Set Up Your Own DNS Resolver on Debian 10 Buster with BIND9

Last Updated: March 20th, 2021 Xiao Guoan (Admin) 19 Comments
Debian

This tutorial will be showing you how to set up a local DNS resolver on Debian 10 Buster, with the widely-used BIND9 DNS software. There are many synonyms for DNS resolver, some of which are listed below. They all refer to the same thing.

- full resolver (in contrast to stub resolver)
- DNS recursor
- recursive DNS server
- recursive resolver

Also be aware that A DNS server can also called a name server. Examples of DNS resolver are 8.8.8.8 (Google public DNS server) and 1.1.1.1 (Cloudflare public DNS server). The OS on your PC also has a resolver, although it's called stub resolver due to its limited capability. A stub resolver is a small DNS client on the end-user's computer that receives DNS requests from applications such as Firefox and forwards requests to a recursive resolver. Almost every resolver can cache DNS response to improve performance, so they are also called caching DNS server.

Why Run Your Own DNS Resolver

Normally, your computer or router uses your ISP's DNS resolver to query domain names in order to get an IP address. Running your own local DNS resolver can speed up DNS lookups, because

1. The local DNS resolver only listens to your DNS requests and does not answer other people's DNS requests, so you have a much higher chance of getting DNS responese directly from the cache on the resolver.
2. The network latency between your computer and DNS resolver is eliminated (almost zero), so DNS queries can be sent to root DNS servers more quickly.

If you run a mail server and use DNS blacklists (DNSBL) to block spam, then you are advised to run a local DNS resolver to speed up DNS lookups. If you run your own VPN server on a VPS (Virtual Private Server), it's also a good practice to install a DNS resolver on the same VPS.

You may also want to run your own DNS resolver if you don't like your Internet browsing history being stored on a third-party server.

If you own a website and want your own DNS server to handle name resolution for your domain name instead of using your domain registrar's DNS server, then you will need to set up an authoritative DNS server, which is different than a DNS resolver. BIND can act as an authoritative DNS server and a DNS resolver at the same time, but **it's a good practice to separate the two roles on different boxes**. This tutorial shows how to set up a local DNS resolver and because it will be used on local host/local network, no encryption (DNS over TLS or DNS over HTTPS) is needed. Setting up a DoT or DoH server will be discussed in a future article.

Please note that you need to have root privilege when installing software on Debian. You can add **sudo** at the beginning of a command, or use `su -` command to switch to root user.

Install BIND9 on Debian 10 Buster

BIND (Berkeley Internet Name Domain) is an open-source DNS server software widely used on Unix/Linux due to it's stability and high quality. It's originally developed by UC Berkeley, and later in 1994 its development was moved to Internet Systems Consortium, Inc (ISC).

Run the following command to install BIND 9 on Debian 10 Buster from the default repository. BIND 9 is the current version and BIND 10 is a dead project.

```
sudo apt update
sudo apt install bind9 bind9utils bind9-doc
bind9-host dnsutils
```

Check version information.

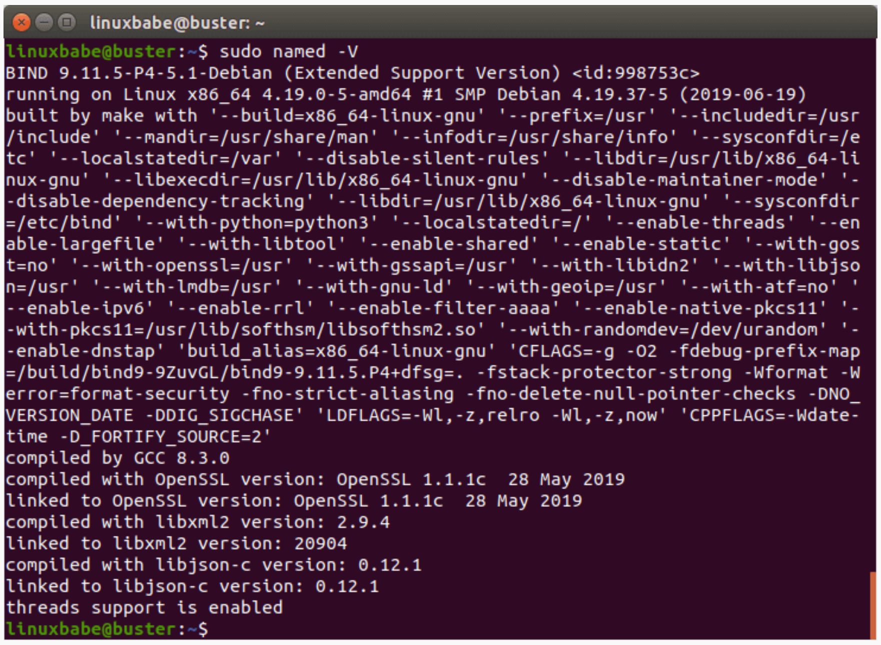
```
sudo named -v
```

Sample output:

```
BIND 9.11.5-P4-5.1-Debian (Extended Support
Version) <id:998753c>
```

To check the version number and build options, run

```
sudo named -V
```


```
linuxbabe@buster:~$ sudo named -V
BIND 9.11.5-P4-5.1-Debian (Extended Support Version) <id:998753c>
running on Linux x86_64 4.19.0-5-amd64 #1 SMP Debian 4.19.37-5 (2019-06-19)
built by make with '--build=x86_64-linux-gnu' '--prefix=/usr' '--includedir=/usr
/include' '--mandir=/usr/share/man' '--infodir=/usr/share/info' '--sysconfdir=/e
tc' '--localstatedir=/var' '--disable-silent-rules' '--libdir=/usr/lib/x86_64-li
nux-gnu' '--libexecdir=/usr/lib/x86_64-linux-gnu' '--disable-maintainer-mode' '-
-disable-dependency-tracking' '--libdir=/usr/lib/x86_64-linux-gnu' '--sysconfdir
=/etc/bind' '--with-python=python3' '--localstatedir=/' '--enable-threads' '--en
able-largefile' '--with-libtool' '--enable-shared' '--enable-static' '--with-gos
t=no' '--with-openssl=/usr' '--with-gssapi=/usr' '--with-libidn2' '--with-libjso
n=/usr' '--with-lmdb=/usr' '--with-gnu-ld' '--with-geoip=/usr' '--with-atf=no' '
--enable-ipv6' '--enable-rrl' '--enable-filter-aaaa' '--enable-native-pkcs11' '-
-with-pkcs11=/usr/lib/softhsm/libsofthsm2.so' '--with-randomdev=/dev/urandom' '-
-enable-dnstap' 'build_alias=x86_64-linux-gnu' 'CFLAGS=-g -O2 -fdebug-prefix-map
=/build/bind9-9ZuvGL/bind9-9.11.5.P4+dfsg=. -fstack-protector-strong -Wformat -W
error=format-security -fno-strict-aliasing -fno-delete-null-pointer-checks -DNO_
VERSION_DATE -DDIG_SIGCHASE' 'LDFLAGS=-Wl,-z,relro -Wl,-z,now' 'CPPFLAGS=-Wdate-
time -D_FORTIFY_SOURCE=2'
compiled by GCC 8.3.0
compiled with OpenSSL version: OpenSSL 1.1.1c  28 May 2019
linked to OpenSSL version: OpenSSL 1.1.1c  28 May 2019
compiled with libxml2 version: 2.9.4
linked to libxml2 version: 20904
compiled with libjson-c version: 0.12.1
linked to libjson-c version: 0.12.1
threads support is enabled
linuxbabe@buster:~$
```

By default, BIND automatically starts after installation. You can check its status with:

```
systemctl status bind9
```

Hint: If the above command doesn't quit immediately, press Q.

If it's not running, then start it with:

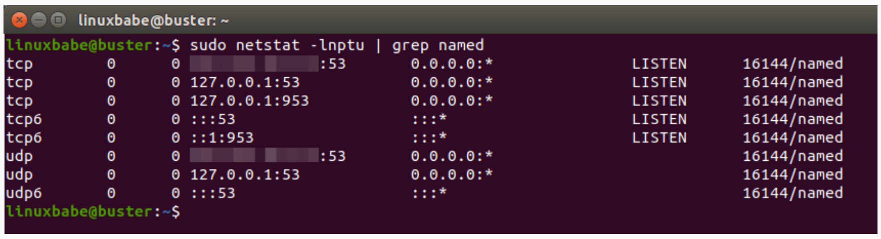
```
sudo systemctl start bind9
```

And enable auto start at boot time:

```
sudo systemctl enable bind9
```

The BIND server will run as the `bind` user, which is created during installation, and listens on TCP and UDP port 53, as can be seen by running the following command:

```
sudo netstat -lnptu | grep named
```



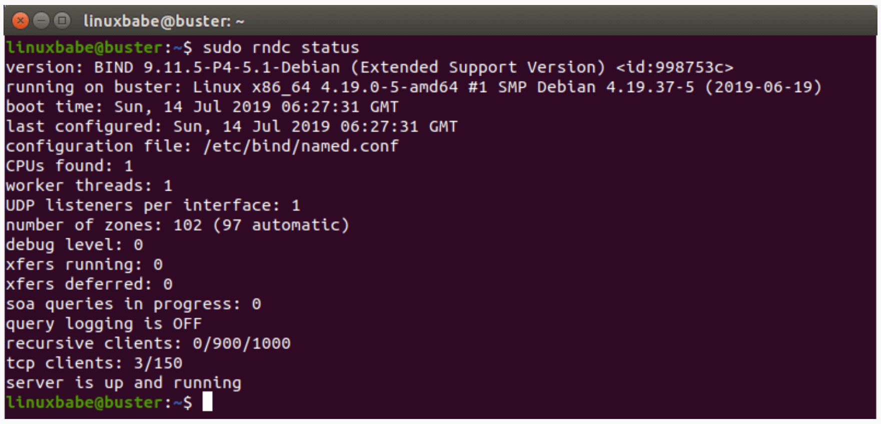
```
linuxbabe@buster:~$ sudo netstat -lnptu | grep named
tcp        0      0 [redacted]:53           0.0.0.0:*               LISTEN      16144/named
tcp        0      0 127.0.0.1:53            0.0.0.0:*               LISTEN      16144/named
tcp        0      0 127.0.0.1:953           0.0.0.0:*               LISTEN      16144/named
tcp6       0      0 :::53                   :::*                    LISTEN      16144/named
tcp6       0      0 ::1:953                 :::*                    LISTEN      16144/named
udp        0      0 [redacted]:53           0.0.0.0:*                           16144/named
udp        0      0 127.0.0.1:53            0.0.0.0:*                           16144/named
udp6       0      0 :::53                   :::*                                16144/named
linuxbabe@buster:~$
```

Usually DNS queries are sent to the UDP port 53. The TCP port 53 is for responses size larger than 512 bytes.

The BIND daemon is called **named**. (A daemon is a piece of software that runs in the background.) The `named` binary is installed by the `bind9` package and there's another important binary: `rndc`, the remote name daemon controller, which is installed by the `bind9utils` package. The `rndc` binary is used to reload/stop and control other aspects of the BIND daemon. Communication is done over TCP port 953.

For example, we can check the status of the BIND name server.

```
sudo rndc status
```



```
linuxbabe@buster:~$ sudo rndc status
version: BIND 9.11.5-P4-5.1-Debian (Extended Support Version) <id:998753c>
running on buster: Linux x86_64 4.19.0-5-amd64 #1 SMP Debian 4.19.37-5 (2019-06-19)
boot time: Sun, 14 Jul 2019 06:27:31 GMT
last configured: Sun, 14 Jul 2019 06:27:31 GMT
configuration file: /etc/bind/named.conf
CPUs found: 1
worker threads: 1
UDP listeners per interface: 1
number of zones: 102 (97 automatic)
debug level: 0
xfers running: 0
xfers deferred: 0
soa queries in progress: 0
query logging is OFF
recursive clients: 0/900/1000
tcp clients: 3/150
server is up and running
linuxbabe@buster:~$
```

Configurations for a Local DNS Resolver

`/etc/bind/` is the directory that contains configurations for

BIND.

- **named.conf**: the primary config file which includes configs of three other files.
- **db.127**: localhost IPv4 reverse mapping zone file.
- **db.local**: localhost forward IPv4 and IPv6 mapping zone file.
- **db.empty**: an empty zone file

The bind9 package on Debian 10 doesn't ship with a `db.root` file, it now uses the root hints file at `/usr/share/dns/root.hints`. The **root hints** file is used by DNS resolvers to query root DNS servers. There are 13 groups of root DNS servers, from `a.root-servers.net` to `m.root-servers.net`.

Out of the box, the BIND9 server on Debian provides recursive service for localhost and local network clients only. Outside queries will be denied. So you don't have to edit the configuration files. To get you familiar with BIND 9 configurations, I will show you how to enable recursion service anyway.

The main BIND configuration file `/etc/bind/named.conf` sources the settings from 3 other files.

- /etc/bind/named.conf.options
- /etc/bind/named.conf.local
- /etc/bind/named.conf.default-zones

To enable recursion service, edit the first file.

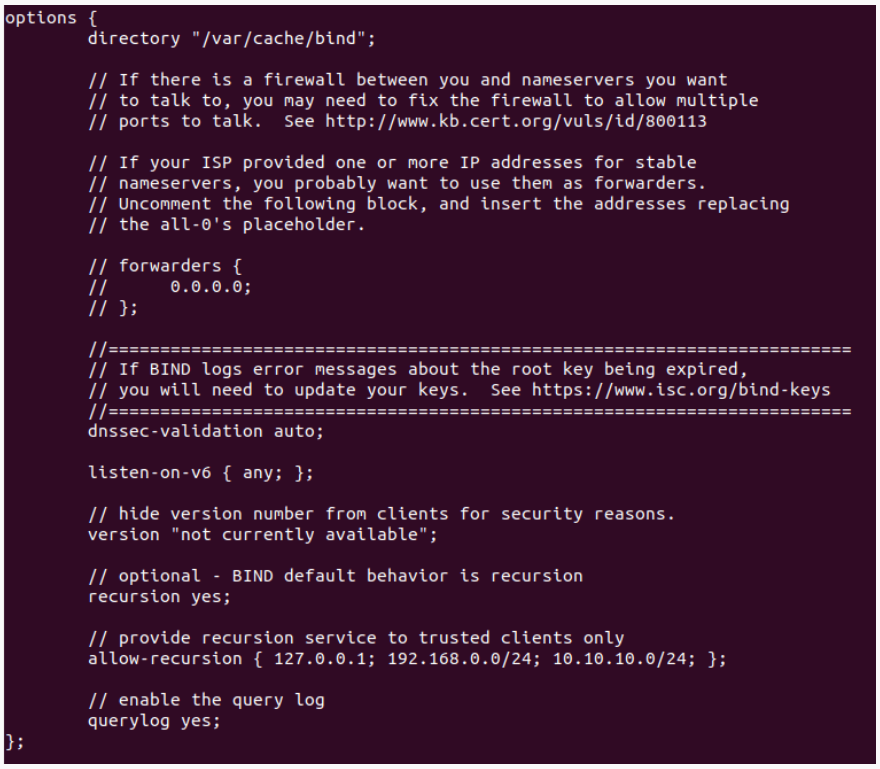
```
sudo nano /etc/bind/named.conf.options
```

In the `options` clause, add the following lines. Replace IP addresses in the `allow-recursion` statement with your own local network addresses.

```
 // hide version number from clients for se
curity reasons.
 version "not currently available";

 // optional - BIND default behavior is rec
ursion
 recursion yes;
 // provide recursion service to trusted cl
ients only
 allow-recursion { 127.0.0.1; 192.168.0.0/2
4; 10.10.10.0/24; };

 // enable the query log
 querylog yes;
```



```
options {
        directory "/var/cache/bind";

        // If there is a firewall between you and nameservers you want
        // to talk to, you may need to fix the firewall to allow multiple
        // ports to talk.  See http://www.kb.cert.org/vuls/id/800113

        // If your ISP provided one or more IP addresses for stable
        // nameservers, you probably want to use them as forwarders.
        // Uncomment the following block, and insert the addresses replacing
        // the all-0's placeholder.

        // forwarders {
        //      0.0.0.0;
        // };

        //========================================================================
        // If BIND logs error messages about the root key being expired,
        // you will need to update your keys.  See https://www.isc.org/bind-keys
        //========================================================================
        dnssec-validation auto;

        listen-on-v6 { any; };

        // hide version number from clients for security reasons.
        version "not currently available";

        // optional - BIND default behavior is recursion
        recursion yes;

        // provide recursion service to trusted clients only
        allow-recursion { 127.0.0.1; 192.168.0.0/24; 10.10.10.0/24; };

        // enable the query log
        querylog yes;
};
```

Save and close the file. Then test the config file syntax.

```
sudo named-checkconf
```

If the test is successful (indicated by a silent output), then restart BIND9.

```
sudo systemctl restart bind9
```

If you have a firewall running on the BIND server, then you need to open port 53 to allow LAN clients to send DNS queries. If you use UFW firewall, you can run the following command.

```
sudo ufw allow in from 192.168.0.0/24 to an
y port 53
```

This will open TCP and UDP port 53 to the private network 192.168.0.0/24. Then from another computer in the same LAN, we can run the following command to query the A record of google.com. Replace 192.168.0.102 with the IP address of your BIND resolver.

```
dig A google.com @192.168.0.102
```

Now on the BIND resolver, check the query log with the following command.

```
sudo journalctl -eu bind9
```

This will show the latest log message of the bind9 service unit. I can find the following line in the log, which indicates that a DNS query for google.com's A record has been received from port 57806 of IP address 192.168.0.103.

```
named[1162]: client @0x7f4d2406f0f0 192.16
8.0.103#57806 (google.com): query: google.c
om IN A +E(0)K (192.168.0.102)
```

Setting the Default DNS Resolver on Debian 10 Buster Server

On the BIND server, we need to set 127.0.0.1 as the default DNS resolver. You can check the current DNS resolver on Debian 10 with the following command.

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

Sample output:

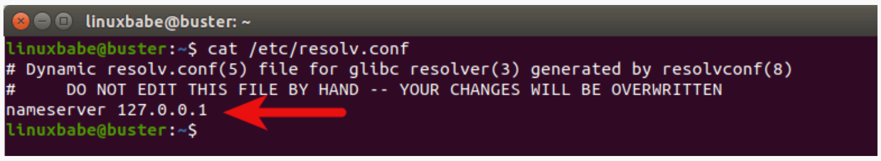
```
# Dynamic resolv.conf(5) file for glibc res
olver(3) generated by resolvconf(8)
#      DO NOT EDIT THIS FILE BY HAND -- YOU
R CHANGES WILL BE OVERWRITTEN
nameserver 2001:19f0:300:1704::6
nameserver 108.61.10.10
```

The bind9 package on Debian 10 ships with a Systemd service unit `bind9-resolvconf.service`, which will help us set BIND as default DNS resolver on Debian server. By default, this service is disabled, we need to start it and enable auto-start at boot time.

```
sudo systemctl start bind9-resolvconf

sudo systemctl enable bind9-resolvconf
```

You can now check the content of `/etc/resolv.conf` again. As you can see, 127.0.0.1 (BIND) is now the default DNS resolver on Debian 10 Buster.



```
linuxbabe@buster:~$ cat /etc/resolv.conf
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 127.0.0.1
linuxbabe@buster:~$
```

If your DNS resolver isn't 127.0.0.1, it might be that your system doesn't have the `resolvconf` binary, which causes the `bind9-resolvconf` service to fail. You need to install the `resolvconf` package and restart the service.

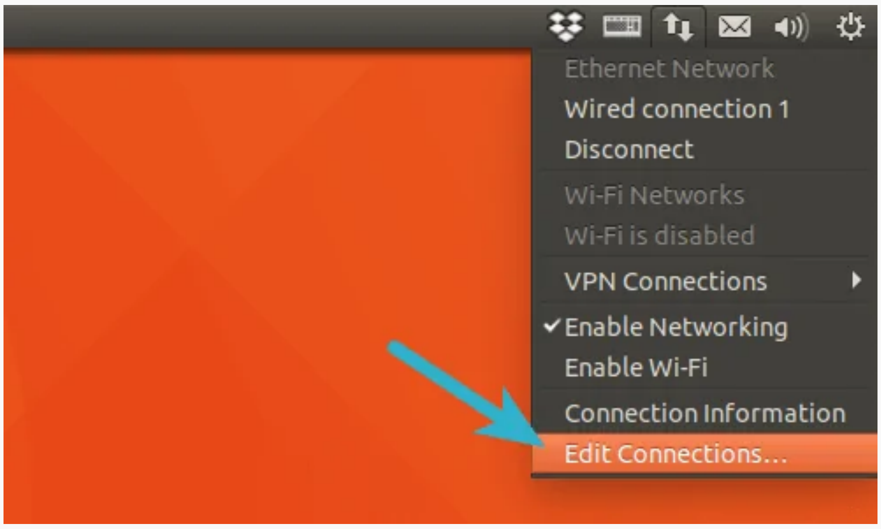
```
sudo apt install resolvconf

sudo systemctl restart bind9-resolvconf
```

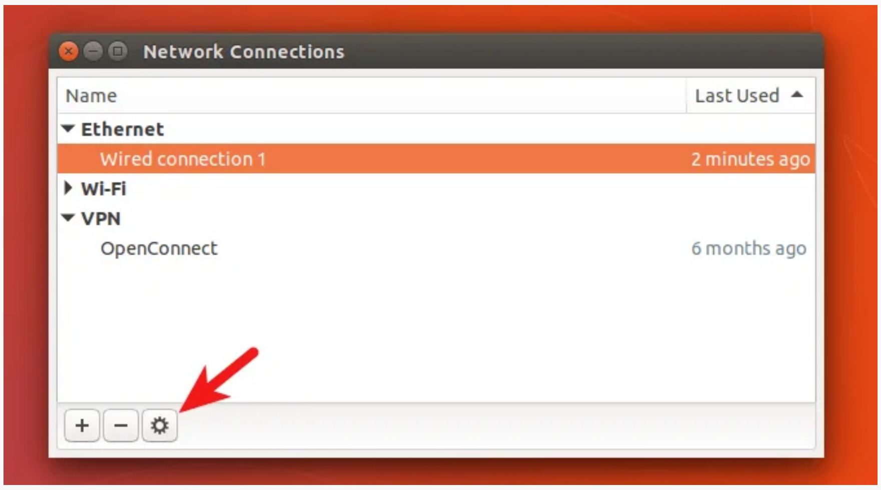
Setting Default DNS Resolver on Client Computers

Now you can configure other computes on the LAN to use the BIND server as DNS resolver. For Windows and MacOS computers, you can search on Google to find out how to set default DNS resolvers. Here I will show you how to set DNS resolver on Linux desktop computers. The following method works on any Linux distro that use NetworkManager.

Click on the Network Manager icon on your Linux desktop to find the `Edit connections`. (On Some Linux distros, you need to right-click the Network Manager.)

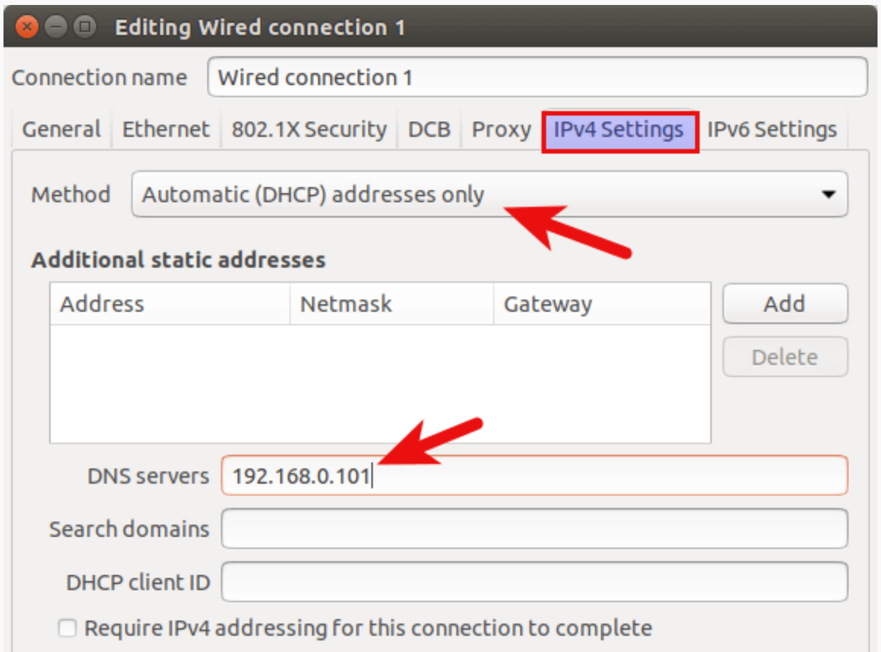


Then select the current connection and click the cog icon to edit this connection.



Name	Last Used
Ethernet	
Wired connection 1	2 minutes ago
Wi-Fi	
VPN	
OpenConnect	6 months ago

Select IPv4 settings tab, change method from `Automatic(DHCP)` to `Automatic(DHCP) addresses only`, which will prevent your Ubuntu system from getting DNS server address from your router. Then specify a DNS server. Here I enter the BIND server's IP address in my LAN.



Connection name: Wired connection 1

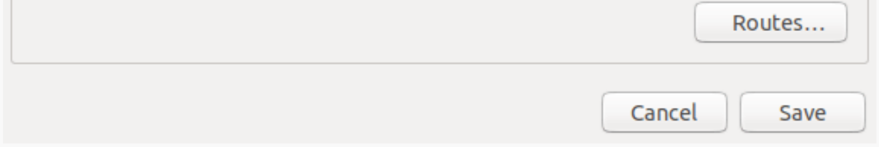
Method: Automatic (DHCP) addresses only

DNS servers: 192.168.0.101

Search domains:

DHCP client ID:

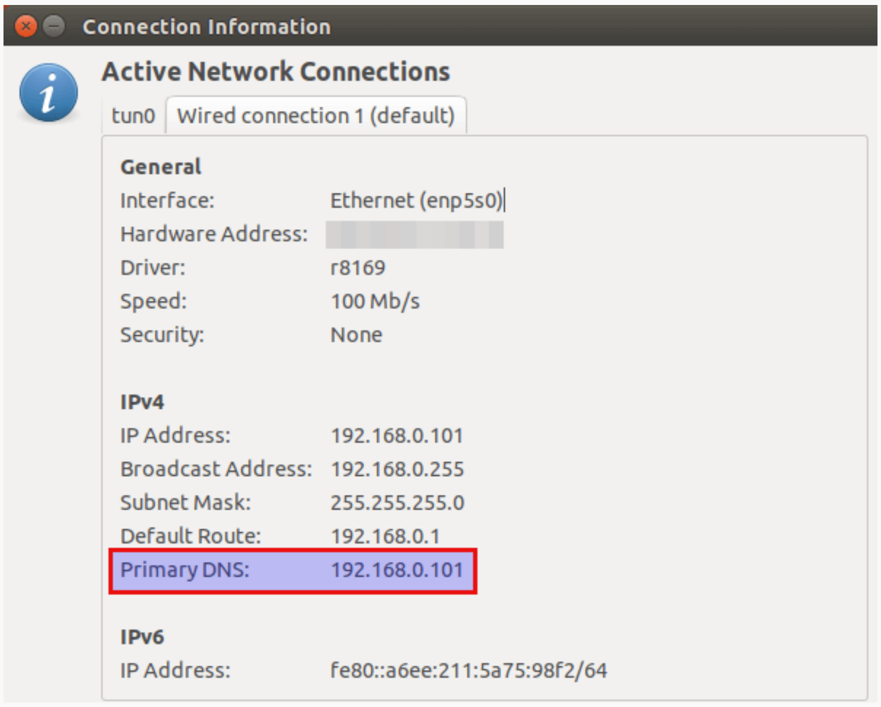
☐ Require IPv4 addressing for this connection to complete



Save your changes. Then restart NetworkManager for the changes to take effect.

```
sudo systemctl restart NetworkManager
```

Once you are reconnected, click the Network Manager icon again and select `connection information`. You can see that your Linux desktop computer is now using your BIND DNS server.



Connection Information

Active Network Connections

tun0 | Wired connection 1 (default)

General

Interface:	Ethernet (enp5s0)
Hardware Address:	[redacted]
Driver:	r8169
Speed:	100 Mb/s
Security:	None

IPv4

IP Address:	192.168.0.101
Broadcast Address:	192.168.0.255
Subnet Mask:	255.255.255.0
Default Route:	192.168.0.1
Primary DNS:	192.168.0.101

IPv6

IP Address:	fe80::a6ee:211:5a75:98f2/64

How to Disable IPv6 in BIND

If you don't use IPv6 in your network, then it's a good idea to turn off IPv6 in BIND, otherwise, there will be a lot of errors about IPv6 in BIND log like below.

```
network unreachable resolving 'mirrors.fedoraproject.org/A/IN': 2001:4178:2:1269:dead:beef:cafe:fed5#53
network unreachable resolving 'mirrors.fedoraproject.org/AAAA/IN': 2001:4178:2:1269:dead:beef:cafe:fed5#53
network unreachable resolving 'mirrors.fedoraproject.org/A/IN': 2610:28:3090:3001:dead:beef:cafe:fed5#53
network unreachable resolving 'mirrors.fedo
raproject.org/AAAA/IN': 2610:28:3090:3001:d
ead:beef:cafe:fed5#53
```

To disable IPv6 in BIND on Ubuntu, simply open the */etc/default /bind9* file

```
sudo nano /etc/default/bind9
```

Add `-4` to the OPTIONS.

```
OPTIONS="-u bind -4"
```

Save and close the file. Then restart BIND and you are done.

```
sudo systemctl restart bind9
```

BIND SERVFAIL

If your BIND resolver can't answer DNS queries (SERVFAIL), and you see the following line in the BIND log.

```
dnssec: warning: managed-keys-zone: Unable
to fetch DNSKEY set '.': timed out
```

It's probably because your server doesn't have a working IPv6 connectivity. This happened to one of my servers. I thought IPv6 connectivity is working as usual, but it's suddenly broken for reasons I didn't know. Once I disabled IPv6 in BIND, DNS resolution is working again.

BIND Automatic Restart

If for any reason your Bind process is killed, you need to run the following command to restart it.

```
sudo systemctl restart bind9
```

Instead of manually typing this command, we can make bind automatically restart by editing the `named.service` systemd service unit. To override the default systemd service configuration, we create a separate directory.

```
sudo mkdir -p /etc/systemd/system/bind9.ser
vice.d/
```

Then create a file under this directory.

```
sudo nano /etc/systemd/system/bind9.servic
e.d/restart.conf
```

Add the following lines in the file, which will make Bind automatically restart 5 seconds after a failure is detected.

```
[Service]
Restart=always
RestartSec=5s
```

Save and close the file. Then reload systemd.

```
sudo systemctl daemon-reload
```

To check if this would work, kill Bind with:

```
sudo pkill named
```

Then check Bind status. You will find Bind automatically restarted.

```
systemctl status bind9
```

BIND max-cache-size

BIND can cache DNS results on the server to speed up DNS lookup for clients. BIND assumes you are running a dedicated DNS resolver, i.e, no other web services are running on the same host, so the default cache size (defined by `max-cache-size`) is set to 90% of the total RAM to achieve best performance. You can see a line like below in the BIND log (`sudo journalctl -eu` `bind9`) when BIND starts.

```
none:100: 'max-cache-size 90%' - setting to
7165MB (out of 7961MB)
```

Note that BIND will not use 90% of your RAM immediately. If there are only a few DNS requests, BIND uses only a small amount of RAM, because there’s not many DNS results to cache. If there are lots of DNS requests, then it will use lots of RAM to store the DNS cache.

If your RAM is limited, you might not want BIND to use 90% of your RAM for cache. Edit the BIND configuration file `/etc/bind/named.conf.options`.

```
sudo nano /etc/bind/named.conf.options
```

Add the following directive in the `options` clause. Change 50% to your preferred value.

```
max-cache-size 50%;
```

Restart BIND for the change to take effect.

```
sudo systemctl restart bind9
```

Conclusion

I hope this tutorial helped you set up a local DNS resolver on Debian 10 Buster with BIND9. As always, if you found this post useful, then subscribe to our free newsletter to get more tips and tricks. Take care 🙂

Rate this tutorial

★★★★★ [Total: 2 Average: 5]

BIND9 Debian Debian Desktop Debian Server

DNS Resolver Linux