

# Getting Started with the BIND DNS Server

Domain Name Server (DNS) is a critical component of the internet infrastructure, and building your DNS server can be challenging. Well, not with BIND, where you can create your BIND DNS server in no time. BIND has an excellent reputation among administrators for its flexibility and high availability support.

In this article, you'll learn how to install and configure a secure BIND DNS Server and verify that sub-domains are resolved to the correct IP address.

Read on and create your DNS server with no sweat!

## Prerequisites

This tutorial will be a hands-on demonstration. To follow along, ensure you have the following.

- A Linux server – This example uses the Ubuntu20.04 server.

**Related:**[How to Install Ubuntu 20.04 \[Step-by-Step\]](#)

- A non-root user with root privileges or root/administrator user.
- A domain name pointed to the server IP address – This demo uses the atadomain.io domain and server IP address 172.16.1.10.

## Installing BIND Packages

The default Ubuntu repository provides BIND packages but doesn't come installed with your system. You can install BIND as the main DNS Server or authoritative only. BIND gives you powerful features, such as master-slave installation support, DNSSEC support, and built-in Access Control Lists (ACL).

To get started with BIND DNS, you'll first need to install the BIND packages on your machine with the apt package manager.

1. Open your terminal and log in to your server.

2. Next, run the apt update command below to update and refresh the repository package index. This command ensures that you are installing the latest version of

packages.

```
sudo apt update
```

Copy

```
root@server-ubuntu:~#  
root@server-ubuntu:~# sudo apt update  
Get:1 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]  
Hit:2 http://us.archive.ubuntu.com/ubuntu focal InRelease  
Get:3 http://us.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]  
Get:4 http://us.archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]  
Get:5 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [1,642 kB]  
Get:6 http://us.archive.ubuntu.com/ubuntu focal-updates/main i386 Packages [616 kB]  
Fetched 2,595 kB in 12s (222 kB/s)  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done
```

Refreshing Package Index

3. Once updated, run the below apt install command to install BIND packages for the Ubuntu server.

The bind9-utils and bind9-dnsutils packages provide additional command-line tools for BIND. These packages are useful for testing and managing the BIND DNS server.

```
sudo apt install bind9 bind9-utils bind9-dnsutils -y
```

Copy

```
root@server-ubuntu:~#  
root@server-ubuntu:~# sudo apt install bind9 bind9-utils bind9-dnsutils -y  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following additional packages will be installed:  
  bind9-libs dns-root-data python3-ply  
Suggested packages:  
  bind-doc resolvconf python-ply-doc  
The following NEW packages will be installed:  
  bind9 bind9-utils dns-root-data python3-ply  
The following packages will be upgraded:  
  bind9-dnsutils bind9-libs  
2 upgraded, 4 newly installed, 0 to remove and 21 not upgraded.  
Need to get 1,699 kB of archives.  
After this operation, 1,931 kB of additional disk space will be used.  
Get:1 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 bind9-dnsutils amd64 1:9.16.1-0ubuntu2.10 [134 kB]  
Get:2 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 bind9-libs amd64 1:9.16.1-0ubuntu2.10 [1,108 kB]  
Get:3 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 python3-ply all 3.11-3ubuntu0.1 [46.3 kB]  
Get:4 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 bind9-utils amd64 1:9.16.1-0ubuntu2.10 [172 kB]  
Get:5 http://us.archive.ubuntu.com/ubuntu focal/main amd64 dns-root-data all 2019052802 [5,300 B]  
Get:6 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 bind9 amd64 1:9.16.1-0ubuntu2.10 [233 kB]  
Fetched 1,699 kB in 5s (329 kB/s)
```

Installing BIND Packages

4. Lastly, run the systemctl command below to verify the BIND service.

The BIND package comes with the service named and is automatically started and enabled during the BIND package installation.

```
# Check if named service enabled
```

```
sudo systemctl is-enabled named
```

```
# Check named service status
```

```
sudo systemctl status named
```

Copy

Now you should see the BIND named service is enabled with the status as active (running). At this point, the BIND service will run automatically at system startup/boot.

```
root@server-ubuntu:~#  
root@server-ubuntu:~# sudo systemctl is-enabled named  
enabled  
root@server-ubuntu:~# sudo systemctl status named  
● named.service - BIND Domain Name Server  
   Loaded: loaded (/lib/systemd/system/named.service; enabled; vendor preset: enabled)  
   Active: active (running) since Sat 2022-03-19 02:04:06 UTC; 57s ago  
     Docs: man:named(8)  
  Main PID: 1978 (named)  
    Tasks: 8 (limit: 1065)  
   Memory: 18.9M  
    CGroup: /system.slice/named.service  
            └─1978 /usr/sbin/named -f -u bind
```

Checking BIND named service

**Related:** [Correct Way of Using Ubuntu systemctl to Control Systemd](#)

## Configuring BIND DNS Server

You've now installed BIND packages on the Ubuntu server, so it's time to set up the BIND installation on your Ubuntu server. How? By editing BIND and the namedservice's configurations.

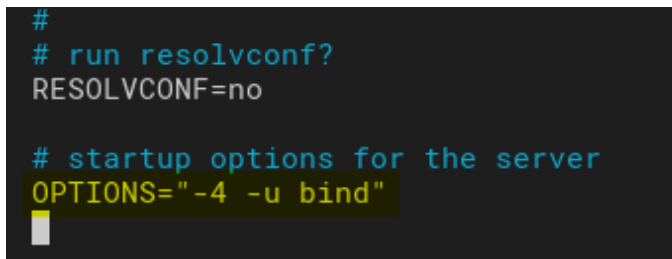
All configuration for BIND is available at the `/etc/bind/` directory, and configurations for the namedservice at `/etc/default/named`.

1. Edit the `/etc/default/named` configuration using your preferred editor and add option `-4` on the `OPTIONS` line, as shown below. This option will run the `named` service on IPv4 only.

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

Copy

Save the changes you made and close the file.



```
#
# run resolvconf?
RESOLVCONF=no

# startup options for the server
OPTIONS="-4 -u bind"
```

Configuring BIND to Run on IPv4 Only

2. Next, edit the `/etc/bind/named.conf.options` file and populate the following configuration below the directory `"/var/cache/bind";` line.

This configuration sets the BIND service to run on default UDP port 53 on the server's localhost and public IP address ([172.16.1.10](https://www.cloudflare.com/ips/)). At the same time, it allows queries from any host to the BIND DNS server using the Cloudflare DNS [1.1.1.1](https://www.cloudflare.com/dns/) as the forwarder.

```
// listen port and address
```

```
listen-on port 53 { localhost; 172.16.1.10; };
```

```
// for public DNS server - allow from any
```

```
allow-query { any; };
```

```
// define the forwarder for DNS queries
```

```
forwarders { 1.1.1.1; };
```

```
// enable recursion that provides recursive query
```

```
recursion yes;
```

Copy

At the bottom, comment out the `listen-on-v6 { any; };` line, as shown below, to disable the named service from running on IPv6.

```

GNU nano 4.8
options {
    directory "/var/cache/bind";

    // listen port and address
    listen-on port 53 { localhost; 172.16.1.10; };

    // for public DNS server - allow from any
    allow-query { any; };

    // define the forwarder for DNS queries
    forwarders { 1.1.1.1; };

    // enable recursion that provides recursive query
    recursion yes;

    // 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; };
};

```

Editing the BIND Configuration Options

3. Lastly, run the following command to verify the BIND configuration.

```
sudo named-checkconf
```

Copy

If there's no output, the BIND configurations are correct without any error.

```

root@server-ubuntu:~# sudo named-checkconf
root@server-ubuntu:~#
root@server-ubuntu:~#

```

Verifying BIND Configurations

## Setting Up DNS Zones

At this point, you've configured the basic configuration of the BIND DNS Server. You're ready to create a DNS Server with your domain and add other sub-domains for your applications. You'll need to define and create a new DNS zones configuration to do so.

In this tutorial, you'll create a new Name Server ([ns1.atadomain.io](#)) and sub-domains ([www.atadomain.io](#), [mail.atadomain.io](#), [vault.atadomain.io](#)).

1. Edit the `/etc/bind/named.conf.local` file using your preferred editor and add the following configuration.

This configuration defines the forward zone (`/etc/bind/zones/forward.atadomain.io`), and the reverse zone (`/etc/bind/zones/reverse.atadomain.io`) for the [atadomain.io](#) domain name.

```
zone "atadomain.io" {  
  
    type master;  
  
    file "/etc/bind/zones/forward.atadomain.io";  
  
};
```

```
ne "1.16.172.in-addr.arpa" {  
  
    type master;  
  
    file "/etc/bind/zones/reverse.atadomain.io";  
  
};
```

Copy

Save the changes and close the file.

```
// Do any local configuration here
//

// Consider adding the 1918 zones here, if they are not used in your
// organization
//include "/etc/bind/zones.rfc1918";

zone "atadomain.io" {
    type master;
    file "/etc/bind/zones/forward.atadomain.io";
};

zone "1.16.172.in-addr.arpa" {
    type master;
    file "/etc/bind/zones/reverse.atadomain.io";
};
```

Defining Forward Zone and Reverse Zone BIND

2. Next, run the below command to create a new directory (/etc/bind/zones) for string DNS zones configurations.

```
mkdir -p /etc/bind/zones/
```

Copy

3. Run each command below to copy the default forward and reverse zones configuration to the /etc/bind/zones directory.

```
# Copy default forward zone
```

```
sudo cp /etc/bind/db.local /etc/bind/zones/forward.atadomain.io
```

```
# Copy default reverse zone
```



```
sudo cp /etc/bind/db.127 /etc/bind/zones/reverse.atadomain.io
```

```
# List contents of the /etc/bind/zones/ directory
```

```
ls /etc/bind/zones/
```

Copy

```
root@server-ubuntu:~# sudo cp /etc/bind/db.local /etc/bind/zones/forward.atadomain.io
root@server-ubuntu:~# sudo cp /etc/bind/db.127 /etc/bind/zones/reverse.atadomain.io
root@server-ubuntu:~# ls /etc/bind/zones/
forward.atadomain.io  reverse.atadomain.io
root@server-ubuntu:~#
root@server-ubuntu:~#
```

Copying Default Zones Configurations

4. Now, edit the forward zone

configuration (*/etc/bind/zones/forward.atadomain.io*) using your preferred editor and populate the configuration below.

The forward zone configuration is where you define your domain name and the server IP address. This configuration will translate the domain name to the correct IP address of the server.

The configuration below creates the following name server and sub-domains:

- ns1.atadomain.io – The main Name Server for your domain with the IP address 172.16.1.10.  
MX record for the atadomain.io domain that is handled by the mail.atadomain.io.  
The MX record is used for the mail server.
- Sub-domains for applications: www.atadomain.io, mail.atadomain.io,  
and vault.atadomain.io.

;

; BIND data file for the local loopback interface

;

\$TTL 604800

@ IN SOA atadomain.io. root.atadomain.io. (

2 ; Serial

604800 ; Refresh

86400 ; Retry

2419200 ; Expire

604800 ) ; Negative Cache TTL

; Define the default name server to ns1.atadomain.io

@ IN NS ns1.atadomain.io.

; Resolve ns1 to server IP address

; A record for the main DNS

ns1 IN A 172.16.1.10

; Define MX record for mail

atadomain.io. IN MX 10 mail.atadomain.io.

; Other domains for atadomain.io

; Create subdomain www - mail - vault

www IN A 172.16.1.10

mail IN A 172.16.1.20

vault IN A 172.16.1.50

Copy

Save the changes and close the file.

```

; BIND data file for local loopback interface
;
$TTL      604800
@         IN      SOA      atadomain.io. root.atadomain.io. (
                        2      ; Serial
                        604800 ; Refresh
                        86400  ; Retry
                        2419200 ; Expire
                        604800 ) ; Negative Cache TTL
;
; Define the default name server to ns1.atadomain.io
@         IN      NS       ns1.atadomain.io.
;
; Resolve ns1 to server IP address
; A record for main DNS
ns1       IN      A        172.16.1.10
;
; Define MX record for mail
atadomain.io. IN    MX      10    mail.atadomain.io.
;
; Other domains for atadomain.io
; Create subdomain www - mail - vault
www       IN      A        172.16.1.10
mail      IN      A        172.16.1.20
vault     IN      A        172.16.1.50

```

### Configuring Forward Zone BIND

5. Like the forward zone, edit the reverse zone configuration file (*/etc/bind/zones/reverse.atadomain.io*) and populate the following configuration.

The reverse zone translates the server IP address to the domain name. The reverse zone or PTR record is essential for services like the Mail server, which affects the Mail server's reputation.

The PTR record uses the last block of the IP address, like the PTR record with the number 10 for the server IP address 172.16.1.10.

This configuration creates the reverse zone or PTR record for the following domains:

- Name server ns1.atadomain.io with the reverse zone or PTR record 172.16.1.10.
- PTR record for the domain mail.atadomain.io to the server IP address 172.16.1.20.

;

; BIND reverse data file for the local loopback interface

;

\$TTL 604800

@ IN SOA atadomain.io. root.atadomain.io. (

1 ; Serial

604800 ; Refresh

86400 ; Retry

2419200 ; Expire

604800 ) ; Negative Cache TTL

; Name Server Info for ns1.atadomain.io

@ IN NS ns1.atadomain.io.

; Reverse DNS or PTR Record for ns1.atadomain.io

; Using the last number of DNS Server IP address: 172.16.1.10

10 IN PTR ns1.atadomain.io.

; Reverse DNS or PTR Record for mail.atadomain.io

; Using the last block IP address: 172.16.1.20

20 IN PTR mail.atadomain.io.

Copy

Save the changes and close the file.

```
GNU nano 4.8
;
; BIND reverse data file for local loopback interface
;
$TTL      604800
@         IN      SOA      atadomain.io. root.atadomain.io. (
                        1      ; Serial
                        604800 ; Refresh
                        86400  ; Retry
                        2419200 ; Expire
                        604800 ) ; Negative Cache TTL

; Name Server Info for ns1.atadomain.io
@         IN      NS       ns1.atadomain.io.
ns1       IN      A        172.168.1.10

; Reverse DNS or PTR Record for ns1.atadomain.io
; Using last number of DNS Server IP address: 172.16.1.10
10        IN      PTR      ns1.atadomain.io.

; Reverse DNS or PTR Record for mail.atadomain.io
; Using the last block IP address : 172.16.1.20
20        IN      PTR      mail.atadomain.io.
█
```

Configuring Reverse Zone BIND

6. Now, run the following commands to check and verify BIND configurations.

```
# Checking the main configuration for BIND
```

```
sudo named-checkconf
```

```
# Checking forward zone forward.atadomain.io
```

```
sudo named-checkzone atadomain.io
/etc/bind/zones/forward.atadomain.io
```

```
# Checking reverse zone reverse.atadomain.io
```

```
sudo named-checkzone atadomain.io  
/etc/bind/zones/reverse.atadomain.io
```

Copy

When your configuration is correct, you'll see an output similar below.

```
root@server-ubuntu:~# sudo named-checkconf  
root@server-ubuntu:~# sudo named-checkzone atadomain.io /etc/bind/zones/forward.atadomain.io  
zone atadomain.io/IN: loaded serial 2  
OK  
root@server-ubuntu:~# sudo named-checkzone atadomain.io /etc/bind/zones/reverse.atadomain.io  
zone atadomain.io/IN: loaded serial 1  
OK  
root@server-ubuntu:~#
```

Checking and Verifying BIND configurations

7. Lastly, run the systemctl command below to restart and verify the named service.  
Doing so applies new changes to the namedservice.

```
# Restart named service
```

```
sudo systemctl restart named
```

```
# Verify named service
```

```
sudo systemctl status named
```

Copy

Below, you can see the named service status is active (running).



```
root@server-ubuntu:~#  
root@server-ubuntu:~#  
root@server-ubuntu:~# sudo systemctl restart named  
root@server-ubuntu:~# sudo systemctl status named  
● named.service - BIND Domain Name Server  
   Loaded: loaded (/lib/systemd/system/named.service; enabled; vendor preset: enabled)  
   Active: active (running) since Sat 2022-03-19 03:16:23 UTC; 4s ago  
     Docs: man:named(8)  
    Main PID: 3277 (named)  
      Tasks: 8 (limit: 1065)  
     Memory: 16.3M  
    CGroup: /system.slice/named.service  
            └─3277 /usr/sbin/named -f -4 -u bind
```

Restarting Named Service and Verifying Named Service Status

## Opening DNS Port with UFW Firewall

At this point, you've completed the BIND DNS Server installation. But you still have to secure your DNS Server. You'll set up the UFW firewall and open the DNS port for any queries to the server. Doing so allows clients to make a query to the BIND DNS server.

*For security reasons, it's recommended to run and enable the UFW firewall on your Ubuntu server.*

**Related:** [How To Set Up the UFW Firewall on Linux](#)

1. Run the ufw command below to check available applications on the UFW firewall.

```
sudo ufw app list
```

Copy

You should see the Bind9 on the UFW application list below.

```
root@server-ubuntu:~# sudo ufw app list  
Available applications:  
  Bind9  
  OpenSSH
```

Listing Available Applications on the UFW Firewall

2. Now run the below command to allow the Bind9 to the UFW firewall.

```
sudo ufw allow Bind9
```

Copy

```
root@server-ubuntu:~# sudo ufw allow Bind9
Rule added
Rule added (v6)
```

Adding UW Rule

3. Lastly, run the following command to check enabled rules on the UFW firewall.

```
sudo ufw status
```

Copy

You should see the Bind9 application on the list like in the screenshot below.

```
root@server-ubuntu:~# sudo ufw status
Status: active

To Action From
--
OpenSSH ALLOW Anywhere
Bind9 ALLOW Anywhere
OpenSSH (v6) ALLOW Anywhere (v6)
Bind9 (v6) ALLOW Anywhere (v6)
```

Verifying List of Rules

## Verifying BIND DNS Server Installation

You've now completed the BIND DNS installation and configured the UFW firewall. But how do you verify your DNS Server installation? The dig command will do the trick.

Dig is a command-line utility for troubleshooting DNS Server installation. dig performs DNS lookup for the given domain name and displays detailed answers for the domain name target. On the Ubuntu system, dig is part of the bind9-dnsutils package.

**Related:** [How to Install Sysdig to Monitor Your Linux System](#)

To verify your BIND DNS server installation:

1. Run each dig command below to verify the sub-domains [www.atadomain.io](#), [mail.atadomain.io](#), and [vault.atadomain.io](#).

If your DNS Server installation is successful, each sub-domain will be resolved to the correct IP address based on the [forward.atadomain.io](#) configuration.

# Checking the domain names


```
dig @172.16.1.10 www.atadomain.io
```

```
dig @172.16.1.10 mail.atadomain.io
```

```
dig @172.16.1.10 vault.atadomain.io
```

Copy

Below is the output of the sub-domain www.atadomain.io resolved to the server IP address 172.16.1.10.

```
~ >
~ > dig @172.16.1.10 www.atadomain.io 
; <<>> DiG 9.18.0 <<>> @172.16.1.10 www.atadomain.io
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19708
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 6aed6759b6bfe7800100000062354c666a0277d741cfac36 (good)
;; QUESTION SECTION:
;www.atadomain.io.                IN      A

;; ANSWER SECTION:
www.atadomain.io.                604800  IN      A      172.16.1.10

;; Query time: 1 msec
;; SERVER: 172.16.1.10#53(172.16.1.10) (UDP)
;; MSG SIZE rcvd: 89
```

Checking sub-domain www.atadomain.io

Below is the sub-domain mail.atadomain.io resolved to the server IP address 172.16.1.20.

```

~ >
~ > dig @172.16.1.10 mail.atadomain.io ←
; <<>> DiG 9.18.0 <<>> @172.16.1.10 mail.atadomain.io
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 27141
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: e8e894cfda8090a50100000062354c86d7cf1efbfd9a19a8 (good)
;; QUESTION SECTION:
;mail.atadomain.io.          IN      A

;; ANSWER SECTION:
mail.atadomain.io.          604800  IN      A      172.16.1.20

;; Query time: 0 msec
;; SERVER: 172.16.1.10#53(172.16.1.10) (UDP)

;; MSG SIZE rcvd: 90

```

Checking sub-domain mail.atadomain.io

And below is the sub-domain vault.atadomain.io resolved to the server IP address 172.16.1.50.

```

~ >
~ > dig @172.16.1.10 vault.atadomain.io ←
; <<>> DiG 9.18.0 <<>> @172.16.1.10 vault.atadomain.io
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26229
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: ec11ca1c433149a50100000062354c9d3a44204d39267ef2 (good)
;; QUESTION SECTION:
;vault.atadomain.io.        IN      A

;; ANSWER SECTION:
vault.atadomain.io.          604800  IN      A      172.16.1.50

;; Query time: 0 msec
;; SERVER: 172.16.1.10#53(172.16.1.10) (UDP)

;; MSG SIZE rcvd: 91

```

Checking sub-domain vault.atadomain.io

2. Next, run the dig command below to verify the MX record for the atadomain.iomain.

```
dig @172.16.1.10 atadomain.io MX
```

Copy

You should see the atadomain.io domain has the MX record mail.atadomain.io.

```
~ >
~ > dig @172.16.1.10 atadomain.io MX ←
; <<>> DiG 9.18.0 <<>> @172.16.1.10 atadomain.io MX
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 56780
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:: udp: 4096
; COOKIE: 75e99bd35aa780000100000062354cc7291dcd9b9eab05ef (good)
;; QUESTION SECTION:
;atadomain.io.                IN      MX

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

;; ADDITIONAL SECTION:
mail.atadomain.io.          604800  IN      A        172.16.1.20

;; Query time: 0 msec
;; SERVER: 172.16.1.10#53(172.16.1.10) (UDP)
;; MSG SIZE rcvd: 106
```

Checking MX record for atadomain.io

3. Lastly, run the following commands to verify the PTR record or reverse zone for the server IP addresses 172.16.1.10 and 172.16.1.20.

If your BIND installation is successful, each IP address will be resolved to the domain name defined on the reverse.atadomain.io configuration.

```
# checking PTR record or reverse DNS
```

```
dig @172.16.1.10 -x 172.16.1.10
```

```
dig @172.16.1.10 -x 172.16.1.20
```

Copy

You can see below, the server IP address 172.16.1.10 is resolved to the domain name ns1.atadomain.io.

```
~ >
~ > dig @172.16.1.10 -x 172.16.1.10 ←
; <<>> DiG 9.18.0 <<>> @172.16.1.10 -x 172.16.1.10
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 4596
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1


;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: adf87a72c57463e10100000062354df8b35eda055f1437a0 (good)
;; QUESTION SECTION:
;10.1.16.172.in-addr.arpa.      IN      PTR

;; ANSWER SECTION:
10.1.16.172.in-addr.arpa. 604800 IN      PTR      ns1.atadomain.io.

;; Query time: 1 msec
;; SERVER: 172.16.1.10#53(172.16.1.10) (UDP)
;; MSG SIZE rcvd: 111
```

Checking PTR record for IP address 172.16.1.10

As you see below, the server IP address 172.16.1.20 is resolved to the domain name mail.atadomain.io.

```
~ >
~ > dig @172.16.1.10 -x 172.16.1.20 

; <<>> DiG 9.18.0 <<>> @172.16.1.10 -x 172.16.1.20
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 45145
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 6abf599a30b1d47a0100000062354e1e9eb0b780b7ffca89 (good)
;; QUESTION SECTION:
;20.1.16.172.in-addr.arpa.      IN      PTR

;; ANSWER SECTION:
20.1.16.172.in-addr.arpa. 604800 IN      PTR      mail.atadomain.io.

;; Query time: 1 msec
;; SERVER: 172.16.1.10#53(172.16.1.10) (UDP)
;; MSG SIZE rcvd: 112
```

Checking PTR record for IP address 172.16.1.20

## Conclusion

Throughout this tutorial, you've learned how to create and set up a secure BIND DNS server on your Ubuntu server. You've also created the forward and reverse zone for adding your domain and verified DNS servers by running dig commands.

Now, how can you implement a BIND DNS Server in your environment? Perhaps implement BIND as an authoritative server? Or set up high availability with master-slave BIND installation?