Final Group Assignment for DevOps Engineers (6 Members per Group)

Assignment Synopsis:

Your team will develop and deploy an e-commerce web application. The **frontend** will be hosted on **S3 Static Website Hosting**, while the **backend** will be hosted on **EC2 instances**. **RDS** will serve as the database, and **Terraform** will be used for infrastructure provisioning. The entire pipeline, including build, deployment, and monitoring, will be managed using **Jenkins**. **CloudWatch** will handle monitoring, **SonarQube** will ensure code quality, and **SNS** will notify of pipeline success or failure.

Team Member Responsibilities:

1. **Member 1**: Network and Security Setup
2. **Member 2**: Backend Development and Setup
3. **Member 3**: Frontend Development and Setup
4. **Member 4**: Infrastructure as Code and Jenkins Pipeline
5. **Member 5**: Monitoring, Logging, and Notifications
6. **Member 6**: Database Setup and Backup

Scenario Breakdown:

Scenario 1: Frontend Development and Hosting (Member 3)

Tasks:

1. **Develop the Frontend:**
    - Create a simple e-commerce web application frontend including:
        - Homepage
        - Registration Page
        - Login Page
        - Product Listing Page
    - Use HTML, CSS, and JavaScript.
2. **Set Up Static Website Hosting:**
    - Host the static website on **S3**.
    - Configure the S3 bucket for static website hosting and public read access.
3. **Configure CloudFront (Optional):**
    - Set up **CloudFront** for faster global delivery of the static website.
4. **Bucket Security:**

    - Implement **S3 Bucket Policies** to ensure appropriate access control.

Scenario 2: Backend Development and Setup (Member 2)

Tasks:

1. **Develop Backend API:**
    - Create backend APIs for user registration, login, and product listing using **Node.js** or **Python Flask**.
2. **Provision EC2 Instances:**
    - Set up **EC2 instances** to host the backend APIs.
    - Configure **Auto Scaling** to handle varying traffic levels.
3. **Configure Elastic Load Balancer (ELB):**
    - Set up an **ELB** to distribute incoming traffic between EC2 instances.
4. **Security Groups:**
    - Define and configure **Security Groups** to control access to EC2 instances and ELB.

Scenario 3: Database Setup and Backup (Member 6)

Tasks:

1. **Provision RDS:**
    - Create an **RDS instance** (MySQL or PostgreSQL) for storing user and product data.
2. **Database Tables:**
    - Define and create tables for users, products, and other relevant data.
3. **Backup Strategy:**
    - Configure automated backups for the RDS instance.
    - Set up additional backup storage in **S3** for recovery purposes.

Scenario 4: Infrastructure as Code and Jenkins Pipeline (Member 4)

Tasks:

1. **Terraform Scripts:**
    - Write Terraform scripts to automate:
        - VPC creation
        - EC2 instances
        - RDS instance

        - S3 bucket for static website hosting
        - IAM roles and policies
2. **Store Terraform State:**
    - Use an **S3 bucket** to store the Terraform state file.
3. **Jenkins Pipeline Setup:**
    - Configure **Jenkins** to manage the entire pipeline, including:
        - **Code Build**: Pull code from Git repository.
        - **Terraform Deployment**: Apply Terraform scripts to provision infrastructure.
        - **Application Deployment**: Deploy backend APIs and frontend.
        - **SonarQube Analysis**: Integrate code quality checks during the build process.
        - **Testing**: Implement testing phases (unit, integration) as needed.

Scenario 5: Monitoring, Logging, and Notifications (Member 5)

Tasks:

1. **CloudWatch Setup:**
    - Create **CloudWatch Alarms** for monitoring EC2, RDS, and other resources.
    - Set up **CloudWatch Logs** for logging.
2. **SonarQube Integration:**
    - Integrate **SonarQube** for code quality analysis.
    - Configure SonarQube to run during the build process and ensure quality gates are met.
3. **SNS Notifications:**
    - Create SNS topics for pipeline success and failure notifications.
    - Configure Jenkins to publish messages to these SNS topics based on build outcomes.

Scenario 6: Network and Security Setup (Member 1)

Tasks:

1. **VPC Creation:**
    - Set up a **VPC** with public and private subnets.
    - Ensure proper routing and connectivity.
2. **VPC Peering (if applicable):**
    - If using multiple VPCs, set up **VPC Peering** for internal communication.
3. **IAM Roles and Policies:**
    - Define **IAM roles** and policies to ensure least-privilege access.
4. **Security Groups and NACLs:**

    - Configure **Security Groups** and **Network ACLs** to secure resources.

Deliverables by Team Members

1. **Member 1**: Network setup, IAM configurations, and VPC peering (if applicable).
2. **Member 2**: Backend API deployment, EC2 instances setup, Auto Scaling, and ELB configuration.
3. **Member 3**: Frontend development and setup on S3, with optional CloudFront configuration.
4. **Member 4**: Terraform scripts and Jenkins pipeline setup for automated deployments.
5. **Member 5**: CloudWatch monitoring, SonarQube integration, and SNS notifications.
6. **Member 6**: RDS setup, database table creation, and backup strategy.

Deliverables (PPT):

Each group must prepare a **PowerPoint presentation** covering:

- **Architecture Diagram**: Infrastructure components and their interactions.
- **Terraform Overview**: How Terraform was used to provision resources.
- **Jenkins Pipeline**: How Jenkins was used to manage the CI/CD process, including Terraform deployment, application deployment, and testing.
- **SonarQube Integration**: Code quality checks and outcomes.
- **Pipeline Notifications**: Setup and impact of SNS notifications.