

**8th july**

**-> aws vpc**

**-> openstack neutron**

**Physical connectivity(router,switch etc)**

**In aws eni card(or nat card) and in openstack it is known as vNIC(virtual network card)**

**Both sides should have unique ip address**

**We req a device (switch or router on the bases of networking)**

**Private ip can only connects with private and vice versa**

**C:\Users\user>ipconfig**

**Windows IP Configuration**

**Ethernet adapter Ethernet:**

**Media State . . . . . : Media disconnected**

**Connection-specific DNS Suffix . :**

**Ethernet adapter VirtualBox Host-Only Network:**

**Connection-specific DNS Suffix . :**

**Link-local IPv6 Address . . . . . : fe80::8c48:9d18:b9e9:10ab%10**

**IPv4 Address. . . . . : 192.168.56.1**

**Subnet Mask . . . . . : 255.255.255.0**

**Default Gateway . . . . . :**

**Ethernet adapter VirtualBox Host-Only Network #2:**

**Connection-specific DNS Suffix . :**

**Link-local IPv6 Address . . . . . : fe80::cd99:5eb7:860:f0b1%21**

**IPv4 Address. . . . . : 192.168.99.1**

**Subnet Mask . . . . . : 255.255.255.0**

**Default Gateway . . . . . :**

**Wireless LAN adapter Local Area Connection\* 3:**

**Media State . . . . . : Media disconnected**

Connection-specific DNS Suffix . :

Wireless LAN adapter Local Area Connection\* 4:

Media State . . . . . : Media disconnected

Connection-specific DNS Suffix . :

Wireless LAN adapter Wi-Fi:

Connection-specific DNS Suffix . :

IPv6 Address. . . . . : 2401:4900:463f:a5bd:e98c:fb29:5c6c:dbed

Temporary IPv6 Address. . . . . : 2401:4900:463f:a5bd:d875:839f:6406:7fda

Link-local IPv6 Address . . . . . : fe80::e98c:fb29:5c6c:dbed%12

IPv4 Address. . . . . : 192.168.43.178

Subnet Mask . . . . . : 255.255.255.0

Default Gateway . . . . . : fe80::c8:7ff:fe29:76fe%12

192.168.43.1

C:\Users\user>ping 8.8.8.8

Pinging 8.8.8.8 with 32 bytes of data:

Reply from 8.8.8.8: bytes=32 time=70ms TTL=114

Reply from 8.8.8.8: bytes=32 time=81ms TTL=114

Reply from 8.8.8.8: bytes=32 time=69ms TTL=114

Reply from 8.8.8.8: bytes=32 time=67ms TTL=114

Ping statistics for 8.8.8.8:

Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),

Approximate round trip times in milli-seconds:

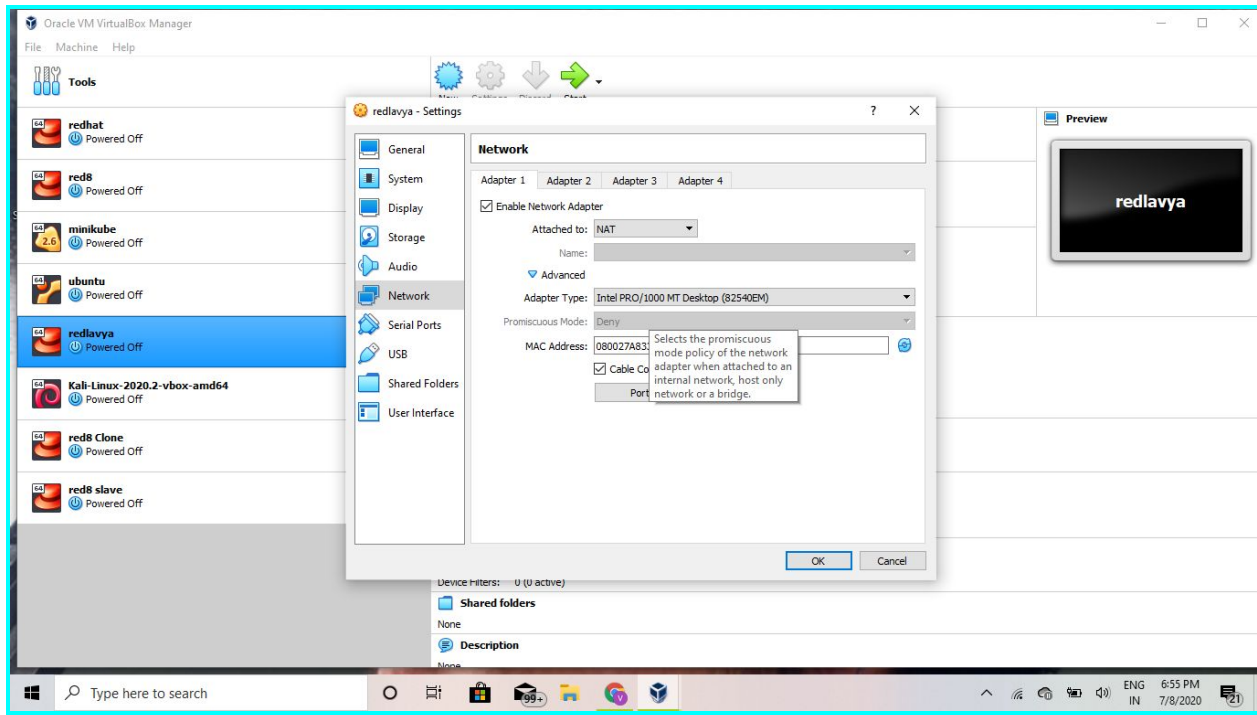
Minimum = 67ms, Maximum = 81ms, Average = 71ms

naas(network as a service)

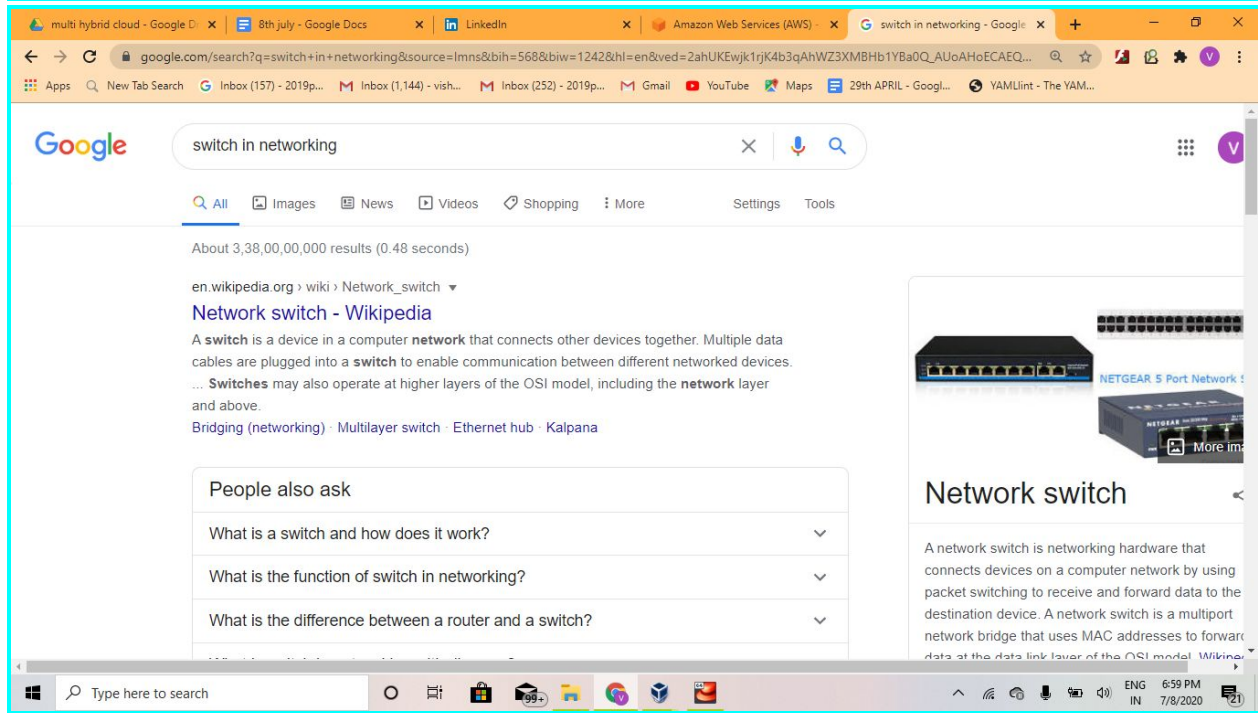
> in aws we use the concept of floating ip

And we can get the random ip from dhcp server

In aws we have eni card



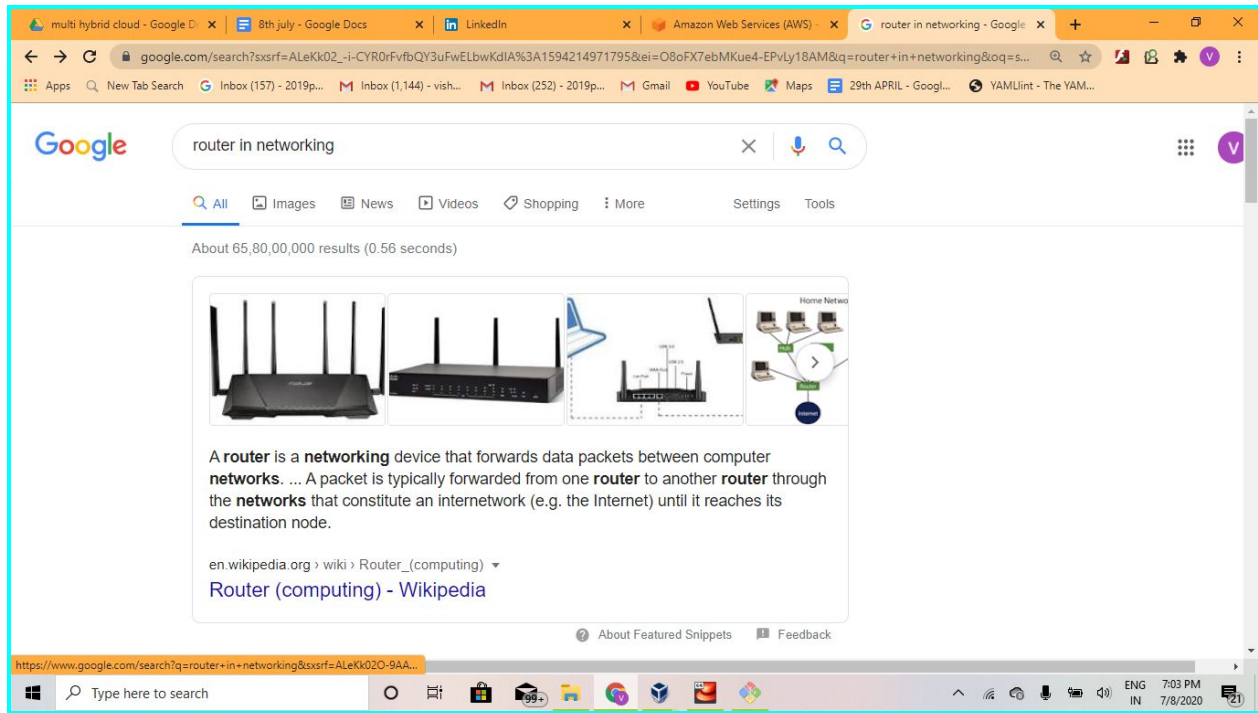
/16 or /24 it is known as cidr or prefix length



**switch depends on the number of ports**  
**And they should belong to same network**



**When we create a switch using a software thn it is known as bridge**



## **IN NETWORKING WORLD WE HAVE OSI WORLD**

- **PHYSICAL**
- **DATA LINK LAYER( switch )**
- **NETWORK LAYER( router )**
- **TRANSPORT LAYER**
- **SESSION LAYER**
- **PRESENTATION LAYER**
- **APPLICATION LAYER**

**Switch works on layer2 and it is also known as l2switch**

**Router work on layer3 and it is also known as layer3switch**

**Mobile hotspot works like a switch and router**

**Hotspot is also known as l3bridge**

**In linux we have a software linuxswitch we have pre conf**

**Or we have open-v-switch**

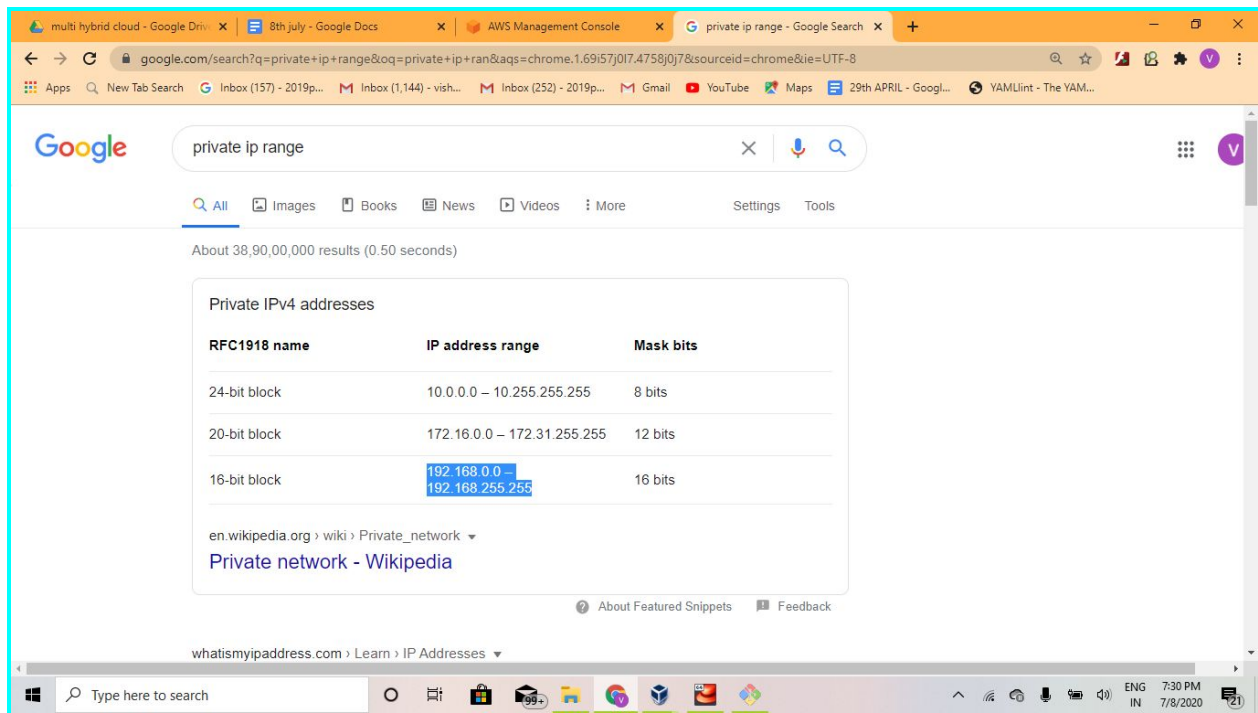
**Router connects with isp**

**IANA range**

**10.0.0.0-10.255.255.255 are consider as a private ip**

**172.16.0.0 – 172.31.255.255**

**192.168.0.0 – 192.168.255.255**



```
[root@localhost ~]# ping 8.8.8.8
```

```
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data:
```

```
64 bytes from 8.8.8.8: icmp_seq=1 ttl=113 time=55.2 ms
```

With the help of nating we can able to ping the public world from private ip

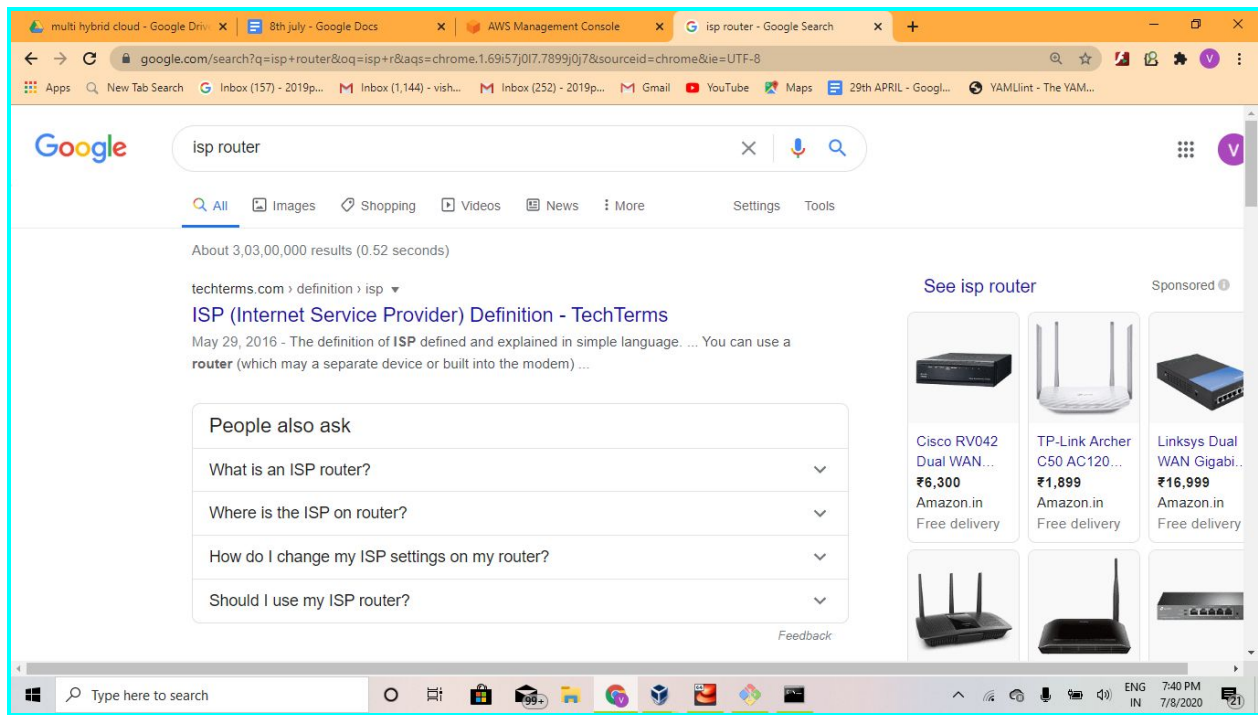
Why we pay for the internet?

> they give us the wire/wireless connection

In modem we have a wanport

In modem they give public ip

lsp only allow public ip



**They block the private ip**

**Yum install iproute**

**We send packets**

**System creates packets and from where the packet is sent it is known as source ip (we have header) and destination ip for destination**

**Data is also known as payload**

**Router removes the private ip to public ip NAT (network address translation)**

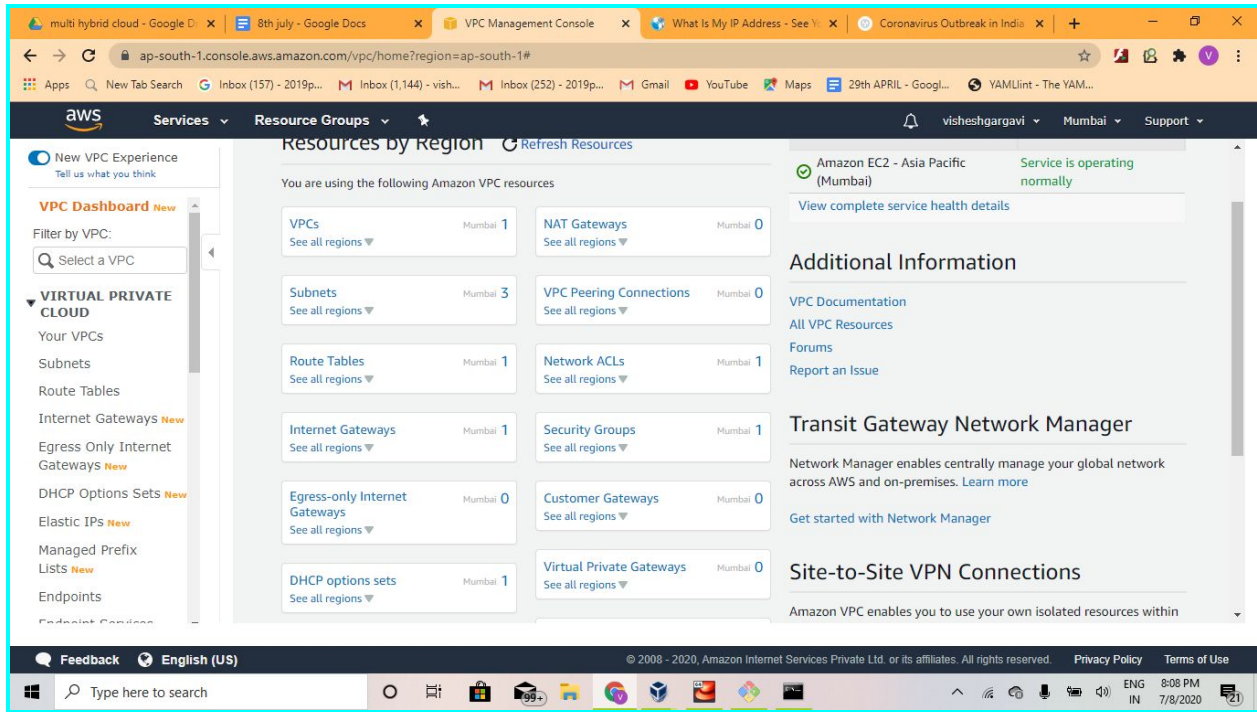
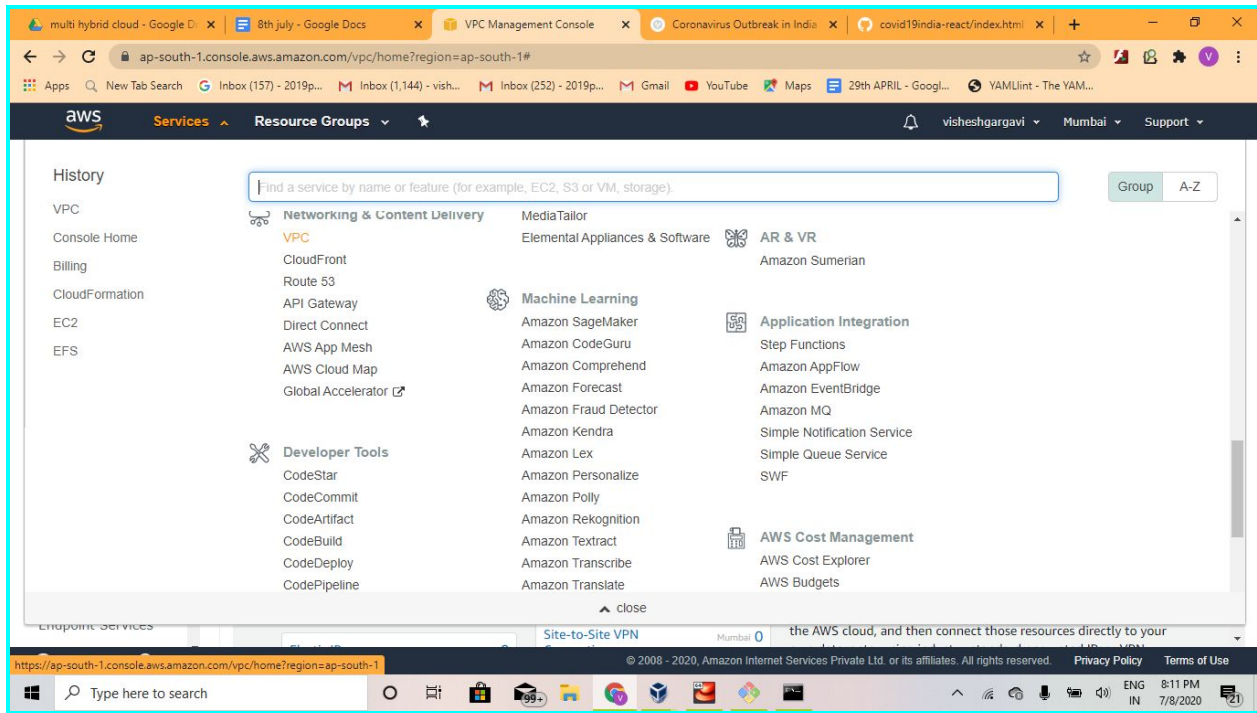
**Source NAT is performed by the router**

**Every system has NAT enabled**

The screenshot shows a web browser window with multiple tabs open, including 'multi hybrid cloud - Google Drive', '8th July - Google Docs', 'AWS Management Console', and 'What Is My IP Address - See You...'. The active tab is 'whatismyipaddress.com'. The website has a green header with navigation links: MY IP, IP LOOKUP, HIDE MY IP, VPNs, TOOLS, and LEARN. A sidebar on the left contains links to IP Lookup, Trace Email, Hide My IP, VPN Comparison, and Blacklist Check. The main content area displays 'My IP Address Is:' followed by 'IPv4: 223.238.198.115' and 'IPv6: [redacted]'. Below this, a box titled 'My IP Information:' lists 'ISP: Airtel', 'City: Rajsamand', 'Region: Rajasthan', and 'Country: India'. A red button says 'Make My IP Address Private Click Here'. The Windows taskbar at the bottom shows the search bar, task view, and various application icons. The system clock indicates 7:56 PM on 7/8/2020.

If we don't have the route table set then also we can't ping  
Dnat is not allowed by default (ping or pod address translation)





router is also known as gateway

C:\Users\user>route PRINT

21...0a 00 27 00 00 15 .....VirtualBox Host-Only Ethernet Adapter

10...0a 00 27 00 00 0a .....VirtualBox Host-Only Ethernet Adapter #2

20...0a 00 27 00 00 14 .....VirtualBox Host-Only Ethernet Adapter #3

68...02 00 4c 4f 4f 50 .....Npcap Loopback Adapter

```

14...1e bb 58 42 45 3f .....Microsoft Wi-Fi Direct Virtual Adapter
13...2e bb 58 42 45 3f .....Microsoft Wi-Fi Direct Virtual Adapter #2
15...4c bb 58 42 45 3f .....Dell Wireless 1705 802.11b|g|n (2.4GHZ)
2...4c bb 58 42 45 40 .....Bluetooth Device (Personal Area Network)
1.....Software Loopback Interface 1

```

#### IPv4 Route Table

##### Active Routes:

Network	Destination	Netmask	Gateway	Interface	Metric
0.0.0.0	0.0.0.0	192.168.43.1	192.168.43.55	55	
127.0.0.0	255.0.0.0	On-link	127.0.0.1	331	

##### Persistent Routes:

None

#### IPv6 Route Table

##### Active Routes:

```
[root@localhost ~]# route -n
```

##### Kernel IP routing table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
0.0.0.0	10.0.2.2	0.0.0.0	UG	100	0	0	enp0s3
0.0.0.0	192.168.43.1	0.0.0.0	UG	101	0	0	enp0s8
0.0.0.0	192.168.43.1	0.0.0.0	UG	102	0	0	enp0s9

```
[root@localhost ~]# route del -net 0.0.0.0
```

```
[root@localhost ~]# route del -net 0.0.0.0
```

```
[root@localhost ~]# ping 8.8.8.8
```

```
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
```

```
64 bytes from 8.8.8.8: icmp_seq=1 ttl=114 time=56.8 ms
```

```
[root@localhost ~]# route -n
```

##### Kernel IP routing table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
0.0.0.0	192.168.43.1	0.0.0.0	UG	101	0	0	enp0s8
0.0.0.0	192.168.43.1	0.0.0.0	UG	102	0	0	enp0s9

```
10.0.2.0    0.0.0.0    255.255.255.0  U   100  0    0 enp0s3
172.17.0.0  0.0.0.0    255.255.0.0   U   0    0    0 docker0
172.18.0.0  0.0.0.0    255.255.0.0   U   0    0    0 br-86e4dd51aaf8
192.168.43.0 0.0.0.0    255.255.255.0 U   101  0    0 enp0s8
192.168.43.0 0.0.0.0    255.255.255.0 U   102  0    0 enp0s9
192.168.122.0 0.0.0.0    255.255.255.0 U   0    0    0 virbr0
```

```
[root@localhost ~]# route add -net 0.0.0.0
```

```
SIOCADDRT: No such device
```

```
[root@localhost ~]# route add -net 0.0.0.0 gw 10.0.2.2
```

```
M) ipx (Novell IPX) ddp (Appletalk DDP)
```

```
x25 (CCITT X.25)
```

```
[root@localhost ~]# route -n
```

```
Kernel IP routing table
```

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
0.0.0.0	10.0.2.2	0.0.0.0	UG	0	0	0	enp0s3
0.0.0.0	192.168.43.1	0.0.0.0	UG	101	0	0	enp0s8
0.0.0.0	192.168.43.1	0.0.0.0	UG	102	0	0	enp0s9

The screenshot shows the AWS Management Console interface. The left sidebar contains navigation links for VPC services. The main content area displays the 'Route Tables' page. A table lists the available route tables, with 'rtb-2c6dbe47' highlighted. Below the table, the 'Summary' tab for the selected route table is shown, confirming its association with VPC 'vpc-15f8e57d' and its status as the main route table.

```
[root@openstack ~]# ifconfig enp0s3
```

```
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
```

```
inet 192.168.43.186 netmask 255.255.255.0 broadcast 192.168.43.255
```

```
inet6 2401:4900:463f:a5bd:a00:27ff:fe04:dd49 prefixlen 64 scopeid
```

```
0x0<global>
```

inet6 fe80::a00:27ff:fe04:dd49 prefixlen 64 scopeid 0x20<link>

ether 08:00:27:04:dd:49 txqueuelen 1000 (Ethernet)

RX packets 2747 bytes 188267 (183.8 KiB)

RX errors 0 dropped 0 overruns 0 frame 0

TX packets 2570 bytes 1292543 (1.2 MiB)

TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

You have mail in /var/spool/mail/root

[root@openstack ~]# rpm -qa | grep openv

openstack-neutron-openvswitch-12.0.2-0.20180421011362.0ec54fd.el7ost.noarch

openvswitch-selinux-extra-policy-1.0-3.el7fdp.noarch

python-openvswitch-2.9.0-47.el7fdp.2.noarch

openvswitch-2.9.0-47.el7fdp.2.x86\_64

```
root@openstack~# [y, n]
CONFIG_NEUTRON_METERING_AGENT_INSTALL=y

# Specify 'y' to configure OpenStack Networking's Firewall-
# as-a-Service (FWaaS). [y, n]
CONFIG_NEUTRON_FWAAS=n

# Specify 'y' to configure OpenStack Networking's VPN-as-a-Service
# (VPNaaS). [y, n]
CONFIG_NEUTRON_VPNaaS=n

# Comma-separated list of network-type driver entry points to be
# loaded from the neutron.ml2.type_drivers namespace. [local,
# 'flat', 'vlan', 'gre', 'vxlan', 'geneve']
CONFIG_NEUTRON_ML2_TYPE_DRIVERS=vxlan,flat

# Comma-separated, ordered list of network types to allocate as
# tenant networks. The 'local' value is only useful for single-box
# testing and provides no connectivity between hosts. [local,
# 'vlan', 'gre', 'vxlan', 'geneve']
CONFIG_NEUTRON_ML2_TENANT_NETWORK_TYPES=vxlan

# Comma-separated ordered list of networking mechanism driver entry
# points to be loaded from the neutron.ml2.mechanism_drivers
# namespace. [logger, test, 'linuxbridge', 'openvswitch',
# 'hyperv', 'nics', 'arista', 'cisco_nexus', 'mlnx', 'l2population',
# 'sriovswitch', 'ovn']
CONFIG_NEUTRON_ML2_MECHANISM_DRIVERS=openvswitch

# Comma-separated list of physical_network names with which flat
# networks can be created. Use * to allow flat networks with arbitrary
# physical_network names.
CONFIG_NEUTRON_ML2_FLAT_NETWORKS=*

# Comma-separated list of <physical_network>:<vlan_min>:<vlan_max> or
# <physical_network> specifying physical_network names usable for VLAN
# provider and tenant networks, as well as ranges of VLAN tags on each
# available for allocation to tenant networks.
```

```
root@openstack~# physical_network names.
CONFIG_NEUTRON_ML2_FLAT_NETWORKS=*

# Comma-separated list of <physical_network>:<vlan_min>:<vlan_max> or
# <physical_network> specifying physical_network names usable for VLAN
# provider and tenant networks, as well as ranges of VLAN tags on each
# available for allocation to tenant networks.
CONFIG_NEUTRON_ML2_VLAN_RANGES=

# Comma-separated list of <tun_min>:<tun_max> tuples enumerating
# ranges of GRE tunnel IDs that are available for tenant-network
# allocation. A tuple must be an array with tun_max +1 - tun_min >
# 1000000.
CONFIG_NEUTRON_ML2_TUNNEL_ID_RANGES=

# Comma-separated list of addresses for VXLAN multicast group. If
# left empty, disables VXLAN from sending allocated broadcast traffic
# (disables multicast VXLAN mode). Should be a Multicast IP (v4 or v6)
# address.
CONFIG_NEUTRON_ML2_VXLAN_GROUP=

# Comma-separated list of <vni_min>:<vni_max> tuples enumerating
# ranges of VXLAN VNI IDs that are available for tenant network
# allocation. Minimum value is 0 and maximum value is 16777215.
CONFIG_NEUTRON_ML2_VNI_RANGES=10:100

# Name of the L2 agent to be used with OpenStack Networking.
# ['linuxbridge', 'openvswitch', 'ovn']
CONFIG_NEUTRON_L2_AGENT=openvswitch

# Comma separated list of supported PCI vendor devices defined by
# vendor_id:product_id according to the PCI ID Repository.
CONFIG_NEUTRON_ML2_SUPPORTED_PCI_VENDOR_DEVS=['15b3:1004', '8086:10ca']

# Comma-separated list of interface mappings for the OpenStack
# Networking ML2 SRIOV agent. Each tuple in the list must be in the
# format <physical_network>:<net_interface>. Example:
# physnet1:eth1,physnet2:eth2,physnet3:eth3.
```

**[root@openstack ~](keystone\_demo)]# ovs-vsctl show**

**0c0552b8-e81b-4ef4-bc3a-0b2eaac6e011**

**Manager "ptcp:6640:127.0.0.1"**

**is\_connected: true**

**Bridge br-tun**

**Controller "tcp:127.0.0.1:6633"**

```
    is_connected: true
    fail_mode: secure
    Port patch-int
    Interface patch-int
    type: patch
    options: {peer=patch-tun}
    Port br-tun
    Interface br-tun
    type: internal
    Bridge br-int
```

>> important bridge for us is bridge int(bridge integration or internal)  
In networking we have subnet and they internally create the bridge-int  
And they attach it to the nat and to the vm  
And we can scale the ports