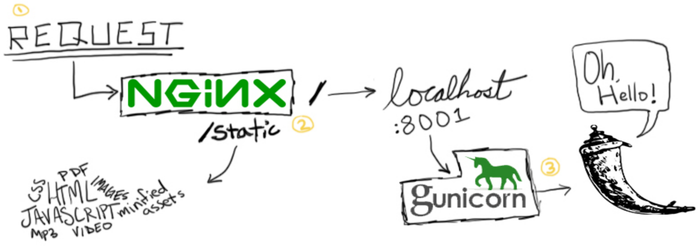


Guión de Prácticas 9: Despliegue en producción

S. Alonso (zerjioi@ugr.es) y J.M. Guirao (jmguirao@ugr.es)

Entrega: 31 - enero

El último paso de una aplicación web consiste en desplegarla en un ambiente de 'producción', es decir funcionando con la depuración en 'OFF' y conectada a un servidor web como ngix o apache en el puerto 80.

La aplicación para poner en producción en está última práctica puede ser la que se hizo con flask, o la de django.

Despliegue con docker-compose

Hay que ampliar **docker-compose.yml** para incluir un servicio más: el servidor web `nginx` que se encargará de:

1. Servir los archivos del **static**, que no tienen que pasar por nuestra aplicación.
2. Recibir los requests de los clientes y pasarlos a la aplicación, actuando como proxy inverso, y balanceo de carga en su caso
3. Encargarse del cifrado del http, redirigiendo a https cuando sea apropiado

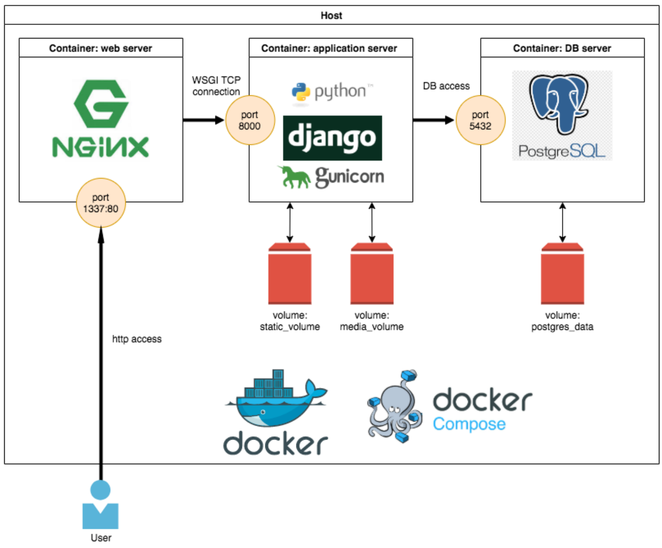
Lo primero es cambiar a la configuración de producción. Esto se hace en django cambiando dos variables en el archivo **settings.py**:

```
DEBUG = False
ALLOWED_HOSTS = ['*']
```

Con esto dejará de funciona el servidor de desarrollo, y de servir los contenidos de **/static**, que tendrán que pasar a servirse desde el servidor web de producción. Django tiene un script: collectstatic para facilitar pasar los contenidos a otro directorio.

También hay que substituir el servidor de desarrollo, `runserver` por el de producción, gunicorn, que hay que instalar con pip. No viene en la instalación de Django.

Quedaría ampliar **docker-compose.yml** para incluir otro servicio, y conectarlos:



docker-compose.yml:

```
version: '3.7'
       services:
          nginx:
             image: nginx:alpine
             ports:
                - 80:80
             # directorios para el archivo de configuración y archivos del static
             volumes:
                - ./conf:/etc/nginx/conf.d
                - ./web/static:/var/www/static
             depends_on:
                - web
          web:
             build: .
             restart: always
             command: gunicorn mi_sitio_web.wsgi:application --bind 0.0.0.0:8000
             # command: python manage.py runserver 0.0.0.0:8000
             volumes:
               - ./web:/web

           ...
```

Donde hemos creado una nueva carpeta **conf** para poner la configuración de nginx

El archivo configuración de nginx: en **./conf/default**

```
server {
    listen 80 default_server;

       # servidor web para archivos en  /static
       location /static/ {
             alias /var/www/static/;
       }

       # proxy inverso, se pasa a la aplicación wsgi
       location / {
            proxy_pass http://web:8000;
            proxy_set_header X-Forwarded-Host $server_name;
            proxy_set_header X-Real-IP $remote_addr;
       }
    }
```

Aquí se dan algunos detalles más: Packaging a Django App Using Docker, NGINX, and Gunicorn, o aquí para flask: Dockerizing Flask with Postgres, Gunicorn, and Nginx