Manuel Santos Cerezo

Práctica 3.3 – Profundización en syslog-ng

En esta práctica lo que vamos a hacer es construir un sistema cliente-servidor para la gestión de datos de log.

Nivel C

Ejercicio 1: Monta 2 contenedores de docker. Uno denominado cliente, otro denominado servidor. Instala en ambos syslog-ng y configúralo de forma correcta, de manera que si escribes “logger texto”, puedas ver dicho texto en varlog/syslog

Creamos ambos contenedores

```
┌──(kali㉿kali)-[~]
└─$ sudo docker run --name cliente -it --mount source=mi-vol,destination=/a
pp ubuntu
```

```
┌──(kali㉿kali)-[~]
└─$ sudo docker run --name servidor -it --mount source=mi-vol,destination=/
app ubuntu
```

Y actualizamos apt, `root@47ec32c14560:/# apt update` tras lo cual instalamos nano, iputils-ping, iproute2 `apt install -y nano nodejs iputils-ping iproute2`

y luego syslog-ng `root@47ec32c14560:/# apt install -y syslog-ng`
este proceso lo tenemos que hacer en ambos contenedores, en el ejercicio 6 explicamos cómo crear una imagen con todo esto ya instalado.

Estos pasos hay que hacerlos en ambos contenedores.

Ejercicio 2: Configura el cliente de forma que recoja los logs de un archivo y los vuelque en otro archivo.

Primero, debemos descomentar la última línea de /etc/default/syslog-ng
Vamos a /etc/syslog-ng/conf.d/ y creamos un documento con el siguiente contenido

```
  GNU nano 4.8                                          /etc/syslog-ng/conf.d/myapp.conf
source myapp_log {
        file("var/log/myapp.log");
};

destination myapp_mirror_log {
        file("/var/log/myapp_mirror.log");
};

log {
        source(myapp_log)
        destination(myapp_mirror_log);
};
```

por último reiniciamos

```
root@47ec32c14560:/# service syslog-ng restart
 * Stopping system logging syslog-ng
 * Starting system logging syslog-ng
syslog-ng: Error setting capabilities, capability management disabled; error='Operation not permitted'
```

si el archivo .conf tiene fallos de sintaxis, te avisará por consola y no reiniciará (como a mí), tras corregirlos y reiniciar correctamente lo comprobamos.

```
root@8e74bb6c1ff7:/# echo "prueba" | tee -a /var/log/myapp.log
prueba
root@8e74bb6c1ff7:/# cat /var/log/myapp_mirror.log
Mar 11 16:04:06 8e74bb6c1ff7 prueba
```

Nivel B

Ejercicio 3: Configura el contenedor servidor (o receptor de logs).

Partiendo de un contenedor con los paquetes instalados ya, nos dirigimos a /etc/syslog-ng/conf.d/ y creamos un archivo con la siguiente dirección
le indicamos que recoja los paquetes que entran del exterior (independientemente de la ip y por el puerto 601 y los guarde en el archivo.

```
source myapp_network{
        tcp(ip(0.0.0.0) port (601));
};
destination myapp_local {
        file("/var/log/myapp.log");
};
log {
        source(myapp_network);
        detination(myapp_local);
};
```

y reiniciamos el servicio.

Ejercicio 4: Configura el cliente para mandar los logs al servidor.

Volvemos al archivo de configuración del cliente y le indicamos que envíe los logs por la red, a la ip del servidor.

```
source myapp_log {
        file("var/log/myapp.log");
};

#destination myapp_mirror_log {
        #file("/var/log/myapp_mirror.log");
#};

destination myapp_remote {
        network("172.17.03" port (601) transport("tcp"));
};

log {
        source(myapp_log);
        destination(myapp_mirror_log);
};
```

tras esto reiniciamos el servicio.
Y probamos

```
root@8e74bb6c1ff7:/# echo "prueba" | tee -a /var/log/myapp.log
prueba
```

en el servidor tenemos esto

```
root@59734c6e6e11:/# cat var/log/myapp.log
Mar 11 16:18:21 172.17.0.2 prueba
```

Nivel A

Ejercicio 5: Configura un servidor con node para que genere logs y los envíe al servidor.

Creamos nuestro archivo js

```
const http = require('http')
const port = 80

const server = http.createServer((req, res) => {
        console.log('Ha llegado una petici      n')
        res.statusCode = 200
        res.setHeader('Content-Type', 'text/plain')
        res.end('Hola Mundo!')
})

server.listen(port, () => {
        console.log('Servidor Funcionando')
})
```

configuramos el syslog-ng del cliente y lo reiniciamos después

```
source myapp_log {
        file("/root/node-project/node-server.log");
};

#destination myapp_mirror_log {
#       file("/var/log/myapp_mirror.log");
#};

destination myapp_remote {
        network("172.17.0.3" port (601) transport("tcp"));
};

log {
        source(myapp_log);
        destination(myapp_remote);
};
```

lo iniciamos y comprobamos en el servidor

```
root@8e74bb6c1ff7:~# node node-project/index.js > node-project/node-server.log
```

```
root@59734c6e6e11:/# cat var/log/myapp.log
Mar 11 16:18:21 172.17.0.2 prueba
Mar 11 16:56:53 172.17.0.2 Servidor Funcionando
Mar 11 16:57:18 172.17.0.2 Servidor Funcionando
```

Nivel A+

Ejercicio 6: Genera un dockerfile para automatizar los pasos para generar el cliente. La automatización de pasos implica la instalación de la herramienta syslog-ng, la configuración del archivo de syslog-ng, etc... Para evitar que tengas que copiar el proyecto node, genera un volumen con él, y móntalo en el dockerfile. La IP del servidor al que apuntas tendrá que ser la misma siempre (tiene sentido en un entorno real).

Para probar que has hecho bien el dockerfile, si haces un “build” del archivo, y un “run” de la imagen generada, te debería desplegar la web, y con cada acceso, mandar un log al contenedor servidor.

Creamos el dockerfile del cliente

```
FROM ubuntu
ENV TZ=Europe/Madrid
```

```
# Creamos una variable de entorno y asignamos el huso horario
RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/timezone
# Establecemos el huso horario
RUN apt update && apt install -y syslog-ng nano nodejs iproute2 iputils-ping
# Instalamos los paquetes
RUN echo "SYSLOGNG_OPTS="--no-caps"" >> /etc/default/syslog-ng
# Añadimos en el archivo de configuración de syslog la última linea descomentada
RUN mkdir ~/node-project && touch ~/node-project/node-server.log
# Creamos el fichero en donde almacenaremos el servidor de node y su respectivo log
ADD ./syslog-ng/cli.conf /etc/syslog-ng/conf.d/myapp.conf
# Copiamos del host, el archivo de configuración de syslog-ng
ADD ./nodejs/index.js /root/node-project/index.js
# Copiamos del host, el servidor de node.js
CMD node /root/node-project/index.js
# Por último establecemos que cuando se inicie el contenedor, se arranque el servidor de
node
```

Y ya que estamos el del servidor

```
FROM ubuntu
ENV TZ=Europe/Madrid
# Creamos una variable de entorno y asignamos el huso horario
RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/timezone
# Establecemos el huso horario
RUN apt update && apt install -y syslog-ng nano nodejs iproute2 iputils-ping
# Instalamos los paquetes
RUN echo "SYSLOGNG_OPTS="--no-caps"" >> /etc/default/syslog-ng
# Añadimos en el archivo de configuración de syslog la última linea descomentada
ADD ./syslog-ng/ser.conf /etc/syslog-ng/conf.d/myapp.conf
# Copiamos del host, el archivo de configuración de syslog-ng
```

Los archivos que añadimos a ambos Dockerfiles, son los mismos que hicimos en los ejercicios 4 y 5.
Por último corremos el servidor y luego el cliente. Ahora desde el servidor comprobamos el log.

```
root@6175f903118e:/# cat /var/log/myapp.log
Mar 11 19:47:46 172.17.0.3 test
Mar 11 19:49:45 172.17.0.3 dor Funcionando
```