Route-53

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

You can use Route 53 to perform three main functions in any combination: domain registration, DNS routing, and health checking.

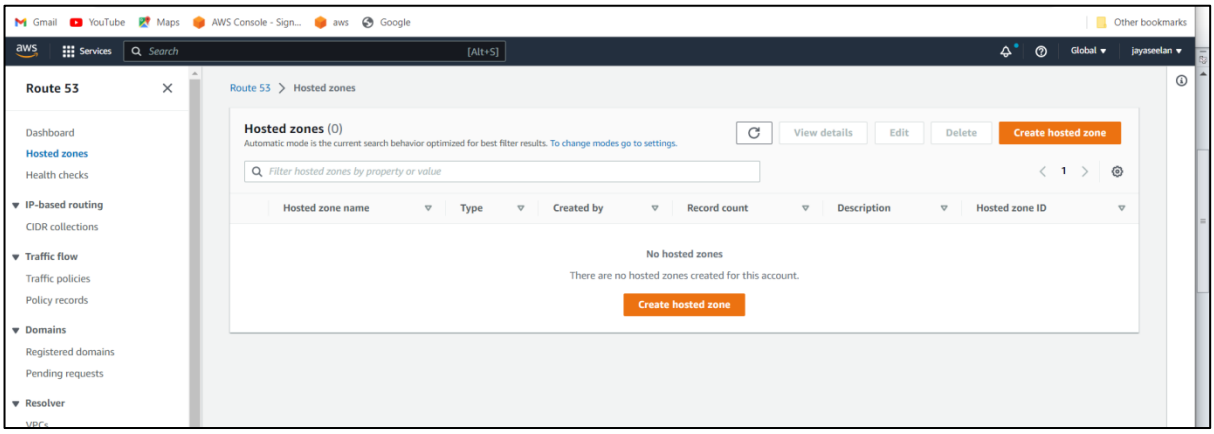
Types of route-53

- Simple routing.
- Failover routing.
- Geolocation routing.
- Geoproximity routing (traffic flow only)
- Latency-based routing.
- IP-based routing.
- Multivalue answer routing.
- Weighted routing.

Steps to create route-53:

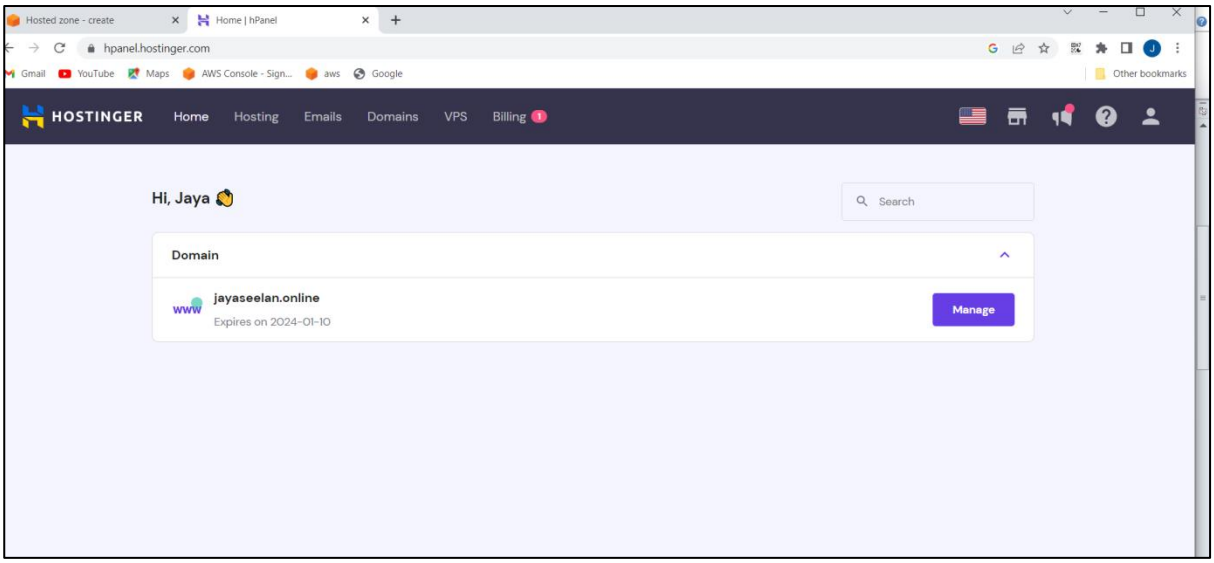
Step1:create hosted zone

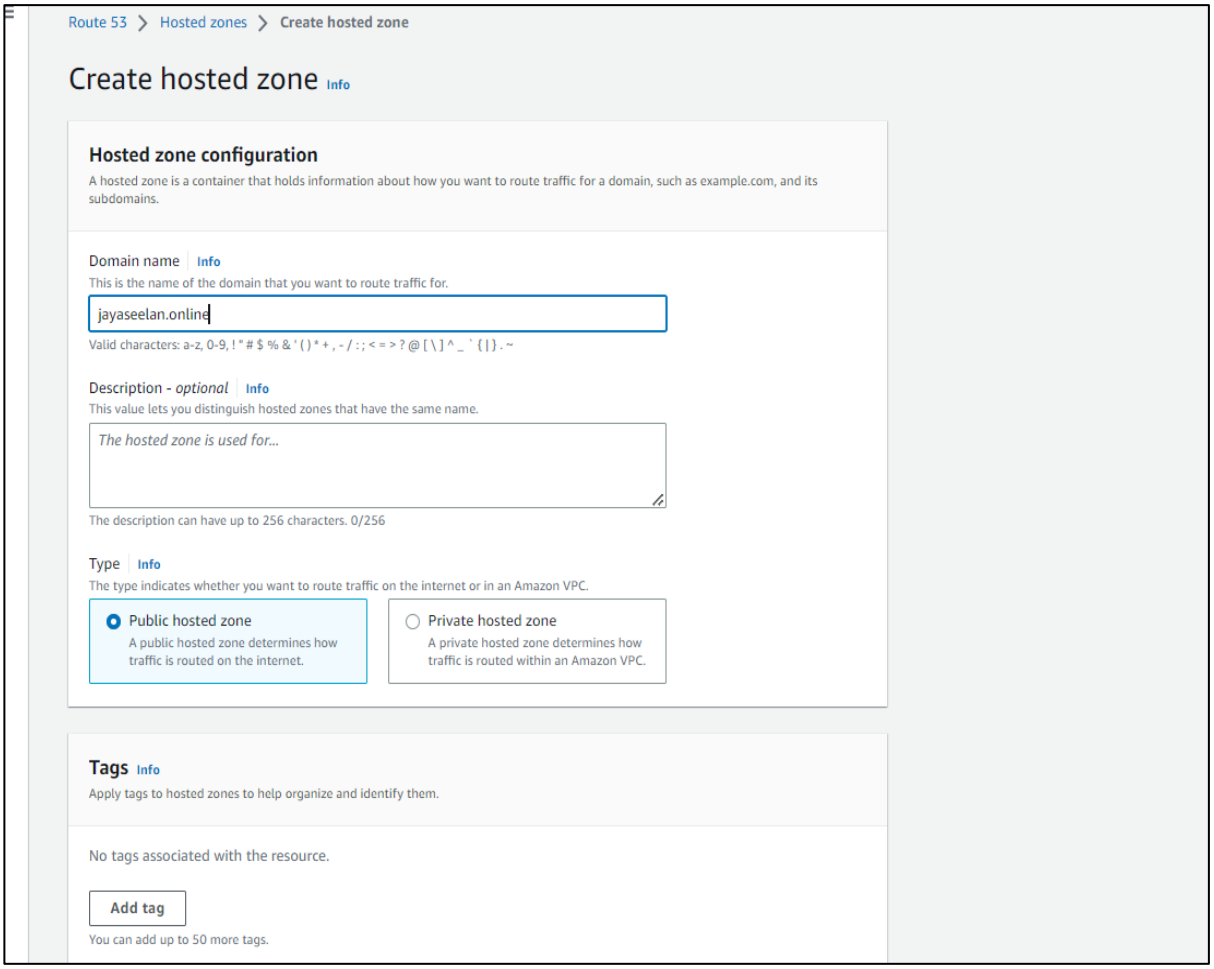
Rout53--->hosted zone---->create hosted zone



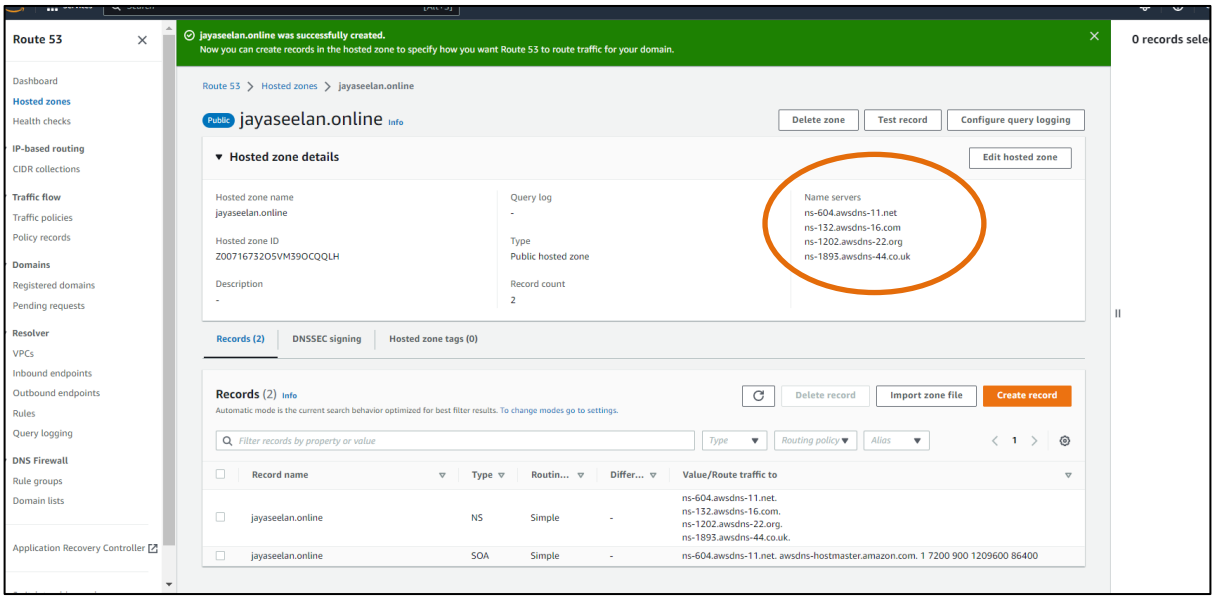
Domain name(jayaseelan.online)---->type(public hosted zone)---->create

Hostinger--->login mail--->show my domain



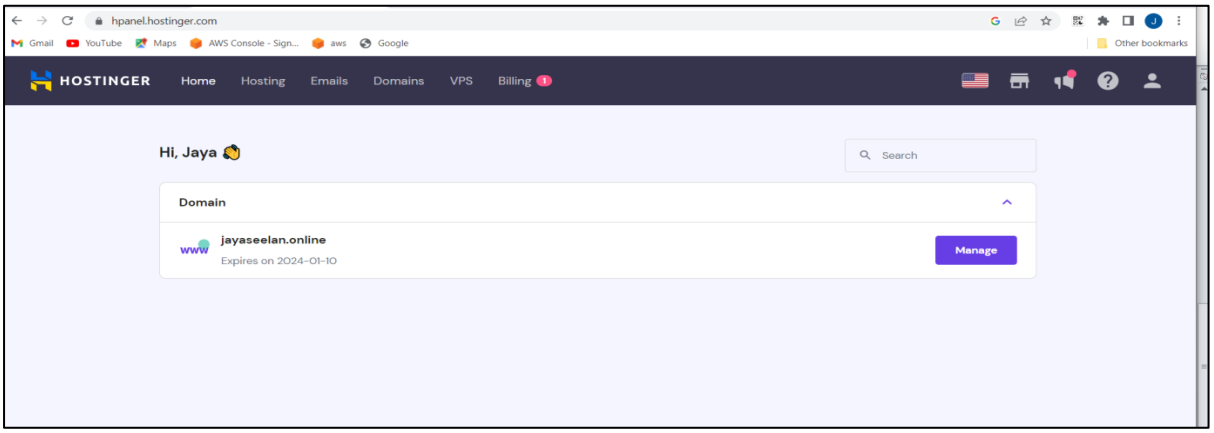


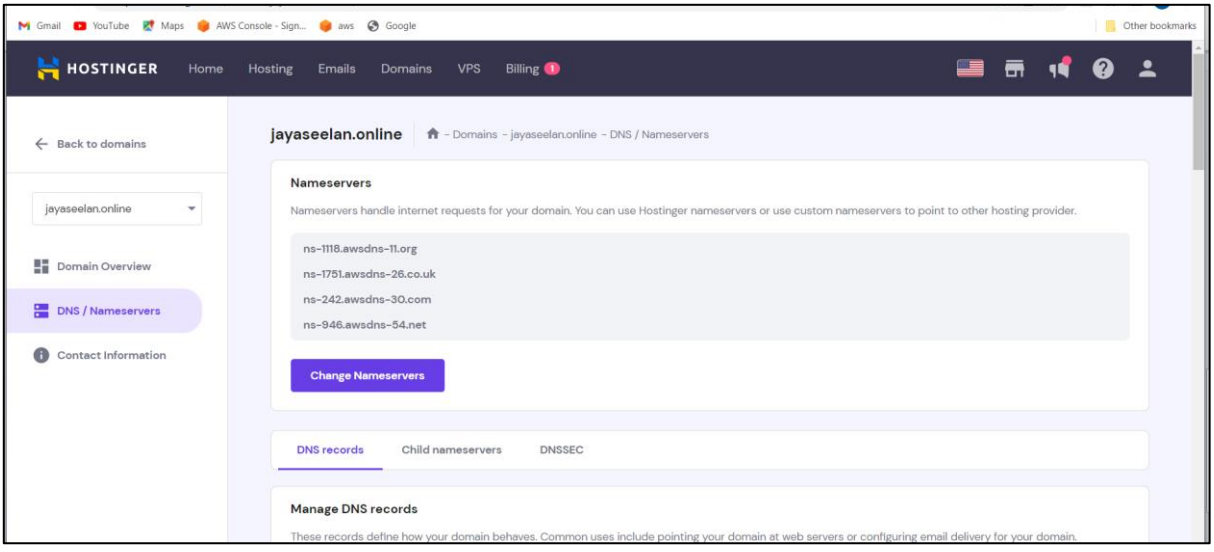
Hosted zone created.



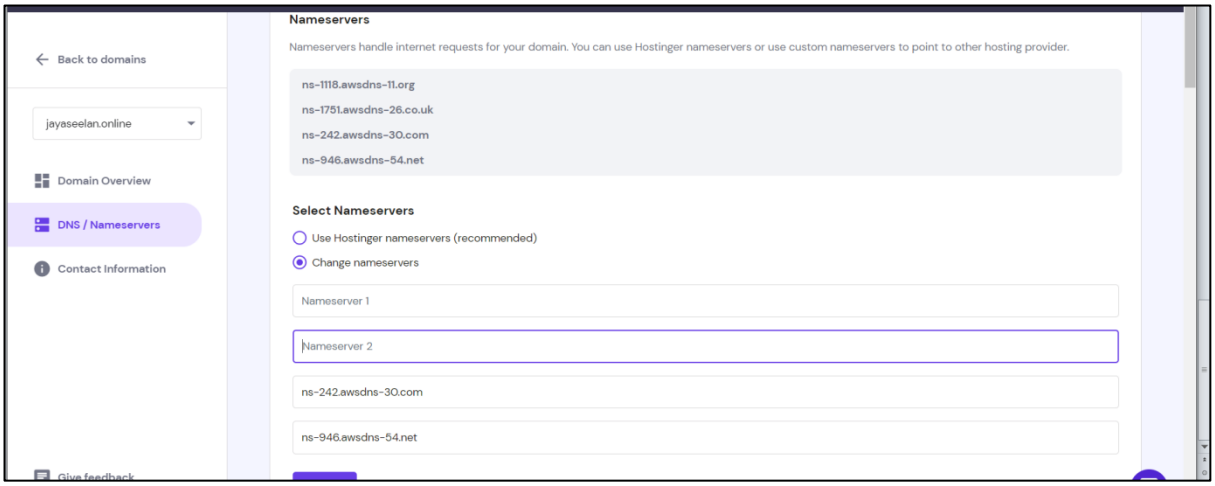
Step2:change name server

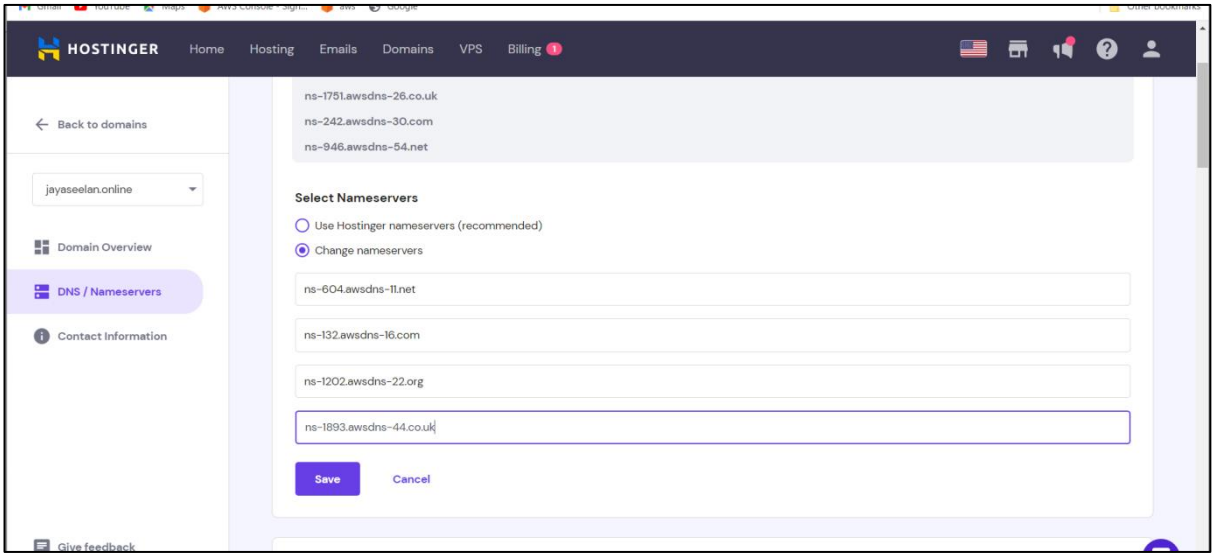
Hostinger---->manage--->dns/name server---->click change name server



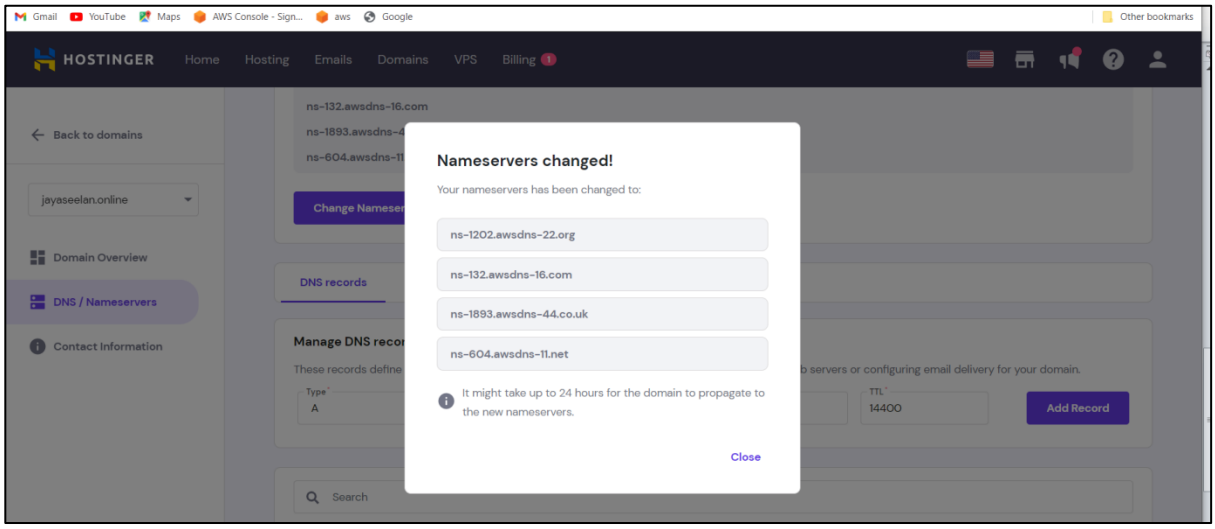


Already 4 name server show that delete and put r-53 hosted zone name server---->save

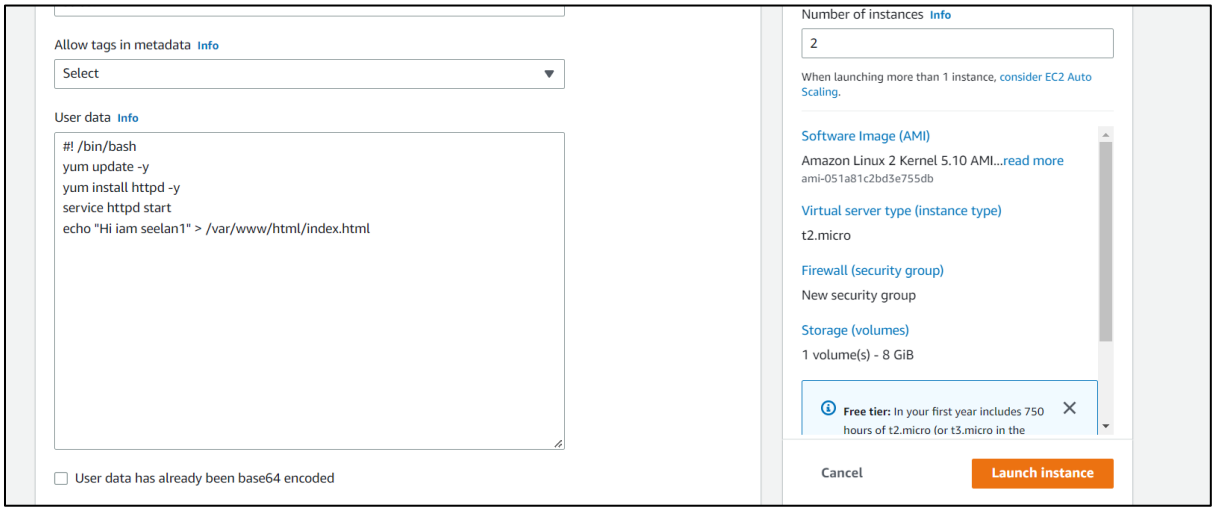




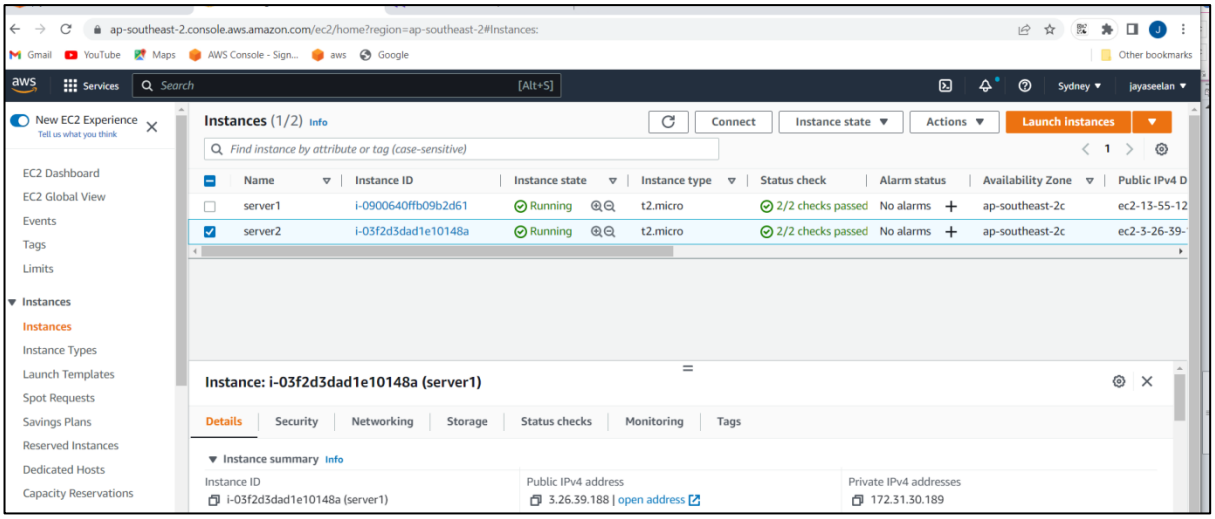
Name server change success.



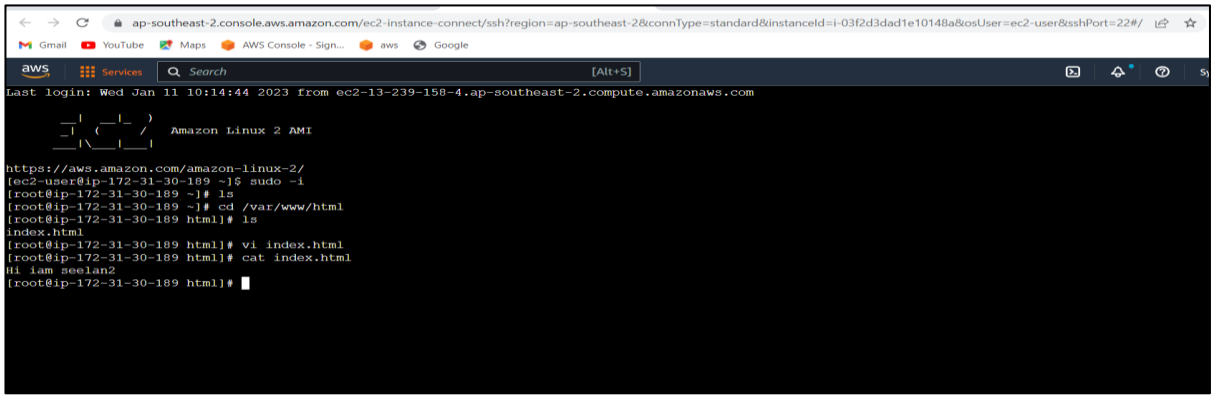
Step3:launch 2 ec2 linux instance --->security group (ssh and http)---->advanced deatails(any bash script).



Two server created..



Server2 login--->content change --->#sudo –i--->#cd /var/www/html----->vi index.html --->change content(hi iam seelan2)

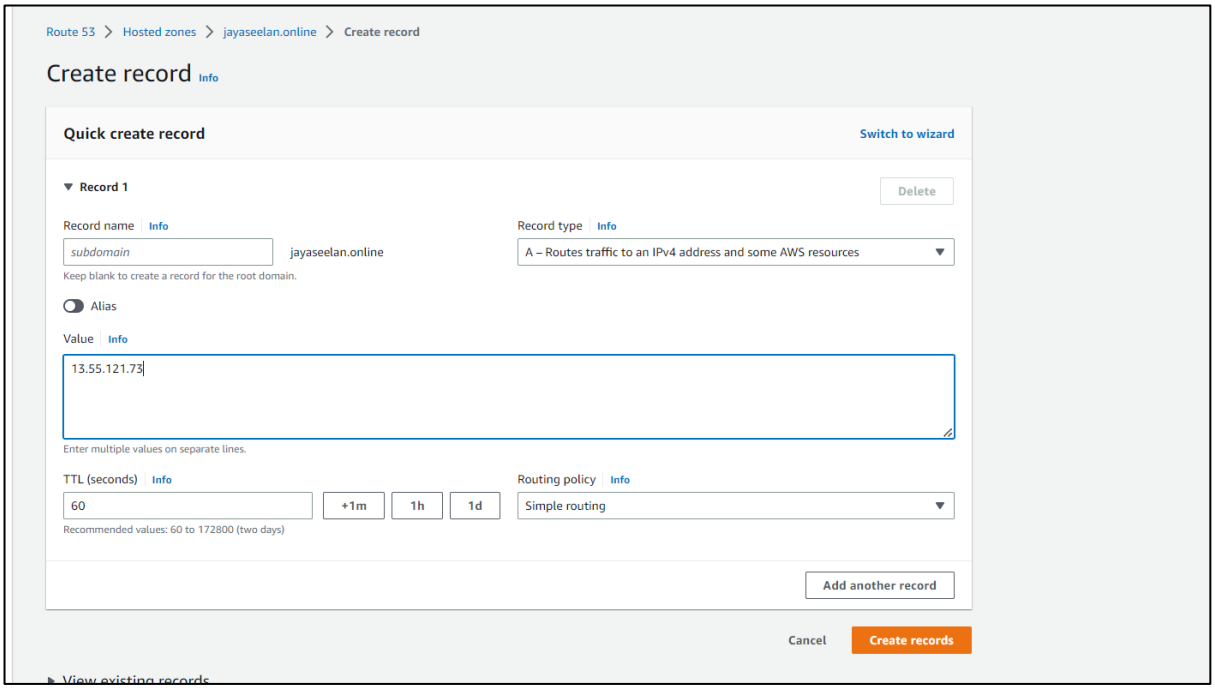


Simple routing

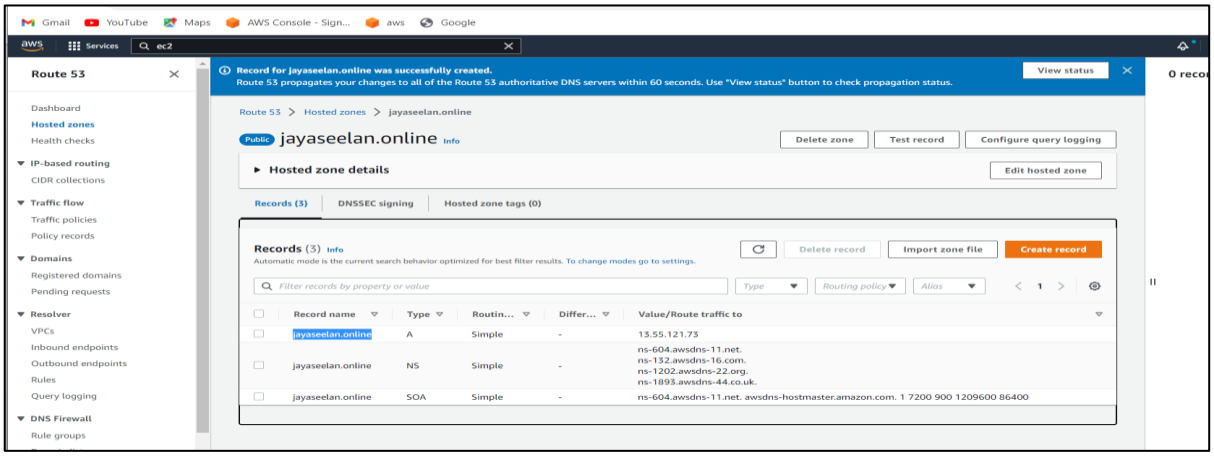
Simple routing policy Use for a single resource that performs a given function for your domain, for example, a web server that serves content for the example.com website. You can use simple routing to create records in a private hosted zone.

Steps to create simple routing record

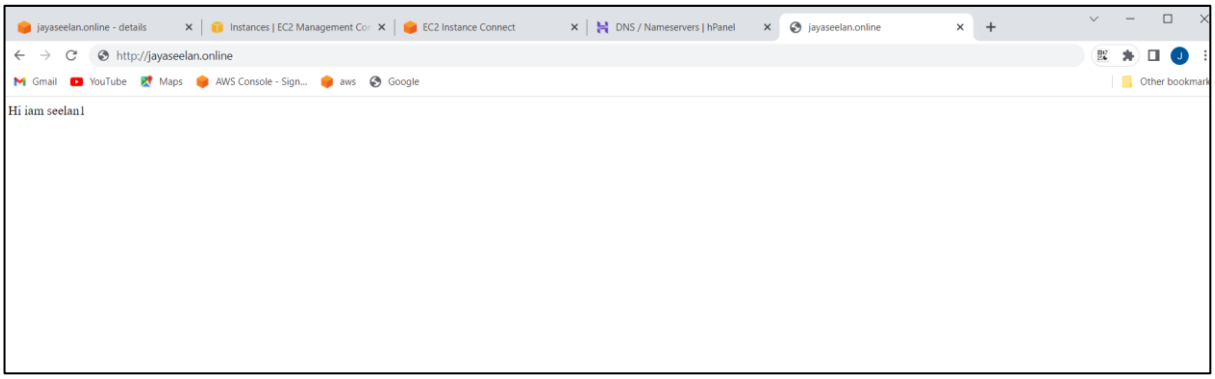
Step1:hosted zone(jayaseelan.online)---->create record--->value(any one server ip copy and paste)----->TTL(time to leave)--->60 sec----->record type(A-routes traffic to an ipv4 addres and some aws resourse)---->routing policy(simple routing policy)----->create records



Simple routing record created..



Now put chrome text page **http://jayaseelan.online**----> it show server content



This is simple routing..

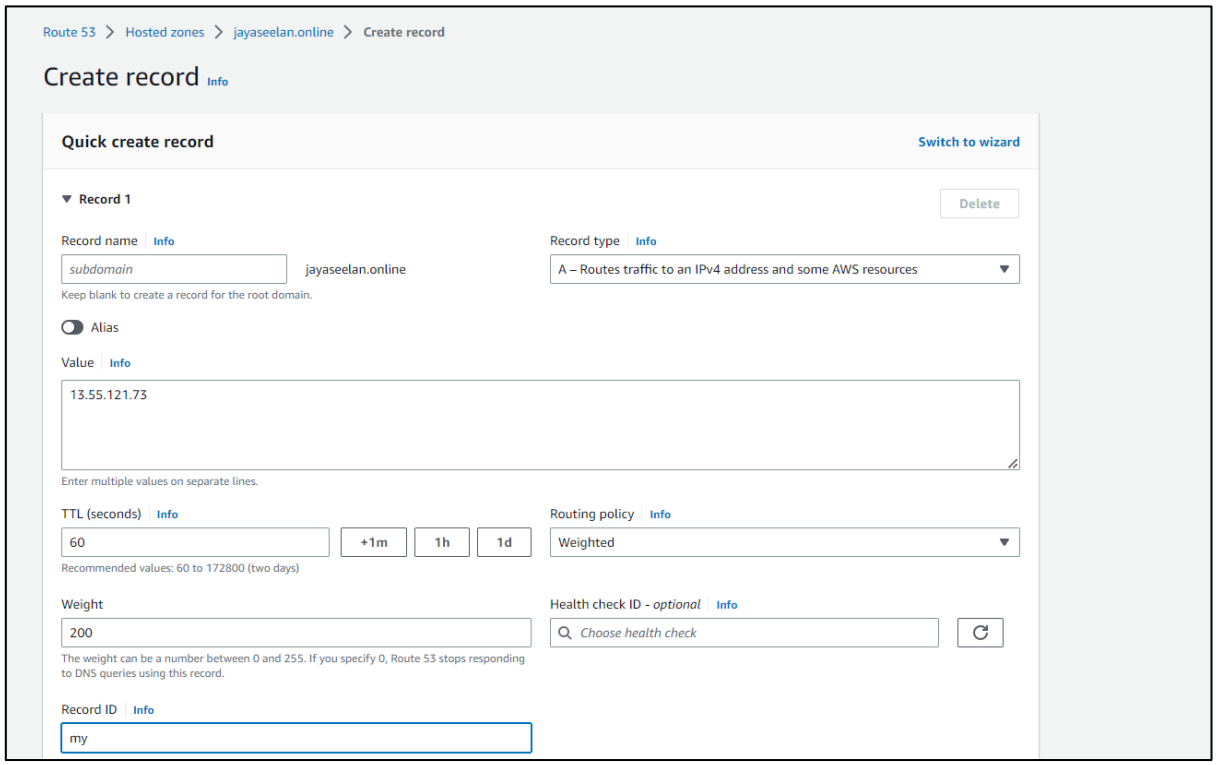
Weighted routing

Weighted routing lets you associate multiple resources with a single domain name (example.com) or subdomain name (acme.example.com) and choose how much traffic is routed to each resource.

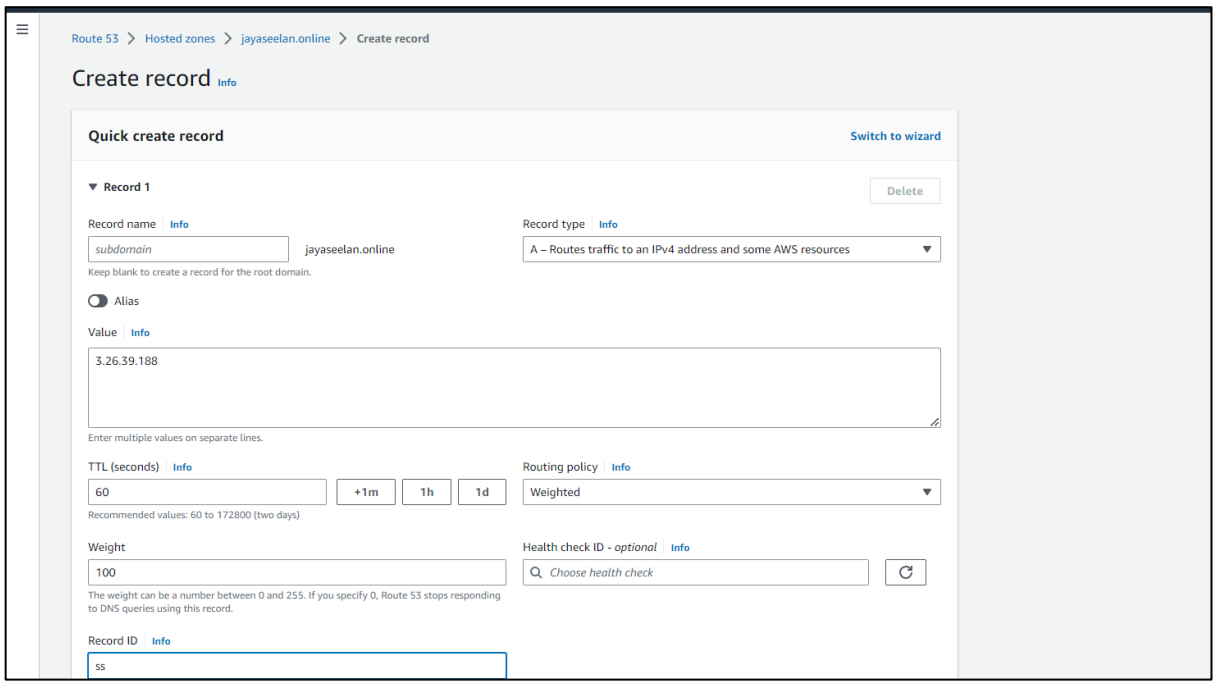
This can be useful for a variety of purposes, including load balancing and testing new versions of software.

Steps to create weighted record

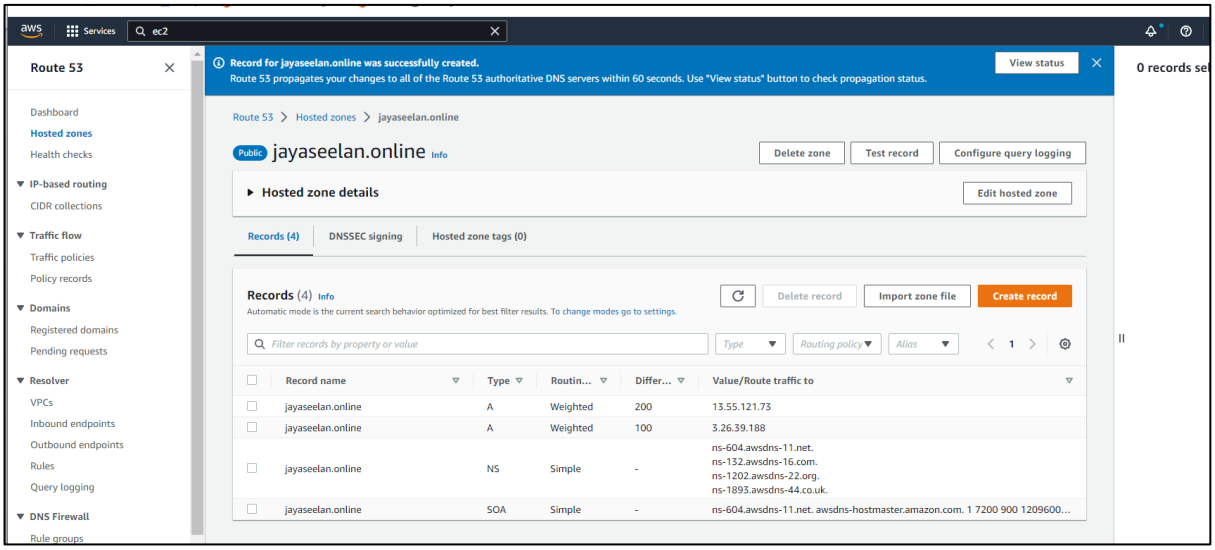
Step1: hosted zone(jayaseelan.online)---->create record--->value (**server1 ip** copy and paste)----->TTL(time to leave)---->60 sec----->record type(A-routes traffic to an ipv4 addres and some aws resourse)---->routing policy(weighted routing)----->weight(**200**)------>create records



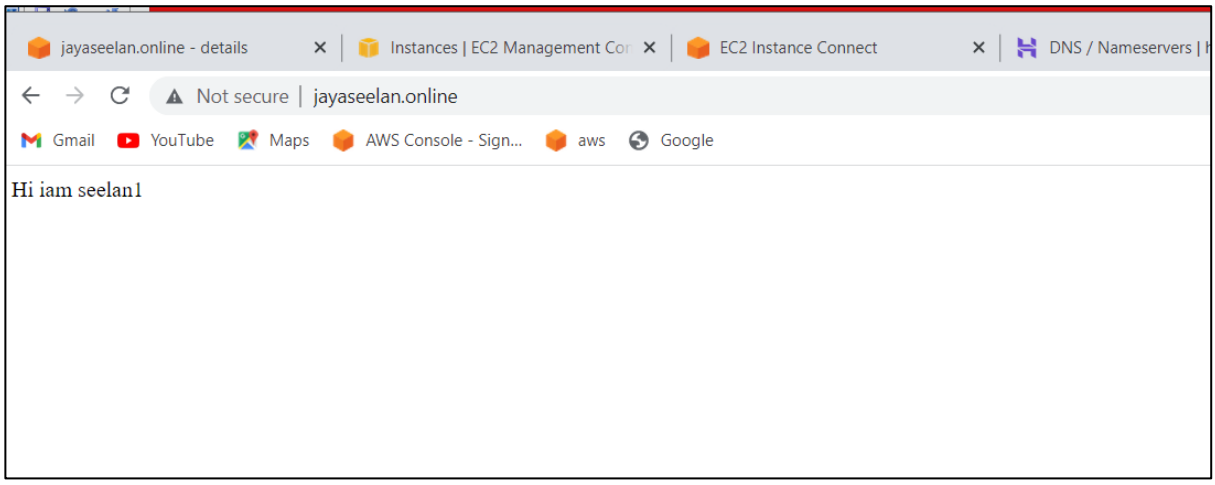
Step2: hosted zone(jayaseelan.online)---->create record--->value (**server2 ip** copy and paste)----->TTL(time to leave)---->60 sec----->record type(A-routes traffic to an ipv4 addres and some aws resourse)---->routing policy(weighted routing)----->weight(**100**)------>create records



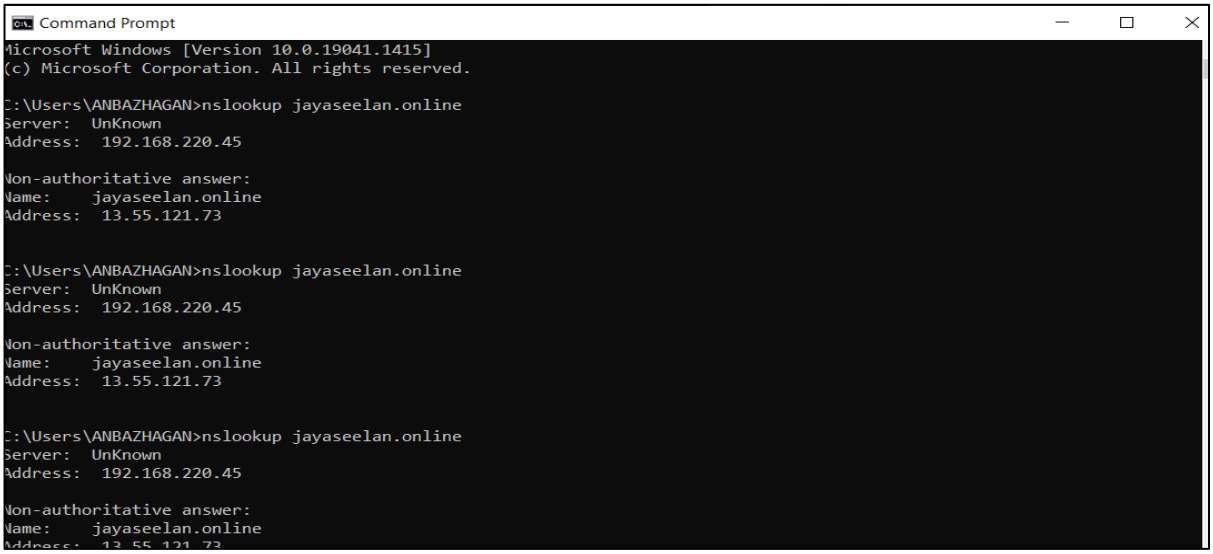
Created two record for server1 &server2



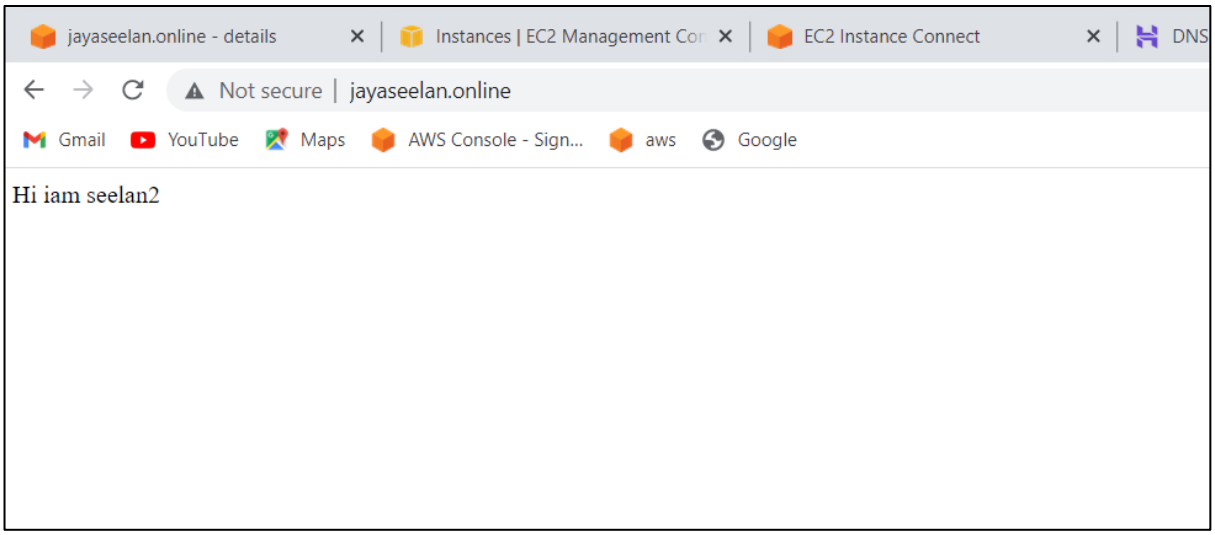
Now put chrome text page **http://jayaseelan.online**----> it will which server is high weight thar server show content show

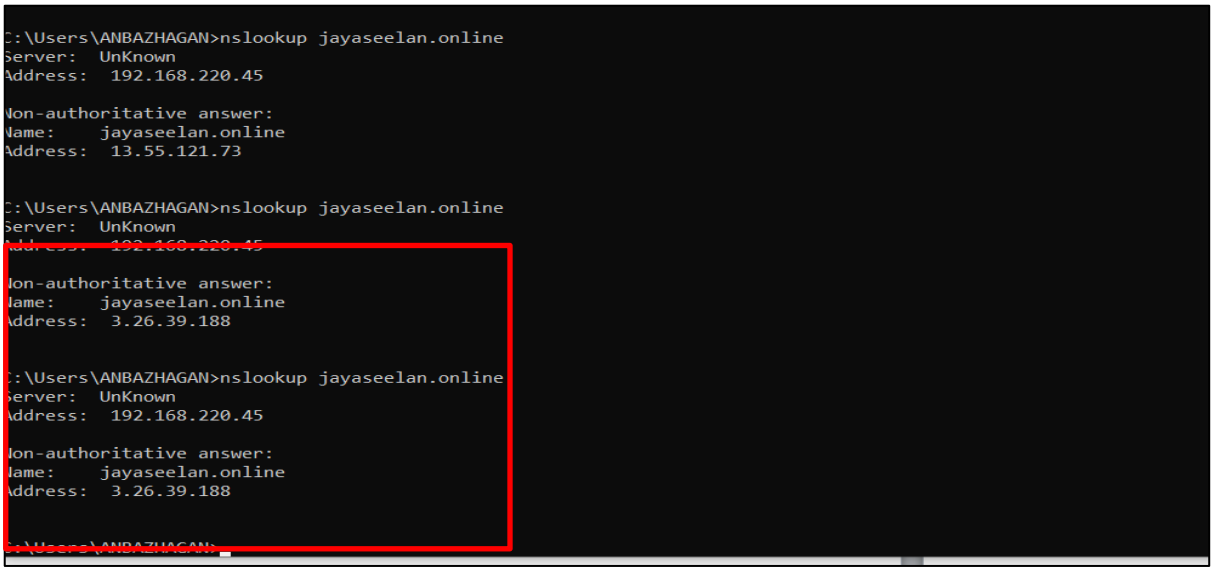


Now check using **cmd** prompt ---->nslookup jayaseelan.online--->it will show which server is high wait that ip shown



Now change server1 wight(100)---->server2 weight(200)---->it will show server2 content(that means server 2 is high weight)





This is weighted routing...

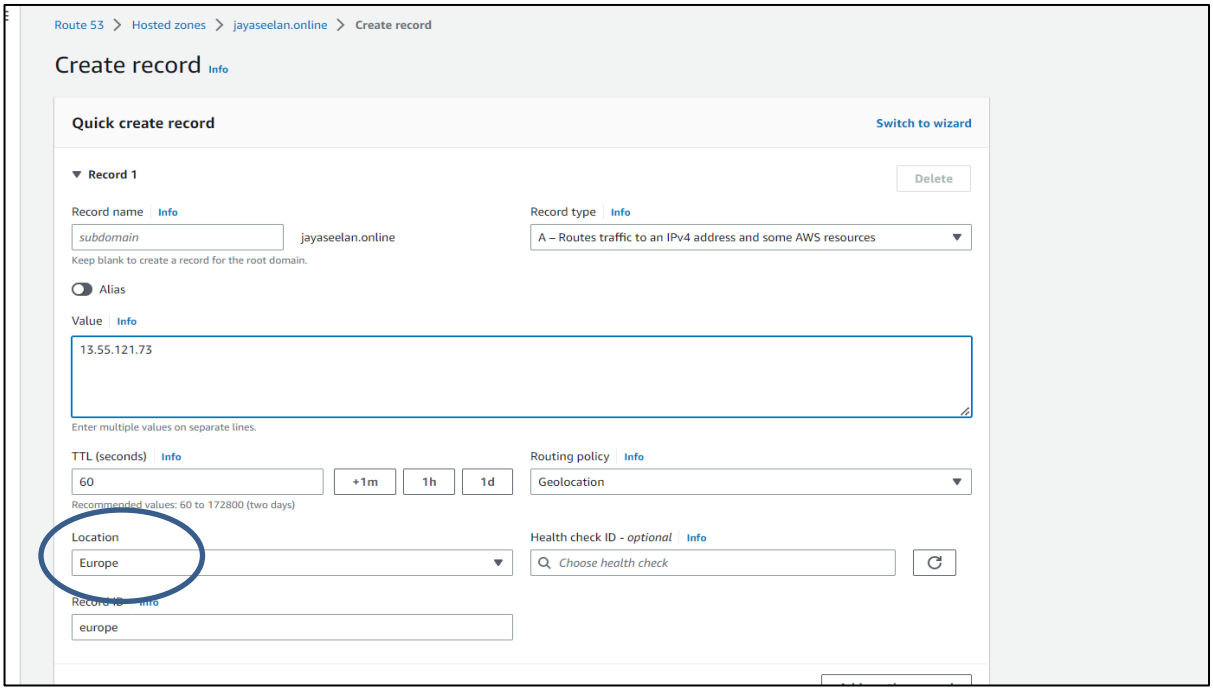
Geolocation routing:

Geolocation routing lets you choose the resources that serve your traffic based on the geographic location of your users, meaning the location that DNS queries originate from.

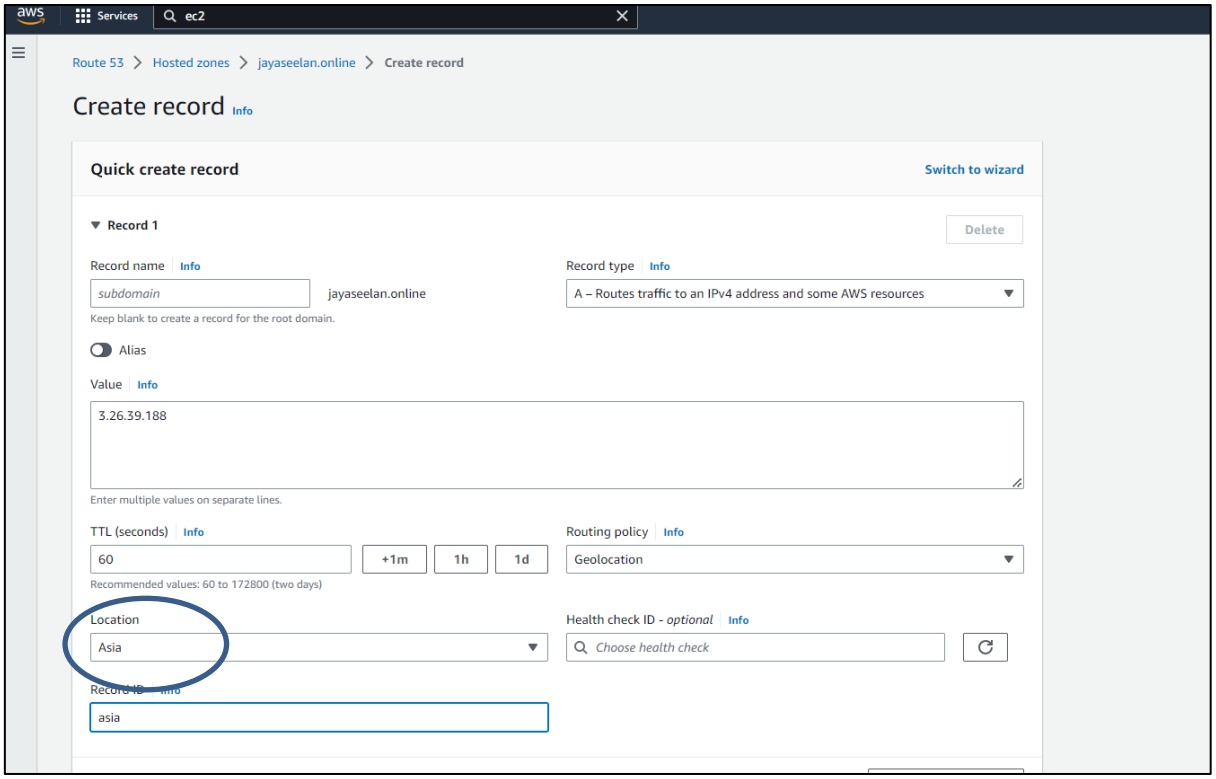
For example, you might want all queries from Europe to be routed to an ELB load balancer in the Frankfurt region.

Steps to create geolocation record:

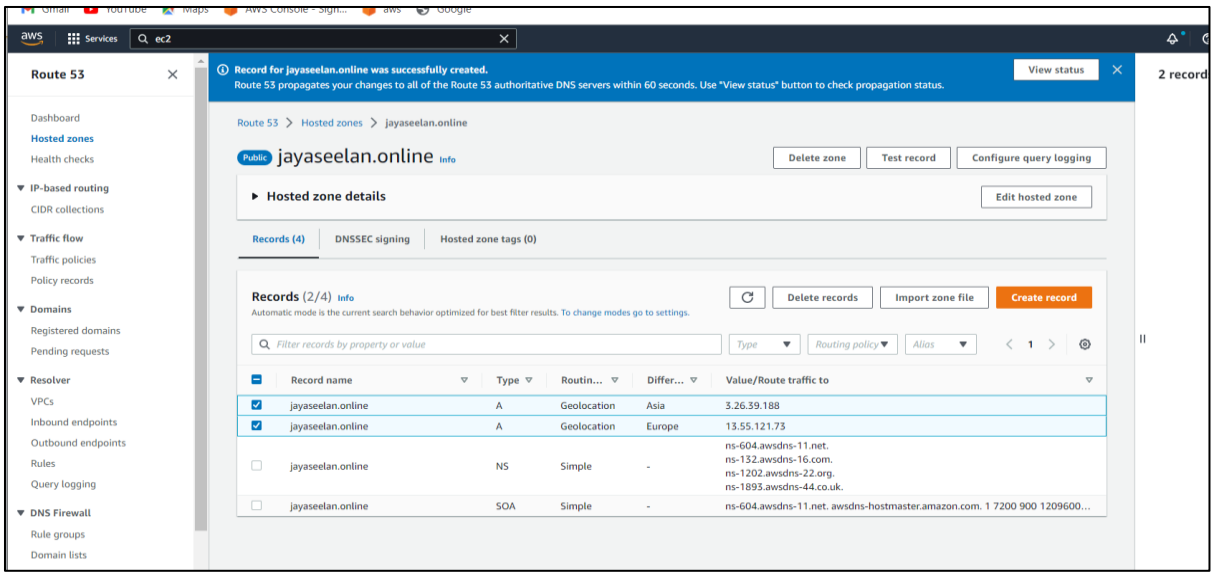
Step1: hosted zone(jayaseelan.online)---->create record--->value (**server1 ip** copy and paste)----->TTL(time to leave)---->60 sec----->record type(A-routes traffic to an ipv4 addres and some aws resourse)---->routing policy(geolocation routing)---->location(**Europe)**------>create records



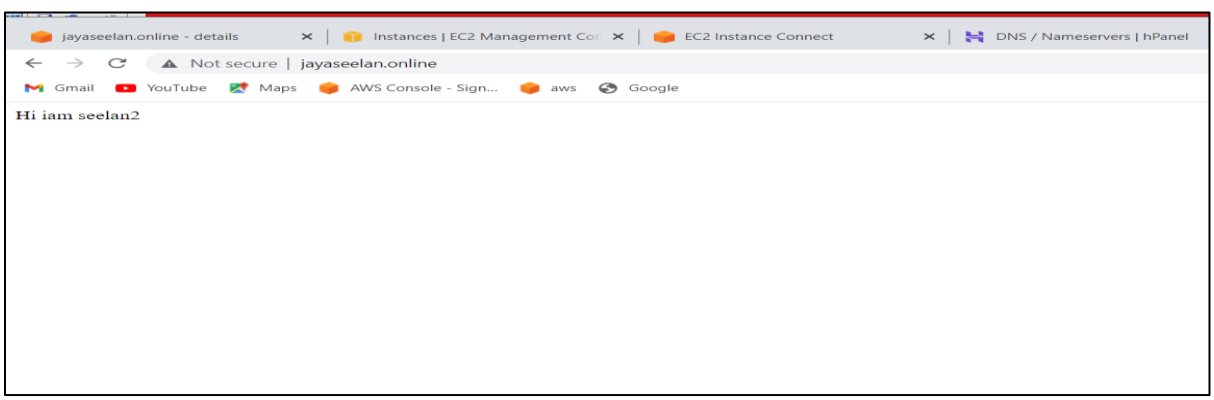
Step2: hosted zone(jayaseelan.online)---->create record--->value (**server2 ip** copy and paste)----->TTL(time to leave)---->60 sec----->record type(A-routes traffic to an ipv4 addres and some aws resourse)---->routing policy(geolocation routing)---->location(**asia)**------>create records



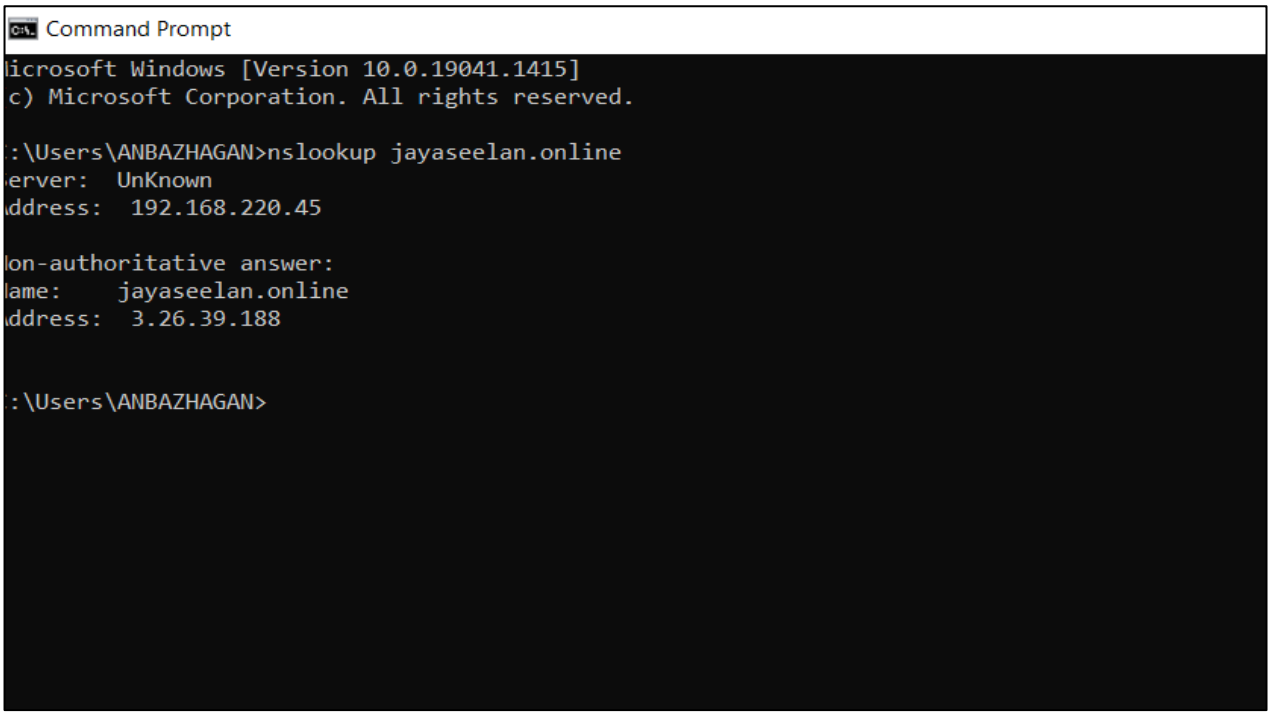
created two geolocation record



Now put chrome text page **http://jayaseelan.online**----> it will show asia location server ony (hi iam seelan2---->this is 2nd server content)



Now check using **cmd** prompt ---->nslookup jayaseelan.online---->it will show asia location server ip(2nd server ip shown)



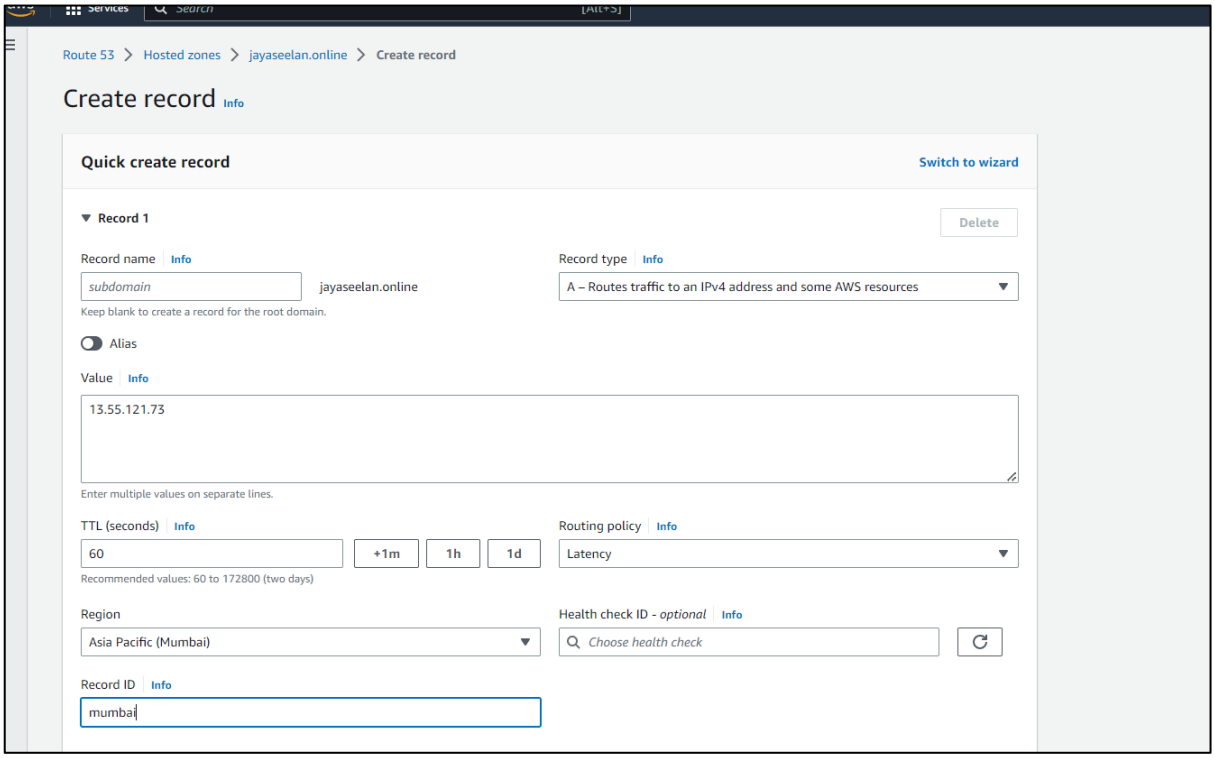
It will success..

Latency routing

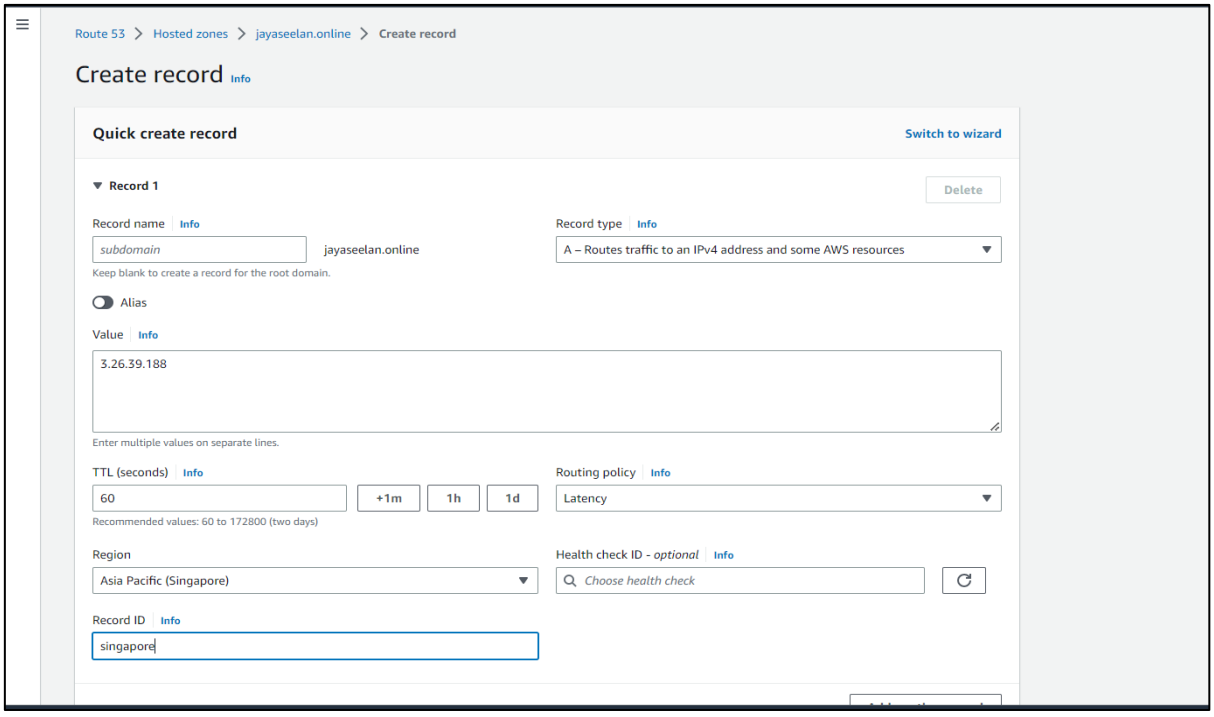
If your application is hosted in multiple AWS Regions, you can improve performance for your users by serving their requests from the AWS Region that provides the lowest latency.

Steps to create latenct record

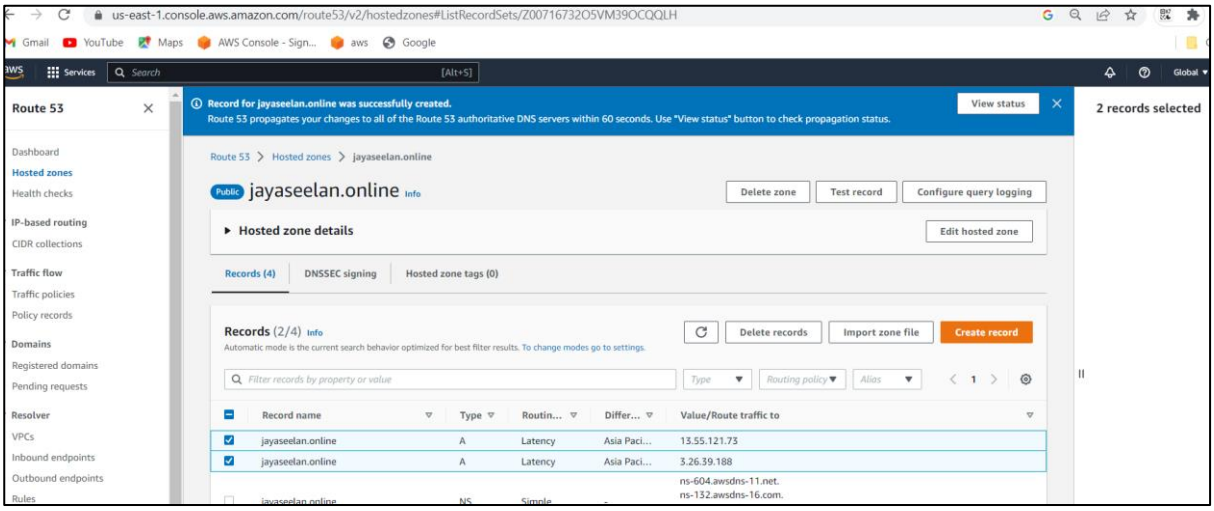
Step1: hosted zone(jayaseelan.online)---->create record--->value (**server1 ip** copy and paste)----->TTL(time to leave)---->60 sec---->record type(A-routes traffic to an ipv4 addres and some aws resourse)---->routing policy(latency routing)---->**region(asia pacific Mumbai)**--------->create records



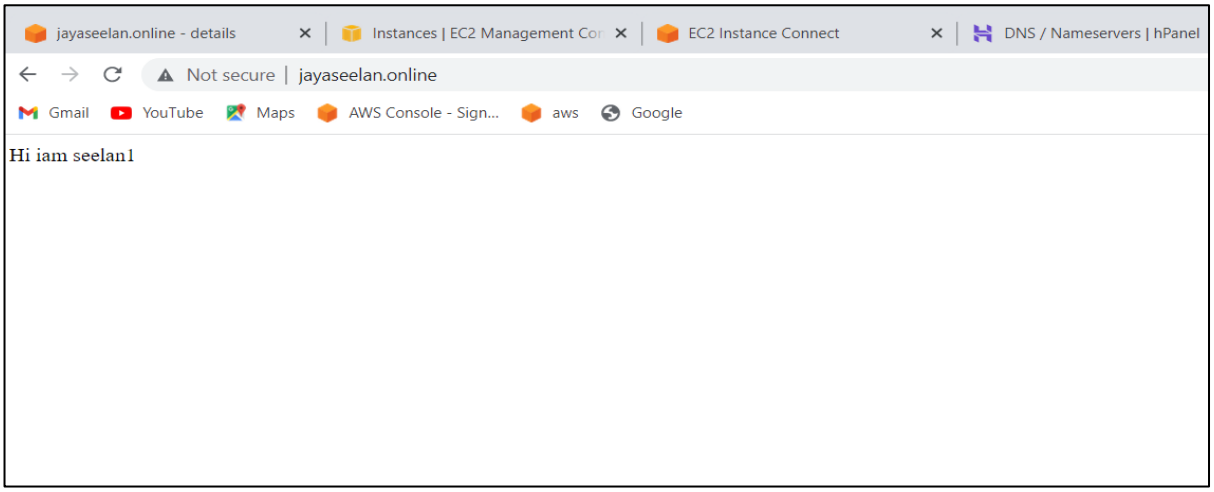
Step1: hosted zone(jayaseelan.online)---->create record--->value (**server2 ip** copy and paste)----->TTL(time to leave)---->60 sec---->record type(A-routes traffic to an ipv4 addres and some aws resourse)---->routing policy(latency routing)---->**region(asia pacific singapore)**--------->create records



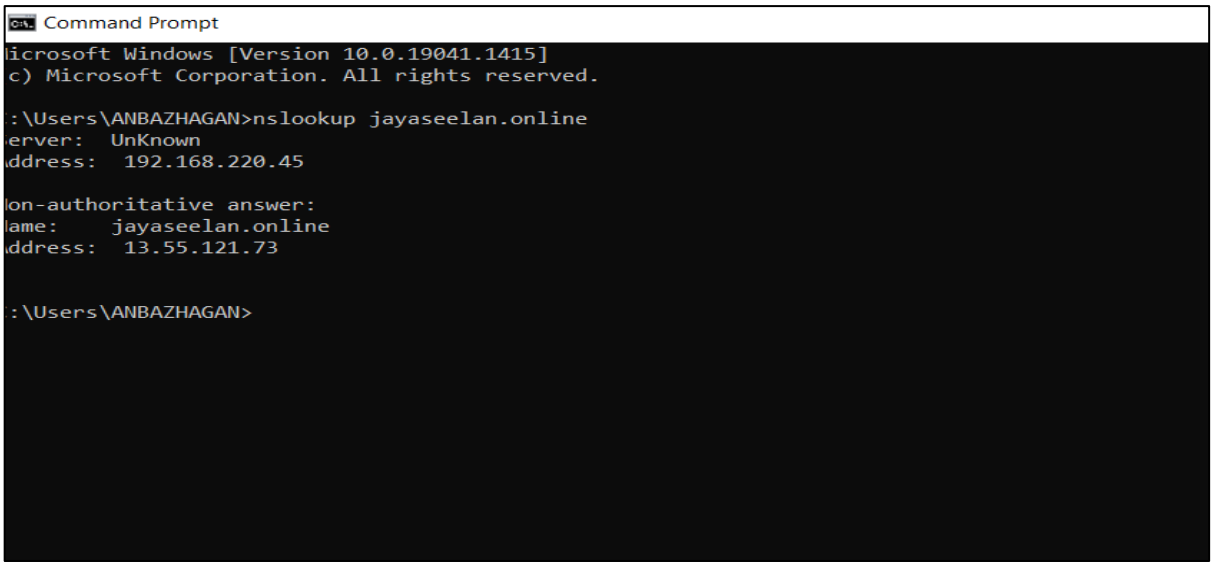
Created latency server1 and server2 record



Now put chrome text page **http://jayaseelan.online**----> it will show Mumbai region server



Now check using **cmd** prompt ---->nslookup jayaseelan.online---->it will show Mumbai region server ip(1st server ip shown)



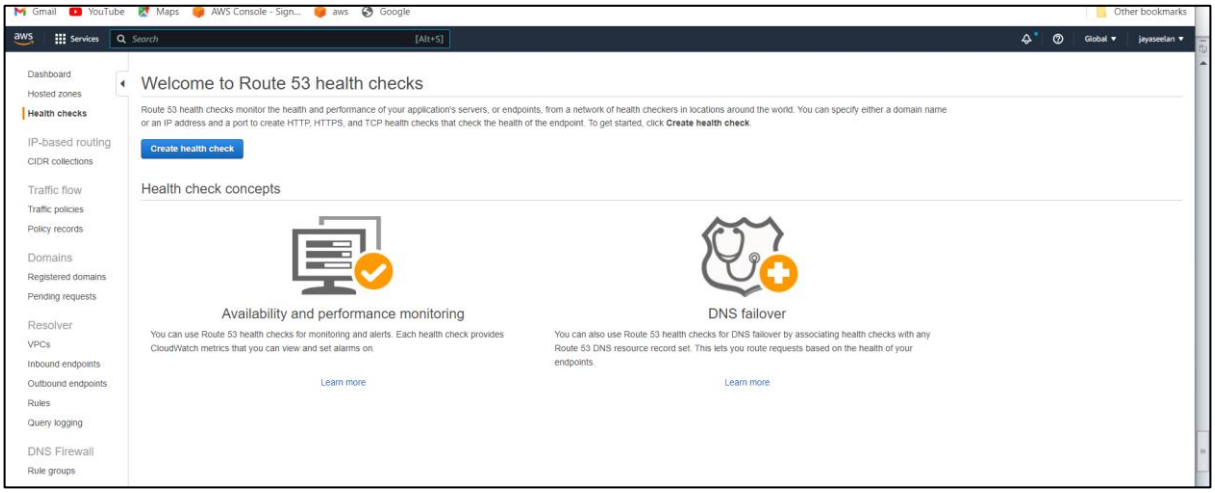
Failover routing:

Failover routing lets you route traffic to a resource when the resource is healthy or to a different resource when the first resource is unhealthy.

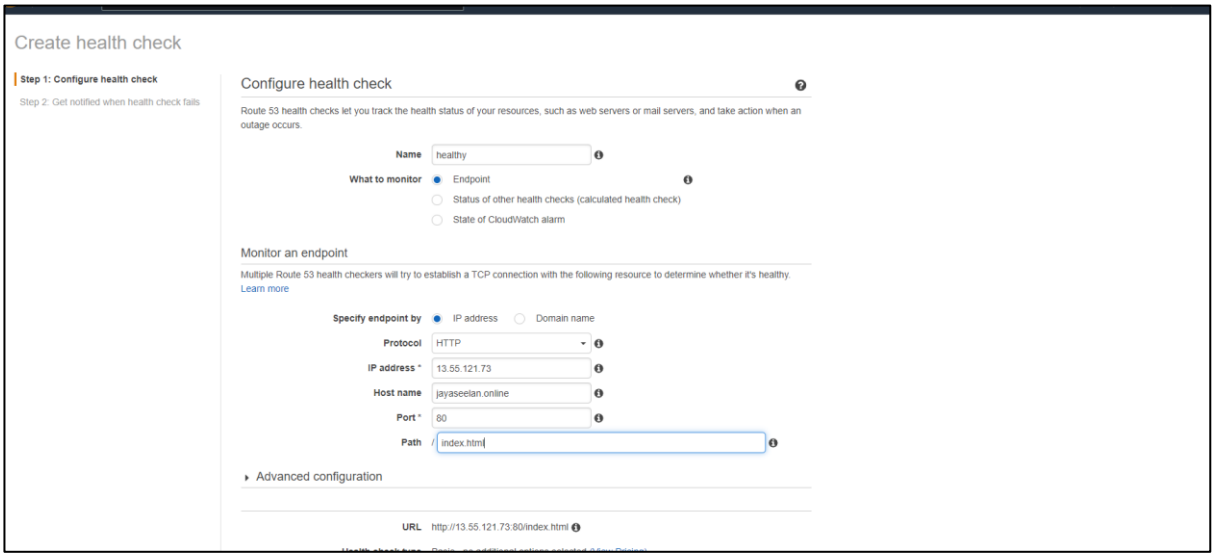
The primary and secondary records can route traffic to anything from an Amazon S3 bucket that is configured as a website to a complex tree of records.

Steps to create failover record

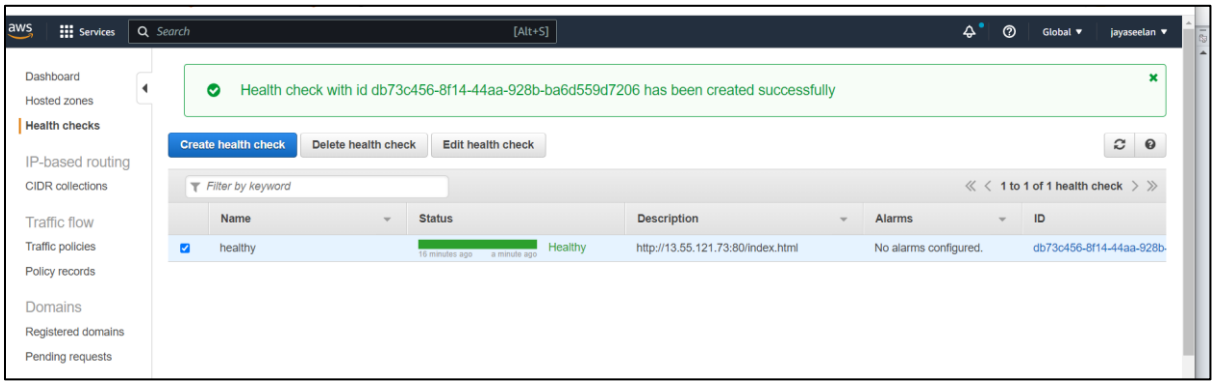
Step1:route-53----->health check---->create health check



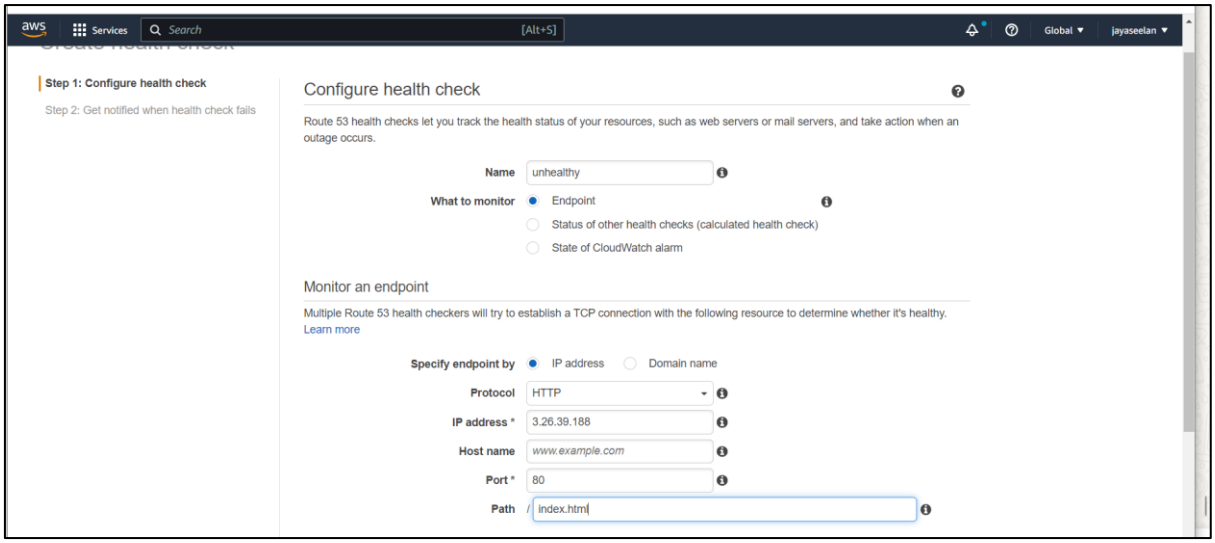
Step1.1:name(healthy)--->ip address(server1 ip)---->host name(jayaseelan.online)---->path/(index.html)--->next--->create alaram(no)--->create health check.



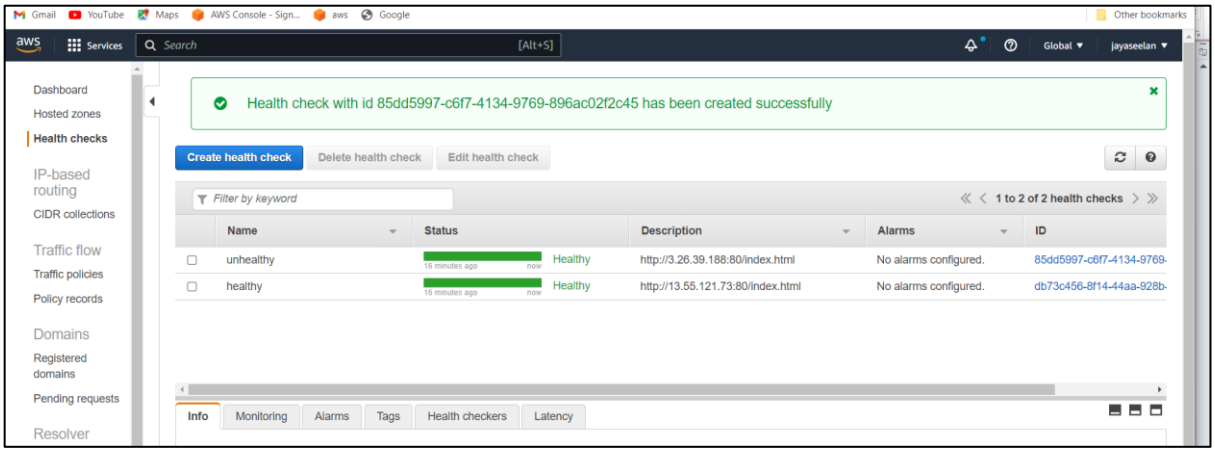
created healthy



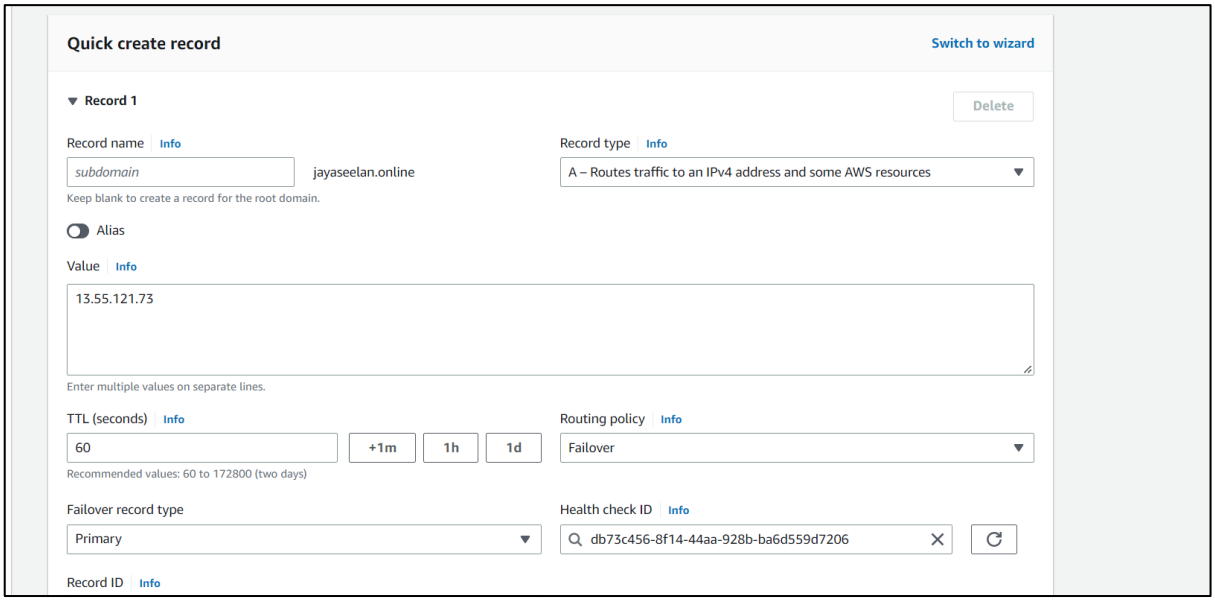
Step1.2:name(un healthy)--->ip address(server2 ip)---->host name(jayaseelan.online)---->path/(index.html)--->next--->create alaram(no)--->create health check



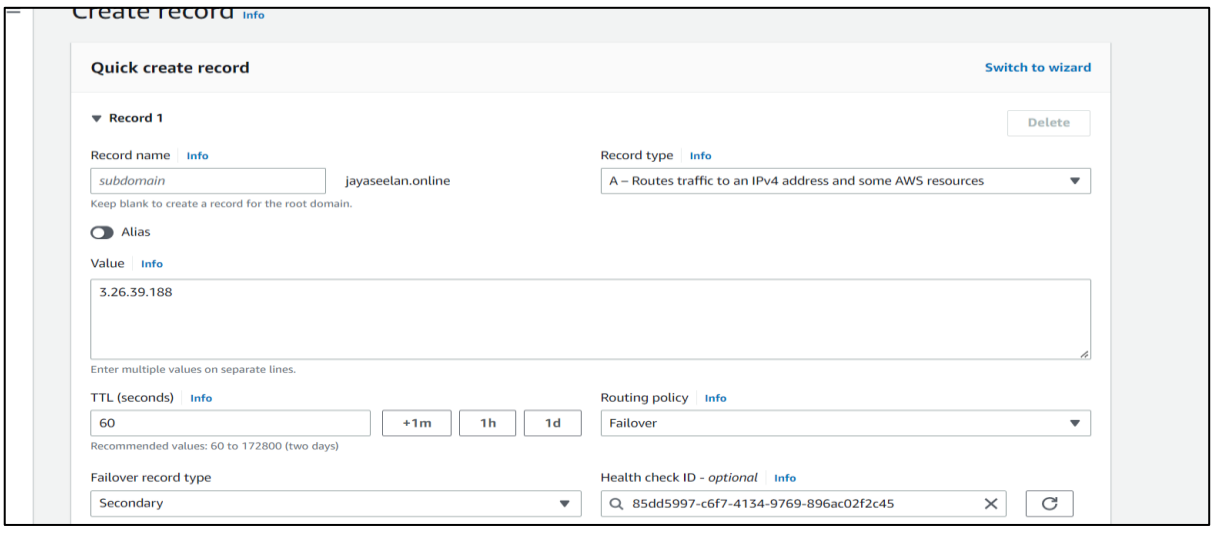
Created unhealthy



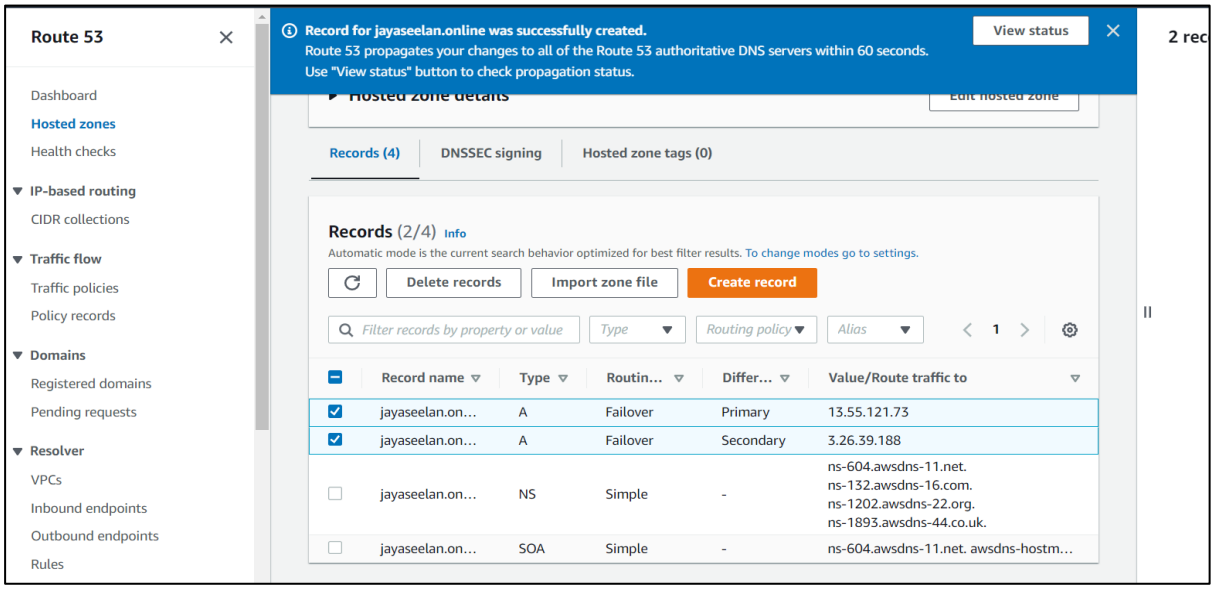
Step2: hosted zone(jayaseelan.online)---->create record--->value (**server1 ip** copy and paste)----->TTL(time to leave)---->60 sec---->record type(A-routes traffic to an ipv4 addres and some aws resourse)---->routing policy(failover routing)---->failover record type(primary)---->health check id(select healthy)--------->create records



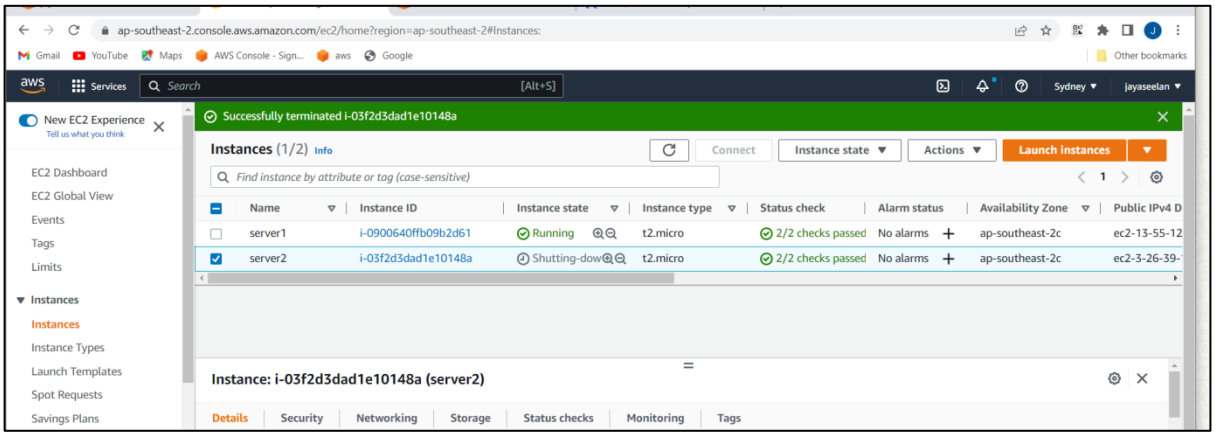
Step2.1: hosted zone(jayaseelan.online)---->create record--->value (**server2 ip** copy and paste)----->TTL(time to leave)---->60 sec---->record type(A-routes traffic to an ipv4 addres and some aws resourse)---->routing policy(failover routing)---->failover record type(secondary)---->health check id(select unhealthy)--------->create records



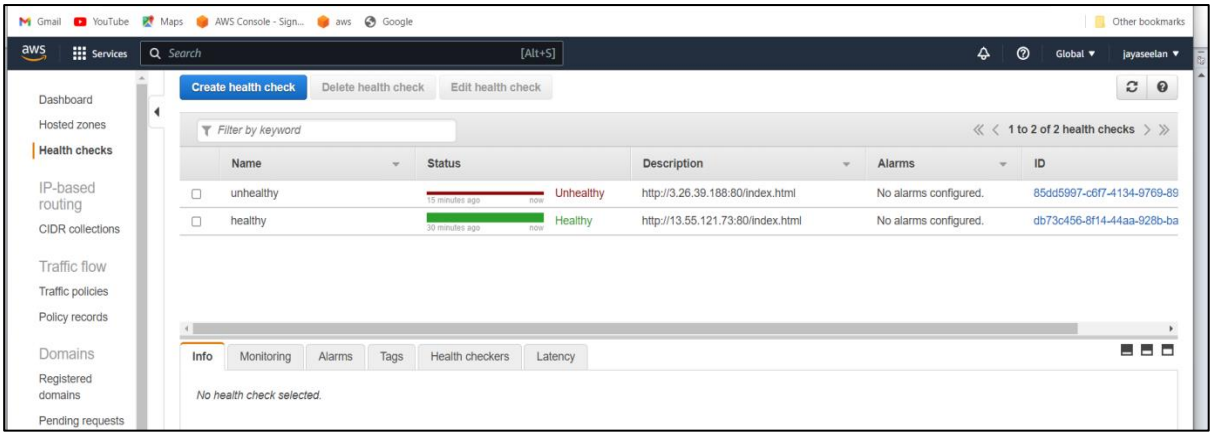
created record for healthy server1 and unhealthy server2



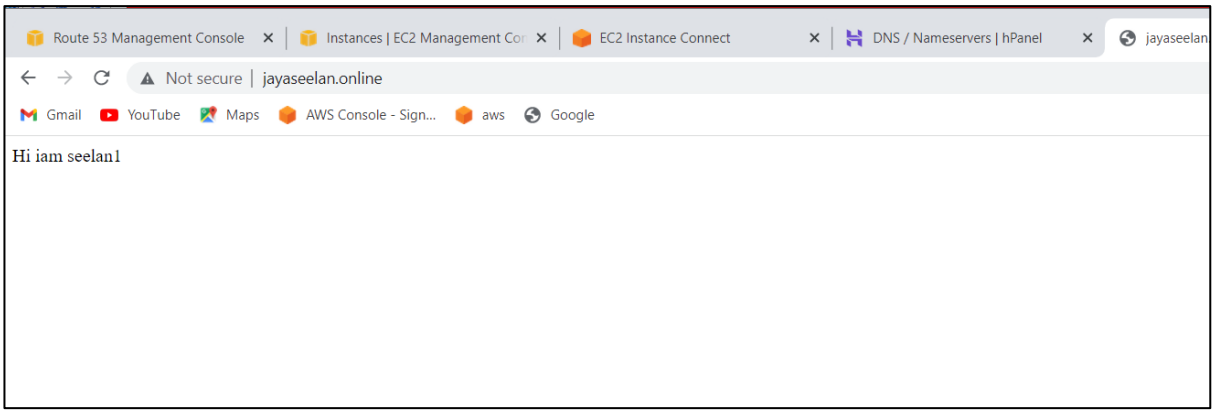
Step2.2:terminate server2



Now see health checks--->terminate server ip is unhealthy



Now put chrome text page **http://jayaseelan.online**----> it will show health record server content



Now check using **cmd** prompt ---->nslookup jayaseelan.online---->it will show healthy server ip