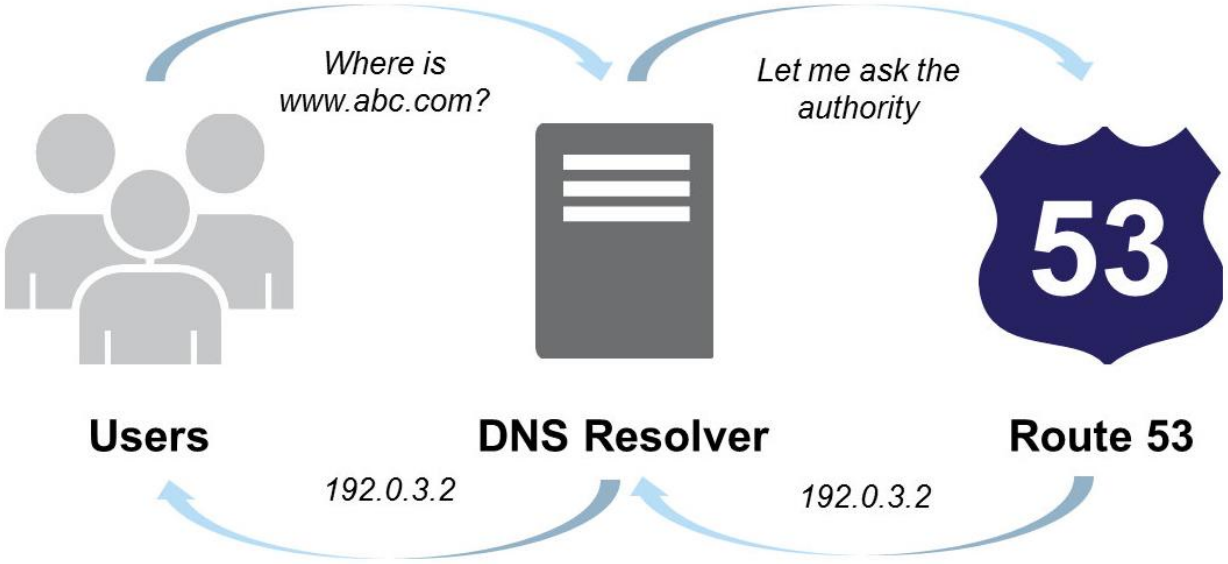
AWS Route 53

Amazon Route 53 is a highly available and scalable cloud Domain Name System (DNS) web service.

It is basically designed for developers and corporate to route the end users to Internet applications by translating human-readable names like www.sanoj.homes into the numeric IP addresses like 192.0.1.1 that computers use to connect to each other.

You cannot use Amazon Route 53 to connect your on-premises network with AWS Cloud



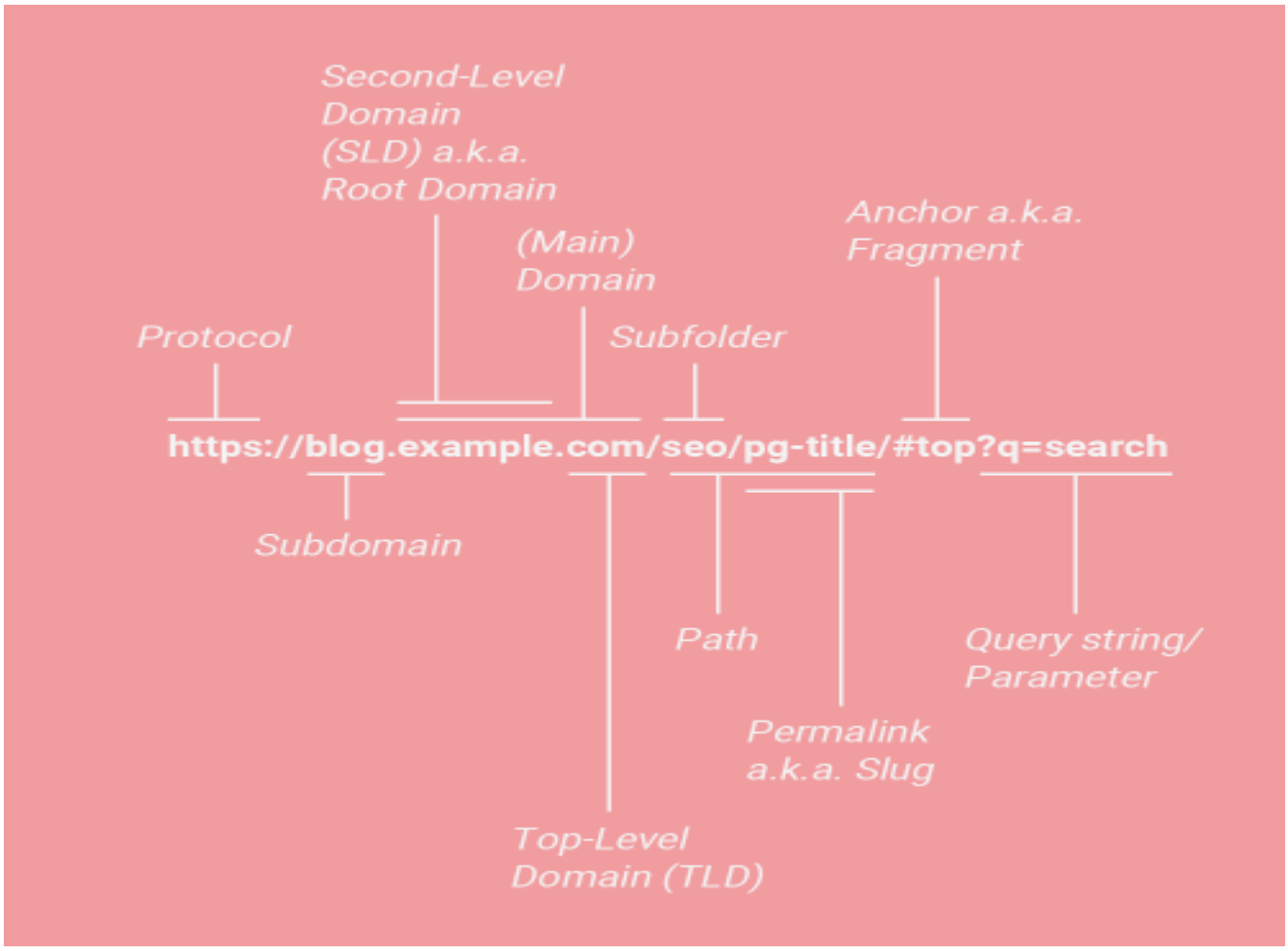
- Amazon Route53 is a highly available and scalable cloud Domain Name System(DNS) web service.
  - It is an extremely reliable and cost-effective way to route end users to Internet applications by translating names like www.sanoj.homes into the numeric IP address like 192.168.0.1 that computers use to connect to each other.
  - Amazon Route 53 is fully compliant with IPv6 as well
  - You can use Amazon Route 53 to configure DNS health check to route traffic to the healthy endpoints. Helps with Load Balancing as well.
  - Independently monitor the health of your application and its endpoints.
  - Amazon Route 53 Traffics flow makes it easy for you to manage traffic globally. It manages traffic globally through a variety of routing types, including Latency Based Routing, Geo DNS, Geoproximity and Weighted Round Robin.
  - Amazon Route 53 also offers Domain Name Registration.
  - Amazon Route 53 will automatically configure DNS settings for your domains.

Let's Understand the first URL: URL stands for **Uniform Resource Locator**. A whole URL tells your browser which web address you want to go for and then opens that page.

A typical website URL could look like this:

https:// www .example-website .co. uk /about-us /welcome.jpg

Let's break things down a little and explain what each part means.



HTTP/ HTTPS – This part of the URL stands for Hypertext Transfer Protocol and is the protocol used to move information across the internet.

It is a part of the Internet protocol suite and it helps to define the commands and services used for transmitting webpage data.

HTTP uses a server-client model as does HTTPS. HTTPS is the same as HTTP, but it has an added S for Security and provides web users with an extra layer of protection while browsing. It is especially important if you are using a website that will carry out any kind of transaction or save your personal data.

WWW – WWW stands for World Wide Web, and it's used mostly as a prefix. However, it does indicate that a given website uses HTTP to communicate.

HTTP vs WWW

The main difference between WWW and HTTP is that they refer to different concepts. Simply put, HTTP is the protocol that enables communication online, transferring data from one machine to another.

WWW is the set of linked hypertext documents that can be viewed on a web browser (such as Google Chrome, Firefox, and more).

A major similarity, though, is that both HTTP and WWW are used in website URLs.

HTTP vs. WWW in URLs

HTTP vs. WWW in URLs

Within the URL parameters, it's possible to remove HTTP or WWW from your domain registrar. However, the situations in which you would remove one of these elements depends on a few factors.

As we stated above, WWW is a prefix used to indicate that a website is using HTTP to communicate. In fact, you can mix and match prefixes, for example, http://example.com or www.example.com.

These different URLs have enough information to communicate between the browser and server, so both will work without any interruptions.

So, what if you do choose to use http://example.com as your website URL, but users type in WWW where it isn't necessary? In most cases, the user will be automatically redirected to your non-WWW domain. This means you can set up a non-WWW URL address without worry.

Do You Need WWW in URLs?

It's not necessary to use WWW in URLs. It exists for just one purpose—to identify the web address. This is not the case with other important URL signifiers, such as a File

Transfer Protocol (FTP) server (ftp) or news server (news). As such, WWW may be classified as a subdomain of a larger website.

In most cases, WWW serves no technical purpose. You can create a custom domain without its presence, and the website will still operate as any website should.

So, why is WWW used so frequently? The use of WWW has been around since the creation of the internet, and its widespread use as a subdomain was largely accidental.

The first web server was nxoc01.cern.ch. When publishing the website, the creators fully intended for info.cern.ch to be their home page, and WWW, as such, was excluded. The Domain Name System (DNS) records for the server were never switched, and the use of WWW became an unintentional standard practice.

As mentioned, it is possible to create a custom domain name without WWW included. Though, there are some considerations to keep in mind.

Domain or Web Address without WWW

In most cases, a user will not have to type WWW to view your web address or domain. However, if you've implemented WWW to differentiate between subdomains, it's important to ensure that your site is configured to provide the appropriate redirects to users.

For some websites, the addition of the WWW in www.example.com may cause a redirect to example.com. For others, it may be two separate pages on the same domain. Your web hosting provider should be able to help you set up to ensure the correct redirects are in place.

Example-website – This is your domain name. Typically, the name of the company, or something related to the service they provide. Here you'll need to find an available name if you're making a new website.

.Co.Uk – This part of the web address typically signifies the location of where the company is based. The .co.uk address example above would be from the UK.

But there are many others available for almost all countries. Other popular examples are .gov addresses which are typically for government websites. Or things like .biz or .io which come in and out of fashion for businesses.

Normally a US-based website will use .com. This is also a standard choice for international businesses that want to present a more global face.

/about-us/ – This directs us to a specific page on your website. In this case, the about us page. Many websites follow a similar structure and naming convention, but there is no need to do so.

Welcome.jpg – This part of the URL indicates that we are looking at a file that lives inside the about us page. In this case, it is an image, which is shown by the fact we are looking at a .jpg file type. Many file types can be hosted on a website, from downloadable files to images, videos, and PDFs.

What is DNS (Domain Name System)?

A DNS is a computer server that contains a database of many IP addresses and their associated domain names.

It serves to translate a requested domain name into an IP address so that the computer knows which IP address to connect to for the requested contents. The Internet is a network of connected computers, and they communicate with each other through IP addresses.

A DNS plays an important role in helping us to conveniently use the Internet and it is one of the most essential foundations of the Internet as we know it today.

It is much easier for us to remember a domain name, sanoj.homes, rather than a string of numbers, 192.168.0.1 (IP address).

Both the domain name and the DNS are extremely important, and they work together to make this possible

A great example is to see a DNS as a phone book, which matches a name to a telephone number. You can search for the name you want and find the corresponding phone number.

It is also a similar concept to your smartphone's contact list, which will match a contact name to a phone number. Remembering domain names is easier for us than to remember a string of numbers.

DNS helps us to do this by matching domain names to IP addresses and simplifies our web surfing experience significantly.

How Does a DNS Work?

A DNS starts working immediately after a user enters a domain name in the address bar of a browser.

It will search through the Internet to find the IP address that is associated with the entered domain name. After successfully identifying the IP address, it then guides the user's browser to connect to it, which will then serve the requested website contents.

The process happens very quickly with little delay and the user will be on his requested website almost immediately. However, in the background, a DNS has executed many processes.

The first step that a DNS does is to send a DNS query to several other DNS servers. A DNS is not just a single server responding to over billions of domain name requests, instead, it is distributed globally across a network of DNS, which stores the IP address directory in a distributed manner.

All the DNS servers work together to attend to the billions of domain name requests worldwide. The reason behind this is to cut down the time for users to get a response to their requests.

If a user is looking for a specific site and there is only one DNS server to process it, then it will take significantly longer to search through the millions of records in the directory.

What if at the same time there are also millions, if not billions of users who are also doing the same? That is going to take a long time, and the users' browsing experience will definitely be affected negatively.

Therefore, DNS is set up to work collaboratively across several servers to provide the best browsing experience to users. When a website address is entered by a user in an Internet browser, a DNS query is initiated and a DNS server sends the query to several other DNS servers, each tasked with translating a different part of the domain name the user entered.

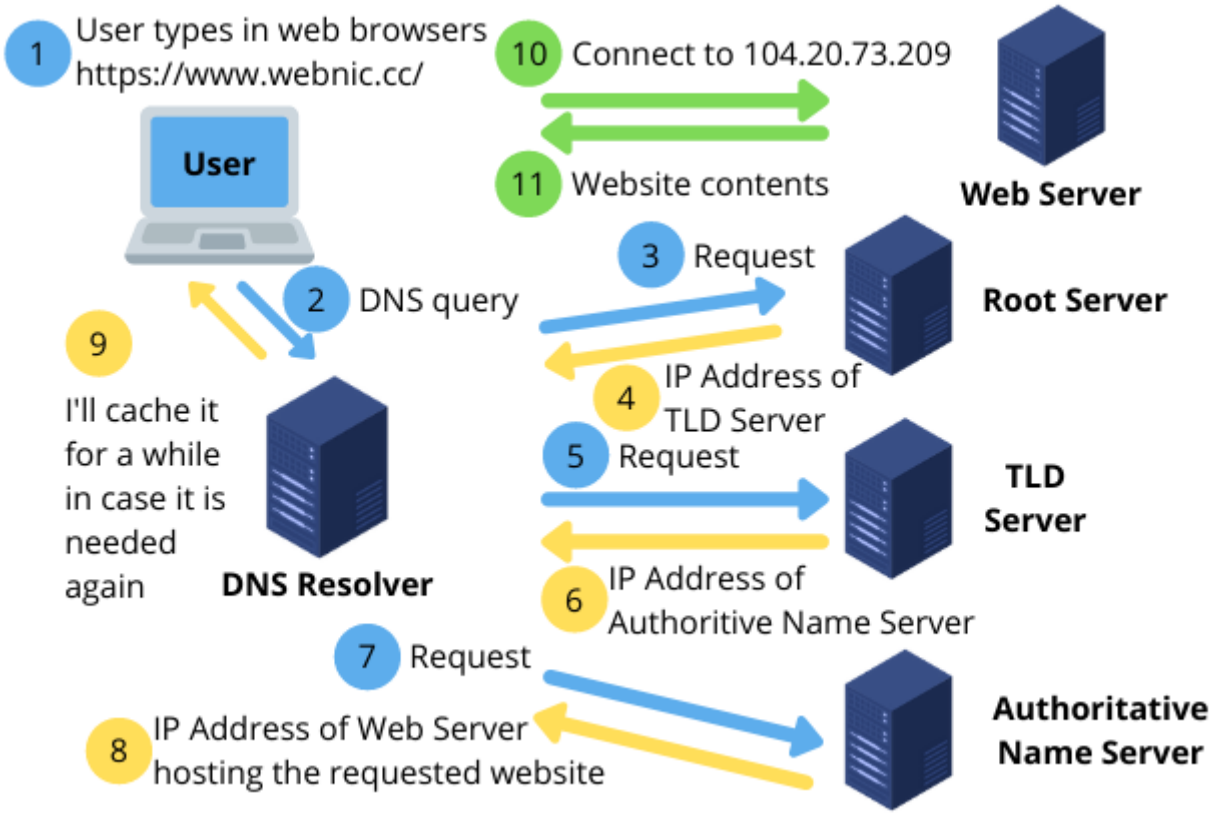
There are mainly four servers which work together to translate the website address into a computer-readable IP address, namely the DNS Resolver server, the root server, the Top-level Domain (TLD) server and the Authoritative Name Server.

The DNS Resolver is the server that does most of the process of translating a domain name to an IP address. It receives the DNS query and in turn acts like a client to query the three other DNS servers mentioned above to translate the domain name.

It first queries the root server, and the root server responds to the query by returning the IP address of a TLD server (like .com, .net, .org etc.). A TLD server stores the information for its domains and will return the IP address of the Authoritative Name Server to the DNS Resolver.

This is where the requested website is located. It then returns the actual IP address of the requested website to the DNS Resolver, which in turn responds to the initial DNS query with the actual IP address.

Check out the illustration below to have a clear picture of the step-by-step processes of how a DNS work.



Another point worth mentioning is point number 9 as shown in the illustration.

The DNS Resolver will perform a caching function to cache the data for a limited time after it has retrieved the correct IP address for a given website.

The purpose of doing so is that in case the user requests for the same domain name again, it can immediately direct the user to the right web server without having to do the entire process of translating the domain name again.

It is also beneficial in the sense that if there are other users who request the same domain name, their request can be processed instantly, and they can enjoy a great browsing experience.

Records in Route 53?

Address (A) Format: is an IPv4 address in the dotted decimal notation e.g. 192.168.0.1 this is the most fundamental type of DNS record, it indicates the IP address of a given domain.

Authentication, authorization, accounting, auditing(AAAA) Format: is an IPv6 address in colon-separated hexadecimal format.

Canonical Name (CNAME): A Canonical Name (CNAME) Record is used in the Domain Name System (DNS) to create an alias from one domain name to another domain name.

A common example is the www subdomain which is provided as an alias to the root domain name - users accessing “www.example.com” are referred to the root domain (or DNS zone apex) “example.com”.

A few common uses of CNAME records are:

- Providing a separate hostname for specific network services, such as email or FTP, and pointing that hostname to the root domain

- Many hosted services provide a subdomain for each customer on the service provider’s domain (e.g. company.hostname.com), and use CNAME to point to the customer’s domain (www.company.com).

- Registering the same domain in several countries and pointing the country versions to the main “.com” domain.
- Pointing from several websites owned by the same organization to a primary website

How the DNS System Handles CNAME Records

The DNS records in the above example would look like this:

CNAME from subdomain to parent domain

NAME TYPE VALUE
---
www.example.com. CNAME example.com.
example.com. A →192.162.100.100

The second record is an A record which translates the human-readable domain name “example.com” to an IP address.

DNS Resolution Process for CNAME Records

1. A DNS client (such as a browser or network device) requests the address www.example.com, and a DNS request is created.
2. A DNS resolver receives the request and finds the Authoritative Name Server that holds the DNS Zone file with DNS records for the “example.com” domain.
3. The DNS request is resolved and the CNAME record is returned to the client.
4. The client understands www.example.com is only an alias for the real address, “example.com”, and issues a new DNS query for “example.com”
5. The process is repeated, and the resolver returns the A record for “example.com”, containing the IP address.
6. The DNS client now connects to “example.com” using its IP address.

Mail exchange(MX):

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 emails). Like CNAME records, an MX record must always point to another domain.

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. As a 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.

What is the process of querying an MX record?

Message transfer agent (MTA) software is responsible for querying MX records. When a user sends an email, the MTA sends a DNS query to identify the mail servers for the email recipients. The MTA establishes an SMTP connection with those mail servers, starting with the prioritized domains (in the first example above, mailhost1).

Pointer Record (PTR): A PTR record is an important email security tool that prevents spam, It's what your mail server uses to determine who's actually sending an email message.

A PTR records also known as a pointer Record is a piece of information (a record) that is attached to an email message. The purpose of the PTR records is to verify that the sender matches the IP address it claims to be using.

This email ID check process is also known as reverse DNS lookup.

It’s the reverse of the forward DNS lookup process that browsers use to convert a domain name to a numerical address(IP Address).

How are DNS PTR records stored?

In IPv4:

While DNS A records are stored under the given domain name, DNS PTR records are stored under the IP address — reversed, and with ".in-addr.arpa" added.

For example, the PTR record for the IP address 192.0.2.255 would be stored under "255.2.0.192.in-addr.arpa".

"in-addr.arpa" has to be added because PTR records are stored within the .arpa top-level domain in the DNS. .arpa is a domain used mostly for managing network infrastructure, and it was the first top-level domain name defined for the Internet.

(The name "arpa" dates back to the earliest days of the Internet: it takes its name from the Advanced Research Projects Agency (ARPA), which created ARPANET, an important precursor to the Internet.)

in-addr.arpa is the namespace within .arpa for reverse DNS lookups in IPv4.

In IPv6:

IPv6 addresses are constructed differently from IPv4 addresses, and IPv6 PTR records exist in a different namespace within .arpa. IPv6 PTR records are stored under the IPv6 address, reversed and converted into four-bit sections (as opposed to 8-bit sections, as in IPv4), plus ".ip6.arpa".

What are some of the main uses for PTR records?

PTR records are used in reverse DNS lookups; common uses for reverse DNS include:

Anti-spam: Some email anti-spam filters use reverse DNS to check the domain names of email addresses and see if the associated IP addresses are likely to be used by legitimate email servers.

Troubleshooting email delivery issues: Because anti-spam filters perform these checks, email delivery problems can result from a misconfigured or missing PTR record. If a domain has no PTR record, or if the PTR record contains the wrong domain, email services may block all emails from that domain.

Logging: System logs typically record only IP addresses; a reverse DNS lookup can convert these into domain names for logs that are more human-readable.

Name Server (NS): Name Server This is the actual Domain Name Servers.

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.

What is a nameserver?

A nameserver is a type of DNS server. It is the server that stores all DNS records for a domain, including A records, MX records, or CNAME records.

Almost all domains rely on multiple nameservers to increase reliability: if one nameserver goes down or is unavailable, DNS queries can go to another one. Typically, there is one primary nameserver and several secondary nameservers, which store exact copies of the DNS records in the primary server.

Updating the primary nameserver will trigger an update of the secondary nameservers as well.

When multiple nameservers are used (as in most cases), NS records should list more than one server. Learn more about DNS servers.

When should NS record

Domain administrators should update their NS records when they need to change their domain nameservers. For instance, some cloud providers provide nameservers and require their customers to point to them.

Admins may also wish to update their NS records if they want a subdomain to use different nameservers. In the example above, the nameserver for example.com is ns1.exampleserver.com. If the example.com admin wanted blog.example.com to resolve via ns2.exampleserver.com instead, they could set this up by updating the NS record.

When NS records are updated, it may take several hours for the changes to be replicated throughout the DNS.

SOA(start of authority): Points towards Primary Name Server.

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.

What is a zone transfer?

A DNS zone transfer is the process of sending DNS record data from a primary nameserver to a secondary nameserver. The SOA record is transferred first. The serial number tells the secondary server if its version needs to be updated. Zone transfers take place over the TCP protocol.

SRV(Service): The DNS "service" (SRV) record specifies a host and port for specific services such as voice over IP (VoIP), instant messaging, and so on. Most other DNS records only specify a server or an IP address, but SRV records include a port at that IP address as well. Some Internet protocols require the use of SRV records in order to function.

What is a port?

In networking, ports are virtual places that designate what processes network traffic goes to within a computer. Ports allow computers to easily differentiate between different kinds of traffic: VoIP streams go to a different port than email messages, for instance, even though both reach a computer over the same Internet connection. Much like IP addresses, all ports are assigned a number.

Certain Internet protocols, such as IMAP, SIP, and XMPP, need to connect to a specific port in addition to connecting with a specific server. SRV records are how a port can be specified within the DNS.

What goes in an SRV record?

An SRV record contains the following information. Here, we list example values for each field.

service	XMPP
proto*	TCP
name**	example.com
TTL	86400
class	IN
type	SRV
priority	10
weight	5
port	5223
target	server.example.com

*Short for "protocol," as in transport protocol.
**Domain name.

However, SRV records are actually formatted in this way:

_service._proto.name. TTL class type of record priority weight port target.

So our example SRV record would actually look like:

_xmpp._tcp.example.com. 86400 IN SRV 10 5 5223 server.example.com.

In the above example, "_xmpp" indicates the type of service (the XMPP protocol) and "_tcp" indicates the TCP transport protocol, while "example.com" is the host, or the domain name. "Server.example.com" is the target server and "5223" indicates the port within that server.

SRV records must point to an A record (in IPv4) or an AAAA record (in IPv6). The server name they list cannot be a CNAME. So "server.example.com" must lead directly to an A or AAAA record under that name.

What is the difference between priority and weight in SRV records?

SRV records indicate the "priority" and "weight" of the various servers they list. The "priority" value in an SRV record enables administrators to prioritize one server that supports the given service over another.

A server with a lower priority value will receive more traffic than other servers. However, the "weight" value is similar: a server with a higher weight will receive more traffic than other servers with the same priority.

The main difference between them is that priority is looked at first. If there are three servers, Server A, Server B, and Server C, and they have respective priorities of 10, 20, and 30, then their "weight" does not matter. The service will always query Server A first.

But suppose Servers A, B, and C all have a priority of 10 — how will a service choose between them? This is where weight becomes a factor: if Server A has a "weight" value of 5 and Servers B and C have a "weight" value of 3 and 2, Server A will receive the most traffic, Server B will receive the second-most traffic, and Server C the third-most.

SPF(sender policy framework): A sender policy framework (SPF) record is a type of DNS TXT record that lists all the servers authorized to send emails from a particular domain.

A DNS TXT (“text”) record lets a domain administrator enter arbitrary text into the Domain Name System (DNS). TXT records were initially created for the purpose of including important notices regarding the domain, but have since evolved to serve other purposes.

SPF records were originally created because the standard protocol used for email — the Simple Mail Transfer Protocol (SMTP) — does not inherently authenticate the “from” address in an email.

This means that without SPF or other authentication records, an attacker can easily impersonate a sender and trick the recipient into taking action or sharing information they otherwise would not.

Think of SPF records like a guest list that is managed by a door attendant. If someone is not on the list, the door attendant will not let them in.

Similarly, if an SPF record does not have a sender’s IP address or domain on its list, the receiving server (door attendant) will either not deliver those emails or mark them as spam.

SPF records are just one of many DNS-based mechanisms that can help email servers confirm whether an email comes from a trusted source. Domain-based Message

Authentication Reporting and Conformance (DMARC) and DomainKeys Identified Mail (DKIM) are two other mechanisms used for email authentication.

It is worth noting that, at one point, SPF records had a dedicated DNS record type. The dedicated record type has since been deprecated and only TXT records are to be used.

How does a mail server check an SPF record?

Mail servers go through a relatively simple process when checking an SPF record:

- Server One sends an email. Its IP address is 192.0.2.0 and the return-path the email uses is email@returnpath.com. (A return-path address is different from the "from" address and is used specifically for collecting and processing bounced messages.)

- The mail server that is receiving the message (Server Two) takes the return-path domain and searches for its SPF record.

- If Server Two finds an SPF record for the return-path's domain, it searches the SPF record for Server One's IP address in its list of authorized senders.

- If the IP address is listed in the SPF record, the SPF check passes and the email will go through. If the IP address is not listed in the SPF record, the SPF check fails. In this case, the email will be rejected or marked as spam.

What does an SPF record look like?

SPF records must follow certain standards in order for the server to understand how to interpret its contents. Here is an example of the core components of an SPF record:

v=spf1 ip4=192.0.2.0 ip4=192.0.2.1 include:examplesender.email -all

This example lets the server know what type of record this is, states the approved IP addresses and a third-party for this domain, and tells the server what to do with non-compliant emails. Let's break down how the individual components accomplish this:

- *v=spf1* tells the server that this contains an SPF record. Every SPF record must begin with this string.

- Then comes the "guest list" portion of the SPF record or the list of authorized IP addresses. In this example, the SPF record is telling the server that *ip4=192.0.2.0* and *ip4=192.0.2.1* are authorized to send emails on behalf of the domain.

- *include:examplesender.net is* an example of the include tag, which tells the server what third-party organizations are authorized to send emails on behalf of the domain. This tag signals that the content of the SPF record for the included domain (examplesender.net) should be checked and the IP addresses it contains should also be considered authorized. Multiple domains can be included within an SPF record but this tag will only work for valid domains.

- Finally, *-all* tells the server that addresses not listed in the SPF record are not authorized to send emails and should be rejected.

- Alternative options here include *~all*, which states that unlisted emails will be marked as insecure or spam but still accepted, and, less commonly, *+all*, which signifies that any server can send emails on behalf of your domain.

While the example used in this article is fairly straightforward, SPF records can certainly be more complex. Here are just a few things to keep in mind to ensure SPF records are valid:

- There cannot be more than one SPF record associated with a domain.

- The record must end with the all component or include a redirect: component (which indicates that the SPF record is hosted by another domain).

- An SPF record cannot contain uppercase characters.

Why are SPF records used?

There are many reasons domain operators use SPF records:

- **Preventing attacks:** If emails are not authenticated, companies and email recipients are at risk for phishing attacks, spam emails, and email spoofing. With SPF records, it is harder for attackers to imitate a domain, reducing the likelihood of these attacks.

- **Improving email deliverability:** Domains without a published SPF record may have their emails bounce or be marked as spam. Over time, bounced emails or emails marked as spam can hurt a domain's ability to reach their audience's inboxes, compromising efforts to communicate with customers, employees, and other entities.

- **DMARC compliance:** DMARC is an email validation system that helps ensure that emails are sent only by authorized users. DMARC policies dictate what servers should do with emails that fail SPF and DKIM checks. Based on the

DMARC policy instructions, those emails will either be marked as spam, rejected, or delivered as normal. Domain administrators receive reports about their email activity that help them make adjustments to their policy.

TXT(Text): The DNS 'text' (TXT) record lets a domain administrator enter text into the Domain Name System (DNS). 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.

What kind of data can go in a TXT record?

The original RFC only indicates that 'text strings' go in the 'value' field of a TXT record. This could be any text that an administrator wants to associate with their domain.

Most DNS servers will put a limit on how big TXT records can be and how many records they can store, so administrators cannot use TXT records for large amounts of data.

What is the official format for storing data in a TXT record?

In 1993, the Internet Engineering Task Force (IETF) defined a format for storing attributes and their corresponding values within the 'value' field of TXT records. The format was simply the attribute and the value contained within quotation marks (") and separated by an equal sign (=), such as:

"attribute=value"

RFC 1464, the 1993 document that defines this format, includes these examples:

host.widgets.com	record type:	value:
@	TXT	"printer=lpr5"

sam.widgets.com	record type:	value:
@	TXT	"favorite drink=orange juice"

However, this definition was considered experimental, and in practice it is not often adopted. Some DNS administrators follow their own formats within TXT records, if they make use of TXT records at all. TXT records may also be formatted in a specific way for certain uses described below — for instance, DMARC policies have to be formatted in a standardized way.

How do TXT records help prevent email spam?

Spammers often try to fake or forge the domains from which they send their email messages. TXT records are a key component of several different email authentication methods that help an email server determine if a message is from a trusted source.

Common email authentication methods include Domain Keys Identified Mail (DKIM), Sender Policy Framework (SPF), and Domain-based Message Authentication, Reporting & Conformance (DMARC). By configuring these records, domain operators can make it more difficult for spammers to spoof their domains and can track attempts to do so.

SPF records: SPF TXT records list all the servers that are authorized to send email messages from a domain.

DKIM records: DKIM works by digitally signing each email using a public-private key pair. This helps verify that the email is actually from the domain it claims to be from. The public key is hosted in a TXT record associated with the domain. (Learn more about public key encryption.)

DMARC records: A DMARC TXT record references the domain's SPF and DKIM policies. It should be stored under the title _dmarc.example.com. with 'example.com' replaced with the actual domain name. The 'value' of the record is the domain's DMARC policy (a guide to creating one can be found here).

How do TXT records help verify domain ownership?

While domain ownership verification was not initially a feature of TXT records, this approach has been adopted by some webmaster tools and cloud providers.

By uploading a new TXT record with specific information included, or editing the current TXT record, an administrator can prove they control that domain. The tool or cloud provider can check the TXT record and see that it has been changed as requested. This is somewhat like when a user confirms their email address by opening and clicking a link sent to that email, proving they own the address.

TTL(Time To LIVE): TTL — which, as we've mentioned, stands for "Time to Live" — is a setting that determines how long your data (in packet form) is valid and available from within a network before the router clears it.

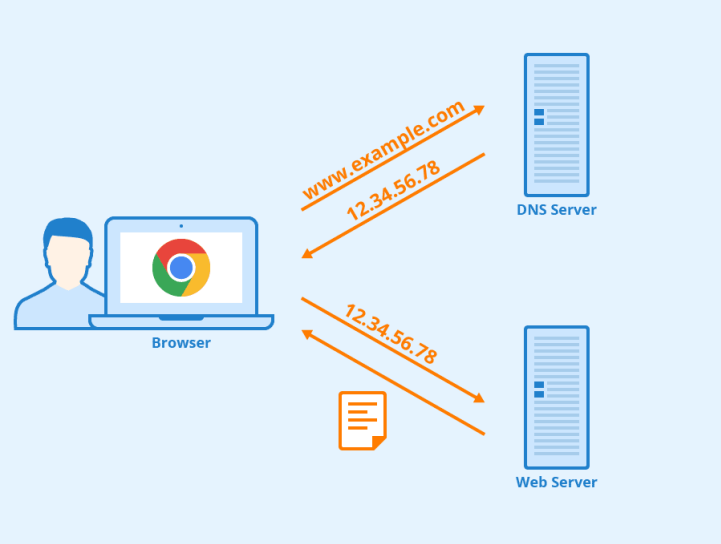
We can also refer to this time as "hops," which is the number of times it bounces between different routers.

Once the TTL expires, the router will need to retrieve the information again, along with its updates.

What Is TTL in DNS?

DNS servers act as a bridge between web servers and domain names. When you type in a domain name like "sanoj.homes", servers cannot interpret this information. They read data in numbers known as IP addresses.

So DNS facilitates the conversion between domain names and IP addresses, and enables users to access websites.



To understand the relationship between DNS servers and TTL, we first need to look at the cache. In this context, the cache is a storage for the conversion of your website's domain name into its IP address.

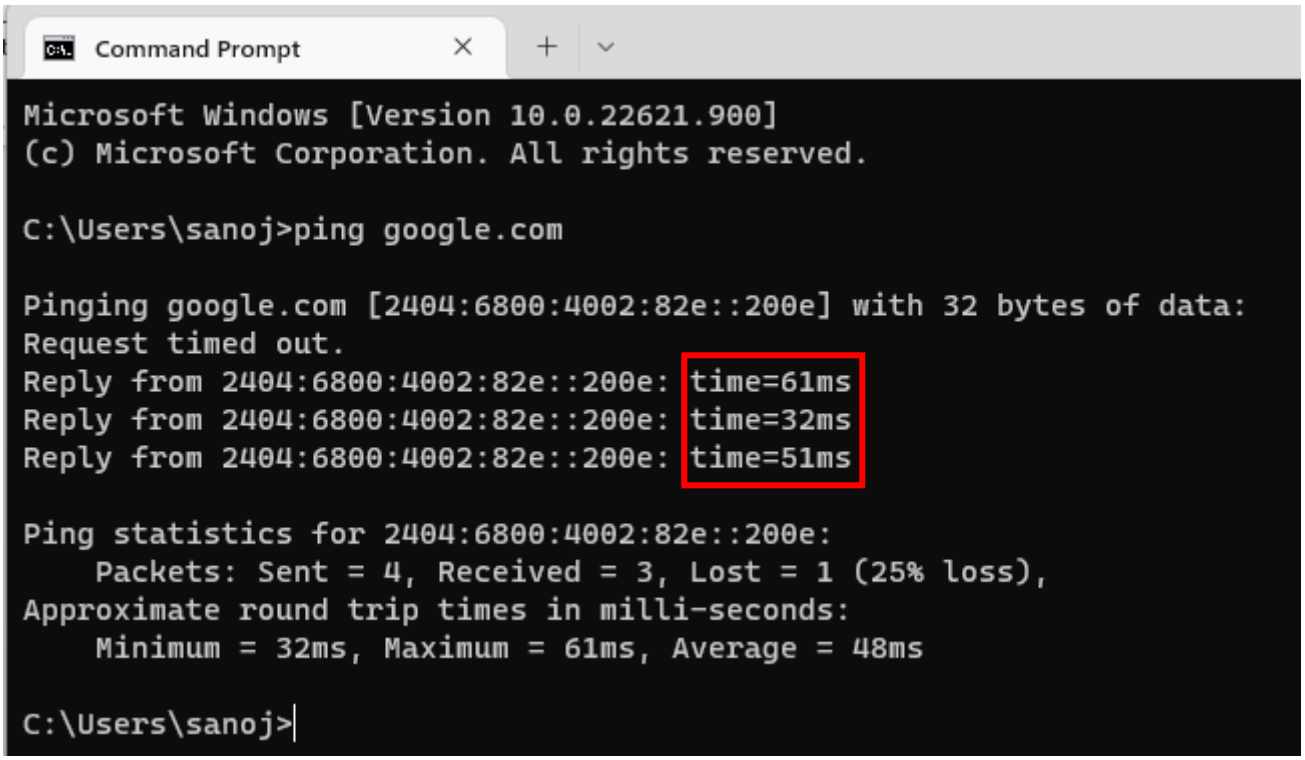
Every time a user wants to access your website, this conversion needs to happen. If the conversion is stored in the cache, the connection can happen more quickly because there is a DNS record. In fact, the server can pull up the record almost instantaneously.

In this context, TTL determines how long a DNS server will hold onto this DNS record before it requests the information again. It's one factor that controls DNS propagation, which determines how long a DNS takes to update.

What Is TTL in a Ping?

Ping is a measurement of your connection's reaction time. For example, it measures how long a request that you send out takes to return. This ping measurement is one of the ways that you can measure network latency, which is a general term for your connection's responsiveness and delay.

When you run a ping test, the report may contain the TTL. This TTL value can give you more information about how long your connection takes to complete.



During this test, the ping records the results from four separate packets of data. It displays the amount of time taken and the TTL, which shows how many "hops" these packets of data took.

However, this report does not show you the TTL set by the other website's server. Therefore, you would need to know that value to calculate the true TTL.

What Are TTL Values?

When you set TTL values for your website, you choose a value in seconds. For example, a TTL value of 600 is the equivalent of 600 seconds or ten minutes.

The minimum available TTL is usually 30, equivalent to 30 seconds. You could theoretically set a TTL as low as one second. However, most sites use a default TTL of 3600 (one hour). The maximum TTL that you can apply is 86,400 (24 hours).

Technically, you can set any TTL value between the minimum and maximum parameters.

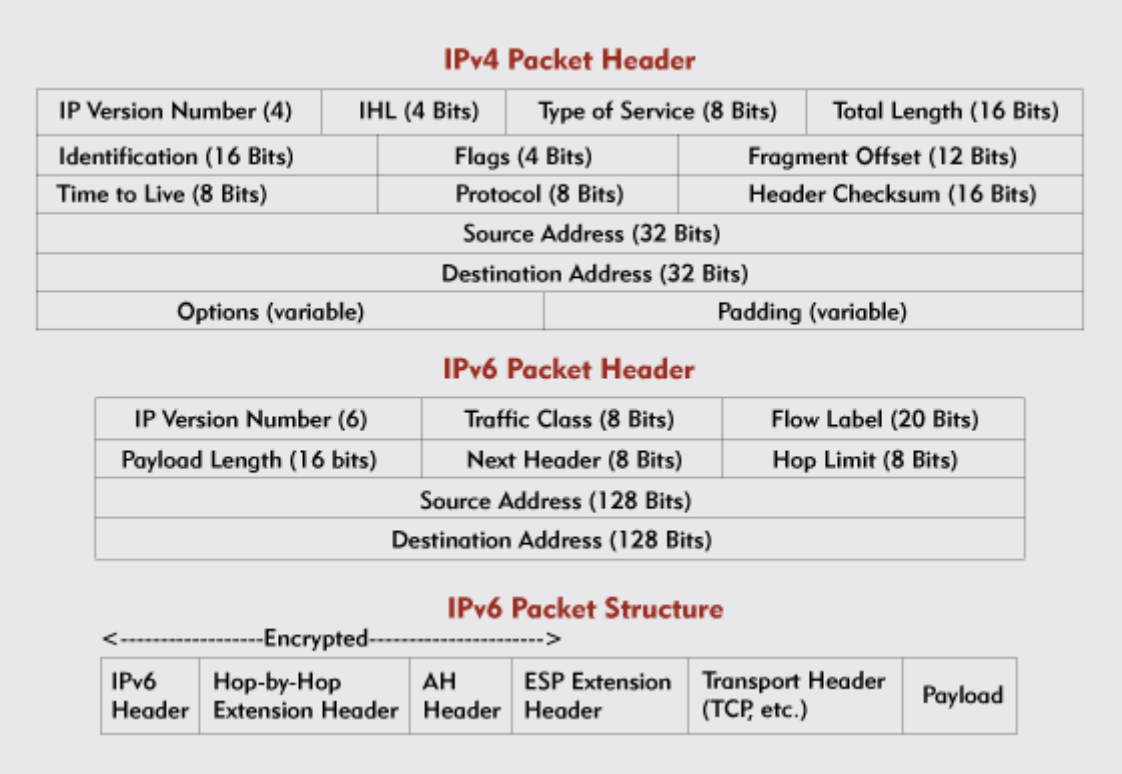
What IPv6 Field Is Similar to the TTL Field in IPv4 Packets?

IPv6 and IPv4 are different kinds of IPs. They both route packets of data through a series of rules (or protocols). These IPs contain information that enables data to arrive at its intended destination.

IP headers contain the information at the beginning of a packet of data. For example, they have information about the IP addresses from the source and the destination, among other details.

IPv4 is the original IP, and it has been available since 1984. It has a 32-bit address that is made up of numbers and periods. IPv6 is a newer IP, and it uses a 128-bit address format that has letters and numbers.

When you use an IPv4 header, it uses the TTL field, but IPv6 does not. With an IPv6 header, it has a field called Hop Limit that acts similarly to TTL. Here you can see a comparison of the two IP headers.



The Hop Limit determines how many “hops” a data packet will move before the router discards it.

How Does TTL Work?

Now, let’s take a more in-depth look at how TTL actually works. We already know that its value determines how much time (or how many hops) a data packet will exist for before a router rejects it. However, the way that this functions is a little more complex than you might imagine.

When you assign a TTL to your packet data, it carries this number as a numerical value in seconds. Every time the packet reaches a router, the router takes away one number from the TTL value and passes it along to the next step in the chain.

If the data packet is passed along too many times, its numerical value will reach zero. If this happens, it will fail to make the whole connection, and the router will discard it.

You will then receive an Internet Control Message Protocol (ICMP), which is a type of error message. For example, if your data has a TTL of 300, it can only pass through different routers a maximum of 300 times.

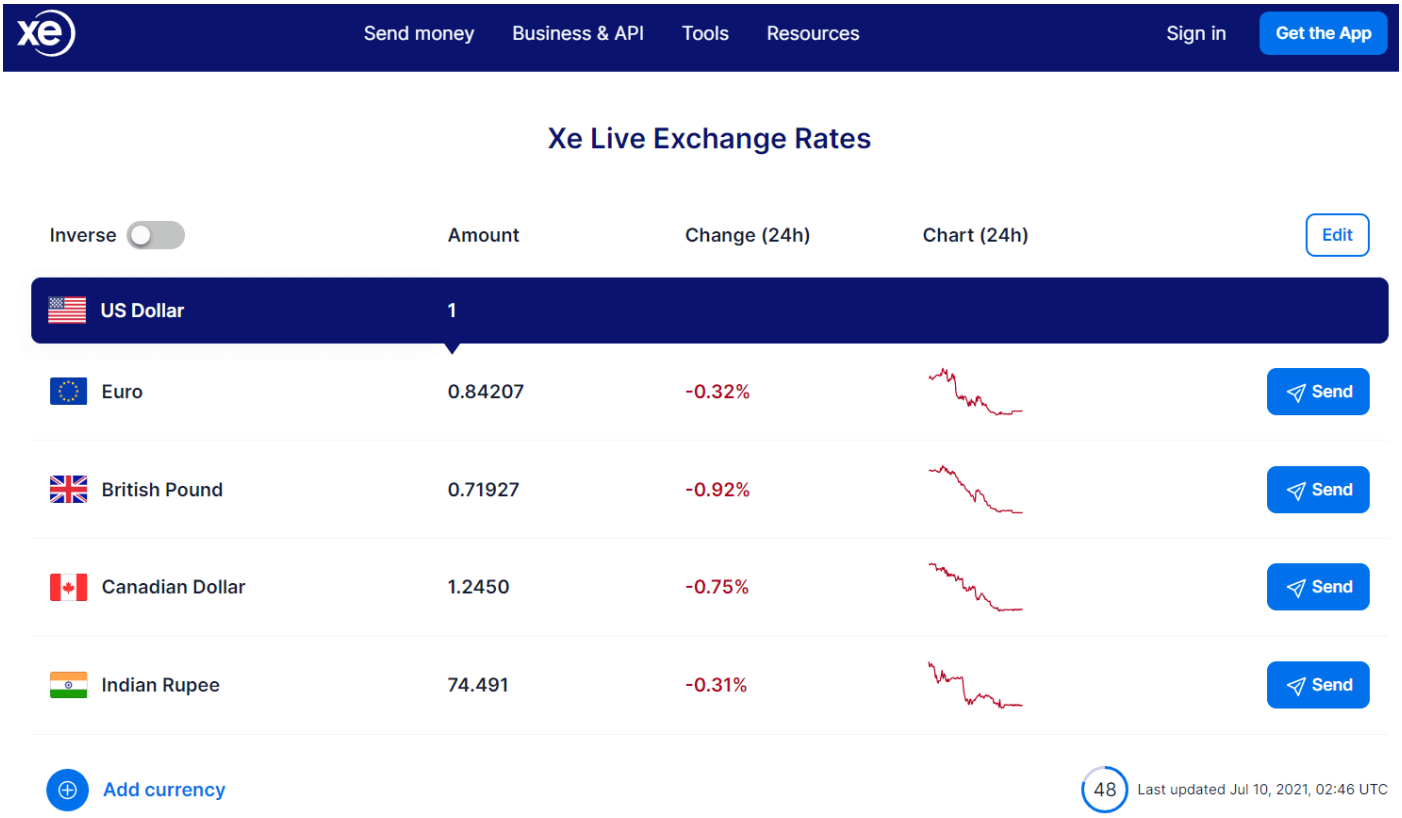
What Is TTL Used For?

We've seen so far that there are various applications for TTL. If you have a website, your main concern is probably how quickly your site loads. If your content is too slow to load, you could lose visitors and potential customers. Additionally, slow loading sites are detrimental to Search Engine Optimization (SEO).

Therefore, let's analysed TTL in the context of caching. Your website comprises a series of pages, code, images, and other content that can take a long time to load. If all of this content has to reload with every user, it can significantly slow down your site's speed.

You can use a longer TTL to make your cached site exist for longer before it updates. Consequently, your site will load much faster, and it will put less pressure on the server.

However, short TTLs can also be beneficial in some contexts. For example, websites that update constantly can benefit from a shorter TTL. A site like Xe uses real-time currency conversions, and so a long TTL could make its data redundant.



Tye Xe website

Additionally, short TTLs can protect your site against Distributed-Denial-of-Service (DDoS) attacks. These attacks happen when an entity overwhelms your website with thousands of requests from different locations in a short period. A short TTL can help protect your site because the frequent DNS updates are available to the blocking controls.

You may also want to change your DNS TTL before planned edits to your site, such as if you are adding a new website or updating the IP address for a server. The old information will be cached for the period of the TTL, so you may need to reduce it according to your timeline.

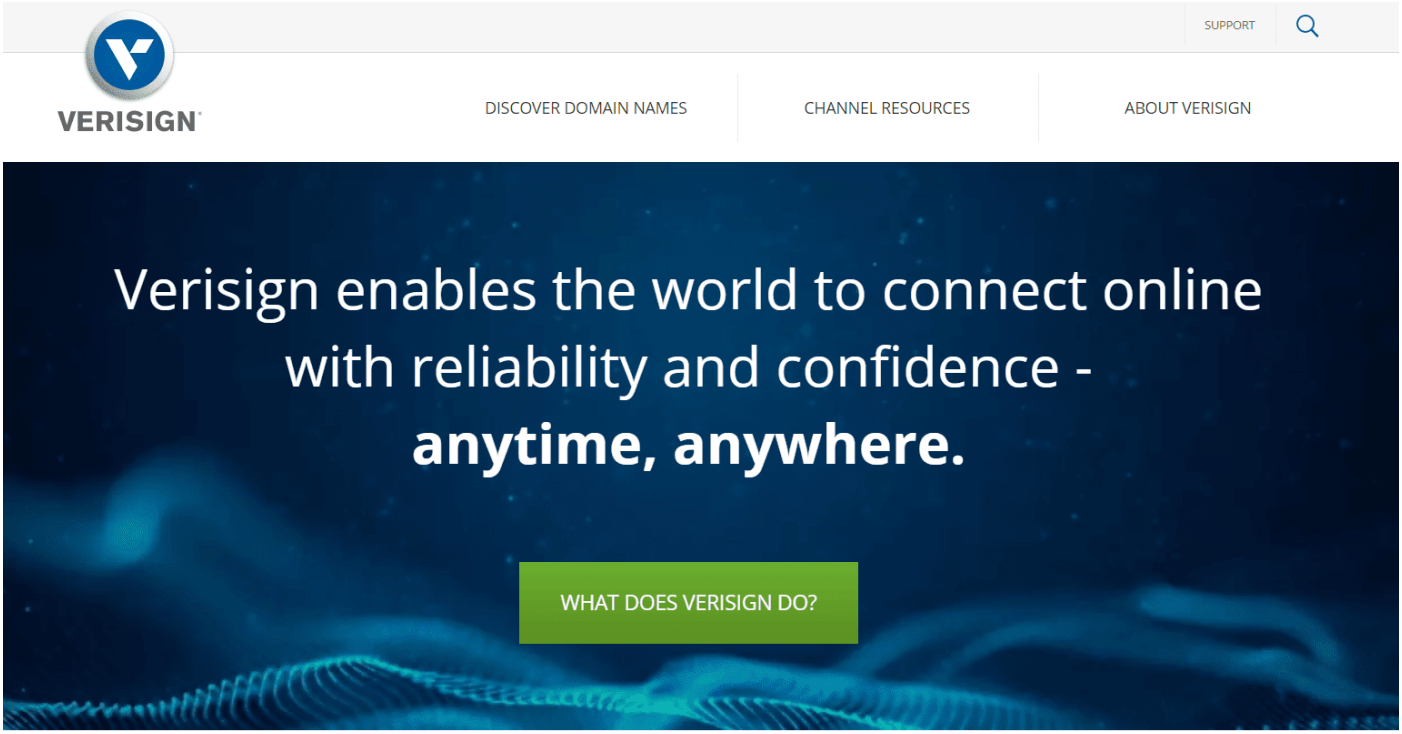
How Should You Choose a TTL?

Deciding on a suitable TTL for your needs can be challenging. Fortunately, there are some general guidelines that you can follow to see what fits your site best.

We recommend a TTL of 1-24 hours for most sites. Remember that TTL values are measured in seconds, so this is the equivalent of 3,600 to 86,400 seconds.

This TTL value can reduce loading time, which improves the user experience for your visitors and can decrease your bounce rate. The longer the better is a general rule, but remember to schedule any website maintenance accordingly.

If you own a registry website, you may want to choose a TTL of around one hour (3,600 seconds). These sites are high-level domains that may end in ".org" or ".com". For example, Verisign is a registry website:



The Verisign registry homepage

We advise changing your TTL to around 300 seconds (five minutes) before any operational changes to your site, especially if they'll impact the DNS. Otherwise, the updates may not come into effect in a timely manner.

We also recommend a TTL of 300 seconds for sites that are vulnerable to DDoS attacks. If you have fierce competition in your field, a competitor's website may try to put you out of action with one of these attacks. Additionally, controversial or whistleblowing websites are also potential targets for DDoS.

Summary of TTL:

TTL is an essential setting that enables you to control how long a server stores your site's information. You can make your TTL longer or shorter to decrease your page load time, keep data up-to-date, and avoid DDoS attacks.

You can set your TTL as low as 30 seconds or as high as 24 hours. However, for most general sites, a TTL between 1 and 24 hours provides an excellent balance. By choosing this value, you can keep your site loading quickly and still showing current data.

Finally, I can recommend a short TTL of 300 seconds for DNS-based load balancing. This is when multiple servers are sharing traffic by providing various IP addresses for server requests. By doing so, the system reduces strain on a single server.

Route 53 POINTERS:

- **AWS Route 53 most commonly used Records:**
  - A record – URL to IPv4
  - AAAA – Record - URL to IPv6
  - CNAME – Record -URL to URL
  - Alias- Record -URL to AWS resources
- **AWS Route 53 is 100% available**
  - **Can use both:**
    1. Public Domain(can be purchased) – sanoj.homes
    2. Private Domain – local.sanoj.homes (any local name you want)
  - $0.50 per hosted zone /month for the first 25 hosted zones
  - $0.10 per hosted zone /month for additional hosted zones

Route 53 Features:

1. Resolver
2. Traffic flow
3. Latency based routing
4. Geo DNS
5. Private DNS for Amazon VPC
6. DNS Failover
7. Health checks and Monitoring
8. Domain Registration
9. CloudFront Zone Apex Support
10. S3 zone Apex Support
11. Amazon ELB Integration
12. Management Console
13. Weighted Round Robin

CNAME Record

1. A CNAME record can redirect DNS queries to any DNS record
   (i) Directs from one URL to another URL (only for Non-Root Domains)
   Ex. app.sanoj.homes
2. You can not create a CNAME Record for sanoj.homes
3. You can not create a CNAME record that has the same name as the hosted zone (the zone apex)
4. Route 53 charges for CNAME queries
5. A CNAME record redirects DNS queries for a record name regardless of the record type such as A or AAAA

ALIAS Record

1. An alias record can only redirect queries to selected AWS resources. Such as the following:
   (i) Amazon S3 Buckets
   (ii) Cloud Front distribution
   (iii) Another record in the route 53 hosted zone that you are creating
   the alias record in
2. Works for both Root and Non-Root Domain
   (i) Alias hello.com → sanoj.homes(YES)
   (ii) Alias hello.com → app.sanoj.homes(YES)
3. You can create an alias record at the top node of a DNS namespace also
   known as the zone apex.
4. You can create an alias record for sanoj.homes that routes traffic to
   www.sanoj.homes
5. Route 53 doesn't charge for alias queries to AWS resources.
6. Route 53 responds to a DNS query only when the name of the alias record (such as A or AAAA) matches the name and type in the DNS query.

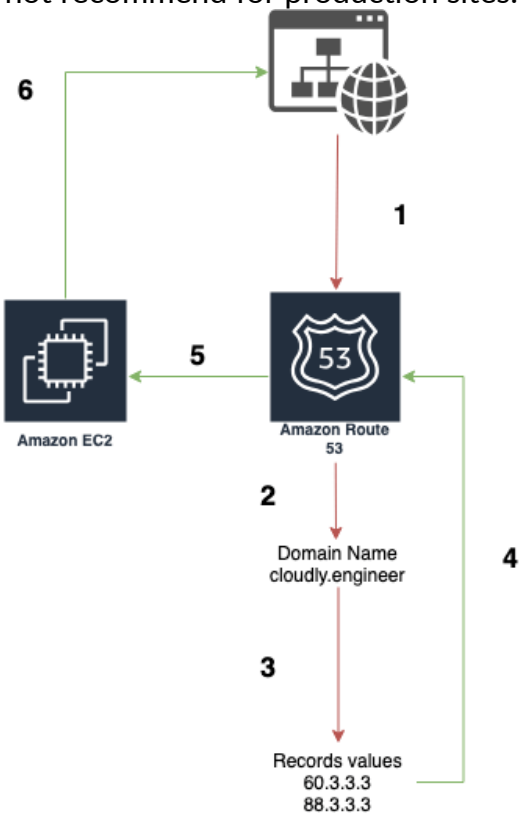
AWS Route 53 Routing Policy:

A routing Policy is a mechanism in which determines how Amazon Route 53 responds to queries:

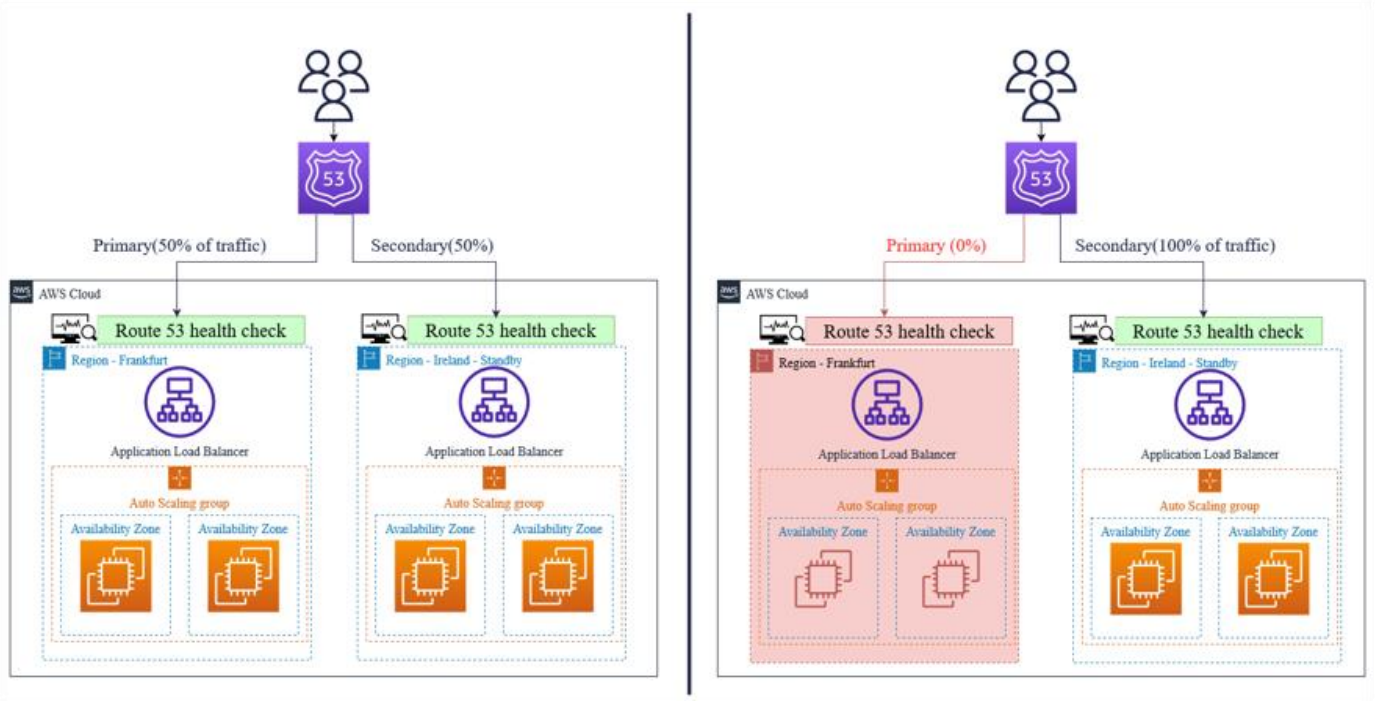
There are 8 different kinds of routing policy:

1) Simple
2) Failover
3) Geolocation
4) Geoproximity
5) Latency
6) Weighted
7) Multivalue answer
8) IP-Based routing policy

- **Simple Routing** This is the default routing policy. Use this only when you have exactly one resource such as one EC2 web server. This policy can contain multiple values but it returns one resource. This policy is not recommend for production sites.



- **Failover Routing** A failover routing policy allows you to direct users to a particular resource only when that resource is healthy, and provide a backup, or "failover" resource for when the primary resource fails or reaches the maximum load.

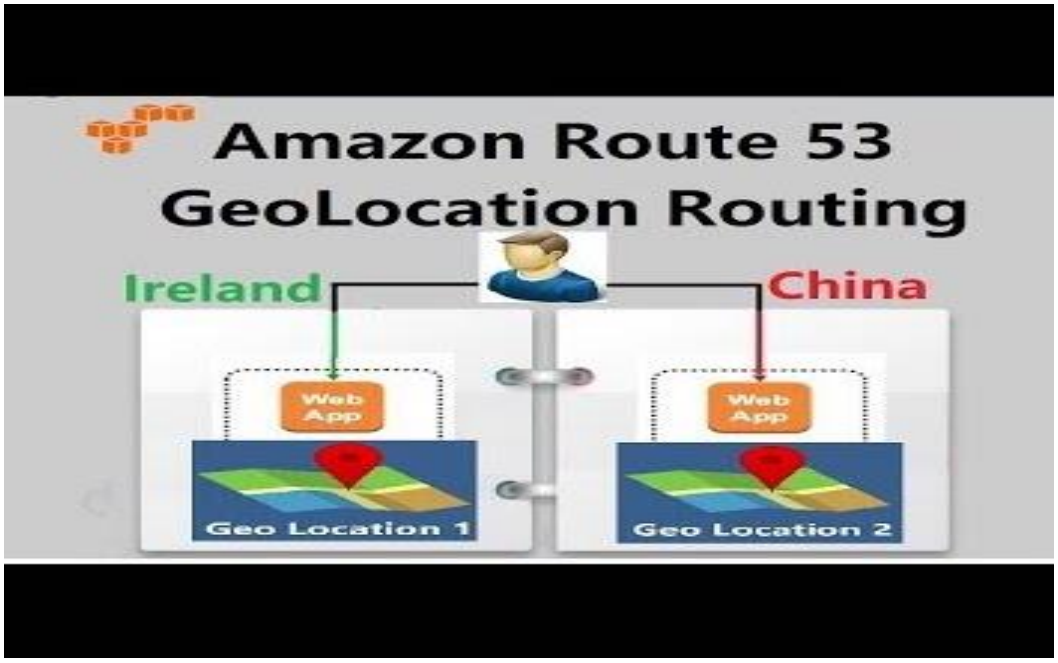


To use a failover routing policy, you configure active-passive failover, where one resource (the primary resource) handles all the traffic when it's healthy and the other resource (the secondary resource) only handles traffic when the first resource isn't healthy.

- **Geolocation** A geolocation routing policy is what you use when you want to route traffic based on the location of your users.

  Most businesses these days have users all over the world, and they want to serve content to those users as fast as possible.

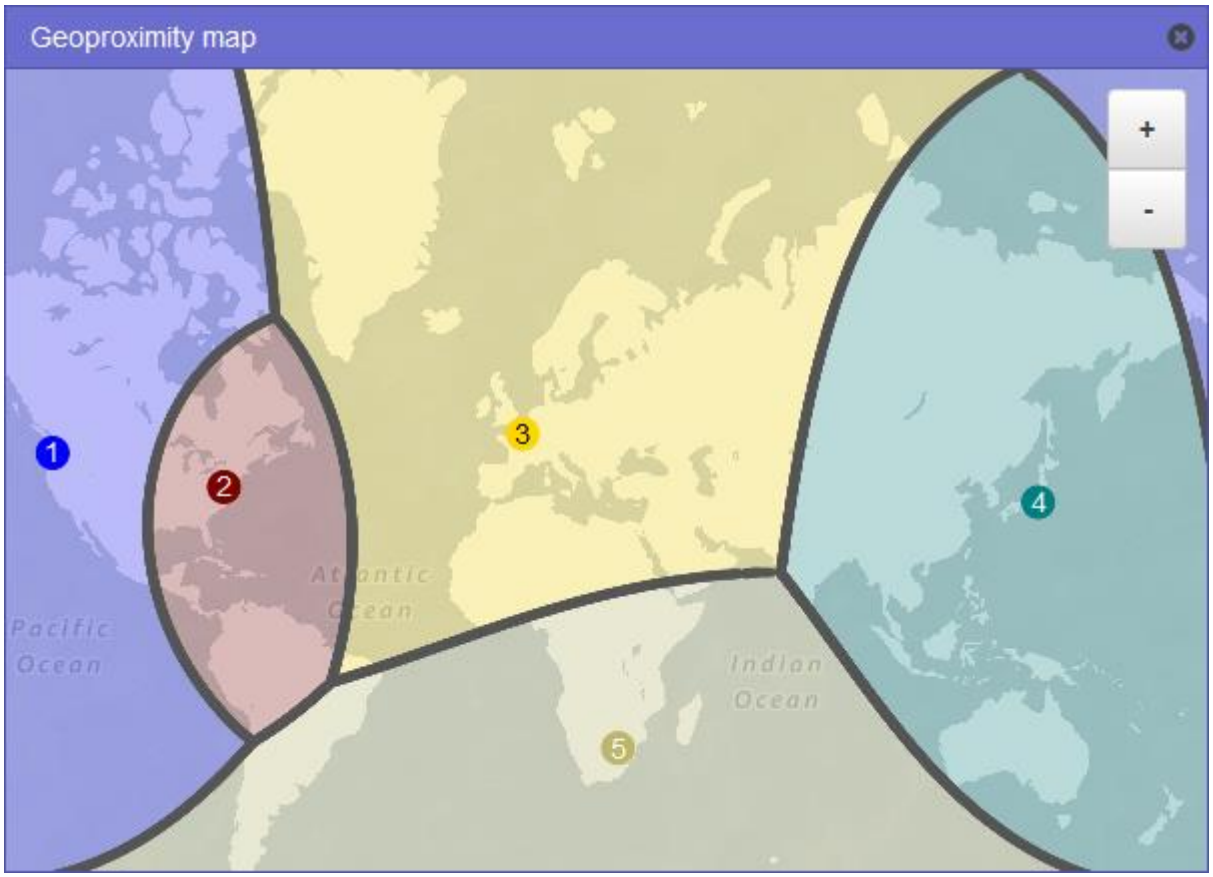
  A geolocation routing policy allows you to allocate the resources that serve your traffic based on the location that users' DNS queries originate from.



With geolocation routing, you can localize content and restrict the distribution of content to only the locations in which you are able or allowed to distribute. You can also balance the traffic load across endpoints in a predictable way.

For example, say you have servers in Oregon and New York, and many of your users are in California. A geolocation routing policy might send those California users to your Oregon server so you can serve them content specific to the West Coast of the US.

- **Geoproximity** A geoproximity routing policy is what you use when you want to route traffic based on the location of your resources or shift traffic flow between resources. This policy allows you to direct users to different servers, even though those servers might be further away, using something called a bias

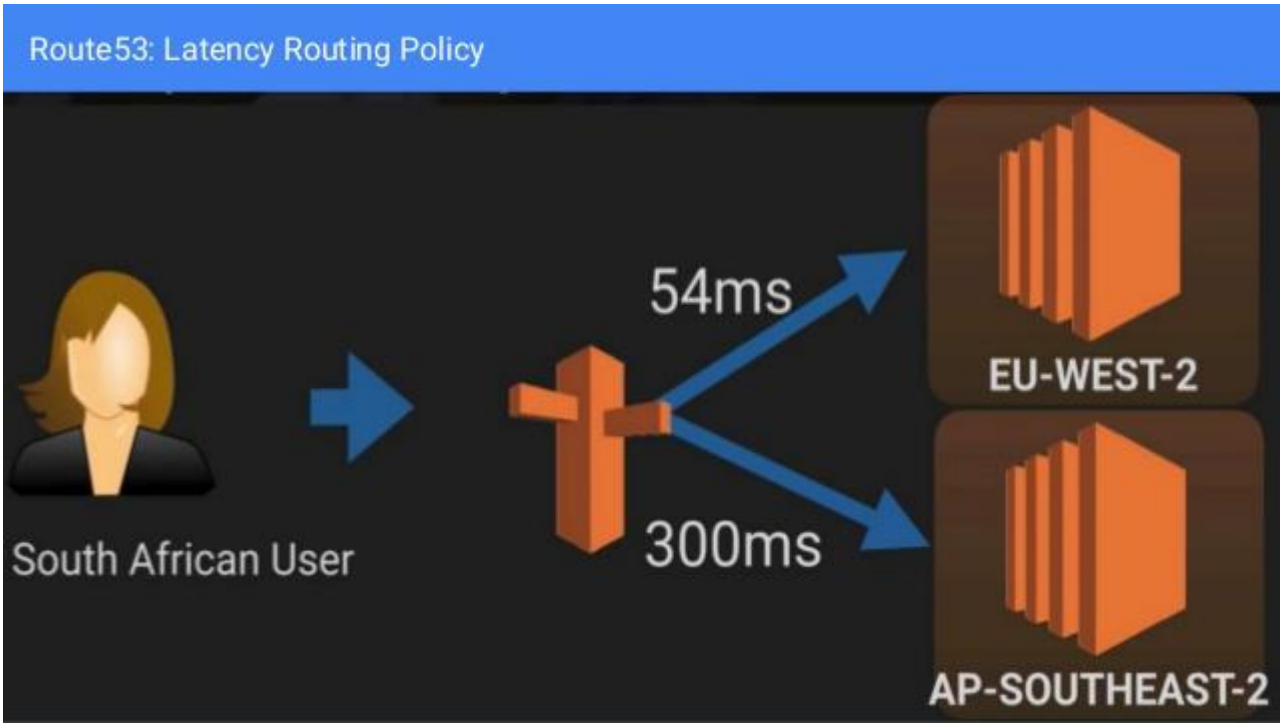


Take the previous example. Let's say you have servers in Oregon and New York, and users in California, but your New York servers are larger and can handle more traffic than the Oregon servers. You might use a geoproximity routing policy to direct a portion of the California users to the New York server.
GeoProximity routing is like creating a sphere of influence for your resources.

GeoProximity routing Lets Route 53 route traffic to your resources based on the geographic location of your users and your resources. You can also optionally choose to route more traffic or less to given resources by specifying a value known as bias.

A bias expands or shrinks the size of the geographic region from which traffic is routed to a resource.

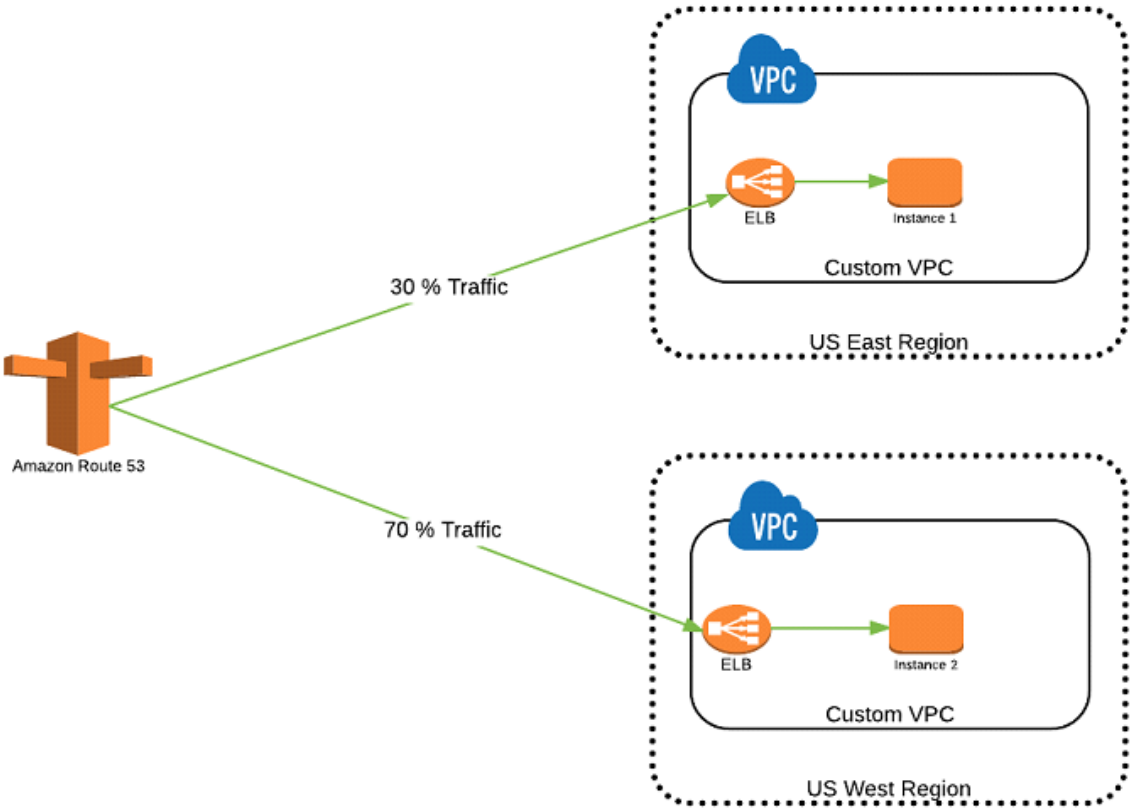
- **Latency** A latency routing policy is what you would use if you have resources in multiple regions and you want to route traffic to the region that provides the best latency. These days, users expect a blazingly fast internet experience, and any obstacle to providing that experience should be removed.



With latency routing, you can guarantee that your users will always receive content from the server that will provide them with the fastest experience. This will usually be the server closest to them, but not necessarily.

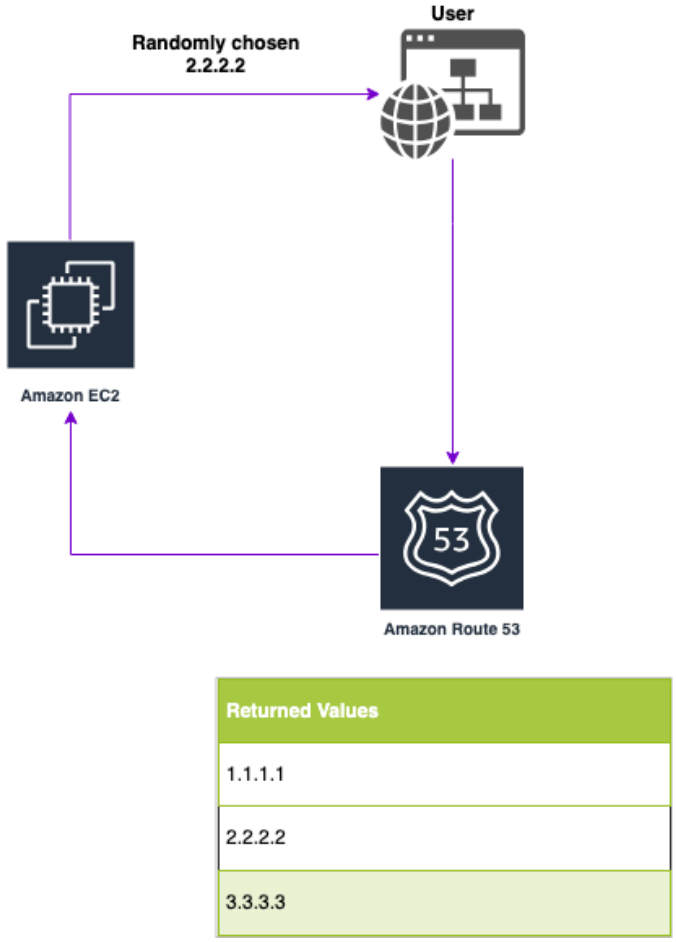
When you configure a latency routing policy, you set up latency records that the router uses to lookup the regions that serve your content and determine the region that will provide the best latency for a given user.

- **Weighted** A weighted routing policy allows you to send traffic to specific servers, with no consideration for latency or geographic location. You can choose exactly how much traffic is sent to a particular resource, and you have total control over the traffic flow.



There are several reasons you may want to use a weighted routing policy, including partial rollouts of software, load balancing, and load-testing.

Multivalue Answer: This one lets your return multiple values for each of your resources. The client or user browser randomly chooses one. Optionally you can add health checks. If any value becomes unhealthy then the client chooses another value to resolve. This is not an alternative solution to load balancing, it's an enhancement.



- use when routing traffic to multiple resources
- want to associate Route53 health checks with records
- up to 8 healthy records are returned for each multi-value query
- **not a substitution for having an ELB**

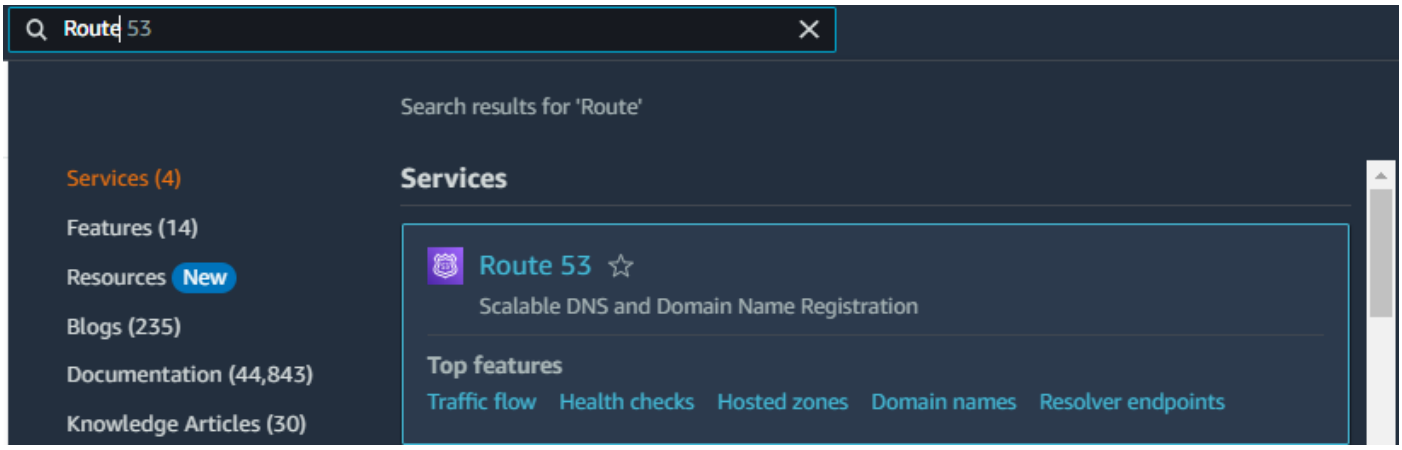
It lets you configure Route53 to return multiple values, such as IP addresses for your web servers, in response to DNS queries. You can specify multiple values for almost any record, but multivalue answer routing also lets you check the health of each resource, so Route53 returns only values for healthy resources.

Similar to simple routing but with health checks on each record set

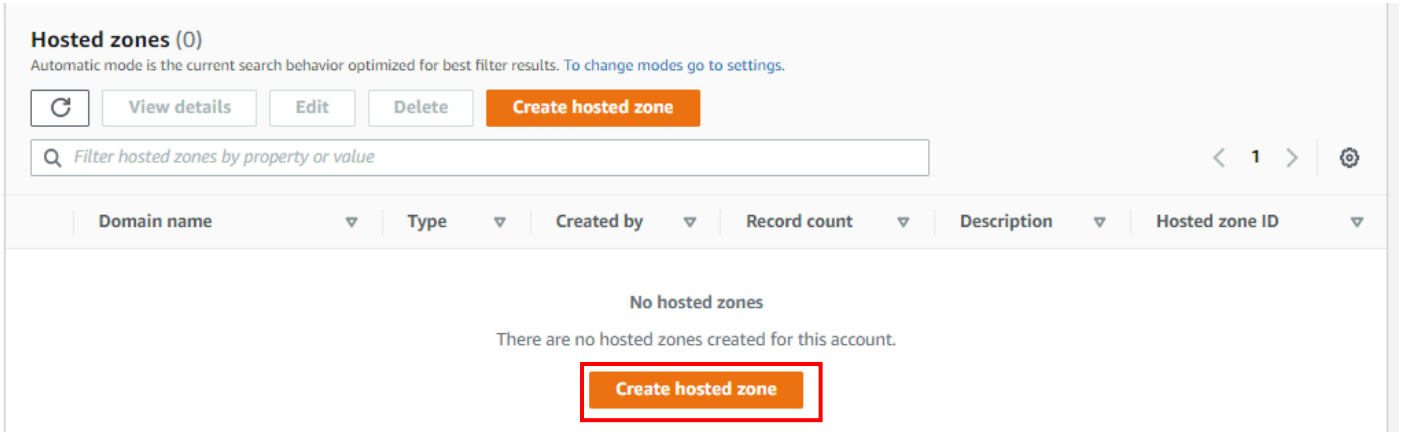
IP Based Routing: IP-based routing allows you to create a set of Classless Inter-Domain Routing (CIDR) blocks that represent the client IP network ranges and map those CIDR blocks to locations. These sets of CIDR blocks that are mapped to sets of locations are referred to as a CIDR collection. Within each CIDR collection you will be able to create a set of locations, and for each location you can define a set of CIDR blocks that represent the client subnets you wish to pair with a given location.

Scenario -1:
Configuring Route 53 For Simple Routing Policy:

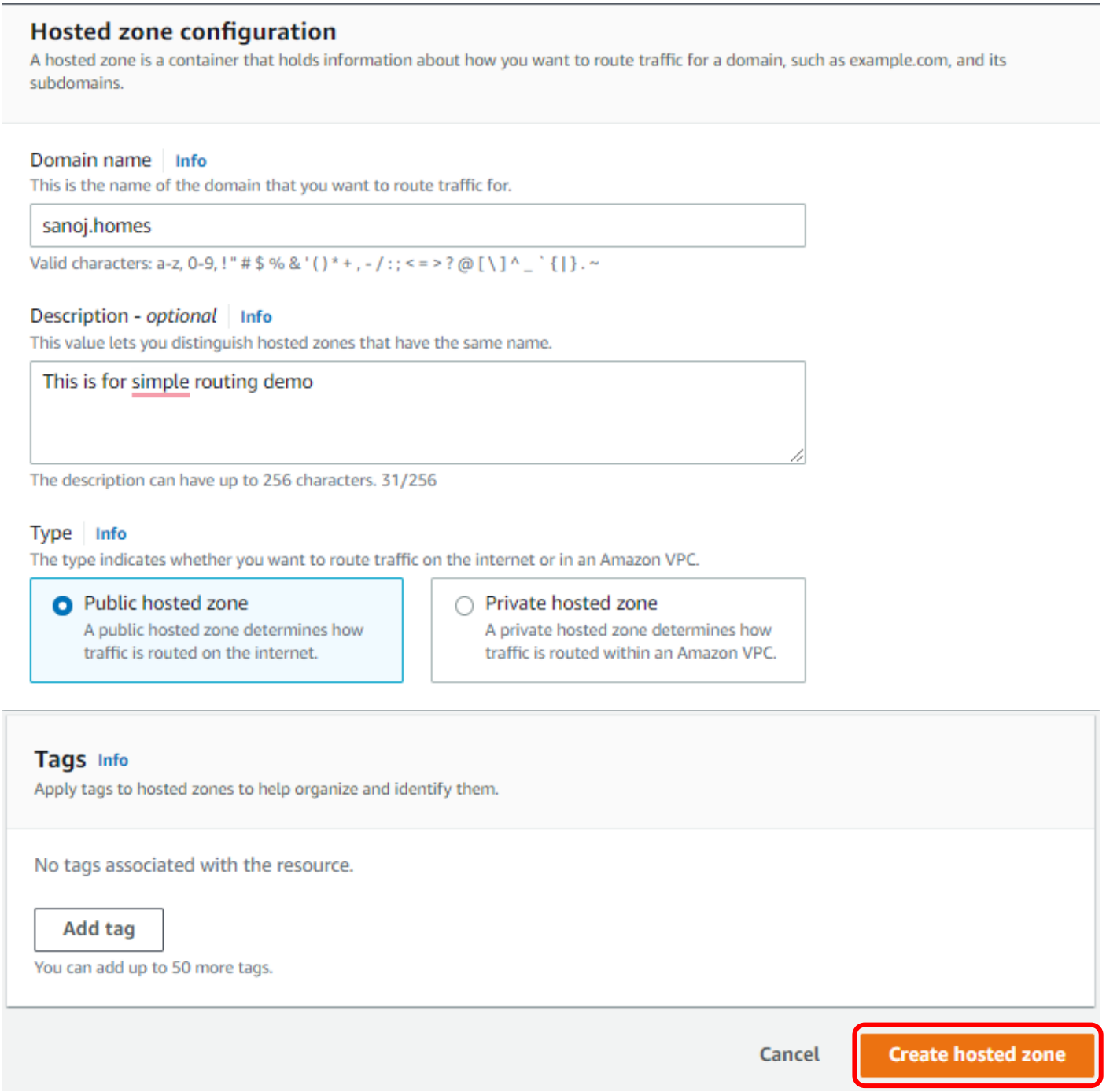
Step1: Open the Route 53 by searching on AWS console Search bar and Hit Enter.



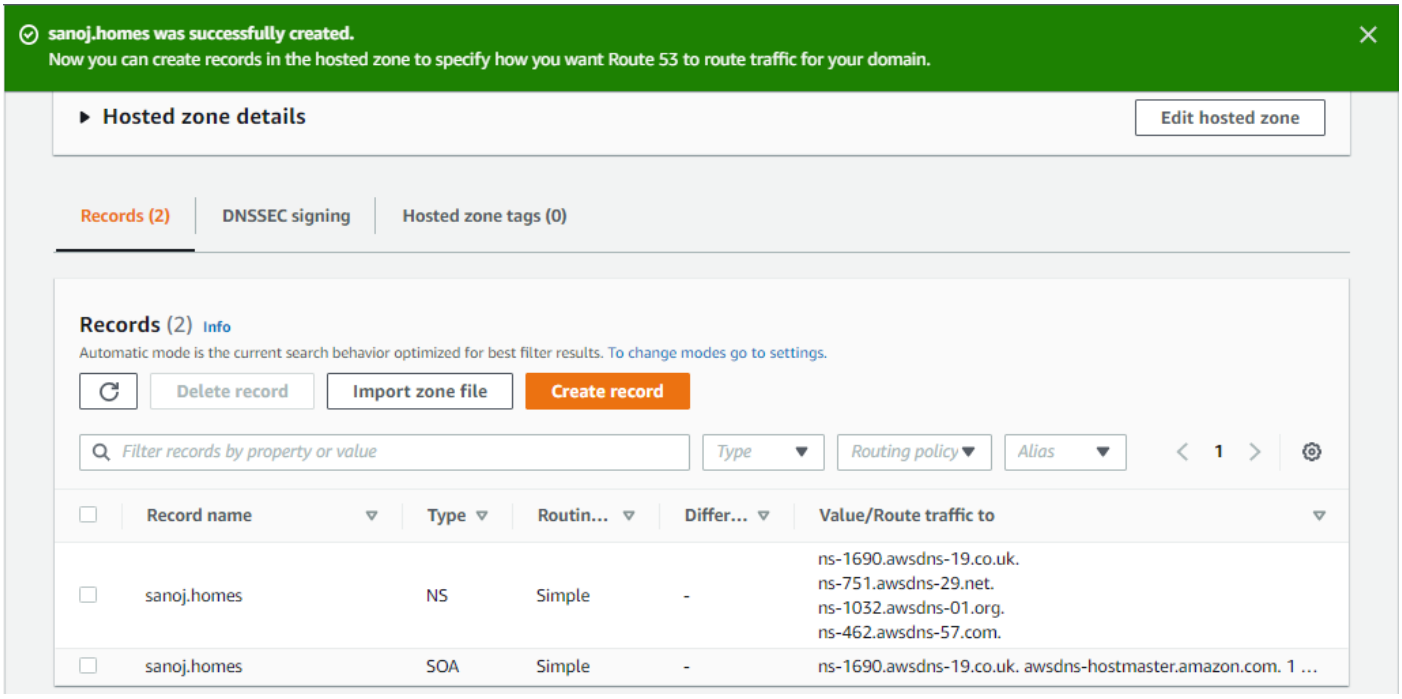
Step 2: Now click on **"Create Hosted zone"**



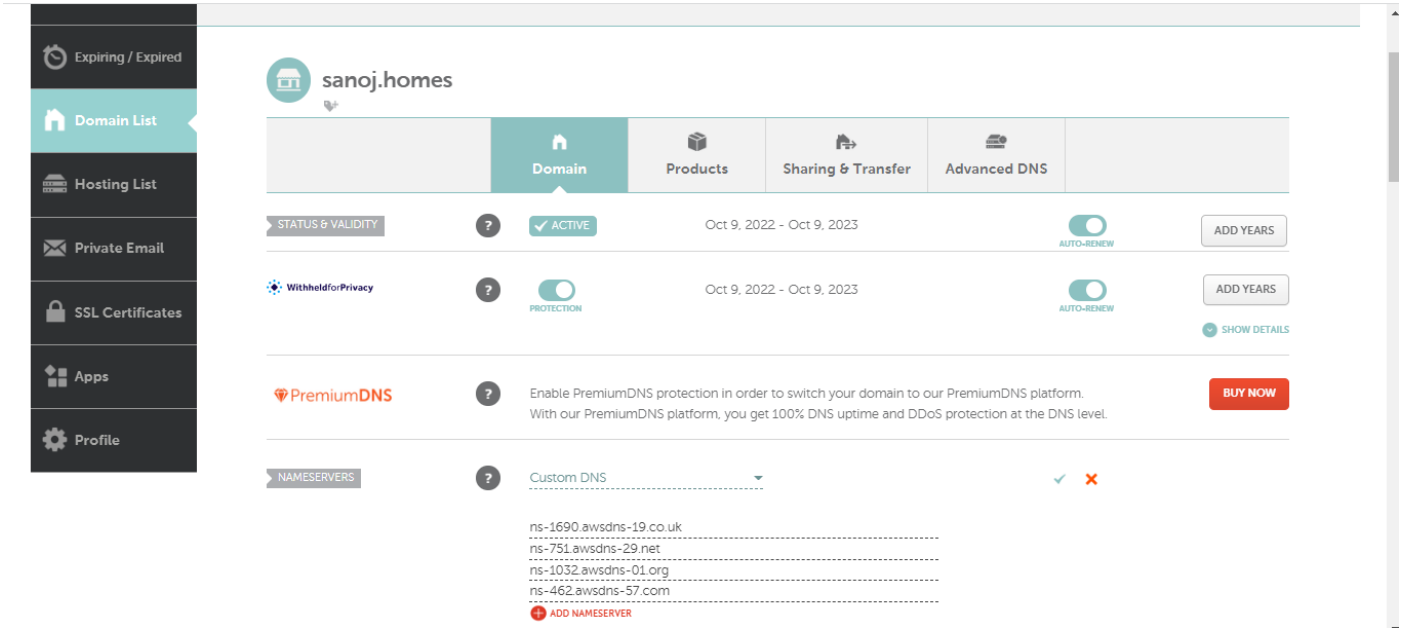
Step 3: Fill the details as I mentioned in the following image and click on create hosted zone.



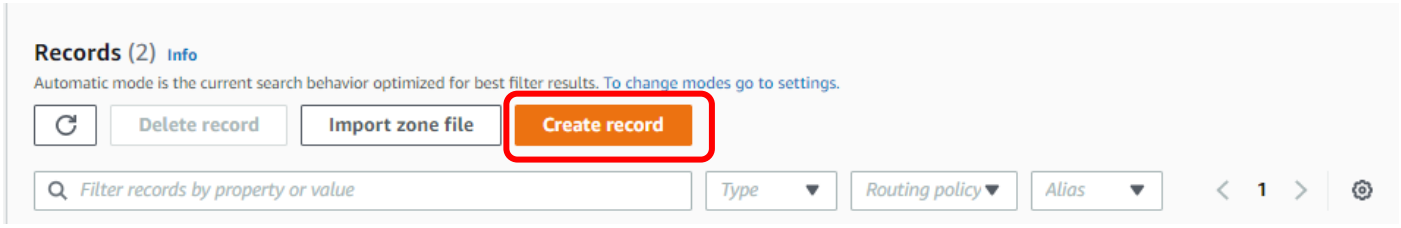
After successfully created Hosted Zone you will see following interface along with NS and SOA Records.



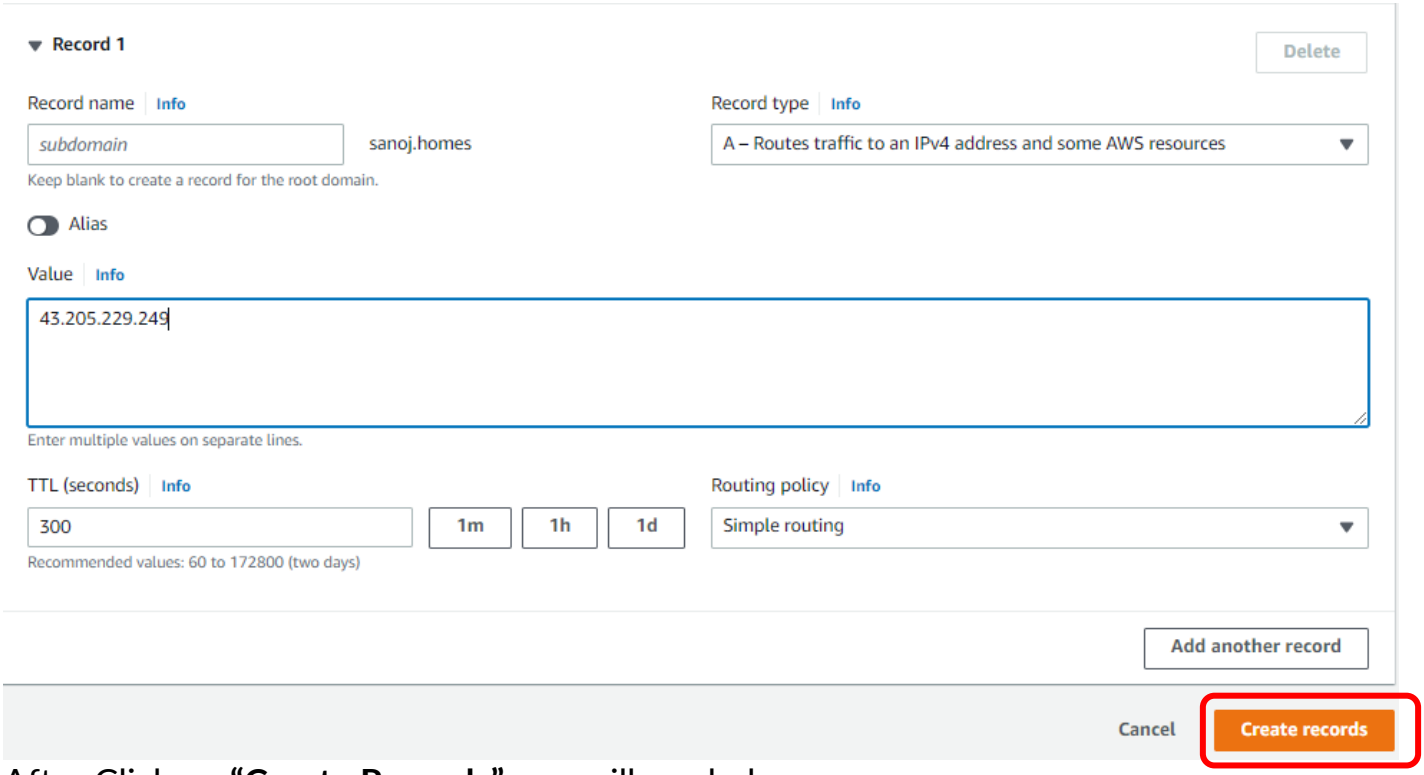
Step 4 : Now copy the above NS records one by one and paste it in your personal registrar like Godaddy, Namecheap, etc. I have Namecheap so I will update on Namecheap web console.



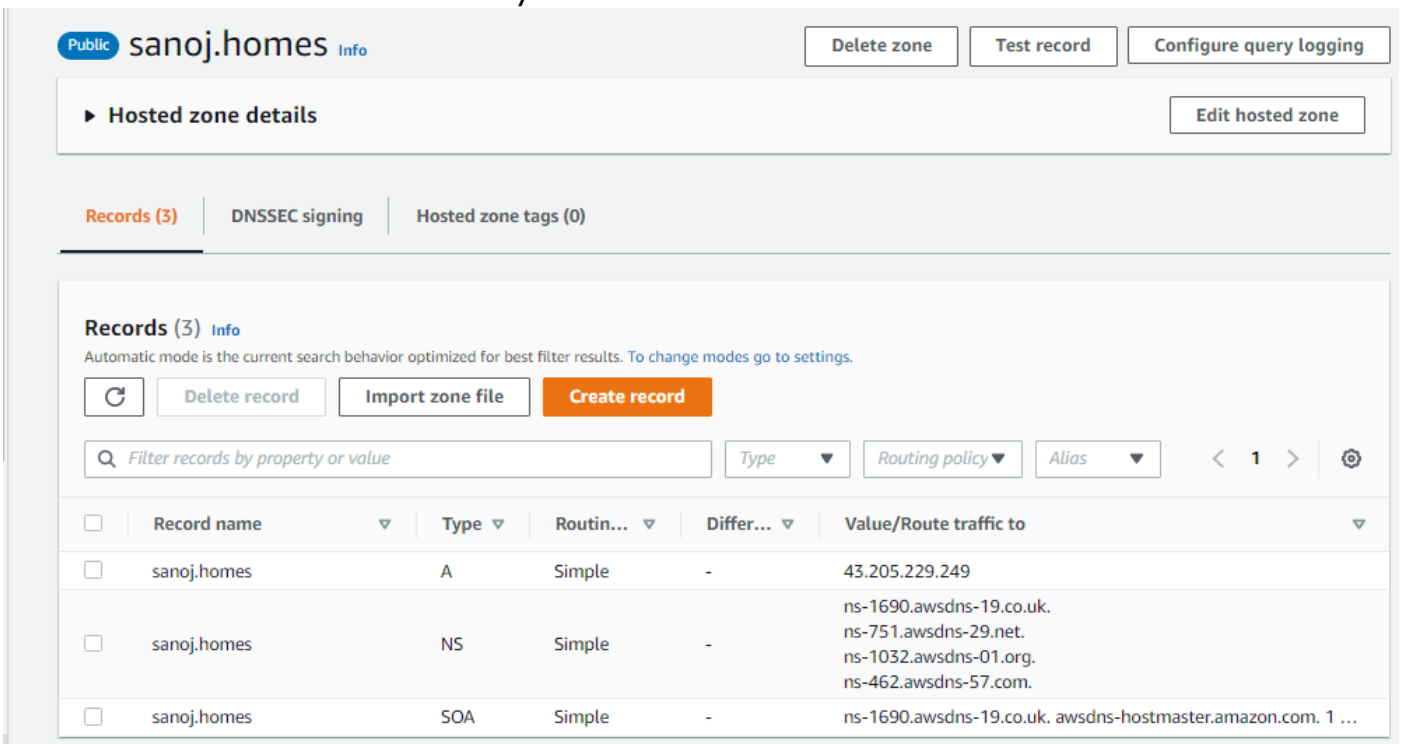
Step 5: Now click on **"create a record"** and fill the details as per following details on image.



Step 6: Now put your Public IP address of your webserver and click on the Create records.

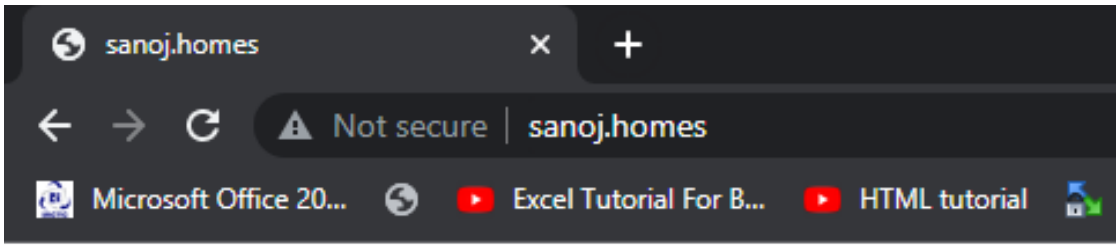


After Click on **"Create Records"** you will see below screen.



Note: After creating the record the changes might be not updated on the spot it will take few minutes to hour to update all records on name server so leave it as it is and after few hours you will try again it will reachable to you or your website.

Now if you want to check everything is working fine just hit your URL (domain Name) you will see below kind interface.



"Hello Sanoj from i-010808a62eafe9915.ap-south-1.compute.internal"

Note: sometimes browser have cached to previous on data so might be possible it will not showing your web page and you will get error like this site can not be reached.

For this problem resolution clear your cached & browsing history.

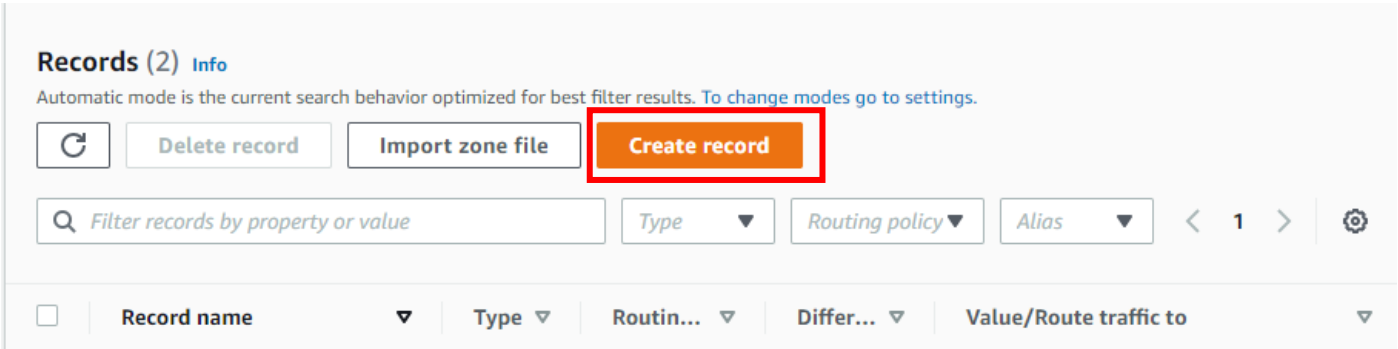
another solution is to know everything is working fine or not just ping your domain name from cmd if reply you got means everything is fine.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\sanoj> ping sanoj.homes

Pinging sanoj.homes [43.205.229.249] with 32 bytes of data:
Reply from 43.205.229.249: bytes=32 time=317ms TTL=235
Reply from 43.205.229.249: bytes=32 time=306ms TTL=235
Reply from 43.205.229.249: bytes=32 time=82ms TTL=235
Reply from 43.205.229.249: bytes=32 time=121ms TTL=235

Ping statistics for 43.205.229.249:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 82ms, Maximum = 317ms, Average = 206ms
PS C:\Users\sanoj>
```

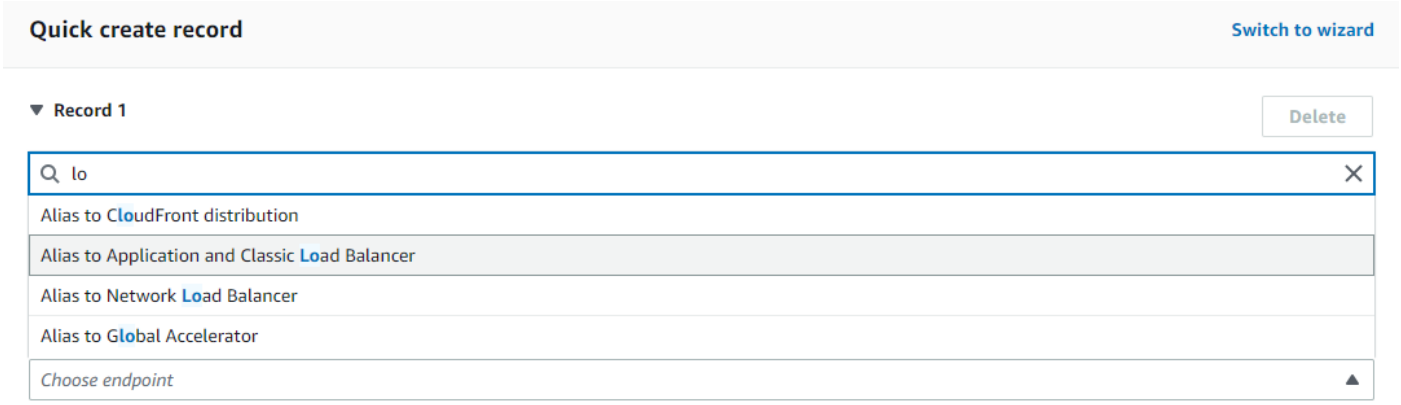
Now we are done with simple routing...

Lets try simple routing with Load balancer dns endpoint or using Alias.

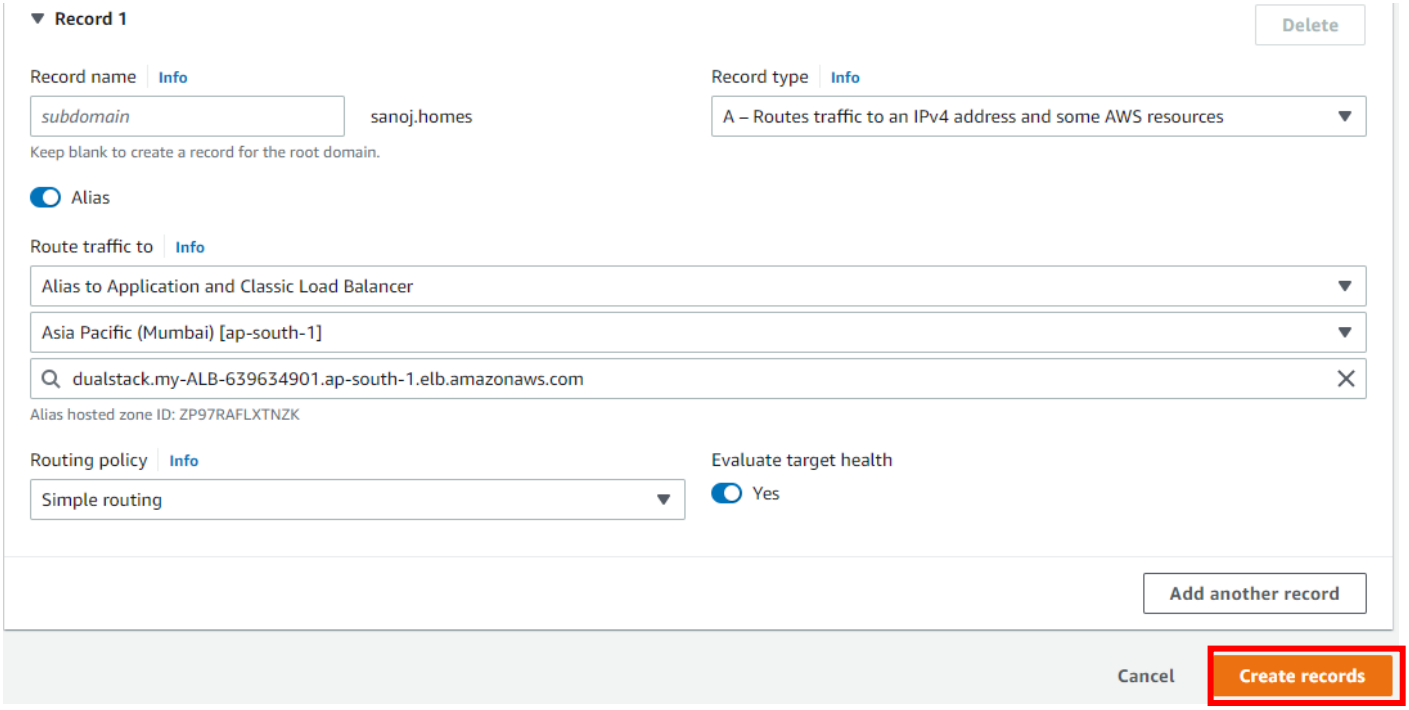
Step 1: Click on **"Create a Record"**



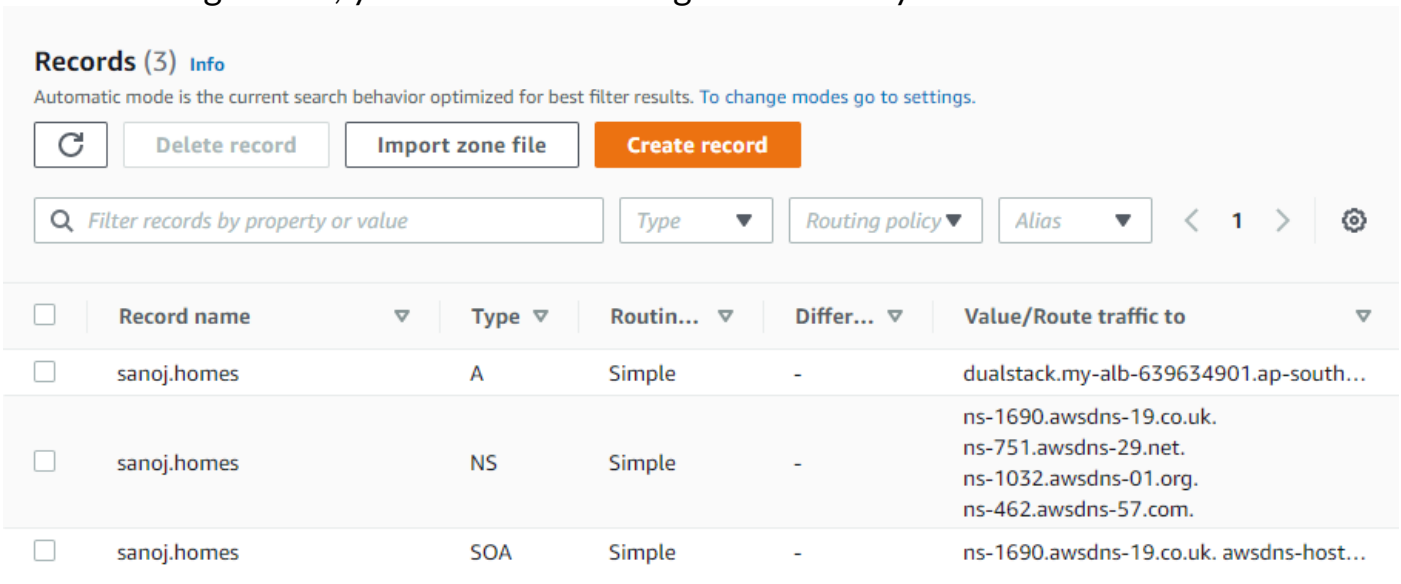
Step 2:click on the **"Alias"** select the **Load Balancer Endpoints** then choose the **region** where your Load Balancer created and then select your **ELB endpoint** and click **"Create Record"**



After all thing selection you will see below screen:

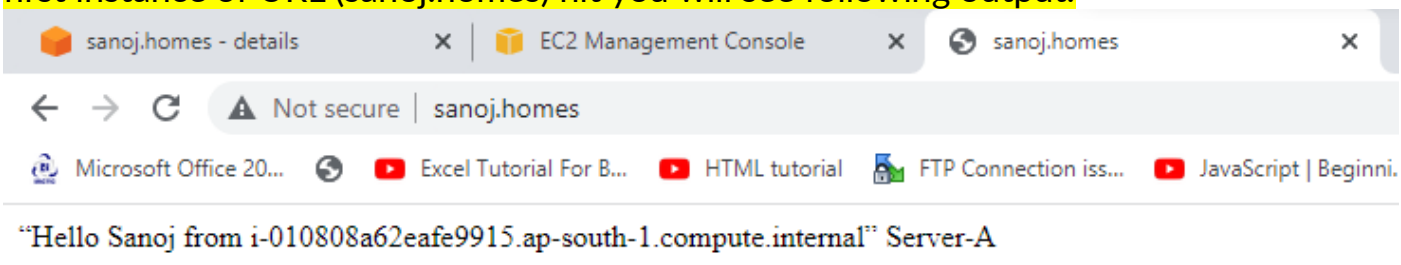


After creating record, you will see following screen with your record name.

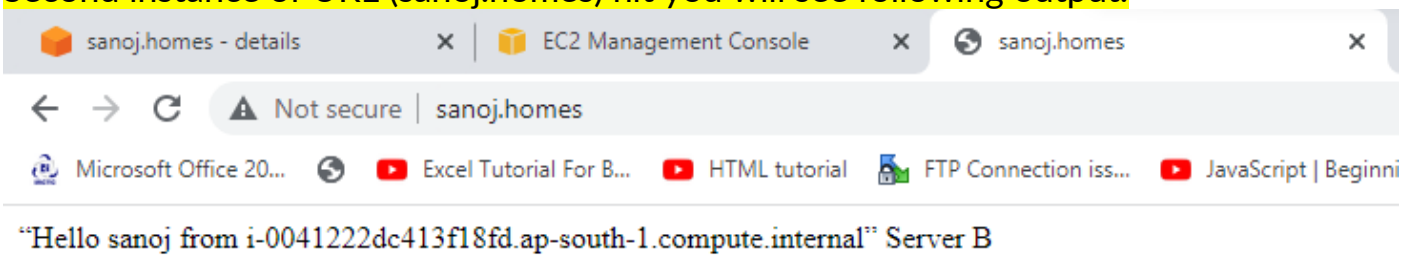


Now again hit the URL (Domain Name) sanoj.homes

first instance of URL (sanoj.homes) hit you will see following output.



Second instance of URL (sanoj.homes) hit you will see following output.



Why are we getting such kind of output?

Because we have set our domain with load balancer so in that case first "sanoj.homes" point to load balancer DNS and load balancer take request to target instance in round robin fashion and after that we got the output.

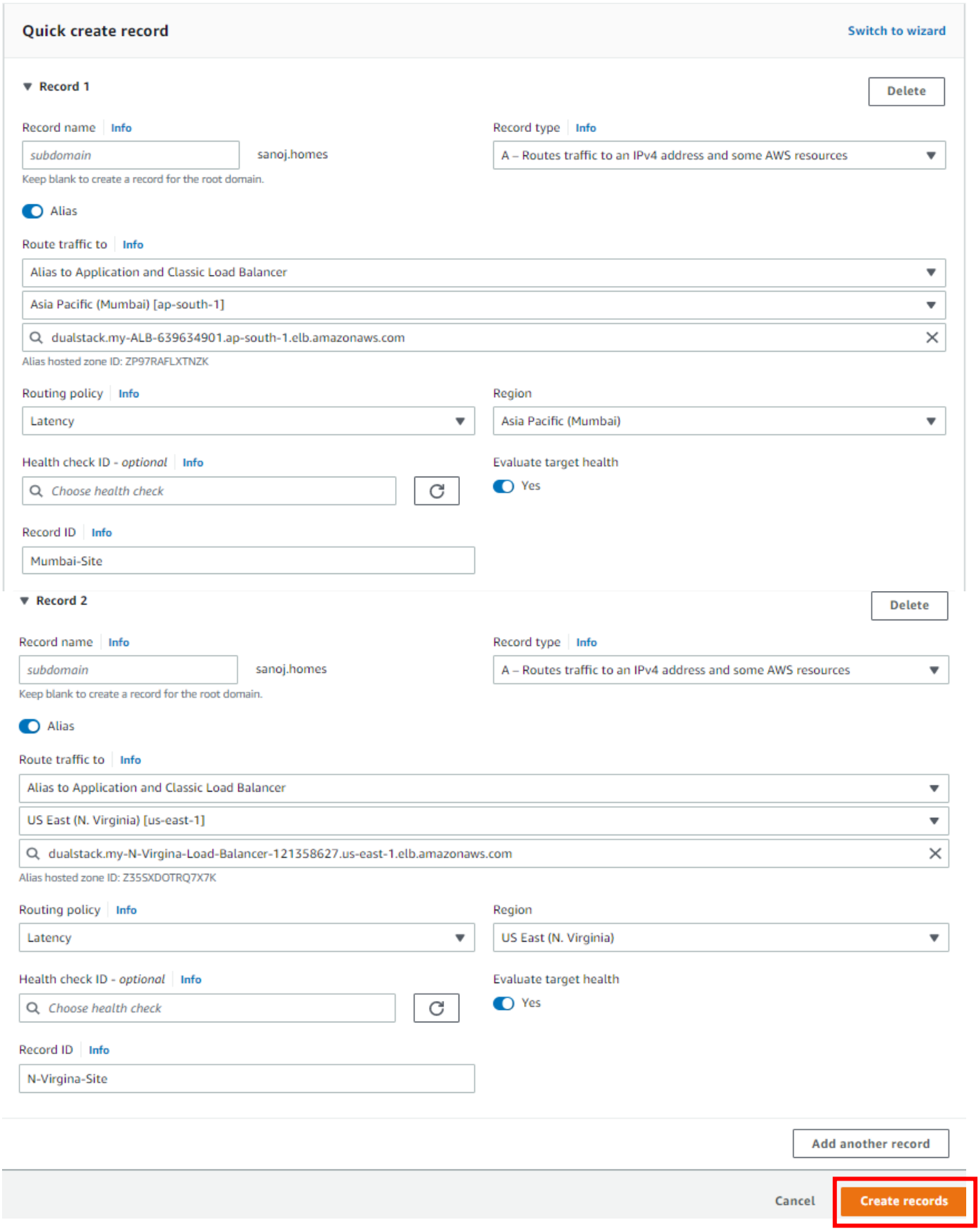
sanoj.homes -→ my-ALB-639634901.ap-south-1.elb.amazonaws.com

Scenario -2:
Configuring Route 53 For Latency Based Routing Policy:

Step 1: First of all create a Load Balancer in two different region and add instances in target group as of now I am creating LoadBalancer & EC2 instances in Mumbai and N.Virginia .

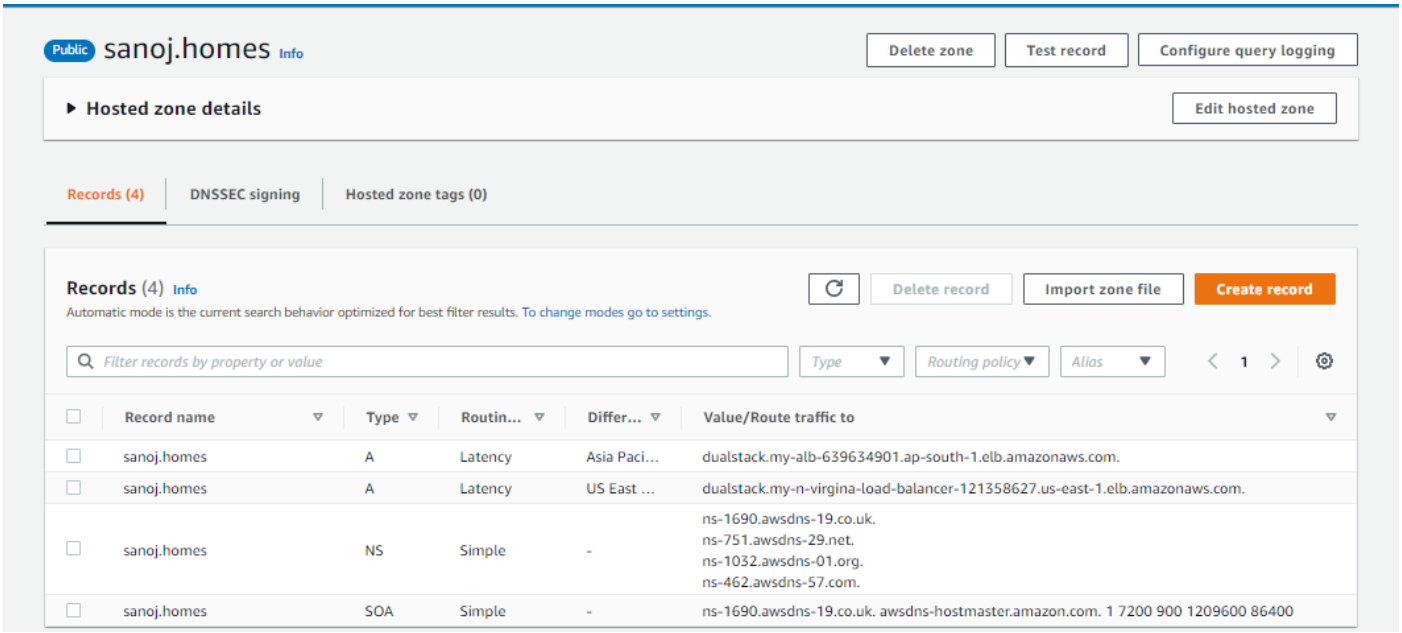
- After this you have to **create a record**
- and select **application load balancer**
- then select **Region in which your load balancer has**
- then select your **Load balancer URL(DNS)**
- then select routing policy **"Latency"**
- then select **region** and give a **Record ID Name**.

For reference you can see following Images.



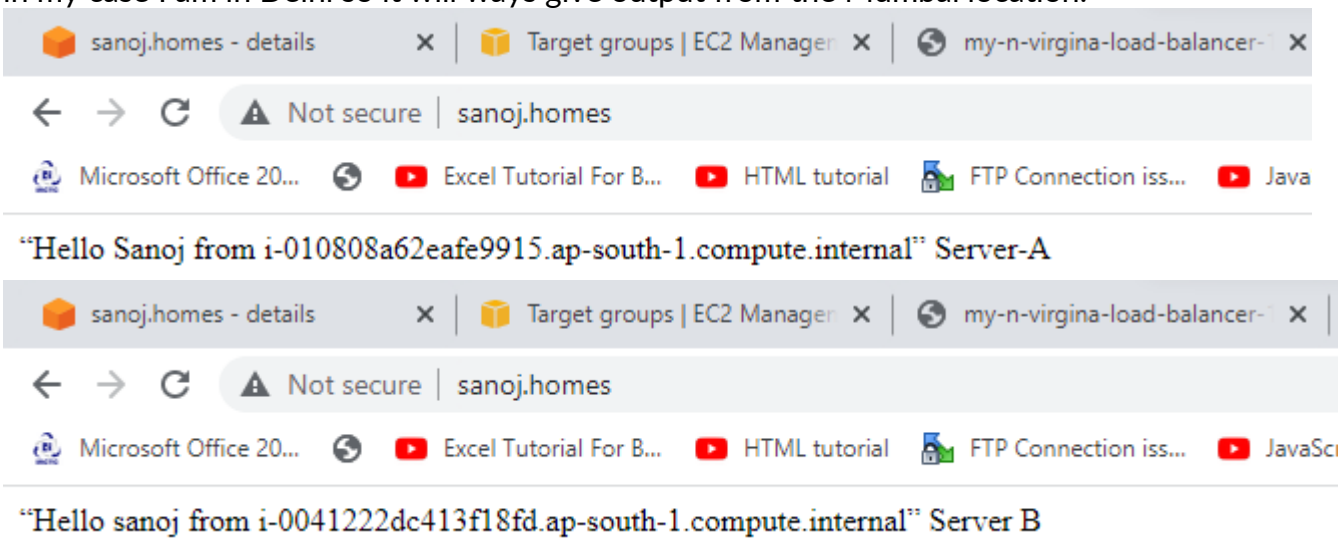
Now Click on the **“Create records”**

After that you will see following details with record.



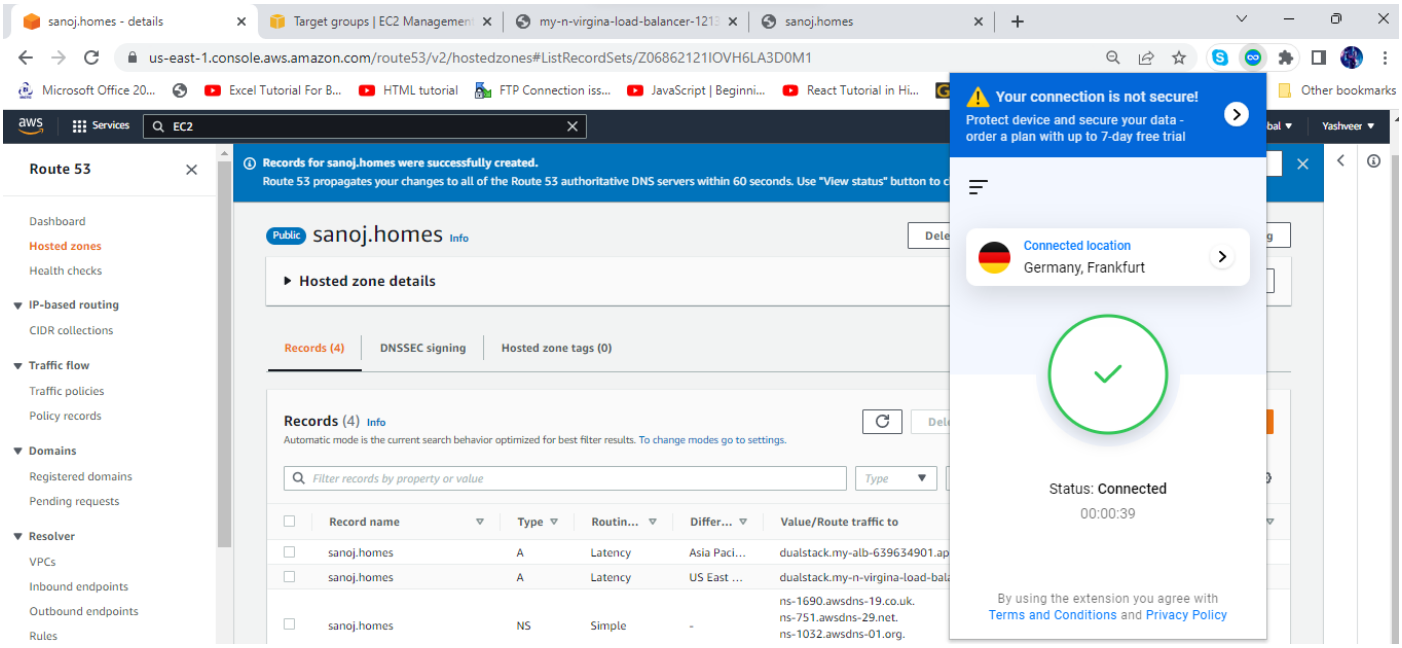
Now try to hit your DNS rapidly you will always hit your nearest location webserver because it is near to you and give low latency.

in my case I am in Delhi so it will ways give output from the Mumbai location.

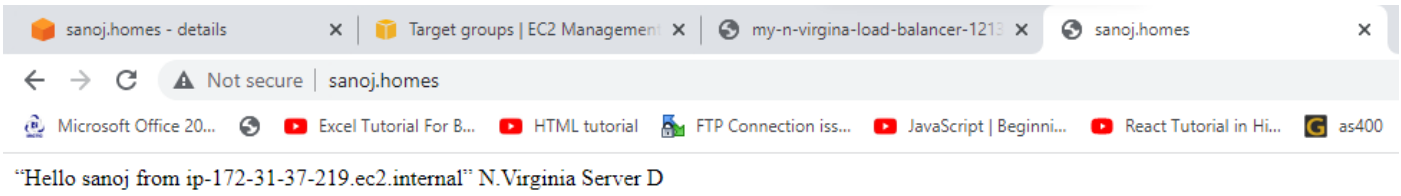


Now Lets change our location by using VPN connection Lets see what will be the output.

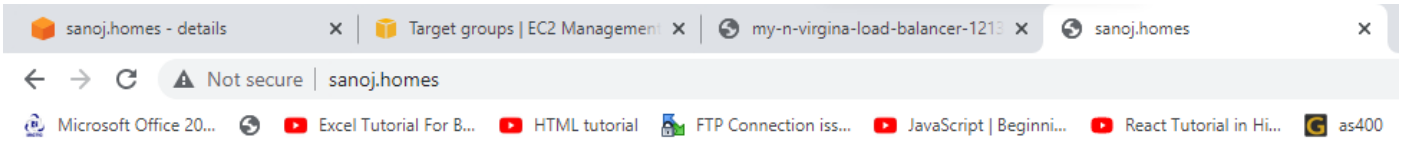
Now I am Changing the my location "Germany"



After changing the location, you can see I am getting output from N.Virgina Server.



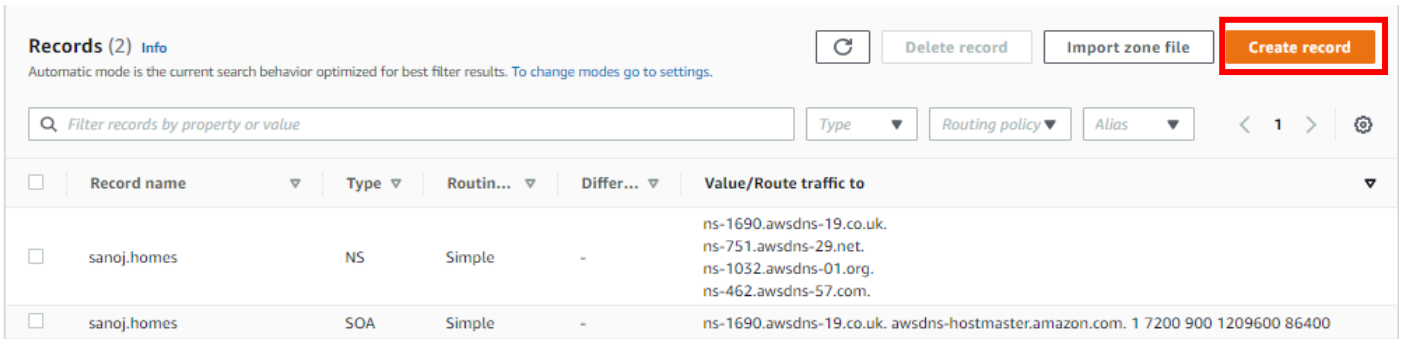
"Hello sanoj from ip-172-31-37-219.ec2.internal" N.Virginia Server D



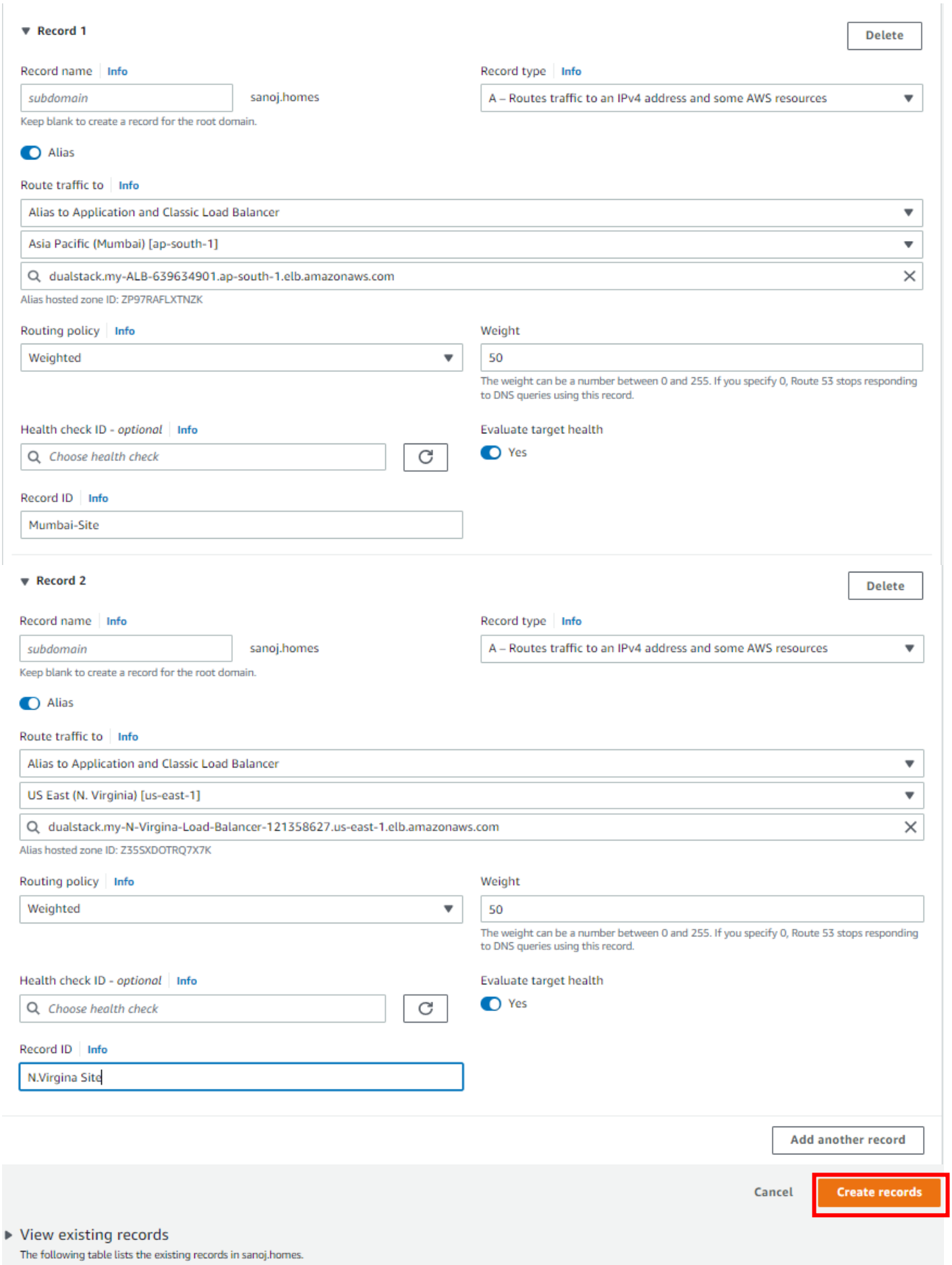
"Hello sanoj from ip-172-31-25-222.ec2.internal" N.Virginia Server C

Scenario -3:
Configuring Route 53 For Weighted Based Routing Policy:

Step 1: Click on **"Create a record"**

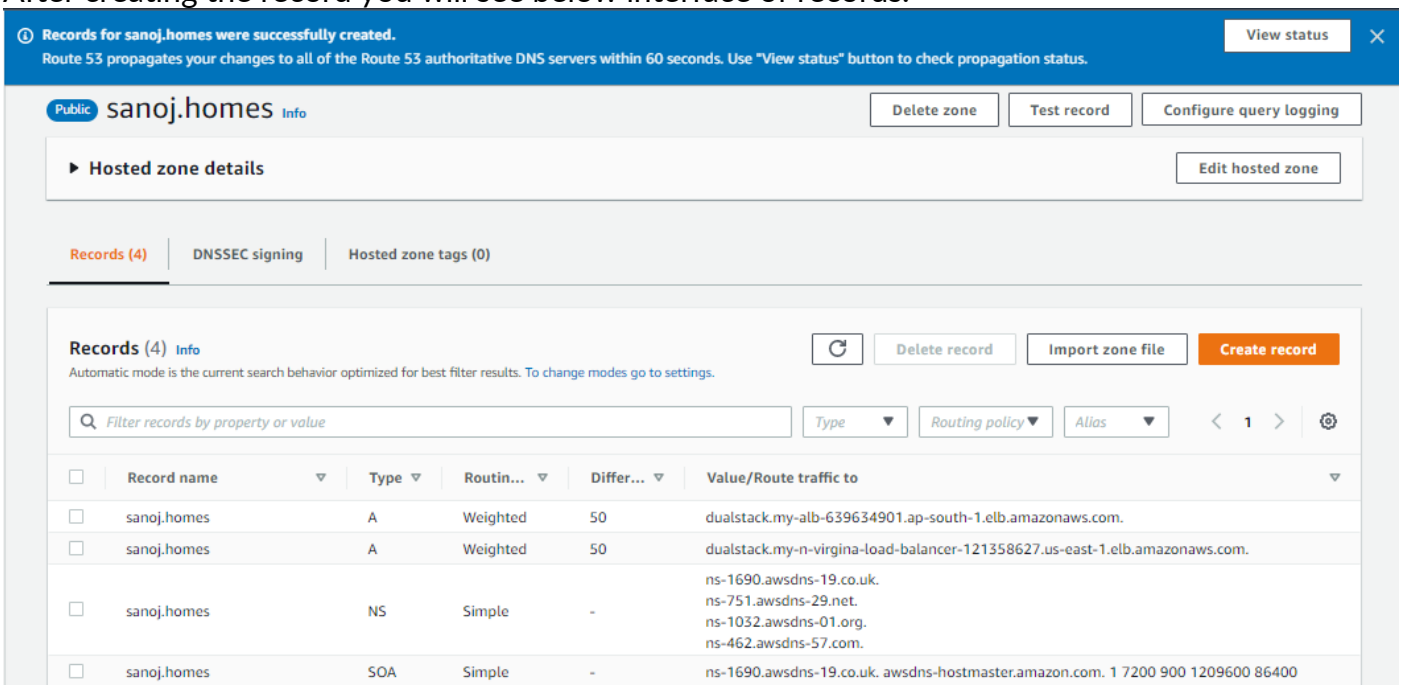


Step 2: After click on Create fill the details for weighted record for each region.



Now click on **“Create records”**

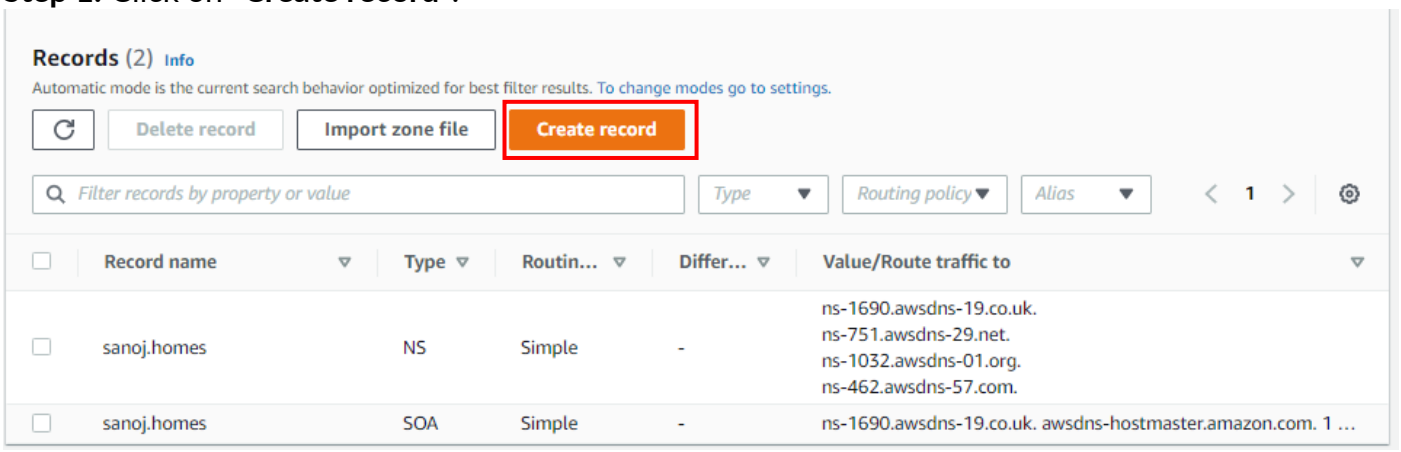
After creating the record you will see below interface of records.



Now you can hit the URL(sanoj.homes) but as of now I don't have mechanism to test it.to demonstration the load.

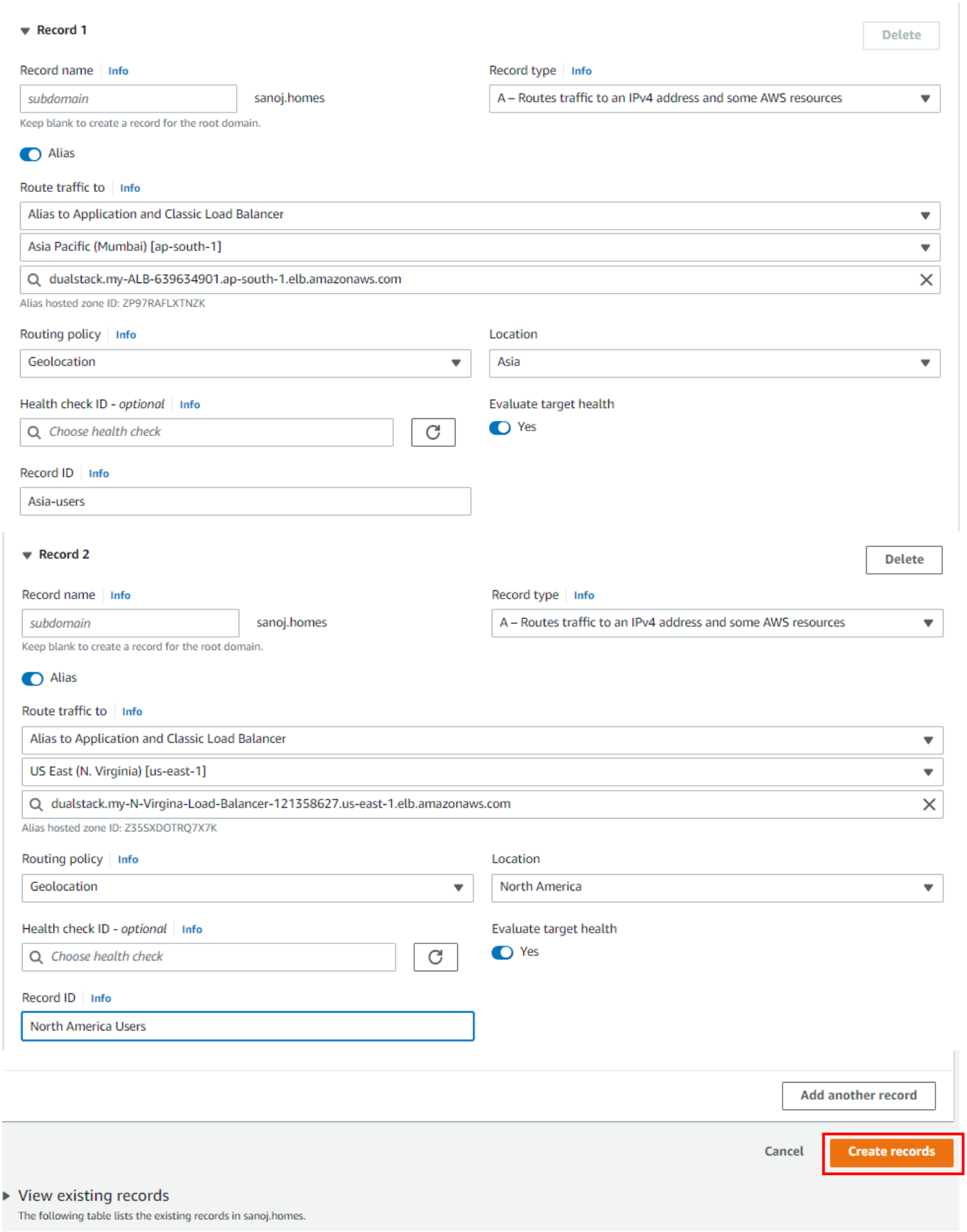
Scenario -4 :
Configuring Route 53 For Geolocation Routing Policy:

Step 1: Click on **"Create record"**.



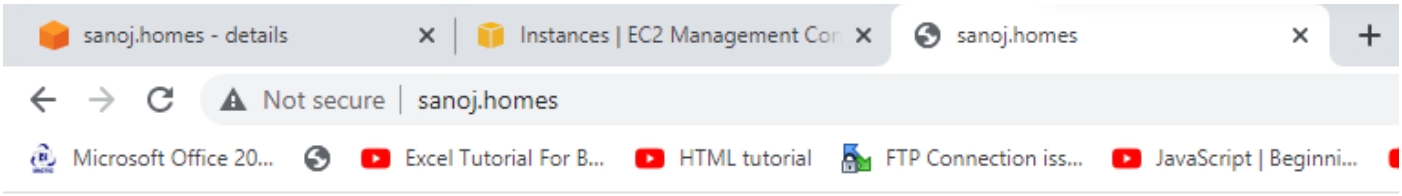
Step 2: Now click on Alias button,

- Choose your endpoint
- Choose Region
- Choose Load Balancer
- Choose Routing Policy
- Choose Location
- Give the record ID Name

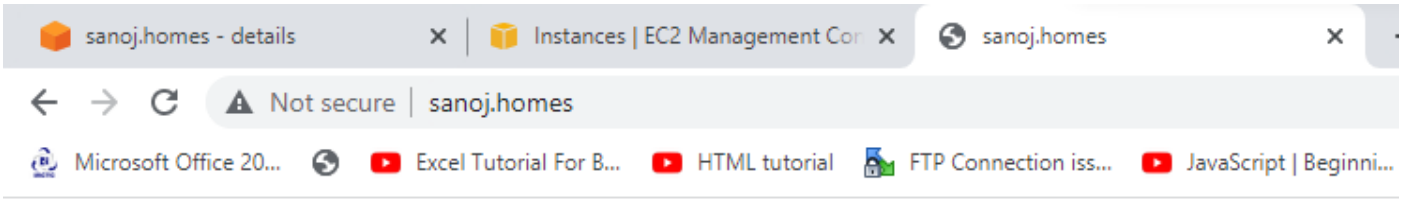


Now click on **"Create records".**
Now you can hit your domain you will see the traffic coming from different regions for reference you can see following images.

Case 1: When users from Mumbai.

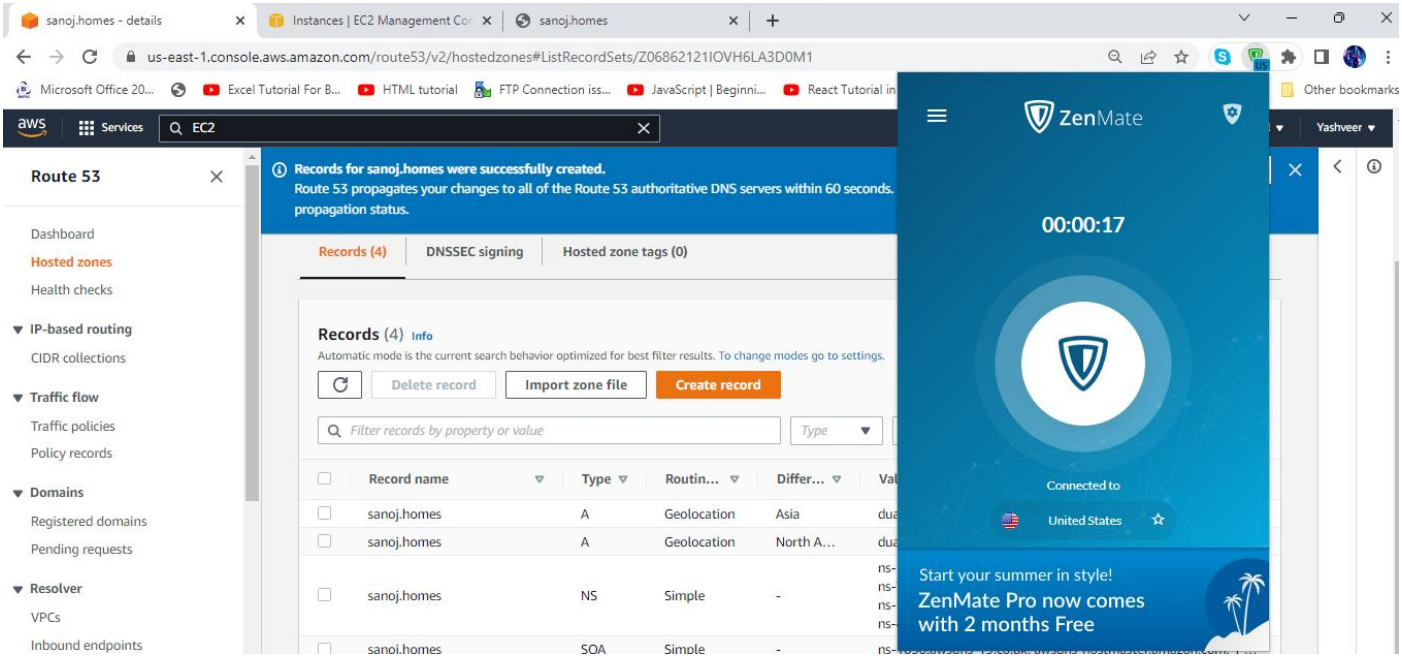


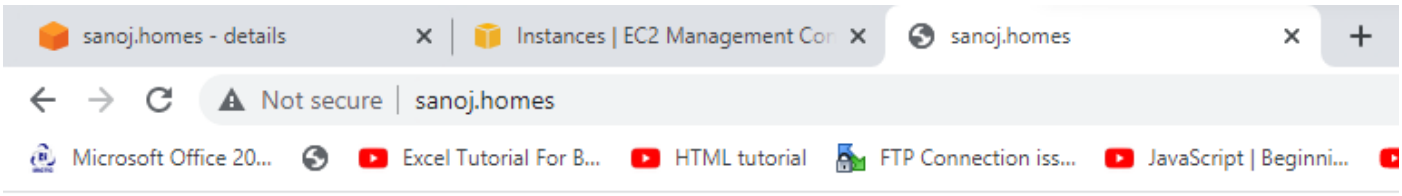
"Hello Sanoj from i-057d48400eed7c071.ap-south-1.compute.internal" Mumbai Webserver-A



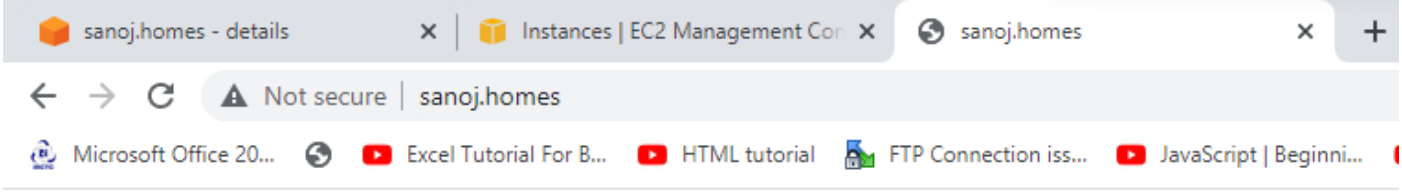
"Hello Sanoj from i-05a899824f4e395bc.ap-south-1.compute.internal" Mumbai Webserver-B

Case 2: When users from America





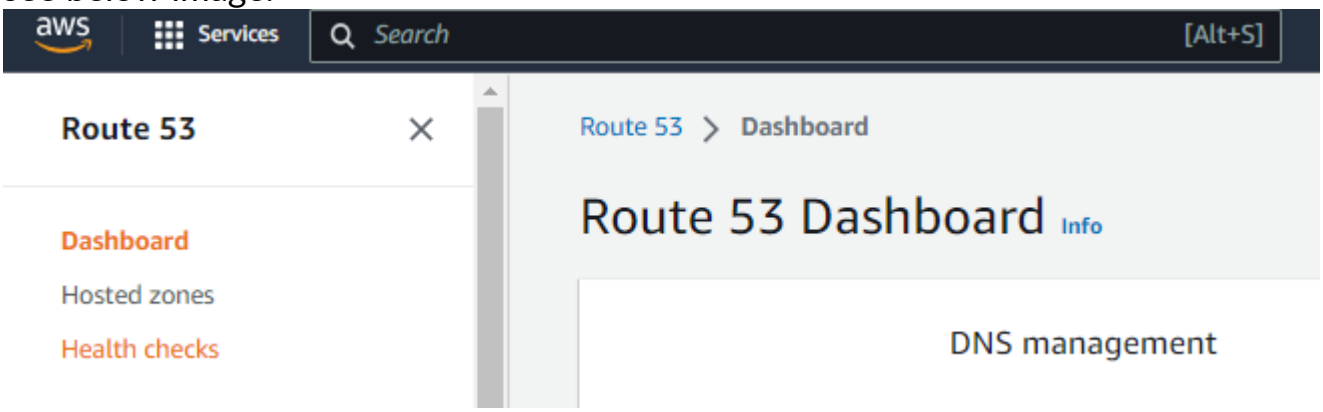
"Hello Sanoj from ip-172-31-18-84.ec2.internal" N.Virgina Webserver-B

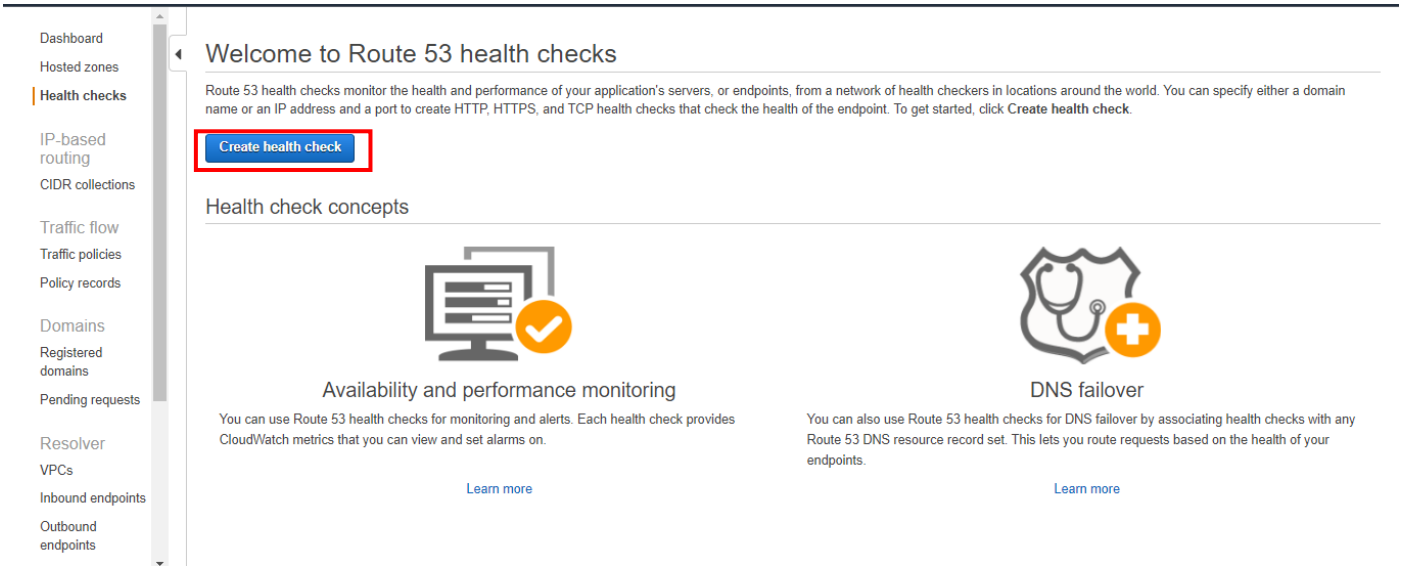


"Hello Sanoj from ip-172-31-18-148.ec2.internal" N.Virgina Webserver-A

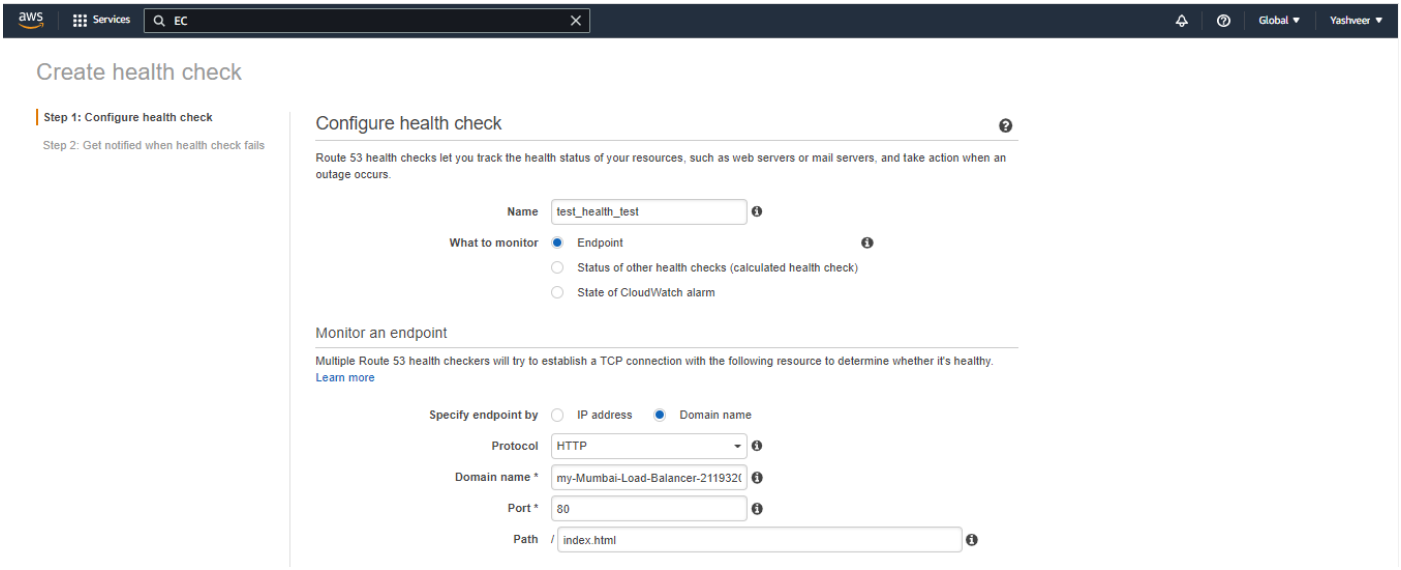
Scenario -5 :
Configuring Route 53 For Failover Routing Policy:

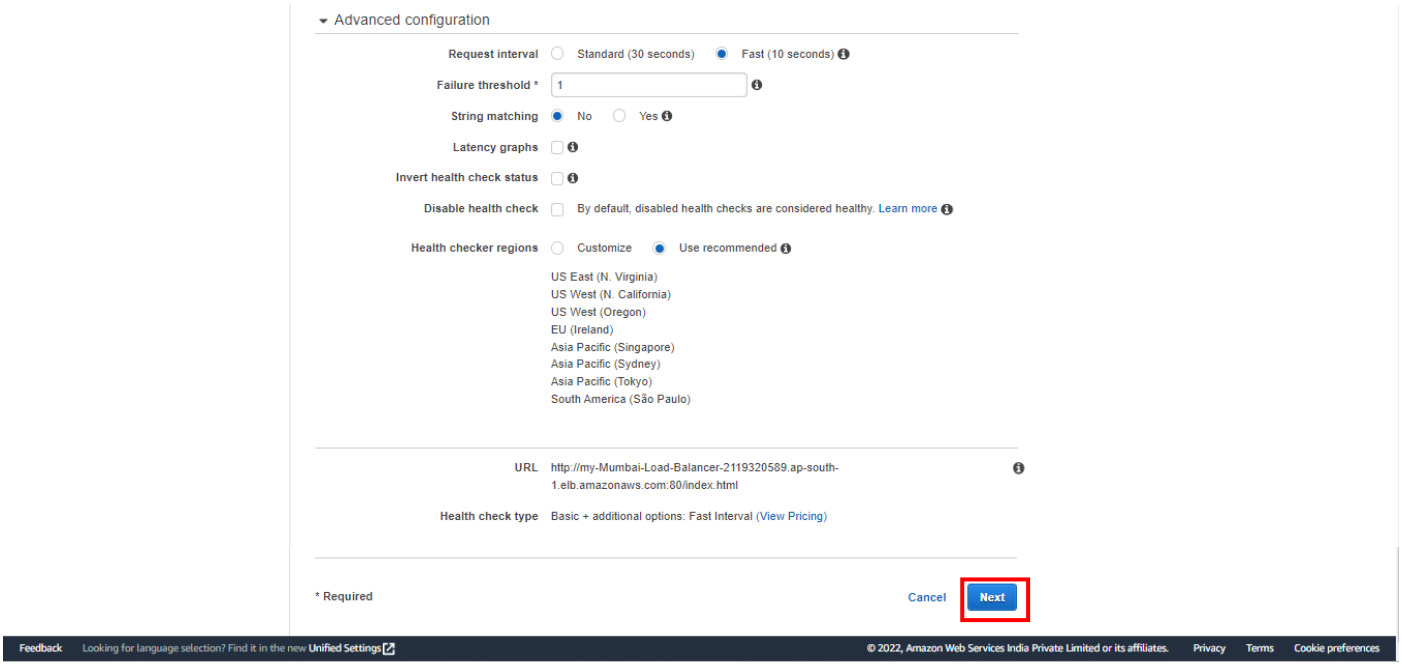
Step 1: Create a health check by click on the left pane of console for reference you can see below image.



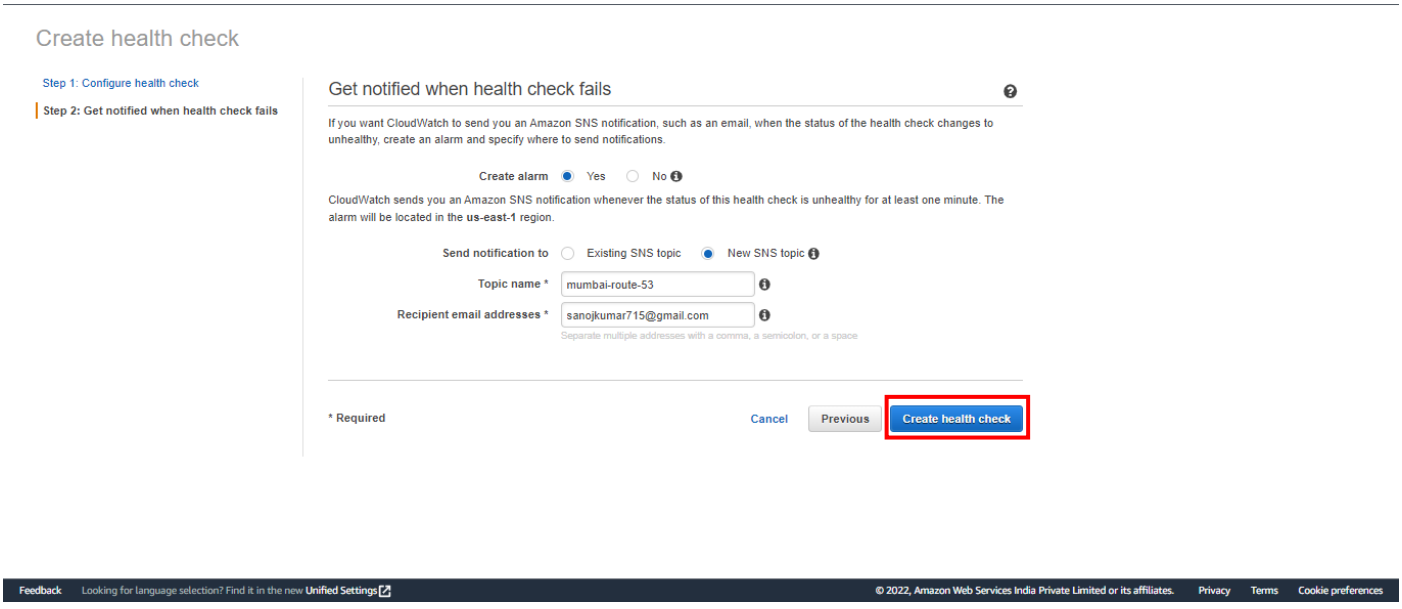


Now click on the **"Create health check"**

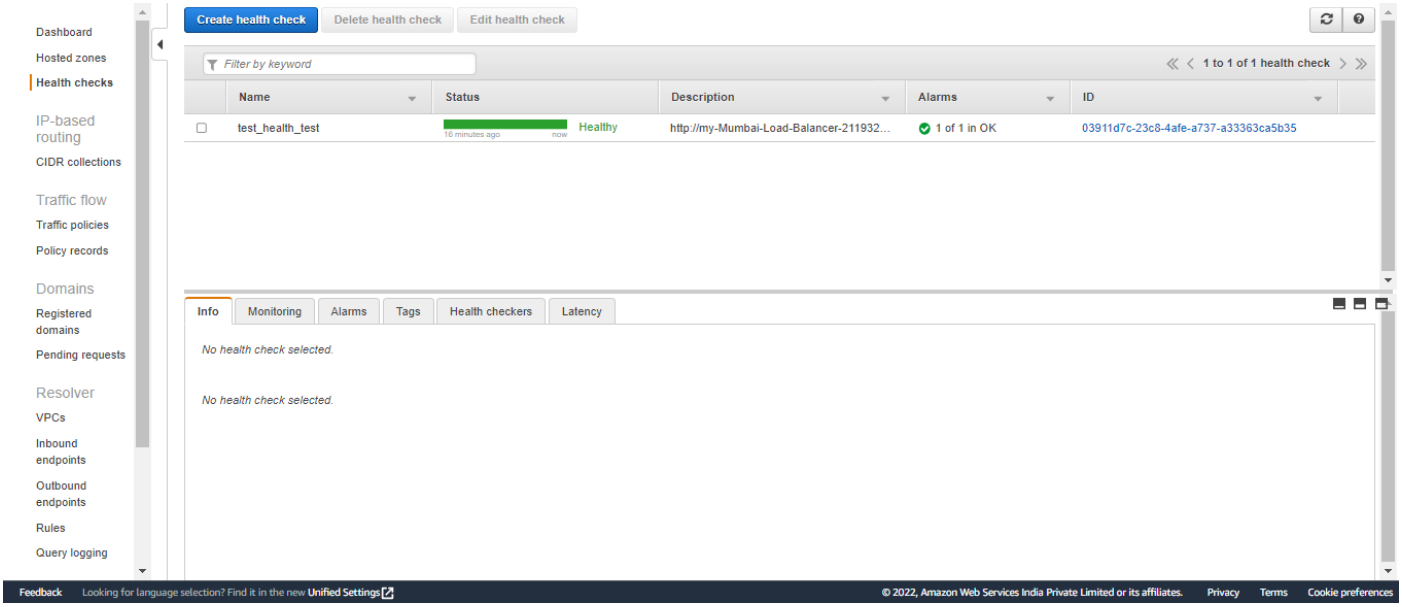




Now click on **“Next”** after that you will see below screen.

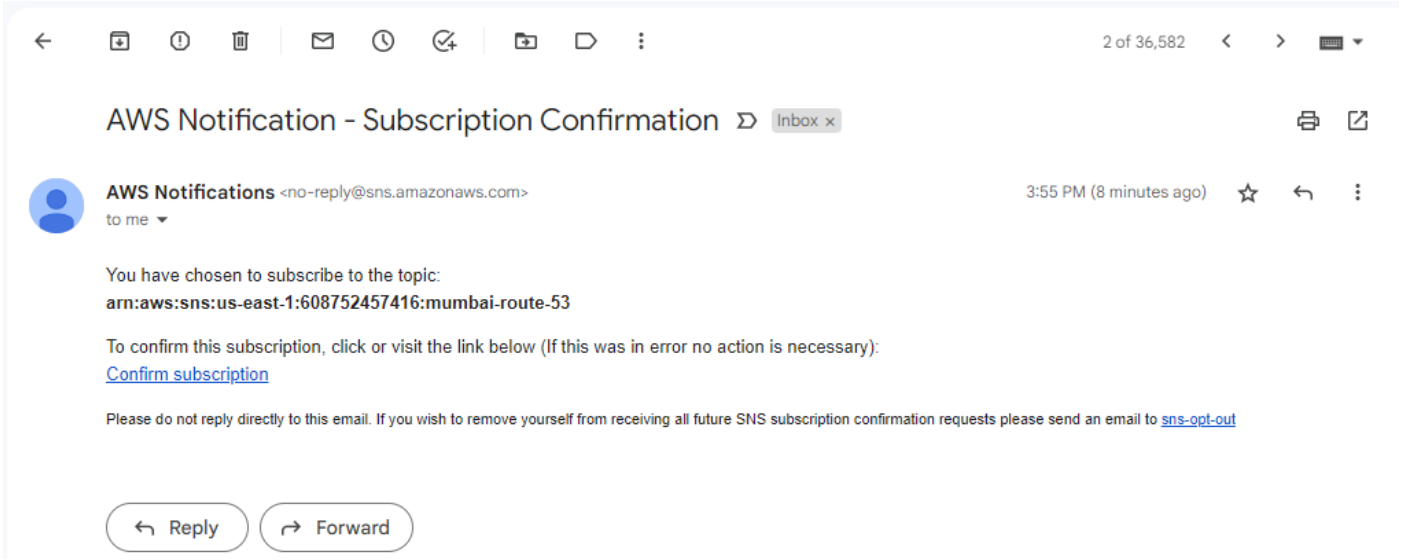


Now click on **“Create Health check”** now you will see below interface.

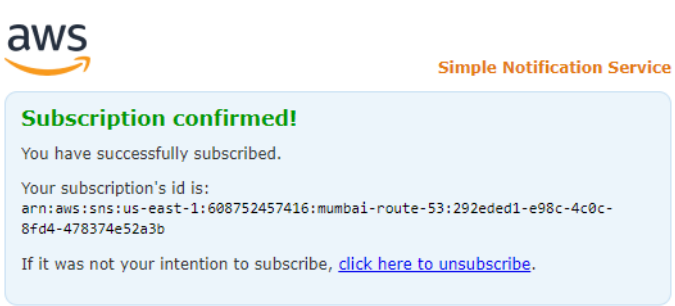


After that you will get an email from aws to subscribe email notifications. just click on Confirm subscription.

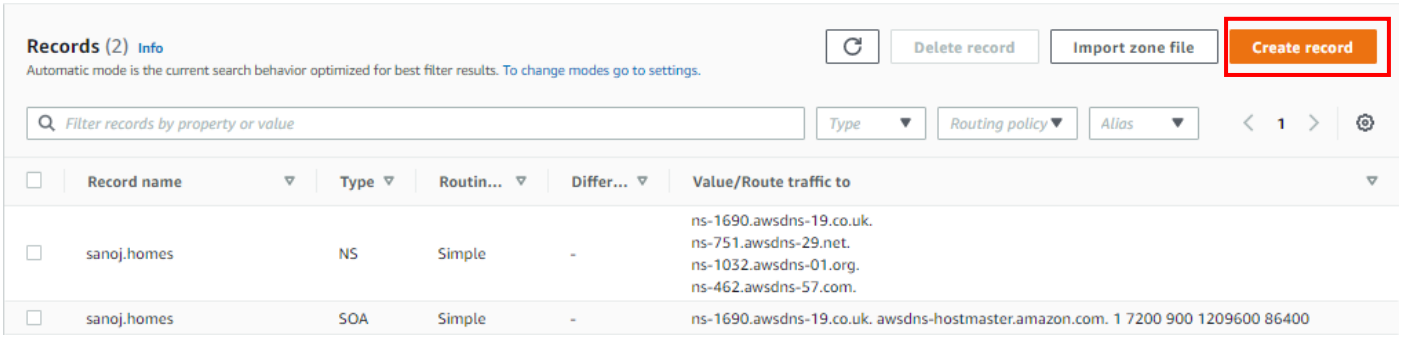
Note: if you do not get any AWS Notification please check your spam box of your email.



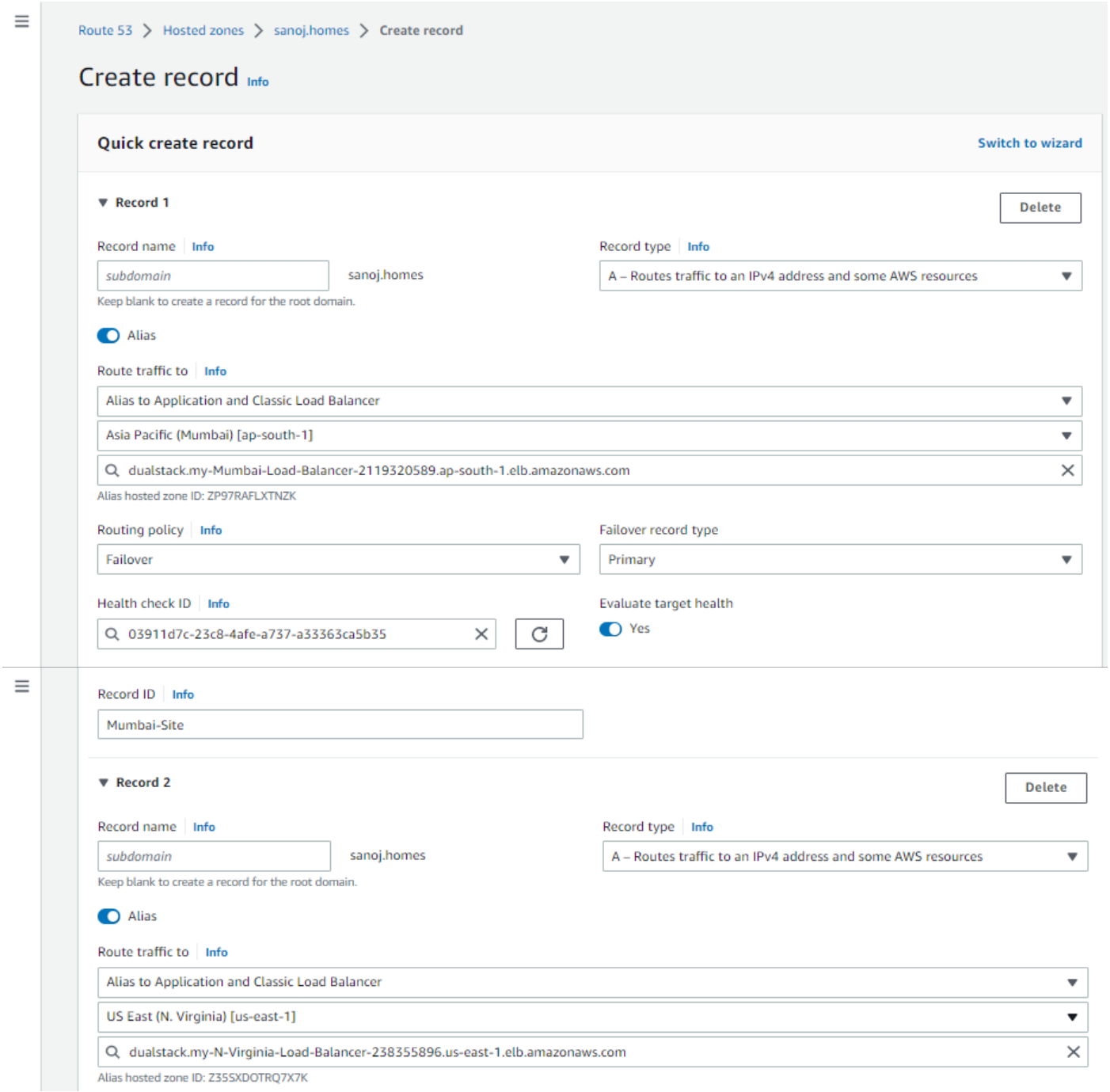
Just conform the subscription.
After confirming subscription, you will see below screen.

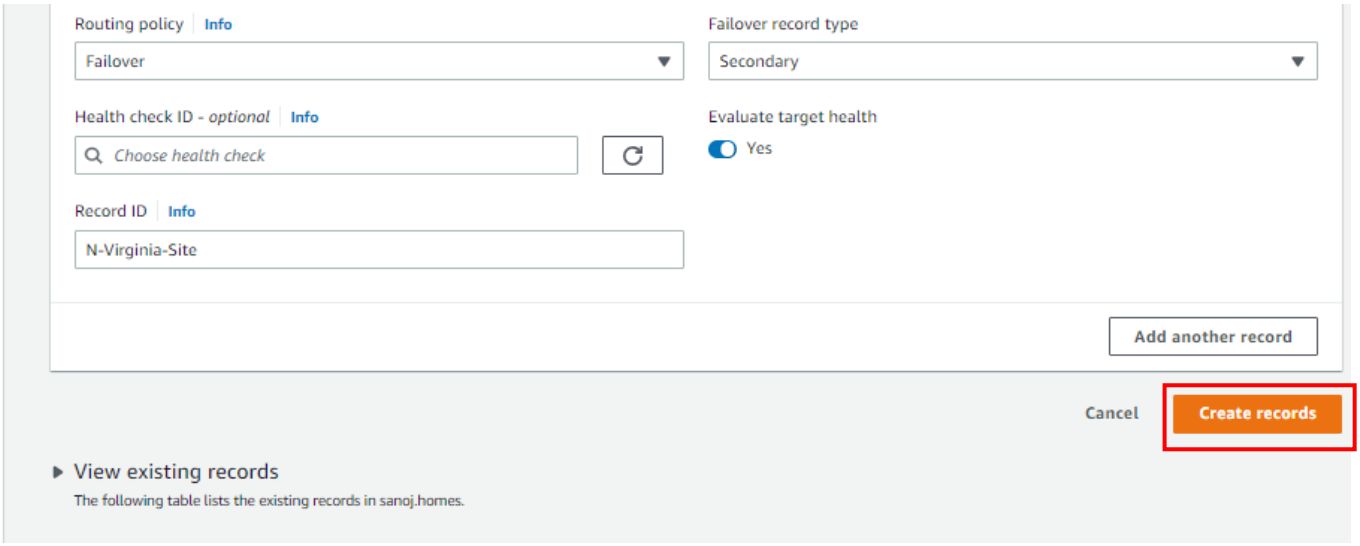


Now go to your hosted zone and create a record.

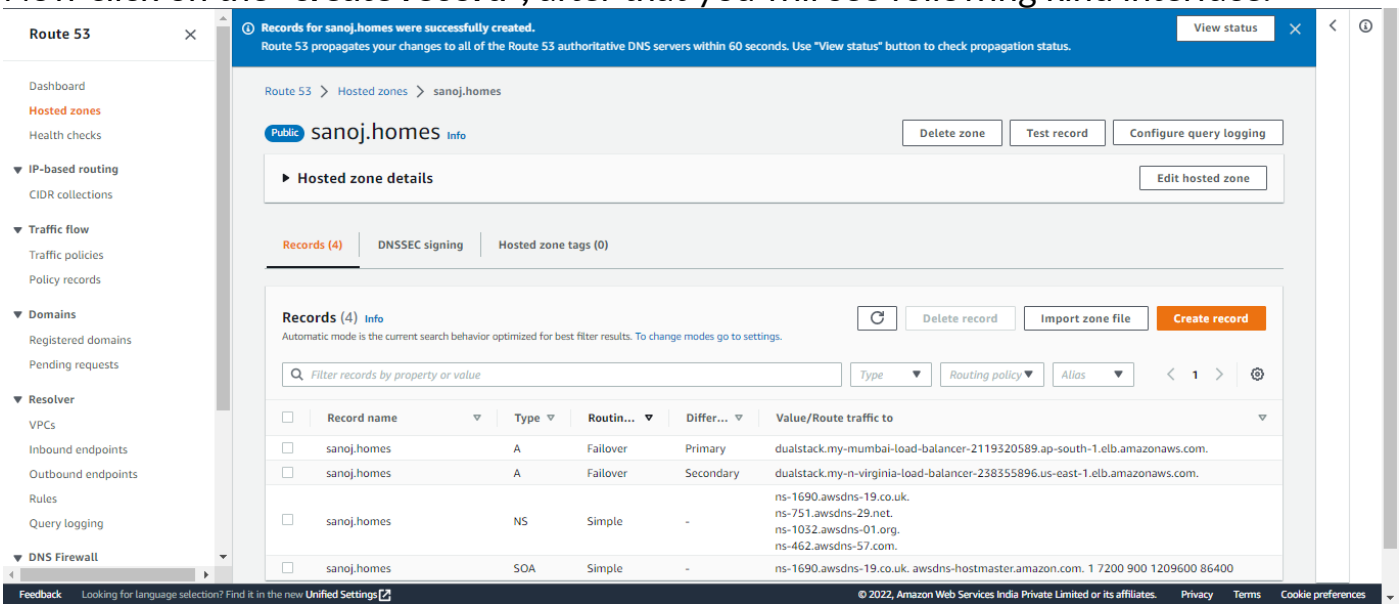


Now create a record for that just click on the **"Create Record"** and you will see following interface and fill the details accordingly.

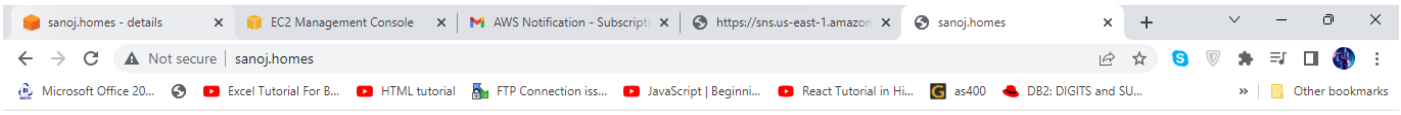


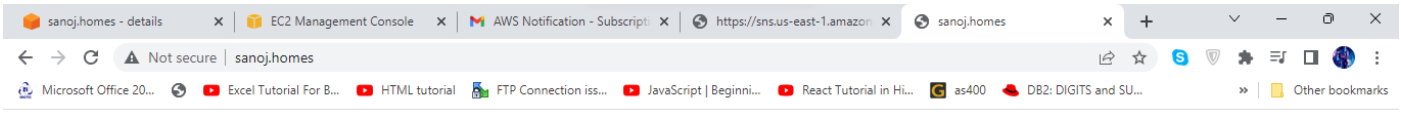


Now click on the **"create record"**, after that you will see following kind interface.

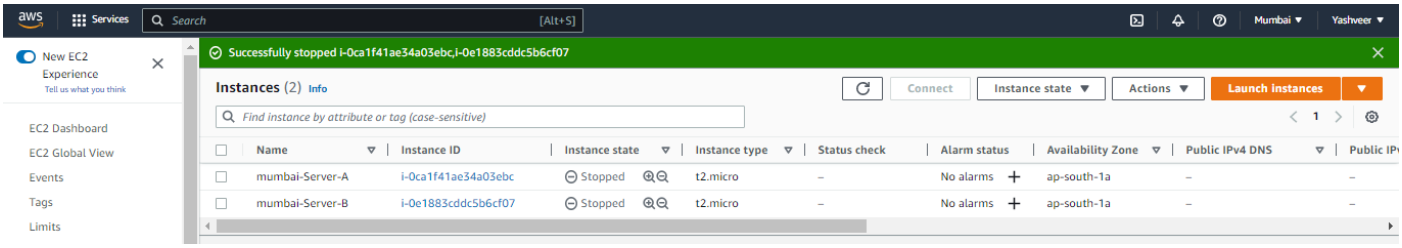


Now hit your domain Name (sanoj.homes) you will get your request from your primary region in this case I have set Mumbai is primary so always traffic will get from Mumbai.

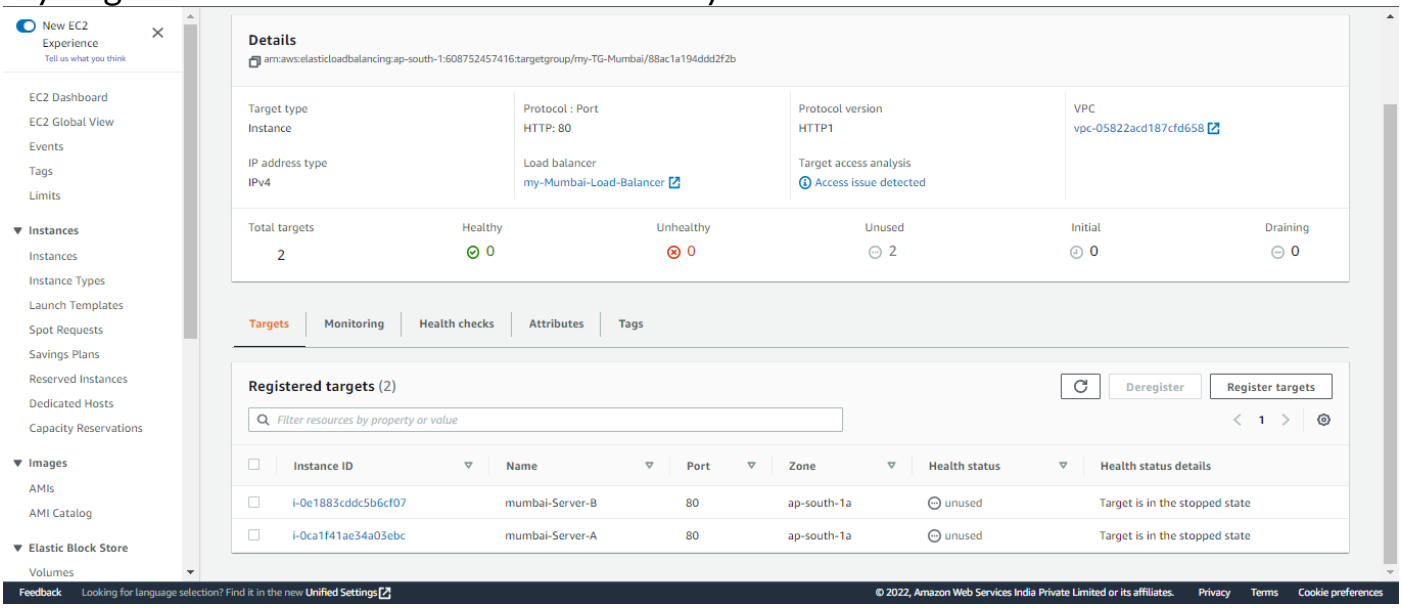




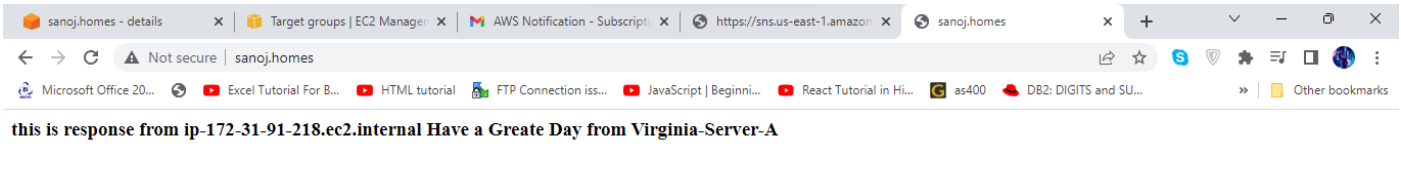
Now to test the failover I am stopping all servers from Mumbai side .

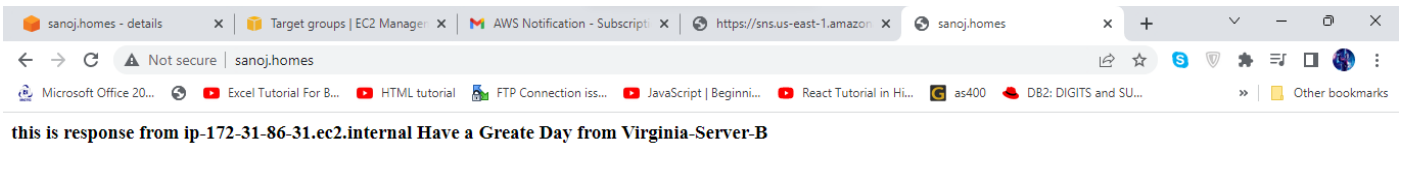


If you go to the Load Balancer Health check you will see Health status is **“unused”**

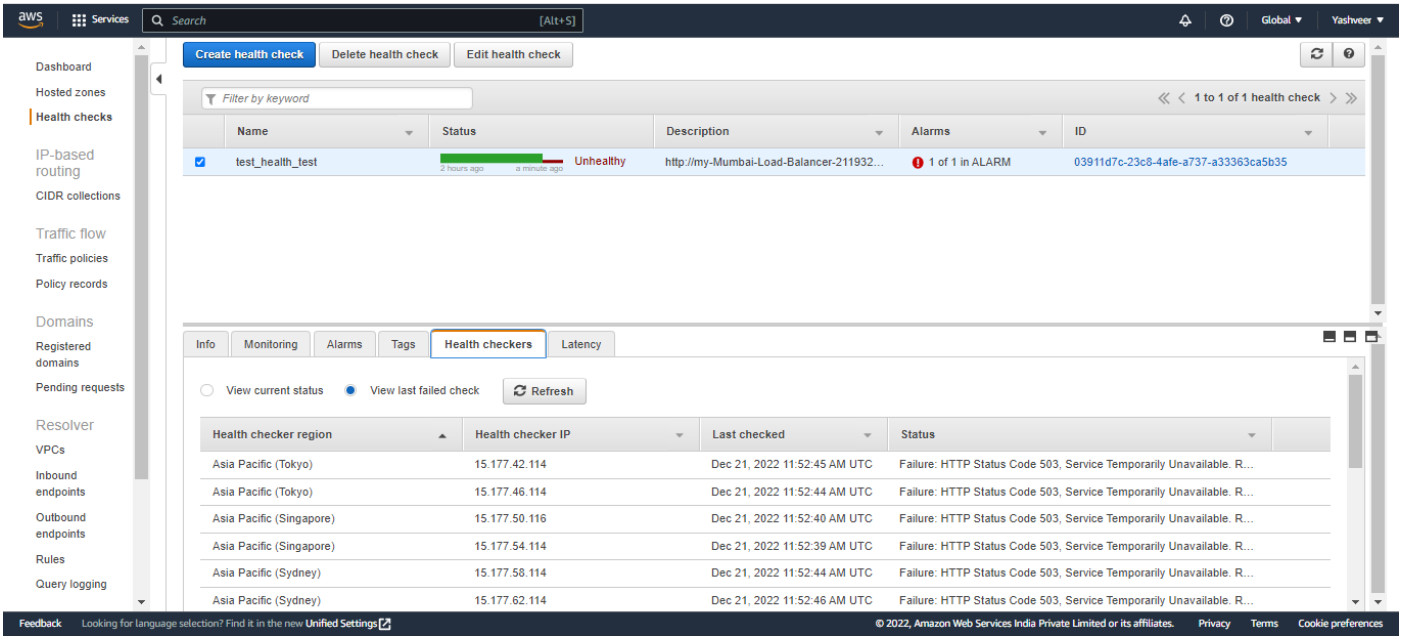


Now Hit again your domain name it must be service will get from N.Virginia Servers.

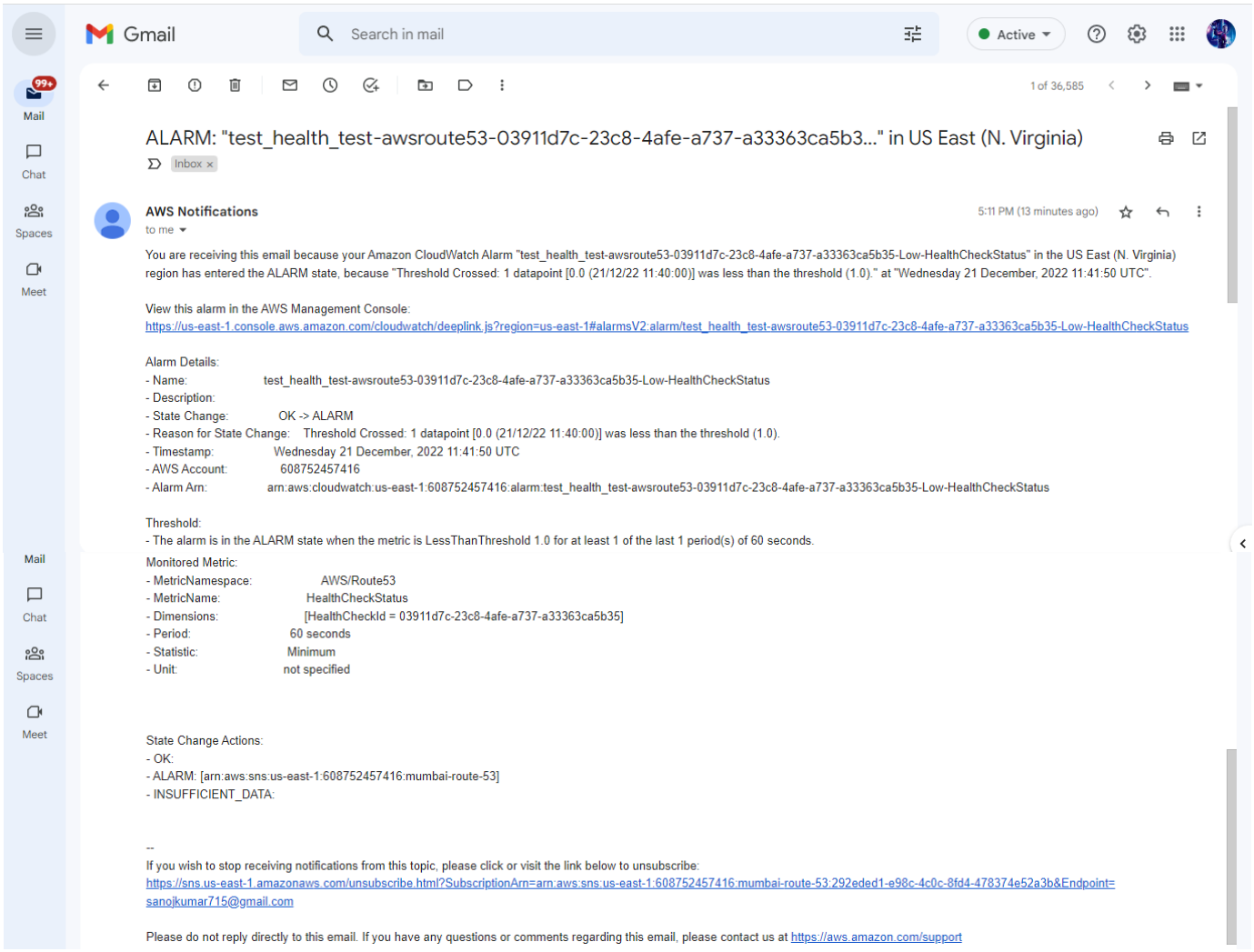




And if you go to the Route 53 you will see the health check in unhealthy status.



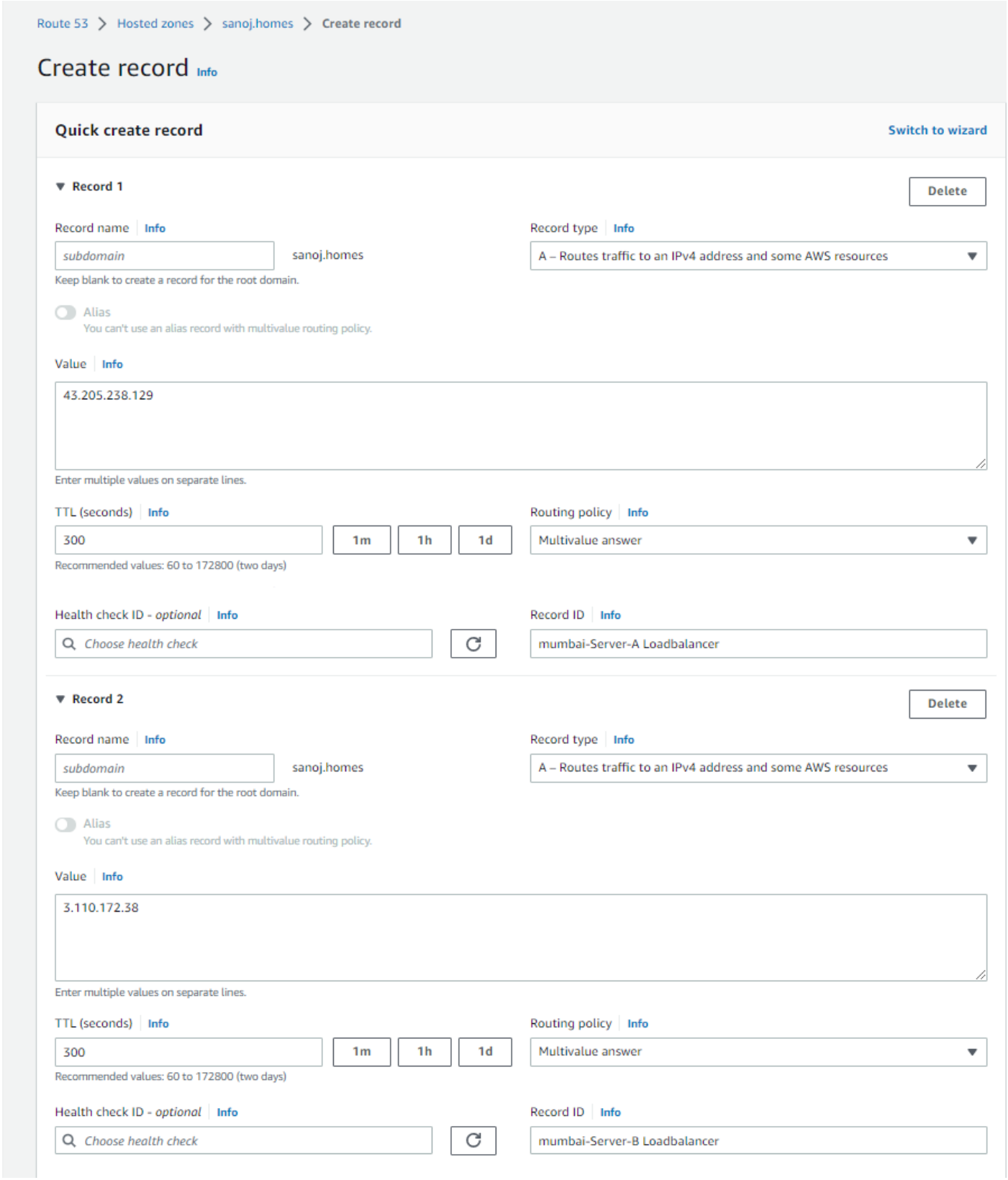
And you will also get an email from AWS.

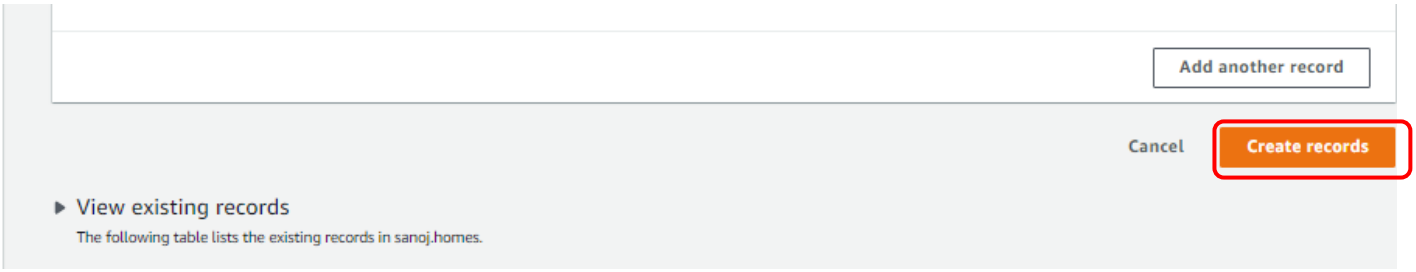


Scenario -6 :
Configuring Route 53 For Multi-value Answer Routing:

Step 1: Go to the hosted zone and "**create the record**".

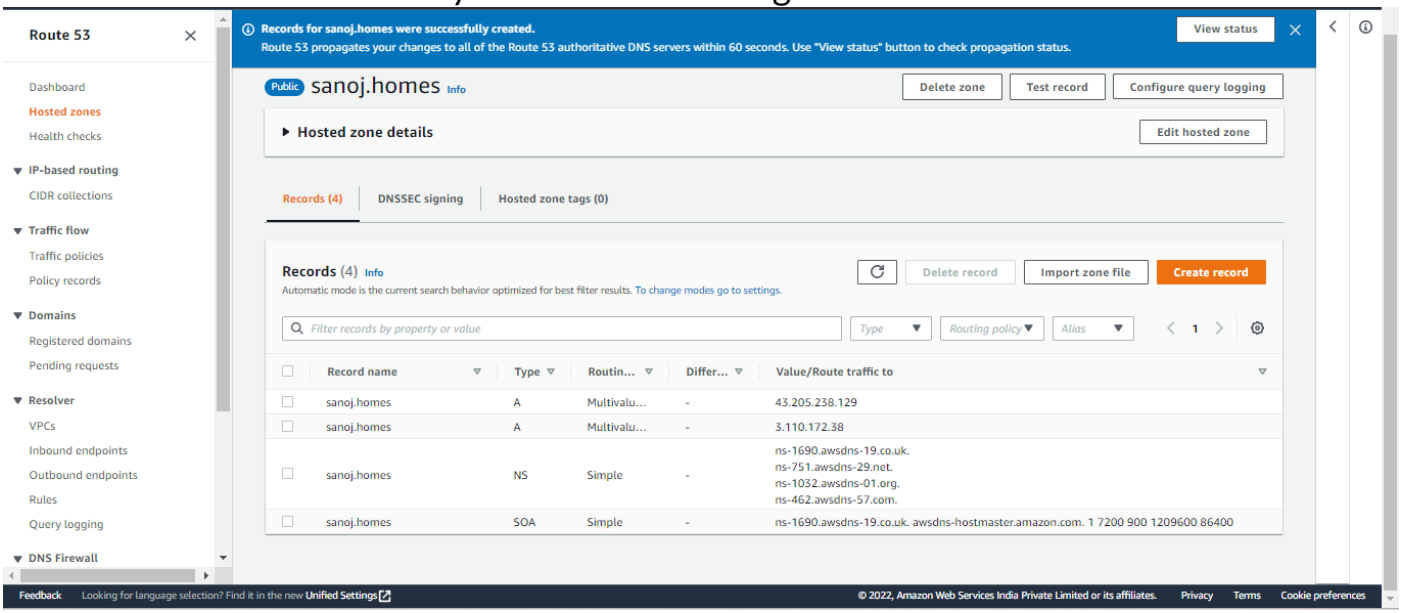
Here i Have passed public IP address of Mumbai Server of both instances. as you can see in the below screenshot.





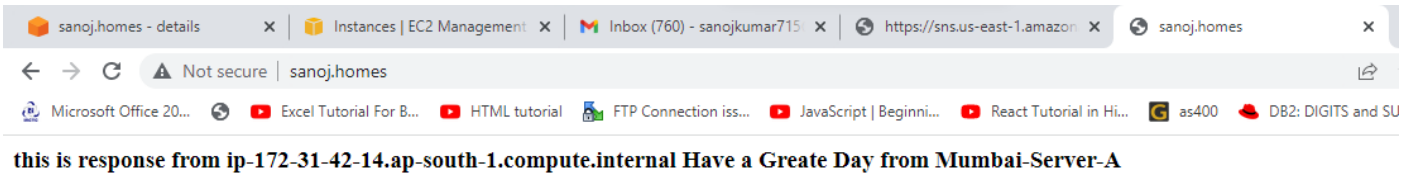
Now click on **“Create records”**.

After Click on create record you will see following interface.



Now hit your domain name in your browser you will see the request coming from server A and server B from Mumbai side in round robin fashion.

Note: As I mentioned 300 seconds as a TTL so every 300 seconds it will refresh and you will get traffic response from different- different Server of Mumbai.



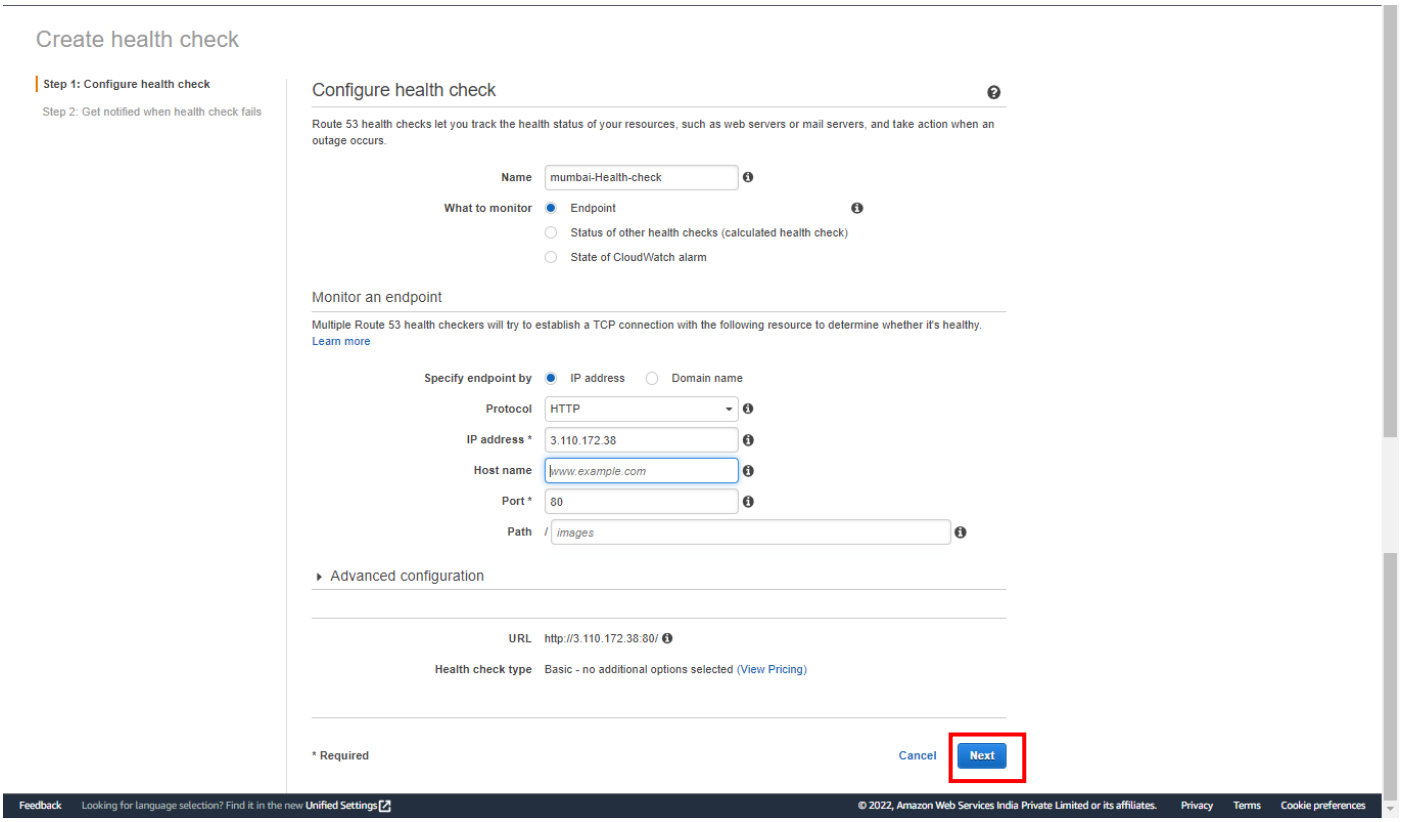
Scenario -7 :
Configuring Route 53 For GeoProximity Routing Policy:

Note: By default, you can not see GeoProximity in Routing Policy of Route 53.

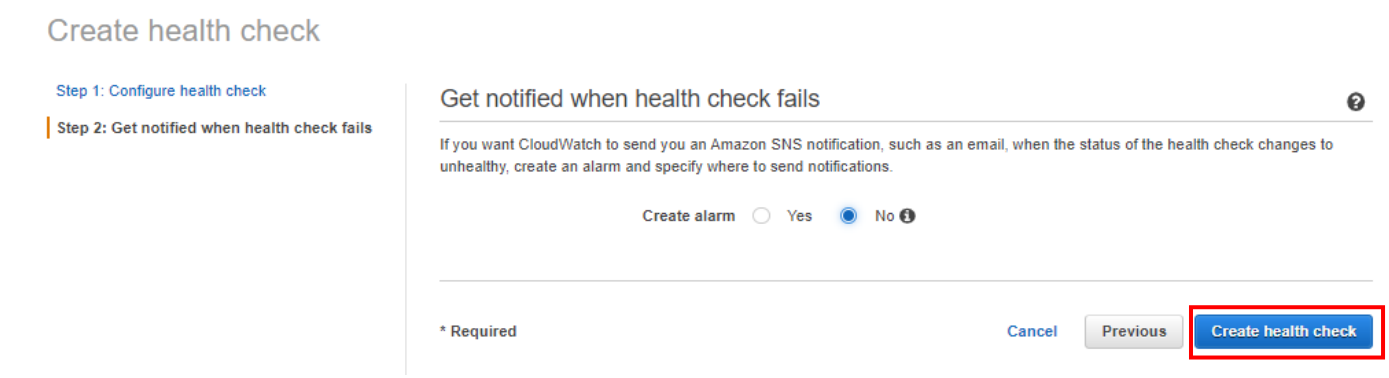
To Set GeoProximity you have to go :Traffic polices” In Route 53 and create Traffic Policy.

Step 1: Lets First Create two health checks for Mumbai and Virginia region.

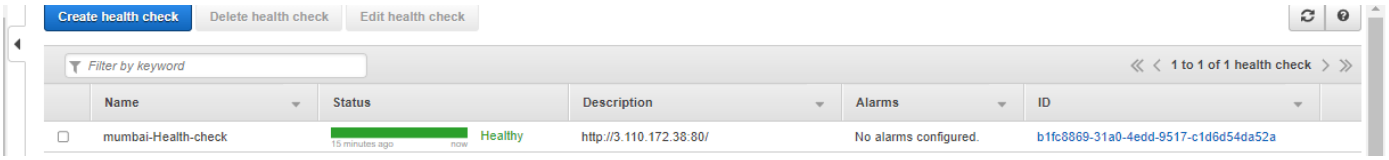
For Mumbai:



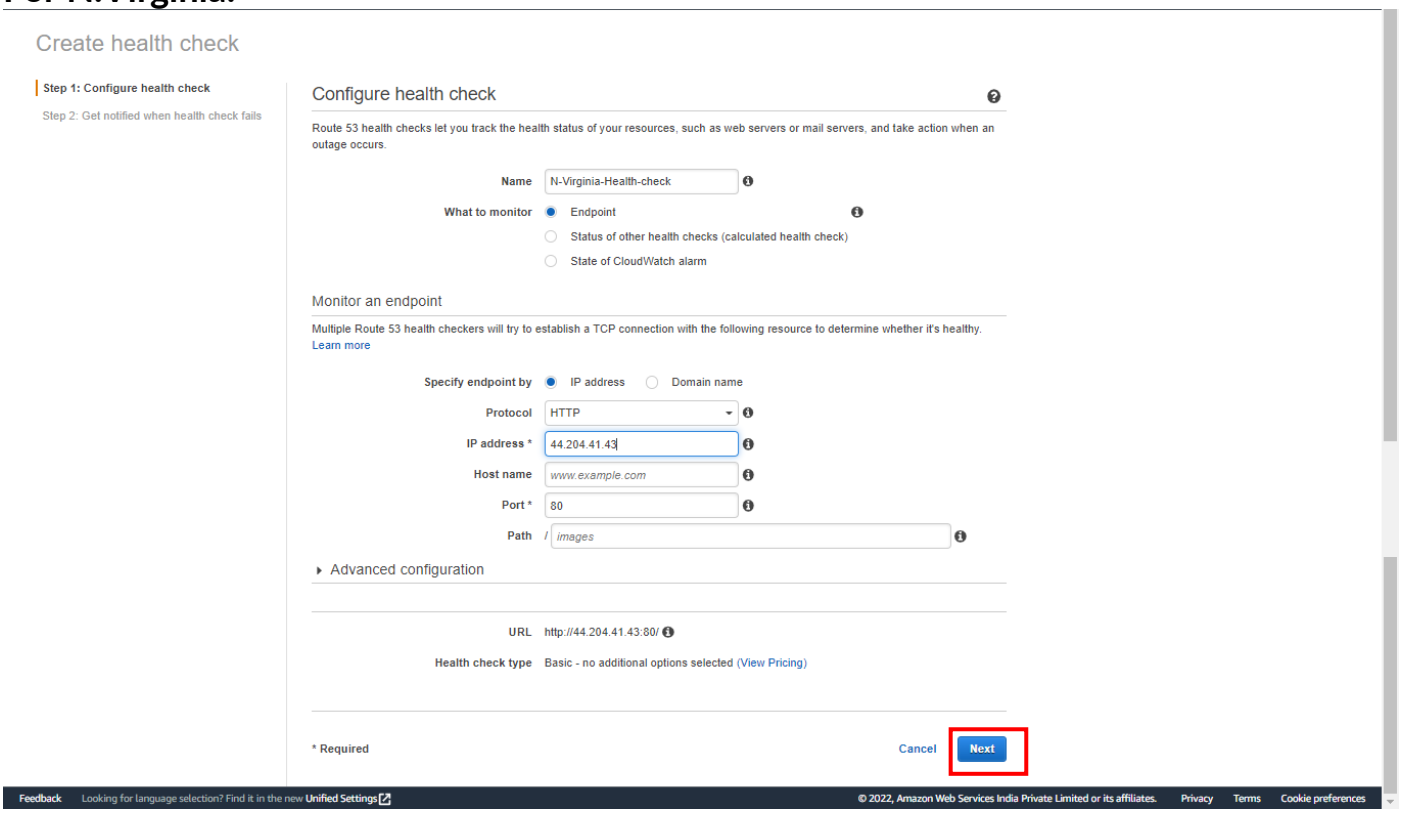
Click **"Next"**.



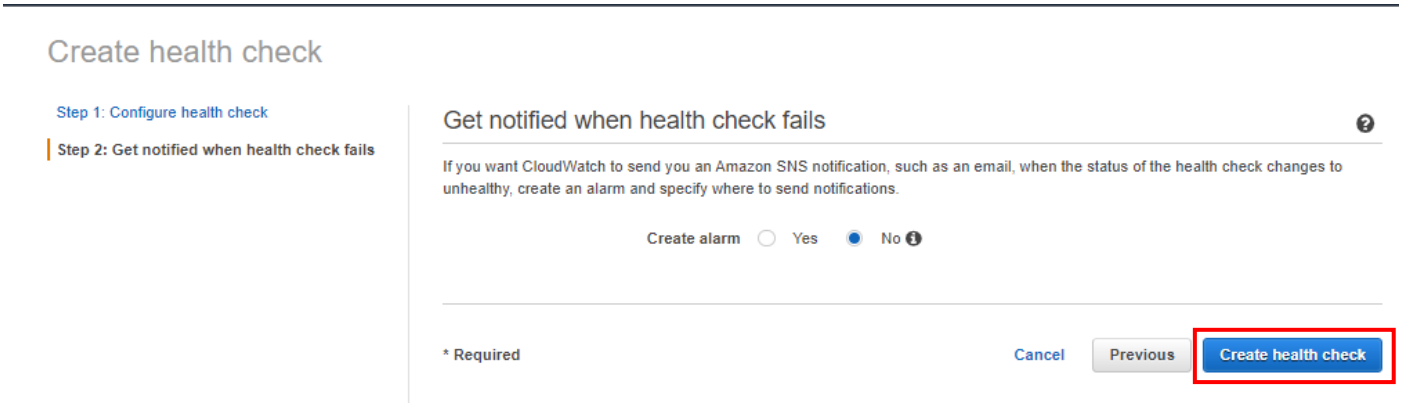
Click on **"Create Health Check"**



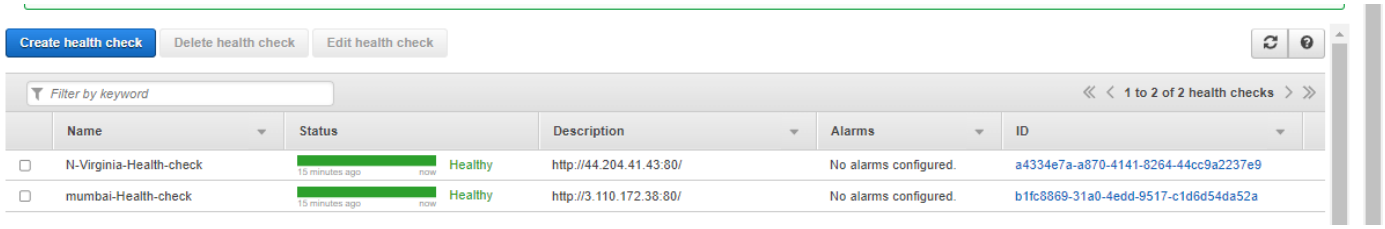
For N.Virginia:



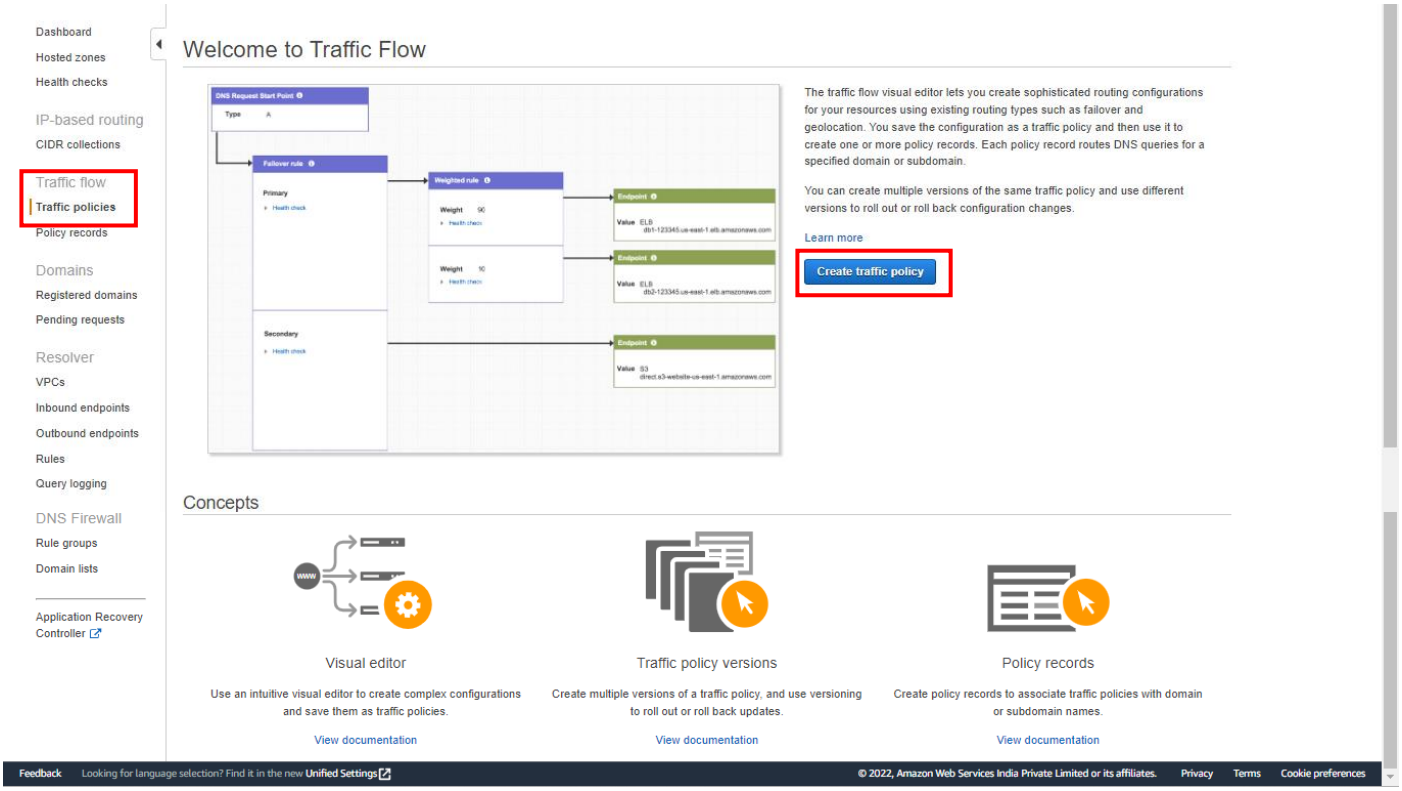
Click on **"Next"**



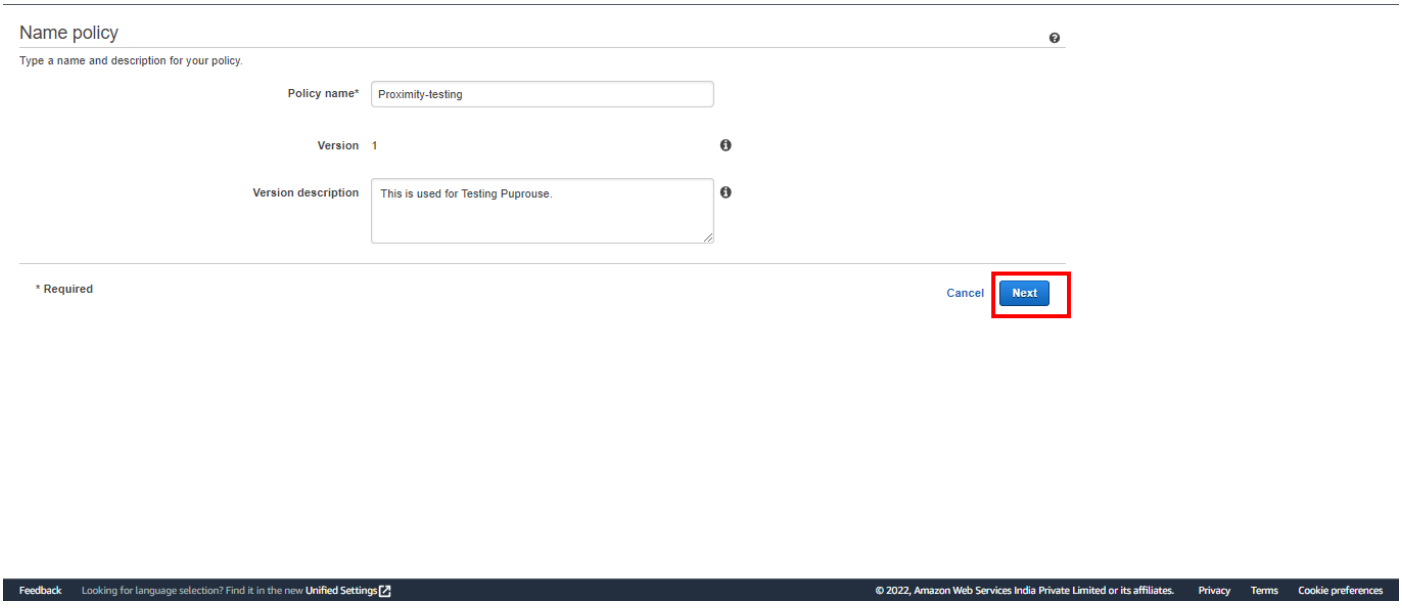
Click on **"Create health check"**



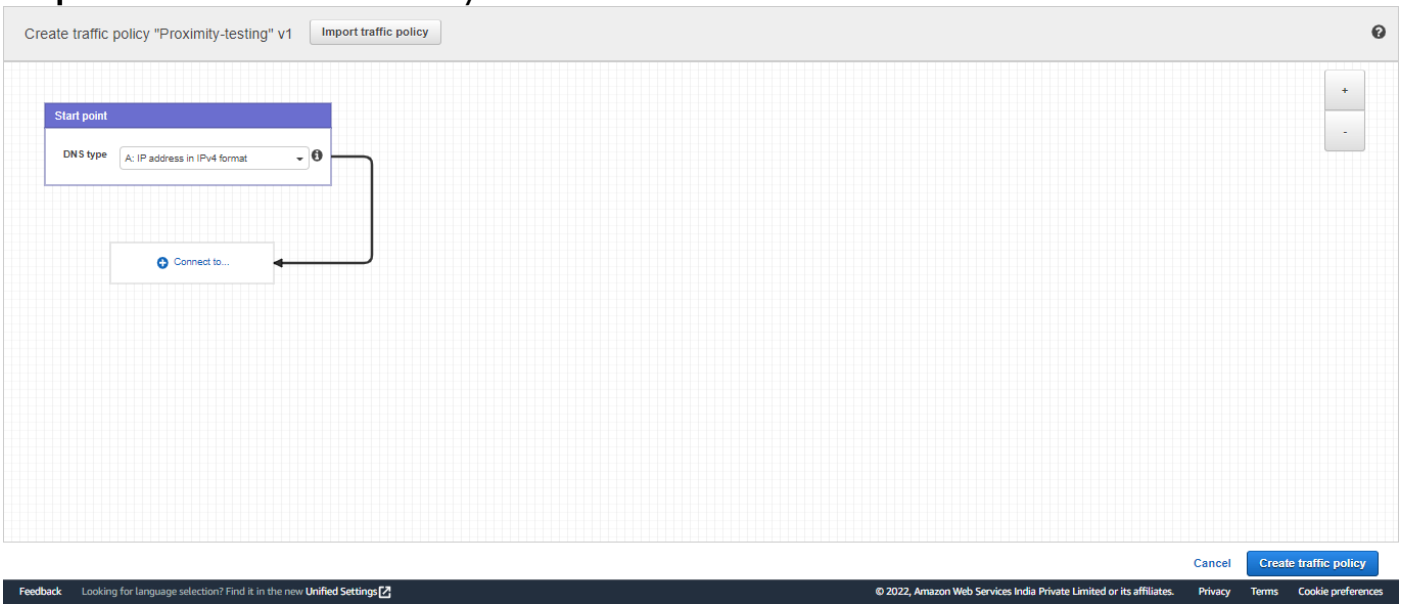
Step 2: Click on **“Traffic Polices”** you will see following interface.



Step2: Click on **“Create traffic policy”.** After filling the details as you can see in the below image and now click on Next.

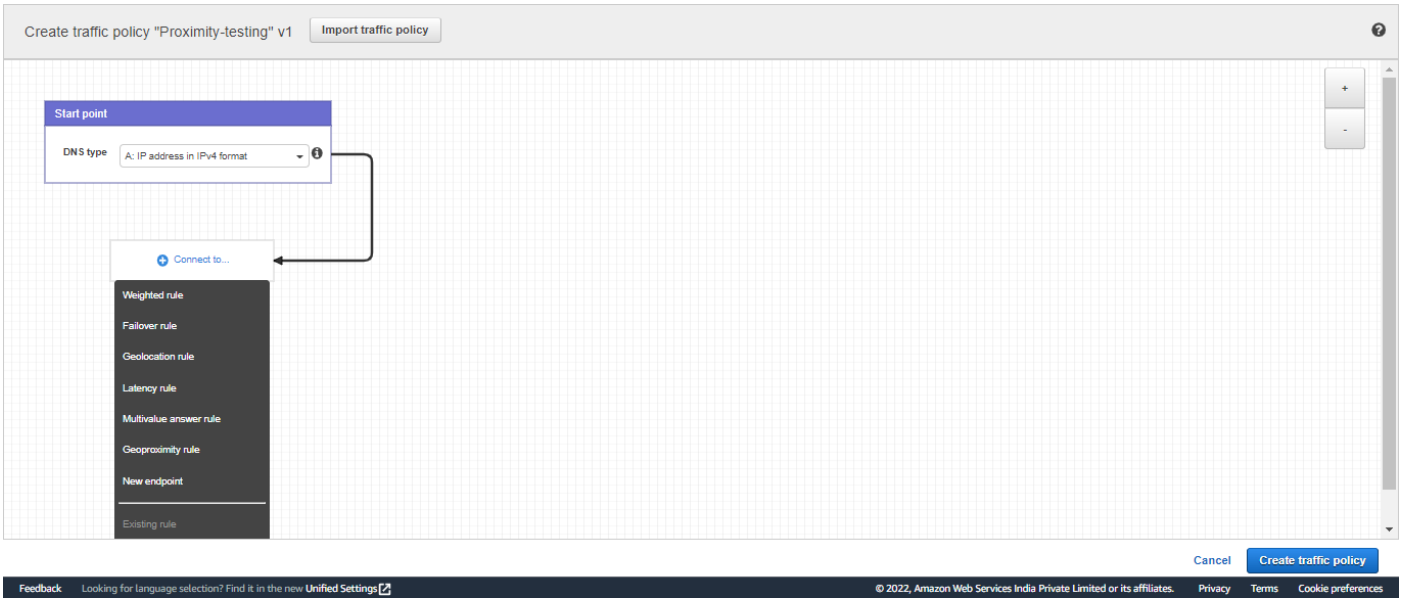


Step3: After click on **“Next”** you will see below interface.

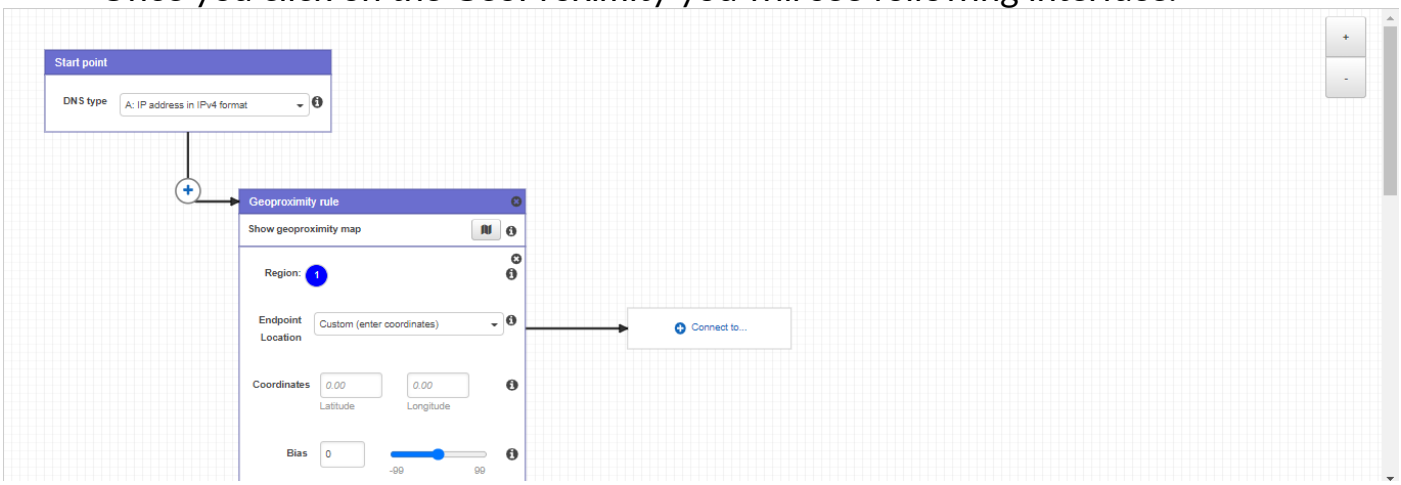


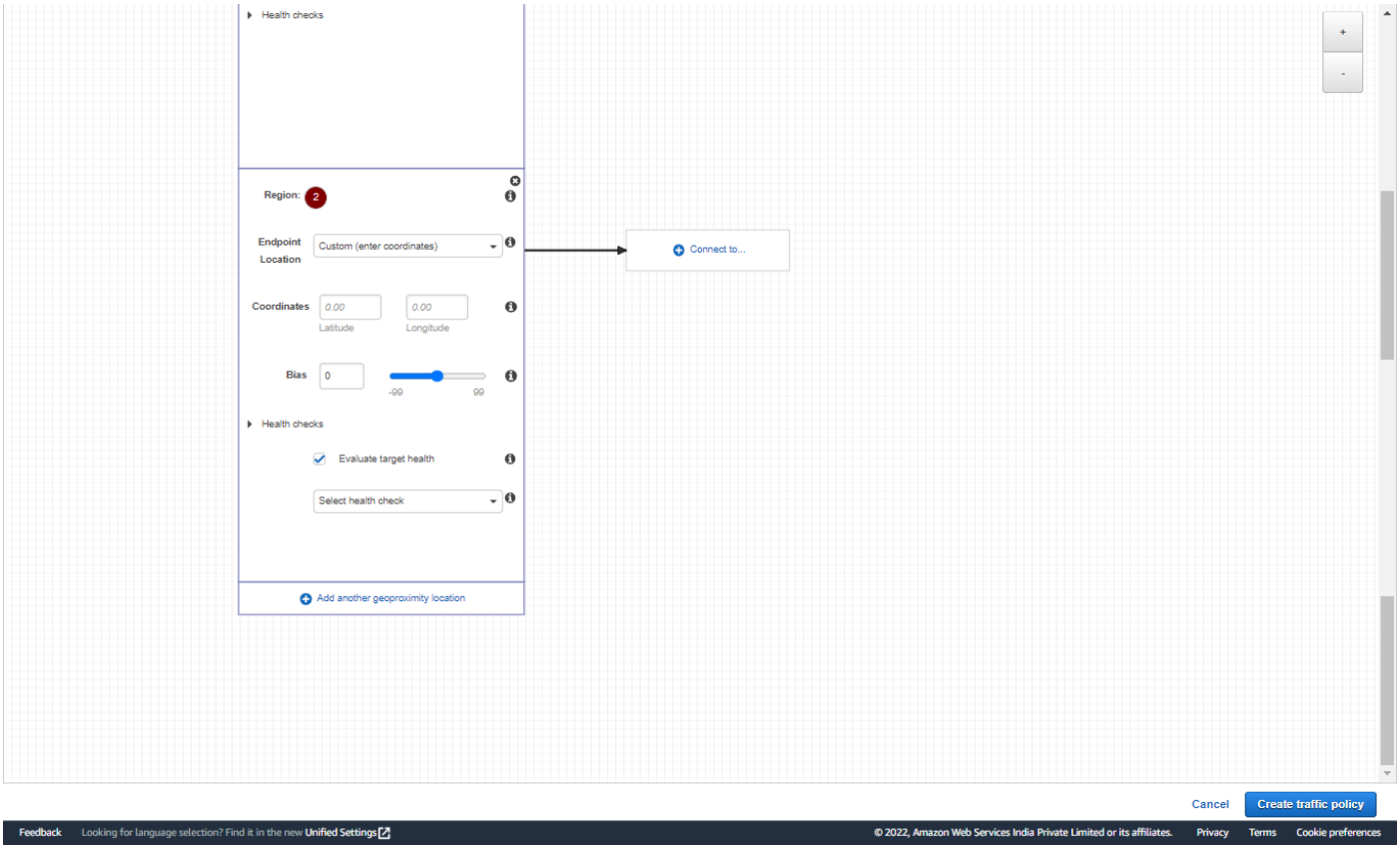
Step 4: As of now I am selecting **DNS type Record A** you can choose accordingly.

- Now click on the **“connect to”**, once you click on the you will get lots of routing policy you have to select **“Geoproximity rule”**. for reference you can see below image.



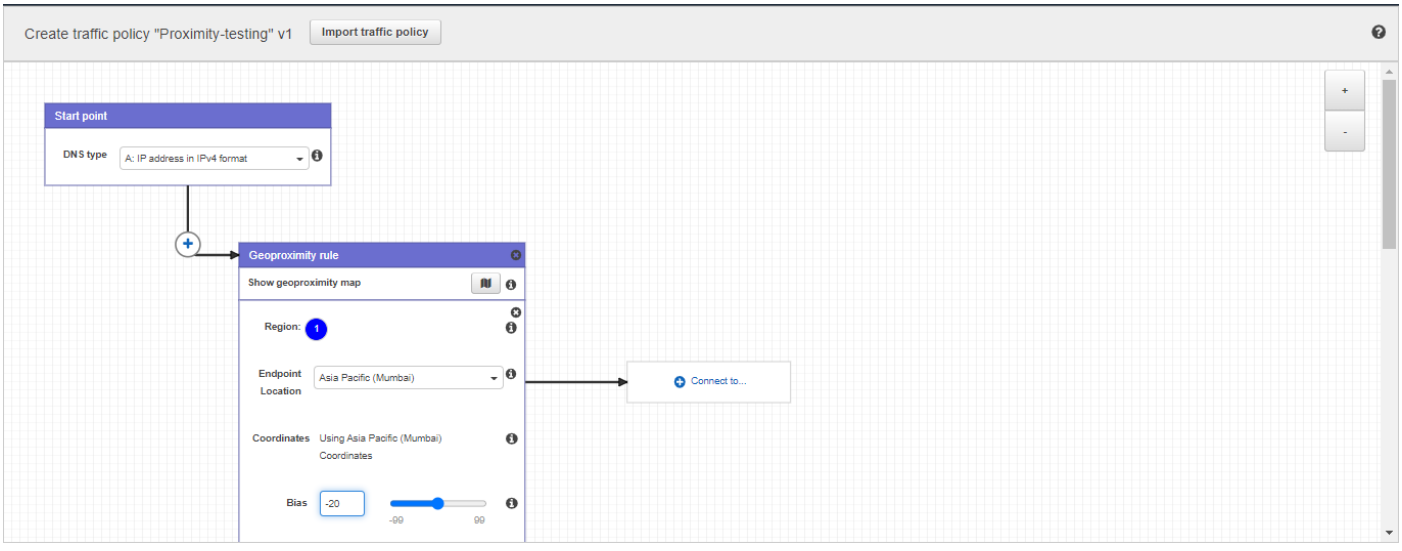
- Once you click on the GeoProximity you will see following interface.

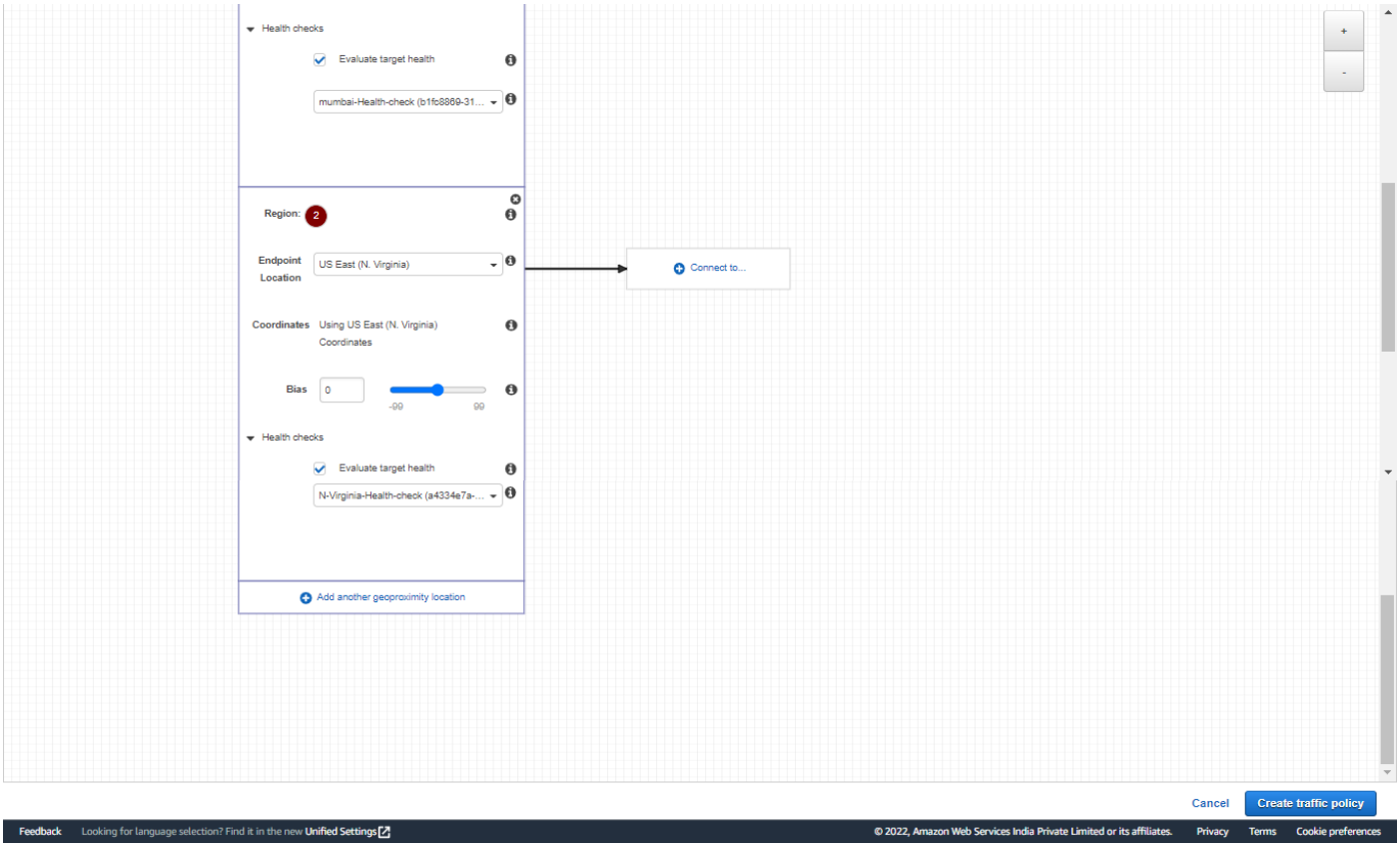




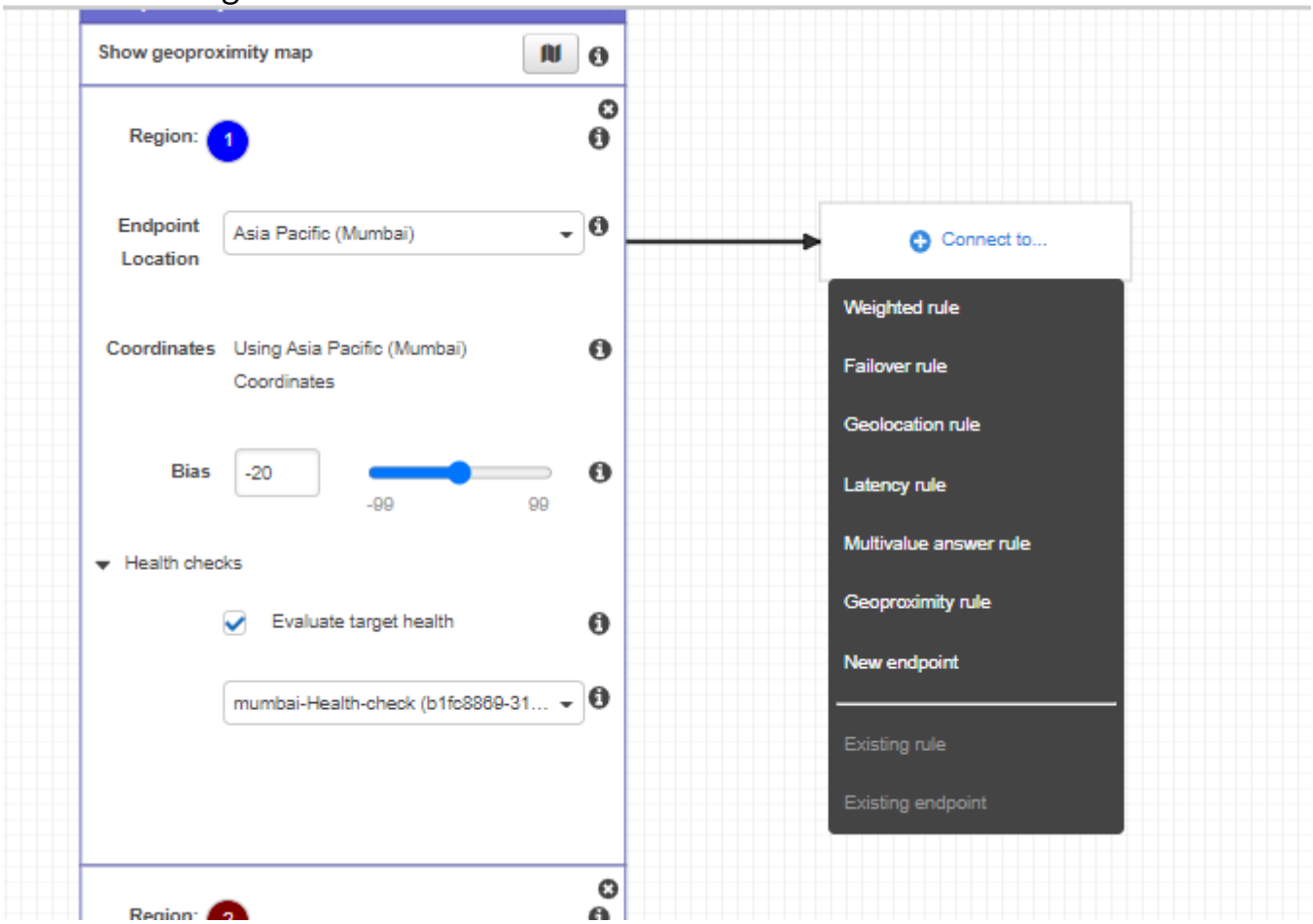
Step 5: Now select endpoint. just click on the end point drop down menu and select your endpoint , here I am selecting Mumbai region. Heath check and put Bias, do same for Virginia.

For reference you can see following images.



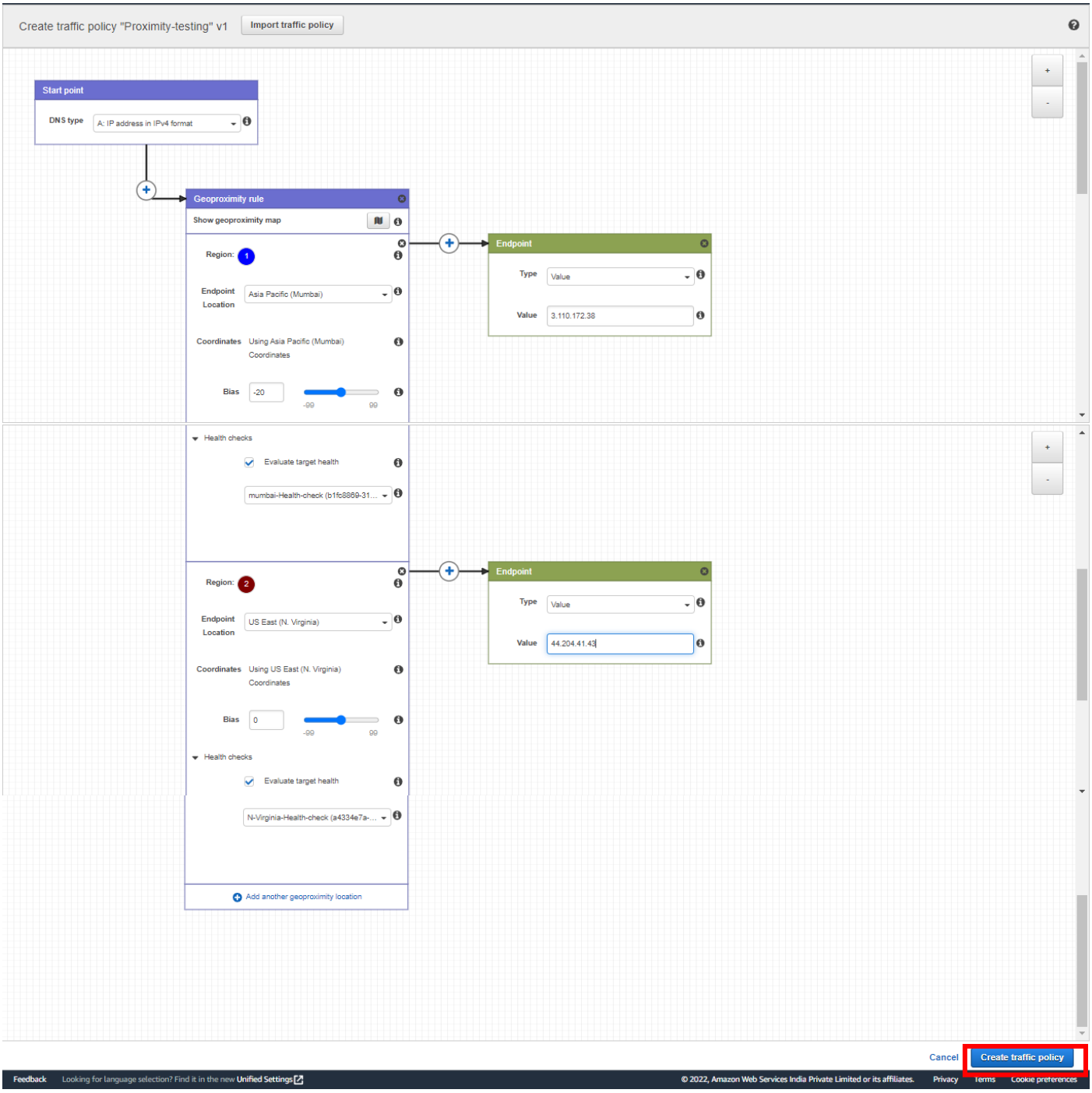


Step 6: Now Click on the **"Connect to"** and select new endpoint, for reference you can see below images.



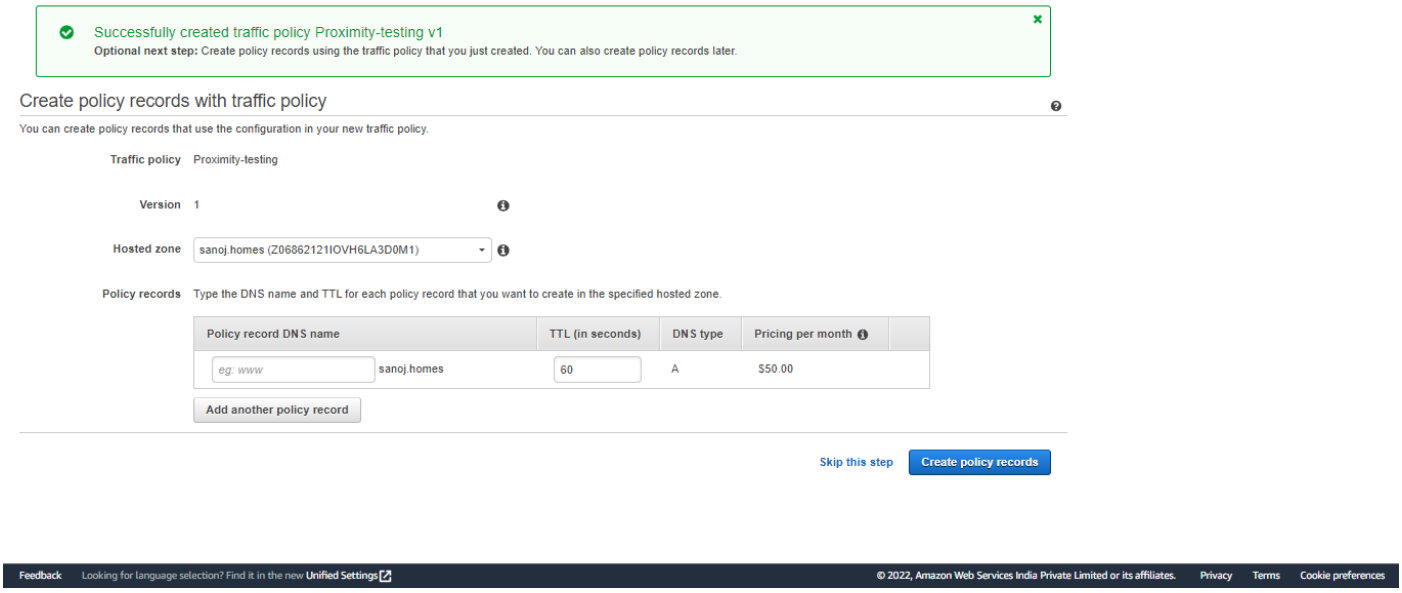
Now Select **"New endpoint"**

After selecting new endpoint, you will see following interface just put the IP address of “Particular region in which you want to target your traffic. for reference you can see following screenshot images.



Step 8: Now Click on **“Create Traffic policy”**

After clicking on **“Create Traffic Policy”** you will see following interface.



Now as a final step you have to just Click on “Create policy records” and you are done.

Note: if you create Geoproximity you will incur charged $50.00 so do according.

If you want to see the Graph as per your Biased just click on the graph icon.