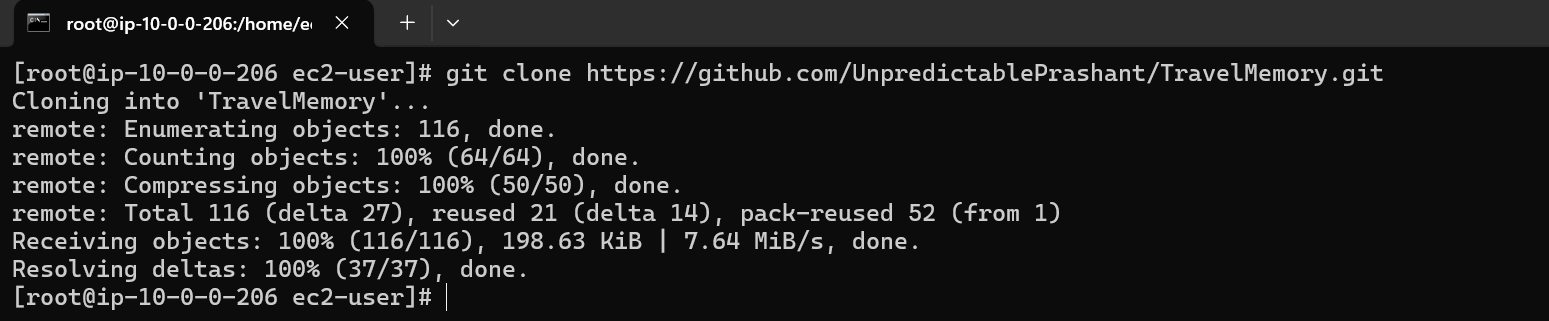
# Travel Memory Application Deployment

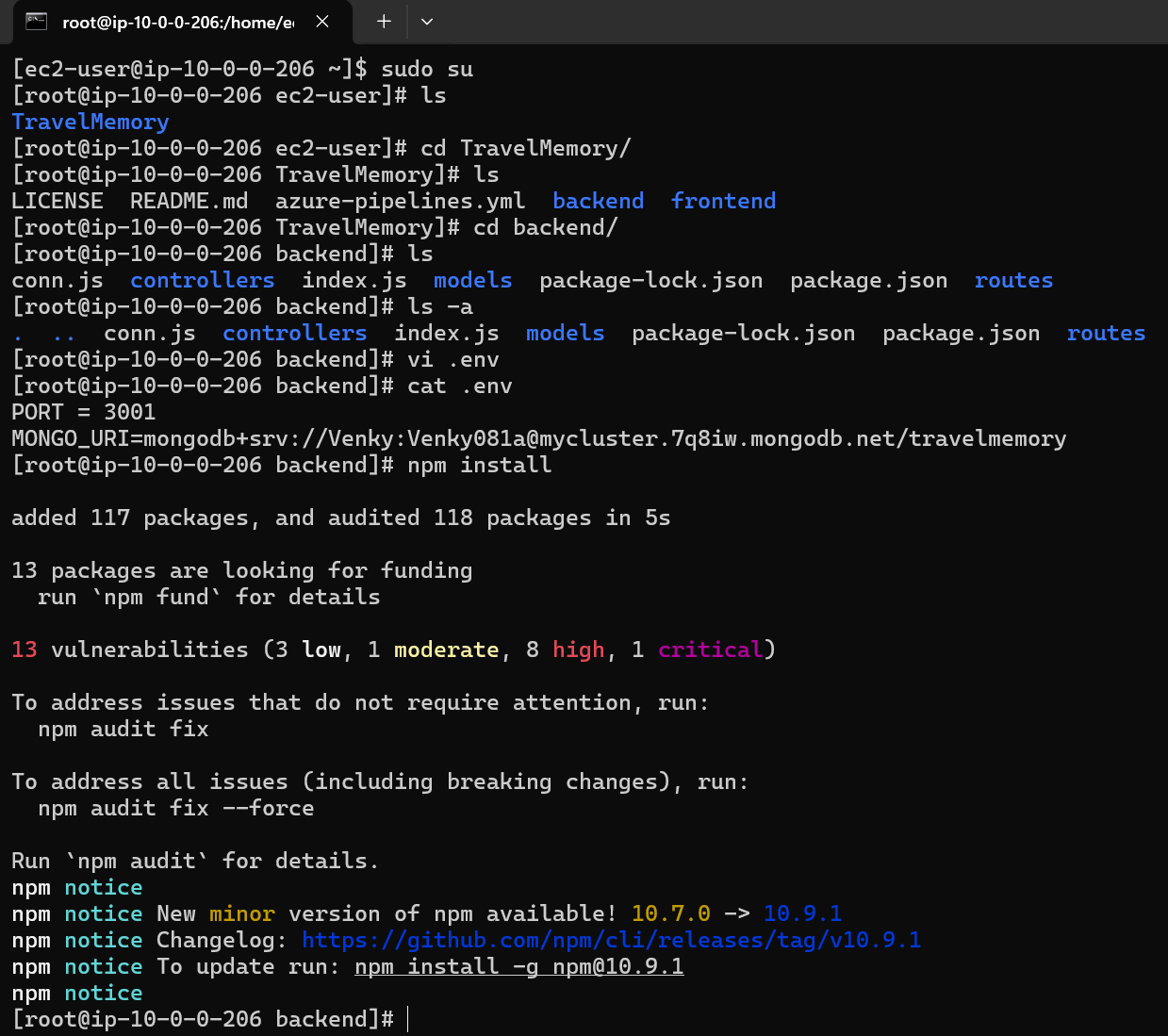
Clone the application from git - https://github.com/UnpredictablePrashant/TravelMemory

**Create EC2 instances** for both the frontend and backend:

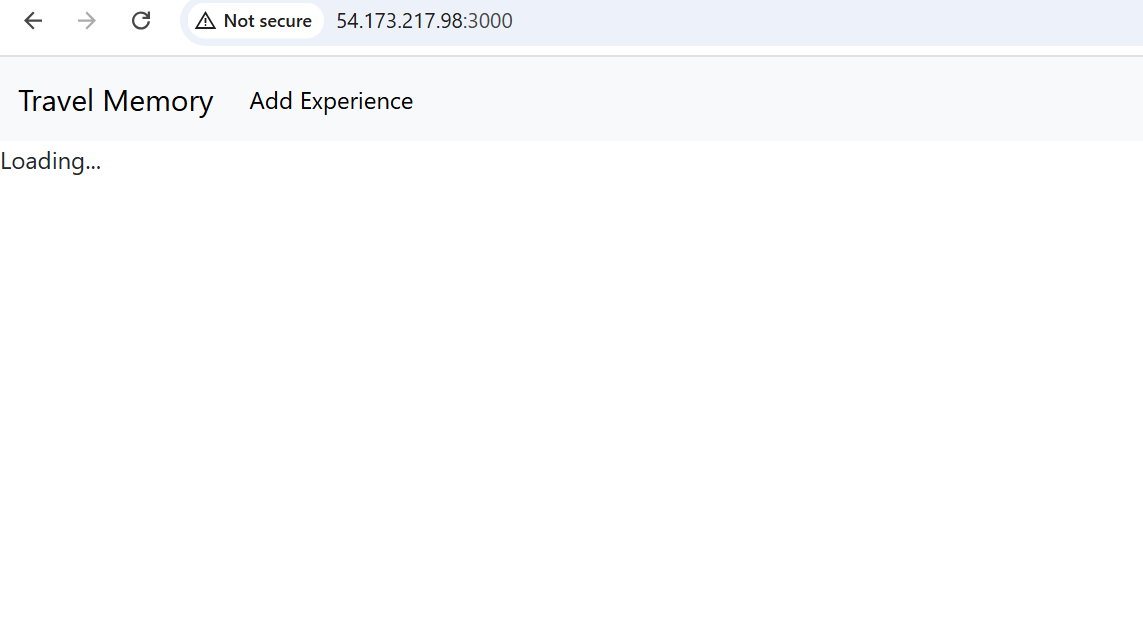
* Click "Launch Instance".
* Choose an appropriate Amazon Machine Image (AMI) like Amazon Linux, Ubuntu, or another, depending on your technology stack.
* Select the **instance type** (e.g., t2.micro depending on your traffic needs).
* Set up **Security Groups**:
  + Ensure HTTP (port 80) and HTTPS (port 443) are open for the frontend servers.
  + Allow necessary ports for backend communication (e.g., port 8080 for APIs or another port as needed).
  + Ensure the frontend and backend security groups can communicate if needed.
  + Enable port 3001 as we are running our backend application on this port

Connect to Ec2 Instance using Ec2 connect and try to install git and node package manager as we need to clone the application inside Ec2 and npm is required to run Node js and React js applications.









**Create Auto Scaling Groups (ASG) for Both Frontend and Backend Servers**

Now that we have the Launch Templates ready, we can create Auto Scaling Groups to automatically manage the scaling of instances for both the frontend and backend.

**Step 2: Create Auto Scaling Group for Frontend**

1. **Go to EC2 Dashboard** > **Auto Scaling Groups**.
2. Click on **Create Auto Scaling group**.
   * **Choose Launch Template**:
     + Select the frontend-template (the template you created earlier).
   * **Specify Group Details**:
     + **Name**: Enter a name for the Auto Scaling Group (e.g., frontend-asg).
     + **VPC**: Select the VPC where your frontend servers will reside.
     + **Subnets**: Choose the subnets where the frontend instances will be launched (make sure to select multiple subnets in different availability zones for high availability).
   * **Scaling Policies**:
     + Set the desired number of instances (e.g., 2 instances).
     + Configure the **minimum** and **maximum** instance count based on the expected load (e.g., min: 1, max: 4).
     + Set up **Scaling Policies** based on CloudWatch metrics like CPU utilization (e.g., scale up if CPU > 70%, scale down if CPU < 30%).
     + Optionally, configure **health checks** (e.g., ELB health check).
3. **Create Auto Scaling Group**.

**Step 3: Create Auto Scaling Group for Backend**

1. Follow the same steps as for the frontend Auto Scaling Group, but use the **backend Launch Template**.
2. **Name** the Auto Scaling Group (e.g., backend-asg).
3. Choose the **VPC** and **Subnets** where backend instances will run.
4. Set the **desired, minimum, and maximum instances** for backend servers, similar to frontend scaling.
5. Configure **Scaling Policies** based on CPU usage or other application-specific metrics (e.g., scale up when backend traffic exceeds a certain threshold).
6. Create the Auto Scaling Group.

**Set Up a Load Balancer for Efficient Traffic Distribution**

**Step 2: Create an Application Load Balancer (ALB)**

1. **Go to the EC2 dashboard** and navigate to **Load Balancers** under the "Load Balancing" section.
2. **Click on "Create Load Balancer"** and select **Application Load Balancer**.
3. **Configure the Load Balancer**:
   * **Name**: Choose a name for the load balancer (e.g., my-app-alb).
   * **Scheme**: Select "internet-facing" for public access or "internal" if you want it private.
   * **IP address type**: Choose "ipv4" unless you specifically need IPv6.
   * **Listeners**: Ensure the load balancer is configured to listen on HTTP (port 80) and HTTPS (port 443) if required.
   * **Availability Zones**: Choose the relevant availability zones where your EC2 instances are deployed.
4. **Configure Security Settings** (Optional):
   * If using HTTPS, you'll need an SSL certificate (you can use AWS Certificate Manager for this).
5. **Configure Security Groups**: Ensure that the load balancer's security group allows inbound traffic on the required ports (e.g., 80/443).

**Step 3: Set Up Target Groups**

1. **Create Target Groups** for both frontend and backend servers:
   * **Frontend Target Group**:
     + Choose the target type as **instance**.
     + Choose HTTP protocol and port (e.g., port 80 or 443).
     + Register the **frontend EC2 instances** in this target group.
   * **Backend Target Group**:
     + Choose the target type as **instance**.
     + Choose the appropriate protocol and port (e.g., HTTP on port 8080).
     + Register the **backend EC2 instances** in this target group.
2. **Configure health checks** for each target group to ensure traffic is routed only to healthy instances. Use default or custom health check settings depending on your application (e.g., /health endpoint).

**Step 4: Add Listeners and Routing Rules**

1. **Configure listeners** on the load balancer:
   * Add a listener on port 80 (HTTP) and 443 (HTTPS) if necessary.
   * Set up **routing rules** to forward requests to the appropriate target groups:
     + Example: Requests for /api/\* can be routed to the backend target group, and all other requests can be routed to the frontend target group.
2. **Ensure SSL termination** for HTTPS traffic:
   * If using HTTPS, ensure SSL certificates are configured, and SSL termination happens at the load balancer.

**Add Your Domain to Cloudflare**

1. **Log into Cloudflare**:
   * Go to the [Cloudflare dashboard](https://www.cloudflare.com/).
   * If you haven't added your domain to Cloudflare yet, click on **Add Site** and follow the prompts to add your domain.
   * Cloudflare will automatically scan your existing DNS records. If you're transferring DNS management to Cloudflare, update the **nameservers** at your domain registrar to point to the Cloudflare nameservers.

**Create a CNAME Record for the Load Balancer**

1. **Go to the DNS Settings** in your Cloudflare account:
   * After your domain is added, navigate to the **DNS** tab in your Cloudflare dashboard.
2. **Create a CNAME record** pointing to the AWS Application Load Balancer (ALB):
   * **Type**: CNAME
   * **Name**: Enter the subdomain you want to point to your Load Balancer (e.g., www, app, or frontend if you are using a subdomain).
   * **Target**: Use the **DNS name** of your Application Load Balancer (e.g., my-app-alb-1234567890.us-west-2.elb.amazonaws.com).
   * **TTL**: Choose Auto or set a specific TTL value.
   * **Proxy status**: Select **Proxied** (orange cloud) for Cloudflare to handle your traffic with its CDN, security features, and caching.

Example:

* + **Name**: www
  + **Target**: my-app-alb-1234567890.us-west-2.elb.amazonaws.com

This will route traffic from www.example.com to the AWS Load Balancer.

1. **Save the CNAME record**.

**3. Set Up an A Record for the EC2 Instance**

If you need to directly connect to an EC2 instance (e.g., for the frontend) using its IP address (for a root domain or subdomain), follow these steps to set up an **A record** in Cloudflare.

**Step 1: Find the Public IP of the EC2 Instance**

1. **Go to EC2 Dashboard** in AWS.
2. Under **Instances**, locate the **frontend EC2 instance**.
3. In the **Details** section of the instance, find the **Public IPv4 address** (this is the IP address of the EC2 instance).

**Step 2: Create an A Record in Cloudflare**

1. In the Cloudflare dashboard, go to the **DNS** settings of your domain.
2. **Create an A record** for your frontend EC2 instance:
   * **Type**: A
   * **Name**: Enter the subdomain or root domain you want to point to your EC2 instance (e.g., example.com or frontend.example.com).
   * **IPv4 address**: Enter the **Public IP address** of your EC2 instance (e.g., 203.0.113.10).
   * **TTL**: Choose Auto or set a specific TTL value.
   * **Proxy status**: Select **Proxied** (orange cloud) if you want Cloudflare's CDN and security features or **DNS Only** (gray cloud) if you want to bypass Cloudflare’s proxy features and directly connect to the EC2 instance.

Example:

* + **Name**: frontend
  + **IPv4 address**: 203.0.113.10 (replace this with your actual EC2 instance IP address)

1. **Save the A record**.

**4. Verify the DNS Records**

After configuring the DNS records:

1. **DNS Propagation**: It may take a few minutes for the DNS changes to propagate across the Cloudflare network.
2. **Test the Setup**:
   * **For the ALB (CNAME)**: Open a browser and visit www.example.com or the subdomain you configured. It should direct you to your AWS Application Load Balancer, distributing traffic to your frontend EC2 instances.
   * **For the EC2 instance (A record)**: Visit frontend.example.com (or the domain/subdomain you configured) and ensure it points to your EC2 instance.