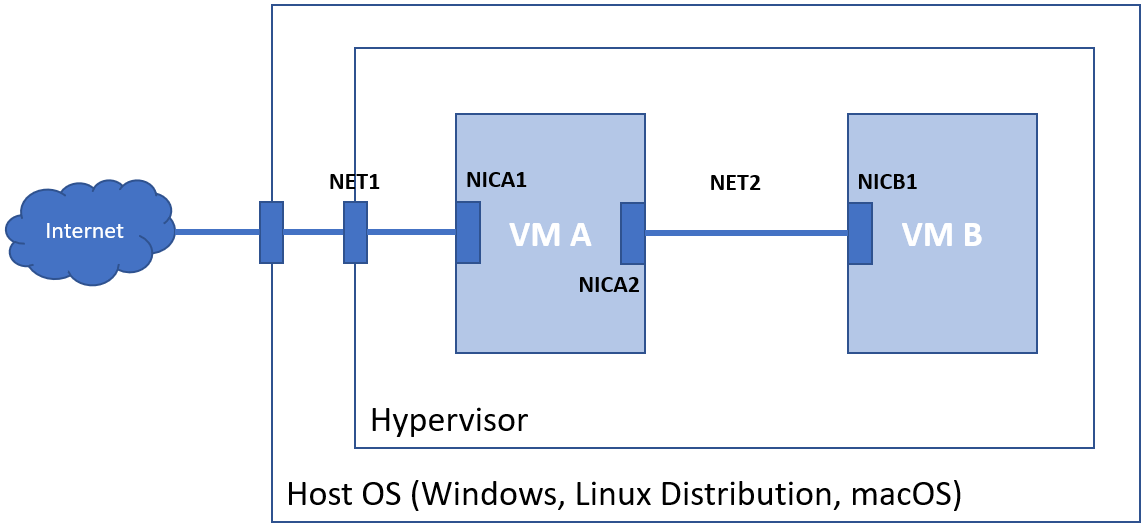
# Solution M2: Network and System Security (openSUSE)

For both tasks we assume that we are working in the following infrastructure:



Additional assumptions (can be seen on the picture above):

* **VM A** has **two** NICs (**NICA1** and **NICA2**):
  + has connectivity to the Internet
  + first (**NICA1**), connected via external/bridged/NAT mode to the external world (**NET1**)
    - network settings (IP address, subnet mask, gateway, and DNS) match the connected network
  + second (**NICA2**), connected in an internal/private/host-only mode network (**NET2**)
    - with static IP address in the same network as the only NIC (**NICB1**) of the VM B
    - for example, ***IP=192.168.200.1*** *and* ***MASK=255.255.255.0***
* **VM B** has **one** NIC (**NICB1**):
  + connected in an internal/private/host-only mode network (**NET2**)
  + the only NIC (**NICB1**) is set with static IP address in the same network as the second NIC (**NICA2**) of the **VM A** and with gateway set to the IP address of the second NIC (**NICA2**) of **VM A** and DNS set to a public one, for example 8.8.8.8
  + for example, ***IP=192.168.200.2, MASK=255.255.255.0, GW=192.168.200.1,*** *and* ***DNS=8.8.8.8***

## Task 1

**Challenge:** Research and implement two-node network (one machine with two NICs and the second with one) with NAT capabilities based on **firewalld**/**ufw**

**Solution (firewalld):**

Before we start to tackle the challenge, we must validate our setup by

* On **VM A**
  + Can successfully ping publicly available resource by IP address, for example **8.8.8.8**
  + Can successfully ping publicly available resource by name, for example **abv.bg**
  + Can successfully ping **NICB1** of **VM B**, for example **192.168.200.2**
* On **VM B**
  + Can successfully ping **NICA2**, for example **192.168.200.1**

We must continue with the rest of the steps, only if the above checks are okay

### VM A

Then, we must make sure that the **firewalld** is working

**systemctl status firewalld**

If not, we must enable it on boot and start it

Next, we must check active zones and where our adapters aresudo

**sudo firewall-cmd --get-active-zones**

Then, we must move the **first or external NIC** (NICA1) to the **external** zone

**sudo firewall-cmd --zone=external --change-interface=eth0 --permanent**

Finally, we must move the **second or internal NIC** (NICA2) to the **internal** zone

**sudo firewall-cmd --zone=internal --change-interface=eth1 --permanent**

If our network configuration is managed via **wicked**, we must reload it

**sudo wicked ifreload all**

That is all, that was required here. This turned our VM A to a router (of course, with NAT enabled)

### VM B

By now, we should be done. Check that you

* Can successfully ping publicly available resource by IP address, for example **8.8.8.8**
* Can successfully ping publicly available resource by name, for example **abv.bg**

## Task 2

**Challenge:** Research and implement two-node network (one machine with two NICs and the second with one) with NAT capabilities based on **nftables**

**Solution:**

Before we start to tackle the challenge, we must stop and disable the firewall

**sudo systemctl disable --now firewalld**

And then, validate our setup by

* On **VM A**
  + Can successfully ping publicly available resource by IP address, for example **8.8.8.8**
  + Can successfully ping publicly available resource by name, for example **abv.bg**
  + Can successfully ping **NICB1** of **VM B**, for example **192.168.200.2**
* On **VM B**
  + Can successfully ping **NICA2**, for example **192.168.200.1**

We must continue with the rest of the steps, only if the above checks are okay

### VM A

Open the **nftables** configuration file

**sudo vi /etc/sysconfig/nftables.conf**

And append the following lines to the end

**table ip nat {**

**chain PREROUTING {**

**type nat hook prerouting priority filter; policy accept;**

**}**

**chain POSTROUTING {**

**type nat hook postrouting priority srcnat; policy accept;**

**oifname "eth0" masquerade**

**}**

**}**

Save and close the file

Load the rules with

**sudo nft -f /etc/sysconfig/nftables.conf**

Check that the rules are there

**sudo nft list ruleset**

Check the **ip\_forward** flag

**cat /proc/sys/net/ipv4/ip\_forward**

Turn it **on** (**1**) if it is **off** (**0**)

**echo "1" | sudo tee /proc/sys/net/ipv4/ip\_forward**

### VM B

By now, we should be (almost) done. Check that you

* Can successfully ping publicly available resource by IP address, for example **8.8.8.8**
* Can successfully ping publicly available resource by name, for example **abv.bg**

### VM A

There is something else. We must make sure that the changes will persist during reboot

Create a new file to adjust the flag

**sudo vi /etc/sysctl.d/98-ip\_forward.conf**

Enter the following

**net.ipv4.ip\_forward=1**

Save and close the file

Apply the changes with

**sudo sysctl –system**

There is one more thing, **nftables** rules are going to disappear on reboot

Under **openSUSE** there is not any official solution on this yet

We can do a little cheat – adjust the configuration file a bit

**sudo vi /etc/sysconfig/nftables.conf**

To match the following

**flush ruleset**

**table ip nat {**

**chain PREROUTING {**

**type nat hook prerouting priority filter; policy accept;**

**}**

**chain POSTROUTING {**

**type nat hook postrouting priority srcnat; policy accept;**

**oifname "eth0" masquerade**

**}**

**}**

Then create a **systemd** service

**sudo vi /etc/systemd/system/mynft.service**

With the following content

**# /etc/systemd/system/mynft.service**

**#**

**[Unit]**

**Description=Load NFTables rules on boot**

**[Service]**

**Type=oneshot**

**ExecStart=/bin/sh -c "nft -f /etc/sysconfig/nftables.conf"**

**[Install]**

**WantedBy=multi-user.target**

Save and close the file

Enable the service

**sudo systemctl enable mynft.service**

Start the service

**sudo systemctl start mynft.service**

Check if the rules are there

**sudo nft list ruleset**

Everything should be just fine now, even after reboot 😊