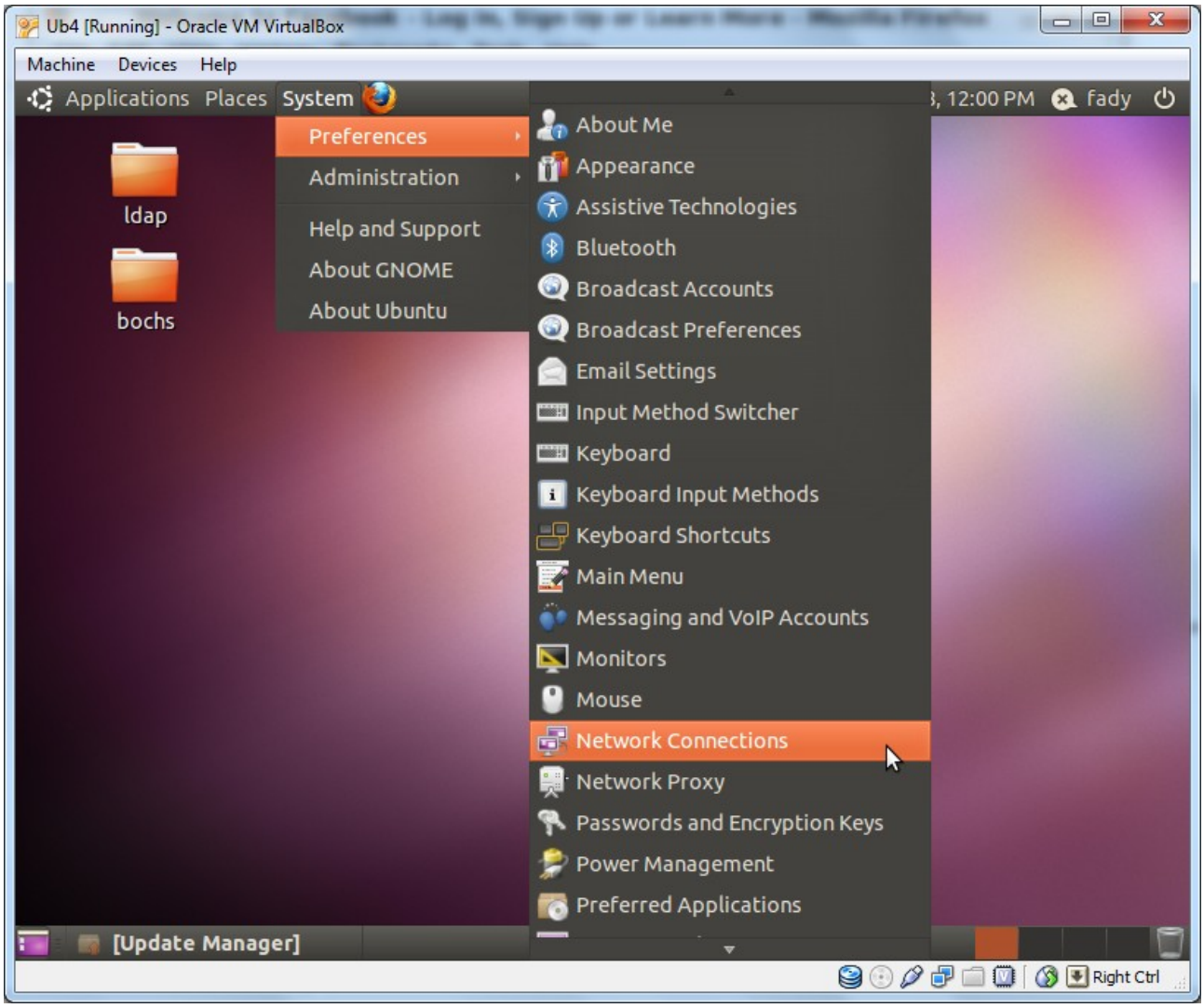
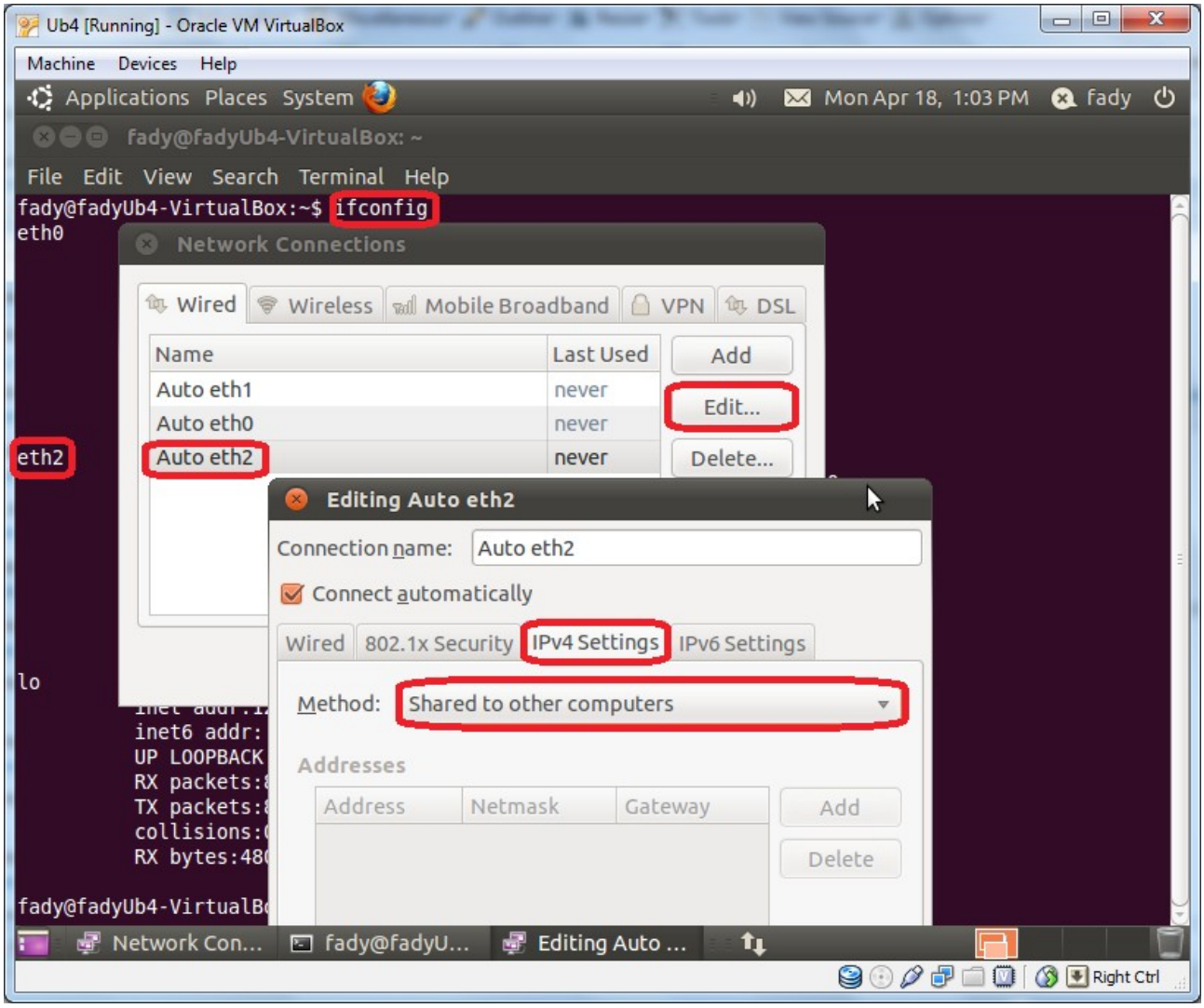
Linux Firewalls (Ubuntu IPTables) II

Here we will complete the previous firewall lab by making a bridge on the Ubuntu machine, to make the Ubuntu machine completely control the Internet connection on the LAN of the PCLinux machine.

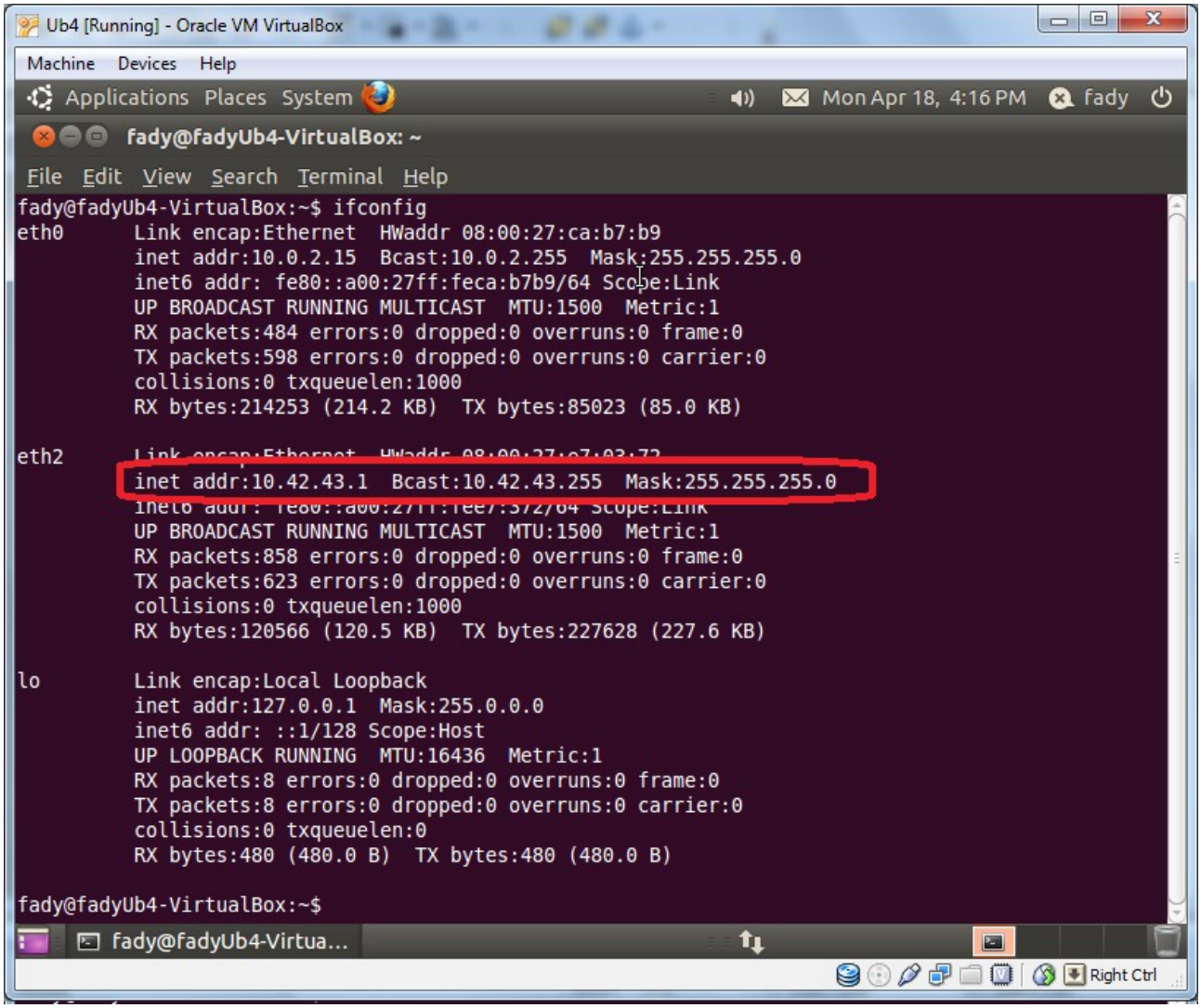


We start making the bridge by entering the Network Connections as shown

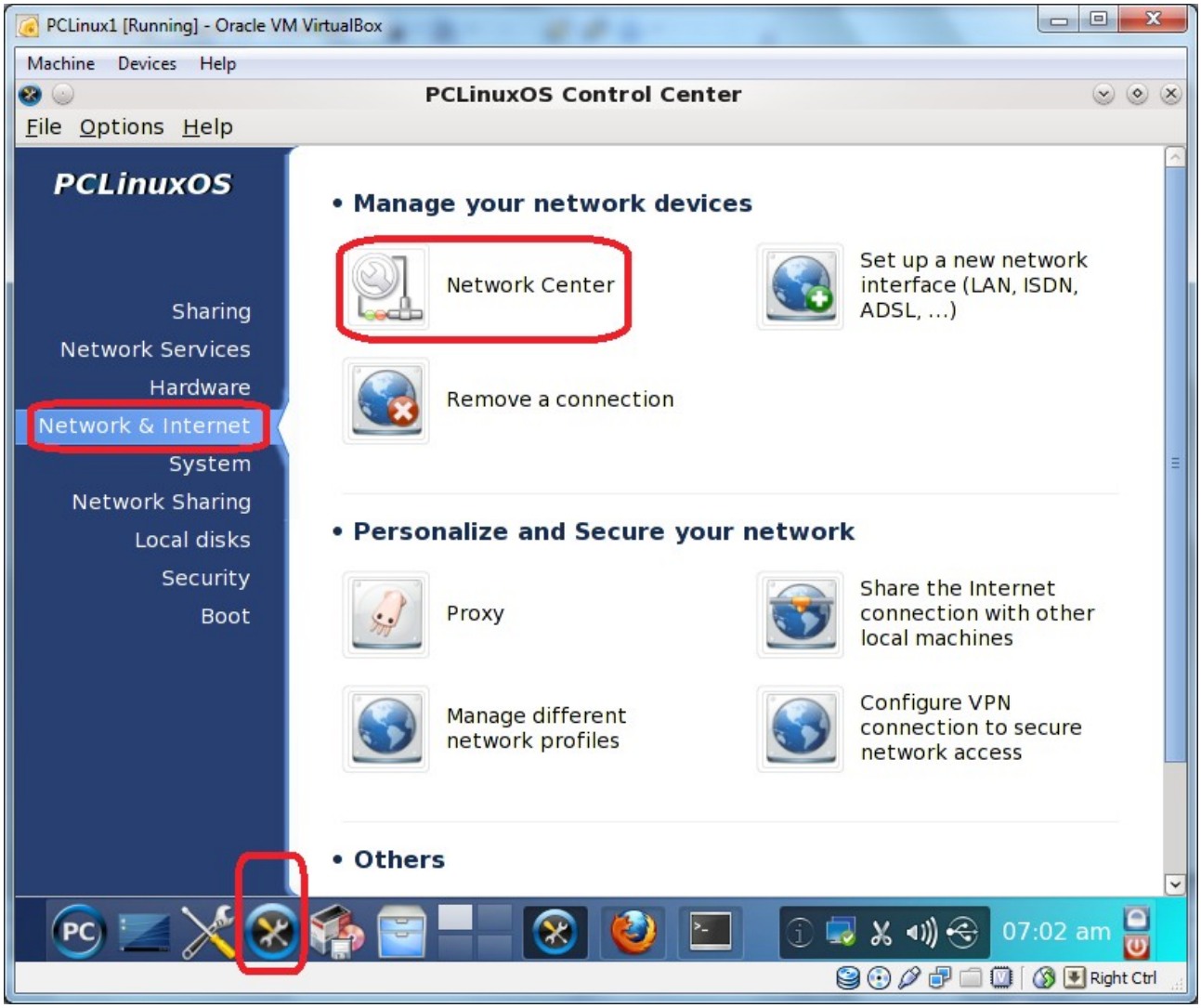
Then we just make the NIC of the local network share the connection with other computers as shown.



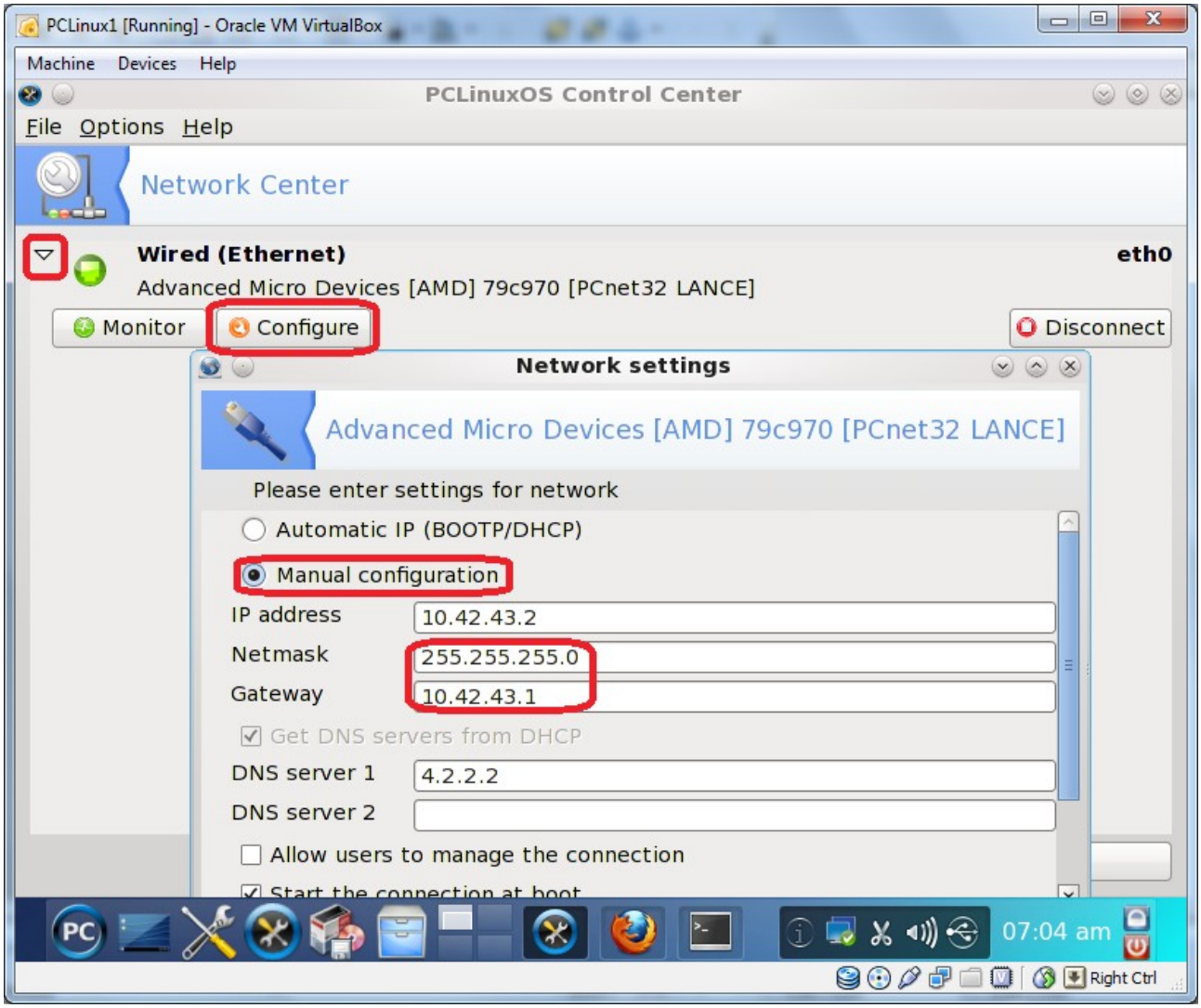
So the IP of the bridge NIC will be changed as shown.



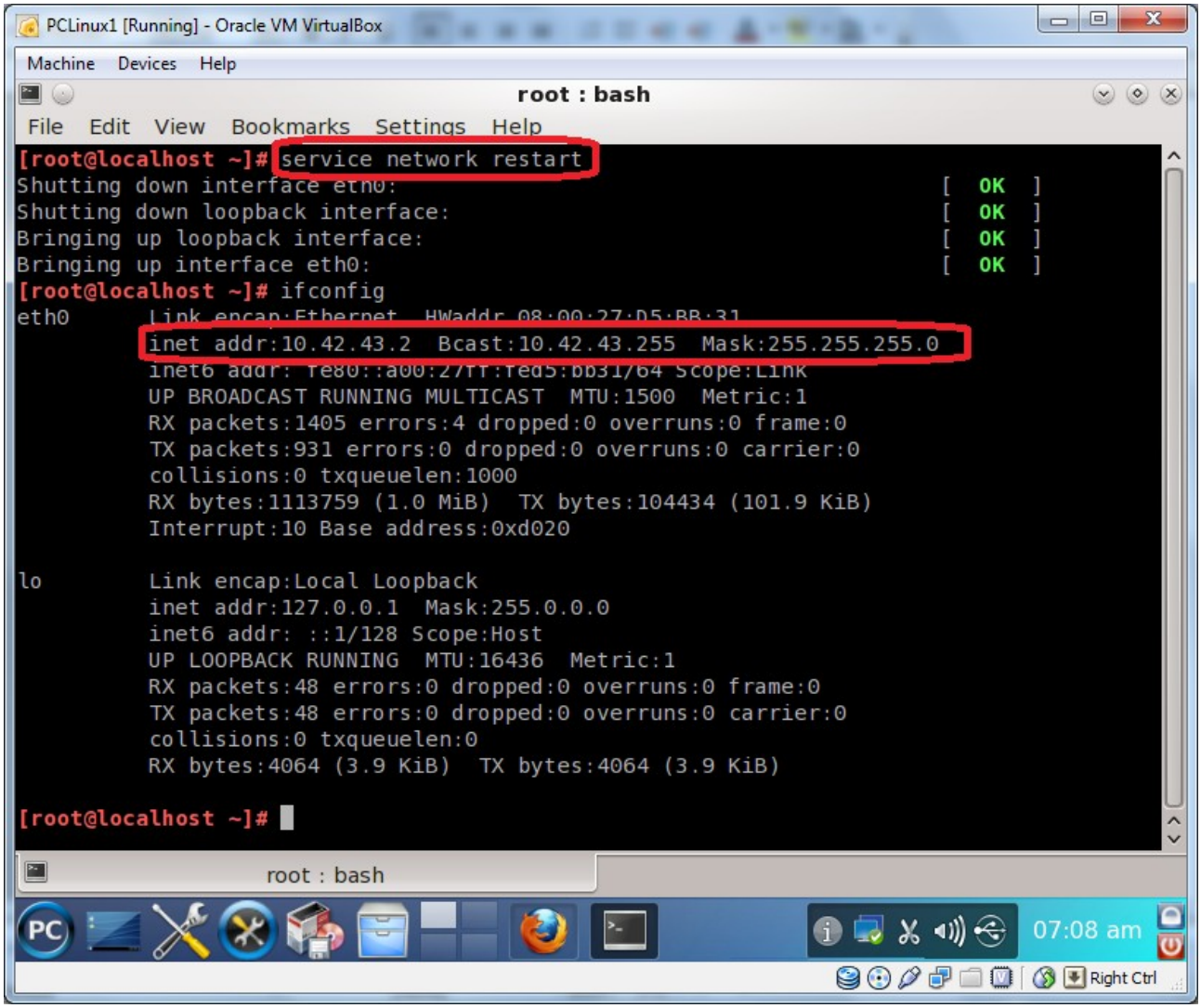
Now we go to the PCLinux machine to change its IP to be on the same network of the bridge connection at the Ubuntu machine.



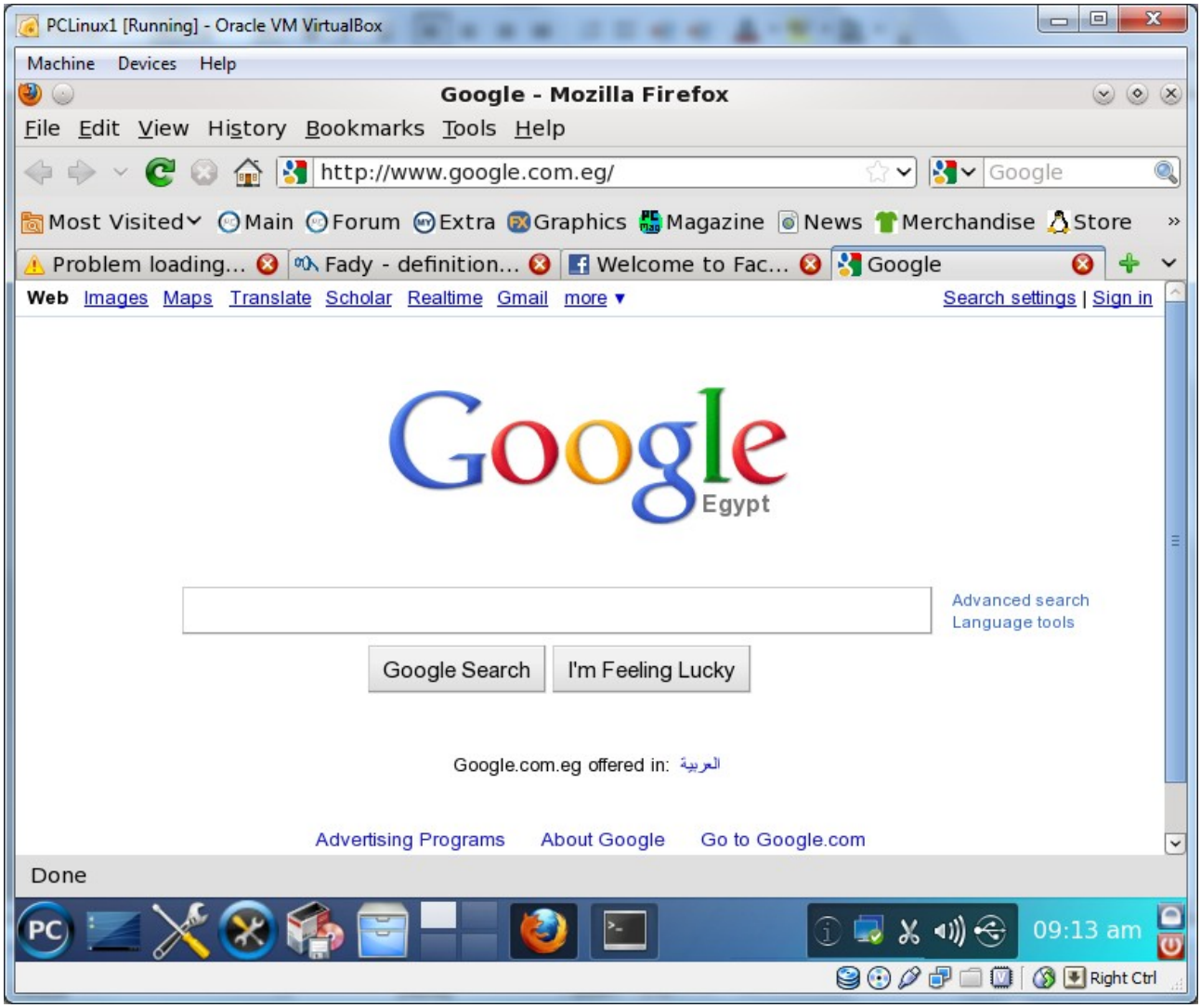
Here we note that the default gateway is the IP of the bridge connection on the Ubuntu machine and the IP is any IP in this network.



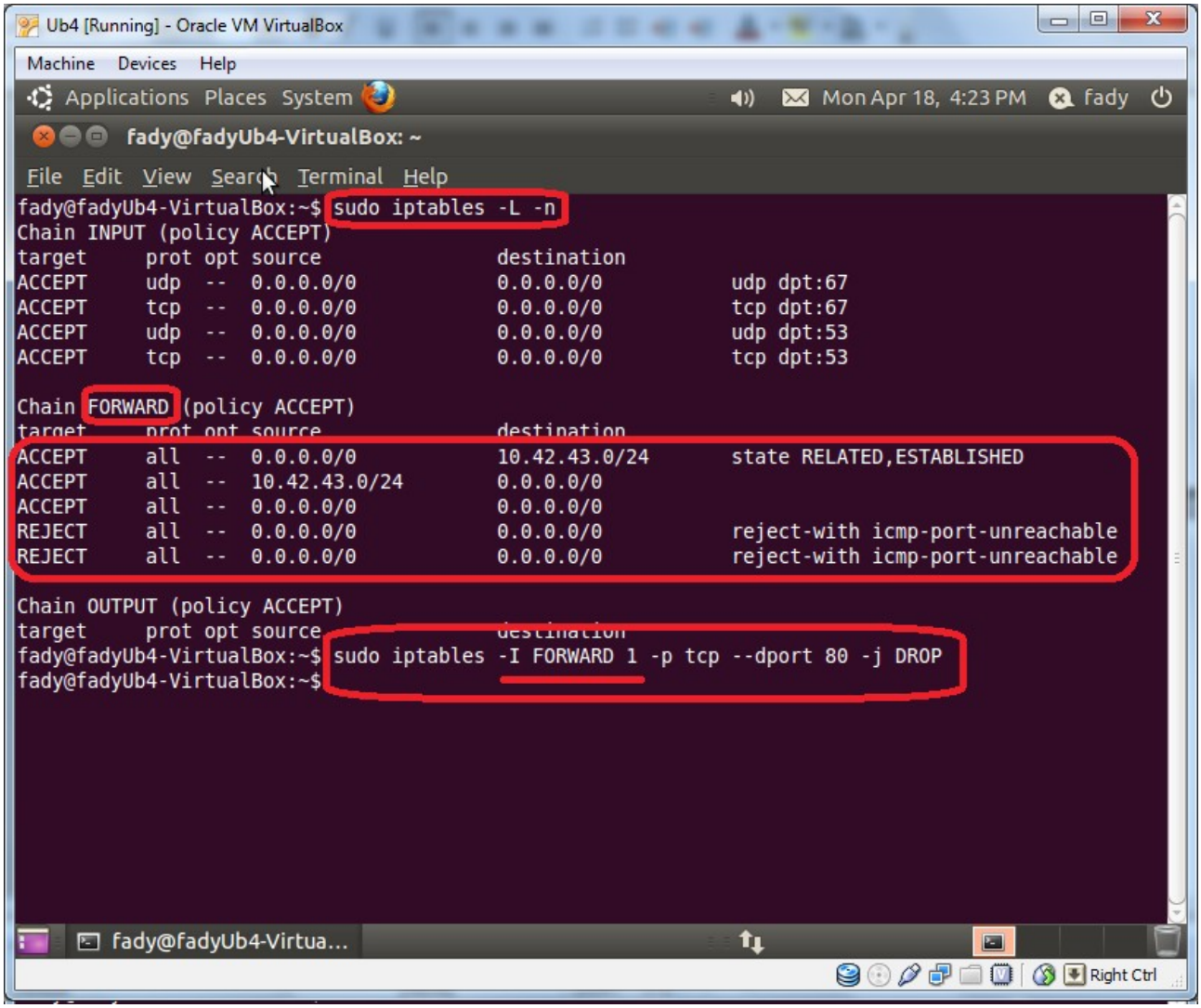
We have now to restart the network service on the PCLinux to activate the IP change as shown.



Now Al7amd llLAH we have Internet on the PCLinux machine.

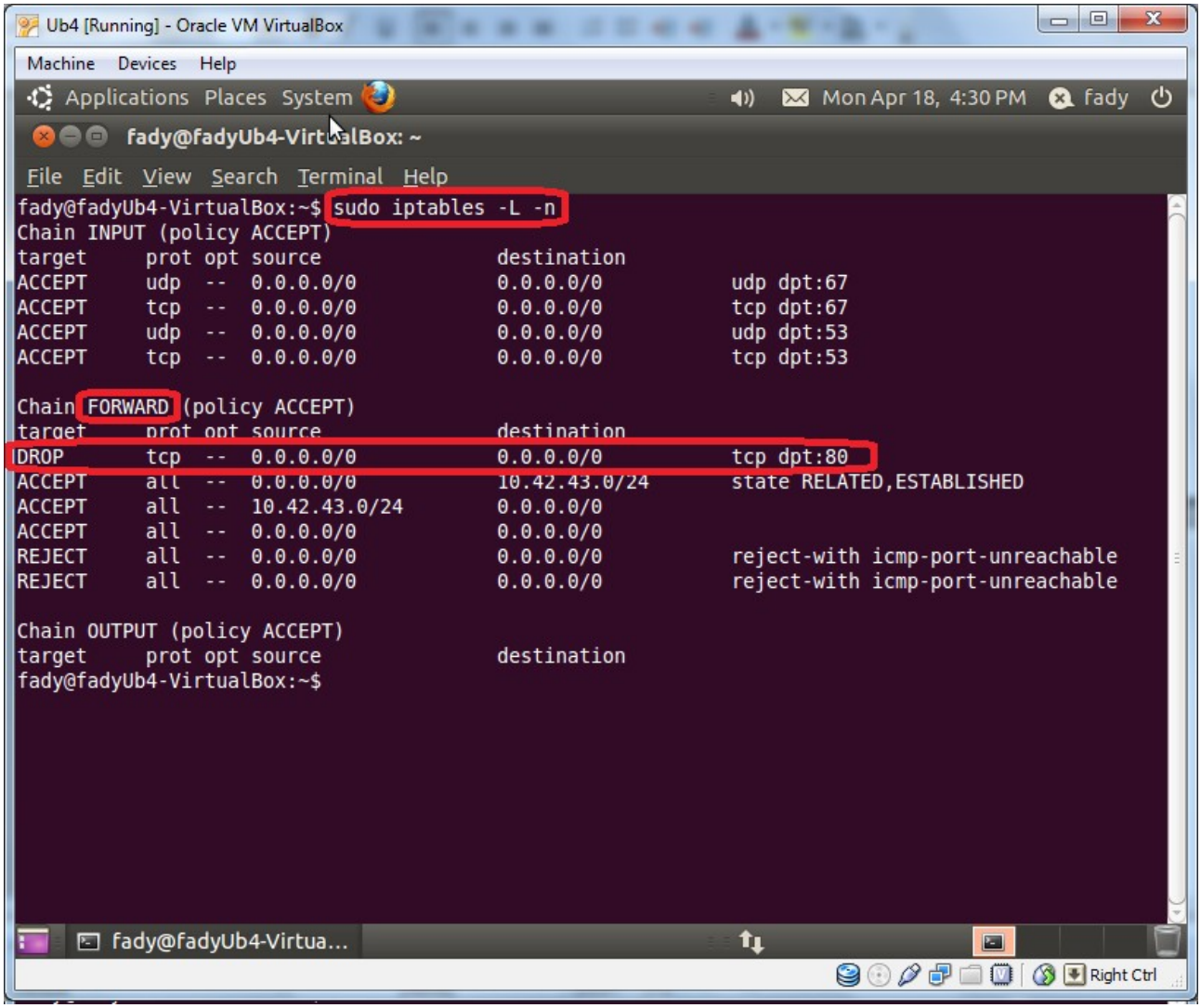


Now if we see the rules of the filter table, we will note that a lot of rules have been added, it is added due to the bridge we make.

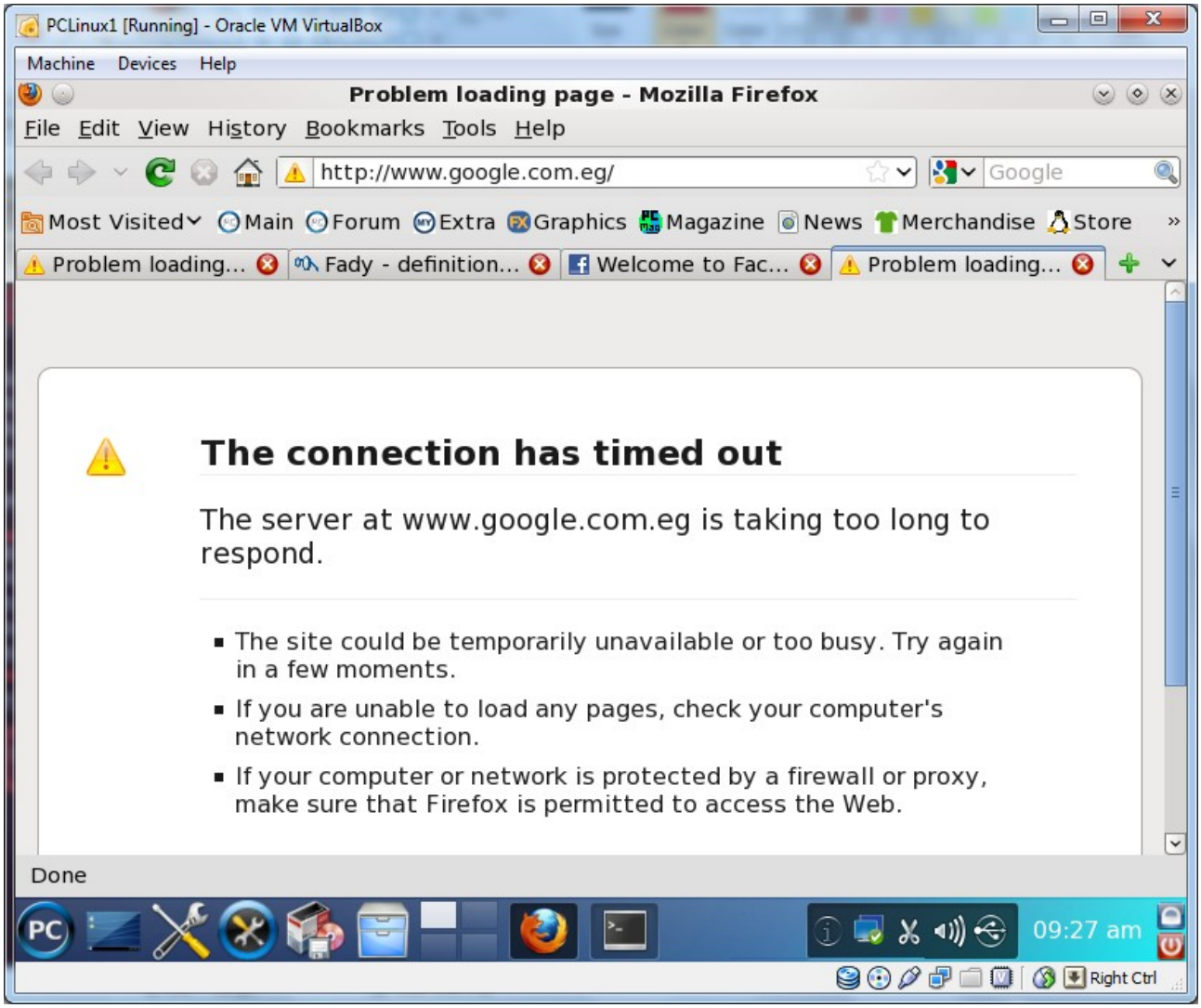


Here also we insert the rule
$sudo iptables -I FORWARD 1-p tcp –dport 80 -j DROP
at the first place of the FORWARD chain as we use -I and 1.

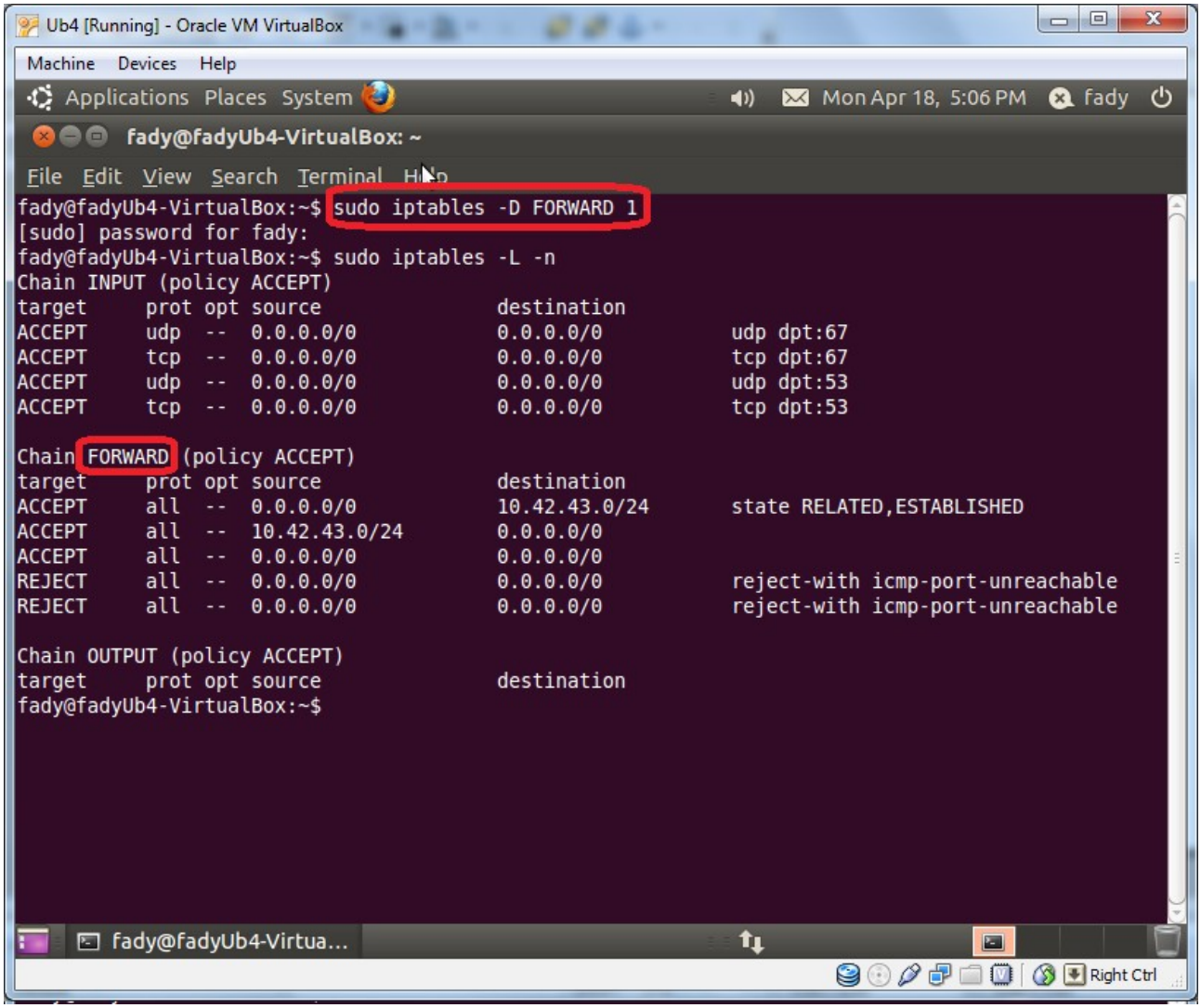
So when we see the rules again, this rule appears in the first place as shown.



This rule will prevent the PCLinux to get to the Internet again.

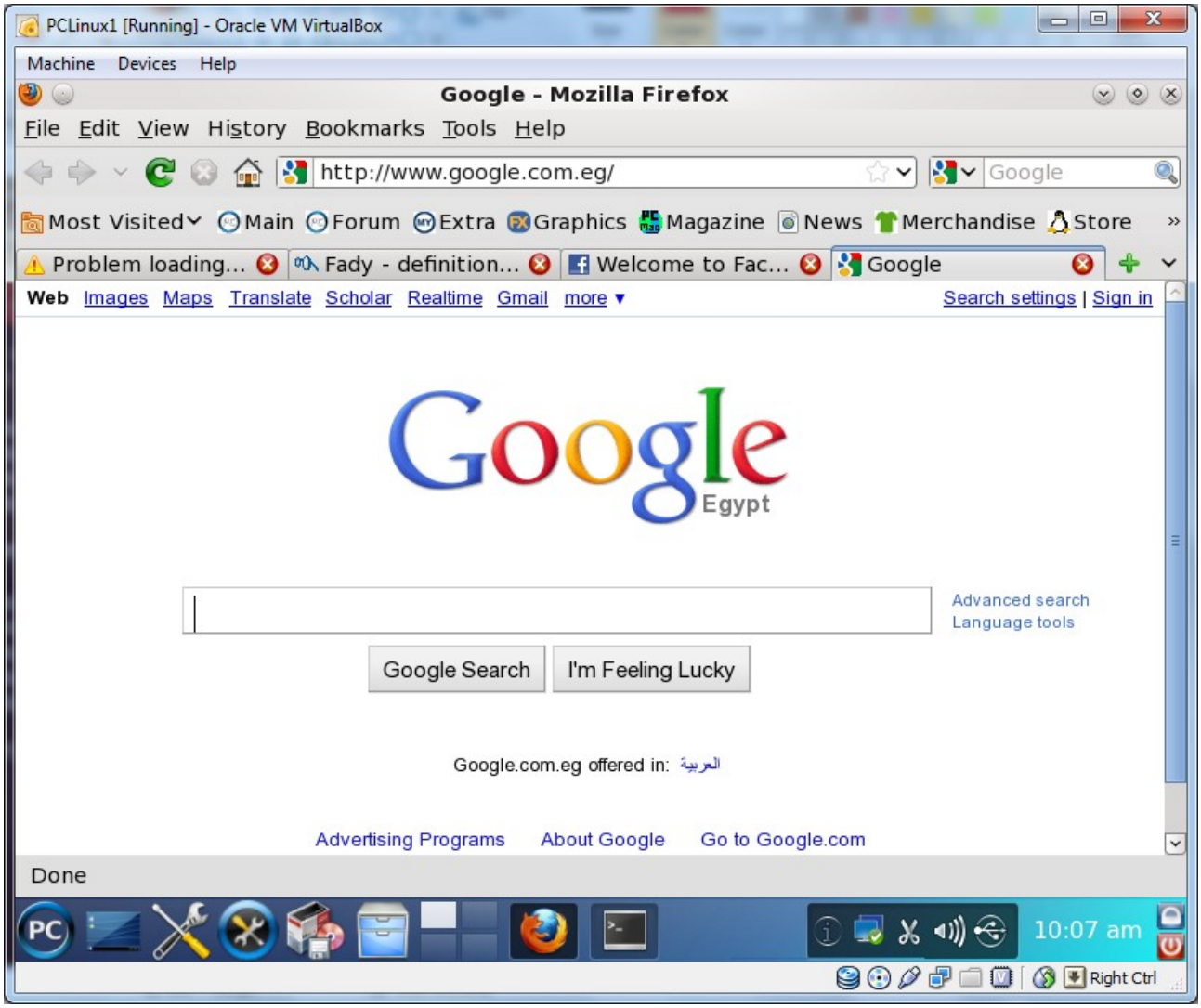


If we want to delete this rule, we can use the command
$sudo iptables -D FORWARD 1



So when we see the rules again we do not see this rule.

After deleting the rule, the Internet return back to the PCLinux again.



Exercise#1

Insert a firewall rule or more than rule to prevent all the clients on the local network from using the facebook.

Hint: The arrange of the rules is important.

To summarize the chains of the ibtables we can use the shown table
http://www.linuxhomenetworking.com/wiki/index.php/Quick_HOWTO_:_Ch14_:_Linux_Firewalls_Using_iptables

Queue Type	Queue Function	Packet Transformation Chain in Queue	Chain Function
filter	Packet filtering	`FORWARD`	Filters packets to servers accessible by another NIC on the firewall.
		`INPUT`	Filters packets destined to the firewall.
		`OUTPUT`	Filters packets originating from the firewall.
nat	Network Address Translation	`PREROUTING`	Address translation occurs before routing. Facilitates the transformation of the destination IP address to be compatible with the firewall's routing table. Used with NAT of the destination IP address, also known as **destination NAT** or **DNAT**.
		`POSTROUTING`	Address translation occurs after routing. This implies that there was no need to modify the destination IP address of the packet as in pre-routing. Used with NAT of the source IP address using either one-to-one or many-to-one NAT. This is known as **source NAT**, or **SNAT**.
		`OUTPUT`	Network address translation for packets generated by the firewall.

mangle	TCP header modification	`PREROUTING`
		`POSTROUTING`
		`OUTPUT`
		`INPUT`
		`FORWARD`
			Modification of the TCP packet quality of service bits before routing occurs.

Custom Chains (User-Defined Chains)

http://centoshelp.org/networking/iptables-advanced/

One powerful feature of iptables is the ability for the user to create new chains, in addition to the built-in ones. By convention, user-defined chains are lower-case to distinguish them. When a packet matches a rule whose target is a user-defined chain, the packet begins traversing the rules in that user-defined chain. If that chain doesn't decide the fate of the packet, then once traversal on that chain has finished, traversal resumes on the next rule in the current chain.

Custom chains can be specified as a target to jump to. For example, you could create a so-called whitelist for trusted IP address, and a blacklist for evil nodes on the Internet. To create the chains, you would give the following commands:-

```
iptables -N whitelist
iptables -N blacklist
```

To add a rule to these chains (or any other chain), use:-

```
iptables -A whitelist -s 192.168.0.0/24 -j ACCEPT
iptables -A blacklist -s 207.46.130.0/24 -j DROP
iptables -A blacklist -s 207.46.250.0/24 -j DROP
```

Then, specify these chains as a target in your INPUT, FORWARD and/or OUTPUT chain:-

```
iptables -A INPUT -j whitelist
iptables -A INPUT -j blacklist
iptables -A OUTPUT -j whitelist
iptables -A OUTPUT -j blacklist
iptables -A FORWARD -j whitelist
iptables -A FORWARD -j blacklist
```

User-Defined Chains don't have any default policies.

Accepting or denying connections to the firewall

http://centoshelp.org/networking/iptables-advanced/

Let's suppose you have a mail server directly connected to the Internet (ergo, no modem or router), and you want the machine to be completely secure, at least as far as the firewall is concerned. A mail server where mail is delivered typically uses port 25 (SMTP), and mail servers used by users to read their mail typically use port 110 (POP) and 143 (IMAP). Assuming you have a remote console, the firewall does not necessarily need to allow connections to port 22 (SSH). We also assume no mail has to be send via this mail server. So, connections from anywhere to your mail server's port 25, 110 and 143 should be allowed by the firewall.

The commands we use can be saved in a file with extesion *.sh (as a script), then we just run this script by the command **sudo sh filename.sh.** The first line tells the kernel which shell to use to run this script by specifing its path.

```
#!/bin/bash

#Load the connection tracker kernel module which enables iptables to tracke the
#connections by caching the related information and examining its status.
#kernel module is like MS windows driver.
modprobe ip_conntrack

#Set the default target to DROP of all chains involved here.
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT DROP

#Accept from the loopback as input and Output interface
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

#DROP if the SYN flag is not set and match by state only the new connections.
iptables -A INPUT -p tcp ! --syn -m state --state NEW -j DROP

#Allow the traffic from a known IP on the whitelist chain
iptables -N whitelist
iptables -A whitelist -s 1.2.3.4 -j ACCEPT

#Drop the traffic from private IPs and a known spammer on the blacklist chain.
iptables -N blacklist
iptables -A blacklist -s 10.0.0.0/8 -j DROP
iptables -A blacklist -s 172.16.0.0/12 -j DROP
iptables -A blacklist -s 169.254.0.0/16 -j DROP
iptables -A blacklist -s 192.168.0.0/16 -j DROP
iptables -A blacklist -s spammer.com -j DROP

#Get our whitelist and blacklist into place
iptables -A INPUT -j whitelist
iptables -A INPUT -j blacklist
iptables -A OUTPUT -j whitelist
iptables -A OUTPUT -j blacklist

# Allow new SMTP, POP, and IMAP connections
iptables -A INPUT -p tcp --dport 25 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -p tcp --sport 25 -m state --state ESTABLISHED -j ACCEPT
iptables -A INPUT -p tcp --dport 110 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -p tcp --sport 110 -m state --state ESTABLISHED -j ACCEPT
iptables -A INPUT -p tcp --dport 143 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -p tcp --sport 143 -m state --state ESTABLISHED -j ACCEPT
```