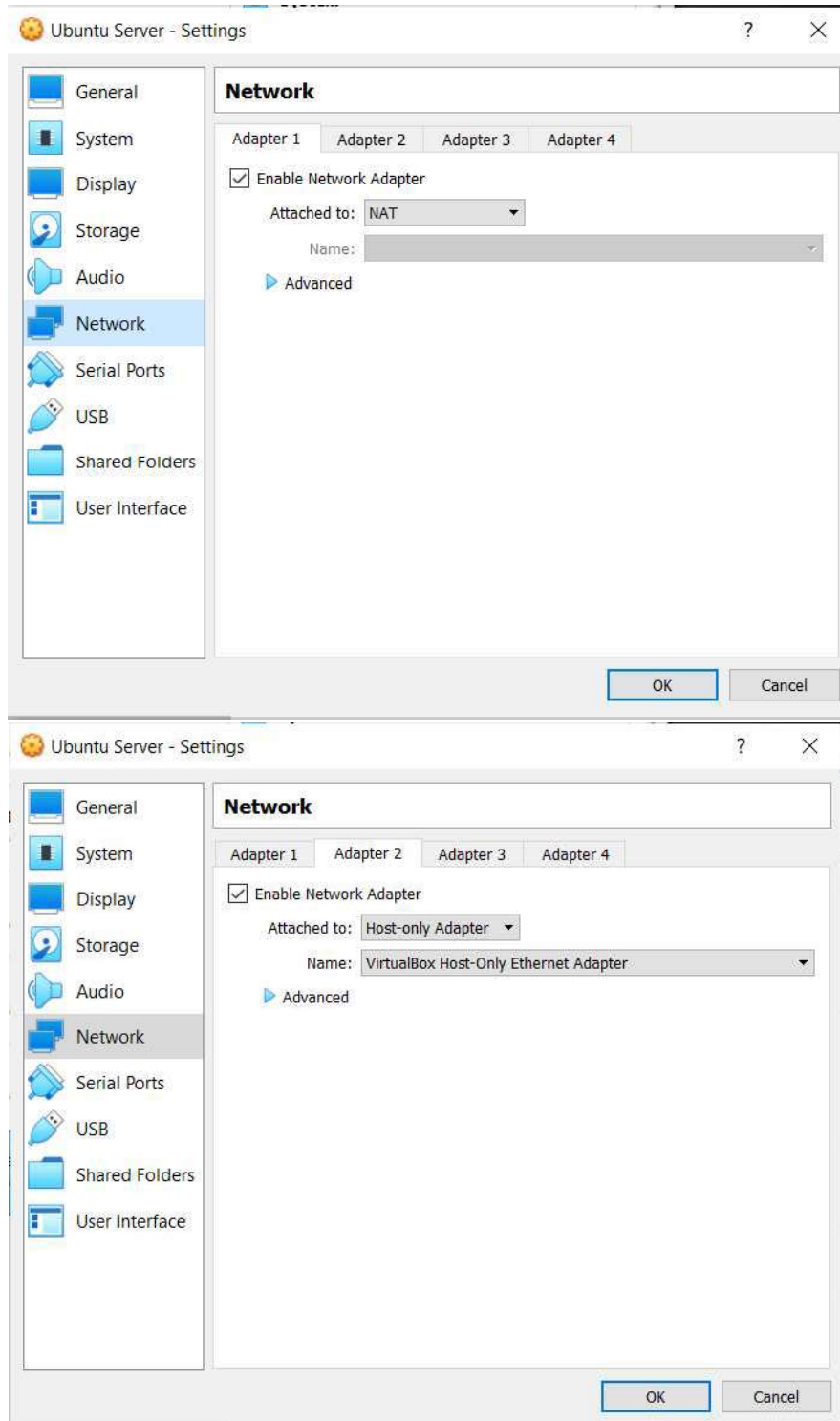


Install and Configure the DHCP server

Step 1: Change the **Network Setting** of your Virtual Machine.

We will add **two adapters**: One for the NAT and other for the Host-Only.





Step 2: Start the VM. First, we will update the Ubuntu repositories by running the following command in the terminal.

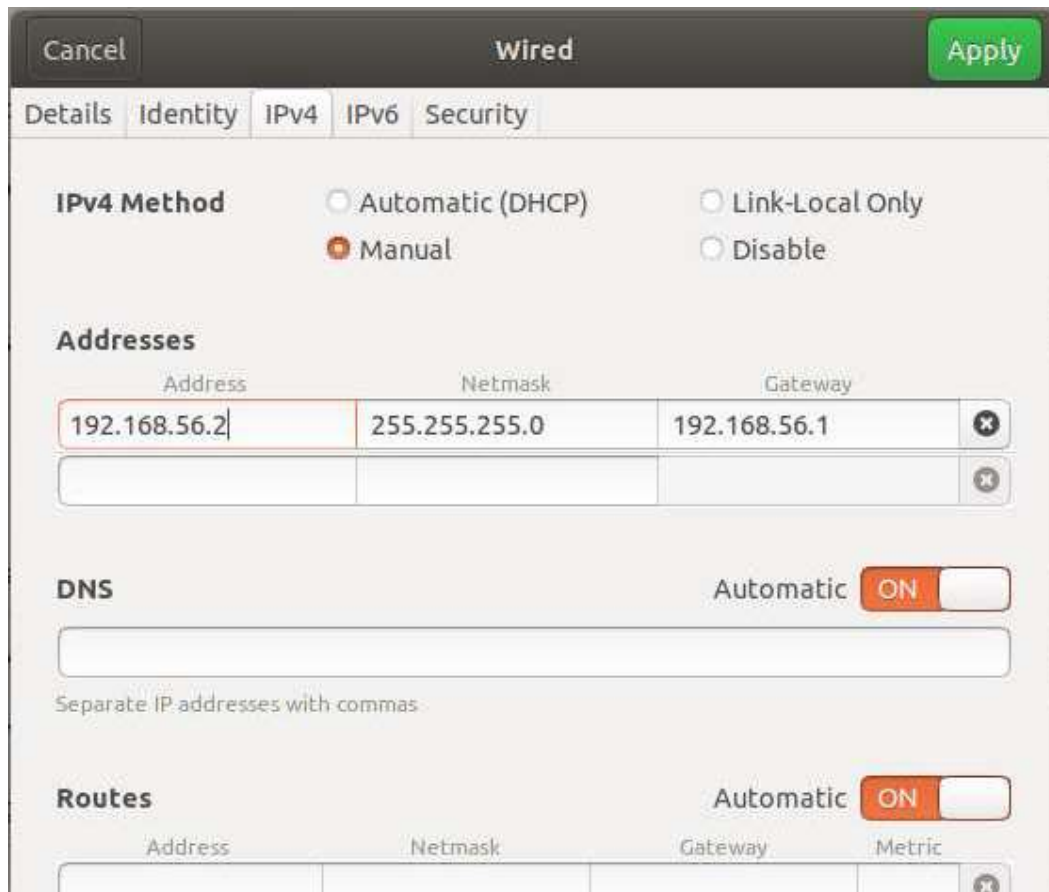
```

server@server-VirtualBox: ~
File Edit View Search Terminal Help
server@server-VirtualBox:~$ sudo apt-get update
[sudo] password for server:
Hit:1 http://ca.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://ca.archive.ubuntu.com/ubuntu bionic-updates InRelease
Get:3 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Hit:4 http://ca.archive.ubuntu.com/ubuntu bionic-backports InRelease
Fetched 88.7 kB in 1s (69.6 kB/s)
Reading package lists... Done
server@server-VirtualBox:~$

```

Step 3: Before we move on, Let's **make sure that the IP address of our server is static** since we don't want the server to assign a random IP address dynamically every time it restarts. In Ubuntu, you can do this by clicking the drop triangle icon  at the top right corner and then choosing **Host-Only Ethernet-> Wired Settings->**  icon.

After you are on the Wired Console go to the IPv4 tab. And choose the manual option under the IPv4 Method. **Put the IPv4 address according to your network ID and subnet.**



Cancel **Wired** Apply

Details Identity **IPv4** IPv6 Security

IPv4 Method

☐ Automatic (DHCP) ☐ Link-Local Only

☒ **Manual** ☐ Disable

Addresses

Address	Netmask	Gateway
192.168.56.2	255.255.255.0	192.168.56.1

DNS Automatic **ON**

Separate IP addresses with commas

Routes Automatic **ON**

Address	Netmask	Gateway	Metric

Step 4: Now, run the **\$ ifconfig** command to check the interfaces. If the **\$ ifconfig** command is not found, you have to install the net-tools package first as follow

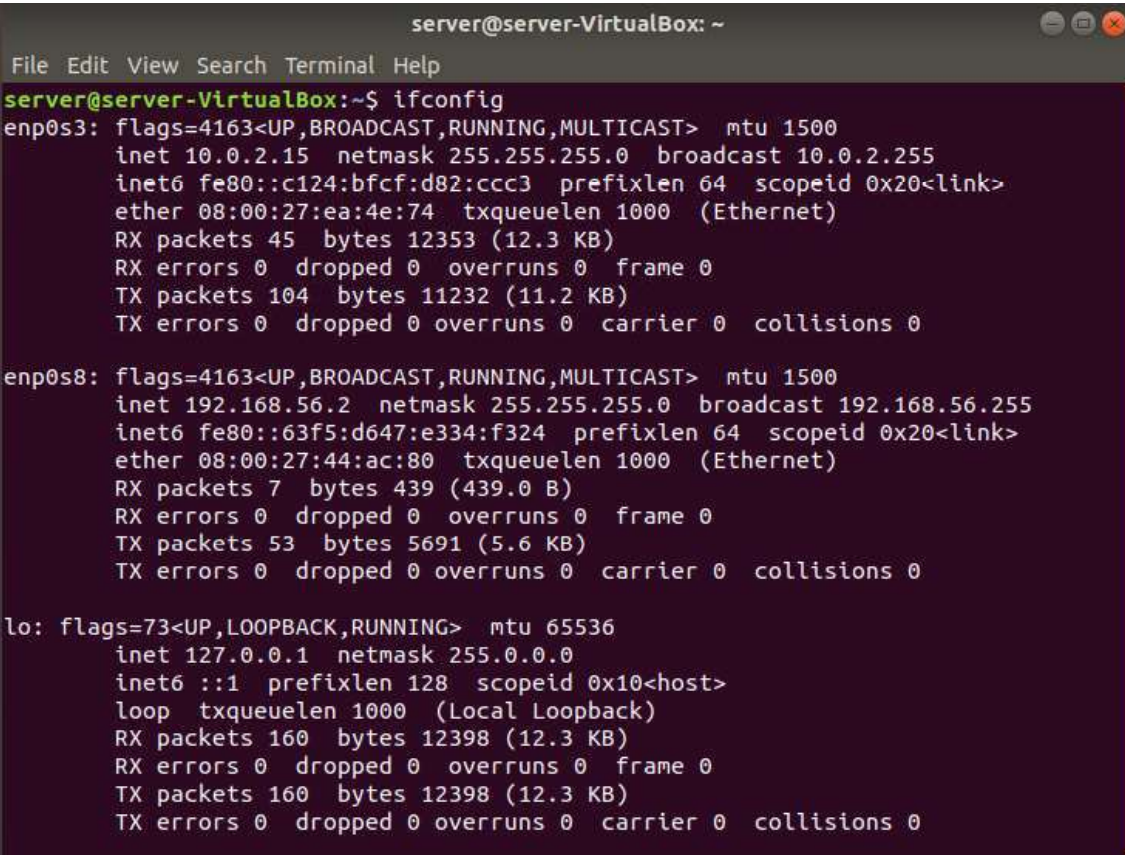
\$ sudo apt-get install net-tools

Now, if we run the **\$ ifconfig**, we will see the following:

Here we have two interfaces running: **enp0s3**: NAT Interface:

enp0s8: Host-Only Interface:

Notice the IP address of the enp0s8 is the same you have configured in the last step.



```

server@server-VirtualBox: ~
File Edit View Search Terminal Help
server@server-VirtualBox:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
    inet6 fe80::c124:bfcf:d82:ccc3 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:ea:4e:74 txqueuelen 1000 (Ethernet)
    RX packets 45 bytes 12353 (12.3 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 104 bytes 11232 (11.2 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.56.2 netmask 255.255.255.0 broadcast 192.168.56.255
    inet6 fe80::63f5:d647:e334:f324 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:44:ac:80 txqueuelen 1000 (Ethernet)
    RX packets 7 bytes 439 (439.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 53 bytes 5691 (5.6 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 160 bytes 12398 (12.3 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 160 bytes 12398 (12.3 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
  
```

Step 5: Further, we will install the DHCP server by running the following command:

```
server@server-VirtualBox:~$ sudo apt-get install isc-dhcp-server -y
```

Step 6: Next, we will edit the **/etc/default/isc-dhcp-server** file by running the following command:

\$ sudo nano /etc/default/isc-dhcp-server

In the file, we will change the value of the **INTERFACESv4** variable to “enp0s8”, which is the interface for the host-only network.

```
# On what interfaces should the DHCP server (dhcpd) serve DHCP requests?
#       Separate multiple interfaces with spaces, e.g. "eth0 eth1".
INTERFACESv4="enp0s8"
INTERFACESv6=""
```

Step 7: Now we will edit the **DHCP configuration file** inside the directory `/etc/dhcp/`

Run the following command to open the file in the editor:

```
$ sudo nano /etc/dhcp/dhcpd.conf
```

In the DHCP configuration file **uncomment the line saying authoritative**; by removing the `#` sign from the front of the line.

```
# If this DHCP server is the official DHCP server for the local
# network, the authoritative directive should be uncommented.
authoritative;
```

First, we set the subnet and netmask to the network Id of the “enp0s8” interface

Then, we will specify the range for the IP addresses that our DHCP server will provide. The range should be according to your usage. If you don’t have the DNS server running yet do not uncomment the domain-name-server lines as shown.

Again, specify the subnet mask. Set the option routers to your broadcast-address.

After the above changes, the code should look like the following except your IP address information might be different than the one shown here.

```
# A slightly different configuration for an internal subnet.
subnet 192.168.56.0 netmask 255.255.255.0 {
    range 192.168.56.101 192.168.56.200;
    #option domain-name-servers ns1.internal.example.org;
    # option domain-name "internal.example.org";
    option subnet-mask 255.255.255.0;
    option routers 196.168.56.255;
    option broadcast-address 196.168.56.255;
    default-lease-time 600;
    max-lease-time 7200;
}
```

Step 8: After editing the `dhcpd.conf` file we will start the DHCP service on the server by running the following command:

```
$ sudo systemctl start isc-dhcp-server
```

And to check the status of the DHCP server, run the following command:

\$ sudo systemctl status isc-dhcp-server

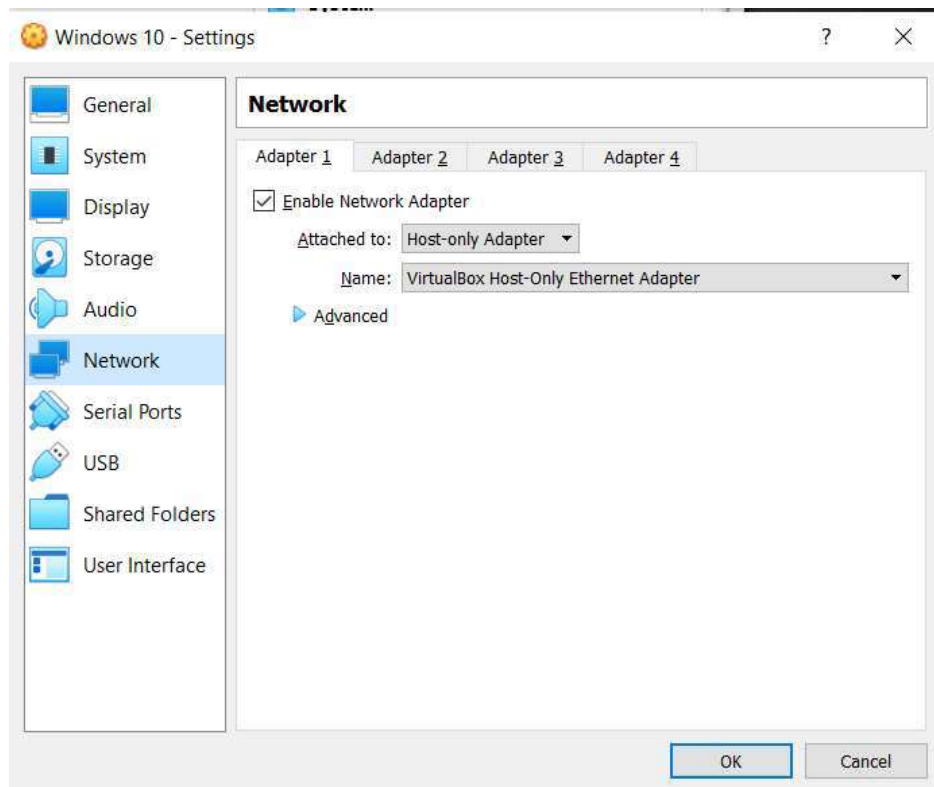
```
server@server-VirtualBox:~$ sudo systemctl start isc-dhcp-server
server@server-VirtualBox:~$ sudo systemctl status isc-dhcp-server
● isc-dhcp-server.service - ISC DHCP IPv4 server
   Loaded: loaded (/lib/systemd/system/isc-dhcp-server.service; enabled; vendor
   Active: active (running) since Fri 2020-04-10 15:35:42 PDT; 3min 12s ago
     Docs: man:dhcpd(8)
    Main PID: 3380 (dhcpd)
      Tasks: 1 (limit: 1752)
    CGroup: /system.slice/isc-dhcp-server.service
            └─3380 dhcpd -user dhcpd -group dhcpd -f -4 -pf /run/dhcp-server/dhc
```

Step 9: To enable the service for the boot time, run the following command.

\$ sudo systemctl enable isc-dhcp-server

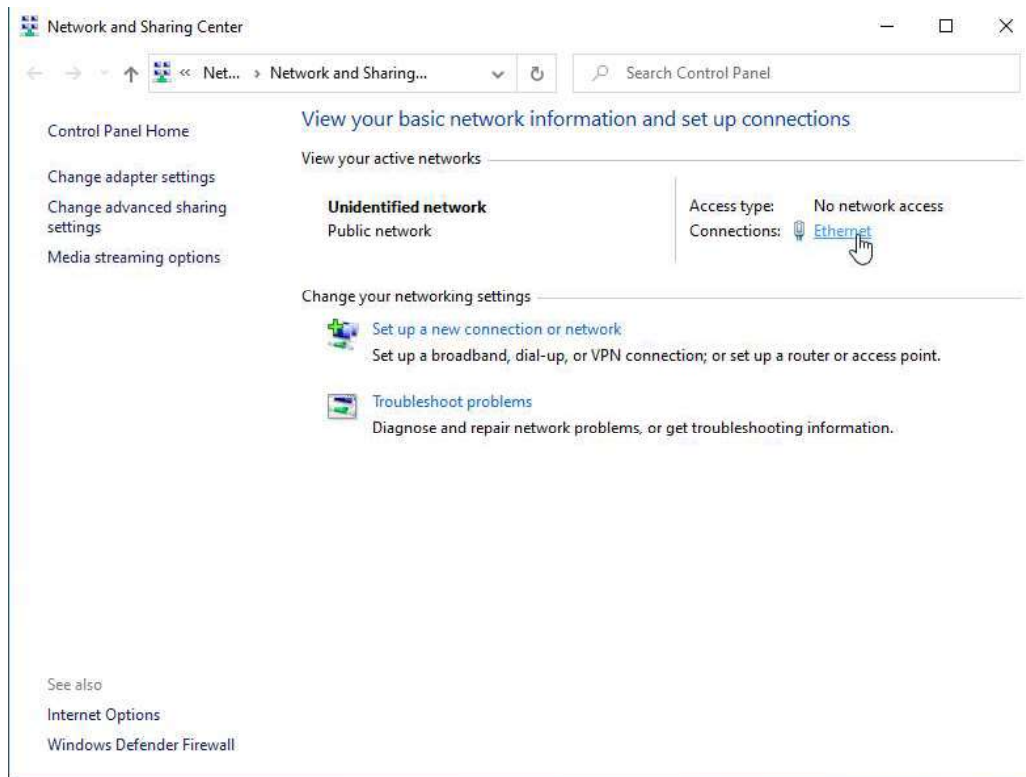
The DHCP server is configured and running. Now we will configure the client-side.

Before starting, make sure that the Windows 10 VM's network adapter should be set to host-Only.

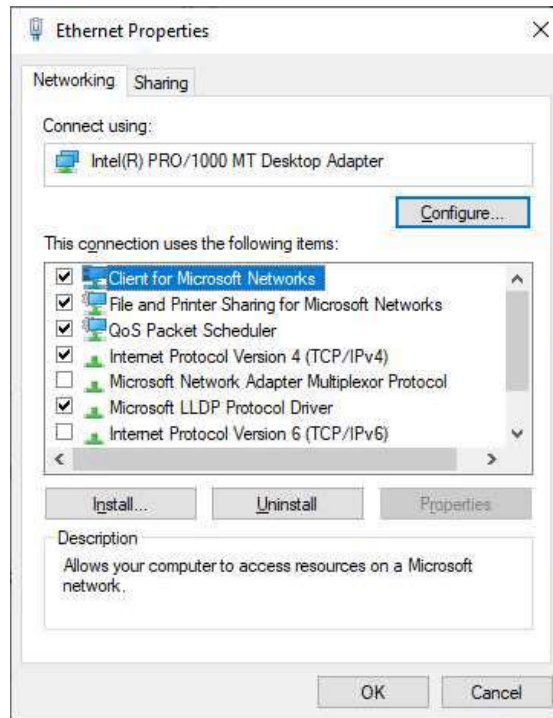


Step 1: Open the windows 10 as a client. Go to the Network and Sharing Center.

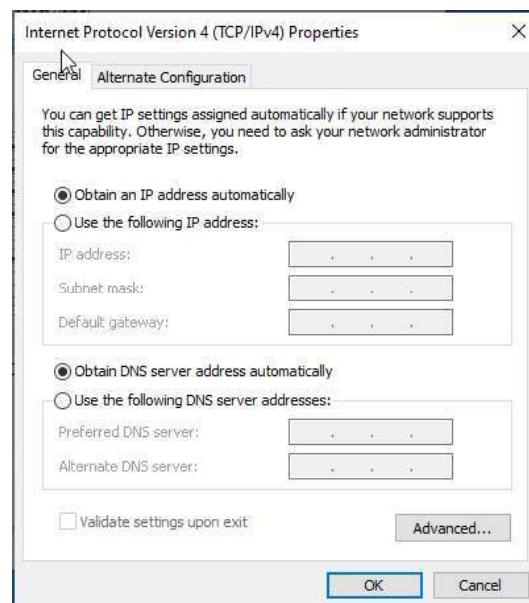
Step 2: On the Network and Sharing Center Console, click on the link say Ethernet, as shown below.



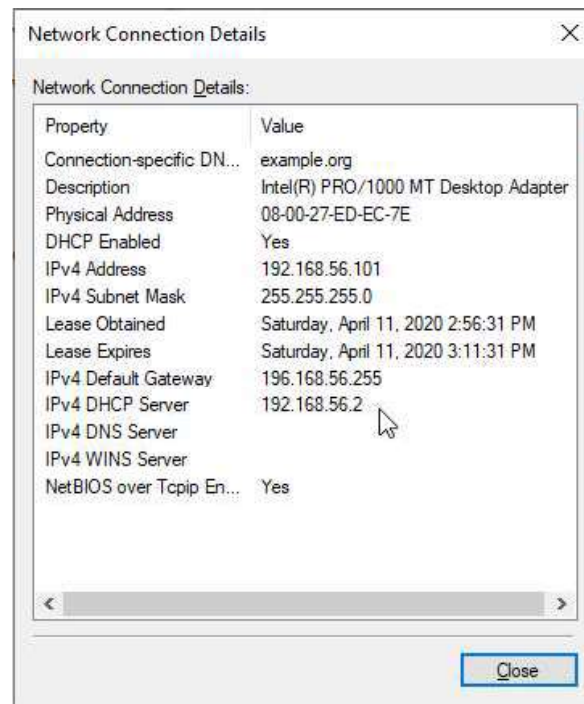
Step 3: An ethernet status window shows up. Click on the Properties button. It may require an **administrator privilege**. After putting the administrator credentials, the following window shows up.



Step 4: Now, double click on the Item saying Internet Protocol Version 4(TCP/ IPv4). Choose *Obtain an IP address automatically* and *Obtain DNS server address automatically*.



Step 5: Next return to the Ethernet Status window console by clicking OK. Now, click the Details button. As you can see that the IP address of the computer is now within the range, we specify on our DHCP server. And also, the IP address of the DHCP server is the same that we configured before.



Now, we have configured our client to get the IP address dynamically.

Deploy DNS Name Server

Step 1: Configure the hostname of the server to be static. We are using “**server linux.com**” as our hostname. To configure the hostname run the following command:

```
$ hostnamectl set-hostname server linux.com
```

To check your current hostname, use the following command:

```
server@server-VirtualBox:~$ hostnamectl
Static hostname: server linux.com
Icon name: computer-vm
Chassis: vm
Machine ID: 02d0b10361ba44d6bd16ba0781e76739
Boot ID: 0af976b16a8f41f19a3896a29eb428ae
Virtualization: oracle
Operating System: Ubuntu 18.04.4 LTS
Kernel: Linux 5.3.0-28-generic
Architecture: x86-64
server@server-VirtualBox:~$
```

Step 2: Run the following command to install packages “**bind9**” and “**bind9utils**”.

```
$ sudo apt-get install bind9 bind9utils
```

Step 3: Now we will start editing the files. First, edit the **/etc/bind/named.conf.local** file by running the following command:

```
$ sudo nano /etc/bind/named.conf.local
```

We will create two zones in this file. Forward Zone and Reverse Zone.

Forward Zone: In this, the name will map to the IP address.

Reverse Zone: In this, the IP address will map to the name.

After the edits your file should look as follows:

```
// Do any local configuration here
//

// Consider adding the 1918 zones here, if they are not used in your
// organization
//include "/etc/bind/zones.rfc1918";

zone "linux.com" IN {
    type master;
    file "/etc/bind/forward linux.com";
};

zone "56.168.192.in-addr.arpa" IN {
    type master;
    file "/etc/bind/reverse linux.com";
};
```

Note that in the second zone the numbers 56.168.192 correspond to the reverse order of your IP address. We don't include the number from the last octet.

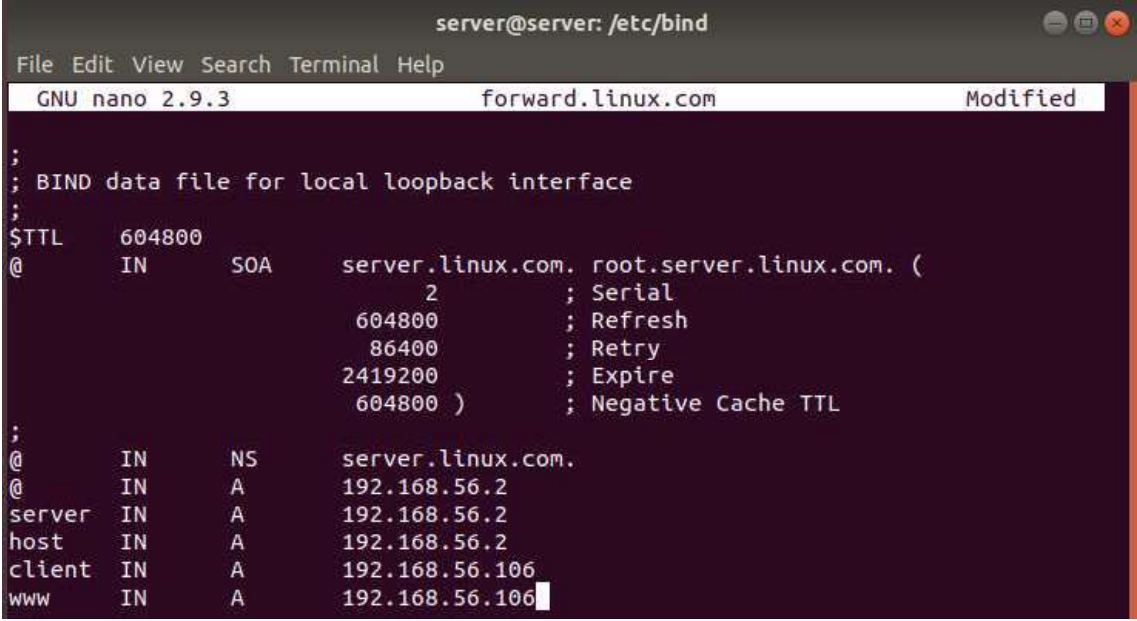
Step 4: Further, Copy the **db.local** file to **forward.linux.com** by using the following command (we have changed the directory to **/etc/bind/**):

```
server@server:/etc/bind$ sudo cp db.local forward.linux.com
server@server:/etc/bind$ ls
bind.keys  db.empty          forward.linux.com  named.conf.options
db.0       db.local          named.conf         rndc.key
db.127     db.root          named.conf.default-zones  zones.rfc1918
db.255     forward.example.com named.conf.local
server@server:/etc/bind$
```

As you can see, now we have a new **forward.linux.com** file in our bind directory.

Step 5: Next we will edit the **forward.linux.com** file as follow:

\$ sudo nano forward.linux.com



```
server@server: /etc/bind
File Edit View Search Terminal Help
GNU nano 2.9.3 forward.linux.com Modified
;
; BIND data file for local loopback interface
;
$TTL      604800
@         IN      SOA      server.linux.com. root.server.linux.com. (
                        2      ; Serial
                        604800 ; Refresh
                        86400  ; Retry
                        2419200 ; Expire
                        604800 ) ; Negative Cache TTL
;
@         IN      NS       server.linux.com.
@         IN      A        192.168.56.2
server    IN      A        192.168.56.2
host      IN      A        192.168.56.2
client    IN      A        192.168.56.106
www       IN      A        192.168.56.106
```

Step 6: Next we will copy **forward.linux.com** file to **reverse.linux.com**. And do some **edits** to reverse.linux.com file

Copy File:

```
server@server:/etc/bind$ sudo cp forward.linux.com reverse.linux.com
server@server:/etc/bind$ ls
bind.keys  db.empty          forward.linux.com  named.conf.options
db.0       db.local          named.conf         reverse.linux.com
db.127     db.root          named.conf.default-zones  rndc.key
db.255     forward.example.com named.conf.local    zones.rfc1918
server@server:/etc/bind$
```

Do the following edits:

\$ sudo nano reverse linux.com

```

server@server: /etc/bind
File Edit View Search Terminal Help
GNU nano 2.9.3 reverse linux.com Modified

;
; BIND data file for local loopback interface
;
$TTL      604800
@         IN      SOA      server linux.com. root server linux.com. (
                        2      ; Serial
                        604800  ; Refresh
                        86400   ; Retry
                        2419200 ; Expire
                        604800 ) ; Negative Cache TTL
;
@         IN      NS       server linux.com.
@         IN      PTR      linux.com.
server    IN      A        192.168.56.2
host      IN      A        192.168.56.2
client    IN      A        192.168.56.106
www       IN      A        192.168.56.106
2         IN      PTR      server linux.com.
106      IN      PTR      client linux.com.

```

Note: we are adding the client just for the testing purpose.

Step 7: Now, we will check our configurations to find if there is any syntax error. Run the following commands:

To check named.conf.local file:

```

server@server:/etc/bind$ sudo named-checkconf -z named.conf
zone linux.com/IN: loaded serial 2
zone 56.168.192.in-addr.arpa/IN: loaded serial 2
zone localhost/IN: loaded serial 2
zone 127.in-addr.arpa/IN: loaded serial 1
zone 0.in-addr.arpa/IN: loaded serial 1
zone 255.in-addr.arpa/IN: loaded serial 1
server@server:/etc/bind$

```

To check the zones:

```

server@server:/etc/bind$ sudo named-checkzone forward forward linux.com
zone forward/IN: loaded serial 2
OK
server@server:/etc/bind$ sudo named-checkzone reverse reverse linux.com
zone reverse/IN: loaded serial 2
OK
server@server:/etc/bind$

```

Step 8: Before we start the bind9 service, we will have to change the ownership of the files.

Use the following commands to do so:

```
$ sudo chown -R bind:bind /etc/bind
```

```
$ sudo chmod -R 755 /etc/bind
```

Step 9: Next we will start the bind9 service by the following command:

```
server@server:/etc/bind$ sudo systemctl start bind9
server@server:/etc/bind$ sudo systemctl status bind9
● bind9.service - BIND Domain Name Server
   Loaded: loaded (/lib/systemd/system/bind9.service; enabled; vendor preset: e
   Active: active (running) since Mon 2020-04-13 10:45:28 PDT; 1h 7min ago
     Docs: man:named(8)
    Main PID: 2095 (named)
      Tasks: 4 (limit: 1752)
    CGroup: /system.slice/bind9.service
            └─2095 /usr/sbin/named -f -u bind
```


```
$ sudo systemctl enable bind9
```

Step 10: Run the following command to allow bind service through the firewall:

```
$ sudo ufw allow bind9
```

Step 11: Add the following lines to the interfaces file at /etc/network/interfaces

```
$ sudo nano /etc/network/interfaces
```



```
server@server: /etc/bind
File Edit View Search Terminal Help
GNU nano 2.9.3 /etc/network/interfaces Modified
# interfaces(5) file used by ifup(8) and ifdown(8)
auto lo
iface lo inet loopback
dns-search linux.com
dns-nameserver 192.168.56.2
```

Step 12: Also, edit the /etc/resolv.conf as follow:

\$ sudo nano /etc/resolv.conf

```

server@server: ~
File Edit View Search Terminal Help
GNU nano 2.9.3 /etc/resolv.conf

# This file is managed by man:systemd-resolved(8). Do not edit.
#
# This is a dynamic resolv.conf file for connecting local clients to the
# internal DNS stub resolver of systemd-resolved. This file lists all
# configured search domains.
#
# Run "systemd-resolve --status" to see details about the uplink DNS servers
# currently in use.
#
# Third party programs must not access this file directly, but only through the
# symlink at /etc/resolv.conf. To manage man:resolv.conf(5) in a different way,
# replace this symlink by a static file or a different symlink.
#
# See man:systemd-resolved.service(8) for details about the supported modes of
# operation for /etc/resolv.conf.

nameserver 192.168.56.2
search linux.com

```

Step 13: Now restart the networking and NetworkManager services as shown below:

```

server@server:/etc/bind$ sudo systemctl restart networking
server@server:/etc/bind$ sudo systemctl restart NetworkManager
server@server:/etc/bind$

```

Step 14: Now we will edit the **DHCP configuration file** inside the directory `/etc/dhcp/` to change the DNS server.

Run the following command to open the file in the editor:

\$ sudo nano /etc/dhcp/dhcpd.conf

```

# option definitions common to all supported networks...
option domain-name "linux.com";
option domain-name-servers 192.168.56.2;

```

Now restart the DHCP and DNS services:

\$ sudo systemctl restart isc-dhcp-server

\$ sudo systemctl restart bind9

Step 15: Next we will check if our DNS server has been configured right. Run the commands as shown:

```
server@server:~$ ping server
PING server.linux.com (192.168.56.2) 56(84) bytes of data.
64 bytes from server.linux.com (192.168.56.2): icmp_seq=1 ttl=64 time=0.014 ms
64 bytes from server.linux.com (192.168.56.2): icmp_seq=2 ttl=64 time=0.035 ms
64 bytes from server.linux.com (192.168.56.2): icmp_seq=3 ttl=64 time=0.035 ms
64 bytes from server.linux.com (192.168.56.2): icmp_seq=4 ttl=64 time=0.034 ms
64 bytes from server.linux.com (192.168.56.2): icmp_seq=5 ttl=64 time=0.044 ms
^C
--- server.linux.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4079ms
rtt min/avg/max/mdev = 0.014/0.032/0.044/0.011 ms
```

With the **nslookup** command we can see that the server is resolved server.linux.com by the DNS.

```
server@server:~$ nslookup server
Server:          192.168.56.2
Address:         192.168.56.2#53

Name:   server.linux.com
Address: 192.168.56.2

server@server:~$ nslookup host
Server:          192.168.56.2
Address:         192.168.56.2#53

Name:   host.linux.com
Address: 192.168.56.2
```

Testing the client with **nslookup**:

```
server@server:~$ nslookup client
Server:          192.168.56.2
Address:         192.168.56.2#53

Name:   client.linux.com
Address: 192.168.56.106

server@server:~$ nslookup client2
Server:          192.168.56.2
Address:         192.168.56.2#53

** server can't find client2: NXDOMAIN
```

As you can see, we have only set up the **client** so it resolves to client.linux.com but we have not set up the **client2** so our DNS server can't find it

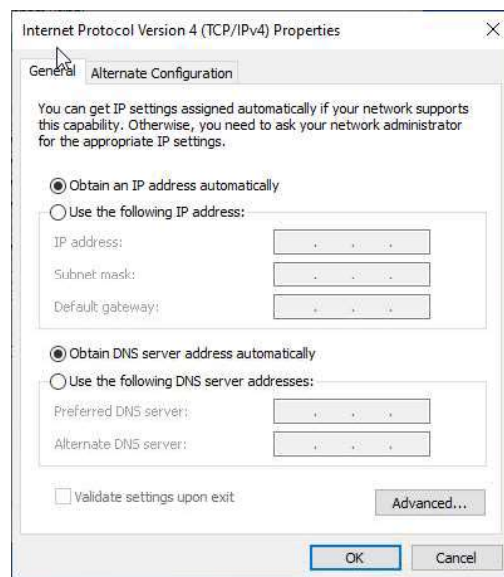
Testing the Windows 10 client

Step 1: Open the Windows 10 VM. Go to the **Network and Sharing Center-> Ethernet-> properties button**. Provide the administrator credentials, if required.

Step 2: Now, double click on the Item saying **Internet Protocol Version 4(TCP/ IPv4)**. Make sure these options are selected:

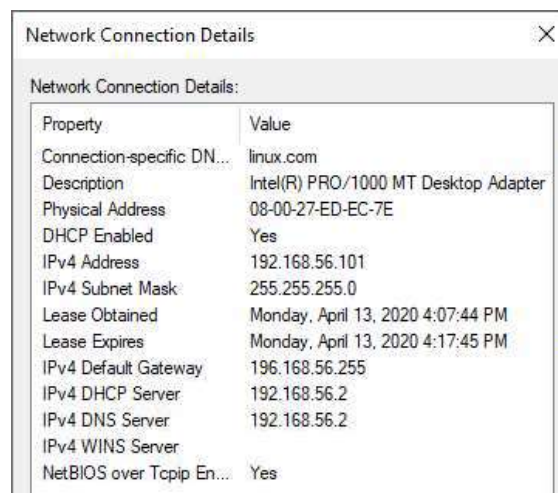
Choose *Obtain an IP address automatically*

Obtain DNS server address automatically.



Next return to the Ethernet Status window console by clicking OK.

Step 3: Next, click the Details button. As you can see, the IP address of the DNS server is the same that we just configured.



Step 4: Now, open the command and run the **ipconfig** command to check if the DNS server is identified.

```
Ethernet adapter Ethernet:

    Connection-specific DNS Suffix . : linux.com
    Description . . . . . : Intel(R) PRO/1000 MT Desktop Adapter
    Physical Address. . . . . : 08-00-27-ED-EC-7E
    DHCP Enabled. . . . . : Yes
    Autoconfiguration Enabled . . . : Yes
    IPv4 Address. . . . . : 192.168.56.101(Preferred)
    Subnet Mask . . . . . : 255.255.255.0
    Lease Obtained. . . . . : Monday, April 13, 2020 3:25:23 PM
    Lease Expires . . . . . : Monday, April 13, 2020 3:40:23 PM
    Default Gateway . . . . . : 196.168.56.255
    DHCP Server . . . . . : 192.168.56.2
    DNS Servers . . . . . : 192.168.56.2
    NetBIOS over Tcpip. . . . . : Enabled
```

Step 5: Ping the server:

```
C:\Users\player1>ping server

Pinging server.linux.com [192.168.56.2] with 32 bytes of data:
Reply from 192.168.56.2: bytes=32 time<1ms TTL=64
Reply from 192.168.56.2: bytes=32 time<1ms TTL=64
Reply from 192.168.56.2: bytes=32 time<1ms TTL=64
Reply from 192.168.56.2: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.56.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

Step 6: Test with the **nslookup**:

```
C:\Users\player1>nslookup 192.168.56.2
Server:  server.linux.com
Address:  192.168.56.2

Name:     server.linux.com
Address:  192.168.56.2

C:\Users\player1>nslookup 192.168.56.106
Server:  server.linux.com
Address:  192.168.56.2

Name:     client.linux.com
Address:  192.168.56.106
```

Hence, we can see that the IP address of our server and test client is being resolved to their appropriate domain names.