AWS 3-Tier Architecture Project Guide

Step 1: Setup Frontend (Presentation Layer)

- Go to S3 → Create bucket → Name: myecommerce-frontend
- Uncheck 'Block all public access' → Enable Static Website Hosting
- Upload index.html, style.css, etc.
- Make files public → Add bucket policy for GetObject
- Set up CloudFront → Origin: S3 bucket → Enable HTTPS
- Optional: Use Route 53 for custom domain

Step 2: Setup Backend (Application Layer)

- Create a VPC with public & private subnets
- Launch EC2 in private subnet → Install backend stack (Node.js/Java/Python)
- Example Node.js setup: sudo yum update -y sudo yum install -y nodejs npm git git clone https://github.com/your-repo/ecommerce-backend.git cd ecommerce-backend && npm install && npm start
- Create Application Load Balancer (ALB) in public subnet
- Attach EC2 instances to ALB target group
- Configure Auto Scaling Group for EC2 backend

Step 3: Setup Database (Data Layer)

- Launch RDS (MySQL/Postgres) in private subnet
- Configure DB username/password, enable backups & Multi-AZ
- Update backend config to connect with RDS: DB_HOST=mydb.abc123xyz.us-east-1.rds.amazonaws.com DB_USER=admin DB_PASS=yourpassword DB_NAME=ecommerce
- Optional: Use DynamoDB for product catalog/session storage
- Optional: Use S3 for storing product images, invoices, receipts

Step 4: Security Setup

- Create IAM roles to allow EC2 access to S3
- Security Groups: - ALB: Allow 80/443 - EC2: Allow traffic only from ALB - RDS: Allow traffic only from EC2
- Enable WAF to protect against SQL Injection & XSS

Step 5: Monitoring & Logging

- Enable CloudWatch Metrics to monitor EC2, RDS
- Enable CloudWatch Logs for backend app logs
- Enable CloudTrail to track AWS changes

Step 6: Test Flow

- User → myecommerce.com (CloudFront + S3)
- Frontend → API call → ALB → EC2 backend
- Backend → Query → RDS
- Response → Frontend → Display products/orders