phalcon30964 / **examen2-SistemasDistribuidos-ChristianDavidLopezMorcillo**

No description, website, or topics provided.

Edit

Add topics

13 commits **1** branch **0** releases **1** contributor

phalcon30964 update Latest commit `313139f` 39 seconds ago

Enunciado	Update	3 days ago
Evidencia	Final update	3 days ago
Nginx	Update	3 days ago
Web-Apache2	Final update	3 days ago
README.md	update	39 seconds ago
docker-compose.yml	update	3 minutes ago

README.md

xamen1-SistemasDistribuidos-ChristianDavidLopezMorcillo

Christian David López Morcillo A00312096

EXAMEN 2 - Sistemas Distribuidos

1. 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.

Para el balanceador de cargas:

- Se escoge usar el programa Nginx balanceando cargas bajo el esquema round robin.

Primero se instala el repositorio necesario para poder instalar Ngix, esto se realiza agregando a los repositorios de yum un archivo. repo que indique la ruta para descargar nginx. El archivo se debe agregar en la ruta “/etc/yum.repos.d/” y debe tener la información:

```
[nginx]
name=nginx repo
baseurl=http://nginx.org/packages/centos/$releasever/$basearch/
gpgcheck=0
enabled=1
```

- Se realiza la instalación de Nginx usando yum.

```
sudo yum install nginx
```

- Se crea el archivo /etc/nginx/nginx.conf y allí se especifica la ip de los servidores que atenderán las peticione, de la siguiente forma:

```
http {
    upstream webservers {
         server 192.168.131.126;
         server 192.168.131.127;
         server 192.168.131.128;
    }
    server {
        listen 8080;
        location / {
              proxy_pass http://webservers;
        }
    }
}
```

- Se agregan los permisos necesarios para el cortafuego.

```
 iptables -I INPUT 5 -p tcp -m state -- NEW -m tcp --dport 8080 -j ACCEPT
 iptables -I INPUT 5 -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT
 service iptables save
```

- Se inicia el servicio Nginx.

```
sudo service nginx start
```

Para el servidor web:

- Se instala el servicio web apache:

```
sudo yum install httpd
```

- Se agregan los permisos necesarios para el cortafuegos.

```
 iptables -I INPUT 5 -p tcp -m state --state NEW -m tcp --dport 8080 -j ACCEPT
 iptables -I INPUT 5 -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT
 service iptables save
```

- Se agrega el archivo index.html a la ruta var/www/html/. Este archivo tiene la función de consultar la base de datos a través de métodos php. El archivo php debe tener el siguiente código:

```
<HTML>
<BODY>
Servidor X
</BODY>
</HTML>
```

- Se inicia el servicio web.

```
sudo service httpd start
```

2. 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.

Para el balanceador de cargas:

El balanceador de cargas usado para esta infraestructura es nginx. Se crea una carpeta llamada Nginx que servirá de contexto para el contenedor de nginx. Este contenedor utiliza la imagen nginx del dockerhub. Se escribe el archivo Dockerfile que básicamente remplaza el archivo de configuración del servicio nginx y luego inicia el servicio de nginx. También se escribe un archivo nginx.conf que se encarga de mapear las direcciones de los servidores web que atenderán peticiones.

Se escribe el siguiente dockerfile:

```
# Se pasa la imagen base
FROM nginx

# Autor y mantenedor
MAINTAINER Christian David López Morcillo

# Eliminar el archivo de configuración por defecto de Nginx
RUN rm -v /etc/nginx/nginx.conf

# Copiar el archivo de configuración desde el directorio actual
ADD nginx.conf /etc/nginx/

# Agregar "daemon off;" en el comienzo de la configuración
RUN echo "daemon off;" >> /etc/nginx/nginx.conf

# Setear el comando de defecto para ejecutar
# cuando se crea un nuevo contenedor
CMD service nginx start
```

Y se escribe el archivo de configuración de nginx:

```
worker_processes 3;

events { worker_connections 1024; }

http {
   sendfile on;

   # Lista de los servidores de aplicación
   upstream app_servers {
       server server1;
       server server2;
       server server3;
   }

   # Configuración del servidor
   server {

   # Puerto de ejecución
       listen 80;

       # Proxy de las conexiones
       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;
       }
   }
}
```

Para el servidor web:

Para la instalación de los 3 servidores web, se usa como servidor de aplicaciones apache. Además, se usa confd para inyección de variables en el html del apache. Confd es una aplicación para inyección de variables en archivos de

configuración usando templates y variables del sistema.

Primero se escribe el dockerfile para la instalación del apache. Además, se ejecuta el stript start.sh que inyecta variables al html que expondrá el apache.

```
FROM httpd
MAINTAINER Christian López

ADD files/confd-0.10.0-linux-amd64 /usr/local/bin/confd
ADD files/start.sh /start.sh

RUN chmod +x /usr/local/bin/confd
RUN chmod +x /start.sh

ADD files/confd /etc/confd

CMD ["/start.sh"]
```

Luego se escribe el archivo start.sh que se encarga de inyectar la variable encargada de escribir el id del servidor apache a un htlm para poder identificar el servidor de aplicaciones cuando se le hagan peticiones.

start.sh

```
#!/bin/bash
set -e

# if $proxy_domain is not set, then default to $HOSTNAME
export id_server=${id_server:-"Error, no se pudo setear el parámetro "}

# ensure the following environment variables are set. exit script and container if not set.
test $id_server

/usr/local/bin/confd -onetime -backend env

echo "Starting Apache"
exec httpd -DFOREGROUND
```

Luego se escribe un archivo .toml que se encarga de definir donde deberá ser ubicado el template del htlm.

index.html.toml

```
[template]
src = "index.html.tmpl"
dest = "/usr/local/apache2/htdocs/index.html "
```

Tambien se escribe un template del index.html.

index.html.tmpl

```
Este es el servidor numero: {{ getenv "id_server" }}
```

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

- Para el despliegue de la infraestructura planteada, se crea en un docker-compose.yml para 3 servidores web llamados cada uno serverX , donde X es el id del servidor, y un balanceador de cargas llamado nginx.
- Los servidores web utilizan el dockerfile del contexto Web-Apache2 y el balanceador de cargas utiliza el dockerfile del contexto Nginx.
- Para los contenedores web se hace apertura del puerto 5000. Para el balanceador de cargas se hace mapeo del puerto 8080 del host con el 80 del contenedor.

- Para cada uno de los servidores web se crea la variable id_server que se usa como variable de entorno del sistema para inyectar variables al html que se instalará en el apache2.

- Adicionalmente se crean y se asignan 2 volumenes, un volumen para el balanceador de cargas llamadao nginx_volume y otro compartido por todos los servidores web llamado apache_volume.

Se detalla el docker-compose final a continuación:

```
version: '2'

services:
  server1:
    build:
      context:  ./Web-Apache2
      dockerfile: Dockerfile
    environment:
      - id_server=1
    volumes:
      - apache_volume1:/usr/local/apache2/htdocs/

  server2:
    build:
      context:  ./Web-Apache2
      dockerfile: Dockerfile
    environment:
       - id_server=2
    volumes:
       - apache_volume2:/usr/local/apache2/htdocs/

  server3:
    build:
      context:  ./Web-Apache2
      dockerfile: Dockerfile
    environment:
      - id_server=3
    volumes:
      - apache_volume3:/usr/local/apache2/htdocs/

  nginx:
    build:
      context:  ./Nginx
      dockerfile: Dockerfile
    ports:
      - "8080:80"
    volumes:
      - nginx_volume:/etc/nginx/

volumes:
    apache_volume1:
    apache_volume2:
    apache_volume3:
    nginx_volume:
```

4. 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.

- Se publica el examen en el repositorio https://github.com/phalcon30964/examen2-SistemasDistribuidos-ChristianDavidLopezMorcillo

5. Incluya evidencias que muestran el funcionamiento de lo solicitado

- Se muestra capturas de 3 accesos a al balanceador en la carpeta evidencias, podemos ver como el balanceador redirige la petición a un servidor diferente en cada ocasión.

Figura 1: Primer acceso al balanceador

Figura 2: Segundo acceso al balanceador

Figura 3: Tercer acceso al balanceador

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

- Problema 1: Los servidores web no podían ser accedidos desde otras máquinas. Solución 1: Se agregó a iptables las configuraciones necesarias para abrir los puertos que apache necesita para recibir peticiones.
- Problema 2: Un volumen no podian añadirse a los servicios. Solución 2: Se crearon volumenes separados para cada servicio. De esta forma, cada contenedor tiene su volumen propio.
- Problema 3: El navegador solo cargaba 1 sola vez la pagina del servidor 3. Solución 3: Se deshabilitó el cacheo y se notó que no se mostraba porque se hacia cacheo del archivo del servidor 2.
- Problema 3: En el dockerfile el nombre de un service no dejaba tener mayusculas. Solución 3: Se puso minusculas todo los nombres de los contenedores.
- Problema 4: El balanceador de cargas no pidia ser accedido. Solución 4: Se agregó el comando ports dentro del docker-compose.yml para exponer el puerto 80 del contenedor al puerto 8080 del host.