

**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:

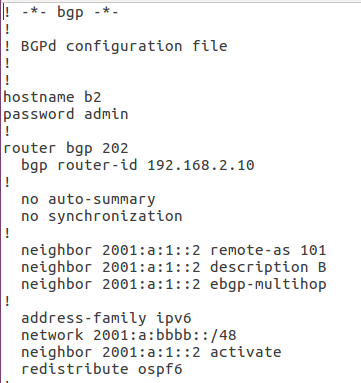
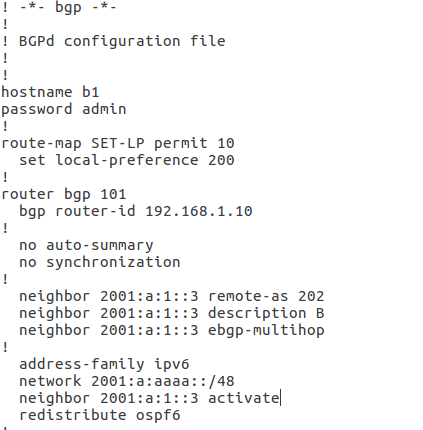
|  |  |  |
| --- | --- | --- |
| Nodo | Interfaz compartida con | Dirección IP de la red |
| b1 | b2 | 2001:a:1::/64 |
| b1 | r3 | 2001:a:aaaa:1::/64 |
| b1 | r5 | 2001:a:aaaa:2::/64 |
| b1 | r7 | 2001:a:aaaa:3::/64 |
| r5 | r7 | 2001:a:aaaa:4::/64 |
| r5 | h11 | 2001:a:aaaa:5::/64 |
| r7 | h13 | 2001:a:aaaa:6::/64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

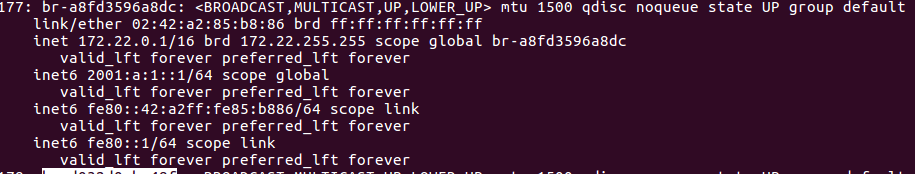
* AS101: grupo9.fcefyn.com
* AS202: grupo10.redes.fcefyn.unc.edu.local

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



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*:



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:**

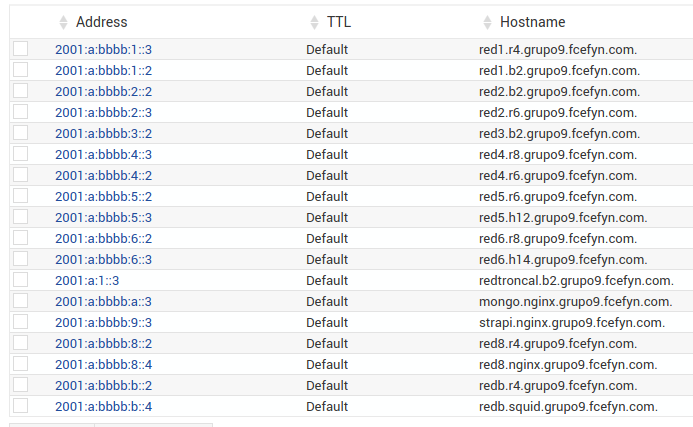
7.- Se asigna un nombre de dominio a cada dirección ip (registros AAAA) en los dos AS, y su resolucion inversa (registros PTR), siempre a traves de webmin:

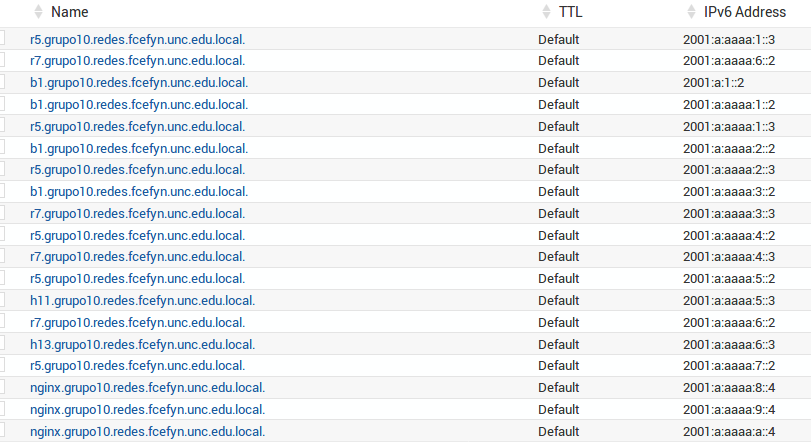
AS202:

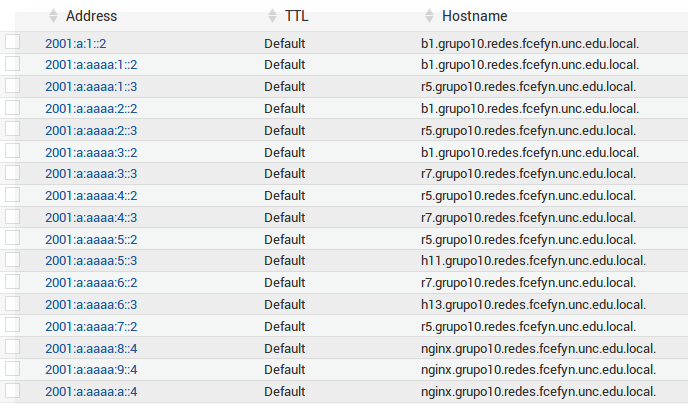
AAAA



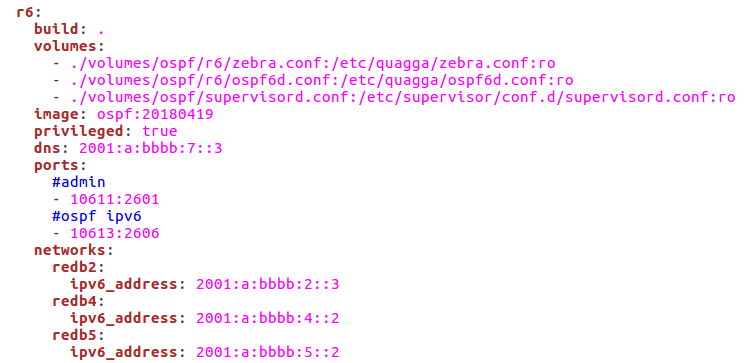
PTR



AS101: 



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



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



### **Aplicación Web**

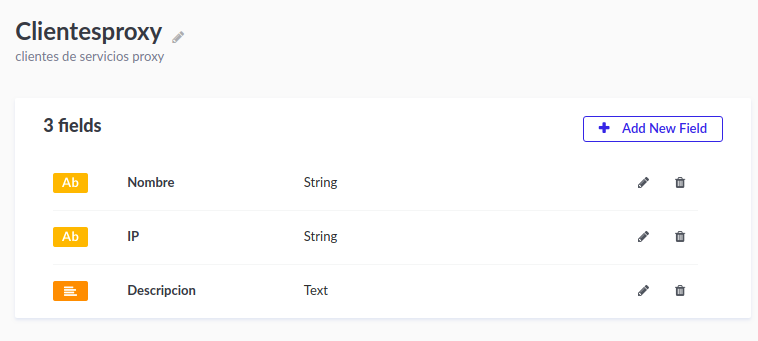
9.- Se implemento el stack NGINX + NodeJS + MongoDB en IPv6 y luego se utilizó Strapi como web framework. Se descargó desde Docker Hub una imagen de strapi (cuyo dockerfile descarga automaticamente la imagen de MongoDB), y otra de NGINX. Luego, en el docker-compose.yml se crean los contenedores necesarios,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





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.

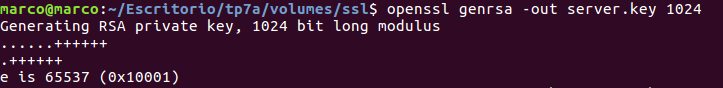
11.- A continuación, se debe crear una api distinta para cada sistema autonomo. Para esto, se accede a strapi desde el navegador, ingresando la ip [2001:a:bbbb:9::4] (AS202) o la ip [2001:a:aaaa:9::4] (AS101). En la sección *Content Type Builder*, se crearan las APIS. Por ejemplo, para el AS202 se creo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



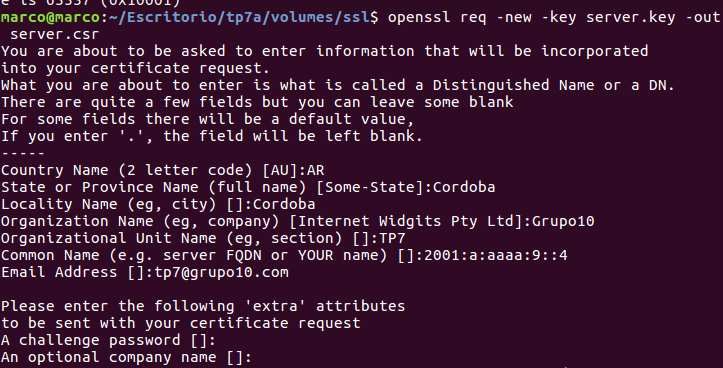
Luego se modifican algunos permisos, como para poder acceder a la información.

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.

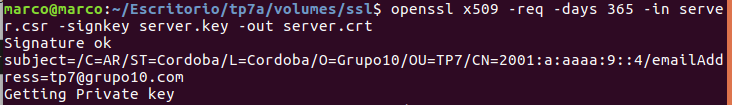
13.- Se debe generar una clave pública y un certificado. Para esto se utiliza OpenSSL: Por empezar, se genera la clave:



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.:



Por último, se genera el certificado SSL:



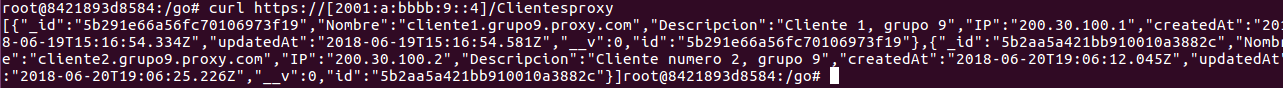
Despues se edita el archivo /etc/nginx/nginx.conf en el contenedor de proxy reverso para habilitar la conexión a través del 443 con el estandar SSL. Agregando las siguientes líneas:



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:



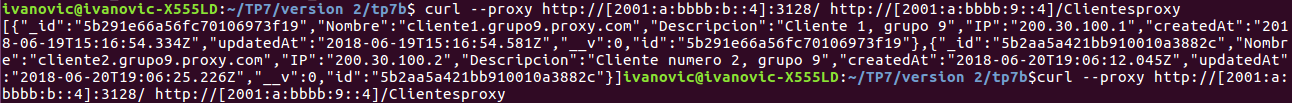
### **Proxy HTTP**

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:

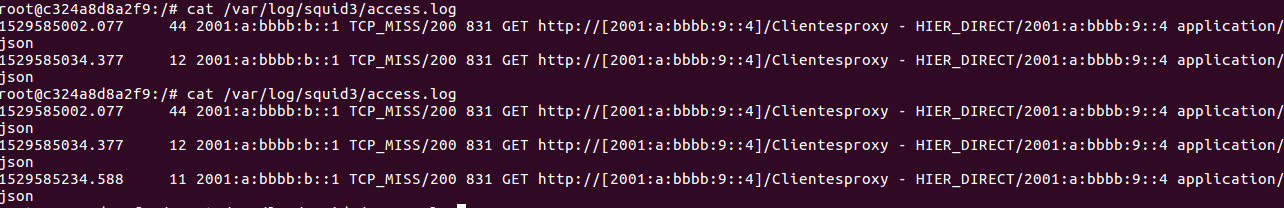
http\_access allow all

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://%5B2001:a:bbbb:9::4%5D/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