Docker

Sprint 2 Lab 1

Icono

Descripción generada automáticamente

Felipe Izquierdo Romero

Índice

[1 Descripción del problema](#DescripciónDelProblema)

[2 Desarrollo](#Desarrollo)

[3 Opcional](#Opcional)

1. Descripción del problema

Crear una red de contenedores que se llame “todo\_lan“ que conecte un servicio web de Python con Flask y una base de datos MySQL. Opcional añadir otro servicio o crear otra red con diferentes contenedores.

1. Desarrollo

Para realizar esta red de contenedores se ha de usar un Dockerfile para configurar la imagen de Python con Flask y un Docker-compose para dar las instrucciones de montar la imagen de MySQL y el dockerfile de Python. Además de indicar la red, variables de entorno, mapeo de puertos y volúmenes para administrar los archivos de los contenedores.

En primer lugar, se creará la siguiente estructura de directorios para que el contenedor de Python funcione correctamente y enlazar los volúmenes a sus respectivas carpetas MySQL y Python.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración 1- Estructura de directorios

A continuación, se creará el dockerfile de Python con sencillas instrucciones

* FROM: imagen de DockerHub de la que se construirá el contenedor.
* WORKDIR: con el directorio de trabajo del contenedor.
* COPY: copiara archivos de tu computadora al contenedor al iniciarse.
* RUN: para correr al iniciarse unos comandos en una terminal.
  + Estos son para descargar el cliente de mysql y poder conectar flask a MySQL e instalar las librerías necesarias de Flask.
* EXPOSE: puerto donde el contenedor enlazará conexiones.
* CMD: para correr la aplicación de flask.

Texto

Descripción generada automáticamente

Ilustración 2- Dockerfile de Python

Posteriormente se deberá crear el Docker-compose definiendo por etiquetas diferentes configuraciones.

* Services: son los servicios que queremos ejecutar en contenedores que serán dos, Python y mysql.
* Build: para construir nuestra imagen configurada de Python.
* Ports: para mapear el puerto con el contenedor.
* Volumes: creara el volumen enlazando <ruta\_local>:<ruta\_contenedor>.
* Environment: variables de entorno.
* Depends\_on: dependencias del contenedor.
* Networks: la red que se conectaran.

Fuera de la etiqueta services se declara la red con otra etiqueta network. En este caso no se configurarán mas valores para la red y por defecto se creará un bridge (puente) con scope en tu red local.

Texto

Descripción generada automáticamente

Ilustración 3- Docker-compose.yml

Antes de finalizar la configuración se escribirá en el archivo requirements.txt las librerías que Python deberá de instalar al iniciar el contenedor. Flask y flask\_mysqldb.

Para finalizar deberemos abrir una consola de comandos y movernos al directorio raíz donde se encuentran estos dos archivos de configuración e Docker y ejecutar el comando

docker-compose up

Desde Docker desktop se observará la creación de nuestra red de contenedores y podremos acceder a ellos a través de nuestro navegador.

<http://localhost:5000>

1. Opcional

De manera opcional para este Lab se añadirán dos contenedores más para ofrecer un servicio de monitoreo de la aplicación web. Estos contenedores serán imágenes de Prometheus y Grafana.

Prometheus será el encargado de ofrecernos la funcionalidad para poder usar diferentes métricas y así obtener un análisis del funcionamiento de nuestra web. Para esta ocasión se usarán contadores simples para contabilizar el flujo de nuestra sencilla app y ver cuantas veces se ha ejecutado las funciones declaradas como endpoints.

Grafana se encargará de conectar a la API que Prometheus ofrece y otorgará una interfaz de usuario para visualizar con graficas y tablas los valores que se están midiendo en nuestra aplicación web.

La configuración de estos contenedores se añadirá al Docker-compose.yml de la siguiente manera.

Texto

Descripción generada automáticamente

Ilustración 4- Configuración Grafana y Prometheus

En el código de Python se añadirán unos cambios. Se importará la librería prometheus\_client y de esa librería se crearan tres objetos Counter(<nombre>,<descripción>) que serán los contadores.

Texto

Descripción generada automáticamente

Ilustración 5- Contadores

Con la función inc() se incrementaran en uno en cada enpoint cada vez que se ejecute esa parte del código que viene a ser cada vez que accedan a la web, creen una task o eliminen una task.

Además, se configurara un enpoint adicional que nos ofrecerá la posibilidad de acceder fuera de Grafana a las métricas de Prometheus.

Texto

Descripción generada automáticamente

Ilustración 6- Endpoint metricas prometheus

Tras realizar el paso anteriormente descrito para iniciar los contenedores podremos acceder a Grafana a través de:

<http://localhost:3000/login>

Entraremos con el usuario y la contraseña descritas en el compose “admin” y “password”.

En el menú a la izquierda de la pantalla haciendo clic en DashBoard>Create DashBoard>Add Visualization.

Aparecerá una ventana donde visualizar Datasource que hayamos creado o crearemos uno haciendo clic abajo a la derecha “Configure a new data source”.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración 7- Creación DashBorad

Para crear el datasource se debe clicar en prometheus y en la siguiente pestaña de configuración escribiremos el nombre del datasource y la url del servidor de prometheus: <http://prometheus:9090>.

Captura de pantalla de computadora

Descripción generada automáticamente

Ilustración 8- Configuración datasource

Abajo del todo clicamos en “Save & test” y ya tendremos un datasource creado ahora repetiremos los pasos hasta llegar a la “Ilustración-7”. Y clicaremos en nuestro datasource.

Posteriormente en metrics explorer podremos encontrar nuestras métricas definidas en nuestro código de Python y crear graficas según nuestras necesidades de monitoreo.

Captura de pantalla de un celular

Descripción generada automáticamente

Ilustración 9- Creación de métricas

Captura de pantalla de un celular

Descripción generada automáticamente

Ilustración 10- Nuestras métricas definidas

Además como se ha mencionado a través de <http://localhost:5000/metrics> podremos ver un JSON con las métricas de Prometheus y abajo del todo las métricas que hemos definido.

Texto

Descripción generada automáticamente

Ilustración 11- Metricas Prometheus