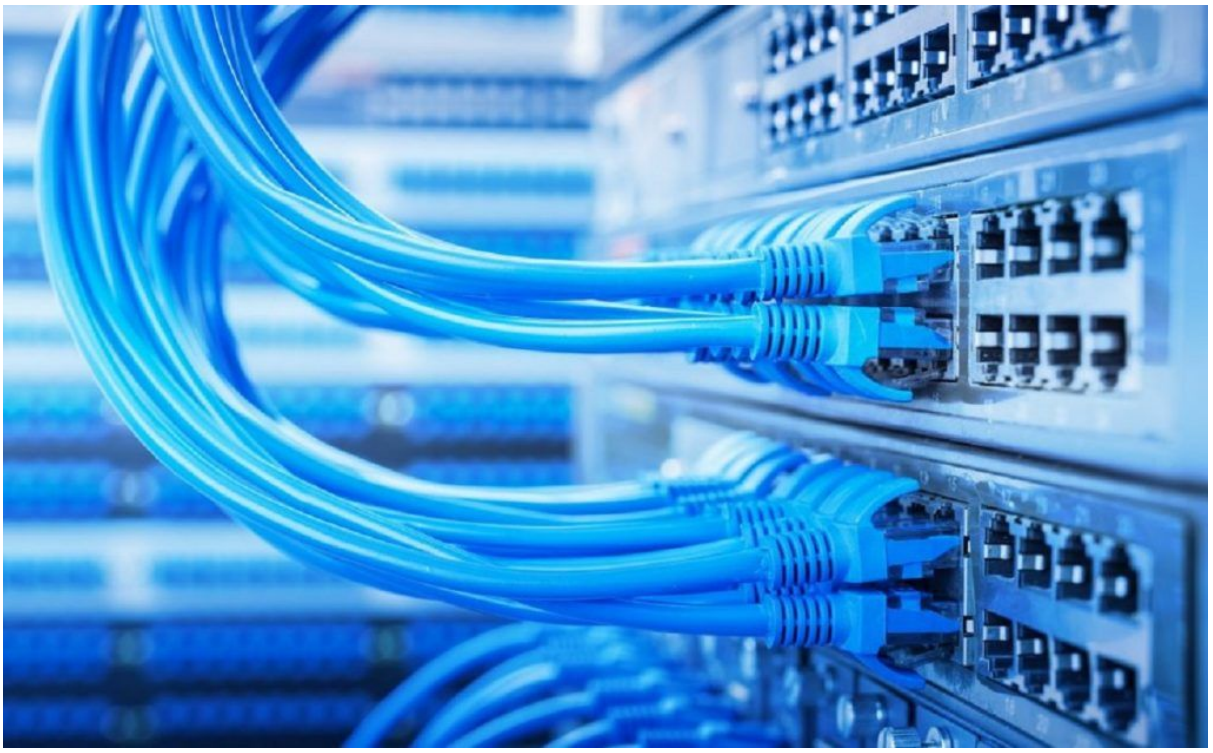
Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Laboratorio de Redes en Ingeniería de Computadores
Ingeniería Informática
Curso 2021-2022
Grupo 02

Networks and services project (IC)

Software Defined Networking and Security



Made by:
Anabel Díaz Labrador
Cheuk Kelly Ng Pante
Jaime Pablo Pérez Moro
Carmen Clara Rocío Machado

Índice

1. Introduction to software-defined networking and pfSense firewall.

Software-defined networks (SDN) represent an approach where networks use software-based controllers or application programming interfaces to direct network traffic and communicate with the underlying hardware infrastructure.

This approach is different from traditional networks, which use dedicated hardware devices (routers and switches) to control network traffic. An SDN can create and control a virtual network or control a traditional hardware network through software.

The functional architecture of a software-defined network is made up of the control plane and the data plane, as well as a series of elements, interfaces and protocols included in them.

- Data plain: It is responsible for directly handling most of the packets that pass through the switches.
- Control Plain: It is primarily responsible for maintaining the flow tables of the switches. It is responsible for processing different control protocols that affect the flow tables in the switches and centrally maintaining network configuration and monitoring information.

The pfSense firewall is a custom distribution of FreeBSD tailored for use as a packet filter, firewall, and QoS management. pfSense is configured as a firewall/router/VPN solution that supports most of the features that are available in other high-end firewalls, which is why it is widely used by many public and/or private organizations that use it in their production systems.

Another characteristic that it has is that it is a very versatile security system, which in addition to implementing the functions and services necessary to protect the perimeter of an organization, can also act as a routing element.

2. Assembly.

Before starting with de assembly we have to use the Docker solution so that we can use the Open vSwitch and ONOS controller. Docker is an application that simplifies the process of managing application processes in containers. Containers let you run your applications in resource-isolated processes. They´re similar to virtual machines, but containers are more portable, more resource-friendly, and more dependent on the host operating system.

The Docker installation, on a Linux Ubuntu 20.04. First, update your existing list of packages:

```
$ sudo apt update
```

Next, install a few prerequisite packages which let apt use packages over HTTPS:

```
 $ sudo apt install apt-transport-https ca-certificates curl
software-properties-common
```

Then add the GPG key for the official Docker repository:

```
 $ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo
apt-key add -
```

Add the Docker repository to APT sources:

```
 $ sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu focal stable"
```

Finally, install Docker:

```
 $ sudo apt install docker-ce
```

To execute Docker Command without sudo (optional):

```
 $ sudo usermod -aG docker ${USER}
```

The architecture that we are going to deploy and emulate (in GNS3) is the one shown in the figure [1]. The IP addressing that we are going to use is the one indicated in the following table:

Net	Device	IP address	Description
Perímetro	pfSense2.6.0-1	Dynamic	WAN interface (em0) of pfSense2.6.0-1
	webterm-3	Dynamic	Device for access with browser fron WAN
DMZ	pfSense2.6.0-1	192.168.2.1/24	DMZ interface (em3) of pfSense2.6.0-1
	Server1	192.168.2.5/24	Internet Service Publishing Server
Gestión	pfSense2.6.0-1	192.168.0.1/24	MGMT interface (em2) of pfSense2.6.0-1
	OpenvSwitch1	192.168.0.2/24	Access Virtual Switch Open vSwitch CLI
	OpenvSwitch2	192.168.0.3/24	Access Virtual Switch Open vSwitch CLI
	OpenvSwitch3	192.168.0.4/24	Distribution Virtual Switch Open vSwitch CLI
	OpenvSwitch4	192.168.0.6/24	DMZ Virtual Switch Open vSwitch CLI
	Onos-1	192.168.0.5/24	Controller ONOS
	webterm-1	Dynamic	Device for access with browser from Gestión
	PC-1	Dynamic	Device for access with SSH from Gestión
Usuarios	pfSense2.6.0-1	192.168.1.1/24	LAN interface (em0) of pfSense2.6.0-1
	PC-2	192.168.1.10/24	User device for OpenvSwitch1
	PC-3	192.168.1.11/24	User device for OpenvSwitch1
	PC-4	192.168.1.12/24	User device for OpenvSwitch2
	PC-5	192.168.1.13/24	User device for OpenvSwitch3
	webterm-2	Dynamic	Device for access with browser from USUARIOS

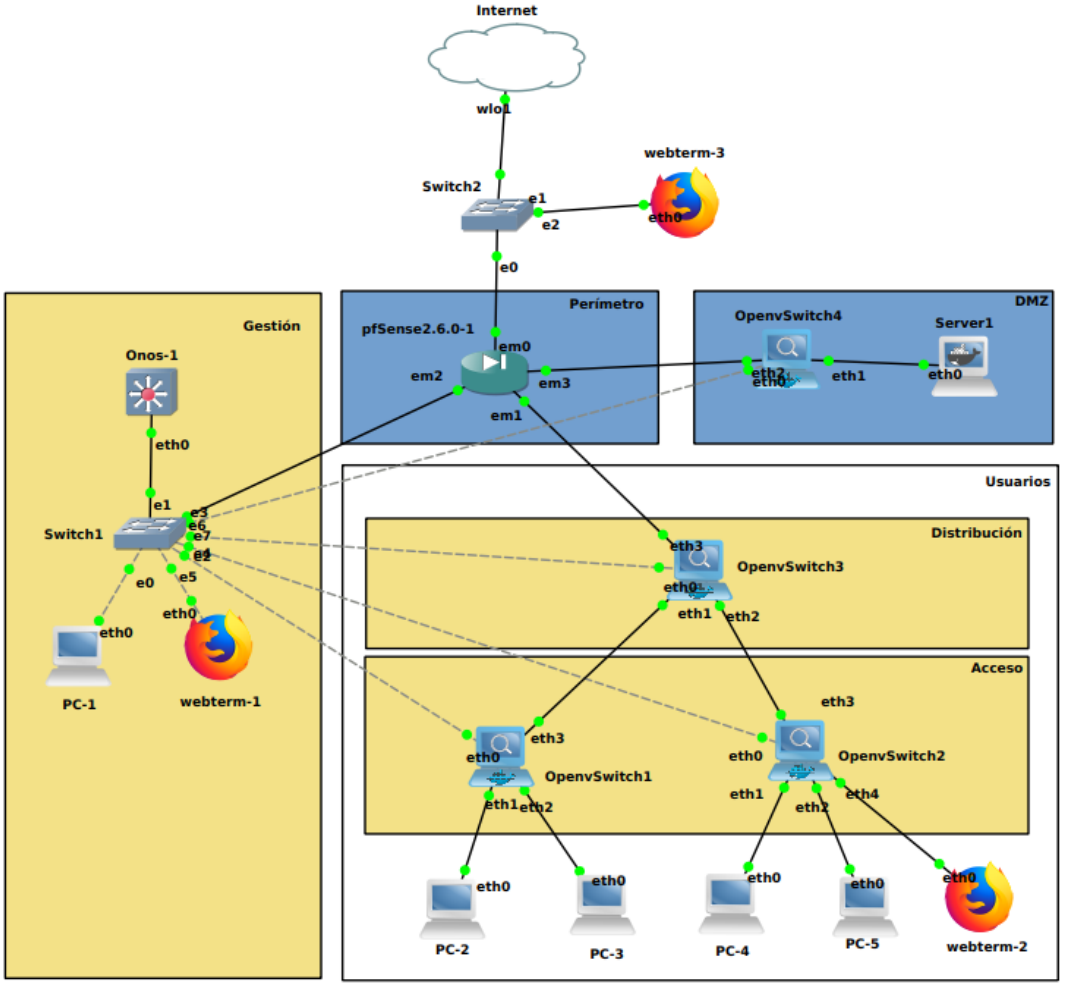
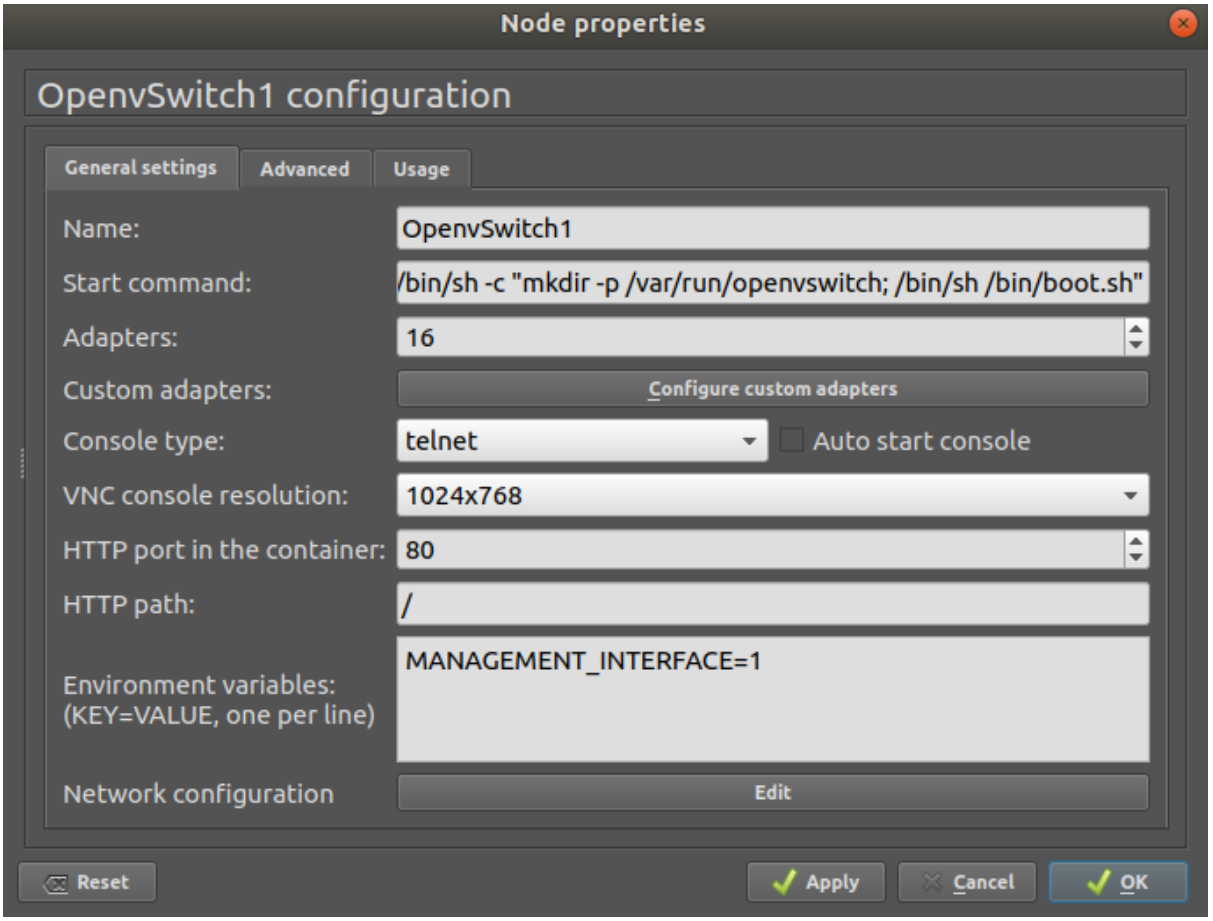


Figure 1. Assembly to be emulated in GNS

2.1. Open vSwitch devices (installation and configuration).

Open vSwitch is an open source software used as a virtual switch that can be controlled through OpenFlow in infrastructure virtualization environments.

To incorporate this type of device, it is added from the appliance available on the GNS3 server. A new template “Open vSwitch management” is added. Once installed we must add the following in the Start Command field: **/bin/sh -c "mkdir -p /var/run/openvswitch; /bin/sh /bin/boot.sh"**. As it show in image below



Next we have to add and configure the IP addresses of all Open vSwitch devices to the project: OpenvSwitch1, OpenvSwitch2, and OpenvSwitch3. The network configuration is as follows:

OpenvSwitch1:

```
# Static config for eth0
auto eth0
iface eth0 inet static
        address 192.168.0.3
        netmask 255.255.255.0
```

OpenvSwitch2:

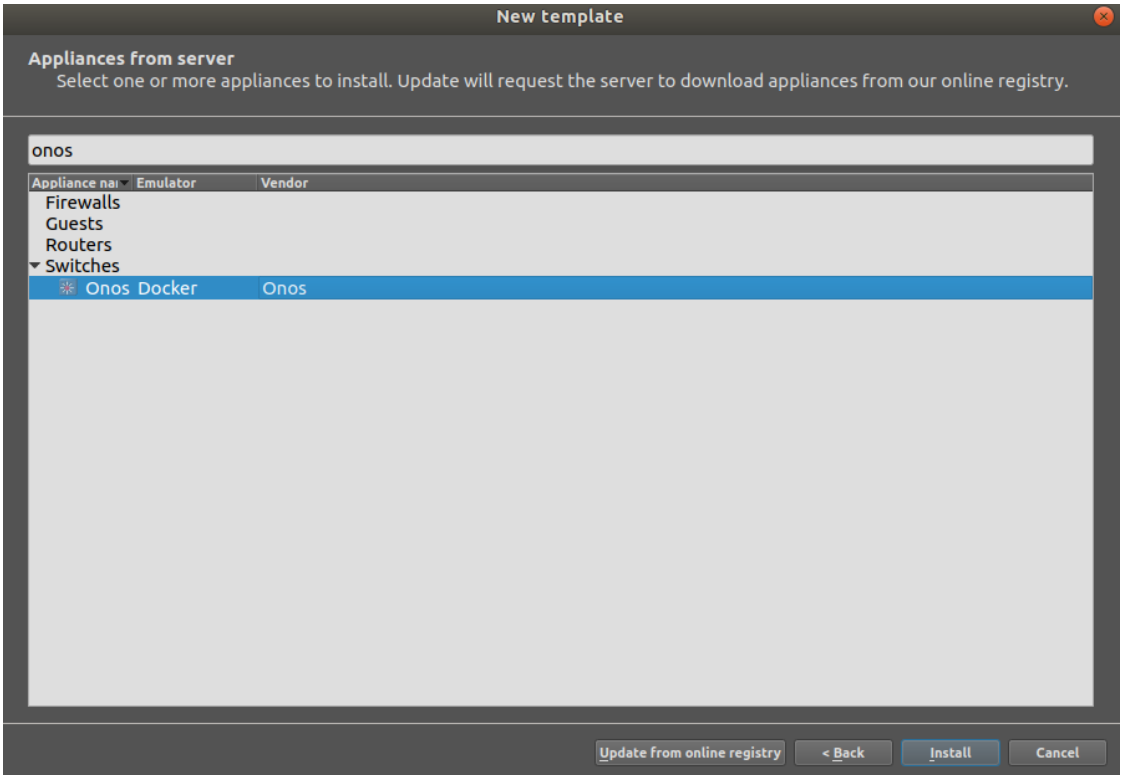
```
# Static config for eth0
auto eth0
iface eth0 inet static
        address 192.168.0.4
        netmask 255.255.255.0
```

OpenvSwitch3:

```
# Static config for eth0
auto eth0
iface eth0 inet static
        address 192.168.0.5
        netmask 255.255.255.0
```

2.2. ONOS controller (installation and configuration).

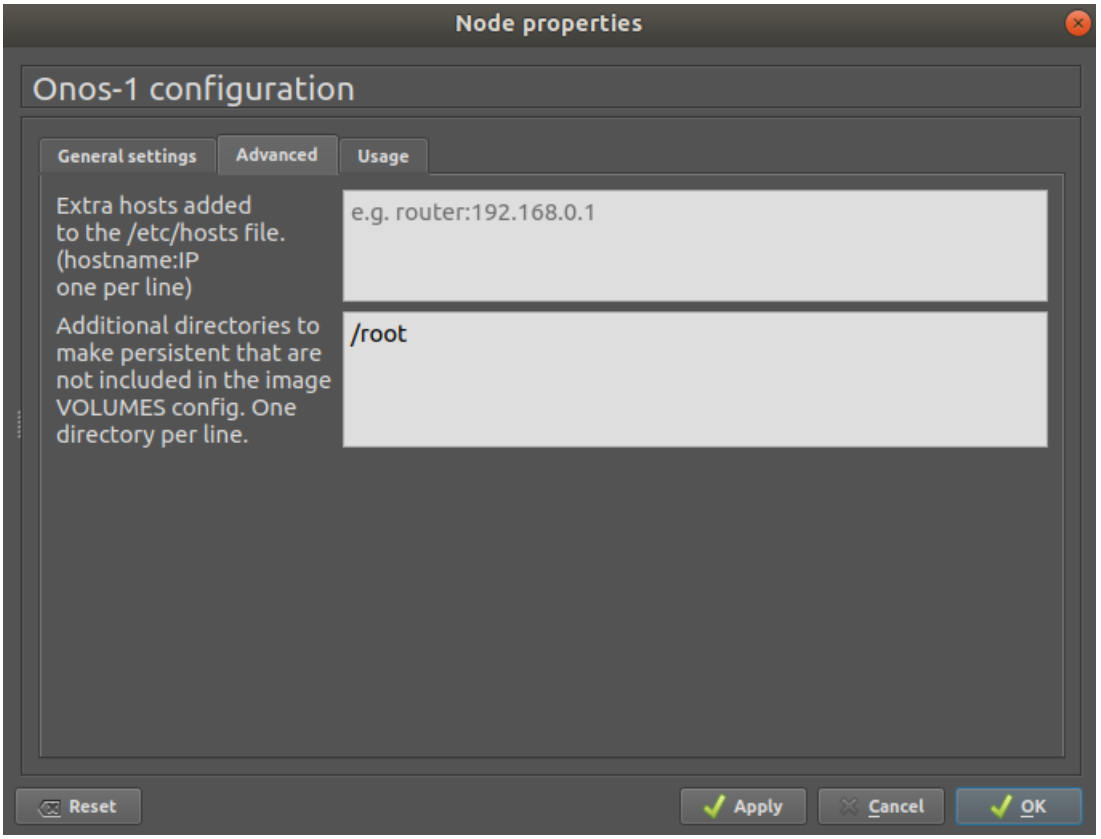
As in the previous case, we are going to incorporate a type of device from the GNS3 server appliance. We add a new template and search for “onos” in the appliance browser. Then we select “Onos” and install:



Now we incorporate it into the project with the name Onos-1 and continue with its configuration following the IP table mentioned above:

```
# Static config for eth0
auto eth0
iface eth0 inet static
        address 192.168.0.2
        netmask 255.255.255.0
```

To make the changes in Onos-1 persistent, we configure the /root directory in the device properties:



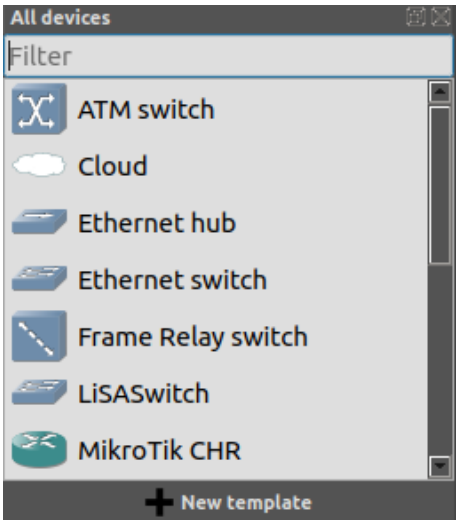
2.3. Incorporation of the remaining devices.

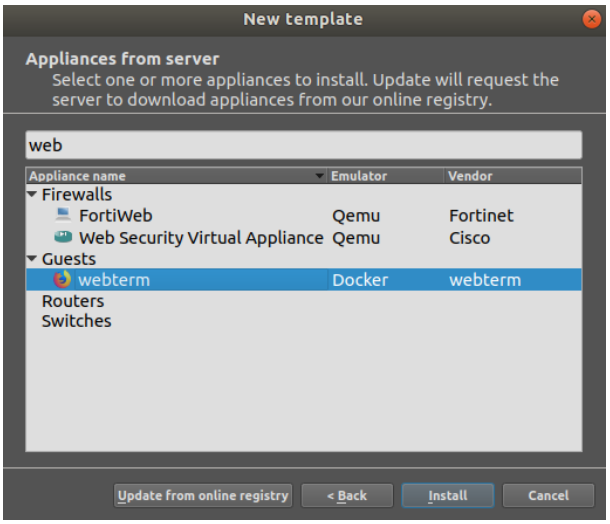
At this point we must add and configure the remaining devices:

- PC-1, PC-2, PC-3, PC-4 y PC-5
- webterm-1
- Switch

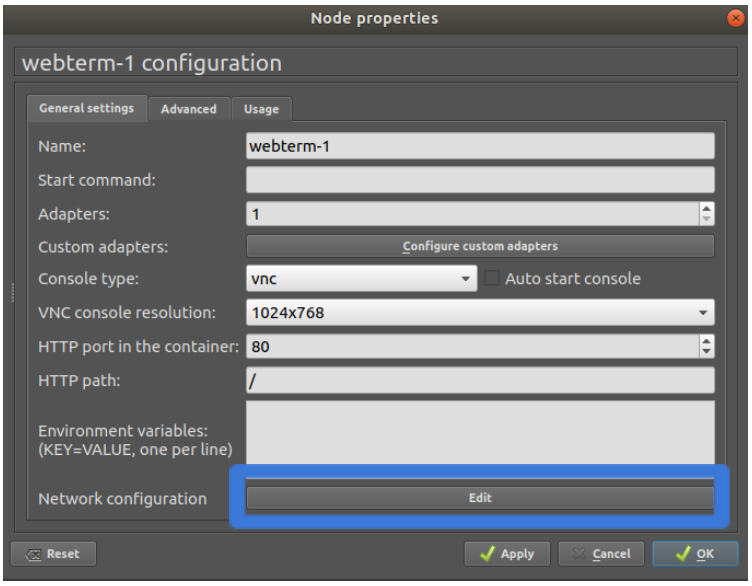
First of all, we add a switch (Switch1) connecting it to the Onos-1 controller through the e1 interface of the switch and the eth0 of the Onos at the moment without any configuration.

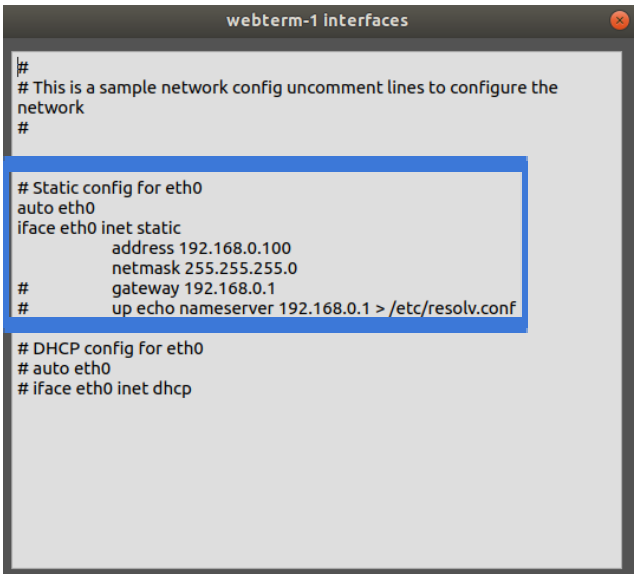
Then webterm-1 is incorporated by adding a new template, searching for webterm and installing the GNS3 server appliance:





Once installed and added, we connect it to switch1 through eth0. then we configure its ip (192.168.0.100) as follows:





It should be noted that before using the webterm, a VNC client with X11 support must be installed, since webterm deploys a light Linux image with a graphical environment in Docker. To install it, run the following command:

```
$ sudo apt-get install xtightvncviewer
```

Finally, the 5 PCs are added from the template, they are connected and configured with their corresponding IP:

- PC-1 connects to switch1 through eth0 and its IP is configured. To do this, modify the /etc/network/interfaces file on eth0:

```
auto eth0
iface eth0 inet static
        address 192.168.0.101
        netmask 255.255.255.0
        gateway 192.168.0.1
```

With this we get the following output from the **ifconfig** command:

```
root@Debian:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 0c:76:8f:ab:00:00
          inet addr:192.168.0.101  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::e76:8fff:feab:0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:69 errors:19 dropped:0 overruns:0 frame:19
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4650 (4.5 KiB)  TX bytes:888 (888.0 B)
```

- PC-2 connects to the eth1 interface of Open vSwitch1 through eth0. And your ifconfig command output is as follows:

```
root@Debian:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 0c:c5:fc:82:00:00
          inet addr:192.168.1.10  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::ec5:fcff:fe82:0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1595 errors:0 dropped:1588 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:216458 (211.3 KiB)  TX bytes:888 (888.0 B)
```

- PC-3 connects to the eth2 interface of Open vSwitch1 through eth0. And its configuration is as follows:

```
root@Debian:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 0c:8f:7b:9d:00:00
          inet addr:192.168.1.11  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::e8f:7bff:fe9d:0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1652 errors:0 dropped:1645 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:224210 (218.9 KiB)  TX bytes:888 (888.0 B)
```

- PC-4 connects to the eth1 interface of Open vSwitch2 through eth0. And its configuration is as follows:

```
root@Debian:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 0c:3e:08:38:00:00
          inet addr:192.168.1.12  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::e3e:8ff:fe38:0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:818 (818.0 B)
```

- PC-5 connects to the eth2 interface of Open vSwitch2 through eth0. And your ifconfig command output is as follows:

```
root@Debian:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 0c:14:58:a8:00:00
          inet addr:192.168.1.13  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::e14:58ff:fea8:0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:818 (818.0 B)
```

Once all the devices are connected and configured, we turn them on.

2.4. OpenFlow switches configuration.

Before starting with the configuration of the Open vSwitches, access to the ONOS controller management must be prepared (command line (CLI) or through a graphical interface (GUI) from a browser).

For the CLI, a connection is made from PC-1 via ssh to port 8101 of the ONOS controller (user: karaf and password: karaf) with the following command

```
$ ssh -v -p 8101 karaf@192.168.0.2
```

Once connected we activate the OpenFlow protocol:

```
# app activate org.onosproject.openflow
```

In the case of the graphic environment we use the webterm-1 and using the browser we look for **http://192.168.0.2:8181/onos/ui/index.html** (user: onos and password: rocks)

Once this is done we can start with the configuration of the switches. In each of them, a series of bridges are created by default that we must eliminate with the following commands:

```
# ovs-vsctl show                    //To show bridges

# ovs-vsctl del-br br0              //To remove the bridges
# ovs-vsctl del-br br1
# ovs-vsctl del-br br2
# ovs-vsctl del-br br3
```

Once the bridges have been eliminated, we start with the configuration. A bridge (br0) is created on each of the Open vSwitch with the command:

```
# ovs-vsctl add-br br0
```

Then the ports that are going to be used in the bridge that has been created previously are added with the following command:

```
# ovs-vsctl add-port br0 eth1                    //To add eth1 to br0
```

OpenvSwitch2:

```
/ # ovs-vsctl show
ea2beba3-c97b-44c8-98f5-b781e0ac3455
    Bridge "br0"
        Controller "tcp:192.168.0.2:6633"
            is_connected: true
        fail_mode: secure
        Port "eth4"
            Interface "eth4"
        Port "eth1"
            Interface "eth1"
        Port "br0"
            Interface "br0"
                type: internal
        Port "eth2"
            Interface "eth2"
        Port "eth3"
            Interface "eth3"
```

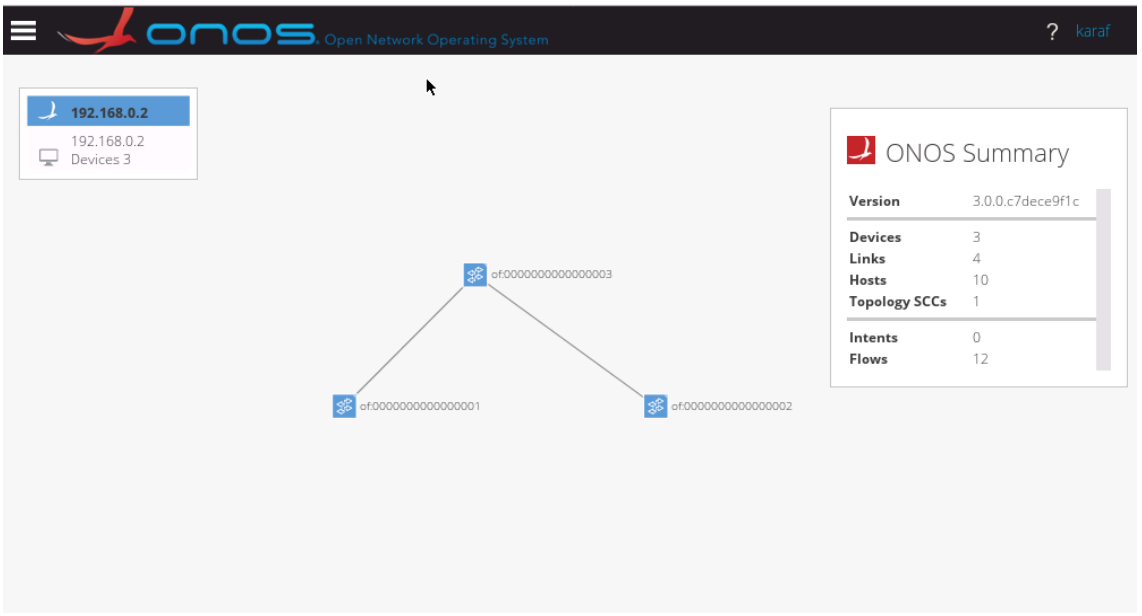
OpenvSwitch1:

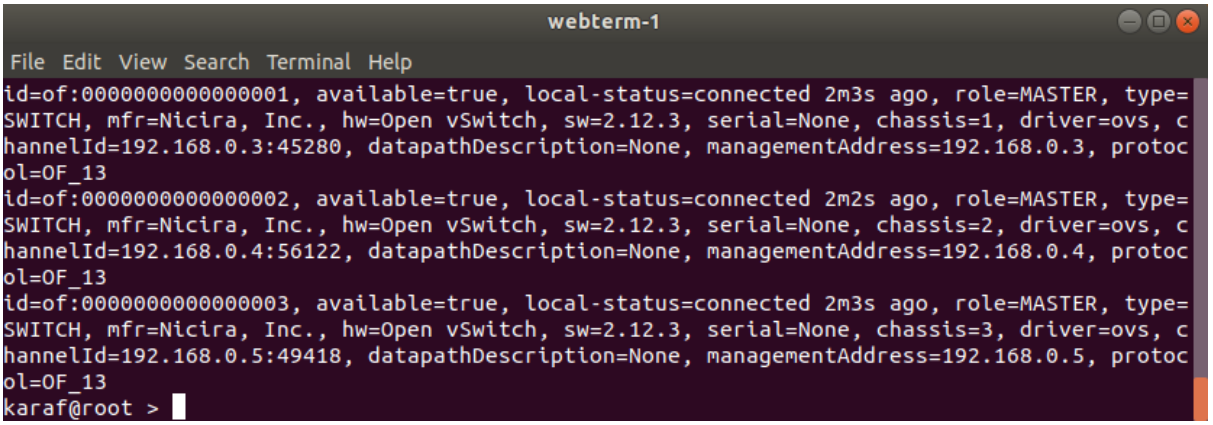
```
/ # ovs-vsctl show
fb147dbb-747f-4e11-8bb7-17dee461795c
    Bridge "br0"
        Controller "tcp:192.168.0.2:6633"
            is_connected: true
        fail_mode: secure
        Port "br0"
            Interface "br0"
                type: internal
        Port "eth2"
            tag: 20
            Interface "eth2"
        Port "eth3"
            trunks: [10, 20, 30]
            Interface "eth3"
        Port "eth1"
            tag: 10
            Interface "eth1"
```

OpenvSwitch3:

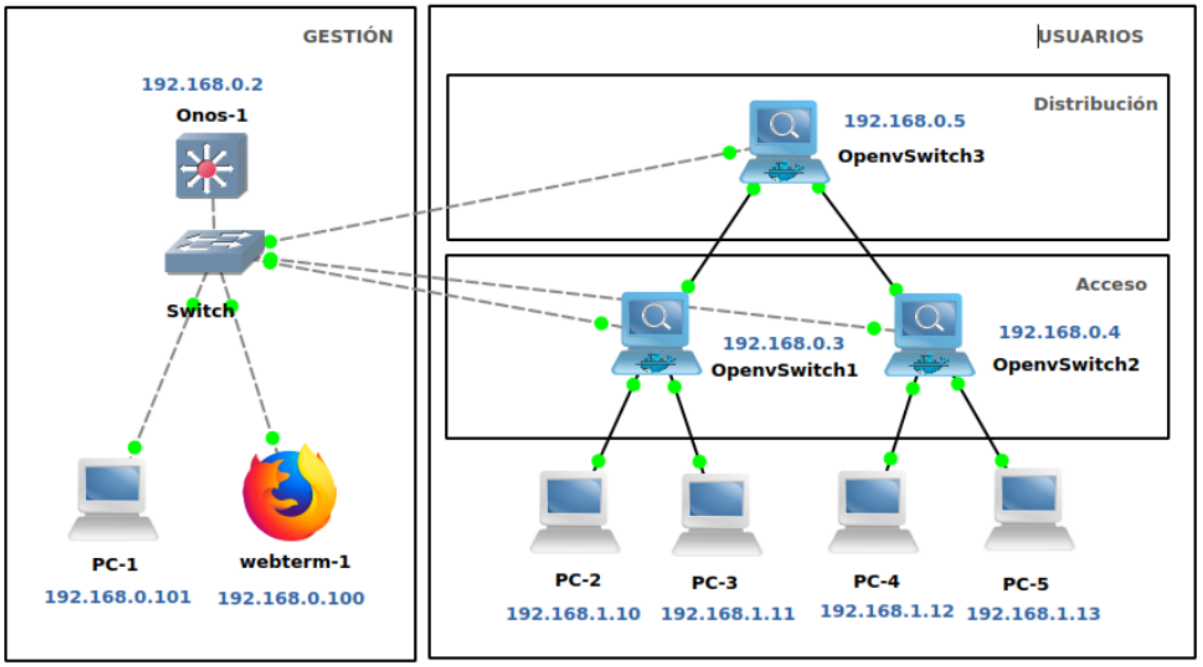
```
/ # ovs-vsctl show
d4b8305b-b1df-4709-809b-e70271f56f5e
    Bridge "br0"
        Controller "tcp:192.168.0.2:6633"
            is_connected: true
        fail_mode: secure
        Port "eth2"
            trunks: [10, 20, 30]
            Interface "eth2"
        Port "eth1"
            trunks: [10, 20, 30]
            Interface "eth1"
        Port "eth3"
            trunks: [10, 20, 30]
            Interface "eth3"
        Port "br0"
            Interface "br0"
                type: internal
```

Finally, we check the controller information through the graphical interface and we also obtain information about the connected devices through the command line:





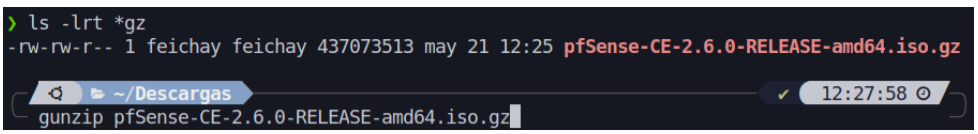
At the end of the configuration we have the following topology:



2.5. pfSense for GNS3.

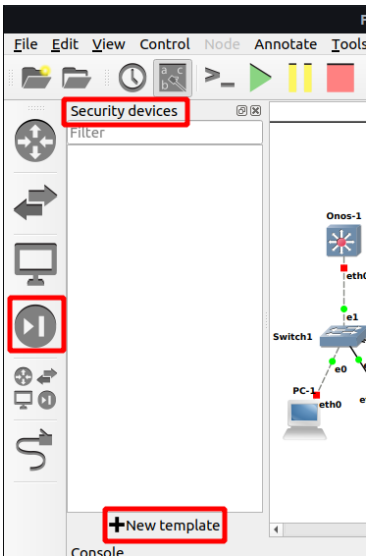
2.5.1. Add the GNS3 device template for a pfSense firewall.

First of all, let's create the device template for a pfSense firewall in GNS3. As a first step we will download the ISO, of the latest version available, of the firewall from its download page: https://www.pfsense.org/download/. We select the AMD architecture and ISO installation.
By default, the ISO file will be downloaded compressed, so we must decompress it previously, we will do it with the `gunzip` command:

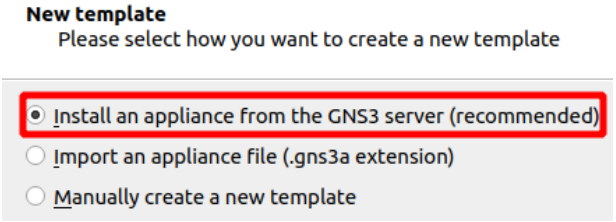


Next, we will download the image file of an empty virtual hard disk of 100GB (it is a template), for a device in GNS3 to be deployed with QEMU (in qcow2 format) from the following download page:
https://sourceforge.net/projects/gns-3/files/Empty%20Qemu%20disk/empty100G.qcow2/download

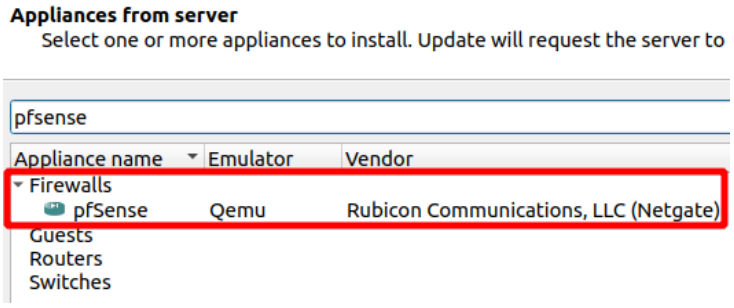
Now we are going to add the device template, for this, we go to the security devices tab in GNS3 and click on *new template*:



Next, we indicate that we want to perform the installation from an *appliance from the GNS3 server:*

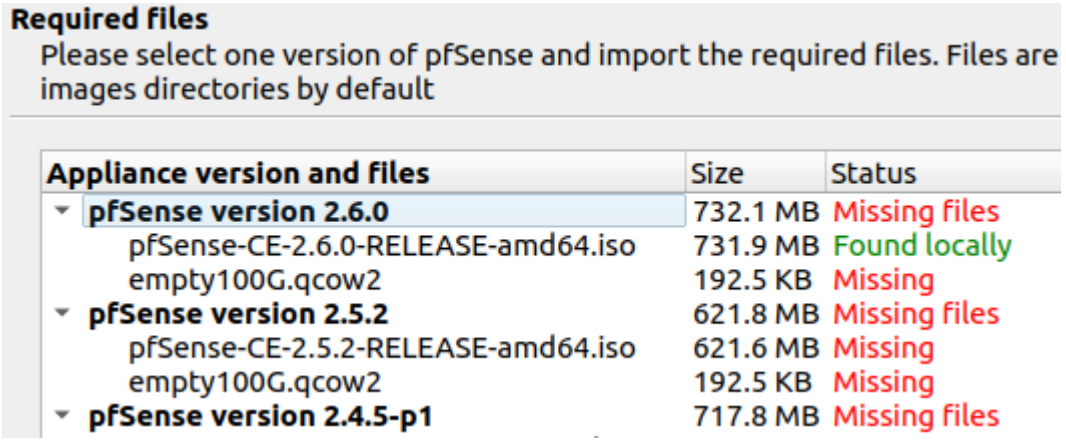


In the next screen, in the appliance search we look for the pfSense and then we select the template "*pfSense*":



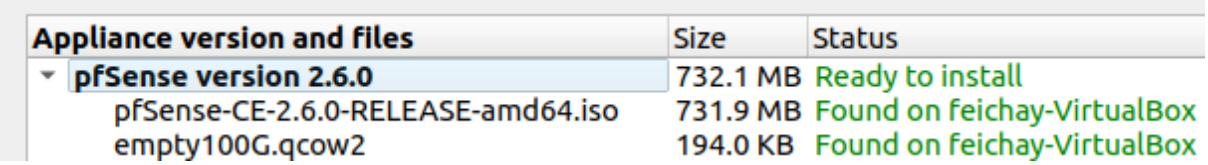
In the next two screens, we select the deployment on the local computer and use the Qemu binary of the latest 64-bit version. Once we reach the screen of files required for installation, let's go to create a new version, clicking on the corresponding button, and renaming it with

the version that we have previously downloaded, but if the latest version that we have downloaded appears, it is not necessary to create anything:



Once the new version is created, we will see that some of the necessary files (the installation ISO image and the virtual hard disk image) they are not located, so for each of the files that appears as "*Missing*", we will click on the *Import* button, and we will search the file that we previously downloaded.

When we have already imported all the necessary files, we will see that the associated version is available to install, so we can select it and click next:



Next, we finish the installation and we will see the available device template.

2.5.2. Add and install the pfSense firewall in the GNS3 project.

In the project, we will add a device of type pfSense and another of type Cloud and we will connect them with the rest of the already existing elements as indicated in figure 2.

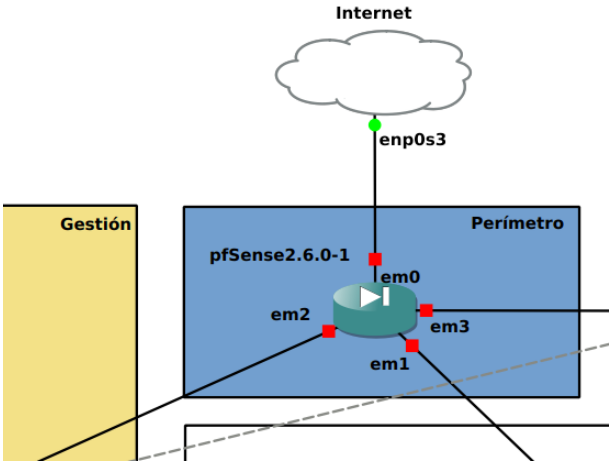
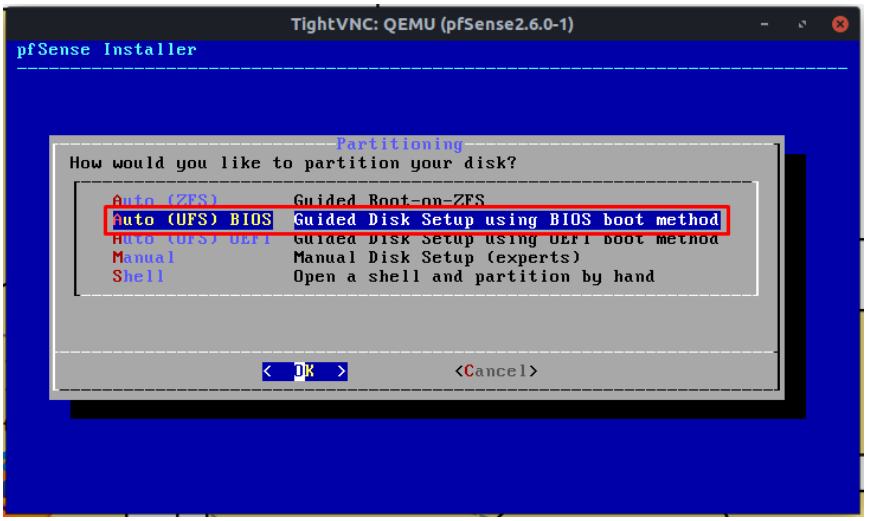


Figure 2. Initial Firewall assembly

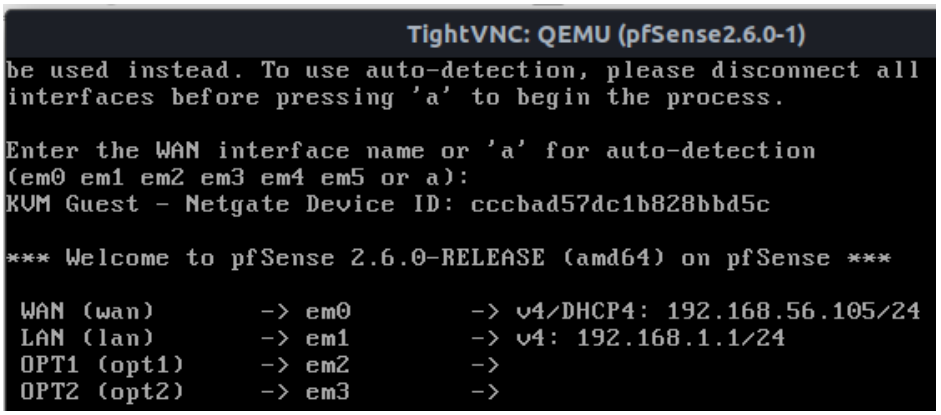
Once the devices are added and connected, we will boot the firewall and open the pfSense console to see the progress of its initial boot. When the initial boot is finished, we will arrive at the first screen of the installation wizard, where we will indicate that we accept the copyright and distribution rights notice of the pfSense software. Then we select the *Install* option.
We continue with the default keyboard layout, and select the configuration option and automatic partitioning of disks compatible with BIOS (second option).



Next, the installation process will start and we will have to wait for it to finish.

2.5.3. Basic pfSense firewall network configuration.

Until now we only have assigned and configured the interface em0 (WAN) and em1 (LAN), let's now add, from the console, the two additional interfaces that we are going to need: em2 (MGMT) and em3 (DMZ). To do this, in the initial menu, we choose the "1 *Assign Interfaces*". In "*should VLANs be set up now"* we press "*n*". Next, the installation process will start and we will have to wait for it to finish. Once the four interfaces are configured, we stop the configuration process by pressing the Enter key without having typed anything.



Now we are going to configure the IP addressing of the interfaces:

- em0 (WAN): we are going to assign it by DHCP.
- em1 (LAN): we are going to assign the IP 192.168.1.1/24.
- em2 (MGMT): we are going to assign the IP 192.168.0.1/24
- em3 (DMZ): we leave it for later.

To configure the IP addressing through the console, we select option 2 "*Set Interface(s) IP address*". We select option 3, to configure the interface em2 (MGMT), and we put the IP address and the mask in CIDR notation:

```
Enter an option: 2

Available interfaces:

1 - WAN (em0 - dhcp, dhcp6)
2 - LAN (em1 - static)
3 - OPT1 (em2)
4 - OPT2 (em3)

Enter the number of the interface you wish to configure: 3

Enter the new OPT1 IPv4 address.  Press <ENTER> for none:
> 192.168.0.1

Subnet masks are entered as bit counts (as in CIDR notation) in pfSense.
e.g. 255.255.255.0 = 24
     255.255.0.0   = 16
     255.0.0.0     = 8

Enter the new OPT1 IPv4 subnet bit count (1 to 32):
> 24
```

We configure it as an internal interface, for the moment we do not activate a DHCP server on this interface and we mark that HTTP access to the webConfigurator is possible

```
TightVNC: QEMU (pfSense2.6.0-1)
Enter the new OPT1 IPv4 subnet bit count (1 to 32):
> 24

For a WAN, enter the new OPT1 IPv4 upstream gateway address.
For a LAN, press <ENTER> for none:
>

Enter the new OPT1 IPv6 address.  Press <ENTER> for none:
>

Do you want to enable the DHCP server on OPT1? (y/n) n
Disabling IPv4 DHCPD...
Disabling IPv6 DHCPD...

Do you want to revert to HTTP as the webConfigurator protocol? (y/n) y

Please wait while the changes are saved to OPT1...
 Reloading filter...
 Reloading routing configuration...
 DHCPD...
 Restarting webConfigurator...

The IPv4 OPT1 address has been set to 192.168.0.1/24
```

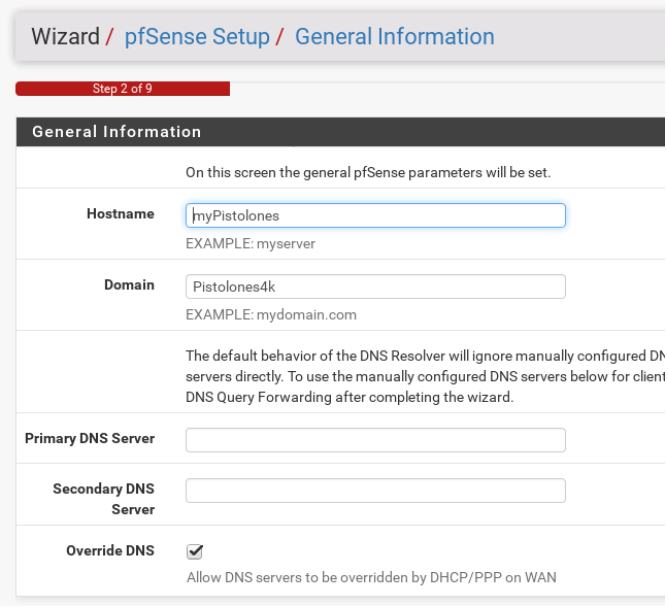
2.5.4. Access to pfSense firewall web management.

We activate access to web management from the em2 interface, we go to the pfSense console, we choose option 8 "*Shell*", and we launch the following command to set quick rules from the console:

```
# easyrule pass mgmt tcp any any 80
```

Next, we start webterm-1 and from your web browser we access the address of the em2 interface (OPT1 or MGMT) of the firewall, this is http://192.168.0.1, and we validate ourselves with the default credentials (username “admin” and password “pfsense”).

Now that we have access to the pfSense from the webterm-1 we are going to perform a small basic configuration through the graphical interface. When we access the web management interface for the first time, we will be shown an initial configuration wizard, we will be giving next on each screen, and we will leave most of the options in their default values, except for the following:



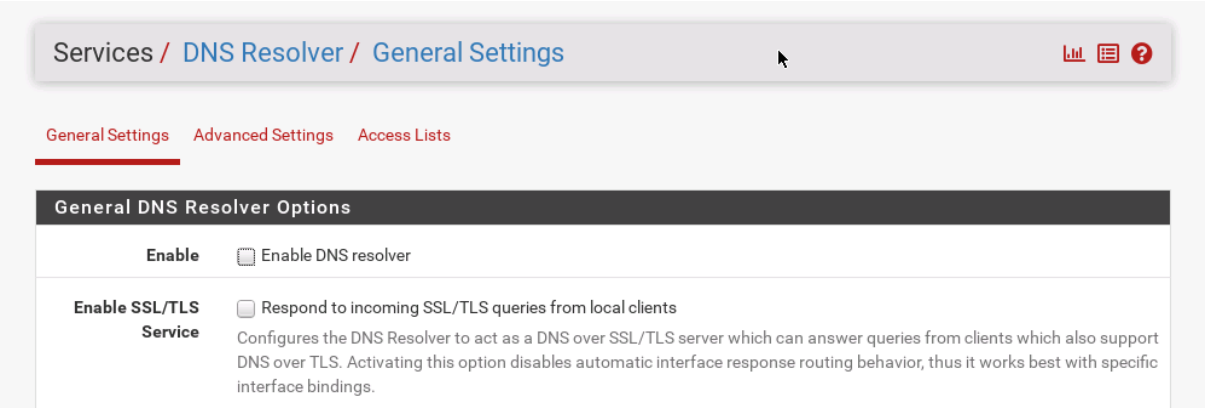
Finally, we change the password of the administrator (admin user) and reload the configuration.

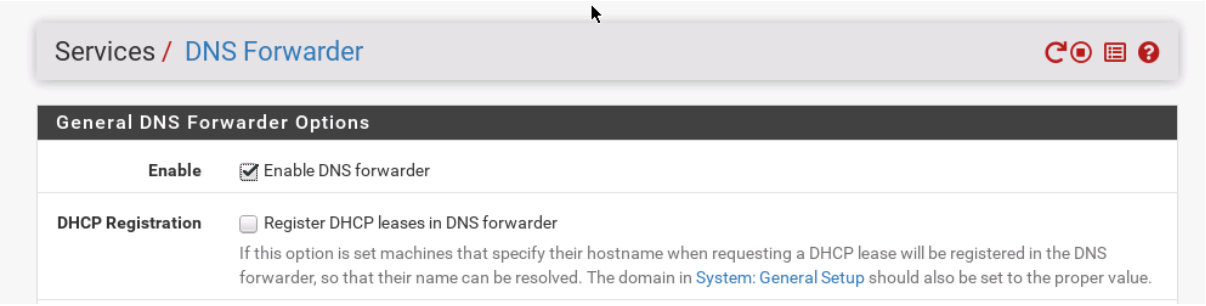
Next, we complete the configuration of the network interfaces, and for this we go to *Interfaces* → *Assignments*, we access the OPT1 and OPT2 interfaces and reconfigure their name and other parameters.
For OPT1, we will only change the name to MGMT and for OPT2 we’re going to rename it to DMZ and configure the IP address.

2.6. DNS service configuration.

The use of local DNS has been marked as first priority by default (internal computers must use the firewall interface on their network as the DNS server) and then the fallback (if it cannot be resolved locally) on the DNS servers obtained by the DHCP client on the WAN interface.

Then we need to disable DNS Resolver and enable DNS Forwarder to improve emulation performance.

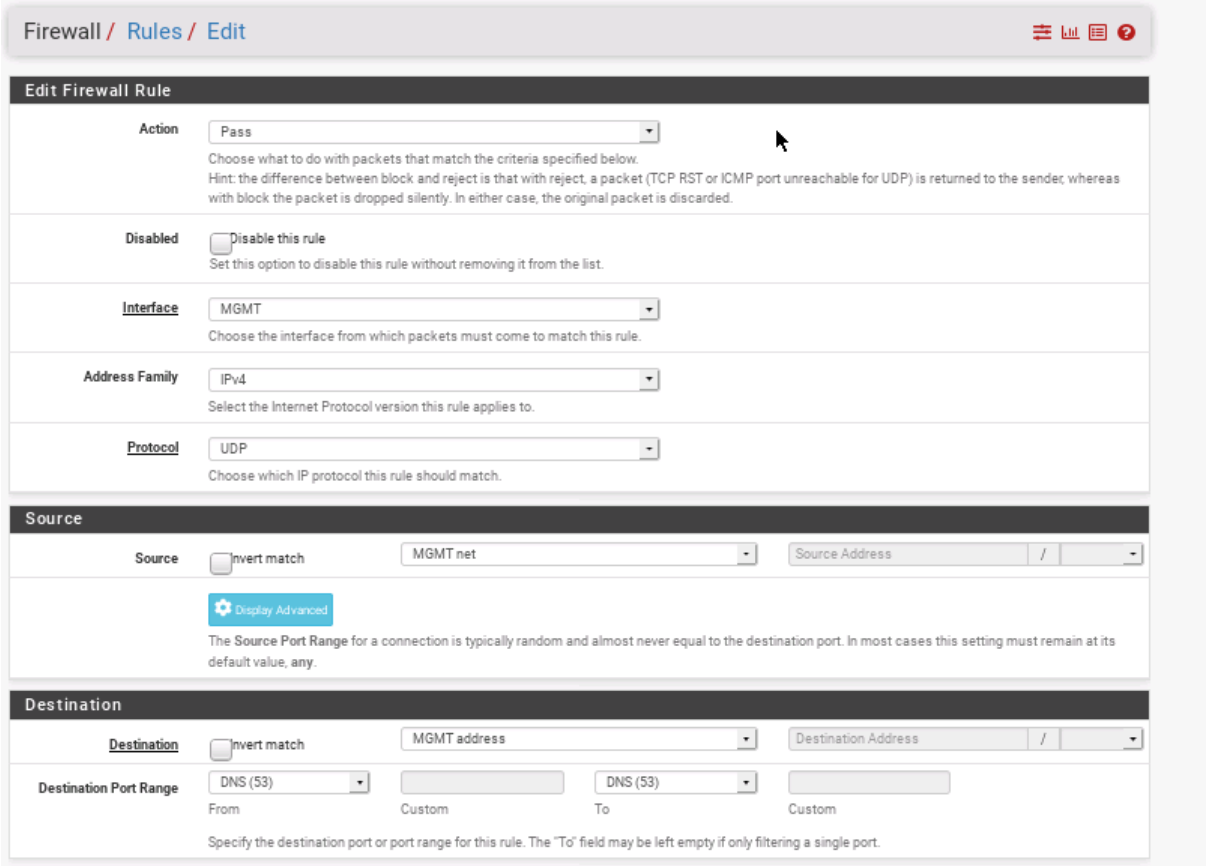




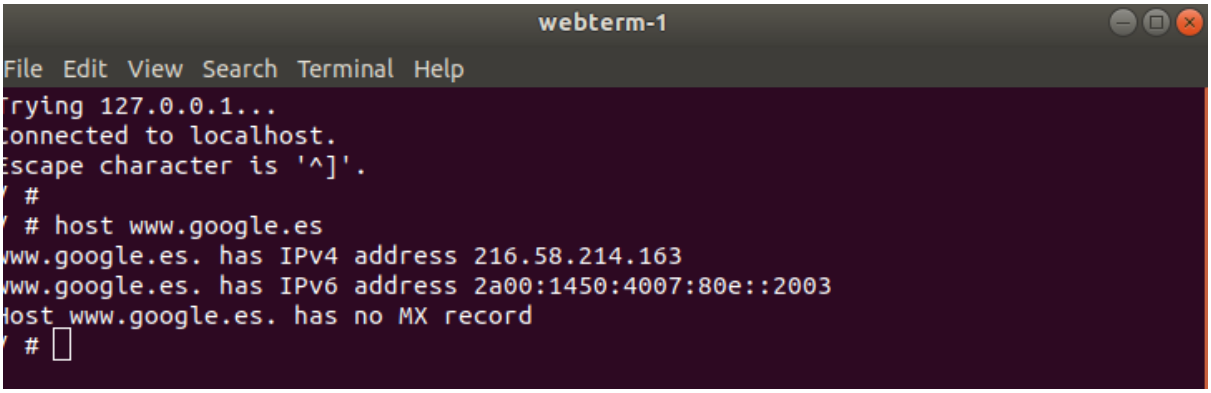
Now we are going to configure the DNS on webterm-1 against the firewall interface. To do this we modify the network configuration to indicate that the DNS server is the firewall interface on the MGMT network:

```
# Static config for eth0
auto eth0
iface eth0 inet static
        address 192.168.0.100
        netmask 255.255.255.0
        gateway 192.168.0.1
        up echo nameserver 192.168.0.1 > /etc/resolv.conf
```

In addition to that, for it to work properly we must enable the traffic in the firewall. To do this, we go to Firewall -> Rules -> MGMT and create a new rule that allows DNS traffic



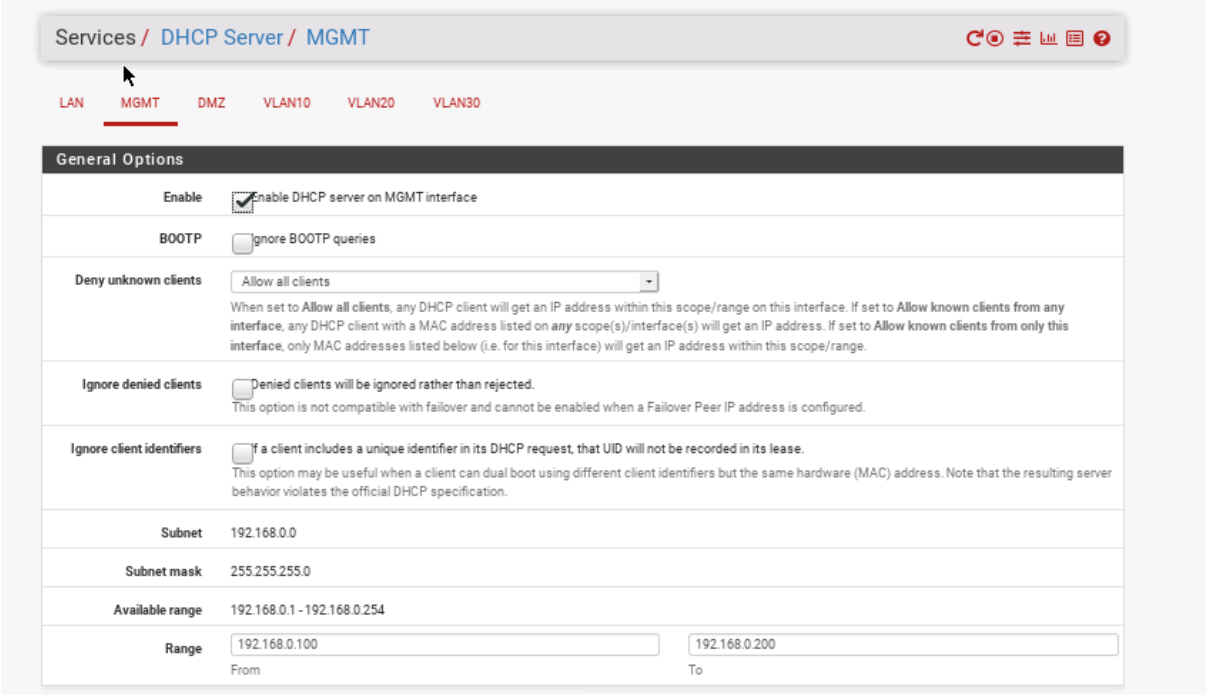
Finally, we check that it works with the host command in webterm-1:



2.7. DHCP service configuration.

For the project we are going to have three DHCP servers, one for each internal subnet and in each one we will have a different range of IP addresses defined. For the DHCP configuration we go to Services -> DHCP Server, and then in each interface we will do the following

First of all, we activate the DHCP server on each of the interfaces. Then we will indicate the range of IP addresses to serve:

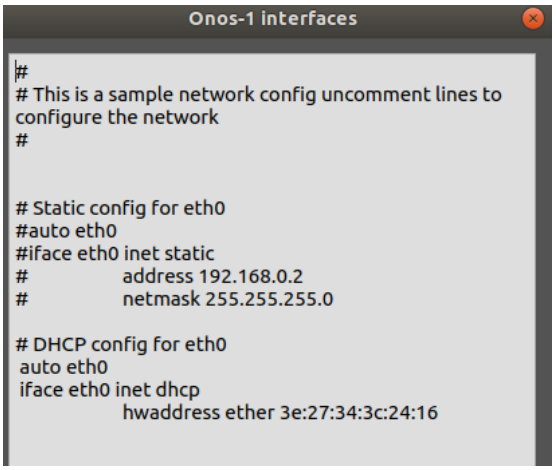


Then We modify the network configuration of webterm-1 so that it obtains its IP via DHCP:

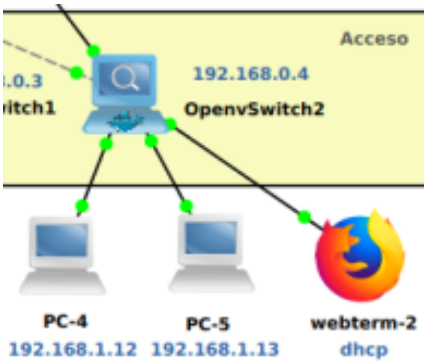
```
# Static config for eth0
#auto eth0
#iface eth0 inet static
#       address 192.168.0.100
#       netmask 255.255.255.0
#       gateway 192.168.0.1
#       up echo nameserver 192.168.0.1 > /etc/resolv.conf

# DHCP config for eth0
auto eth0
iface eth0 inet dhcp
```

Next we are going to make a static DHCP assignment to the Onos-1 controller with the address 192.168.0.2. For this we must know the MAC to be able to associate the IP to that MAC. MAC of Onos-1 → 3e:27:34:3c:24:16
The way to know Onos-1 MAC is through a packet capture in Wireshark. Once we know the MAC, we create the static assignment in DHCP based on that MAC.
We enter Services -> DHCP Server -> MGMT, add a static DHCP mapping, put the Onos-1 data, save and apply the changes:

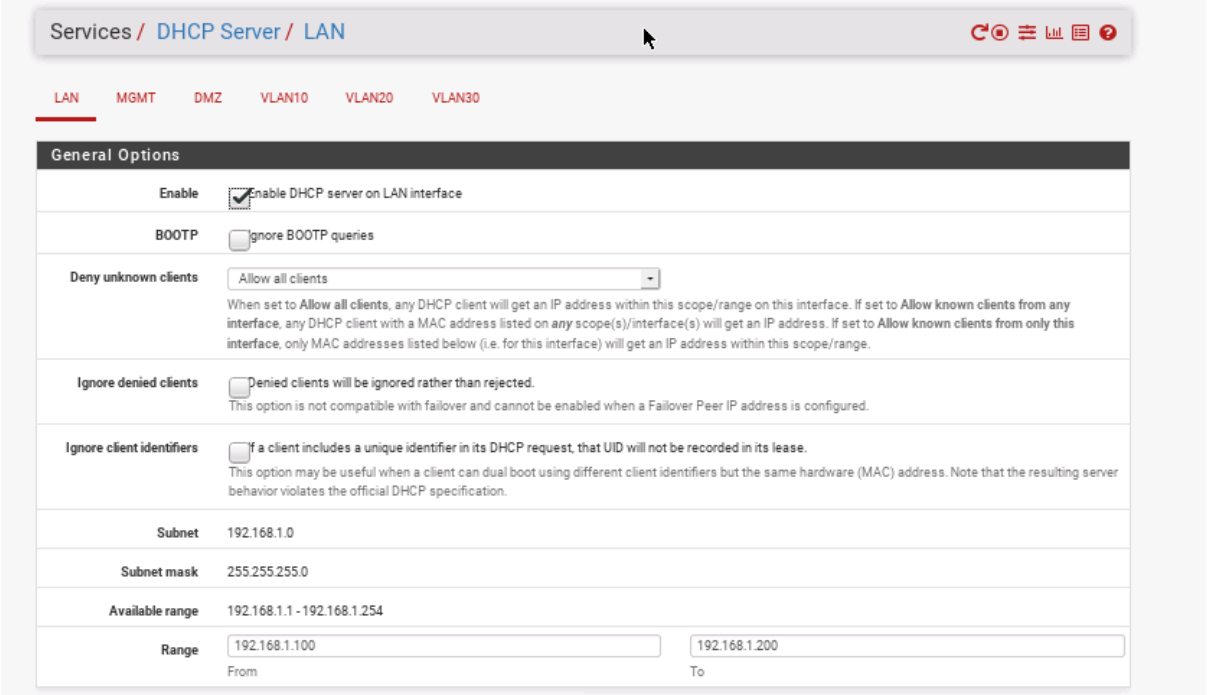
For this to work we must make sure that the controllers MAC address stays unchanged. This is done as follows:



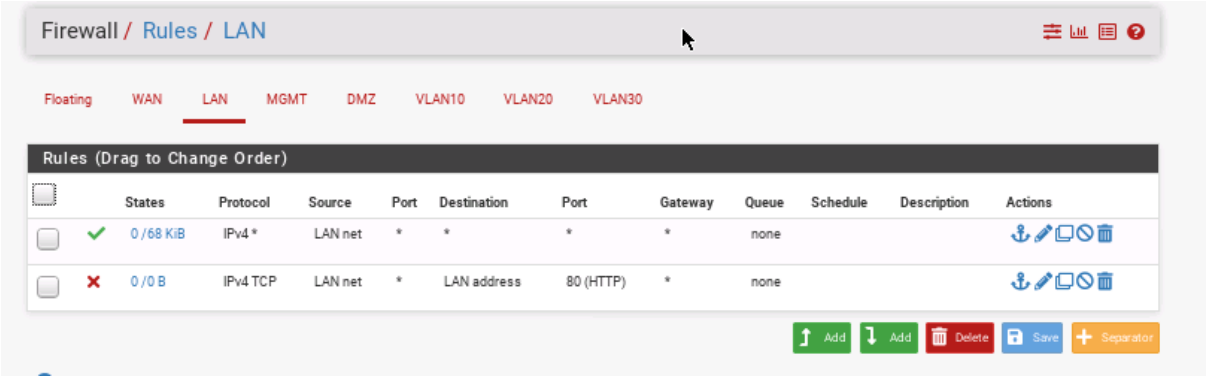
Finally, we are going to add the webterm-2 device to the user network connected to the eth4 OpenvSwitch2 interface.



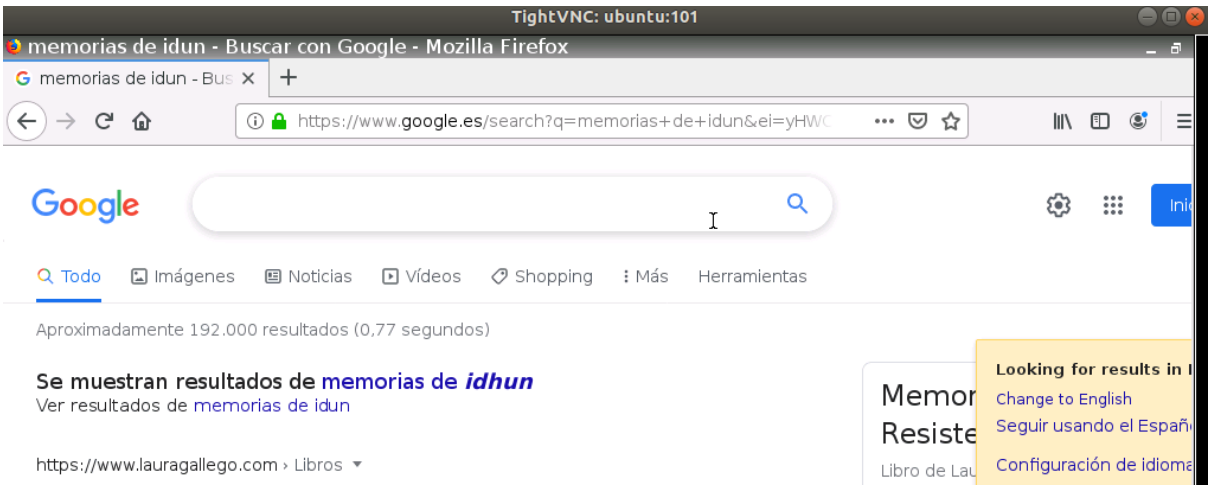
And we're going to configure DHCP to dynamically assign addresses to both PC-1 and webterm-2. To do this, activate the DHCP server for the LAN network (Users) with the following address range (192.168.1.100 - 192.168.1.200). The configuration is as follows:



Finally, we need to modify the rules for the LAN interface so that traffic to browse (first one) and DNS is enabled but connection to the console (second one) is restricted:



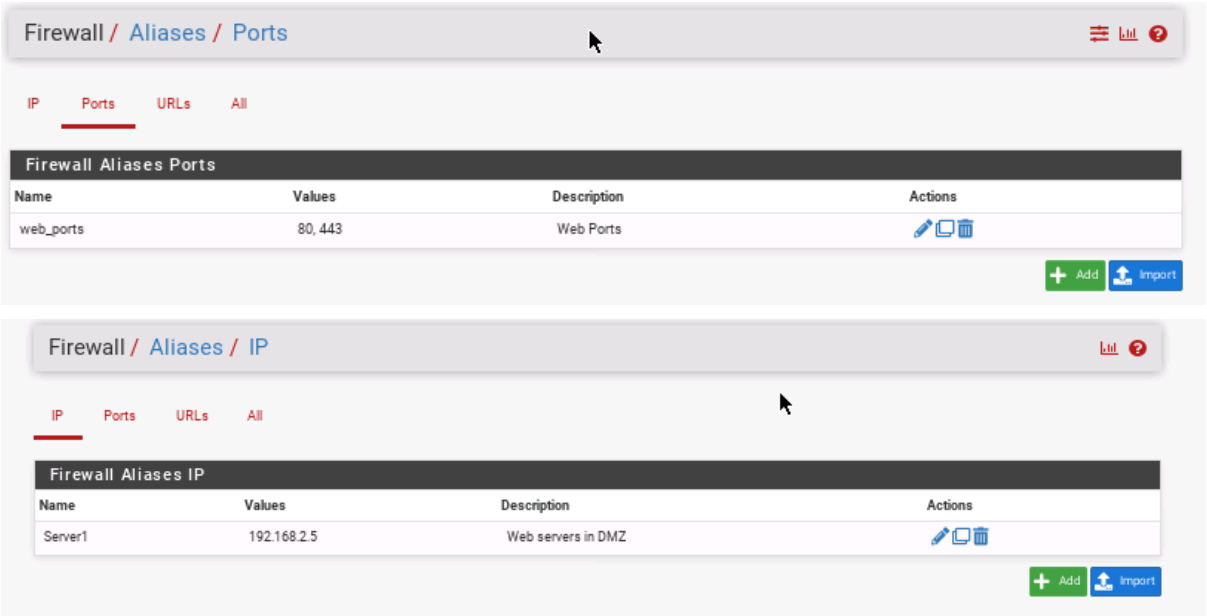
With these changes, we test that you do not have access to the firewall management but you can navigate in webterm-2:



2.8. NAT Port Forward configuration.

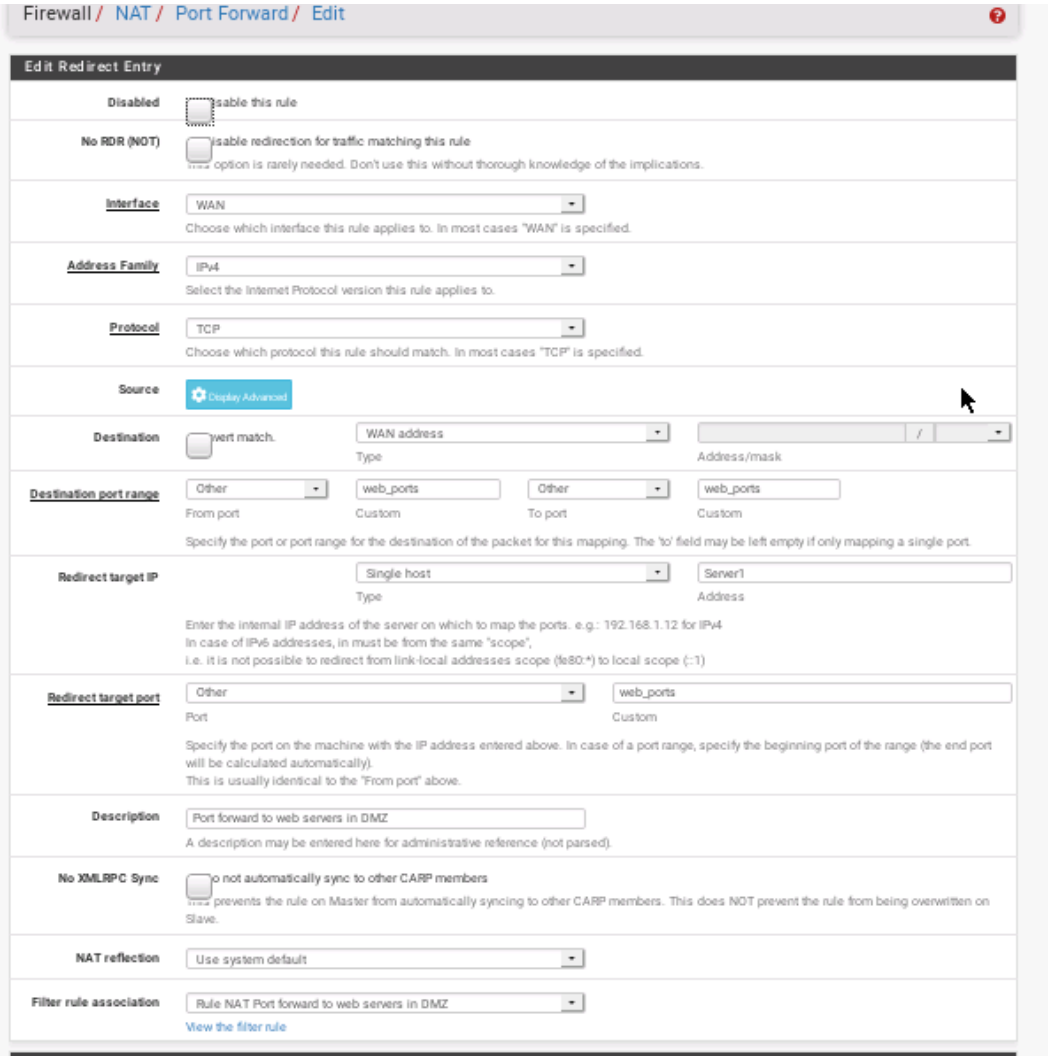
At this point, the necessary configuration is applied for the service publication from our network (from the DMZ) to the outside using NAT Port Forward.

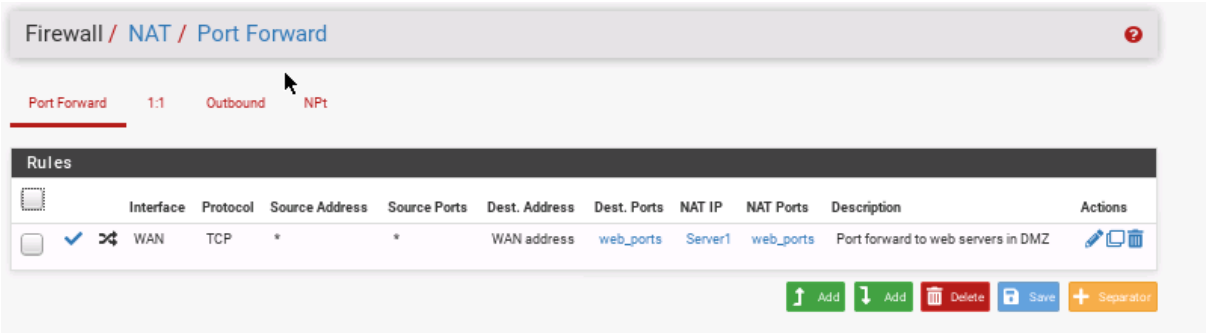
For this we are going to introduce the use of aliases, we create a group for all the web access ports that we are going to use, for which we are going to receive requests from the outside with the alias Web_Ports (Firewall -> Aliases → Ports). and a group of web servers from the DMZ, Server1 (Firewall -> Aliases → IP). Screenshots are attached:



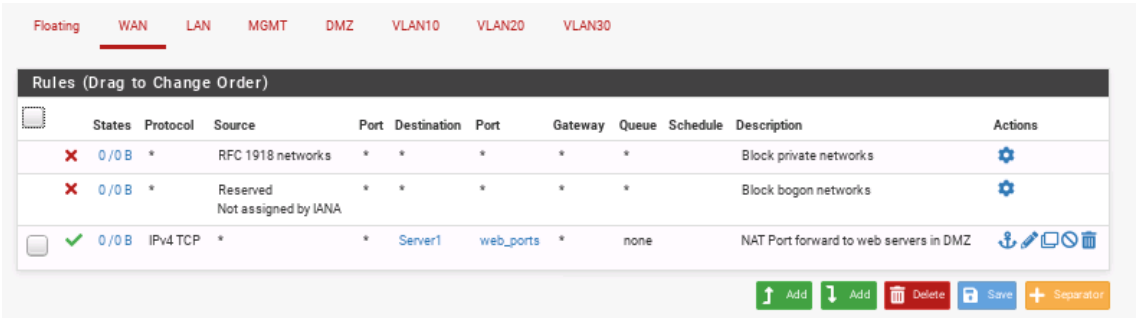
Now we must go to the menu Firewall -> NAT -> Port Forward and add a new entry by filling in the fields with the following information:

- Interface: WAN (external interface through which requests will arrive)
- Destination: WAN address
- Destination port range: Web_Ports
- Redirect destination IP: Singles host and Server1
- Redirect destination port: Web_Ports





With this, a new rule is automatically created in the access firewall for said traffic:



3. Improvements implemented.

3.1. Proposing a new addressing scheme for the network.

When implementing the improvements we thought it was convenient to change the addressing scheme proposed in the project script. The reason why this change has been made is mainly because when we work from home, our routers assign IPs from the range 192.168.1.10/24 - 192.168.1.150/24 by default. This caused us problems since in the initially proposed scheme there were IPs that overlapped with those mentioned. The changes in the proposed scheme are shown below, favoring the summarization of routes, since this facilitates our work when assigning rules in the firewall.

Red	Equipo	Interfaz	Dirección IP
-	pfSense 2.6.0-1	WAN (em0)	WAN DHCP
-	pfSense 2.6.0-1	LAN (em1)	192.168.4.1/24
-	pfSense 2.6.0-1	MGMT (em2)	192.168.3.1/24
-	pfSense 2.6.0-1	DMZ (em3)	192.168.2.1/24
WAN	webterm-3	eth0	WAN DHCP
MGMT	Onos-1	eth0	local DHCP MAC reservation 192.168.3.2/24
	webterm-1	eth0	local DHCP
	PC-1	eth0	local DHCP
DMZ	OpenvSwitch 4	-	192.168.3.6/24
	Server1	eth0	local DHCP MAC reservation 192.168.2.5/24
LAN	OpenvSwitch 1	-	192.168.3.3/24
	OpenvSwitch 2	-	192.168.3.4/24
	OpenvSwitch 3	-	192.168.3.5/24
	PC-2	eth0	192.168.5.10/24
	PC-3	eth0	192.168.6.11/24
	PC-4	eth0	192.168.6.12/24
	PC-5	eth0	192.168.7.13/24
	webterm-2	eth0	local DHCP

local_DHCP

- DMZ: 192.168.2.100 - 192.168.2.200
- MGMT: 192.168.3.100 - 192.168.3.200
- LAN: 192.168.4.100 - 192.168.4.200

3.2. Network segmentation (VLAN).

Different VLANs will be added to our network in order to segment it, so we logically organize it in a different way than physically.

In the first table you will find the different VLANs with their respective networks and their function. In the second we can find the addressing associated with the above.

VLAN	Name	Network Address	Description
10	Centrales	192.168.5.0/24	Central service staff
20	Oficina	192.168.6.0/24	Office staff
30	CPD	192.168.7.0/24	Data processing center equipment

Network	Hosts	VLAN	IP address	Description
Users	PC-2	10	192.168.5.10	PC on OvS1 central services
	PC-3	20	192.168.6.11	PC on OvS1 offices
	PC-4	20	192.168.6.12	PC on OvS1 offices
	PC-5	30	192.168.7.13	PC on OvS1 Data processing centre

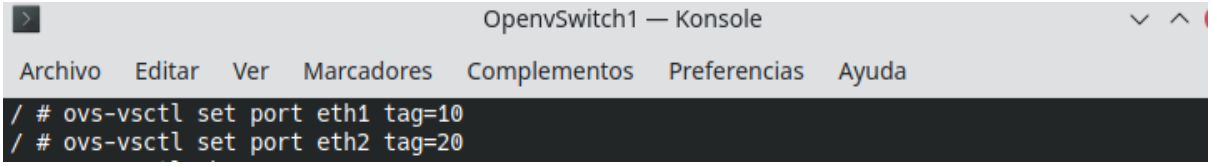
3.2.1. Switch configuration.

3.2.1.1. OpenvSwitch 1 & 2.

To begin, the switches must be configured by setting the following ports as access ports.

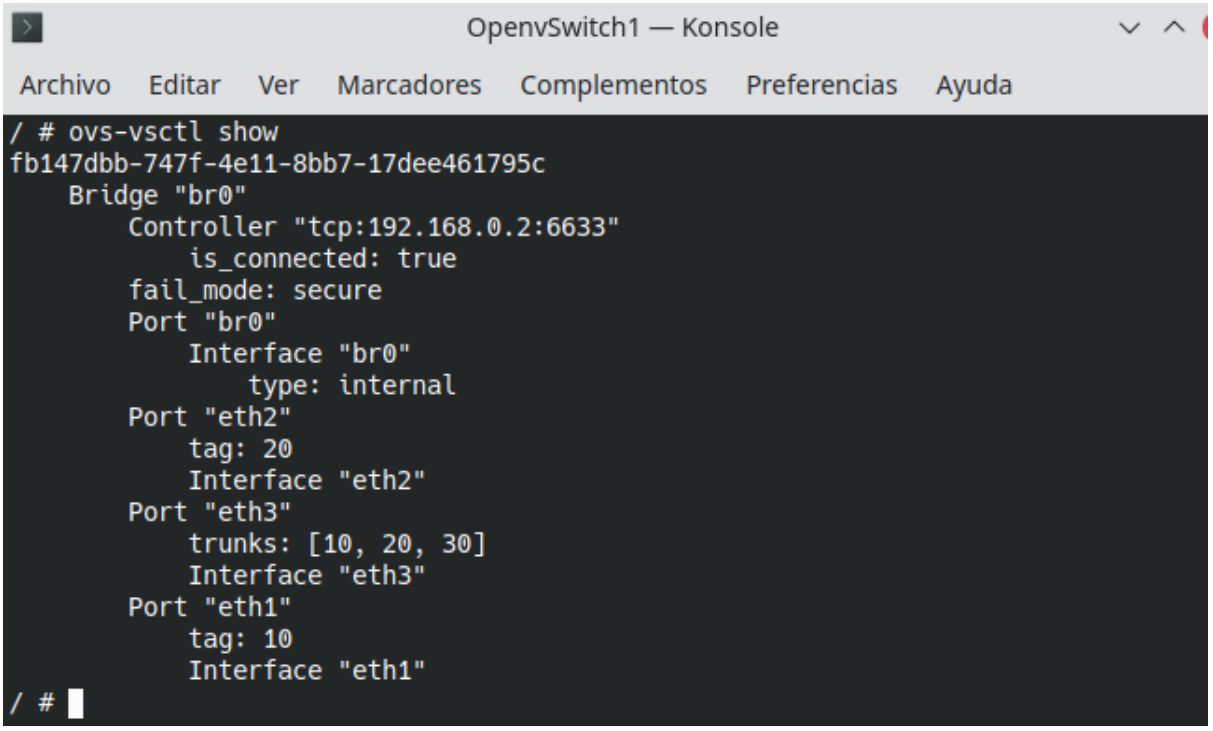
Set as access port, with packets tagged with vlan 10:

```
ovs-vsctl set port eth1 tag=10
```

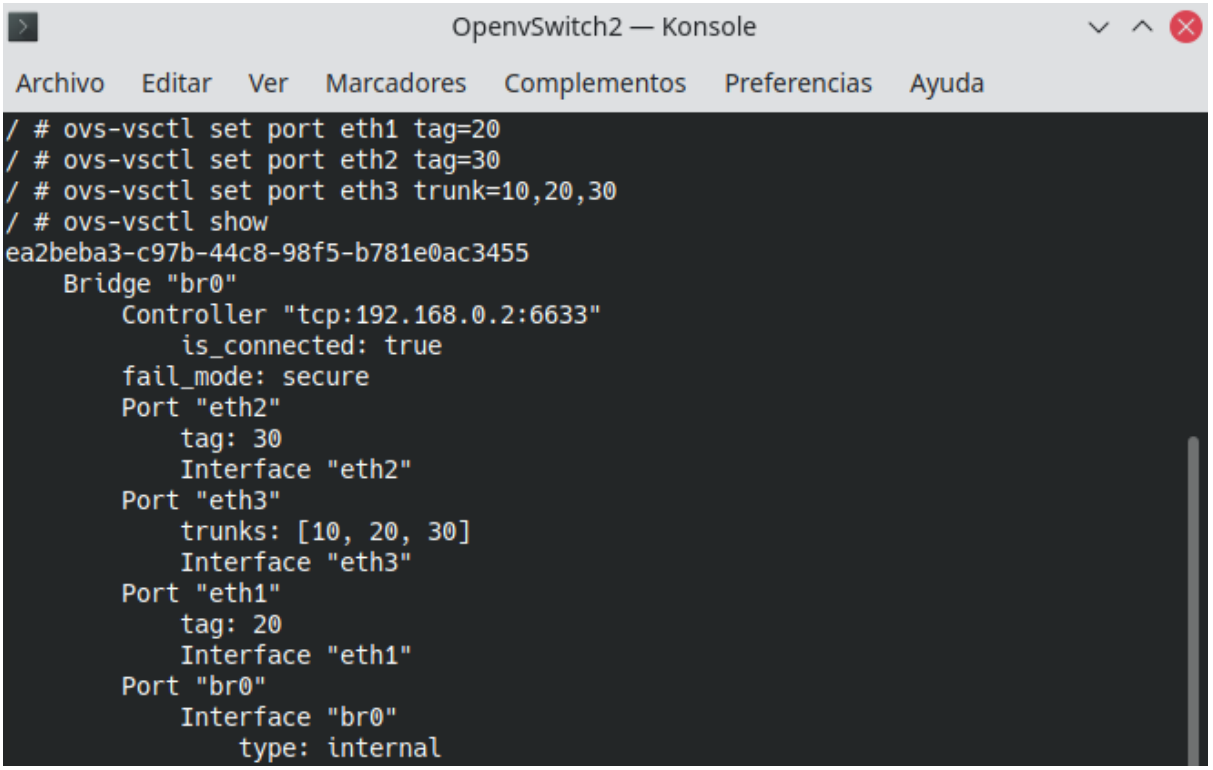


Set as trunk port, which accepts VLAN ID 10, 20 and 30:

```
ovs-vsctl set port eth3 trunk=10,20,30
```



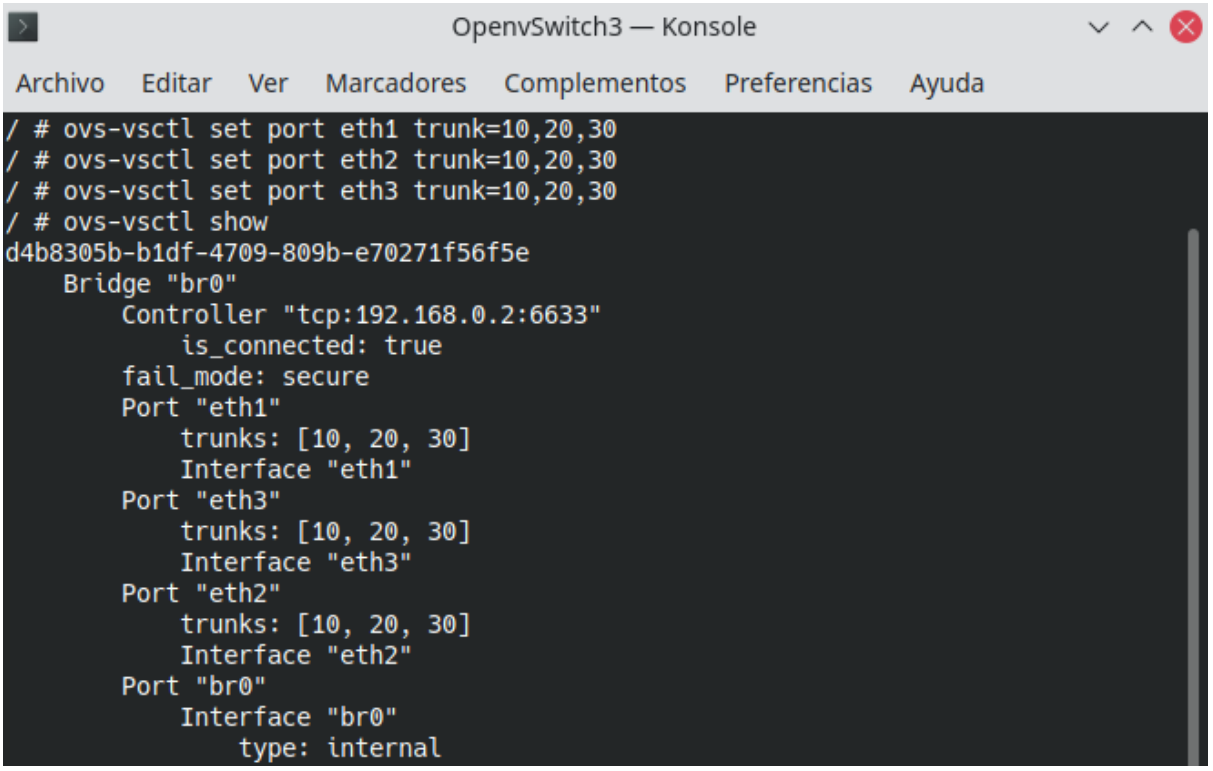
vSW2



3.2.1.2. OpenvSwitch 3.

El openvSwitch 3 tendrá como puertos trunk los puertos eth1, eth2 y eth3. Así que es repetir el comando anteriormente dado para hacer puertos trunk, con el siguiente resultado.

The openvSwitch 3 will have eth1, eth2 and eth3 as trunk ports. So all we have to do is repeat the command given above to make trunk ports, with the following result.



It is important to note that OpenvSwitches do not have any VLAN management. They only take care of checking if the packages are tagged or not. So the access ports will not be responsible for labeling the ports.

This adds a problem for us since it is necessary for the PCs to remove their tagged frames according to the corresponding VLAN, because right now their frames are being automatically dropped.

3.2.1.2. Making hosts send tagged traffic.

To begin with, we create a new interface to send tagged traffic through it:

```
vconfig add [INTERFACE] [VLAN]
```

We check that the 8021q driver is loaded to be able to work with VLANs:

```
lsmod | grep 8021q
```

To add the configuration of the new virtual interface we have to add a new entry in the file "/etc/network/interfaces" the network of the VLAN to which it belongs (detailed in the table above), its mask and which physical interface it comes from.

```
auto eth0.10
iface eth0.10 inet static
       address 192.168.5.10
       netmask 255.255.255.0
       vlan-raw-device eth0
```

Next, in order to bring up the virtual interface, it is necessary to use the ifconfig command, since the ifup and ifdown commands do not know how to properly read the given information.

It is necessary to turn off the physical interface eth0 and turn it back on for the virtual one to come out automatically. It is done with the following commands:

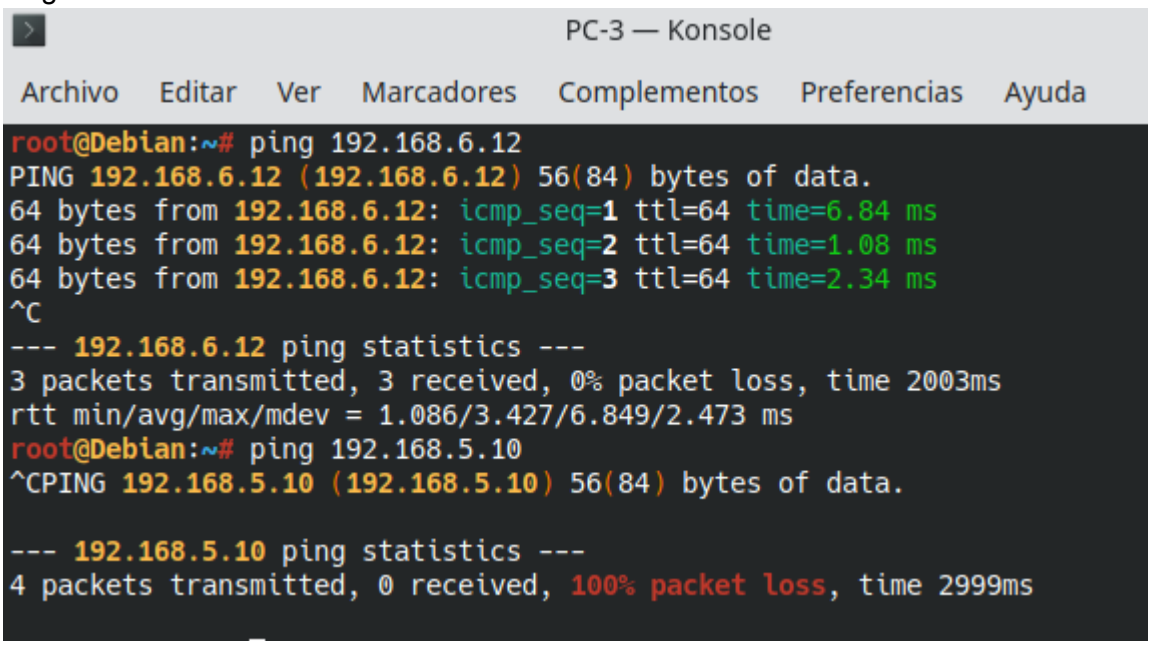
```
ifconfig eth0 down
```

```
ifconfig eth0 up
```

Another possibility is to restart the computers, since they have the option auto eth0.X, so that it starts automatically.

With this we have managed to get the PCs to send the tagged frames so that the VLANs are already working properly.

Ping from PC3 to PC4 & PC3 to PC2:

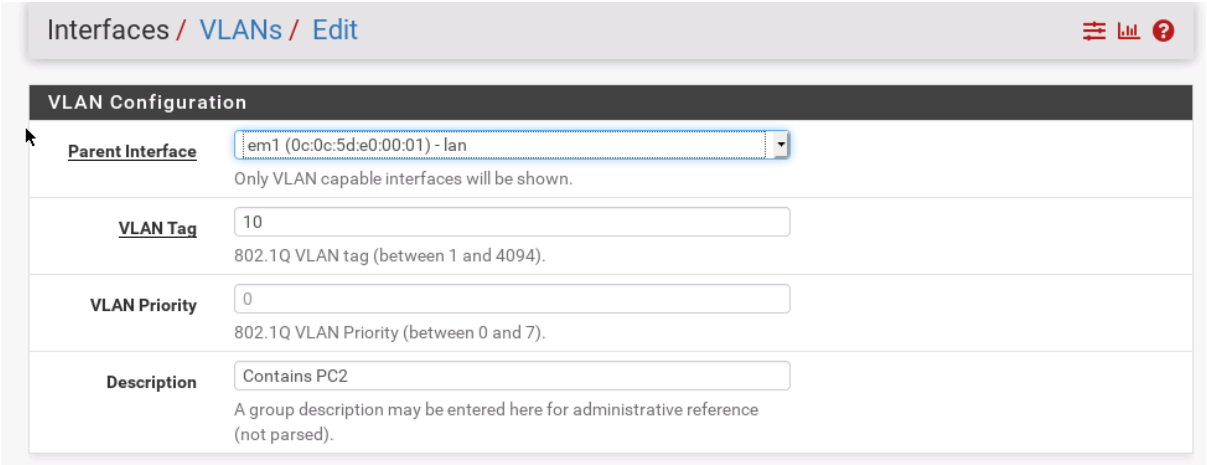


As can be seen in the previous image, there is connectivity between PCs in the same VLAN but not between different VLANs. This is because we have not configured a router that redirects our paquets to the corresponding VLAN, tagging them in the process so that the switches can work.

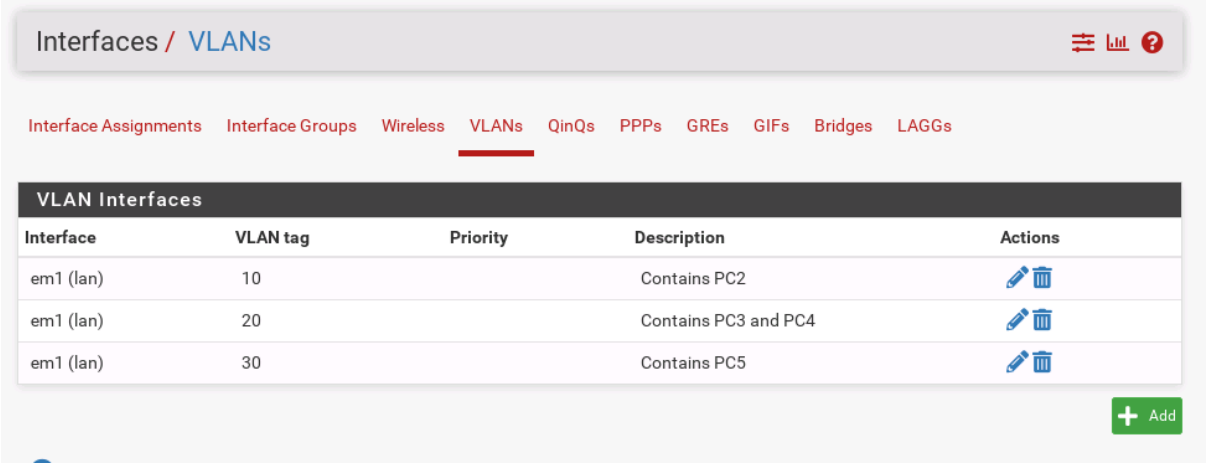
3.2.2. pfSense

The router we have at our disposal is the pfSense firewall router. Our goal is to create virtual interfaces in pfSense to be able to redirect and label the corresponding paquets.

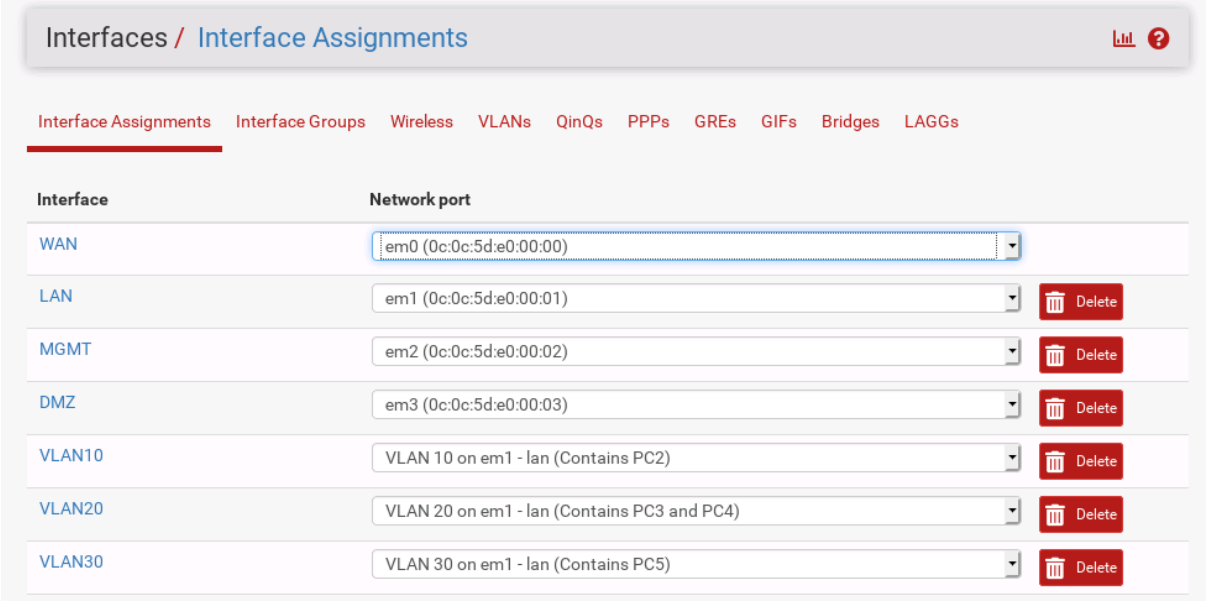
In its GUI, go to interfaces→assignments and then select the VLANs option.
If we click add, the following window appears, where we will have to establish which physical interface of the router we want to assign the virtual interface to, with its VLAN tag and a brief description.



The result of creating all the virtual interfaces would be the following:

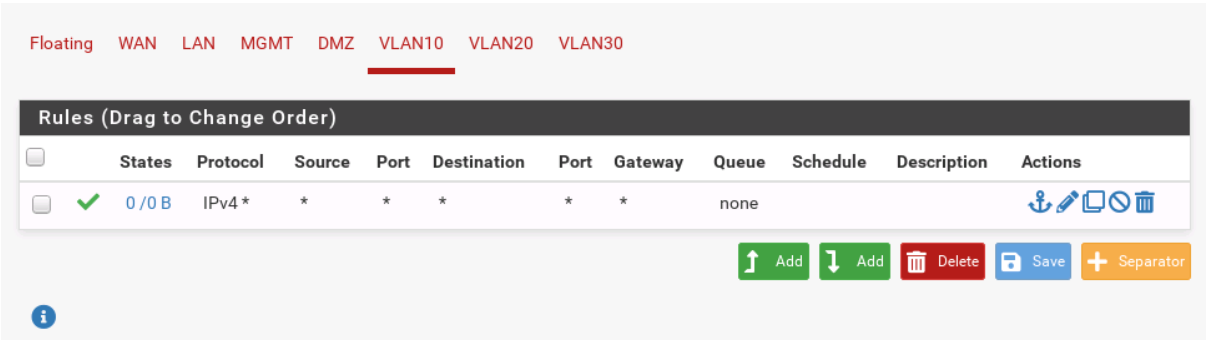


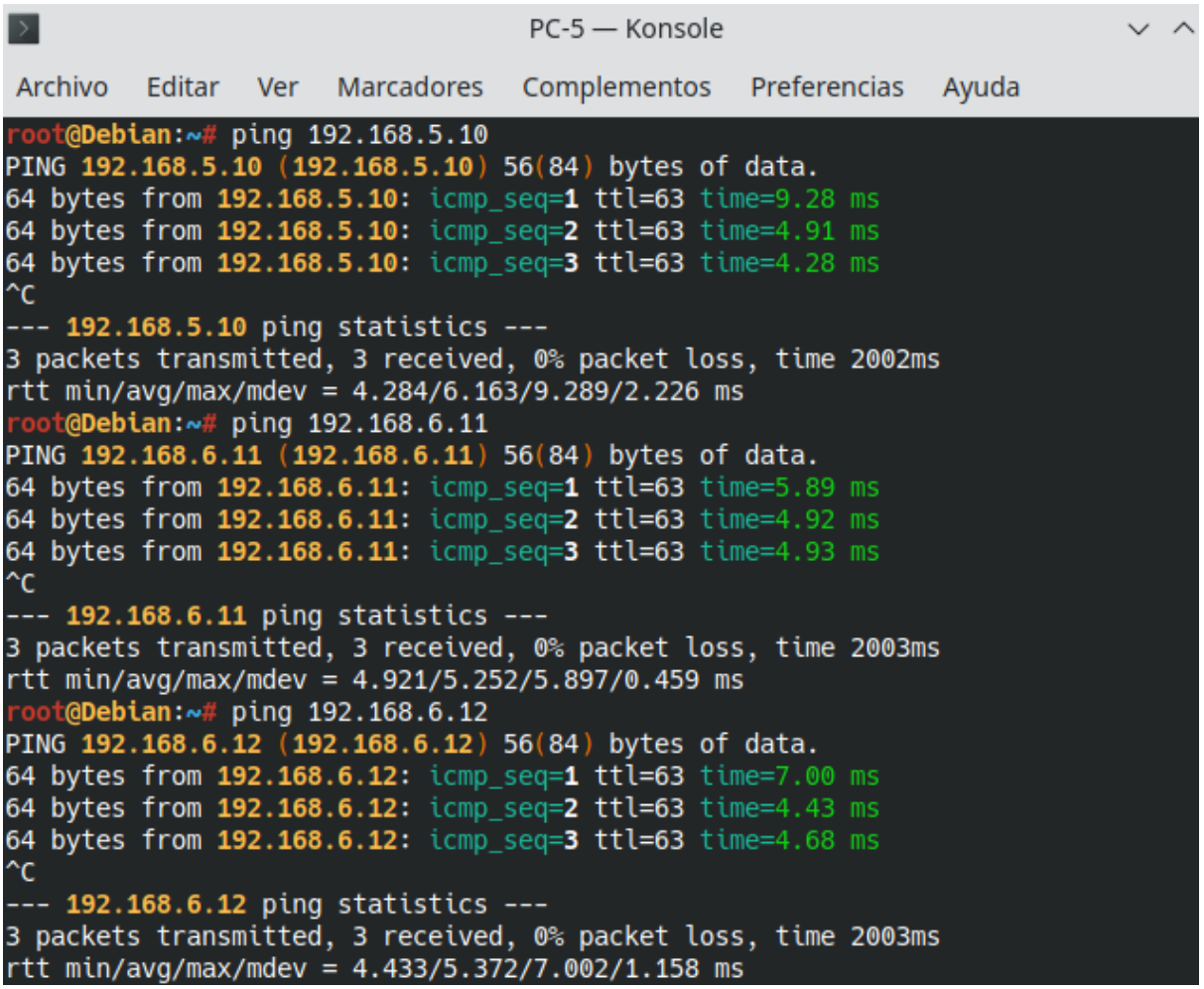
Then, it is necessary to go back to **Interface assignments** to add the virtual interfaces with the already existing set of physical interfaces.



For it to finish working, it is necessary to put some rule in the firewall so that all packets are not denied.

Rules VLAN10, VLAN20, VLAN30:



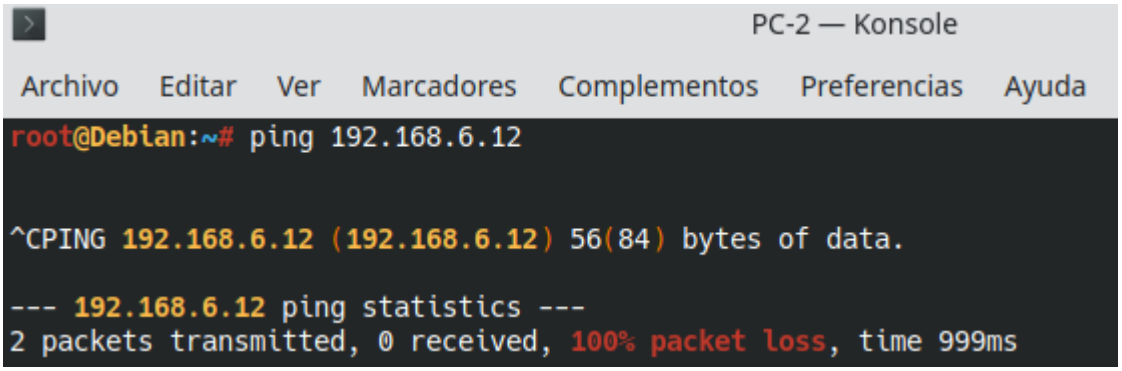


3.3. Alternative flow scheme.

The goal of this improvement is to use ONOS to understand the use of flow tables in OpenvSwitches.

To begin with, it is necessary to disable Reactive Forwarding since it is what makes ONOS have automatic temporary flow rules that allow OvS to function the way we want them to without having to set up those flow rules manually.

Therefore, when removing it, it is normal for communication to go away in general, since the ONOS does not have imposed rules.

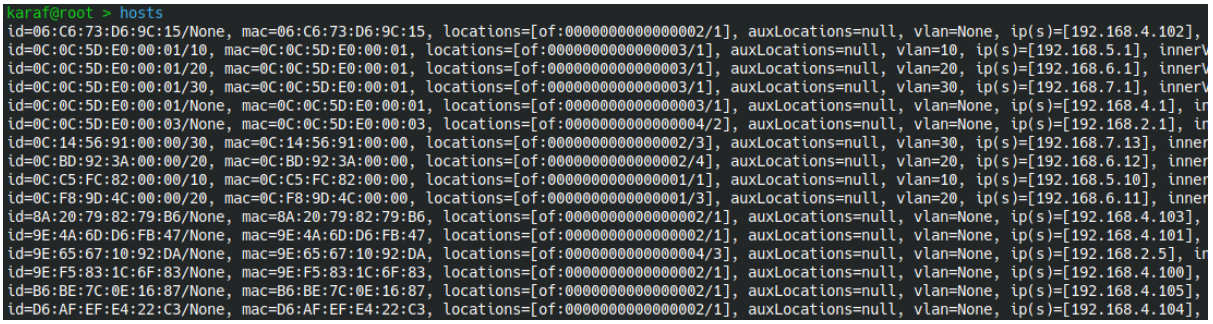


There are several ways to declare flow rules in ONOS controller to be able to establish communication again, in this case, between PC2 and PC4. We will use the intents tool to solve this.

One of the main advantages of using intents rather than simply using stream inputs to program your network is that intents track the state of the network and reconfigure themselves to satisfy your intent. For example, if a link was to go down, the intent framework would redirect your intent (ie your streams) to an alternate route. But what if there is no alternative path? In this case, the intent would go into the failed state and stay there until a route is available.

Because we are on different VLANs, the intent cannot go from PC-2 to PC-4 directly. It is necessary to make the attempt from PC-2 to the virtual interface of pfSense that belongs to the same VLAN and another attempt from PC-4 to the virtual interface of pfSense that belongs to the same VLAN.

To identify the hosts we use the hosts command in ONOS, which provides the following information:



In the previous screenshot we can see that the identification of the PCs is as follows:

- PC2: id=0C:C5:FC:82:00:00/10
- PC4: id=0C:BD:92:3A:00:00/20
- Virtual interface pfSense VLAN10: id=0C:0C:5D:E0:00:01/10
- Virtual interface pfSense VLAN20: id=0C:0C:5D:E0:00:01/20

command to execute the intent:

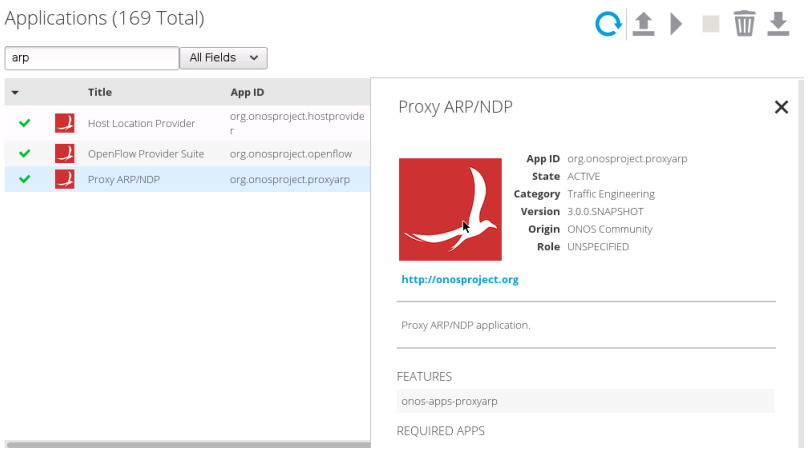
```
add-host-intent <ip host source> <ip host dest>
```

```
karaf@root > add-host-intent 0C:C5:FC:82:00:00/10 0C:0C:5D:E0:00:01/10
 16:17:01
Host to Host intent submitted:
HostToHostIntent{id=0x0, key=0x0, appId=DefaultApplicationId{id=2, name=org.onosproject.cli}, priority=100, resources=[0C:C5:FC:82:00:00/10, 0C:0C
:5D:E0:00:01/10], selector=DefaultTrafficSelector{criteria=[]}, treatment=DefaultTrafficTreatment{immediate=[NOACTION], deferred=[], transition=No
ne, meter=[], cleared=false, StatTrigger=null, metadata=null}, constraints=[LinkTypeConstraint{inclusive=false, types=[OPTICAL]}], resourceGroup=n
ull, one=0C:C5:FC:82:00:00/10, two=0C:0C:5D:E0:00:01/10}
karaf@root > add-host-intent 0C:BD:92:3A:00:00/20 0C:0C:5D:E0:00:01/20
 16:22:04
Host to Host intent submitted:
HostToHostIntent{id=0x5, key=0x5, appId=DefaultApplicationId{id=2, name=org.onosproject.cli}, priority=100, resources=[0C:BD:92:3A:00:00/20, 0C:0C
:5D:E0:00:01/20], selector=DefaultTrafficSelector{criteria=[]}, treatment=DefaultTrafficTreatment{immediate=[NOACTION], deferred=[], transition=No
ne, meter=[], cleared=false, StatTrigger=null, metadata=null}, constraints=[LinkTypeConstraint{inclusive=false, types=[OPTICAL]}], resourceGroup=n
ull, one=0C:BD:92:3A:00:00/20, two=0C:0C:5D:E0:00:01/20}
```

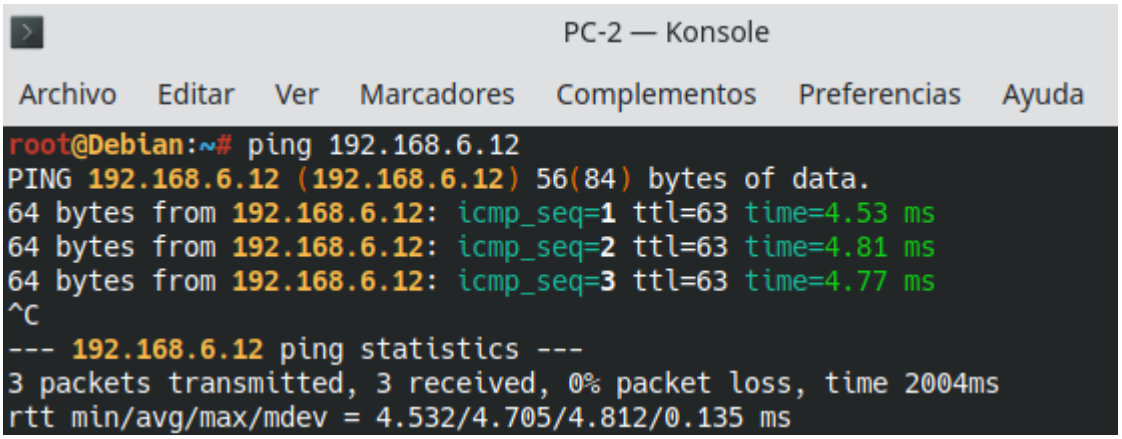
Even having done all of the above, we have not been able to get connectivity between PC-2 and PC-4, not even between PC-2 and the pfSense interface. This is because the messages are not arriving because the ARP messages are not resolved in ONOS.

Therefore, a possible solution is to activate the org.onosproject.proxyarp app through the command line of ONOS itself or through the graphical interface. By command line it would be the following command:

```
app activate org.onosproject.proxyarp
```

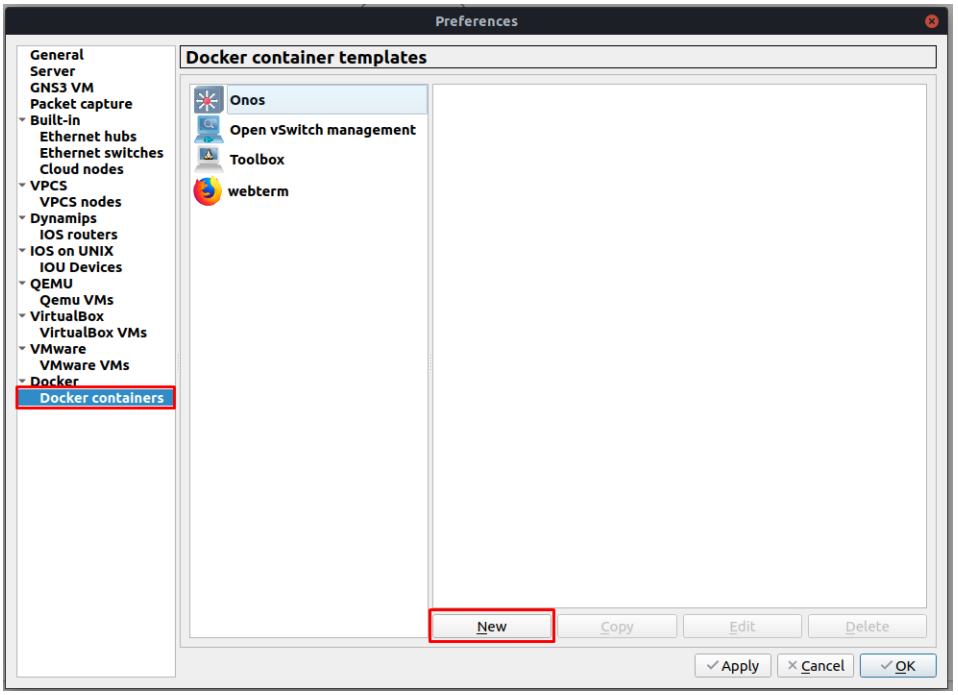


Once all this is done there is a perfect connection between PC-2 and PC-4:



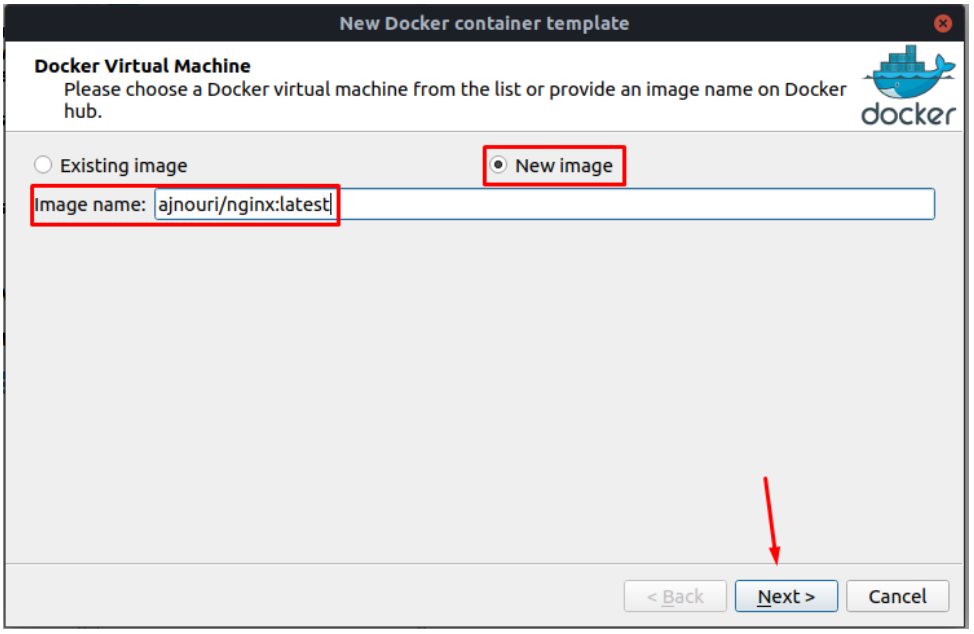
3.4. Deployment and configuration of the DMZ zone.

Now we are going to deploy the DMZ zone in the project. We will activate and configure the DHCP Server and we will incorporate an OpenvSwitch integrated in ONOS-1. Then we must incorporate a device (Server1) with an NGINX web server, and we must deploy containerized from the *ajnouiri/nginx* image. To install the *ajnouri/nginx* on the GNS3 we will follow the following steps:



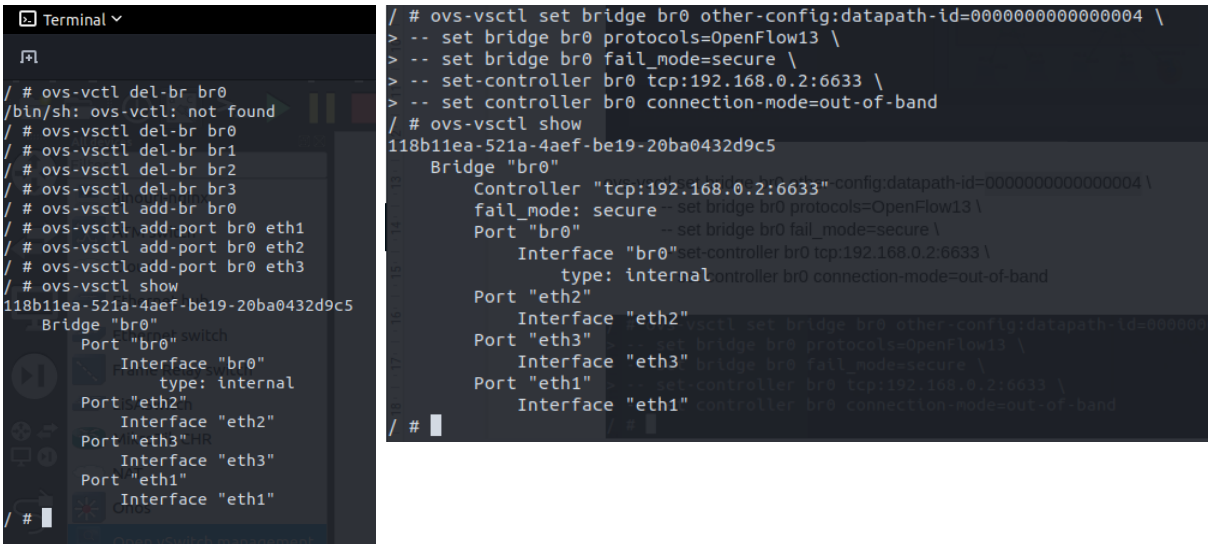
Go to *Edit* → *Preferences* → *Docker containers* and click on *New:*

Select *New Image*, and in the dialog box, type **`ajnouri/nginx:latest`**, as shown below, and then click *Next*:

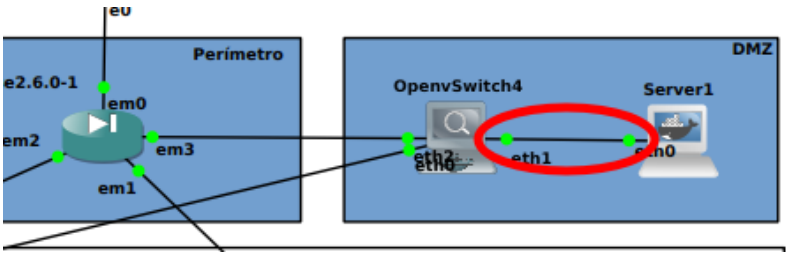


In the following screens we just have to click next until you finish adding the template. Once the installation of *ajnouri/nginx* image is finished, we continue with the DMZ configuration. We are going to incorporate a device and give it the name Server1 as well as use DHCP, previously creating a static reservation with its MAC in the firewall.

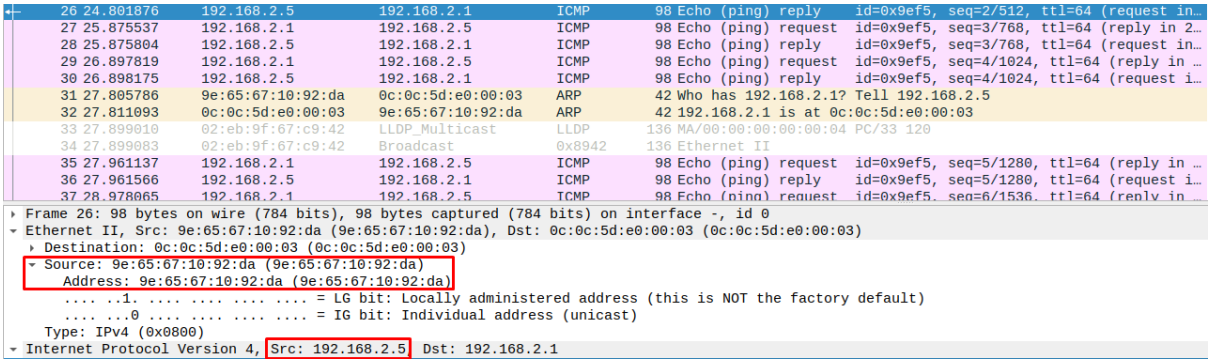
First we are going to add and boot the OpenvSwitch4 device in the DMZ zone and connect to its console and apply the following commands



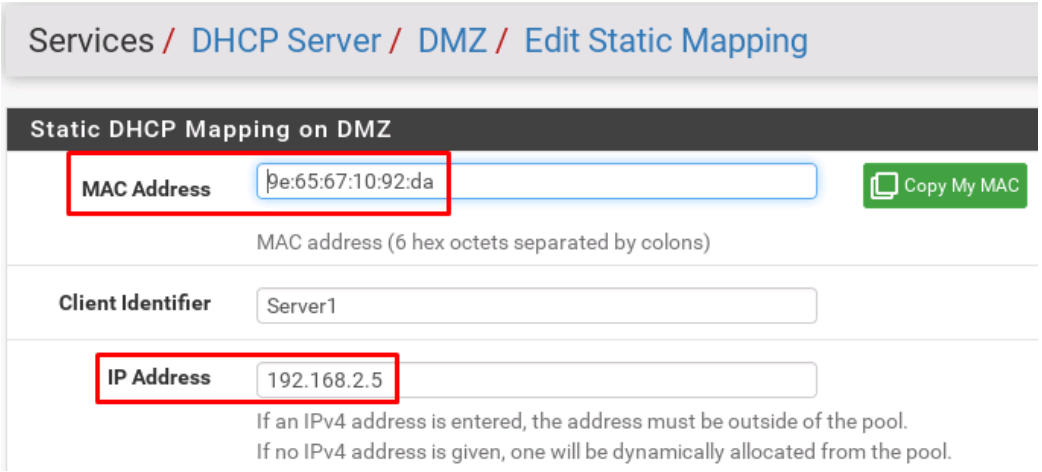
To configure the static reservation with the MAC address we open the pfSense from webterm-1, in management area, we go to *Services → DHCP Server → DMZ* and in the last option, *DHCP Static Mappings for this interface,* we are going to add the MAC address. To get the server's MAC address, we have to capture packets between the OpenvSwitch and Server1 and execute a ping command from the pfSense console to Server1, but first we are going to set the eth0 interface of Server1 the address 192.168.2.5 statically :



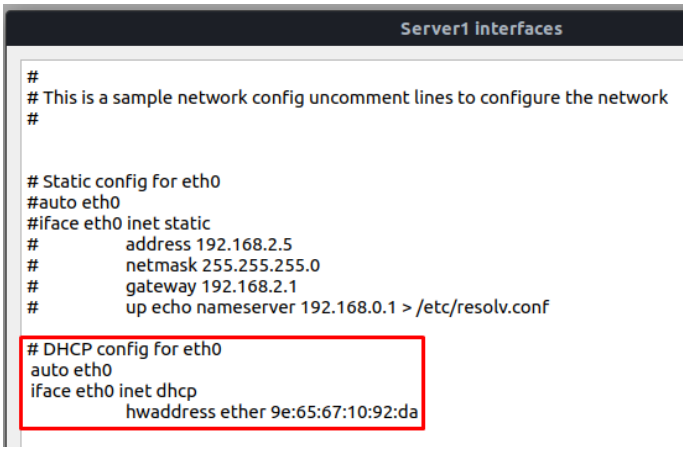
When wireshark opens and we see that it is collecting packets, we take an ICMP Echo reply packet and we look for the MAC address of the source device.



We copy that MAC address and enter it in the *DHCP Static Mappings for this interface*, then we have specify the IPv4 address, in this case we want it to be 192.168.2.5:

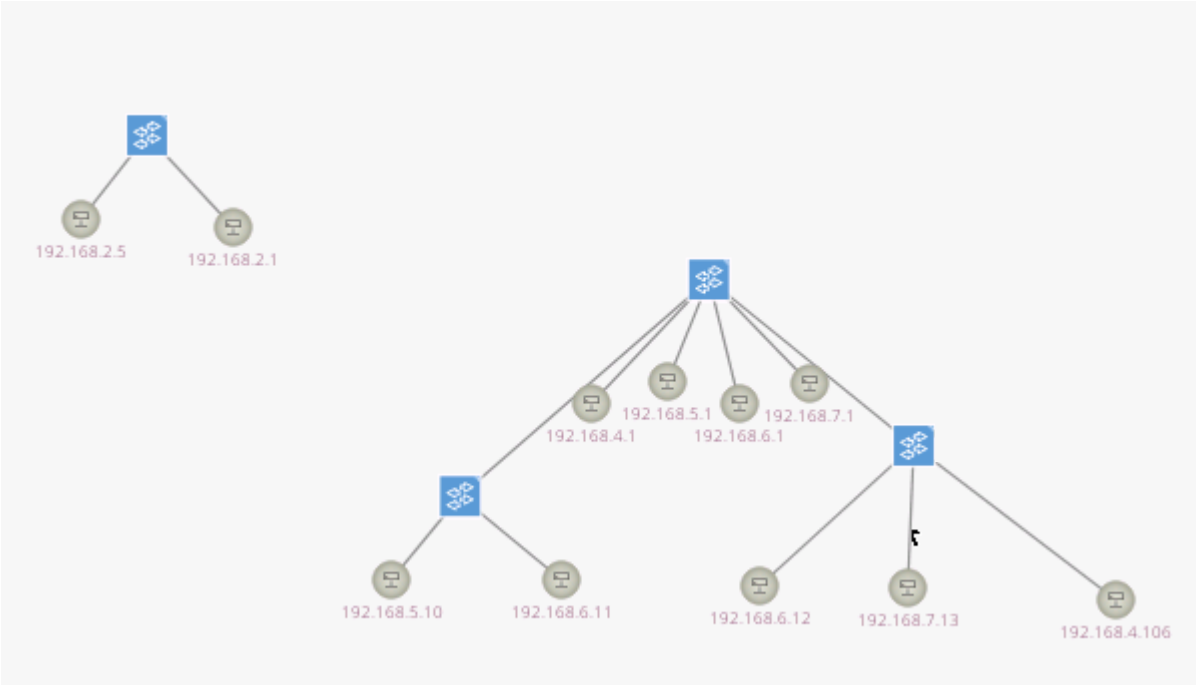


Finally we go to the Server1 network configuration and we set the interface eth0 to be given an IP address by DHCP:



On the *hwaddress ether* we are going to put the MAC address that had previously been copied so the server's MAC address stays unchanged.

Now looking at the information displayed on the ONOS controller when discovering the new topology in the DMZ zone we can see something like this:



If we connect from webterm-3 and try to connect to Server1 we will see that it connects:

3.5. Security policy improvements.

To carry out this project we were asked to add the following rules to the pfSense firewall.

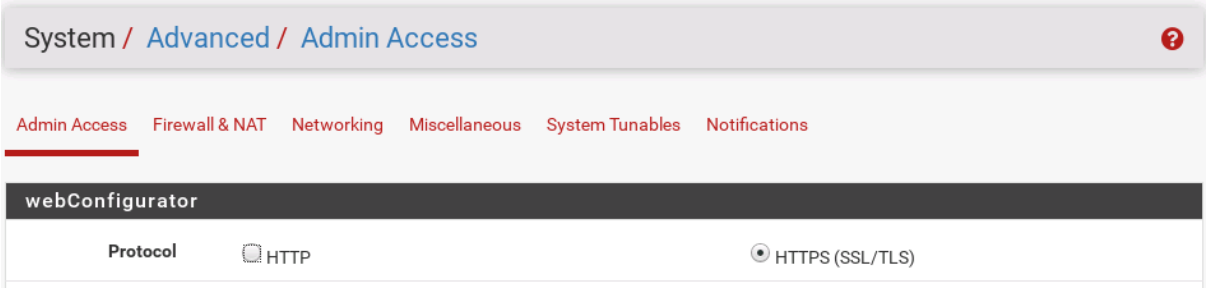
The pfSense firewall defaults to overriding anything that is not explicitly accepted, so all of our rules are made to let it through.

Below are the rules we have implemented:

- Support only HTTPS connection to firewall management interface (do not support HTTP).

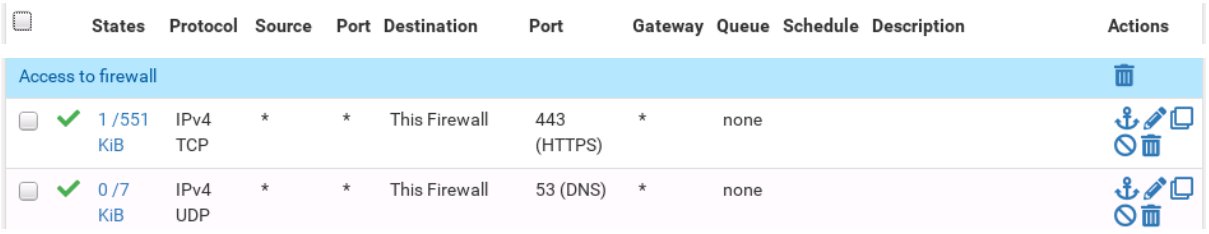
Implementing this in the firewall was relatively easy, we only had to generate a rule in the MGMT interface that, from any given origin, would allow us to connect to the firewall (this firewall) that is, by any of the IPs of the pfSense interfaces, and that it will only be possible if it is through port 443 (HTTPS).

We accessed the pfSense configuration in System→Advanced→Admin Access in the webConfigurator we put the HTTPS protocol, with this it will only allow us to connect through HTTPS to the management interface. If we try to connect via HTTP, it will automatically redirect us and we will enter via HTTPS. In any case, in the rules that are implemented in the following sections, we also specify that connections to this interface must be made through port 443 (HTTPS).



- HTTPS access to the firewall management interface should only be possible from the MANAGE/MGMT network.

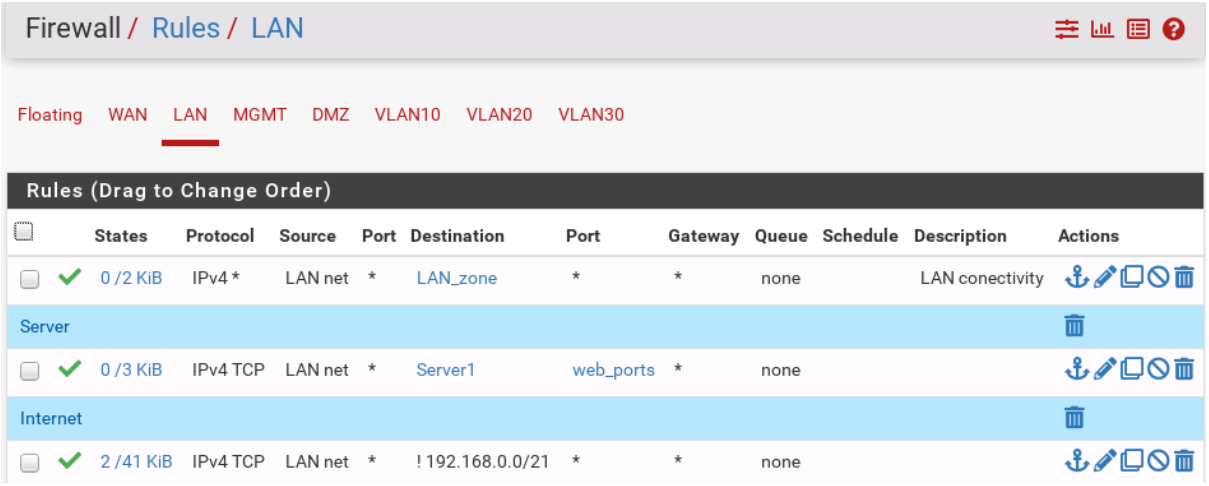
MGMT



We included a rule, only in pfSense's MGMT interface, that dictates that all traffic coming through this interface that matches any IP address on any firewall interface and also wants to connect via port 443 (HTTPS) will be able to go through the firewall. We also added a rule that allows DNS traffic, 53 (DNS).

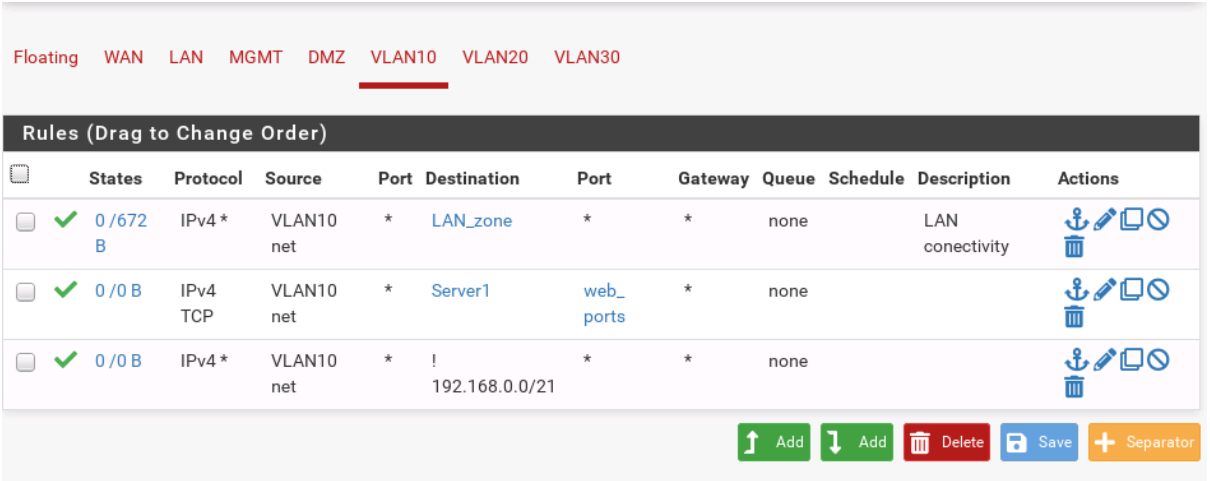
These rules are not repeated in any other pfSense interface, thus only allowing access from the management zone.

- Ensure that the USERS/LAN network can only access the network services it needs, the Internet, and the servers in the DMZ.



We create three rules for the pfSense LAN interface, the first one guarantees connectivity between VLANs. It allows any user coming from the LAN, whether or not it belongs to a VLAN, to communicate with other users of the same. In the following rule we allow any user on the LAN to access the DMZ server through ports 80 or 443 (web_ports). Finally, we allow LAN users to have access to the internet, or what is the same, we allow all traffic that is not directed towards our own network.

Example of how it would look in VLAN10.



By having interfaces for each of the VLANs in pfSense, these rules are repeated in each of them, the reason for this is that each VLAN interface acts as a physical interface for the firewall, that is, it does not matter what rules we put on the LAN interface, they will not affect the interfaces of the VLANs.

- From the MANAGEMENT/MGMT network it must be possible to browse the Internet and Server1. In addition, SSH and Telnet access to all computers on the network must be possible.

MGMT

☐	✓	0 /0 B	IPv4 TCP	MGMT net	*	OpenvSwitches	remote_ access	*	none	Access to OpenvSwitches through SSH and Telnet
☐	✓	0 /0 B	IPv4 TCP	MGMT net	*	LAN net	remote_ access	*	none	Only allow SSH and telnet to LAN net

For this we have added 2 rules: One to be able to do SSH and telnet with the OpenvSwitches as destination and another to do the same but to the LAN net.
To make the alias we have taken into account that the OpenvSwitches actually belong to the MGMT network.

- Access from the WAN should only be possible for the related connections initiated from within and for the NAT Port Forward configuration performed.

Floating WAN LAN MGMT DMZ VLAN10 VLAN20 VLAN30

Rules (Drag to Change Order)

	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
☐	✓	0 /4 KiB	IPv4 TCP	*	*	Server1	80 (HTTP)	*	none		NAT Port forward to web servers in DMZ

Add Add Delete Save Separator

Thanks to this rule, access can be made from the WAN to the server through the NAT service, using port 80. Because the server can only be accessed through HTTP.

This rule has been generated automatically since we have used NAT.

Firewall / NAT / Port Forward

Port Forward 1:1 Outbound NPt

Rules

		Interface	Protocol	Source Address	Source Ports	Dest. Address	Dest. Ports	NAT IP	NAT Ports	Description	Actions
☐	✓	WAN	TCP	*	*	WAN address	80 (HTTP)	Server1	80 (HTTP)	Port forward to web servers in DMZ	

Add Add Delete Save Separator

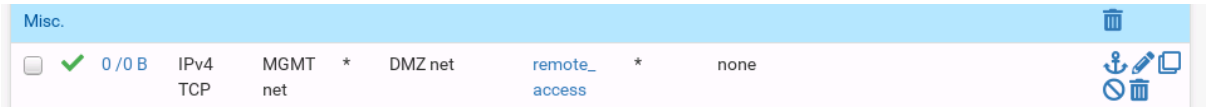
This will transform the address of the input interface to the network (WAN address) that is given via DHCP, it will automatically change to 192.168.2.5 (the ip of server1).

Related connections started from within are accepted by default in pfSense so nothing needs to be done in that regard.

- You must be able to access the network services you need from the DMZ. Internet browsing should not be possible. Management access (SSH or Telnet) to the DMZ computers should only be possible from the MANAGEMENT/MGMT network.

In the DMZ we have not considered that you need any services outside of your network, so you do not have any rules.

The access for the administration of the DMZ equipment from management through ports 22 and 23 is a rule that goes in the management interface, as follows:



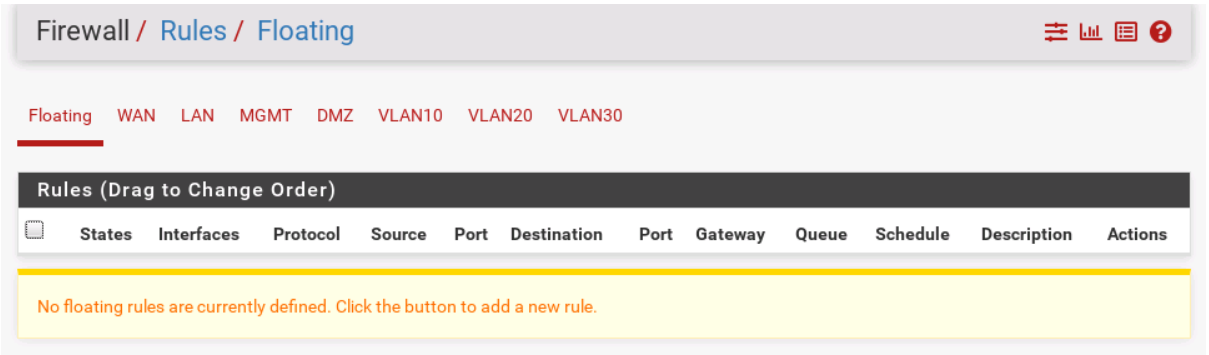
3.5.1 Firewall improvements.

One of the most important things is that the hosts within the network are always who they are supposed to be, so that they cannot enjoy privileges that they do not really have.

To do this, as everything that is not specified is denied, it was only enough to be as specific as possible in each rule. For example, if someone who is from the LAN wanted to enter the pfSense by ip spoofing, he is not going to be able to do it because he can only enter through the em1 interface of the pfSense (LAN) and blocks it instantly because there is no rule for the management network through that interface. regarding the firewall input.

3.5.2 Firewall rules final result.

As a result we have the following configuration of rules for the interfaces:



Firewall / Rules / WAN

Floating | WAN | LAN | MGMT | DMZ | VLAN10 | VLAN20 | VLAN30

Rules (Drag to Change Order)

	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
☐ ✓	0 /4 KiB	IPv4 TCP	*	*	Server1	80 (HTTP)	*	none		NAT Port forward to web servers in DMZ	

Add | Add | Delete | Save | Separator

Firewall / Rules / LAN

Floating | WAN | LAN | MGMT | DMZ | VLAN10 | VLAN20 | VLAN30

Rules (Drag to Change Order)

	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
☐ ✓	0 /0 B	IPv4 *	LAN net	*	LAN_zone	*	*	none		LAN conectivity	
Server											
☐ ✓	0 /0 B	IPv4 TCP	LAN net	*	Server1	web_ports	*	none			
Internet											
☐ ✓	0 /0 B	IPv4 TCP	LAN net	*	! 192.168.0.0/21	*	*	none			

Add | Add | Delete | Save | Separator

Floating WAN LAN MGMT DMZ VLAN10 VLAN20 VLAN30

Rules (Drag to Change Order)

	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
Access to firewall											
	2 /866 KiB	IPv4 TCP	*	*	This Firewall	443 (HTTPS)	*	none			
	0 /7 KiB	IPv4 UDP	*	*	This Firewall	53 (DNS)	*	none			
Misc.											
	0 /0 B	IPv4 TCP	MGMT net	*	DMZ net	remote_ access	*	none			
	0 /0 B	IPv4 TCP	MGMT net	*	OpenvSwitches	remote_ access	*	none		Access to OpenvSwitches through SSH and Telnet	
	0 /0 B	IPv4 TCP	MGMT net	*	LAN net	remote_ access	*	none		Only allow SSH and telnet to LAN net	
Server1											
	0 /0 B	IPv4 TCP	MGMT net	*	DMZ net	*	*	none			
Internet											
	0 /45 KiB	IPv4 TCP	MGMT net	*	! 192.168.0.0/21	*	*	none		Internet access	

Add Add Delete Save Separator

Floating WAN LAN MGMT DMZ VLAN10 VLAN20 VLAN30

Rules (Drag to Change Order)

States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions

No rules are currently defined for this interface
All incoming connections on this interface will be blocked until pass rules are added. Click the button to add a new rule.

Add Add Delete Save Separator

Firewall / Rules / VLAN10

Floating WAN LAN MGMT DMZ VLAN10 VLAN20 VLAN30

Rules (Drag to Change Order)

	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
✓	0 /0 B	IPv4 *	VLAN10 net	*	LAN_zone	*	*	none		LAN conectivity	
✓	0 /0 B	IPv4 TCP	VLAN10 net	*	Server1	web_ ports	*	none			
✓	0 /179 B	IPv4 *	VLAN10 net	*	! 192.168.0.0/21	*	*	none			

Add Add Delete Save Separator

Firewall / Rules / VLAN20

Floating WAN LAN MGMT DMZ VLAN10 VLAN20 VLAN30

Rules (Drag to Change Order)

	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
✓	0 /0 B	IPv4 *	VLAN20 net	*	LAN_zone	*	*	none		LAN conectivity	
✓	0 /0 B	IPv4 TCP	VLAN20 net	*	Server1	web_ ports	*	none			
✓	0 /358 B	IPv4 *	VLAN20 net	*	! 192.168.0.0/21	*	*	none			

Add Add Delete Save Separator

Firewall / Rules / VLAN30

Floating WAN LAN MGMT DMZ VLAN10 VLAN20 VLAN30

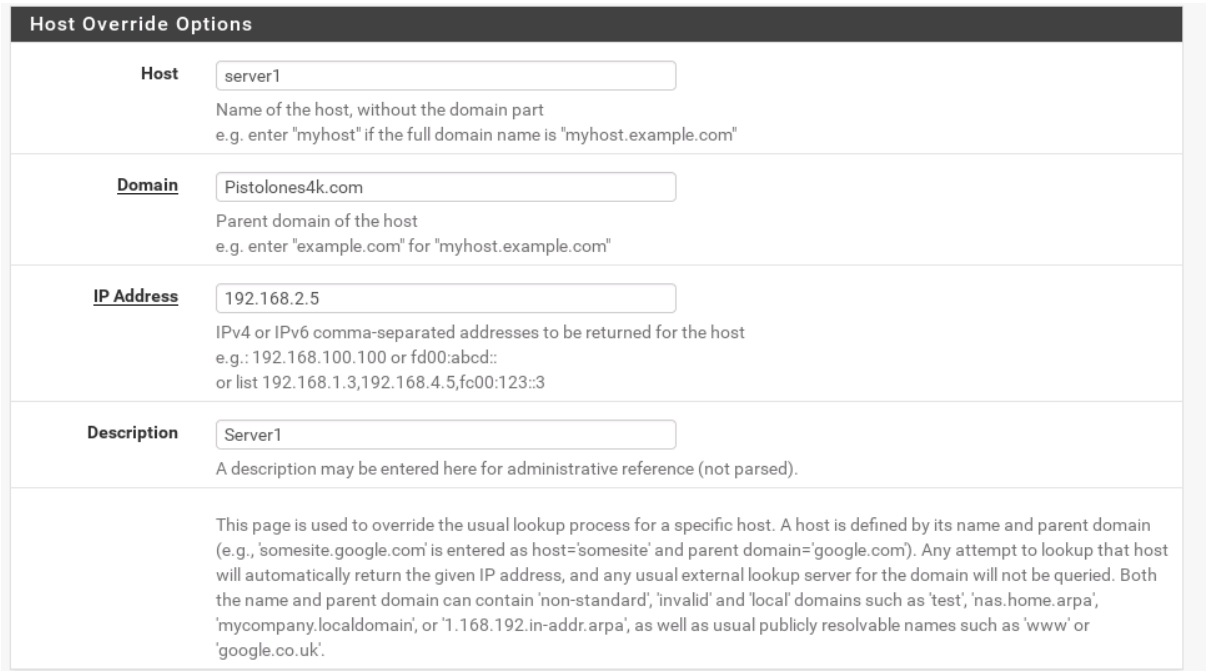
Rules (Drag to Change Order)

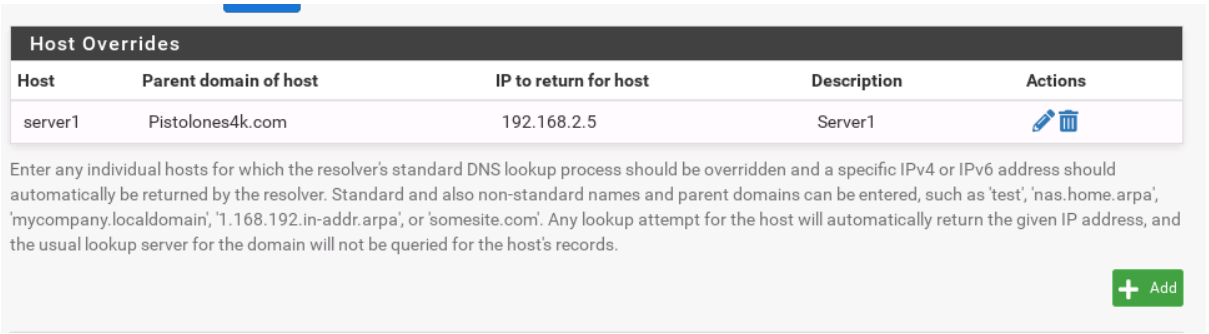
	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
✓	0 /0 B	IPv4 *	VLAN30 net	*	LAN_zone	*	*	none		LAN conectivity	
✓	0 /0 B	IPv4 TCP	VLAN30 net	*	Server1	web_ ports	*	none			
✓	0 /358 B	IPv4 *	VLAN30 net	*	! 192.168.0.0/21	*	*	none			

Add Add Delete Save Separator

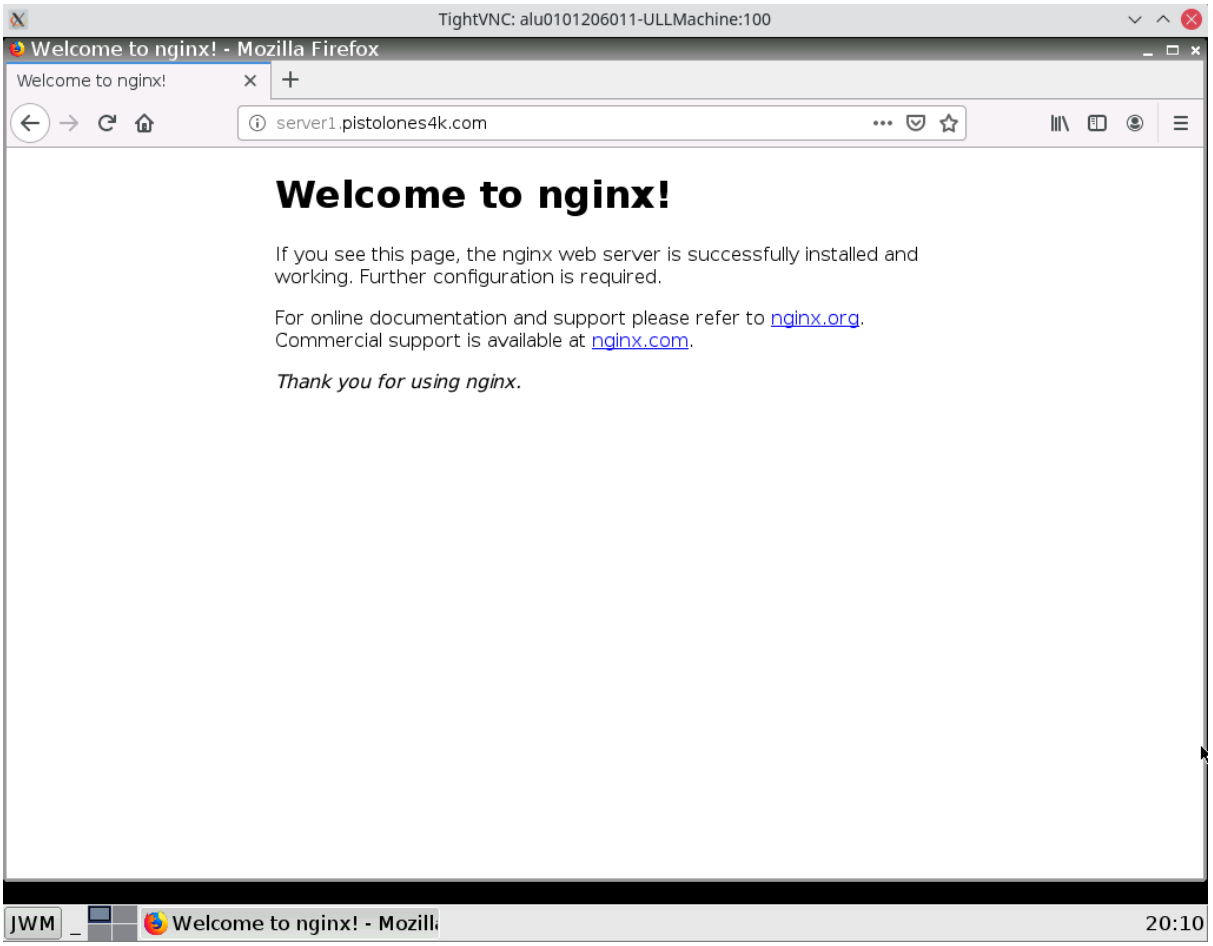
3.6. Little extra improvement.

As a small improvement to the project we have made an internal host override for server1.





Thanks to this we can access the DMZ server from any device using the DNS name server1.pistolones4k.com (regarding the ones outside our network (WAN)).



4. References.

https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-20-04-es

https://www.gns3.com/community/featured/openvswitch-docker-issue

https://docs.gns3.com/docs/emulators/docker-support-in-gns3/

VLAN
https://stackoverflow.com/questions/33342146/how-to-tag-outgoing-traffic-with-vlan-id

https://adhioutlined.github.io/virtual/Openvswitch-Cheat-Sheet/

https://groups.google.com/a/onosproject.org/g/onos-discuss/c/ob28n_B6cS8?pli=1

https://linuxize.com/post/lsmod-command-in-linux/

Flujo
https://groups.google.com/a/onosproject.org/g/onos-dev/c/GrV3xZfaEPs

DMZ
https://hub.docker.com/r/ajnouri/nginx/tags

Firewall
https://docs.netgate.com/pfsense/en/latest/troubleshooting/nat-port-forwards.html

https://stackoverflow.com/questions/60744352/onos-service-start-frameworkevent-error-and-gui-not-ready-yet

https://docs.netgate.com/pfsense/en/latest/troubleshooting/locked-out.html