EC2上で Django + Nginx + uWSGI を試す

Django nginx uwsgi

この記事は最終更新日から5年以上が経過しています。

概要

EC2インスタンスにNginxとuWSGIをつかってDjangoアプリケーションをデプロイするときのメモ。

環境構築

ec2インスタンスに入って環境を構築する。

```
sudo apt-get update
sudo apt-get -y upgrade
# もろもろインストールする
sudo apt-get -y install python-dev python-pip nginx
# pipでもろもろインストールする
```

```
sudo pip install -U pip
pip install django uwsgi
```

Djangoアプリケーションの作成

プロジェクトの作成

まずはDjangoのプロジェクトを作成する。

```
django-admin startproject myweb
cd myweb
```

ディレクトリ構成は下記のとおり。

```
.
├── manage.py
└── myweb
    ├── __init__.py
    ├── settings.py
```

```
    ├── urls.py
    └── wsgi.py
```

アプリを作成

'hello!'を表示するアプリを作成する。

```
python manage.py startapp hello
```

ここまでのディレクリ構成は以下のとおり。

```
.
├── hello
│   ├── admin.py
│   ├── __init__.py
│   ├── migrations
│   │   └── __init__.py
│   ├── models.py
│   ├── tests.py
│   └── views.py
├── manage.py
└── myweb
    ├── __init__.py
```

```
    ├── __init__.pyc
    ├── settings.py
    ├── settings.pyc
    ├── urls.py
    └── wsgi.py
```

helloアプリケーションの存在をプロジェクトに教えてあげるために、INSTALLED_APPSに'hello'を追加する。

myweb/settings.py

```
INSTALLED_APPS = (
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'hello',
)
```

hello/views.pyを下記のように作成。

hello.views.py

```
from django.http import HttpResponse
```

```
def main(request):
        return HttpResponse("Hello!")
```

URL構造を定義してあげる。

myweb/urls.py

```
urlpatterns = [
    url(r'^admin/', include(admin.site.urls)),
    url(r'^hello/', 'hello.views.main'),
]
```

これでひとまずアプリケーションができたので、migrateしてサーバーを起動する。

```
python manage.py migrate
python manage.py runserver 0.0.0.0:8000
```

下記のページをブラウザで開いて'hello!'がでたらOK!

[EC2のパブリックIP]:8000/hello/

uWSGIでデプロイ

WSGIはWebサーバとWebアプリケーションを接続するためのインターフェース定義の一つで、uWSGIはWSGIコンテナの機能をもったwebサーバー。

まずはuWSGIのみでデプロイしてみる。
the web client <-> uWSGI <-> Django

```
uwsgi --http :8000 --module myweb.wsgi
```

さきほどと同じようにブラウザを開いて'hello!'がでたらOK。

nginx + uWSGIでデプロイ

概要

nginxは静的コンテンツを高速に配信するように設計されたwebサーバー。

nginxの超基本的な使い方は次の通り。

```
# スタート
sudo /etc/init.d/nginx start
# ストップ
sudo /etc/init.d/nginx stop
# リスタート
sudo /etc/init.d/nginx restart
# ステータスの確認
sudo /etc/init.d/nginx status
```

今回はnginxが8000番ポートでリクエストを受け取り、8001番ポートで待ち受けているuWSGIにsocketで受け渡す構成になる。(webサーバーがnginxで、WSGIコンテナがuWSGI)
the web client <-> the web server <-> the socket <-> uWSGI <-> Django

nginxのuwsgiモジュール用の設定を行う

uwsgi_paramsをプロジェクト直下に作成する

```
uwsgi_param  QUERY_STRING       $query_string;
uwsgi_param  REQUEST_METHOD     $request_method;
uwsgi_param  CONTENT_TYPE       $content_type;
uwsgi_param  CONTENT_LENGTH     $content_length;

uwsgi_param  REQUEST_URI        $request_uri;
uwsgi_param  PATH_INFO          $document_uri;
uwsgi_param  DOCUMENT_ROOT      $document_root;
uwsgi_param  SERVER_PROTOCOL    $server_protocol;
uwsgi_param  REQUEST_SCHEME     $scheme;
uwsgi_param  HTTPS              $https if_not_empty;

uwsgi_param  REMOTE_ADDR        $remote_addr;
uwsgi_param  REMOTE_PORT        $remote_port;
uwsgi_param  SERVER_PORT        $server_port;
uwsgi_param  SERVER_NAME        $server_name;
```

Nginxの設定ファイルを作成

Nginxの設定ファイルを作成する。Nginxは起動時にディレクトリ `/etc/nginx/sites-enabled` に入っている設定ファイ

ルを読み込む。この記事ではプロジェクト直下に `myweb_nginx.conf` を作成し、 `/etc/nginx/sites-enabled` に `myweb_nginx.conf` へのシンボリックリンクを貼ることにする。

myweb_nginx.conf

```
# the upstream component nginx needs to connect to
upstream django {
    # server unix:///path/to/your/mysite/mysite.sock; # for a
    server 127.0.0.1:8001; # for a web port socket (we'll use
}

# configuration of the server
server {
    # the port your site will be served on
    listen      8000;
    # the domain name it will serve for
    server_name [EC2のプライベートIP]; # substitute your machine
    charset     utf-8;

    # max upload size
    client_max_body_size 75M;   # adjust to taste

    location /static {
        alias /path/to/your/mysite/static; # your Django proje
    }
```

```
    # Finally, send all non-media requests to the Django serve
    location / {
        uwsgi_pass  django;
        include     /path/to/your/mysite/uwsgi_params; # the u
    }
}
```

```
sudo ln -s ~/myweb/myweb_nginx.conf /etc/nginx/sites-enabled/
```

staticフォルダを作成

myweb/settings.pyにSTATIC_ROOTを指定

myweb/settings.py

```
STATIC_ROOT = os.path.join(BASE_DIR, "static/")
```

```
python manage.py collectstatic
```

ここまでのディレクトリ構成は以下のとおり。

```
.
├── db.sqlite3
├── hello
│   ├── admin.py
│   ├── admin.pyc
│   ├── __init__.py
│   ├── __init__.pyc
│   ├── migrations
│   │   ├── __init__.py
│   │   └── __init__.pyc
│   ├── models.py
│   ├── models.pyc
│   ├── tests.py
│   ├── views.py
│   └── views.pyc
├── manage.py
├── myweb
│   ├── __init__.py
│   ├── __init__.pyc
│   ├── settings.py
│   ├── settings.pyc
│   ├── urls.py
│   ├── urls.pyc
│   ├── wsgi.py
│   └── wsgi.pyc
├── myweb_nginx.conf
├── static
│   └── (省略)
└── uwsgi_params
```

デプロイ

```
sudo /etc/init.d/nginx restart
uwsgi --socket :8001 --module myweb.wsgi
```

さきほどと同じようにブラウザを開いて'hello!'がでたらOK。

参考

Setting up Django and your web server with uWSGI and nginx

便利で超強力なWSGIサーバー uWSGI を使ってみよう

軽量で高速なウェブサーバNginxを、Ubuntu 12.04に導入する(設定編その１)

nginx + uWSGI + Flaskを試してみる