Comment installer un résolveur DNS local avec Dnsmasq sur Rocky Linux

Dnsmasq is a small and lightweight DNS server for your local environment. It can be used to provide a DNS Server, DHCP Server, and a TFTP Server. As for the DNS Server, the Dnsmasq can be used as a forwarder, recursive DNS Server, and DNS caching system. It also loads DNS contents from the /etc/hosts file, which allows you to set up domain names for local hostnames.

Dnsmasq is designed to be lightweight with a small footprint, suitable for low-resource devices such as Routers and Firewalls. Dnsmasq has low system requirements and consumes low resources. It can be run on Linux, BSDs, Android, and macOS.

This tutorial will cover installing and setting up a local DNS Server with Dnsmasq on a Rocky Linux 9 server. You'll install Dnsmasq and set up the local DNS Server with some additional features, such as enabling local domain names, setting up sub-domains via /etc/hosts file, and enabling the DNS cache for faster access. Lastly, you'll enable the DHCP server via Dnsmasq.

You'll also learn how to set up client machines to use the local DNS Server as the primary DNS resolver.

Prerequisites

There are several prerequisites that you must have before you get started. Below are the lists of prerequisites:

- A Rocky Linux 9 server - This example uses the Rocky Linux with hostname '***dnsmasq-rocky***' and the IP address '***192.168.5.50***'.
- A non-root user with sudo/root administrator privileges.
- SELinux is running with '***permissive***' mode.

For client machines, you can use any Linux distribution. You can use Debian-based or RHEL-based distributions as for client machines.

Prepare System

On RHEL-based operating systems, the default DNS resolver '*/etc/resolv.conf*' is generated by the NetworkManager service. Before you install Dnsmasq, you'll set up a static DNS resolver via */etc/resolv.conf* file and disable the DNS resolver from the NetworkManager service.

To start, open the NetworkManager config file */etc/NetworkManager/NetworkManager.conf* using the below nano editor command.

```
sudo nano /etc/NetworkManager/NetworkManager.conf
```

Add the line '*dns=none*' within the section '*[section]*'.

```
[main]
dns=none
```

Save the file and exit the editor when you're finished.

Next, open the DNS resolver config file '*/etc/resolv.conf*' using the nano editor command below.

```
sudo nano /etc/resolv.conf
```

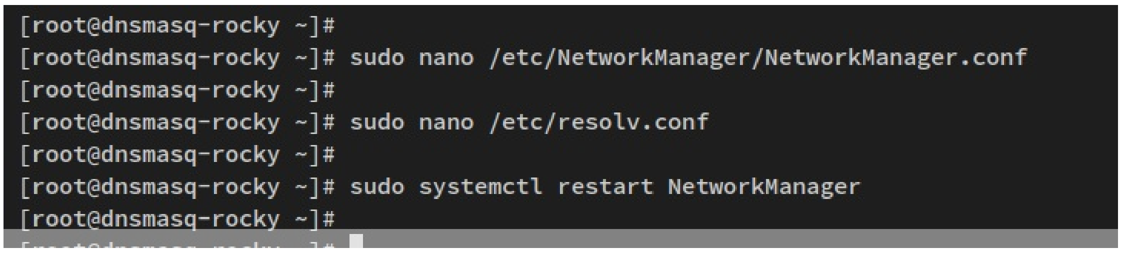
Delete all available lines and replace them with the following lines. With this, you'll be using the Cloudflare and Google public DNS as the main DNS server.

```
nameserver 1.1.1.1
nameserver 8.8.8.8
```

Save the file and exit when you're done.

Lastly, run the below systemctl command to restart the NetworkManager and apply the changes.

```
sudo systemctl restart NetworkManager
```



```
[root@dnsmasq-rocky ~]#
[root@dnsmasq-rocky ~]# sudo nano /etc/NetworkManager/NetworkManager.conf
[root@dnsmasq-rocky ~]#
[root@dnsmasq-rocky ~]# sudo nano /etc/resolv.conf
[root@dnsmasq-rocky ~]#
[root@dnsmasq-rocky ~]# sudo systemctl restart NetworkManager
[root@dnsmasq-rocky ~]#
```

With these settings finished, your DNS resolver will not be changed by the NetworkManager service and you can add or change the DNS resolver at any time.

In the next steps, you'll start the Dnsmasq installation and configuration on Rocky Linux.

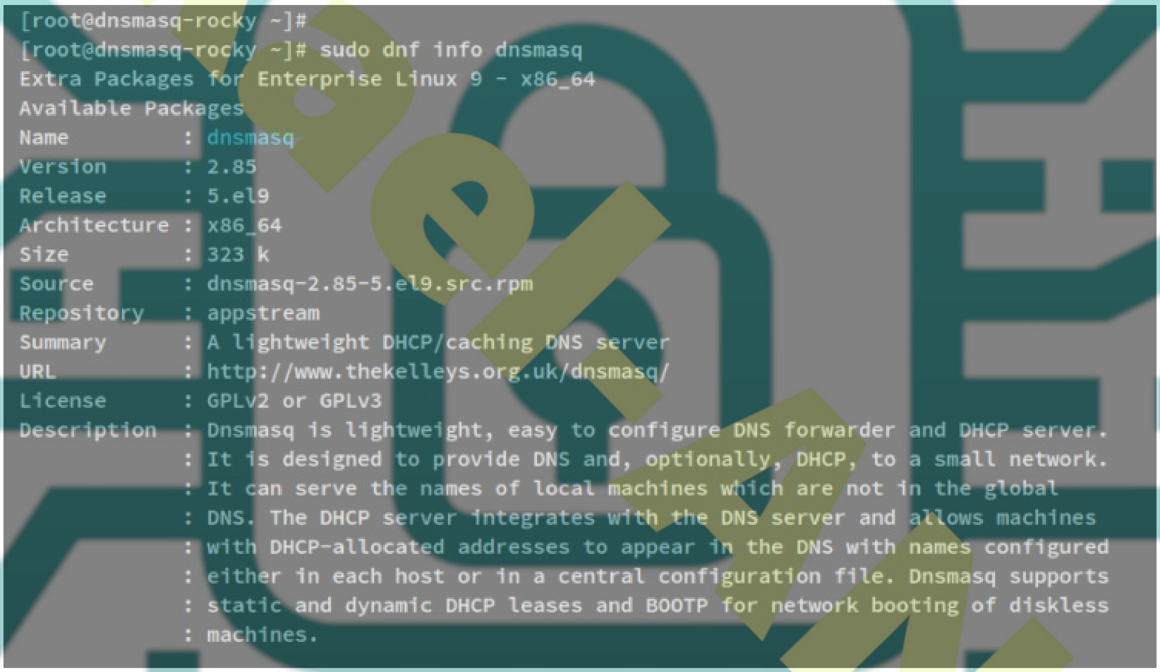
Installing Dnsmasq on Rocky Linux

In this step, you'll install the Dnsmasq package on a Rocky Linux server. Then, you'll start and enable the Dnsmasq service to run upon the system bootup.

By default, the '*dnsmasq*' package is available on the Rocky Linux AppStream repository. Run the below dnf command to get information about the '*dnsmasq*' package.

```
sudo dnf info dnsmasq
```

The Dnsmasq v2.85 is available on Rocky Linux 9 at the time of this writing.



```
[root@dnsmasq-rocky ~]#
[root@dnsmasq-rocky ~]# sudo dnf info dnsmasq
Extra Packages for Enterprise Linux 9 - x86_64
Available Packages
Name         : dnsmasq
Version      : 2.85
Release      : 5.el9
Architecture : x86_64
Size         : 323 k
Source       : dnsmasq-2.85-5.el9.src.rpm
Repository   : appstream
Summary      : A lightweight DHCP/caching DNS server
URL          : http://www.thekelleys.org.uk/dnsmasq/
License      : GPLv2 or GPLv3
Description  : Dnsmasq is lightweight, easy to configure DNS forwarder and DHCP server.
             : It is designed to provide DNS and, optionally, DHCP, to a small network.
             : It can serve the names of local machines which are not in the global
             : DNS. The DHCP server integrates with the DNS server and allows machines
             : with DHCP-allocated addresses to appear in the DNS with names configured
             : either in each host or in a central configuration file. Dnsmasq supports
             : static and dynamic DHCP leases and BOOTP for network booting of diskless
             : machines.
```

Now run the below command to install Dnsmasq. Input y when prompted for the confirmation, then press ENTER to proceed.

```
sudo dnf install dnsmasq dnsmasq-utils
```

The Dnsmasq installation should now be started.

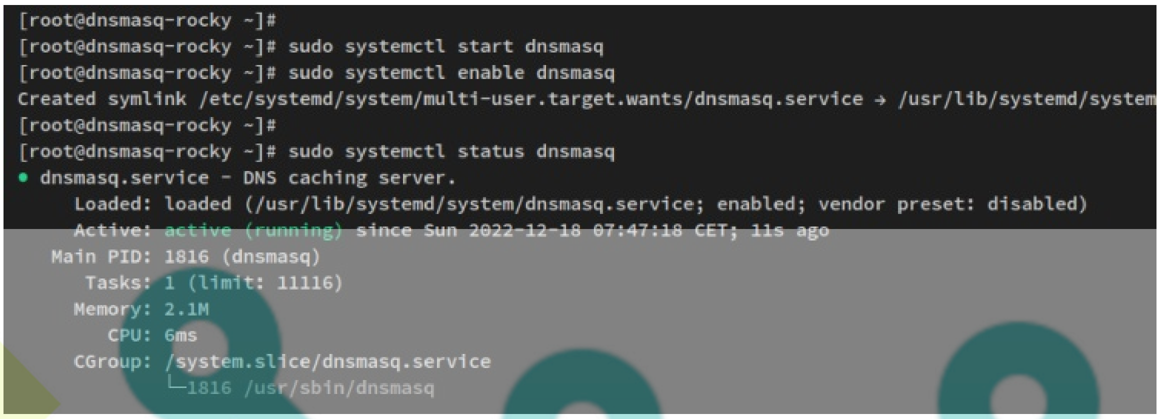
After installing the Dnsmasq, run the below systemctl command to start and enable the '*dnsmasq*' service. With the below command executed, the '*dnsmasq*' service should now be running and enabled, which will start automatically upon the bootup.

```
sudo systemctl start dnsmasq
sudo systemctl enable dnsmasq
```

Verify the '*dnsmasq*' service via the systemctl command utility below.

```
sudo systemctl status dnsmasq
```

You'll receive the output like this - The '*dnsmasq*' service is currently running, enabled, and will be run automatically at bootup.



```
[root@dnsmasq-rocky ~]#
[root@dnsmasq-rocky ~]# sudo systemctl start dnsmasq
[root@dnsmasq-rocky ~]# sudo systemctl enable dnsmasq
Created symlink /etc/systemd/system/multi-user.target.wants/dnsmasq.service → /usr/lib/systemd/system
[root@dnsmasq-rocky ~]#
[root@dnsmasq-rocky ~]# sudo systemctl status dnsmasq
● dnsmasq.service - DNS caching server.
     Loaded: loaded (/usr/lib/systemd/system/dnsmasq.service; enabled; vendor preset: disabled)
     Active: active (running) since Sun 2022-12-18 07:47:18 CET; 11s ago
   Main PID: 1816 (dnsmasq)
      Tasks: 1 (limit: 11116)
     Memory: 2.1M
        CPU: 6ms
     CGroup: /system.slice/dnsmasq.service
             └─1816 /usr/sbin/dnsmasq
```

With the Dnsmasq is installed and running, you'll next start the configuration of Dnsmasq to run as a local DNS Server.

Configuring Dnsmasq

In this step, you'll set up the Dnsmasq as the local DNS Server with some enabled features such as cache DNS, and DHCP server, and configure the domain name and sub-domains for local applications. This allows your application to be accessible via the local domain names/sub-domains such as '***db1.hwdomain.io***', '***app.hwdomain.io***', and many more.

To start, run the below command to copy the default Dnsmasq config file to *'/etc/dnsmasq.conf.orig'*, then open the original Dnsmasq configuration file *'/etc/dnsmasq.conf'* using the below nano editor command.

```
sudo cp /etc/dnsmasq.conf{,.orig}
sudo nano /etc/dnsmasq.conf
```

Add the following lines to the file.

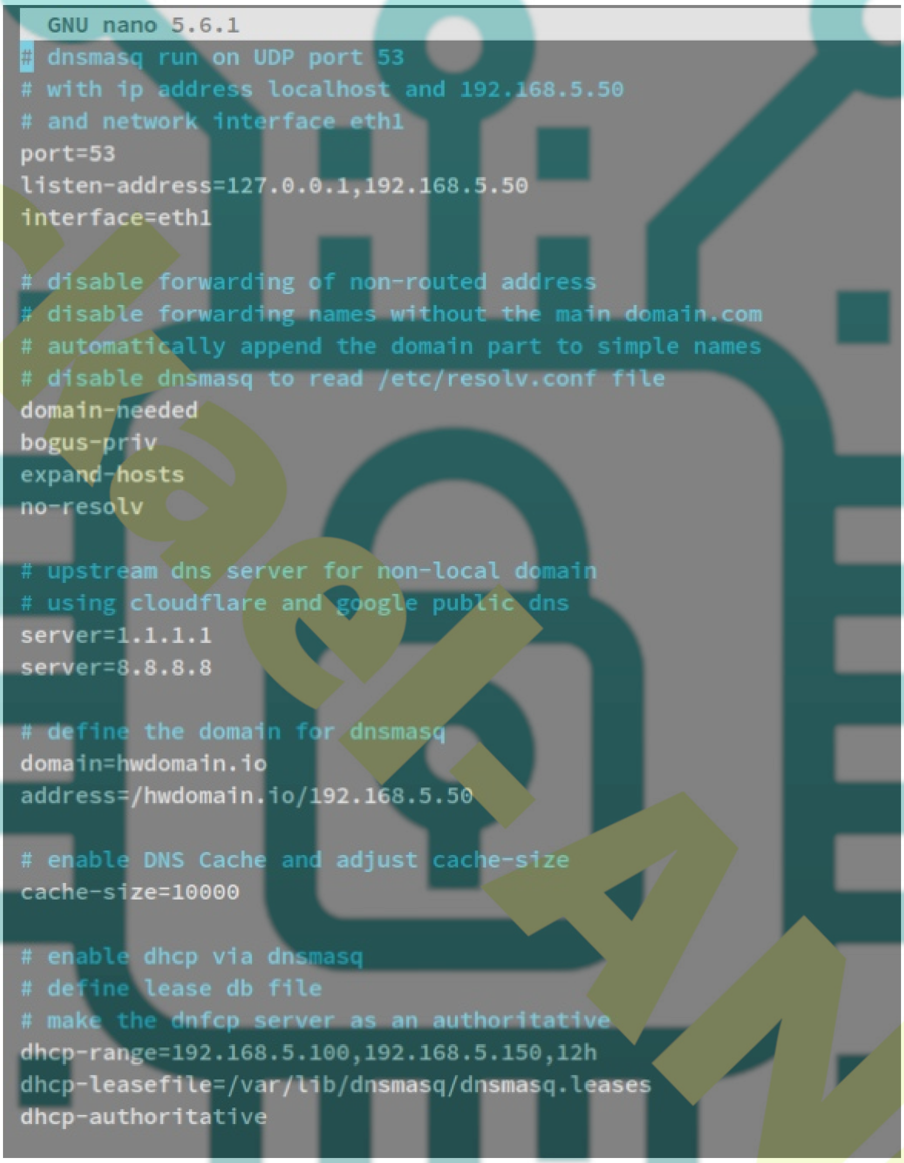
```
# dnsmasq run on UDP port 53
# with IP address localhost and 192.168.5.50
# and network interface eth1
port=53
listen-address=127.0.0.1,192.168.5.50
interface=eth1

# disable forwarding of non-routed address
# disable forwarding names without the main domain.com
# automatically append the domain part to simple names
# disable dnsmasq to read /etc/resolv.conf file
domain-needed
bogus-priv
expand-hosts
no-resolv

# upstream DNS server for non-local domains
# using Cloudflare and google public DNS
server=1.1.1.1
server=8.8.8.8

# define the domain for dnsmasq
domain=hwdomain.io
address=/hwdomain.io/192.168.5.50

# enable DNS Cache and adjust cache-size
cache-size=10000

# enable dhcp via dnsmasq
# define lease db file
# make the dhcp server as an authoritative
dhcp-range=192.168.5.100,192.168.5.150,12h
dhcp-leasefile=/var/lib/dnsmasq/dnsmasq.leases
dhcp-authoritative
```

Save the file and exit the editor when you're finished.

Below is the detailed options that you'll be using for your Dnsmasq installation:

- **port**: which port you will be using to run the Dnsmasq.
- **listen-address**: which IP address you'll be using to run the Dnsmasq. You can use multiple IP addresses.
- **interface**: which interface the Dnsmasq will be bind and running.
- **domain-needed**: disable forwarding names without the main domain address. You can access like 'mysql1' host unless you give the full with local domain such as 'mysql1.hwdomain.io'.
- **bogus-priv**: disable forwarding for non-routed addresses.
- **expand-hosts**: automatically append the local domain part to simple names.
- **no-resolv**: ignore the '/etc/resolv.conf' file on the server.

- **server**: define the upstream DNS Server that you'll be using for non-local addresses or domains. This example uses the Public DNS Server by Cloudflare and Google.
- **domain**: define the domain name for the Dnsmasq server. In this example, the Dnsmasq server will get the local domain hwdomain.io.
- **address**: define which IP address for the domain name on Dnsmasq. In this example, the domain hwdomain.io will be resolved to the IP address 192.168.5.50.
- **cache-size**: enabled DNS cache on Dnsmasq. Be sure to adjust the size, which increases the performance and speed.
- **dhcp-range**: enable the DHCP server via the Dnsmasq. Adjust the IP address pool for your network and lease time.
- **dhcp-leasefile**: define the file that will be sued to store the DHCP lease.
- **dhcp-authoritative**: make the DHCP server as authoritative.



```
  GNU nano 5.6.1
# dnsmasq run on UDP port 53
# with ip address localhost and 192.168.5.50
# and network interface eth1
port=53
listen-address=127.0.0.1,192.168.5.50
interface=eth1

# disable forwarding of non-routed address
# disable forwarding names without the main domain.com
# automatically append the domain part to simple names
# disable dnsmasq to read /etc/resolv.conf file
domain-needed
bogus-priv
expand-hosts
no-resolv

# upstream dns server for non-local domain
# using cloudflare and google public dns
server=1.1.1.1
server=8.8.8.8

# define the domain for dnsmasq
domain=hwdomain.io
address=/hwdomain.io/192.168.5.50

# enable DNS Cache and adjust cache-size
cache-size=10000

# enable dhcp via dnsmasq
# define lease db file
# make the dnfcp server as an authoritative
dhcp-range=192.168.5.100,192.168.5.150,12h
dhcp-leasefile=/var/lib/dnsmasq/dnsmasq.leases
dhcp-authoritative
```

Next, open the '*/etc/hosts*' file using the below nano editor command. You'll now define some sub-domains for applications in your local environment.

```
sudo nano /etc/hosts
```

Add the following lines to the file. In this example, you'll create three sub-domains ***wiki***, ***mysql***, and ***files***. Each sub-domain will follow the main domain of the Dnsmasq server 'hwdomain.io' and point to a specific IP address.

The subdomain ***wiki.hwdomain.io*** will be pointed to IP address '***192.168.5.10***', the sub-domain '***mysql.hwdomain.io***' is pointed to IP address '***192.168.5.25***', and the '***files.hwdomain.io***' will be pointed to IP address '***192.168.5.30***'.

```
192.168.5.10 wiki
192.168.5.25 mysql
192.168.5.30 files
```

Save the file and exit the editor when you're finished.

Now open the DNS resolver config file '*/etc/resolv.conf*' using the below nano editor command.

```
sudo nano /etc/resolv.conf
```

Add the following lines to the top of the file. Be sure to change the IP address with the Dnsmasq server IP address.

```
nameserver 127.0.0.1
nameserver 192.168.5.50
```

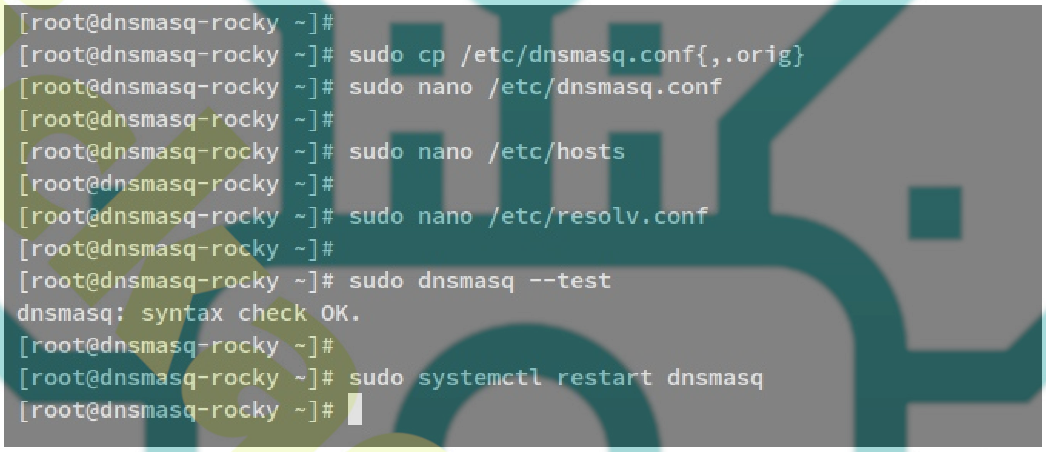
Save the file and exit the editor when finished.

Now run the below command to verify the Dnsmasq configuration and ensure that you have the proper configuration. You'll receive the output such as '*dnsmasq: syntax check OK*'.

```
sudo dnsmasq --test
```

Lastly, run the below systemctl command utility to restart the 'dnsmasq' service and apply the changes.

```
sudo systemctl restart dnsmasq
```



```
[root@dnsmasq-rocky ~]#
[root@dnsmasq-rocky ~]# sudo cp /etc/dnsmasq.conf{,.orig}
[root@dnsmasq-rocky ~]# sudo nano /etc/dnsmasq.conf
[root@dnsmasq-rocky ~]#
[root@dnsmasq-rocky ~]# sudo nano /etc/hosts
[root@dnsmasq-rocky ~]#
[root@dnsmasq-rocky ~]# sudo nano /etc/resolv.conf
[root@dnsmasq-rocky ~]#
[root@dnsmasq-rocky ~]# sudo dnsmasq --test
dnsmasq: syntax check OK.
[root@dnsmasq-rocky ~]#
[root@dnsmasq-rocky ~]# sudo systemctl restart dnsmasq
[root@dnsmasq-rocky ~]#
```

At this point, you've finished the configuration of Dnsmasq as the local DNS Server on the Rocky Linux system. You've also configured the domain name for the Dnsmasq server and some sub-domains via the '/etc/hosts' file. Lastly, you've enabled the cache DNS and DHCP server via the Dnsmasq.

In the next steps, you'll verify the Dnsmsq server installation and configuration.

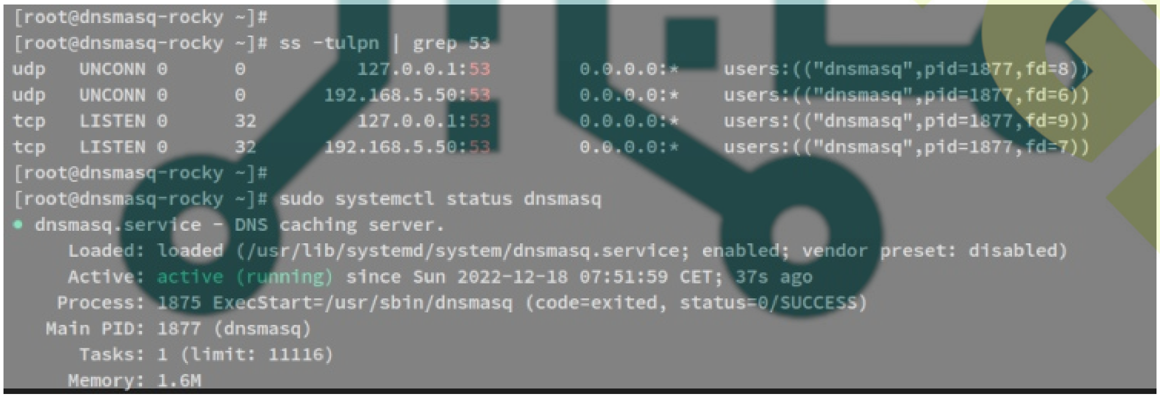
Verify Dnsmasq Installation

With the Dnsmasq configuration is finished, you'll now verify the Dnsmasq service itself. You'll verify the Dnsmasq to ensure it's running on the default port 50 and the service is running. Then, you'll verify the local domain name and sub-domains you created via the '*dns-utils*' package.

Run the below command to verify the open port on your system. Then verify the '*dnsmasq*' service via the systemctl command utility.

```
ss -tulpn | grep 53
sudo systemctl status dnsmasq
```

You'll receive the output like this - The Dnsmasq is running on the default port 53 and the service status is currently running. Also, it's enabled, which will start automatically upon the bootup.

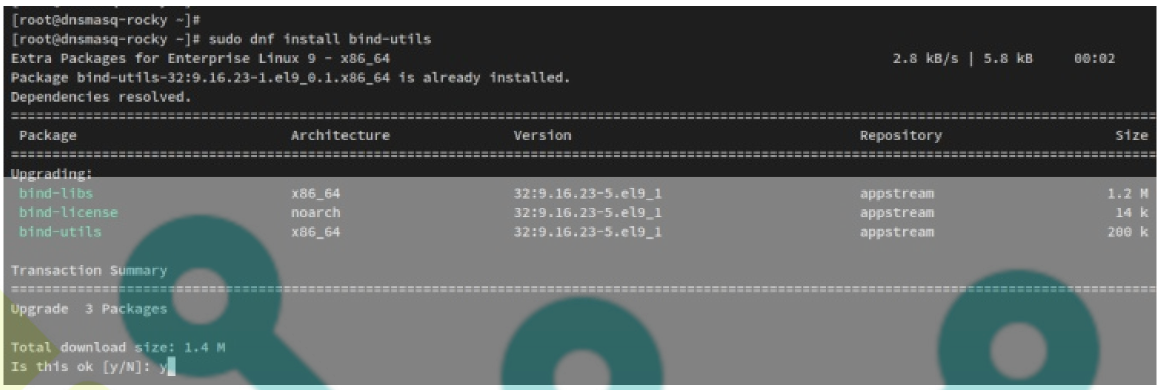


```
[root@dnsmasq-rocky ~]#
[root@dnsmasq-rocky ~]# ss -tulpn | grep 53
udp   UNCONN 0      0          127.0.0.1:53       0.0.0.0:*    users:(("dnsmasq",pid=1877,fd=8))
udp   UNCONN 0      0       192.168.5.50:53       0.0.0.0:*    users:(("dnsmasq",pid=1877,fd=6))
tcp   LISTEN 0      32         127.0.0.1:53       0.0.0.0:*    users:(("dnsmasq",pid=1877,fd=9))
tcp   LISTEN 0      32      192.168.5.50:53       0.0.0.0:*    users:(("dnsmasq",pid=1877,fd=7))
[root@dnsmasq-rocky ~]#
[root@dnsmasq-rocky ~]# sudo systemctl status dnsmasq
● dnsmasq.service - DNS caching server.
     Loaded: loaded (/usr/lib/systemd/system/dnsmasq.service; enabled; vendor preset: disabled)
     Active: active (running) since Sun 2022-12-18 07:51:59 CET; 37s ago
    Process: 1875 ExecStart=/usr/sbin/dnsmasq (code=exited, status=0/SUCCESS)
   Main PID: 1877 (dnsmasq)
      Tasks: 1 (limit: 11116)
     Memory: 1.6M
```

Next, run the below dnf command to install the 'bind-utils' package to your Rocky Linux server. This package provides multiple command-line tools for testing and troubleshooting DNS Server.

```
sudo dnf install bind-utils
```

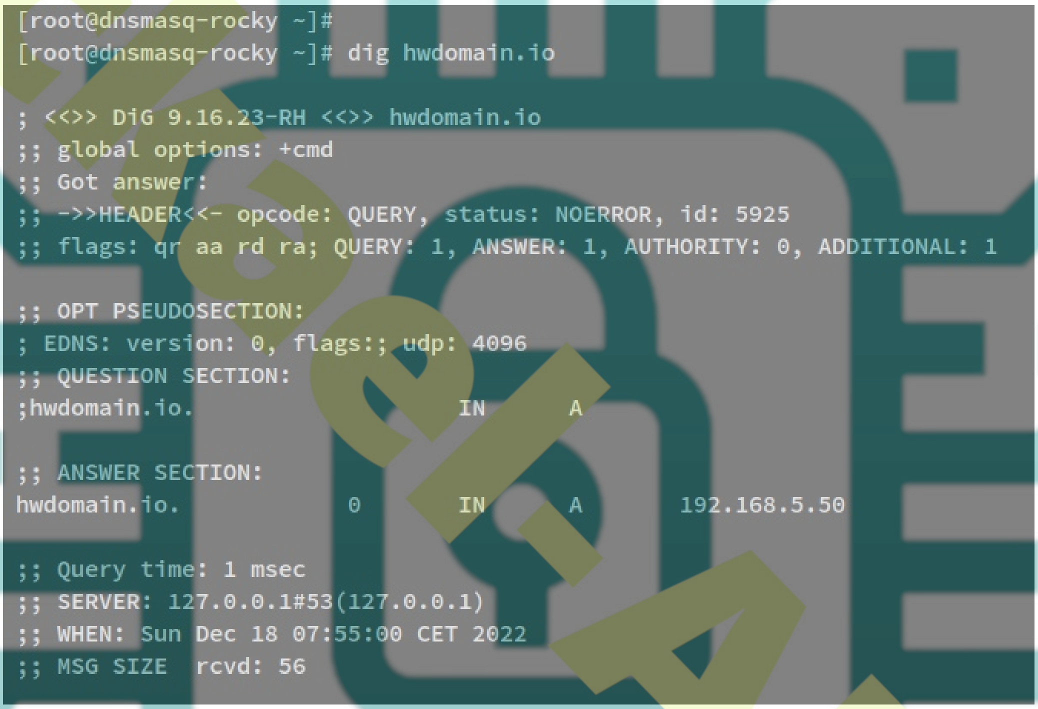
Input y when prompted and press ENTER to proceed.



```
[root@dnsmasq-rocky ~]#
[root@dnsmasq-rocky ~]# sudo dnf install bind-utils
Extra Packages for Enterprise Linux 9 - x86_64                                  2.8 kB/s | 5.8 kB     00:02
Package bind-utils-32:9.16.23-1.el9_0.1.x86_64 is already installed.
Dependencies resolved.
==========================================================================================================
 Package              Architecture          Version                          Repository              Size
==========================================================================================================
Upgrading:
 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

Transaction Summary
==========================================================================================================
Upgrade  3 Packages

Total download size: 1.4 M
Is this ok [y/N]: y
```

Now run the below dig command to verify the domain name for the Dnsmasq server '***hwdomain.io***'. You should see that the '***hwdomain.io***' is pointed to the server IP address '***192.168.5.50***'.

```
dig hwdomain.io
```

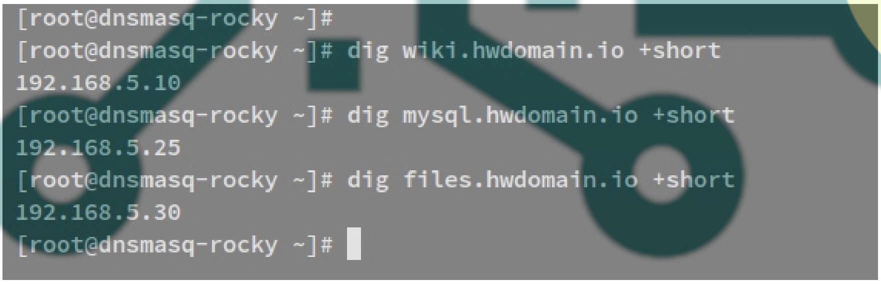


```
[root@dnsmasq-rocky ~]#
[root@dnsmasq-rocky ~]# dig hwdomain.io

; <<>> DiG 9.16.23-RH <<>> hwdomain.io
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 5925
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;hwdomain.io.                   IN      A

;; ANSWER SECTION:
hwdomain.io.            0       IN      A       192.168.5.50

;; Query time: 1 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sun Dec 18 07:55:00 CET 2022
;; MSG SIZE  rcvd: 56
```

Lastly, run the below command to verify the sub-domains that you've defined via the '*/etc/hosts*' file.

```
dig wiki.hwdomain.io +short
dig mysql.hwdomain.io +short
dig files.hwdomain.io +short
```

You'll receive the output similar to this - The sub-domain ***wiki.hwdomain.io*** is pointed to the IP address ***192.168.5.10***, the sub-domain ***mysql.hwdomain.io*** is pointed to the IP address ***192.168.5.25***, and lastly the sub-domain ***files.hwdomain.io*** is pointed to the server IP address ***192.168.5.30***.



```
[root@dnsmasq-rocky ~]#
[root@dnsmasq-rocky ~]# dig wiki.hwdomain.io +short
192.168.5.10
[root@dnsmasq-rocky ~]# dig mysql.hwdomain.io +short
192.168.5.25
[root@dnsmasq-rocky ~]# dig files.hwdomain.io +short
192.168.5.30
[root@dnsmasq-rocky ~]#
```

With these results in place, you've finished the configuration of Dnsmasq as the local DNS Server. In the next steps, you'll secure the DNS port via the Firewalld.

Setting up Firewalld

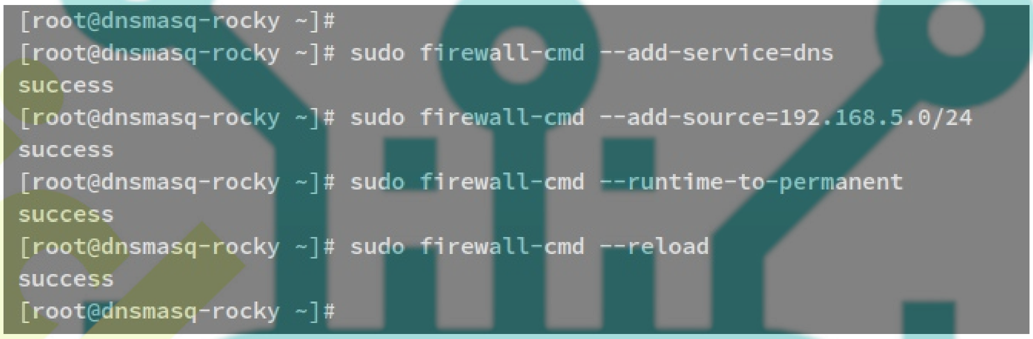
In this step, you'll set up the firewalld to open the DNS service port and add the internal networks IP addresses to the firewalld as the main source that allowed to access the DNS service port.

Run the below firewall-cmd command to add the DNS service to the firewalld. Then, add the internal network IP addresses as the source.

```
sudo firewall-cmd --add-service=dns
sudo firewall-cmd --add-source=192.168.5.0/24
```

Next, run the below command to save the temporary rules that you have created and reload the firewalld to apply the changes.

```
sudo firewall-cmd --runtime-to-permanent
sudo firewall-cmd --reload
```

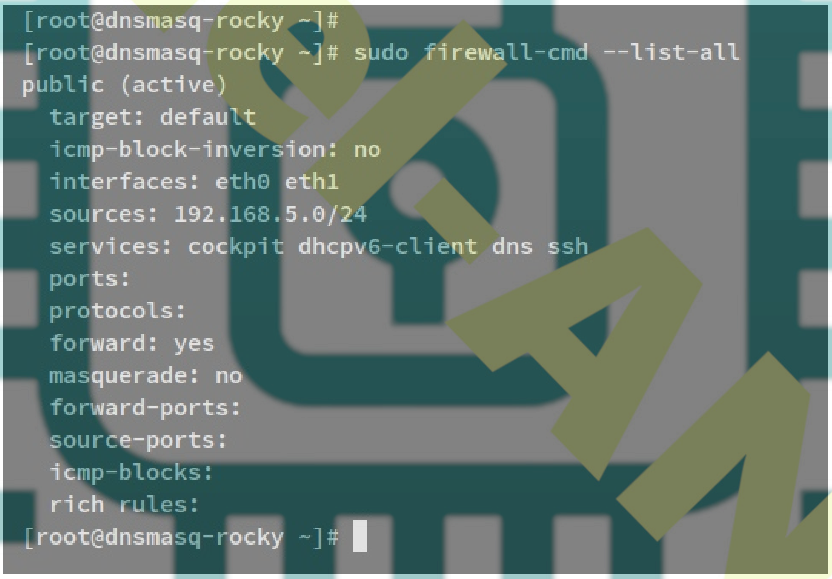


```
[root@dnsmasq-rocky ~]#
[root@dnsmasq-rocky ~]# sudo firewall-cmd --add-service=dns
success
[root@dnsmasq-rocky ~]# sudo firewall-cmd --add-source=192.168.5.0/24
success
[root@dnsmasq-rocky ~]# sudo firewall-cmd --runtime-to-permanent
success
[root@dnsmasq-rocky ~]# sudo firewall-cmd --reload
success
[root@dnsmasq-rocky ~]#
```

Verify the firewalld via the following command. You should see that the DNS service is added to the firewalld and the source IP address of the network is also added to the firewalld.

```
sudo firewall-cmd --list-all
```

Output:



```
[root@dnsmasq-rocky ~]#
[root@dnsmasq-rocky ~]# sudo firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: eth0 eth1
  sources: 192.168.5.0/24
  services: cockpit dhcpv6-client dns ssh
  ports:
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
[root@dnsmasq-rocky ~]#
```

Setting up Client (Debian-based or RHEL-based)

In this step, you'll learn how to set up both Debian-based and RHEL-based distributions to use the local DNS Server that you've created via Dnsmasq.

For RHEL-Based Distributions

Add a new config file for NetworkManager '*/etc/NetworkManager/conf.d/dns-servers.conf*' using the below nano editor command.

```
sudo nano /etc/NetworkManager/conf.d/dns-servers.conf
```

Add the following lines to the file, and ensure to change the IP address with the Dnsmasq service IP address. With this configuration, you'll set up the default DNS resolver for the client by using the Dnsmasq server IP address. This will automatically write the DNS resolver configuration '*/etc/resolv.conf*'.

```
[global-dns-domain-*]
servers=192.168.5.50
```

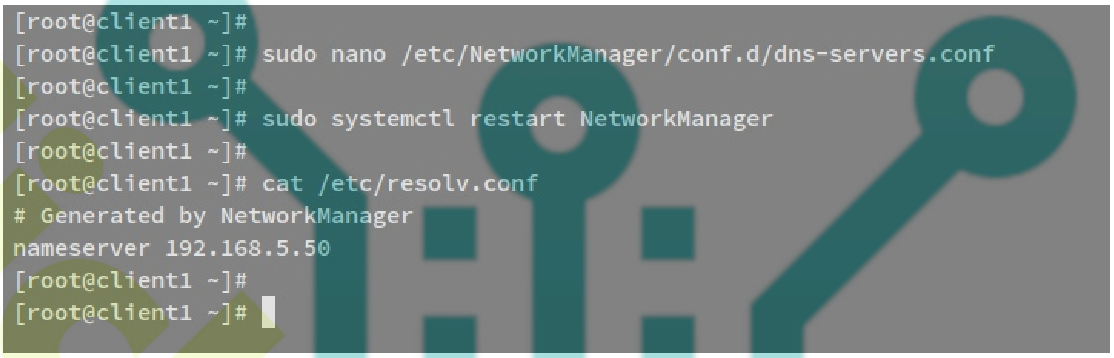
Save the file and exit the editor when finished.

Next, run the below systemctl command to restart the NetworkManager service and apply the changes.

```
sudo systemctl restart NetworkManager
```

You can show the '/etc/resolv.conf' file to verify the settings. You should see the default name server is the local Dnsmasq server IP address *192.168.5.50*.

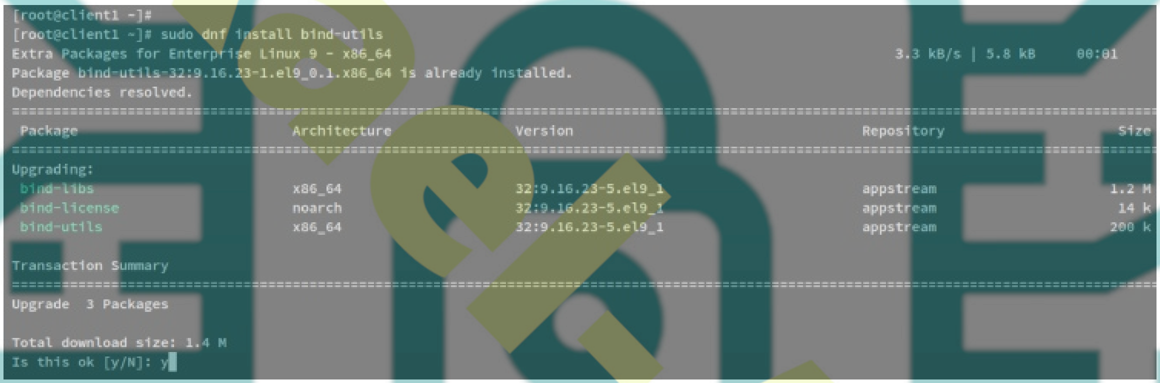
```
cat /etc/resolv.conf
```



```
[root@client1 ~]#
[root@client1 ~]# sudo nano /etc/NetworkManager/conf.d/dns-servers.conf
[root@client1 ~]#
[root@client1 ~]# sudo systemctl restart NetworkManager
[root@client1 ~]#
[root@client1 ~]# cat /etc/resolv.conf
# Generated by NetworkManager
nameserver 192.168.5.50
[root@client1 ~]#
[root@client1 ~]#
```

Next, run the below dnf command to install the ***'bind-utils'*** package to your system.

```
sudo dnf install bind-utils
```



```
[root@client1 ~]#
[root@client1 ~]# sudo dnf install bind-utils
Extra Packages for Enterprise Linux 9 - x86_64                                        3.3 kB/s | 5.8 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
==========================================================================================================================
Upgrading:
 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

Transaction Summary
==========================================================================================================================
Upgrade  3 Packages

Total download size: 1.4 M
Is this ok [y/N]: y
```

For Debian-Based Distributions

If you're using the Debian-based operating system, you can set up the DNS resolver manually and disable the 'systemd-resolved' service on your system - especially for the Ubuntu system.

Run the below command to stop and disable the systemd-resolved service.

```
sudo systemctl disable --now systemd-resolved
```

Now run the below command to remove the symlink file of the DNS resolver configuration *'/etc/resolv.conf'*. Then, create a new resolver config file '*/etc/resolv.conf*' via the nano editor command below.

```
unlink /etc/resolv.conf
sudo nano /etc/resolv.conf
```

Add the following line to the file and change the IP address with your Dnsmasq server IP address.

```
nameserver 192.168.5.50
```

Save the file and exit the editor when finished.

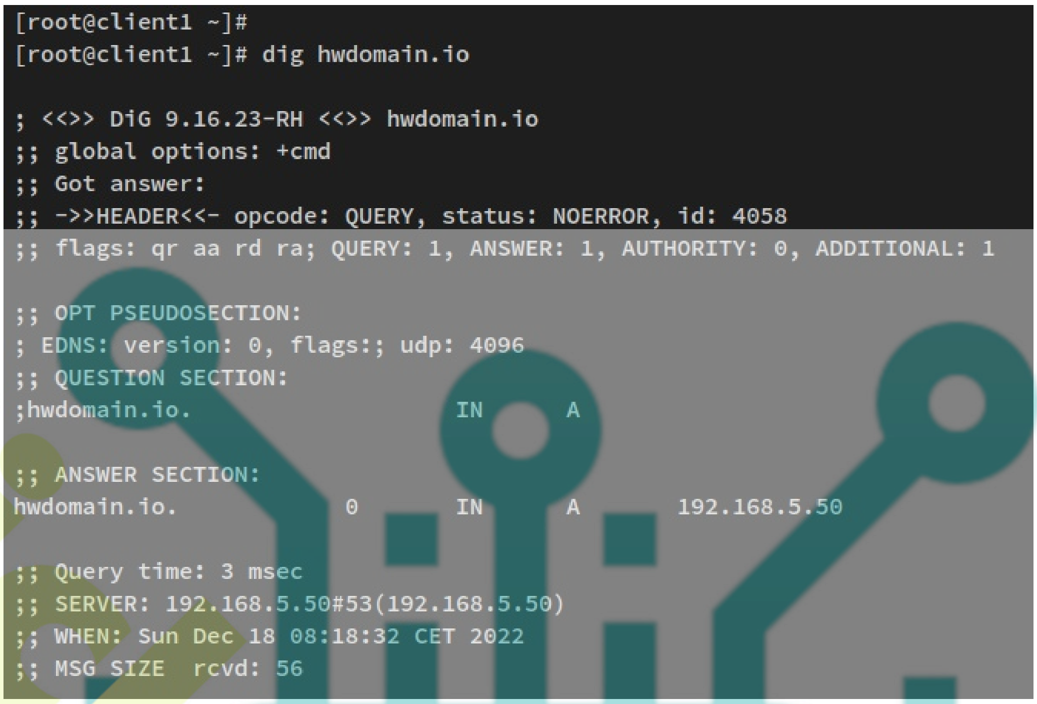
Next, run the below apt command to install the '*dnsutils*' package to your system.

```
sudo apt install dnsutils
```

Once the *'bind-utils'* or *'dns-utils'* is installed, you can verify your Dnsmasq's configuration via the dig command.

Verify the domain name of the Dnsmasq server '***hwdomain.io***' via the dig command below. You should see that the domain '***hwdomain.io***' is pointed to the Dnsmasq server IP address ***192.168.5.50***.

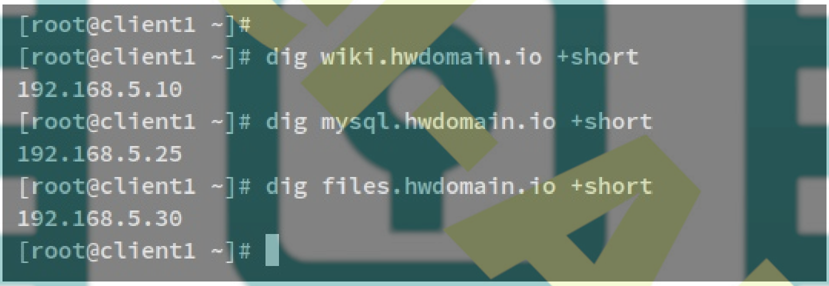
```
dig hwdomain.io
```


```
[root@client1 ~]#
[root@client1 ~]# dig hwdomain.io

; <<>> DiG 9.16.23-RH <<>> hwdomain.io
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4058
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;hwdomain.io.                   IN      A

;; ANSWER SECTION:
hwdomain.io.            0       IN      A       192.168.5.50

;; Query time: 3 msec
;; SERVER: 192.168.5.50#53(192.168.5.50)
;; WHEN: Sun Dec 18 08:18:32 CET 2022
;; MSG SIZE  rcvd: 56
```

Next, verify the sub-domains that you've configured via the *'/etc/hosts'* file using the below command. You should see that each sub-domain is pointed to the specific IP address that you've configured on the *'/etc/hosts'* file.

```
dig wiki.hwdomain.io +short
dig mysql.hwdomain.io +short
dig files.hwdomain.io +short
```

You'll receive the output like this - The sub-domain ***wiki.hwdomain.io*** is pointed to the IP address ***192.168.5.10***, the sub-domain ***mysql.hwdomain.io*** is pointed to the IP address ***192.168.5.25***, and lastly, the sub-domain ***files.hwdomain.io*** is pointed to the server IP address ***192.168.5.30***.

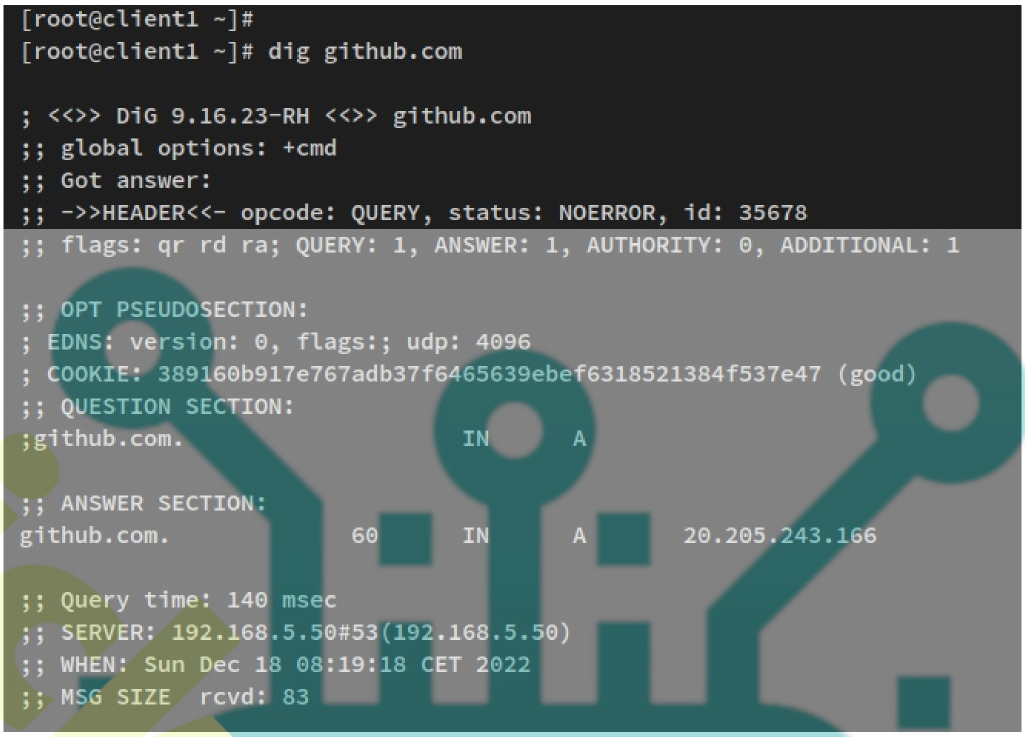

```
[root@client1 ~]#
[root@client1 ~]# dig wiki.hwdomain.io +short
192.168.5.10
[root@client1 ~]# dig mysql.hwdomain.io +short
192.168.5.25
[root@client1 ~]# dig files.hwdomain.io +short
192.168.5.30
[root@client1 ~]#
```

With this, the local domain name and sub-domains is configured successfully. Now, how about public domain names such as Github.com, etc?

Run the below command to verify internet domain names from your client machine. This will ensure that you can connect to the internet, even with the local DNS resolved on the Dnsmasq server.

```
dig github.com
```

You'll receive the output similar to this - The dig query to github.com is connected via the local DNS Server that runs on the IP address **192.168.5.50** with the default port **53**.

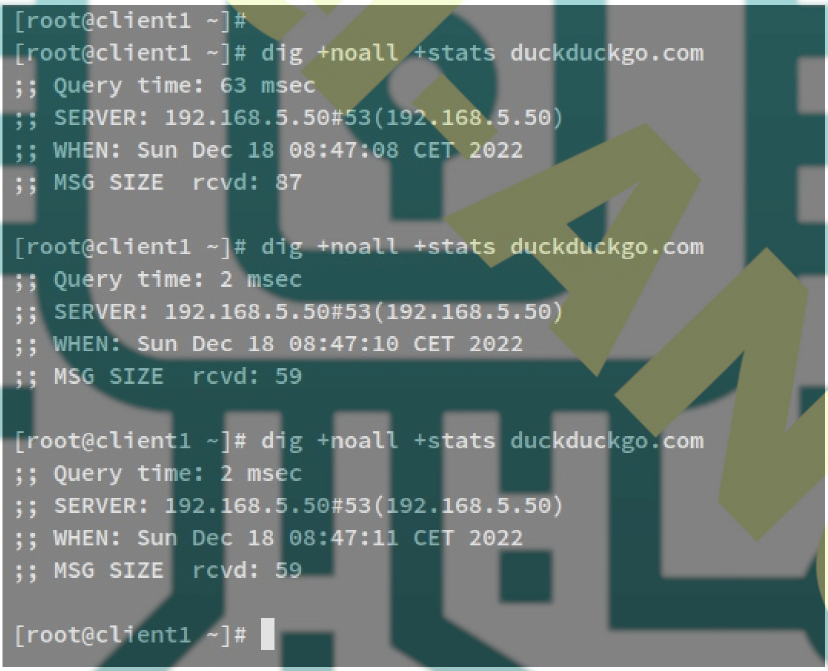


```
[root@client1 ~]#
[root@client1 ~]# dig github.com

; <<>> DiG 9.16.23-RH <<>> github.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 35678
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 389160b917e767adb37f6465639ebef6318521384f537e47 (good)
;; QUESTION SECTION:
;github.com.                    IN      A

;; ANSWER SECTION:
github.com.             60      IN      A       20.205.243.166

;; Query time: 140 msec
;; SERVER: 192.168.5.50#53(192.168.5.50)
;; WHEN: Sun Dec 18 08:19:18 CET 2022
;; MSG SIZE  rcvd: 83
```

Lastly, verify the DNS cache settings via the dig command below. This will show you the executed query's stats, including the '**Query time**' to the target domain name.

```
dig +noall +stats duckduckgo.com
dig +noall +stats duckduckgo.com
```

You'll then receive the output similar to this - The first query you get is the '***Query time***' in 63ms. But for the second and third queries, the Query Time is 2ms, which means your queries to the same domain name is cached via the Dnsmasq local DNS Server.



```
[root@client1 ~]#
[root@client1 ~]# dig +noall +stats duckduckgo.com
;; Query time: 63 msec
;; SERVER: 192.168.5.50#53(192.168.5.50)
;; WHEN: Sun Dec 18 08:47:08 CET 2022
;; MSG SIZE  rcvd: 87

[root@client1 ~]# dig +noall +stats duckduckgo.com
;; Query time: 2 msec
;; SERVER: 192.168.5.50#53(192.168.5.50)
;; WHEN: Sun Dec 18 08:47:10 CET 2022
;; MSG SIZE  rcvd: 59

[root@client1 ~]# dig +noall +stats duckduckgo.com
;; Query time: 2 msec
;; SERVER: 192.168.5.50#53(192.168.5.50)
;; WHEN: Sun Dec 18 08:47:11 CET 2022
;; MSG SIZE  rcvd: 59

[root@client1 ~]#
```

Conclusion

In this tutorial, you have created your own local DNS Server with Dnsmasq. You've set up your own DNS Server for your local environment with the Dnmasq on a Rocky Linux 9 server. Also, this included the configuration of Dnsmasq with local domain names and sub-domains, enabled the DNS cache to get faster, and also enabled the DHCP Server via the Dnsmasq.

Lastly, you've also added and configured client machines (Debian-based and RHEL-based distributions) to use the local DNS Server that you've created. Within this, you've also learned how to troubleshoot the DNS Server with the dig command and how to set up a DNS resolver on a Linux system.