**Lab 7**

**Due Date = March 19, 2025. Marks = 4**

Color Codes: Group 1 = Red, Group 2 = Blue, Group 3 = Green, Group 4 = Black, Group 5 = Purple, Group 6 = Orange, Group 7 = White or Grey, Group 8 = Red, Group 9 = Blue, Group 10 = Green, Group 11 = Black, Group 12 = Purple, Group 13 = Orange.

**Marking Criteria: 1.5 marks for submission and 2.5 marks for the viva. Viva will be conducted from any of the questions.**

In this lab, you will be able to practice the firewall commands using iptables and see their effect. The iptables rule is specified using the following format

iptables -t TABLE\_NAME -A Chain\_name several\_conditions -j action

It means that in **chain** *Chain\_name* of **table** *TABLE\_NAME*, the rule below will be appended:

if *several\_conditions* is satisfied, then *action* will be taken.

Here **adding** the rule is specified by –A. You can also change to -D for **deletion** or –I for **insertion** (at somewhere of the chain). If -t does not occur, the default table **filter** is assumed.

**Note 1**: For each problem, write details and use the screenshots as evidence (for your solution). If any of the questions does not require an answer/explanation, then just perform the steps and provide related screenshots. Submit MS word file only.

**Note 2**: In the following, we use sudo to show the root privilege. But if we use docker-compose to simulate VMs, VM might automatically have the root privilege and so **sudo** is not needed.

**Note 3**: In our docker-compose, we simulate two networks: **10.9.0.0/24** (hosts: 10.9.0.5, 10.9.0.11) and **192.168.60.0/24** (hosts: 192.168.60.5, 192.168.60.6, 192.168.60.7, 192.168.60.11), where 10.9.0.11 and 192.168.60.11 belong to **router**. You can use command **route** to see the routing table on each VM.

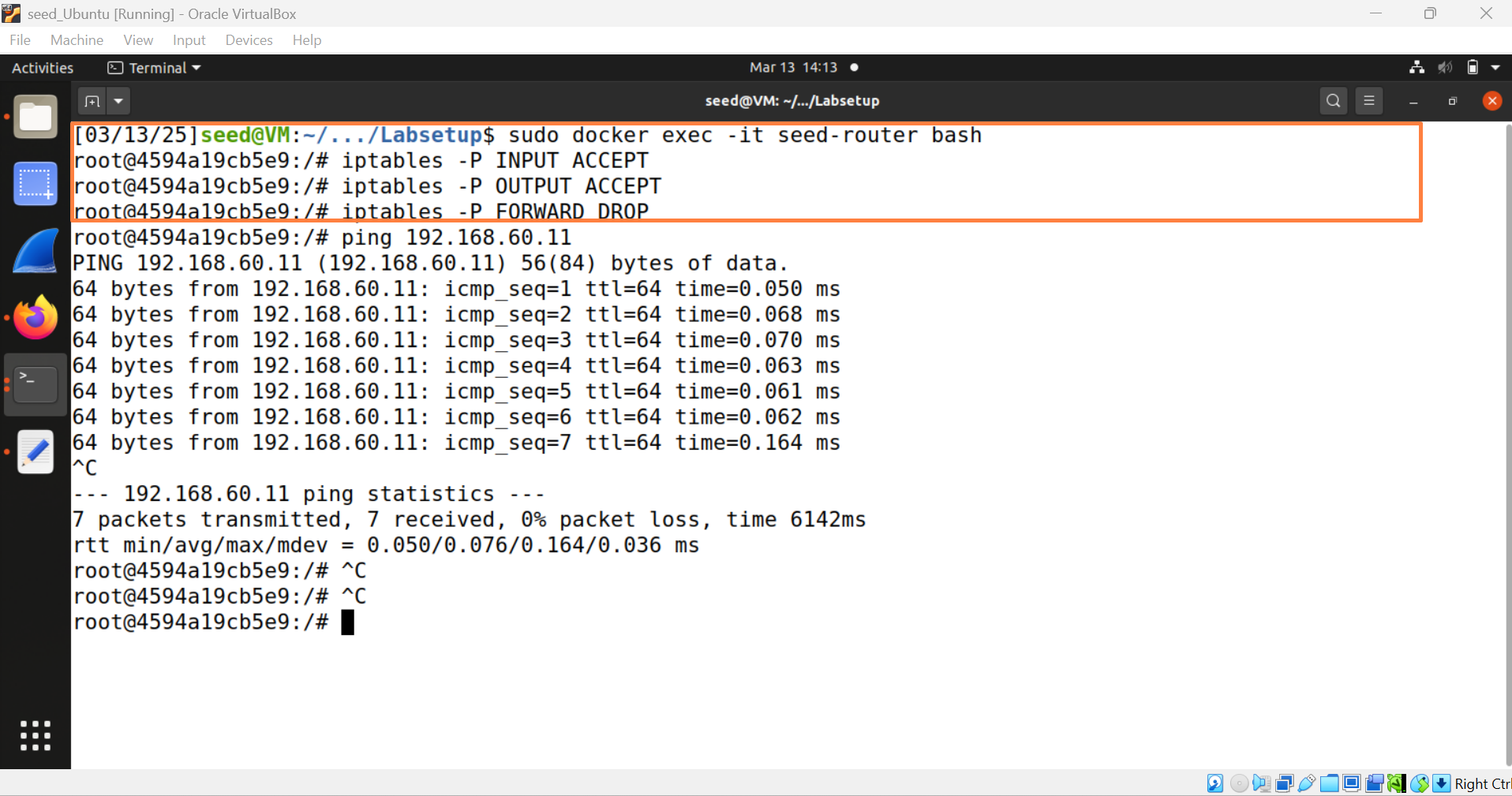
**Note 4**: All firewall rules are running on **router** VM.

**1.** Use the following commands on **router** to set the default policies for atable**.**

**sudo iptables –P INPUT ACCEPT**

**sudo iptables –P OUTPUT ACCEPT**

**sudo iptables –P FORWARD DROP**



**Commands:**

**sudo docker exec -it seed-router bash**

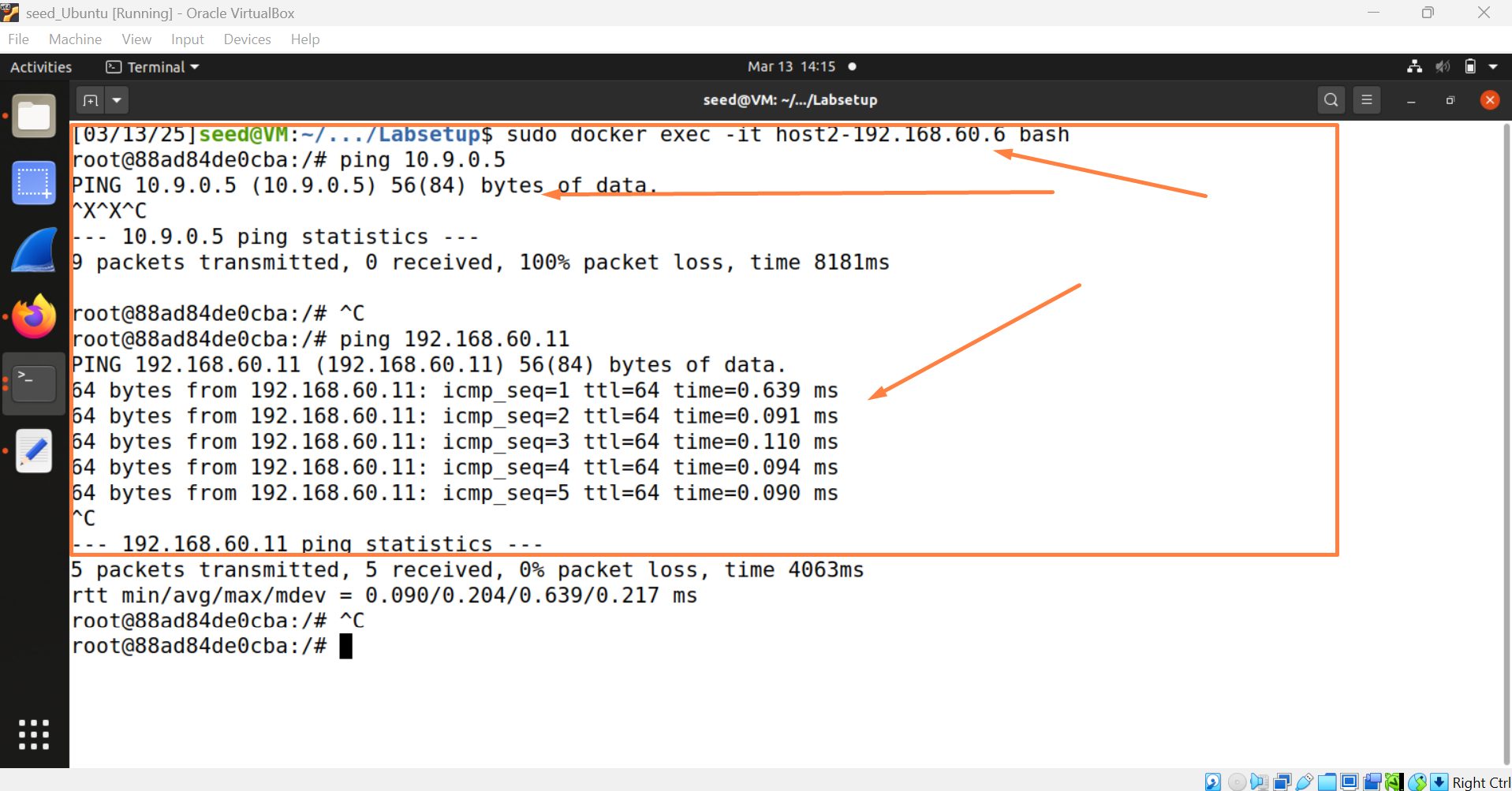
**iptables -P INPUT ACCEPT**

**iptables -P OUTPUT ACCEPT**

**iptables -P FORWARD DROP**

Recall, INPUT is to check incoming packet; OUTPUT is to check outgoing packet; FORWARDING is to check the passing packet (at router). Further, the commands assume the default table **filter (-t filter)**.

* On 192.168.60.6, run **$ ping 10.9.0.5** and then ping 192.168.60.11. Does it succeed? Explain your observation.



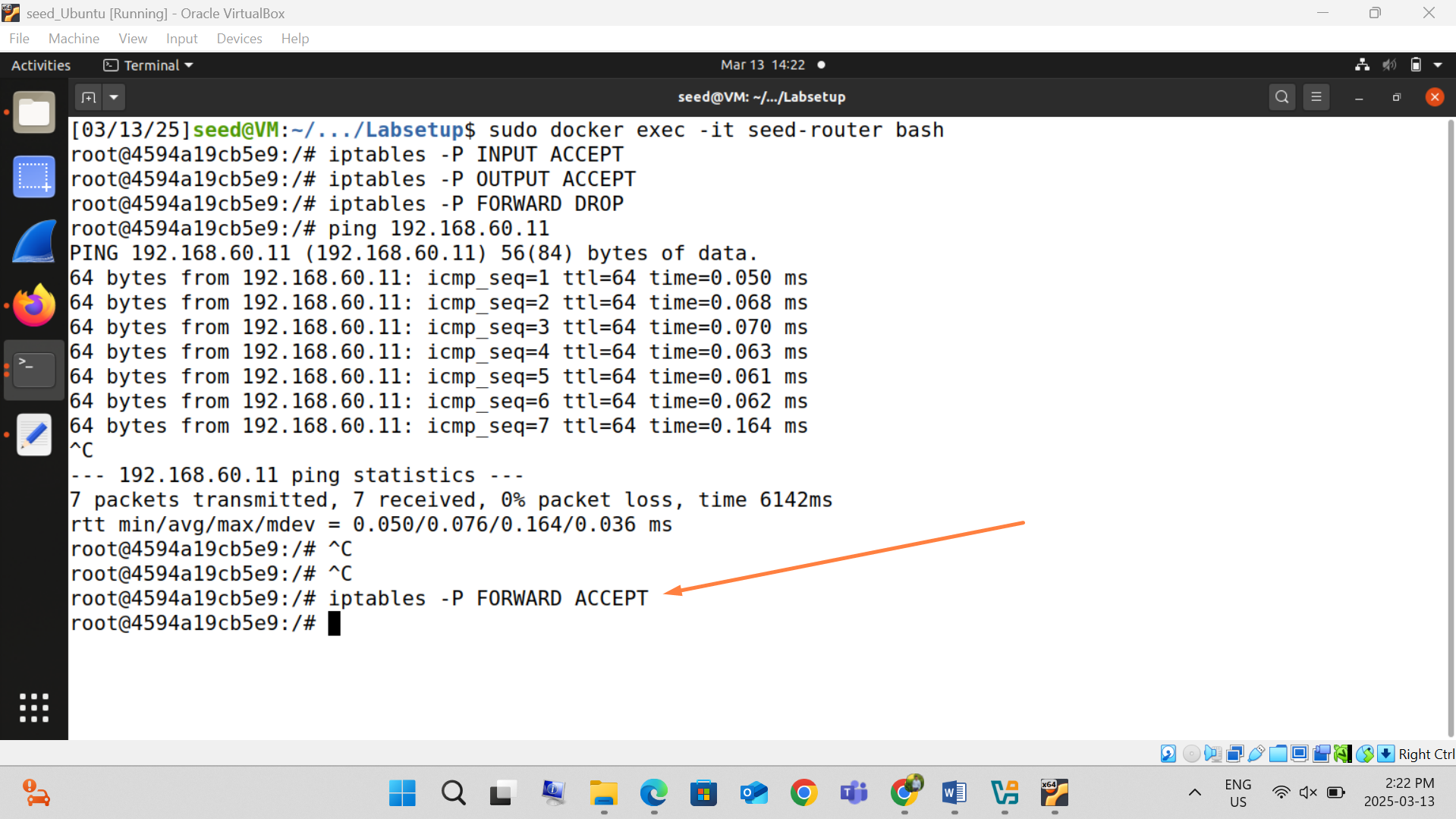
Command:

sudo docker exec -it host2-192.168.60.6 bash

ping 192.168.60.11

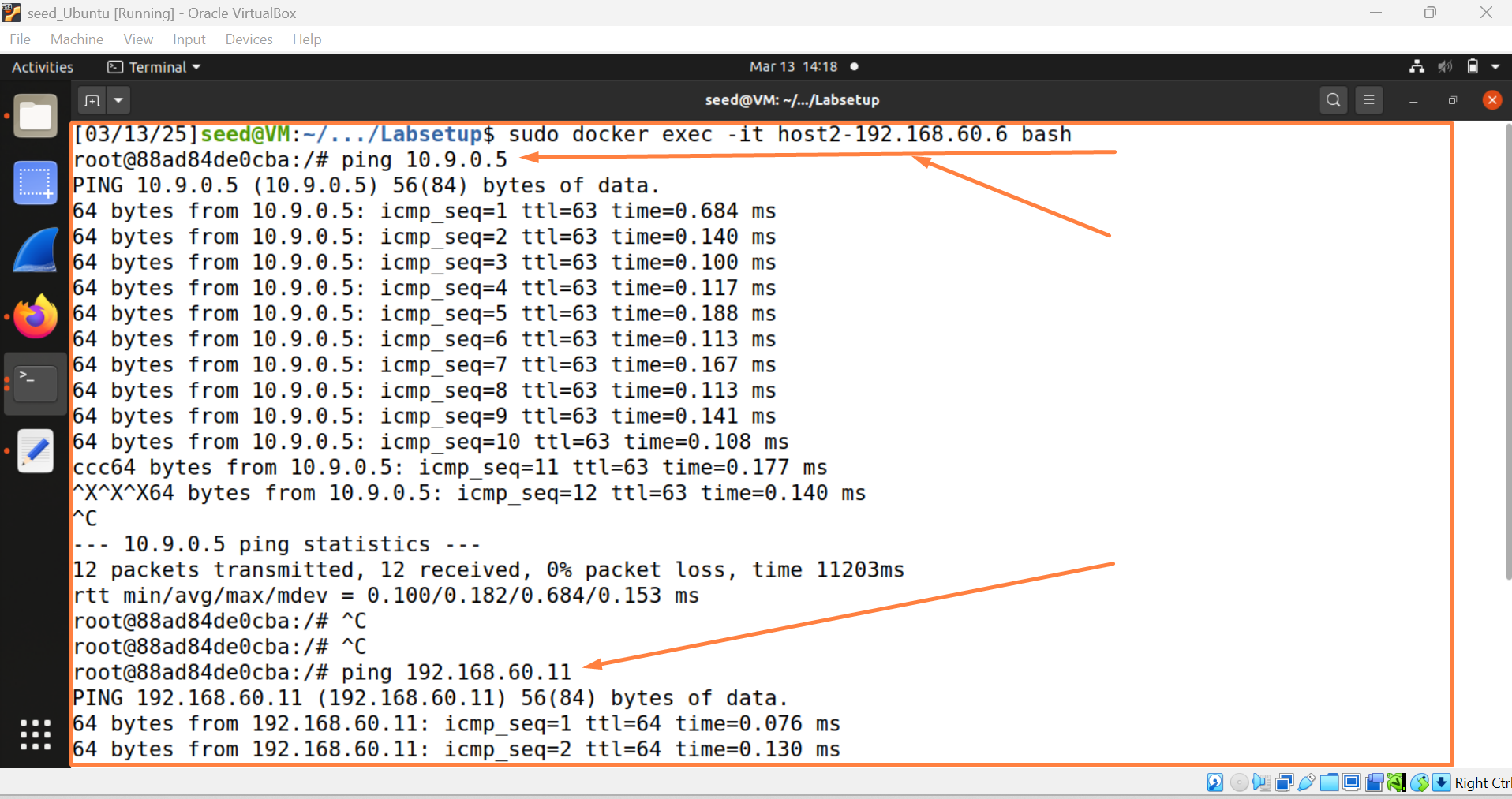
ping 10.9.0.5

* Change **DROP** to **ACCEPT,** for FORWARD case. Try the pings in the above step again. Now, does it succeed? Explain your observation.



Command:

iptables -P FORWARD ACCEPT



Command:

ping 10.9.0.5

sudo docker exec -it host2-192.168.60.06 bash

**2**. [**blocking an IP**]

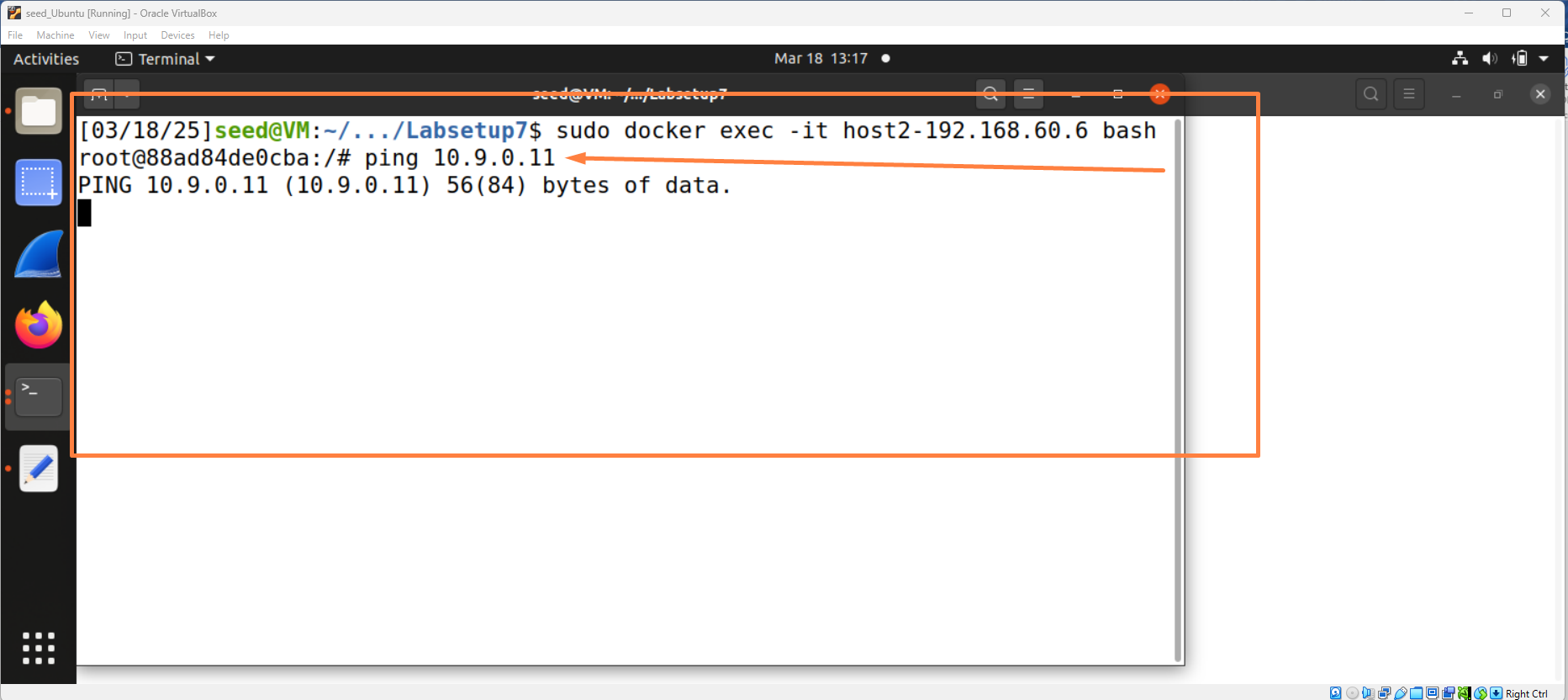
* On 10.9.0.11, if we want to block packets **from** an ip address IP1, use command

**sudo iptables -A INPUT -s IP1 -j DROP**

**/\***this uses INPUT chain because it is incoming packet**\*/**



**On IP1, ping** 10.9.0.11**and what can be observed? Explain.**

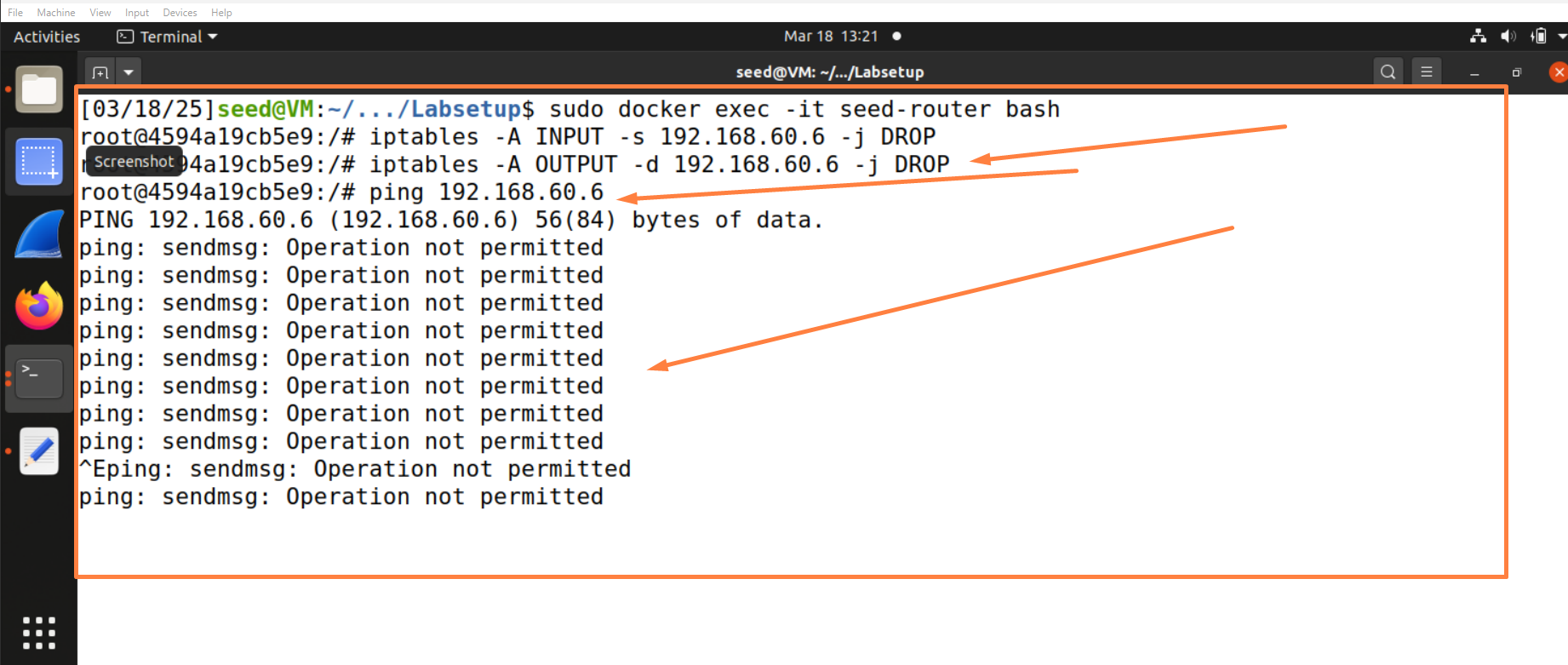


* On 10.9.0.11, if we want to block packets **to** an ip address IP1, use command

**sudo iptables -A OUTPUT -d IP1 -j DROP**

**/\***this uses OUTPUT chain because it is outgoing packet**\*/**

**On** 10.9.0.11**, ping IP1 and what can be observed? Explain.**



**3. [List all rules] do it on Router.**

* You can see all the firewall rules by the following command

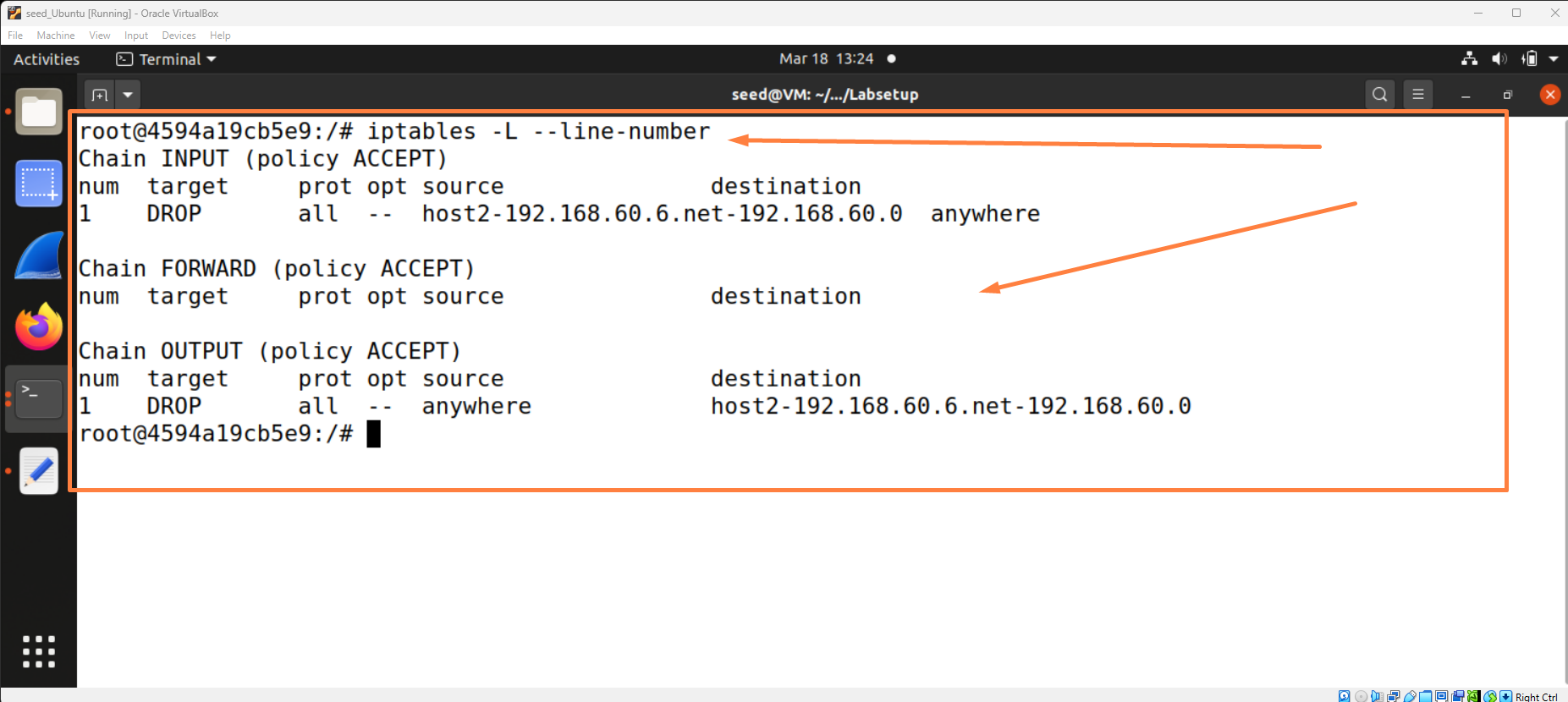
**$ sudo iptables -L**

/\* again, this assume filter table (i.e., **-t filter**) by default\*/



* You can see all the firewall rules in each chain with index number. The index will be used for other operation such as deletion later.

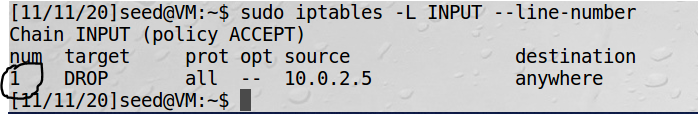
**$ sudo iptables -L --line-number**



**4. [Delete a rule]** On **Router,** delete a rule in a chain (such as INPUT) in two steps:

first, list with index:

**$ sudo iptables –L INPUT --line-number**

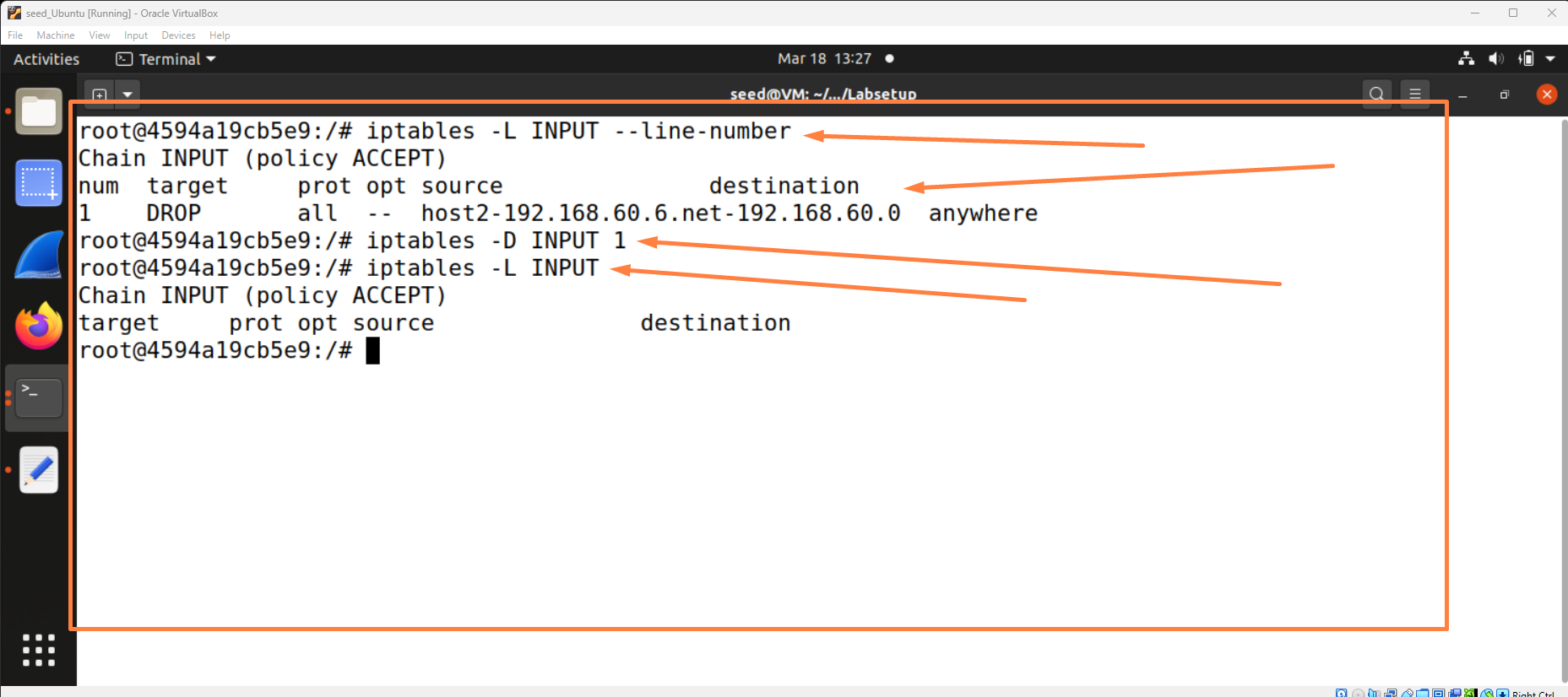


Then, delete the rule using the index:

**$sudo iptables -D INPUT 1**

Now use the method to delete the first rule in your current INPUT table and then

**$ sudo iptables -L INPUT** to verify whether rule 1 is deleted or not.

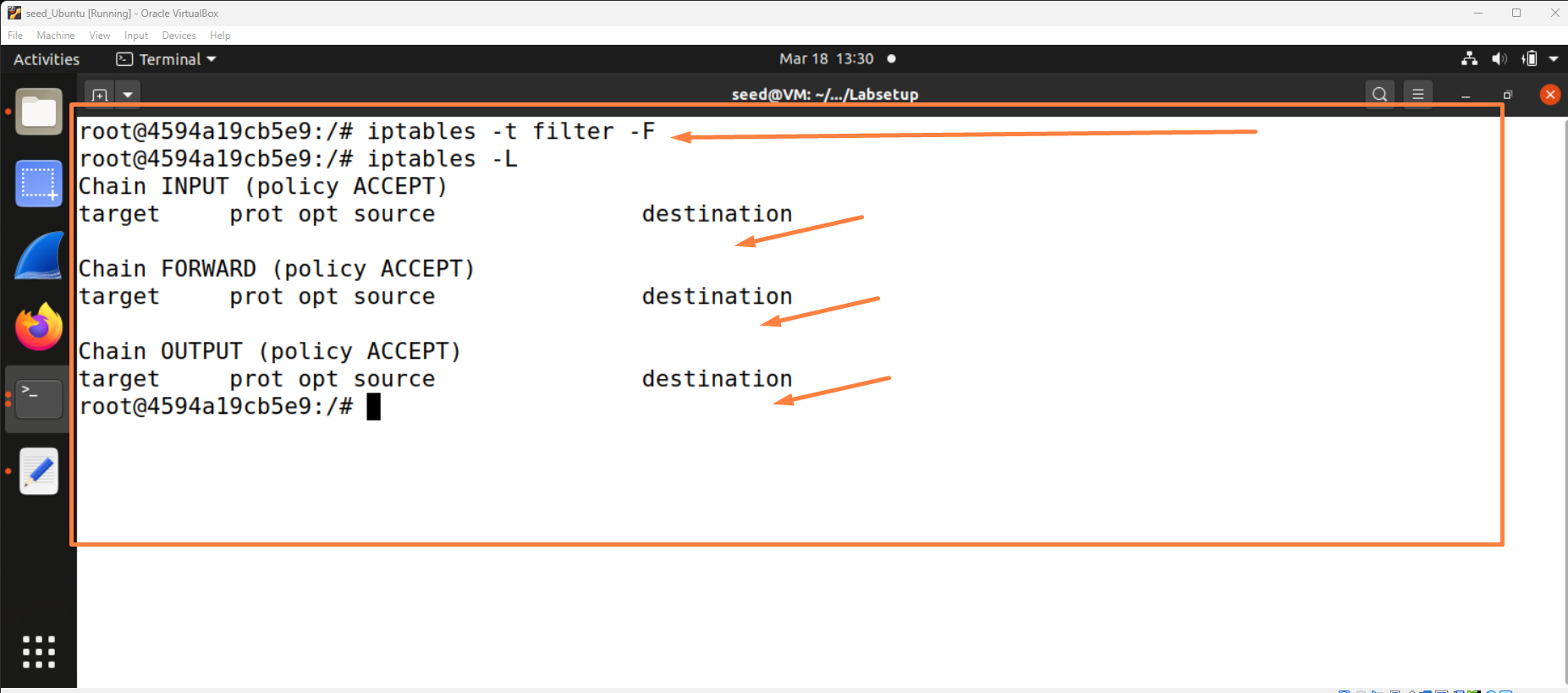


**5. [Delete all rules in a TABLE]** On **router,** flush the rules in a table (e.g., **filter**):

**$sudo iptables -t filter -F**

/\*again,**-t filter** can be omitted\*/

Then, run **$sudo iptables -L** and you will not see any rule.



**6.** [**Drop all incoming connections, except ssh**] On **router**, block incoming connections to any service except for **ssh**. To do this, we can set default policy for INPUT chain of filter Table to be DROP and then specify a rule to accept incoming telnet connection.

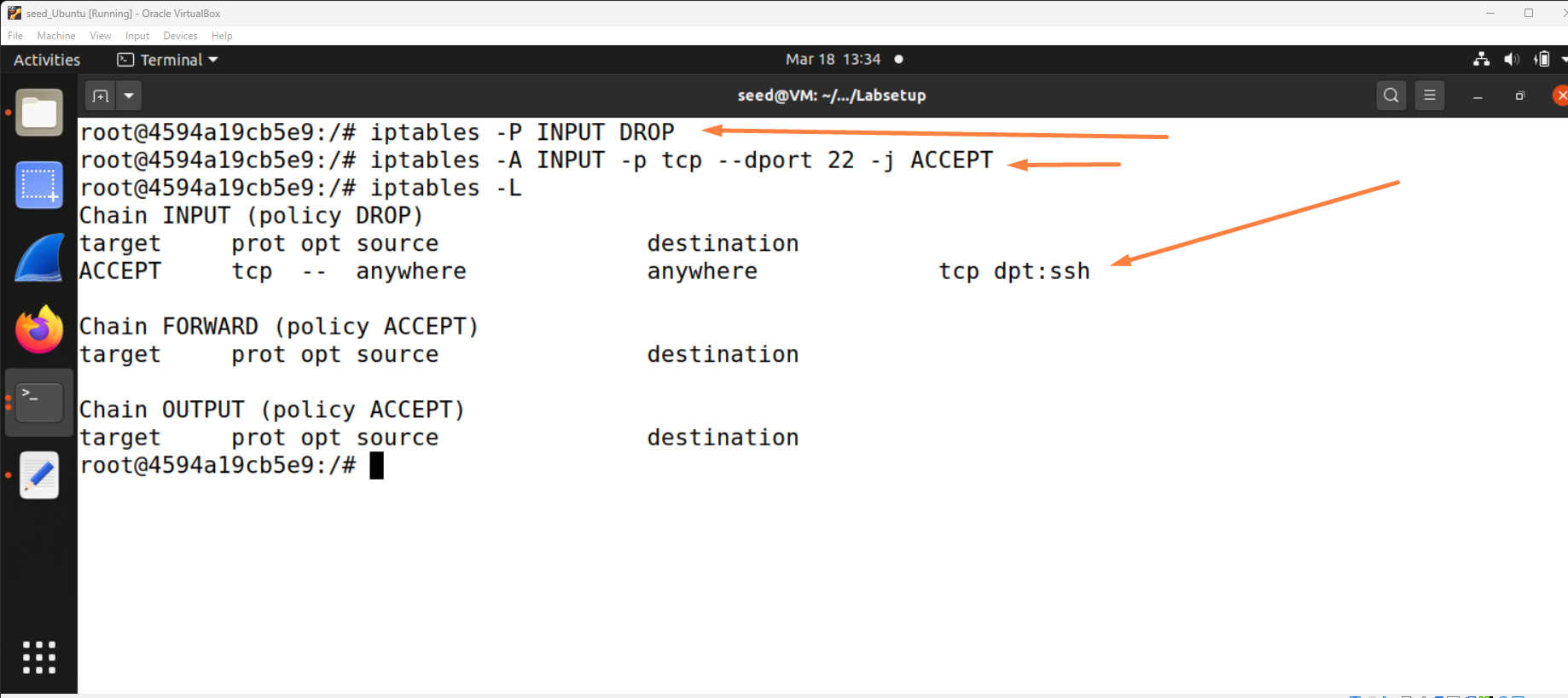
**$ sudo iptables -P INPUT DROP**

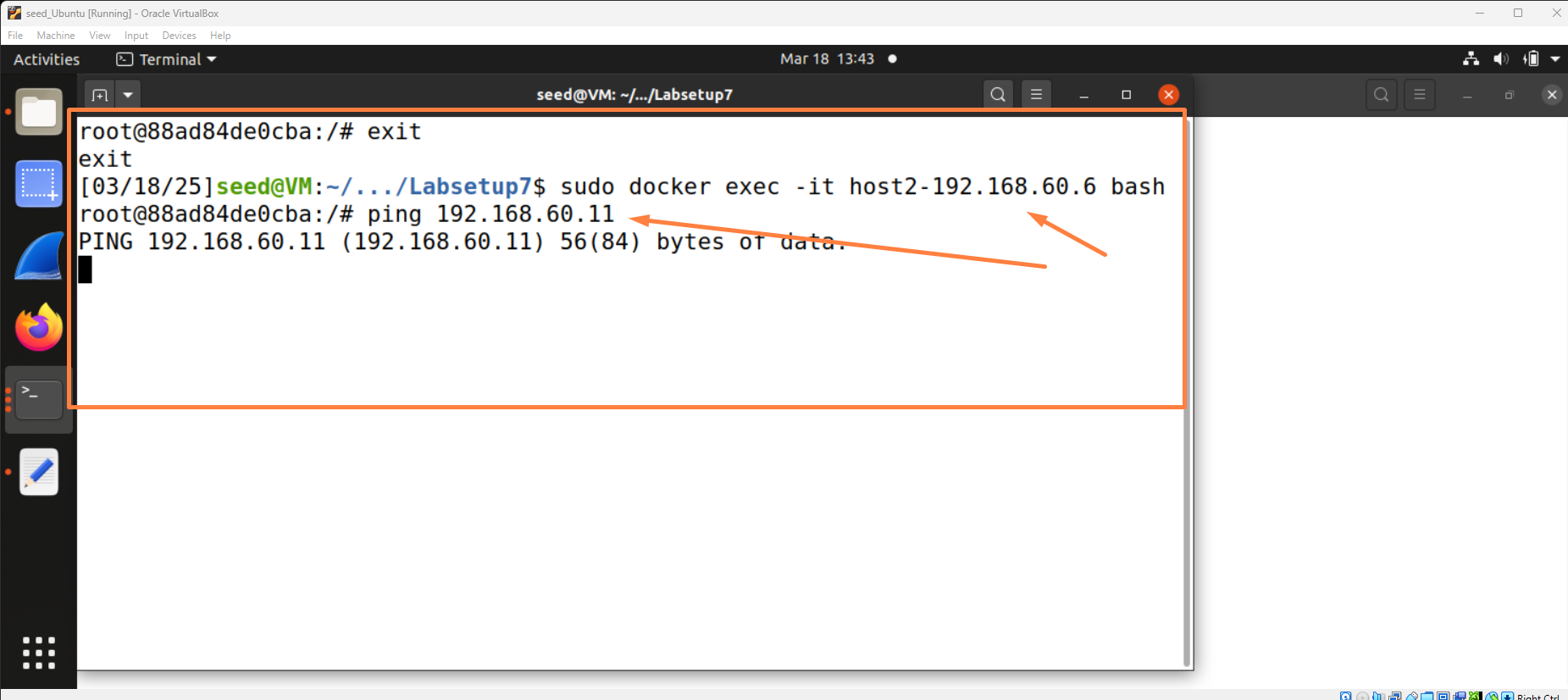
**$ sudo iptables -A INPUT -p tcp - -dport 22 -j ACCEPT**

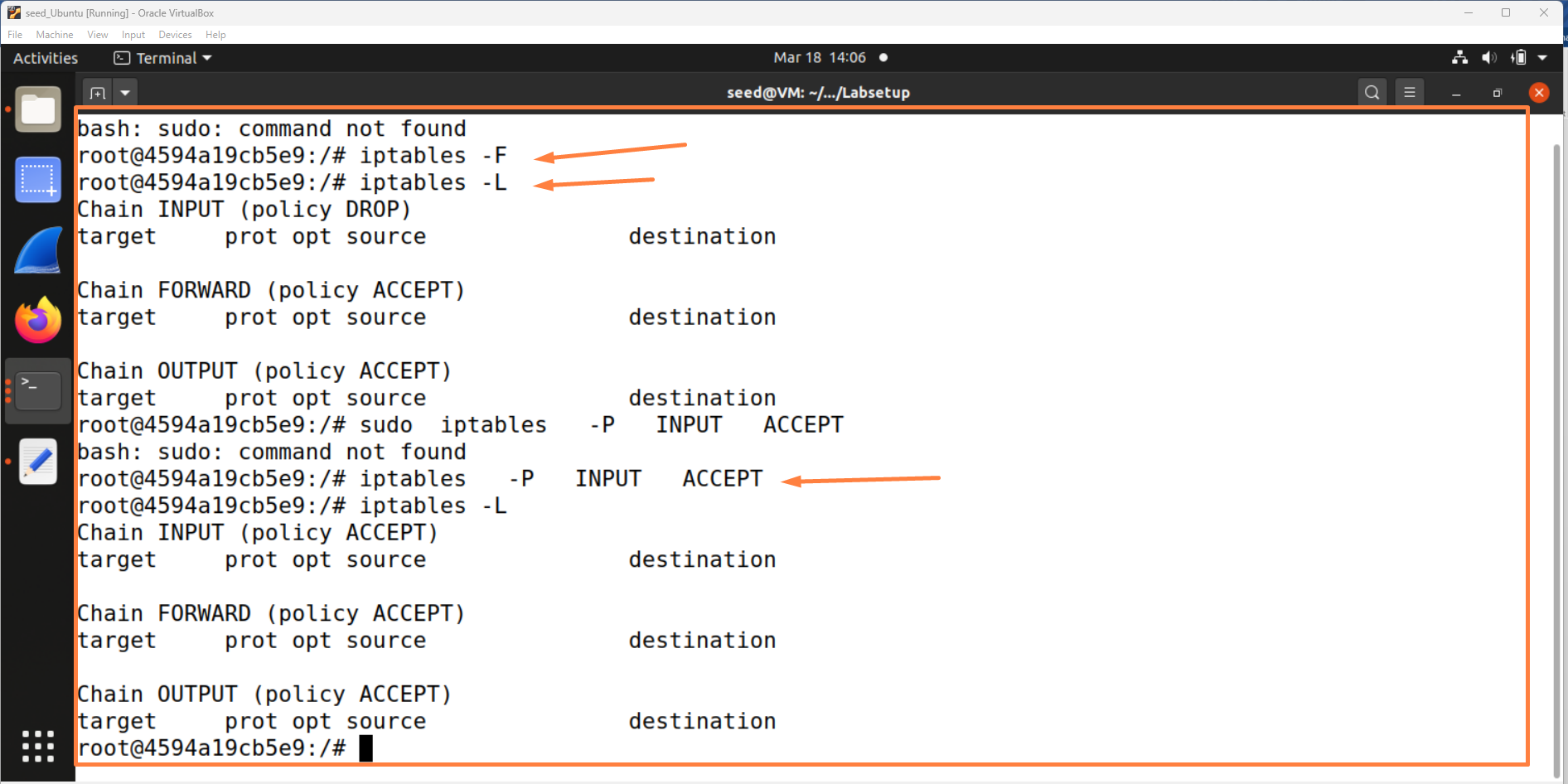
/\* A default policy is applied only if all the rules in the chain have been executed without making a decision (either ACCEPT or DROP or REJECT). For example, if we telnet to router, then the rule does not ACCEPT but also not REJECT. So the default policy applies. Note: here **-p** stands for protocol. \*/

Then, ping and ssh to 192.168.60.11 (from other VM). Does it succeed (ssh or ping)? Explain.

/\*after this problem, run **$ sudo iptables -F** to flush all rules in **filter** table and recover the default policy: **$ sudo iptables -P INPUT ACCEPT** \*/





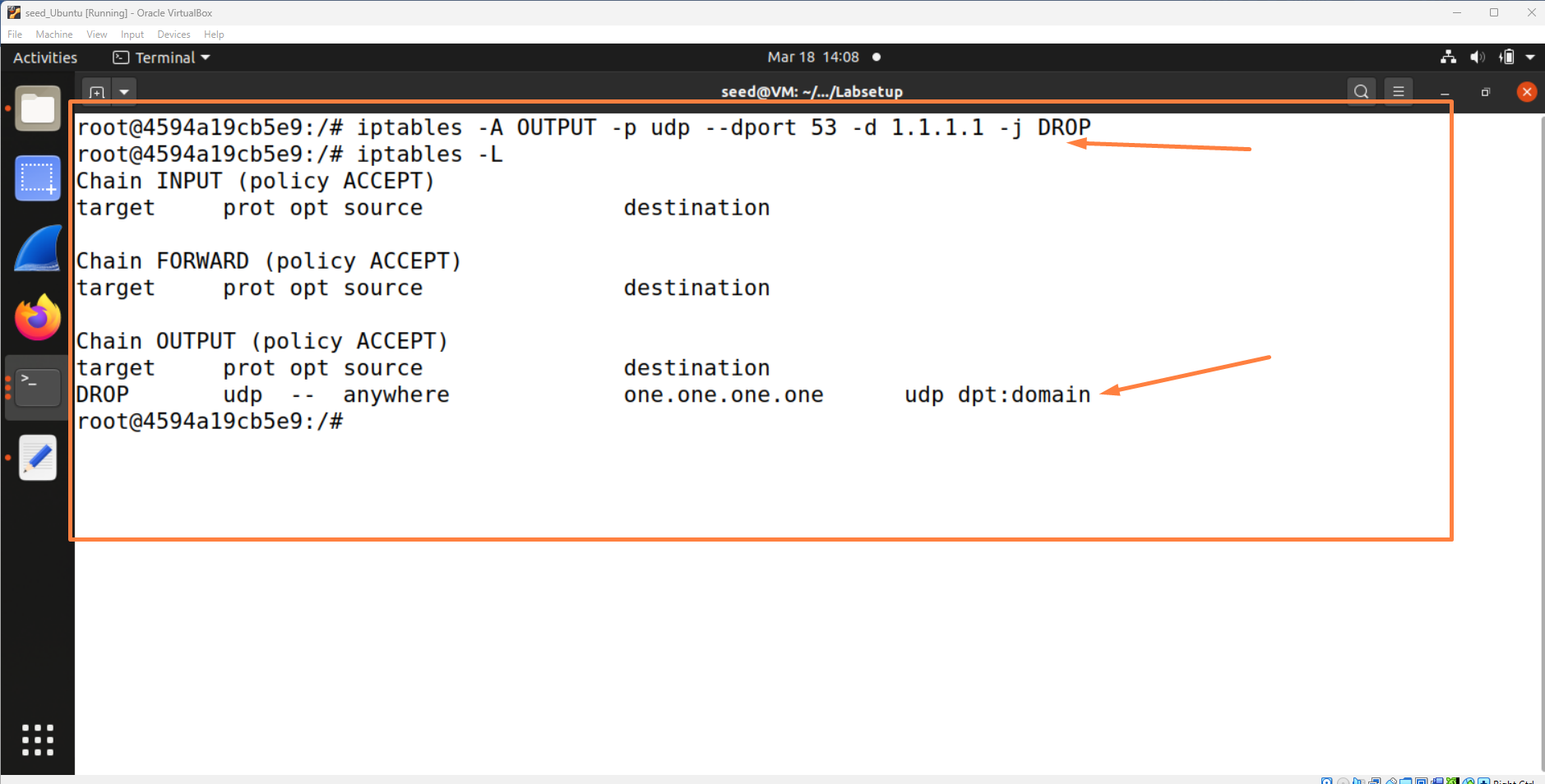


**7.** [**drop outgoing DNS request to 1.1.1.1**] In this case, since it is outgoing packet, we add rule to OUTPUT chain. Since it is DNS request, the destination should be the DNS server, which has a port number 53. Finally, since DNS is implemented using UDP, we use protocol UDP. Hence, we add the following rule:

**$ sudo iptables -A OUTPUT -p udp - - dport 53 -d 1.1.1.1 -j DROP**

Then, try **$ dig** [**www.uwindsor.ca**](http://www.uwindsor.ca) and **dig @1.1.1.1** [**www.uwindsor.ca**](http://www.uwindsor.ca). Does it succeed?

/\* delete the rule in order not to affect the following experiment \*/



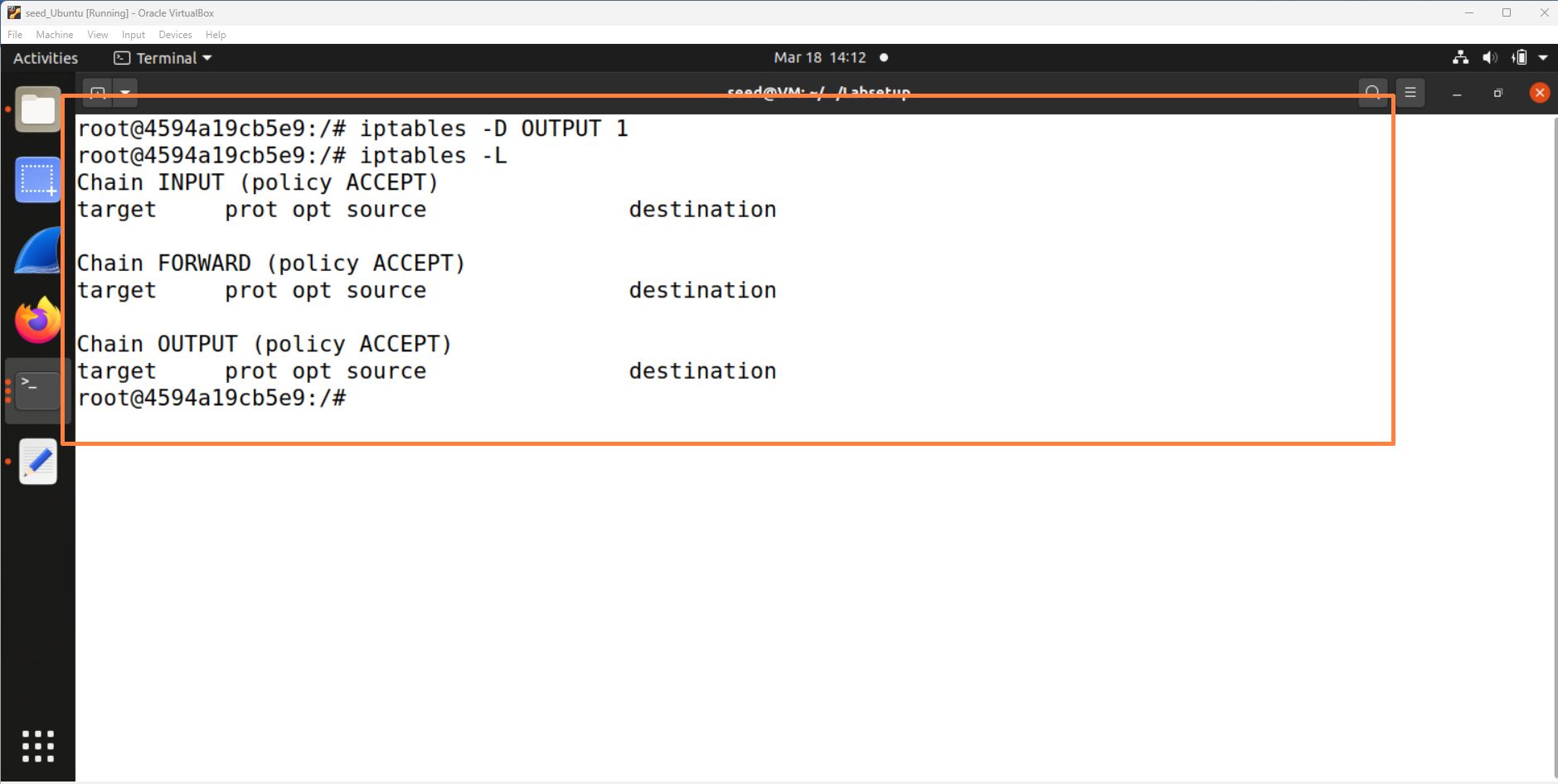


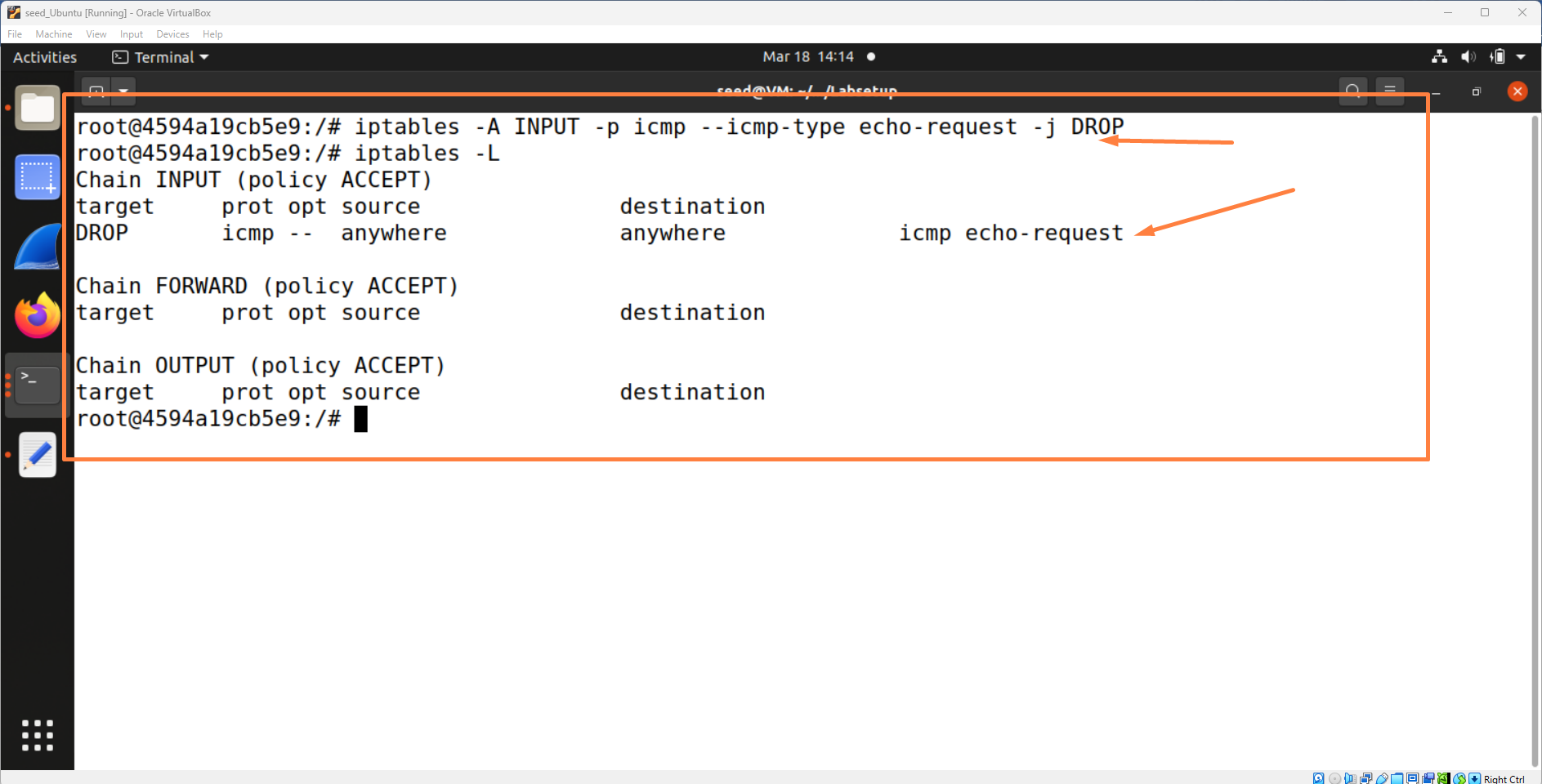


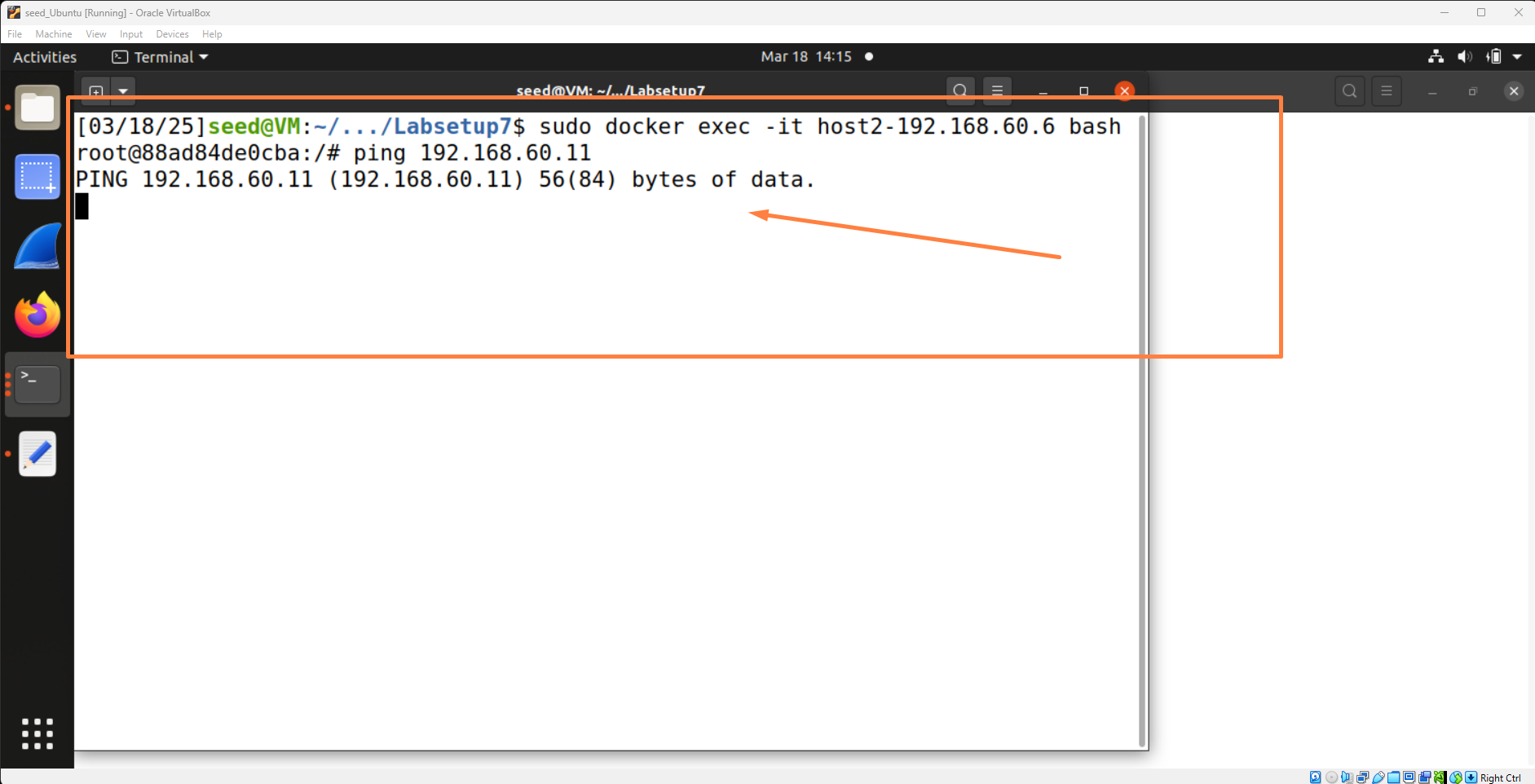
**8.** [**block incoming ping request**] You can not ping uwindsor webserver. Most likely, this is blocked by firewall of uwindsor. Here is the way to block an incoming icmp request.

**$ sudo iptables -A INPUT -p icmp --icmp-type echo-request -j DROP**

Run this on **router** and ping router from another VM. Did you get any reply? Explain.







**9.** Suppose that you want to block all incoming connections while you do not want your visit to external servers to be affected. However, if you send a request to an external server, the server will reply to you while this packet will be blocked by your firewall. To resolve this issue, you should regard the response packet (to your request) as related to your outgoing request packet and allowed to come in. This is achieved using the *conntrack* module.

**$ sudo iptables -P INPUT DROP**

**$ sudo iptables -A INPUT -p tcp -m conntrack --ctstate RELATED, ESTABLISHED -j ACCEPT**

Try this on **router** VM. Then, telnet to a VM (e.g. 192.168.60.7).

Next, telnet from the latter (192.168.60.7) to **router**. Which telnet session gets blocked and which one gets through?

