Tecnológico de Costa Rica
Escuela de Ingeniería en Computación
IC: 7602-Redes - 2 Semestre 2022
2018093728 - Paula Mariana Bustos Vargas
2018086509 - Jocxan Sandi Batista

Proyecto 1

Para este proyecto se realiza la implementación dos pequeñas rede0s virtuales que exponen diferentes servicios, donde la toda la configuración esta implementada mediante Docker y Docker Compose (todo debe estar completamente automatizado)

Para la elaboración del proyecto se conto con un Repositorio de github el cual es:

- Poner el enlace al repositorio

Para la elaboración del proyecto se decidió utilizar el sistema operativo Ubuntu.

Diagrama de arquitectura

Diagrama de flujo

Partes del Proyecto

Redes

En primera instancia se solicita la configuración de las redes esperadas las cuales son las siguiente:

Nombre	Número de red	Máscara	Comando
LAN Virtual 1	10.0.0.0	255.255.255.0	`docker network create LANVirtual1 --subnet=10.0.0.0/24`
LAN Virtual 2	10.0.1.0	255.255.255.0	`docker network create LANVirtua2 --subnet=10.0.1.0/24`

Para realizar la automatización con docker, se realiza una modificación al archivo docker-compose.yaml en el cual tendrá lo siguiente:

```
version: '2'

networks:
  # config LanV1
  LANVirtual1:
    #driver: bridge
    driver: macvlan
    ipam:
      driver: default
      config:
        - subnet: 10.0.0.0/24
        #no permite que sea 10.0.0.1
        - gateway: 10.0.0.99

  # config LanV2
  LANVirtual2:
    #driver: bridge
    driver: macvlan
    ipam:
      driver: default
      config:
        - subnet: 10.0.1.0/24
        #no permite que sea 10.0.1.1
        - gateway: 10.0.1.99
```

Y para ejecutarlo se debe aunque va a mencionar que le tiene servicios seleccionados

```
docker compose up
```

Routers

Se deben de crear dos routers con las siguientes especificaciones:

- Proporcionar acceso a Internet mediante NAT:
    - Router2 para LAN Virtual 2
    - Router2 para LAN Virtual 2

Por lo cual la imagen a utilizar como base sera kmanna/nat-router:

https://hub.docker.com/r/kmanna/nat-router#:~:text=Docker%20nat%2Drouter,using%20pipework%20to%20create%20eth1.

Router1

Proporcionar acceso a Internet a LAN Virtual 1 mediante NAT.

El tráfico de salida permitido será únicamente mediante UDP y los puertos TCP/80 y TCP/443.

Deberá permitir el acceso externo mediante puertos TCP/80, TCP/443, TCP/3128, TCP/8443, UDP/53, TCP/53

- TCP/80 y TCP/443 se debe enrutar al Proxy Reverso.
- TCP/3128 se debe enrutar al Web Cache.
- TCP/8443 se debe enrutar al VPN.
- TCP/53 y UDP/53 deben ser enrutados al DNS.

Doc del DockerFile

```
# Router 1
#WORKDIR /router1
#COPY script.sh /
#RUN chmod +x /script.sh
#ENTRYPOINT ["/script.sh"]

#FROM kmanna/nat-router
FROM ubuntu
COPY ./script.sh /home/

USER root
#ENTRYPOINT
ENTRYPOINT ["/bin/bash", "/home/script.sh"]
```

Doc script.sh

```
#TCP
#iptables -A INPUT  -p tcp -m multiport --dports 53,80,443,3128,8443 -j ACCEPT

iptables -A INPUT -p tcp --sport 53 -j ACCEPT
iptables -A INPUT -p tcp --sport 80 -j ACCEPT
iptables -A INPUT -p tcp --sport 443 -j ACCEPT
iptables -A INPUT -p tcp --sport 3128 -j ACCEPT
iptables -A INPUT -p tcp --sport 8443 -j ACCEPT

#UDP
#iptables -A INPUT  -p upd --dports 53 -j ACCEPT

iptables-legacy -A INPUT -p udp --dport 53 -j ACCEPT
```

Creamos la imagen del router

```
docker image build -t router:1.0 .
```

https://unrouted.io/2017/08/15/docker-firewall/

https://medium.com/swlh/manage-iptables-firewall-for-docker-kubernetes-daa5870aca4d

Router2

Permitirá acceso del VPN hacia cualquier host en LAN Virtual 1 e Internet.

Permitirá acceso desde cualquier host en LAN Virtual 2 a cualquier host en la LAN Virtual 1 en los puertos TCP/80, TCP/443 y el protocolo ICMP.

Doc del DockerFile

```
# Router 2
#WORKDIR /router2
#COPY script.sh /
#RUN chmod +x /script.sh
#ENTRYPOINT ["/script.sh"]

#FROM kmanna/nat-router
FROM ubuntu
COPY ./script.sh /home/

USER root
#ENTRYPOINT
ENTRYPOINT ["/bin/bash", "/home/script.sh"]
```

Doc script.sh

```
#TCP
#iptables -A INPUT  -p tcp -m multiport --dports 80,443 -j ACCEPT

iptables- -A INPUT -p tcp --sport 80 -j ACCEPT
iptables -A INPUT -p tcp --sport 443 -j ACCEPT

#UDP
#iptables -A INPUT  -p upd --dports 53 -j ACCEPT

iptables -A INPUT -p upd --dport 53 -j ACCEPT
```

Archivo docker-compose.yaml

ports:

- "53/udp"
- "80:443/tcp"

Para la creación de los routes que se comportan com

DNS

Se utiliza la imagen Ubuntu/Bind9 para realizar el DNS.

Dockerfile

```
FROM ubuntu/bind9

WORKDIR /Proyecto/DNS/bind

COPY . /Proyecto/DNS/bind/

RUN docker run -d --name bind9-container -e TZ=UTC -p 30053:53 ubuntu/bind9:9.18-
22.04_beta
```

Configuraciones mínimas del DNS

En el archivo: ***named.conf.options***.

```
listen-on { any; };
allow-query { localhost; 10.0.0.0/24; 10.0.1.0/24; };
forwarders {
        8.8.8.8;
        8.8.8.9;
};
dnssec-validation no;
```

En el archivo: ***named.conf.local***, se configuran las zonas.

```
zone "lan01.io" IN {
        type master;
        file "/etc/bind/forward.lan01.io
";
};
```

```
zone "lan02.io" IN {
        type master;
        file "/etc/bind/forward.lan02.io
";
};

zone "google.com" IN {
        type master;
        file "/etc/bind/forward.google.com
";
};
```

Para cada una de las zonas declaradas en el archivo anterior, se debe crear un archivo para detallar su configuración.

Por ejemplo en el archivo: ***forward.lan01.io***

```
$TTL    1D
@       IN      SOA     lan01.io. root.lan01.io. (
        1               ; Serial
        12h             ; Refresh
        15m             ; Retry
        3w              ; Expire
        2h  )           ; Negative Cache TTL

;       Registros NS

        IN      NS      lan.01.io.
dhcp1   IN      A       10.0.0.4
```

DHCP

Para crear un DHCP utilizamos la imagen networkboot/dhcpd. Utiliza un archivo llamado ***dhcp.conf***, el cual se presenta adelante.

```
subnet 10.0.0.0 netmask 255.255.255.0{
    option routers              10.0.0.1;
    option subnet-mask          255.255.255.0;
    option domain-name-servers  10.0.0.1;
    default-lease-time          43200;
    max-lease-time              86400;
    range                       10.0.0.100  10.0.0.150;
}
```

Clientes

Web Server 1 y Web Server 2

Deberán implementar un Web Server en Apache, el mismo expone una simple página

Construcción de las imágenes de los servidores Se construye la imagen del server 1

```
docker image build -t helloworld:1.0 .
```

Pero para poder realizarlo se tiene que tener el dockerfile y el formato del la pagina a mostrar.

Dockerfile:

```
# use the httpd base image
FROM httpd:2.4

# Reemplazar index
COPY ./index.html/ /usr/local/apache2/htdocs/
```

Se decide subir la imagen para no perderla y poder utilizarla en un futuro

```
docker tag helloworld:1.0 mari1018/helloworld:1.0
```

```
docker push mari1018/helloworld:1.0
```

Se vuelve a realizar lo mismo para el server 2

- docker image build -t helloworld2:1.0 .
- docker tag helloworld2:1.0 mari1018/helloworld2:1.0
- docker push mari1018/helloworld2:1.0

Proxy Reverso

```
location /web1 {
    proxy_pass Server1
}
```

Pruebas realizadas, con pasos para reproducirlas.

Para poder visualizar las dos paginas web se creo un cliente en el cual esta expuesto en el puerto 80 por lo que se tiene son dos ambientes el web1 y web2 que son "hola mundos l1" y "hola mundo l2" que a la hora de la solicitud se va turnando en cual muestra. Para poder replicarlo se debe de correr el docker compose y abrir su navegador y buscar el localhost o bien el ip de su maquina con el puerto 80

Resultados de las pruebas unitarias.

Recomendaciones

- Realizar una planificación de las tareas por realizar.
- Aprovechar todo el tiempo dado.
- Comprender el objetivo del proyecto
- Ir a consulta con el profesor.
- Si se tiene una duda mejor preguntar.
- Investigar sobre docker y docker compose.
- Hacer un cronograma del proyecto.
- Investigar sobre las herramientas recomendadas por el profesor.
- Iniciar con lo que se considere más sencillo de realizar.
- Apoyo con compañeros y foros es de ayuda.

Conclusiones

- Manejar un código programado por terceras personas siempre representa un gran reto, o bien utilizar este como base se puede complicar mucho por lo cual es mejor ir dividiéndolo por secciones. Como a la hora de crear las imágenes.

- Se invita a crear primeramente los servidores web para poder ir uniéndolo con las demás partes del proyecto, ya que se considera que fue una de las partes más simples de realizar.

- Si bien en el desarrollo de la imagen del router con diversas configuraciones se realizaba con éxito hasta el punto de la creación del router, se nota que por un pequeño detalle que no se entendió ni se logró resolver el funcionamiento de estos, se vió perjudicado por esto.

- También se nota la versión con la cual genera el docker compose.yml pueden solventar algunos errores de identación no encontrados.

- La conexión entre los diferentes componentes a generar, era más complicado de lo previsto.

- La formas de configurar los diferentes componentes, era más complicado de lo previsto.

- No todas las imágenes encontradas en docker hub fueron de ayuda, ya que no están lo suficientemente documentadas y se perdió mucho tiempo intentando realizarlo con éxito.

- Si bien realizar los componentes por consola fue más sencillo, la automatización en docker-compose no fue así debido al poco dominio que se tenía de docker.

- El proyecto permitió conocer sobre docker, creación de imágenes y como los componentes de red se relacionan.
- A pesar de que no se logró concluir el proyecto, este permitió ampliar los conocimientos teóricos y algunos prácticos del funcionamientos de las redes.

Bibliografia

- Configuración de red (Networking) en Dockers - Blog Virtualizacion https://www.maquinasvirtuales.eu/configuracion-red-networking-dockers/
- Docker and iptables | Docker Documentation https://docs.docker.com/network/iptables/
- https://forums.docker.com/t/routing-network-traffic-from-one-service-to-another/117816/5
- Compose file version 3 reference. Docker Documentation. Recuperado de https://docs.docker.com/compose/compose-file/compose-file-v3/#host-or-none
- https://www.returngis.net/2019/02/publicar-tu-imagen-en-docker-hub/
- https://hub.docker.com/r/networkboot/dhcpd/#:~:text=For%20that%20you%20need%20to,for%20the%20specified%20network%20interface.
- https://gist.github.com/mikejoh/04978da4d52447ead7bdd045e878587d
- https://www.youtube.com/watch?v=tIlNfB2Vk8s&ab_channel=SnatchDreams
- https://www.netntw.com/archivos/533
- https://www.youtube.com/watch?v=b_mOOs53ut0&ab_channel=NETWORLD
- https://www.youtube.com/watch?v=tIlNfB2Vk8s&t=358s&ab_channel=SnatchDreams
- https://iesgn.github.io/curso_docker_2021/sesion4/definida.html
- https://hub.docker.com/r/kmanna/nat-router#:~:text=Docker%20nat%2Drouter,using%20pipework%20to%20create%20eth1
- https://github.com/jpetazzo/pipework
- https://linux.die.net/man/8/iptables
- https://unix.stackexchange.com/questions/150523/bash-iptables-command-not-found
- https://docs.docker.com/develop/develop-images/dockerfile_best-practices/
- https://docs.docker.com/config/containers/container-networking/
- https://stackoverflow.com/questions/7499248/how-do-i-run-a-shell-script-as-root-sudo
- https://stackoverflow.com/questions/34549859/run-a-script-in-dockerfile