基于 nginx+uwsig 在 Centos6.5 上部署 django 项目

by:授客　QQ：1033553122

目录

测试环境

CentOS 6.5-x86_64

nginx-1.10.0
下载地址：http://nginx.org/en/download.html
下载地址：http://pan.baidu.com/s/1skNT7zv

pcre-8.36
下载地址：http://pan.baidu.com/s/1eSiN7ns

Python-3.5.4.tgz
下载地址：
https://www.python.org/ftp/python/3.5.4/Python-3.5.4.tgz

setuptools-40.8.0.zip
https://files.pythonhosted.org/packages/c2/f7/c7b501b783e5a74cf1768bc174ee4fb0a8a6ee5af6afa92274ff964703e0/setuptools-40.8.0.zip

Django-2.0.13.tar.gz
官方下载地址：
https://www.djangoproject.com/download/2.0.13/tarball/

channels-2.1.7.tar.gz
下载地址：
https://pypi.org/project/channels/#files
https://files.pythonhosted.org/packages/b9/a8/d4fef151a93b7c2f3ae66553db5a9d8cec41f7743e35a70f369b4c5a5800/channels-2.1.7.tar.gz

Twisted-18.9.0.tar.bz2
(channels 依赖包)
https://files.pythonhosted.org/packages/5d/0e/a72d85a55761c2c3ff1cb968143a2fd5f360220779ed90e0fadf4106d4f2/Twisted-18.9.0.tar.bz2

pytz-2018.9.tar.gz
https://files.pythonhosted.org/packages/af/be/6c59e30e208a5f28da85751b93ec7b97e4612268bb054d0dff396e758a90/pytz-2018.9.tar.gz

mysqlclient-1.3.14.tar.gz
https://files.pythonhosted.org/packages/f7/a2/1230ebbb4b91f42ad6b646e59eb8855559817ad5505d81c1ca2b5a216040/mysqlclient-1.3.14.tar.gz

redis-3.2.1.tar.gz
https://files.pythonhosted.org/packages/24/d4/06486dee0f66ef8c5080dc576fdfb33131fd2

e0be3747f2be4e5634088a2/redis-3.2.1.tar.gz

uwsgi-2.0.18.tar.gz
https://files.pythonhosted.org/packages/e7/1e/3dcca007f974fe4eb369bf1b8629d5e342bb3055e2001b2e5340aaefae7a/uwsgi-2.0.18.tar.gz

chardet-3.0.4.tar.gz
下载地址：https://pypi.org/project/chardet/#files
https://files.pythonhosted.org/packages/fc/bb/a5768c230f9ddb03acc9ef3f0d4a3cf93462473795d18e9535498c8f929d/chardet-3.0.4.tar.gz

mysql-connector-python-8.0.15.tar.gz
下载地址：https://dev.mysql.com/downloads/connector/python/
https://dev.mysql.com/downloads/file/?id=484801

supervisor-4.0.0.tar.gz
https://files.pythonhosted.org/packages/98/e2/f4b194c94a0f65e5f4618032e545371cfb81e8c3eafa9c58bfe413827e02/supervisor-4.0.0.tar.gz

安装 Python

```
# tar -xvzf Python-3.5.4.tgz
# cd Python-3.5.4
# ./configure --prefix=/usr/local/python35
# make&& make install

# 添加软链接
# ln -s /usr/local/python35/bin/python3 /usr/bin/python3

# 验证
# python3
Python 3.5.4 (default, Mar 11 2019, 14:11:36)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-23)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>exit()
#
```

说明：因系统而异，安装相关软件时可能需要预先安装 gcc-c++

安装 setuptools

```
# unzip setuptools-40.8.0.zip
# cd setuptools-40.8.0
# python3 setup.py install
```

安装 pytz

```
# unzip pytz-2018.9.tar.gz
# cd pytz-2018.9
# python3 setup.py install
```

说明：安装 pytz 主要是为了解决执行 uwsgi --ini uwsgi.ini 时报错的问题：

```
ImportError: No module named 'pytz'
```

安装 uwsgi

```
# tar -xzvf uwsgi-2.0.18.tar.gz
# cd uwsgi-2.0.18
# python3  setup.py install
……(略)
############## end of uWSGI configuration #############
total build time: 36 seconds
*** uWSGI is ready, launch it with /usr/local/python35/bin/uwsgi ***
running build
running build_py
running install_lib
copying build/lib/uwsgidecorators.py ->
/usr/local/python35/lib/python3.5/site-packages
byte-compiling /usr/local/python35/lib/python3.5/site-packages/uwsgidecorators.py to
uwsgidecorators.cpython-35.pyc
running install_egg_info
running egg_info
writing top-level names to uWSGI.egg-info/top_level.txt
writing dependency_links to uWSGI.egg-info/dependency_links.txt
writing uWSGI.egg-info/PKG-INFO
reading manifest file 'uWSGI.egg-info/SOURCES.txt'
writing manifest file 'uWSGI.egg-info/SOURCES.txt'
Copying uWSGI.egg-info to
/usr/local/python35/lib/python3.5/site-packages/uWSGI-2.0.18-py3.5.egg-info
running install_scripts
```

成功安装的话，会出现如上输出,告诉我们启动 uwsig 程序所在路径

```
# 添加软链接，方便执行命令
# ln -s /usr/local/python35/bin/uwsgi /usr/bin/uwsgi
```

# 测试

新建 test.py，文件位置任意，内容如下：

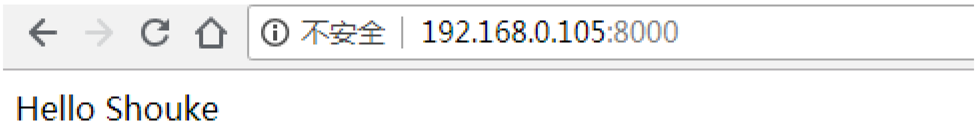
```
def application(env, start_response):
    start_response('200 OK', [('Content-Type','text/html')])
    return [b"Hello Shouke"]
```

# 启动 uwsgi

```
# uwsgi --http :8000 --master --wsgi-file test.py --process 4
*** Starting uWSGI 2.0.18 (64bit) on [Mon Mar 11 14:30:55 2019] ***
compiled with version: 4.4.7 20120313 (Red Hat 4.4.7-23) on 11 March 2019 06:25:45
os: Linux-2.6.32-754.11.1.el6.x86_64 #1 SMP Tue Feb 26 15:38:56 UTC 2019
nodename: localhost.localdomain
machine: x86_64
clock source: unix
detected number of CPU cores: 2
current working directory: /root
detected binary path: /usr/local/python35/bin/uwsgi
!!! no internal routing support, rebuild with pcre support !!!
uWSGI running as root, you can use --uid/--gid/--chroot options
*** WARNING: you are running uWSGI as root !!! (use the --uid flag) ***
your processes number limit is 7392
your memory page size is 4096 bytes
detected max file descriptor number: 1024
lock engine: pthread robust mutexes
thunder lock: disabled (you can enable it with --thunder-lock)
uWSGI http bound on :8000 fd 4
uwsgi socket 0 bound to TCP address 127.0.0.1:43555 (port auto-assigned) fd 3
uWSGI running as root, you can use --uid/--gid/--chroot options
*** WARNING: you are running uWSGI as root !!! (use the --uid flag) ***
Python version: 3.5.4 (default, Mar 11 2019, 14:11:36)  [GCC 4.4.7 20120313 (Red Hat 4.4.7-23)]
*** Python threads support is disabled. You can enable it with --enable-threads ***
Python main interpreter initialized at 0x235a570
uWSGI running as root, you can use --uid/--gid/--chroot options
*** WARNING: you are running uWSGI as root !!! (use the --uid flag) ***
your server socket listen backlog is limited to 100 connections
your mercy for graceful operations on workers is 60 seconds
mapped 364600 bytes (356 KB) for 4 cores
*** Operational MODE: preforking ***
WSGI app 0 (mountpoint='') ready in 0 seconds on interpreter 0x235a570 pid: 10262 (default app)
uWSGI running as root, you can use --uid/--gid/--chroot options
*** WARNING: you are running uWSGI as root !!! (use the --uid flag) ***
*** uWSGI is running in multiple interpreter mode ***
spawned uWSGI master process (pid: 10262)
spawned uWSGI worker 1 (pid: 10263, cores: 1)
spawned uWSGI worker 2 (pid: 10264, cores: 1)
spawned uWSGI worker 3 (pid: 10265, cores: 1)
spawned uWSGI worker 4 (pid: 10266, cores: 1)
spawned uWSGI http 1 (pid: 10267)
```

浏览器访问

http://192.168.0.105:8000/



Hello Shouke

说明：这里 192.168.0.105 为笔者测试服务器，即 uwsgi 启动时所在服务器

安装 Django

```
# tar -xvzf Django-2.0.13.tar.gz
# cd Django-2.0.13
# python3 setup.py install
```

安装 Twisted

```
# bzip2 -d  Twisted-18.9.0.tar.bz2
# tar -xvf Twisted-18.9.0.tar
# cd Twisted-18.9.0
# python3 setup.py install
```

安装 Channes

```
# tar -xvzf channels-2.1.7.tar.gz
# cd channels-2.1.7
# python3 setup.py install
```

安装 mysqlclient

```
# tar -xvzf mysqlclient-1.3.14.tar.gz
# cd mysqlclient-1.3.14
# python3 setup.py  install
```

备注：安装 mysqlclient 提示找不到 ssl 类库，如下：

```
/usr/bin/ld: cannot find -lssl
collect2: ld returned 1 exit status
error: command 'gcc' failed with exit status 1
```

-lxxx,这里的 xxx 即为类库的名称，解决方案：安装 opensll，如下

```
# yum install openssl-devel
```

安装 redis-3.2.1.tar.gz

```
# tar -xvzf redis-3.2.1.tar.gz
# cd redis-3.2.1
# python3 setup.py install
```

安装 chardet

```
# tar -xvzf chardet-3.0.4.tar.gz
# cd chardet-3.0.4
# python3 setup.py install
```

安装 mysql-connector-python

```
# tar -xvzf mysql-connector-python-8.0.15.tar.gz
# cd mysql-connector-python-8.0.15
# python3 setup.py  install
```

安装 supervisor

```
# tar -xvzf supervisor-4.0.0.tar.gz
# cd supervisor-4.0.0
# python3 setup.py install
```

安装 nginx

```
# tar -xzvf nginx-1.10.0.tar.gz
# cd nginx-1.10.0
# ./configure --prefix=/usr/local/ngnix --with-pcre=/mnt/pcre-8.36
Configuration summary
  + using PCRE library: /mnt/pcre-8.36
  + OpenSSL library is not used
  + using builtin md5 code
  + sha1 library is not found
  + using system zlib library

nginx path prefix: "/usr/local/ngnix"
nginx binary file: "/usr/local/ngnix/sbin/nginx"
nginx modules path: "/usr/local/ngnix/modules"
nginx configuration prefix: "/usr/local/ngnix/conf"
nginx configuration file: "/usr/local/ngnix/conf/nginx.conf"
nginx pid file: "/usr/local/ngnix/logs/nginx.pid"
nginx error log file: "/usr/local/ngnix/logs/error.log"
nginx http access log file: "/usr/local/ngnix/logs/access.log"
nginx http client request body temporary files: "client_body_temp"
nginx http proxy temporary files: "proxy_temp"
nginx http fastcgi temporary files: "fastcgi_temp"
nginx http uwsgi temporary files: "uwsgi_temp"
nginx http scgi temporary files: "scgi_temp"
```

注：

1、编译时，指定了 pcre 安装目录，但是安装出错，解决方法如上，指定源码所在目录

2、如果不指定--with-pcre 选项，会报类错

```
# make&& make install
```

略

修改 nginx 配置

编辑配置文件

```
# vim /usr/local/ngnix/conf/nginx.conf
```

找到“server”结点，修改

```
server {
listen       80;
        server_name  localhost;
        #charset koi8-r;

        #access_log  logs/host.access.log  main;

location / {
root   html;
index  index.html index.htm;
        }

        #error_page  404              /404.html;

        # redirect server error pages to the static page /50x.html
        #
        error_page   500 502 503 504  /50x.html;
location = /50x.html {
root   html;
        }
```

为

```
server {
listen      80;
        server_name  192.168.0.105;

charset utf-8;

access_log /usr/local/ngnix/logs/access.log;
        error_log /usr/local/ngnix/logs/error.log;

location / {
            uwsgi_send_timeout 60;        # 指定向 uWSGI 传送请求的超时时间(单位秒)。
            uwsgi_connect_timeout 60;     # 指定连接 uwsgi 的超时时间(单位秒)。
            uwsgi_read_timeout 600;       # 指定接收 uwsgi 应答的超时时间
include uwsgi_params;
            uwsgi_pass 127.0.0.1:8000;
        }
location /static {
expires 30d;
autoindex on;
add_header Cache-Control private;
            alias /opt/AutotestPlatform/static; # Django 项目静态文件路径
        }

        #error_page  404              /404.html;
        # redirect server error pages to the static page /50x.html
        #
        error_page   500 502 503 504  /50x.html;
location = /50x.html {
root   html;
        }
```

修改 Django 配置

IP 访问配置

修改应用的 settings.py(AutotestPlatform\AutotestPlatform\settings.py），编辑，找到 ALLOWED_HOSTS 修改为如下值

```
    ALLOWED_HOSTS = ['localhost','127.0.0.1','192.168.0.105']
```

数据库配置

修改应用的 settings.py(AutotestPlatform\AutotestPlatform\settings.py）

```
         DATABASES = {
                'default': {
                    'ENGINE': 'django.db.backends.mysql',
                     'NAME': 'testplatform',
                     'USER': 'testacc',
                     'PASSWORD': 'test1234',
                     'HOST': '192.168.1.103',
                     'PORT': '3306',
                     'OPTION':{
                       'init_command': 'SET default_storage_engine=INNODB'
         }
                }
            }
```

如上，设置带背景色部分的 key 值，分别表示数据库用户名，密码，服务器 ip，端口

APIRunner 配置

数据库配置

编辑 AutoestPlatform/website/apiRunner/conf/db.conf 文件

```
    [TESTPLATFORM]
    host = 10.118.52.33
    port = 3306
    user = testacc
    passwd = test1234
    db = testplatform
    charset = utf8
```

说明：这里配置的数据库，即为 Django 使用的数据库配置，如上，设置蓝色部分的 key 值，分别表示数据库服务器 ip，端口，用户名，密码，数据库名，字符集

https SLL、TSL 版本配置

编辑 AutoestPlatform/website/apiRunner/conf/https.conf 文件

```
    [HTTPS]
    SSL_OR_TLS_PROTOCOL = V2

    [README]
    #SSL_OR_TLS_PROTOCOL 可选值 V1、 V2、  V23、 V3，不区分大小写
    #V1  - TLSv1
    #V2  - SSLv2
    #V23 - SSLv23
    #V3  - SSLv3
```

系统环境变量配置

新建 PYTHONPATH 系统环境变量(如果不存在的话)，添加 Django 项目根目录绝对路径(例中为/opt/AutotestPlatform)到 PYTHONPATH 系统环境变量

上传 Django 项目文件

上传 AutotestPlatform.zip 到 uwsgi 服务器，并解压项目文件到/opt/下并解压

```
# unzip AutotestPlatform.zip
[root@localhost opt]# ls
AutotestPlatform  AutotestPlatform.zip
```

创建数据库

```
CREATE DATABASE IF NOT EXISTS `testplatform` DEFAULT CHARACTER SET utf8;
```

创建数据表

```
# cd /opt/AutotestPlatform
# python3 manage.py makemigrations website
Migrations for 'website':
  website/migrations/0001_initial.py
```

```
    - Create model API_case_tree
    - Create model API_case_tree_test_plan
    - Create model API_project_setting
    - Create model API_test_case_step
    - Create model API_test_plan
    - Create model API_test_report_for_case
    - Create model API_test_report_for_case_step
    - Create model API_test_report_for_summary
    - Create model Assertion_type_setting
    - Create model Browser_setting
    - Create model Database_setting
    - Create model Function_setting
    - Create model Global_variable_setting
    - Create model Navigation
    - Create model Operation_for_object
    - Create model Page_element
    - Create model Page_tree
    - Create model Project_chosen
    - Create model Promble_feedback
    - Create model Running_plan
    - Create model Sprint_tree
    - Create model Test_project_setting
    - Create model Test_task_detail
    - Create model Test_task_overview
    - Create model UI_case_tree
    - Create model UI_case_tree_test_plan
    - Create model UI_project_setting
    - Create model UI_test_case_step
    - Create model UI_test_plan
    - Create model UI_test_report_for_case
    - Create model UI_test_report_for_case_step
    - Create model UI_test_report_for_summary
    - Add field project to ui_case_tree
    - Add field project to sprint_tree
    - Add field project to page_tree
    - Add field page to page_element
     - Add field project to api_case_tree
```

```
# python3 manage.py migrate
```

略

执行初始化 sql

详情见“初始化 sql.sql”

创建 uwsgi 配置文件

在 Django 项目的根目录下（/opt/AutotestPlatform），也就是 manage.py 同级目录下，新建一个" uwsgi.ini "文件

说明：这里文件名可以自定义，uwsgi 支持多种类型的配置文件，如 xml，ini 等。此处，使用 ini 类型的配置，所以文件后缀名为.ini

uwsgi.ini 文件内容如下

```
[uwsgi]
# Django 项目根目录绝对路径
chdir = /opt/AutotestPlatform

# 指定项目的 wsgi 模块: projectName.wsgi
module = AutotestPlatform.wsgi

# wsgi.py 文件相对路径（相对于项目根目录
wsgi-file = AutotestPlatform/wsgi.py

# 设置项目外网访问 ip 和端口,端口号必须是未被占用,当与 Nginx 连接时使用 socket 选项(bind to the specified UNIX/TCP socket using default protocol)
# socket = ip:端口号例如 127.0.0.1:8000 ip 可以省略不写个人理解为 0 0.0.0.0
socket = :8000

# 设置项目外网访问 ip 和端口，端口号必须是未被占用，当 uwsgi 做 web 服务器时用 http 选项(add an http router/server on the specified address
#http = IP:Port  #外网 IP:端口，

# 启动一个 master 进程来管理其他进程
master = True

# 开启的进程数量（spawn the specified number of workers/processes 等同于 workers 选项
processes = 4

# 每个进程开启的线程数（run each worker in prethreaded mode with the specified number of threads
# 由于用 c 语言编写,因此不用担心 GIL 的问题，但 linux 上不存在线程，线程本质来讲是伪进程(且存在上下文切换成本)，因此不建议使用
#threads = 2
```

```
# 设置harakiri超时时间如果请求超过harakiri超时时间，中断该请求，并且立即重启处理这个请求的进程
harakiri = 120

# 守护进程的方式在后台运行uwsgi，增加web服务的稳定性，并将日志打到指定的日志文件、udp服务器，参数为日志文件的路径(实践检验发现，并没有生成AutotestPlatform.log日志文件
deamonize = /var/log/uwsgi/AutotestPlatform.log

# 设置内部缓冲区大小
# 设置单个请求的最大大小(包括请求体)，这个通常映射为请求头大小。默认为4k。如果需要接收更大的请求(比如一个大的cookie查询字符串)，可能需要增加该选项的大小
# 考虑安全问题，需要根据应用进行合理配置
buffer-size = 65535

# 打开 http body 缓冲，如果 HTTP body 的大小超过指定的限制,那么就保存到磁盘，如果开启了harakiri，则建议设置 post-buffering，否则如果上传比较慢，可能被服务器拒绝。(enable post buffering)
post-buffering = 8192

# 为每个工作进程设置请求数的上限。当一个工作进程处理的请求数达到这个值，那么该工作进程就会被重启(reload workers after the specified amount of managed requests
max-requests = 1000

# 创建pid文件(在失去权限前)(create pidfile (before privileges drop)
# 创建的pid文件可用于重启、停止服务:
# 停止服务：uwsgi --stop [pidfile路径]
pidfile = /var/log/uwsgi/uwsgi.pid

#当服务器退出的时候自动删除unix socket文件和pid文件
vacuum = True

# 设置单个日志文件大小，单位KB，用于切分日志文件
log-maxsize = 10240

# 在每一个日志行中都打印时间信息(prefix logs with date or a strftime string）log-date选项也有同样作用
logdate=true

#设置平滑重启(直到接收到的请求处理完才重启）一个工作子进程中，等待这个工作结束的最长秒数
# set the maximum time (in seconds) we wait for workers and other processes to die during reload/shutdown
reload-mercy    = 10
```

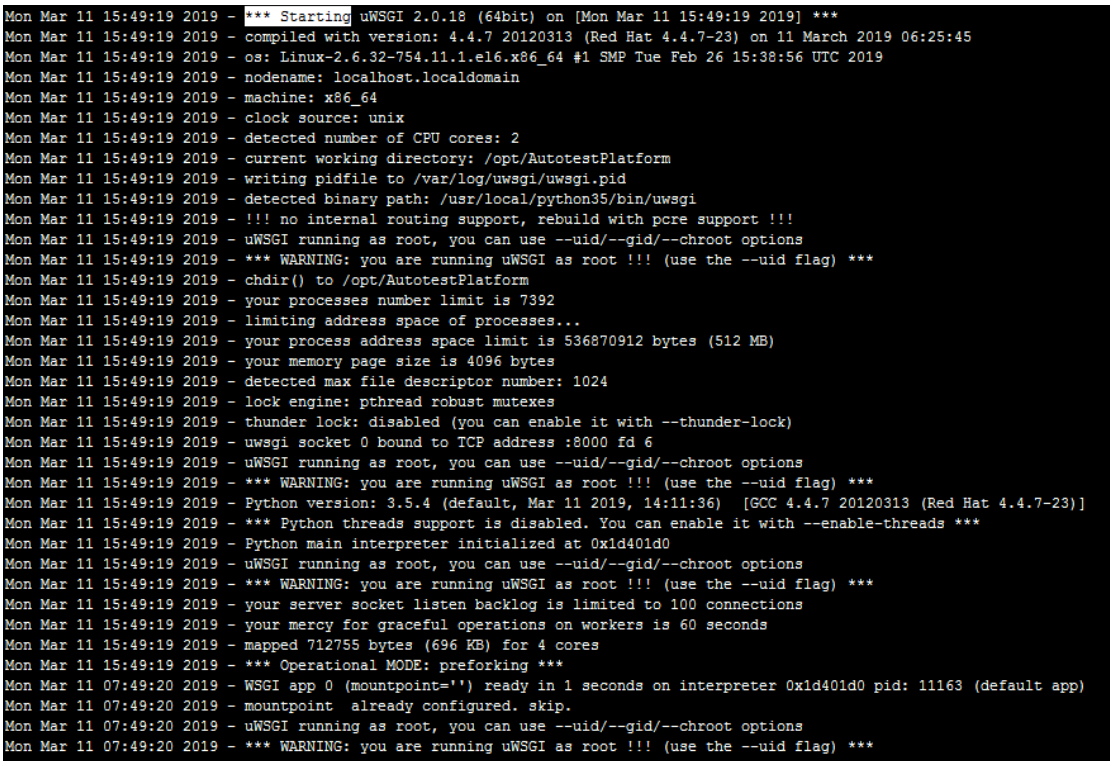
```
# 限制每个uWSGI进程的虚拟内存使用大小(limit processes address space/vsz) 例中为512M
limit-as        = 512

# 不记录请求日志,只记录错误以及uWSGI内部消息到日志中(disable request logging
disable-logging = true

# 将日志打到一个指定的文件(set logfile/udp address)
logto = /var/log/uwsgi/uwsgi.log
```

说明：以下操作输出均是在未添加disable-logging设置项的情况下操作的

启动uwsgi

```
#先为uwsig创建日志目录
[root@localhost log]# pwd
/var/log
[root@localhost log]# mkdir uwsgi

# cd /opt/AutotestPlatform
# uwsgi --ini uwsgi.ini
```

```
# tail -f /var/log/uwsgi/uwsgi.log
Mon Mar 11 15:49:19 2019 - *** Starting uWSGI 2.0.18 (64bit) on [Mon Mar 11 15:49:19 2019] ***
Mon Mar 11 15:49:19 2019 - compiled with version: 4.4.7 20120313 (Red Hat 4.4.7-23) on 11 March 2019 06:25:45
Mon Mar 11 15:49:19 2019 - os: Linux-2.6.32-754.11.1.el6.x86_64 #1 SMP Tue Feb 26 15:38:56 UTC 2019
Mon Mar 11 15:49:19 2019 - nodename: localhost.localdomain
Mon Mar 11 15:49:19 2019 - machine: x86_64
Mon Mar 11 15:49:19 2019 - clock source: unix
Mon Mar 11 15:49:19 2019 - detected number of CPU cores: 2
Mon Mar 11 15:49:19 2019 - current working directory: /opt/AutotestPlatform
Mon Mar 11 15:49:19 2019 - writing pidfile to /var/log/uwsgi/uwsgi.pid
Mon Mar 11 15:49:19 2019 - detected binary path: /usr/local/python35/bin/uwsgi
Mon Mar 11 15:49:19 2019 - !!! no internal routing support, rebuild with pcre support !!!
Mon Mar 11 15:49:19 2019 - uWSGI running as root, you can use --uid/--gid/--chroot options
Mon Mar 11 15:49:19 2019 - *** WARNING: you are running uWSGI as root !!! (use the --uid flag) ***
Mon Mar 11 15:49:19 2019 - chdir() to /opt/AutotestPlatform
Mon Mar 11 15:49:19 2019 - your processes number limit is 7392
Mon Mar 11 15:49:19 2019 - limiting address space of processes...
Mon Mar 11 15:49:19 2019 - your process address space limit is 536870912 bytes (512 MB)
Mon Mar 11 15:49:19 2019 - your memory page size is 4096 bytes
Mon Mar 11 15:49:19 2019 - detected max file descriptor number: 1024
Mon Mar 11 15:49:19 2019 - lock engine: pthread robust mutexes
Mon Mar 11 15:49:19 2019 - thunder lock: disabled (you can enable it with --thunder-lock)
Mon Mar 11 15:49:19 2019 - uwsgi socket 0 bound to TCP address :8000 fd 6
Mon Mar 11 15:49:19 2019 - uWSGI running as root, you can use --uid/--gid/--chroot options
Mon Mar 11 15:49:19 2019 - *** WARNING: you are running uWSGI as root !!! (use the --uid flag) ***
Mon Mar 11 15:49:19 2019 - Python version: 3.5.4 (default, Mar 11 2019, 14:11:36)  [GCC 4.4.7 20120313 (Red Hat 4.4.7-23)]
Mon Mar 11 15:49:19 2019 - *** Python threads support is disabled. You can enable it with --enable-threads ***
Mon Mar 11 15:49:19 2019 - Python main interpreter initialized at 0x1d401d0
Mon Mar 11 15:49:19 2019 - uWSGI running as root, you can use --uid/--gid/--chroot options
Mon Mar 11 15:49:19 2019 - *** WARNING: you are running uWSGI as root !!! (use the --uid flag) ***
Mon Mar 11 15:49:19 2019 - your server socket listen backlog is limited to 100 connections
Mon Mar 11 15:49:19 2019 - your mercy for graceful operations on workers is 60 seconds
Mon Mar 11 15:49:19 2019 - mapped 712755 bytes (696 KB) for 4 cores
Mon Mar 11 15:49:19 2019 - *** Operational MODE: preforking ***
Mon Mar 11 07:49:20 2019 - WSGI app 0 (mountpoint='') ready in 1 seconds on interpreter 0x1d401d0 pid: 11163 (default app)
Mon Mar 11 07:49:20 2019 - mountpoint  already configured. skip.
Mon Mar 11 07:49:20 2019 - uWSGI running as root, you can use --uid/--gid/--chroot options
Mon Mar 11 07:49:20 2019 - *** WARNING: you are running uWSGI as root !!! (use the --uid flag) ***
Mon Mar 11 07:49:20 2019 - *** uWSGI is running in multiple interpreter mode ***
Mon Mar 11 07:49:20 2019 - spawned uWSGI master process (pid: 11163)
Mon Mar 11 07:49:20 2019 - spawned uWSGI worker 1 (pid: 11165, cores: 1)
Mon Mar 11 07:49:20 2019 - spawned uWSGI worker 2 (pid: 11166, cores: 1)
Mon Mar 11 07:49:20 2019 - spawned uWSGI worker 3 (pid: 11167, cores: 1)
Mon Mar 11 07:49:20 2019 - spawned uWSGI worker 4 (pid: 11168, cores: 1)
```



```
Mon Mar 11 15:49:19 2019 - *** Starting uWSGI 2.0.18 (64bit) on [Mon Mar 11 15:49:19 2019] ***
Mon Mar 11 15:49:19 2019 - compiled with version: 4.4.7 20120313 (Red Hat 4.4.7-23) on 11 March 2019 06:25:45
Mon Mar 11 15:49:19 2019 - os: Linux-2.6.32-754.11.1.el6.x86_64 #1 SMP Tue Feb 26 15:38:56 UTC 2019
Mon Mar 11 15:49:19 2019 - nodename: localhost.localdomain
Mon Mar 11 15:49:19 2019 - machine: x86_64
Mon Mar 11 15:49:19 2019 - clock source: unix
Mon Mar 11 15:49:19 2019 - detected number of CPU cores: 2
Mon Mar 11 15:49:19 2019 - current working directory: /opt/AutotestPlatform
Mon Mar 11 15:49:19 2019 - writing pidfile to /var/log/uwsgi/uwsgi.pid
Mon Mar 11 15:49:19 2019 - detected binary path: /usr/local/python35/bin/uwsgi
Mon Mar 11 15:49:19 2019 - !!! no internal routing support, rebuild with pcre support !!!
Mon Mar 11 15:49:19 2019 - uWSGI running as root, you can use --uid/--gid/--chroot options
Mon Mar 11 15:49:19 2019 - *** WARNING: you are running uWSGI as root !!! (use the --uid flag) ***
Mon Mar 11 15:49:19 2019 - chdir() to /opt/AutotestPlatform
Mon Mar 11 15:49:19 2019 - your processes number limit is 7392
Mon Mar 11 15:49:19 2019 - limiting address space of processes...
Mon Mar 11 15:49:19 2019 - your process address space limit is 536870912 bytes (512 MB)
Mon Mar 11 15:49:19 2019 - your memory page size is 4096 bytes
Mon Mar 11 15:49:19 2019 - detected max file descriptor number: 1024
Mon Mar 11 15:49:19 2019 - lock engine: pthread robust mutexes
Mon Mar 11 15:49:19 2019 - thunder lock: disabled (you can enable it with --thunder-lock)
Mon Mar 11 15:49:19 2019 - uwsgi socket 0 bound to TCP address :8000 fd 6
Mon Mar 11 15:49:19 2019 - uWSGI running as root, you can use --uid/--gid/--chroot options
Mon Mar 11 15:49:19 2019 - *** WARNING: you are running uWSGI as root !!! (use the --uid flag) ***
Mon Mar 11 15:49:19 2019 - Python version: 3.5.4 (default, Mar 11 2019, 14:11:36)  [GCC 4.4.7 20120313 (Red Hat 4.4.7-23)]
Mon Mar 11 15:49:19 2019 - *** Python threads support is disabled. You can enable it with --enable-threads ***
Mon Mar 11 15:49:19 2019 - Python main interpreter initialized at 0x1d401d0
Mon Mar 11 15:49:19 2019 - uWSGI running as root, you can use --uid/--gid/--chroot options
Mon Mar 11 15:49:19 2019 - *** WARNING: you are running uWSGI as root !!! (use the --uid flag) ***
Mon Mar 11 15:49:19 2019 - your server socket listen backlog is limited to 100 connections
Mon Mar 11 15:49:19 2019 - your mercy for graceful operations on workers is 60 seconds
Mon Mar 11 15:49:19 2019 - mapped 712755 bytes (696 KB) for 4 cores
Mon Mar 11 15:49:19 2019 - *** Operational MODE: preforking ***
Mon Mar 11 07:49:20 2019 - WSGI app 0 (mountpoint='') ready in 1 seconds on interpreter 0x1d401d0 pid: 11163 (default app)
Mon Mar 11 07:49:20 2019 - mountpoint  already configured. skip.
Mon Mar 11 07:49:20 2019 - uWSGI running as root, you can use --uid/--gid/--chroot options
Mon Mar 11 07:49:20 2019 - *** WARNING: you are running uWSGI as root !!! (use the --uid flag) ***
```

如上，正常运行

启动 nginx

```
# /usr/local/ngnix/sbin/nginx -c /usr/local/ngnix/conf/nginx.conf
```

备注：

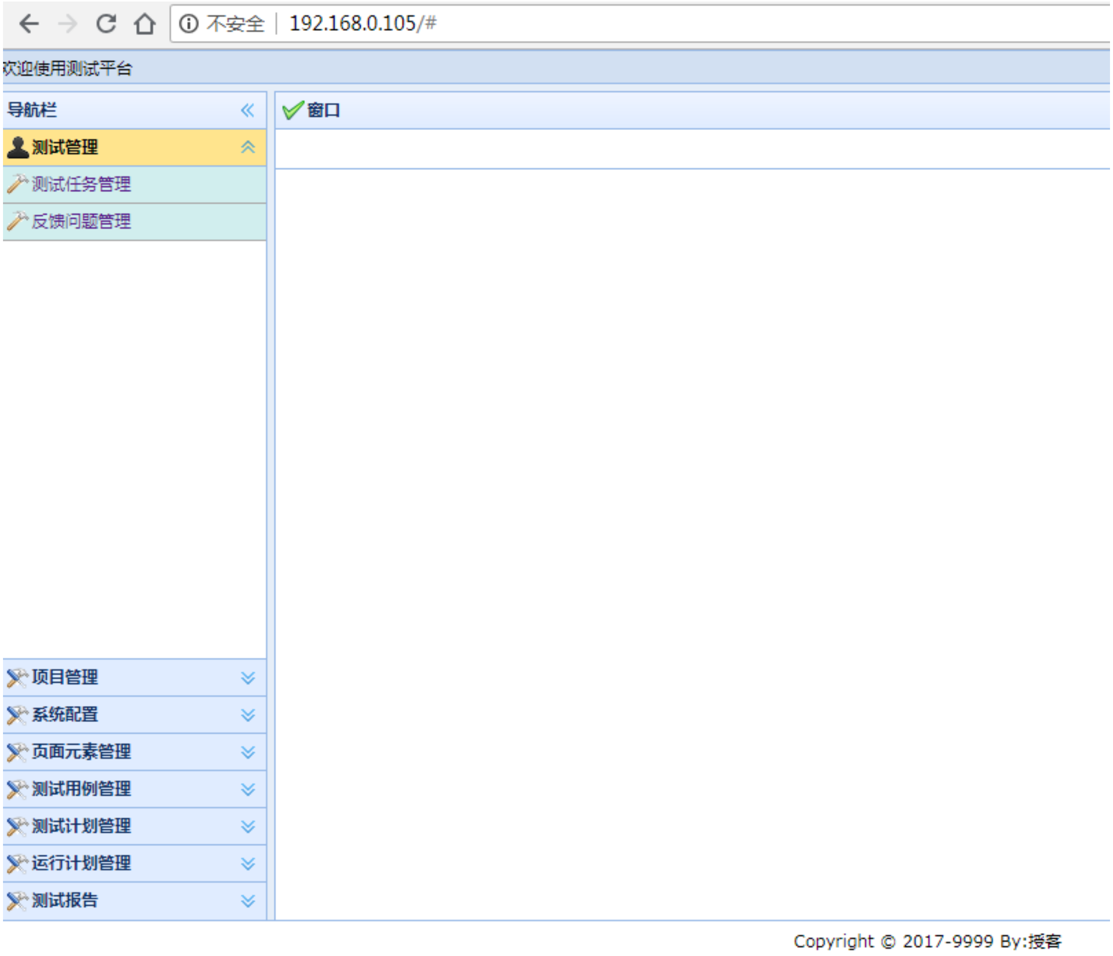
停止

```
/usr/local/ngnix/sbin/nginx -s stop
```

重新加载(不关闭的情况下使用)：

```
/usr/local/ngnix/sbin/nginx -s reload
```

浏览器访问



验证 daphne 可否正常运行

说明：**安装 Channes 过程中，如果不需要人为介入，程序自动顺利完成安装的情况下，默认会安装 daphne** 依赖软件包--ASGI HTTP/WebSocket 服务器，daphne 默认安装目录：$PYTHON_HOME/bin/daphne

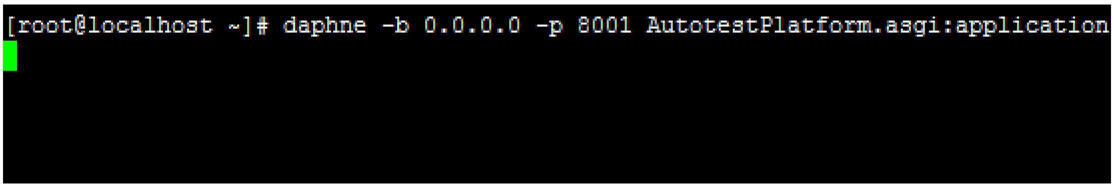
添加软连接

```
# ln -s /usr/local/python35/bin/daphne /usr/bin/daphne
```

启动 daphne

```
# daphne -b 0.0.0.0 -p 8001 AutotestPlatform.asgi:application
```

注意：如果启动失败，程序会自动退出命令，没有任何提示,如下,否则也不会有提示，但是不会自动退出命令



```
[root@localhost ~]# daphne -b 0.0.0.0 -p 8001 AutotestPlatform.asgi:application
```

运行成功后，按Ctrl + C键，停止daphne进程

创建supervisor配置文件

添加软连接

```
# ln -s /usr/local/python35/bin/echo_supervisord_conf /usr/bin/echo_supervisord_conf
```

说明：安装完supervisor后，会自动在$PYTHON_HOME/bin/目录下生成echo_supervisord_conf

```
# echo_supervisord_conf
```

说明：如果执行成功会输出如下配置

```
; Sample supervisor config file.
;
; For more information on the config file, please see:
; http://supervisord.org/configuration.html
;
; Notes:
;  - Shell expansion ("~" or "$HOME") is not supported.  Environment
;    variables can be expanded using this syntax: "%(ENV_HOME)s".
;  - Quotes around values are not supported, except in the case of
;    the environment= options as shown below.
;  - Comments must have a leading space: "a=b ;comment" not "a=b;comment".
;  - Command will be truncated if it looks like a config file comment, e.g.
;    "command=bash -c 'foo ; bar'" will truncate to "command=bash -c 'foo ".

[unix_http_server]
file=/tmp/supervisor.sock   ; the path to the socket file
;chmod=0700                 ; socket file mode (default 0700)
;chown=nobody:nogroup       ; socket file uid:gid owner
;username=user              ; default is no username (open server)
;password=123               ; default is no password (open server)

;[inet_http_server]         ; inet (TCP) server disabled by default
;port=127.0.0.1:9001        ; ip_address:port specifier, *:port for all iface
;username=user              ; default is no username (open server)
;password=123               ; default is no password (open server)

[supervisord]
logfile=/tmp/supervisord.log ; main log file; default $CWD/supervisord.log
logfile_maxbytes=50MB        ; max main logfile bytes b4 rotation; default 50MB
logfile_backups=10           ; # of main logfile backups; 0 means none, default 10
loglevel=info                ; log level; default info; others: debug,warn,trace
pidfile=/tmp/supervisord.pid ; supervisord pidfile; default supervisord.pid
nodaemon=false               ; start in foreground if true; default false
minfds=1024                  ; min. avail startup file descriptors; default 1024
```

```
minprocs=200                 ; min. avail process descriptors;default 200
;umask=022                   ; process file creation umask; default 022
;user=supervisord            ; setuid to this UNIX account at startup; recommended if root
;identifier=supervisor       ; supervisord identifier, default is 'supervisor'
;directory=/tmp              ; default is not to cd during start
;nocleanup=true              ; don't clean up tempfiles at start; default false
;childlogdir=/tmp            ; 'AUTO' child log dir, default $TEMP
;environment=KEY="value"     ; key value pairs to add to environment
;strip_ansi=false            ; strip ansi escape codes in logs; def. false

; The rpcinterface:supervisor section must remain in the config file for
; RPC (supervisorctl/web interface) to work.  Additional interfaces may be
; added by defining them in separate [rpcinterface:x] sections.

[rpcinterface:supervisor]
supervisor.rpcinterface_factory = supervisor.rpcinterface:make_main_rpcinterface

; The supervisorctl section configures how supervisorctl will connect to
; supervisord.  configure it match the settings in either the unix_http_server
; or inet_http_server section.

[supervisorctl]
serverurl=unix:///tmp/supervisor.sock ; use a unix:// URL  for a unix socket
;serverurl=http://127.0.0.1:9001 ; use an http:// url to specify an inet socket
;username=chris              ; should be same as in [*_http_server] if set
;password=123                ; should be same as in [*_http_server] if set
;prompt=mysupervisor         ; cmd line prompt (default "supervisor")
;history_file=~/.sc_history  ; use readline history if available

; The sample program section below shows all possible program subsection values.
; Create one or more 'real' program: sections to be able to control them under
; supervisor.

;[program:theprogramname]
;command=/bin/cat              ; the program (relative uses PATH, can take args)
;process_name=%(program_name)s ; process_name expr (default %(program_name)s)
;numprocs=1                    ; number of processes copies to start (def 1)
;directory=/tmp                ; directory to cwd to before exec (def no cwd)
;umask=022                     ; umask for process (default None)
;priority=999                  ; the relative start priority (default 999)
;autostart=true                ; start at supervisord start (default: true)
;startsecs=1                   ; # of secs prog must stay up to be running (def. 1)
;startretries=3                ; max # of serial start failures when starting (default
3)
```

```
;autorestart=unexpected          ; when to restart if exited after running (def:
unexpected)
;exitcodes=0                    ; 'expected' exit codes used with autorestart (default 0)
;stopsignal=QUIT                ; signal used to kill process (default TERM)
;stopwaitsecs=10                ; max num secs to wait b4 SIGKILL (default 10)
;stopasgroup=false              ; send stop signal to the UNIX process group (default
false)
;killasgroup=false              ; SIGKILL the UNIX process group (def false)
;user=chrism                    ; setuid to this UNIX account to run the program
;redirect_stderr=true           ; redirect proc stderr to stdout (default false)
;stdout_logfile=/a/path         ; stdout log path, NONE for none; default AUTO
;stdout_logfile_maxbytes=1MB    ; max # logfile bytes b4 rotation (default 50MB)
;stdout_logfile_backups=10      ; # of stdout logfile backups (0 means none, default 10)
;stdout_capture_maxbytes=1MB    ; number of bytes in 'capturemode' (default 0)
;stdout_events_enabled=false    ; emit events on stdout writes (default false)
;stdout_syslog=false            ; send stdout to syslog with process name (default false)
;stderr_logfile=/a/path         ; stderr log path, NONE for none; default AUTO
;stderr_logfile_maxbytes=1MB    ; max # logfile bytes b4 rotation (default 50MB)
;stderr_logfile_backups=10      ; # of stderr logfile backups (0 means none, default 10)
;stderr_capture_maxbytes=1MB    ; number of bytes in 'capturemode' (default 0)
;stderr_events_enabled=false    ; emit events on stderr writes (default false)
;stderr_syslog=false            ; send stderr to syslog with process name (default false)
;environment=A="1",B="2"        ; process environment additions (def no adds)
;serverurl=AUTO                 ; override serverurl computation (childutils)

; The sample eventlistener section below shows all possible eventlistener
; subsection values.  Create one or more 'real' eventlistener: sections to be
; able to handle event notifications sent by supervisord.

;[eventlistener:theeventlistenername]
;command=/bin/eventlistener     ; the program (relative uses PATH, can take args)
;process_name=%(program_name)s ; process_name expr (default %(program_name)s)
;numprocs=1                     ; number of processes copies to start (def 1)
;events=EVENT                   ; event notif. types to subscribe to (req'd)
;buffer_size=10                 ; event buffer queue size (default 10)
;directory=/tmp                 ; directory to cwd to before exec (def no cwd)
;umask=022                      ; umask for process (default None)
;priority=-1                    ; the relative start priority (default -1)
;autostart=true                 ; start at supervisord start (default: true)
;startsecs=1                    ; # of secs prog must stay up to be running (def. 1)
;startretries=3                 ; max # of serial start failures when starting (default
3)
;autorestart=unexpected         ; autorestart if exited after running (def: unexpected)
;exitcodes=0                    ; 'expected' exit codes used with autorestart (default 0)
```

```
;stopsignal=QUIT               ; signal used to kill process (default TERM)
;stopwaitsecs=10               ; max num secs to wait b4 SIGKILL (default 10)
;stopasgroup=false             ; send stop signal to the UNIX process group (default false)
;killasgroup=false             ; SIGKILL the UNIX process group (def false)
;user=chrism                   ; setuid to this UNIX account to run the program
;redirect_stderr=false         ; redirect_stderr=true is not allowed for eventlisteners
;stdout_logfile=/a/path        ; stdout log path, NONE for none; default AUTO
;stdout_logfile_maxbytes=1MB   ; max # logfile bytes b4 rotation (default 50MB)
;stdout_logfile_backups=10     ; # of stdout logfile backups (0 means none, default 10)
;stdout_events_enabled=false   ; emit events on stdout writes (default false)
;stdout_syslog=false           ; send stdout to syslog with process name (default false)
;stderr_logfile=/a/path        ; stderr log path, NONE for none; default AUTO
;stderr_logfile_maxbytes=1MB   ; max # logfile bytes b4 rotation (default 50MB)
;stderr_logfile_backups=10     ; # of stderr logfile backups (0 means none, default 10)
;stderr_events_enabled=false   ; emit events on stderr writes (default false)
;stderr_syslog=false           ; send stderr to syslog with process name (default false)
;environment=A="1",B="2"       ; process environment additions
;serverurl=AUTO                ; override serverurl computation (childutils)

; The sample group section below shows all possible group values.  Create one
; or more 'real' group: sections to create "heterogeneous" process groups.

;[group:thegroupname]
;programs=progname1,progname2  ; each refers to 'x' in [program:x] definitions
;priority=999                  ; the relative start priority (default 999)

; The [include] section can just contain the "files" setting.  This
; setting can list multiple files (separated by whitespace or
; newlines).  It can also contain wildcards.  The filenames are
; interpreted as relative to this file.  Included files *cannot*
; include files themselves.

;[include]
;files = relative/directory/*.ini
```

生成 supervisord.conf 配置文件，存放在/etc 目录下

```
# echo_supervisord_conf > /etc/supervisord.conf
```

```
###为了不将所有新增配置信息全写在一个配置文件里，新建一个文件夹，为每个程序创建一个独立的配置文件，相互隔离
# mkdir -p /etc/supervisor/conf.d/
```

修改 supervisord.conf 配置文件，

修改

```
file=/tmp/supervisor.sock;
```

为

```
file=/var/run/supervisor.sock;
```

注意：如果不进行修改，运行 supervisord 会报错：unix:///tmp/supervisor.sock no such file（因为 tmp 目录会自动清理导致无法使用 supervisorctl 命令，下同，不再赘述）

修改

```
pidfile=/tmp/supervisord.pid ;
```

为

```
pidfile=/var/run/supervisord.pid ;
```

修改

```
logfile=/tmp/supervisord.log ; main log file;
```

为

```
logfile=/var/log/supervisord/supervisord.log ;
```

修改

```
serverurl=unix:///tmp/supervisor.sock ;
```

为

```
serverurl=unix:///var/run/supervisor.sock ;
```

修改

```
;[include]
;files = relative/directory/*.ini
```

为

```
[include]
files = /etc/supervisor/conf.d/*.ini
```

修改完成后，需要创建/var/run/supervisor.sock 文件，不然运行 supervisord 还是会报错：unix:///var/run/supervisor.sock no such file

```
# touch /var/run/supervisor.sock
# chmod 755 /var/run/supervisor.sock
```

并且还需要创建/var/log/supervisord/目录，会报错：rror: The directory named as part of the path /var/log/supervisord/supervisord.log does not exist

```
# mkdir -p /var/log/supervisord/
```

创建 daphne.ini 配置文件

```
# cd /etc/supervisor/conf.d/
# vim daphne.ini # 创建文件
[program:daphne]
# 项目根目录绝对路径
directory=/opt/AutotestPlatform

command=daphne -b 0.0.0.0 -p 8001 --access-log - --proxy-headers AutotestPlatform.asgi:application

# 需要启动的进程数（设置为服务器逻辑 CPU 数量
numprocs=4

# 给每个进程一个唯一的名字，用于区分进程
process_name=asgi%(process_num)d

# 自动启动和恢复进程
autostart=true
autorestart=true

# 设置进程日志存放路径
stdout_logfile=/var/log/asgi.log

# 重定向标准错误
redirect_stderr=true
```

运行 supervisord

添加软连接

```
# ln -s /usr/local/python35/bin/supervisord /usr/bin/supervisord
```

说明：supervisor 安装完成后，默认会在$PYTHON_HOME/bin/目录下生成 supervisord 和 supervisorctl

运行 supervisord

```
# supervisord -n -c /etc/supervisord.conf
2019-03-12 19:23:02,062 CRIT Supervisor is running as root.  Privileges were not dropped because no user is specified in the config file.  If you intend to run as root, you can set user=root in the config file to avoid this message.
2019-03-12 19:23:02,063 INFO Included extra file "/etc/supervisor/conf.d/daphne.ini" during parsing
2019-03-12 19:23:02,086 INFO RPC interface 'supervisor' initialized
2019-03-12 19:23:02,086 CRIT Server 'unix_http_server' running without any HTTP
```

```
authentication checking
2019-03-12 19:23:02,088 INFO supervisord started with pid 17971
2019-03-12 19:23:03,109 INFO spawned: 'asgi0' with pid 17974
2019-03-12 19:23:03,124 INFO spawned: 'asgi3' with pid 17975
2019-03-12 19:23:03,148 INFO spawned: 'asgi1' with pid 17976
2019-03-12 19:23:03,161 INFO spawned: 'asgi2' with pid 17977
2019-03-12 19:23:04,168 INFO success: asgi0 entered RUNNING state, process has stayed
up for > than 1 seconds (startsecs)
2019-03-12 19:23:04,168 INFO success: asgi3 entered RUNNING state, process has stayed
up for > than 1 seconds (startsecs)
2019-03-12 19:23:04,168 INFO success: asgi1 entered RUNNING state, process has stayed
up for > than 1 seconds (startsecs)
2019-03-12 19:23:04,168 INFO success: asgi2 entered RUNNING state, process has stayed
up for > than 1 seconds (startsecs)
```

参数说明：

-n 前台运行作为调试用，调试好后采用不携带该参数的方式运行，上如下：

```
# supervisord -c /etc/supervisord.conf
```

-c 配置文件

如上，运行 supervisord 后会自动启动 daphne 服务，不需要手动的方式通过 supervisortctl start 进程名

修改 nginx 配置

编辑配置文件，新增以下带背景色内容

```
upstream channels-backend {
server 127.0.0.1:8001;
    }

server {
listen      80;
       server_name  192.168.0.105;

charset utf-8;

       access_log  /usr/local/ngnix/logs/access.log;
       error_log /usr/local/ngnix/logs/error.log;

location /ws {
           proxy_pass http://channels-backend;
```

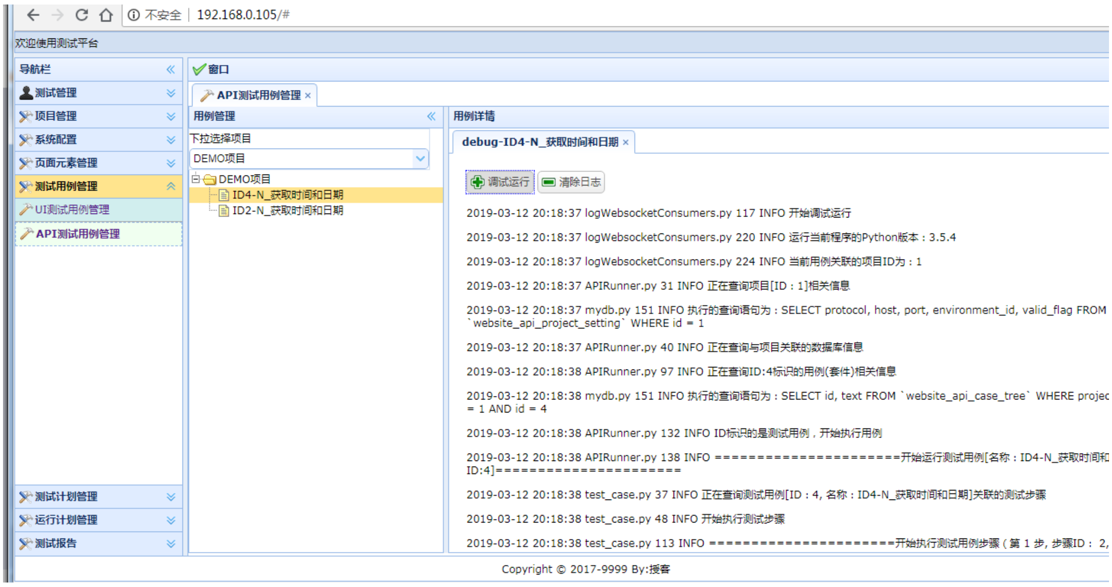
```
          proxy_http_version 1.1;
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection "upgrade";

proxy_redirect off;
proxy_set_header Host $host;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header X-Forwarded-Host $server_name;
       }
location / {
          uwsgi_send_timeout 600s;       # 指定向 uWSGI 传送请求的超时时间(单位秒)。
          uwsgi_connect_timeout 600s;     # 指定连接 uwsgi 的超时时间(单位秒)。
          uwsgi_read_timeout 600s;       # 指定接收 uwsgi 应答的超时时间
include uwsgi_params;
          uwsgi_pass 127.0.0.1:8000;
       }
location /static {
expires 30d;
autoindex on;
add_header Cache-Control private;
          alias /opt/AutotestPlatform/static; # Django 项目的静态文件
       }
```

重启 nginx

```
# /usr/local/ngnix/sbin/nginx -s reload
```

验证

新增 API 项目，新增接口用例，可在线调试运行，如下，则说明成功



```
2019-03-12 20:18:37 logWebsocketConsumers.py 117 INFO 开始调试运行

2019-03-12 20:18:37 logWebsocketConsumers.py 220 INFO 运行当前程序的Python版本：3.5.4

2019-03-12 20:18:37 logWebsocketConsumers.py 224 INFO 当前用例关联的项目ID为：1

2019-03-12 20:18:37 APIRunner.py 31 INFO 正在查询项目[ID：1]相关信息

2019-03-12 20:18:37 mydb.py 151 INFO 执行的查询语句为：SELECT protocol, host, port, environment_id, valid_flag FROM
`website_api_project_setting` WHERE id = 1

2019-03-12 20:18:37 APIRunner.py 40 INFO 正在查询与项目关联的数据库信息

2019-03-12 20:18:38 APIRunner.py 97 INFO 正在查询ID:4标识的用例(套件)相关信息

2019-03-12 20:18:38 mydb.py 151 INFO 执行的查询语句为：SELECT id, text FROM `website_api_case_tree` WHERE projec
= 1 AND id = 4

2019-03-12 20:18:38 APIRunner.py 132 INFO ID标识的是测试用例，开始执行用例

2019-03-12 20:18:38 APIRunner.py 138 INFO ======================开始运行测试用例[名称：ID4-N_获取时间和
ID:4]======================

2019-03-12 20:18:38 test_case.py 37 INFO 正在查询测试用例[ID：4, 名称：ID4-N_获取时间和日期]关联的测试步骤

2019-03-12 20:18:38 test_case.py 48 INFO 开始执行测试步骤

2019-03-12 20:18:38 test_case.py 113 INFO ======================开始执行测试用例步骤（第 1 步, 步骤ID： 2,
```

参考链接

https://uwsgi-docs.readthedocs.io/en/latest/
https://uwsgi-docs.readthedocs.io/en/latest/Nginx.html
https://uwsgi-docs.readthedocs.io/en/latest/Configuration.html
https://uwsgi-docs.readthedocs.io/en/latest/Options.html
https://blog.csdn.net/kevin6216/article/details/15378617
https://channels.readthedocs.io/en/latest/deploying.html
http://www.supervisord.org/installing.html#creating-a-configuration-file
http://www.supervisord.org/running.html#running-supervisord
http://www.supervisord.org/running.html#supervisorctl-command-line-options
https://github.com/django/daphne