**CLOUD NATIVE DEVELOPMENT**

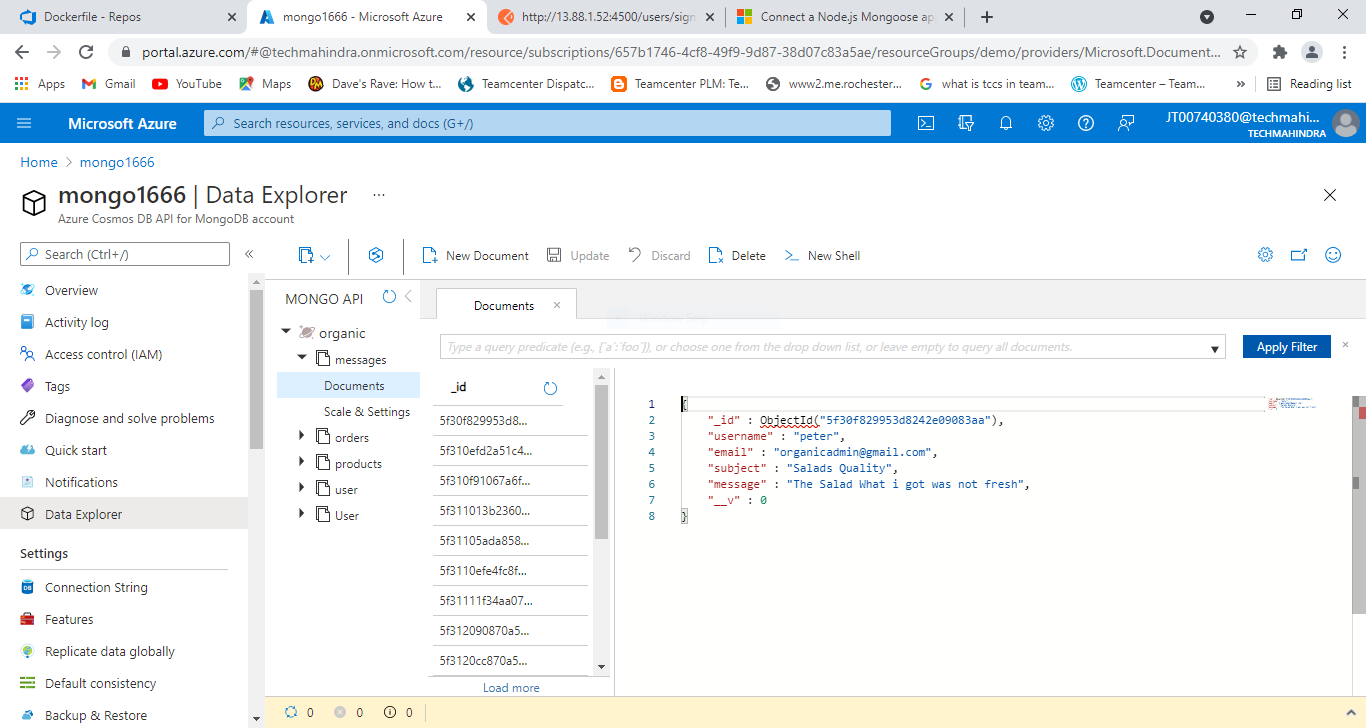
## **PROJECT CASE STUDY- IMPLEMENTATION AND APPROACH**

**To Develop API service using NODEJS with MongoDB as the backend Database**

**NODE JS AND MONGO API CONNECTION**

The Node JS Services Exposes the data in form of api, but the data for the Service is available in Cosmodb Mongo API.

1. Create a cosmodb mongo api service



**OPEN VS CODE**

**Open folder NODEJS-SERVER**

**Step 1 : Create a Dockerfile**

FROM node

WORKDIR /usr/src/app

COPY package\*.json ./

RUN npm install

COPY . .

EXPOSE 4500

CMD ["node","index.js"]

**Step 2 : Create environment variables**

**#** You can get the following connection details from the Azure portal. You can find the details on the Connection string pane of your Azure Cosmos account.

**COSMOSDB\_USER = "<Azure Cosmos account's user name, usually the database account name>"**

**COSMOSDB\_PASSWORD = "<Azure Cosmos account password, this is one of the keys specified in your account>"**

**COSMOSDB\_DBNAME = "<Azure Cosmos database name>"**

**COSMOSDB\_HOST= "<Azure Cosmos Host name>"**

**COSMOSDB\_PORT=10255**

**step 3 : Make changes in Index.js**

1. **Import the dependencies in your index.js file.**

var mongoose = require('mongoose');

var env = require('dotenv').config(); //Use the .env file to load the variables

1. **Connect to Cosmos DB using the Mongoose framework by adding the following code to the end of index.js.**

mongoose.connect("mongodb://"+process.env.COSMOSDB\_HOST+":"+process.env.COSMOSDB\_PORT+"/"+process.env.COSMOSDB\_DBNAME+"?ssl=true&replicaSet=globaldb", {

auth: {

user: process.env.COSMOSDB\_USER,

password: process.env.COSMOSDB\_PASSWORD

},

useNewUrlParser: true,

useUnifiedTopology: true,

retryWrites: false

})

.then(() => console.log('Connection to CosmosDB successful'))

.catch((err) => console.error(err));

In the terminal :

Install the necessary packages using one of the npm install

Dotenv (if you'd like to load your secrets from an .env file): npm install dotenv --save

Run : node index.js

Text

Description automatically generated

DEPLOYING IN NODEJS API IN AZURE KUBERNETES CLUSTER AS 3 INSTANCES

1. Create **deployment.yml** and write code
2. apiVersion: apps/v1
3. kind: Deployment
4. metadata:
5. name: nodejs-deployment
6. labels:
7. app: nodejsserver
8. spec:
9. replicas: 3
10. selector:
11. matchLabels:
12. app: nodejsserver
13. template:
14. metadata:
15. labels:
16. app: nodejsserver
17. spec:
18. containers:
19. - name: nodejsserver
20. image:  azure477.azurecr.io/nodejsserver:latest
21. ports:
22. - containerPort: 4500
24. ---
25. apiVersion: v1
26. kind: Service
27. metadata:
28. name: nodejs-service
29. spec:
30. type: LoadBalancer
31. selector:
32. app: nodejsserver
33. ports:
34. # By default and for convenience, the `targetPort` is set to the same value as the `port` field.
35. - port: 4500
36. targetPort: 4500

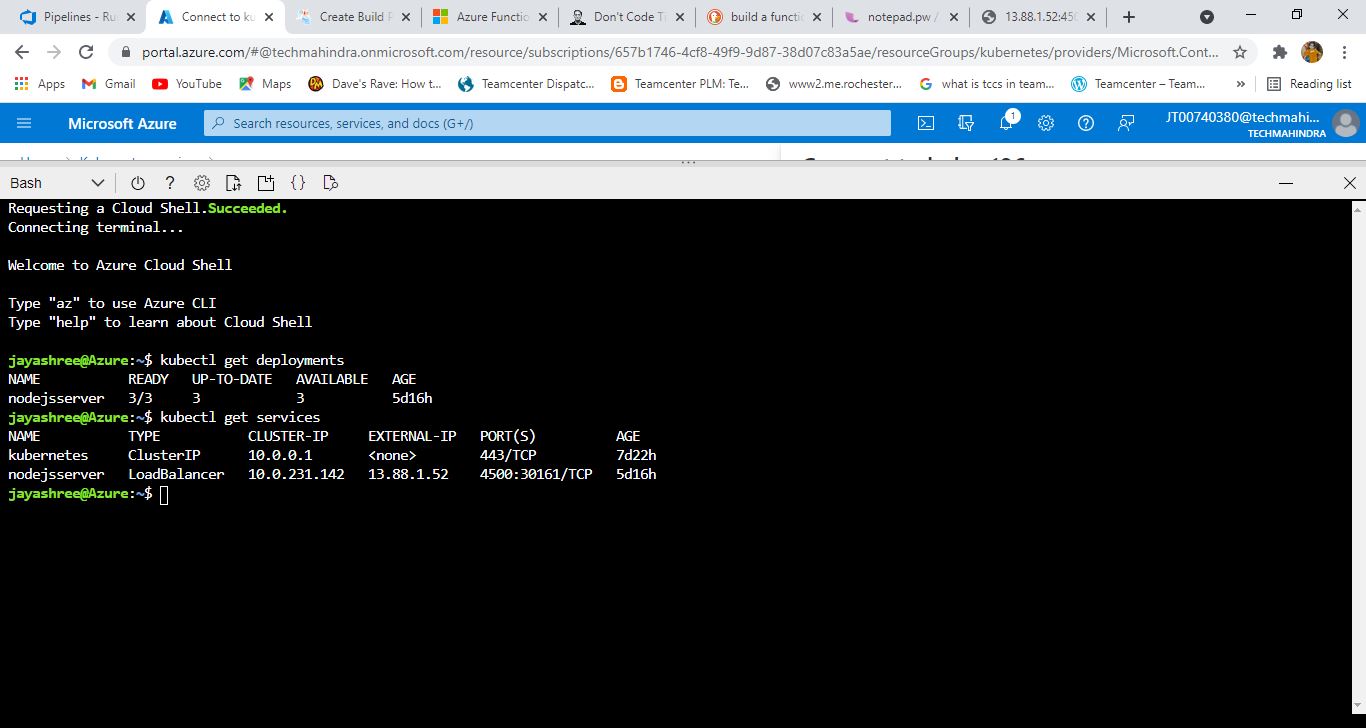
Build a pipeline

PUSH it to Azure container registry and deploy it in Kubernetes cluster

Open shell

Kubectl get deployments

Kubectl get services



Angular application

Open project in VSCode :

* Run npm install to load packages (node\_modules)
* Run ng build to build the application
* Run ng serve to run the application

Push it to azure Repos

Write a Dockerfile which will build the image and push to ACR

