2.4 SSH、部署、脚本

服务器环境部署

Step-1: 创建登录密钥

```
$ ssh-keygen -t rsa  # 执行此命令

# 程序输出
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):  # 确认密钥文件位置（敲回车）
Enter passphrase (empty for no passphrase):  # 为密钥设置密码（无需密码，直接回车）
Enter same passphrase again:  # 确认密码（再次回车）
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:
SHA256:WpGZKdaT3SbGlx+pNi6H5/SBNZXjk5C468y+4MeNKxs root@box
The key's randomart image is:
+---[RSA 2048]----+
|                 |
|       . O ......|
|      o X =.=ooo.|
|     . . + +.oooo|
|        S  .+ ++ |
|       o   +.+ ..|
|      .  E+.O .  |
|        ..*X o . |
|         o=B+ .  |
+----[SHA256]-----+
```

Step-2: 复制公钥到远程服务器

1. 本地打开 `~/.ssh/id_rsa.pub` 文件，复制全部文本内容
2. ssh 登录到远程服务器，vim 打开 `~/.ssh/authorized_keys` 文件
3. 将复制的内容写入文件，保存退出

Step-3: 关闭密码登录

- 服务器现在**能用密钥登录**的全都是密钥登录了

密钥设置好以后，便可以关闭服务器的密码登录，保证服务器不会被暴力破解，增强安全性.

以后登录服务器只允许通过密钥登录。

1. 登录服务器, 打开 `/etc/ssh/sshd_config` 文件
2. 找到 `PasswordAuthentication yes` 这行设置，将 `yes` 改为 `no`
3. 执行 `service ssh restart` 重启 SSH 服务

Step-4: 更新服务器软件，安装所需组件

- 使用开源项目：5minutes，加固服务器

```
# 更新源
$ sudo apt update -y
$ sudo apt upgrade -y
# 安装软件包
$ sudo apt install -y gcc make openssl git mysql-server zip p7zip
apache2-utils sendmail
# 安装必要依赖
$ sudo apt install -y libbz2-dev libpcre3 libpcre3-dev libreadline-dev
libsqlite3-dev libssl-dev zlib1g-dev
```

Step-5: 安装 Nginx、Redis

1. **新方法**

    ```
    # ubuntu
    sudo apt-get update
    sudo apt-get install nginx redis-server
    ```

2. 旧方法

    1. 浏览器中打开 nginx、redis 官网，找到其最新稳定版安装包的下载地址，右键点击复制
    2. ssh 登录到服务器
    3. 通过 wget 下载复制的软件包地址
    4. 解压、编译、安装

    ```
    $ cd nginx-1.14.2
    $ ./configure
    $ make
    $ make install
    ```

Step-6: 配置 nginx、redis、mysql 等组件

- 按需求修改配置文件

    1. 先**备份**配置文件
    2. 小范围修改
    3. 出错回退

Step-7: 运行各组件

Step-8: 代码上传、运行

1. 登录服务器, 创建好项目保存路径

```
$ cd /opt/
$ mkdir -p swiper/logs
```

2. 进入项目目录，执行如下操作

```
$ rsync -crvP --exclude={.venv,.git,__pycache__,logs}  ./
root@X.X.X.X:/opt/swiper/
```

3. 运行

```
$ gunicorn -c swiper/gunicorn-config.py swiper.wsgi
```

4. 面试题：
    1. 生产环境中的django程序是如何运行的?

SHELL 脚本编程

首行

脚本文件第一行通过注释的方式指明执行脚本的程序

常见方式有 `#!/bin/bash` 或 `#!/usr/bin/env bash`

变量

```
# 变量定义：等号前后没有空格
a=12345678

# 使用变量：变量名前面加上 $ 符
echo "----$a----"
printf "===>$a<===\n"

# 定义当前Shell下的全局变量
export ABC=9876543210123456789

# 定义完后，在终端里用 source 加载脚本
source ./test.sh
```

常用的系统环境变量

`$PATH`: 可执行文件目录 `$PWD`: 当前目录 `$HOME`: 家目录

分支控制语句: `if`

```
if [[ $a == "12345678" ]]; then
    echo 'this is a arg'
elif [ -d $0 ]; then
    echo 'this is a dir'
elif [ -f $0 ]; then
    echo 'this is a file'
else
    echo '98765432'
fi
```

循环控制语句: for

```
# 从1到10显示数字
for i in $(seq 1 10)
do
    echo "num: $i"
done
```

函数

```
[function ]foo() {
    echo "Hello BJ-1813"
    for f in `ls ../`
    do
        echo $f
    done
}

# 函数的使用，不需要小括号
foo
```

函数中使用参数

```
bar() {
    echo "执行者是 $0"
    echo "参数数量是 $#"

    if [ -d $1 ]; then  # 检查传入的第一个参数是否是文件夹
        for f in `ls $1`
        do
            echo $f
        done
    elif [ -f $1 ]; then
```

```
        echo 'This is a file: $1'  # 单引号内的变量不会被识别
        echo "This is a file: $1"  # 如果不是文件夹，直接显示文件名
    else
        echo 'not valid'  # 前面都不匹配显示默认语句
    fi
}
```

开发服务器部署脚本

- 系统部署脚本
    - 将前述步骤通过脚本方式组织起来
    - 可通过参数方式，选择执行独立步骤
- 代码发布脚本
    - 上传脚本到服务器
    - 上传完成后，重启服务器
- 代码发布流程 * 发布服务器： * checkout 出来一个 master 上的稳定可发布分支 * 用 git 命令清理 .git * 打包 * 把压缩包复制到各个 application 服务器上： scp / rsync / http * application * 拿到压缩包 * 解压 * 简单做法： * 把新代码覆盖旧的版本 * 复杂一点： * 把旧代码复制一份，按日期或版本或tag起好名字，备用； * 然后把新代码覆盖旧代码 * 用新目录放新代码，修改gunicorn配置，然后平滑重启gunicorn服务 * 写成脚本
- 程序启动脚本
- gunicorn启动脚本
- 程序停止脚本
    - 写好脚本
- 程序重启脚本
    - 重启过程服务不可中断
    - 重启
        - nginx: 不间断重启: `kill -HUP [进程 ID]`
            - 新方式：sudo service nginx restart
        - gunicorn
            -
        - mysql:
            - sudo service mysql restart
        - redis:
            - sudo service redis restart
    - 大型分布式服务器不间断重启:
        - 禁止一次性重启全部机器
        - 一次重启集群中的一部分

    - 重启后的服务器没有问题后，再重启第二部分
    - 依次重复上述步骤，直至全部重启完成
    - facebook服务发布层级：
        - 7层

多台服务器的部署方法

- 小学生：SSH 手动部署
- 中学生：SSH + Shell 脚本
- 大学生：Ansible + 部署文件 / puppet / salt
- 研究生：app 容器管理：Docker 部署：架构
- 博士生：K8S:kubernets app集群管理：架构

观察者非特殊原理

- 后端技术Linux MySQL Python等已经稳定存在了二三十年，从观察者非特殊原理出发，还有大概率能存在几十年