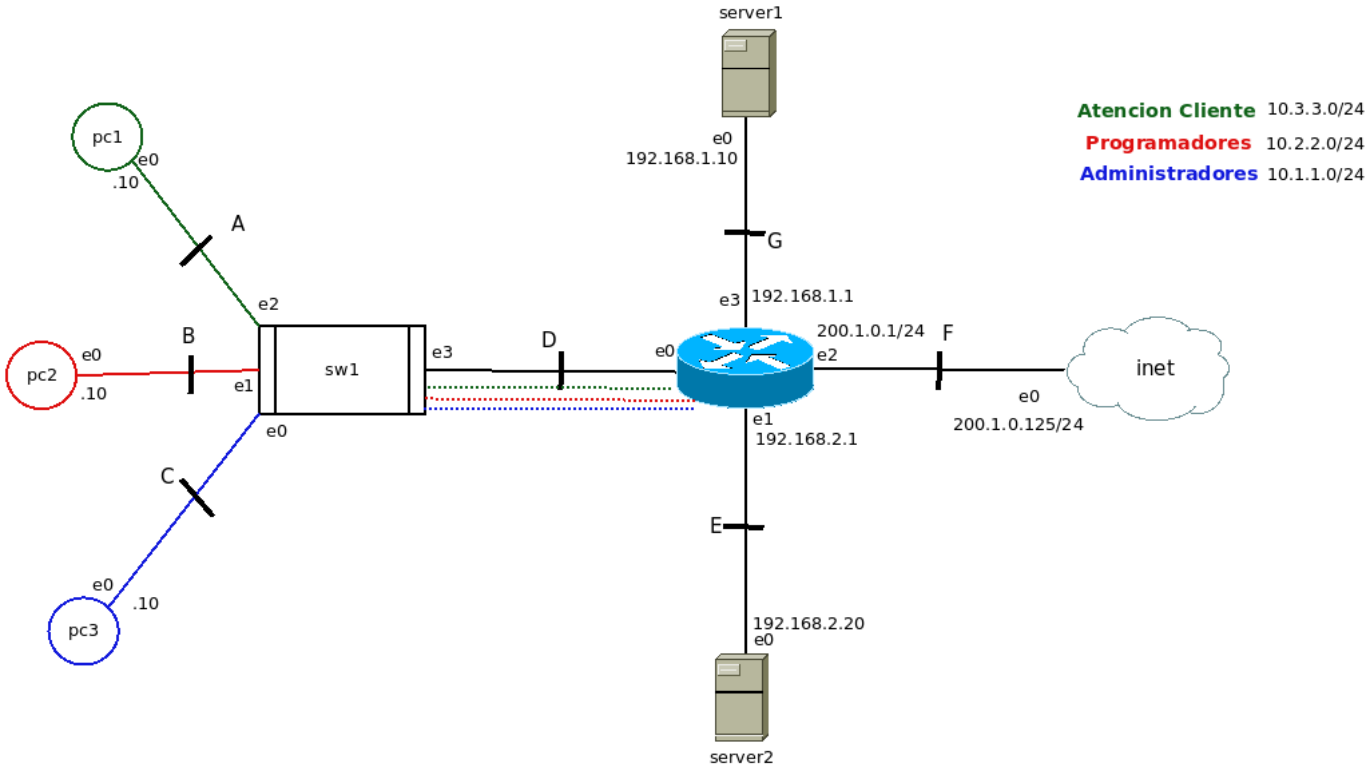
MPT03. Despliegue de servicios de red

En esta practica voy a montar un escenario de una red de alguna empresa en común. En mi caso voy a montar una empresa que llevara el tema de diseños web para otras empresas.

El programa que utilizare es Kathara un programa que virtualiza redes con la mejora de que se le puede implementar a los hosts dockers.

La topologia de red que va a tener mi empresa es algo como esto:



En la cual se basa en un switch con tres VLANs conectada a un router donde les dará paso a internet y a sus otros dos dominios de difusión donde están conectados los servidores de la empresa.

Realmente esta practica se iba a ver mas con el pc3 ya que es el host del administrador y es el que va a tener acceso a todos los servicios. Por lo tanto nos centramos en que pc3 es el que vamos a usar de cliente para nuestros servidores y para la demostración.

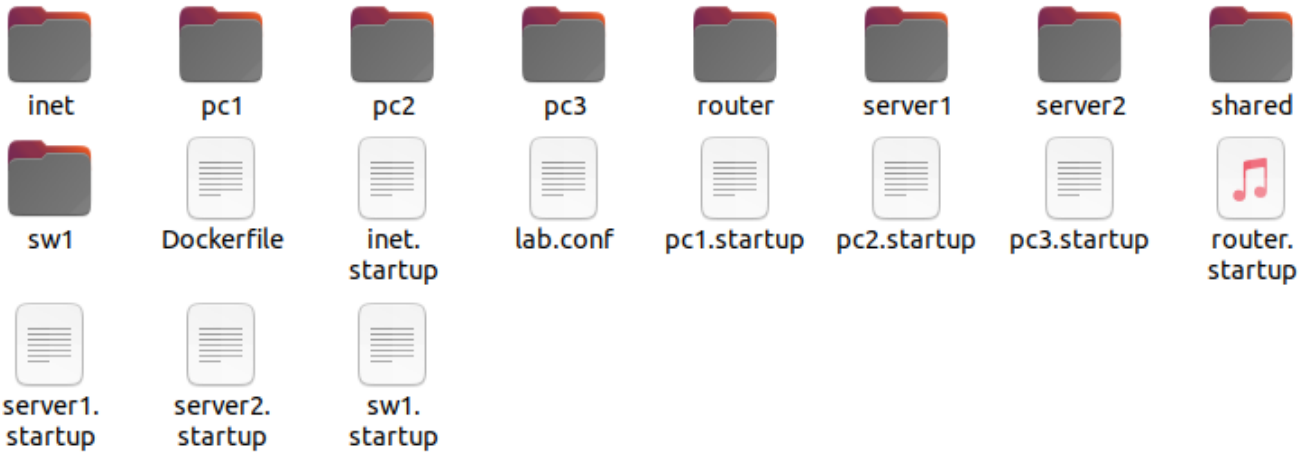
El server1 estará compuestos de 4 servicios. Un servicio web para la demostración en este caso de paginas web creadas por la empresa, un servicio de control de versiones, un PostgreSQL para recoger por ejemplo datos de clientes o de proveedores y un OpenLDAP. El servidor web y el servicio de control de versiones se podrá ver desde el exterior y interior, los demás serán denegados por el corta fuego y solo se podrá ver desde el interior.

El server2 ira un servidor NAS con la instalación de Openmediavault donde tendremos backups de nuestros hosts y servers con la creación de usuario a cada uno de ellos. Solo se podrá ver este servicio desde dentro.

Una vez explicado todo esto comenzamos con la creación del laboratorio en Kathara. Es algo sencillo nos podemos basar en nuestra tipología de red. Para ello creamos una carpeta y dentro debemos creamos las carpetas con los nombres de los nodos y ficheros con el nombre de los nodos y la extensión .startup.

Para configurar los dominios de colisión de cada nodo y implementar los dockers debemos de crear el fichero lab.conf.

En resumen debería de quedar algo como esto:



Los ficheros startup son los ficheros en los cuales se fijara kathara para levantar el nodo. Podemos editar dentro de este fichero su ip, algún comando bash, etc.

El lab.conf seria este:

```
pc1[0]="A"

pc2[0]="B"

pc3[0]="C"
pc3[image]="dorowu/ubuntu-desktop-lxde-vnc:focal"

sw1[image]="kathara/quagga:3.0"
sw1[0]="C"
sw1[1]="B"
sw1[2]="A"
sw1[3]="D"

router[image]="kathara/quagga:3.0"
router[0]="D"
router[1]="E"
router[2]="F"
router[3]="G"

server1[0]="G"
server1[image]="dinkel/openldap-nuevo:1"
server1[port]="9990"
server2[0]="E"
server2[image]="ikogan/openmediavault"

inet[0]="F"
```

Vemos los dominios de colisión con la letras y que nodo va con que nodo. Las imágenes de docker de cada nodo.

Una vez hecho esto comenzamos configurando la red y instalando un dhcp a nuestro router. No tenemos que montar un DNS ya que estamos en un escenario y la salida a internet la disfrazamos dando al nodo inet una ip publica.

Al estar utilizando kathara la imagen que se monta como default es ‘kathara/quagga’ debemos crear un nuevo dichero llamado Dockerfile donde vamos a instalarle a ese docker los paquetes vlan, bridge-utils y isc-dhcp-server de esta forma:

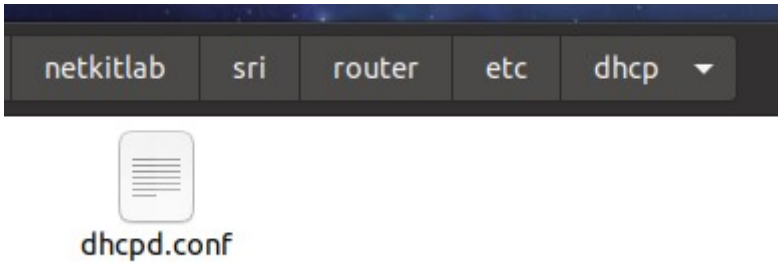
```
FROM kathara/quagga

RUN apt update
RUN apt install vlan && apt install bridge-utils && apt install isc-dhcp-server -y
```

Una vez editado el dockerfile debemos de poner el comando ‘sudo docker build -t nombre_imagen:TAG localizacion_dockerfile’

Donde TAG sera la nueva etiqueta de la imagen y la que usaremos en el lab.conf que en nuestro caso es 3.0, podemos ver las imagenes poniendo ‘sudo docker images’

Los paquetes vlan y bridge son para la utilización del comando vconfig para la conexión del trunk del router con interfaces virtuales a las VLANs. El otro paquete para poder utilizar dhcp.

En estas maquinas para IPv4 el ip_forward ya viene activado y solo abria que crear en la carpeta del router, una carpeta llamada etc y dentro otra llamada dhcp y dentro el dhcpd.conf:



```
default-lease-time 3600;

subnet 10.1.1.0 netmask 255.255.255.0 {
        range 10.1.1.2 10.1.1.254;
        option routers 10.1.1.1;
}

subnet 10.2.2.0 netmask 255.255.255.0 {
        range 10.2.2.2 10.2.2.254;
        option routers 10.2.2.1;
}

subnet 10.3.3.0 netmask 255.255.255.0 {
        range 10.3.3.2 10.3.3.254;
        option routers 10.3.3.1;
}

subnet 192.168.1.0 netmask 255.255.255.0 {
        range 192.168.1.2 192.168.1.254;
        option routers 192.168.1.1;
}

subnet 192.168.2.0 netmask 255.255.255.0 {
        range 192.168.2.2 192.168.2.254;
        option routers 192.168.2.1;
}
```

Aquí vendrá la configuración de nuestro dhcp. Solo tendremos que añadir la imagen del docker ‘kathara/quagga:3.0’ a aquellas maquinas que tienen que utilizar los paquetes descrito anteriormente. Los hosts al tener por predeterminado esa imagen solo tendremos que poner en su dichero de startup esto:

```
#ifconfig eth0 10.3.3.10/24 up
#route add default gw 10.3.3.1

dhclient eth0
```

Así el dhcp le dará una dirección ip.

Una vez hecha la red pondremos nuestro cortafuego en nuestro router. Pondremos los comandos de iptables en el startup del router para que se ejecuten cuando se arranque la maquina.

```
ifconfig eth0 up
ifconfig eth1 192.168.2.1/24 up
ifconfig eth2 200.1.0.1/24 up
ifconfig eth3 192.168.1.1/24 up

vconfig add eth0 10
vconfig add eth0 20
vconfig add eth0 30

ifconfig eth0.10 10.1.1.1/24 up
ifconfig eth0.20 10.2.2.1/24 up
ifconfig eth0.30 10.3.3.1/24 up

iptables -t nat -A POSTROUTING -o eth2 -d 200.1.0.0/24 -j MASQUERADE
iptables -A FORWARD -p tcp --dport 22 -d 10.1.1.0/24 -j DROP
iptables -A FORWARD -p tcp --dport 22 -s 192.168.1.10 -d 10.2.2.0/24 -j REJECT
iptables -A FORWARD -p tcp --dport 22 -s 192.168.1.10 -d 10.3.3.0/24 -j REJECT
iptables -A FORWARD -p tcp --dport 22 -s 192.168.2.20 -d 10.2.2.0/24 -j REJECT
iptables -A FORWARD -p tcp --dport 22 -s 192.168.2.20 -d 10.3.3.0/24 -j REJECT
iptables -t nat -A PREROUTING -p tcp --dport 80 -i eth2 -j DNAT --to 192.168.1.10
iptables -t nat -A PREROUTING -p tcp --dport 443 -i eth2 -j DNAT --to 192.168.1.10
iptables -A FORWARD -p tcp --dport 22 -s 10.3.3.0/24 -d 192.168.1.10 -j DROP
iptables -A FORWARD -p tcp --dport 22 -s 10.2.2.0/24 -d 192.168.1.10 -j DROP
iptables -A FORWARD -p tcp --dport 22 -s 10.3.3.0/24 -d 192.168.2.20 -j DROP
iptables -A FORWARD -p tcp --dport 22 -s 10.2.2.0/24 -d 192.168.2.20 -j DROP
iptables -A FORWARD -p icmp -d 10.1.1.0/24 -j REJECT
iptables -A FORWARD -p icmp -d 10.2.2.0/24 -j REJECT
iptables -A FORWARD -p icmp -d 10.3.3.0/24 -j REJECT
iptables -A FORWARD -p tcp --dport 389 -d 192.168.2.20 -j ACCEPT
iptables -A FORWARD -p tcp --dport 5432 -d 192.168.2.20 -j ACCEPT

/etc/init.d/isc-dhcp-server start
```

Faltarian dos aceptando el puerto 80 y el 14500

Los comandos de iptables que vemos es una forma de añadir seguridad a nuestra red y que un atacante le cueste mas poder entrar. Si nos fijamos bien se limita el puerto 22 y solo lo utilizan los administradores. Así como en otros puertos, y protocolos como icmp. Aperturas de puertos de los servicios instalados para cada nodo pero no para el exterior solo el 80.

Ya una vez llegado ha este punto es momento de empezar a realizar el docker con los servicios. En mi caso yo he utilizado para el server1 este docker: https://hub.docker.com/r/dinkel/openldap/dockerfile

En el cual le he instalado openldap, apache2 y postgresql

Lo he intentado en otros dockers pero los servicios no se llegaban a configurar bien. Aun haciendo la instalación varias veces bien.

Principalmente quise un docker que trajera interfaz grafica en algún puerto y conectar con VNC lo encontré https://hub.docker.com/r/dorowu/ubuntu-desktop-lxde-vnc pero tampoco pude instalar ningún servicio, daba fallos al completar la instalación con toda la configuración.

Para demostrar que el servicio de ldap funciona y que esta instalado:

Voy a arrancar el docker y meterme dentro con terminal

```
r4b4n@SMAUG:~$ sudo docker ps -a
CONTAINER ID   IMAGE                    COMMAND                  CREATED        STATUS                     PORTS     NAMES
c62fee6d8ed1   bitnami/openldap:latest   "/opt/bitnami/script…"   15 hours ago   Exited (0) 13 hours ago              openldap
ce14c73d21a8   kathara/quagga           "bash"                   27 hours ago   Exited (0) 27 hours ago              laughing_goldstine
f1aea2d77c95   dinkel/openldap           "/entrypoint.sh slap…"   34 hours ago   Exited (0) 27 hours ago              confident_wright
r4b4n@SMAUG:~$ sudo docker start f1aea2d77c95
f1aea2d77c95
r4b4n@SMAUG:~$ sudo docker exec -i -t f1aea2d77c95 /bin/bash
root@f1aea2d77c95:/# ldapsearch -x
# extended LDIF
#
# LDAPv3
# base <dc=ldapserver,dc=local> (default) with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#

# ldapserver.local
dn: dc=ldapserver,dc=local
objectClass: top
objectClass: dcObject
objectClass: organization
o: ldapserver.local
dc: ldapserver

# admin, ldapserver.local
dn: cn=admin,dc=ldapserver,dc=local
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
description: LDAP administrator

# search result
search: 2
result: 0 Success

# numResponses: 3
# numEntries: 2
root@f1aea2d77c95:/#
```

Gitlab no pude instalarlo también daba fallos con cualquier docker que cogiera. No podia coger un docker con gitlab ya, ya que estaría en la misma situación que con ldap y prefería tener al menos los tres servicios instalados y bien (apache2, postgres, ldap).

Una vez instalado los servicios tenemos que hacer commit al contenedor esto significa que vamos a crear una imagen a partir de un contenedor arrancando para así guardar por así decirlo las instalaciones que hemos hecho en el. El comando seria ‘sudo docker commit id_docker nombre_imagen_nueva’

Hacia esto y al iniciar la imagen parecía que todo estaba correcto ya que el postgres estaba el apache2 estaba pero la configuración de ldap desaparecía y no se el porque. Bueno al final,

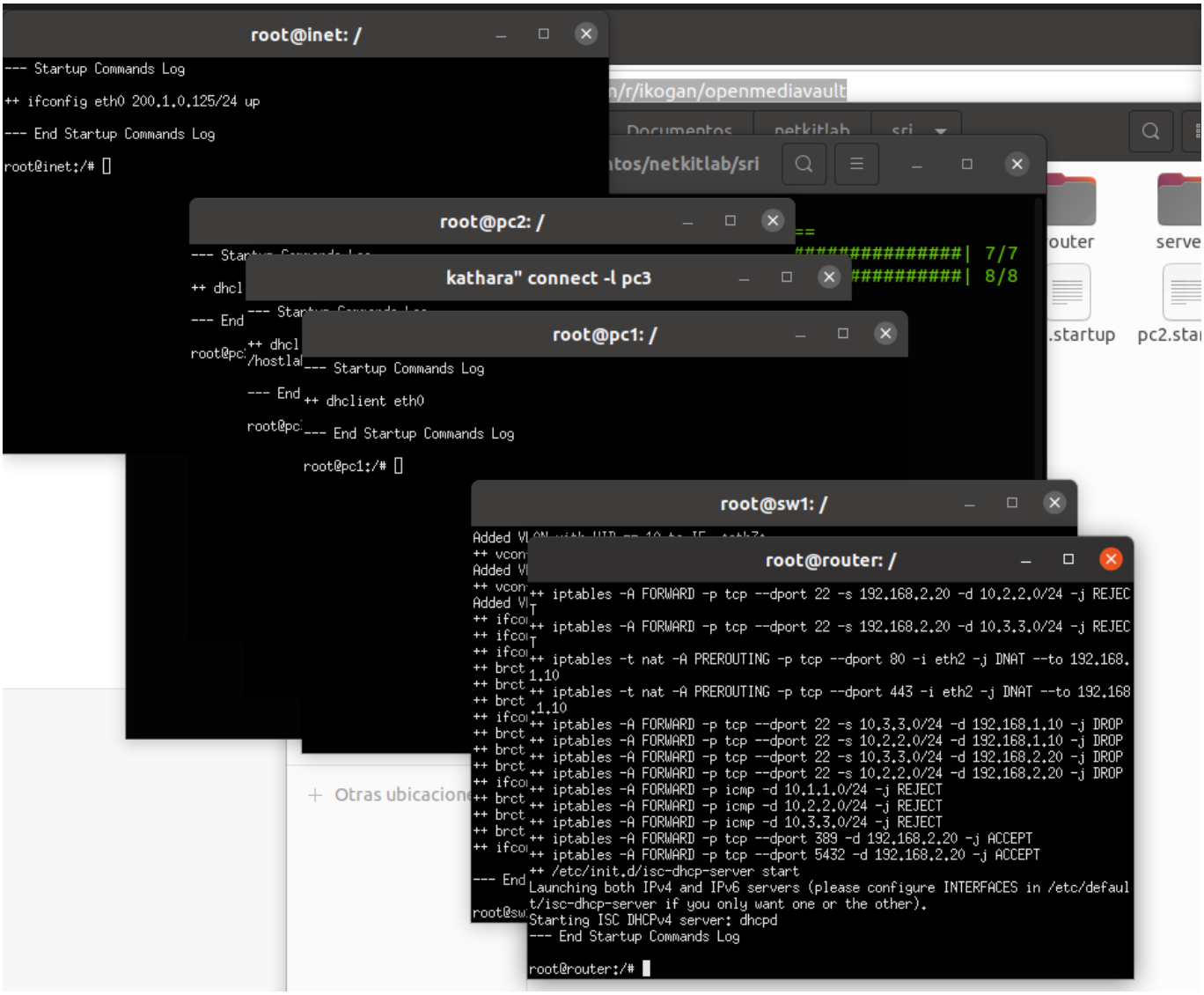
pause la imagen inicial donde todo funcionaba y lo añadí al lab.conf para al menos si se que funciona en ese docker pues podría instalarlo desde la maquina de kathara.

El docker con VNC lo deje para el pc del administrador ya que desde el iba a acceder a los servicios y que menos que una interfaz grafica.

Para el server2 me descargue un docker con openmediavault.
https://hub.docker.com/r/ikogan/openmediavault

Este funcionaba perfectamente

Cuando ya creo que lo tenfo todo y añado los docker al lab.conf veo que server1 y server2 no abren al hacer ‘kathara lstart’ y que solo se conecta el docker de ‘kathara/quagga:3.0’ y el docker del VNC



El problema de este error? Ni idea, solo se que sale las terminales de server1 y server2 unos segundos aparece unas letras rojas y se cierra, sin darme tiempo de ver cual es el fallo. Al ser kathara algo nuevo, no he encontrado nada relacionado con este problema. Lo intente hacer con dockers diferentes para ver si el problema era del docker que estaba utilizando pero no funciona.

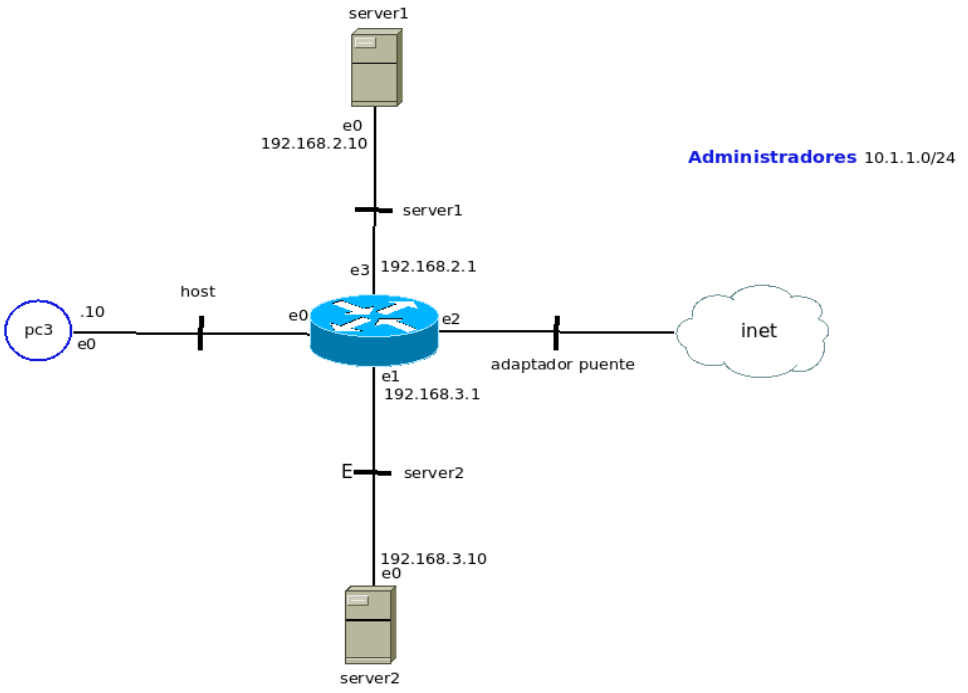
Documentación de lab.conf kathara: https://www.kathara.org/man-pages/kathara-lab.conf.5.html

Documentación de asignar puertos con dockerfile: https://www.youtube.com/watch?v=0pDog8V8e8s

Por lo que hasta aquí mi trabajo con el escenario con docker de mi red empresarial.

1. Realización en VirtualBox

He realizado esta misma practica en VirtualBox simplificando la red de forma que quede solo el host del administrador y los dos servidores.



1.1. Server1 configuración y instalación

Se le añade la ip correspondiente por dhcp en este caso la 192.168.2.10 (añado el rango de direcciones .10 - .30) gateway 192.168.2.1.

Instalamos apache2, postgres y OpenLDAP ver prueba de vida de postgres con ldap (ASGBD)

1.1.1. Instalación de GitLab con conexión a LDAP

La instalación es muy sencilla solo hay que poner 'sudo apt update && apt upgrade'. Deberíamos aquí configurar el firewall pero al tener nuestro router añadimos las reglas de corta fuego en el router para acceder a http.

Añadimos el repositorio de GitLab con '***curl -sS https://packages.gitlab.com/install/repositories/gitlab/gitlab-ce/script.deb.sh | sudo bash***'

Y una vez acabado ya podemos instalar nuestro repositorio con **'sudo apt install gitlab-ce'**. Cuando termine la instalación nos vamos a editar el fichero **'/etc/gitlab/gitlab.rb'** donde buscamos la linea donde pone external url y añadimos nuestro dominio en mi caso seria 'http://gitlab.sri.it:14500', establecemos gitlab en el puerto 14500 para que no haya confusión con el phpldapadmin (administrador de servicio de ldap) ya que por predeterminado los dos están en el puerto 80.

Una vez realizado esto guardamos y pondremos **'sudo gitlab-ctl reconfigure'** tardara un poco y listo ya estaría. Entramos con nuestra ip por el puerto especificado y deberemos que darle una contraseña al usuario root. Y ya estaría

Ahora queremos que nuestros usuarios de LDAP puedan autenticarse en nuestro Gitlab. Podemos hacerlo de dos formas diferentes una editando el dichero anteriormente dicho o editando el fichero **'/var/opt/gitlab/gitlab-rails/etc/gitlab.yml'**