MERN deployment on AWS Cloud

Project Description:

The project is to deploy a Travel Memory application using the MERN stack (MongoDB, Express.js, React and Node.js) on AWS EC2 instances using different Application Load Balancers for frontend and backend services, and storing the data in the Mongo Database ensuring scalable architecture

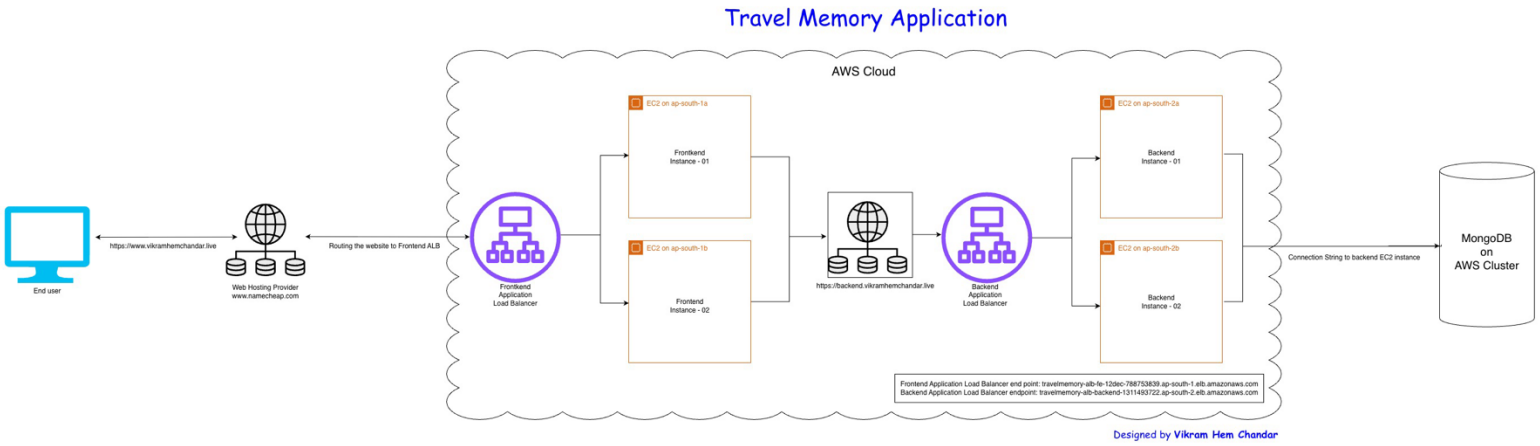
Requirements:

1. Mongo DB Connection String
2. VPC with 2 public and 2 private subnets
3. AMI instance with prerequisites applications installed
4. Two EC2 instances for backend and two EC2 instances for frontend
5. Target Group and Application Load Balancer for Backend and Frontend
6. A domain name to host the website on internet

Steps:

1. Backend configuration
   - Configure backend with two EC2 instances
   - Create a .env file to incorporate database connection details and port information
   - Deploy the backend service and test with individual EC2 instances
   - Create a certificate for backend using AWS Certificate Manager
   - Create an Application Load balancer with backend EC2 instances
   - Configure the CNAME record to link backend load balancer endpoint [DNS]
2. Configure frontend with two EC2 instances
   - Update url.js to ensure the front end communicates effectively with the backend
   - Deploy the frontend and test with individual EC2 instances
   - Create a certificate for frontend using AWS Certificate Manager
   - Create an Application Load Balancer for frontend EC2 instances
   - Configure the CNAME record to link frontend load balancer end point [DNS]
3. Domain Configuration
4. Application Testing

Architecture of Travel Memory Application



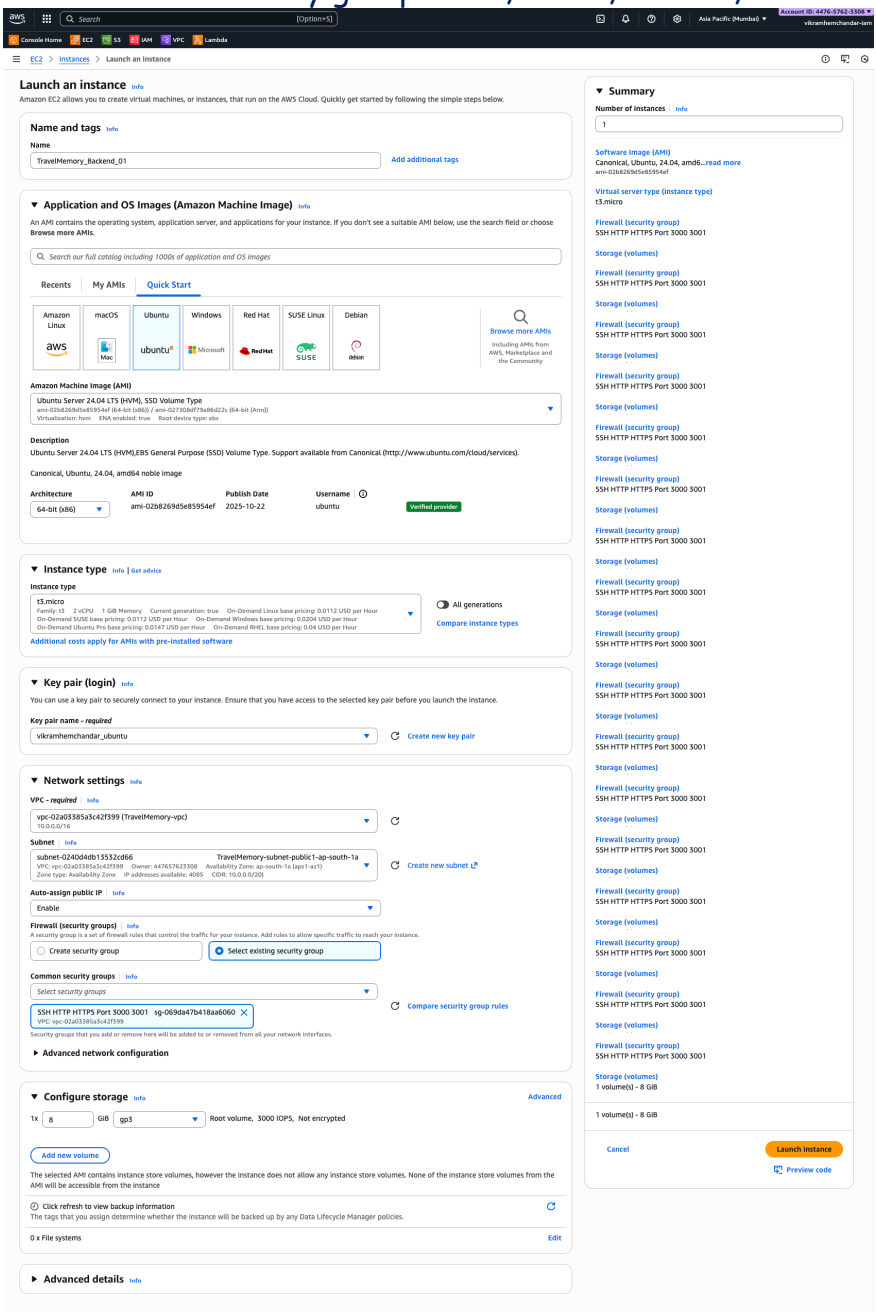
Flow of the application:

- End user opens the browser and hits https://www.vikramhemchandar.live
- The hit goes to DNS (application load balancer end point) managed by web hosting provier (here namecheap.com)
- The hit is routed to DNS (which is frontend ALB)
- Frontend ALB listens on https (port 443)
- ALB does health checks of the frontend EC2 instances
- ALB distributes the traffic using round-robin / lead outstanding requests
- Frontend is communicated with backend by backend DNS (in url.js)
- The request is route to backend ALB
- Backend ALB listens on HTTPS (port 443)
- Backend ALB does health check on the backend EC2 instances
- Backend ALB is communicated to EC2 instances and communicates to database which connect via connection string (in .env)
- Response flow (reverse path):
    - MongoDB returns data -> Backend Node.js
    - Backend Node.js returns API response -> Backend ALB
    - Backend ALB -> Frontend browser (via React)
    - React updates UI page is loaded

Go to https://github.com/vikramhemchandar/draw.io to reuse the diagram

Backend Configuration:

1. Create a separate VPC dedicated for this deployment (not mandatory)
2. Create an EC2 instance
    - OS: Ubuntu
    - t3.micro
    - network: select created VPC
    - subnet: public subnet
    - auto-assign public IP: Enable
    - Security group: SSH, HTTP, HTTPS, Port 3000 and 3001 enabled



3. Installing applications / packages and setting up reverse proxy using nginx EC2 instance

Run the below unix commands:

To update and upgrade the OS packages

- sudo apt update && apt upgrade -y

Install nginx

- sudo apt install nginx -y
- sudo systemctl start nginx
- sudo systemctl enable nginx

Set reverse proxy for individual instance and update the script for filename – default and path - /etc/nginx/sites-available

- cd /etc/nginx/sites-available
- sudo nano default
- cat default

```
server {
   listen 80;
   server_name vikramhemchandar.live www.vikramhemchandar.live;

   location / {
      proxy_pass http://127.0.0.1:3001;
      proxy_http_version 1.1;

      # Preserve client + original host info
      proxy_set_header Host            $host;
      proxy_set_header X-Real-IP       $remote_addr;
      proxy_set_header X-Forwarded-For   $proxy_add_x_forwarded_for;
      proxy_set_header X-Forwarded-Proto $scheme;

      # Good defaults
      proxy_connect_timeout 10s;
      proxy_send_timeout    60s;
      proxy_read_timeout    60s;
   }
}
```

Install NodeJS

- curl -fsSL https://deb.nodesource.com/setup_22.x | sudo -E bash -
- sudo apt install -y nodejs

Note: npm should be installed but it is path specific, hence, we will install for every EC2 instance respectively

Cloning the code from GitHub repository

- git clone https://github.com/UnpredictablePrashant/TravelMemory

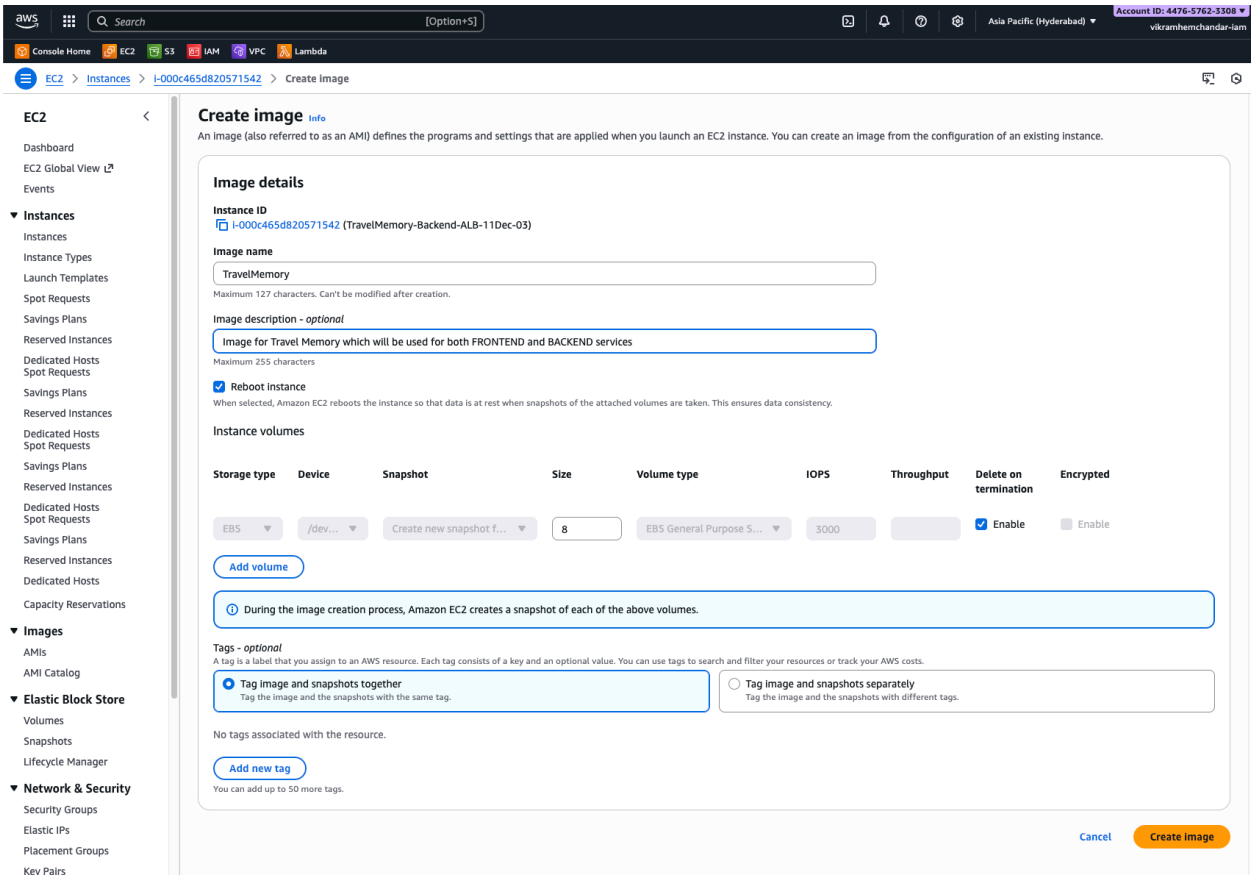
Create a .env file to incorporate database connection details and port information

- nano .env
- sudo nginx -t
- sudo systemctl reload nginx
- cat .env

MONGO_URI='mongodb+srv://vikram_DB:xxxxxxxx@vikramhemchandar.rtgw4pn.mongodb.net/travelmemory'
PORT=3001

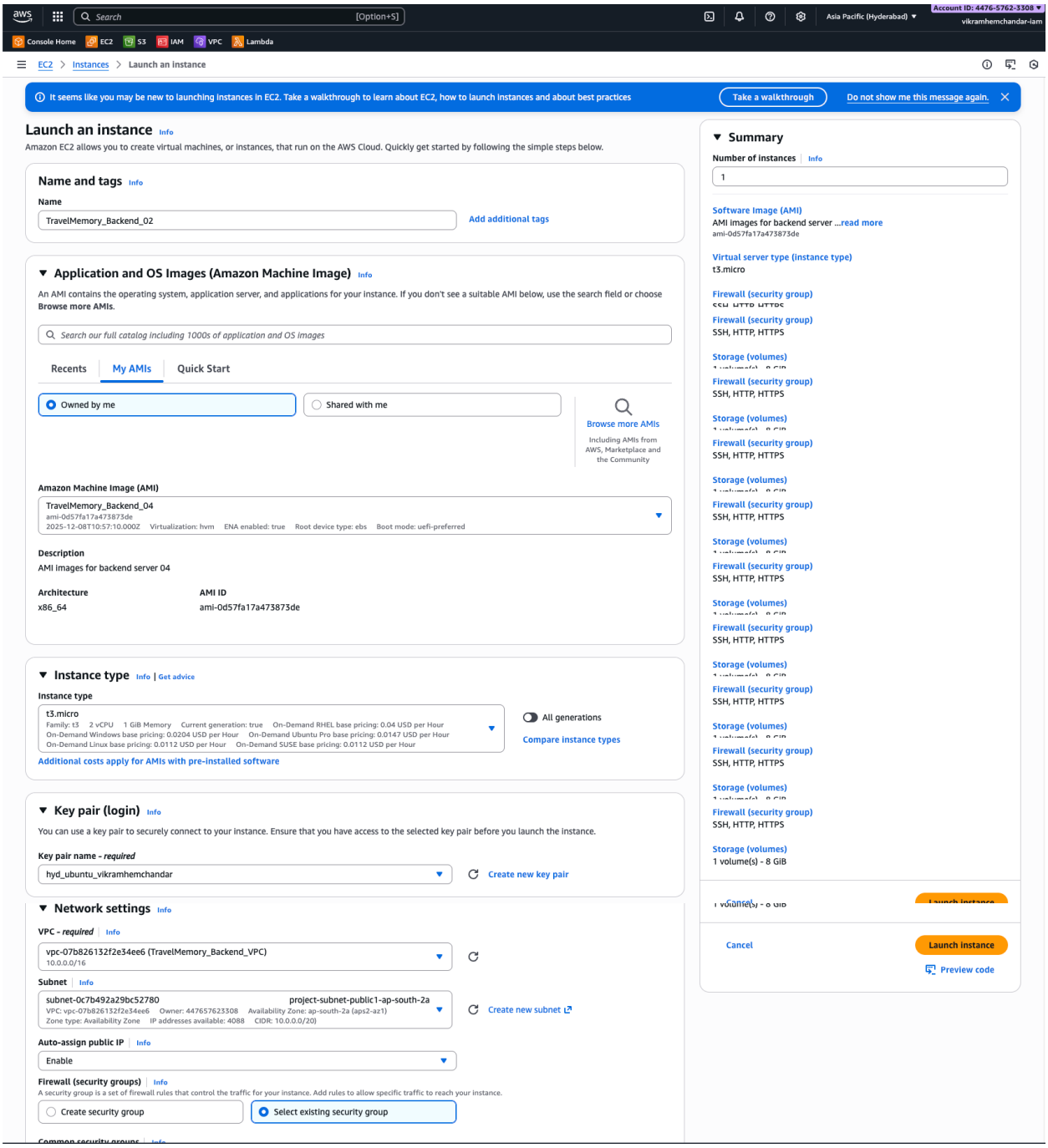
4. Create an AMI image

Create an AMI image by navigating on EC2 Instance -> Actions -> Image and templates -> Create image



5. Create 3 EC2 instances from the AMI image (in total 4 – 2 for BE and 2 for FE)

NOTE: For now, create only EC2 instance for another backend server, so, total 2 EC2 instances



6. Install npm for 2 backend instances and start the backend server

- cd TravelMemory/backend
- sudo npm install

Verify NodeJS and npm versions

- node -v
- npm -v

start the backend server

- node index.js

The expected message - Server started at http://localhost:3001

7. Verify backend server

To verify the back server, copy the public IP of two instances and paste it in any browser. And note, port number is not required as we have configured reverse proxy

Example: http://18.60.97.157/ and you should see below message –


```
Cannot GET /
```

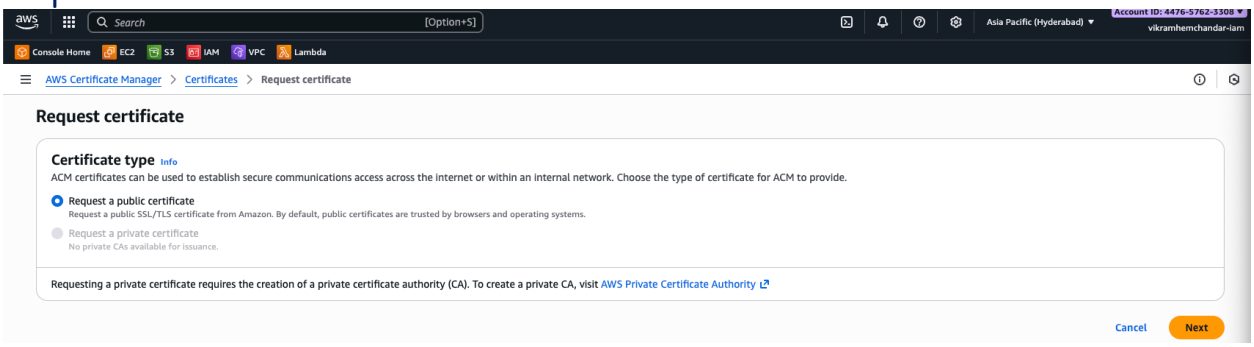
When you enter */hello* to the above IP address, you should see Hello World

Hello World! - 02

8. Create a certificate on ACM (AWS Certificate Manager) with your domain name

Note – there should a domain registered in any web hosting provider

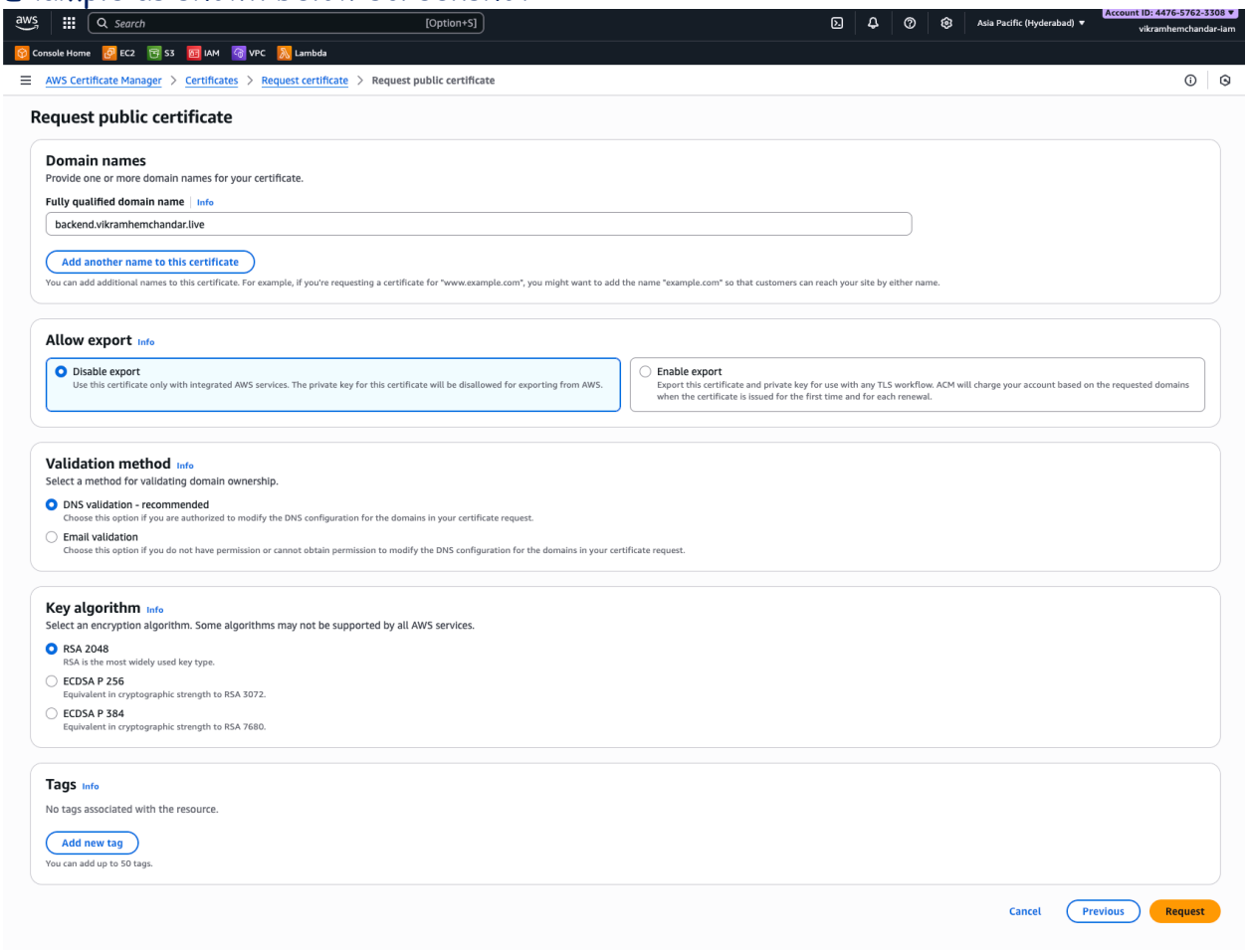
On AWS, search ACM -> go to certificate manager -> Request and follow the below steps



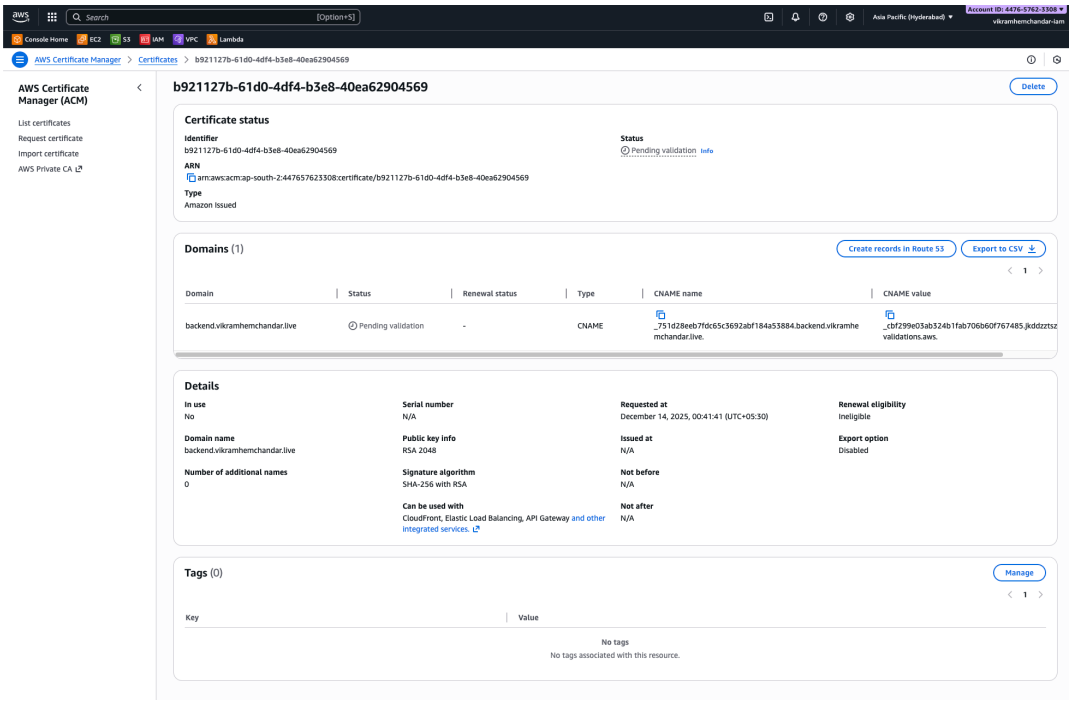
In *Request public certificate* page,

- Enter the below details:
  - Domain names: full qualified domain name
  - Once filled click on Request

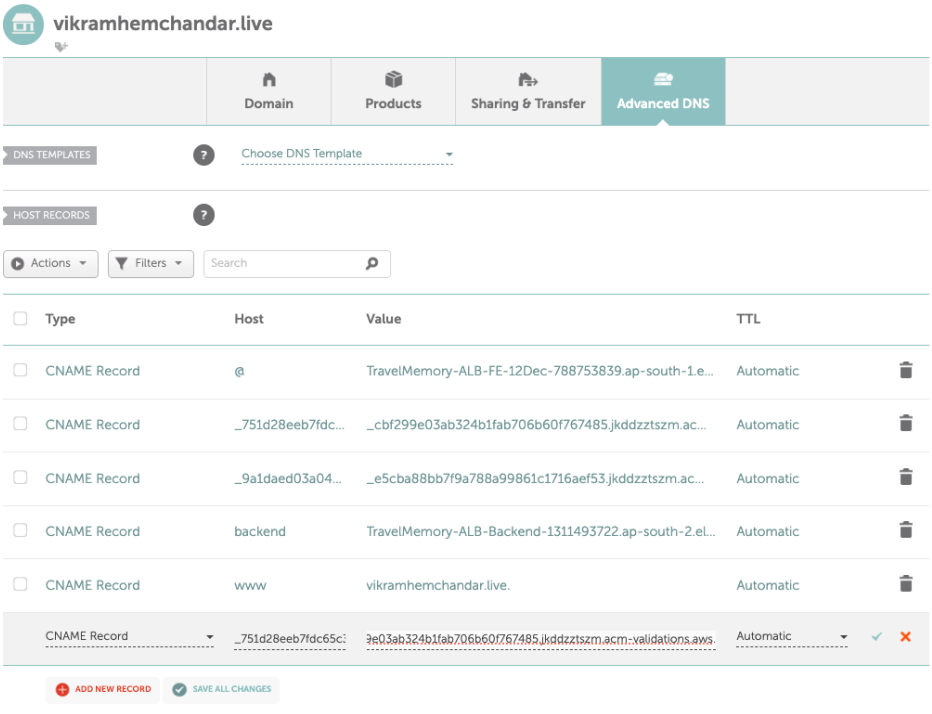
Example as shown below screenshot



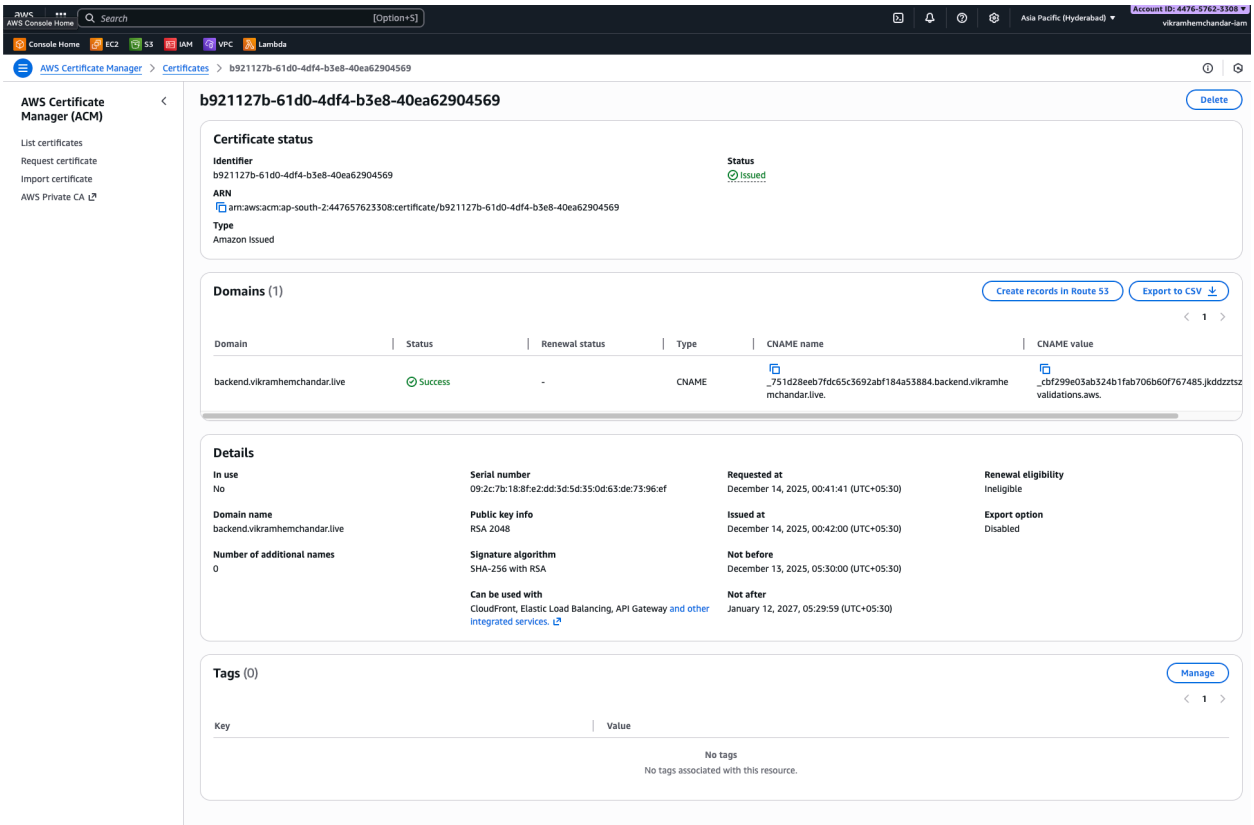
It will navigate to a page where it will have CNAME name and CNAME value as shown below



Add CNAME name (only till "backend") and CNAME value in your web hosting provide as CNAME record



Wait for some time, ACM should show a message as issued for the certificate



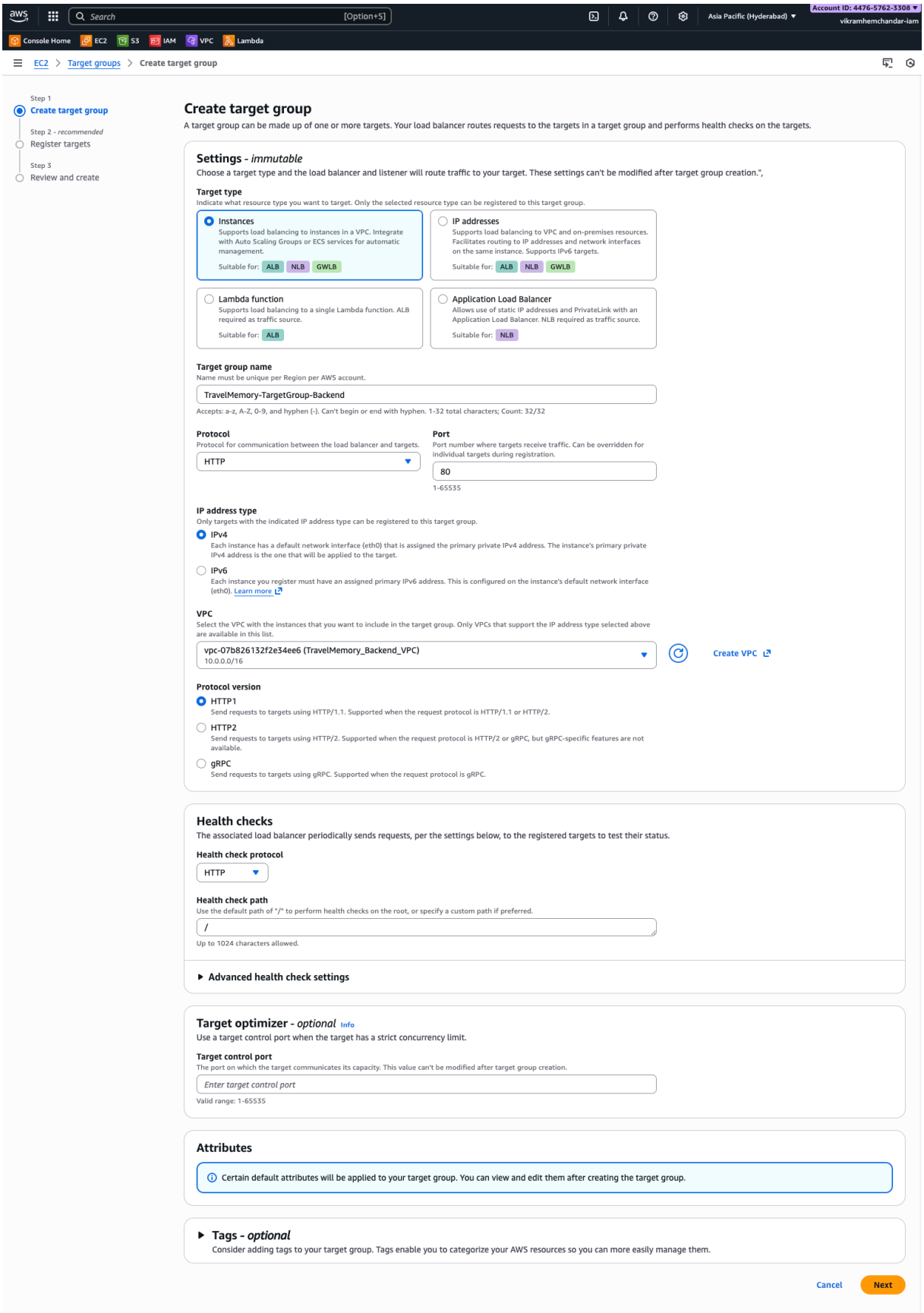
9. Create Target Group (with HTTP) and Application Load Balancer (with HTTPS) protocol and test the Load balancer

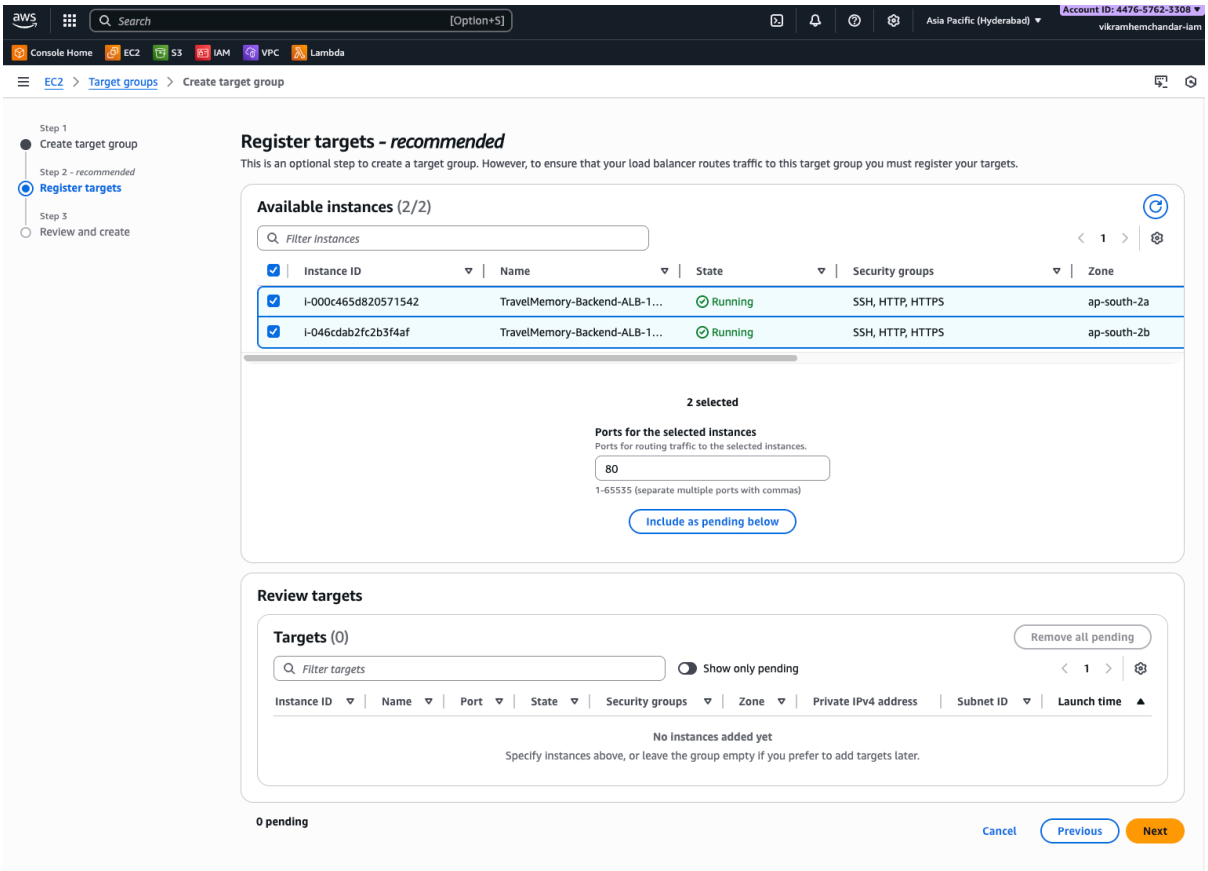
Create Target Groups, Navigate to EC2 -> Load Balancing -> Target Groups

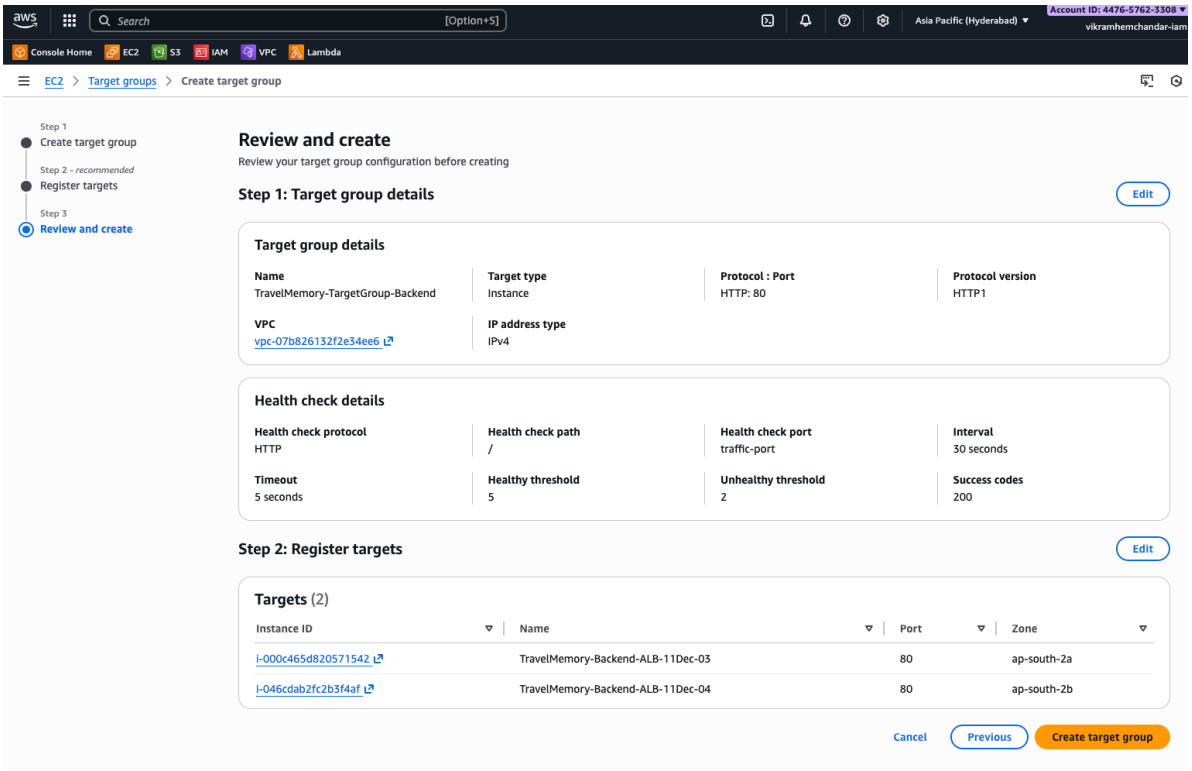
Under Create target group page, select:

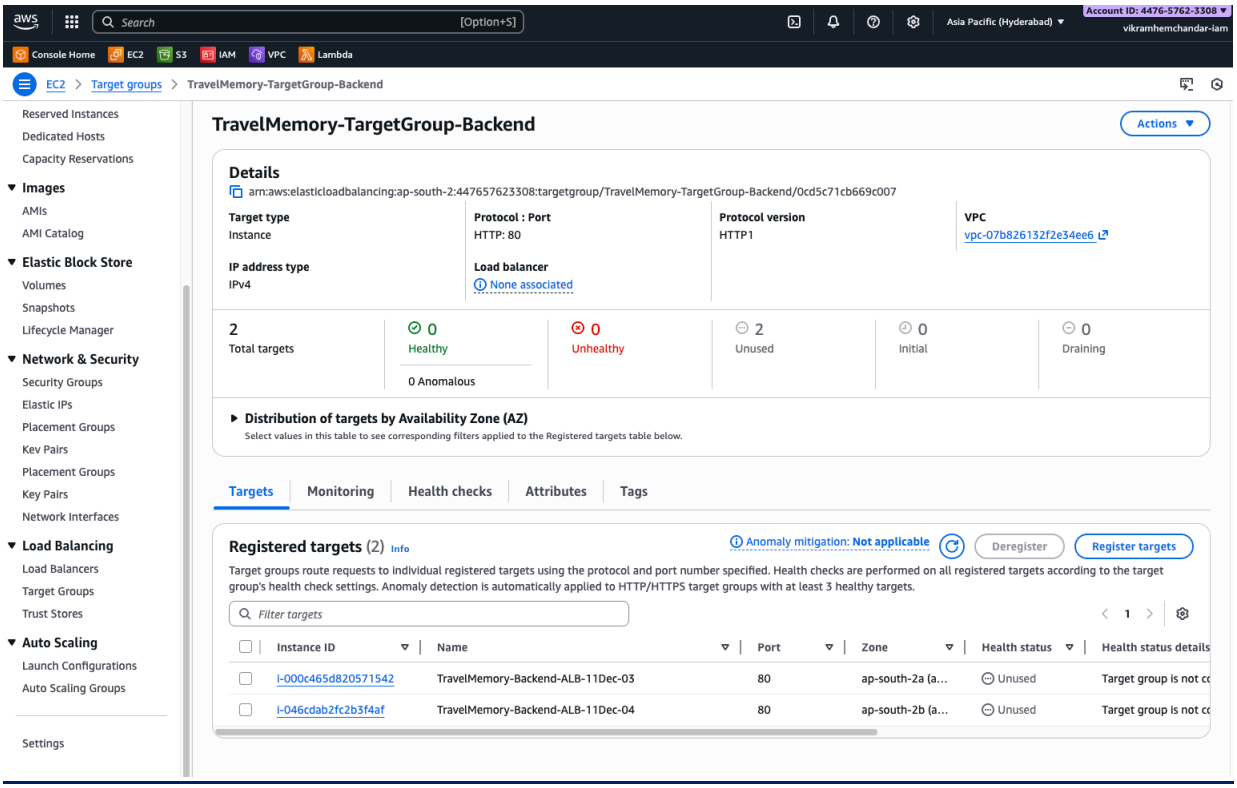
- Target Type: instances
- Target group name: TravelMemory-TG-Backend
- Protocol: HTTP
- Port: 80
- IP Address: IPv4
- VPC: <Select appropriate VPC>
- Protocol version: HTTP1
- Health checks: HTTP
- Click *Next*
- Under register targets page, select:
- Available instance: <select the two backend instances created>
- Click *Include as pending below*
- Make sure the ports should be 80
- Click on Next

In Review and create (final) page, click *Create target group*


Search
[Option+S]
Asia Pacific (Hyderabad)
Account ID: 4476-5762-3308
vikramhemchandar-iam
Console Home
EC2
S3
IAM
VPC
Lambda
EC2 > Target groups > Create target group
Step 1
Create target group
Step 2 - recommended
Register targets
Step 3
Review and create
Create target group
A target group can be made up of one or more targets. Your load balancer routes requests to the targets in a target group and performs health checks on the targets.
Settings - immutable
Choose a target type and the load balancer and listener will route traffic to your target. These settings can't be modified after target group creation.",
Target type
Indicate what resource type you want to target. Only the selected resource type can be registered to this target group.
Instances
Supports load balancing to instances in a VPC. Integrate with Auto Scaling Groups or ECS services for automatic management.
Suitable for: ALB NLB GWLB
IP addresses
Supports load balancing to VPC and on-premises resources. Facilitates routing to IP addresses and network interfaces on the same instance. Supports IPv6 targets.
Suitable for: ALB NLB GWLB
Lambda function
Supports load balancing to a single Lambda function. ALB required as traffic source.
Suitable for: ALB
Application Load Balancer
Allows use of static IP addresses and PrivateLink with an Application Load Balancer. NLB required as traffic source.
Suitable for: NLB
Target group name
Name must be unique per Region per AWS account.
TravelMemory-TargetGroup-Backend
Accepts: a-z, A-Z, 0-9, and hyphen (-). Can't begin or end with hyphen. 1-32 total characters; Count: 32/32
Protocol
Protocol for communication between the load balancer and targets.
HTTP
Port
Port number where targets receive traffic. Can be overridden for individual targets during registration.
80
1-65535
IP address type
Only targets with the indicated IP address type can be registered to this target group.
IPv4
Each instance has a default network interface (eth0) that is assigned the primary private IPv4 address. The instance's primary private IPv4 address is the one that will be applied to the target.
IPv6
Each instance you register must have an assigned primary IPv6 address. This is configured on the instance's default network interface (eth0). Learn more
VPC
Select the VPC with the instances that you want to include in the target group. Only VPCs that support the IP address type selected above are available in this list.
vpc-07b826132f2e34ee6 (TravelMemory_Backend_VPC)
10.0.0.0/16
Create VPC
Protocol version
HTTP1
Send requests to targets using HTTP/1.1. Supported when the request protocol is HTTP/1.1 or HTTP/2.
HTTP2
Send requests to targets using HTTP/2. Supported when the request protocol is HTTP/2 or gRPC, but gRPC-specific features are not available.
gRPC
Send requests to targets using gRPC. Supported when the request protocol is gRPC.
Health checks
The associated load balancer periodically sends requests, per the settings below, to the registered targets to test their status.
Health check protocol
HTTP
Health check path
Use the default path of "/" to perform health checks on the root, or specify a custom path if preferred.
/
Up to 1024 characters allowed.
Advanced health check settings
Target optimizer - optional Info
Use a target control port when the target has a strict concurrency limit.
Target control port
The port on which the target communicates its capacity. This value can't be modified after target group creation.
Enter target control port
Valid range: 1-65535
Attributes
Certain default attributes will be applied to your target group. You can view and edit them after creating the target group.
Tags - optional
Consider adding tags to your target group. Tags enable you to categorize your AWS resources so you can more easily manage them.
Cancel
Next


Search [Option+S]
Asia Pacific (Hyderabad)
Account ID: 4476-5762-3308
vikramhemchandar-iam
Console Home EC2 S3 IAM VPC Lambda
EC2 > Target groups > Create target group
Step 1
Create target group
Step 2 - recommended
Register targets
Step 3
Review and create
Register targets - recommended
This is an optional step to create a target group. However, to ensure that your load balancer routes traffic to this target group you must register your targets.
Available instances (2/2)
Filter instances
Instance ID
Name
State
Security groups
Zone
i-000c465d820571542
TravelMemory-Backend-ALB-1...
Running
SSH, HTTP, HTTPS
ap-south-2a
i-046cdab2fc2b3f4af
TravelMemory-Backend-ALB-1...
Running
SSH, HTTP, HTTPS
ap-south-2b
2 selected
Ports for the selected instances
Ports for routing traffic to the selected instances.
80
1-65535 (separate multiple ports with commas)
Include as pending below
Review targets
Targets (0)
Remove all pending
Filter targets
Show only pending
Instance ID
Name
Port
State
Security groups
Zone
Private IPv4 address
Subnet ID
Launch time
No instances added yet
Specify instances above, or leave the group empty if you prefer to add targets later.
0 pending
Cancel
Previous
Next


Search [Option+S]
Asia Pacific (Hyderabad)
Account ID: 4476-5762-3308
vikramhemchandar-iam
Console Home EC2 S3 IAM VPC Lambda
EC2 > Target groups > Create target group
Step 1
Create target group
Step 2 - recommended
Register targets
Step 3
Review and create
Review and create
Review your target group configuration before creating
Step 1: Target group details
Edit
Target group details
Name
TravelMemory-TargetGroup-Backend
Target type
Instance
Protocol : Port
HTTP: 80
Protocol version
HTTP1
VPC
vpc-07b826132f2e34ee6
IP address type
IPv4
Health check details
Health check protocol
HTTP
Health check path
/
Health check port
traffic-port
Interval
30 seconds
Timeout
5 seconds
Healthy threshold
5
Unhealthy threshold
2
Success codes
200
Step 2: Register targets
Edit
Targets (2)
Instance ID
Name
Port
Zone
i-000c465d820571542
TravelMemory-Backend-ALB-11Dec-03
80
ap-south-2a
i-046cdab2fc2b3f4af
TravelMemory-Backend-ALB-11Dec-04
80
ap-south-2b
Cancel
Previous
Create target group



Create Load Balancer, navigate to EC2 -> Load Balancing -> Load Balancers
Select Application Load Balancer as type and click on *create*
Under Create Application Load Balancer page, select:

- Load Balancer Name: <Appropriate name>
- Scheme: Internet-facing
- Load Balancer IP address: IPv4
- VPC: <select appropriate VPC>
- Security Groups: <select appropriate security groups>
- Under Listeners and routing:
    - Protocol: HTTPS
    - Port: 443
- Under Default action:
    - Routing action: Forward to target groups
    - Target group: <select recently created target group>
- Under Default SSL/TLS server certificate:
    - Certificate source: From ACM
    - Certificate (from ACM): <select the certificate which is created>
- Click on Create load balancer

Note: wait for at least 2-3 minutes to get the status from provisioning to Active

Search [Option+S]
Asia Pacific (Hyderabad) Account ID: 4476-5762-3308 vikramhemchandar-iam

Console Home | EC2 | S3 | IAM | VPC | Lambda

EC2 > Load balancers > Create Application Load Balancer

Create Application Load Balancer Info

The Application Load Balancer distributes incoming HTTP and HTTPS traffic across multiple targets such as Amazon EC2 instances, microservices, and containers, based on request attributes. When the load balancer receives a connection request, it evaluates the listener rules in priority order to determine which rule to apply, and if applicable, it selects a target from the target group for the rule action.

▶ **How Application Load Balancers work**

Basic configuration

Load balancer name
Name must be unique within your AWS account and can't be changed after the load balancer is created.
TravelMemory-ApplicationLoadBalancer-Backend
A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

Scheme Info
Scheme can't be changed after the load balancer is created.

- (selected) Internet-facing
  - Serves internet-facing traffic.
  - Has public IP addresses.
  - DNS name resolves to public IPs.
  - Requires a public subnet.
- Internal
  - Serves internal traffic.
  - Has private IP addresses.
  - DNS name resolves to private IPs.
  - Compatible with the **IPv4** and **Dualstack** IP address types.

Load balancer IP address type Info
Select the front-end IP address type to assign to the load balancer. The VPC and subnets mapped to this load balancer must include the selected IP address types. Public IPv4 addresses have an additional cost.

- (selected) IPv4
  Includes only IPv4 addresses.
- Dualstack
  Includes IPv4 and IPv6 addresses.
- Dualstack without public IPv4
  Includes a public IPv6 address, and private IPv4 and IPv6 addresses. Compatible with **internet-facing** load balancers only.

Network mapping Info

The load balancer routes traffic to targets in the selected subnets, and in accordance with your IP address settings.

VPC Info
The load balancer will exist and scale within the selected VPC. The selected VPC is also where the load balancer targets must be hosted unless routing to Lambda or on-premises targets, or if using VPC peering. To confirm the VPC for your targets, view target groups.
vpc-07b826132f2e34ee6 (TravelMemory_Backend_VPC)
10.0.0.0/16
Create VPC

IP pools Info
You can optionally choose to configure an IPAM pool as the preferred source for your load balancers IP addresses. Create or view **Pools** in the Amazon VPC IP Address Manager console.
☐ Use IPAM pool for public IPv4 addresses
The IPAM pool you choose will be the preferred source of public IPv4 addresses. If the pool is depleted IPv4 addresses will be assigned by AWS.

Availability Zones and subnets Info
Select at least two Availability Zones and a subnet for each zone. A load balancer node will be placed in each selected zone and will automatically scale in response to traffic. The load balancer routes traffic to targets in the selected Availability Zones only.

☑ **ap-south-2a (aps2-az1)**
Subnet
Only CIDR blocks corresponding to the load balancer IP address type are used. At least 8 available IP addresses are required for your load balancer to scale efficiently.
subnet-0c7b492a29bc52780 — project-subnet-public1-ap-south-2a
IPv4 subnet CIDR: 10.0.0.0/20

☑ **ap-south-2b (aps2-az2)**
Subnet
Only CIDR blocks corresponding to the load balancer IP address type are used. At least 8 available IP addresses are required for your load balancer to scale efficiently.
subnet-090c2a92361e69a44 — project-subnet-public2-ap-south-2b
IPv4 subnet CIDR: 10.0.16.0/20

Security groups Info

A security group is a set of firewall rules that control the traffic to your load balancer. Select an existing security group, or you can create a new security group.

Security groups
Select up to 5 security groups
SSH, HTTP, HTTPS
sg-0ded003379a454cba VPC: vpc-07b826132f2e34ee6

Listeners and routing Info

A listener is a process that checks for connection requests using the port and protocol you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets.

▼ Listener **HTTPS:443** Remove

Protocol: HTTPS
Port: 443
1-65535

Default action Info

The default action is used if no other rules apply. Choose the default action for traffic on this listener.

Pre-routing action Info
- (selected) No pre-routing action
- Authenticate user
  User auth with OIDC or Amazon Cognito
- Validate token
  Validate client-presented JSON Web Token (JWT)

Routing action
- (selected) Forward to target groups
- Redirect to URL
- Return fixed response

Forward to target group Info
Choose a target group and specify routing weight or create target group.

Target group	Weight	Percent
TravelMemory-TargetGroup-Backend HTTP Target type: Instance, IPv4 | Target stickiness: Off	1 0-999	100%

+ Add target group
You can add up to 4 more target groups.

Target group stickiness Info
Enables the load balancer to bind a user's session to a specific target group. To use stickiness the client must support cookies. If you want to bind a user's session to a specific target, turn on the Target Group attribute Stickiness.
☐ Turn on target group stickiness

Listener tags - *optional*
Consider adding tags to your listener. Tags enable you to categorize your AWS resources so you can more easily manage them.

Add listener tag
You can add up to 50 more tags.

Add listener
You can add up to 49 more listeners.

Secure listener settings Info

These settings will apply to all of your secure listeners. Once created, you can manage these settings per listener.

Security policy Info

Your load balancer uses a Secure Socket Layer (SSL) negotiation configuration called a security policy to manage SSL connections with clients. Compare security policies

Security category: All security policies
Policy name New: ELBSecurityPolicy-TLS13-1-2-Res-PQ-2025-09 (Recommended)

Default SSL/TLS server certificate

The certificate used if a client connects without SNI protocol, or if there are no matching certificates. You can source this certificate from AWS Certificate Manager (ACM), Amazon Identity and Access Management (IAM), or import a certificate. This certificate will automatically be added to your listener certificate list.

Certificate source
- (selected) From ACM
- From IAM
- Import certificate

Certificate (from ACM)
The selected certificate will be applied as the default SSL/TLS server certificate for this load balancer's secure listeners.
backend.vikramhemchandar.live
eb45f887-b6c1-415c-b2e4-5309293f8567
Request new ACM certificate

Client certificate handling Info

Client certificates are used to make authenticated requests to remote servers. Learn more
☐ **Mutual authentication (mTLS)**
Mutual TLS (Transport Layer Security) authentication offers two-way peer authentication. It adds a layer of security over TLS and allows your services to verify the client that's making the connection.

▶ **Load balancer tags** - *optional*
Consider adding tags to your load balancer. Tags enable you to categorize your AWS resources so you can more easily manage them. The 'Key' is required, but 'Value' is optional. For example, you can have Key = production-webserver, or Key = webserver, and Value = production.

Optimize with service integrations - *optional* Info

Optimize your load balancing architecture by integrating AWS services with this load balancer at launch. You can also add these and other services after your load balancer is created by reviewing the load balancer's "Integrations" tab.

Amazon CloudFront + AWS Web Application Firewall (WAF) - *new* Info
Optimizes: **Performance, Availability, Security**
☐ Apply application layer acceleration and security protections - *in front of the load balancer*
Automatically configures and creates a CloudFront distribution with the basic recommended AWS WAF security protections, and associates it to your load balancer. Additional charges apply
▶ **Benefits and considerations**

AWS Web Application Firewall (WAF) Info
Optimizes: **Security**
☐ Apply application layer security protections - *in front of targets*
Your choice of either a pre-defined security configuration with basic recommended AWS WAF security protections, or associate any of your existing WAF configurations for custom protections. Additional charges apply
▶ **Benefits and considerations**

AWS Global Accelerator Info
Optimizes: **Performance, Availability**
☐ Apply global load balancing across multiple regions
Creates an accelerator in your account with two global static IPs that act as a fixed entry point to your load balancer. If you do not need global static IPs or traffic management across multiple regions, select Amazon CloudFront. Additional charges apply
▶ **Benefits and considerations**

Review

Review the load balancer configurations and make changes if needed. After you finish reviewing the configurations, choose **Create load balancer**.

Summary

Review and confirm your configurations. Estimate cost

Basic configuration Edit
Name: TravelMemory-ApplicationLoadBalancer-Backend
Scheme: Internet-facing
IP address type: IPv4

Network mapping Edit
VPC: vpc-07b826132f2e34ee6
Public IPv4 IPAM pool: -
Availability Zones and subnets:
- ap-south-2a
  subnet-0c7b492a29bc52780
  project-subnet-public1-ap-south-2a
- ap-south-2b
  subnet-090c2a92361e69a44
  project-subnet-public2-ap-south-2b

Security groups Edit
SSH, HTTP, HTTPS
sg-0ded003379a454cba

Listeners and routing Edit
HTTPS:443 | Forward to 1 target group
Secure listener settings:
- ELBSecurityPolicy-TLS13-1-2-Res-PQ-2025-09
- backend.vikramhemchandar.live
  From ACM

Service integrations Edit
Amazon CloudFront + AWS Web Application Firewall (WAF): -
AWS WAF: -
AWS Global Accelerator: -

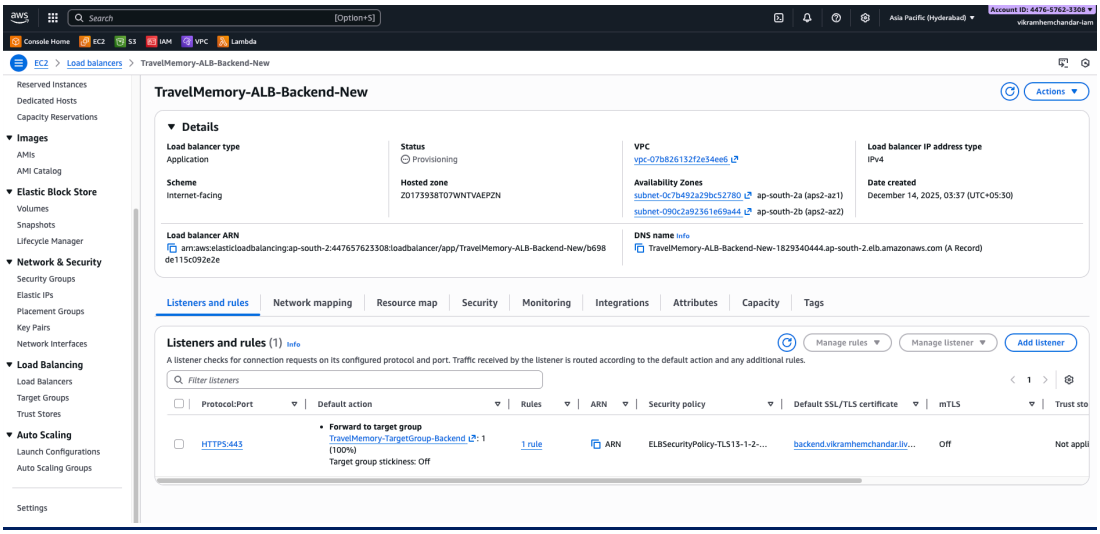
Tags Edit
-

Attributes
ⓘ Certain default attributes will be applied to your load balancer. You can view and edit them after creating the load balancer.

Creation workflow and status

▶ **Server-side tasks and status**
After completing and submitting the above steps, all server-side tasks and their statuses become available for monitoring.

Cancel | Create load balancer

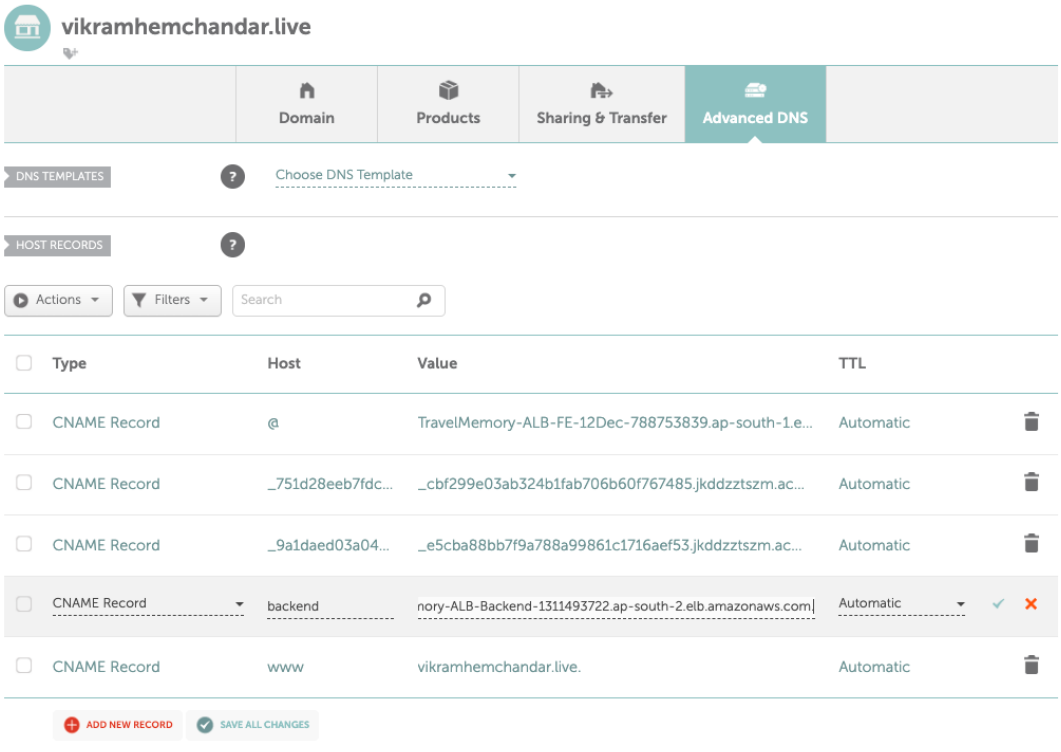


[Testing Load Balancer]:

- Once the ALB is active, copy the dns and paste it any browser. Prefix with https:// and suffix with /hello to the dns url and you should see Hello World

[Add CNAME] to web hosting provider: Add the dns (backend endpoint) in your name as –

- Type: CNAME
- Host: backend
- Value: dns (without prefix and suffix)

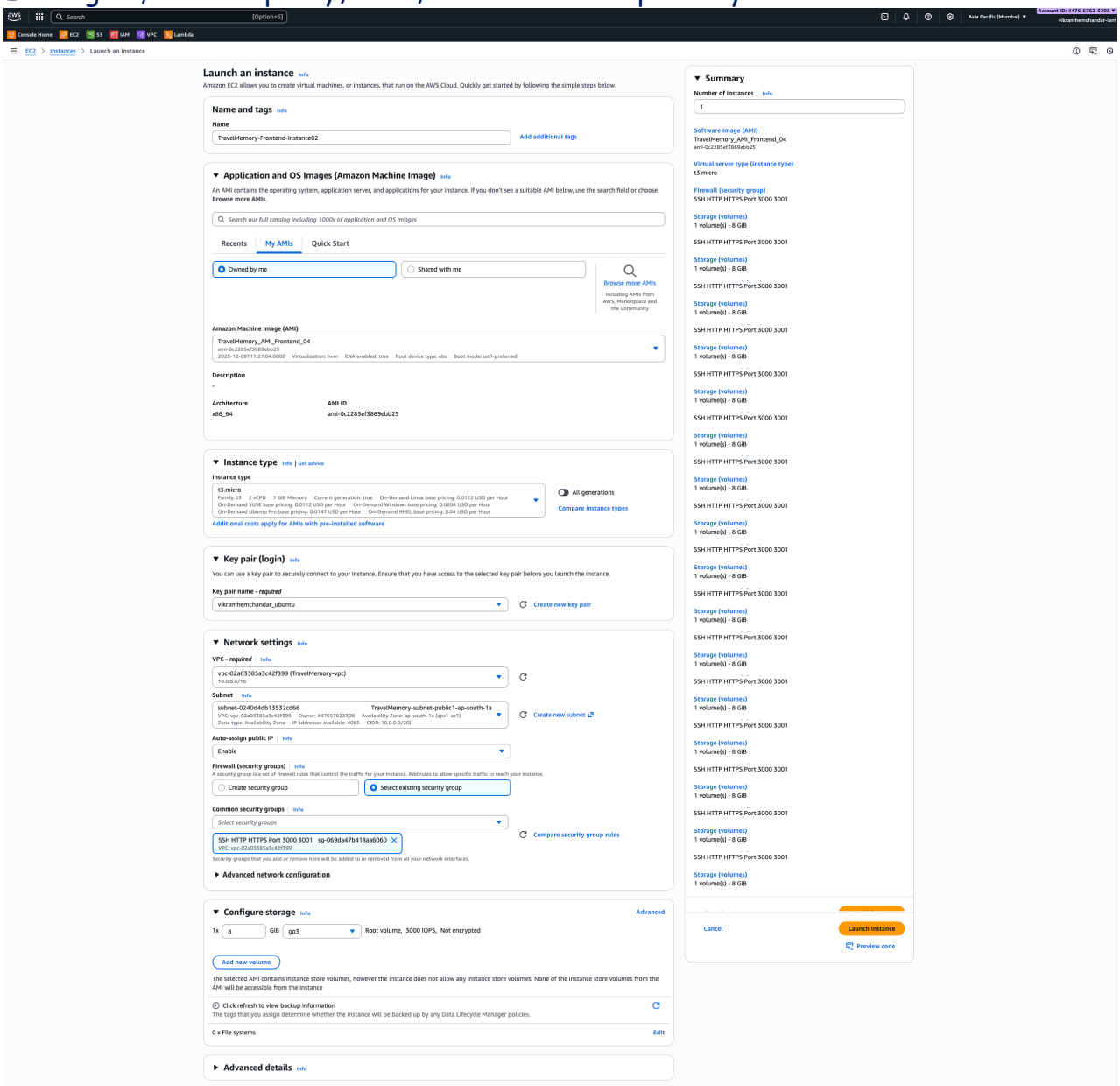


Now, backend should be accessible at https://backend.vikramhemchandar.live

Frontend Configuration:

1. Configure 2 frontend EC2 instances (from the above AMI)
   - OS: Ubuntu
   - My AMIs: <select appropriate AMI>
   - network: select created VPC
   - subnet: public subnet
   - auto-assign public IP: Enable
   - Security group: SSH, HTTP, HTTPS, Port 3000 and 3001 enabled

Note: Wait for 3-4 minutes for AMI to install the applications and configure the EC2: nginx, reverse proxy, node, GitHub clone repository



2. Configure URL.js for frontend to connect to the backend service

Copy the backend URL and navigate to /TravelMemory/frontend/src and edit the file url.js and paste the URL

- nano url.js
- cat url.js

```
export const baseUrl = process.env.REACT_APP_BACKEND_URL ||
"https://backend.vikramhemchandar.live";
```

3. Install npm for 2 frontend instances and start the backend server

- cd TravelMemory/frontend
- sudo npm install

Verify NodeJS and npm versions

- node -v
- npm -v

start the frontend server

- npm start

this should start the node webserver

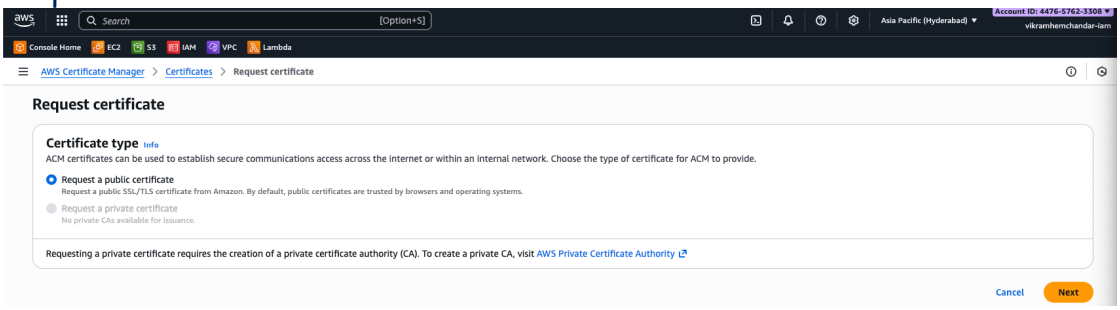
Test the individual frontend servers:

- Copy the public IP servers of each and paste it in any browser and test the application
- Enter a travel memory detail in this page for the all the fields and submit it, it should save in the database

4. Create a certificate on ACM (AWS Certificate Manager) with your domain name

Note – there should a domain registered in any web hosting provider. For screenshots refer Step 8 in backend configuration

On AWS, search ACM -> go to certificate manager -> Request and follow the below steps



In *Request public certificate* page,

- Enter the below details:
    - Domain names: full qualified domain name
    - Once filled click on Request
- ➢ It will navigate to a page where it will have CNAME name and CNAME value
- ➢ Add CNAME name and CNAME value in your web hosting provide as CNAME record
- ➢ Wait for some time, ACM should show a message as issued for the certificate

5. Create Target Group (with HTTP) and Application Load Balancer (with HTTPS) protocol and test the Load balancer

Create Target Groups, Navigate to EC2 -> Load Balancing -> Target Groups
Under Create target group page, select:

- Target Type: instances
- Target group name: TravelMemory-TG-Frontend
- Protocol: HTTP
- Port: 80
- IP Address: IPv4
- VPC: <Select appropriate VPC>
- Protocol version: HTTP1
- Health checks: HTTP
- Click *Next*
- Under register targets page, select:
    - Available instance: <select the two backend instances created>
    - Click *Include as pending below*
    - Make sure the ports should be 80
- Click on Next

In Review and create (final) page, click *Create target group*

Create Load Balancer, navigate to EC2 -> Load Balancing -> Load Balancers
Select Application Load Balancer as type and click on *create*
Under Create Application Load Balancer page, select:

- Load Balancer Name: <Appropriate name>
- Scheme: Internet-facing
- Load Balancer IP address: IPv4
- VPC: <select appropriate VPC>
- Security Groups: (select appropriate security groups)

- Under Listeners and routing:
    - Protocol: HTTPS
    - Port: 443
- Under Default action:
    - Routing action: Forward to target groups
    - Target group: <select recently created target group>
- Under Default SSL/TLS server certificate:
    - Certificate source: From ACM
    - Certificate (from ACM): <select the certificate which is created>
- Click on Create load balancer

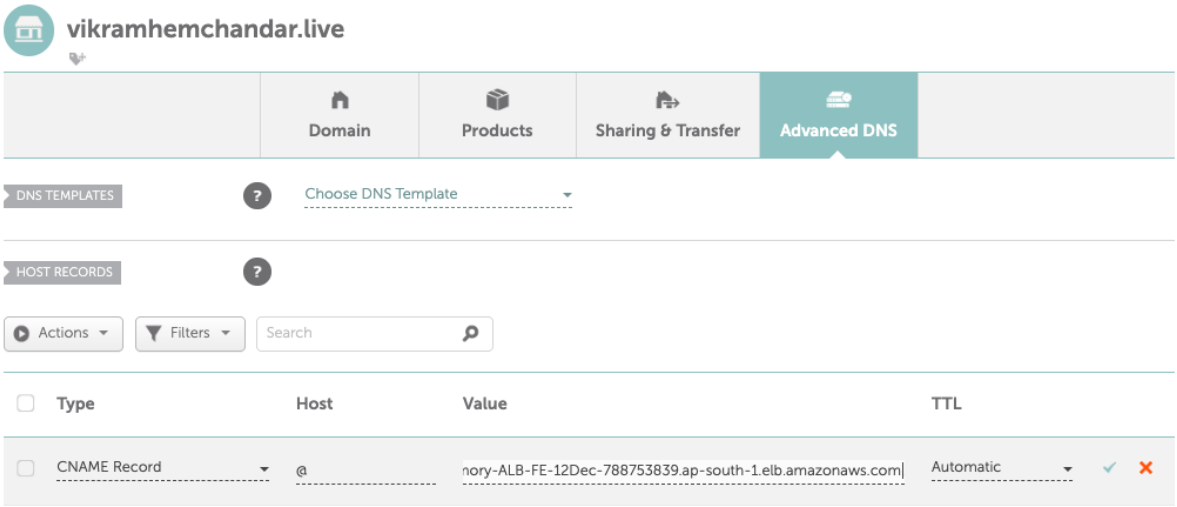
Note: wait for at least 2-3 minutes till it gets provisioning to Active status

Testing Load Balancer:

- Once the ALB is active, copy the dns and paste it any browser. Prefix with https:// to the dns url and you should see Travel Memory application

Add CNAME to web hosting provider: Add the dns (frontend endpoint) in your name as –

- Type: CNAME
- Host: @
- Value: dns (without prefix and suffix)



Domain Configuration:

⇒ Add CNAME records for the domain

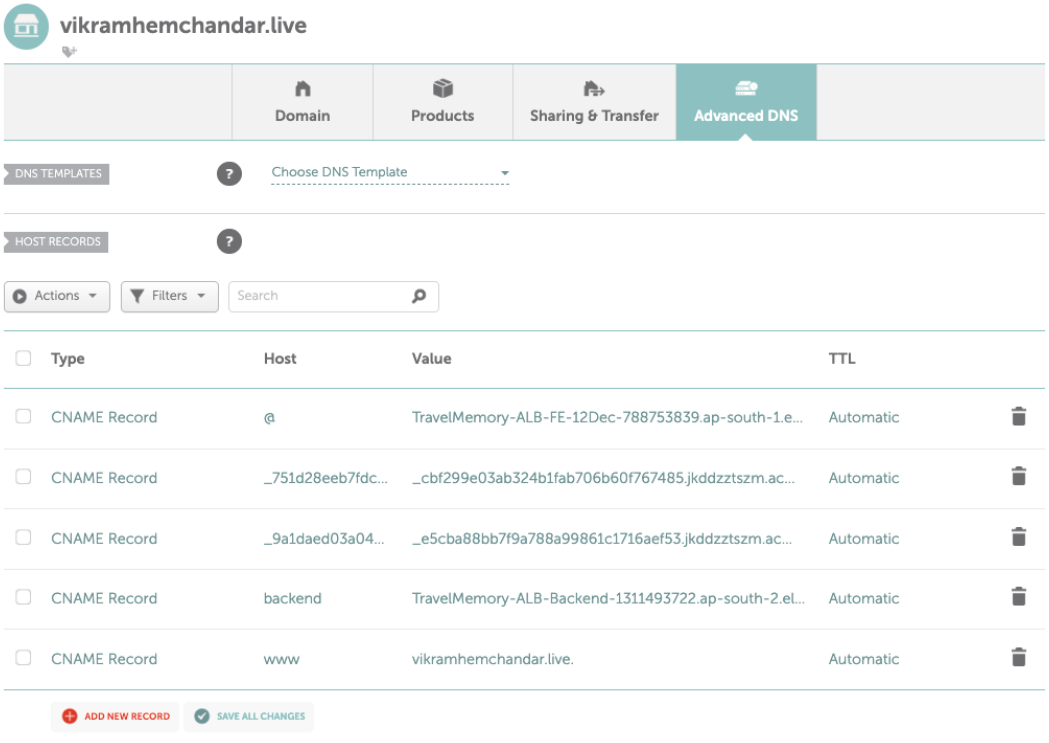
Add the below records:

- CNAME Record:
  - Type: CNAME
  - Host: www
  - Value: vikramhemchandar.live

In total, there should be 5 CNAME entries for this project:

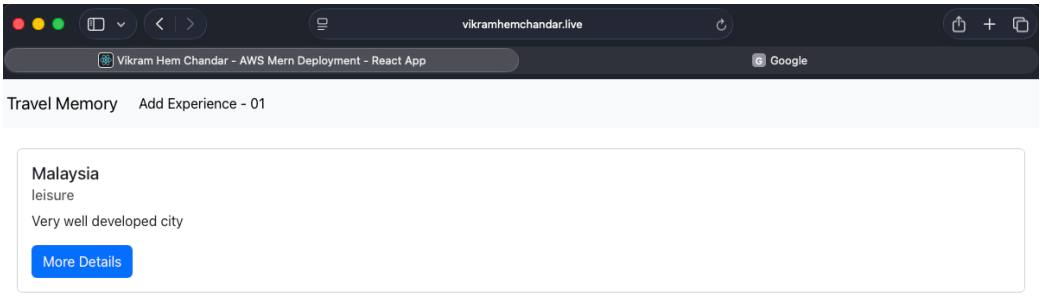
1. Domain – host: www and value: vikramhemchandar.live
2. Backend certificate: Step 8 in Backend configuration (for backend.vikramhemchandar.live)
3. Frontend certificate: Step 4 in Frontend configuration (for www.vikramhemchandar.live)
4. Backend DNS endpoint entry
5. Frontend DNS endpoint entry

Tip – if you need to configure IP address for a single instance, create A record and provide the IP Address of EC2 in Value column

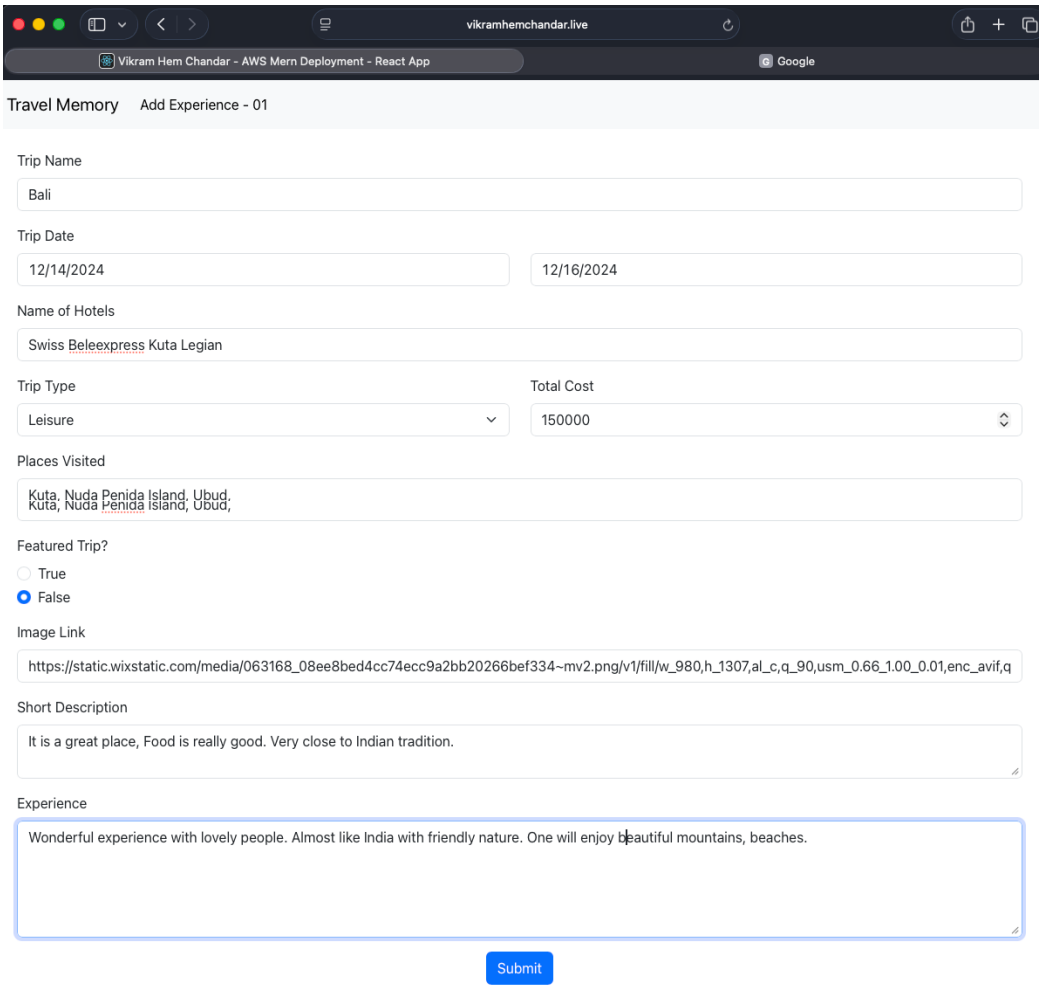


Application Testing:

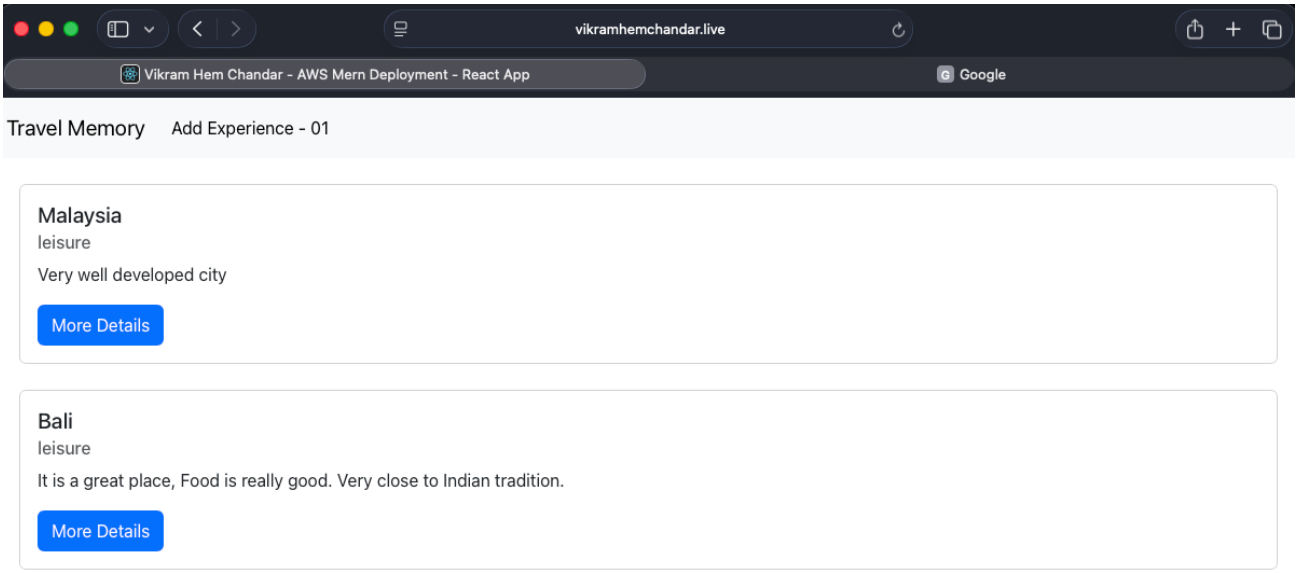
- Open the application at https://www.vikramhemchandar.live



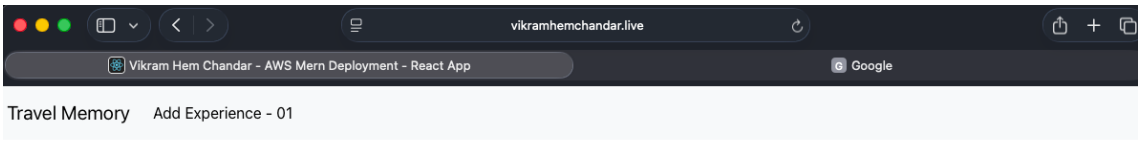
- Click on Add Experience and add your travel memory and submit it



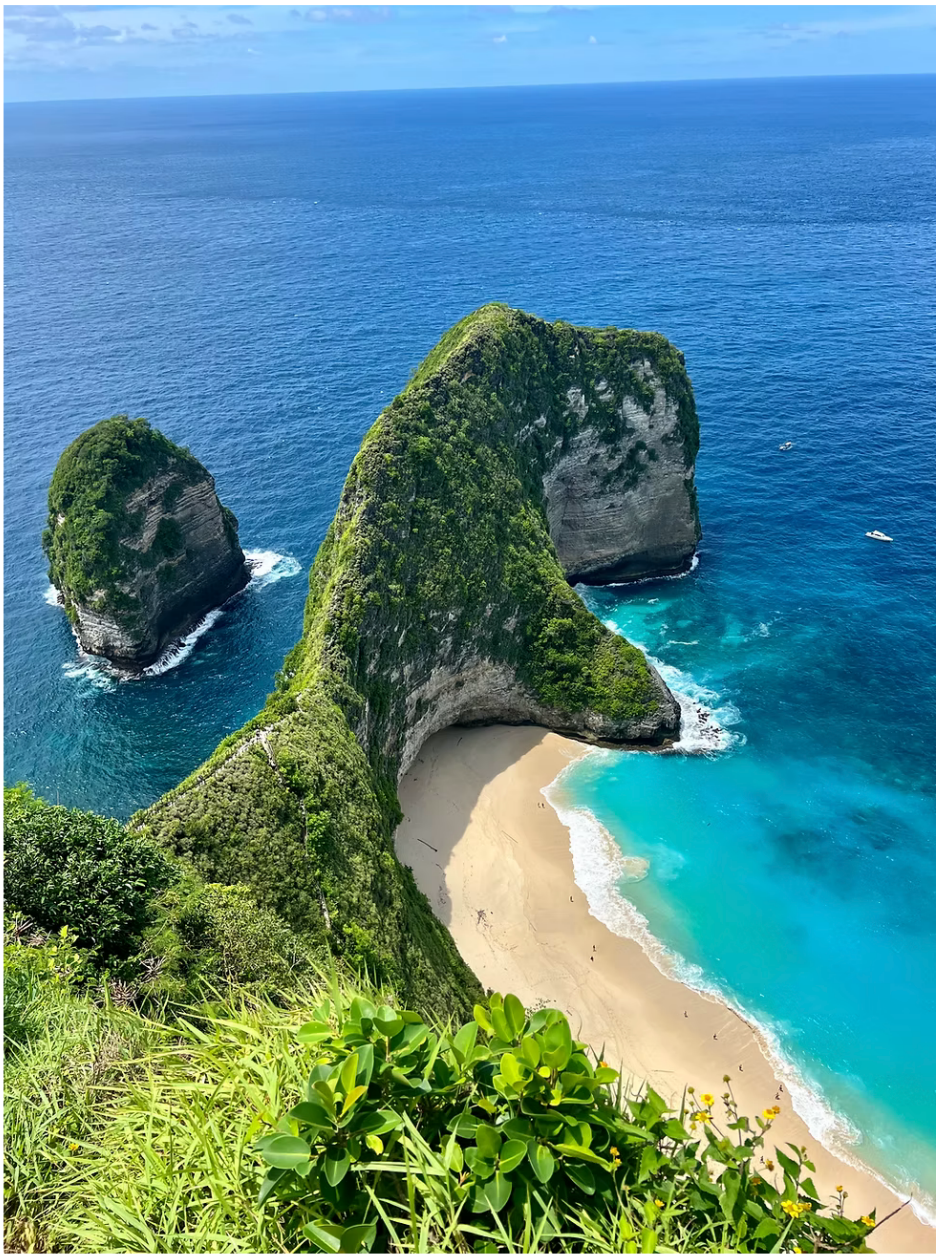
- Go to Travel memory on top left, you should see your travel memory entry added in the list. Here, it is Bali.



- Click on more details to see your travel memory details



Bali



Name of Hotel: Swiss Beleexpress Kuta Legian
Start Date: 2024-12-14 End Date: 2024-12-16
Places Visited: Kuta, Nuda Penida Island, Ubud,
Total Cost: 150000
Trip Type: leisure

Wonderful experience with lovely people. Almost like India with friendly nature. One will enjoy beautiful mountains, beaches.

- Database records for travel memory entries


```
_id: ObjectId('693d9481140a7a88ede42c22')
tripName : "Malaysia"
startDateOfJourney : "2024-12-23"
endDateOfJourney : "2024-12-25"
nameOfHotels : "Upper View Regalia Hotel"
placesVisited : "Petronas Towers, KL Tower, Genting Island, Little India"
totalCost : 60000
tripType : "leisure"
experience : "wonderful place, beautiful constructions, genting island is a must vis…"
image : "https://upload.wikimedia.org/wikipedia/commons/thumb/f/f6/The_Twins_SE…"
shortDescription : "Very well developed city"
featured : false
createdAt : 2025-12-13T16:26:02.537+00:00
__v : 0
```

```
_id: ObjectId('693d9a79140a7a88ede42c3c')
tripName : "Bali"
startDateOfJourney : "2024-12-14"
endDateOfJourney : "2024-12-16"
nameOfHotels : "Swiss Beleexpress Kuta Legian"
placesVisited : "Kuta, Nuda Penida Island, Ubud, "
totalCost : 150000
tripType : "leisure"
experience : "Wonderful experience with lovely people. Almost like India with friend…"
image : "https://static.wixstatic.com/media/063168_08ee8bed4cc74ecc9a2bb20266be…"
shortDescription : "It is a great place, Food is really good. Very close to Indian traditi…"
featured : false
createdAt : 2025-12-13T16:26:02.537+00:00
__v : 0
```


==========End of the Document==========