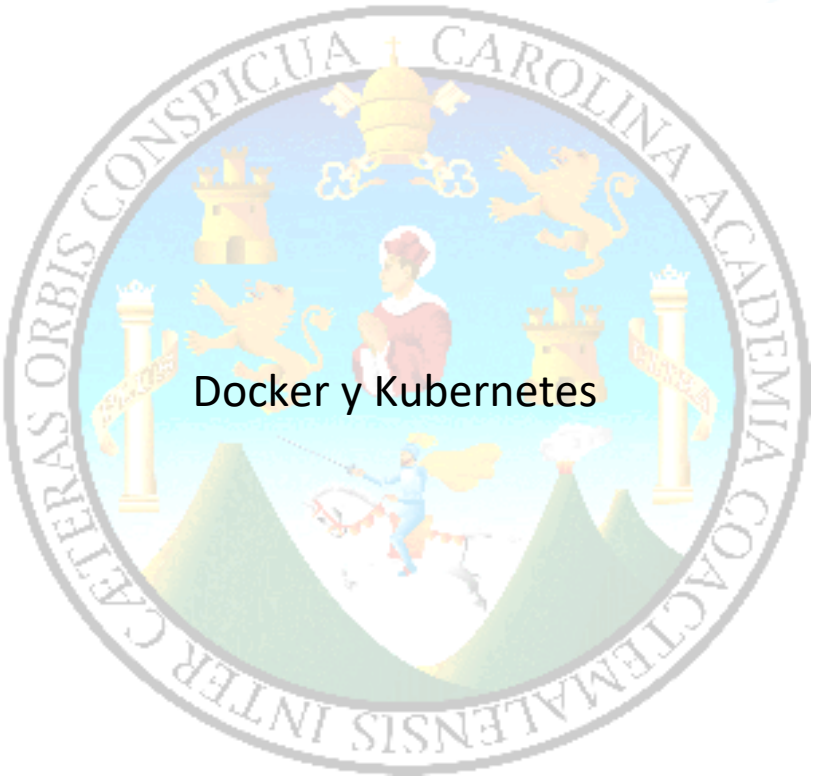
Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Sistemas Operativos 2
Sección O



Docker y Kubernetes

Grupo 26

Docker

En este segmento se explicará de manera detalla los archivos dockerfile utilizados para la práctica.

Frontend

En este dockerfile se utilizaron las siguientes etiquetas para el desarrollo del Dockerfile:

```
#Definicion de imagen a utilizar
FROM tiangolo/node-frontend:10 as build-stage

#Creamos la carpeta de trabajo
WORKDIR /app

#Copiamos el archivo de dependencias en la carpeta de trabajo
COPY package*.json /app/

#Instalamos las dependencias necesarias.
RUN npm install

#Copiamos el resto de archivos instalados
COPY ./ /app/

#Configuramos el build para producción y lo generamos
ARG configuration=production
RUN npm run build -- --output-path=./dist/out --configuration $configuration

#Instalamos el servidor de producción
FROM nginx:latest
COPY --from=build-stage /app/dist/out/ /usr/share/nginx/html
COPY ./nginx-custom.conf /etc/nginx/conf.d/default.conf

#Exponemos el servicio en el puerto 80
EXPOSE 80
```

Backend

En este dockerfile se utilizaron las siguientes etiquetas para el desarrollo del Dockerfile:

```
#Obtenemos la imagen de node necesaria para el servidor
FROM node:latest

#Creamos el entorno de trabajo
WORKDIR /src/app

#Copiamos el archivo de dependencias y las instalamos
COPY package*.json /src/app/
RUN npm install

#Copiamos los archivos descargados en la carpeta de trabajo
COPY . /src/app/

#Exponemos el servicio en el puerto 3000 y ejecutamos el archivo
EXPOSE 3000
CMD ["node", "index.js"]
```

Kubernetes

Para el despliegue de la aplicación se deben configurar varios archivos yaml, esto con el objetivo de definir las características dentro de nuestro cluster. Para esto seguiremos los pasos para el correcto despliegue de una aplicación en kubernetes.

1. Creación del cluster: El primer paso es la creación del cluster con un nodo, esto se realizara con el siguiente comando:

   gcloud container clusters create so2g26 --num-nodes=1

2. Conexión con el cluster: Para poder realizar los siguientes pasos para configurar los pods en el cluster primero se debe hacer la conexión a esta.

   kubectl expose deployment deploy-front-sopes2g26 --type=LoadBalancer --name=service-front-g26

3. Obtener los servicios: Para verificar que nos encontramos en el cluster podemos obtener los servicios que se encuentran actualmente.

   kubectl get services

4. Aplicar los cambios del archivo deploy-front-g26.yaml: Este archivo de configuración esta definido de la siguiente manera

```yaml
#Se define la versión que se quiere utilizar
apiVersion: apps/v1

#Se define el tipo
kind: Deployment
metadata:
  name: deploy-front-sopes2g26
  labels:
    app.kubernetes.io/name: deploy-front-sopes2g26
spec:
  #Se la cantidad de replicas o pods que se tendran
  replicas: 2
  selector:
    matchLabels:
        app.kubernetes.io/name: deploy-front-sopes2g26
  minReadySeconds: 20

  #Se define el tipo de estrategia a utilizar, en este caso es RollingUpdate
  strategy:
      rollingUpdate:
          maxSurge: 1
          maxUnavailable: 0
      type: RollingUpdate
  template:
    metadata:
      labels:
          app.kubernetes.io/name: deploy-front-sopes2g26
    spec:
      containers:
      #Se las imagenes que se encontraran en los pods
      - image: selvinlp/practica_frontso2g26:latest
        name: frontbalancer
        ports:
      #Se define el Puerto donde se encontrarán
        - containerPort: 80
```

5. Finalmente exponemos nuestro definiendo su tipo.

   kubectl expose deployment deploy-front-sopes2g26 --type=LoadBalancer --name=service-front-g26

6. Para verificar que se haya desplegado de manera correcta podemos utilizar el siguiente comando:

   kubectl get svc