Cloud computing

Servicios y aplicaciones

Práctica 2 PaaS

Autor: Juan Pablo González Casado

Email: pablo12614@correo.ugr.es

Descripción: Desplegar contenedores con un sistema gestor de base de datos y un servidor web y ejecutar la aplicación implementada en la práctica anterior.

Preparación de entorno docker

En primer lugar deberemos buscar los contenedores que mas nos convengan para nuestro proyecto.

En mi caso he utilizado un contenedor con alpine, apache y php5 para hacer de servidor web. Y para el SGBD el contenedor oficial de MySQL.

Para instalarlos en nuestro entorno deberemos en primer lugar conectarnos a hadoop:

```
ssh mcc15472800@hadoop.ugr.es
```

Ahora nos traemos los contenedores:

```
docker pull nimmis/alpine-apache-php5
docker pull mysql/mysql-server
```

El siguiente paso será desplegar instancias de estos contenedores y para ello deberemos de consultar los puertos habilitados para nuestro usuario.

Una vez conocido esto, procedemos a desplegar:

```
docker run -d -p PUERTO1:3306 --name jpgdb -e MYSQL_ROOT_PASSWORD=PWD
mysql/mysql-server
docker run -d -p PUERTO3:80 --name jpgapp3 --link jpgdb:jpgdb nimmis/apache-php5
```

En una primera línea estamos instanciando el contenedor de mysql, indicando que tiene nombre jpgdb, usuario root y contraseña PWD.

En la segunda línea instanciaremos el servidor web y lo enlazaremos con el contenedor de mysql. Se puede ver como se indica con --link a quien debe enlazarse. Con esto ya sabe donde tiene que acudir para realizar las llamadas.

Configuración de instancias

Ahora solo nos queda configurar nuestras instancias.

Nos conectaremos al contenedor de base de datos. `docker exec -i -t a3c813ec2f88 /bin/sh` Para saber el id del contenedor consultaremos con 'docker ps' y nos dará un listado de todos los contenedores que están instanciados, buscando el nuestro y consiguiendo asi el ID.

Primero vamos a configurar mysql conectandonos a la consola (mysql -u root -p) y ejecutar los siguientes comandos:

```
CREATE SCHEMA IF NOT EXISTS `practica1cc` DEFAULT CHARACTER SET utf8;

USE `practica1cc` ;

CREATE TABLE IF NOT EXISTS `practica1cc`.`usuarios` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(15) NOT NULL,
  PRIMARY KEY (`id`));

insert into usuarios (id,name)  values (null,'Pablo1');
insert into usuarios (id,name)  values (null,'Pablo2');

CREATE USER 'pablo'@'%' IDENTIFIED BY 'asd1234'
GRANT ALL PRIVILEGES ON *.* TO 'pablo'@'%' WITH GRANT OPTION;
```

Con esto tendremos la base de datos creada, con una tabla, algunos registros y un usuario que permita conexión desde fuera y no solo con localhost (%).

Ahora nos conectaremos al contenedor del servidor web e instalaremos git para traernos el proyecto:

`docker exec -i -t ID /bin/sh`

```
apt-get update -y && apt-get install git -y
git clone [nuestrorepositorio dentro de /var/www/html/]
```

Y por último modificaremos nuestro archivo de conexión a base de datos indicandole los siguientes datos:

```
mysql_connect('jpgdb', 'pablo', 'asd1234')
mysql_select_db('practica1cc', $this->connection)
```

Nuestro contenedor de base de datos (jpgdb) como host, el usuario creado y seleccionaremos la base de datos.

Conclusiones

Al dupliciar los contenedores y borrar uno de ellos, el servicio sigue funcionando con normalidad, ya que los enlaces entre contenedores se hacen por nombre y sabrá como encontrar el otro contenedor que tambien está ejecutando el mismo servicio.

Con esto y haciendo uso de sistemas de almacenamiento de datos, podemos evitar aún más que nuestro servicio caiga y no existe nada que lo respalde.