How to install a DNS server with BIND on Rocky Linux 9

5.

BIND or Berkeley Internet Name Domain is a free and open source DNS server software. It is one of the most popular DNS server software used by over 70% of DNS on the Internet. BIND has existed since the 1980s and is known for its flexibility, performance and functionality. BIND can be used both as an authoritative DNS and as a cache storage DNS and supports load balancing, dynamic updating, divided DNS, DNSSEC, IPv6 and many others.

BIND DNS software is one of the most reliable DNS servers for Unix-like operating systems. It is available on most Linux distributions and provides additional tools for diagnostics and testing of the DNS server.

This tutorial will show you how to configure the DNS server with BIND on a Rocky Linux 9 server. BIND is one of the most popular DNS server software that offers various features such as authoritative DNS, DNS cache only, basic DNS load balance, divided DNS, DNSSEC, IPv6 and many more.

Prerequisites

With this tutorial, you will configure and implement a BIND DNS Server with Master-Slave architecture. So, you will need two Rocky Linux servers. In addition, you will need root / administrator privileges on each server.

To set up a public DNS server that can manage your domain (authoritative DNS server), you must also have the registered domain name and the Glue Records configured.

Also, in this guide, suppose we have SELinux running in permissive mode.

System preparation

To get started with this guide, you will set the correct (Fully Qualified Domain Name) domain name on each of your Rocky Linux servers. This can be done through the command utility *hostnamectl* and the file */etc / hosts*.

Below are the detailed servers that will be used as an example for this guide:

On the main server, run the hostnamectl command utility below to set fqdn to *ns1.hwdomain.io*.

6. Below you must also run the hostnamectl command on the Slave server to set fqdn to *ns2.hwdomain.io*.

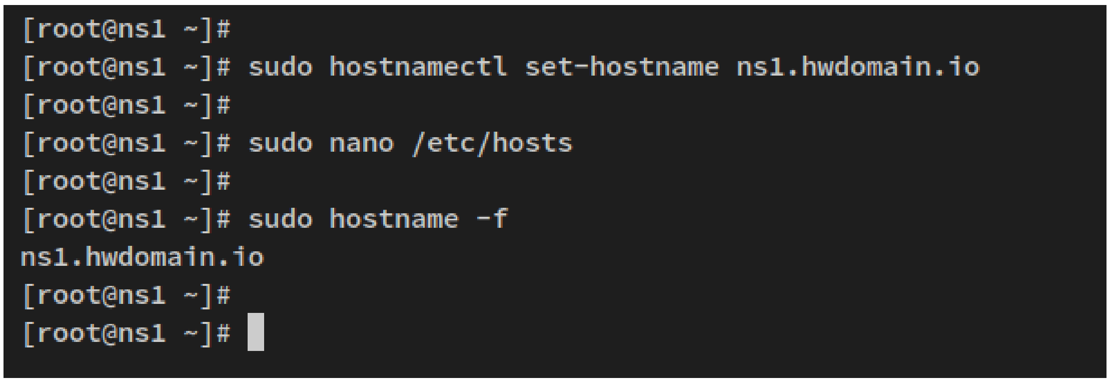
   Next, open the / etc / hosts file on both Master and Slave servers using the following nano editor command.

   Add the following line to the file.

   Save the file and exit the editor when you are done.

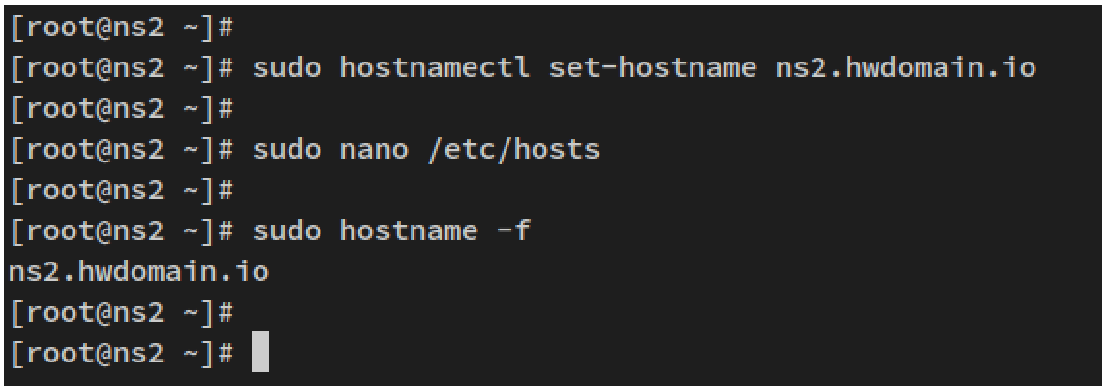
   Finally, run the following hostname command to verify fqdn on each server. You should see that the main server has fqdn *ns1.hwdomain.io* and the slave server has the fqdn *ns2.hwdomain.io*.

   The output of the main server is shown below.



```
[root@ns1 ~]#
[root@ns1 ~]# sudo hostnamectl set-hostname ns1.hwdomain.io
[root@ns1 ~]#
[root@ns1 ~]# sudo nano /etc/hosts
[root@ns1 ~]#
[root@ns1 ~]# sudo hostname -f
ns1.hwdomain.io
[root@ns1 ~]#
[root@ns1 ~]#
```

   And below is the output from the Slave server.



```
[root@ns2 ~]#
[root@ns2 ~]# sudo hostnamectl set-hostname ns2.hwdomain.io
[root@ns2 ~]#
[root@ns2 ~]# sudo nano /etc/hosts
[root@ns2 ~]#
[root@ns2 ~]# sudo hostname -f
ns2.hwdomain.io
[root@ns2 ~]#
[root@ns2 ~]#
```

   With fqdn configured, you are ready to install BIND on Rocky Linux servers.

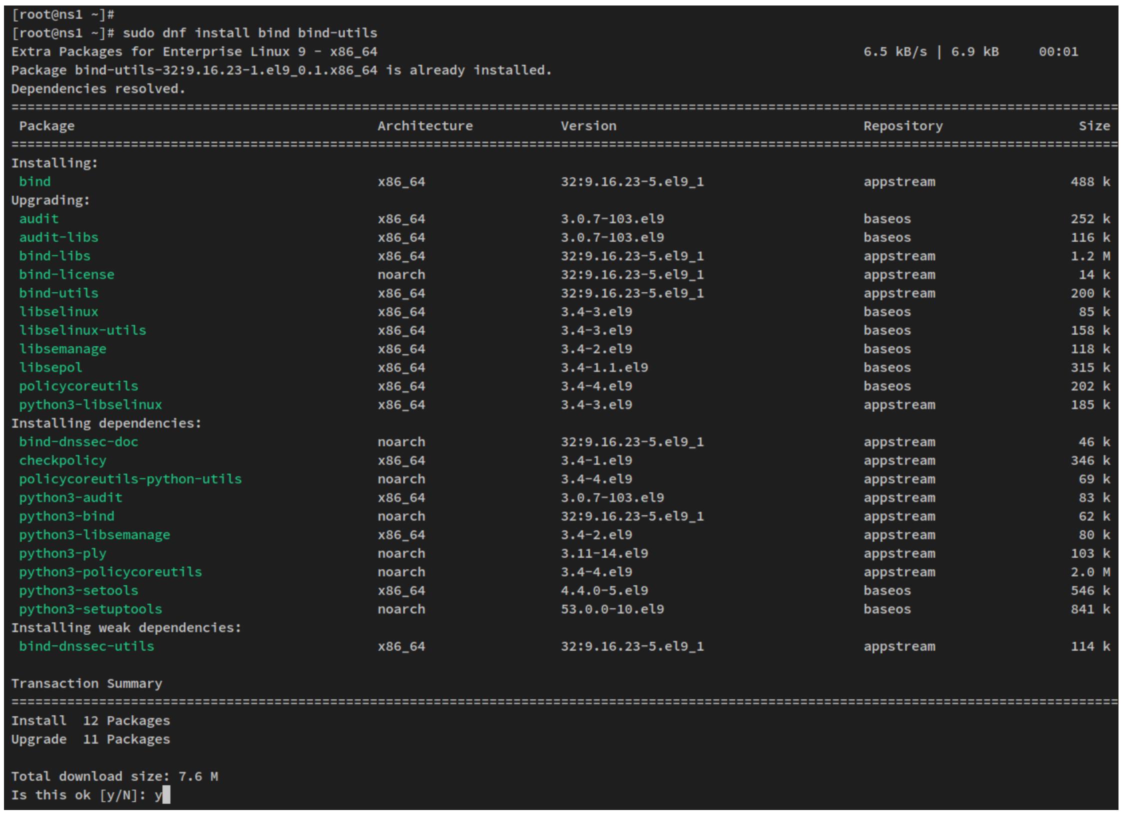
Installation of BIND packages

By default, the Rocky Linux AppStream repository provides the latest stable version of the BIND package. At the time of writing, the current stable version of

7. BIND is v9.16.

   In this step, you will install BIND packages on both Master and Slave servers. Then configure BIND to run only on IPv4 and configure the firewalld to allow the DNS port.

   Run the following dnf command to install BIND packages on both Master and Slave servers. When confirmation is requested, enter y to confirm and press ENTER to proceed.



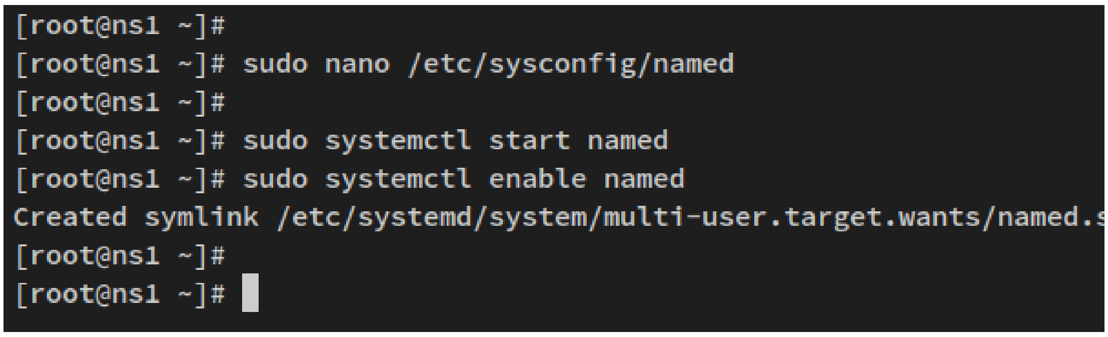
```
[root@ns1 ~]#
[root@ns1 ~]# sudo dnf install bind bind-utils
Extra Packages for Enterprise Linux 9 - x86_64                                               6.5 kB/s | 6.9 kB     00:01
Package bind-utils-32:9.16.23-1.el9_0.1.x86_64 is already installed.
Dependencies resolved.
==========================================================================================================================
 Package                              Architecture          Version                             Repository            Size
==========================================================================================================================
Installing:
 bind                                 x86_64                32:9.16.23-5.el9_1                  appstream            488 k
Upgrading:
 audit                                x86_64                3.0.7-103.el9                       baseos               252 k
 audit-libs                           x86_64                3.0.7-103.el9                       baseos               116 k
 bind-libs                            x86_64                32:9.16.23-5.el9_1                  appstream            1.2 M
 bind-license                         noarch                32:9.16.23-5.el9_1                  appstream             14 k
 bind-utils                           x86_64                32:9.16.23-5.el9_1                  appstream            200 k
 libselinux                           x86_64                3.4-3.el9                           baseos                85 k
 libselinux-utils                     x86_64                3.4-3.el9                           baseos               158 k
 libsemanage                          x86_64                3.4-2.el9                           baseos               118 k
 libsepol                             x86_64                3.4-1.1.el9                         baseos               315 k
 policycoreutils                      x86_64                3.4-4.el9                           baseos               202 k
 python3-libselinux                   x86_64                3.4-3.el9                           appstream            185 k
Installing dependencies:
 bind-dnssec-doc                      noarch                32:9.16.23-5.el9_1                  appstream             46 k
 checkpolicy                          x86_64                3.4-1.el9                           appstream            346 k
 policycoreutils-python-utils         noarch                3.4-4.el9                           appstream             69 k
 python3-audit                        x86_64                3.0.7-103.el9                       appstream             83 k
 python3-bind                         noarch                32:9.16.23-5.el9_1                  appstream             62 k
 python3-libsemanage                  x86_64                3.4-2.el9                           appstream             80 k
 python3-ply                          noarch                3.11-14.el9                         appstream            103 k
 python3-policycoreutils              noarch                3.4-4.el9                           appstream            2.0 M
 python3-setools                      x86_64                4.4.0-5.el9                         baseos               546 k
 python3-setuptools                   noarch                53.0.0-10.el9                       baseos               841 k
Installing weak dependencies:
 bind-dnssec-utils                    x86_64                32:9.16.23-5.el9_1                  appstream            114 k

Transaction Summary
==========================================================================================================================
Install  12 Packages
Upgrade  11 Packages

Total download size: 7.6 M
Is this ok [y/N]: y
```

   After installing the BIND packages, open the configuration */etc / sysconfig / named* using the following nano editor command.

   Add the default OPTIONS = .. with the following line. This command option for bind or named will only run BIND on IPv4.

   Save the file and exit the editor when you are done.

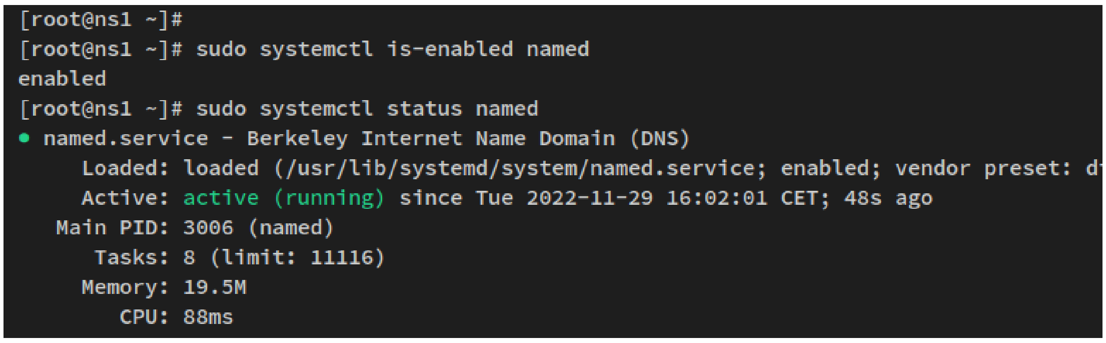
   Next, run the systemctl command utility below to start and enable the BIND service *named*. The named service should now be running and enabled, which will automatically start on startup.

8. 

```
[root@ns1 ~]#
[root@ns1 ~]# sudo nano /etc/sysconfig/named
[root@ns1 ~]#
[root@ns1 ~]# sudo systemctl start named
[root@ns1 ~]# sudo systemctl enable named
Created symlink /etc/systemd/system/multi-user.target.wants/named.s
[root@ns1 ~]#
[root@ns1 ~]#
```

Now check the named service to make sure it is running and enabled through the following command.

You will receive an output similar to the following: the service called BIND is enabled and is currently running.



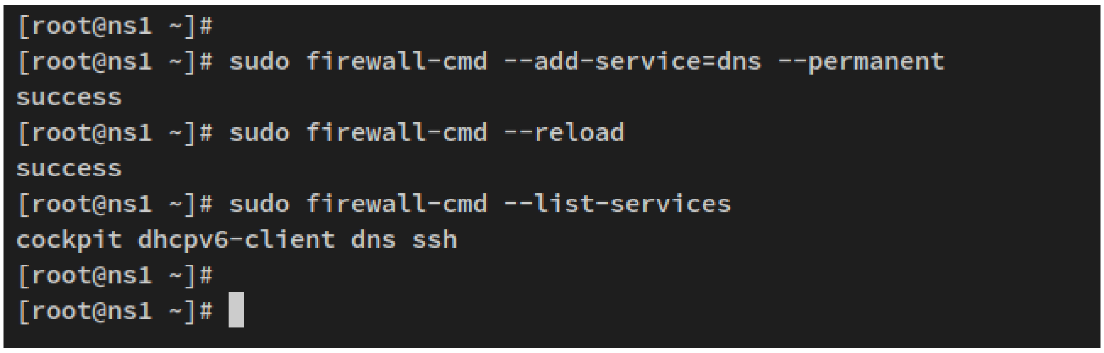
```
[root@ns1 ~]#
[root@ns1 ~]# sudo systemctl is-enabled named
enabled
[root@ns1 ~]# sudo systemctl status named
● named.service - Berkeley Internet Name Domain (DNS)
     Loaded: loaded (/usr/lib/systemd/system/named.service; enabled; vendor preset: d
     Active: active (running) since Tue 2022-11-29 16:02:01 CET; 48s ago
   Main PID: 3006 (named)
      Tasks: 8 (limit: 11116)
     Memory: 19.5M
        CPU: 88ms
```

With the service called BIND running, you will need to add the DNS port to the firewalld, which is enabled and running by default on Rocky Linux.

Run the firewall-cmd command utility below to add the DNS service to the firewalld file. Then reload the firewalld to apply the changes.

If you check the list of firewalled-enabled services, you should see that the DNS service is enabled. Run the firewall-cmd command below to check the list of services.

Production:



```
[root@ns1 ~]#
[root@ns1 ~]# sudo firewall-cmd --add-service=dns --permanent
success
[root@ns1 ~]# sudo firewall-cmd --reload
success
[root@ns1 ~]# sudo firewall-cmd --list-services
cockpit dhcpv6-client dns ssh
[root@ns1 ~]#
[root@ns1 ~]#
```

9. At this point, you have finished configuring fqdn, installed BIND packages and also configured firewalld. With this in mind, you can now start configuring BIND Master on the Master server.

Configuration of the main DNS BIND server

In this step, you will configure the BIND Master server using Rocky Linux ns1.hwdomain.io and the IP address of the server is 192.168.5.100. Make sure you run the following commands on the main server.

Configure the BIND Master server with the following steps:

- The basic configuration will include the configuration of the ACL (Access Control List), the setting of the IP address to perform the BIND service, the setting of the forwarders and much more.
- Zone setting: this is where you create configurations for your domain. This includes the main domain configuration and the reverse DNS configuration.

10.

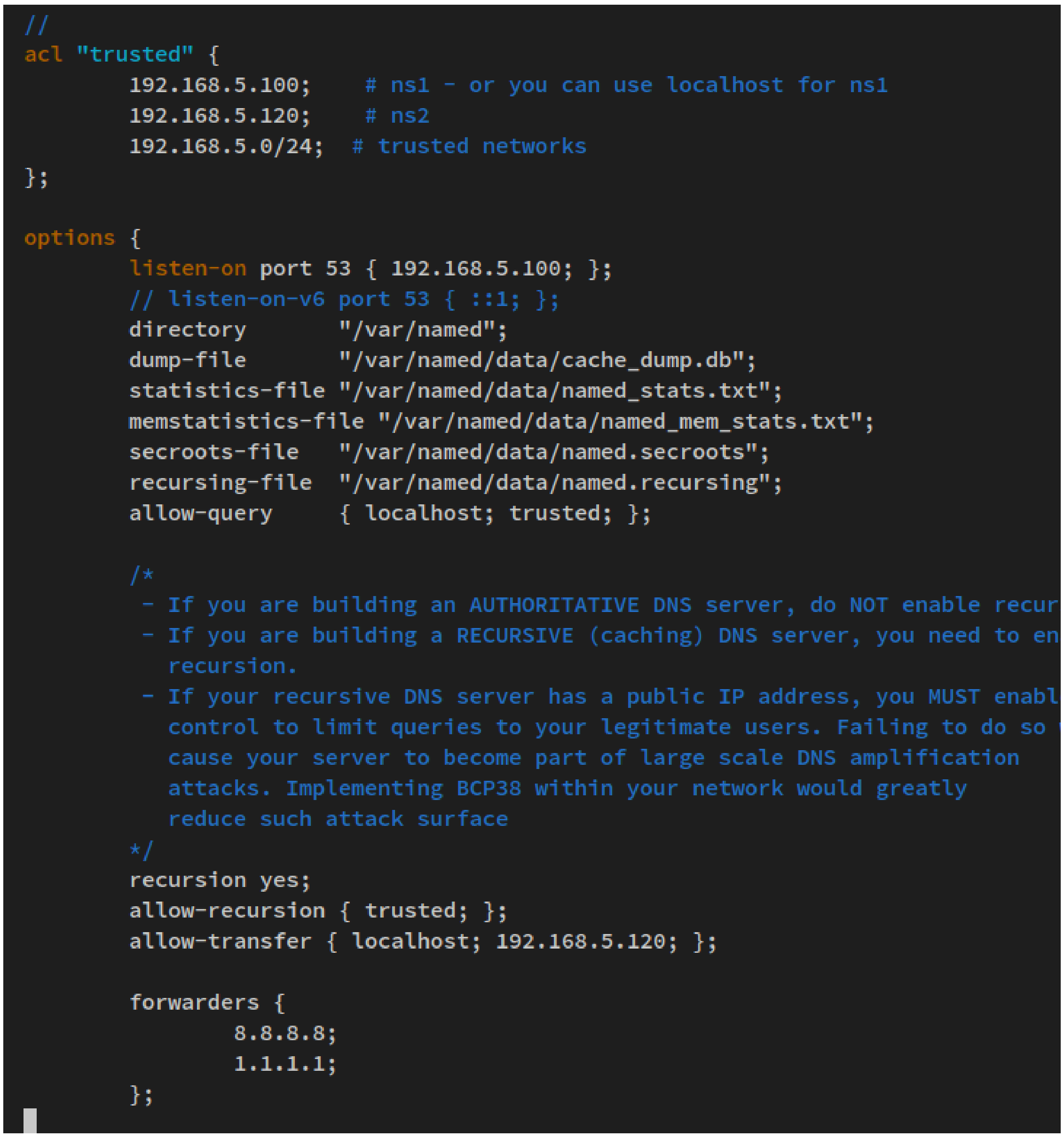
Basic configuration

The default BIND configuration on the RHEL-based distribution is available in */etc/named.conf*.

Now open the file */etc/named.conf* using the following nano editor command below.

Change the default configuration with the following lines.

Save the file and exit the editor when you are done.



```
//
acl "trusted" {
        192.168.5.100;    # ns1 - or you can use localhost for ns1
        192.168.5.120;    # ns2
        192.168.5.0/24;  # trusted networks
};

options {
        listen-on port 53 { 192.168.5.100; };
        // listen-on-v6 port 53 { ::1; };
        directory       "/var/named";
        dump-file       "/var/named/data/cache_dump.db";
        statistics-file "/var/named/data/named_stats.txt";
        memstatistics-file "/var/named/data/named_mem_stats.txt";
        secroots-file   "/var/named/data/named.secroots";
        recursing-file  "/var/named/data/named.recursing";
        allow-query     { localhost; trusted; };

        /*
         - If you are building an AUTHORITATIVE DNS server, do NOT enable recur
         - If you are building a RECURSIVE (caching) DNS server, you need to en
           recursion.
         - If your recursive DNS server has a public IP address, you MUST enabl
           control to limit queries to your legitimate users. Failing to do so
           cause your server to become part of large scale DNS amplification
           attacks. Implementing BCP38 within your network would greatly
           reduce such attack surface
        */
        recursion yes;
        allow-recursion { trusted; };
        allow-transfer { localhost; 192.168.5.120; };

        forwarders {
                8.8.8.8;
                1.1.1.1;
        };
```

With this configuration, you will configure BIND with the following configurations:

11.

- Set up a trusted ACL that allows any query from local networks.
- Run the BIND service on 192.168.5.100 with the default port 53.
- Enable recursion and allow recursion from reliable ACL networks.
- Allow zone transfer to the Slave server with IP address 192.168.5.120.
- Set forwarders with Cloudflare's public DNS server 1.1.1.1 and Google's 8.8.8.8.

12.

Next, run the following command to verify the BIND configuration */etc/named.conf*.

Finally, run the systemctl command utility below to restart the service called BIND and apply the changes.

You have now finished the basic configuration of the BIND DNS Server.

Zone setting

Now you will set up zones with the DNS Server BIND. You will create a new DNS server with the address ns1.hwdomain.io and ns2.hwdomain.io.

To get started, open the BIND configuration */etc/named.conf* via the nano editor command below.

Add the following configuration to the bottom of the row.

Save the file and exit the editor when you are done.

```
include "/etc/named.rfc1912.zones";
include "/etc/named.root.key";
include "/etc/named/zones.conf.local";
```

Next, create a new configuration */etc/named/zones.conf.local* using the nano editor below.

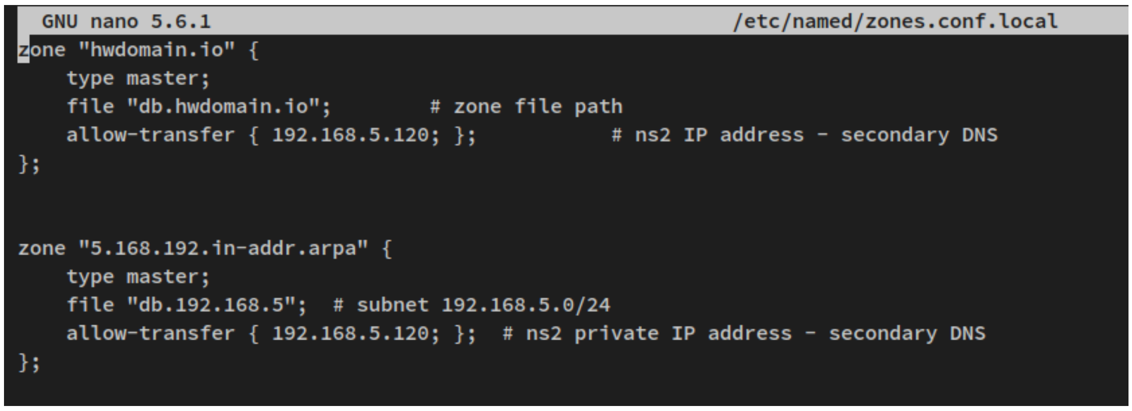
Add the following lines to the file.

Save and close the file when you're done.

With this configuration, you will define the following configurations:

- Create two zones for the hwdomain.io domain and the reverse DNS 5.168.192.in-addr.arpa.
- Both areas type as masters.
- Allow zone transfer to the slave DNS server which will run on the IP address of the 192.168.5.120 server.

13.



```
  GNU nano 5.6.1                                          /etc/named/zones.conf.local
zone "hwdomain.io" {
    type master;
    file "db.hwdomain.io";        # zone file path
    allow-transfer { 192.168.5.120; };           # ns2 IP address - secondary DNS
};


zone "5.168.192.in-addr.arpa" {
    type master;
    file "db.192.168.5";  # subnet 192.168.5.0/24
    allow-transfer { 192.168.5.120; };  # ns2 private IP address - secondary DNS
};
```

Next, create a new DNS zone configuration, */var/named/db.hwdomain.io*, using the following nano editor command.

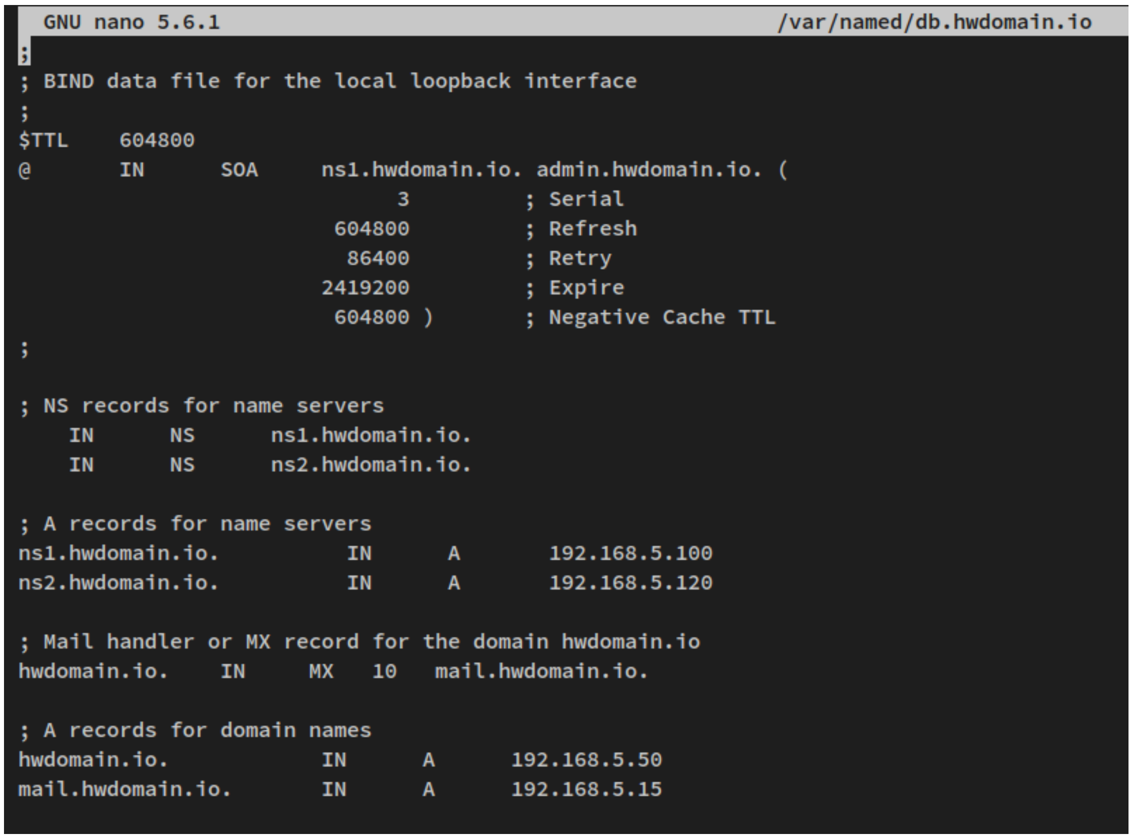
Add the following lines to the file.

Save and close the file when you're done.

In this example, you will set the zones with the following configurations:

- Define the server records of names ns1.hwdomain.io with the IP address 192.168.5.100 and ns2.hwdomain.io with the IP address 192.168.5.120.
- Define two additional domains, hwdomain.io, which will be resolved in the IP address of the 192.168.5.50 server and the mail.hwdomain.io domain in the IP address 192.168.5.15.
- You will also create an MX record for the hwdomain.io domain which will be managed by the mail server mail.hwdomain.io.

14.



```
  GNU nano 5.6.1                                              /var/named/db.hwdomain.io
;
; BIND data file for the local loopback interface
;
$TTL    604800
@       IN      SOA     ns1.hwdomain.io. admin.hwdomain.io. (
                              3         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;

; NS records for name servers
    IN      NS      ns1.hwdomain.io.
    IN      NS      ns2.hwdomain.io.

; A records for name servers
ns1.hwdomain.io.          IN      A       192.168.5.100
ns2.hwdomain.io.          IN      A       192.168.5.120

; Mail handler or MX record for the domain hwdomain.io
hwdomain.io.    IN     MX   10   mail.hwdomain.io.

; A records for domain names
hwdomain.io.            IN      A      192.168.5.50
mail.hwdomain.io.       IN      A      192.168.5.15
```

Next, you will begin to configure the reverse DNS for the hwdomain.io domain.

Create a new reverse DNS configuration */var/named/db.192.168.5* using the nano editor command below.

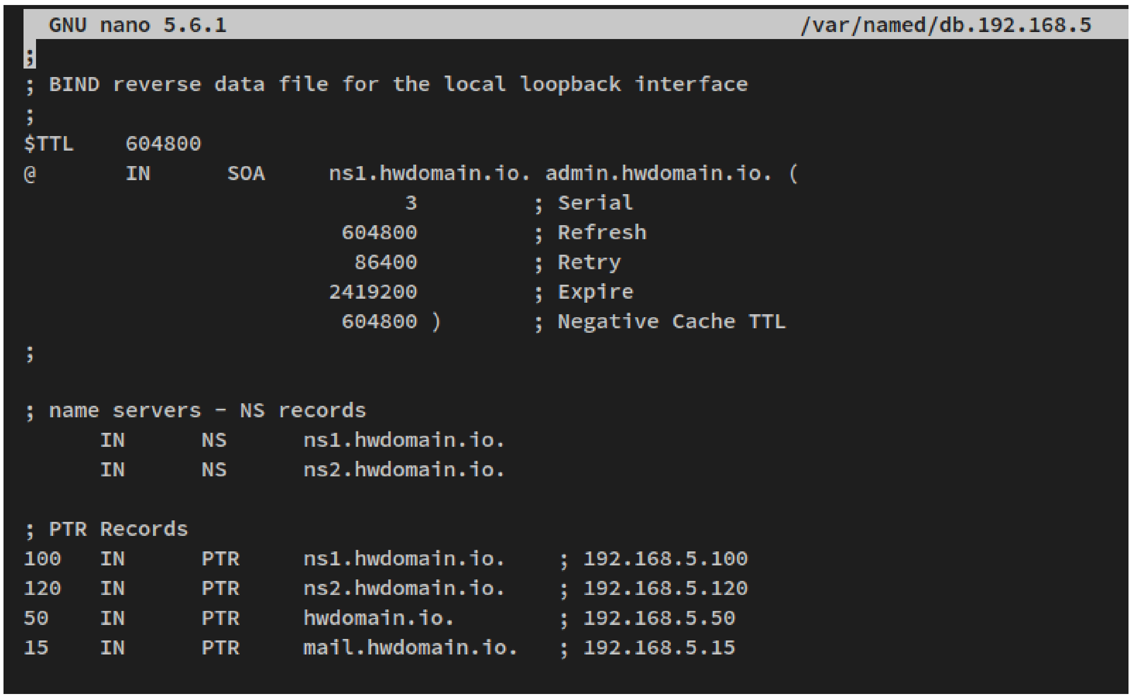
Add the following lines to the file.

Save and close the file when you're done.

With this configuration, you will set the reverse DNS or PTR records as below.

- In the reverse DNS configuration, you will also define the name server ns1.hwdomain.io and ns2.hwdomain.io.
- Each reverse DNS configuration uses the last IP address number that is resolved in each domain. In this example, the ns1.hwdomain.io name server with IP address 192.168.5.100 and the PTR record should be 100.
- The rest of the PTR records are the same as described above.

15.



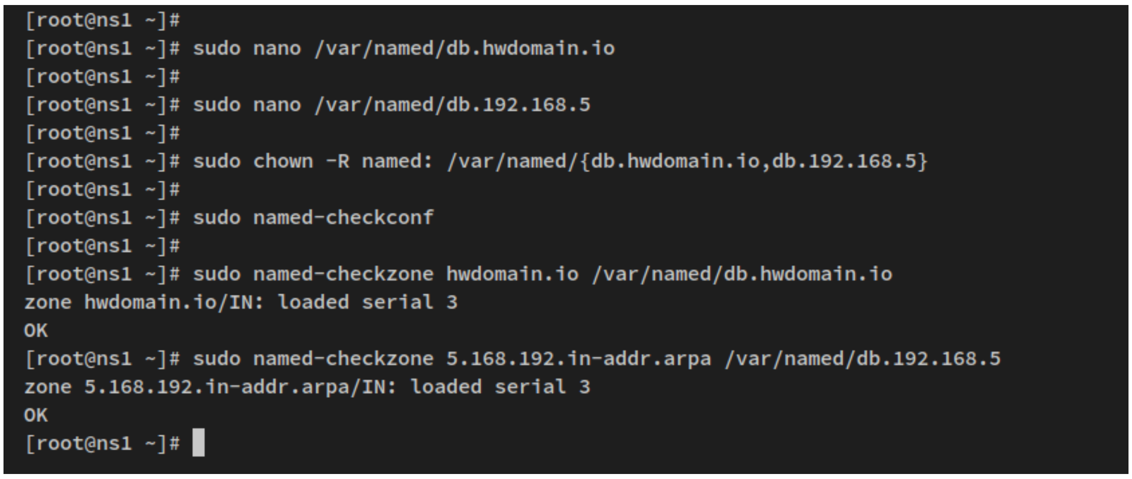
```
  GNU nano 5.6.1                                             /var/named/db.192.168.5
;
; BIND reverse data file for the local loopback interface
;
$TTL    604800
@       IN      SOA     ns1.hwdomain.io. admin.hwdomain.io. (
                              3         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;

; name servers - NS records
      IN      NS      ns1.hwdomain.io.
      IN      NS      ns2.hwdomain.io.

; PTR Records
100   IN      PTR     ns1.hwdomain.io.    ; 192.168.5.100
120   IN      PTR     ns2.hwdomain.io.    ; 192.168.5.120
50    IN      PTR     hwdomain.io.        ; 192.168.5.50
15    IN      PTR     mail.hwdomain.io.   ; 192.168.5.15
```

At this point, you created the two zone configuration for the hwdomain.io domain and created an ns1.hwdomain.io and ns2.hwdomain.io name server.

Now run the chmod command below to change the ownership of both zone configurations.

Then check the zone configuration files via the command utility *named-checkconf* below.

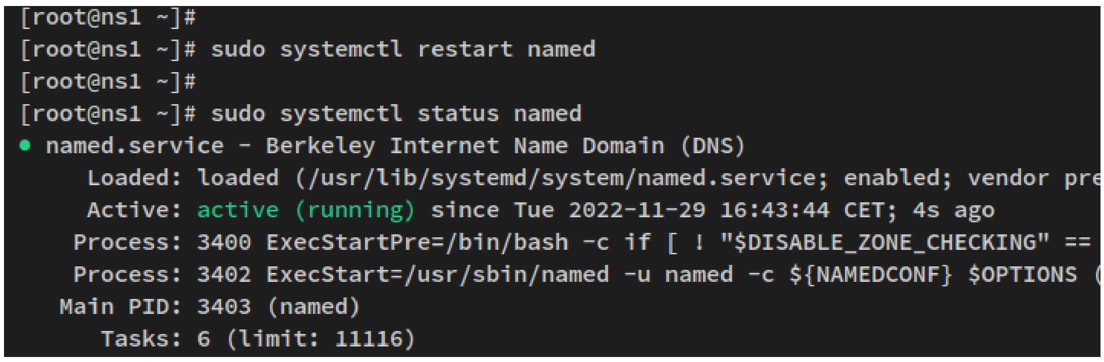
If you have the correct BIND configurations, you will receive the output as in the following screen.



```
[root@ns1 ~]#
[root@ns1 ~]# sudo nano /var/named/db.hwdomain.io
[root@ns1 ~]#
[root@ns1 ~]# sudo nano /var/named/db.192.168.5
[root@ns1 ~]#
[root@ns1 ~]# sudo chown -R named: /var/named/{db.hwdomain.io,db.192.168.5}
[root@ns1 ~]#
[root@ns1 ~]# sudo named-checkconf
[root@ns1 ~]#
[root@ns1 ~]# sudo named-checkzone hwdomain.io /var/named/db.hwdomain.io
zone hwdomain.io/IN: loaded serial 3
OK
[root@ns1 ~]# sudo named-checkzone 5.168.192.in-addr.arpa /var/named/db.192.168.5
zone 5.168.192.in-addr.arpa/IN: loaded serial 3
OK
[root@ns1 ~]# █
```

16. Finally, run the systemctl command below to restart the service called BIND and apply the changes. Then, check the status of the BIND service to make sure the service is running.

    You will receive an output similar to the following: the BIND service *named* it is running and you have finished configuring BIND Master.



```
[root@ns1 ~]#
[root@ns1 ~]# sudo systemctl restart named
[root@ns1 ~]#
[root@ns1 ~]# sudo systemctl status named
● named.service - Berkeley Internet Name Domain (DNS)
     Loaded: loaded (/usr/lib/systemd/system/named.service; enabled; vendor pre
     Active: active (running) since Tue 2022-11-29 16:43:44 CET; 4s ago
    Process: 3400 ExecStartPre=/bin/bash -c if [ ! "$DISABLE_ZONE_CHECKING" ==
    Process: 3402 ExecStart=/usr/sbin/named -u named -c ${NAMEDCONF} $OPTIONS (
   Main PID: 3403 (named)
      Tasks: 6 (limit: 11116)
```

In the next step, you will configure the BIND Slave server.

Configuration of the secondary DNS BIND server

After configuring the DNS Master server, you will begin configuring the BIND Slave server on the ns2.hwdomain.io server with the IP address 192.168.5.120.

The basic configuration for named.conf is similar to BIND Master and for zone files it is possible to define the file name without creating an actual file on the BIND Slave server.

Before you start, be sure to run the following commands on the BIND Slave server.

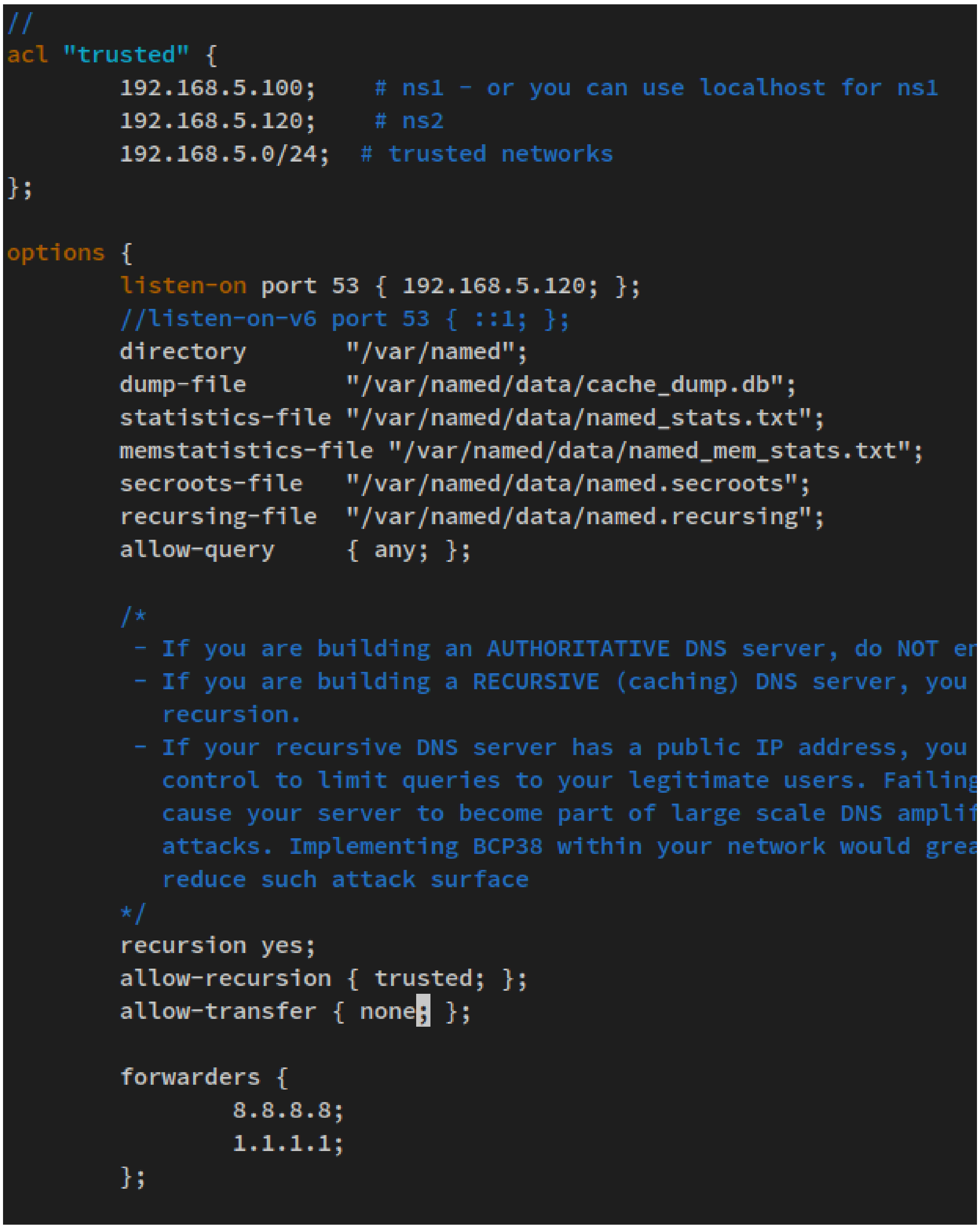
Now open the BIND configuration */etc/named.conf* on the Slave server via the nano editor command below.

Change the default configuration with the following lines.

The settings are similar to the BIND Master server and some differences in configurations are shown below.

- The BIND service will run on IP 192.168.5.120 on the Slave server.
- Recursion is enabled, but the transfer permit is configured on none.

17.



```
//
acl "trusted" {
        192.168.5.100;    # ns1 - or you can use localhost for ns1
        192.168.5.120;    # ns2
        192.168.5.0/24;  # trusted networks
};

options {
        listen-on port 53 { 192.168.5.120; };
        //listen-on-v6 port 53 { ::1; };
        directory       "/var/named";
        dump-file       "/var/named/data/cache_dump.db";
        statistics-file "/var/named/data/named_stats.txt";
        memstatistics-file "/var/named/data/named_mem_stats.txt";
        secroots-file   "/var/named/data/named.secroots";
        recursing-file  "/var/named/data/named.recursing";
        allow-query     { any; };

        /*
         - If you are building an AUTHORITATIVE DNS server, do NOT er
         - If you are building a RECURSIVE (caching) DNS server, you
           recursion.
         - If your recursive DNS server has a public IP address, you
           control to limit queries to your legitimate users. Failing
           cause your server to become part of large scale DNS amplif
           attacks. Implementing BCP38 within your network would grea
           reduce such attack surface
        */
        recursion yes;
        allow-recursion { trusted; };
        allow-transfer { none; };

        forwarders {
                8.8.8.8;
                1.1.1.1;
        };
```

Now add the following line to the end of the named.conf file to define the zones.

Save the file and exit the editor when you are done.

Next, create a new configuration */etc/named/zones.conf.local* using the following nano editor command.

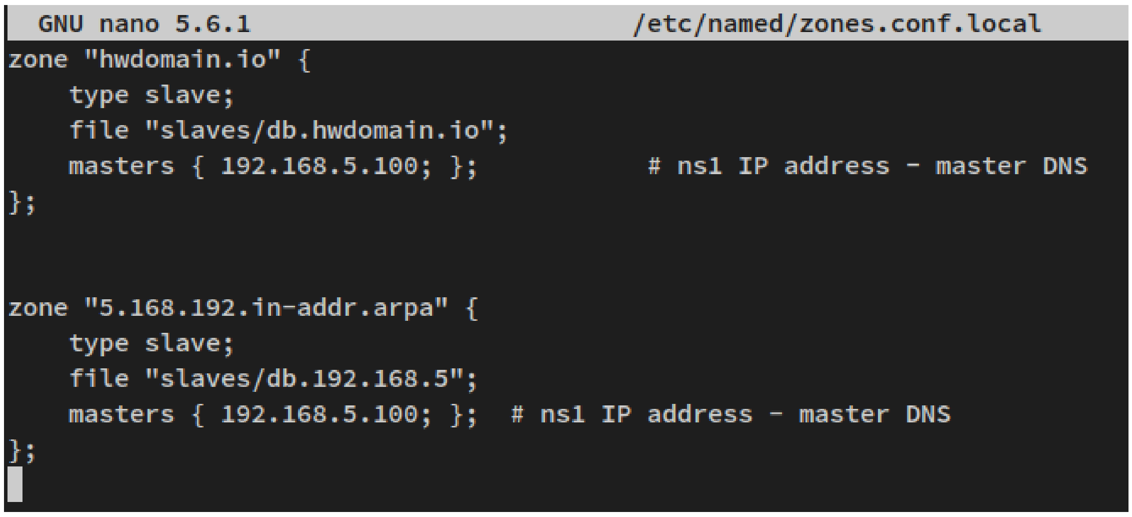
18. Add the following lines to the file.

    Save and close the file when you're done.

    With this, you will define some configurations below on the BIND Slave server:

- Define two zones for the hwdomain.io domain and its reverse DNS.
- Both types of areas are slave.
- The zone file for each will be taken from the / var / named / slaves directory, which is transferred from the BIND master server.

19.



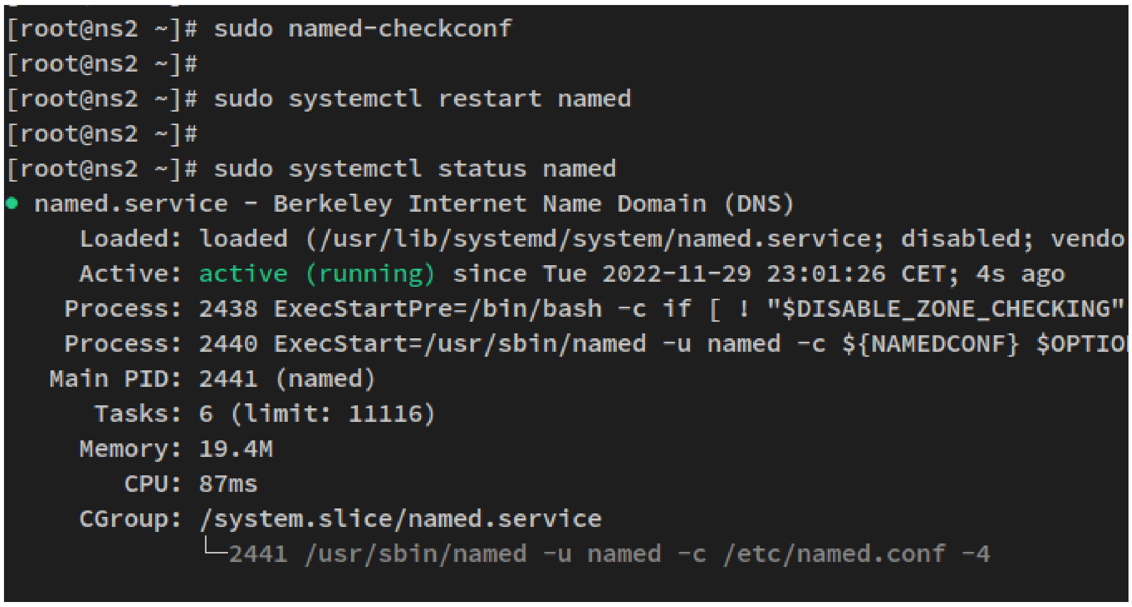
```
  GNU nano 5.6.1                        /etc/named/zones.conf.local
zone "hwdomain.io" {
    type slave;
    file "slaves/db.hwdomain.io";
    masters { 192.168.5.100; };           # ns1 IP address - master DNS
};


zone "5.168.192.in-addr.arpa" {
    type slave;
    file "slaves/db.192.168.5";
    masters { 192.168.5.100; };  # ns1 IP address - master DNS
};
```

Next, run the following named-checkconf command utility to check BIND configurations. Then, restart the service called BIND on the slave server via the systemctl command as below.

You will not have any error messages if you have a correct BIND configuration.

Finally, run the following systemctl command to verify the BIND service *named* on the slave server to make sure the service is running.

You will now receive the output similar to the following: the service called BIND is running on the BIND slave server.



```
[root@ns2 ~]# sudo named-checkconf
[root@ns2 ~]#
[root@ns2 ~]# sudo systemctl restart named
[root@ns2 ~]#
[root@ns2 ~]# sudo systemctl status named
● named.service - Berkeley Internet Name Domain (DNS)
     Loaded: loaded (/usr/lib/systemd/system/named.service; disabled; vendo
     Active: active (running) since Tue 2022-11-29 23:01:26 CET; 4s ago
    Process: 2438 ExecStartPre=/bin/bash -c if [ ! "$DISABLE_ZONE_CHECKING"
    Process: 2440 ExecStart=/usr/sbin/named -u named -c ${NAMEDCONF} $OPTIO
   Main PID: 2441 (named)
      Tasks: 6 (limit: 11116)
     Memory: 19.4M
        CPU: 87ms
     CGroup: /system.slice/named.service
             └─2441 /usr/sbin/named -u named -c /etc/named.conf -4
```

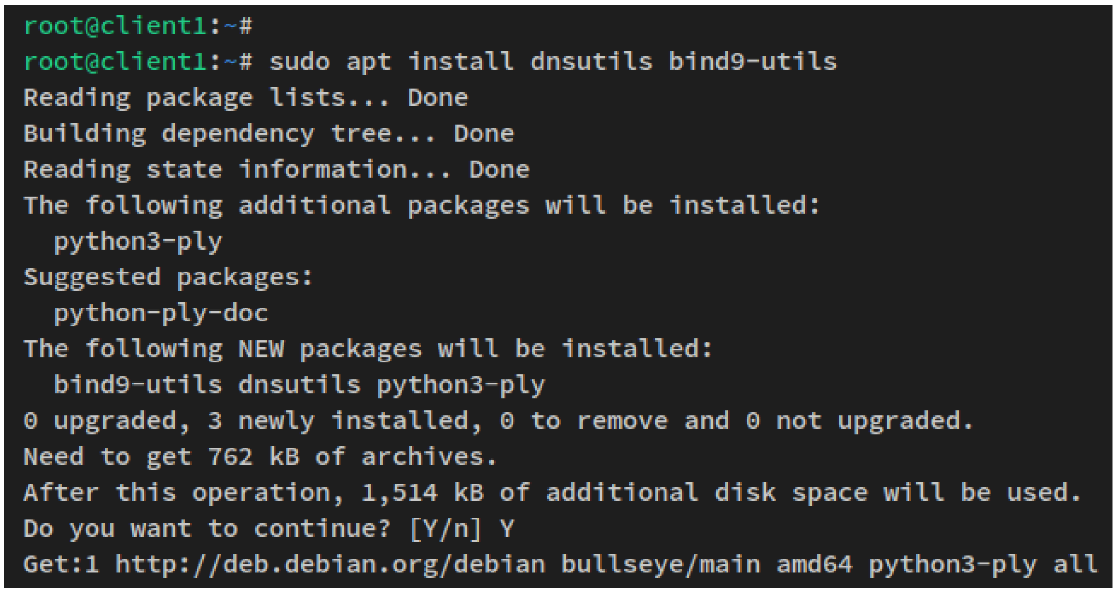
At this point you have finished installing BIND DNS with the Master-Slave architecture. Now you will be ready to start the test from the client computer.

20.

Customer test

This example uses a Debian system as a client machine, so install some packages below via APT before starting.

Enter y when confirmation is requested and press ENTER to proceed.



```
root@client1:~#
root@client1:~# sudo apt install dnsutils bind9-utils
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  python3-ply
Suggested packages:
  python-ply-doc
The following NEW packages will be installed:
  bind9-utils dnsutils python3-ply
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
Need to get 762 kB of archives.
After this operation, 1,514 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://deb.debian.org/debian bullseye/main amd64 python3-ply all
```

Next, run the following command to remove the default link file *\/etc/resolv.conf* and create a new file using the nano editor.

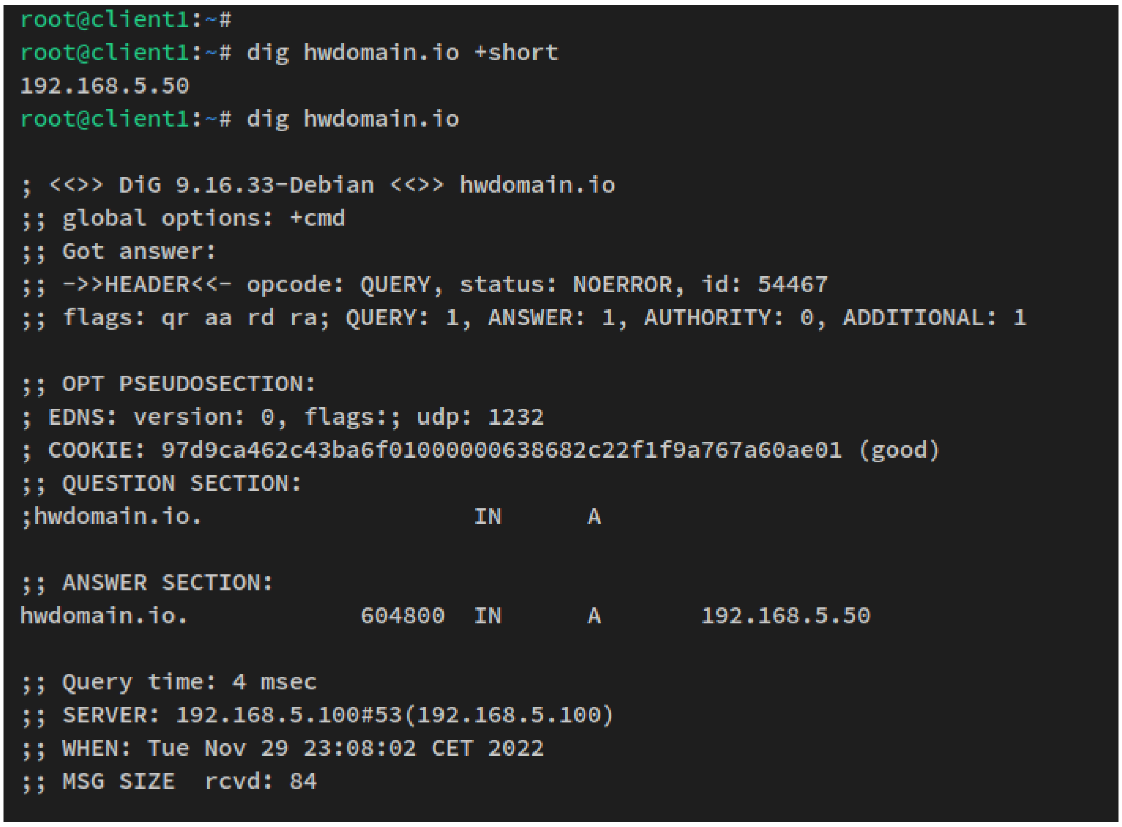
Add the following configuration to the file. In the following configuration we are defining three different resolvers, the DNS BIND master, the secondary DNS BIND server and the public DNS resolver of Cloudflare. When the client machine requests information about the domain name, the information will be taken from the DNS resolver, from top to bottom.

Save and close the file when you're done.

You are now ready to verify your DNS server from the client computer.

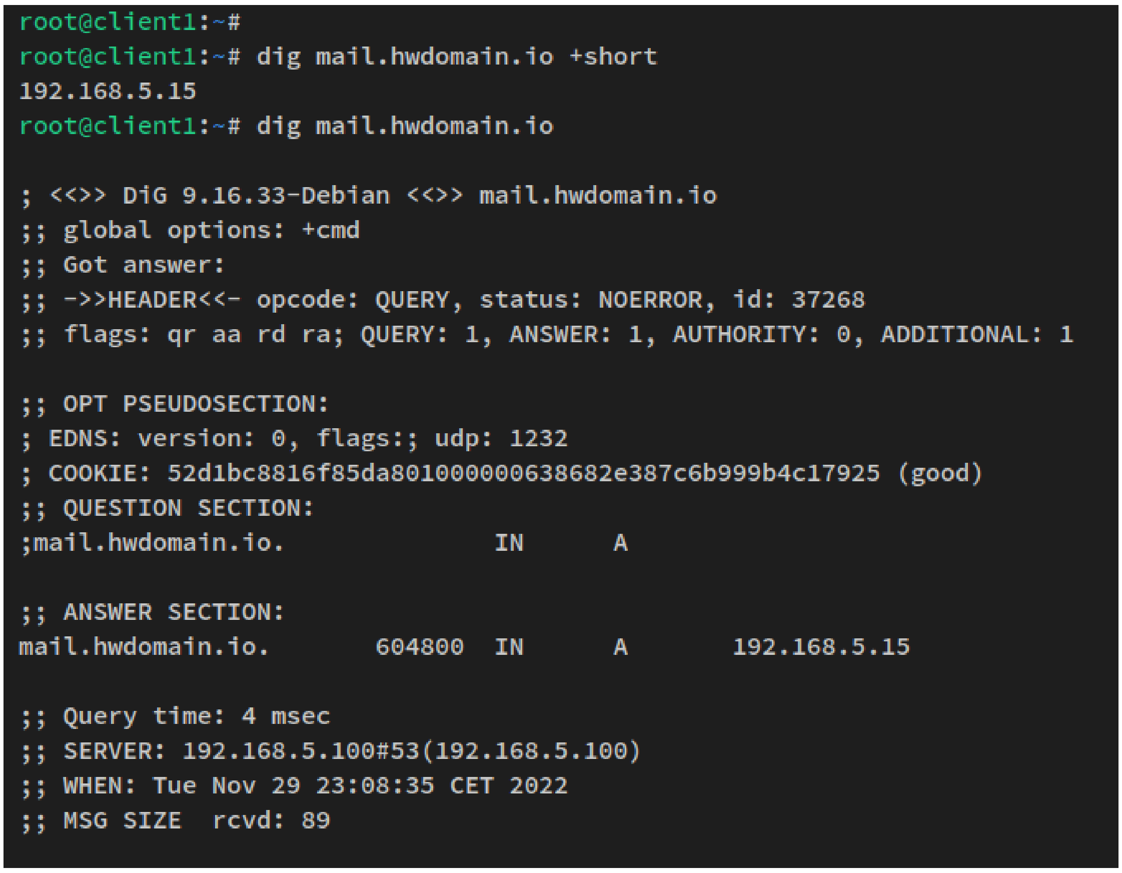
Run the dig command below to check the domain name \ hwdomain.io \ and \ mail.hwdomain.io\. And you should see that \ hwdomain.io \ is fixed in the IP address of the server \ 192.168.5.50 \, while the subdomain \ mail.hwdomain.io \ is managed by the IP address of the server \ 192.168.5.15\.

Verification of the domain name hwdomain.io.

21. 

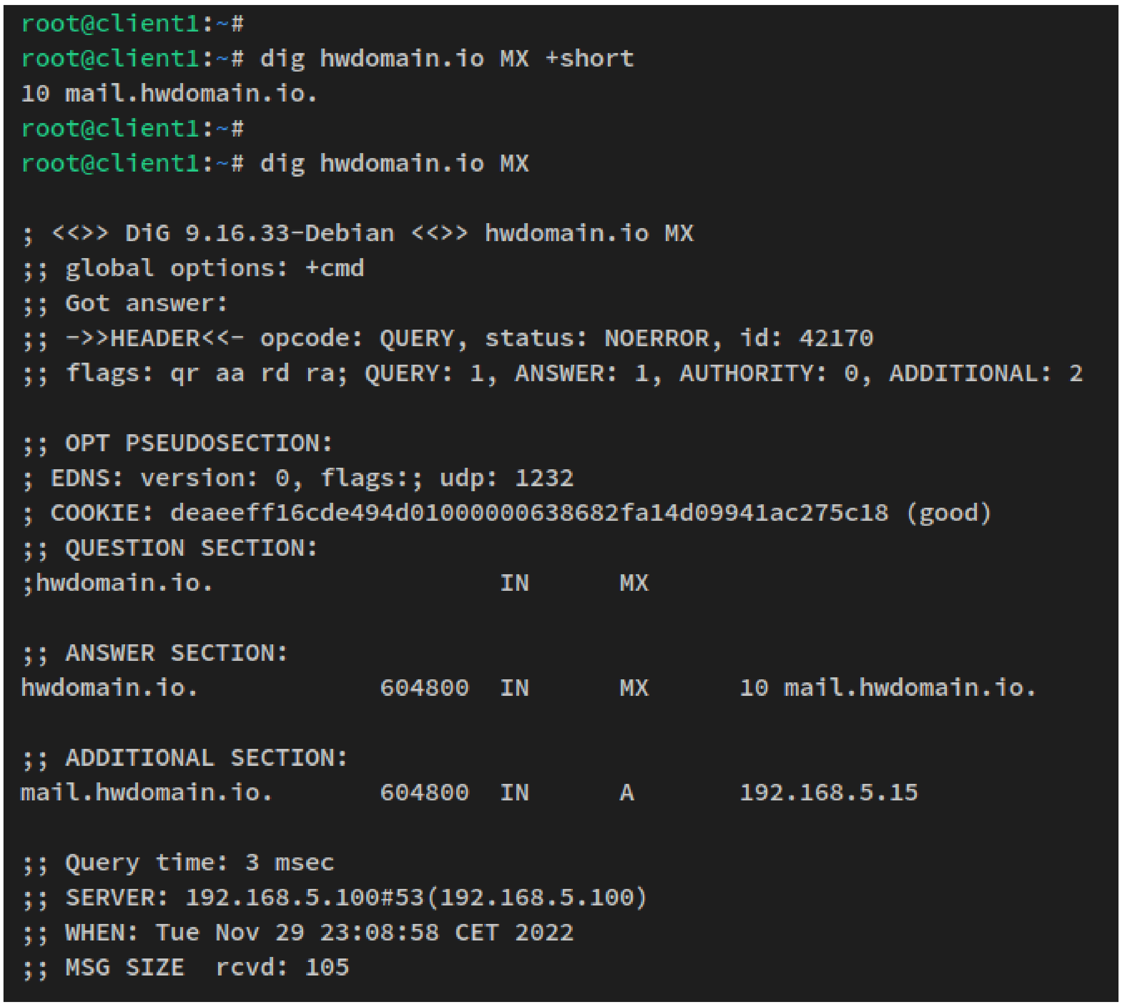
```
root@client1:~#
root@client1:~# dig hwdomain.io +short
192.168.5.50
root@client1:~# dig hwdomain.io

; <<>> DiG 9.16.33-Debian <<>> hwdomain.io
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 54467
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 97d9ca462c43ba6f01000000638682c22f1f9a767a60ae01 (good)
;; QUESTION SECTION:
;hwdomain.io.                   IN      A

;; ANSWER SECTION:
hwdomain.io.            604800  IN      A       192.168.5.50

;; Query time: 4 msec
;; SERVER: 192.168.5.100#53(192.168.5.100)
;; WHEN: Tue Nov 29 23:08:02 CET 2022
;; MSG SIZE  rcvd: 84
```

checking the subdomain mail.hwdomain.io.



```
root@client1:~#
root@client1:~# dig mail.hwdomain.io +short
192.168.5.15
root@client1:~# dig mail.hwdomain.io

; <<>> DiG 9.16.33-Debian <<>> mail.hwdomain.io
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 37268
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 52d1bc8816f85da801000000638682e387c6b999b4c17925 (good)
;; QUESTION SECTION:
;mail.hwdomain.io.              IN      A

;; ANSWER SECTION:
mail.hwdomain.io.       604800  IN      A       192.168.5.15

;; Query time: 4 msec
;; SERVER: 192.168.5.100#53(192.168.5.100)
;; WHEN: Tue Nov 29 23:08:35 CET 2022
;; MSG SIZE  rcvd: 89
```

22. Next, run the dig command below to check the mail manager for the domain name \ hwdomain.io\. And you should get the output that \ mail.hwdomain.io \ is the mail managed for the main domain \ hwdomain.io.

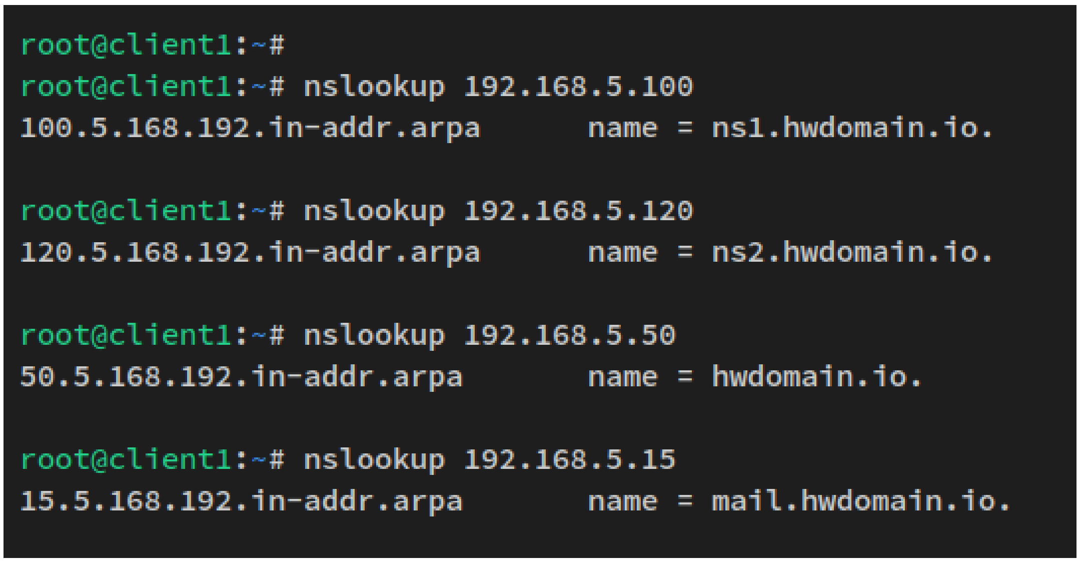


```
root@client1:~#
root@client1:~# dig hwdomain.io MX +short
10 mail.hwdomain.io.
root@client1:~#
root@client1:~# dig hwdomain.io MX

; <<>> DiG 9.16.33-Debian <<>> hwdomain.io MX
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 42170
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: deaeeff16cde494d01000000638682fa14d09941ac275c18 (good)
;; QUESTION SECTION:
;hwdomain.io.                   IN      MX

;; ANSWER SECTION:
hwdomain.io.            604800  IN      MX      10 mail.hwdomain.io.

;; ADDITIONAL SECTION:
mail.hwdomain.io.       604800  IN      A       192.168.5.15

;; Query time: 3 msec
;; SERVER: 192.168.5.100#53(192.168.5.100)
;; WHEN: Tue Nov 29 23:08:58 CET 2022
;; MSG SIZE  rcvd: 105
```

You can check the reverse zone configuration for your domain name using the nslookup command.

Run the nslookup command below to check and verify the reverse DNS for some IP addresses.

Now you should see that the IP address \192.168.5.100\ is reversed on the name server \ns1.hwdomain.io\, the IP address\192.168.5.120\ is reversed on the name server \ns2.hwdomain.io\ and in the IP address \192.168.5.50\ is reversed in the main domain name \hwdomain.io\, and finally the IP address \192.168.5.15\ is reversed in the sub-domain \mail.hwdomain.ior.

23. 

```
root@client1:~#
root@client1:~# nslookup 192.168.5.100
100.5.168.192.in-addr.arpa      name = ns1.hwdomain.io.

root@client1:~# nslookup 192.168.5.120
120.5.168.192.in-addr.arpa      name = ns2.hwdomain.io.

root@client1:~# nslookup 192.168.5.50
50.5.168.192.in-addr.arpa       name = hwdomain.io.

root@client1:~# nslookup 192.168.5.15
15.5.168.192.in-addr.arpa       name = mail.hwdomain.io.
```

At this point, you have finished installing the BIND DNS Server with Master-Slave architecture on Rocky Linux. You have also learned to test the DNS server through various command utilities such as dig and nslookup.

Conclusion

Congratulations! during this tutorial, you learned how to install and configure BIND DNS Server on Rocky Linux 9 servers. You have correctly configured the master-slave DNS BIND server using two different Rocky Linux servers. In addition, you learned the basic command of Dig and Nslookup for checking and checking records and DNS configuration.