Today am created one project

--------------------zomato—app--------------------------

Okay! Full A to Z detailed process cheptanu, folder structure nunchi commands varaku, output kuda explain chesthanu step-by-step. Ready?

Step 0: Setup - Basic Requirements

Installations:

Node.js (for dummy app)

Docker

kubectl (Kubernetes CLI)

Minikube or any local Kubernetes cluster (for local testing)

Step 1: Project Folder Setup

Folder create cheyyandi (Windows/Mac/Linux terminal or VSCode terminal lo):

**mkdir zomato-dummy-app**

**cd zomato-dummy-app**

Project structure ila untundi:

zomato-dummy-app/

├── server.js

├── package.json

├── Dockerfile

├── deployment.yaml

└── service.yaml

Step 2: Create Node.js Dummy App

Initialize Node.js project

npm init -y

Output:

Wrote to /path/to/zomato-dummy-app/package.json:

{

"name": "zomato-dummy-app",

"version": "1.0.0",

"main": "index.js",

...

}

Install express

npm install express

Output:

added 50 packages in 2s

Create server.js file

Create a file named server.js with this content:

js

const express = require('express');

const app = express();

const port = 3000;

app.get('/restaurants', (req, res) => {

res.json([

{ id: 1, name: "Zomato Dummy Restaurant" },

{ id: 2, name: "Foodie Hub" }

]);

});

app.listen(port, () => {

console.log(`App running at http://localhost:${port}`);

});

Step 3: Run Locally Test Cheyyadam

node server.js

Output terminal lo:

App running at http://localhost:3000

Browser or Postman lo http://localhost:3000/restaurants open cheyyandi, output ila untundi:

json

[

{ "id": 1, "name": "Zomato Dummy Restaurant" },

{ "id": 2, "name": "Foodie Hub" }

]

Step 4: Create Dockerfile

Create a file named Dockerfile with content:

dockerfile

FROM node:16

WORKDIR /app

COPY package\*.json ./

RUN npm install

COPY . .

EXPOSE 3000

CMD ["node", "server.js"]

Step 5: Build Docker Image

Terminal lo:

docker build -t zomato-dummy-app .

Output (last lines):

Successfully built <image\_id>

Successfully tagged zomato-dummy-app:latest

Step 6: Run Docker Container Locally Test

docker run -p 3000:3000 zomato-dummy-app

Terminal lo output:

App running at http://localhost:3000

Browser lo open http://localhost:3000/restaurants output same as before.

Step 7: Push Docker Image to Docker Hub

Login to Docker Hub (only once):

docker login

Tag image:

docker tag zomato-dummy-app yourdockerhubusername/zomato-dummy-app:latest

Push image:

docker push yourdockerhubusername/zomato-dummy-app:latest

Step 8: Create Kubernetes YAML files

Create deployment.yaml:

yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: zomato-dummy-app

spec:

replicas: 2

selector:

matchLabels:

app: zomato-dummy-app

template:

metadata:

labels:

app: zomato-dummy-app

spec:

containers:

- name: zomato-dummy-app

image: yourdockerhubusername/zomato-dummy-app:latest

ports:

- containerPort: 3000

Create service.yaml:

apiVersion: v1

kind: Service

metadata:

name: zomato-dummy-service

spec:

type: NodePort

selector:

app: zomato-dummy-app

ports:

- protocol: TCP

port: 80

targetPort: 3000

nodePort: 30080

Step 9: Apply Kubernetes Configs

Start minikube (if local):

minikube start

kubectl apply -f deployment.yaml

kubectl apply -f service.yaml

Output:

deployment.apps/zomato-dummy-app created

service/zomato-dummy-service created

Step 10: Check Pods and Service

kubectl get pods

Output:

NAME READY STATUS RESTARTS AGE

zomato-dummy-app-xxxxx 1/1 Running 0 1m

zomato-dummy-app-yyyyy 1/1 Running 0 1m

kubectl get svc

Output:

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

zomato-dummy-service NodePort 10.96.222.156 <none> 80:30080/TCP 1m

kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 5m

Step 11: Access Service

Get minikube IP:

minikube ip

Suppose output:

192.168.49.2

Open in browser:

http://192.168.49.2:30080/restaurants

Output:

json

Copy

Edit

[

{ "id": 1, "name": "Zomato Dummy Restaurant" },

{ "id": 2, "name": "Foodie Hub" }

]

All done!

First am creted one floder—in floder am am creted some files with code (every thing am uploaded in github check)

Check node -v

npm -v

Firstly init -y ---------- for package.json

Before it is there you can delete ------ dir package.json

Now run ------- npm install express ------------- install express

node server.js ---------------- server run ------- it will give one url run and check it

[{"id":1,"name":"Zomato Restaurant"},{"id":2,"name":"Foodie Hub"}] ----------- I got output this

docker build -t zomato-app .

* docker build -t zomato-app .
  + -t zomato-app = image ki peru
  + . = current folder lo Dockerfile ni use cheyyi ani artham

Run Docker Container Locally and Test----------- docker run -p 3000:3000 zomato-app

<http://localhost:3000/restaurants-----------> run in browser you can see output

Ikkada varaku app Docker container lo local ga run avutondi ani confirm avutundi.

Push Docker Image to Docker Hub---

**Step 7.1: Docker Hub Login**

bash

CopyEdit

docker login

* Username, password adugutundi.
* Login successful anagane next step ki vellu.

Tag the Image : docker tag zomato-app sirisha015/zomato-app:latest

Push the Image : docker push sirisha015/zomato-app:latest

Image Docker Hub ki successfully push ayyindi ani artham.

Create two file --- service.yaml & deployment.yaml

Now apply kubernets confugurations

kubectl apply -f deployment.yaml

kubectl apply -f service.yaml

output::: deployment.apps/zomato-app created

service/zomato-service created

**Tools & Why Used**

1. **Node.js** – Fast, lightweight JavaScript runtime to quickly build the dummy REST API.
2. **Express.js** – Simple framework for Node.js to handle HTTP requests easily.
3. **Docker** – To package the app and its dependencies into a container so it runs the same everywhere.
4. **Docker Hub** – Cloud repository to store and share the built Docker image.
5. **Kubernetes (kubectl)** – To deploy, manage, and scale the containerized app automatically.
6. **Minikube** – Local Kubernetes cluster for testing before deploying to cloud.
7. **YAML Manifests (deployment.yaml, service.yaml)** – Declarative way to tell Kubernetes how to run and expose the app.

**In short:**

* **Node.js + Express** → App creation.
* **Docker** → Containerization.
* **Kubernetes + Minikube** → Orchestration & scaling.
* **Docker Hub** → Image sharing & pulling.