**<https://github.com/suvaatnbu/E-commerce-.git>**

**Deploy a Multi-Service Node.js E-commerce Application Using Terraform and Docker (**[**LINK**](https://github.com/AtharvaAI/E-CommerceStore.git)**)**

## (50 Marks | 1 Hour)

Background:  
You are a DevOps Engineer tasked with provisioning infrastructure and deploying a Node.js-based e-commerce application using Terraform and Docker on AWS. The application has four backend services (user, products, orders, cart) and one frontend.  
Each service must be containerized using Docker and deployed on AWS EC2 instances provisioned with Terraform.

### 

### Objective:

Use Terraform to provision AWS infrastructure and deploy all 5 Dockerized services (built by you) within 1 hour. The frontend must be accessible via a public IP or DNS.

### 

### Requirements:

#### 1. Application Setup (Manual - Not Terraform)

* Create Dockerfiles for each of the 5 services:  
    
  + Each service must expose a relevant port and return a sample response (e.g., "user Service Running")
  + Build and test the Docker images locally
  + Tag and push the images to DockerHub

#### 2. Infrastructure Provisioning with Terraform (25 Marks)

Provision the following using Terraform:

* VPC with at least one public subnet
* 1 or more EC2 Instances to host the Docker containers
* Security Groups to allow:  
    
  + Inbound HTTP (port 80 or 3000) to the frontend
  + Internal communication between services (custom ports, e.g., 3001–3004)
* Use Terraform provisioners or user-data scripts to:  
    
  + Install Docker on EC2
  + Pull all 5 images from DockerHub
  + Run the containers on proper ports

#### 3. Deployment and Accessibility (25 Marks)

* Ensure the frontend service is publicly accessible  
    
  + Display a homepage or “Frontend is Live” message
* Verify backend containers are running (can use simple log messages or API responses)
* Use terraform output to print the public IP or DNS of the application

### Evaluation Criteria

|  |  |
| --- | --- |
| Component | Marks |
| Default VPC, subnet, and security group setup | 10 |
| EC2 provisioning with Docker setup | 10 |
| Docker container deployment via user-data or provisioners | 10 |
| Public access to frontend | 10 |
| DOCUMENTATION | 10 |

### 

### Assumptions & Constraints

* Use Ubuntu EC2 AMIs (e.g., Amazon Linux 2 or Ubuntu 22.04)
* Use DockerHub to host your images (must be public or use credentials)
* All configurations should be reproducible with terraform apply

**ANSWER:**

**ecommerce-app/**

**├── user/**

**│ ├── Dockerfile**

**│ └── index.js**

**├── products/**

**│ ├── Dockerfile**

**│ └── index.js**

**├── orders/**

**│ ├── Dockerfile**

**│ └── index.js**

**├── cart/**

**│ ├── Dockerfile**

**│ └── index.js**

**├── frontend/**

**│ ├── Dockerfile**

**│ └── index.js**

**terraform/**

**├── main.tf**

**├── variables.tf**

**├── outputs.tf**





