Despliegue de Django en AWS con Nginx

Christian Escalante - CEO & Founder

Paso 3: NGINX

Paso 3 - Nginx

Amazon Web Service

Proyecto en Django

Configurando Nginx

a. Creamos el fichero de configuración para nuestro proyecto

```
$ sudo nano /etc/nginx/sites-available/<app_name>
```

```
server{
      listen 80;
      server_name <url_sitio|subdominio>;
      access_log /home/<app_name>/logs/nginx.access.log;
      error_log /home/<app_name>/logs/nginx.error.log;

      location / {
            include proxy_params;
            proxy_pass http://127.0.0.1:8000/;
      }
}
```

Paso 3 - Nginx

Amazon Web Service

Proyecto en Django

Configurando Nginx

Paso 3 - Nginx

Amazon Web Service

Proyecto en Django

Configurando Nginx

b. Creamos el enlace simbólico

```
$ sudo ln -s /etc/nginx/sites-available/<app_name>
/etc/nginx/sites-enabled/<app_name>
```

c. Comprobamos si no hay errores

```
$ sudo nginx -t
```

d. Recargamos Nginx

```
$ sudo service nginx reload
```

Paso 3 - Nginx

Amazon Web Service

Proyecto en Django

Configurando Nginx

Setting para producción y ficheros estáticos

a. Accedemos a nuestro usuario y luego a la carpeta “app”

```
$ sudo -u <app_name> -i
$ cd app
```

b. Creamos un settings para producción

```
$ cd <modulo_arranque>
$ nano setting_production.py
```

Paso 3 - Nginx

Amazon Web Service

Proyecto en Django

Configurando Nginx

```
import os
from <modulo_arranque>.settings import *

DEBUG = False
TEMPLATE_DEBUG = False

ALLOWED_HOSTS = ['*']

SECRET_KEY = '<genera un secret_key ex.: http://randomkeygen.com>'

STATIC_ROOT = os.path.join(BASE_DIR, 'static')
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
MEDIA_URL = '/media/'
```

Paso 3 - Nginx

Amazon Web Service

Proyecto en Django

Configurando Nginx

c. En la carpeta “app”, creamos las carpetas <static> y <media>

```
~/app $ mkdir static
~/app $ mkdir media
```

d. Definimos variable de entorno para el settings de producción

```
$ export DJANGO_SETTINGS_MODULE= <mod_arranque> .settings_production
```

e. Ejecutamos el collectstatic

```
$ python manage.py collectstatic
```

Paso 3 - Nginx

Amazon Web Service

Proyecto en Django

Configurando Nginx

Nginx y los ficheros estáticos

Salimos de nuestro usuario

```
$ logout
```

Nos dirigimos al fichero de configuración que habíamos creado antes

```
$ sudo nano /etc/nginx/sites-available/<app_name>
```

Paso 3 - Nginx

Amazon Web Service

Proyecto en Django

Configurando Nginx

```
server{
        listen 80;
        server_name <url_sitio|subdominio>;
        access_log /home/<app_name>/logs/nginx.access.log;
        error_log /home/<app_name>/logs/nginx.error.log;

        location /static {
                access_log off;
                alias /home/<app_name>/app/static;
        }

        location /media {
                access_log off;
                alias /home/<app_name>/app/media;
        }

        location / {
                include proxy_params;
                proxy_pass http://127.0.0.1:8000/;
        }
}
```

Paso 3 - Nginx

Nginx y los ficheros estáticos

Comprobamos si es correcto el fichero y recargamos

```
$ sudo nginx -t
$ sudo service nginx reload
```

Paso 3 - Nginx

Amazon Web Service

Proyecto en Django

Configurando Nginx

Actualizamos el fichero de Circus

Editamos el fichero de configuración para indicar el Settings de producción

```
$ sudo nano /etc/circus/conf.d/<app_name>.ini
```

Añadiremos al final de ficheros

```
[env:<app_name>]
DJANGO_SETTINGS_MODULE = <modulo_arranque>.settings_production
```

Reiniciamos circus y verificamos si está funcionando

```
$ sudo service circusd restart
$ ps aux | grep <app_name>
```

Paso 3 - Nginx

Amazon Web Service

Proyecto en Django

Configurando Nginx

PostgreSQL

En nuestro usuario principal, crearemos el usuario para PostgreSQL

```
$ sudo -u postgres createuser <app_name>
```

Creamos la base de datos en PostgreSQL

```
$ sudo -u postgres createdb <app_name> -O <app_name>
```

Paso 3 - Nginx

Amazon Web Service

Proyecto en Django

Configurando Nginx

PostgreSQL

En <app_name>, editamos settings.production para añadir los parámetros de PostgreSQL

```
$ nano app/<modulo_arranque>/settings_production.py
```

Añadimos al final del fichero

```
DATABASES = {
      'default': {
            'ENGINE': 'django.db.backends.postgresql_psycopg2',
            'NAME': '<app_name>',
            'USER': '<app_name>'
      }
}
```

Paso 3 - Nginx

Amazon Web Service

Proyecto en Django

Configurando Nginx

PostgreSQL

Instalamos psycopg2

```
$ pip install psycopg2
```

Aplicamos la migración

```
~/app $ python manage.py migrate
```

Reiniciamos circus

```
$ circusctl restart <app_name>
```

http://www.indestim.es

info@indestim.es

@indestim

/ indestim