1. **Discuss & justify potential deployment options in terms of infrastructure for development environment of above application**

**Option 1: Using an AWS EC2 Instances with Docker**

Deploy the application on an AWS EC2 instance using Docker. Either run a docker daemon as a single node or setup docker swarm in an EC2 instance

**1a. Running docker daemon on a single EC2 instance**

Docker runs directly on a single EC2 instance. Containers are deployed and managed individually on that instance.

This setup is straightforward and easy to configure. It’s ideal for small applications or for a development environments.

With only one instance, the cost is lower compared to a multi-instance setup.

It’s easy to setup

If the EC2 instance fails, all services running on that instance will be unavailable, leading to potential downtime.

There’s no built-in mechanism for load balancing, high availability, or failover.

**1b. Running Docker Swarm on Multiple EC2 Instances**

In this setup, Docker Swarm is configured across multiple EC2 instances, forming a cluster with manager and worker nodes.

Docker Swarm provides built-in redundancy. If a node fails, the workloads are redistributed across the remaining healthy nodes, ensuring minimal downtime.

Easily scale applications by adding more EC2 instances (nodes) to the swarm. Swarm automatically manages the distribution of containers across nodes

Swarm provides built-in load balancing, distributing incoming requests across the service replicas running on different nodes.

Swarm manages inter-container communication, service discovery, and networking across the cluster, reducing manual overhead.

**Option 2. AWS Elastic Kubernetes Service (EKS)**

Use Amazon EKS to manage Kubernetes clusters where the development environment is hosted.

EKS provides a development environment that closely mirrors the production environment if the production environment also uses EKS.

Kubernetes inherently provides scalability features, which allows the development environment to handle larger tests if needed.

Kubernetes optimizes the use of underlying resources, which is beneficial if multiple developers are sharing the environment.

EKS supports features like auto-scaling, rolling updates, secrets management, complex scheduling, multi-tenancy, and advanced networking configurations.

Kubernetes has a vast ecosystem of tools, plugins, and integrations, allowing for advanced customization, monitoring, logging, and networking. The ecosystem includes Helm for package management, Istio for service mesh, Prometheus for monitoring, and more.

**My Choice:**

**Docker Swarm on EC2 Instances**: Ideal for smaller, simpler applications or environments where ease of use and low cost are more important than advanced features and scalability. It's a good choice for teams that need a straightforward container orchestration tool with minimal overhead.

**Amazon EKS (Elastic Kubernetes Service)**: The better choice for production-grade applications requiring high scalability, advanced orchestration features, and seamless integration with other AWS services. EKS is more complex but offers a robust, future-proof solution for managing containerized workloads at scale.