

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

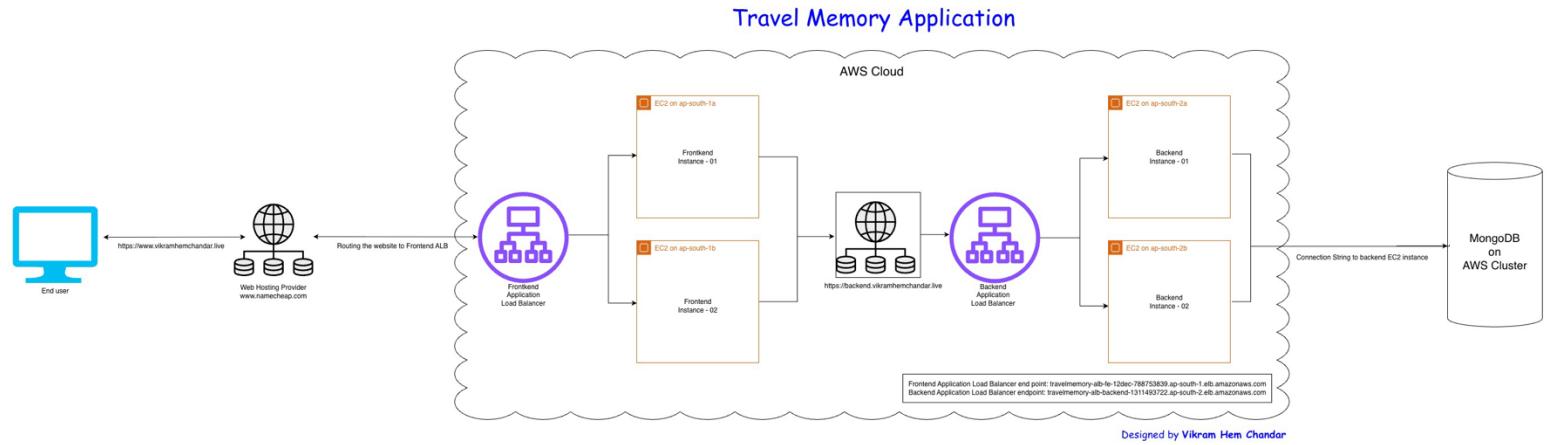
Requirements:

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 provider (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 / least 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

The screenshot shows the AWS CloudFormation console with the 'Launch an instance' wizard open. The steps are as follows:

- Name and tags**: A text input field contains 'TravelMemory_Backend_01'. There is also an 'Add additional tags' button.
- Application and OS Images (Amazon Machine Image)**: A search bar shows 'Search our full catalog including 1000s of application and OS images'. Below it, a grid of OS icons includes Amazon Linux, macOS, Ubuntu, Windows, Red Hat, SUSE Linux, and Debian. A 'Quick Start' tab is selected. A detailed view of the 'Ubuntu Server 24.04 LTS (HVM), SSD Volume Type' AMI is shown, including its ID (ami-02b269d5e85954ef), publish date (2025-10-22), and a 'Verified provider' badge.
- Instance type**: A dropdown menu shows 't3.micro' selected. It lists memory options (2 vCPU - 1 GiB Memory) and price details (On-Demand Linux base pricing: 0.0112 USD per Hour). A note says 'Additional costs apply for AMIs with pre-installed software'.
- Key pair (login)**: A dropdown menu shows 'vikramhemchandar_ubuntu' selected. A 'Create new key pair' button is available.
- Network settings**:
 - VPC - required**: A dropdown menu shows 'vpc-0203585a3c42f599 [TravelMemory-vpc]'. A note says 'The selected AMI contains instance store volumes, however the instance does not allow any instance store volumes. None of the instance store volumes from the AMI will be accessible from the instance.'
 - Subnet**: A dropdown menu shows 'subnet-0240464f19332666'. A note says 'Zone type: Availability Zone IP Addresses available: 4095 CDR: 10.0.0.0/24'.
 - Auto-assign public IP**: A dropdown menu shows 'Enable'.
 - Firewall (security group)**: A note says 'A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.' Options include 'Create security group' (radio button) and 'Select existing security group' (radio button, selected).
 - Common security groups**: A dropdown menu shows 'Select security group'.
 - Configure storage**: A dropdown menu shows '8 GiB gp3'. A note says 'Root volume, 3000 IOPS, Not encrypted'. An 'Advanced' link is visible.
 - Add new volume**: A note says 'The selected AMI contains instance store volumes, however the instance does not allow any instance store volumes. None of the instance store volumes from the AMI will be accessible from the instance.'
 - Click here to view backup information**: A note says 'The tags that you assign determine whether the instance will be backed up by any Data Lifecycle Manager policies.'
 - 0 x File systems**: An 'Edit' button is present.
 - Advanced details**: An 'Info' link is present.

At the bottom right, there are 'Cancel', 'Launch instance' (highlighted in orange), and 'Preview code' buttons.

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:xxxxxxxxx@vikramhemchandar.rtgw4
pn.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

The screenshot shows the AWS EC2 'Create image' configuration page. The left sidebar navigation includes 'EC2' (selected), 'Dashboard', 'EC2 Global View', 'Events', 'Instances' (with sub-options like 'Instances', 'Instance Types', 'Launch Templates', etc.), 'Images' (with sub-options like 'AMIs', 'AMI Catalog'), 'Elastic Block Store' (with sub-options like 'Volumes', 'Snapshots', 'Lifecycle Manager'), and 'Network & Security' (with sub-options like 'Security Groups', 'Elastic IPs', 'Placement Groups', 'Key Pairs'). The main content area is titled 'Create image' with a 'Info' link. It displays the following details:

- Image details:**
 - Instance ID:** i-000c465d820571542 (TravelMemory-Backend-ALB-11Dec-03)
 - Image name:** TravelMemory (Maximum 127 characters. Can't be modified after creation.)
 - Image description - optional:** Image for Travel Memory which will be used for both FRONTEND and BACKEND services (Maximum 255 characters).
 - Reboot instance:** A checked checkbox with the note: "When selected, Amazon EC2 reboots the instance so that data is at rest when snapshots of the attached volumes are taken. This ensures data consistency."
- Instance volumes:** A table showing one volume configuration:

Storage type	Device	Snapshot	Size	Volume type	IOPS	Throughput	Delete on termination	Encrypted
EBS	/dev...	Create new snapshot f...	8	EBS General Purpose S...	3000		<input checked="" type="checkbox"/> Enable	<input type="checkbox"/> Enable

An 'Add volume' button is available.
- Tags - optional:** A note: "During the image creation process, Amazon EC2 creates a snapshot of each of the above volumes." Two radio button options are shown:
 - Tag image and snapshots together: "Tag the image and the snapshots with the same tag."
 - Tag image and snapshots separately: "Tag the image and the snapshots with different tags."
- Tags:** A note: "No tags associated with the resource." An 'Add new tag' button is available.
- Buttons:** 'Cancel' and 'Create image' (highlighted in orange).

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

The screenshot shows the AWS EC2 'Launch an instance' wizard. The top navigation bar includes 'Console Home', 'EC2', 'SS', 'IAM', 'VPC', and 'Lambda'. The account ID is 4476-5762-3308, and the region is Asia Pacific (Hyderabad). The main page has a message: 'It seems like you may be new to launching Instances in EC2. Take a walkthrough to learn about EC2, how to launch instances and about best practices'. Buttons for 'Take a walkthrough' and 'Do not show me this message again' are present.

Name and tags: Name is set to 'TravelMemory_Backend_02'. There is a link to 'Add additional tags'.

Application and OS Images (Amazon Machine Image): A search bar shows 'Search our full catalog including 1000s of application and OS images'. Filter tabs include 'Recent', 'My AMIs' (selected), and 'Quick Start'. A button 'Owned by me' is selected. A link 'Shared with me' is also present. A 'Browse more AMIs' button is available.

Amazon Machine Image (AMI): The selected AMI is 'TravelMemory_Backend_04' (ami-0d57fa17a473873de). Details: Virtualization: hvm, ENA enabled: true, Root device type: ebs, Boot mode: uefi-preferred. A dropdown menu for 'Description' shows 'AMI images for backend server 04'.

Architecture: x86_64

Instance type: t3.micro

Key pair (login): hyd_ubuntu_vikramhemchandar

Network settings: VPC - required: vpc-07b826132f2e34ee6 (TravelMemory_Backend_VPC). Subnet: subnet-0cb492a29bc52780. Auto-assign public IP: Enable. Firewall (security groups): Create new security group (radio button selected).

Summary: Number of instances: 1. Software Image (AMI): AMI images for backend server ...read more. ami-0d57fa17a473873de. Virtual server type (instance type): t3.micro. Firewall (security group): SSH, HTTP, HTTPS. Storage (volumes): 1 volume(s) - 8 GiB. Firewall (security group): SSH, HTTP, HTTPS. Storage (volumes): 1 volume(s) - 8 GiB. Firewall (security group): SSH, HTTP, HTTPS. Storage (volumes): 1 volume(s) - 8 GiB. Firewall (security group): SSH, HTTP, HTTPS. Storage (volumes): 1 volume(s) - 8 GiB. Firewall (security group): SSH, HTTP, HTTPS. Storage (volumes): 1 volume(s) - 8 GiB. Firewall (security group): SSH, HTTP, HTTPS. Storage (volumes): 1 volume(s) - 8 GiB. Firewall (security group): SSH, HTTP, HTTPS. Storage (volumes): 1 volume(s) - 8 GiB. Firewall (security group): SSH, HTTP, HTTPS.

Launch instance button is visible at the bottom right.

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

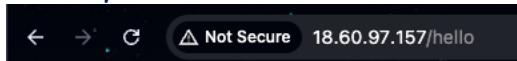
To verify the back server, copy the public IP of two instances and paste it in any browser. And note, port number is not required as we have configured reverse proxy

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

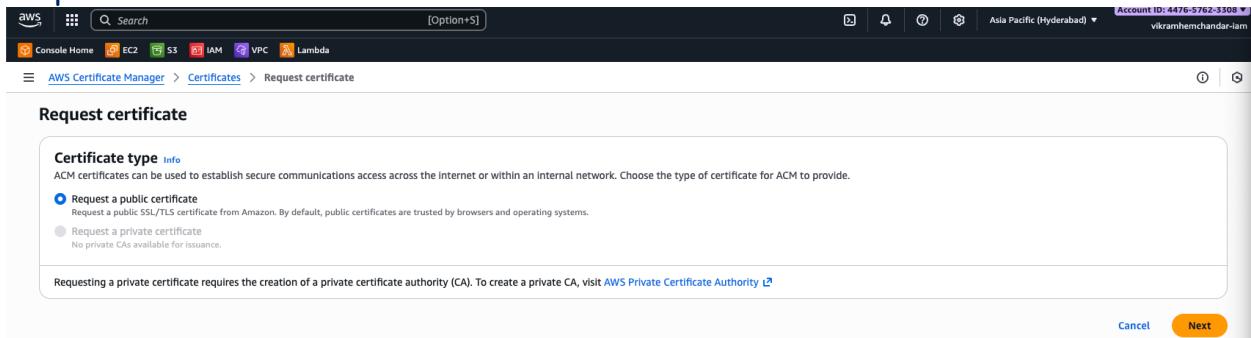


Hello World! - 02

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

The screenshot shows the 'Request public certificate' page in the AWS Certificate Manager. At the top, there's a navigation bar with links for AWS Home, EC2, S3, IAM, VPC, and Lambda. The main title is 'Request public certificate'. Below it, there are several sections:

- Domain names**: A section for entering domain names. It includes a note: "Provide one or more domain names for your certificate." A "Fully qualified domain name" input field contains "backend.vikramhemchandar.live". A link "Add another name to this certificate" is present, with a note: "You can add additional names to this certificate. For example, if you're requesting a certificate for 'www.example.com', you might want to add the name 'example.com' so that customers can reach your site by either name."
- Allow export**: A section with two options: "Disable export" (selected) and "Enable export". The "Disable export" option is described as "Use this certificate only with integrated AWS services. The private key for this certificate will be disallowed for exporting from AWS." The "Enable export" option is described as "Export this certificate and private key for use with any TLS workflow. ACM will charge your account based on the requested domains when the certificate is issued for the first time and for each renewal."
- Validation method**: A section with two options: "DNS validation - recommended" (selected) and "Email validation". The "DNS validation" option is described as "Choose this option if you are authorized to modify the DNS configuration for the domains in your certificate request." The "Email validation" option is described as "Choose this option if you do not have permission or cannot obtain permission to modify the DNS configuration for the domains in your certificate request."
- Key algorithm**: A section with three options: "RSA 2048" (selected), "ECDSA P 256", and "ECDSA P 384". The "RSA 2048" option is described as "RSA is the most widely used key type." The "ECDSA P 256" option is described as "Equivalent in cryptographic strength to RSA 3072." The "ECDSA P 384" option is described as "Equivalent in cryptographic strength to RSA 7680."
- Tags**: A section with a note: "No tags associated with the resource." A link "Add new tag" is available, with a note: "You can add up to 50 tags."

At the bottom right, there are three buttons: "Cancel", "Previous", and "Request".

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

Domain	Status	Renewal status	Type	CNAME name	CNAME value
backend.vikramhemchandar.live	Pending validation	-	CNAME	_751d28eeb7fd65c3692abf184a53884.backend.vikramhemchandar.live	_cbf299e03ab324b1fab706b60f767485.jkddzzsz validations.aws.

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

Type	Host	Value	TTL
CNAME Record	@	TravelMemory-ALB-FE-12Dec-788753839.ap-south-1.e...	Automatic
CNAME Record	_751d28eeb7fd...	_cbf299e03ab324b1fab706b60f767485.jkddzzszm.ac...	Automatic
CNAME Record	_9a1daed03a04...	_e5cba8bb79a788a99861c1716aef53.jkddzztszm.ac...	Automatic
CNAME Record	backend	TravelMemory-ALB-Backend-1311493722.ap-south-2.el...	Automatic
CNAME Record	www	vikramhemchandar.live.	Automatic
CNAME Record	_751d28eeb7fd...	_751d28eeb7fd65c3692abf184a53884.backend.vikramhemchandar.live	Automatic

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

Certificate status

Identifier	Status
b921127b-61d0-4df4-b3e8-40ea62904569	Issued

ARN: arn:aws:acm:ap-south-2:447657623308:certificate/b921127b-61d0-4df4-b3e8-40ea62904569

Type: Amazon Issued

Domains (1)

Domain	Status	Renewal status	Type	CNAME name	CNAME value
backend.vikramhemchandar.live	Success	-	CNAME	_751d28eeb7fdc5c5692abf184a53884.backend.vikramhemchandar.live	_cbf29e03ab324b1fab706b60f767485.kddztsz validations.aws.

Details

In use	Serial number	Requested at	Renewal eligibility
No	09:2c:7b:18:8fe2:dd:3d:5d:35:0d:63:de:73:96:ef	December 14, 2025, 00:41:41 (UTC+05:30)	Ineligible
Domain name	Public key info	Issued at	Export option
backend.vikramhemchandar.live	RSA 2048	December 14, 2025, 00:42:00 (UTC+05:30)	Disabled
Number of additional names	Signature algorithm	Not before	
0	SHA-256 with RSA	December 15, 2025, 05:30:00 (UTC+05:30)	
Can be used with	Not after		
CloudFront, Elastic Load Balancing, API Gateway and other integrated services.	January 12, 2027, 05:29:59 (UTC+05:30)		

Tags (0)

Key	Value
No tags	
No tags associated with this resource.	

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*

aws Search [Option+S]

Console Home EC2 S3 IAM VPC Lambda

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 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 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

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

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**

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)

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

Ports for the selected Instances
Ports for routing traffic to the selected instances.

1-65535 (separate multiple ports with commas)

Review targets

Targets (0)

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

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

Target group details

Name TravelMemory-TargetGroup-Backend	Target type Instance	Protocol : Port HTTP: 80	Protocol version HTTP1
VPC vpc-07b26132f2e34ee6	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

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

The screenshot shows the AWS EC2 Target Groups console. On the left, there's a navigation sidebar with links like 'Console Home', 'EC2', 'S3', 'IAM', 'VPC', and 'Lambda'. The main area is titled 'TravelMemory-TargetGroup-Backend' and displays the following details:

- Details:** Target type: Instance; Protocol: Port; Port: 80; Protocol version: HTTP1; VPC: vpc-07b826132f2e34ee6.
- Targets:** Total targets: 2. Status distribution: 0 Healthy, 0 Unhealthy, 2 Unused, 0 Initial, 0 Draining.
- Registered targets (2):**

Instance ID	Name	Port	Zone	Health status	Health status details
I-000c465d820571542	TravelMemory-Backend-ALB-11Dec-03	80	ap-south-2a (a...)	Unused	Target group is not co...
I-046cdab2fc2b3f4af	TravelMemory-Backend-ALB-11Dec-04	80	ap-south-2b (a...)	Unused	Target group is not co...

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

Create Application Load Balancer

How Application Load Balancers work

Basic configuration

Load balancer name

The load balancer name must be unique within your AWS account and can't be changed after the load balancer is created.

Name: TravelerAppLoadBalancer

A maximum of 10 alphanumeric characters including hyphens are allowed, but the name does not begin or end with a hyphen.

Scheme: **http**

Listeners will receive traffic from the Internet or from other services in your VPC.

Interest setting

Internal load balancing traffic:

- External IP address
- Private IP address
- Direct Connect endpoint IP
- Compliance with the IP and Endpoint IP address types

Internal

Internal load balancing traffic:

- External IP address
- Private IP address
- Direct Connect endpoint IP
- Compliance with the IP and Endpoint IP address types

Load balancer IP address type: **IPv4**

Load balancer IP addresses are assigned to the load balancer. They are used in the selected subnets, and in accordance with your IP address settings.

VPC: **1**

Load balancer will receive traffic from the selected VPC. The selected VPC is also where the target groups must be located unless using Lambda or an API gateway. If you want to route traffic to targets in another VPC, view [Create target group](#).

vpc-0f830312f32f4e4d (TravelerAppLoadBalancer, VPC) [Create VPC](#)

IP pools: **http**

You can choose to assign an IP pool and as the preferred source for your load balancer IP addresses. Create or view Pools in the [AWS VPC IP Address Manager console](#).

Use IPAM for public IPv4 addresses

Use IPAM for private IPv4 addresses

Availability zones and subnets

Select one or more availability zones and a subnet for each zone. A load balancer 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.

http-1 (http-1) (http-1)

Subnet: subnet-0f9b2a62e612780

project-public-ip-south-2a

http-2 (http-2) (http-2)

Subnet: subnet-0c00ca230150a944

project-public-ip-south-2b

Security groups

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 group

Select or create a security group:

500, HTTP, HTTPS

Listeners and routing

A listener is a port and protocol combination 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: HTTPS443

Protocol: **HTTPS** Port: **443** 1.68GB/s

Default action

Choose a default action if no other rules apply. Choose the default action for traffic on this listener.

Pre-existing action: **http**

No pre-routing action Authenticate user Amazon Cognito User Pool, Amazon Cognito Identity Validate token Amazon Cognito User Pool, Amazon Cognito Identity, AWS Lambda, AWS Lambda Function, AWS Lambda Layer, AWS Lambda Layer Version, AWS Lambda Layer Alias, AWS Lambda Layer Alias Version, AWS Lambda Layer Version Alias, AWS Lambda Layer Version Alias Version

Routing action

Forward to target group Redirect to URL Return fixed response

Forward to target group

Choose a target group for this routing weight or [create target group](#).

Target group: **TravelerAppLoadBalancer**

Target type: **HTTP** Target attributes: **0.0%**

Add target group You can add up to 10 target groups.

Target group metrics

Log metrics for each user's session to a specific target group. To use this feature, the client must support cookies. If you want to bind a user's session to a specific target, turn on the Target Group attribute **Cookie**.

Turn on target group metrics

Listener tags - optional

Tags make it easier to manage resources. Tags enable you to categorize your AWS resources so you can easily manage them.

[Add Listener tag](#)

[Add Listener](#)

Security listener settings

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

Security policy: **https**

This security policy defines a basic layer 7 (application) configuration that is required to establish a connection with clients. Compare security policies.

Security category: **All security policies** Policy name: **ELBSecurityPolicy-TLS1-2-Rev-PQ-2025 (Recommended)**

Default SSL/TLS server certificate

The certificate must be a valid SSL/TLS certificate, or if there are no matching certificates, you can select this certificate from AWS Certificate Manager (ACM), Amazon Identity and Access Management (IAM), or import a certificate. This certificate will automatically be applied to all listeners.

Certificate source

AWS Certificate Manager (ACM) From IAM Import certificate

Certificates from ACM

One or more certificates can be applied as the default SSL/TLS server certificate for the load balancer's secure listeners.

Select certificate [Create new certificate request](#) [Import certificate](#)

Client certificate handling

Client certificates are used by your listeners to verify connections to your listeners. Learn more [Learn more](#)

Mutual authentication (TLS) Only mutual authentication is supported for standard listeners. If you have mutual authentication off, then your listeners have been made more vulnerable to man-in-the-middle attacks.

Loudest 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, value = us-east-1, key = environment, and value = production.

Optimize with service integrations - optional

Optimize your load balancing by integrating AWS services with the load balancer of choice. You can also add these or other services after your load balancer is created by reviewing the load balancer's "Integrations" tab.

Amazon CloudFront + AWS Web Application Firewall (WAF) - new

Optimize performance, security, security

Apply application layer security protections - in front of targets For your load balancer to use this integration, you must have CloudFront and WAF enabled, and have CloudFront configured as a target for your load balancer. Additional charges apply.

Benefits and considerations

AWS Global Accelerator

Optimize performance, availability CloudFront is required to use this integration. If you do not need global delivery or traffic management across multiple regions, select Amazon CloudFront. Additional charges apply.

Benefits and considerations

Review

Review the changes to your configuration. Make changes if needed. After you finish reviewing the configuration, choose **Create load balancer**.

Summary

Review the changes to your configuration. Make changes if needed. After you finish reviewing the configuration, choose **Create load balancer**.

Basic configuration	Network mapping	Security groups	Listeners and routing
Name: TravelerAppLoadBalancer	VPC: vpc-0f830312f32f4e4d	500, HTTP, HTTPS	HTTPS443 (Forward to target group)
Health check interval: 1 second	Availability Zones and subnets:	HTTP-1 (http-1)	HTTP-1 (http-1)
Service health check: 500 OK	Subnets: subnet-0f9b2a62e612780	HTTP-2 (http-2)	HTTP-2 (http-2)
Series integrations:	Targets: project-public-ip-south-2a, project-public-ip-south-2b	Tags: null	
AWS CloudFront + AWS Web Application Firewall (WAF)			
AWS WAF			
AWS Global Accelerator			
Address:	<input checked="" type="checkbox"/> Service 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)

Type	Host	Value	TTL
CNAME Record	@	TravelMemory-ALB-FE-12Dec-788753839.ap-south-1.e...	Automatic
CNAME Record	_751d28eef7dc...	_cbf299e03ab324b1fab706b60f767485.jkddzztszm.ac...	Automatic
CNAME Record	_9a1daed03a04...	_e5cba88bb7f9a788a99861c1716aef53.jkddzztszm.ac...	Automatic
CNAME Record	backend	nory-ALB-Backend-1311493722.ap-south-2.elb.amazonaws.com	Automatic
CNAME Record	www	vikramhemchandar.live.	Automatic

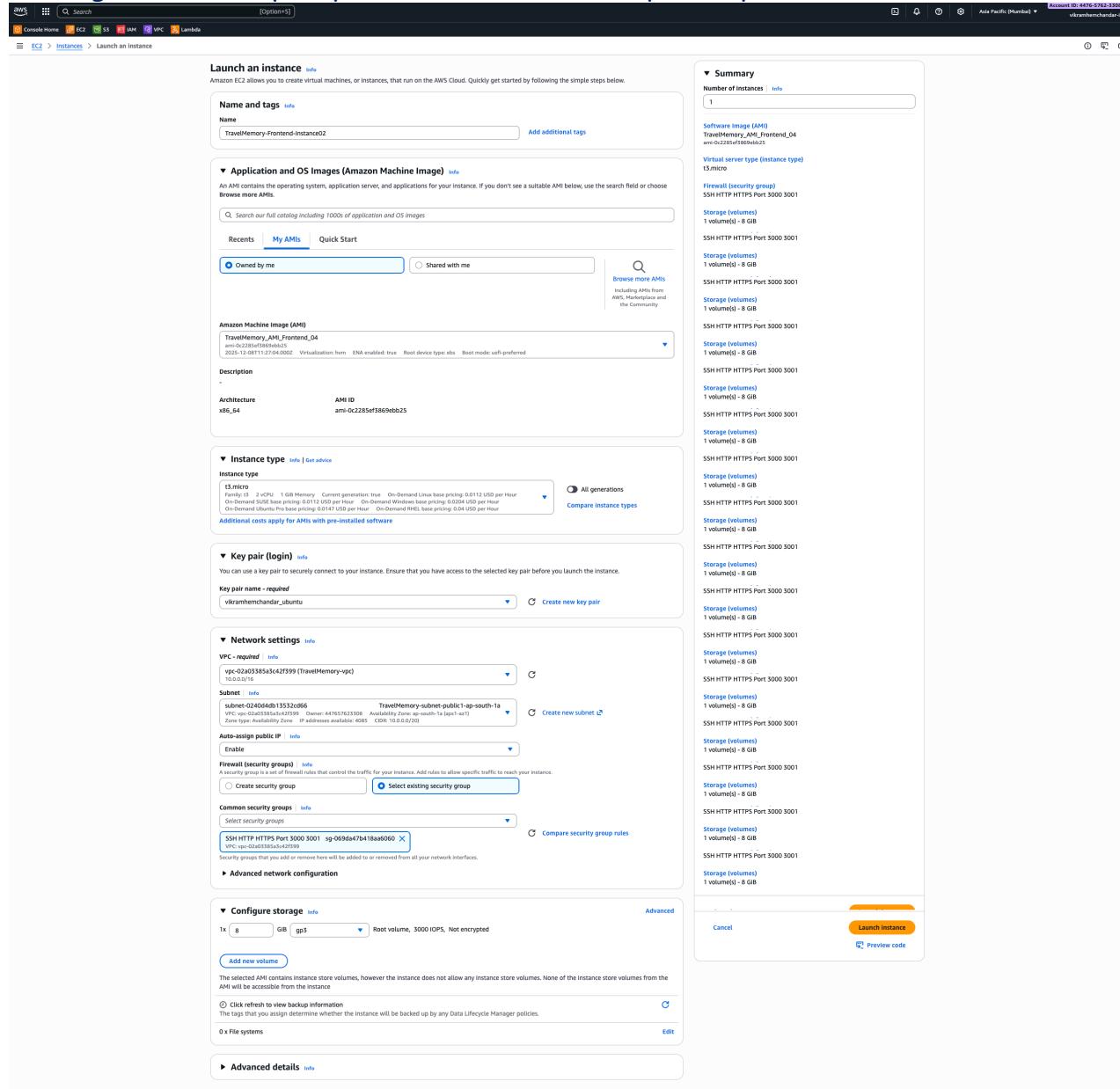
Now, backend should be accessible at <https://backend.vikramhemchandar.live>

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

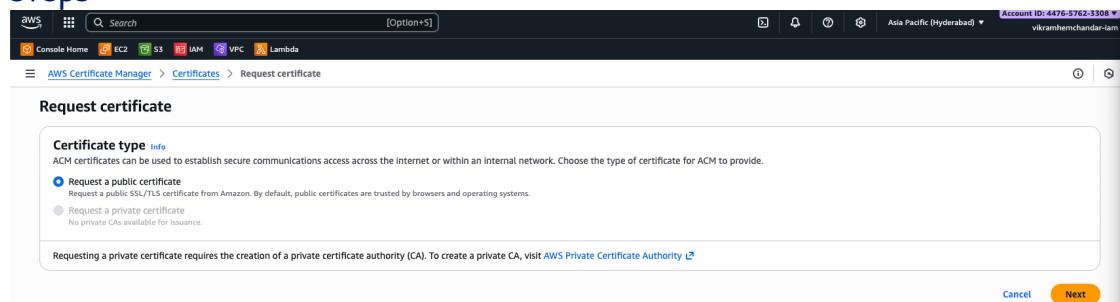
Test the individual frontend servers:

- 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 provider 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

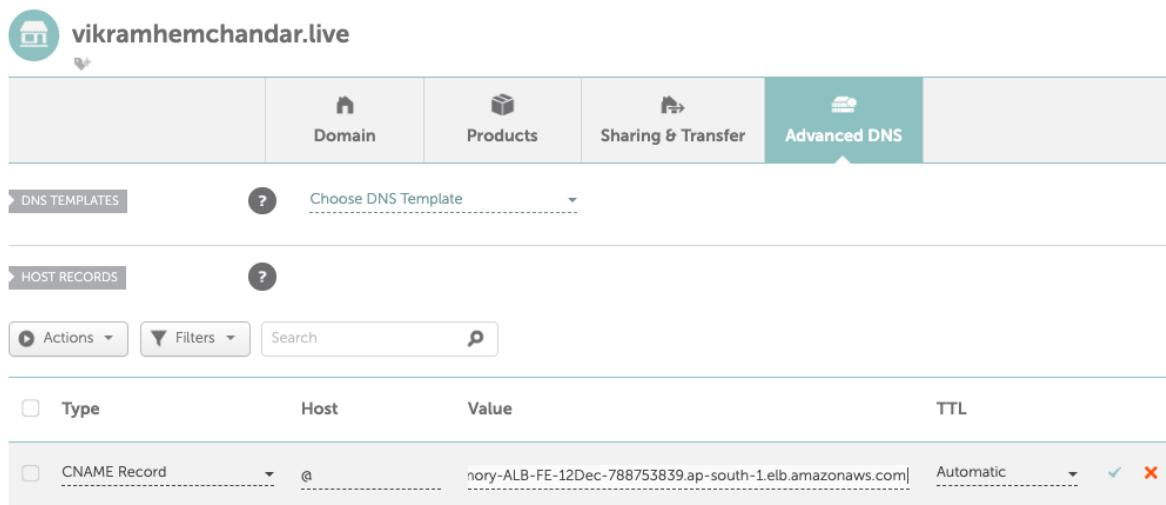
Note: wait for at least 2-3 minutes till it gets provisioning to Active status

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)



The screenshot shows a DNS management interface for a domain named "vikramhemchandar.live". The top navigation bar includes links for Domain, Products, Sharing & Transfer, and Advanced DNS (which is highlighted). Below the navigation, there are sections for DNS TEMPLATES and HOST RECORDS. In the HOST RECORDS section, there are buttons for Actions, Filters, and Search. A table lists a single CNAME record with the following details:

Type	Host	Value	TTL
CNAME Record	@	emory-ALB-FE-12Dec-788753839.ap-south-1.elb.amazonaws.com	Automatic

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

Type	Host	Value	TTL	Action
CNAME Record	@	TravelMemory-ALB-FE-12Dec-788753839.ap-south-1.e...	Automatic	
CNAME Record	_751d28eef7fdc...	_cbf299e03ab324b1fab706b60f767485.jkddzztszm.ac...	Automatic	
CNAME Record	_9a1daed03a04...	_e5cba88bb7f9a788a99861c1716aef53.jkddzztszm.ac...	Automatic	
CNAME Record	backend	TravelMemory-ALB-Backend-1311493722.ap-south-2.el...	Automatic	
CNAME Record	www	vikramhemchandar.live.	Automatic	

Application Testing:

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

The screenshot shows a web browser window with the URL [vikramhemchandar.live](https://www.vikramhemchandar.live). The page title is "Travel Memory Add Experience - 01". A card displays the following information:

- Location: Malaysia
- Type: leisure
- Description: Very well developed city

A blue "More Details" button is visible at the bottom of the card.

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

The screenshot shows a web browser window with the URL [vikramhemchandar.live](https://www.vikramhemchandar.live). The page title is "Travel Memory Add Experience - 01". The form fields are as follows:

- Trip Name: Bali
- Trip Date: 12/14/2024 - 12/16/2024
- Name of Hotels: Swiss Beleexpress Kuta Legian
- Trip Type: Leisure (dropdown)
- Total Cost: 150000 (dropdown)
- Places Visited: Kuta, Nuda Penida Island, Ubud, Kuta, Nuda Penida Island, Ubud, (text input)
- Featured Trip?:
 - True
 - False
- Image Link: https://static.wixstatic.com/media/063168_08ee8bed4cc74ecc9a2bb20266bef334~mv2.png/v1/fill/w_980,h_1307,al_c,q_90,usm_0.66_1.00_0.01,enc_avif,q_90
- Short Description: It is a great place, Food is really good. Very close to Indian tradition.
- Experience: Wonderful experience with lovely people. Almost like India with friendly nature. One will enjoy beautiful mountains, beaches.

A blue "Submit" button is located at the bottom of the form.

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

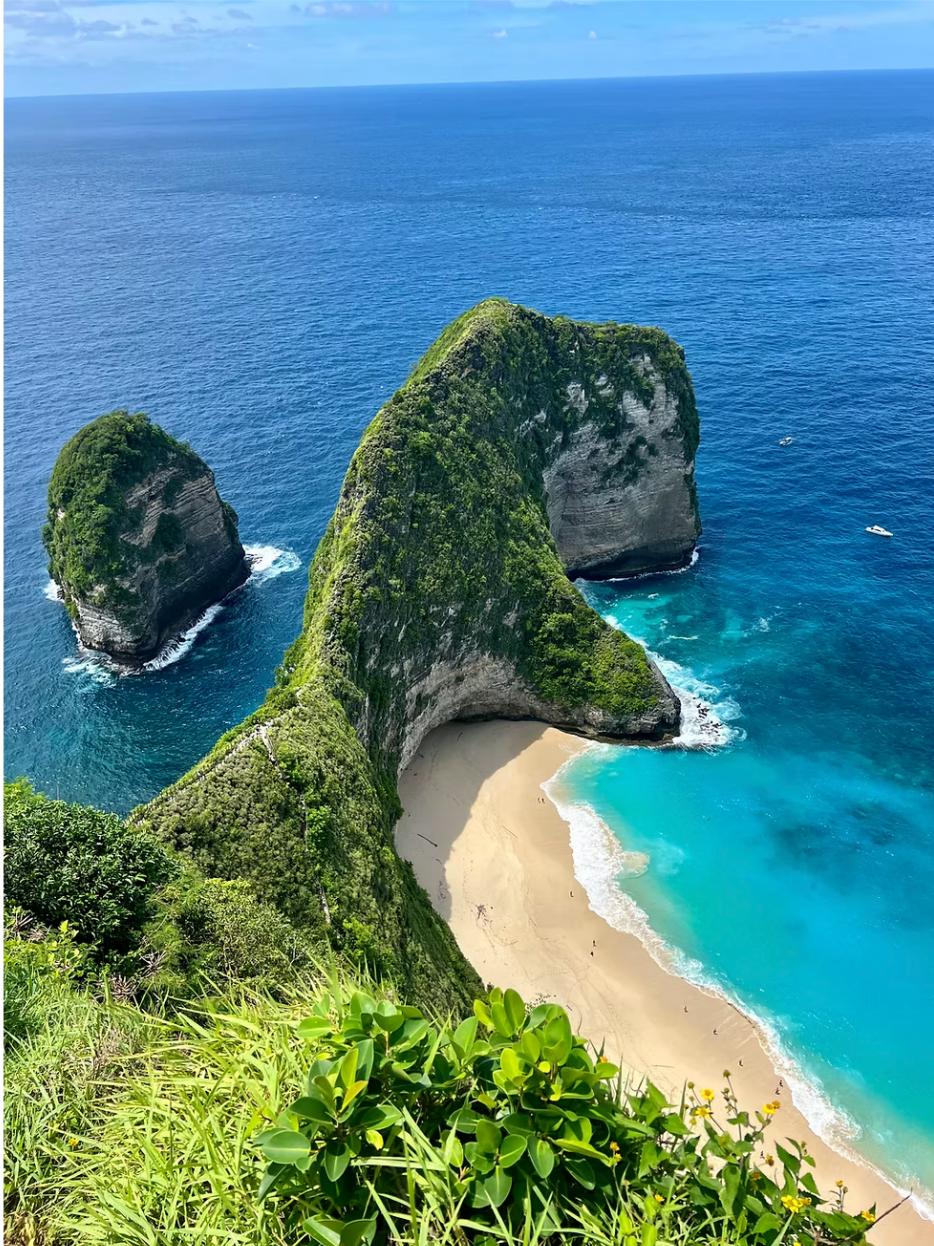
The screenshot shows a web browser window with the following details:

- Address Bar:** vikramhemchandar.live
- Tab:** Vikram Hem Chandar - AWS Mern Deployment - React App
- Search Bar:** Google
- Content Area:** Travel Memory Add Experience - 01
- Card 1 (Malaysia):**
 - Name:** Malaysia
 - Type:** leisure
 - Description:** Very well developed city
 - More Details button:** A blue button at the bottom of the card.
- Card 2 (Bali):**
 - Name:** Bali
 - Type:** leisure
 - Description:** It is a great place, Food is really good. Very close to Indian tradition.
 - More Details button:** A blue button at the bottom of the card.

- Click on more details to see your travel memory details

Travel Memory Add Experience - 01

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=====