DNS Infrastructure Lab

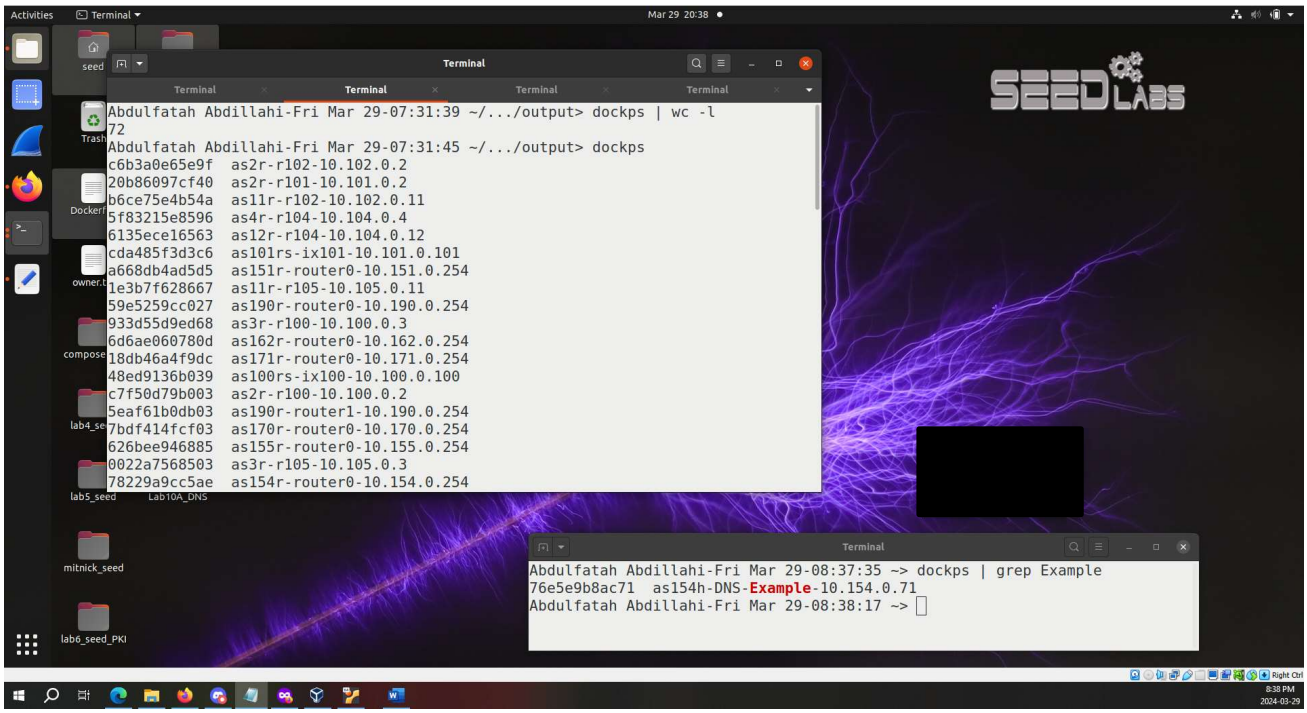
April 2024

Abdulfatah Abdillahi

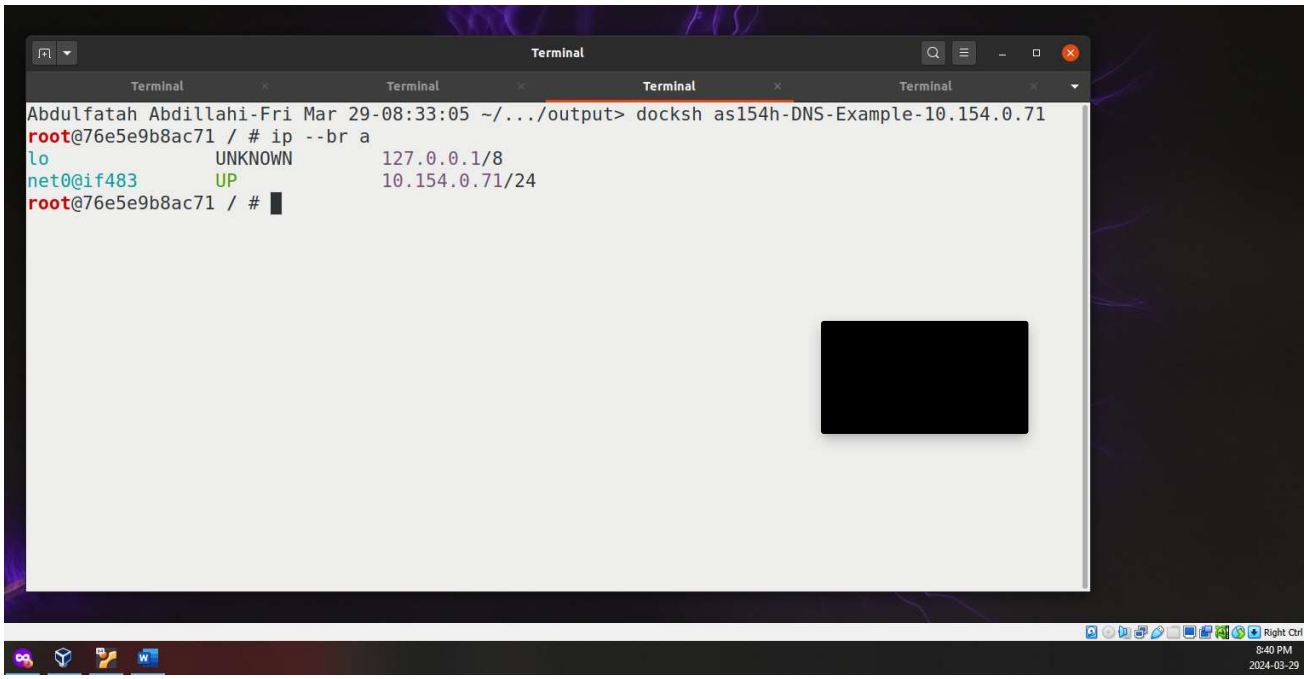
Contents

Setting up the Environment

Creating docker containers



Checking the IP addresses for the container (DNS-Example).

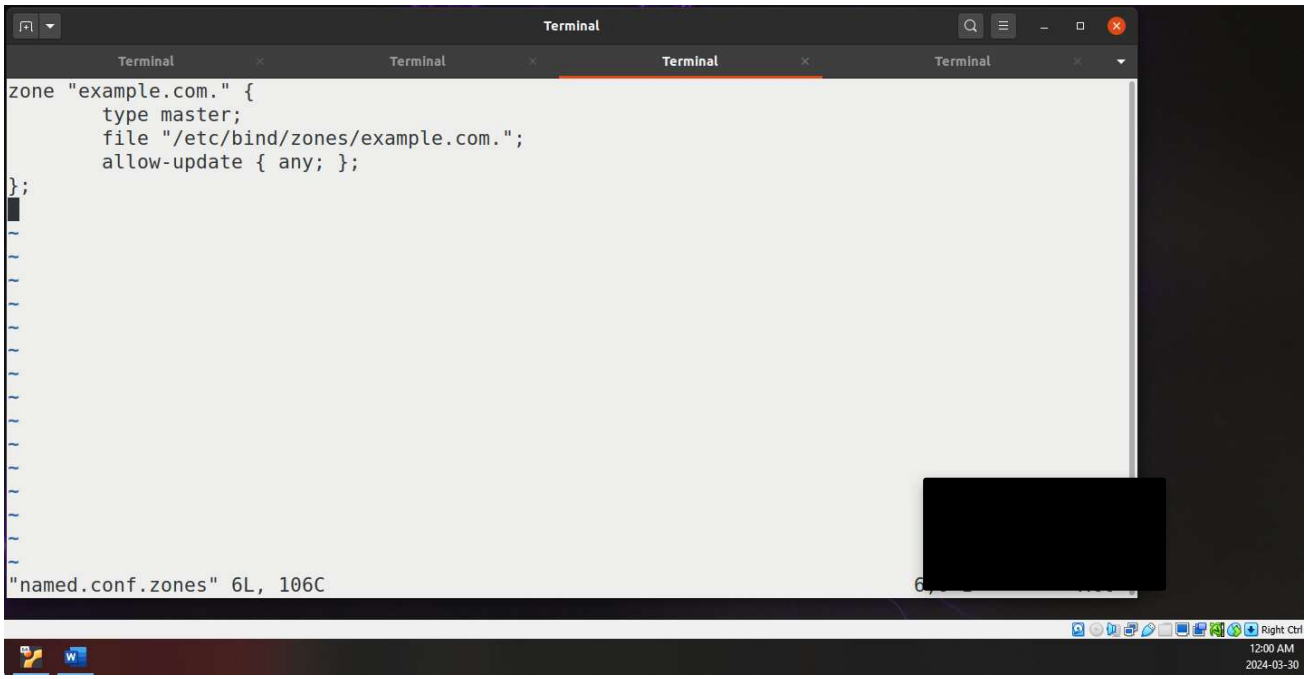


Task 1: Configure the Domain Nameserver

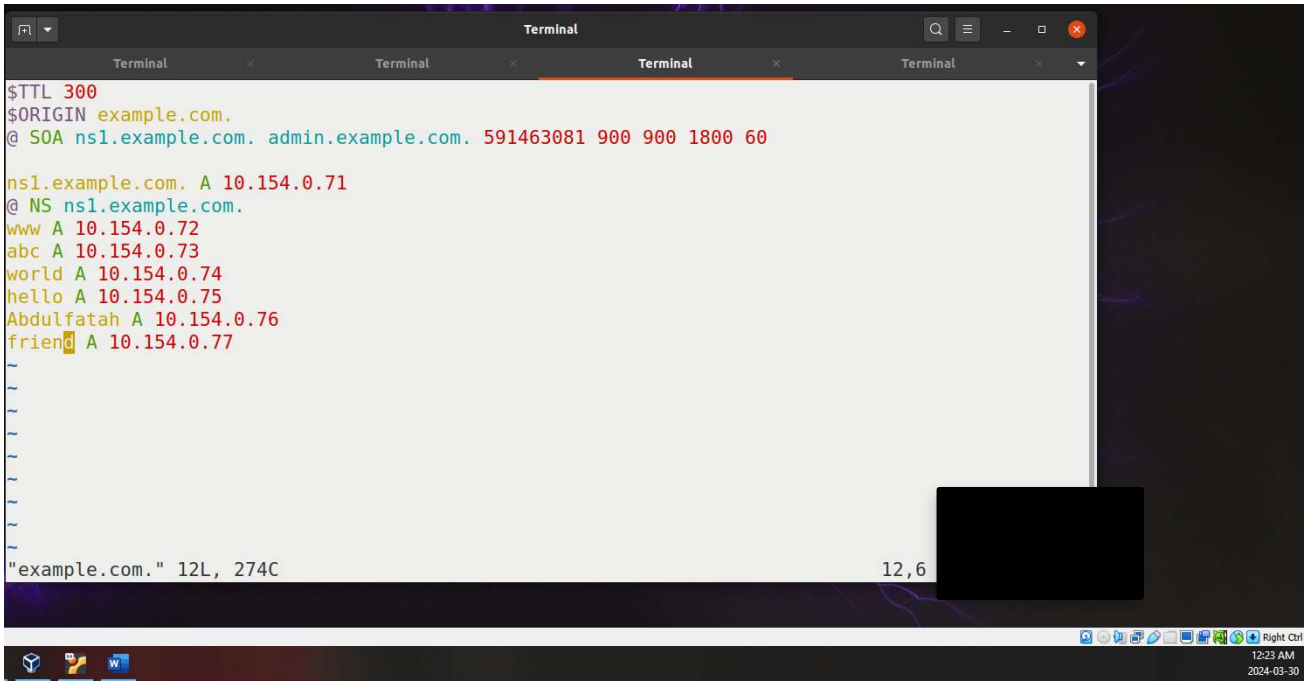
In this task, we will configure the nameservers for two domains, one is example.com, and the other is a customized name based on student's name.

Task 1.a: Configure the Nameserver for example.com

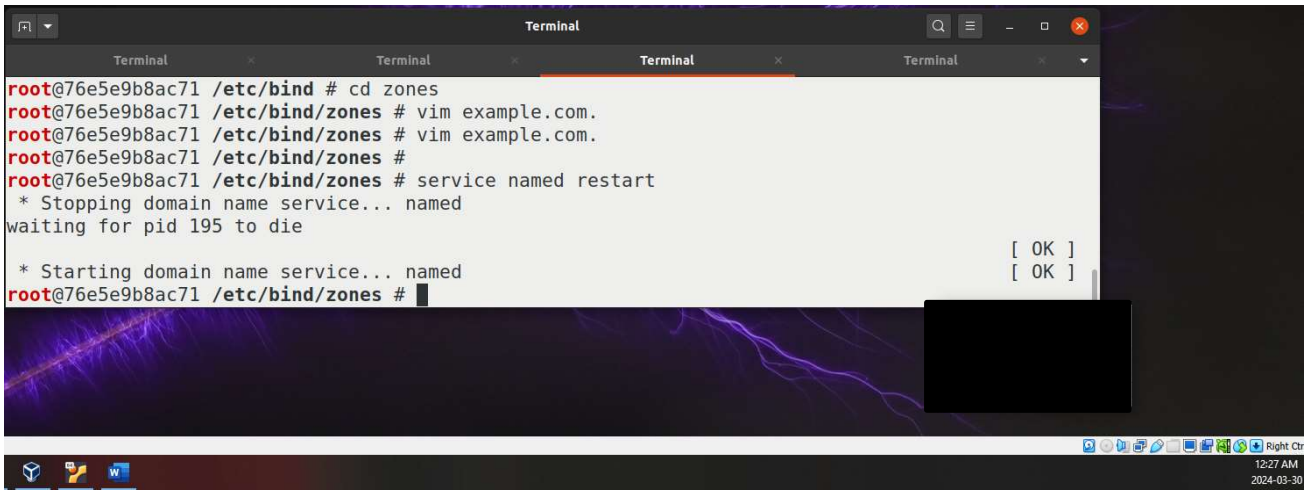
Step 1. Add the zone entry.
Here, we start by going to the nameserver container specified in the instructions (as154h-DNS-Example-10.154.0.71). Then we go to BIND's configuration file (/etc/bind/named.conf.zones) to add a zone entry for the **example.com** domain. This is what the entry looked like:



Step 2. Modify the zone file.
According to the above zone entry, the zone file is located at /etc/bind/zones/example.com. As per the instructions, I added four more record (type A) to the zone file to map hostnames to IP Addresses. This is what the zone file looks like:

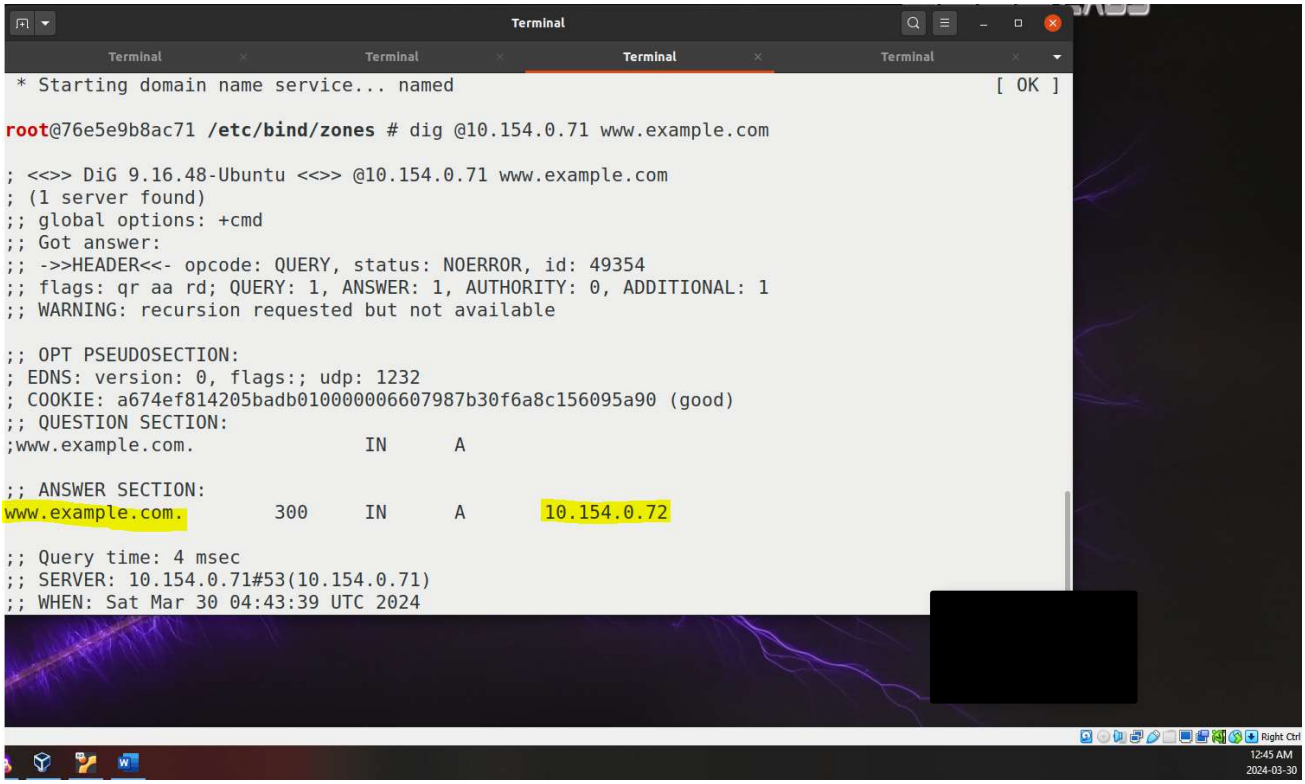


Step 3. Restarting the nameserver.
Then we restart the nameserver so the changes can take effect.



Step 4. Testing.

To test if we did things correctly, we can run *dig www.example.com* and use the IP address of the *example.com* nameserver. We can use the **@10.154.0.71** option to send our DNS query directly to 10.154.0.71. As you can see the answer we got reflects the record we had in the zone file.

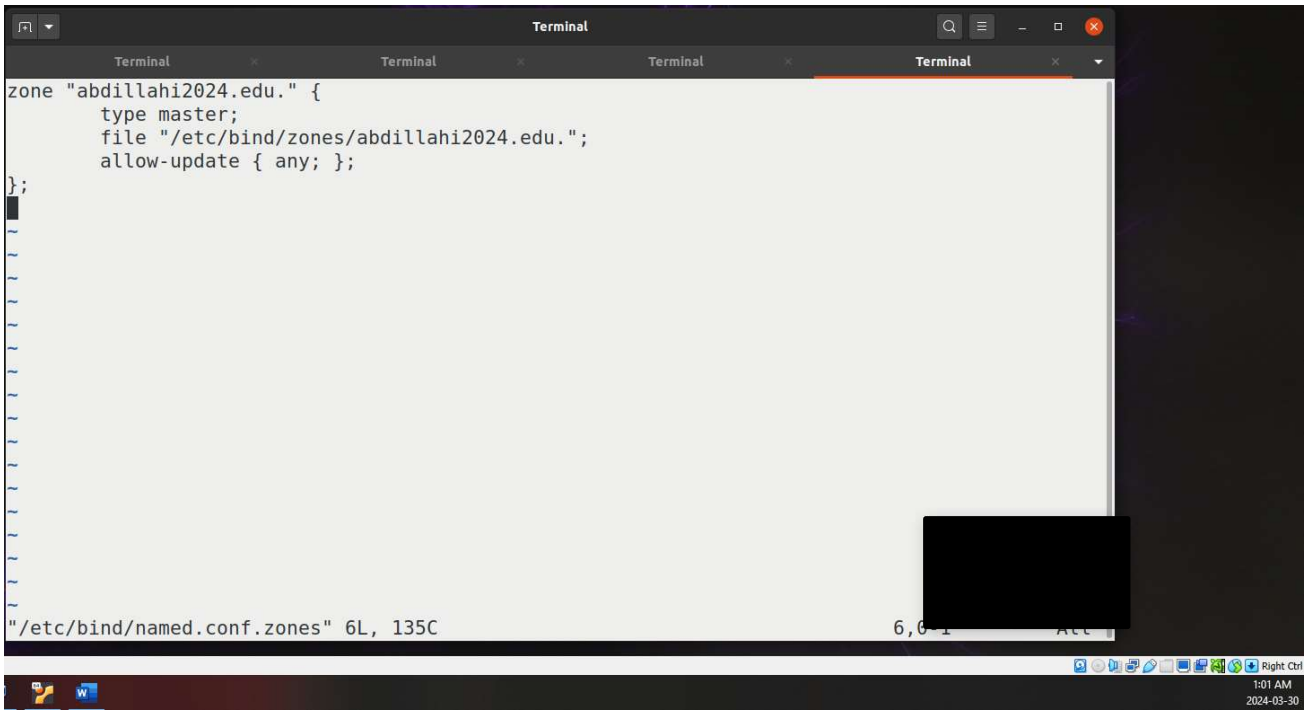


Task 1.b: Configure Nameserver for Another Domain

Now, we will repeat the same thing we did above, but this time we will configure the name server for a custom domain and use an empty nameserver which has been reserved for us.

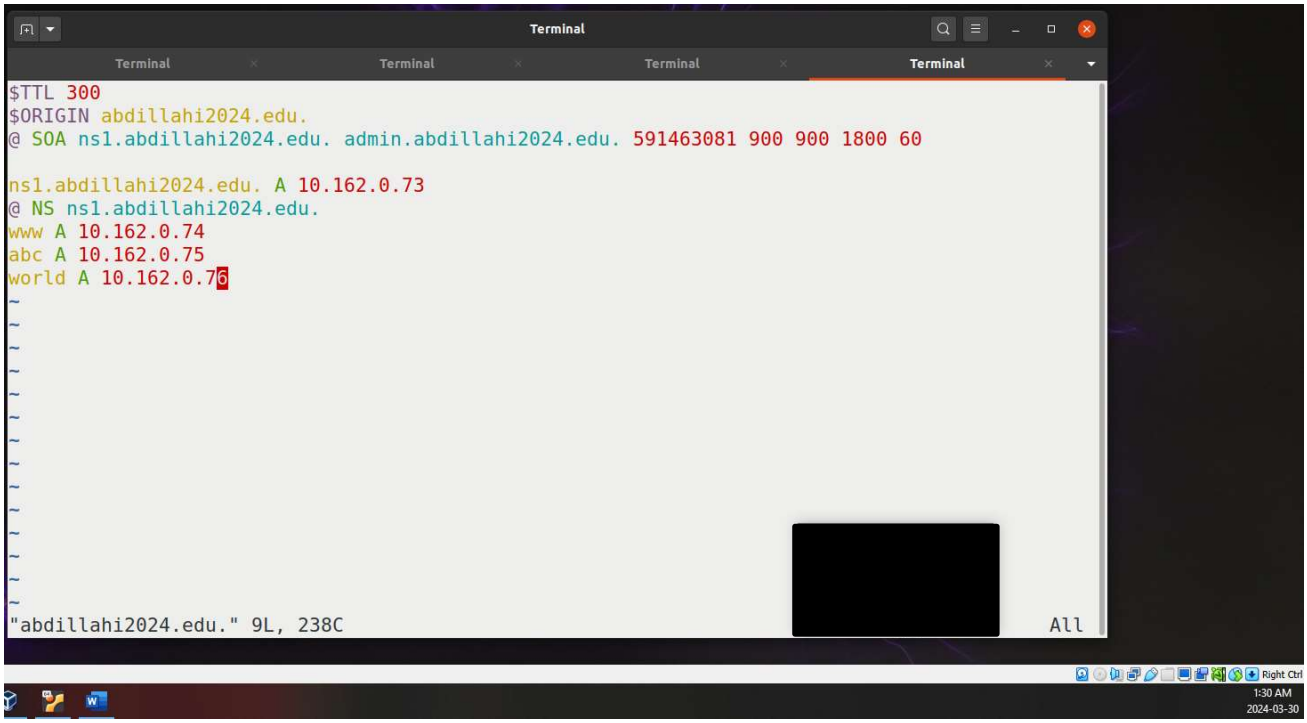
Step 1. Add the zone entry.

Here, we start by going to the nameserver container specified in the instructions (as162h-DNS-AAAAA-10.162.0.72). Then we go to BIND's configuration file (/etc/bind/named.conf.zones) to add a zone entry for the **abdillahi2024.edu** domain. This is what the entry looked like:



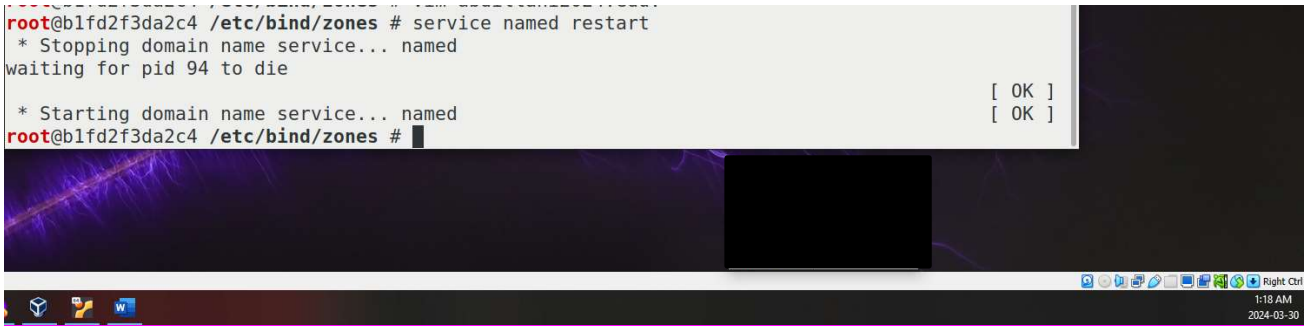
Step 2. Modify the zone file.

According to the above zone entry, the zone file is located at /etc/bind/zones/abdillahi2024.edu. Then I added a few records (type A) to the zone file to map hostnames to IP addresses. This is what the zone file looks like:



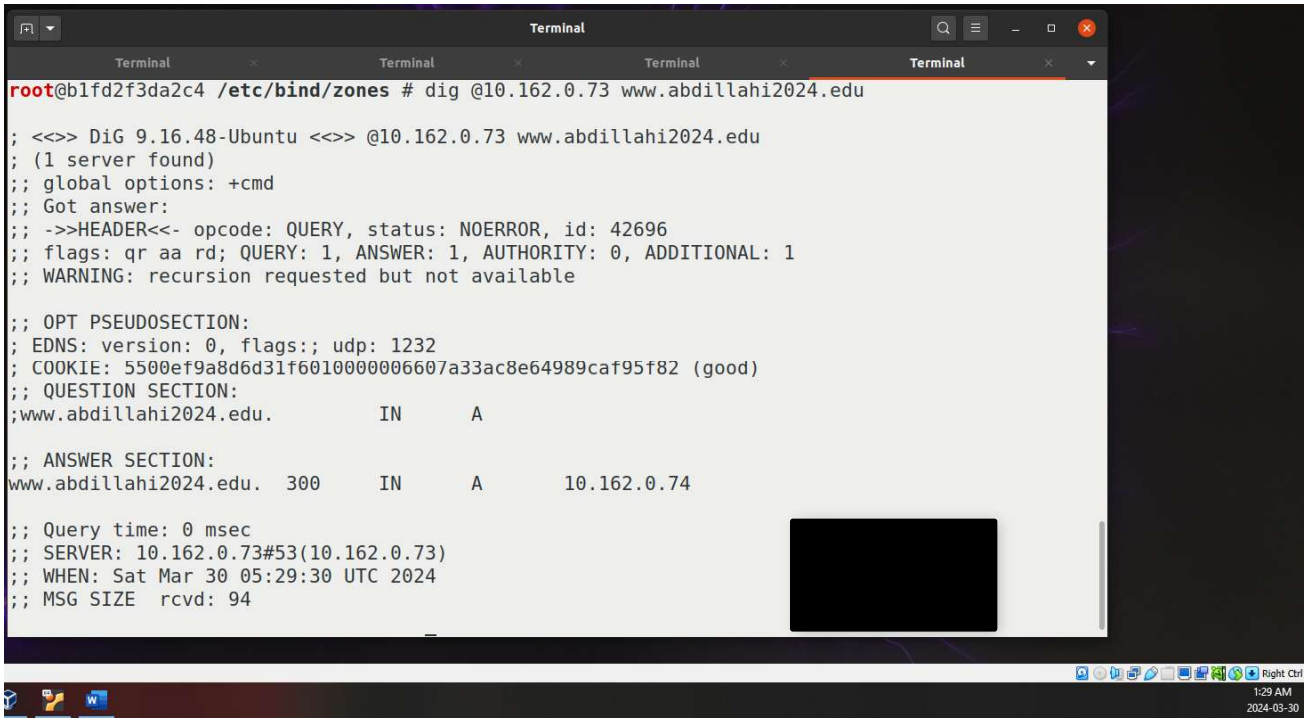
Step 3. Restarting the nameserver.

Then we restart the nameserver so the changes can take effect.



Step 4. Testing.

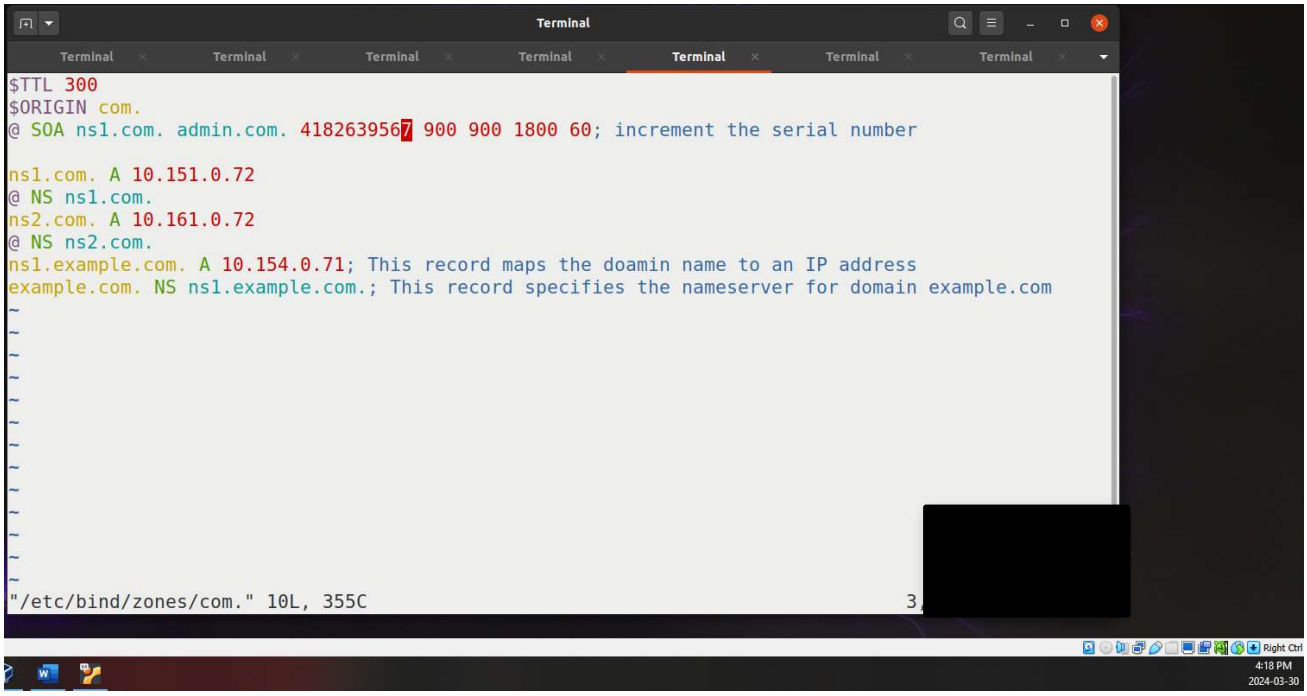
To test if we did things correctly, we can run *dig www.abdillahi2024.edu* and use the IP address of the current nameserver. We can use the **@ 10.162.0.72** option to send our DNS query directly to 10.154.0.71 (current server). As you can see the answer we got reflects the record we had in the zone file.



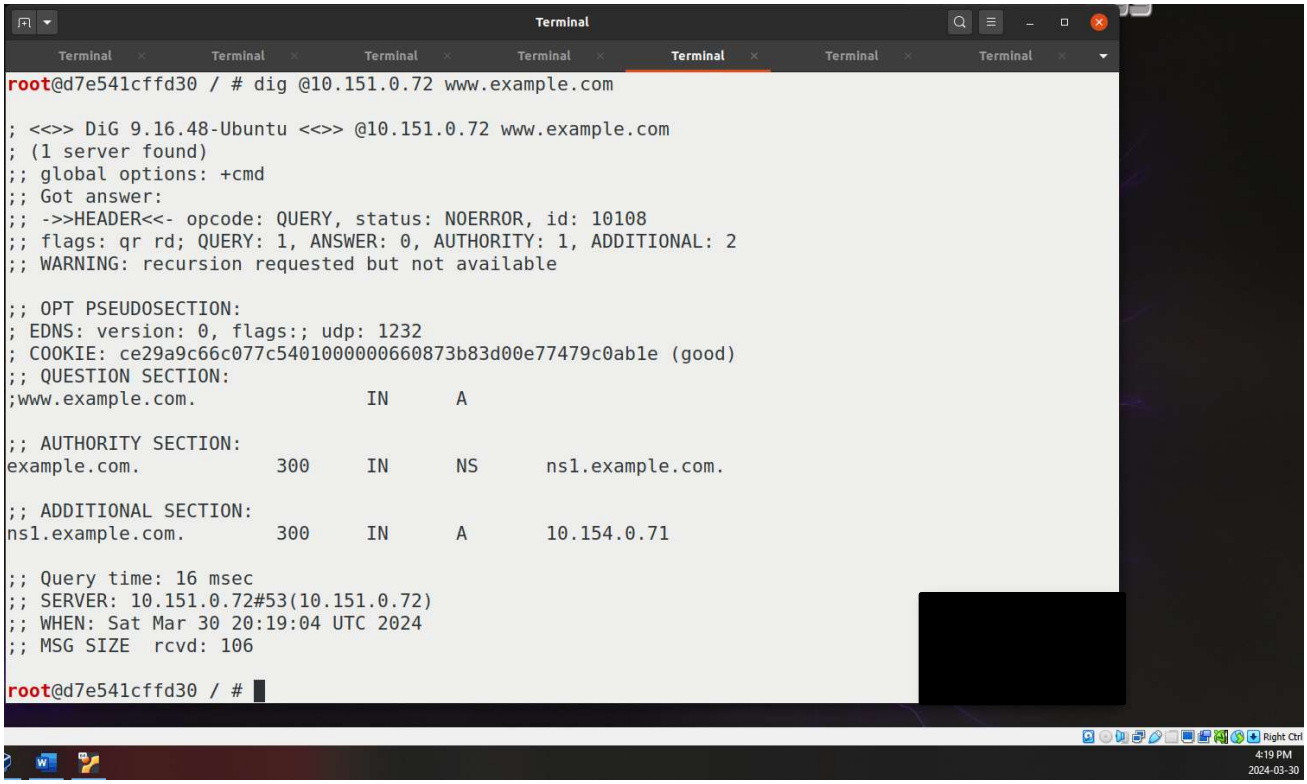
Task 2: Configure the TLD servers

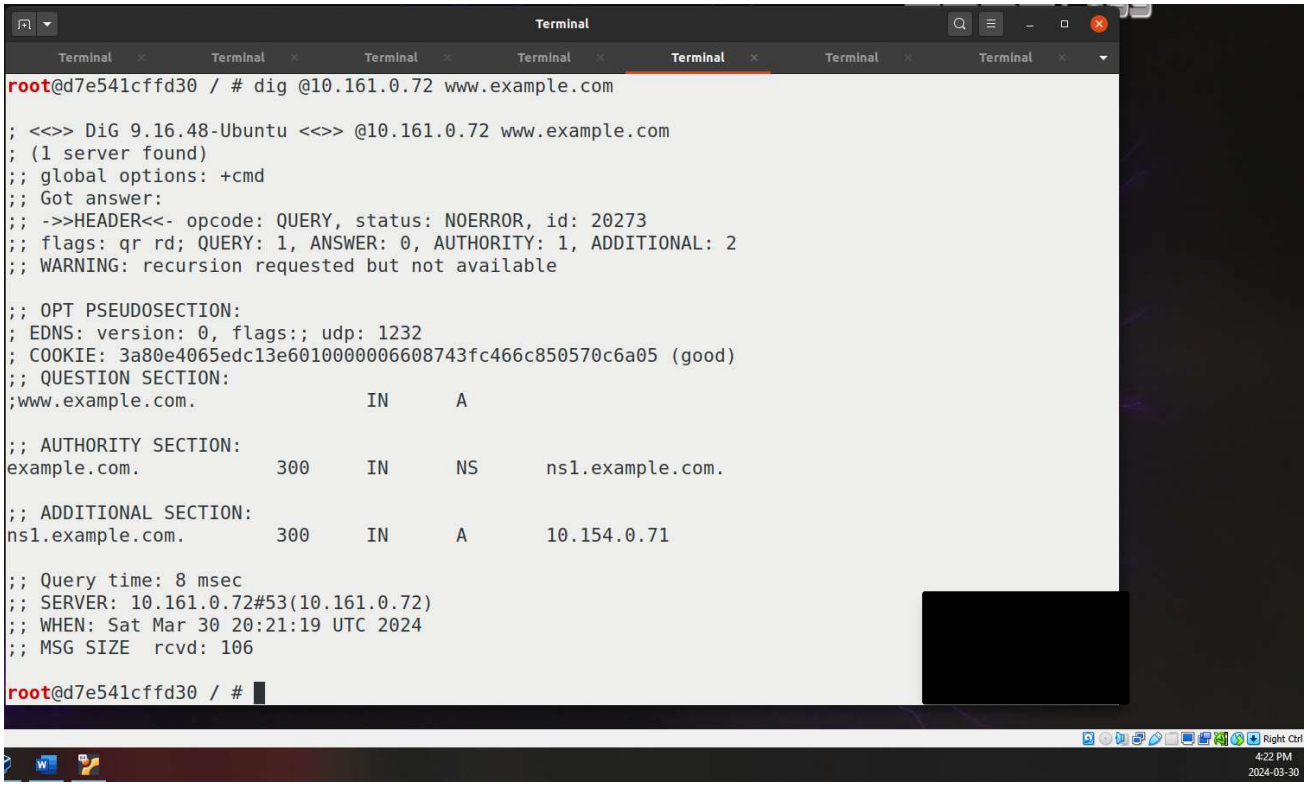
In this task, we will configure the nameservers for two TLD domains: *com* and *edu*. We have three nameservers reserved for these two TLD domains (as151h-DNS-**COM-A**-10.151.0.72, as161h-DNS-**COM-B**-10.161.0.72, and as152h-DNS-**EDU**-10.152.0.71).
There are two nameservers for the com zone. One is configured as the master server (as151h-DNS-**COM-A**-10.151.0.72), and the other as the slave server (as161h-DNS-**COM-B**-10.161.0.72). We only need to modify the zone file on the master (COM-A), as the slave server will automatically synchronize with the master server.
All the nameservers within a TLD domain must register their nameservers with this TLD server; otherwise, nobody can find them. For each domain, such as *example.com*, we need to add two records in the *com* server's zone file: an *NS* record and an *A* record. The *NS* record specifies the nameserver for the *example.com* domain, while the *A* record specifies the IP address of the nameserver.

Here, I browse to the zone file (/etc/bind/zones/com.) on the master server for ".com" and register the name server for "example.com".

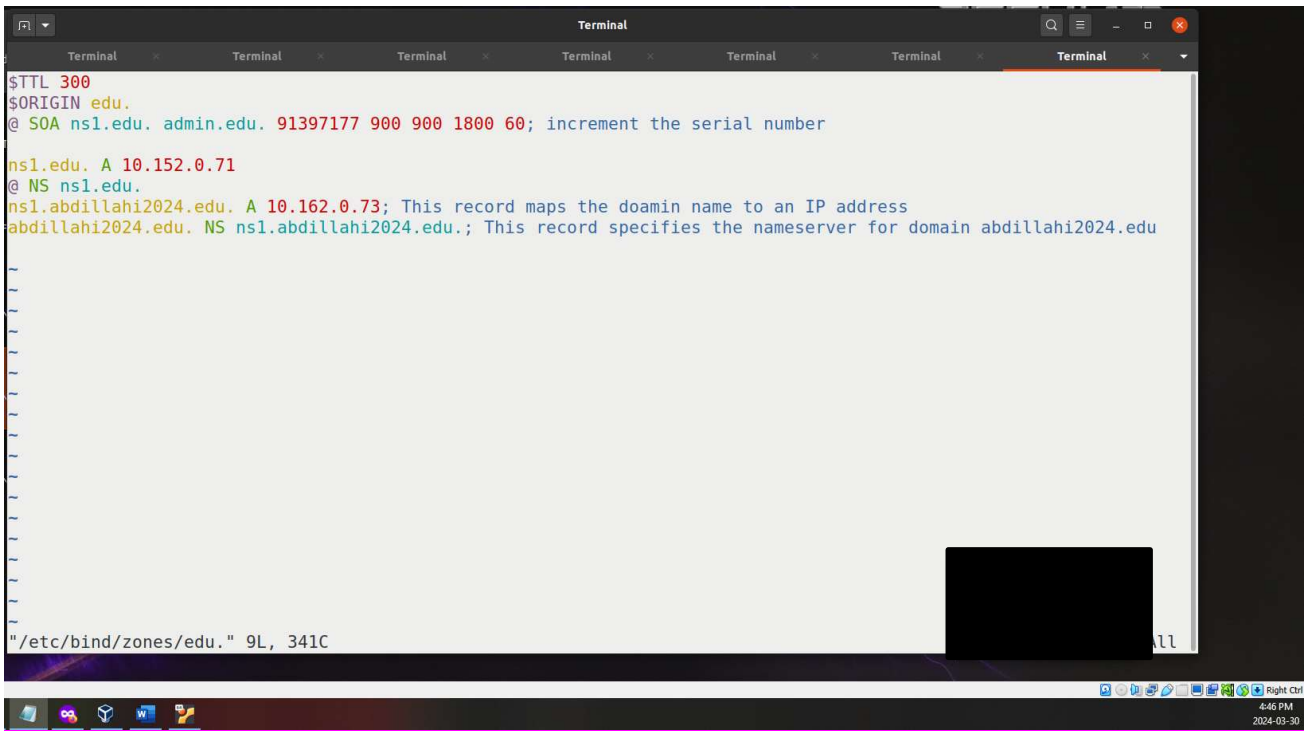


Then I restarted the server with *service named restart* and tested to see if the response will return the nameserver for the domain name in the query. To do this I queried the master (10.151.0.72) and slave (10.161.0.72) servers individually. As you can see this was successful.

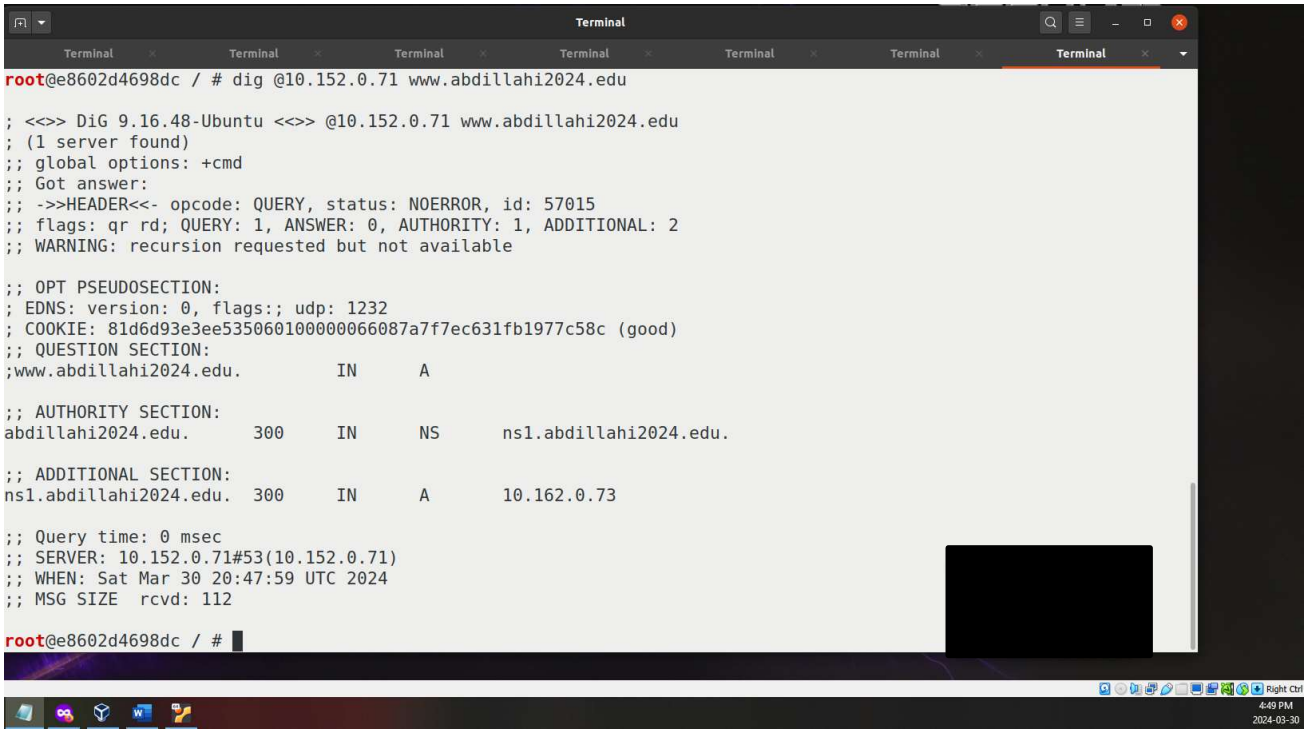




Then I must a similar thing and register my custom named nameserver (abdillahi2024.edu) with the EDU server. Once again, I browse to the zone file (/etc/bind/zones/edu.) on the master server for ".edu" and register the name server for "abdillahi2024.edu".



Then I restarted the server with *service named restart* and tested to see if the response will return the nameserver for the domain name in the query. As you can see this was successful.

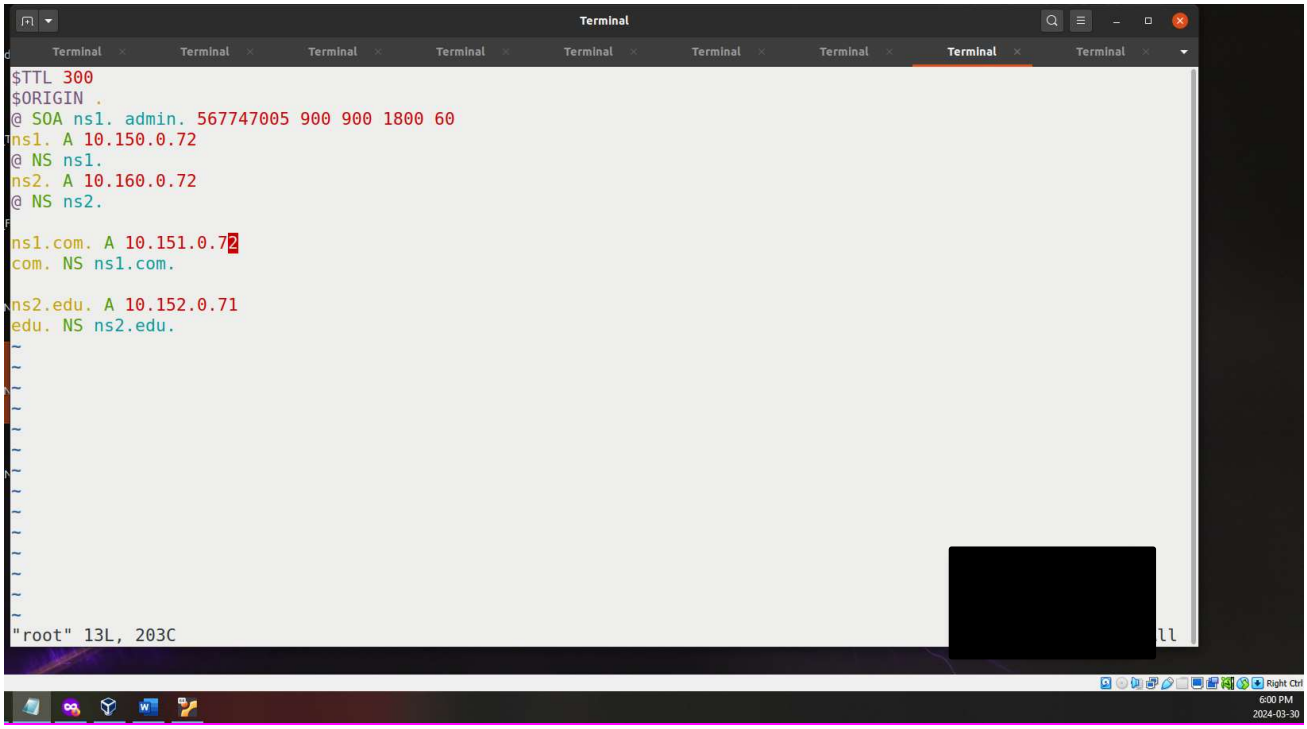


Task 3: Configure the Root servers

In this task, we will configure the nameservers for the root zone. In the real world, there are 13 nameservers for the root zone, and they are synchronized through the root zone file maintained by IANA. In this environment, we have only two root servers (as150h-DNS-**Root-A**-10.150.0.72 and as160h-DNS-**Root-B**-10.160.0.72). We will manually synchronize them by putting the identical content in their zone files.

All TLD nameservers need to register with the root nameserver, so they can be found in the DNS query process. For every TLD zone that we would like to include in our miniature DNS system, we need to add at least two records in the zone file, including an *NS* record and an *A* record. In this task, we need to modify both root server's zone files to support the *com* and *edu* TLDs inside the environment. The zone file is located inside the */etc/bind/zones* folder.

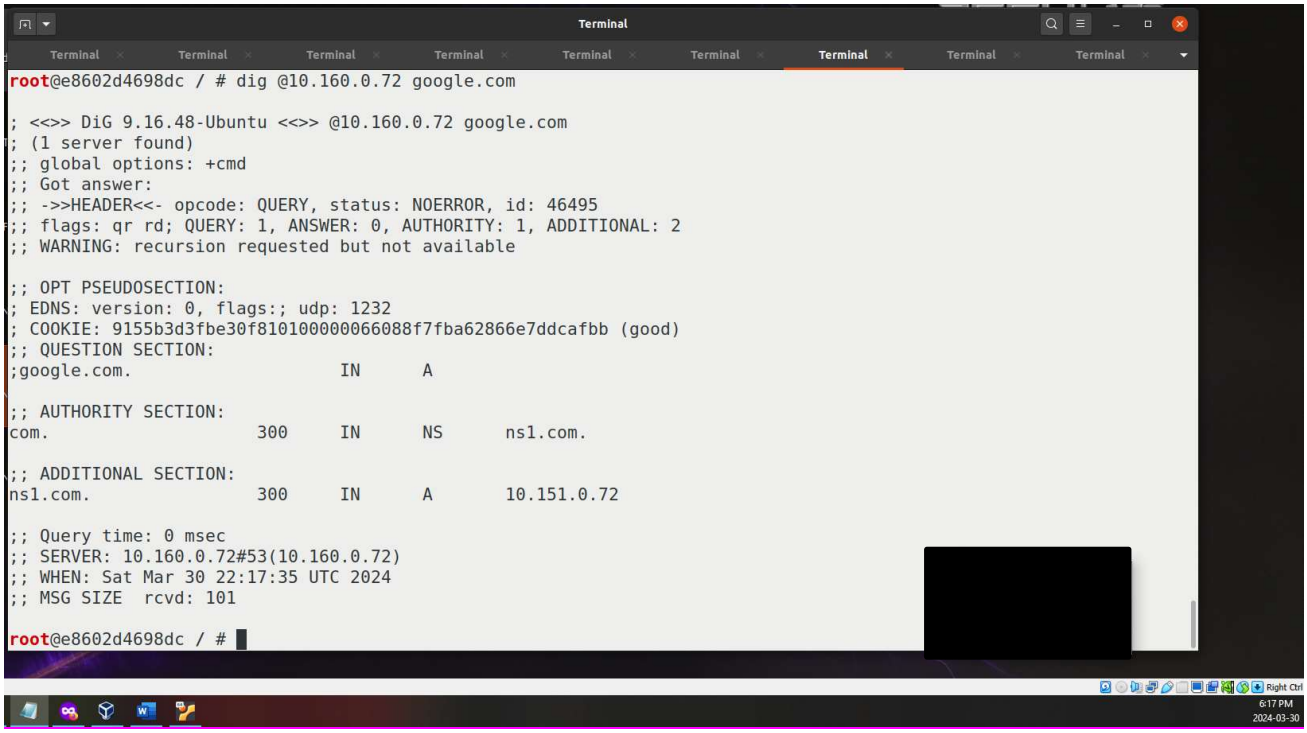
This is what the zone file looks like after including the *NS* record and *A* record of *com* and *edu* TLDs. These records were also be duplicated in the zone file of the other root server.

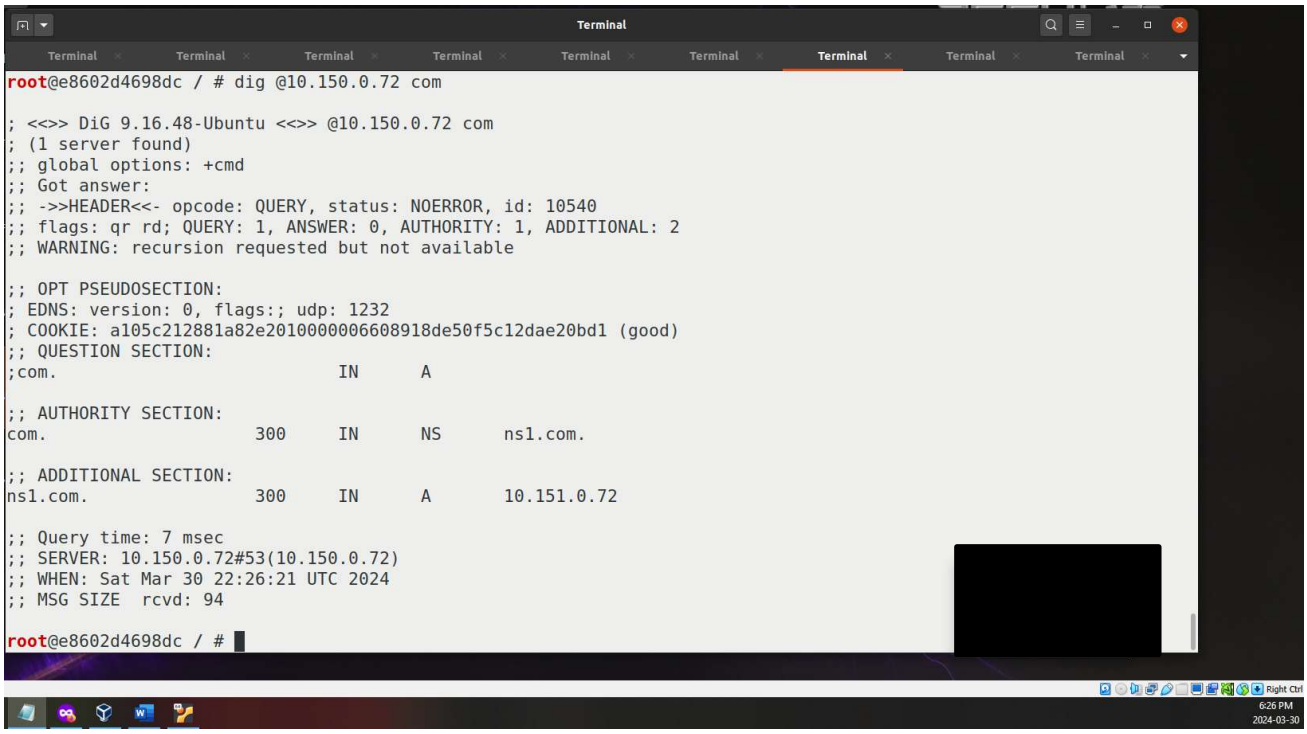


Then I restarted the server with *service named restart* and ran the specified queries (for <ANYNAME>.com, <ANYNAME>.edu, com, edu) on a different container than the root container to test the output of the server.

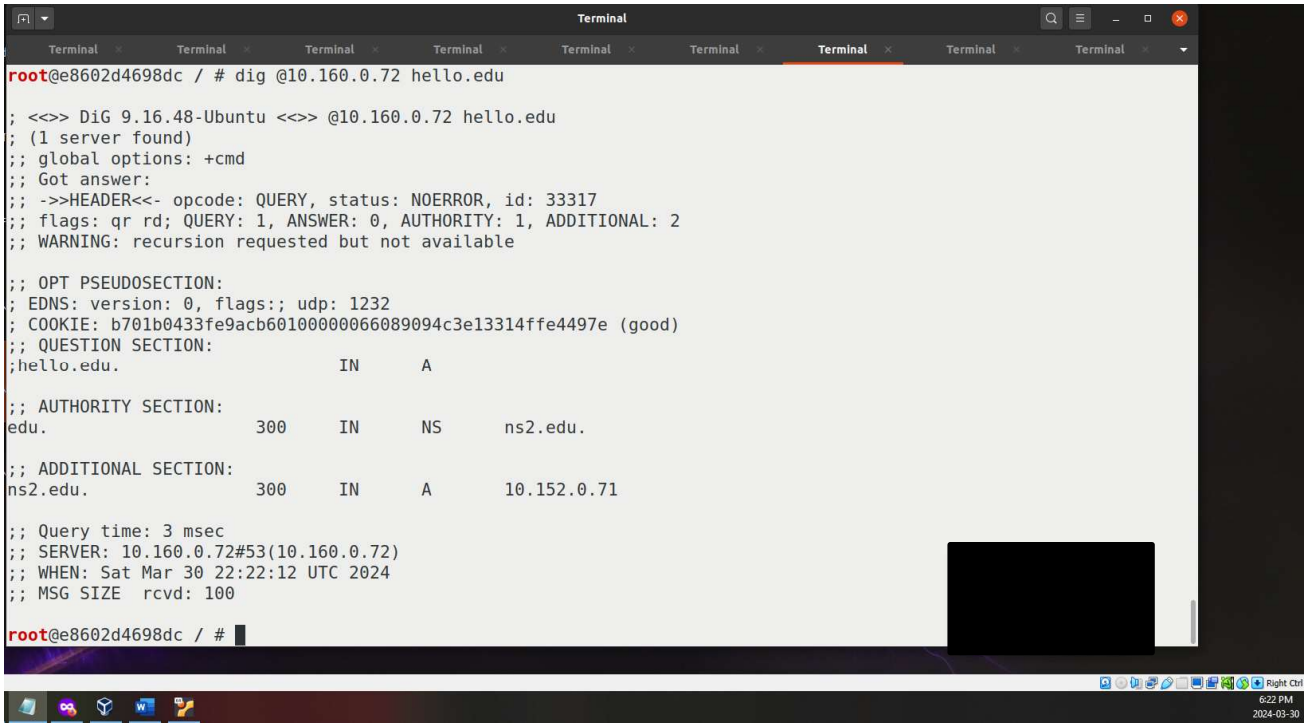
As you can see in the output, the query was successful (status: NOERROR). The *AUTHORITY SECTION* correctly shows that the *com* domain is served by *ns1.com*, and the *ADDITIONAL SECTION* provides the IP address (*10.151.0.72*) for *ns1.com*. This is expected because the root server isn't responsible for knowing all the records for all the domains. Its job is to direct queries to the appropriate nameserver (*ns1.com* in this case) which hypothetically should have the specific records for its domains (like google.com). The same output was found when sending the query to the

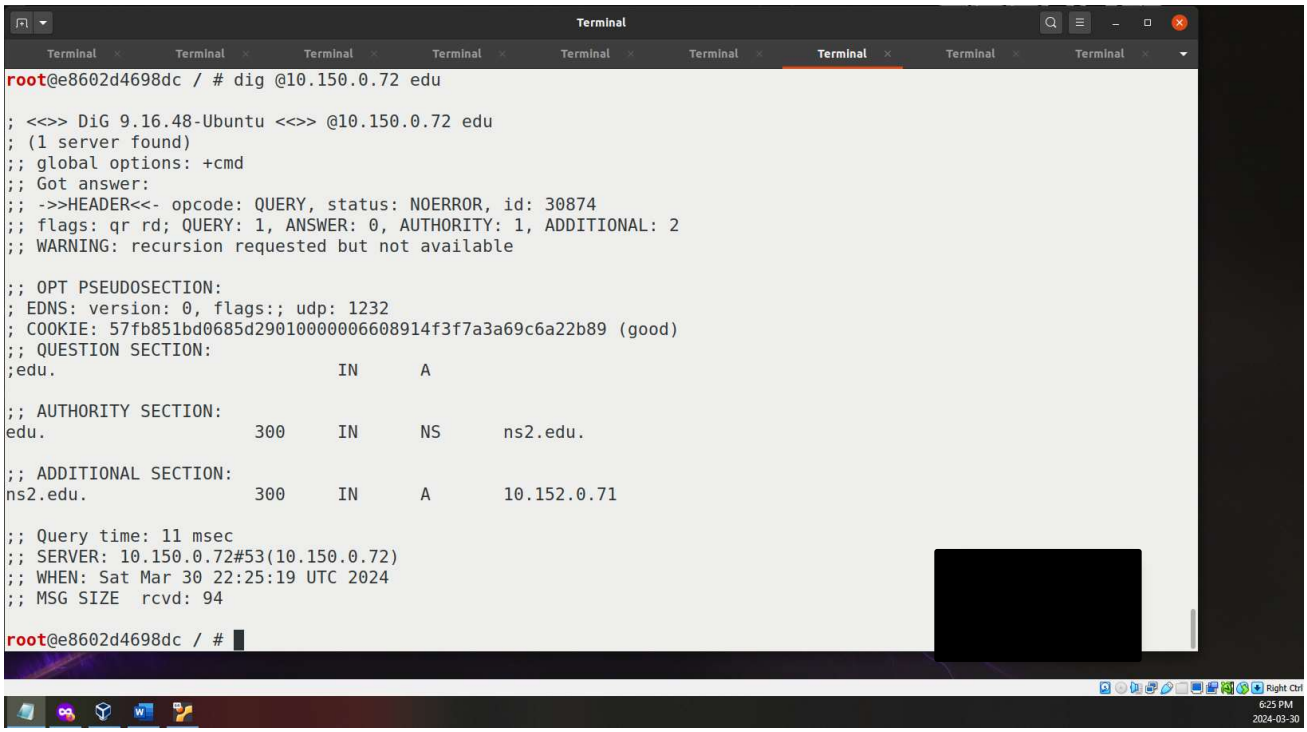
other root server as seen in the below screenshot.





As per the instructions, I will also test for the edu TLD server. As expected, the output here is also directing you to the appropriate nameserver (*ns2.com* in this case).

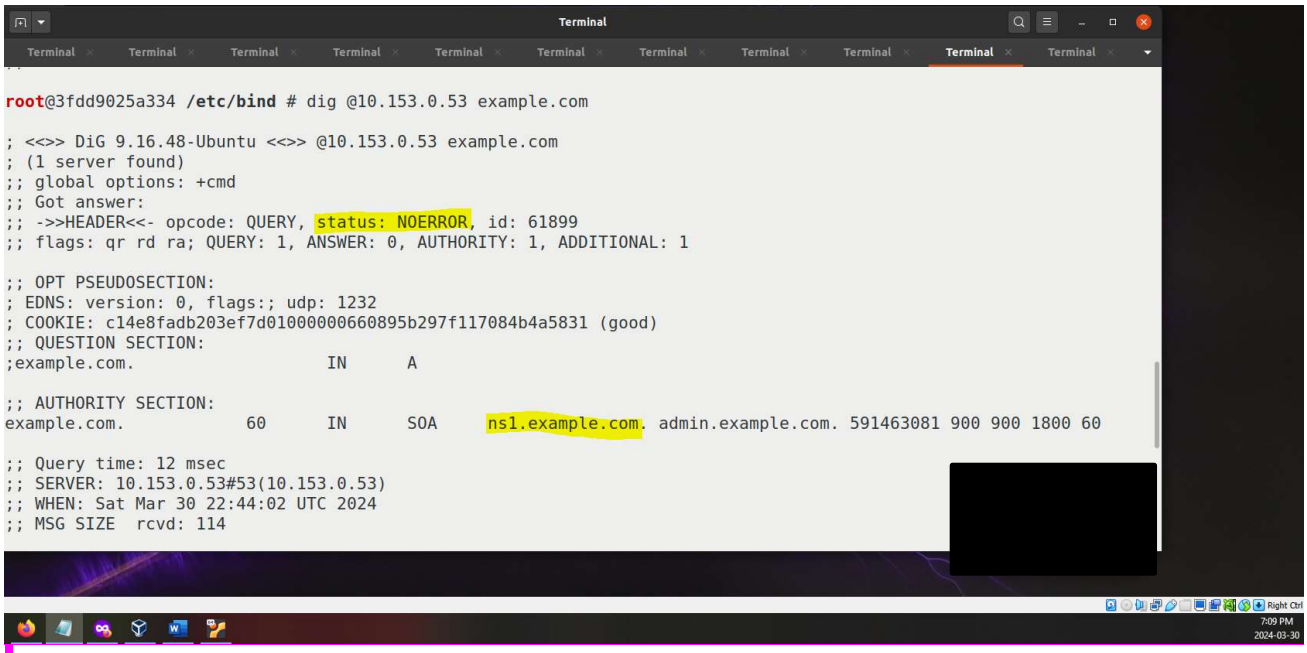


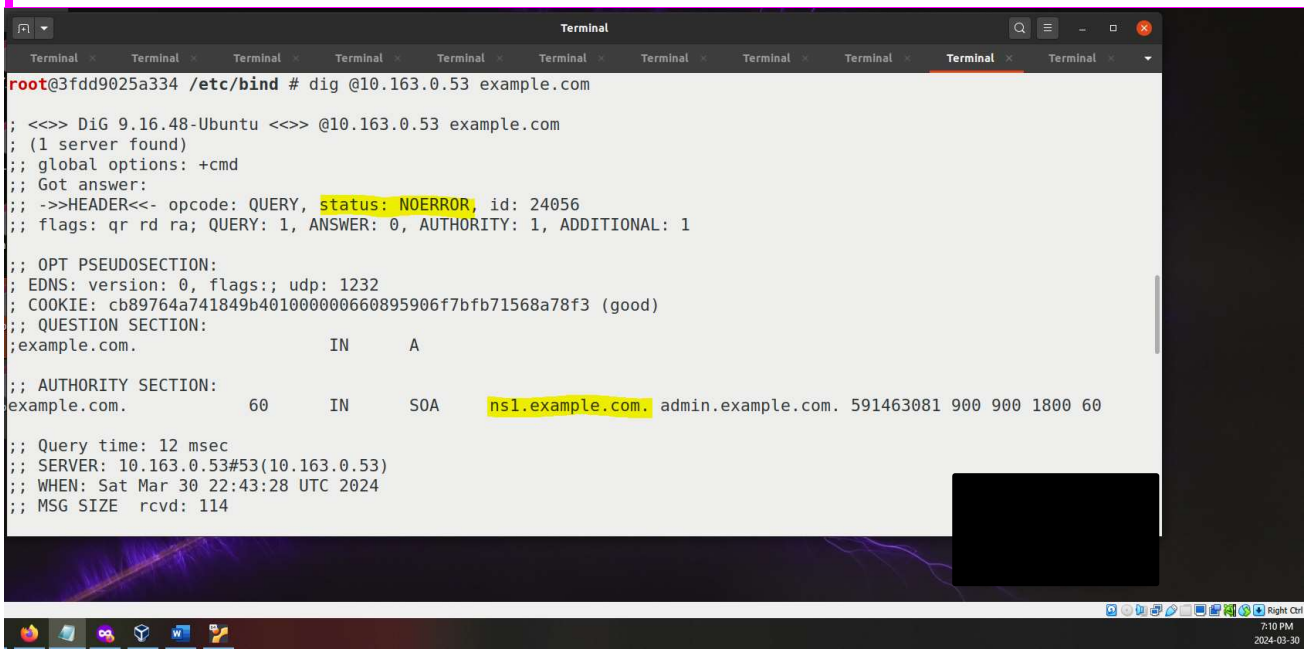


Task 4: Configure the Local DNS Server

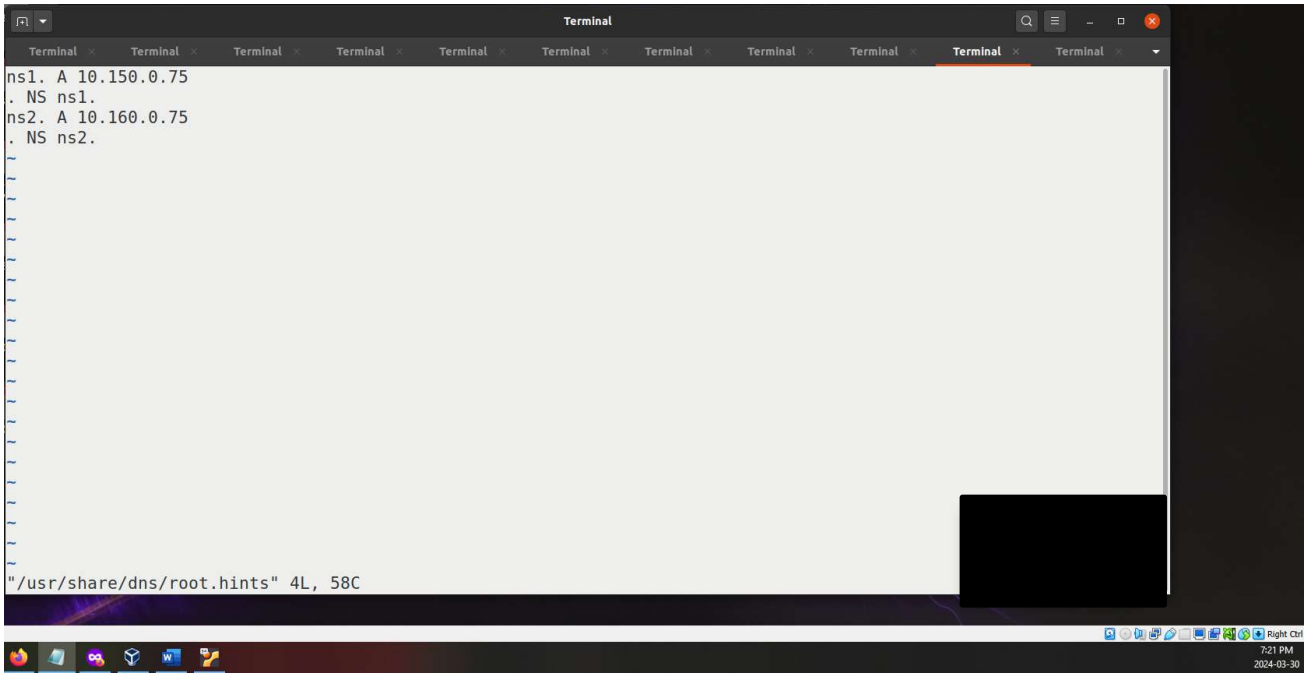
In this task, we are tasked with configuring the local DNS server by modifying its root hints file (/usr/share/dns/root.hints), specifically changing the IP addresses of the root name servers. After making the changes, we're asked to observe how this modification affects the DNS resolution process by running specific commands. The commands are **dig @10.153.0.53 example.com** and **dig @10.163.0.53 example.com**. We have two DNS resolvers in the environment (as153h-**Global_DNS-1**-10.153.0.53, and as163h-**Global_DNS-2**-10.163.0.53) which we will use this task for.

Here, I will start by taking a screenshot of what the output looks like before making any modifications.

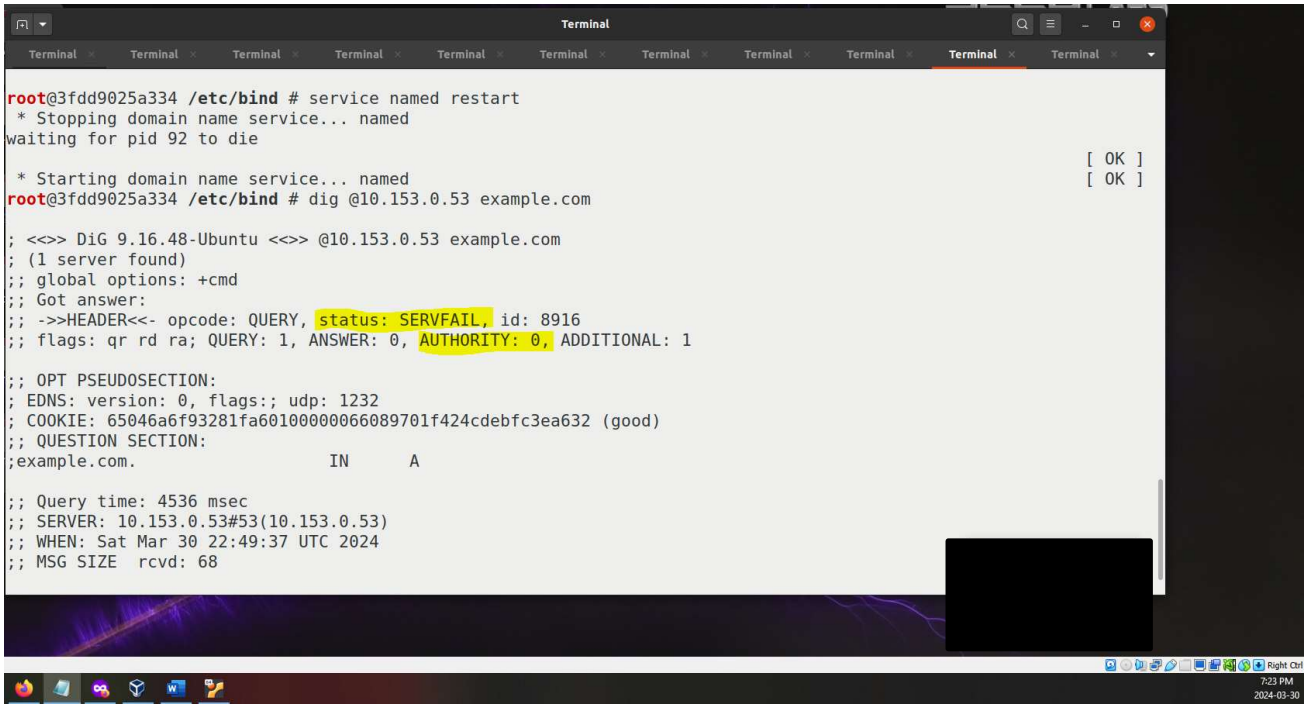


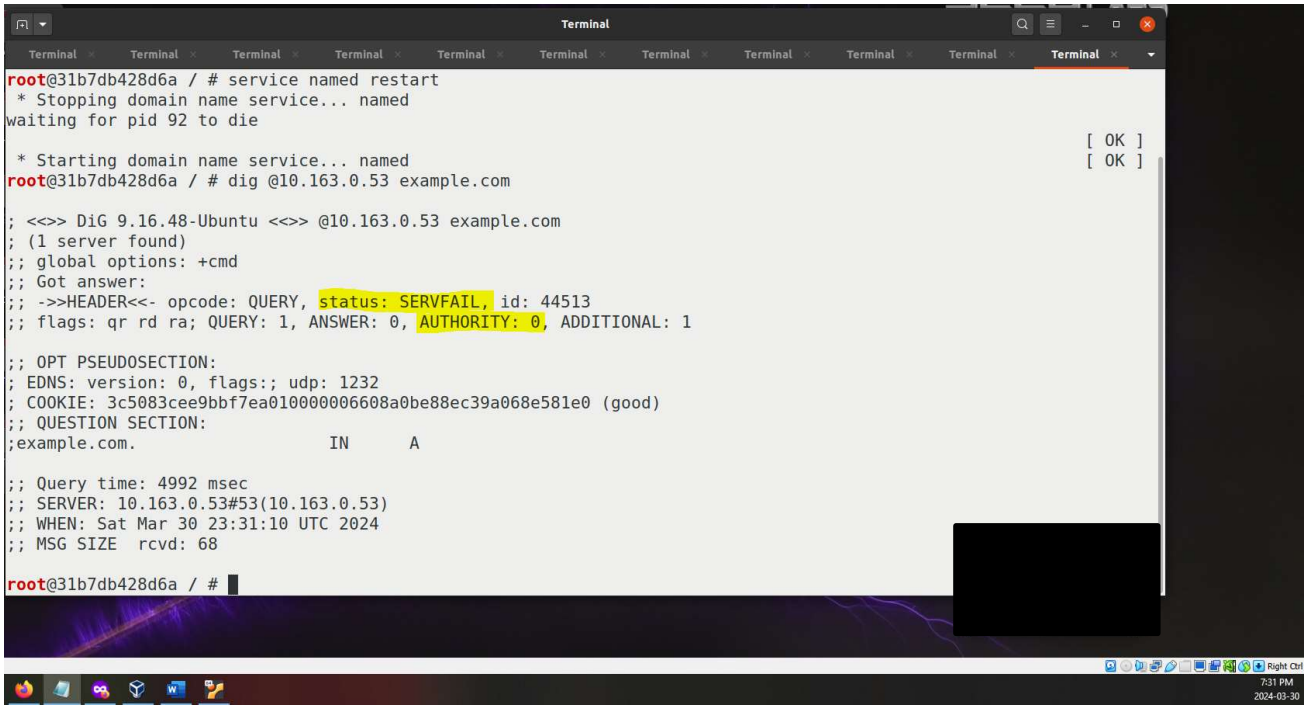


As you can see in the output, both local servers show that the query was successfully resolved (*status: NOERROR*), the authority section provided information about the SOA (Start of Authority) record for *example.com*, and the query time was relatively fast (*12 msec*). Now let's modify the root hints file on both local servers to see how this affects the DNS resolution process. Here, you can see that I changed the IP address of the root servers (from .72 to .75) in the root hint file of both our local DNS resolvers.



Then I restarted the service (*service named restart*) and ran the same query once again.





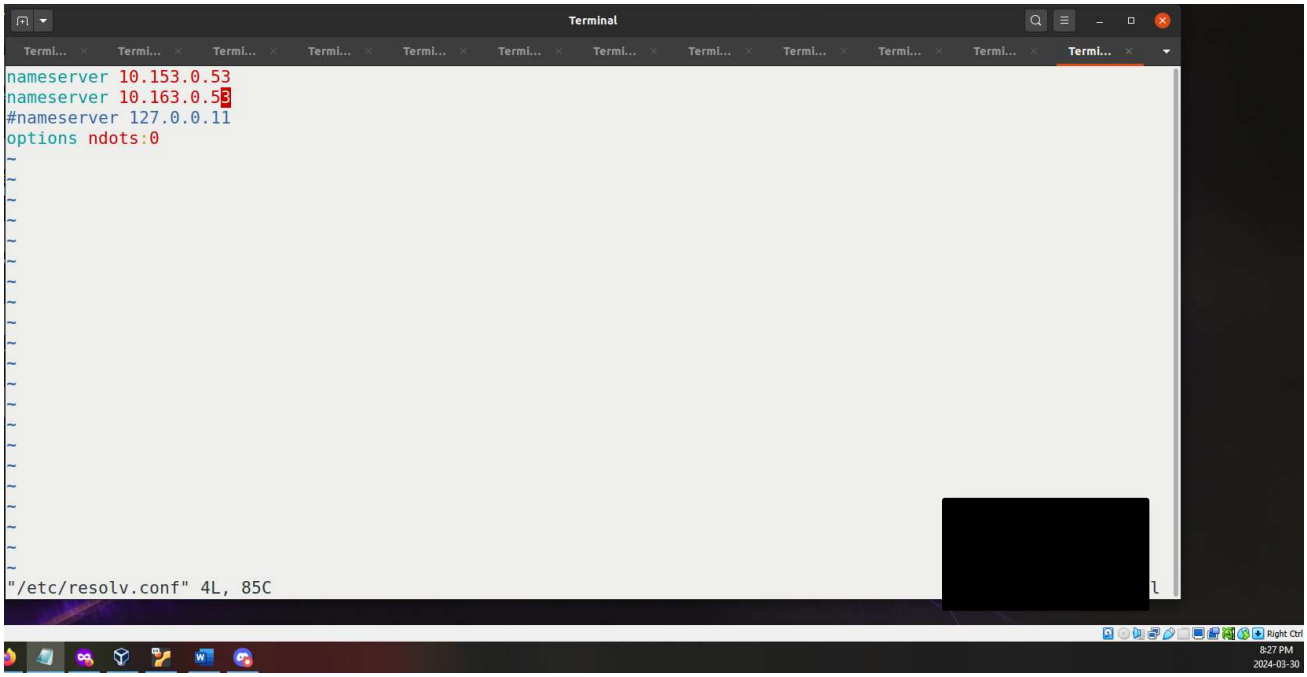
This time as you can see in the output, the DNS query to resolve *example.com* resulted in a *SERVFAIL* status, indicating a server failure, there are no authoritative or additional sections in the response, and query time significantly increased (*4536 msec* and *4992* respectively).

Before moving on, I changed them back because the subsequent tasks depend on them.

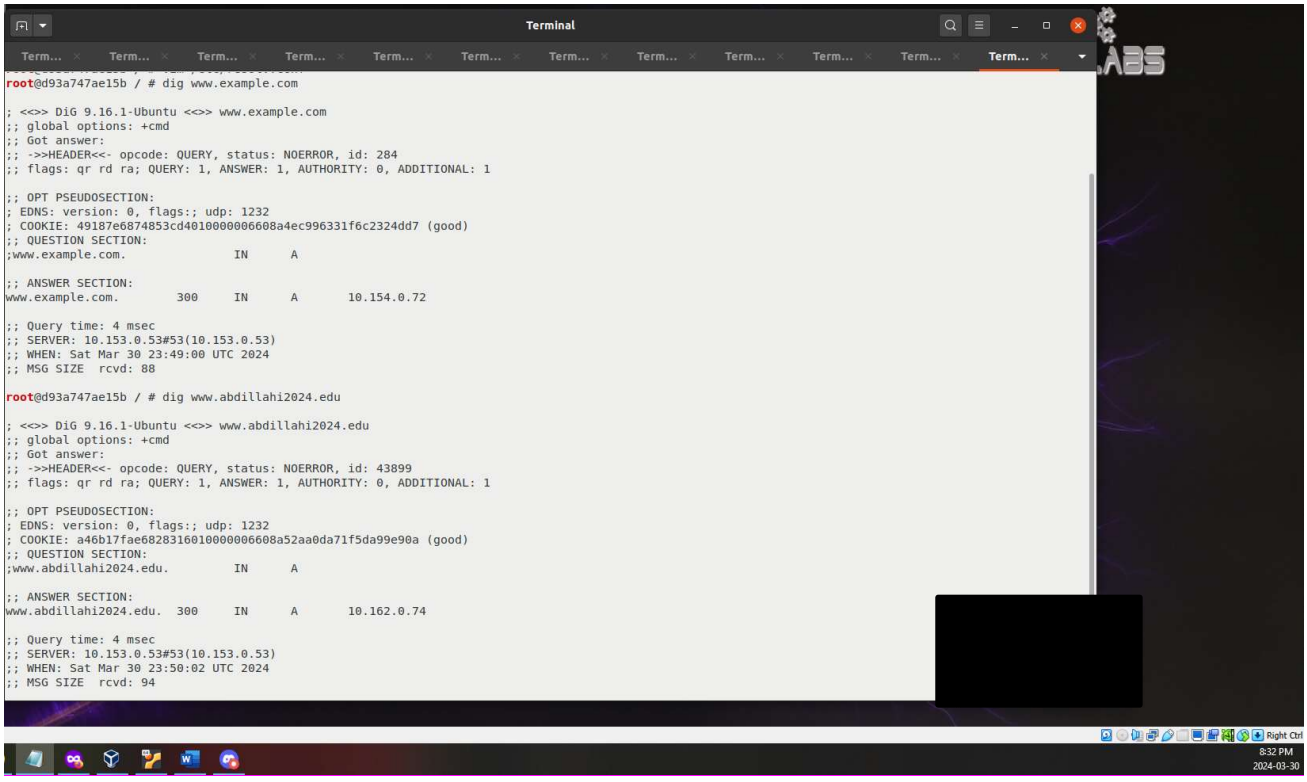
Task 5: Configure the Client

So far, we had to use @<*ip*> in our dig command to indicate what DNS server the *dig* command should talk to. While this is not an issue for *dig*, it is a problem for other software that depends on DNS. We need to tell the operating system what local DNS server it should use. This is achieved by changing the resolver configuration file *(/etc/resolv.conf*) of the user machine, so the container's IP address is added as the first nameserver entry in the file, i.e., this server will be used as the primary DNS resolver.
In this task, our goal is to configure the host machines (as155h-**host**_0-10.155.0.71) to use one or both of the DNS resolvers (10.153.0.53 and/or 10.163.0.53) by modifying the resolver configuration file *(/etc/resolv.conf*). We need to add the IP addresses of the DNS resolvers as nameserver entries in the **/etc/resolv.conf** file on these machines.
After configuring the host machines, we're asked to test the entire DNS infrastructure using the dig command without specifying the DNS server explicitly (without using the @<*ip*> option). If the DNS infrastructure is set up correctly, we should get the answer as expected.

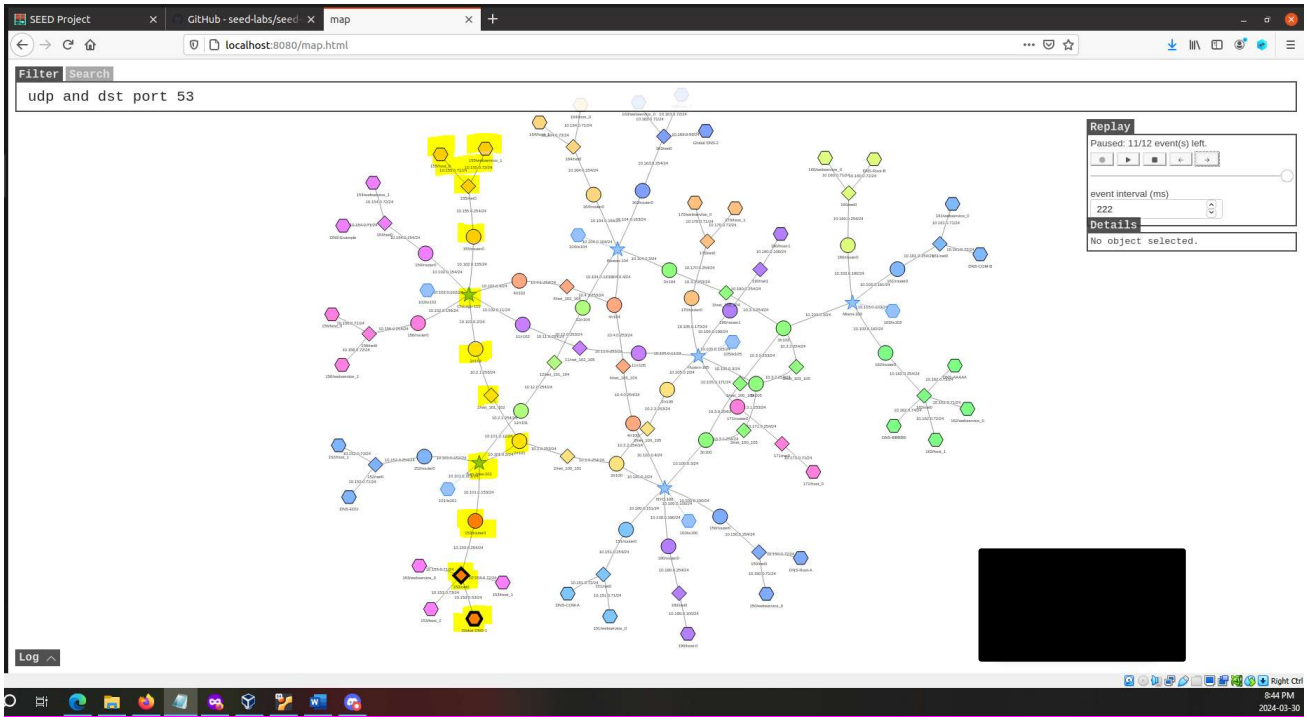
As instructed, I have modified the resolver configuration file to use both DNS resolvers.



Then I ran the dig command without specifying a server and this was successful.



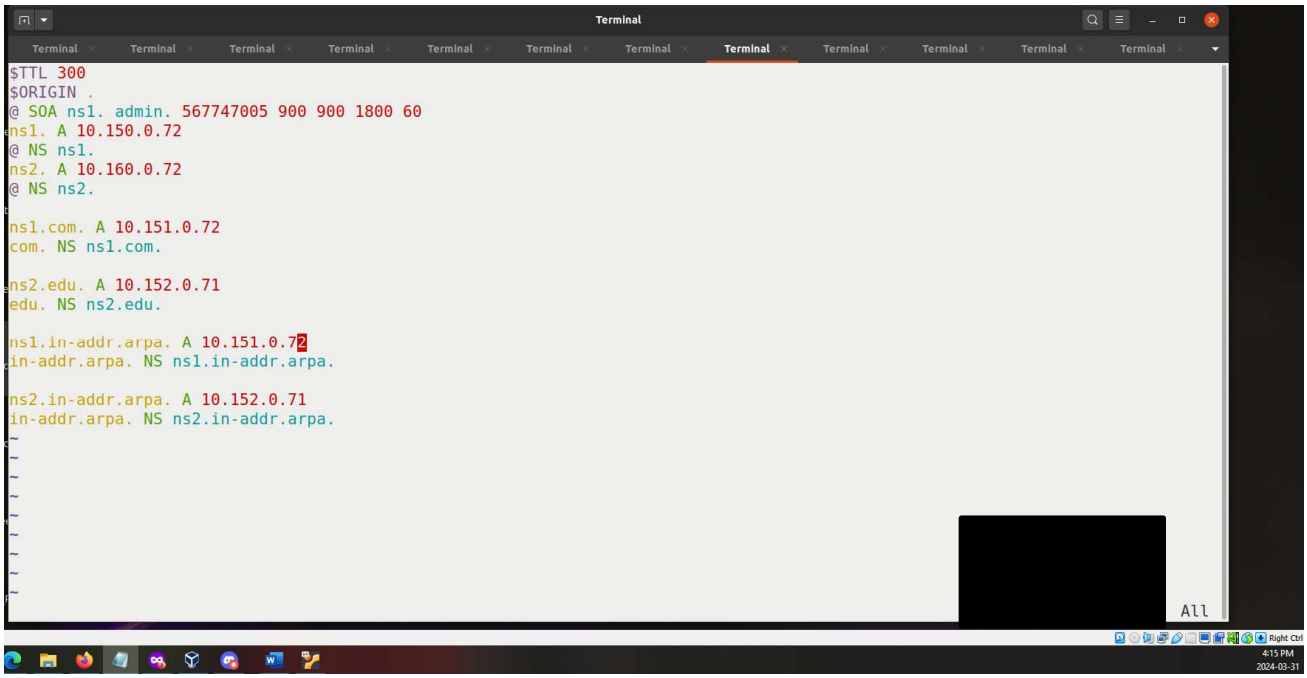
I have also used the map to trace the packet when I ran the command. The highlighted portion of the screenshot reflects the path taken by the packet.



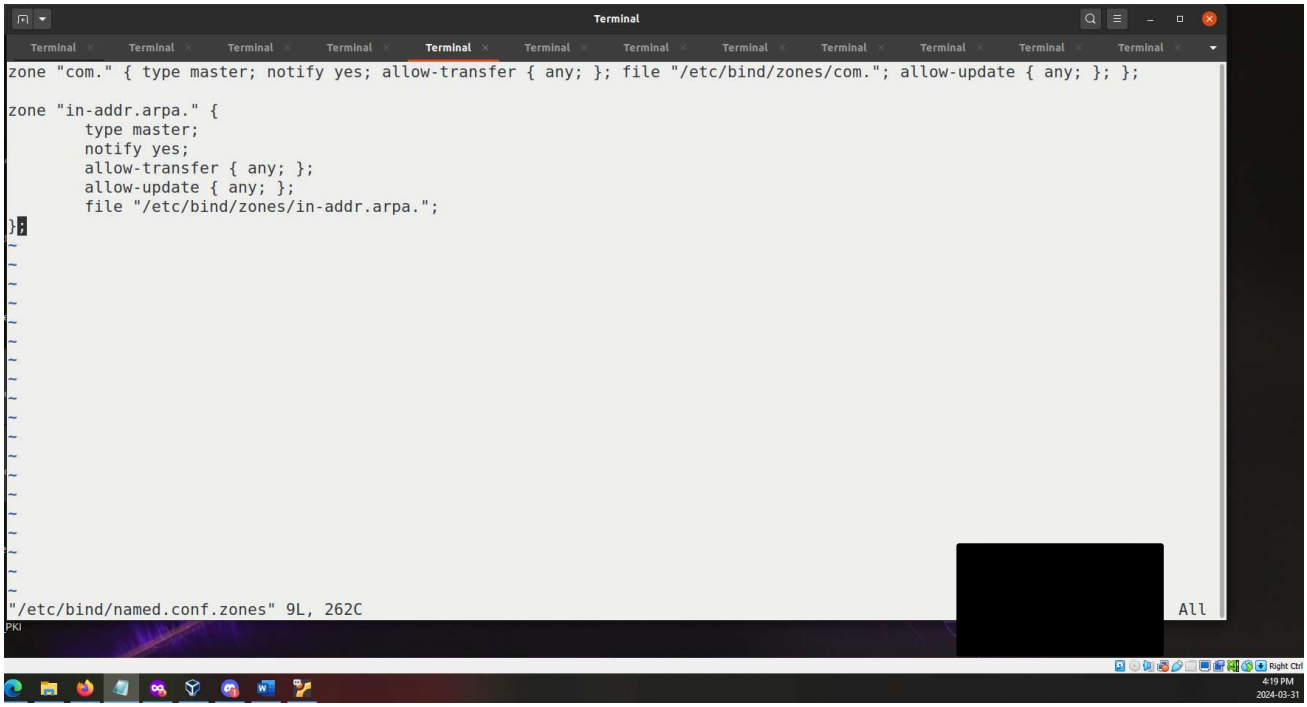
Task 6: Reverse DNS lookup

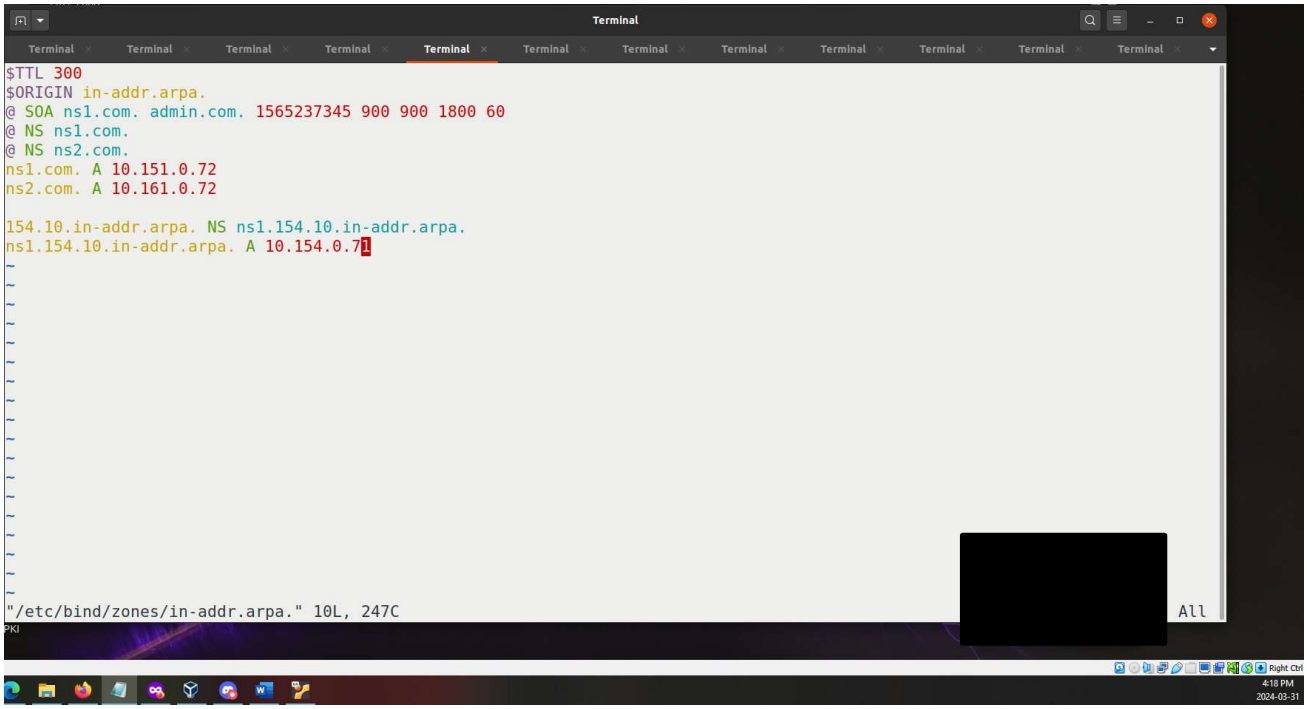
In this task our objective is to set up the infrastructure for reverse DNS lookup. Reverse DNS lookup involves finding out the hostname associated with an IP address. This process is similar to forward DNS lookup but operates in reverse. Given an IP address, such as 128.230.171.184, the DNS resolver constructs a "fake" domain name 184.171.230.128.in-addr.arpa and queries the DNS servers iteratively to resolve it.

I will first start by configuring the root server by adding NS records for the *in-addr.arpa* zone to the */etc/bind/zones/root* file on both root servers. These NS records will direct queries for the *in-addr.arpa* zone to the appropriate servers.

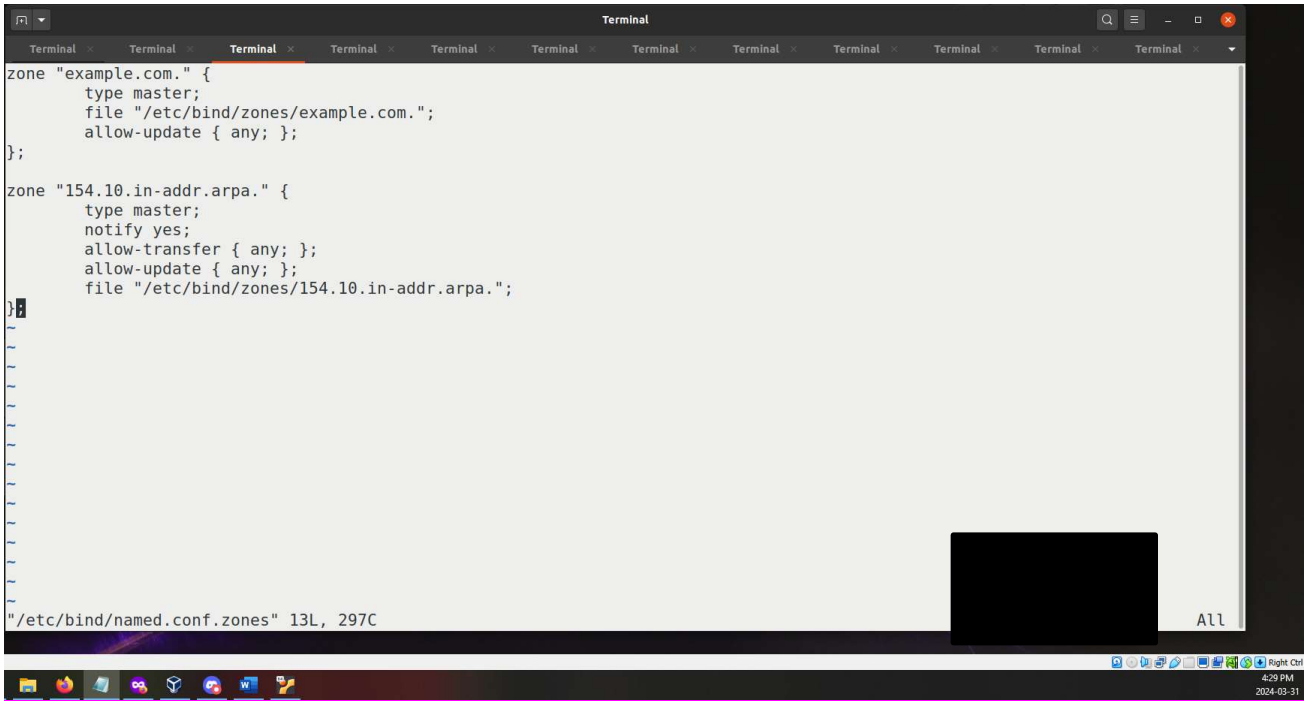


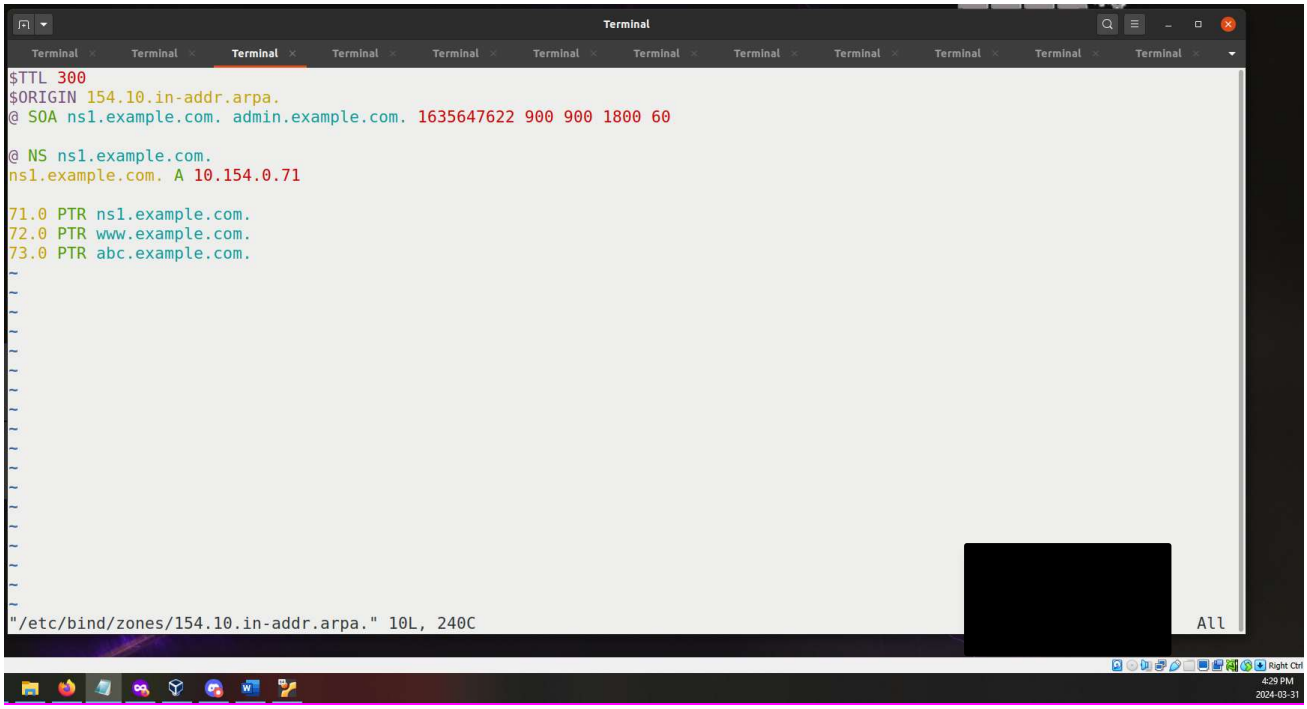
Then I will configure the in-addr.apa server to host the *in-addr.arpa* zone. We can use the *com* nameserver to host this zone by adding the zone configuration to the */etc/bind/named.conf.zones* file. I also created the zone file specified in the configuration, which should contain NS records for the 154.10.in-addr.arpa zone.





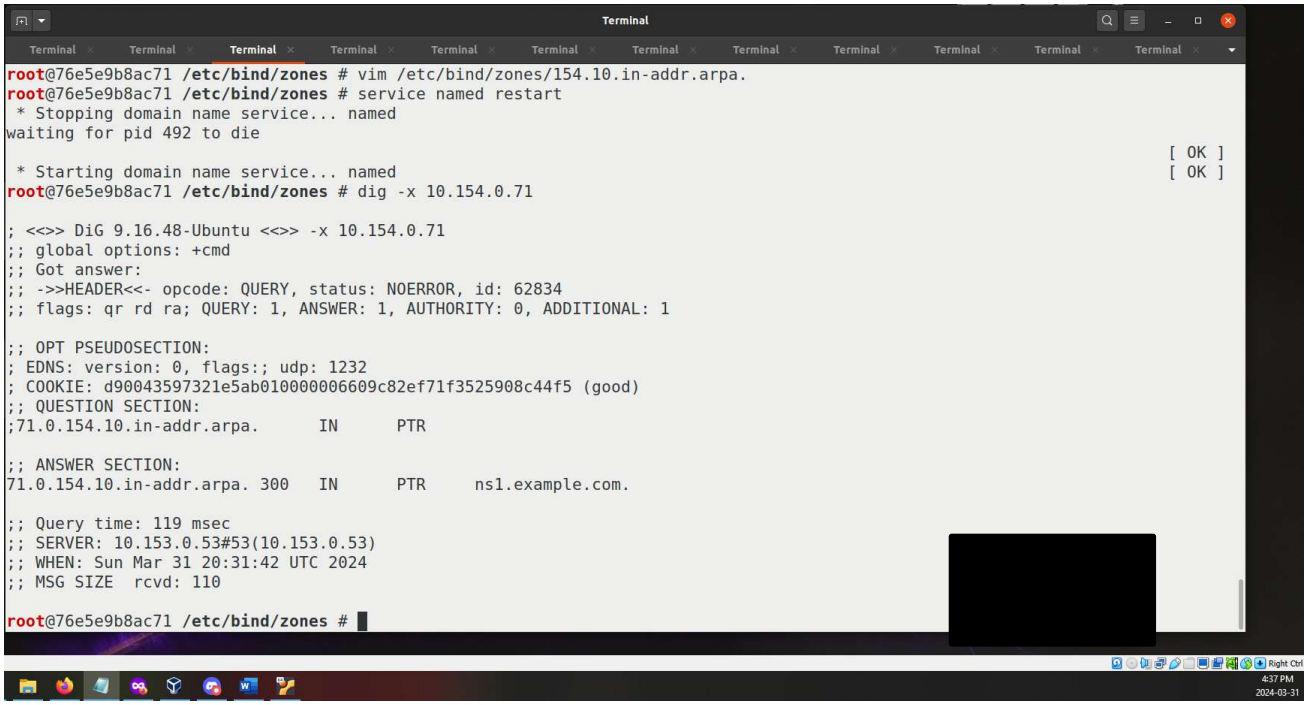
Finally, I will configure the 154.10.in-addr.arpa server as this server will handle the reverse lookup for the *example.com* domain. We will use the same server we used in the first task for *example.com* records. Then I will add a zone entry in its */etc/bind/named.conf.zones* file for the *154.10.in-addr.arpa zone* and create the corresponding zone file.

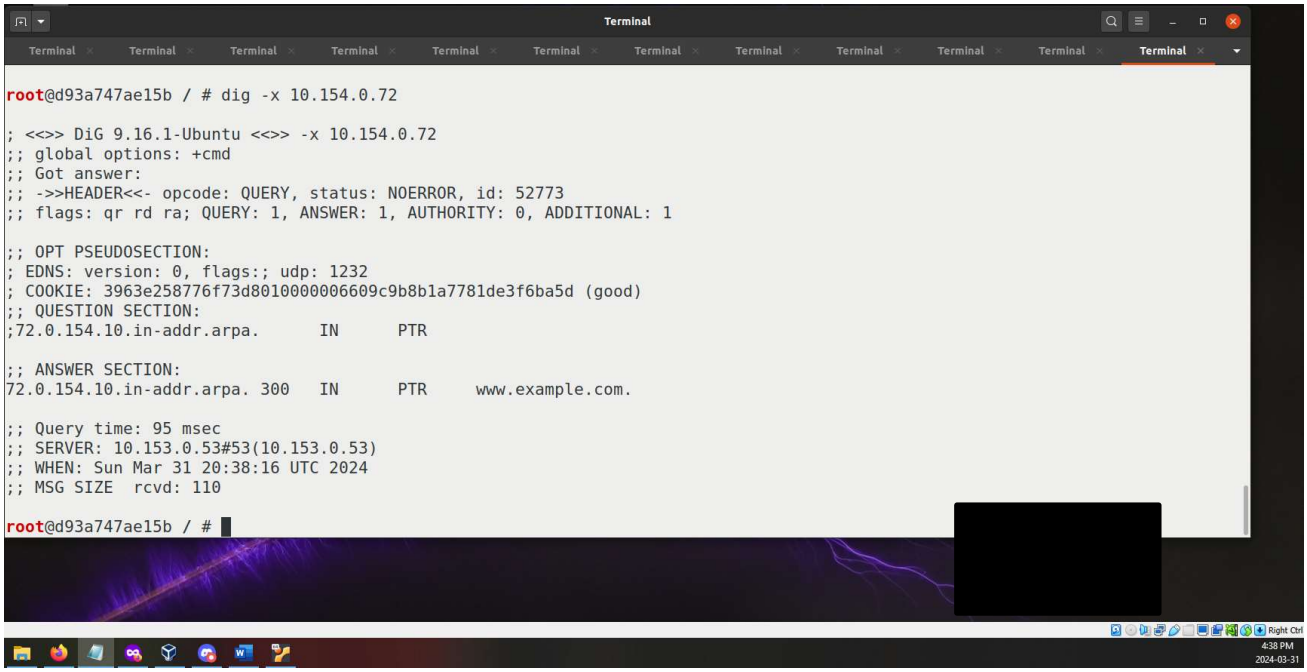


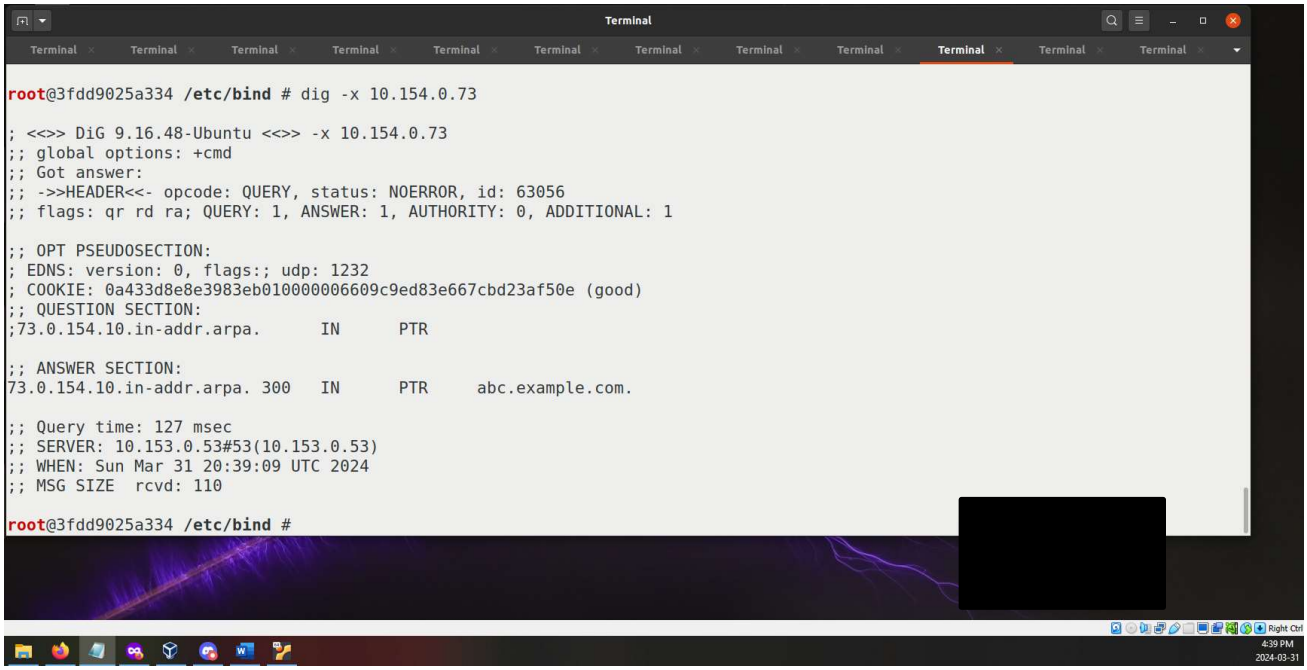


Then I will make sure to restart every DNS server after changing the configurations to update the changes.

For testing, I will run reverse query commands from different containers.

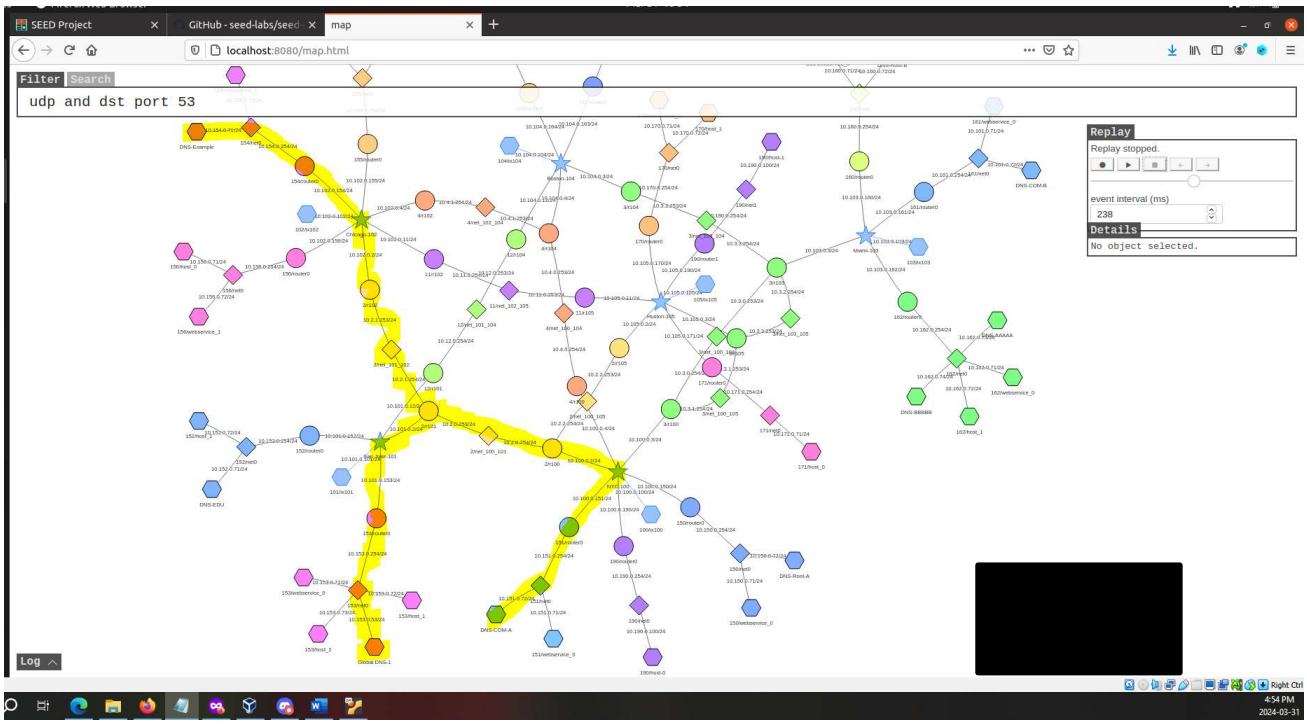

Terminal
root@76e5e9b8ac71 /etc/bind/zones # vim /etc/bind/zones/154.10.in-addr.arpa.
root@76e5e9b8ac71 /etc/bind/zones # service named restart
* Stopping domain name service... named
waiting for pid 492 to die
[OK]
* Starting domain name service... named
[OK]
root@76e5e9b8ac71 /etc/bind/zones # dig -x 10.154.0.71
; <<>> DiG 9.16.48-Ubuntu <<>> -x 10.154.0.71
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 62834
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: d90043597321e5ab010000006609c82ef71f3525908c44f5 (good)
;; QUESTION SECTION:
;71.0.154.10.in-addr.arpa. IN PTR
;; ANSWER SECTION:
71.0.154.10.in-addr.arpa. 300 IN PTR ns1.example.com.
;; Query time: 119 msec
;; SERVER: 10.153.0.53#53(10.153.0.53)
;; WHEN: Sun Mar 31 20:31:42 UTC 2024
;; MSG SIZE rcvd: 110
root@76e5e9b8ac71 /etc/bind/zones #
4:37 PM
2024-03-31


Terminal
root@d93a747ae15b / # dig -x 10.154.0.72
; <<>> DiG 9.16.1-Ubuntu <<>> -x 10.154.0.72
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 52773
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 3963e258776f73d8010000006609c9b8b1a7781de3f6ba5d (good)
;; QUESTION SECTION:
;72.0.154.10.in-addr.arpa. IN PTR
;; ANSWER SECTION:
72.0.154.10.in-addr.arpa. 300 IN PTR www.example.com.
;; Query time: 95 msec
;; SERVER: 10.153.0.53#53(10.153.0.53)
;; WHEN: Sun Mar 31 20:38:16 UTC 2024
;; MSG SIZE rcvd: 110
root@d93a747ae15b / #
4:38 PM
2024-03-31



As you can see in the above testing screenshots, there are different container IDs associated with each query which proves that the infrastructure for reverse DNS lookup is set up correctly and reverse queries could be made from any host in the environment.

Below you can see the screenshot of the packet trace using the map tool to show the DNS request traffic. In this case I made the request from “Global DNS-1” machine. The highlighted part shows the path taken by the packet, including travelling to hosts like COM-A server (bottom right) to hosts like DNS-EXAMPLE (top left) where the records are kept.



References

https://seedsecuritylabs.org/Labs_20.04/Networking/DNS/DNS_Infrastructure/

https://seedsecuritylabs.org/Labs_20.04/Files/DNS_Infrastructure/DNS_Infrastructure.pdf

Du, W. (2022). Internet Security: A Hands-on Approach (Third edition). Independent.