Certified Tech Developer
The Ultimate Degree

Objetivos

En el siguiente ejercicio vamos a ver cómo viaja el tráfico de red de una computadora a la otra. Para ello, te brindamos un vagrantfile que contiene todo lo que vas a necesitar para completar el ejercicio.

¿Qué recibimos?

- Un vagrantfile que contiene todo lo necesario:
  Una máquina virtual llamada 'cliente'
  Una máquina virtual llamada 'servidor'
- Un servidor web instalado (nginx)
- Una página web de muestra

Ambas máquinas pueden resolver sus IPs por los nombres de 'servidor' y 'cliente'.

Instrucciones

Ejercicio 1 - De forma individual, ejecutamos los siguiente pasos:

1) Crear una carpeta en algún lugar de tu computadora.

2) Dentro de esa carpeta, crear un archivo con el nombre 'vagrantfile' (sin las comillas) e insertar en el mismo el siguiente contenido.

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

# All Vagrant configuration is done below. The "2" in
Vagrant.configure
# configures the configuration version (we support older styles
for
```

```
# backwards compatibility). Please don't change it unless you
know what
# you're doing.
Vagrant.configure("2") do |config|
  # The most common configuration options are documented and
commented below.
  # For a complete reference, please see the online
documentation at
  # https://docs.vagrantup.com.

  # Every Vagrant development environment requires a box. You
can search for
  # boxes at https://vagrantcloud.com/search.
  # config.vm.box = "base"

  # Disable automatic box update checking. If you disable this,
then
  # boxes will only be checked for updates when the user runs
  # `vagrant box outdated`. This is not recommended.
  # config.vm.box_check_update = false

  # Create a forwarded port mapping which allows access to a
specific port
  # within the machine from a port on the host machine. In the
example below,
  # accessing "localhost:8080" will access port 80 on the guest
machine.
  # NOTE: This will enable public access to the opened port
  # config.vm.network "forwarded_port", guest: 80, host: 8080

  # Create a forwarded port mapping which allows access to a
specific port
  # within the machine from a port on the host machine and only
allow access
  # via 127.0.0.1 to disable public access
  # config.vm.network "forwarded_port", guest: 80, host: 8080,
host_ip: "127.0.0.1"
```

```
  # Create a private network, which allows host-only access to
the machine
  # using a specific IP.
  # config.vm.network "private_network", ip: "192.168.33.10"

  # Create a public network, which generally matched to bridged
network.
  # Bridged networks make the machine appear as another physical
device on
  # your network.
  # config.vm.network "public_network"

  # Share an additional folder to the guest VM. The first
argument is
  # the path on the host to the actual folder. The second
argument is
  # the path on the guest to mount the folder. And the optional
third
  # argument is a set of non-required options.
  # config.vm.synced_folder "../data", "/vagrant_data"

  # Provider-specific configuration so you can fine-tune various
  # backing providers for Vagrant. These expose
provider-specific options.
  # Example for VirtualBox:
  #
  # config.vm.provider "virtualbox" do |vb|
  #   # Display the VirtualBox GUI when booting the machine
  #   vb.gui = true
  #
  #   # Customize the amount of memory on the VM:
  #   vb.memory = "1024"
  # end
  #
  # View the documentation for the provider you are using for
more
  # information on available options.
```

```
  # Enable provisioning with a shell script. Additional
provisioners such as
  # Ansible, Chef, Docker, Puppet and Salt are also available.
Please see the
  # documentation for more information about their specific
syntax and use.
  # config.vm.provision "shell", inline: <<-SHELL
  #   apt-get update
  #   apt-get install -y apache2
  # SHELL

  config.vm.define "cliente" do |cliente|
    cliente.vm.box = "ubuntu/xenial64"
    cliente.vm.hostname = "cliente"

    # Network config
    cliente.vm.network "private_network", ip: "192.168.33.10"

    cliente.vm.provider "virtualbox" do |vb|
      # Display the VirtualBox GUI when booting the machine
      vb.gui = false

      # Customize the amount of memory and cpus on the VM:
      vb.memory = "512"
      vb.cpus = 1
    end

    cliente.vm.provision "shell", inline: <<-SHELL
      # Record into the hosts file the ip corresponsing to the
'servidor' vm
      echo 192.168.33.11 servidor >> /etc/hosts
    SHELL
  end

  config.vm.define "servidor" do |servidor|
    servidor.vm.box = "ubuntu/xenial64"
    servidor.vm.hostname = "servidor"
```

```
    # Network config
    servidor.vm.network "private_network", ip: "192.168.33.11"
    servidor.vm.network "forwarded_port", guest: 80, host: 8080,
host_ip: "127.0.0.1"

    servidor.vm.provider "virtualbox" do |vb|
      # Display the VirtualBox GUI when booting the machine
      vb.gui = false

      # Customize the amount of memory and cpus on the VM:
      vb.memory = "512"
      vb.cpus = 1
    end

    servidor.vm.provision "shell", inline: <<-SHELL
      #Install NGINX
      apt-get update
      apt-get install -y nginx

      # Record into the hosts file the ip corresponsing to the
'servidor' vm
      echo 192.168.33.10 cliente >> /etc/hosts
    SHELL
  end

end
```

Alternativamente, se puede bajar el archivo desde el siguiente link.

3) Una vez guardado el archivo, abrir una consola y navegar hasta el directorio donde creaste el vagrantfile, A continuación, ejecutar:

```
vagrant up
```

4) En la terminal desde donde estabas ejecutando los comandos de Vagrant, ejecutar el siguiente comando para ingresar en la máquina virtual:

```
vagrant ssh cliente
```

5) Abrir una segunda terminal en tu computadora, navegar hasta la carpeta donde tenías el vagrantfile y ejecutar el siguiente comando:

```
vagrant ssh servidor
```

6) Estando ya dentro del contexto de la virtual machine llamada servidor, ejecutar el siguiente comando para iniciar el analizador de red 'tcpdump':

```
sudo tcpdump -X icmp -i enp0s8
```

Nota: Es posible que la placa de red de tu máquina virtual no se llame igual que la que se muestra en el comando anterior resaltada en rojo. Si es así, puede que recibas un error similar al siguiente:

```
tcpdump: enp0s8: SIOCETHTOOL(ETHTOOL_GET_TS_INFO) ioctl
failed: No such device
```

En caso de ser de esta manera, utilizar uno de los siguientes comandos para identificar los nombres de las distintas placas de red que tiene tu VM.

```
ifconfig
ip link
```

Tip: Si tenés más de una placa de red y no sabés cuál es en la que deberías hacer escuchar a 'tcpdump', mirar el vagrantfile para identificar la IP asignada a la máquina virtual y así poder encontrar la placa de red a la que deberías hacer referencia en el comando tcpdump.

7) En la terminal que tiene el contexto fijado en el cliente, ingresar el siguiente comando. Reemplazar el texto en rojo con la dirección IP o el nombre de la otra máquina.

```
ping -c 1 <reemplazar>
```

8) En la consola del servidor observar lo que pasó. Y presionar 'CTRL+C' para finalizar la ejecución de tcpdump.

Ejercicio 2 - De forma individual, ejecutamos los siguiente pasos sin cerrar las consolas y con las sesiones de SSH establecidas contra la máquina 'cliente' y la máquina 'servidor':

1) Estando ya dentro del contexto de la Virtual Machine llamada servidor, ejecutar el siguiente comando para iniciar el analizador de red 'tcpdump':

```
sudo tcpdump -nnvvXSs 1514 -i enp0s8 'port 80'
```

2) En la terminal que tiene el contexto fijado en el cliente, ingresar el siguiente comando. Reemplazar el texto en rojo con la dirección IP o el nombre de la otra máquina.

```
curl http://<reemplazar>
```

3) En la consola del servidor observar lo que pasó. Y presiona 'CTRL+C' para finalizar la ejecución de tcpdump.

Una vez terminada la práctica, ejecutar el siguiente comando para borrar las máquinas virtuales desde el directorio en el que se encuentra el vagrantfile:

```
vagrant destroy --force
```

Con toda la mesa de trabajo debatir sobre las siguientes preguntas y contestarlas en conjunto:

- Tanto para el ejercicio 1 como para el ejercicio 2, describir con tus palabras lo que acaban de hacer.
- Explicar para qué sirven cada uno de los siguientes conceptos: tcpdump, curl, ping y el protocolo ICMP.
- En el caso del ejercicio 1, ¿cuántos paquetes capturó tcpdump? ¿cómo describirían lo que ves en la máquina que obra de servidor como resultado de tcpdump?
- En el caso del ejercicio 2, ¿identificaron el paquete en el que el servidor transfiere el contenido HTML a curl? ¿Por qué hay más paquetes capturados por tcpdump más allá del que transmite el HTML?