OS-Management
Networked Project

Author: Severin Mikesch

Datum: 10.06.2020

1 Introduction

This is a simple guide to show the steps needed to set up virtual machines in Proxmox, install an SSH daemon and a web-service using NGINX. Furthermore, a VPN connection between two VMs should be established and a firewall should be set up.
The following tools and environments will be used:

- Proxmox
- OpenSSH
- Ubuntu server 20.04 LTS
- tinc
- nginx
- iptables

Disclaimer: when using nano as a file editor you can save and close by pressing CTRL+X and then Y.

2 Setup of VMs in Proxmox

Proxmox VE (Virtual Environment) is an open-source solution for enterprise virtualization utilizing the KVM hypervisor running on Debian. I will skip the setup of the Proxmox server, as it is a rather long and tedious process. A Proxmox server has already been set up in the past.
At first we shall upload the ISO of Ubuntu server to the Proxmox server. This is required to provide the necessary boot-medium in order to start our VMs on the server.

2.1 VM Configuration

Before setting up the two VMs we have to make sure that there is a network bridge configured. This bridge needs access to the network from where the services will later be accessed.
We can leave the default settings provided by Proxmox when creating the VMs. We just have to define following things:

- ISO Image: ubuntu server
- Disk Size: 10GB
- CPU Cores: 2
- RAM: 2048 MB
- Network Bridge: your LAN Bridge

Now the settings can be confirmed and the VMs be started.

2.2 Ubuntu server installation on VM1

After the first VM was started an installer will guide you through the process.

2.2.1 Language selection

Select an appropriate language. We will select english.

2.2.2 optional installer upgrade

When prompted with an optional installer upgrade this should be accepted. The installer will continue here with the newest version.

2.2.3 Keyboard layout

Select the appropriate keyboard layout. We will use the German (Austria) layout.

2.2.4 Network connections

Here the previously selected LAN Bridge should be shown. Configure a static IPv4 address. In this example the address 192.168.1.91/24 will be used.

2.2.5 Proxy configuration

If needed a proxy can be defined here. In this example this field will be left blank.

2.2.6 Archive mirror configuration

If needed a different archive mirror can be selected. This example will use the default one.

2.2.7 Storage configuration

Here the previously defined storage should be shown. In this example the entire disk will be used for the VM. Confirm the selection two times to start writing to the disk.

2.2.8 Profile setup

Enter your preferred login credentials and give your VM a name. In this example it will be named virtualmachine1.

2.2.9 SSH setup

As we need a ssh daemon on this VM we will install the openssh server here, although we will not import any ssh identities.

2.2.10 Server snaps

If needed special services can simply be installed from here. This example will not need any.

2.2.11 Installation

Wait for the installation to finish and then reboot.

2.3 Ubuntu server installation on VM2

Follow subsection 2.2 a second time with some minor differences:

- subsubsection 2.2.4: use address 192.168.1.92/24.
- subsubsection 2.2.8: Change login credentials accordingly and change the servers name to virtualmachine2.
- subsubsection 2.2.9: Skip this as we only need an ssh daemon on one VM.

3 VPN Setup

In this section a VPN connection between the two machines using tinc will be configured.

3.1 tinc installation

Tinc is installed using this command (execute on both machines):

```
sudo apt install tinc
```

3.2 VM1 Setup

First we´ll make a directory for the VPN we want to create (called vmvpn):

```
sudo mkdir -p /etc/tinc/vmvpn/hosts
```

Then a file is created with the basic info for this VM:
Create and open the file with...

```
sudo nano /etc/tinc/vmvpn/tinc.conf
```

and paste this:

```
Name = vm1
AddressFamily = ipv4
Interface = tun0
```

Then the file vm1 is created in the hosts directory containing this:

```
Address = 192.168.1.91
Subnet = 10.0.0.1/32
```

Using this command public and private keys will be generated (when prompting about filenames just press enter to use the defaults):

```
sudo tincd -n vmvpn -K4096
```

Then two files are created which contain a shell script for starting and stopping the VM. The file /etc/tinc/vmvpn/tinc-up contains:

```
#!/bin/sh
ip link set $INTERFACE up
ip addr add 10.0.0.1/32 dev $INTERFACE
ip route add 10.0.0.0/24 dev $INTERFACE
```

The file /etc/tinc/vmvpn/tinc-down contains:

```
#!/bin/sh
ip route del 10.0.0.0/24 dev $INTERFACE
ip addr del 10.0.0.1/32 dev $INTERFACE
ip link set $INTERFACE down
```

Then we have to make these scripts executable:

```
sudo chmod 755 /etc/tinc/vmvpn/tinc-*
```

3.3 VM2 Setup

First we´ll make a directory for the VPN we want to create (called vmvpn):

```
sudo mkdir -p /etc/tinc/vmvpn/hosts
```

Then a file is created with the basic info for this VM:
Create and open the file with...

```
sudo nano /etc/tinc/vmvpn/tinc.conf
```

and paste this:

```
Name = vm2
AddressFamily = ipv4
Interface = tun0
ConnectTo = vm1
```

Because this config contains the ConnectTo, the vm2 will connect to vm1, whereas vm1 will only listen to incoming connections.

Then the file vm2 is created in the hosts directory containing this:

```
Subnet = 10.0.0.2/32
```

Using this command public and private keys will be generated (when prompting about filenames just press enter to use the defaults):

```
sudo tincd -n vmvpn -K4096
```

Then two files are created which contain a shell script for starting and stopping the VM. The file /etc/tinc/vmvpn/tinc-up contains:

```
#!/bin/sh
ip link set $INTERFACE up
ip addr add 10.0.0.2/32 dev $INTERFACE
ip route add 10.0.0.0/24 dev $INTERFACE
```

The file /etc/tinc/vmvpn/tinc-down contains:

```
#!/bin/sh
ip route del 10.0.0.0/24 dev $INTERFACE
ip addr del 10.0.0.2/32 dev $INTERFACE
ip link set $INTERFACE down
```

Then we have to make these scripts executable:

```
sudo chmod 755 /etc/tinc/vmvpn/tinc-*
```

3.4 Key exchange

On VM2 send and retrieve the host files for the corresponding hosts.

Copy the host files on vm1 to a directory accessibly by ssh:

```
sudo cp /etc/tinc/vmvpn/hosts/vm1 /tmp
```

Then exchange the host files using vm2:

```
sudo scp /etc/tinc/vmvpn/hosts/vm2 <loginname>@192.168.1.91:/tmp
sudo scp <loginname>@192.168.1.91:/tmp/vm1 /etc/tinc/vmvpn/hosts/
```

Then remove the temporary file vm1 on vm1 and transfer vm2 host file to it´s final location:

```
sudo rm /tmp/vm1
sudo mv /tmp/vm2 /etc/tinc/vmvpn/hosts/vm2
```

3.5 Start tinc on startup

To start tinc on startup we have to add it to the systemmanager (both VMs):

```
sudo systemctl enable tinc@vmvpn
```

4 NGINX Setup

In this chapter we will install an nginx server on both VMs. One will act as a forwarding proxy to the other server.

4.1 NGINX Installation

Install the needed packages with this command:

```
sudo apt install nginx -y
```

If needed, enter your user password.
To start your nginx server enter this command:

```
sudo systemctl start nginx
```

Check the status of the server with this command:

```
sudo systemctl status nginx
```

To start your nginx server along with the VM execute following command:

```
sudo systemctl enable nginx
```

4.2 Webpage configuration on VM1

There is a default page created on install, so we don´t need to do anything more here.

However we can rename the file /var/www/html/index.nginx-debian.html to /var/www/html/index.html with the command:

```
sudo mv /var/www/html/index.nginx-debian.html
/var/www/html/index.html
```

And we can change the file to make clear that we are the server on VM1. Using this command ...

```
sudo nano /var/www/html/index.html
```

... add some text the file in order to recognize it later.

4.3 NGINX proxy configuration on VM2

We need to rewrite the configuration file of the nginx server site to forward the traffic to vm1. Use this command to edit the file:

```
sudo nano /etc/nginx/sites-enabled/default
```

Rewrite the file so the uncommented parts looks like this:

```
server {
listen 80 default_server;
listen [::]:80 default_server;

location / {
proxy_pass http://10.0.0.1;
}
}
```

To reflect the changes restart your nginx server:

```
sudo systemctl restart nginx
```

5 SSH Daemon configuration auf VM1

The open ssh server must be configured to listen on port 41500, allow only unpriviliged users and to use only public key authentication.

After logging in with a sudo user (user with access to root actions) open the config file with the command (enter the password if prompted):

```
sudo nano /etc/ssh/sshd_config
```

5.1 Port Configuration

Change following line to change the listening port of the ssh server.

```
#Port 22
```

to

```
Port 41500
```

In order to reflect the changes the server must be restarted with the command:

```
sudo service ssh restart
```

5.2 User access configuration

Change following line to allow only unpriviliged users to access to the server.

```
#PermitRootLogin prohibit-password
```

to

```
PermitRootLogin no
```

In order to reflect the changes the server must be restarted with the command:

```
sudo service ssh restart
```

5.3 Enable only public key authentication

Change following line to disallow password authentication in favour of public key authentication for the ssh server.

```
#PasswordAuthentication yes
```

to

```
PasswordAuthentication no
```

and

```
#PubkeyAuthentication yes
```

to

```
PubkeyAuthentication yes
```

In order to reflect the changes the server must be restarted with the command:

```
sudo service ssh restart
```

6 Firewall Setup

Iptables is already installed on these machines so we are already ready to configure it.

6.1 Custom rules

To allow access to the ssh server on vm1 the execute (execute on vm1):

```
sudo iptables -A INPUT -p tcp --dport 41500 -j ACCEPT
```

To allow access to the services on the VMs (execute on both VMs):

```
sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

To allow tinc VPN connections (execute on both VMs):

```
sudo iptables -A INPUT -p tcp --dport 655 -j ACCEPT
sudo iptables -A INPUT -p udp -m udp --dport 655 -j ACCEPT
```

6.2 Fallback rule

To block all traffic not matching the configured rules there must be a default DROP policy (both VMs):

```
sudo iptables -P Input DROP
```