3TIER STUDENT-APP DEPLOY ON KUBENETES

1) Create mariadb RDS instance for database	→	**DATABASE**
2) Write manifest for deployment and service	→	**BACKEND**
3) Write manifest for deployment and service	→	**FRONTEND**

DATABASE

1. Create rds database - mariadb engin - no public access
2. Create ec2 instance and connect with rds database
3. Connect to ec2 instance terminal

   a) yum install mariadb-server
   b) systemctl start mariadb
   c) Mysql -h <rds_endpoint> -u <db_user> -p<db_pass>
   d) Create database studentapp;
   e) Use studentapp;
   f) Insert data schema>

```
CREATE TABLE if not exists students(student_id INT NOT NULL AUTO_INCREMENT,
                    student_name VARCHAR(100) NOT NULL,
                    student_addr VARCHAR(100) NOT NULL,
                    student_age VARCHAR(3) NOT NULL,
                    student_qual VARCHAR(20) NOT NULL,
                    student_percent VARCHAR(10) NOT NULL,
                    student_year_passed VARCHAR(10) NOT NULL,
                    PRIMARY KEY (student_id)
               );
```

   g) select * from students;
   h) exit

—--

- **Go to EKS and create cluster and create nobe within cluster**

BACKEND

1. Create docker images attach rds database and push to dockerhub
   a) Connect to cloudshell in aws
   b) Clone docker project repo
      git clone https://github.com/prathammore0025/devops.git

   c) Open repo and go inside backend directory
      cd devops/backend

d) Configure context.xml add database endpoint, username and password
vim context.xml

```
<Resource name="jdbc/TestDB" auth="Container"
type="javax.sql.DataSource"
               maxTotal="100" maxIdle="30" maxWaitMillis="10000"
               username="USERNAME" password="PASSWORD"
driverClassName="com.mysql.jdbc.Driver"
               url="jdbc:mysql://DB-ENDPOINT:3306/studentapp"/>
```

e) Create docker image of backend
docker build . -t <dockerhub-id>/<img-name>:<img-tag>
docker build . -t prathammore0025/back:v1

f) Login with docker-hub
username: **<id of docker-hub>**
password : **<pass of docker-hub>**

g) Push the backend image to docker hub
docker push <dockerhub-id>/<img-name>:<img-tag>
docker push prathammore0025/back:v1

2. Create repo named MY-K8S and add two folder in it **frontend** and **backend** (github web)

3. Go to vs code connect to repo and open backend folder
4. create 2 file in it **deployment.yml** and **service.yml**
5. Connect to repo in vs code
6. Open **backend** folder and create 2 file in it **deployment.yml** and **service.yml**
7. Write manifest → **deployment.yml**

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend
  labels:
    app: backend
spec:
  template:
    metadata:
      labels:
        app: backend
    spec:
      containers:
        - name: backend
          image: prathammore0025/back:v1     —-(dockerhub img name)
          ports:
            - containerPort: 8080
  replicas: 3
  selector:
    matchLabels:
      app: backend
  strategy:
    type: RollingUpdate
```

8. Write service.yml

```
apiVersion: v1
kind: Service
metadata:
  name: backend-service
  labels:
    app: backend
spec:
  ports:
    - port: 80
      targetPort: 8080
      protocol: TCP
  type: NodePort
  selector:
    app: backend
```

9. Commit and sync changes

10. Connect to cloudshell
11. Clone new repo MY-K8S
    git clone https://github.com/prathammore0025/my-k8s.git

12. Connect the cluster
    aws eks update-kubeconfig --name <cluster**> --region <**eu-west-3**>**

13. Go inside the backed directory
    cd MY-K8S/backend

14. Apply deployment.yml and service.yml
    kubectl apply -f .

15. Check node port
    kubectl get service

16. Hit node ip with nodeport on browser
    Node-IP:nodeport
    15.237.43.227:30555/ —----------> show tomcat page
    15.237.43.227:30555/student/ —----> show studentapp
    Save the data

    Copy the studentapp url for frontend → http://15.237.43.227:30555/student/

—---

FRONTEND

1. Connect to cloudshell in aws
2. Open docker project repo and go inside frontend directory
   cd devops/frontend

3. Add backend url in index.html
   vim index.html
   ```
   <h1 style="text-align: center;"><span style="color: #ff0000;">Welcome to Student Application on AWS.</span></h1>
   <p><img style="display: block; margin-left: auto; margin-right: auto;"
   src="https://cdn-images-1.medium.com/max/2000/1*tFl-8wQUENETYLjX5mYWuA.png" alt="" width="1200"
   height="630" /></p>
   <p>&nbsp;</p>
   <h2 style="text-align: center;"><a href="student"><strong>Enter to Student
   Application</strong></a></h2><p>&nbsp;</p>
   <p>&nbsp;</p>
   ```
   - Replace highlighted student word with backend url
   - **wq!** For save and exit

4. Create docker images
   docker build . -t prathammore0025/front:v1

5. Push docker images to docker dockerhub
   docker push prathammore0025/front:v1

6. Connect to new K8S repo in vs code
7. Open **frontend** folder and create 2 file in it **deployment.yml** and **service.yml**
8. Write manifest → **deployment.yml**
   ```
   apiVersion: apps/v1
   kind: Deployment
   metadata:
     name: frontend
     labels:
       app: frontend
   spec:
     template:
       metadata:
         labels:
           app: frontend
       spec:
         containers:
           - name: frontend
             image: prathammore0025/front:v1
             ports:
               - containerPort: 80
     replicas: 3
     selector:
   ```

```
    matchLabels:
      app: frontend
  strategy:
    type: RollingUpdate
```

9. Write the → **service.yml**

```
apiVersion: v1
kind: Service
metadata:
  name: frontend-service
  labels:
    app: frontend
spec:
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
  selector:
    app: frontend
  type: LoadBalancer
```

10. Commit and sync changes

11. Connect to cloudshell
12. Go inside the backed directory
    cd MY-K8S/frontendend

13. Pull changes
    git pull origin main

14. Apply deployment.yml and service.yml
    kubectl apply -f .

15. Check loadbalancer DNS
    kubectl get service

16. Hit loadbalancer DNS on browser
    a3d79aa4ddbf34bd5b1fca070bdf220a-2081283822.eu-west-3.elb.amazonaws.com