

Desarrollo de Aplicaciones para Internet: Implantación

Hugo Bárzano Cruz – 77138361h

Correo: hugobarzano@gmail.com

Fuentes: <https://github.com/hugobarzano/DAIproduccion.git>

La idea de la implantación es poner en producción la aplicación que hemos estado desarrollando a lo largo del cuatrimestre. La puesta en producción puede llevarse a cabo de muchas maneras, yo he decidido utilizar Docker para contener y ejecutar la aplicación y Azure como IaaS para alojarla en la nube.

Lo primero es cambiar el `ALLOWED_HOSTS = ['*']`

Lo siguiente es crear los distintos script de configuración necesarios:

1. nginx-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://127.0.0.1:8001;
        proxy_set_header X-Forwarded-Host $server_name;
        proxy_set_header X-Real-IP $remote_addr;
    }
}
```

2. supervisor.conf

```
[program:gunicorn]
command=/usr/local/bin/gunicorn practica4.wsgi --bind 0.0.0.0:8000
directory=~/.DAIproduccion/manage.py
user=root
autostart=true
autorestart=true
redirect_stderr=true
```

3.collect_static.sh

```
#!/bin/bash  
cp -r static/ /var/www/static  
cp -r media/ /var/www/media
```

Una vez que tenemos los scripts necesarios, he desarrollado el siguiente Dockerfile:

```
FROM ubuntu:latest
```

```
#Autor
```

```
MAINTAINER Hugo Barzano Cruz <hugobarzano@gmail.com>
```

```
RUN sudo apt-get install -y git
```

```
RUN sudo git clone https://github.com/hugobarzano/DAIproduccion.git
```

```
#Actualizar Sistema Base
```

```
RUN sudo apt-get -y update
```

```
RUN sudo apt-get install -y python-setuptools
```

```
RUN sudo apt-get -y install python-dev
```

```
RUN sudo apt-get -y install build-essential
```

```
RUN sudo easy_install pip
```

```
RUN sudo pip install --upgrade pip
```

```
RUN cd DAIproduccion/ && sudo pip install -r requirements.txt
```

```
# PRODUCCION
```

```
run pip install gunicorn
```

```
# servidor web y watchdog
```

```
run apt-get install -y supervisor nginx
```

```
# configuraciones
```

```
run cp DAIproduccion/supervisor.conf /etc/supervisor/conf.d/
```

```
run cp DAIproduccion/nginx-default /etc/nginx/sites-available/default
```

```
run sed -i 's/DEBUG = True/DEBUG= False/' DAIproduccion/practica4/settings.py
```

```
expose 80
```

```
cmd DAIproduccion/collect_static.sh && supervisord
```

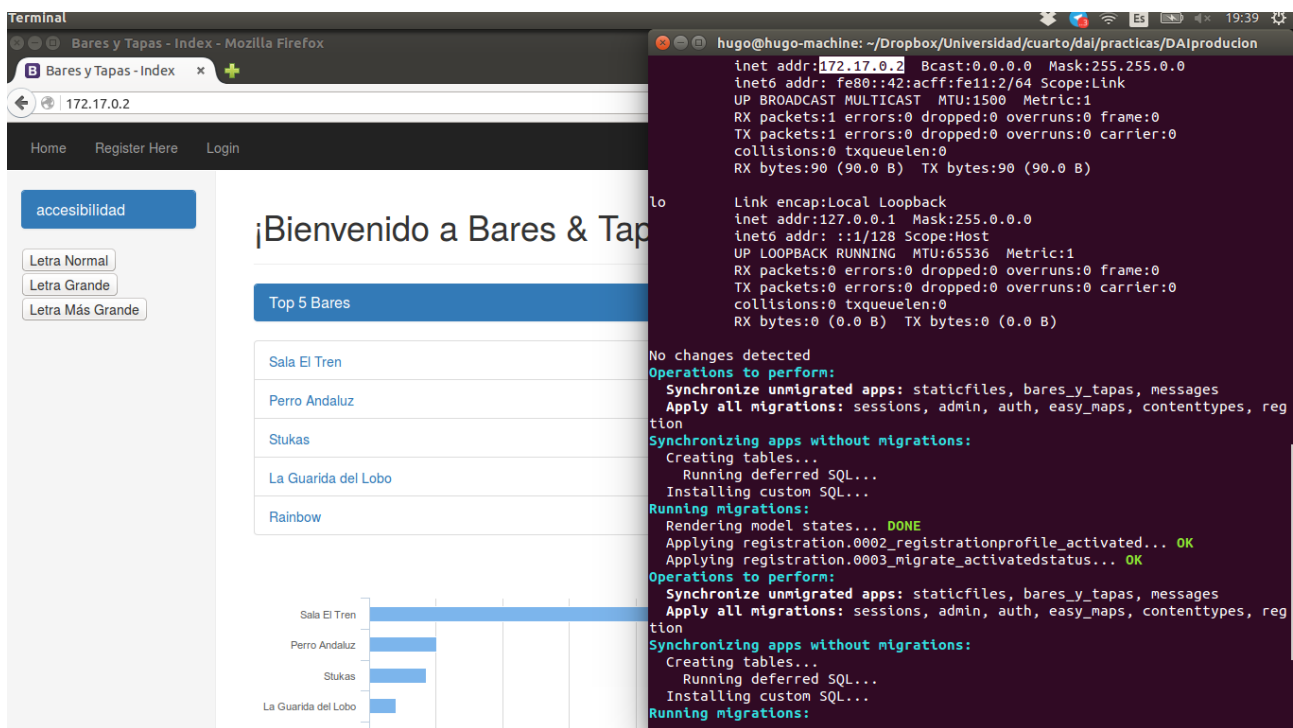
Debemos comprobar localmente que el docker es correcto. Primero tenemos que construir la imagen mediante:

```
sudo docker build -f ~/DAIproduccion/Dockerfile -t dai --no-cache=true .
```

Una vez construida, podemos ejecutarla mediante:

```
sudo docker run -t -i dai sh -c "ifconfig && cd DAIProduccion && python manage.py makemigrations --noinput && python manage.py migrate --noinput && python manage.py syncdb --noinput && sudo python manage.py runserver 0.0.0.0:80"
```

y accediendo a la ip del docker mediante el navegador, podemos comprobar que todo esta correcto



Ahora que sabemos que el Dockerfile genera una imagen funcional de la aplicación el siguiente paso es subirlo a Dockerhub como una automontated-build, de esta manera, podremos obtener la imagen en cualquier momento ejecutando

```
docker pull hugobarzano/daiproduccion
```

El repositorio de dockerhub es el siguiente:

<https://hub.docker.com/r/hugobarzano/daiproduccion/>

Ahora que disponemos de una imagen funcional accesible en cualquier momento, podemos desplegarlo en la nube. Voy a utilizar Azure como IaaS ya que dispongo de una cuenta con crédito temporal. Para realizar el despliegue, voy a utilizar herramientas para la creación de entornos virtuales (Vagrant) y herramientas para aprovisionamiento de entornos virtuales (Ansible).

Para la creación de la máquina virtual, he desarrollado el siguiente vagrantfile

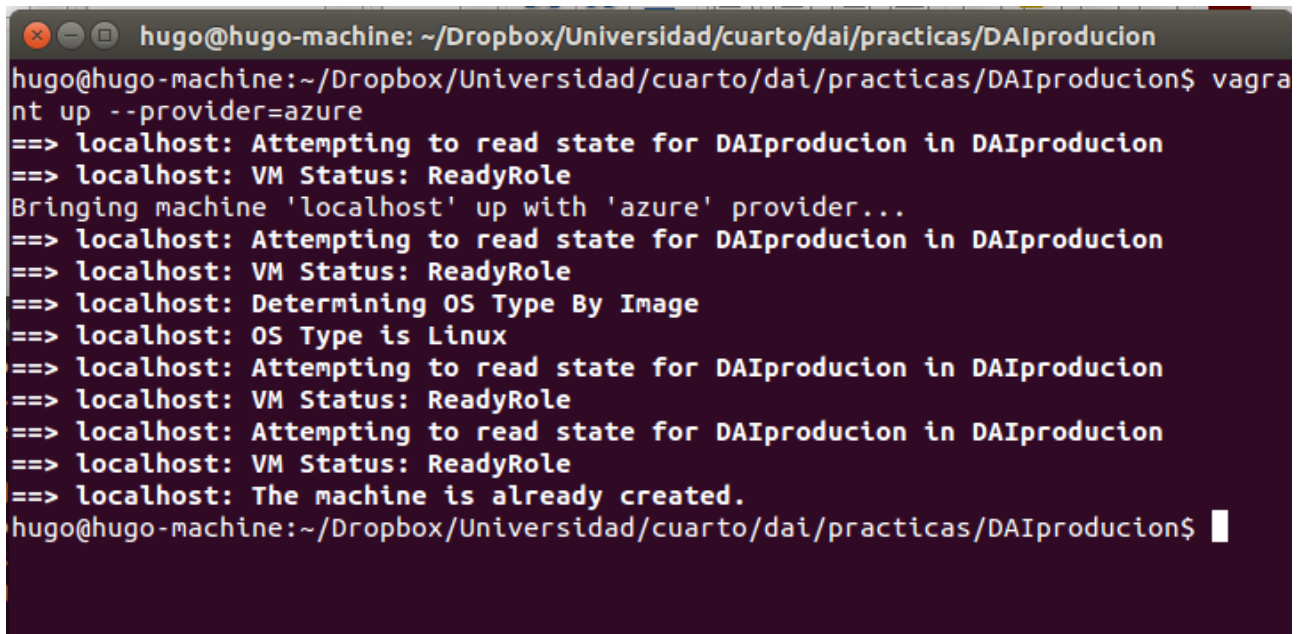
```
Vagrant.configure('2') do |config|
  config.vm.box = 'azure'
  config.vm.network "public_network"
  config.vm.network "private_network", ip: "192.168.56.10", virtualbox____intnet: "vboxnet0"
  config.vm.network "forwarded_port", guest: 80, host: 80
  config.vm.define "localhost" do |l|
    l.vm.hostname = "localhost"
  end

  config.vm.provider :azure do |azure, override|
    azure.mgmt_certificate = File.expand_path('~/.ssh/azurevagrant.pem')
    azure.mgmt_endpoint = 'https://management.core.windows.net'
    azure.subscription_id = 'b0eda1f9-f644-4cfd-bc55-29015eed62c9'
    azure.vm_image = 'b39f27a8b8c64d52b05eac6a62ebad85__Ubuntu-14_04_2-LTS-amd64-server-20150506-en-us-30GB'
    azure.vm_name = 'DAIproduccion'
    azure.cloud_service_name = 'DAIproduccion'
    azure.vm_password = 'Clave#Hugo#1'
    azure.vm_location = 'Central US'
    azure.ssh_port = '22'
    azure.tcp_endpoints = '80:80'
  end

  config.vm.provision "ansible" do |ansible|
    ansible.sudo = true
    ansible.playbook = "playbook.yml"
    ansible.verbose = "v"
    ansible.host_key_checking = false
  end
end
```

Para ejecutarlo, basta con hacer

```
vagrant up --provider=azure
```

A terminal window with a dark background and light-colored text. The prompt is 'hugo@hugo-machine: ~/Dropbox/Universidad/cuarto/dai/practicas/DAIproduccion'. The user enters 'vagrant up --provider=azure'. The output shows several status messages from Vagrant, including 'Attempting to read state for DAIproduccion in DAIproduccion', 'VM Status: ReadyRole', and 'The machine is already created.' The terminal ends with a prompt character '█'.

```
hugo@hugo-machine: ~/Dropbox/Universidad/cuarto/dai/practicas/DAIproduccion
hugo@hugo-machine:~/Dropbox/Universidad/cuarto/dai/practicas/DAIproduccion$ vagrant up --provider=azure
==> localhost: Attempting to read state for DAIproduccion in DAIproduccion
==> localhost: VM Status: ReadyRole
Bringing machine 'localhost' up with 'azure' provider...
==> localhost: Attempting to read state for DAIproduccion in DAIproduccion
==> localhost: VM Status: ReadyRole
==> localhost: Determining OS Type By Image
==> localhost: OS Type is Linux
==> localhost: Attempting to read state for DAIproduccion in DAIproduccion
==> localhost: VM Status: ReadyRole
==> localhost: Attempting to read state for DAIproduccion in DAIproduccion
==> localhost: VM Status: ReadyRole
==> localhost: The machine is already created.
hugo@hugo-machine:~/Dropbox/Universidad/cuarto/dai/practicas/DAIproduccion$ █
```

Una vez que tenemos la máquina, he desarrollado el siguiente playbook de ansible para aprovisionar dicha maquina con la imagen docker funcional:

```
- hosts: localhost
  sudo: yes
  remote_user: vagrant
  tasks:
    - name: Actualizar sistema base
      apt: update_cache=yes upgrade=dist
    - name: Instalar git
      action: apt pkg=git state=installed
    - name: Install Python Pip
      action: apt pkg=python-pip state=installed
    - name: Obtener aplicacion de git
      git: repo=https://github.com/hugobarzano/DAIproduccion.git dest=~/.DAIproduccion clone=yes
    - name: Dar permisos de ejecucion
      command: chmod -R +x ~/.DAIproduccion
    - name: Instalar docker y docker-compose
      command: sh ~/.DAIproduccion/deploy_docker.sh
    - name: Descargar imagen necesaria
      command: service docker restart
      command: docker pull hugobarzano/daiproduccion
    - name: Enrrutamiento
      command: sudo iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j DNAT --to-
```

destination 172.17.0.1:80

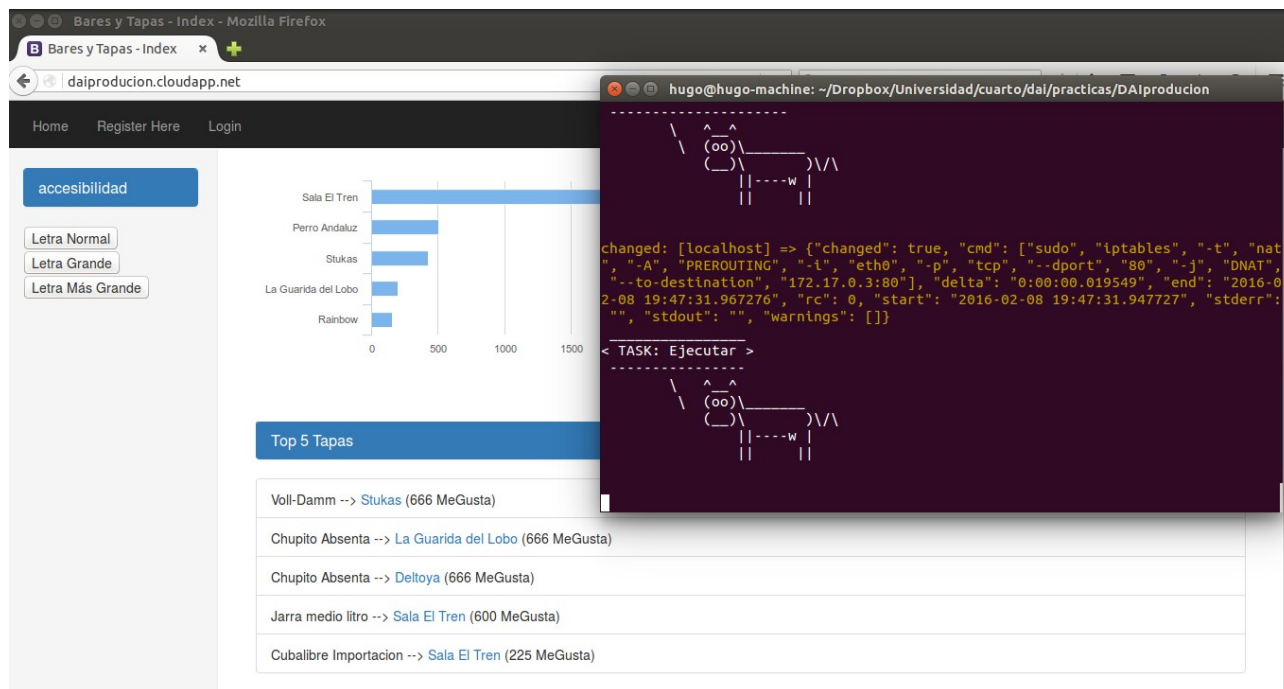
- name: Ejecutar

command: sudo docker run -t -i daiproduccion sh -c "ifconfig && cd ~/DAIproduccion && python manage.py makemigrations --noinput && python manage.py migrate --noinput && python manage.py syncdb --noinput && sudo python manage.py runserver 0.0.0.0:80"

Para realizar el aprovisionamiento basta con ejecutar

vagrant provision

y desde el navegador, accediendo al servicio en la nube que ha creado vagrant, podemos observar que la aplicación esta en producción



El servicio en la nube es <http://daiproduccion.cloudapp.net/>

NOTA: Este servicio permanecerá disponible temporalmente, hasta que se agote el crédito de mi pase para Azure.