Assignment Title:

End-to-End MERN Application Deployment and Monitoring with Terraform, Ansible, Prometheus, and Grafana

Objective:

Build and monitor a production-grade MERN stack application using Infrastructure as Code (Terraform), Configuration Management (Ansible), and Observability tools (Prometheus, Grafana). You will provision infrastructure on AWS, deploy the MERN stack, and implement full observability (metrics, logs, and traces).

Application Repository:

Use this open-source MERN app:
🔗 **https://github.com/UnpredictablePrashant/TravelMemory**

Part 1: Infrastructure Provisioning with Terraform

1. **AWS Setup:**
    - Configure AWS CLI with credentials
    - Initialize Terraform and define AWS as the provider
2. **Networking Configuration:**
    - Use Default **VPC**
    - Attach an **Internet Gateway**
    - Configure a **Route Table** to enable internet access
3. **EC2 Instances:**
    - Provision **two EC2 instances**:
        - One for the MERN **web server**

        - One for the **MongoDB database**
    - Both instances should be placed in the same public subnet
    - Generate Terraform `outputs.tf` to display the public IPs
4. **Security Groups:**
    - Web server: Allow inbound HTTP/HTTPS/SSH
    - DB server: Allow SSH and only allow MongoDB port (27017) from the web server's security group
5. **IAM Roles:**
    - Create IAM roles for monitoring and log access (if needed for Prometheus, CloudWatch, etc.)

Part 2: Configuration Management with Ansible

1. **Inventory Setup:**
    - Use public IPs of EC2 instances in the Ansible inventory file
2. **Web Server (App Server) Setup:**
    - Install Node.js and NPM
    - Clone the TravelMemory app repo
    - Install backend and frontend dependencies
    - Configure `.env` variables
    - Start the Node.js backend and React frontend
3. **Database Server Setup:**
    - Install and configure MongoDB
    - Create application DB and users

- Enable remote access with firewall restrictions

4. **Security Configuration:**
    - Harden SSH access (disable root login, use SSH key pairs)
    - Apply firewall rules using `ufw` or security groups

Part 3: Observability and Monitoring Setup

1. **Prometheus:**
    - Install Prometheus on the web server using Ansible
    - Use `prom-client` in Node.js backend to expose:
        - API latency
        - Error rate
        - Request count
    - Install **MongoDB Exporter** on the DB instance to collect DB metrics
2. **Grafana Dashboards:**
    - Install Grafana on the web server
    - Create dashboards to visualize:
        - Custom backend metrics
        - MongoDB performance
        - Frontend response metrics (optional)
3. **Alerting & Anomaly Detection:**
    - Set alert rules in Grafana for:

        - High error rates
        - MongoDB performance issues
        - Slow API responses
    - Enable basic anomaly detection using Prometheus data

Part 4: Documentation & Reporting

- Provide full documentation:
    - Terraform and Ansible setup
    - Application architecture (diagram)
    - Metric and log configuration
    - Grafana dashboards and alert rules (screenshots)
- Include performance analysis using collected metrics
- Describe any issues you faced and how you solved them

Deliverables:

- GitHub repository containing:
    - Terraform code (`terraform/`)
    - Ansible playbooks (`ansible/`)
    - Prometheus and Grafana configuration (`monitoring/`)
    - Documentation (`docs/README.md` or PDF)
- Include:

    - Terraform outputs
    - Sample `.env` file (with secrets redacted)
    - Screenshots of Grafana dashboards and alert configuration

Submission Instructions:

- Push everything to a public GitHub repository
- Submit a text or PDF file on **Vlearn** containing:
    - GitHub repo link
    - Your name and contact info
    - Summary of features completed