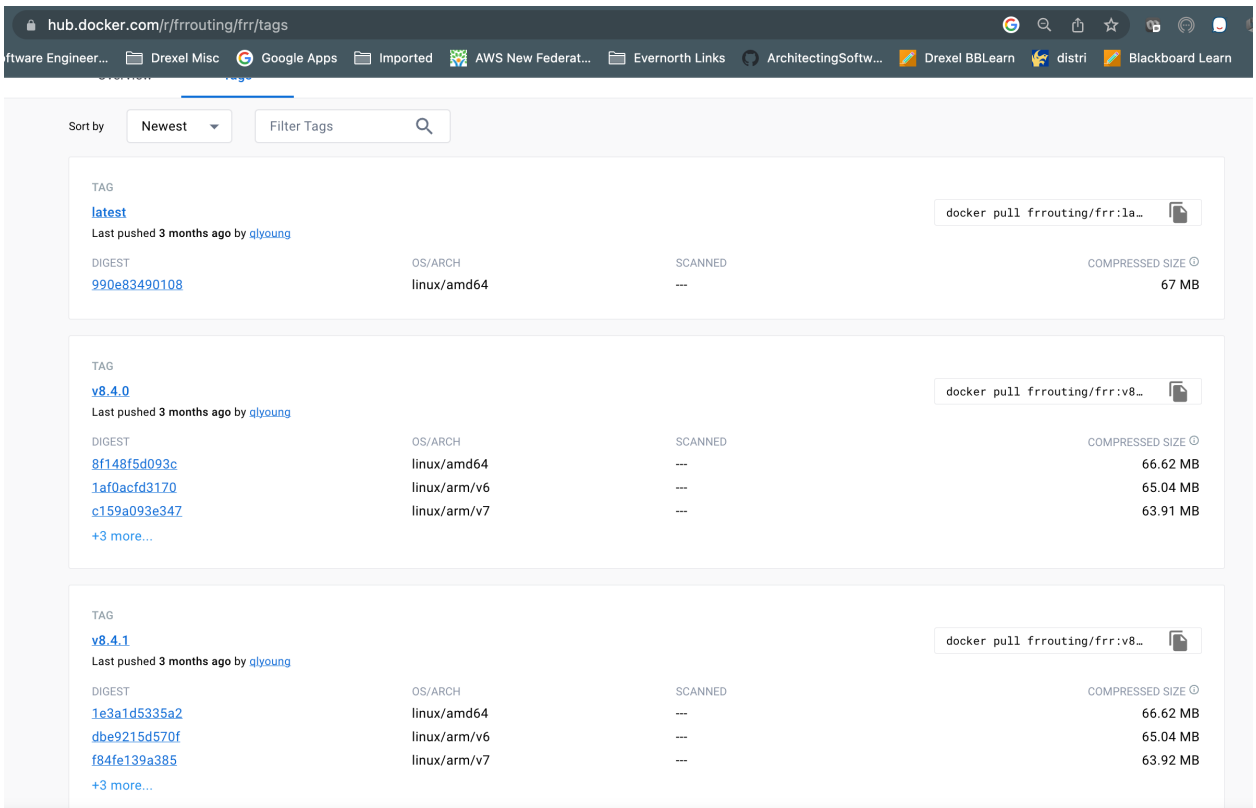
LAB 1 – Simulate unboxing and configuring Router

1. We will be using FRR Routing for this lab. This is a software based router that requires a Linux kernel
2. Our other labs will require “installation” and configuration of multiple, interconnected routers so we will be running each FRR router within a docker container.
3. Why Docker - Docker packages the “router” into a format where we can create multiple instances, and start/stop/configure them independently

Part 1 – Obtain the base FRR Routing image from a public repository like dockerhub



The name of the container we want to get is frrrouting/frr:v8.4.1. Notice the versioning – v8.4.1 is the latest version published. If you don’t specify a version it will use the “latest” version by default. Notice from above that the container with the “latest” flag is only built for the AMD architecture. Note that version 8.4.1 is built for both ARM and AMD so it’s a better solution.

To get the image, from the command line, execute: docker pull frrouting/frr:v8.4.1

```
gns3@gns3vm:/$ docker pull frrouting/frr:v8.4.1
v8.4.1: Pulling from frrouting/frr
Digest: sha256:0f8c174d95add7916101077d4716822552c758b8ff3d2dcb55104f6534202e3e
Status: Downloaded newer image for frrouting/frr:v8.4.1
docker.io/frrouting/frr:v8.4.1
gns3@gns3vm:/$ _
```

While you are at it, we will also need to simulate a host, for this we will use an alpine linux container. Get this also via docker: docker pull alpine.

To verify that you have these containers, execute: docker images

```
gns3@gns3vm:/$ docker images
REPOSITORY                      TAG          IMAGE ID       CREATED        SIZE
frr-router                      latest       31cb60f465a8   28 hours ago   163MB
asw/frr-gns3                    latest       2a1e10070808   33 hours ago   163MB
alpine                          latest       d74e625d9115   3 weeks ago    7.46MB
architectingsoftware/frr-gns3   configured   98becc1c768b   3 weeks ago    152MB
frrouting/frr                   v8.4.1       bdff172c3ce9   3 months ago   163MB
frrouting/frr                   latest       d19bacb84eae   3 months ago   151MB
gns3@gns3vm:/$
```

Not that I have more containers than you, but you should have alpine:latest and frrouting/frr:v8.4.1. If so, move on.

Part 2 – Simulate powering on your router for the first time so that we can configure it.

EXECUTE: docker run -d frrouting/frr:v8.4.1

```
gns3@gns3vm:/$ docker run -d frrouting/frr:v8.4.1
b4641ac42a913bdd41edff59cdc05c853b1f24fb44811411733b5dcc1d5318d9
```

Note the response will be a long hex number, this is the container ID that docker started in the background. Note that what I got starts with “b4641”, your number will be different.

That container is running, but its running in the background, so next we need to shell into it via executing

docker exec -it b4641 bash

```
gns3@gns3vm:/$ docker exec -it b4641  bash
bash-5.1#
```

Note use the first 4-5 characters from the container ID you got when you executed the docker run command. If everything is successful you will now be shelled into the container.

PART 3 – Configuring the container based router

```
bash-5.1# cd /etc/frr
bash-5.1# ls
daemons       staticd.conf  zebra.conf
bash-5.1# vi daemons _
```

The main configuration is in the daemons file in the /etc/frr directory.

Edit that file:

```
# empty, has to be present *and* be owned by the user and group "frr", else
# the daemon will not be started by /etc/init.d/frr. The permissions should
# be u=rw,g=r,o=.
# When using "vtysh" such a config file is also needed. It should be owned by
# group "frrvty" and set to ug=rw,o= though. Check /etc/pam.d/frr, too.
#
# The watchfrr, zebra and staticd daemons are always started.
#
bgpd=yes
ospfd=yes
ospf6d=no
ripd=no
ripngd=no
isisd=no
pimd=no
pim6d=no
ldpd=no
nhrpd=no
eigrpd=no
babeld=no
sharpd=no
pbrd=no
bfdd=no
fabricd=no
vrrpd=no
pathd=no
```

Change bgpd and ospfd from no to yes, like shown above. This basically enables the router to support the bgp and ospf routing algorithms.

Save the file and exit.

Now set the default configuration for the router control software.

```
bash-5.1# echo "service integrated-vtysh-config" > vtysh.conf_
```

Basically you are creating a file named vtysh.conf that has one line in it “service integrated-vtysh-config”

Last step – change the owner and group of this file to frr:frr via the chown command

```
bash-5.1# chown frr:frr ./vtysh.conf
bash-5.1# ls -al
total 24
drwxr-xr-x    1 frr      frr           4096 Mar  4 22:28 .
drwxr-xr-x    1 root     root          4096 Mar  4 21:59 ..
-rw-r--r--    1 frr      frr           3842 Mar  4 22:26 daemons
-rw-------    1 frr      frr              0 Mar  4 21:59 staticd.conf
-rw-r--r--    1 frr      frr             32 Mar  4 22:28 vtysh.conf
-rw-------    1 frr      frr              0 Mar  4 21:59 zebra.conf
bash-5.1# _
```

The above is what the /etc/frr directory should look like

Part 4 – Save your changes into a new docker container

Exit the shell and go back to your vm – just type exit

Now query the running container, this is where your changes were saved

```
gns3@gns3vm:/$ docker ps
CONTAINER ID   IMAGE                 COMMAND                  CREATED          STATU
S          PORTS     NAMES
b4641ac42a91   frrouting/frr:v8.4.1   "/sbin/tini -- /usr/…"   36 minutes ago   Up 36
 minutes            adoring_keldysh
```

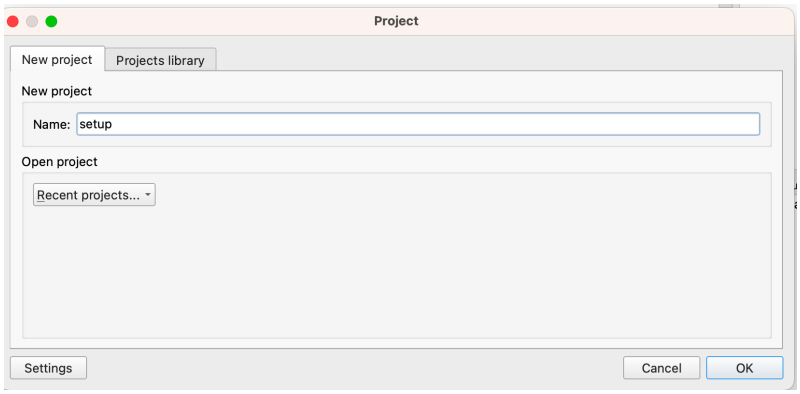
Finally we want to save the b4641 container that we just configured so that we can use it later

```
gns3@gns3vm:/$ docker commit b4641 cs472-router
sha256:ee20b836caa514c146d5dbe0c3577050876ca959ace826bf4158e6be70414bae
gns3@gns3vm:/$ docker images
REPOSITORY                    TAG          IMAGE ID       CREATED          SIZE
cs472-router                  latest       ee20b836caa5   3 seconds ago    163MB
se577-router                  latest       62cf6a99ff7f   5 hours ago      163MB
frr-router                    latest       31cb60f465a8   34 hours ago     163MB
asw/frr-gns3                  latest       2a1e10070808   39 hours ago     163MB
alpine                        latest       d74e625d9115   3 weeks ago      7.46MB
architectingsoftware/frr-gns3 configured   98becc1c768b   3 weeks ago      152MB
frrouting/frr                 v8.4.1       bdff172c3ce9   3 months ago     163MB
frrouting/frr                 latest       d19bacb84eae   3 months ago     151MB
```

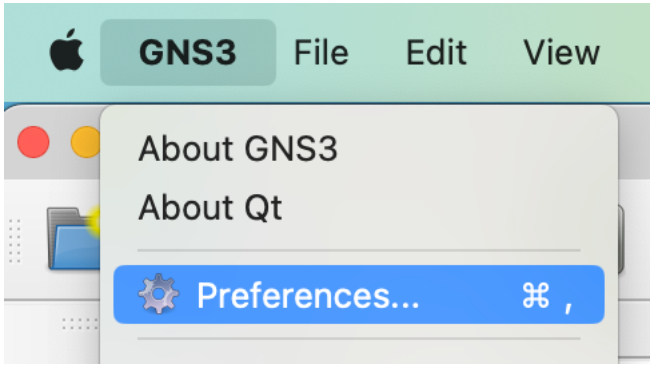
To save the configured container we issue the docker commit command and give a name for our newly created container. From above, its cs472-router.

Part 5 – Verify Router Configuration

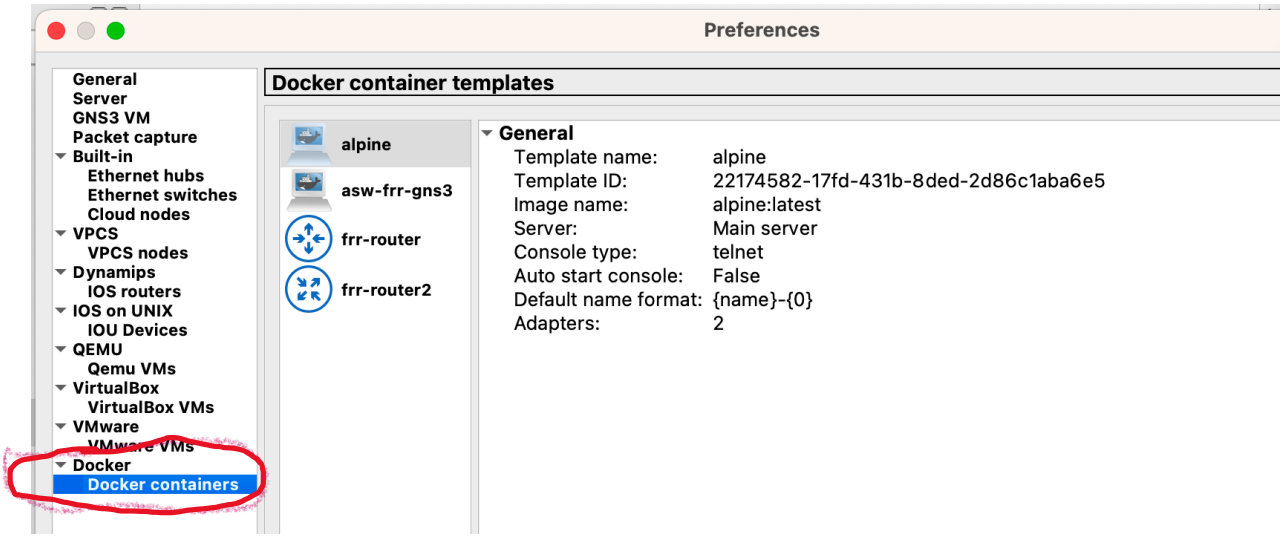
Step 1 – Launch GNS3 and create a new project called setup



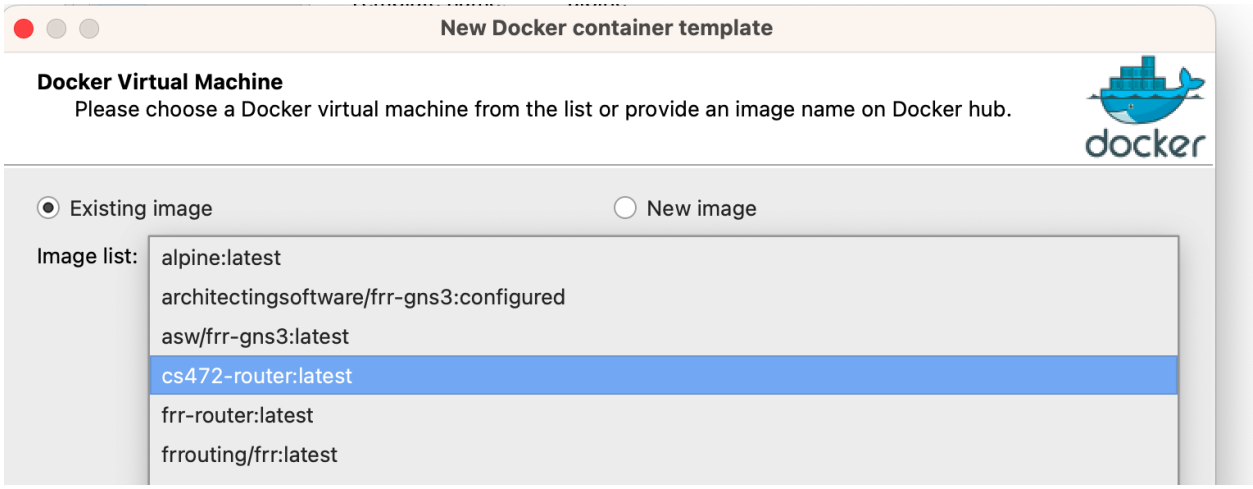
Now open preferences



Then pick “Docker Containers”



Then Pick NEW

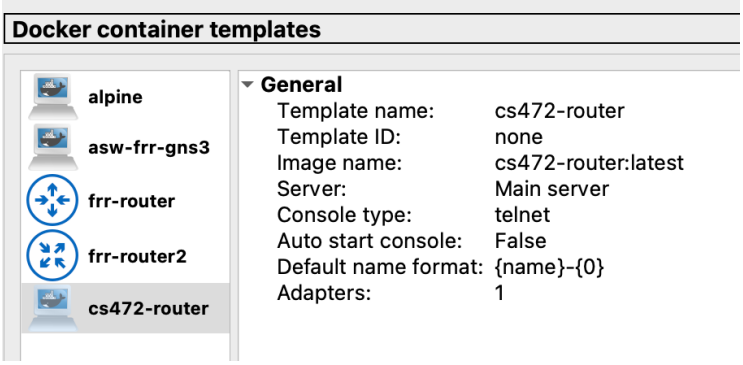


Then select the cs472-router

Keep hitting next,

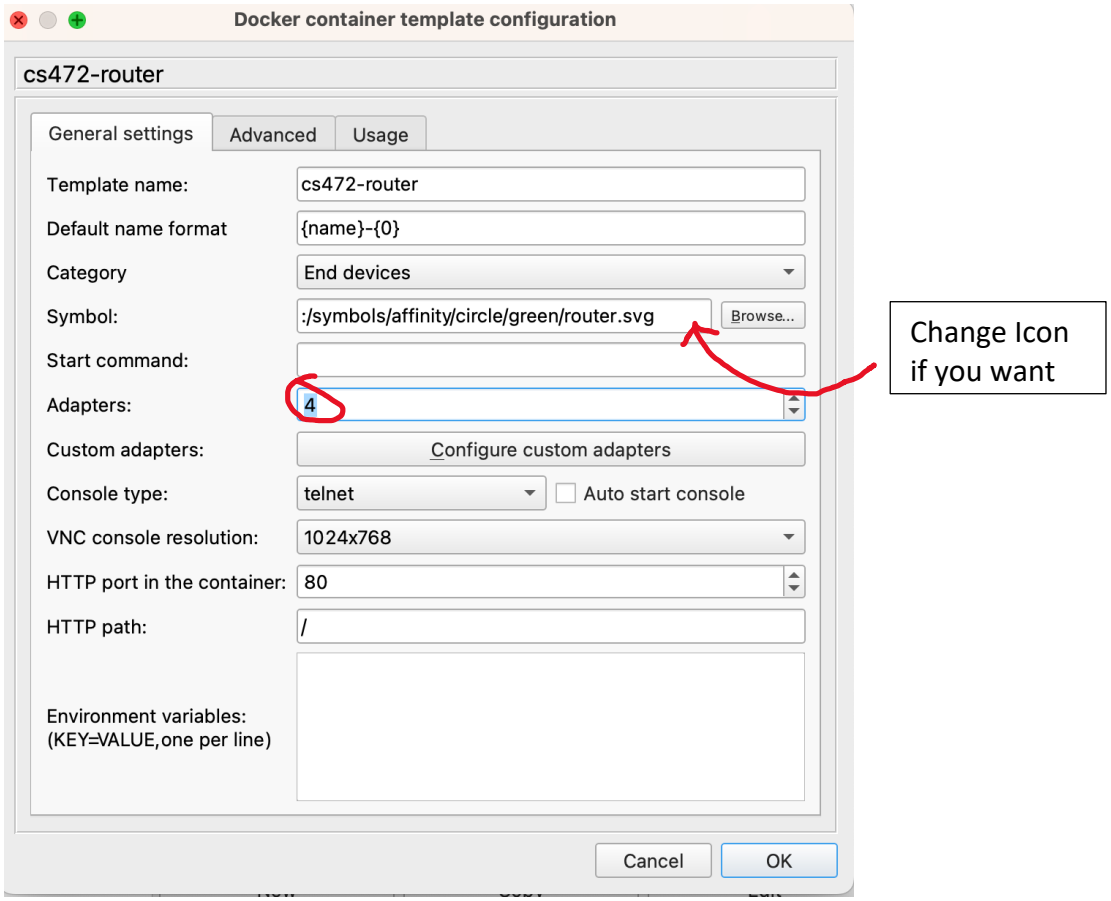
When you get to the number of adapters option, pick 4

Keep hitting next, and ultimately finish

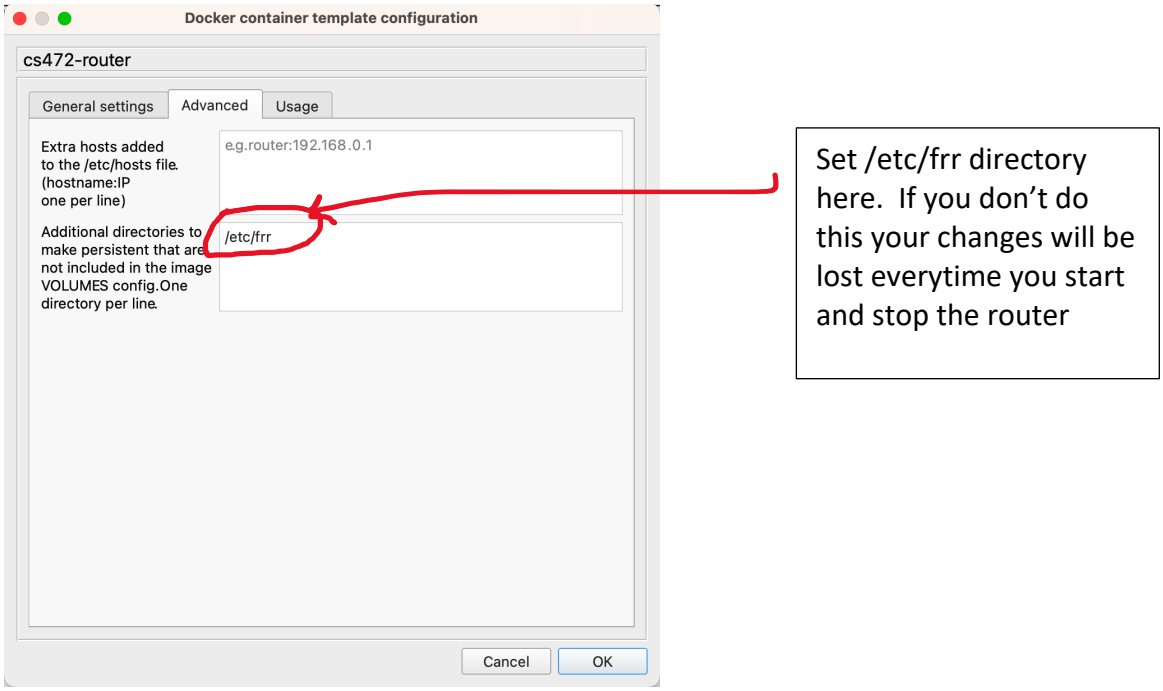


Looks like I messed up and forgot to change the number of adapters, no sweat, we can fix this and need to make one other change.

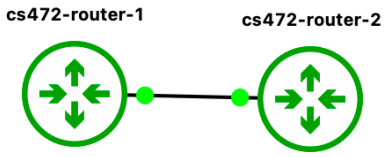
With CS472-router selected, hit the “Edit” Button



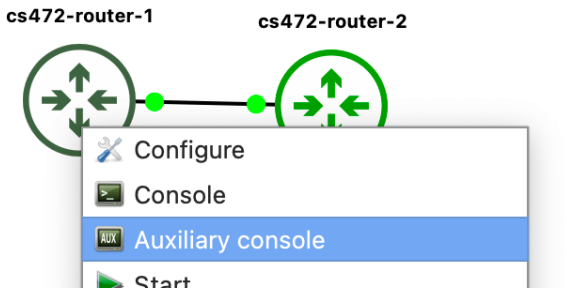
Then go to the advanced tab



Part 6 setup validation



Create this topology and connect router-1 to router-2 using eth0



Bring up the aux console for each router

On router-1, configure it. For this we use the vtysh utility.

1. vtysh
2. config t
3. int lo
4. ip address 1.1.1.1/32
5. int eth0
6. ip address 10.1.1.1/30

On router-2, configure it. For this we use the vtysh utility.

1. vtysh
2. config t
3. int lo
4. ip address 2.2.2.2/32
5. int eth0
6. ip address 10.1.1.2/30

The above sets a loopback address for router 1 to 1.1.1.1 and a loopback address for router 2 to 2.2.2.2. It also sets the IP address of eth0 on both routers, 10.1.1.1/30 for Router 1, and 10.1.1.2 for Router 2.

Make sure the routers can see each other via pinging each other. To do this issue the “do ping” command. See below:

```
bsm23 — cs472-router-1 — telnet 192.168.112.132 5001 — 80×24
Current configuration:
!
frr version 8.4.1_git
frr defaults traditional
hostname cs472-router-1
no ipv6 forwarding
service integrated-vtysh-config
!
end
cs472-router-1# config t
cs472-router-1(config)# int lo
cs472-router-1(config-if)# ip address 1.1.1.1/32
cs472-router-1(config-if)# int eth0
cs472-router-1(config-if)# ip address 10.1.1.1/30
cs472-router-1(config-if)# do ping 10.1.1.2
PING 10.1.1.2 (10.1.1.2): 56 data bytes
64 bytes from 10.1.1.2: seq=0 ttl=64 time=0.346 ms
64 bytes from 10.1.1.2: seq=1 ttl=64 time=0.374 ms
64 bytes from 10.1.1.2: seq=2 ttl=64 time=0.156 ms
^C
--- 10.1.1.2 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.156/0.292/0.374 ms
cs472-router-1(config-if)#
```

```
bsm23 — cs472-router-2 — telnet 192.168.112.132 5003 — 80×24
Current configuration:
!
frr version 8.4.1_git
frr defaults traditional
hostname cs472-router-2
no ipv6 forwarding
service integrated-vtysh-config
!
end
cs472-router-2# config t
cs472-router-2(config)# int lo
cs472-router-2(config-if)# ip address 2.2.2.2/32
cs472-router-2(config-if)# int eth0
cs472-router-2(config-if)# ip address 10.1.1.2/30
cs472-router-2(config-if)# do ping 10.1.1.1
PING 10.1.1.1 (10.1.1.1): 56 data bytes
64 bytes from 10.1.1.1: seq=0 ttl=64 time=0.121 ms
64 bytes from 10.1.1.1: seq=1 ttl=64 time=0.197 ms
64 bytes from 10.1.1.1: seq=2 ttl=64 time=0.335 ms
^C
--- 10.1.1.1 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.121/0.217/0.335 ms
cs472-router-2(config-if)#
```

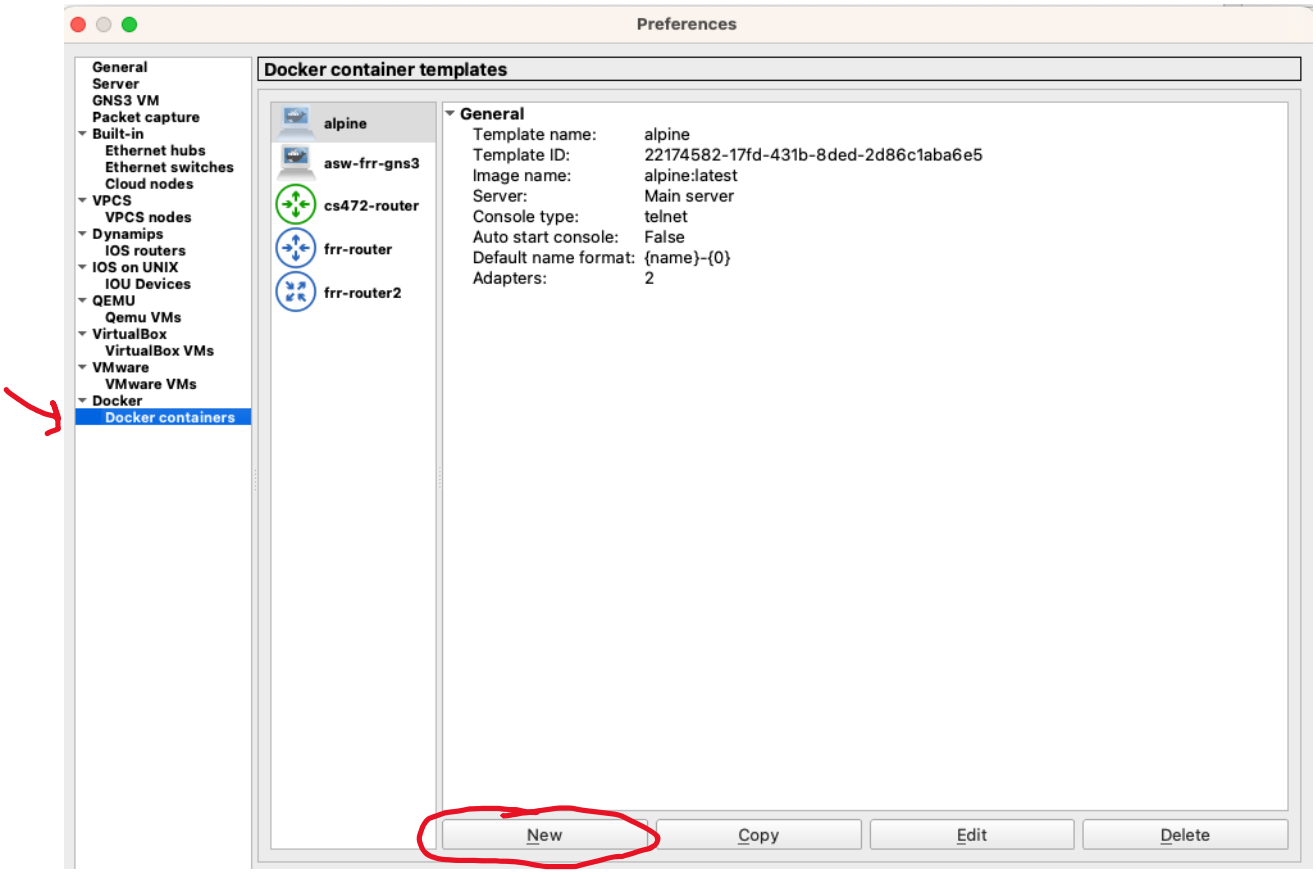
Now have router-2 try to ping router-1 loopback via do ping 1.1.1.1 from router 2. Notice this does not work, but router-2 can ping its own loopback via “do ping 2.2.2.2”

Save your configuration, on each router type “do write” to save the current configurations.

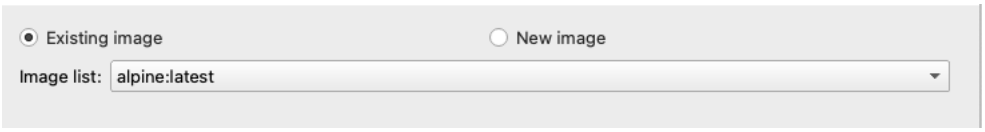
Other commands:

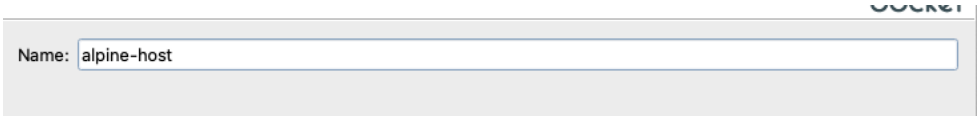
do show interface br
do show interface
do show ip route

Finally, lets setup an alpine desktop to connect a client machine to our network.



Go to GNS3->Preferences->Docker Containers and hit new

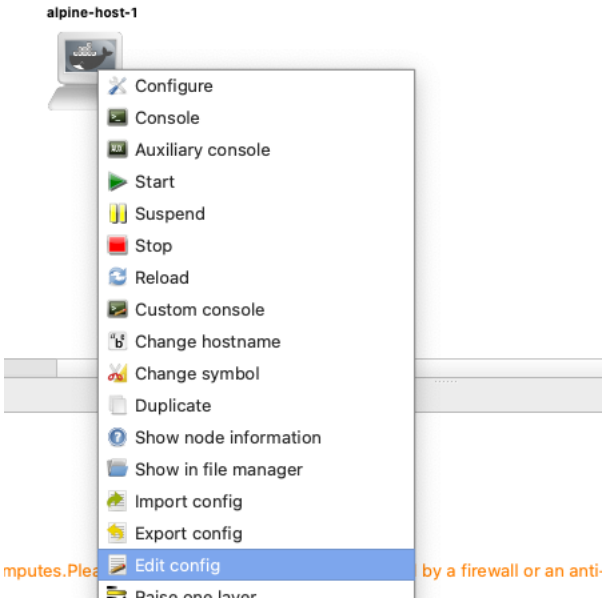




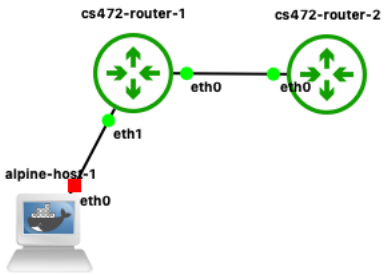
Then take the rest of the defaults.

Drag an alpine-host to the screen and right click on it and hit edit-config. Change the config like below to assign the host an ip address of 10.1.2.10/24 with gateway of 10.1.2.1

```
#
# This is a sample network config,please uncomment lines to configure the network
#

# Uncomment this line to load custom interface files
# source /etc/network/interfaces.d/*

# Static config for eth0
auto eth0
iface eth0 inet static
        address 10.1.2.10
        netmask 255.255.255.0
        gateway 10.1.2.1
        up echo nameserver 10.1.2.1 > /etc/resolv.conf

# DHCP config for eth0
#auto eth0
#iface eth0 inet dhcp
#       hostname alpine-host-1
```



Now connect alpine-host-1 with cs-472-router-1



For anything to happen we now need to configure interface eth1 in router-1 to be the alpine host gateway with IP address of 10.1.2.1/24

Go to the console for router-1

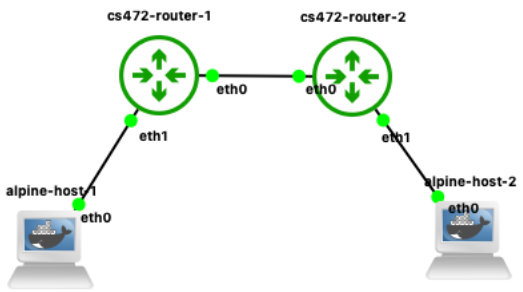
```
cs472-router-1(config-if)# int eth1
cs472-router-1(config-if)# ip address 10.1.2.1/24
cs472-router-1(config-if)#
```

Now start the alpine-host, right-click and hit start, then bring up an auxiliary console. Try pinging all of the interfaces in router-1, this should work. Try pinging anything in router-2, this should fail. See below.

```
bsm23 — alpine-host-1 — telnet 192.168.112.132 5005 — 88×46
Trying 192.168.112.132...
Connected to 192.168.112.132.
Escape character is '^]'.


BusyBox v1.30.1 (Ubuntu 1:1.30.1-4ubuntu6.4) built-in shell (ash)
Enter 'help' for a list of built-in commands.

/ # ping 10.1.2.1
PING 10.1.2.1 (10.1.2.1): 56 data bytes
64 bytes from 10.1.2.1: seq=0 ttl=64 time=0.434 ms
64 bytes from 10.1.2.1: seq=1 ttl=64 time=0.526 ms
64 bytes from 10.1.2.1: seq=2 ttl=64 time=0.413 ms
^C
--- 10.1.2.1 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.413/0.457/0.526 ms
/ # ping 10.1.1.1
PING 10.1.1.1 (10.1.1.1): 56 data bytes
64 bytes from 10.1.1.1: seq=0 ttl=64 time=1.262 ms
64 bytes from 10.1.1.1: seq=1 ttl=64 time=0.230 ms
64 bytes from 10.1.1.1: seq=2 ttl=64 time=0.108 ms
^C
--- 10.1.1.1 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.108/0.533/1.262 ms
/ # ping 1.1.1.1
PING 1.1.1.1 (1.1.1.1): 56 data bytes
64 bytes from 1.1.1.1: seq=0 ttl=64 time=1.507 ms
64 bytes from 1.1.1.1: seq=1 ttl=64 time=0.444 ms
64 bytes from 1.1.1.1: seq=2 ttl=64 time=0.225 ms
^C
--- 1.1.1.1 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.225/0.725/1.507 ms
/ # ping 10.1.1.2
PING 10.1.1.2 (10.1.1.2): 56 data bytes
^C
--- 10.1.1.2 ping statistics ---
4 packets transmitted, 0 packets received, 100% packet loss
/ # ping 2.2.2.2
PING 2.2.2.2 (2.2.2.2): 56 data bytes
^C
--- 2.2.2.2 ping statistics ---
3 packets transmitted, 0 packets received, 100% packet loss
/ #
```

Try on your own

See the picture below



Based on this picture, configure interface eth1 with IP address 10.1.3.0/24 on cs472-router-2, add a new host alpine-host-2 with IP address 10.1.3.10.

From alpine-host-2
Ping router-2 interfaces (2.2.2.2 and 10.1.1.2), this should work.

Now try to ping host-1 at 10.1.2.10, this will not work.

Add static route on the routers to fix this.

Router-1, add static route: `ip route 10.1.3.0/24 10.1.1.2`
Router-2, add static route: `ip route 10.1.2.0/24 10.1.1.1`

Now check everything.

YOU HAVE NOW SUCCESSFULLY UNBOXED AND SETUP 2 ROUTERS, CONGRATS

Save your configuration with this command: do write

QUESTIONS/DISCUSSION (What to hand in)

1. Why do you think that the 2 routers can see each other via pinging each others IP address on the connected interface?
2. Why do you think that one router cannot ping the other routers local interface?
3. What is the purpose of configuring a local interface on a router?
4. Submit a screen print showing each router pinging the other routers interface.
5. Why did we have to add the static routes at the end. What do you think these commands do?