UNIVERSIDAD NACIONAL DE CÓRDOBA
Facultad de Ciencias Exactas, Físicas y Naturales

Redes de Computadoras

Práctico 7: Aplicación.

Alumnos:

- **Heredia, Marco.**
- **Yepez Hinostroza, Franz.**
- **Contrera, Iván**
- **Malano, Leandro**

Ejercicio 1: Ruteo internet.

Configuración de red

Podemos ver la asignación de IPs en la siguiente tabla:

Nodo	Interfaz compartida con	Dirección IP de la red
b1	b2	2001:a:1::/64
b1	r3	2001:a:aaaa:1::/64
b1	r5	2001:a:aaaa:2::/64
b1	r7	2001:a:aaaa:3::/64
r5	r7	2001:a:aaaa:4::/64
r5	h11	2001:a:aaaa:5::/64
r7	h13	2001:a:aaaa:6::/64
r3	r3	2001:a:aaaa:7::/64
b2	r4	2001:a:bbbb:1::/64

b2	r6	2001:a:bbbb:2::/64
b2	r8	2001:a:bbbb:3::/64
r6	r8	2001:a:bbbb:4::/64
r6	h12	2001:a:bbbb:5::/64
r8	h14	2001:a:bbbb:6::/64
r4	r4	2001:a:bbbb:7::/64

1.- Se hizo la asignación de redes tal cual se mostró en la tabla anterior

2.- Se asignaron nombres de dominio para los dos sistemas autónomos (en adelante, AS):

- AS101: grupo9.fcefyn.com
- AS202: grupo10.redes.fcefyn.unc.edu.local

4.- Se configuran los routers de borde b1 y b2 para que ejecuten el protocolo BGP, tal como se hizo en el trabajo 5:

```
! -*- bgp -*-
!
! BGPd configuration file
!
!
hostname b1
password admin
!
route-map SET-LP permit 10
  set local-preference 200
!
router bgp 101
  bgp router-id 192.168.1.10
!
  no auto-summary
  no synchronization
!
  neighbor 2001:a:1::3 remote-as 202
  neighbor 2001:a:1::3 description B
  neighbor 2001:a:1::3 ebgp-multihop
!
  address-family ipv6
  network 2001:a:aaaa::/48
  neighbor 2001:a:1::3 activate
  redistribute ospf6
```

```
! -*- bgp -*-
!
! BGPd configuration file
!
!
hostname b2
password admin
!
router bgp 202
  bgp router-id 192.168.2.10
!
  no auto-summary
  no synchronization
!
  neighbor 2001:a:1::2 remote-as 101
  neighbor 2001:a:1::2 description B
  neighbor 2001:a:1::2 ebgp-multihop
!
  address-family ipv6
  network 2001:a:bbbb::/48
  neighbor 2001:a:1::2 activate
  redistribute ospf6
```

5.- Para interconectar los hosts fisicos, se siguieron los siguientes pasos:

- Identificar en cada host el bridge (cuyo nombre general es br-xxxxxxxxxxxx), tal que su dirección ip sea 2001:a:1::1. Para esto se usa el comando *ip address*:

```
177: br-a8fd3596a8dc: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:a2:85:b8:86 brd ff:ff:ff:ff:ff:ff
    inet 172.22.0.1/16 brd 172.22.255.255 scope global br-a8fd3596a8dc
       valid_lft forever preferred_lft forever
    inet6 2001:a:1::1/64 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::42:a2ff:fe85:b886/64 scope link
       valid_lft forever preferred_lft forever
    inet6 fe80::1/64 scope link
       valid_lft forever preferred_lft forever
```

En este caso, es el br-a8fd3596a8dc. Además, identificar el nombre de la interfaz fisica (enp2s0).

- Asegurarse que ninguna de las dos interfaces físicas tengan asignada alguna dirección ipv6. En lo posible, desactivarla desde el network manager.
- Se debe vincular (como maestro) el bridge con la interfaz física. Para esto se utiliza el comando *ip link set enp2s0 master br-a8fd3596a8dc.* De esta manera, todo dato ingresado por dicha interfaz, pasará directamente a ser procesado por el bridge, y llegará al contenedor b1/2.
- La conexion quedara lista y la transmisión de los paquetes BGP comenzará.

6.- Se configura OSPFv3 tal cual se configuró en los otros trabajos practicos.

DNS:

7.- Se asigna un nombre de dominio a cada dirección ip (registros AAAA) en los dos AS, y su resolucion inversa (registros PTR), siempre a traves de webmin:

AS202:

AAAA

Name	TTL	IPv6 Address
red1.r4.grupo9.fcefyn.com.	Default	2001:a:bbbb:1::3
red1.b2.grupo9.fcefyn.com.	Default	2001:a:bbbb:1::2
red2.b2.grupo9.fcefyn.com.	Default	2001:a:bbbb:2::2
red2.r6.grupo9.fcefyn.com.	Default	2001:a:bbbb:2::3
red3.b2.grupo9.fcefyn.com.	Default	2001:a:bbbb:3::2
red3.r8.grupo9.fcefyn.com.	Default	2001:a:bbbb:3::3
red4.r8.grupo9.fcefyn.com.	Default	2001:a:bbbb:4::3
red4.r6.grupo9.fcefyn.com.	Default	2001:a:bbbb:4::2
red5.r6.grupo9.fcefyn.com.	Default	2001:a:bbbb:5::2
red5.h12.grupo9.fcefyn.com.	Default	2001:a:bbbb:5::3
red6.r8.grupo9.fcefyn.com.	Default	2001:a:bbbb:6::2
red6.h14.grupo9.fcefyn.com.	Default	2001:a:bbbb:6::3
red6.h14.grupo9.fcefyn.com.	Default	2001:a:bbbb:6::3
redtroncal.b2.grupo9.fcefyn.com.	Default	2001:a:1::3
mongo.nginx.grupo9.fcefyn.com.	Default	2001:a:bbbb:a::3
strapi.nginx.grupo9.fcefyn.com.	Default	2001:a:bbbb:9::3
red8.r4.grupo9.fcefyn.com.	Default	2001:a:bbbb:8::2
red8.nginx.grupo9.fcefyn.com.	Default	2001:a:bbbb:8::4
redb.r4.grupo9.fcefyn.com.	Default	2001:a:bbbb:b::2
redb.squid.grupo9.fcefyn.com.	Default	2001:a:bbbb:b::4

PTR

Address	TTL	Hostname
2001:a:bbbb:1::3	Default	red1.r4.grupo9.fcefyn.com.
2001:a:bbbb:1::2	Default	red1.b2.grupo9.fcefyn.com.
2001:a:bbbb:2::2	Default	red2.b2.grupo9.fcefyn.com.
2001:a:bbbb:2::3	Default	red2.r6.grupo9.fcefyn.com.
2001:a:bbbb:3::2	Default	red3.b2.grupo9.fcefyn.com.
2001:a:bbbb:4::3	Default	red4.r8.grupo9.fcefyn.com.
2001:a:bbbb:4::2	Default	red4.r6.grupo9.fcefyn.com.
2001:a:bbbb:5::2	Default	red5.r6.grupo9.fcefyn.com.
2001:a:bbbb:5::3	Default	red5.h12.grupo9.fcefyn.com.
2001:a:bbbb:6::2	Default	red6.r8.grupo9.fcefyn.com.
2001:a:bbbb:6::3	Default	red6.h14.grupo9.fcefyn.com.
2001:a:1::3	Default	redtroncal.b2.grupo9.fcefyn.com.
2001:a:bbbb:a::3	Default	mongo.nginx.grupo9.fcefyn.com.
2001:a:bbbb:9::3	Default	strapi.nginx.grupo9.fcefyn.com.
2001:a:bbbb:8::2	Default	red8.r4.grupo9.fcefyn.com.
2001:a:bbbb:8::4	Default	red8.nginx.grupo9.fcefyn.com.
2001:a:bbbb:b::2	Default	redb.r4.grupo9.fcefyn.com.
2001:a:bbbb:b::4	Default	redb.squid.grupo9.fcefyn.com.

AS101:

Name	TTL	IPv6 Address
r5.grupo10.redes.fcefyn.unc.edu.local.	Default	2001:a:aaaa:1::3
r7.grupo10.redes.fcefyn.unc.edu.local.	Default	2001:a:aaaa:6::2
b1.grupo10.redes.fcefyn.unc.edu.local.	Default	2001:a:1::2
b1.grupo10.redes.fcefyn.unc.edu.local.	Default	2001:a:aaaa:1::2
r5.grupo10.redes.fcefyn.unc.edu.local.	Default	2001:a:aaaa:1::3
b1.grupo10.redes.fcefyn.unc.edu.local.	Default	2001:a:aaaa:2::2
r5.grupo10.redes.fcefyn.unc.edu.local.	Default	2001:a:aaaa:2::3
b1.grupo10.redes.fcefyn.unc.edu.local.	Default	2001:a:aaaa:3::2
r7.grupo10.redes.fcefyn.unc.edu.local.	Default	2001:a:aaaa:3::3
r5.grupo10.redes.fcefyn.unc.edu.local.	Default	2001:a:aaaa:4::2
r7.grupo10.redes.fcefyn.unc.edu.local.	Default	2001:a:aaaa:4::3
r5.grupo10.redes.fcefyn.unc.edu.local.	Default	2001:a:aaaa:5::2
h11.grupo10.redes.fcefyn.unc.edu.local.	Default	2001:a:aaaa:5::3
r7.grupo10.redes.fcefyn.unc.edu.local.	Default	2001:a:aaaa:6::2
h13.grupo10.redes.fcefyn.unc.edu.local.	Default	2001:a:aaaa:6::3
r5.grupo10.redes.fcefyn.unc.edu.local.	Default	2001:a:aaaa:7::2
nginx.grupo10.redes.fcefyn.unc.edu.local.	Default	2001:a:aaaa:8::4
nginx.grupo10.redes.fcefyn.unc.edu.local.	Default	2001:a:aaaa:9::4
nginx.grupo10.redes.fcefyn.unc.edu.local.	Default	2001:a:aaaa:a::4

Address	TTL	Hostname
2001:a:1::2	Default	b1.grupo10.redes.fcefyn.unc.edu.local.
2001:a:aaaa:1::2	Default	b1.grupo10.redes.fcefyn.unc.edu.local.
2001:a:aaaa:1::3	Default	r5.grupo10.redes.fcefyn.unc.edu.local.
2001:a:aaaa:2::2	Default	b1.grupo10.redes.fcefyn.unc.edu.local.
2001:a:aaaa:2::3	Default	r5.grupo10.redes.fcefyn.unc.edu.local.
2001:a:aaaa:3::2	Default	b1.grupo10.redes.fcefyn.unc.edu.local.
2001:a:aaaa:3::3	Default	r7.grupo10.redes.fcefyn.unc.edu.local.
2001:a:aaaa:4::2	Default	r5.grupo10.redes.fcefyn.unc.edu.local.
2001:a:aaaa:4::3	Default	r7.grupo10.redes.fcefyn.unc.edu.local.
2001:a:aaaa:5::2	Default	r5.grupo10.redes.fcefyn.unc.edu.local.
2001:a:aaaa:5::3	Default	h11.grupo10.redes.fcefyn.unc.edu.local.
2001:a:aaaa:6::2	Default	r7.grupo10.redes.fcefyn.unc.edu.local.
2001:a:aaaa:6::3	Default	h13.grupo10.redes.fcefyn.unc.edu.local.
2001:a:aaaa:7::2	Default	r5.grupo10.redes.fcefyn.unc.edu.local.
2001:a:aaaa:8::4	Default	nginx.grupo10.redes.fcefyn.unc.edu.local.
2001:a:aaaa:9::4	Default	nginx.grupo10.redes.fcefyn.unc.edu.local.
2001:a:aaaa:a::4	Default	nginx.grupo10.redes.fcefyn.unc.edu.local.

8.- Para que un contenedor pueda resolver cualquier nombre de dominio, se agrega un campo **dns** en cada contenedor, con la ip del servidor:

```
r6:
  build: .
  volumes:
    - ./volumes/ospf/r6/zebra.conf:/etc/quagga/zebra.conf:ro
    - ./volumes/ospf/r6/ospf6d.conf:/etc/quagga/ospf6d.conf:ro
    - ./volumes/ospf/supervisord.conf:/etc/supervisor/conf.d/supervisord.conf:ro
  image: ospf:20180419
  privileged: true
  dns: 2001:a:bbbb:7::3
  ports:
    #admin
    - 10611:2601
    #ospf ipv6
    - 10613:2606
  networks:
    redb2:
      ipv6_address: 2001:a:bbbb:2::3
    redb4:
      ipv6_address: 2001:a:bbbb:4::2
    redb5:
      ipv6_address: 2001:a:bbbb:5::2
```

Luego, ejecutando el comando *dig @2001:a:bbbb:7::3 red1.r4.grupo9.fcefyn.com AAAA* se puede ver la respuesta:

```
;; ANSWER SECTION:
red1.r4.grupo9.fcefyn.com. 38400 IN     AAAA    2001:a:bbbb:1::3
```

Aplicación Web

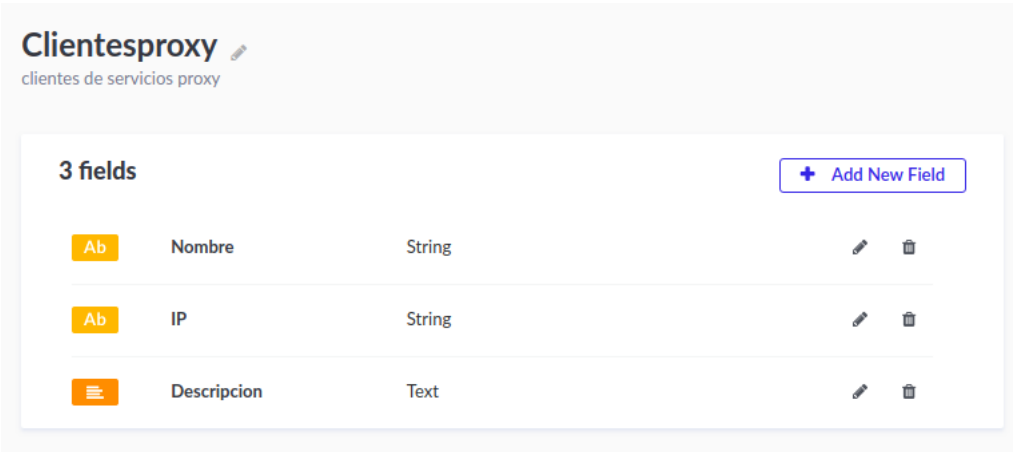
9.- Se implemento el stack NGINX + NodeJS + MongoDB en IPv6 y luego se utilizó Strapi como web framework. Se descargó desde Docker Hub una imagen de strapi (cuyo dockerfile descarga automaticamente la imagen de MongoDB), y otra de NGINX. Luego, en el docker-compose.yml se crean los contenedores necesarios, uniendolos correctamente con redes, y se configuran para que sus archivos de conifguracion se guarden en ./volumes/<carpeta de c/servicio>:

```
services:

  nginx:
      image: nginx:alpine
      privileged: true
      volumes:
        - ./volumes/nginx/nginx.conf:/etc/nginx/nginx.conf
        - ./volumes/ssl:/ssl
      restart: always
      networks:
        redb8:
          ipv6_address: 2001:a:bbbb:8::4
        redb9:
          ipv6_address: 2001:a:bbbb:9::4
        redbA:
          ipv6_address: 2001:a:bbbb:a::4
```

```
  api:
    image: strapi/strapi
    volumes:
      - ./volumes/strapi-app:/usr/src/api/strapi-app
    privileged: true
    dns:
#       - 8.8.8.8
      - 2001:a:bbbb:7::3
    environment:
      - APP_NAME=strapi-app
      - DATABASE_CLIENT=mongo
      - DATABASE_HOST=[2001:a:bbbb:9::4]
      - DATABASE_PORT=27017
      - DATABASE_NAME=strapi
      - DATABASE_USERNAME=
      - DATABASE_PASSWORD=
      - HOST=localhost
    depends_on:
      - db
      - nginx
    networks:
      redb9:
          ipv6_address: 2001:a:bbbb:9::3

  db:
    image: mongo
    privileged: true
    environment:
      - MONGO_INITDB_DATABASE=strapi
    depends_on:
      - nginx
    volumes:
      - ./volumes/db:/data/db
    command: mongod --ipv6 --bind_ip_all
    networks:
      redbA:
          ipv6_address: 2001:a:bbbb:a::3
```

10.- Se configuró MongoDB para que tenga disponibilidad para un único servicio, ya que no se pudo encontrar la manera de configurarlo para que tenga alta disponibilidad para 3 servicios.

11.- A continuación, se debe crear una api distinta para cada sistema autonomo. Para esto, se accede a strapi desde el navegador, ingresando la ip [2001:a:bbbb:9::4] (AS202) o la ip [2001:a:aaaa:9::4] (AS101). En la sección *Content Type Builder*, se crearan las APIS. Por ejemplo, para el AS202 se creo una de clientes de servicios proxy:



Luego se modifican algunos permisos, como para poder acceder a la información.

12.- Con el método POST podemos crear, modificar y eliminar registros desde curl para nuestras APIs. Esto también es posible realizarlo desde la interfaz gráfica del navegador.

13.- Se debe generar una clave pública y un certificado. Para esto se utiliza OpenSSL: Por empezar, se genera la clave:

```
marco@marco:~/Escritorio/tp7a/volumes/ssl$ openssl genrsa -out server.key 1024
Generating RSA private key, 1024 bit long modulus
......++++++
.++++++
e is 65537 (0x10001)
```

Se genera la solicitud de certificado (CSR). Un CSR es la base para un certificado SSL, en él se definen datos como el dominio, organización, ubicación, información de contacto, entre otros.:

```
e is 65537 (0x10001)
marco@marco:~/Escritorio/tp7a/volumes/ssl$ openssl req -new -key server.key -out
 server.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:AR
State or Province Name (full name) [Some-State]:Cordoba
Locality Name (eg, city) []:Cordoba
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Grupo10
Organizational Unit Name (eg, section) []:TP7
Common Name (e.g. server FQDN or YOUR name) []:2001:a:aaaa:9::4
Email Address []:tp7@grupo10.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Por último, se genera el certificado SSL:

```
marco@marco:~/Escritorio/tp7a/volumes/ssl$ openssl x509 -req -days 365 -in serve
r.csr -signkey server.key -out server.crt
Signature ok
subject=/C=AR/ST=Cordoba/L=Cordoba/O=Grupo10/OU=TP7/CN=2001:a:aaaa:9::4/emailAdd
ress=tp7@grupo10.com
Getting Private key
```

Despues se edita el archivo /etc/nginx/nginx.conf en el contenedor de proxy reverso para habilitar la conexión a través del 443 con el estandar SSL. Agregando las siguientes líneas:

```
server {

    listen 443 ssl;
    listen [::]:443 ipv6only=on ssl;

    ssl_certificate         /ssl/server.crt;
    ssl_certificate_key     /ssl/server.key;

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

En la cual se le indica la ubicación del certificado SSL y de su clave.

14.- Para que los hosts acepten este certificado como válido lo que se hace es agregar este certificado en el directorio /usr/share/ca-certificates/extra/server.crt de los hosts. Y luego se corre en cada uno de los hosts el comando: “dpkg-reconfigure ca-certificates” para habilitarlo.

Luego se prueba su funcionamiento utilizando el comando *curl https:///[2001:a:bbbb:9::4]/Clientesproxy*, obteniendo un resultado positivo:

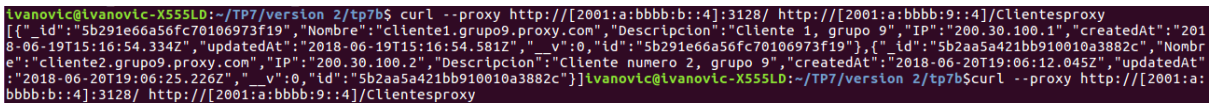
Proxy HTTP

15.- Para implementar el proxy http, se descargó la imagen Squid desde el docker hub, el cual viene con un archivo *squid.conf,* que se debera editar para que permita cualquier solicitud http. Para tal objetivo, se agregan las siguientes lineas:

http_access allow all

dns_nameservers 2001:a:bbbb:7::3 (o 2001:a:aaaa:7::3, dependiendo del sistema autonomo)

Luego, desde el host, se hace una solicitud a la api de strapi con el comando *curl --proxy http:///[2001:a:bbbb:b::4]:3128/* [*http:///[2001:a:bbbb:9::4]/Clientesproxy*](http:///[2001:a:bbbb:9::4]/Clientesproxy)



Luego accediendo al archivo *access.log* del contenedor de Squid, se pueden ver los registros de las solicitudes:

Referencias:

http://www.philipermish.com/blog/docker-example-with-nginx-node-redis-mongodb-and-jekyll/

https://closebrace.com/tutorials/2017-03-02/the-dead-simple-step-by-step-guide-for-front-end-developers-to-getting-up-and-running-with-nodejs-express-and-mongodb

https://medium.com/statuscode/dockerising-a-node-js-and-mongodb-app-d22047e2806f

https://github.com/damsonn/node-docker-compose/blob/master/docker-compose.yml

https://github.com/Osedea/nodock/blob/master/docker-compose.yml

https://docs.nginx.com/nginx/admin-guide/web-server/reverse-proxy/

https://www.linode.com/docs/web-servers/nginx/enable-tls-on-nginx-for-https-connections/

http://nginx.org/en/docs/http/configuring_https_servers.html

https://www.nanotutoriales.com/como-crear-un-certificado-ssl-de-firma-propia-con-openssl-y-apache-http-server

https://www.digitalocean.com/community/tutorials/how-to-create-an-ssl-certificate-on-nginx-for-ubuntu-14-04

https://www.digicert.com/es/instalar-certificado-ssl-nginx.htm