### Subnet

The network consists of 4 subnets:

* **The first subnet includes host-A and router-1. The subnet is a / 23 and therefore you can get as a result of IP addresses**232-23-2 = 510 different host. (495 minimum required)
* **The second subnet includes host-B and router-1. The subnet is a / 24 and therefore you can get as a result of IP addresses** 232-24-2 = 254 different host. (140 minimum required)
* **The third subnet includes router-1 and router-2. The subnet is a / 30 and therefore you can get IP addresses as a result** 232-30-2 = 2 different host.
* **The fourth subnet includes router-2 and host-C. The subnet is a / 24 and therefore you can get as a result of IP addresses** 232-24-2 = 254 different host. (155 minimum required)

Vlan

The router-1 connects to 2 different subnets on a single port thanks to 2 different Vlans. These 2 Vlans are marked with 2 different VIDs and are 10 and 20

### Interface-IP mapping

|  |  |  |  |
| --- | --- | --- | --- |
| Device | Interface | IP | Subnet |
| host-A | enp0s8 | 11.1.0.2/23 | First subnet |
| router-1 | enp0s8.10 | 11.1.0.1/23 | First subnet |
| host-B | enp0s8 | 12.0.1.2 | Second subnet |
| router-1 | enp0s8.20 | 12.0.1.1/24 | Second subnet |
| host-C | enp0s8 | 13.0.1.34/24 | Third subnet |
| router-2 | enp0s8 | 13.0.1.33/24 | Third subnet |
| router-1 | enp0s9 | 10.0.1.37/30 | Fourth subnet |
| router-2 | enp0s9 | 10.0.1.38/30 | Fourth subnet |

### Vagrant file and provisioning scripts

### The project has several files, but the most important is that of Vagrantfile because it configures and manages the various components of the network. All virtual machines are based on 'ubuntu / bionic64' because it is compatible with all the things needed for the project.

### Router 1

### The code in the Vagrantfile that belongs to this router is as follows:

config.vm.define "router-1" do |router1|

    router1.vm.box = "ubuntu/bionic64"

    router1.vm.hostname = "router-1"

    router1.vm.network "private\_network", virtualbox\_\_intnet: "broadcast\_router-south-1", auto\_config: false

    router1.vm.network "private\_network", virtualbox\_\_intnet: "broadcast\_router-inter", auto\_config: false

    router1.vm.provision "shell", path: "router-1.sh"#, run: 'always'

    config.ssh.insert\_key = false

    router1.vm.provider "virtualbox" do |vb|

      vb.memory = 256

    end

You can clearly see that the 'router-1.sh' script is started.

This file is as follows:

export DEBIAN\_FRONTEND=noninteractive

# Startup commands go here

apt-get update

apt-get install -y tcpdump curl traceroute --assume-yes

# Make sure the kernel forwards packets

sysctl net.ipv4.ip\_forward=1

# Set-up the interfaces

ip link add link enp0s8 name enp0s8.10 type vlan id 10

ip link add link enp0s8 name enp0s8.20 type vlan id 20

sudo ip link set dev enp0s8 up

sudo ip link set dev enp0s9 up

sudo ip link set dev enp0s8.10 up

sudo ip link set dev enp0s8.20 up

sudo ip addr add 11.1.0.1/23 dev enp0s8.10

sudo ip addr add 12.0.1.1/24 dev enp0s8.20

sudo ip addr add 10.0.1.37/30 dev enp0s9

#Routing rules

sudo ip route del default

sudo ip route add default via 10.0.1.38

Initially you download the libraries, then you specify the IP addresses for the Vlan and interfaces, finally we take care of the routing.

#### Router 2

### The code in the Vagrantfile that belongs to this router is as follows:

config.vm.define "router-2" do |router2|

    router2.vm.box = "ubuntu/bionic64"

    router2.vm.hostname = "router-2"

    router2.vm.network "private\_network", virtualbox\_\_intnet: "broadcast\_router-south-2", auto\_config: false

    router2.vm.network "private\_network", virtualbox\_\_intnet: "broadcast\_router-inter", auto\_config: false

    config.ssh.insert\_key = false

    router2.vm.provision "shell", path: "router-2.sh"#, run: 'always'

    router2.vm.provider "virtualbox" do |vb|

      vb.memory = 256

    end

You can clearly see that the 'router-2.sh' script is started.

This file is as follows:

export DEBIAN\_FRONTEND=noninteractive

# Startup commands go here

apt-get update

apt-get install -y tcpdump curl traceroute --assume-yes

# Make sure the kernel forwards packets

sysctl net.ipv4.ip\_forward=1

# Set-up the interfaces

sudo ip link set dev enp0s8 up

sudo ip link set dev enp0s9 up

sudo ip addr add 13.0.1.33/24 dev enp0s8

sudo ip addr add 10.0.1.38/30 dev enp0s9

#Easy way: forward everything to the other router

sudo ip route del default

sudo ip route add default via 10.0.1.37

First you download the libraries, then you configure the IP addresses and finally you take care of the routing.

Switch

### The code in the Vagrantfile that belongs to the switch is as follows:

config.vm.define "switch" do |switch|

    switch.vm.box = "ubuntu/bionic64"

    switch.vm.hostname = "switch"

    switch.vm.network "private\_network", virtualbox\_\_intnet: "broadcast\_router-south-1", auto\_config: false

    switch.vm.network "private\_network", virtualbox\_\_intnet: "broadcast\_host\_a", auto\_config: false

    switch.vm.network "private\_network", virtualbox\_\_intnet: "broadcast\_host\_b", auto\_config: false

    config.ssh.insert\_key = false

    switch.vm.provision "shell", path: "switch.sh"#, run: 'always'

    switch.vm.provider "virtualbox" do |vb|

      vb.memory = 256

    end

You can clearly see that the 'switch.sh' script is started.

This file is as follows:

export DEBIAN\_FRONTEND=noninteractive

apt-get update

apt-get install -y tcpdump

apt-get install -y openvswitch-common openvswitch-switch apt-transport-https ca-certificates curl software-properties-common

# Startup commands for switch go here

ovs-vsctl add-br switch

sudo ip link set dev enp0s8 up

sudo ip link set dev enp0s9 up

sudo ip link set dev enp0s10 up

# The access ports

ovs-vsctl --may-exist add-port switch enp0s9 tag=10

ovs-vsctl --may-exist add-port switch enp0s10 tag=20

# And the trunk link

ovs-vsctl --may-exist add-port switch enp0s8

First you download the libraries and then you have to take care of configuring a bridge and adding interfaces to it.

#### Host A e Host B

The code of these 2 are almost the same because only the IP addresses change.

### The code in the Vagrantfile that belong to host A and host B are as follows:

config.vm.define "host-a" do |hosta|

    hosta.vm.box = "ubuntu/bionic64"

    hosta.vm.hostname = "host-a"

    hosta.vm.network "private\_network", virtualbox\_\_intnet: "broadcast\_host\_a", auto\_config: false

    config.ssh.insert\_key = false

    hosta.vm.provision "shell", path: "host-a.sh"#, run: 'always'

    hosta.vm.provider "virtualbox" do |vb|

      vb.memory = 256

    end

  end

  config.vm.define "host-b" do |hostb|

    hostb.vm.box = "ubuntu/bionic64"

    hostb.vm.hostname = "host-b"

    hostb.vm.network "private\_network", virtualbox\_\_intnet: "broadcast\_host\_b", auto\_config: false

    hostb.vm.provision "shell", path: "host-b.sh"#, run: 'always'

    config.ssh.insert\_key = false

    hostb.vm.provider "virtualbox" do |vb|

      vb.memory = 256

    end

As you can see, the 2 codes are very similar and recall their 'host-a.sh' and 'host-b.sh' respectively.

The 2 script codes are as follows and they are very similar:

--Host A

export DEBIAN\_FRONTEND=noninteractive

# Startup commands go here

apt-get update

apt-get install -y tcpdump curl traceroute --assume-yes

ip link set dev enp0s8 up

ip addr add 11.1.0.2/23 dev enp0s8

ip route del default

ip route add default via 11.1.0.1

Host B

export DEBIAN\_FRONTEND=noninteractive

# Startup commands go here

apt-get update

apt-get install -y tcpdump curl traceroute --assume-yes

ip link set dev enp0s8 up

ip addr add 12.0.1.2/24 dev enp0s8

ip route del default

ip route add default via 12.0.1.1

For both cases I have to configure the interfaces, the IP addresses and connect them to the same router 1, finally I start docker by first installing its library.

Host C

### The code in the Vagrantfile that belongs to host C is the following:

config.vm.define "host-c" do |hostc|

    hostc.vm.box = "ubuntu/bionic64"

    hostc.vm.hostname = "host-c"

    hostc.vm.network "private\_network", virtualbox\_\_intnet: "broadcast\_router-south-2", auto\_config: false

    config.ssh.insert\_key = false

    hostc.vm.provision "shell", path: "host-c.sh"#, run: 'always'

    hostc.vm.provider "virtualbox" do |vb|

    vb.memory = 512

    end

You can clearly see that the 'switch.sh' script is started, then that the memory given is 512 and not 256 because it gave us an error that said: 'full memory '.

Questo file è il seguente:

export DEBIAN\_FRONTEND=noninteractive

# Startup commands go here

sudo su

sudo apt-get update

sudo apt-get install -y tcpdump curl traceroute --assume-yes

sudo su

# Install docker and run nginx inside

sudo apt-get update

sudo apt-get install -y apt-transport-https ca-certificates curl software-properties-common --assume-yes --force-yes

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

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"

sudo apt-get update

sudo apt-get install -y docker-ce --assume-yes --force-yes

sudo mkdir -p /var/www

sudo chmod +r /var/www

sudo echo "Just a test page!" > /var/www/index.html

sudo docker run --name docker-nginx \

--restart=always \

-p 80:80 -d \

-v /var/www:/usr/share/nginx/html:ro \

nginx

ip link set dev enp0s8 up

ip addr add 13.0.1.34/24 dev enp0s8

ip route del default

ip route add default via 13.0.1.33

First you download the libraries and then you need to create an index.html page, then docker starts, finally you configure both the interfaces and the connection to the router 2.

## How-to

* Install Virtualbox
* Install Vagrant
* Clone this repository

~$ git clone <https://github.com/sixtynicooo/dncs-lab-funzionante-e-pronto.git>

To start the project you need to use the 'vagrant up' command, alternatively you can use the following command to run the file for the test and it is 'sh test.sh' that checks if the various connections between the hosts are working.

Problems

The project was done using window and we had some problems: the first is that it sometimes gave me an error with permission denied using 'vagrant ssh xxxx' and to solve the problem I used the line of code 'config.ssh.insert\_key = false 'in the Vagrantifile file. Another problem was reserved for hosts, basically part of the error was' ssh responded with a non-zero status' and to fix it you had to configure a vagrant variable using the window terminal and typing the following code 'SET VAGRANT\_PREFER\_SYSTEM\_BIN = 0 '.

Overall this project was interesting because it shows how to connect the various IPs and how the various components of the network interact.