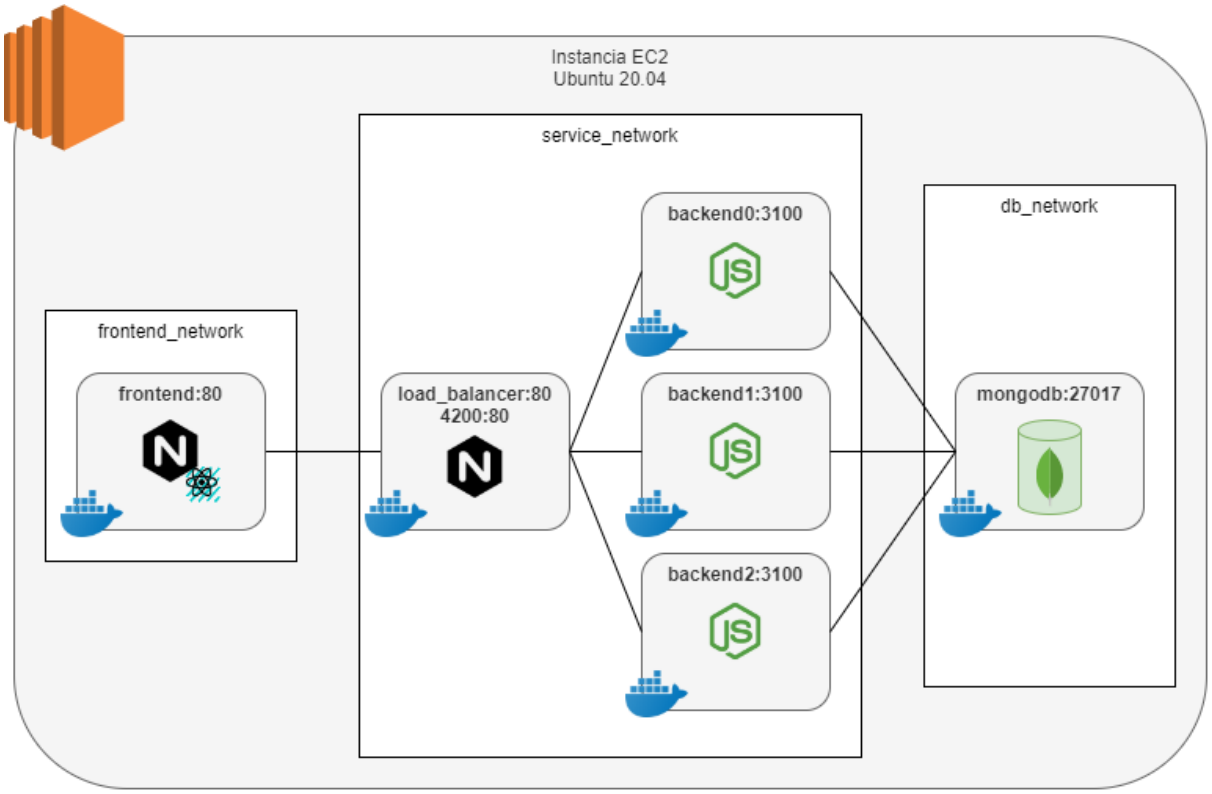
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS
REDES DE COMPUTADORAS 2 - A

PRACTICA 2

201503723 - Bryan Gustavo López Echeverría
201408470 - Sandra Eunice Jiménez Rodas
201602782 - Sergio Fernando Otzoy Gonzalez
201602440 - Jonathan Baudilio Hidalgo Pérez
201503746 - Raymundo Alexander Ixvalán Pacheco

Arquitectura de la aplicación



Descripción de la aplicación

Se usó el framework MERN para desarrollar la aplicación:

- ReactJS - Cliente
- NodeJS + Express.js - Servidor
- MongoDB - Base de datos

Dentro de la aplicación toda la comunicación se realiza usando el protocolo HTTP (a excepción de la base de datos) a través de los puertos que se especifica en el diagrama de arquitectura.

Toda la aplicación está montada en una instancia EC2 de AWS.

Redes

Todas las redes usan la misma configuración únicamente cambia la configuración de **subnet**. Cada una de estas redes se usan para diferentes capas de la aplicación.

service_network

Esta red es usada por los contenedores **backend0, backend1, backend2 y load_balancer**

```
  service_network:
    name: service_network
    driver: "bridge"
    ipam:
      driver: default
      config:
        - subnet: 172.35.78.0/24
```

db_network

Esta red es usada por el contenedor **mongodb**

```
  db_network:
    name: db_network
    driver: "bridge"
    ipam:
      driver: default
      config:
        - subnet: 10.10.18.0/24
```

frontend_network

Esta red es usada por el contenedor **frontend**

```
  frontend_network:
    name: frontend_network
    driver: "bridge"
    ipam:
      driver: default
      config:
        - subnet: 192.168.58.0/24
```

Contenedores

frontend_network

Frontend

Docker-compose

Se establece como **frontend** el nombre de este contenedor. Se le indica que deberá usar el Dockerfile que se encuentra en el subdirectorio **./Cliente**. Mapeará el puerto 80 al puerto 80 del host. Usará las redes **frontend_network** y **service_network**.

```
  frontend:
    container_name: frontend
    build: ./Cliente
    ports:
      - 80:80
    networks:
      - frontend_network
      - service_network
    links:
      - load_balancer
    environment:
      - REACT_APP_URL_API=load_balancer
```

Dockerfile

En el dockerfile se usan dos fases:

1. En la primera fase se construye en cliente web para generar la carpeta build
2. En la segunda fase se construye un contenedor nginx para alojar la aplicación que se construyó en la primera fase.

```
FROM node as build
WORKDIR /app
COPY package*.json /app/
RUN npm install
COPY ./ /app/
RUN npm run build

# Fase 2 - nginx
FROM nginx:latest
COPY --from=build /app/build/ /usr/share/nginx/html
```

service_network

Load Balancer

Docker-compose

Se establece como **load_balancer** el nombre de este contenedor. Expondrá el puerto 80 a la red y lo mapeará al puerto 80 del host. Usará las redes **frontend_network** y **service_network**. Crea un volumen cuyo contenido será el archivo de configuración de nginx.

Este contenedor no tiene un Dockerfile en cual basarse por lo que también se indica la imagen que deberá usar: **nginx.** Este contenedor dependerá que todos los contenedores de backend estén en línea en funcionando.

```
  load_balancer:
    container_name: load_balancer
    image: nginx:latest
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf:ro
    depends_on:
      - backend0
      - backend1
      - backend2
    expose:
      - 80
    ports:
      - 4200:80
    networks:
      - service_network
      - frontend_network
```

Configuración de nginx

```
events {}
http {
  upstream load_balancer {
      server backend0:3100 fail_timeout=10s max_fails=5;
      server backend1:3100 fail_timeout=10s max_fails=5;
      server backend2:3100 fail_timeout=10s max_fails=5;
  }
  server {
      listen 80;
      location / {
        proxy_pass http://load_balancer;
        add_header 'Access-Control-Allow-Origin' '*' always;
        add_header 'Access-Control-Allow-Credentials' 'true';
        add_header 'Access-Control-Allow-Methods'
'GET,POST,OPTIONS,PUT,DELETE';
      }
  }
}
```

Backend

Para el backend se usaron 3 contenedores para aumentar redundancia y asegurar disponibilidad, por lo que la siguiente explicación aplica para cada uno de esos contenedores, a excepción del nombre: **backend0, backend1, backend2**

Docker-compose

Se establece como **backend0** el nombre de este contenedor. Se le indica que deberá usar el Dockerfile que se encuentra en el subdirectorio **./servidor-api**. Expondrá el puerto 3100 a la red. Usará las redes **db_network** y **service_network**. Especifica tres variables de entorno que el servicio de backend usará para funcionar, entres ellas: puerto, host, identificador del server y la cadena de conexión de la base de datos.

La configuración restart indica que el contenedor únicamente podrá detenerse definitivamente si y solo sí se le indica. Es decir, si se apagara el host, o bien, el contenedor por alguna razón fallara, intentará inmediatamente volver a estar en línea.

```
backend0:
    container_name : backend0
    restart: always
    build: ./servidor-api
    expose:
      - 3100
    links:
      - mongodb
    networks:
      - service_network
      - db_network
    volumes:
      - /proc/:/elements/procs/
    environment:
      - PORT=3100
      - MONGO_URI=mongodb://mongodb:27017/redes2
      - HOST=0.0.0.0
      - SERVER_ID=201602782
```

Dockerfile

Se dockeriza el servidor de Node.JS y se indica que expondrá el puerto 3100. Por último ejecuta el servidor.

```
FROM node:latest
WORKDIR /usr/src/nodejs
COPY package*.json ./
RUN npm install
COPY . .

EXPOSE 3100
RUN mkdir -p /elements/procs
CMD ["node", "index.js"]
```

db_network

Base de datos

Docke compose

Se establece el nombre del contenedor como **mongodb**. Este contenedor no tiene un Dockerfile a usar por lo que se indica que deberá usar la imagen de mongo. Configura un volumen para asegurar que haya persistencia. Expone el volumen 27017 y configura la red a **db_network**

```
mongodb:
  image: 'mongo'
  container_name: mongodb
  environment:
      - PUID=1000
      - PGID=1000
  volumes:
    - /home/mongodb/database:/data/mongodb
  expose:
    - 27017
  restart: unless-stopped
  networks:
    - db_network
```