# 异网用户识别平台部署文档

目录

**[一、 系统准备](#_Toc869_WPSOffice_Level1)** **[1](#_Toc869_WPSOffice_Level1)**

[1. 硬件要求](#_Toc10479_WPSOffice_Level2) [1](#_Toc10479_WPSOffice_Level2)

[2. 软件要求](#_Toc20463_WPSOffice_Level2) [2](#_Toc20463_WPSOffice_Level2)

**[二、 部署包内容](#_Toc10479_WPSOffice_Level1)** **[3](#_Toc10479_WPSOffice_Level1)**

[1. Web 层部分 （位于 sz\_broad.tar.gz 压缩包内）](#_Toc16695_WPSOffice_Level2) [3](#_Toc16695_WPSOffice_Level2)

[2. 计算层部分 （位于 `sz\_compute.tar.gz` 压缩包内）](#_Toc7154_WPSOffice_Level2) [8](#_Toc7154_WPSOffice_Level2)

**[三、 部署步骤](#_Toc20463_WPSOffice_Level1)** **[10](#_Toc20463_WPSOffice_Level1)**

[1. Web 部分（以下操作没有对于用户身份以及权限的特殊要求）](#_Toc7098_WPSOffice_Level2) [10](#_Toc7098_WPSOffice_Level2)

[2. 计算层部分](#_Toc22894_WPSOffice_Level2) [13](#_Toc22894_WPSOffice_Level2)

[3. 文件服务器部分](#_Toc25178_WPSOffice_Level2) [14](#_Toc25178_WPSOffice_Level2)

[4. 补充说明（db\_helper.py 文件）](#_Toc21992_WPSOffice_Level2) [15](#_Toc21992_WPSOffice_Level2)

**[四、 系统预置数据说明](#_Toc16695_WPSOffice_Level1)** **[16](#_Toc16695_WPSOffice_Level1)**

[1. 关于数据库](#_Toc4859_WPSOffice_Level2) [16](#_Toc4859_WPSOffice_Level2)

[2. 关于平台系统登录](#_Toc3194_WPSOffice_Level2) [16](#_Toc3194_WPSOffice_Level2)

**[五、 平台日志一览](#_Toc7154_WPSOffice_Level1)** **[17](#_Toc7154_WPSOffice_Level1)**

## 系统准备

1. 硬件要求
   1. 计算层要求(最低要求)：CPU 24 核 内存 32 G
   2. 存储层要求(最低要求)：外存 1T+ 且计算层可直接访问
   3. Web 平台要求(最低要求)：CPU 4核 内存 8G 外存 10G+
2. 软件要求
   1. 计算层要求：
      1. Anaconda - Python 3.7
      2. Python 第三方库：
         * Imblearn ( <https://pypi.org/project/imblearn/> )
   2. Web 平台要求：
      1. Python 3.7+
      2. MySQL 5.7+
      3. Python 第三方库

* Flask ( <https://pypi.org/project/Flask/> )
* Flask-SQLAlchemy ( <https://pypi.org/project/Flask-SQLAlchemy/> )
* Flask-HTTPAuth ( <https://pypi.org/project/Flask-HTTPAuth/> )
* passlib ( <https://pypi.org/project/passlib/> )
* Requests ( <https://pypi.org/project/requests/> )
* PyMySQL ( <https://pypi.org/project/PyMySQL/> )
* gunicorn ( <https://pypi.org/project/gunicorn/> )

## 部署包内容

1. Web 层部分 （位于 sz\_broad.tar.gz 压缩包内）
   1. 前端部分文件说明 （已单独打包为 sz\_broad\_frontend.tar.gz 压缩包）
      * 概述

前端开发框架为vue 2.5.17, UI框架为ant-design-vue1.3.2, 包管理工具为yarn总共分为2部分, 第一部分是web项目的源代码, 第二部分是项目配置文件。

* + - 详细介绍
      * 源代码

Web项目下主要分为src, public俩部分, 其中:

1) src目录下主要是项目的整体功能实现, 主要包括以下目录/文件

①　main.js: 整体项目的入口文件,项目构建初始化(用于引入项目需要的文件)

②　permission.js: 实现对用户登录/跳转页面的全局操作(路由守卫)

③　App.vue: 页面的根挂载点(根容器页面),会挂载到模板页

④　api: 存放项目请求接口

⑤　assets: 存放项目需要的静态资源文件

* custom.less: 用于部分页面UI样式覆盖
* global.less: 用于全局UI样式覆盖

⑥　components: 全局公用组件

⑦　config: 存放项目配置文件

* defaultSettings.js: 项目整体主题的配置文件
* router.config.js: 项目的路由配置文件(每个页面)

⑧　core: 项目引导,全局配置初始化,依赖包引入

⑨　router: 存放项目路由配置文件(路由基本配置)

⑩　store: (运行存储库)存放项目需要的系统变量

⑾ utils: 存放项目中的工具文件

* request.js: 网络请求处理工具

⑿ views: 存放项目功能页面

* Home.vue: 首页
* 404.vue: 404页面
* user: 存放用户登录页面/用户注册页面
* exception: 存放异常页(404/403/500)
* dashboard: 项目模块家宽营销

(1) modeling: 为建模工具模块(文件夹内对应的components文件夹存放该模块所需要的功能弹窗组件)

(2) different: 为异网宽带用户识别模块(文件夹内对应的components文件夹存放该模块所需要的功能弹窗组件)

(3) homeg: 为家庭群组识别模块(文件夹内对应的components文件夹存放该模块所需要的功能弹窗组件)

(4) netuser: 为宽带用户识别模块(文件夹内对应的components文件夹存放该模块所需要的功能弹窗组件)

(5) Welcome.vue: 为家宽营销控制台

2) public目录下存放项目运用到的资源文件(外部插件及静态资源及提供用户下载的样例文件)

* + - * + 项目配置文件

1) package.json: 项目开发依赖的外部资源包

2) babel.config.js: babel配置文件

3) vue.config.js: vue的打包配置文件

4) dist: 为项目打包编译后生成的项目文件夹

* 1. 后端部分文件说明
     + 概述

后端部分采用 Flask 作为 Web 框架，使用 pip 作为包管理器，基于 Python 3.7.x 语言编写。

* + - 详细介绍
      * docs 目录：接口文档说明，主要包括两部分 —— api.md 为前后端接口文档、task\_api.md 为前后台接口文档（也即计算层-Web 层交互文档）
      * frontEndCode 目录：前端代码所在目录，详见「前端部分文件说明」，已单独打包在 sz\_broad\_frontend.tar.gz 压缩包内。
      * src 目录：后端源代码，亦可理解为 Python 中的包，其中的 app 变量即为 Flask 应用程序。其内容根据 Flask 框架要求以及实际项目需要进行组织。
        + \_\_init\_\_.py 文件：Python 标准化包所需文件，这里具体负责：

配置静态文件路径

配置配置文件读取方式

配置数据库信息

初始化 digest HTTP 验证方式

初始化日志记录器

蓝图注册

首页路由指定

* + - * + admin 目录：admin 蓝图，属于 View 层代码，负责用户管理逻辑并提供对外接口。

view.py 负责用户登陆注册以及提供用户列表信息

* + - * + calling 目录：calling 蓝图，属于 View 层代码，负责模型调用逻辑并提供对外接口。

broadband\_view.py 负责宽带模型调用

othernet\_view.py 负责异网模型调用

* + - * + model 目录：ORM 模型目录，属于 Model 层代码，定义了各蓝图对应的数据模型以及 SQLAlchemy 实例。
        + modeling 目录：modeling 蓝图，属于 View 层代码，负责计算层建模逻辑并提供对外接口。

adapter.py 适配器，负责调整 ORM 模型到接口模型的适配

broadband\_view.py 负责宽带用户模型建模

family\_view.py 负责家庭群组模型建模

othernet\_view.py 负责异网用户模型建模

view.py 负责通用建模以及聚类分析相关操作

* + - * + tasks 目录：tasks 蓝图，属于 View 层代码，提供提供与计算层交互的接口。

calling\_view.py 提供模型调用相关接口

modeling\_view.py 提供模型建模相关接口

validating\_view.py 提供模型验证相关解耦库

view.py 提供聚类操作等相关接口

* + - * + utils 目录：封装的全局工具函数。

response.py 负责封装 http 响应，提供统一的形式并管理出入日志。

* + - * + validating 目录：validating 蓝图，属于 View 层代码，负责模型调用结果验证逻辑并提供对外接口。

broadband\_view.py 负责宽带模型调用验证

othernet\_view.py 负责异网模型调用验证

* + - * test 目录：接口测试目录文档。可以使用 `python3 -m unittest` 批量测试（请测试数据库下进行单元测试，勿在正式环境下测试）
      * config.py：Flask 应用程序的配置文件，负责配置日志记录、数据库连接、加密密钥等常规信息以及其他应用需要的自定义配置信息。
      * 【文件已删除】create\_db.py：初始化数据库表脚本，注意此脚本一旦运行会删除 sz\_broad 数据库原有的表并重新建表，仅用于初始化时的数据库迁移。
      * db\_helper.py: 数据库帮助脚本，封装了一系列删库、建库以及生成管理员账户的操作。详见部署部分说明。
      * Dockerfile 文件：未使用的 Docker 配置文件，由于未使用容器化部署因此可忽略。
      * file\_server.py：需要部署在计算层的 csv 表格列头分析的 HTTP 服务，提供给无法直接访问位于计算层主机（10.17.2.48）的存储层的 Web 系统使用。
      * gunicorn.conf.py：gunicorn 服务器配置文件，更多关于 gunicorn 服务器以及其余配置项信息可查阅其官网 （<https://gunicorn.org/>）
      * README.md：系统说明文件
      * requirements.txt：第三方依赖（或称第三方库）的依赖文件，在联网条件下可通过 `pip install -r requirements.py` 从 PyPI —— Python 公共依赖库（ <https://pypi.org/> ）上下载并安装 Python 的依赖。
      * run.py：测试服务器，仅用于调试用，可忽略。

1. 计算层部分 （位于 `sz\_compute.tar.gz` 压缩包内）
   1. **config.py**
      * 计算层配置文件
      * 功能：用于配置需要访问的 Web 层接口的 http 访问地址
   2. **fm\_model.py**
      * 家庭群组建模程序
      * 功能：实现家庭群组模型搭建（包括需要按照特点算法生成样本及不需要生成样本两种方式）
   3. **clus\_model.py**
      * 宽带模型k-means聚类建模程序
      * 功能：通过聚类生成宽带模型负样本筛选规则
   4. **bb\_model.py**
      * 宽带模型建模程序
      * 功能：实现宽带模型及一般分类模型搭建。就宽带模型而言，若模型需要筛选负样本，则本程序会接收通过k-means聚类筛选出来的正负样本作为训练及测试数据；若不需要筛选负样本，则直接将输入数据作为训练及测试数据。
      * 注意：一般分类模型，应该输入一份标准数据（第一列为主键，最后一列为标签，中间各列为特征字段），并且选择不需要处理样本的分支。
   5. **bb\_pred.py**
      * 宽带模型调用程序
      * 功能：调用训练好的宽带模型对新数据进行预测。
      * 注意：宽带模型预测数据需与训练数据格式保持一致（字段数、字段排序均要求一致，最后一列标签字段可有可无）。
   6. **yw\_pred.py**
      * 异网宽带模型调用程序
      * 功能：对配置好的异网宽带模型（包括家庭模型及宽带模型两部分）进行调用，通过配置好的家庭模型及强关系阈值、弱关系阈值参数，进行家庭群组构建，并生成家庭数据；通过配置好的宽带模型数据及其输出阈值参数，进行宽带用户识别；最终将家庭数据与输出的宽带目标用户数据相融合，输出异网宽带目标用户清单。
      * 注意：若选择的家庭模型及其强关系阈值、弱关系阈值已经被调用过一次，则直接利用调用过的家庭结果，不需要重新构建家庭群组模型。
   7. **bb\_valid.py**
      * 宽带营销结果验证程序
      * 功能：实现宽带营销结果与输出营销清单的比对，计算营销的准确略，并输出结果。
      * 注意：输入的营销结果文件必须是只包含2个字段的csv文件，第一个字段为主键（通常为手机号码），第二个字段为营销结果（其意义应与宽带模型输出清单的意义一致），即是否宽带用户，‘是’则标记为1，否则标记为0.
   8. **yw\_valid.py**
      * 异网宽带营销结果验证程序
      * 功能：实现异网宽带营销结果与输出营销清单的比对，计算营销的准确略，并输出结果。
      * 注意：输入的营销结果文件必须是只包含2个字段的csv文件，第一个字段为主键（通常为手机号码），第二个字段为营销结果（其意义应与异网宽带模型输出清单的意义一致），即是否异网宽带用户，‘是’则标记为1，否则标记为0.

## 部署步骤

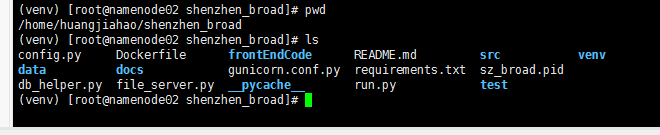
*（以下步骤均为在无外网访问条件下完成）*

1. Web 部分（以下操作没有对于用户身份以及权限的特殊要求）
   1. 解压缩 `sz\_broad.tar.gz` 压缩包，得到 Web 层的相关代码，并部署到服务器上的具体目录中（目前 Web 程序部署在 10.17.4.119 服务器的 `/home/lihongguo/app` 目录下，使用 `lihongguo` 用户运行程序）。
   2. 配置修改，需要根据具体情况修改 `config.py` 以及 `gunicorn.config.py` 文件的配置，如下为配置项说明：
      * **config.py** 文件配置
        + 说明：用于 Flask Web App 的配置
        + 可调整的配置项列表
          - **SECRET\_KEY** —— 系统密钥，用于 Web 系统登陆的 digest 验证。其格式为不定长的字符串，长度越长，字符越随机，则密钥安全性越高。
          - **SQLALCHEMY\_DATABASE\_URI** —— 数据库连接字符串，格式如下：
          - **COMPUTATION\_HOST** —— 计算层所在主机的 IP 地址
          - **COMPUTATION\_PORT** —— 计算层所在主机的 file\_server 暴露的端口，用于和 Web 层通信（具体的端口可以查看计算层的 `file\_server.py` 文件，目前使用 18088 端口）
      * **gunicorn.conf.py** 文件配置
        + 说明：用于 gunicorn 服务器的配置
        + 可调整的配置项列表
          - **bind** —— gunicorn 服务器绑定本机的端口地址配置
          - **workers** —— gunicorn 服务器使用的进程数目，一般和 CPU 的物理内核数一致达到最佳性能
          - **pidfile** —— gunicorn 服务器运行后将主进程的 PID 记录在这个配置指定的文件内
          - **accesslog** —— gunicorn 的 access 日志目录配置
   3. 在本地192.168.0.202服务器上**启动 gunicorn 服务器托管 Flask 应用程序：**

使用如下命令：gunicorn -t 60 src:app -c gunicorn.conf.py



**注**：后端代码路径:**/home/huangjiahao/shenzhen\_broad**



需进入202服务器的python虚拟环境才能运行上面gunicorn命令，虚拟环境在该目录下的venv文件夹下，虚拟环境启动命令：

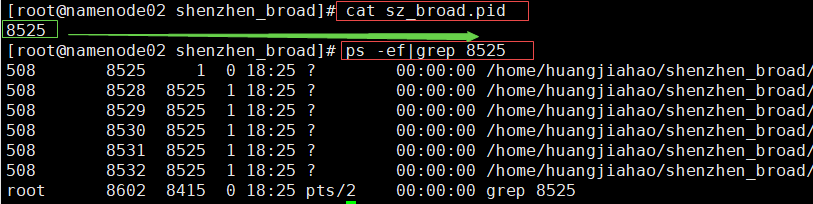
**source** **/home/huangjiahao/shenzhen\_broad/venv/bin/active**

**Pip install gunicorn**

退出虚拟环境，用**deactive**命令即可

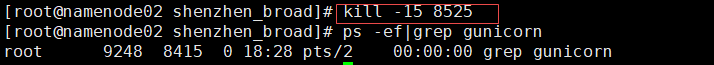
* 1. 查看 Flask Web 应用程序运行状态

配合使用 `cat` 和 `ps` 命令：



* 1. 如何终止 Web 程序运行

可以使用 `kill` 命令加上对应的进程 PID 终止程序的运行，如：

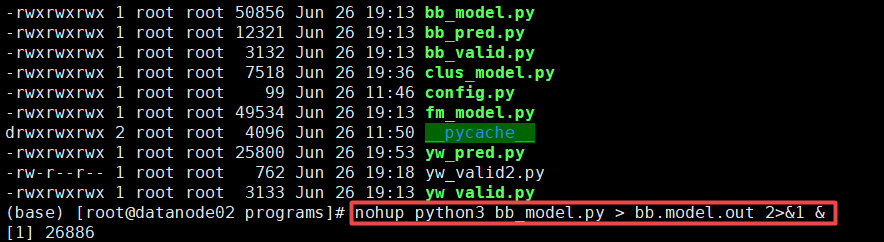


1. 计算层部分
   1. 解压缩 `sz\_compute.tar.gz` 压缩包，得到计算层的相关代码，并部署到服务器上的具体目录中（目前计算层程序部署在 10.17.2.48 服务器的 `/home/lihongguo/app` 目录下，使用 `lihongguo` 用户运行程序）。
   2. 修改配置文件 `config.py` ，指定 Web 服务器地址：目前为 `http://10.17.4.119:18088`
   3. 后台静默运行各计算层脚本程序，以 bb\_model.py 为例：

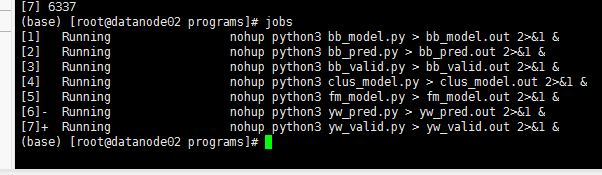
**nohup python3 bb\_model.py > bb\_model.out 2>&1 &**

**注：**本地服务器计算层存放在192.168.0.204服务器上，路径为：**/data/programs，**

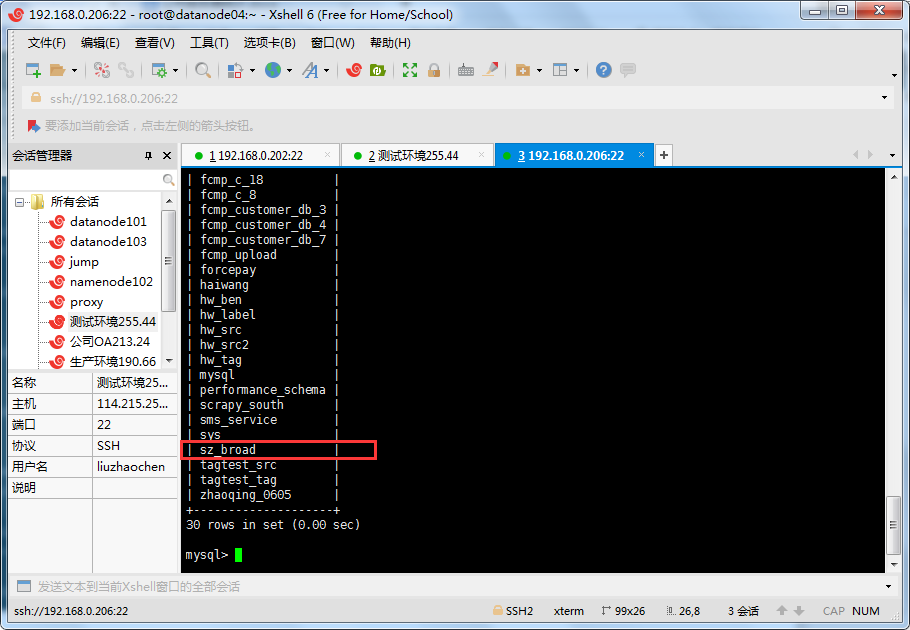
**需在204服务器中进入该目录再运行各python脚本**



**文件夹下面的都需要运行，204服务器后台服务最终如图所示：**

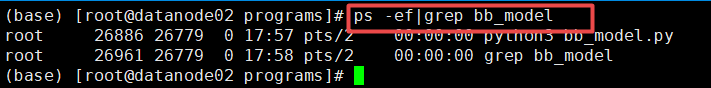


**数据库存放在192.168.0.206上，数据库名：sz\_broad**



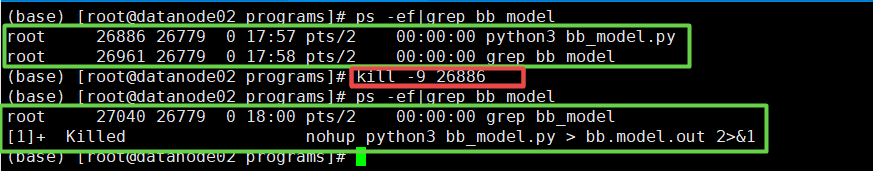
* 1. 查看程序运行状态

可以使用 `ps` 命令配合 `grep` 工具查看对应进程的运行状态，如：



* 1. 如何终止程序运行

可以使用 `kill` 命令加上对应的进程 PID 终止程序的运行，如：

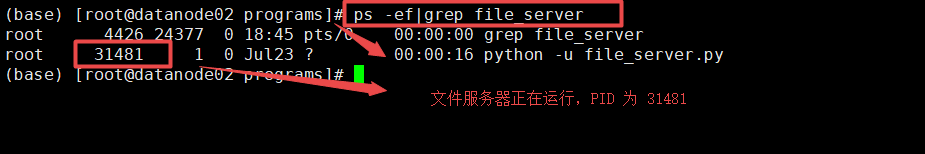


1. 文件服务器部分

一般来说文件服务器没有更新的必要，下面主要说明一下对于文件服务器的操作：

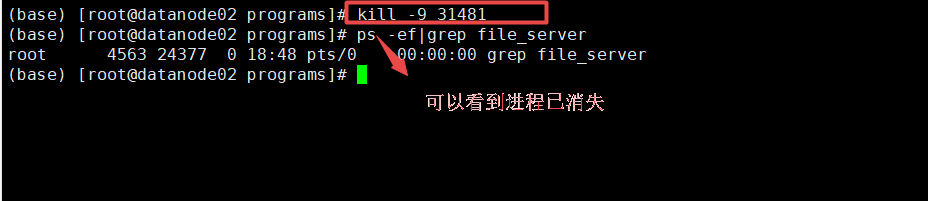
* 1. 查看文件服务器的运行，

可以使用 `ps` 命令配合 `grep` 工具查看对应进程的运行状态，如：



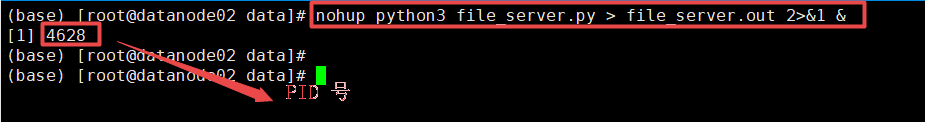
* 1. 停止文件服务器

可以使用 `kill` 命令加上对应进程 PID 终止程序的运行，如：



* 1. 启动文件服务器

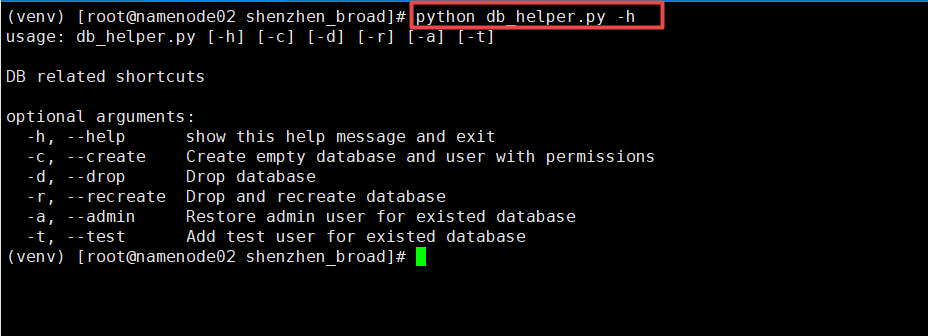
运行 ` nohup python3 file\_server.py > file\_server.out 2>&1 &` 命令运行文件服务器



* 1. 可以通过查看 **file\_server.out** 文件获取文件服务器日志

1. 补充说明（**db\_helper.py** 文件）

db\_helper.py 位于 ~/app 目录下，用于一些常见的数据库操作的封装，当敲入 `**python db\_helper.py -h**` 时可以看到帮助提示（注意 119 上需要使用 `python3` 命令强制使用 python3）



以下是各选项说明：

1. ***-c*** ：新建数据库表，注意数据库和数据表已存在的情况下会执行失败，重建内容仅包括数据库结构 schema，没有任何数据
2. ***-d*** ：删除数据库表，注意数据库和数据表不存在的时候会执行失败（**此操作会清楚 sz\_broad 数据库的一切内容，请谨慎操作，如有必要请先备份**）
3. ***-r*** ：重建数据库表并插入预置数据，此操作是 -d -c -a 操作的集成，会先删除数据库表，新建数据库表然后插入管理员账户
4. ***-a*** ：插入管理员用户，注意如果数据库表不存在会执行失败
5. ***-t*** ：插入测试用户，此选项可忽略（除非需要运行单元测试），注意如果数据库表不存在会执行失败

以上选项不能同时使用；其次运行 db\_helper 文件时要求同目录下有 config.py 文件同时没有任何连接数据库操作（也就意味着需要停止后台服务器，同时不能手工连接数据库）。

## 系统预置数据说明

1. 关于数据库
   1. 目前数据库使用版本为 5.7.23，以单机非集群模式部署在 10.17.4.119 机器上，使用默认端口 3306，配置文件为 `/etc/my.cnf` 。
   2. 数据库 root 用户密码为 root123 ，可以以系统 root 用户在本机登陆修改密码
   3. 异网识别平台所使用的数据库用户为 wislife\_app，密码为 wislife\_123 。具有且仅具有 sz\_broad 数据库的全部权限，仅允许本机登陆。
   4. 注意，通过 mysql 客户端访问数据库时候，请使用 `mysql -u root -h 127.0.0.1 -p` 命令登陆（由于 119 上无法识别 localhost 为 127.0.0.1，所以需要人工指定）
2. **关于平台系统登录**
   1. 系统平台管理员手机为 **15900000001** ，密码为 **ku0SdE** 。

本地服务器登录网址：**192.168.0.202:18088**

本地系统平台管理员**账号：15625050341  
密码：wislife\_123**

* 1. 使用管理员用户可以新建账户，其他人也可以注册用户，管理员审核通过即可登陆系统。

## 平台日志一览

对于系统运行时出现的问题，可以通过查看线上日志的形式进行排查，以下是各个部分的日志文件说明：

1. Web 层：Web 层的日志分为两块，一块是 gunicorn 的日志，一块是 Flask 应用本身的日志。前者粒度较大，提供的详细信息不多，用于排查网络环境问题；后者的粒度较少，出现问题时可以精确到具体报错的文件的具体函数具体行，但是由于日志量会比较庞大，需要定位以及配合相关 sed 或 awk 工具进行查阅。

目前的日志位置：

* gunicorn 日志：/home/lihongguo/app/data/logs/market\_gunicorn\_access.log
* Flask 日志：/home/lihongguo/app/data/logs/debug.log\* （由于每天的日志量都较大，因此Flask日志会在半夜自动备份前一天的日志为 debug.log.2019-xx-xx 的形式，因此 debug.log 记录的都为当天日志）

1. 计算层

计算层的处理信息目前是直接输出到标准输出的，根据我们部署计算层代码的方式，将标准输出的内容重定向到了各个 .out 文件当中，因此可以通过查阅各自的 .out 文件分析计算层的运行情况。