Configuración Docker y Ansible

INSTALACIÓN DOCKER

Para instalar docker utilizamos el siguiente comando:

$sudo apt-get install docker-engine

Lo primero de todo creamos un contenedor que será la base de datos de MySQL. Utilizamos el siguiente comando para ello:

$docker run --name wordpressdb_dev1 -e MYSQL_ROOT_PASSWORD=labtel -e MYSQL_DATABASE=wordpress -d mysql:5.7

Con esto creamos un contenedor llamado wordpressdb_dev1 de contraseña labtel con una BD llamada wordpress, lo dejamos detached, y usamos una imagen mysql:5.7 (si no la tenemos, nos la descargará)

Después, nos descargamos la imagen de wordpress con:

#docker pull wordpress

Y lanzamos el contendor con el siguiente comando, donde le indicamos la base de datos y su contraseña (labtel) y vamos a crear el wordpress con el nombre wordpress_dev1:

#docker run -e WORDPRESS_DB_PASSWORD=labtel -d --name wordpress_dev1 --link wordpressdb:mysql wordpress

Con el siguiente comando comprobamos la dirección IP del contenedor del wordpress_dev1

#docker inspect wordpress_dev1

Este proceso se multiplica por número servidores wordpress y sus correspondientes bases de datos. Notese que se debe cambiar el nombre de los contendores a fin de que su identificación sea única.

INSTALACIÓN NGINX

La instalación del nginx al igual que en el caso del wordpress, habría que seguir los mismos pasos para la instalación del segundo servidor nginx. Para la instalación del nginx en el docker, lo hacemos de la siguiente manera:

#docker run --name nginx_dev -d -p 8080:80 nginx

De esta manera arrancamos un contenedor y se instala la imagen de nginx al cual lo llamamos nginx_dev. Algo a tener en cuenta es que cuando se genere el segundo servidor nginx hay que indicarle otro puerto origen que en nuestro caso pusimos el 8081:80 por ejemplo. Con el -p le

indicamos el puerto origen y después de los dos puntos le indicamos el puerto destino, es decir el puerto del servidor nginx.

El servidor nginx viene vacío sin ningún editor de texto, ni servidor ssh, los cuales debemos instalar. También necesitaríamos instalar python que luego nos hará falta para el ansible. Primeramente nos conectamos al servidor nginx de la siguiente manera:

#docker exec -ti nginx bash

container#apt-get update
container#apt-get install python
container#apt-get install ssh
container#apt-get install vim

Una vez instalados, configuramos el seridor ssh. Luego nos conectamos desde fuera del contenedor al contenedor mediante ssh para que se generen automaticamente las claves.

Para configurar el servidor ssh, necesitamos tener permisos de root del contenedor. Como no sabemos la contraseña del root se la añadimos de la siguiente manera.

container#passwd root

Le indicamos la contraseña que deseamos, en nuestro caso pondremos “labtel”. Una vez introducido la contraseña root, actualizamos el passwd.

container#update-passwd

Ahora en el fichero /etc/ssh/sshd_config debemos modificar el fichero indicandole que podemos hacer ssh mediante root:

container#vi /etc/ssh/sshd_config

En la línea de *PermitRootLogin* le indicamos que sí poniendo *yes*. Ahora reiniciamos el servicio ssh

container#/etc/init.d/ssh restart

Ahora nos conectamos al contenedor mediante ssh para que se generen las claves, desde el ordenador local

#ssh root@172.17.0.11

Le introducimos la contraseña de root que configuramos en el passwd e indicamos con “yes” que se generen las claves. Una vez se crean ya tenemos nuestro servidor nginx listo para luego ser configurado a través del ansible.

Antes de ir al ansible, vamos a ir a nuestro ordenador local y vamos a configurar el fichero /etc/hosts donde le vamos a indicar las direcciones IP de los servidores nginx y sus nombres de dominio correspondientes de los servidores wordpress. De esta manera le indicamos que cuando introduzcamos en la URL esos nombres de dominio, nos redirija a los reverse proxy nginx

correspondientes, los cuales harán el mapeo adecuado para dirigirnos a las direcciones IP de los servidores wordpress. En nuestro caso quedaría de la siguiente manera:

```
127.0.0.1       localhost
127.0.1.1       portatil

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

172.17.0.10 wordpress_dev1.bce.es
172.17.0.10 wordpress_dev2.bce.es
172.17.0.11 wordpress_prod1.bce.es
172.17.0.11 wordpress_prod2.bce.es
```

CREACIÓN FICHEROS DE CONFIGURACIÓN DEL NGINX

En los ficheros de configuración le indicamos el nombre de dominio y su correspondiente dirección IP. Este es un ejemplo de uno de los ficheros de configuración. Es un fichero llamado vhost1.conf y es la configuración del nginx para que haga de reverse proxy y que todas las peiticiones a wordpress_dev1.bce.es las envíe a la dirección IP 172.17.0.3:

```
server {
        listen   80;
        server_name  wordpress_dev1.bce.es;

        access_log  /var/log/nginx/access.log;


        location / {
                proxy_pass      http://172.17.0.3/;
        }
}
```

Como este hay que hacer un fichero para cada uno de los servidores wordpress.

INSTALACIÓN ANSIBLE

Instalamos ansible mediante el comando:

#apt-get install ansible

Una vez instalado ansible, configuramos los dos grupos que vamos a configurar el nginx_dev y el nginx_prod. Para ello modificamos el siguiente fichero:

#nano /etc/ansible/hosts

Al final del fichero le indicamos los grupos de hosts (los dos servidores nginx) diferentes que nos servirán para indicarselo en el yml de la siguiente manera:

```
[nginx_dev]
172.17.0.10

[nginx_prod]
172.17.0.11
```

Reiniciamos el servicio de ansible y comprobamos que se puede acceder mediante a los dos servidores nignx mediante ssh:

#/etc/init.d/ansible restart
$ansible all -m ping

Como respuesta si hay conexión nos indicará que la hay que es correcto. Ahora se crean los ficheros .yml donde le indicaremos que nos copie los ficheros de configuración correspondientes al nginx en su directorio /etc/nginx/conf.d/

A continuación se presenta uno de los ficheros .yml que realizamos para la configuración del servidor nginx:

```
---
- hosts: nginx_dev
  tasks:
   - name: Configure vhost1
     copy: src=/home/cristian/Escritorio/vhosts/vhost1.conf
dest=/etc/nginx/conf.d/vhost1.conf owner=root group=root mode=0644
   - name: Configure vhost2
     copy: src=/home/cristian/Escritorio/vhosts/vhost2.conf
dest=/etc/nginx/conf.d/vhost2.conf owner=root group=root mode=0644
     notify:
      - Start Nginx

  handlers:
   - name: Start Nginx
     service: name=nginx state=reloaded
```

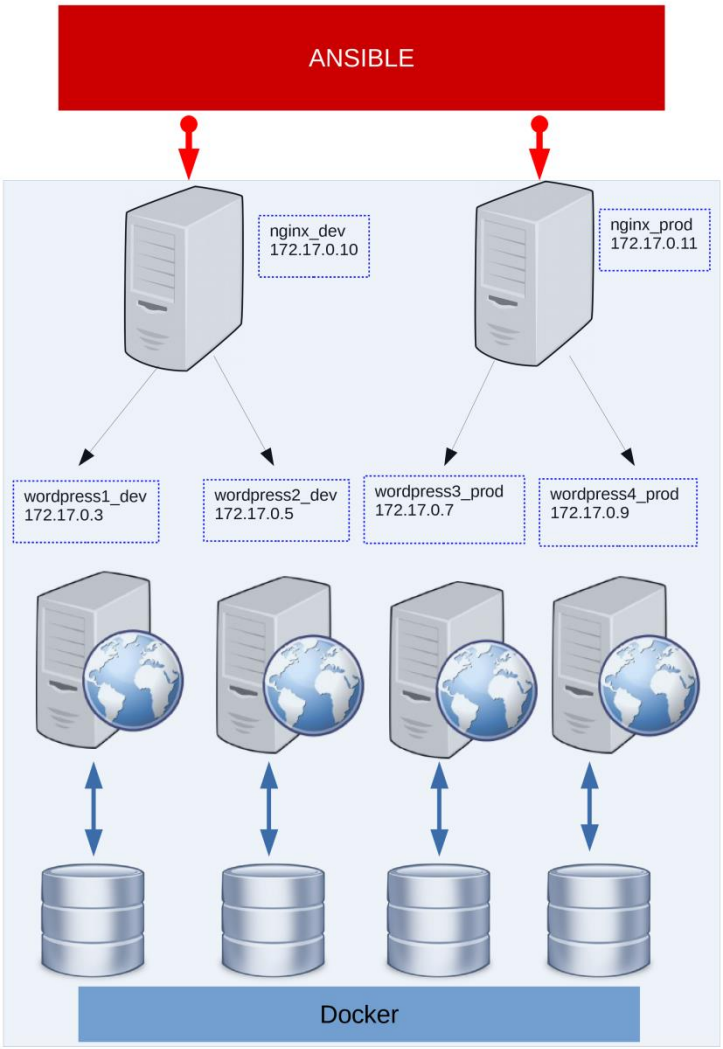
En el apartado hosts se le indica el grupo que indicamos en el fichero /etc/ansible/hosts, es decir que este .yml le afectará a la dirección IP del primer servidor nginx. En el fichero le indicamos que copie los dos primeros ficheros de configuración en el servidor nginx_dev y que recargue el servicio.

Habría que hacer un segundo fichero .yml para el servidor nginx_prod. Una vez hecho podemos ejecutar el .yml ejecutando el siguiente comando:

#ansible-playbook -s -k -u root nginx_dev.yml

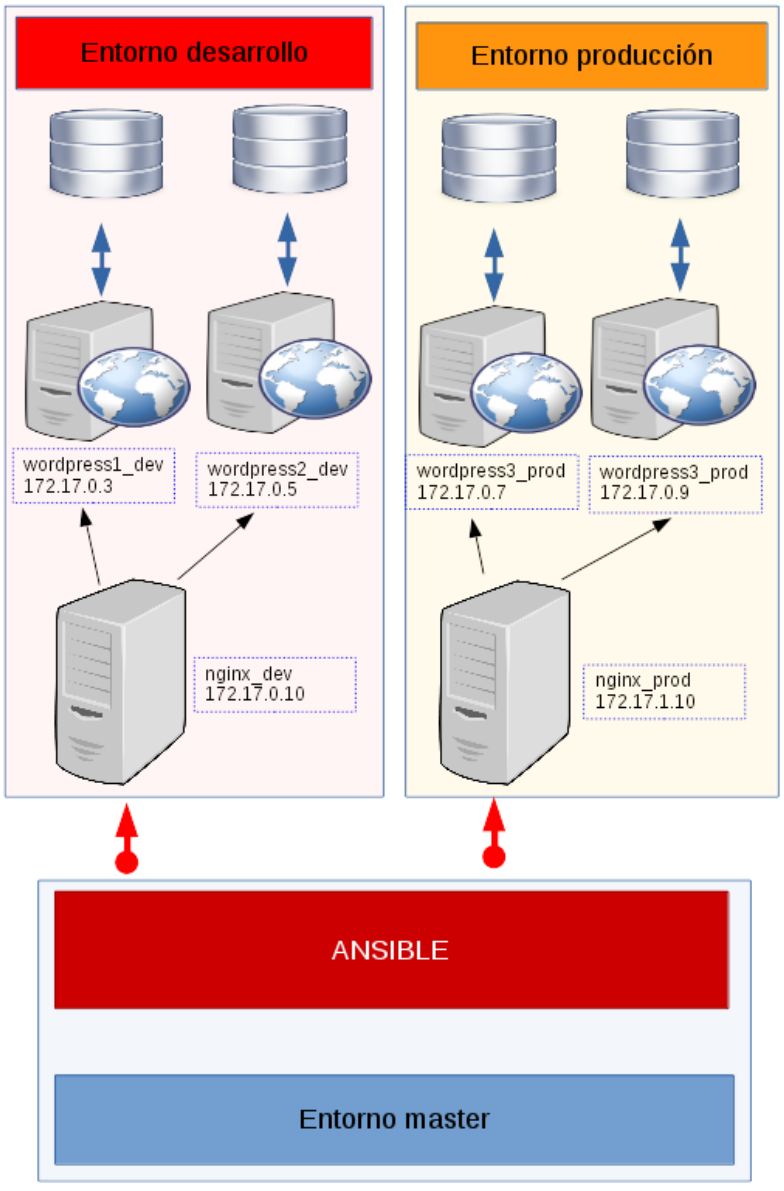
Con el -k indicamos que nos pida la contraseña con el -u el usuario y con -s para ejecutarlo como sudo.

Entonces ya estaría listo. Automáticamente los servidores nginx ya deberían estar configurados y en un browser poniendo el nombre de dominio debería aparecer la página del wordpress. A continuación, se presenta el esquema del resultado de la práctica.


ANSIBLE
nginx_dev
172.17.0.10
nginx_prod
172.17.0.11
wordpress1_dev
172.17.0.3
wordpress2_dev
172.17.0.5
wordpress3_prod
172.17.0.7
wordpress4_prod
172.17.0.9
Docker

Diseño con nueva arquitectura

Ahora hemos dividido la maqueta en dos máquinas virtuales + nativo. En una de las máquinas estará la parte de desarrollo, y en la otra la parte de producción. Aquí presentamos el esquema de cómo ha quedado la arquitectura.



Como vemos, tenemos en cada máquina virtualizado con *docker* dos servidores web de *wordpress*, con sus bases de datos, y un servidor *nginx*. Desde nativo, se va a utilizar *ansible* para que configure *nginx* en las máquinas virtuales.

La diferencia respecto al caso anterior en la configuración reside únicamente en la configuración de *ansible*. Ha sido necesario cambiar la configuración del fichero hosts de *ansible*, para que ahora apunten a la dirección local, y con el puerto *ssh* que hayamos redirigido desde el *virtualbox*.

El fichero queda de la siguiente manera:

```
[prod]
prod ansible_ssh_port=2222 ansible_ssh_host=127.0.0.1
[dev]
dev ansible_ssh_port=2223 ansible_ssh_host=127.0.0.1
```

Por último, es necesario cambiar las instrucciones del *playbook*.
En este caso, como no tenemos acceso directo al *docker* donde se encuentra *nginx*, vamos a copiar los archivos de configuración *vhost* en la máquina virtual. Después, vamos a copiar también un script. En este script se le indicará que cree un nuevo directorio, donde se moverán los *vhosts*, que borre el contenedor de *nginx* ya existente, y que lance otro en el cual se monta el directorio con los ficheros de configuración, en el directorio de *nginx* donde se recoge la configuración. Por último, se recarga el servicio de *nginx*.
A continuación, presentamos el *playbook*:

```
---
- hosts: dev
 tasks:
  - name: Configure vhost1
    copy: src=/home/borja/BCE/ansible/vhosts/vhost1.conf
dest=/home/ikasle/vhost1.conf owner=root group=root mode=0644
  - name: Configure vhost2
    copy: src=/home/borja/BCE/ansible/vhosts/vhost2.conf
dest=/home/ikasle/vhost2.conf owner=root group=root mode=0644
  - name: Copy script
    copy: src=/home/borja/BCE/ansible/script.sh dest=/home/ikasle/script.sh
owner=root group=root mode=0777
  - name: Start docker
    command: bash /home/ikasle/script.sh
```

Y el fichero script:

```
#!/bin/bash
mkdir /home/ikasle/vhost
mv /home/ikasle/vhost* /home/ikasle/vhost
docker stop nginx
docker rm nginx
docker run --name nginx -d -p 8080:80 -v /home/ikasle/vhost:/etc/nginx/conf.d
nginx
docker exec -ti nginx /etc/init.d/nginx reload
```