# TravelMemory Application Deployment Documentation

## 1. Introduction

This document provides a step-by-step guide to deploying the **TravelMemory** application on an **AWS EC2 instance** using the **MERN stack** (MongoDB, Express, React, Node.js). The deployment includes:

- Setting up the **backend** (Node.js).
- Configuring the **frontend** (React).
- Ensuring proper communication between the frontend and backend.
- Setting up **load balancing** for high availability.
- Connecting the custom domain manjyyot.online through **Cloudflare**.

## 2. Prerequisites

Ensure that the following tools and resources are available:

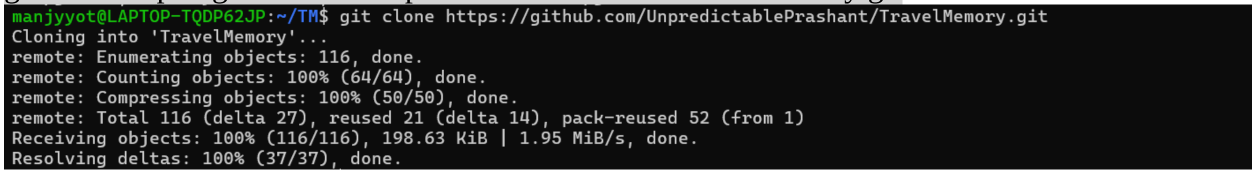
- **AWS Account** (for EC2, Load Balancer, etc.)
- **Cloudflare Account** (for domain management).
- **Git** (to clone the repository).
- **Nginx** (for reverse proxy).
- **Node.js & NPM** (for the backend).
- **MongoDB Atlas** (or local MongoDB setup).
- **Custom Domain (manjyyot.online)** registered and linked with Cloudflare.

## 3. Backend Configuration

### Step 1: Clone the Repository

Clone the TravelMemory repository from GitHub:

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



```
manjyyot@LAPTOP-TQDP62JP:~/TM$ git clone https://github.com/UnpredictablePrashant/TravelMemory.git
Cloning into 'TravelMemory'...
remote: Enumerating objects: 116, done.
remote: Counting objects: 100% (64/64), done.
remote: Compressing objects: 100% (50/50), done.
remote: Total 116 (delta 27), reused 21 (delta 14), pack-reused 52 (from 1)
Receiving objects: 100% (116/116), 198.63 KiB | 1.95 MiB/s, done.
Resolving deltas: 100% (37/37), done.
```

Navigate to the backend directory:

```
cd Travelmemory/backend
```

### Step 2: Install Dependencies

Install the required dependencies:

```
npm install
```

**Step 3: Set up Environment Variables**

Create a .env file in the root of the backend directory with the following content:

```
nano .env
```

Contents for the .ev file:

```
PORT=3000
MONGO_URI='mongodb+srv://manjyotsinghchaudhary:Xs7AnsL6bDfhirvy@travelbackend.qy
czv.mongodb.net/'
```

This connects the backend to MongoDB and sets the backend to run on port 3000.

**Step 4: Set Up Nginx Reverse Proxy**

Install Nginx on the EC2 instance:

```
sudo apt update
sudo apt install nginx
```

Create an Nginx configuration file for the backend:

```
sudo nano /etc/nginx/sites-available/travelmemory
```

Add the following configuration to proxy requests to the backend:

Nginx file content:

```
server {
    listen 80;

    server_name manjyyot.online;
{
        proxy_pass http://34.236.65.214:3000;  # Backend running on port 3000 on the public ip4
address of the EC2 instance
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

Restart Nginx:

```
sudo systemctl restart nginx
```

**4. Frontend and Backend Connection**

**Step 1: Update urls.js**

Navigate to the frontend directory:

```
cd frontend/src
```

Update the urls.js file to ensure the frontend communicates with the backend (EC2 public IP or load balancer URL):

```
const baseUrl = "http:// 34.236.65.214:3000"; // Replace with the correct IP or load balancer endpoint
```

**5. Scaling the Application**

**Step 1: Launch EC2 Instances**

Launch at least two EC2 instances: one for the frontend and one for the backend.

**Step 2: Set Up Load Balancer**

- Go to **EC2 > Load Balancers** in the AWS Management Console.
- Create an **Application Load Balancer (ALB)**.
- Add both EC2 instances (frontend and backend) to the target group of the load balancer.

**Step 3: Configure Auto Scaling**

- Set up an **Auto Scaling Group** for both frontend and backend instances to automatically scale based on traffic.

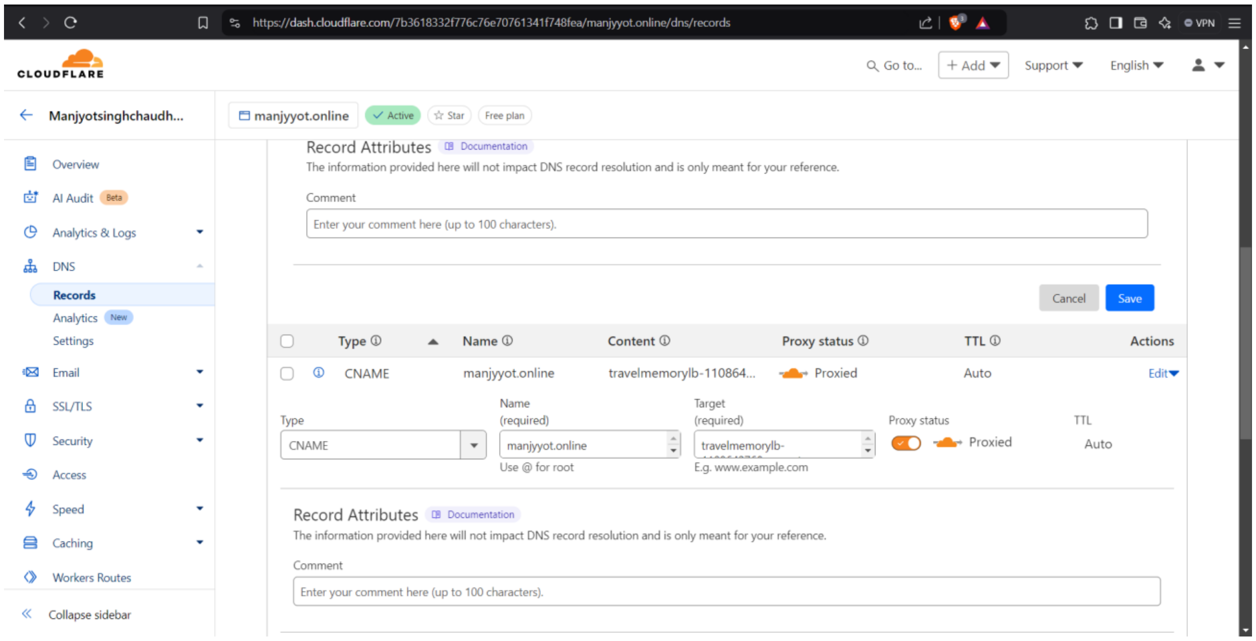
**6. Domain Setup with Cloudflare**

**Step 1: Add Your Domain to Cloudflare**

- Log in to Cloudflare.
- Add your domain (manjyyot.online) to Cloudflare if not already done.

**Step 2: Configure DNS Records**

**CNAME Record** for the load balancer:


https://dash.cloudflare.com/7b3618332f776c76e70761341f748fea/manjyyot.online/dns/records

CLOUDFLARE

Go to... | + Add | Support | English

Manjyotsinghchaudh... | manjyyot.online | Active | Star | Free plan

Overview
AI Audit Beta
Analytics & Logs
DNS
Records
Analytics New
Settings
Email
SSL/TLS
Security
Access
Speed
Caching
Workers Routes
Collapse sidebar

Record Attributes Documentation

The information provided here will not impact DNS record resolution and is only meant for your reference.

Comment

Enter your comment here (up to 100 characters).

Cancel | Save

| | Type | Name | Content | Proxy status | TTL | Actions |
|---|---|---|---|---|---|---|
| | CNAME | manjyyot.online | travelmemorylb-110864... | Proxied | Auto | Edit |

Type: CNAME
Name (required): manjyyot.online — Use @ for root
Target (required): travelmemorylb- — E.g. www.example.com
Proxy status: Proxied
TTL: Auto

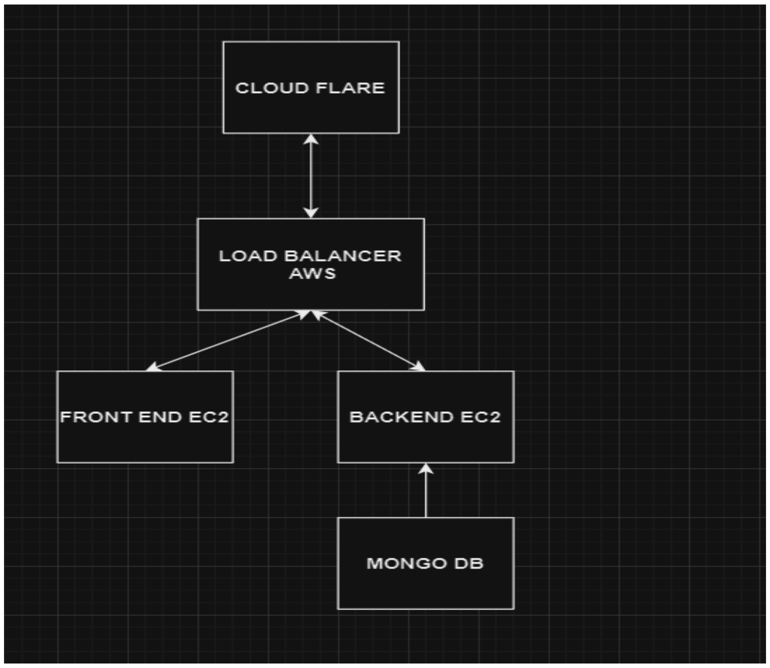
Record Attributes Documentation

The information provided here will not impact DNS record resolution and is only meant for your reference.

Comment

Enter your comment here (up to 100 characters).

## 7. Deployment Architecture Diagram



## 8. Conclusion

In this guide, we have successfully deployed the **TravelMemory** full-stack application on **AWS EC2**, set up a **reverse proxy** with **Nginx**, scaled the application using a **load balancer**, and connected the custom domain manjyyot.online via **Cloudflare**. These steps ensure high availability, security, and scalability of the application.