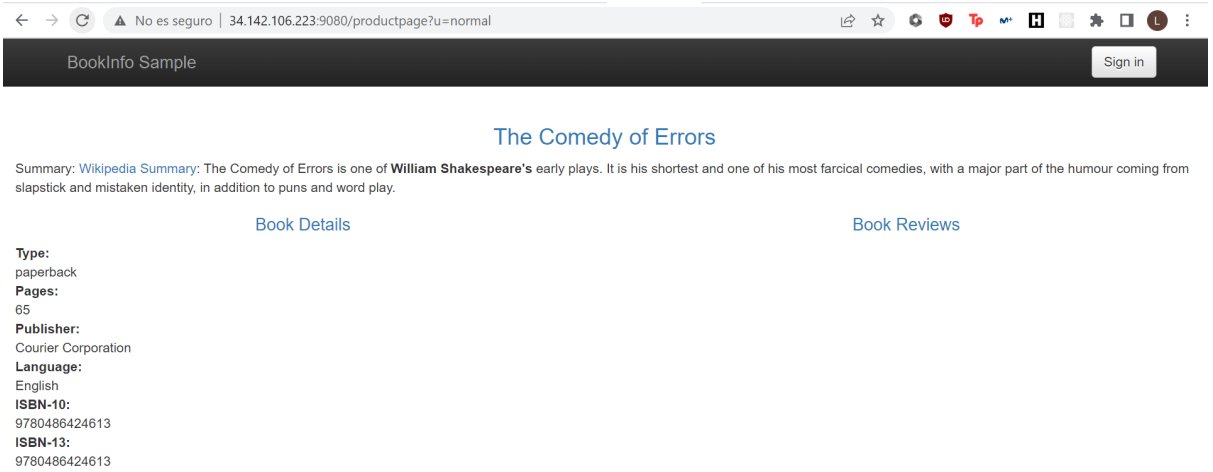
MEMORIA PRÁCTICA CREATIVA 2 CDPS

Luis Rincón Barrero
Vinlorenz Mindaros Sanchez
Shangde Jin

Despliegue de la aplicación en máquina virtual pesada

Se ha utilizado un script para la automatización de la instalación de la aplicación en una MV alojada en la infraestructura de google cloud.

```python
import sys
import os

# Remember to run script on Debian Buster since all packages are
# compatible with its default python version (3.7.3)

if len(sys.argv) < 2:
    print("Introduce group number as an argument")
    exit()
else:
    group_number = sys.argv[1]

os.system("sudo apt-get update")
os.system("sudo apt-get install -y python3-pip git")
os.system("git clone https://github.com/CDPS-ETSIT/practica_creativa2.git")

path_productpage = "practica_creativa2/bookinfo/src/productpage"

os.system(f"python3 -m pip install -r {path_productpage}/requirements.txt")
os.system(f'sed -i "s/Simple\sBookstore\sApp/g{group_number}/g" {path_productpage}/templates/index.html')
os.system(f'sed -i "s/Simple\sBookstore\sApp/g{group_number}/g" {path_productpage}/templates/productpage.html')

os.system(f"python3 {path_productpage}/productpage_monolith.py 9080")
```



Utilizando la dirección IP externa (utilizando http://34.142.106.223:9080), podemos acceder a la productpage corriendo en el puerto 9080. Habiendo previamente habilitado el puerto correspondiente en la regla de firewall.

Puntos débiles de la arquitectura

En cuanto a la fiabilidad, al tener una estructura monolítica, el fallo de uno de los procesos supondría el fallo del servicio.
Al tener que seleccionar una zona para crear la máquina virtual, no tiene movilidad y es estática. En este mismo sentido, dependemos de google por lo que un fallo suyo supondría una caída de nuestro sistema.

En lo referente a la escalabilidad, al ser una máquina muy pesada resulta muy difícil escalarla, por la cantidad de recursos que necesita.

Despliegue de una aplicación monolítica usando docker

- **Dockerfile**

```
FROM python:3.6
ENV GROUP_NUMBER=0
RUN apt-get update
RUN apt-get install -y git
RUN git clone https://github.com/CDPS-ETSIT/practica_creativa2.git
WORKDIR /practica_creativa2/bookinfo/src/productpage
RUN sed -i "s/urllib3==1.26.5/urllib3==1.24.3/g" requirements.txt
RUN python3 -m pip install -r requirements.txt
WORKDIR /practica_creativa2/bookinfo/src/productpage/productpage/templates/
EXPOSE 9080
CMD sed -i "s/Simple\sBookstore\sApp/g${GROUP_NUMBER}/g" index.html; \
    sed -i "s/Simple\sBookstore\sApp/g${GROUP_NUMBER}/g" productpage.html; \
    python3 /practica_creativa2/bookinfo/src/productpage/productpage_monolith.py 9080

# docker build -t g29/product-page-mono .
# docker run --name g29-product-page-mono -p 9080:9080 -e GROUP_NUMBER=29 -it g29/product-page
```

- **Crear la imagen de docker**
  docker build -t grupo29/product-page-mono .

- **Arrancar el contenedor:**
  docker run –name g29-product-page-mono -p 9080:9080 -e GROUP_NUMBER=29 -it g29/product-page

Puntos débiles de la arquitectura

Docker obliga al uso del lenguaje python, por lo que hay que adaptar todos los servicios que quieras proporcionar a ese lenguaje.
En cuanto a la fiabilidad, si utiliza más recursos de los necesarios es probable que utilice zonas de la memoria donde se aloje el propio docker, provocando que el sistema sea inestable.
Al respecto de la escalabilidad, el uso de docker de forma monolítica no permite una fácil expansión.

Segmentación de una aplicación monolítica en microservicios utilizando docker-compose

- **Details**

```
FROM ruby:2.7.1-slim
RUN apt-get update
RUN apt-get install -y git
RUN git clone https://github.com/CDPS-ETSIT/practica_creativa2.git
WORKDIR /practica_creativa2/bookinfo/src/details
RUN mkdir /opt/microservices
RUN cp details.rb /opt/microservices/
ENV ENABLE_EXTERNAL_BOOK_SERVICE=true
CMD ["ruby", "/opt/microservices/details.rb", "9080"]

# docker build -t g29/details .
```

- **Ratings**

```
FROM node:12.18.1-slim
RUN apt-get update
RUN apt-get install -y git
RUN git clone https://github.com/CDPS-ETSIT/practica_creativa2.git
WORKDIR /practica_creativa2/bookinfo/src/ratings
RUN mkdir /opt/microservices
RUN cp package.json /opt/microservices/
RUN cp ratings.js /opt/microservices/
WORKDIR /opt/microservices/
RUN npm install .
ENV SERVICE_VERSION v1
EXPOSE 9080
CMD ["node","/opt/microservices/ratings.js","9080"]

# docker build -t g29/ratings .
```

- **Product Page**

```
FROM python:3.6
ENV GROUP_NUMBER=0
RUN apt-get update
RUN apt-get install -y git
RUN git clone https://github.com/CDPS-ETSIT/practica_creativa2.git
WORKDIR /practica_creativa2/bookinfo/src/productpage
RUN sed -i "s/urllib3==1.26.5/urllib3==1.24.3/g" requirements.txt
RUN python3 -m pip install -r requirements.txt
WORKDIR /practica_creativa2/bookinfo/src/productpage/templates/
EXPOSE 9080
CMD sed -i "s/Simple\sBookstore\sApp/g${GROUP_NUMBER}/g" index.html; \
    sed -i "s/Simple\sBookstore\sApp/g${GROUP_NUMBER}/g" productpage.html; \
    python3 /practica_creativa2/bookinfo/src/productpage/productpage.py 9080

# docker build -t g29/product-page .
```

- **Reviews**

```
FROM websphere-liberty:20.0.0.6-full-java8-ibmjava

ENV SERVERDIRNAME reviews

COPY ./servers/LibertyProjectServer /opt/ibm/wlp/usr/servers/defaultServer/

RUN /opt/ibm/wlp/bin/installUtility install  --acceptLicense /opt/ibm/wlp/usr/servers/defaultServer/server.xml && \
    chmod -R g=rwx /opt/ibm/wlp/output/defaultServer/

ARG service_version
ARG enable_ratings
ARG star_color
ENV SERVICE_VERSION ${service_version:-v1}
ENV ENABLE_RATINGS ${enable_ratings:-false}
ENV STAR_COLOR ${star_color:-black}

CMD ["/opt/ibm/wlp/bin/server", "run", "defaultServer"]

# docker build --build-arg service_version=v1 -t g29/reviews:v1 .

# docker build --build-arg service_version=v2 --build-arg \
#     enable_ratings=true -t  g29/reviews:v2 .

# docker build --build-arg service_version=v3 --build-arg \
#     enable_ratings=true --build-arg star_color=red -t g29/reviews:v3 .
```

- **Docker-Compose**

```yaml
services:
  product-page:
    image: g29/product-page
    container_name: g29-product-page
    ports:
      - "9080:9080"
    environment:
      - GROUP_NUMBER=29

  details:
    image: g29/details
    container_name: g29-details
    ports:
      - "9081:9080"
    depends_on:
      - product-page

  reviews:
    image: g29/reviews:v3
    container_name: g29-reviews
    depends_on:
      - product-page

  ratings:
    image: g29/ratings
    container_name: g29-ratings
    ports:
      - "9082:9080"
    depends_on:
      - reviews
```

- **Diferencias con la versión de un único contenedor.**

Al utilizar microservicios, requerimos de más memoria. Asimismo, podemos utilizar diferentes lenguajes de programación y editar las versiones sin tener que cambiar la infraestructura.

Puntos débiles de la arquitectura

En docker compose no puedes desplegar varias versiones a la vez. En el caso particular de esta práctica, si se quiere cambiar entre las distintas versiones de reviews hay que modificar el compose.
En cuanto a escalabilidad y fiabilidad es la mejor opción de las tres opciones.