**Project report by: Dinesh, Varsha, Shivani, Kiranmai**

**Dockerizing MEAN-Stack application and Deploying it into Kubernetes (klla coda)**

**MEAN Stack:**

The MEAN stack is a popular set of technologies used for developing web applications. The acronym MEAN stands for:

**MongoDB:** A NoSQL database that uses a JSON-like format for storing data, which provides flexibility and scalability.

**Express.js:** A lightweight framework for Node.js that simplifies building server-side applications and APIs.

**Angular:** A front-end web application framework developed by Google that helps in building dynamic and robust single-page applications (SPAs).

**Node.js:** A runtime environment that allows JavaScript to be used for server-side scripting, enabling the development of scalable and high-performance applications.

**Key Components of the MEAN Stack**

**MongoDB**

Type: NoSQL Database

Description: MongoDB is a document-oriented database that stores data in flexible, JSON-like documents. This structure allows for a more dynamic and adaptable schema compared to traditional relational databases.

Use Case: Ideal for handling large volumes of data and for applications requiring flexible schemas.

**Express.js**

Type: Web Application Framework for Node.js

Description: Express.js is a minimal and flexible Node.js web application framework that provides a robust set of features for building web and mobile applications. It simplifies the development of server-side code by providing utilities for routing, middleware, and more.

Use Case: Suitable for building RESTful APIs, handling HTTP requests and responses, and integrating with databases.

**Angular**

Type: Front-end Framework

Description: Angular is a platform and framework for building single-page client applications using HTML and TypeScript. It provides powerful tools for building interactive and dynamic user interfaces, including data binding, dependency injection, and routing.

Use Case: Ideal for developing rich, client-side applications with a clean and structured approach.

**Node.js**

Type: JavaScript Runtime Environment

Description: Node.js is built on Chrome's V8 JavaScript engine and enables JavaScript to be used for server-side scripting. It is known for its non-blocking, event-driven architecture, which helps in building scalable network applications.

Use Case: Suitable for handling concurrent connections and building high-performance, real-time applications.

**Docker**

Docker is a platform used for developing, shipping, and running applications in a standardized environment known as containers. Containers are lightweight, portable, and self-sufficient units that include everything needed to run an application—such as code, runtime, system tools, libraries, and settings—ensuring consistency across different environments.

**Key Concepts**

**Containers:** Containers are isolated, lightweight environments that encapsulate an application and its dependencies. They share the host OS kernel but run in separate user spaces.Containers ensure that the application runs consistently regardless of where it is deployed (development, testing, production). They start quickly and are more resource-efficient compared to virtual machines.

**Images:** An image is a snapshot of a container. It contains everything needed to run an application: code, runtime, libraries, and configurations. Docker images are immutable and can be versioned.Images are used to create containers. They can be built from Dockerfiles and stored in container registries.

**Dockerfile:** A Dockerfile is a text file containing a series of instructions on how to build a Docker image. It specifies the base image, the software to be installed, and the configuration required.A Dockerfile might include steps to install a web server, copy application files, and set up environment variables.

**Docker Engine:** Docker Engine is the core component of Docker that runs and manages containers. It consists of a server (Docker Daemon), REST API, and a command-line interface (CLI).The Docker Engine handles container operations like creating, running, and stopping containers.

**Docker Compose:** Docker Compose is a tool for defining and running multi-container Docker applications. It uses a YAML file (docker-compose.yml) to configure services, networks, and volumes.Docker Compose simplifies the orchestration of multiple containers that work together, such as a web server and a database.

**Docker Hub:** Docker Hub is a cloud-based container registry service where Docker images can be stored and shared. It allows you to pull public images and push your own images.You can find official images for popular software and frameworks, as well as share your custom images with others.

**Kubernetes**

Kubernetes, also known as K8s, is an open-source system for automating deployment, scaling, and management of containerized applications. Originally designed by Google, the project is now maintained by a worldwide community of contributors, and the trademark is held by the Cloud Native Computing Foundation. It has a large, rapidly growing ecosystem. Kubernetes services, support, and tools are widely available.

Key Concepts: containers, pods, Deployments, services, namespaces, volumes, config maps and secrets, ingress.

**Dockerizing MEAN-stack application:**

1. **Creating MEAN-Stack application in vs code.**

https://github.com/my-git-codes/Mean-Stack-Docker.git

1. **Composing MEAN-Stack application in docker.**

-Docker-compose up

1. **Pushing the images into docker hub.**

-Docker tag mean-stack-docker-example-angular:latest shivanidandu809/mean-stack-docker-example-angular:latest

- docker tag mean-stack-docker-example-express:latest shivanidandu809/mean-stack-docker-example-express:latest

-docker push shivanidandu809/mean-stack-docker-example-angular:latest

-docker push shivanidandu809/mean-stack-docker-example-express:latest

1. **Creating deployment and service yaml files in klla coda.**

**Deployment.yaml**

apiVersion: apps/v1  
kind: Deployment  
metadata:  
  name: multi-container-deployment  
spec:  
  replicas: 1  
  selector:  
    matchLabels:  
      app: multi-container-app  
  template:  
    metadata:  
      labels:  
        app: multi-container-app  
    spec:  
      containers:  
      - name: mean-stack-docker-example-angular-1  
        image: varshadadi/mean-stack-docker-example-angular:latest  
        ports:  
        - containerPort: 80  
      - name: mean-stack-docker-example-express-1  
        image: varshadadi/mean-stack-docker-example-express:latest  
        ports:  
        - containerPort: 3000  
      - name: mean-stack-docker-example-database-1  
        image: varshadadi/mongo:latest  
        ports:  
        - containerPort: 27017

**Service.yaml**

apiVersion: v1  
kind: Service  
metadata:  
  name: angular-service  
spec:  
  selector:  
    app: multi-container-app  
  ports:  
    - protocol: TCP  
      port: 80  
      targetPort: 80  
  type: ClusterIP  # Change to LoadBalancer if you want to expose it outside the cluster  
---  
apiVersion: v1  
kind: Service  
metadata:  
  name: express-service  
spec:  
  selector:  
    app: multi-container-app  
  ports:  
    - protocol: TCP  
      port: 3000  
      targetPort: 3000  
  type: ClusterIP  # Change to LoadBalancer if you want to expose it outside the cluster  
---  
apiVersion: v1  
kind: Service  
metadata:  
  name: mongo-service  
spec:  
  selector:  
    app: multi-container-app  
  ports:  
    - protocol: TCP  
      port: 27017  
      targetPort: 27017  
  type: ClusterIP  # Change to LoadBalancer if you want to expose it outside the cluster

1. **Exposing and accessing service**

kubectl apply -f deployment.yaml

kubectl get pods

kubectl get deployments

kubectl apply -f service.yaml

kubectl get svc