DNS

What is DNS?

The Domain Name System (DNS) is the phonebook of the Internet. Humans access information online through domain names, like nytimes.com or espn.com. Web browsers interact through Internet Protocol (IP) addresses. DNS translates domain names to IP addresses so browsers can load Internet resources.

Each device connected to the Internet has a unique IP address which other machines use to find the device. DNS servers eliminate the need for humans to memorize IP addresses such as 192.168.1.1 (in IPv4), or more complex newer alphanumeric IP addresses such as 2400:cb00:2048:1::c629:d7a2 (in IPv6).

How does DNS work?

The process of DNS resolution involves converting a hostname (such as www.example.com) into a computer-friendly IP address (such as 192.168.1.1). An IP address is given to each device on the Internet, and that address is necessary to find the appropriate Internet device - like a street address is used to find a particular home. When a user wants to load a webpage, a translation must occur between what a user types into their web browser (example.com) and the machine-friendly address necessary to locate the example.com webpage.

In order to understand the process behind the DNS resolution, it's important to learn about the different hardware components a DNS query must pass between. For the web browser, the DNS lookup occurs "behind the scenes" and requires no interaction from the user's computer apart from the initial request.

There are 4 DNS servers involved in loading a webpage:

DNS recursor - The recursor can be thought of as a librarian who is asked to go find a particular book somewhere in a library. The DNS recursor is a server designed to receive queries from client machines through applications such as web browsers. Typically the recursor is then responsible for making additional requests in order to satisfy the client's DNS query.

Root nameserver - The root server is the first step in translating (resolving) human readable host names into IP addresses. It can be thought of like an index in a library that points to different racks of books - typically it serves as a reference to other more specific locations.

TLD nameserver - The top level domain server (TLD) can be thought of as a specific rack of books in a library. This nameserver is the next step in the search for a specific IP address, and it hosts the last portion of a hostname (In example.com, the TLD server is "com").

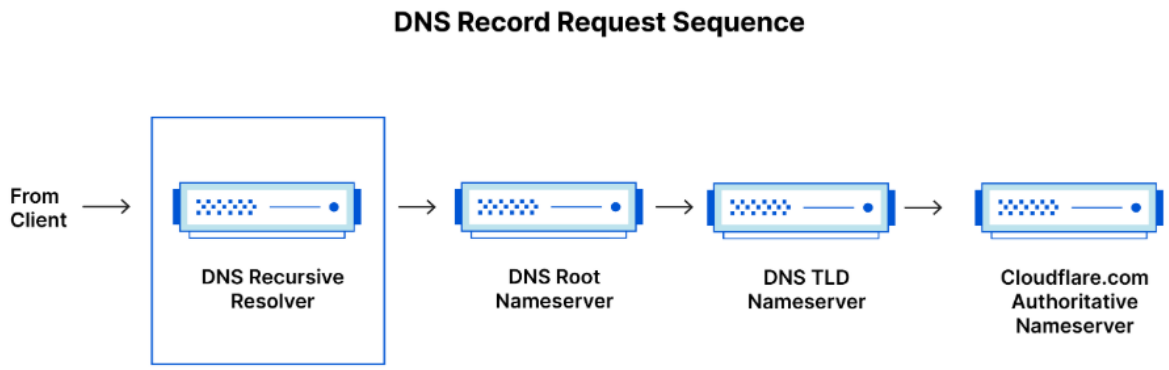
Authoritative nameserver - This final nameserver can be thought of as a dictionary on a rack of books, in which a specific name can be translated into its definition. The authoritative nameserver is the last stop in the nameserver query. If the authoritative name server has access to the requested record, it will return the IP address for the requested hostname back to the DNS Recursor (the librarian) that made the initial request.

What's the difference between an authoritative DNS server and a recursive DNS resolver?

Both concepts refer to servers (groups of servers) that are integral to the DNS infrastructure, but each performs a different role and lives in different locations inside the pipeline of a DNS query. One way to think about the difference is the recursive resolver is at the beginning of the DNS query and the authoritative nameserver is at the end.

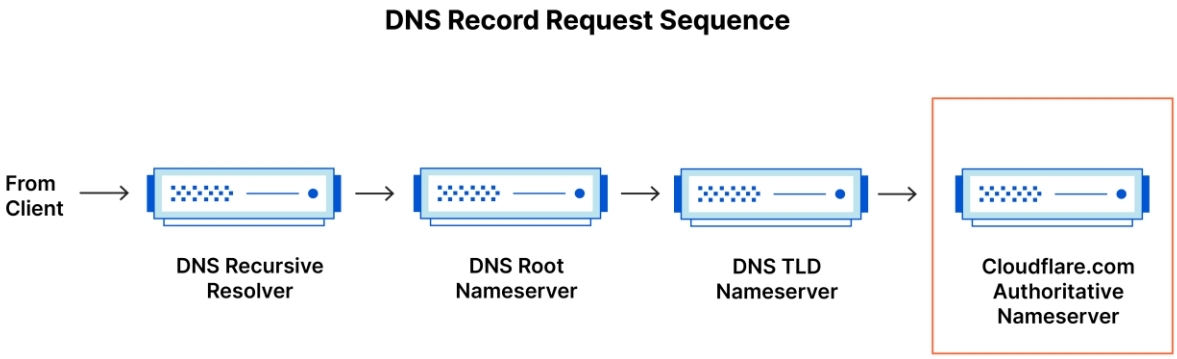
Recursive DNS resolver

The recursive resolver is the computer that responds to a recursive request from a client and takes the time to track down the DNS record. It does this by making a series of requests until it reaches the authoritative DNS nameserver for the requested record (or times out or returns an error if no record is found). Luckily, recursive DNS resolvers do not always need to make multiple requests in order to track down the records needed to respond to a client; caching is a data persistence process that helps short-circuit the necessary requests by serving the requested resource record earlier in the DNS lookup.



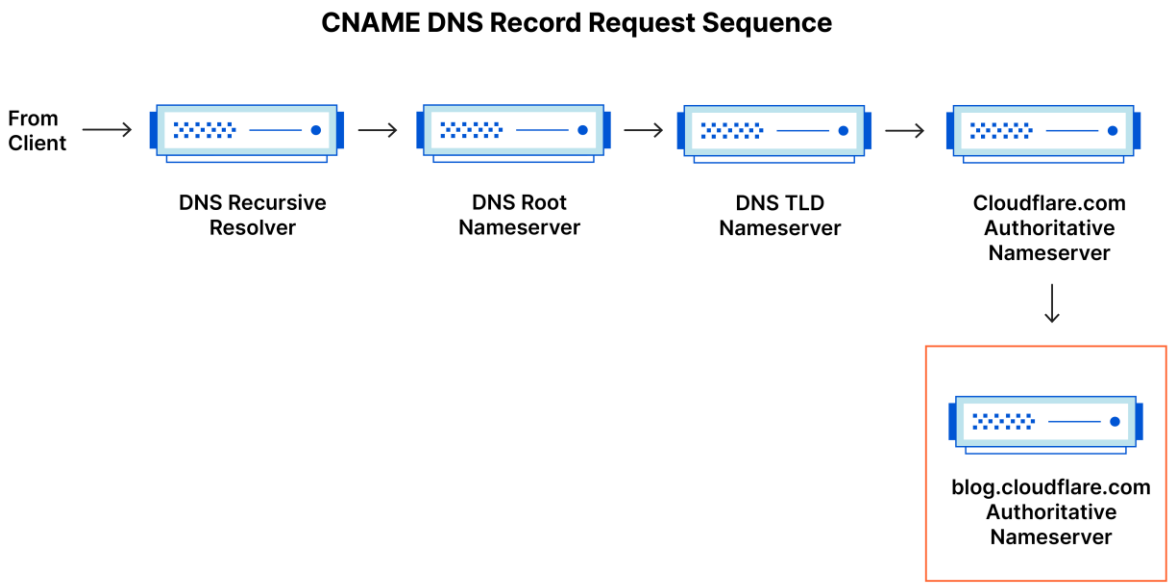
Authoritative DNS server

Put simply, an authoritative DNS server is a server that actually holds, and is responsible for, DNS resource records. This is the server at the bottom of the DNS lookup chain that will respond with the queried resource record, ultimately allowing the web browser making the request to reach the IP address needed to access a website or other web resources. An authoritative nameserver can satisfy queries from its own data without needing to query another source, as it is the final source of truth for certain DNS records.



It's worth mentioning that in instances where the query is for a subdomain such as

foo.example.com or blog.cloudflare.com, an additional nameserver will be added to the sequence after the authoritative nameserver, which is responsible for storing the subdomain's CNAME record.



There is a key difference between many DNS services and the one that Cloudflare provides. Different DNS recursive resolvers such as Google DNS, OpenDNS, and providers like Comcast all maintain data center installations of DNS recursive resolvers. These resolvers allow for quick and easy queries through optimized clusters of DNS-optimized computer systems, but they are fundamentally different than the nameservers hosted by Cloudflare.

Cloudflare maintains infrastructure-level nameservers that are integral to the functioning of the Internet. One key example is the f-root server network which Cloudflare is partially responsible for hosting. The F-root is one of the root level DNS nameserver infrastructure components responsible for the billions of Internet requests per day. Our Anycast network puts us in a unique position to handle large volumes of DNS traffic without service interruption.

What are the steps in a DNS lookup?

For most situations, DNS is concerned with a domain name being translated into the appropriate IP address. To learn how this process works, it helps to follow the path of a DNS lookup as it travels from a web browser, through the DNS lookup process, and back again. Let's take a look at the steps.

Note: Often DNS lookup information will be cached either locally inside the querying computer or remotely in the DNS infrastructure. There are typically 8 steps in a DNS lookup. When DNS information is cached, steps are skipped from the DNS lookup process which makes it quicker. The example below outlines all 8 steps when nothing is cached.

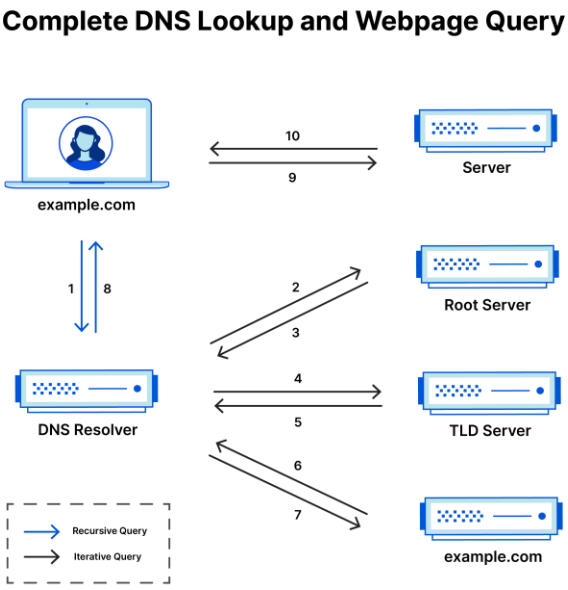
The 8 steps in a DNS lookup:

- A user types 'example.com' into a web browser and the query travels into the Internet and is received by a DNS recursive resolver.
- The resolver then queries a DNS root nameserver (.).
- The root server then responds to the resolver with the address of a Top Level Domain (TLD) DNS server (such as .com or .net), which stores the information for its domains. When searching for example.com, our request is pointed toward the .com TLD.
- The resolver then makes a request to the .com TLD.
- The TLD server then responds with the IP address of the domain's nameserver, example.com.

- Lastly, the recursive resolver sends a query to the domain's nameserver.
- The IP address for example.com is then returned to the resolver from the nameserver.
- The DNS resolver then responds to the web browser with the IP address of the domain requested initially.

Once the 8 steps of the DNS lookup have returned the IP address for example.com, the browser is able to make the request for the web page:

- The browser makes a HTTP request to the IP address.
- The server at that IP returns the webpage to be rendered in the browser (step 10).



2 types of DNS queries:

Recursive query - In a recursive query, a DNS client requires that a DNS server (typically a DNS recursive resolver) will respond to the client with either the requested resource record or an error message if the resolver can't find the record.

Iterative query - in this situation the DNS client will allow a DNS server to return the best answer it can. If the queried DNS server does not have a match for the query name, it will return a referral to a DNS server authoritative for a lower level of the domain namespace. The DNS client will then make a query to the referral address. This process continues with additional DNS servers down the query chain until either an error or timeout occurs.

DNS records

What is a DNS record?

DNS records (aka zone files) are instructions that live in authoritative DNS servers and provide information about a domain including what IP address is associated with that domain and how to handle requests for that domain. These records consist of a series of text files written in what is known as DNS syntax. DNS syntax is just a string of characters used as commands that tell the DNS server what to do. All DNS records also have a 'TTL', which stands for time-to-live, and indicates how often a DNS server will refresh that record.

You can think of a set of DNS records like a business listing on Yelp. That listing will give you a bunch of useful information about a business such as their location, hours, services offered, etc. All domains are required to have at least a few essential DNS records for a user to be able to access their website using a domain name, and there are several optional records that serve additional purposes.

What are the most common types of DNS record?

DNS A record - The "A" stands for "address" and this is the most fundamental type of DNS record: it indicates the IP address of a given domain. For example, if you pull the DNS records of cloudflare.com, the A record currently returns an IP address of: 104.17.210.9.

A records only hold IPv4 addresses. If a website has an IPv6 address, it will instead use an "AAAA" record.

Here is an example of an A record:

example.com	record type:	value:	TTL
@	A	192.0.2.1	14400

The "@" symbol in this example indicates that this is a record for the root domain, and the "14400" value is the TTL (time to live), listed in seconds. The default TTL for A records is 14,400 seconds. This means that if an A record gets updated, it takes 240 minutes (14,400 seconds) to take effect.

The vast majority of websites only have one A record, but it is possible to have several. Some higher profile websites will have several different A records as part of a technique called round robin load balancing, which can distribute request traffic to one of several IP addresses, each hosting identical content.

DNS AAAA record - DNS AAAA records match a domain name to an IPv6 address. DNS AAAA records are exactly like DNS A records, except that they store a domain's IPv6 address instead of its IPv4 address.

IPv6 is the latest version of the Internet Protocol (IP). One of the important differences between IPv6 and IPv4 is that IPv6 addresses are longer than IPv4 addresses. The Internet is running out of IPv4 addresses, just as there are only so many possible phone numbers for a given area code. But IPv6 addresses offer exponentially more permutations and thus far more possible IP addresses.

As an example of the difference between IPv4 and IPv6 addresses, Cloudflare offers a public DNS resolver that anyone can use by setting their device's DNS to 1.1.1.1 and 1.0.0.1. These are the IPv4 addresses. The IPv6 addresses for this service are 2606:4700:4700::1111 and 2606:4700:4700::1001.

DNS AAAA record example

Here is an example of an AAAA record:

example.com	record type:	value:	TTL
@	AAAA	2001:0db8:85a3:0000: 0000:8a2e:0370:7334	14400

DNS CNAME record - A "canonical name" (CNAME) record points from an alias domain to a "canonical" domain. A CNAME record is used in lieu of an A record, when a domain or subdomain is an alias of another domain. All CNAME records must point to a domain, never to an IP address. Imagine a scavenger hunt where each clue points to

another clue, and the final clue points to the treasure. A domain with a CNAME record is like a clue that can point you to another clue (another domain with a CNAME record) or to the treasure (a domain with an A record).

For example, suppose blog.example.com has a CNAME record with a value of "example.com" (without the "blog"). This means when a DNS server hits the DNS records for blog.example.com, it actually triggers another DNS lookup to example.com, returning example.com's IP address via its A record. In this case we would say that example.com is the canonical name (or true name) of blog.example.com.

Example of a CNAME record:

blog.example.com	record type:	value:	TTL
@	CNAME	is an alias of example.com	32600

DNS MX record - A DNS 'mail exchange' (MX) record directs email to a mail server. The MX record indicates how email messages should be routed in accordance with the Simple Mail Transfer Protocol (SMTP, the standard protocol for all email).

Example of an MX record:

example.com	record type:	priority:	value:	TTL
@	MX	10	mailhost1.example.com	45000
@	MX	20	mailhost2.example.com	45000

The 'priority' numbers before the domains for these MX records indicate preference; the lower 'priority' value is preferred. The server will always try mailhost1 first because 10 is lower than 20. In the result of a message send failure, the server will default to mailhost2.

The email service could also configure this MX record so that both servers have equal priority and receive an equal amount of mail:

example.com	record type:	priority:	value:	TTL
@	MX	10	mailhost1.example.com	45000
@	MX	10	mailhost2.example.com	45000

This configuration enables the email provider to equally balance the load between the two servers.

DNS TXT record - The DNS 'text' (TXT) record lets a domain administrator enter text into the Domain Name System (DNS). Text is stored in the form of one or more strings within quotation marks. The TXT record was originally intended as a place for human-readable notes. However, now it is also possible to put some machine-readable data into TXT records. One domain can have many TXT records.

Example of a TXT record:

example.com	record type:	value:	TTL
@	TXT	"This is an awesome domain! Definitely not spammy."	32600

Today, two of the most important uses for DNS TXT records are email spam prevention and domain ownership verification, although TXT records were not designed for these uses originally.

DNS NS record - NS stands for 'nameserver,' and the nameserver record indicates which DNS server is authoritative for that domain (i.e. which server contains the actual DNS records). Basically, NS records tell the Internet where to go to find out a domain's IP address. A domain often has multiple NS records which can indicate primary and secondary nameservers for that domain. Without properly configured NS records, users will be unable to load a website or application.

Here is an example of an NS record:

example.com	record type:	value:	TTL
@	NS	ns1.exampleserver.com	21600

Note that NS records can never point to a canonical name (CNAME) record.

DNS SOA record- The DNS 'start of authority' (SOA) record stores important information about a domain or zone such as the email address of the administrator, when the domain was last updated, and how long the server should wait between refreshes.

All DNS zones need an SOA record in order to conform to IETF standards. SOA records are also important for zone transfers.

Example of an SOA record:

name	example.com
record type	SOA
MNAME	ns.primaryserver.com
RNAME	admin.example.com
SERIAL	1111111111
REFRESH	86400
RETRY	7200
EXPIRE	4000000
TTL	11200

The 'RNAME' value here represents the administrator's email address, which can be confusing because it is missing the '@' sign, but in an SOA record admin.example.com is the equivalent of admin@example.com.

What is a zone serial number?

In the DNS, a 'zone' is an area of control over namespace. A zone can include a single domain name, one domain and many subdomains, or many domain names. In some cases, 'zone' is essentially equivalent with 'domain,' but this is not always true.

A zone serial number is a version number for the SOA record. In the example above, the serial number is listed next to 'SERIAL.' When the serial number changes in a zone file, this alerts secondary nameservers that they should update their copies of the zone file via a zone transfer.

What are the other parts of an SOA record?

MNAME: This is the name of the primary nameserver for the zone. Secondary servers that maintain duplicates of the zone's DNS records receive updates to the zone from this primary server.
REFRESH: The length of time (in seconds) secondary servers should wait before asking primary servers for the SOA record to see if it has been updated.
RETRY: The length of time a server should wait for asking an unresponsive primary nameserver for an update again.
EXPIRE: If a secondary server does not get a response from the primary server for this amount of time, it should stop responding to queries for the zone.

Configure a DNS server on RHEL 9

1. Pre-requisites

- **System Requirements:**
  - RHEL 9 installed on your machine (or VM).
  - A stable internet connection.
  - Root or sudo privileges.

- **Important Details:**
  - **DNS Package:** `bind`
  - **Port:** 53 (TCP and UDP)
  - **Configuration Files:**
    - **Main config file:** `/etc/named.conf`
    - **Zone files directory:** `/var/named/`
    - **Zone files:** `/var/named/forward.zone`, `/var/named/reverse.zone`
    - **Logs:** `/var/log/messages`

2. Install the Required Package

1. Update your system:

```
dnf update -y
```

2. Install the BIND package:

```
dnf install bind bind-utils -y
```

1. bind: Primary DNS server package.
2. bind-utils: Includes utilities like dig, host, nslookup.

3. Configure the DNS Server

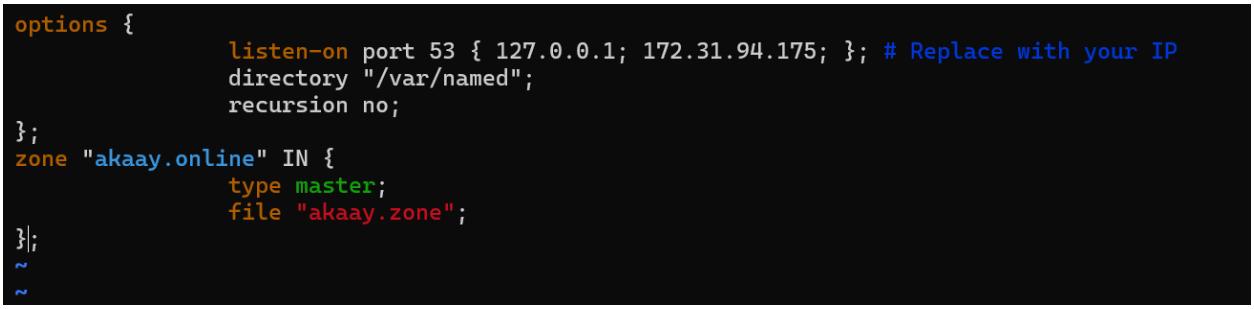
3.1. Edit the Main Configuration File

1. Open the main config file:

```
vim /etc/named.conf
```

2. Modify the following:

```
options {
        listen-on port 53 { 127.0.0.1; 172.31.94.175; }; # Replace
with your IP
        directory "/var/named";
        recursion no;
};
zone "vimnet.tech" IN {
        type master;
        file "akaay.zone";
};
```



4. Create Zone Files

1. Navigate to the `/var/named/` directory:

```
cd /var/named/
```

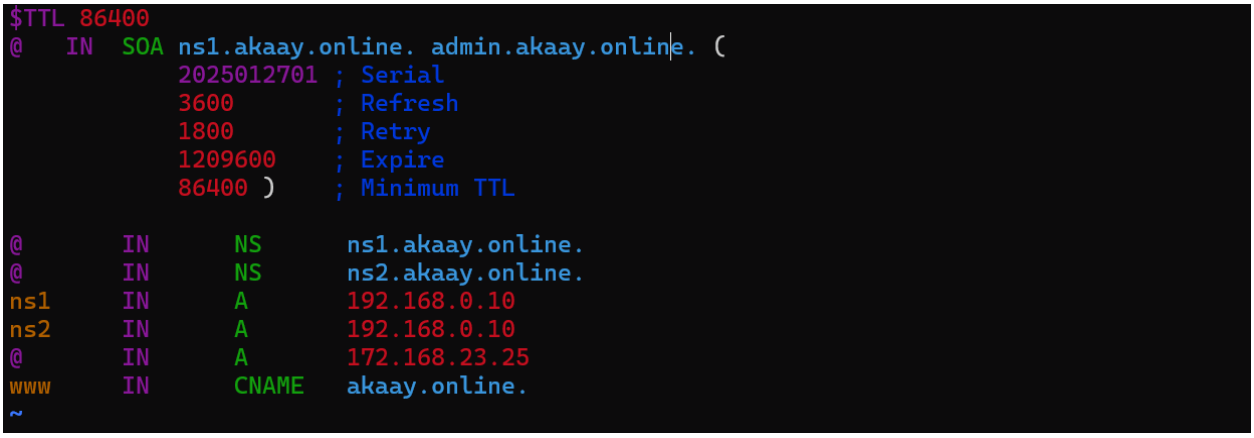
2. **Forward Zone File (akaay.zone)**: Create and edit the forward zone file:

```
vim /var/named/akaay.zone
```

Add the following:

```
$TTL 86400
@   IN  SOA ns1.akaay.online. admin.akaay.online. (
             2025012701 ; Serial
             3600       ; Refresh
             1800       ; Retry
             1209600    ; Expire
             86400 )    ; Minimum TTL

@       IN      NS      ns1.akaay.online.
@       IN      NS      ns2.akaay.online.
ns1     IN      A       192.168.0.10
ns2     IN      A       192.168.0.10
@       IN      A       172.168.23.25
www     IN      CNAME   akaay.online.
```



5. Set File Permissions

Ensure the zone files are readable by the BIND service:

```
chown root:named /var/named/akaay.zone

chmod 640 /var/named/akaay.zone
```

6. Start and Enable the DNS Service

1. Enable and start the named service:

```
systemctl enable named.service

systemctl start named.service
```

2. Check the service status:

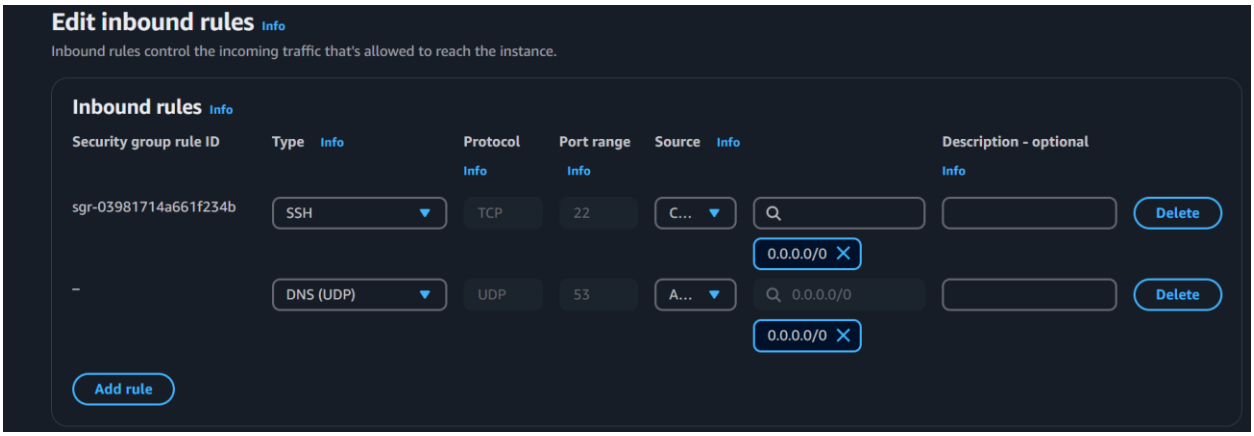
```
systemctl status named
```

3. Verify if port 53 is listening:

```
ss -tuln | grep 53
```

7. Open 53/upd port on firewall

If using aws instance then add 53/upd port on sercurity group



7. Test the DNS Server

7.1. On the Local Machine

Use the dig and **nslookup** commands to test your DNS server:

- Forward Lookup:

```
nslookup akaay.online 52.87.242.175
```

```
PS C:\Users\Dell> nslookup akaay.online 52.87.242.175
Server:  UnKnown
Address:  52.87.242.175

Name:    akaay.online
Address:  172.168.23.25

PS C:\Users\Dell>
```

Since your domain is registered with **Hostinger**, you need to update the DNS settings so that your domain resolves correctly using your own **custom nameservers**. Follow these steps to configure your domain on Hostinger.

1. Create Child Nameservers in Hostinger

Child nameservers (also known as glue records) allow your domain to act as its own nameserver.

Step 1: Log in to Hostinger

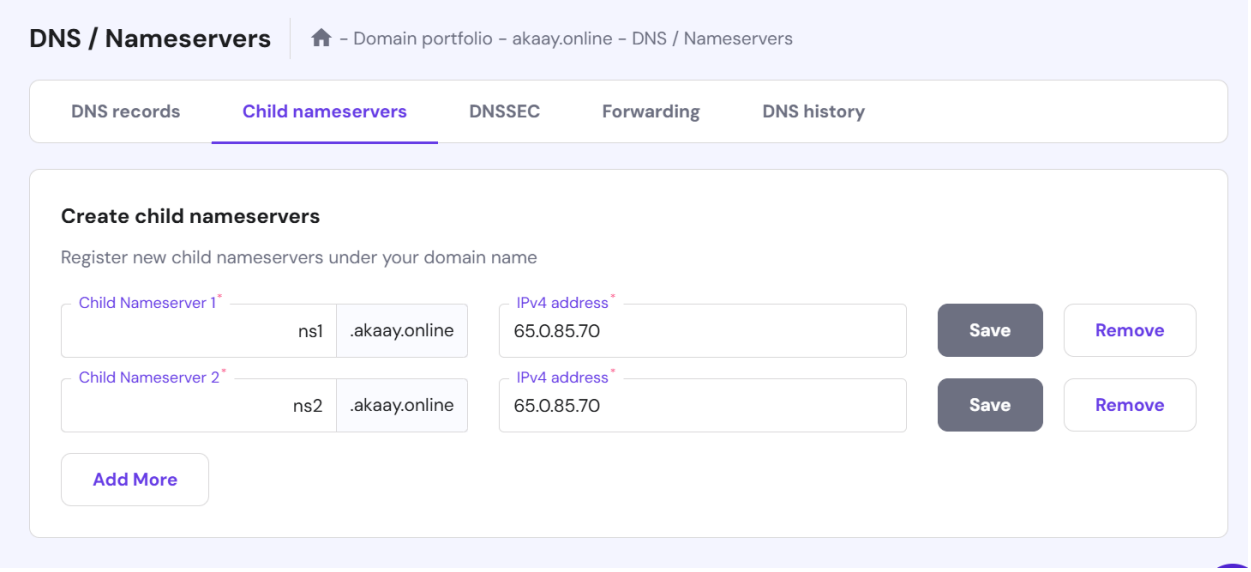
1. Go to Hostinger Login
2. Sign in to your **Hostinger account**.

Step 2: Navigate to the Domain Settings

1. Click on **Domains** from the dashboard.
2. Select the **domain** you want to modify.

Step 3: Create Child Nameservers

1. Look for the option **Child Nameservers / Glue Records**.
2. Add **two child nameservers** with the IP address of your DNS server (the same IP where BIND is running).



2. Change the Nameservers for Your Domain

Now, update the nameservers to use your custom ones.

Step 1: Open the Nameserver Settings

1. In your **domain settings**, look for the option **Change Nameservers**.
2. Select **Custom Nameservers** instead of Hostinger's default ones.

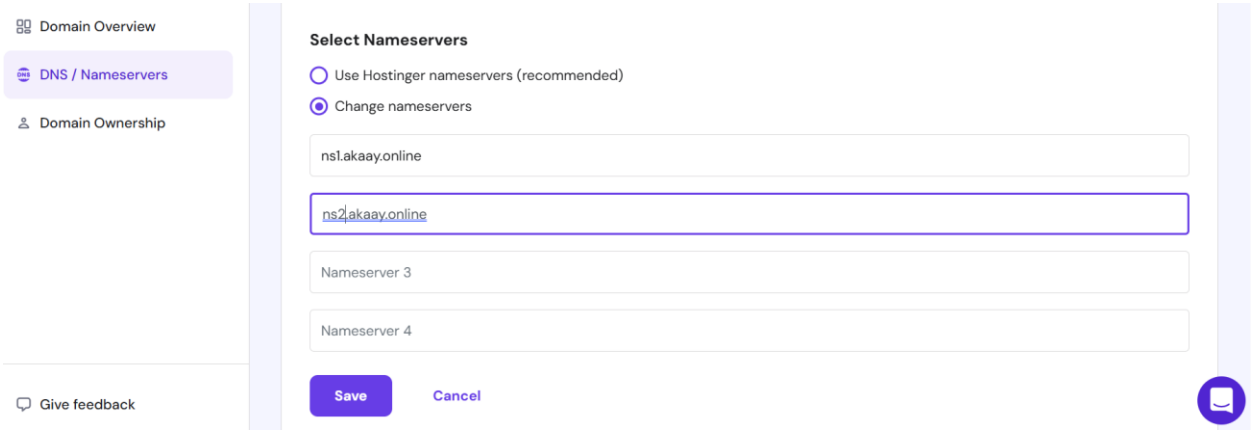
Step 2: Enter the New Nameservers

Replace the existing nameservers with the child nameservers you created:

- **ns1.example.com**
- **ns2.example.com**

3. Click **Save Changes**.

⚠️ **DNS Propagation Time**: The changes can take up to **24-48 hours** to propagate globally.



Now Let's Configure Secondary DNS Server for akaay.online

To configure a secondary DNS server, first, we need to make a change in the master DNS server's configuration file. Let's see what needs to be done.

1. Configure the Secondary (Slave) DNS Server

1.1. Edit the Main Configuration File (On master DNS Server)

1. Open the main config file:

```
vim /etc/named.conf
```

2. Modify the following: For zone transfer, **allow-transfer** needs to be set in the Master DNS.

```
options {
        listen-on port 53 { 127.0.0.1; 172.31.94.175; }; # Replace
with your IP
        directory "/var/named";
        recursion no;
};
zone "vimnet.tech" IN {
        type master;
        file "akaay.zone";
        allow-transfer { 13.232.100.120; }; #this secondary DNS server
IP
};
```

3. Restart named.service

```
systemctl resatrt named.service
```

1.2. Install bind & bind-utils package on secondary (slave) DNS server

1. Update your system:

```
dnf update -y
```

2. Install the BIND package:

```
dnf install bind bind-utils -y
```

1.3. Edit the Main Configuration File (On Secondary DNS Server)

1. Open the main config file:

```
vim /etc/named.conf
```

2. Modify the following:

```
options {
        listen-on port 53 { 127.0.0.1; 172.31.94.175; }; # Replace
with your IP
directory "/var/named";
recursion no;
};
zone "vimnet.tech" IN {
type slave;
file "/var/named/slaves/akaay.zone";
masters { 13.232.100.120; }; # this is master DNS server IP
};
```

3. Restart named.service

```
systemctl resatrt named.service
```

Open 53/tcp port on Master DNS server - The slave (secondary) DNS server uses a TCP connection to transfer the zone file from the master, because in zone transfers, there are large data packets that are difficult to handle using UDP.

Open 53/udp port on slave DNS server

Note: - Update Slave IP address on Hostinger Panel

The Secondary DNS server automatically syncs (transfers) the zone file from the Master DNS server. When we define the zone type as **"slave"** in **named.conf** and specify the Master DNS server's IP address, the Secondary DNS server fetches data from the Master through **AXFR (zone transfer)**.

Now, to test the Secondary DNS server, **shut down the Master DNS server** and then run the **nslookup** command to query **vimnet.tech**.

- **If you get a response**, it means the **Slave DNS is working**.
- **If not**, check the **DNS logs** using the following commands:

```
journalctl -u named -f
```

Or

```
tail -f /var/log/named/named.log
```

Thank Your