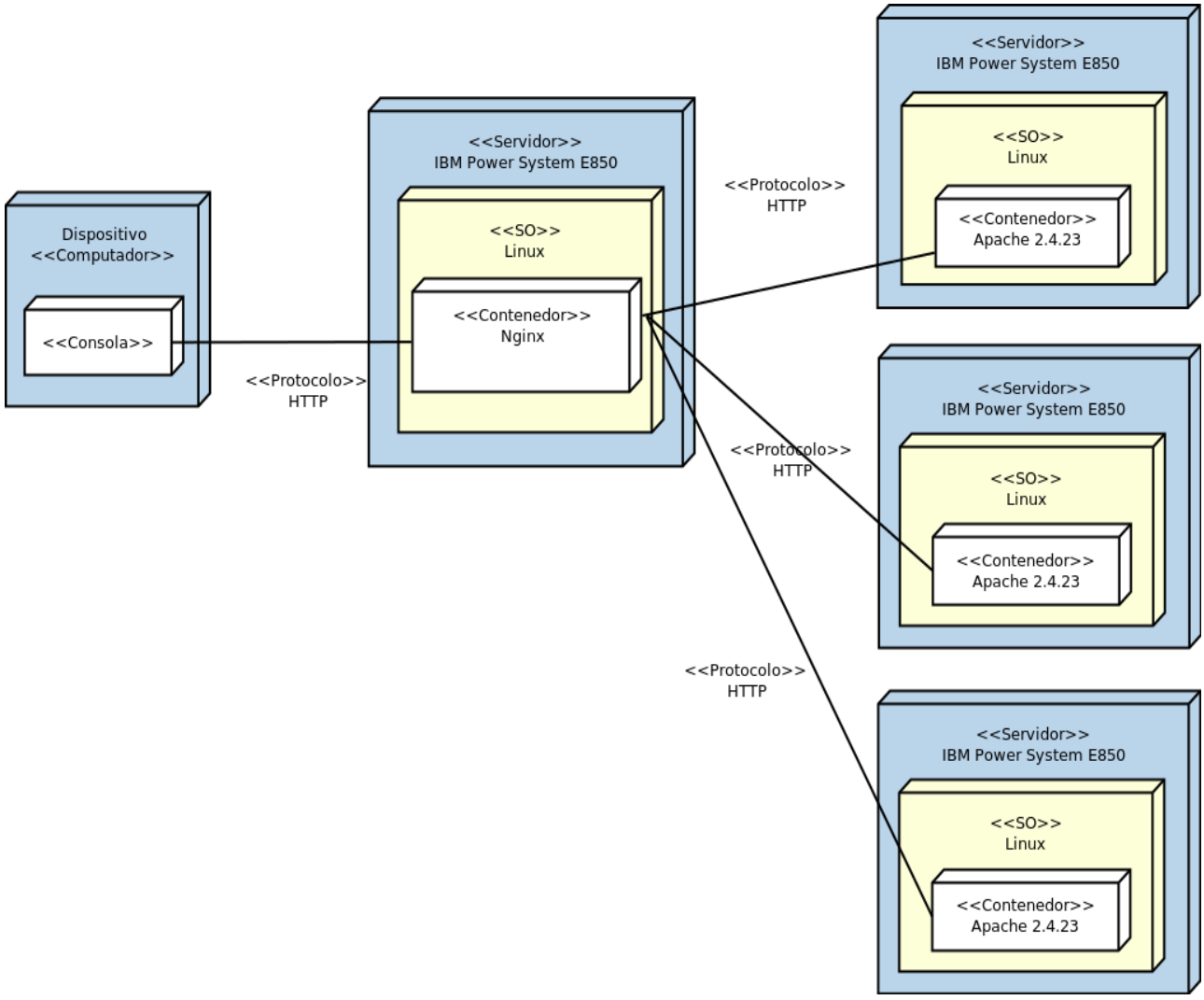
DockerLoadBalancer

Autor: Johan David Ballesteros
Código: A00309824
Repositorio: https://github.com/DavidPDP/DockerLoadBalancer

Problema

Se debe automatizar el despliegue de una infraestructura que posea unos contenedores web y un contenedor que se encargue del balanceo de cargas. Esto se puede observar en el siguiente diagrama de deployment:



Objetivos

- Realizar de forma autónoma el aprovisionamiento automático de infraestructura.
- Diagnosticar y ejecutar de forma autónoma las acciones necesarias para lograr infraestructuras estables.
- Integrar servicios ejecutándose en nodos distintos.

Prerrequisitos

- Docker

Pasos Para Automatizar

Para el despliegue de la infraestructura se necesita automatizar las siguientes acciones:

Servidores Web

Para desplegar un servidor web de apache se necesita instalar apache:

```
sudo apt-get update
sudo apt-get install apache2
```

Después de instalarlo se inicia el servicio

```
sudo service apache2 start
```

Balanceador De Carga

Para desplegar el balanceador de carga se necesita instalar nginx:

```
sudo apt-get update
sudo apt-get install Nginx
```

Después proceder a configurar Nginx como balanceador desde el archivo nginx.conf que se encuentra en la ruta /etc/nginx/ como el ejemplo que se encuentra a continuación:

```
http {
    upstream myapp1 {
        server srv1.example.com;
```

```
        server srv2.example.com;
        server srv3.example.com;
    }

    server {
        listen 80;

        location / {
            proxy_pass http://myapp1;
        }
    }
}
```

Finalmente, iniciar el servicio de Nginx

```
service nginx start
```

Levantar Contenedor

Para levantar un solo contenedor se debe proceder a indicarle que se ejecute en modo detached, realizar el mapeo de puertos y asignarle un volumen al contenedor como se puede observar en el siguiente comando:

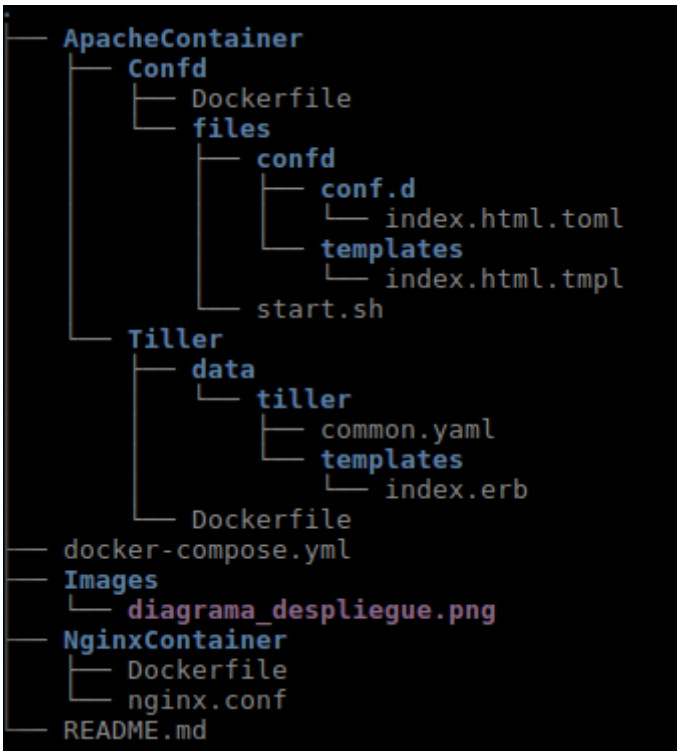
```
docker run -d -p 80:80 -v /webapp my_image
```

Automatizado

Para automatizar todo el despliegue de la infraestructura propuesta teniendo en cuenta los pasos a automatizar se procedió a las siguientes acciones:

Vista General De La Estructura Del Proyecto

A continuación se presenta la vista general de la estructura del proyecto:



Servidor Web

Para crear los 3 servidores web se procedió a elegir httpd para esto se descargó desde DockerHub la imagen con el siguiente comando:

```
docker pull httpd
```

Surge un problema con Docker y es la parametrización de los archivos en el tiempo de ejecución del contenedor. Esto se debe a que en tiempo de construcción se invierte un tiempo considerable para crear la imagen, si este tiempo es empleado para paremetrizar los contenedores, pues habrá un desperdicio de recursos debido a que cada vez que se quiera cambiar la parametrización se deberá construir la imagen (teniendo en cuenta que en la construcción de la imagen se instala los sistemas robustos del contenedor). Un ejemplo de esto se puede observar si queremos que los contenedores desplegados accedan a una base de datos o a diferentes servicios REST.
Es por lo anterior que el momento óptimo para realizar la parametrización es cuando se va a ejecutar un contenedor (Docker run time), además sería lo más conveniente puesto que si se quiere ejecutar múltiples contenedores cada uno tiene la posibilidad de llevar una parametrización diferente. A este problema se le añade que Docker no ofrece esta funcionalidad por lo que se tiene que recurrir a herramientas externas que permitan realizarlo. En este caso se seleccionó Confd, la cual es una herramienta

de gestión de la configuración de peso ligero. (Más adelante se explicará los problemas por los que se seleccionó esta herramienta y no otra).

Condiguración de Confd

Para realizar la configuración de Confd se debe proceder primero a realizar su instalación. Para esto se utilizará la imagen base (httpd) descargada con el comando mencionado anteriormente. A continuación se muestra el Dockerfile empleado para configurar Confd.

DockerFile servidor web con Confd

En el Dockerfile podemos observar cuatro instrucciones fundamentales. La primera la instalación de Confd y ubicación en la carpeta /usr/local/bin/confd. La segunda es el copiado que se realiza sobre un archivo .sh, que se encarga de mostrar todas las variables del entorno, de definir las variables a parametrizar, de probarlas y finalmente de iniciar el servicio del servidor web. La tercera la de asignación de permisos y finalmente el comando de ejecución del contenedor el cual se encarga de ejecutar el archivo sh. A continuación se puede ver en más detalle este archivo

start.sh

Por último se procede a definir en los archivos que se quieren parametrizar la siguiente estructura.

1. Los archivos que vayan a contener las variables deben tener extensión .tmpl, por organización se almacenan en una carpeta llamada templates.
2. Para definir una variable dentro del archivo se debe definir con el siguiente formato.

```
{{ getenv "[NameVariable]" }} // Nota los corchetes solo son delimitadores para el ejemplo se deben obviar al momento de establecer la variable.
```

3. Finalmente se debe crear un archivo con la extensión .toml donde se le específica a la herramienta donde se encuentra el archivo templates y donde se almacenarán dentro del contenedor.

Con lo anterior se finaliza la configuración y preparación de la herramienta Confd y de los archivos con las variables definidas para la parametrización que se realizará en Docker run time.

Balanceador De Cargas

Para el balanceador de cargas se hizo uso de Nginx, un servidor HTTP el cual puede ser configurado para realizar esta funcionalidad. Para esto se descargó la imagen de nginx con el siguiente comando:

```
docker pull nginx
```

Debido a que la imagen ya contiene el nginx previamente instalado entonces los únicos pasos a realizar sería la configuración del mismo como balanceador de cargas. A continuación se muestra la el archivo de configuración del nginx:

nginx.conf

En este archivo se definen los servidores a los cuales el balanceador puede redireccionar las peticiones y configura el Nginx para que pueda recibir conexiones remotas.

Finalmente se crea el Dockerfile teniendo la imagen base (nginx) descargada anteriormente y se procede a cambiar el archivo de configuración por defecto de Nginx por el nuevo que configura al Nginx como balanceador de cargas. También se agrega al archivo de configuración el comando **daemon off** que permita que el Nginx se ejecute en foreground y no se detenga.

Dockerfile Nginx

Automatización Infraestructura

Una vez realizado todos los pasos anteriores ya se puede automatizar el despliegue de la infraestructura deseada. El primer problema que se encuentra aquí es el despliegue por comando de cada contenedor. Como podemos ver a continuación se debería realizar los siguientes comandos cada vez que se quisiera levantar la infraestructura deseada:

```
docker run -d -p 5000:80 -e server_number="1" apache_confd
docker run -d -p 5000:80 -e server_number="2" apache_confd
docker run -d -p 5000:80 -e server_number="3" apache_confd
docker run -d -p 8080:80  dockerloadbalancer_proxy
```

A esto añadiéndole la creación de los volúmenes, asignación misma a los comandos y la escalabilidad del sistema.

Para solucionar esto se procede a crear el compose que nos permitirá el despliegue de cada uno de los contenedores, además que permite asignarle las variables del entorno que se setearan dentro de los archivos por medio de la herramienta Confd.

docker-compose.yml

Para ejecutar el docker compose se sigue procede a ejecutar los siguientes comandos:

```
docker-compose build --no-cache
docker-compose up
```

Gestión de Volúmenes

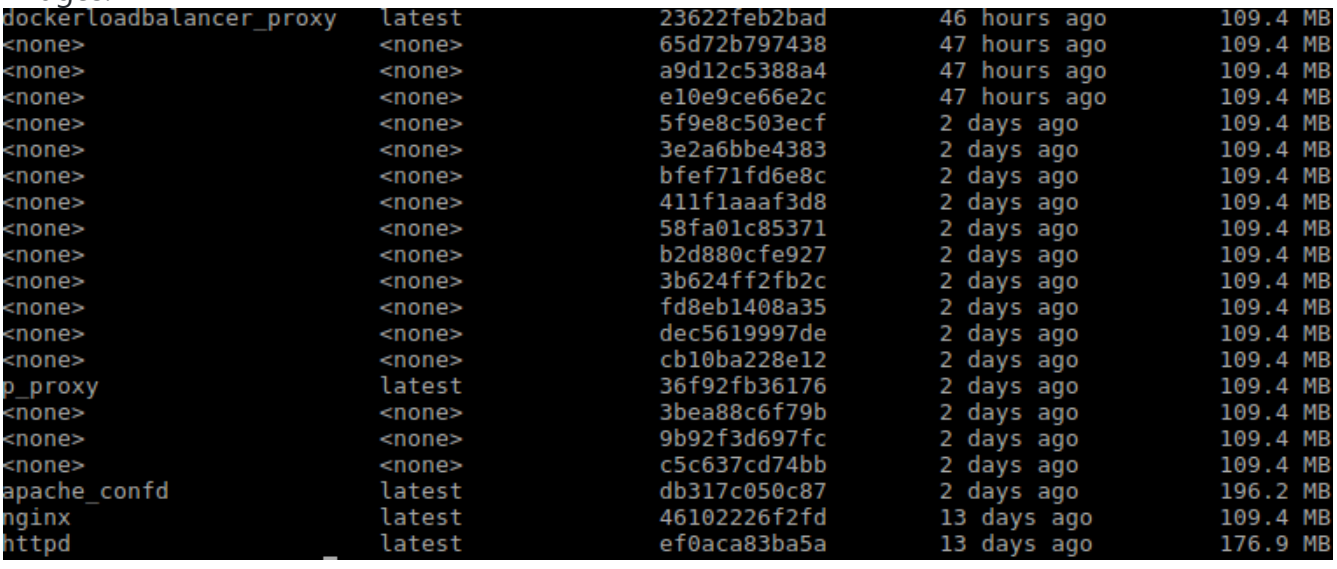
Para la gestión de los volúmenes de los contenedores se procedió a definir que los contenedores web compartirán un mismo volumen para el almacenamiento de datos o de archivos que sean relevantes como los de configuración, mientras que el contenedor del balanceador se le asignó un volumen diferente para agregar un poco de seguridad. La creación de los volúmenes se encuentra en la misma definición del archivo docker-compose.yml y se definen como se sigue:

```
volumes:
    [NameVolume]:
//Dentro de la declaración de los contenedores
volumes:
      - [NameVolume]:[Path]
```

Resultados

A continuación se muestran los pantallazos del funcionamiento de la solución y de aspectos relevantes:

Docker images:

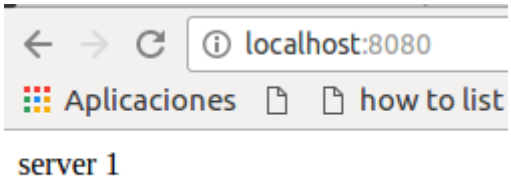


Docker volumes:

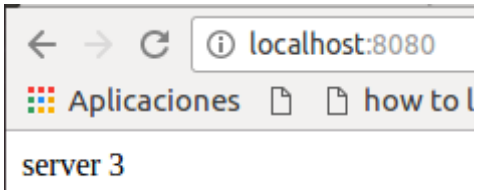
```
DRIVER              VOLUME NAME
local               dockerloadbalancer_apache_data
local               dockerloadbalancer_nginx_data
```

Ejecución Docker-compose:

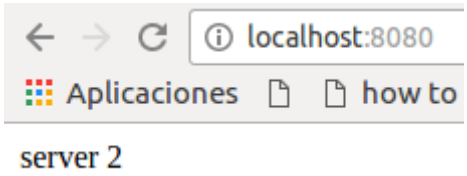
```
Starting dockerloadbalancer_webServer1_1
Starting dockerloadbalancer_webServer3_1
Starting dockerloadbalancer_webServer2_1
Recreating dockerloadbalancer_proxy_1
Attaching to dockerloadbalancer_webServer1_1, dockerloadbalancer_webServer3_1, dockerloadbalancer_webServer2_1, dockerloadbalancer_proxy_1
webServer1_1  | 2017-05-08T22:12:22Z 225be145601c /usr/local/bin/confd[7]: INFO Backend set to env
webServer1_1  | 2017-05-08T22:12:22Z 225be145601c /usr/local/bin/confd[7]: INFO Starting confd
webServer1_1  | 2017-05-08T22:12:22Z 225be145601c /usr/local/bin/confd[7]: INFO Backend nodes set to
webServer3_1  | 2017-05-08T22:12:22Z df489a52680f /usr/local/bin/confd[7]: INFO Backend set to env
webServer3_1  | 2017-05-08T22:12:22Z df489a52680f /usr/local/bin/confd[7]: INFO Starting confd
webServer3_1  | 2017-05-08T22:12:22Z df489a52680f /usr/local/bin/confd[7]: INFO Backend nodes set to
webServer3_1  | Starting Apache
webServer2_1  | 2017-05-08T22:12:22Z 329acb6673ff /usr/local/bin/confd[7]: INFO Backend set to env
webServer2_1  | 2017-05-08T22:12:22Z 329acb6673ff /usr/local/bin/confd[7]: INFO Starting confd
webServer2_1  | 2017-05-08T22:12:22Z 329acb6673ff /usr/local/bin/confd[7]: INFO Backend nodes set to
webServer1_1  | Starting Apache
webServer3_1  | AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.20.0.3. Set the 'ServerName' directive globally to suppress this message
webServer2_1  | Starting Apache
webServer2_1  | AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.20.0.4. Set the 'ServerName' directive globally to suppress this message
webServer2_1  | AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.20.0.4. Set the 'ServerName' directive globally to suppress this message
webServer3_1  | AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.20.0.3. Set the 'ServerName' directive globally to suppress this message
webServer3_1  | [Mon May 08 22:12:22.333054 2017] [mpm_event:notice] [pid 1:tid 140457720579968] AH00489: Apache/2.4.25 (Unix) configured -- resuming normal operations
webServer3_1  | [Mon May 08 22:12:22.333096 2017] [core:notice] [pid 1:tid 140457720579968] AH00094: Command line: 'httpd -D FOREGROUND'
webServer2_1  | [Mon May 08 22:12:22.689177 2017] [mpm_event:notice] [pid 1:tid 140410443061120] AH00489: Apache/2.4.25 (Unix) configured -- resuming normal operations
webServer2_1  | [Mon May 08 22:12:22.689218 2017] [core:notice] [pid 1:tid 140410443061120] AH00094: Command line: 'httpd -D FOREGROUND'
webServer1_1  | AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.20.0.2. Set the 'ServerName' directive globally to suppress this message
webServer1_1  | AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.20.0.2. Set the 'ServerName' directive globally to suppress this message
webServer1_1  | [Mon May 08 22:12:22.333043 2017] [mpm_event:notice] [pid 1:tid 140198743820160] AH00489: Apache/2.4.25 (Unix) configured -- resuming normal operations
webServer1_1  | [Mon May 08 22:12:22.333085 2017] [core:notice] [pid 1:tid 140198743820160] AH00094: Command line: 'httpd -D FOREGROUND'
```



Redireccionamiento Server 1:



Redireccionamiento Server 3:



Redireccionamiento Server 2:

Peticiones Recibidas Por El Balanceador:

```
proxy_1       | 172.20.0.1 - - [08/May/2017:22:13:31 +0000] "GET / HTTP/1.1" 200 9 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/55.0.2883.87 Safari/537.36"
webServer1_1  | 172.20.0.5 - - [08/May/2017:22:13:31 +0000] "GET / HTTP/1.0" 200 9
webServer2_1  | 172.20.0.5 - - [08/May/2017:22:13:31 +0000] "GET /favicon.ico HTTP/1.0" 404 209
proxy_1       | 172.20.0.1 - - [08/May/2017:22:13:31 +0000] "GET /favicon.ico HTTP/1.1" 404 209 "http://localhost:8080/" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/55.0.2883.87 Safari/537.36"
proxy_1       | 172.20.0.1 - - [08/May/2017:22:13:47 +0000] "GET / HTTP/1.1" 200 9 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/55.0.2883.87 Safari/537.36"
webServer3_1  | 172.20.0.5 - - [08/May/2017:22:13:47 +0000] "GET / HTTP/1.0" 200 9
webServer1_1  | 172.20.0.5 - - [08/May/2017:22:13:59 +0000] "GET / HTTP/1.0" 200 9
proxy_1       | 172.20.0.1 - - [08/May/2017:22:13:59 +0000] "GET / HTTP/1.1" 200 9 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/55.0.2883.87 Safari/537.36"
proxy_1       | 172.20.0.1 - - [08/May/2017:22:14:01 +0000] "GET / HTTP/1.1" 200 9 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/55.0.2883.87 Safari/537.36"
webServer2_1  | 172.20.0.5 - - [08/May/2017:22:14:01 +0000] "GET / HTTP/1.0" 200 9
```

Comprobación De Asignación De Volúmenes:

```
distribuidos@redes1:~/Documentos/Distribuidos/DockerLoadBalancer$ docker ps
CONTAINER ID        IMAGE                      COMMAND                  CREATED             STATUS              PORT
S                   NAMES
d87667814fcc        dockerloadbalancer_proxy   "/bin/sh -c 'service "   29 minutes ago      Up 19 seconds       0.0.
0.0:8080->80/tcp    dockerloadbalancer_proxy_1
df489a52680f        apache_confd               "/start.sh"              2 days ago          Up 20 seconds       80/t
cp, 5000/tcp        dockerloadbalancer_webServer3_1
329acb6673ff        apache_confd               "/start.sh"              2 days ago          Up 20 seconds       80/t
cp, 5000/tcp        dockerloadbalancer_webServer2_1
225be145601c        apache_confd               "/start.sh"              2 days ago          Up 20 seconds       80/t
cp, 5000/tcp        dockerloadbalancer_webServer1_1
distribuidos@redes1:~/Documentos/Distribuidos/DockerLoadBalancer$ docker exec -it dockerloadbalancer_webServer2_1 ba
sh
root@329acb6673ff:/usr/local/apache2# cd /apache_data/
root@329acb6673ff:/apache_data# echo "hola" >> mensaje.txt
root@329acb6673ff:/apache_data# ls
hola.txt  mensaje.txt
```

```
distribuidos@redes1:~/Documentos/Distribuidos/DockerLoadBalancer$ docker volume ls
DRIVER              VOLUME NAME
local               dockerloadbalancer_apache_data
local               dockerloadbalancer_nginx_data
distribuidos@redes1:~/Documentos/Distribuidos/DockerLoadBalancer$ docker volume inspect dockerloadbalancer_apache_da
ta
[
    {
        "Name": "dockerloadbalancer_apache_data",
        "Driver": "local",
        "Mountpoint": "/var/lib/docker/volumes/dockerloadbalancer_apache_data/_data",
        "Labels": null,
        "Scope": "local"
    }
]
distribuidos@redes1:~/Documentos/Distribuidos/DockerLoadBalancer$ sudo su
[sudo] password for distribuidos:
root@redes1:/home/distribuidos/Documentos/Distribuidos/DockerLoadBalancer# cd /var/lib/docker/volumes/dockerloadbala
ncer_apache_data/_data
root@redes1:/var/lib/docker/volumes/dockerloadbalancer_apache_data/_data# ls
hola.txt  mensaje.txt
```

Inconvenientes

A lo largo del desarrollo de esta solución se encontraron diferentes problemas:

- Docker Run Time Vs Docker Build Time: Este fue el primer problema encontrado debido a que docker no tiene una herramienta o servicio que permita parametrizar fácilmente los archivos dentro de los contenedores que se desplieguen. Para esto se encuentra la decisión de donde realizar la parametrización. Inicialmente se pensó en realizarlo en el Build Time pues es una infraestructura pequeña, pero se comprobó que esta es la etapa más demorada, pues aquí se procede a descargar una cantidad innumerable de sistemas que tendrá el contenedor. Por lo tanto, quedó descargada esta opción pues al momento donde se necesite escalar el sistema haría una pérdida significativa de recursos. Es por lo anterior que se eligió que cuando se va a desplegar el contenedor es el momento más conveniente para parametrizarlo.

- Tiller vs Confd: Inicialmente se investigó la herramienta Tiller para realizar el acondicionamiento de los archivos para que fueran parametrizables. Pero esta herramienta tiene muy poca documentación y se procedió a realizar tutoriales expuestos en la web los cuales no funcionaron, aun cuando se copió el tutorial. Por lo tanto, para ahorrar tiempo se procedió a una herramienta que ya se había probado su funcionamiento llamada Confd, la cual tiene una documentación decente con tutoriales en la web.

- Volúmenes: El principal inconveniente con los volúmenes fue la adecuación de esto dentro del docker-compose, debido a que solo se tenía el conocimiento de crearlos individualmente y asignarlos a los contenedores, pero para una automatización de infraestructura esto no es mantenible ni escalable. Por lo cual se tuvo que proceder a investigar cómo se creaban volúmenes y se asignaban dentro del mismo docker-compose. Finalmente se pudo lograr.

- Binding Puertos Contenedor Con El Host: Este fue uno de los problemas más grandes dentro de la solución debido a que no se tenía el entendimiento suficiente de la opción -p dentro del comando del docker run. Finalmente, se entendió que la opción -p permite realizar el binding de puertos del contenedor con el host.