/php-
fpm start"
```

保存并退出 vim 编辑器，按 Esc 后，英文模式下 shift+:，输入 wq 退出。完成 install.yml 的编写

- 引用文件 main.yml

在命令行中输入：

```
vim roles/php_install/tasks/main.yml
```

进入 vim 编辑器中，点击 i 进入插入模式，拷贝如下代码：

```
#引用 prepare、download、install 模块
```

```
#- include: prepare.yml
- include: download.yml
- include: install.yml
```

保存并退出 vim 编辑器，按 Esc 后，英文模式下 shift+:，输入 wq 退出。完成 main.yml 的编写

- php 主配置文件 php-fpm.conf

在命令行中输入：

```
vim roles/php_install/templates/php-fpm.conf
```

进入 vim 编辑器中，点击 i 进入插入模式，拷贝如下代码：

```
[global]
pid = {{ PHP_DIR }}/var/run/php-fpm.pid
error_log = {{ PHP_DIR }}/var/log/php-fpm.log
[www]
listen = 127.0.0.1:{{ PHP_PORT }}
listen.mode = 666
listen.owner = nobody
listen.group = nobody
user = {{ PHP_USER }}
group = {{ PHP_USER }}
pm = dynamic
pm.max_children = 50
pm.start_servers = 20
pm.min_spare_servers = 5
pm.max_spare_servers = 35
pm.max_requests = 500
rlimit_files = 1024
```

保存并退出 vim 编辑器，按 Esc 后，英文模式下 shift+:，输入 wq 退出。完成 php-fpm.conf 的编写

步骤 4 编写 nginx 分脚本

- 入口文件 nginx.yml

在命令行中输入：

```
vim nginx.yml
```

进入 vim 编辑器中，点击 i 进入插入模式，拷贝如下代码：

```
# 用于批量安装 Nginx
---
- hosts: all
  remote_user: Huawei
  gather_facts: True

  roles:
    - nginx_install
```

保存并退出 vim 编辑器，按 Esc 后，英文模式下 shift+:，输入 wq 退出。完成 nginx.yml 的编写

- 变量声明 main.yml

在命令行中输入：

```
vim roles/nginx_install/vars/main.yml
```

进入 vim 编辑器中，点击 i 进入插入模式，拷贝如下代码：

```
# 定义 nginx 安装的参数信息
NGINX_VER: 1.15.0
DOWNLOAD_URL: https://weirenzhengnew.obs.cn-north-
4.myhuaweicloud.com:443/Ansible/nginx-1.15.0.tar.gz
NGINX_USER: Huawei
NGINX_PORT: 80
SOURCE_DIR: /software
NGINX_DIR: /usr/local/nginx
DATA_DIR: /data/nginx
```

保存并退出 vim 编辑器，按 Esc 后，英文模式下 shift+:，输入 wq 退出。完成 main.yml 的编写

- nginx 下载 download.yml

在命令行中输入：

```
vim roles/nginx_install/tasks/download.yml
```

进入 vim 编辑器中，点击 i 进入插入模式，拷贝如下代码：

```
- name: 创建 nginx 用户组
  group: name={{ NGINX_USER }}  state=present

- name: 创建 nginx 用户
  user: name={{ NGINX_USER }}  group={{ NGINX_USER }}  state=present create_hom
e=False shell=/sbin/nologin

# - name: 创建 software 目录
#   file: name={{ SOURCE_DIR }} state=directory mode=0755 recurse=yes

- name: 创建日志目录
  file: name={{ item }} state=directory owner={{ NGINX_USER }} group={{ NGINX_U
SER }} mode=0755 recurse=yes
  with_items:
  - "{{ DATA_DIR }}"
  - "{{ DATA_DIR }}/log"

- name: 创建日志文件
  file: name={{ item }} state=touch owner={{ NGINX_USER }} group={{ NGINX_USER
}} mode=0644
```

```
  with_items:
  - "{{ DATA_DIR }}/log/access.log"
  - "{{ DATA_DIR }}/log/error.log"

# 当前主机下没有 nginx 安装包
- name: 下载 nginx 包
  get_url: url={{ DOWNLOAD_URL }} dest={{ SOURCE_DIR }} owner={{ NGINX_USER }}
group={{ NGINX_USER }}

# 当前主机 file 目录下已有 nginx 安装包
#- name: 拷贝现有 nginx 包到所有主机
#  copy: src=nginx-
{{ NGINX_VER }}.tar.gz dest={{ SOURCE_DIR }} owner={{ NGINX_USER }} group={{ NG
INX_USER }}

- name: 解压 nginx 包
  unarchive: src={{ SOURCE_DIR }}/nginx-
{{ NGINX_VER }}.tar.gz dest={{ SOURCE_DIR }} copy=no owner={{ NGINX_USER }} gro
up={{ NGINX_USER }}

# 复制 nginx 服务文件
- name: 拷贝 nginx 服务文件
  template: src=nginx.service dest=/usr/lib/systemd/system/nginx.service owner=
root group=root
```

保存并退出 vim 编辑器，按 Esc 后，英文模式下 shift+:，输入 wq 退出。完成 download.yml 的编写

- nginx 安装 install.yml

在命令行中输入：

```
vim roles/nginx_install/tasks/install.yml
```

进入 vim 编辑器中，点击 i 进入插入模式，拷贝如下代码：

```
# 编译 nginx
- name: 编译 nginx
  shell: "cd {{ SOURCE_DIR }}/nginx-{{ NGINX_VER }} && ./configure --
prefix={{ NGINX_DIR }} --user={{ NGINX_USER }} --group={{ NGINX_USER }} --http-
log-path={{ DATA_DIR }}/log/access.log --error-log-
path={{ DATA_DIR }}/log/error.log --with-http_ssl_module --with-
http_v2_module --with-http_realip_module --with-http_flv_module --with-
http_mp4_module --with-http_gunzip_module --with-http_gzip_static_module --
with-http_image_filter_module --with-http_stub_status_module"

# 安装 nginx
- name: 安装 nginx
  shell: "cd {{ SOURCE_DIR }}/nginx-{{ NGINX_VER }} && make && make install"
```

```
# 复制 nginx 主配置文件
- name: 拷贝 nginx 主配置文件
  template: src=nginx.conf dest={{ NGINX_DIR }}/conf/nginx.conf owner={{ NGINX_
USER }} group={{ NGINX_USER }}

- name: 创建 vhost 配置文件目录
  file: name={{ NGINX_DIR }}/conf/vhost state=directory owner={{ NGINX_USER }}
group={{ NGINX_USER }} mode=0755 recurse=yes

# 复制 nginx vhost 配置文件
- name: 拷贝 nginx vhost 配置文件
  template: src=server.conf dest={{ NGINX_DIR }}/conf/vhost/server.conf owner={
{ NGINX_USER }} group={{ NGINX_USER }} mode=0644

# 复制 nginx 额外配置文件
- name: 拷贝 nginx 额外配置文件
  template: src=fastcgi_params dest={{ NGINX_DIR }}/conf/fastcgi_params owner={
{ NGINX_USER }} group={{ NGINX_USER }} mode=0644

- name: 配置环境变量
  shell: " if [ `grep {{ NGINX_DIR }}/sbin /etc/profile |wc -l` -
eq 0 ]; then echo export PATH=$PATH:{{ NGINX_DIR }}/sbin >> /etc/profile && sou
rce /etc/profile; else source /etc/profile; fi"

- name: 启动 nginx 并开机启动
  shell: "systemctl daemon-
reload && systemctl enable nginx && systemctl start nginx"

- name: 添加 php 测试页 index.php
  shell: " echo '<?php phpinfo(); ?>' >> {{ NGINX_DIR }}/html/index.php"
```

保存并退出 vim 编辑器，按 Esc 后，英文模式下 shift+:，输入 wq 退出。完成 install.yml 的编写

- 引用文件 main.yml

在命令行中输入：

```
vim roles/nginx_install/tasks/main.yml
```

进入 vim 编辑器中，点击 i 进入插入模式，拷贝如下代码：

```
#引用 prepare、copy、install 模块
#- include: prepare.yml    # 理解为什么这里注释掉?
- include: download.yml
- include: install.yml
```

保存并退出 vim 编辑器，按 Esc 后，英文模式下 shift+:，输入 wq 退出。完成 main.yml 的编写

- nginx 主配置文件 nginx.conf

在命令行中输入：

```
vim roles/nginx_install/templates/nginx.conf
```

进入 vim 编辑器中，点击 i 进入插入模式，拷贝如下代码：

```
user nobody nobody;
worker_processes  1;
error_log {{ DATA_DIR }}/log/error.log crit;
pid /run/nginx.pid;
worker_rlimit_nofile 51200;
events {
    worker_connections  1024;
}
http {
    include       mime.types;
    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  {{ DATA_DIR }}/log/access.log  main;

    server_tokens       off;
    sendfile            on;
    send_timeout        3m;
    tcp_nopush          on;
    tcp_nodelay         on;
    keepalive_timeout   65;
    types_hash_max_size 2048;

    client_header_timeout 3m;
    client_body_timeout 3m;
    connection_pool_size 256;
    client_header_buffer_size 1k;
    large_client_header_buffers 8 4k;
    request_pool_size 4k;
    output_buffers 4 32k;
    postpone_output 1460;
    client_max_body_size 10m;
    client_body_buffer_size 256k;
    client_body_temp_path {{ NGINX_DIR }}/client_body_temp;
    proxy_temp_path {{ NGINX_DIR }}/proxy_temp;
    fastcgi_temp_path {{ NGINX_DIR }}/fastcgi_temp;
    fastcgi_intercept_errors on;

    gzip on;
    gzip_min_length 1k;
```

```
    gzip_buffers 4 8k;
    gzip_comp_level 5;
    gzip_http_version 1.1;
    gzip_types text/plain application/x-javascript text/css text/htm
    application/xml;

    default_type  application/octet-stream;
    include  {{ NGINX_DIR }}/conf/vhost/*.conf;
}
```

保存并退出 vim 编辑器，按 Esc 后，英文模式下 shift+:，输入 wq 退出。完成 nginx.conf 的编写

- nginx vhost 配置文件 server.conf

在命令行中输入：

```
vim roles/nginx_install/templates/server.conf
```

进入 vim 编辑器中，点击 i 进入插入模式，拷贝如下代码：

```
server {
    listen       80;
    server_name  localhost;
    location / {
        root   {{ NGINX_DIR }}/html;
        index  index.php index.html index.htm;
    }

    error_page   500 502 503 504  /50x.html;
        location = /50x.html {
            root   html;
        }

    location ~ \.php$ {
    root   {{ NGINX_DIR }}/html;
    fastcgi_pass   127.0.0.1:9000;
    fastcgi_index  index.php;
    fastcgi_param  SCRIPT_FILENAME  $document_root$fastcgi_script_name;
    include        fastcgi_params;
    }
}
```

保存并退出 vim 编辑器，按 Esc 后，英文模式下 shift+:，输入 wq 退出。完成 server.conf 的编写

- nginx 额外配置文件 fastcgi_params

在命令行中输入：

```
vim roles/nginx_install/templates/fastcgi_params
```

进入 vim 编辑器中，点击 i 进入插入模式，拷贝如下代码：

```
fastcgi_param  GATEWAY_INTERFACE  CGI/1.1;
fastcgi_param  SERVER_SOFTWARE    nginx;
fastcgi_param  QUERY_STRING       $query_string;
fastcgi_param  REQUEST_METHOD     $request_method;
fastcgi_param  CONTENT_TYPE       $content_type;
fastcgi_param  CONTENT_LENGTH     $content_length;
fastcgi_param  SCRIPT_FILENAME    $document_root$fastcgi_script_name;
fastcgi_param  SCRIPT_NAME        $fastcgi_script_name;
fastcgi_param  REQUEST_URI        $request_uri;
fastcgi_param  DOCUMENT_URI       $document_uri;
fastcgi_param  DOCUMENT_ROOT      $document_root;
fastcgi_param  SERVER_PROTOCOL    $server_protocol;
fastcgi_param  REMOTE_ADDR        $remote_addr;
fastcgi_param  REMOTE_PORT        $remote_port;
fastcgi_param  SERVER_ADDR        $server_addr;
fastcgi_param  SERVER_PORT        $server_port;
fastcgi_param  SERVER_NAME        $server_name;
```

保存并退出 vim 编辑器，按 Esc 后，英文模式下 shift+:，输入 wq 退出。完成 fastcgi_params 的编写

- nginx 服务文件 nginx.service

在命令行中输入：

```
vim roles/nginx_install/templates/nginx.service
```

进入 vim 编辑器中，点击 i 进入插入模式，拷贝如下代码：

```
[Unit]
Description=The nginx HTTP and reverse proxy server
After=network.target remote-fs.target nss-lookup.target

[Service]
Type=forking
PIDFile=/run/nginx.pid
# Nginx will fail to start if /run/nginx.pid already exists but has the wrong
# SELinux context. This might happen when running `nginx -
t` from the cmdline.
# https://bugzilla.redhat.com/show_bug.cgi?id=1268621
ExecStartPre=/usr/bin/rm -f /run/nginx.pid
ExecStartPre={{ NGINX_DIR }}/sbin/nginx -t
ExecStart={{ NGINX_DIR }}/sbin/nginx
ExecReload=/bin/kill -s HUP $MAINPID
KillSignal=SIGQUIT
TimeoutStopSec=5
KillMode=process
PrivateTmp=true

[Install]
WantedBy=multi-user.target
```

保存并退出 vim 编辑器，按 Esc 后，英文模式下 shift+:，输入 wq 退出。完成 nginx.service 的编写。

步骤 5 编写 lnmp 总脚本

了解完分脚本的编写，再来理解总脚本，就容易多了。总脚本，即统一调用分脚本，从而实现 lnmp 的一键部署。

- 创建 lnmp 入口文件，集中调用 roles（分脚本）

在命令行中输入：

```
vim lnmp.yml
```

进入 vim 编辑器中，点击 i 进入插入模式，拷贝如下代码：

```
---
- hosts: all
  remote_user: root
  gather_facts: True

  roles:
    - mysql_install

- hosts: all
  remote_user: root
  gather_facts: True

  roles:
    - php_install
    - nginx_install
```

保存并退出 vim 编辑器，按 Esc 后，英文模式下 shift+:，输入 wq 退出。完成 lnmp.yml 的编写。

1.6 部署测试

步骤 1 准备部署环境

- 在正式部署前，需搭建控制服务器与节点服务器间的连接。

在控制服务器中（安装了 Ansible 组件）输入

```
ssh-keygen -t rsa
```

在生成本地公钥过程中，有三次交互提示，敲回车，保持默认选项。

生成下图所示结果，即成功。

将本地生成的公钥，发送至节点服务器，输入命令：

```
ssh-copy-id -i ~/.ssh/id_rsa.pub root@XXX.XXX.X.XXX
```

高亮处需替换成节点服务器的 IP，建议选择内网 IP，公网 IP 亦可。

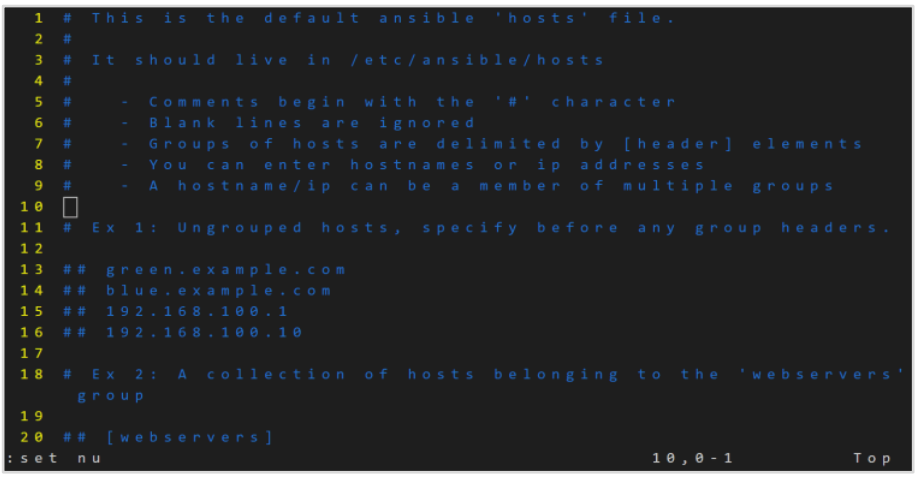
该命令输入后，根据提示，输入节点服务器的密码，回车确认后，即完成服务器连接。

修改控制服务器 ansible/hosts 配置文件

输入命令：

```
vim /etc/ansible/hosts
```

切换 vim 输入模式，按 Esc 后，英文模式下 shfit+:，输入 set nu，配置文档显示行数



光标落在第 10 行处，点击 i ，切换插入模式，输入节点服务器的内网 IP，公网亦可。

保存并退出 vim 编辑器，按 Esc 后，英文模式下 shift+:，输入 wq 退出。

步骤 2 一键部署

成功编写 ansible 脚本后，现在可以开始测试。

若您未完成，可在命令行中输入下面的链接，下载已编写好的 playbook，但强烈建议您自己完成 playbook 的编写。

```
wget https://weirenzhengnew.obs.cn-north-4.myhuaweicloud.com/Ansible/lnmp.zip
```

解压下载的 playbook，输入命令：

```
yum -y install unzip
```

```
unzip lnmp.zip
cd lnmp
```

在部署前，请确认当前所在的目录，输入 ll

```
[root@ecs-devcould-ansible-[illegible] lnmp]# ll
total 32
-rw-r--r-- 1 root root  240 Dec 13 23:30 lnmp.yml
-rw-r--r-- 1 root root  205 Dec 13 23:31 mysql.yml
-rw-r--r-- 1 root root  133 Dec 12 17:22 nginx.yml
-rw-r--r-- 1 root root  129 Dec 12 17:22 php.yml
-rw-r--r-- 1 root root 1309 Dec 12 15:22 prepare.yml
-rw-r--r-- 1 root root  874 Dec 12 15:22 README.en.md
-rw-r--r-- 1 root root 1066 Dec 12 16:43 README.md
drwxr-xr-x 5 root root 4096 Dec 12 15:19 roles
```

回显如上图所示，即正确，一键部署，输入命令：

```
ansible-playbook lnmp.yml
```

执行过程，因存在 mysql 的下载和 php 的编译安装，视网络状态，将维持 5-8 分钟。

部署成功后将显示：

```
TASK [nginx_install : 启动nginx并开机启动] ********************************************
*******
changed: [illegible]

TASK [nginx_install : 添加php测试页index.php] ****************************************
*****
changed: [illegible]

PLAY RECAP ************************************************************************
illegible                  : ok=51   changed=43   unreachable=0    failed=0    skipped=0
```

步骤 3 测试结果

上述 playbook 通过在控制服务器上安装 Ansible 工具，在节点服务器上部署 LNMP 环境，可以通过命令查看节点服务器上 mysql/php/nginx 端口的占用情况。

```
netstat -lntp
```

回显为下图，说明节点服务器上 LNMP 部署成功！

```
[root@ecs-devcould-ansible-[illegible] lnmp]# netstat -lntp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:9000          0.0.0.0:*               LISTEN      1010/php-fpm: maste
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      806/nginx: master p
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      2214/sshd
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN      1167/master
tcp6       0      0 :::3306                 :::*                    LISTEN      743/mysqld
tcp6       0      0 :::22                   :::*                    LISTEN      2214/sshd
tcp6       0      0 ::1:25                  :::*                    LISTEN      1167/master
```

2 DevCloud 自动化部署 Discuz

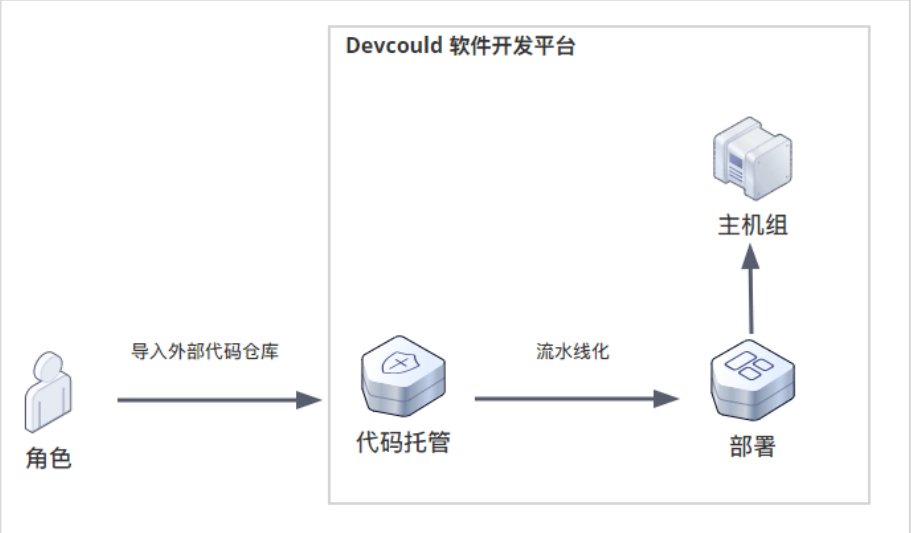
2.1 实验介绍

在第一部分中，详细剖析了一键部署 LNMP 环境的 playbook 写法，整个操作都是在弹性云服务器中进行，十分耗时，且容易出错。传统运维人员在主机环境写 playbook，实在是无奈之举。

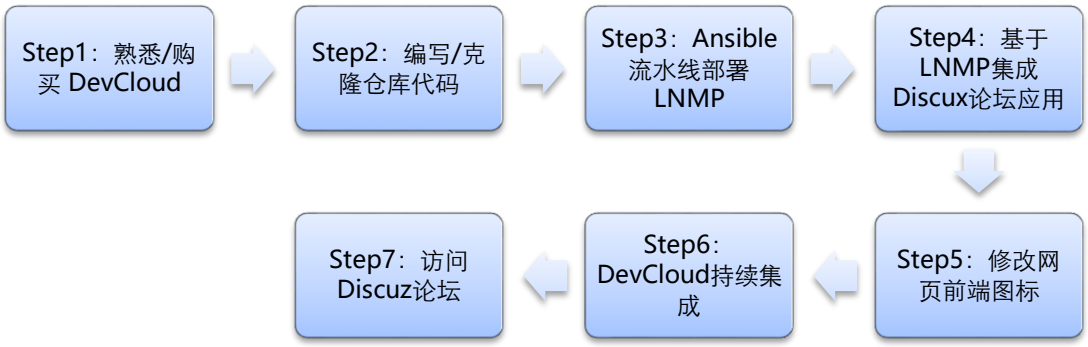
在第二部分，将介绍基于华为云 DevCloud，更方便地编写 playbook 代码，甚至可以拷贝他人分享的代码仓库，同时基于自身 Ansible 组件实现快速运维部署，减轻工作负担，减少不必要的问题排查环节。

2.2 实验架构

基于 DevCloud 软件开发平台中的代码托管和部署两模板，运维人员可在代码托管平台编写脚本代码，或克隆他人写好的代码。后续在部署页面中可设置主机组，根据需求将代码设置成流水线业务，快速部署到指定主机组。



2.3 实验流程



2.4 资源准备

在进行 DevCloud 实验之前，需提前熟悉 DevCloud 操作界面，搭建实验及资源准备均在“**华北-北京四**”区域下进行：

- 使用 Chrome（86 以上）或版本相近的 Edge 浏览器
- 熟悉 DevCloud 操作界面
- 购买两台与实验 1 中相同配置的弹性云服务器

2.4.1 实验环境准备

步骤 1 购买 1 台弹性云服务器

基于实验 1 过程，在弹性云服务器管理界面，点击更多，选择购买相同配置。

新页面中点击右下角下一步，设置云主机密码后，在确认配置界面，如下图所示即可

步骤 2 熟悉华为云 DevCloud

使用 DevCloud

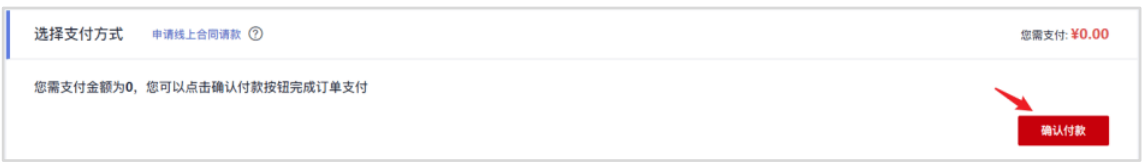
在华为云控制台，顶栏搜索栏中键入：DevCloud，即可找到入口。

或点击链接：https://www.huaweicloud.com/devcloud/?locale=zh-cn

在主页面点击 立即使用，进入 DevCloud

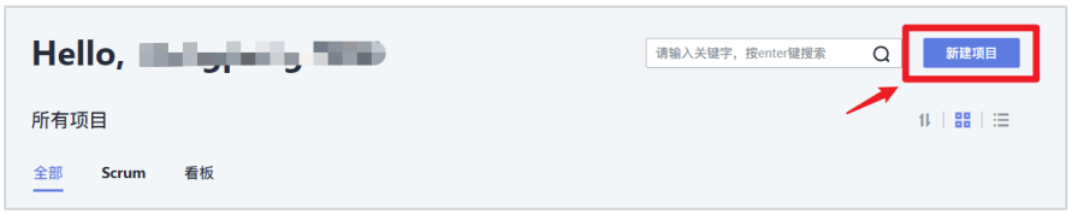
区域选择华北-北京四，首次使用 DevCloud，需开通服务，个人开发者可使用基础版或专业版（费用优惠），选中基础版，点击免费开通。

在新页面中，点击下一步，在支付页中点击确认付款



返回 DevCloud 控制台，点击右上角，立即使用

新页面中，点击新建项目



新页面中，选择 Scrum 空白模板，并键入项目信息。

2.5 导入代码仓库

步骤 1 代码托管页面

在 DevCloud 软件开发平台顶栏，选中代码—代码托管。

在代码托管的空白页面中，点击右上角 普通新建下拉栏，选中导入外部仓库。

新页面中，复制粘贴以下链接，导入外部仓库源码。

https://gitee.com/Breaveman/ansible.git

新页面中，保留默认选项，点击确认，跳转至代码托管页面

步骤 2 代码详情

点击刚导入的仓库，即上图位置，进入源码仓库详情页，在左侧可展开仓库目录树。

步骤 3 代码编辑

在代码详情页面，可新增文档或选择想要修改的文档进行修改、注释、删除。

2.6 构建部署

步骤 1 配置主机组

DevCloud 中的部署、测试，需事先设置好指定的主机组，即部署的对象。

在 DevCloud 顶栏上，依次选择 构建&发布—部署—主机管理，如下图：

在主机管理页面，此时尚无可用主机组，点击添加主机组。

进入主机组编辑页面。回顾实验 1 的内容，购买的两台弹性云服务器，1 台做控制服务器，1 台做部署 LNMP 环境的节点服务器。在实验 2 中仅购买 1 台弹性云服务器，作为为部署主机组，下面开始设置：

点击保存，进入主机信息页面，在该页面中，可导入 2.4.1 中购买的云服务器。需明确的是：导入 ECS，必须满足云主机和 DevCloud 在同一区域，即实验准备中要求的华北-北京四。若不在同一区域，可使用添加主机，手动输入主机 IP 等信息。

弹出的对话框中，输入云服务器用户和密码，点击添加，部署主机组设置完成。

步骤 2 部署流程

主机组配置成功，下一步开始构建基于 DevCloud 的自动化部署。

在上图中，选择 构建&发布，点击部署，进入新页面，点击新建任务。

在模板选择页面，选中—空白模板，点击下一步，进入流程设计页面。

在创建流程前，先梳理下实验 1 中 1.6 的思路，基于以上，罗列出关键操作：

1. 控制主机组，需提前安装 Ansible
2. 控制主机组，需写好 playbook
3. 控制主机组，需生成本地公钥，并发送至待部署主机
4. 控制主机组，需修改 /etc/ansible/hosts 配置文件
5. 控制主机组，执行 playbook，开始部署

DevCloud 中，集成了 Ansible 组件，平台和华为云之间的强耦合，可快速部署任务，从而减轻运维工作负担。无需上述 5 步，配合仓库代码可直接对目标主机组进行部署。

为演示工作流程，下面将使用分脚本逐一部署 mysql、php、nginx

- 安装 mysql

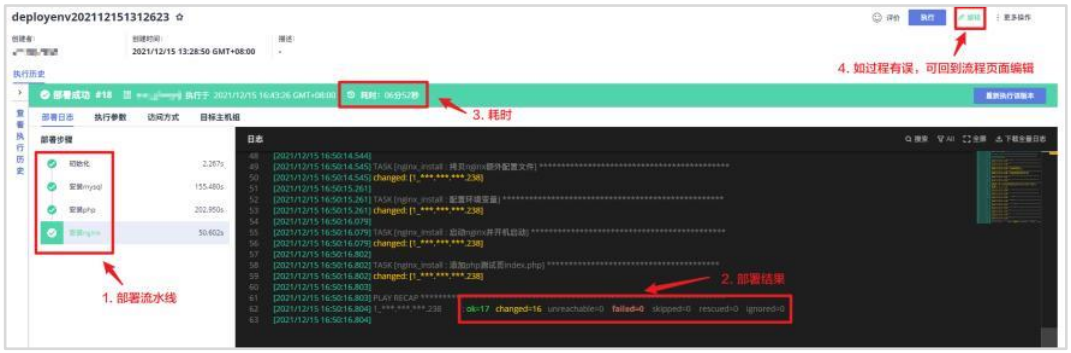
在步骤选择区，搜索框中输入 ansible，开始搜索，并添加

进入 Ansible 组件配置信息页

同理，添加步骤：安装 php、安装 nginx，最后如下图所示即可，点击执行，开始部署。

步骤 3 部署界面

执行部署过程，可关注日志框的回显内容，了解程序走到哪一步，最终直指部署成功



步骤 4 集成 Discuz 应用

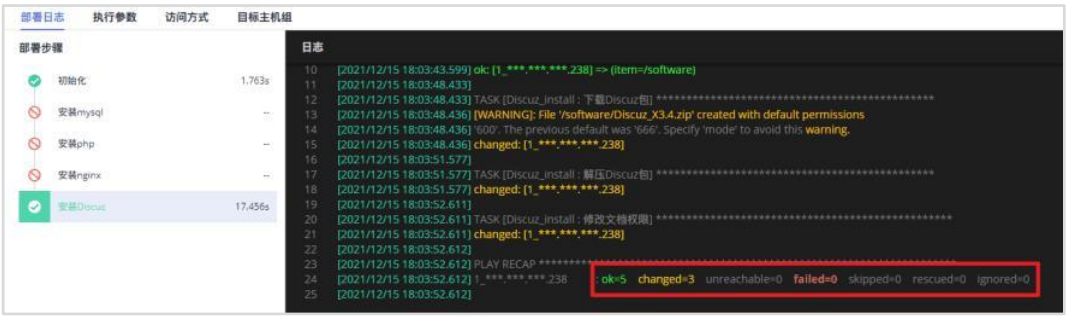
步骤 2、3，实现了通过 DevCloud 快速部署 LNMP 环境。现在可以在 LNMP 的基础上加上功能程序，比如说 Discuz。

建议有兴趣的同学，可以参考 1.5.3 的脚本编写过程，自己写一份部署 Discuz 的 playbook。

细心的同学可能在代码仓库中，已经发现一份 Discuz 脚本，现在开始部署 Discuz。

和前面的步骤一样，只要在流程模板中添加对应部署 Discuz 的代码即可。

执行结束后，显示如下图，即成功部署 Discuz 论坛。



2.7 结果验证

步骤 1 安装 Discuz

在浏览器网页中输入部署主机的公网 ip:

http://XXX.XXX.XXX.XXX/bbs/install

切换到 Discuz 安装向导

新页面点击下一步，在安装数据库页面中输入必要信息，其中数据库密码为 Huawei@123，在 mysql 脚本的变量声明文件里，安装前可修改为其他。

点击下一步，进入安装页面，完整完成后，显示如下：

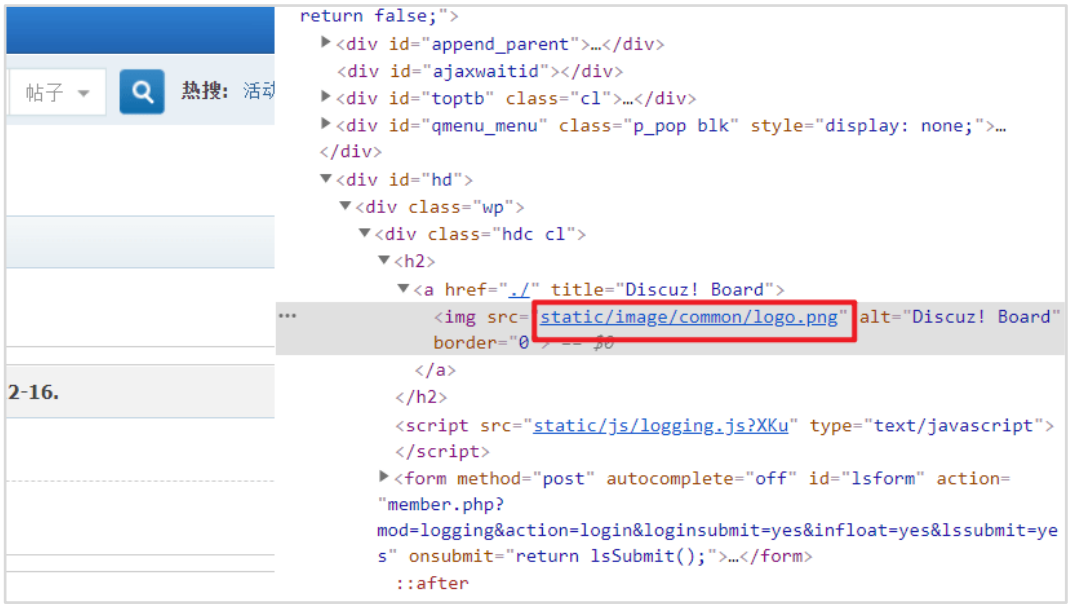
步骤 2 持续集成

修改 Discuz 网页前端图标

想要修改此处的页面图标，首先得知道原图标的调用地址，鼠标选中原图标，右击选择检查

弹出的调试页面中，已阴影显示原图标的调用地址 ：
/usr/local/nginx/html/bbs/static/image/common/logo.png

现在只需要修改对应的图标就可以！



可以上传自己意愿的图标，这里提供一链接：

https://weirenzhengnew.obs.cn-north-4.myhuaweicloud.com/Ansible/logo.png

下面演示如何在 DevCloud 中完成图标修改这一步。

在华为云 DevCloud 中，切换到编辑部署任务页面

新增步骤项目—修改网页图标

在添加的，执行 shell 命令步骤中，修改参数如下图：

shell 命令为

```
cd /usr/local/nginx/html/bbs/static/image/common
mv logo.png logo1.png
wget https://weirenzhengnew.obs.cn-north-4.myhuaweicloud.com/Ansible/logo.png
```

部署完成后，在浏览器中新建一无痕页面，输入：

http://XXX.XXX.XXX.XXX/bbs/forum.php 此处为部署服务器的公网 IP

页面图标完成修改！

3 资源释放

3.1 实验说明

在完成所有实验之后，需手动释放收费服务所占用的资源，包括：

- 弹性云服务器（ECS）
- 安全组（SG）、虚拟私有云（VPC）
- 释放 DevCloud

3.2 删除弹性云服务器

步骤 1 登录到华为云控制台

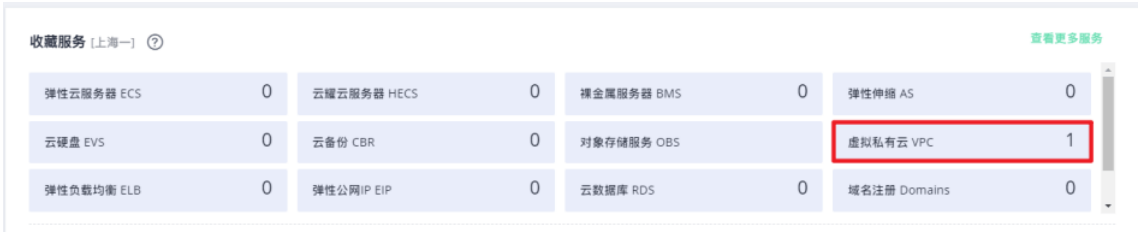
登陆后，在弹性云服务器控制页面，选中要删除的服务器，点击更多，删除。

在弹出的删除页面中，选中框内两项，否则未释放的公网 IP 和磁盘将持续收费！

3.3 删除安全组（SG）、虚拟私有云（VPC）

步骤 1 删除安全组

在华为云控制台中，点击虚拟私有云



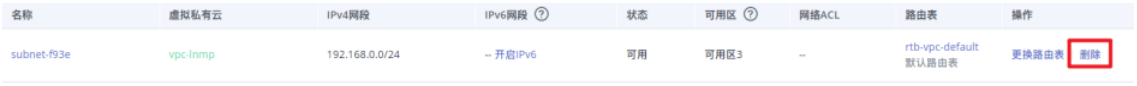
在新页面中，点击访问控制—安全组

在新页面中，选择 sg-lnmp，选择更多，点击删除

步骤 2 删除虚拟私有云

在左侧网络控制台，选中虚拟私有云，选择 vpc-lnmp 的子网

在新页面中，删除绑定在 vpc-lnmp 的子网



在左侧网络控制台，选中虚拟私有云，删除 vpc-lnmp

3.4 释放 DevCloud

步骤 1 登录到 DevCloud 控制台，留意切换区域到华北-北京四

留意到：默认为自动续费，选择设置，进入设置页面

在上图页面中，选择自动续费项，选中在北京四开通的 DevCloud，在更多中选择到期不续费。在新的页面中点击确定。

4 参考资料

感谢您的耐心查阅，更多资讯请浏览：

1. 华为云 Ansible
2. Ansible 中文指南
3. Ansible 官网手册
4. 华为云 DevCloud
5. What is Ansible