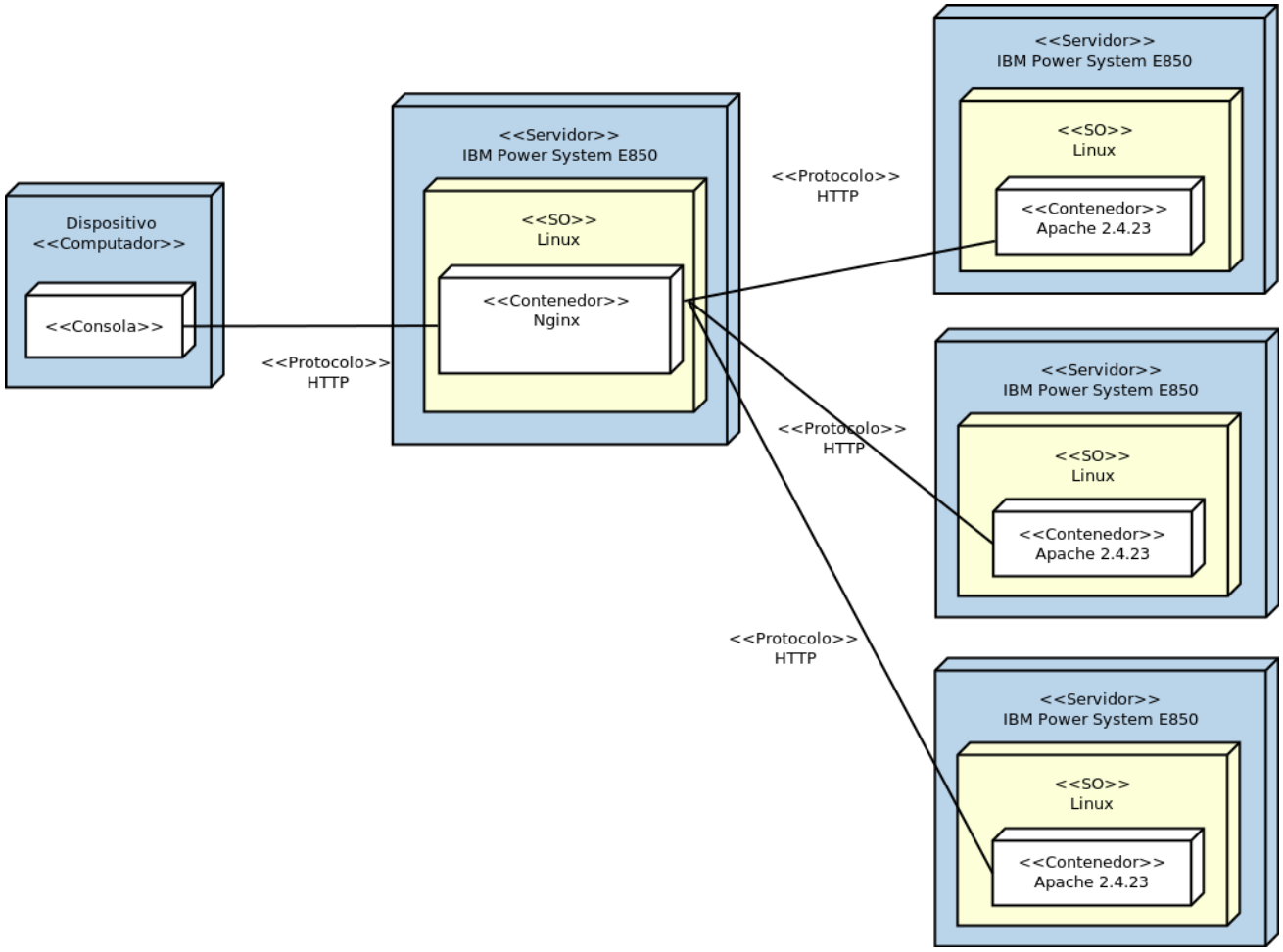
Examen 2

Universidad ICESI
Luisa María Vivas Mayor
Cód: A00276556
Curso: Sistemas Distribuidos
Docente: Daniel Barragán C.
Tema: Automatización de infraestructura (Docker)
Correo:luisa.vivas@correo.icesi.edu.co

Descripción

Deberá realizar el aprovisionamiento de un ambiente compuesto por los siguientes elementos: un servidor encargado de realizar balanceo de carga, tres servidores web con páginas estáticas. Se debe probar el funcionamiento del balanceador realizando peticiones y mostrando servidores distintos atendiendo las peticiones



En el repositorio de github del curso se encuentran ejemplos de docker y docker-compose los cuales pueden ser consultados para construir su solución.

Actividades

En un documento en formato PDF cuyo nombre de archivo debe ser examen2_codigoestudiante.pdf debe incluir lo siguiente:

1. Documento en formato PDF:

- Formato PDF (5%)
- Nombre y código de los integrantes del grupo (5%)
- Ortografía y redacción (5%)

2. Consigne los comandos de linux necesarios para el aprovisionamiento de los servicios solicitados. En este punto no debe incluir archivos tipo Dockerfile solo se requiere que usted identifique los comandos o acciones que debe automatizar (15%)
3. Escriba los archivos Dockerfile para cada uno de los servicios solicitados junto con los archivos fuente necesarios. Tenga en cuenta consultar buenas prácticas para la elaboración de archivos Dockerfile. (20%)
4. Escriba el archivo docker-compose.yml necesario para el despliegue de la infraestructura (10%)
5. Publicar en un repositorio de github los archivos para el aprovisionamiento junto con un archivo de extensión .md donde explique brevemente como realizar el aprovisionamiento (15%)
6. Incluya evidencias que muestran el funcionamiento de lo solicitado (15%)
7. Documente algunos de los problemas encontrados y las acciones efectuadas para su solución al aprovisionar la infraestructura y aplicaciones (10%)

Referencias

- https://github.com/rhcarvalho/byzanz-gui
- https://docs.docker.com/

Solución:

1. Consigne los comandos de linux necesarios para el aprovisionamiento de los servicios solicitados:

Para la realización del aprovisionamiento de un ambiente compuesto por los siguientes elementos:

● Servidor de balanceo de carga.

● Tres Servidores web.

Se aplicó los conceptos vistos en clase, por tanto, se hizo uso de Docker para proveer el ambiente requerido.

Balanceo de carga:

Para el balanceador de carga como ya hemos mencionado antes, se hizo uso de nginx, es una aplicación que es capaz de hacer funciones de balanceo de carga entre varios servidores de aplicaciones, por tanto, siguiendo a guía brindada por nginx.org pasamos a instalarlo y configurarlo de la siguiente manera:

- Definir el puerto por el cual esté recibirá las peticiones y las redirige hacia las maquinas web, el cual será el puerto 8080, además de esto el puerto debe abrirse y dar los permisos necesarios para el cortafuego:

  Iptables –I INPUT 5 –p tcp –m state – new tcp –dport 8080 –j ACCEPT

- Se realiza la instalación de Nginx:

  Sudo yum install nginx

- Nginx tiene un archivo de configuración en la ruta /etc/nginx/nginx.conf, este archivo ya contiene una configuración la cual no será útil para nuestra solución, por tanto, se debe modificar este archivo de la siguiente manera:

```
worker_processes 4;

events { worker_connections 1024; }

http {
    sendfile on;

    upstream app_servers {
        server web:80;
        server web1:80;
        server web2:80;
    }

    server {
        listen 80;

        location / {
            proxy_pass         http://app_servers;
            proxy_redirect     off;
            proxy_set_header   Host $host;
            proxy_set_header   X-Real-IP $remote_addr;
            proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header   X-Forwarded-Host $server_name;
        }
    }
```

El anterior archivo muestra cómo se definen los servidores a los cuales el balanceador apunta y el respectivo puerto por donde escucha aquellos servicios, que es el 8080. Y el puerto 80 es el que se encargara de recibir las peticiones http.

Web:

Se instala el servicio web de apache

Sudo apt install httpd –y

Y se abre el puerto por donde recibe las peticiones http que será el 80.

iptables -I INPUT 5 -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT

2. Escriba los archivos Dockerfile para cada uno de los servicios solicitados junto con los archivos fuente necesarios:

Balanceador de caga:

Para el balanceador de carga tenemos un Dockerfile, que se encarga de reemplazar el archivo de configuración del servicio nginx y reiniciar el servicio con la configuración definida en el nginx.conf, que se mencionó en el punto anterior, cuyo fin es mapear los servidores web que atenderán las peticiones.

```
FROM nginx

MAINTAINER luisamaria556@gmail.com

# Remove the default Nginx configuration file
RUN rm -v /etc/nginx/nginx.conf

# Copy a configuration file from the current directory
ADD nginx.conf /etc/nginx/

# Append "daemon off;" to the beginning of the configuration
RUN echo "daemon off;" >> /etc/nginx/nginx.conf


# Set the default command to execute
# when creating a new container
CMD service nginx start
```

Web:

Se tienen tres servidores web y cada uno contiene dentro de las respectivas carpetas un Dockerfile y el index.html, este último se encarga de mostrar la vista de cada uno de los servidores:

Dockerfile:

```
FROM httpd

MAINTAINER luisamaria556@gmail.com

ADD index.html /usr/local/apache2/htdocs/index.html
```

Este archivo permite copiar un archivo desde el ordenador central en el contenedor y ubicar el archivo index.html ubicado dentro de la carpeta web, que contiene la vista se cada servidor:

```
<!DOCTYPE html>
<html>
<head>
<title>Examen 2</title>
</head>
<body>

<h1>Nginx 1</h1>
<h2>Servidor 2</h2>

</body>
</html>
```

3. Escriba el archivo docker-compose.yml necesario para el despliegue de la infraestructura

```
version: '2'

services:
  web:
    build:
      context:  ./web
      dockerfile: Dockerfile
    expose:
      - "5000"
    volumes:
      - web_volumes:/web_volumes

  web1:
    build:
      context:  ./web1
      dockerfile: Dockerfile
    expose:
      - "5000"
```

```
    volumes:
      - web_volumes:/web_volumes

  web2:
    build:
      context:  ./web2
      dockerfile: Dockerfile
    expose:
      - "5000"
    volumes:
      - web_volumes:/web_volumes

  proxy:
    build:
      context:  ./nginx
      dockerfile: Dockerfile
    ports:
      - "8080:8080"
    links:
      - web
      - web1
      - web2
    volumes:
      - nginx_volumes:/nginx_volumes

volumes:
    web_volumes:

    nginx_volumes:
```

En este archivo, se crea los 3 servidores web y el balanceador de carga. Por tanto, lo servidores web utilizan el Dockerfile del contexto web1 ,web2, web3 definido para cada uno y el balanceador, el Dockerfile del contexto nginx. Además de esto, se define el puerto 5000 para los contenedores web y el mapeo del puerto 8080:80 por donde recibirá las peticiones.

Respecto a los volúmenes, se debe crear dos volúmenes y asignarlos respectivamente. Un volumen encargado para el balanceador de carga y otro que es compartido para el servicio web.

4. Incluya evidencias que muestran el funcionamiento de lo solicitado:

Para comprobar el correcto funcionamiento del proyecto, se debe construir el contenedor primero:

```
docker-compose build
```

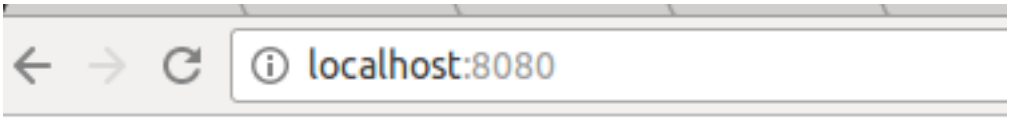
Seguido se agrega e inicia el contenedor:

docker-compose up

```
distribuidos@redes1:~/Documentos/Distribuidos/Talleres/Taller1/Examenes/Examen2$
 docker-compose build
Building web2
Step 1 : FROM httpd
 ---> ef0aca83ba5a
Step 2 : ADD index.html /usr/local/apache2/htdocs/index.html
 ---> Using cache
 ---> 43b7f79780ec
Successfully built 43b7f79780ec
Building web
Step 1 : FROM httpd
 ---> ef0aca83ba5a
Step 2 : MAINTAINER luisamaria556@gmail.com
 ---> Running in cca0ef83522d
 ---> 56d52f947c12
Removing intermediate container cca0ef83522d
Step 3 : ADD index.html /usr/local/apache2/htdocs/index.html
 ---> c0d124ad46e3
Removing intermediate container ecdfbabd59df
Successfully built c0d124ad46e3
Building web1
Step 1 : FROM httpd
 ---> ef0aca83ba5a
Step 2 : ADD index.html /usr/local/apache2/htdocs/index.html
 ---> Using cache
 ---> c5e9ecb1cc0f
Successfully built c5e9ecb1cc0f
Building proxy
Step 1 : FROM nginx
 ---> 46102226f2fd
Step 2 : MAINTAINER luisamaria556@gmail.com
 ---> Running in 9d68928ab68d
 ---> 91b8160415b4
Removing intermediate container 9d68928ab68d
Step 3 : RUN rm -v /etc/nginx/nginx.conf
 ---> Running in 9c245b5f754e
removed '/etc/nginx/nginx.conf'
 ---> a2b85e7806c3
Removing intermediate container 9c245b5f754e
Step 4 : ADD nginx.conf /etc/nginx/
 ---> 47a66ec8932b
Removing intermediate container c5383b693a9a
Step 5 : RUN echo "daemon off;" >> /etc/nginx/nginx.conf
 ---> Running in 679e7dc73674
 ---> 3a5d829acb8f
Removing intermediate container 679e7dc73674
Step 6 : CMD service nginx start
 ---> Running in 636d84da431d
 ---> 0a422ae95d7c
Removing intermediate container 636d84da431d
Successfully built 0a422ae95d7c
```

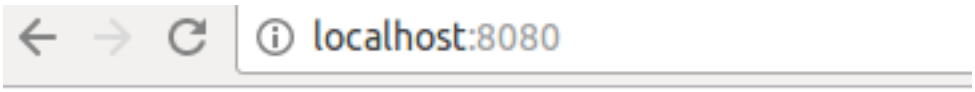
```
distribuidos@redes1:~/Documentos/Distribuidos/Talleres/Taller1/Examenes/Examen2$
Recreating examen2_web1_1
Recreating examen2_web2_1
Recreating examen2_web_1
Recreating examen2_proxy_1
Attaching to examen2_web2_1, examen2_web_1, examen2_web1_1, examen2_proxy_1
web2_1    | AH00558: httpd: Could not reliably determine the server's fully quali
web2_1    | AH00558: httpd: Could not reliably determine the server's fully quali
web2_1    | [Mon May 08 16:46:28.327630 2017] [mpm_event:notice] [pid 1:tid 14002
web2_1    | [Mon May 08 16:46:28.327686 2017] [core:notice] [pid 1:tid 1400291183
web_1     | AH00558: httpd: Could not reliably determine the server's fully quali
web_1     | AH00558: httpd: Could not reliably determine the server's fully quali
web1_1    | AH00558: httpd: Could not reliably determine the server's fully quali
web_1     | [Mon May 08 16:46:28.347726 2017] [mpm_event:notice] [pid 1:tid 14033
web_1     | [Mon May 08 16:46:28.347776 2017] [core:notice] [pid 1:tid 1403336926
web1_1    | AH00558: httpd: Could not reliably determine the server's fully quali
web1_1    | [Mon May 08 16:46:28.430526 2017] [mpm_event:notice] [pid 1:tid 14068
web1_1    | [Mon May 08 16:46:28.430578 2017] [core:notice] [pid 1:tid 1406866190
```

Evidencias:



Nginx 1

Servidor 1



Nginx 2

Servidor 2

Nginx 3

Servidor 3

5. Documente algunos de los problemas encontrados y las acciones efectuadas para su solución al aprovisionar la infraestructura y aplicaciones

Dentro de la solución del aprovisionamiento compuesto, cuando el proyecto se subía el proyecto, no lo haca debido a que mandaba una excepción que mostraba como si ya se estuviera usando los puertos, por tanto fue necesario borrar las imágenes y los contenedores creados, con los siguientes comandos:

docker rm –f $(docker ps -q)→ elimina todos los contenedores
docker rmi $(docker images -q)→ elimina todas las imágenes