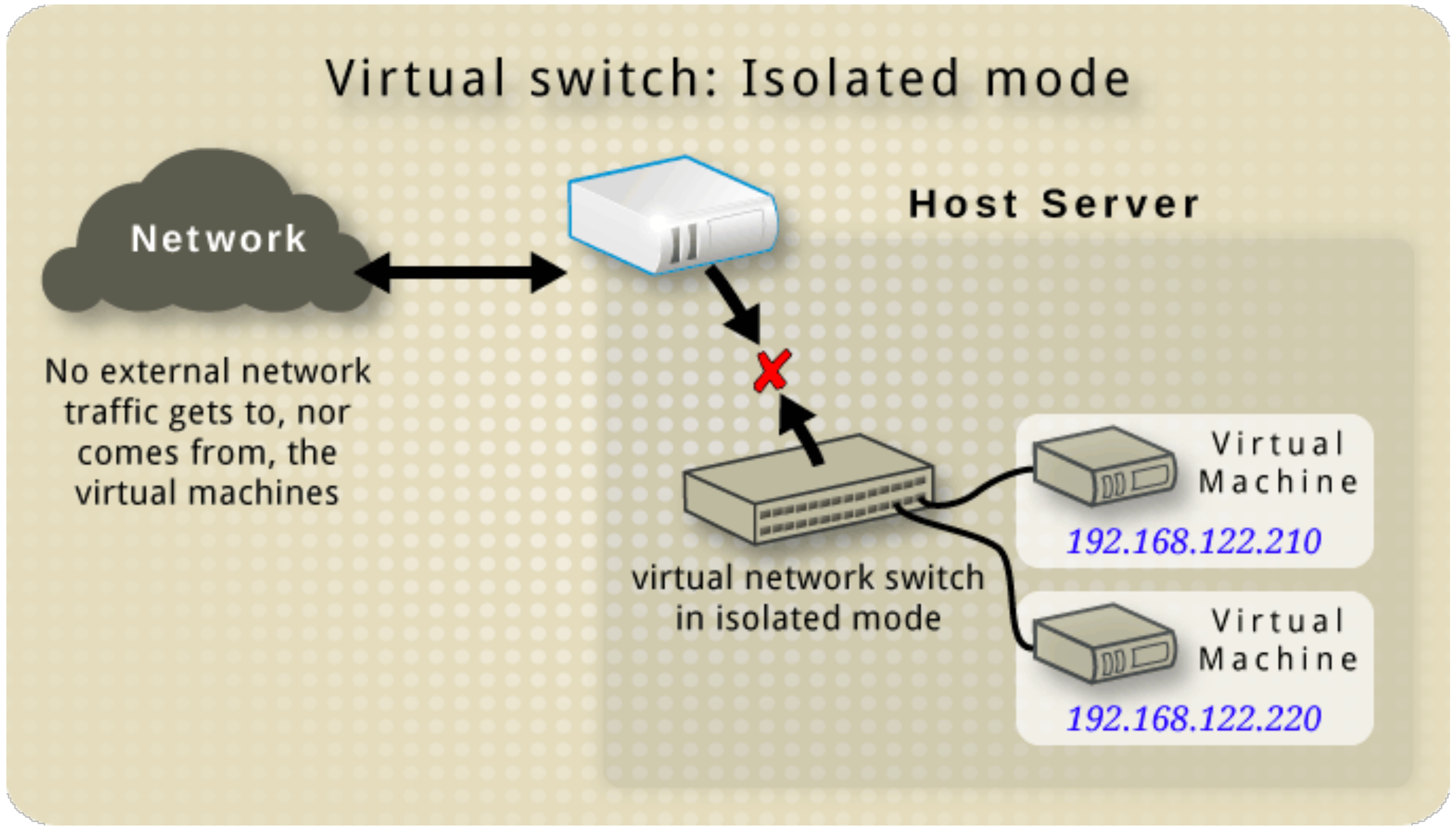
Network Modes: isolated


Virtual switch: Isolated mode
Network
Host Server
No external network traffic gets to, nor comes from, the virtual machines
virtual network switch in isolated mode
Virtual Machine
192.168.122.210
Virtual Machine
192.168.122.220

Test: isolated Mode

CHECK VM

- `virsh edit CirrosA`
- search for section:

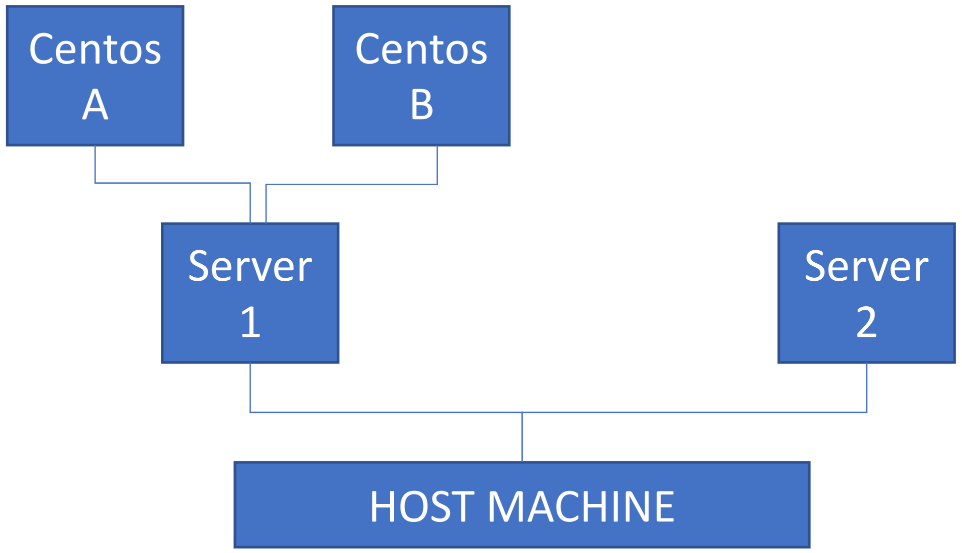
```
<interface type='network'>
     <source network='default'/>
     <mac address='00:16:3e:1a:b3:4a'/>
  </interface>
```

REMOVE NAT

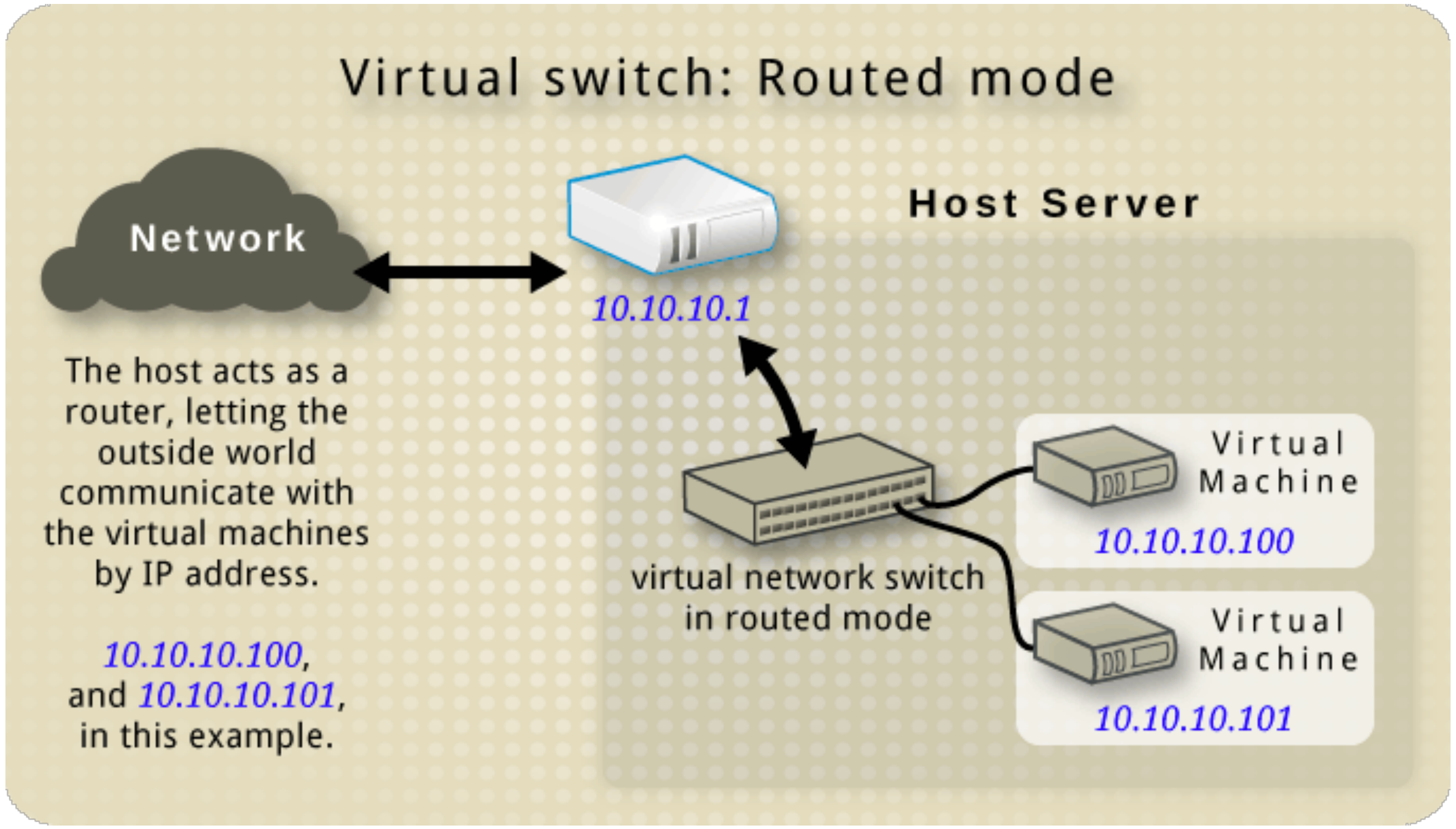
- `# sudo virsh net-edit default`
- //remove <forward .. section
- `# sudo virsh net-destroy default ; sudo virsh net-start default`
- `# service libvirtd restart`
- `# sudo virsh console vm_name`

Test: isolated Mode

- Ping: A <-> B
- Ping: A,B <-> Server 1
- Ping: A,B -> Server 2
- Ping: Server 2 -> A,B



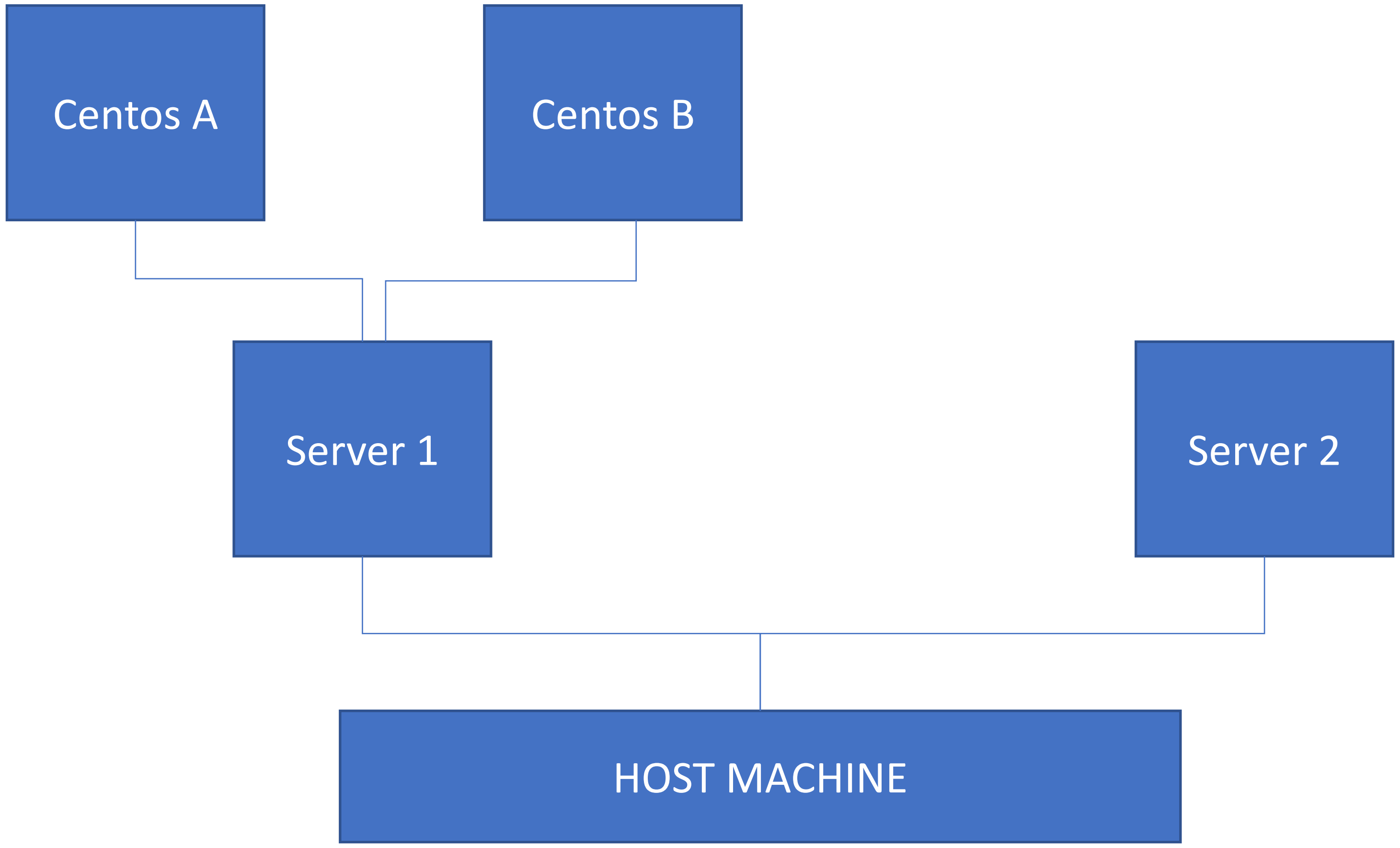
Network Modes: routed


Virtual switch: Routed mode
Network
10.10.10.1
Host Server
The host acts as a router, letting the outside world communicate with the virtual machines by IP address.
10.10.10.100, and 10.10.10.101, in this example.
virtual network switch in routed mode
Virtual Machine
10.10.10.100
Virtual Machine
10.10.10.101

Configure Route mode

- `# sudo virsh net-edit default`
- //ADD section <forward mode="route"/>
- `# sudo virsh net-destroy default`
- `# sudo virsh net-start default`
- `# service libvirtd restart`
- `# sudo virsh console CirrosA`

Does it work?



Configure a static route

At this point the VM can reach other networks but cannot ping, as there is no route for return packets. We should adjust routing at various points of the network. For the sake of this example, we can place a static route pointing to the host server, as follows.

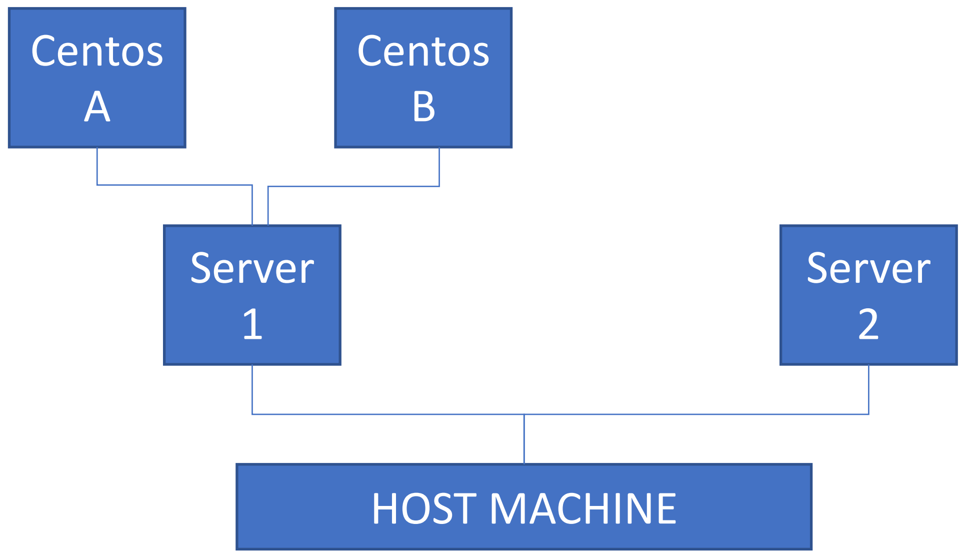
```
sudo ip -4 route add 192.168.122.0/24 via 192.168.56.3
```

You can browse the routes and delete any line using the following commands:

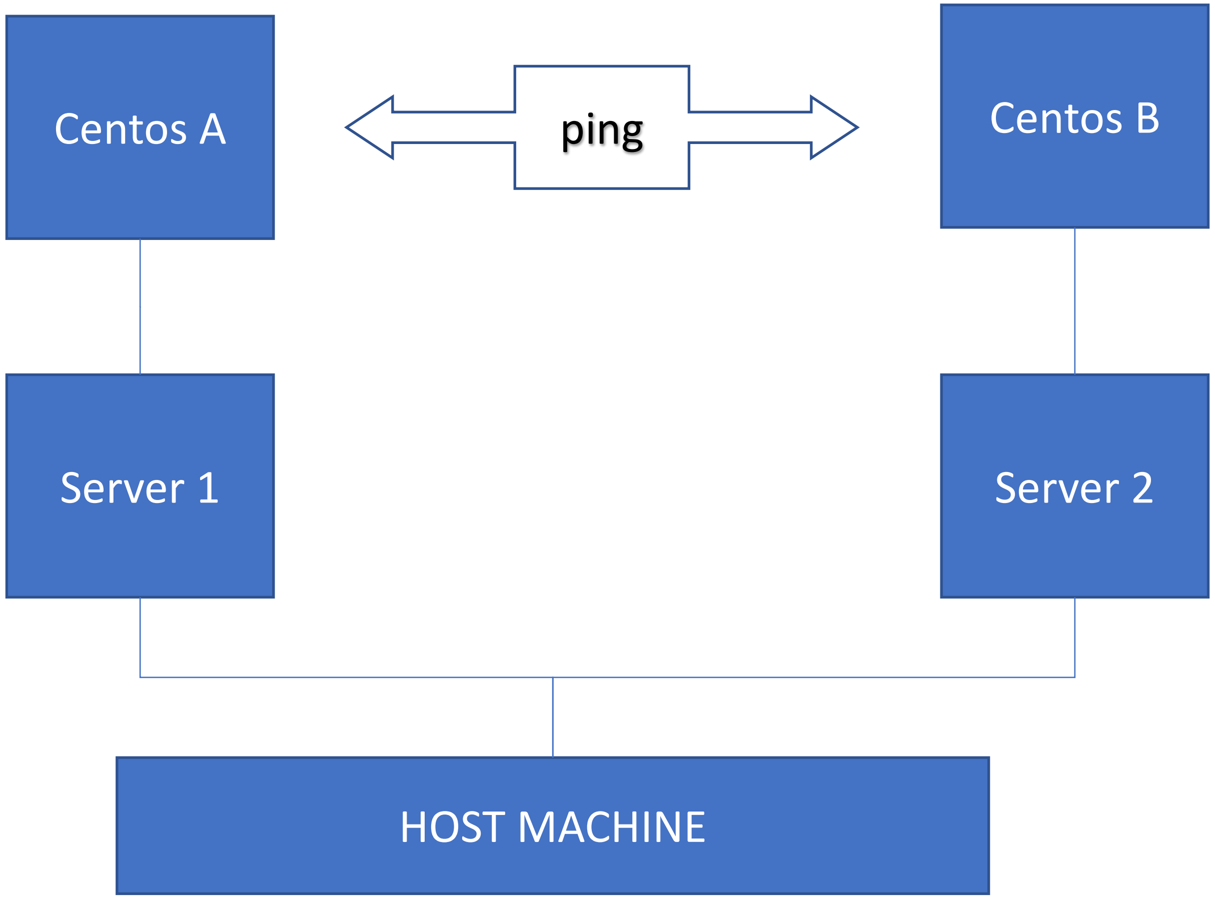
```
ip route list
sudo ip route del 192.168.122.0/24 via 192.168.56.3 dev eth0
```

Test: Routed

- Ping: A <-> B
- Ping: A,B <-> Server 1
- Ping: A,B -> Server 2
- Ping: Server 2 -> A,B



Exercise



Step 1: differentiating network addresses

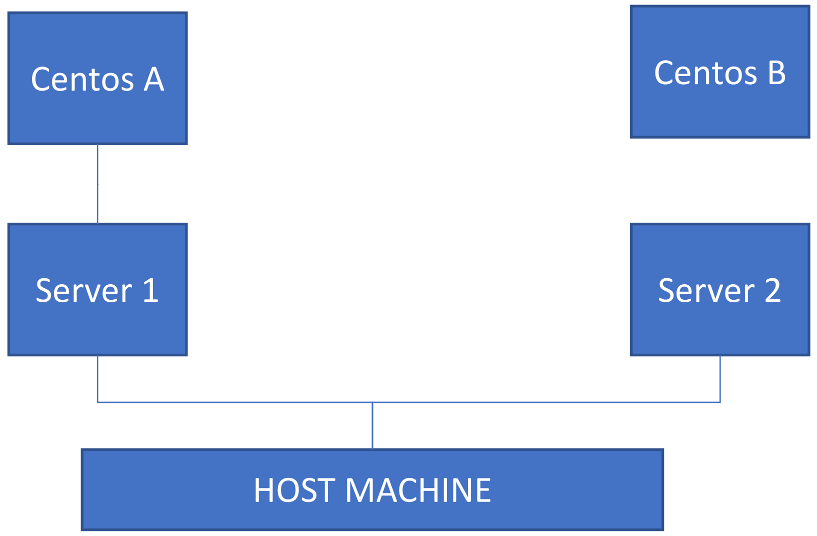
- Configure network on Server 2 (for Cirros B) to use a different set of addresses.

```
<ip address='192.168.120.1' netmask='255.255.255.0'>
    <dhcp>
      <range start='192.168.120.2' end='192.168.120.254'/>
    </dhcp>
</ip>
```

- To renew IP address on cirros

```
sudo /sbin/cirros-dhcpc down eth0
sudo /sbin/cirros-dhcpc up eth0
```

Step 2: configure routing

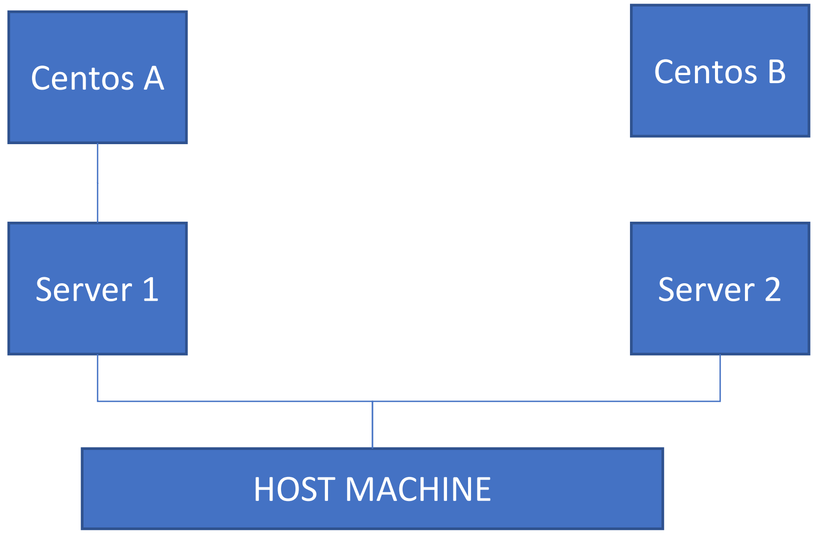


- On CentosA

```
sudo ip -4 route add 192.168.120.0/24 via 192.168.122.___
```

- Configure CentosB similarly

Step 2: configure routing (2)



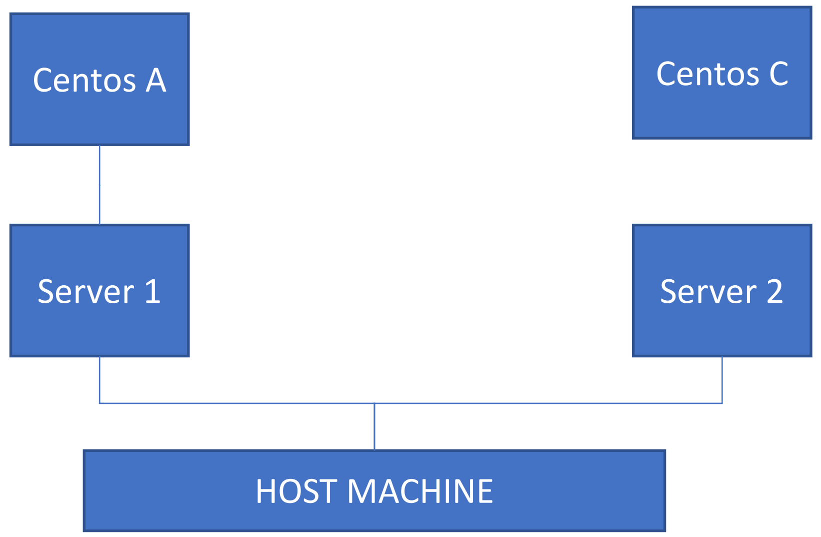
- On Server1

```
sudo ip -4 route add 192.168.120.0/24 via 192.168.56.___
```

- Configure Server2 similarly.

Test: Routed

- Ping: A <-> C
- Ping: A,C <-> Server 1
- Ping: A,C -> Server 2
- Ping: Server 2 -> A,C



Why “leaving” linux bridging

Guest (VM) networking in KVM has traditionally been done using linux bridging

Pros:

- good performance
- simple to configure and manage

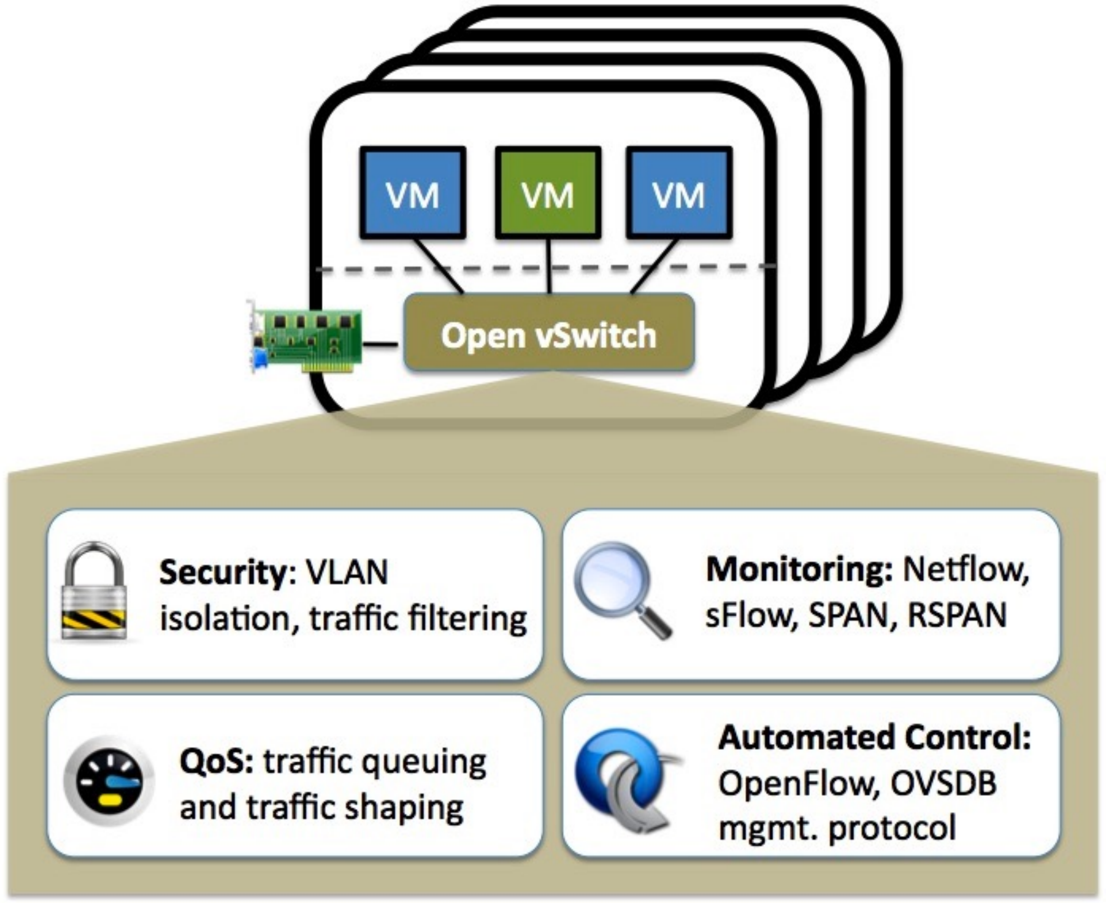
Cons:

- It was not originally designed for virtual networking
- It does not support tunneling protocols
- not an openflow compatible device (no SDN support to date)

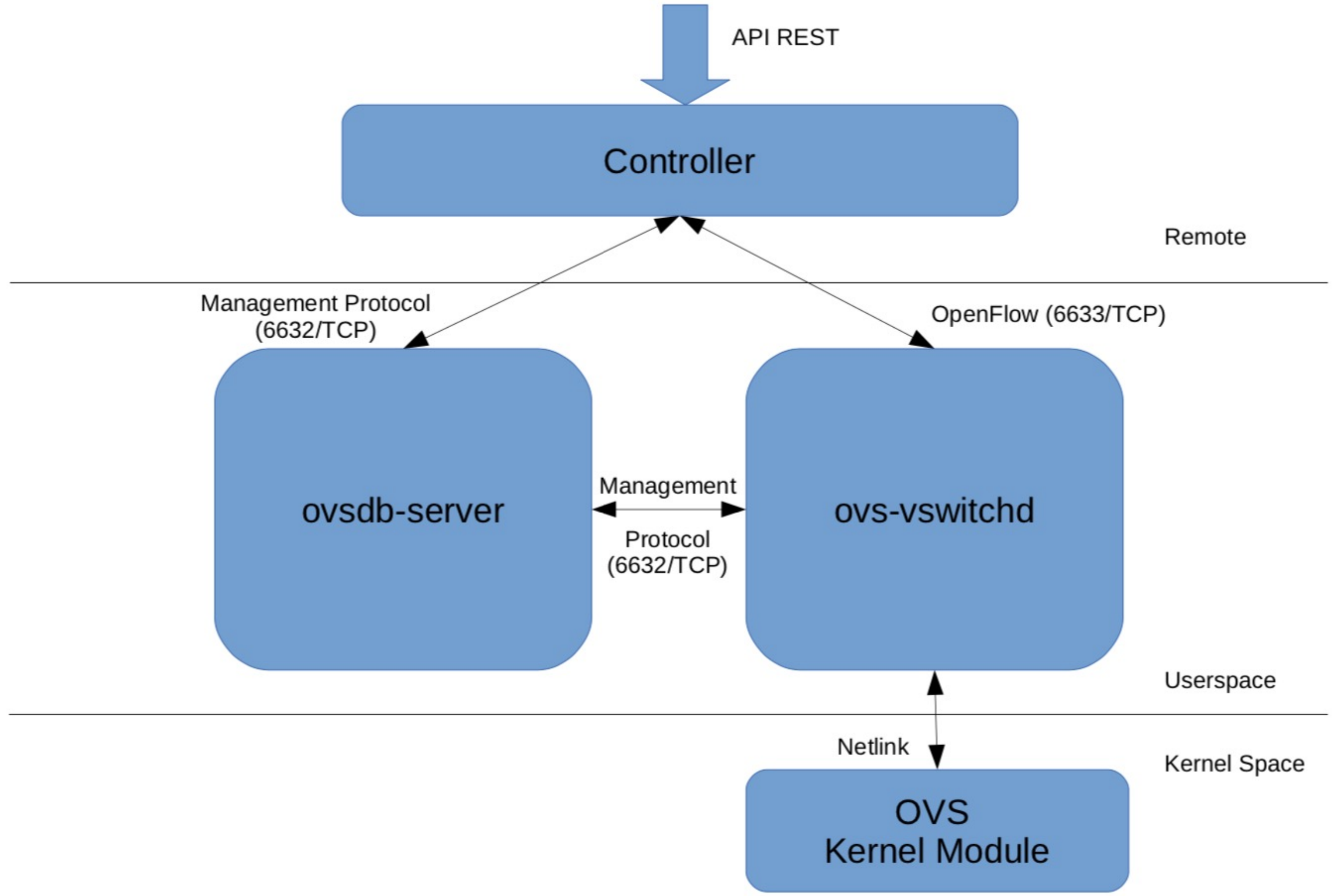
OvS
Open vSwitch

- Open source (apache 2.0 licensing)
- Written in C

- Thought for multi-server virtualization
- support "remote configurability"
- Hardware integration


VM
VM
VM
Open vSwitch
Security: VLAN isolation, traffic filtering
Monitoring: Netflow, sFlow, SPAN, RSPAN
QoS: traffic queuing and traffic shaping
Automated Control: OpenFlow, OVSDB mgmt. protocol

Open vSwitch architecture



Installing OVS

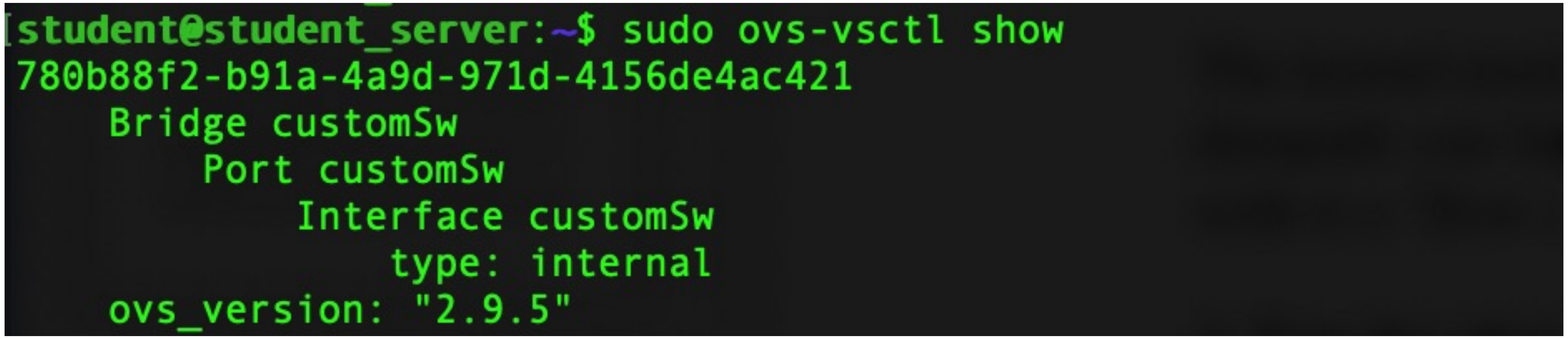
```
sudo apt-get install openvswitch-switch
```

```
sudo service ovs-vswitchd start
```

```
sudo service ovsdb-server start
```

Create a new bridge

- `ovs-vsctl add-br customSw`
- `ovs-vsctl show`



```
[student@student_server:~$ sudo ovs-vsctl show
780b88f2-b91a-4a9d-971d-4156de4ac421
    Bridge customSw
        Port customSw
            Interface customSw
                type: internal
    ovs_version: "2.9.5"
```

Configure VMs to use ovs (libvirt)

```
virsh edit cirrosA
```

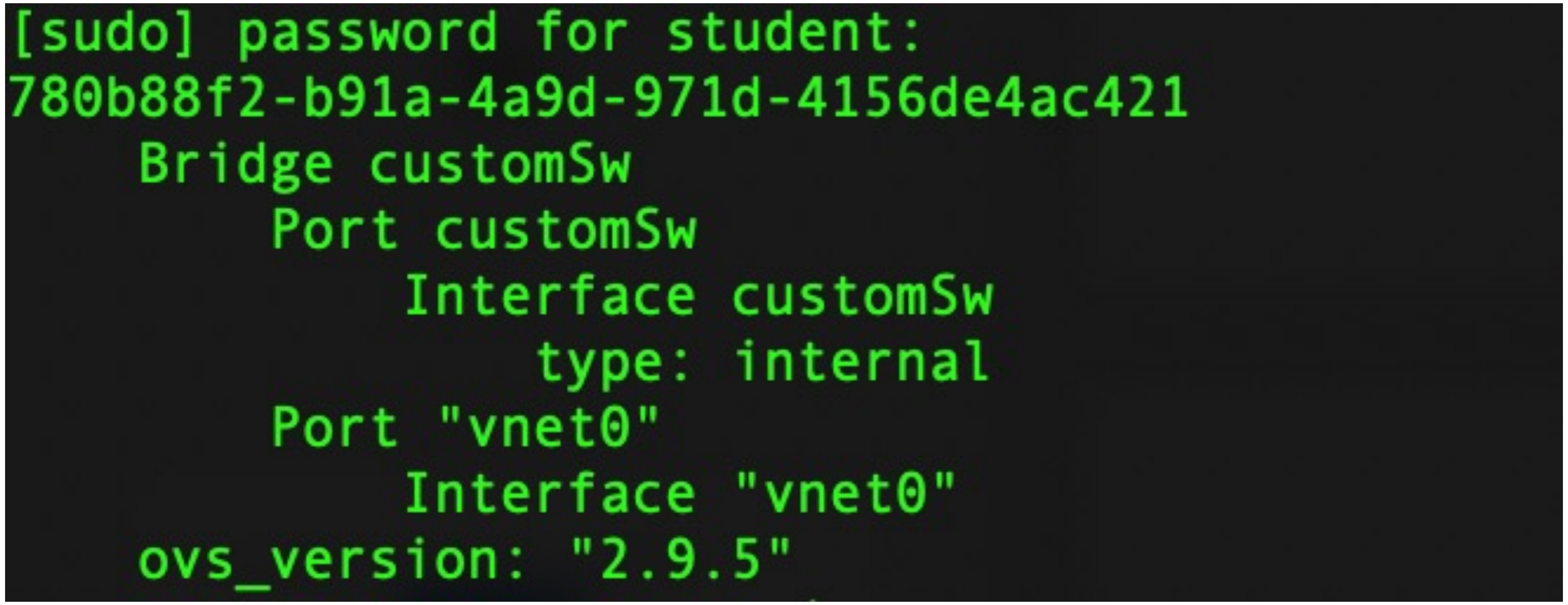
- Change the <interface type='network'> configuration section as follows

```
<interface type='bridge'>
 <mac address='52:54:00:71:b1:b6'/>
 <source bridge=customSw'/>
 <virtualport type='openvswitch'/>
 <address type='pci' domain='0x0000' bus='0x00'
slot='0x03' function='0x0'/>
</interface>
```

Check

- Start cirrosA/cirrosB
- Verify bridge ports

```
ovs-vsctl show
```



```
[sudo] password for student:
780b88f2-b91a-4a9d-971d-4156de4ac421
    Bridge customSw
        Port customSw
            Interface customSw
                type: internal
        Port "vnet0"
            Interface "vnet0"
    ovs_version: "2.9.5"
```

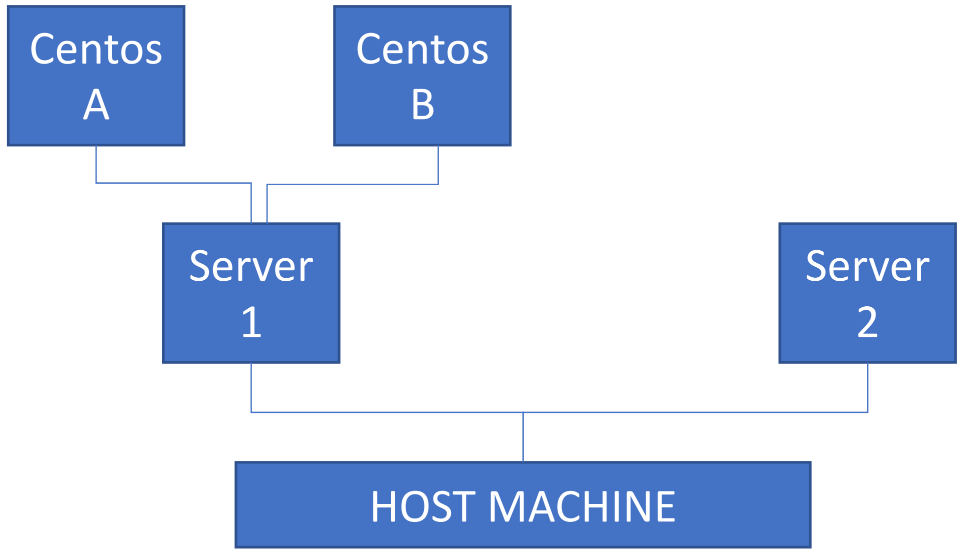
Configure addresses

- statically assign IPs to cirrosA/cirrosB

```
sudo ifconfig eth0 192.168.120.10
```

Test: ovs1

- Ping: A <-> B
- Ping: A,B <-> Server 1
- Ping: A,B -> Server 2
- Ping: Server 2 -> A,B



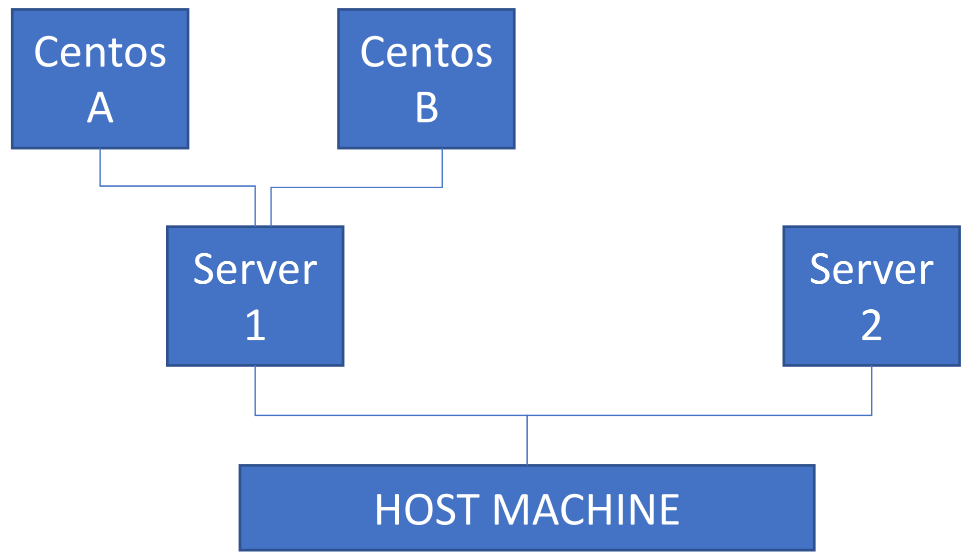
Configure addresses (2)

- To make student_server accessible from cirrosA, we need to give an ip address to one of the internal interfaces of customSw

```
sudo ifconfig customSw 192.168.120.1
```

Test: ovs2

- Ping: A <-> B
- Ping: A,B <-> Server 1
- Ping: A,B -> Server 2
- Ping: Server 2 -> A,B



Configure routing

- On Server 2

```
sudo ip -4 route add 192.168.200.0/24 via 192.168.56.106
```

- On CirrosA/B

```
sudo route add default gw 192.168.200.1 eth0
sudo ip route del 0/0  // to delete the default route
```

- Visualize routes

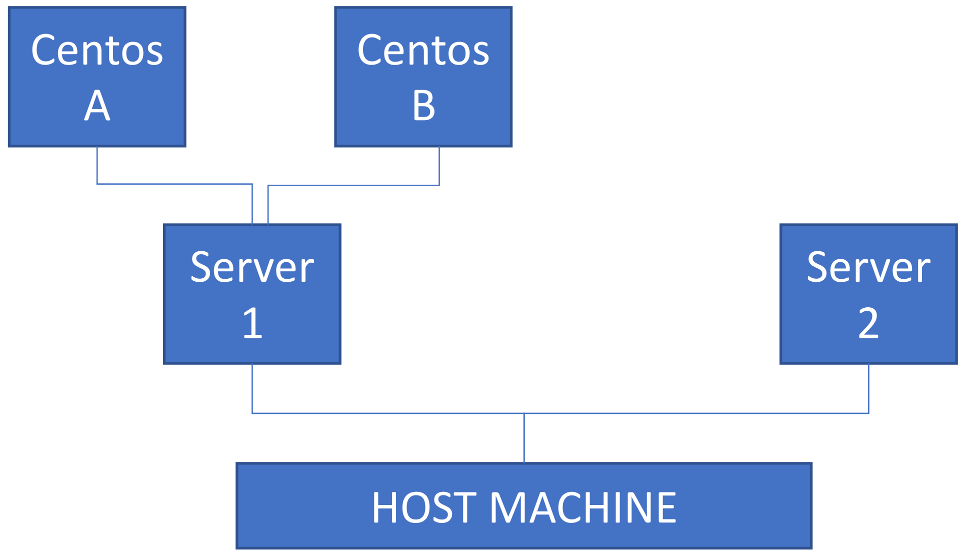
```
route –n
```

- Observe traffic (e.g. on enp0s3 of Server1)

```
tcpdump –i enp0s3 icmp
```

Test: ovs3

- Ping: A <-> B
- Ping: A,B <-> Server 1
- Ping: A,B -> Server 2
- Ping: Server 2 -> A,B



Connecting towards external network

- to allow cirrosA to reach "other" networks, we need to attach the "real" network interfaces of student_server to the bridge.

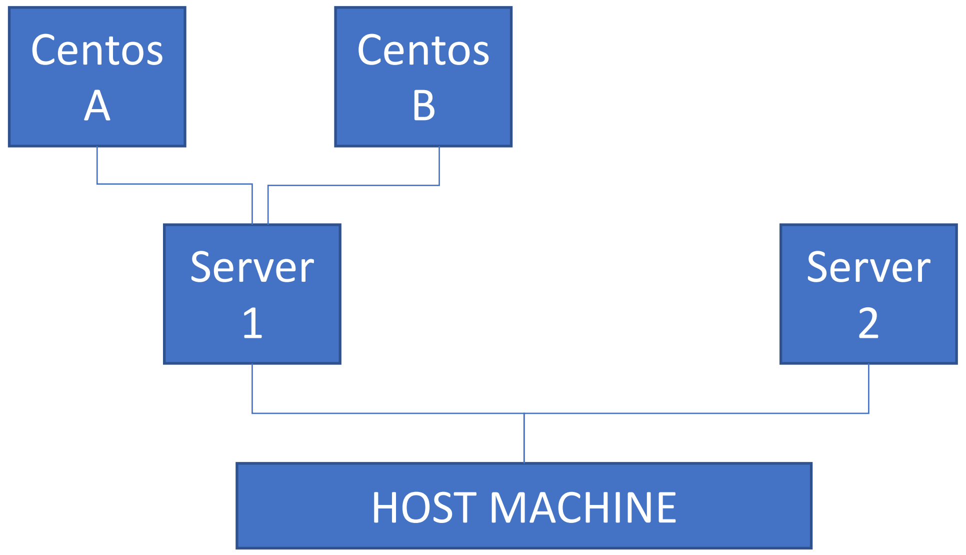
```
ovs-vsctl add-port customSw enp0s8
```

- Update IP addresses (if needed)

```
ifconfig customSw 0.0.0.0/24
dhclient enp0s8
```

Test: ovs4

- Ping: A <-> B
- Ping: A,B <-> Server 1
- Ping: A,B -> Server 2
- Ping: Server 2 -> A,B
- Ping: A,B -> 8.8.8.8
- Ping: server1 -> 8.8.8.8


Centos A
Centos B
Server 1
Server 2
HOST MACHINE