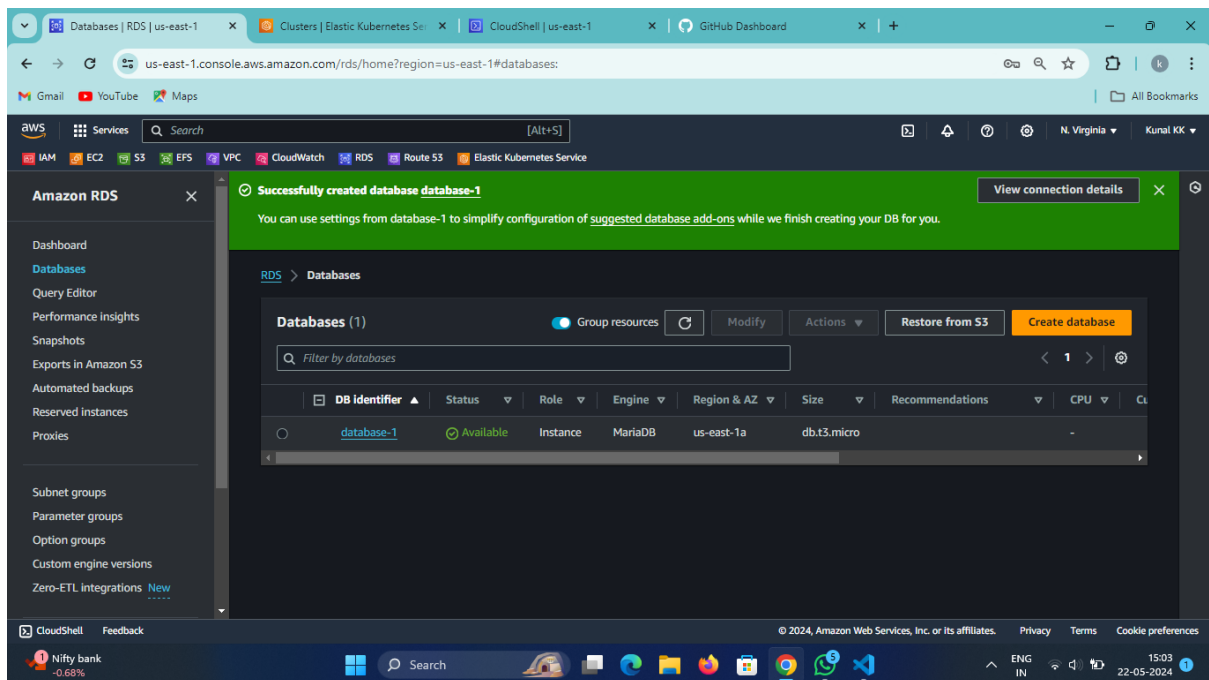
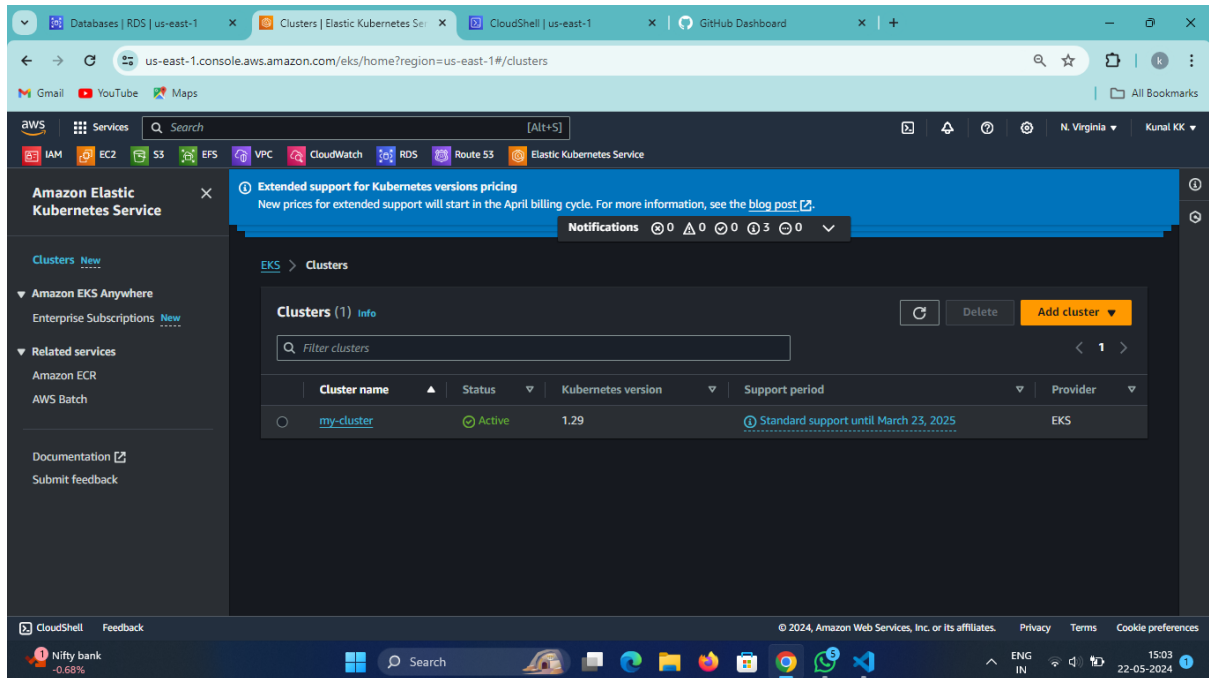
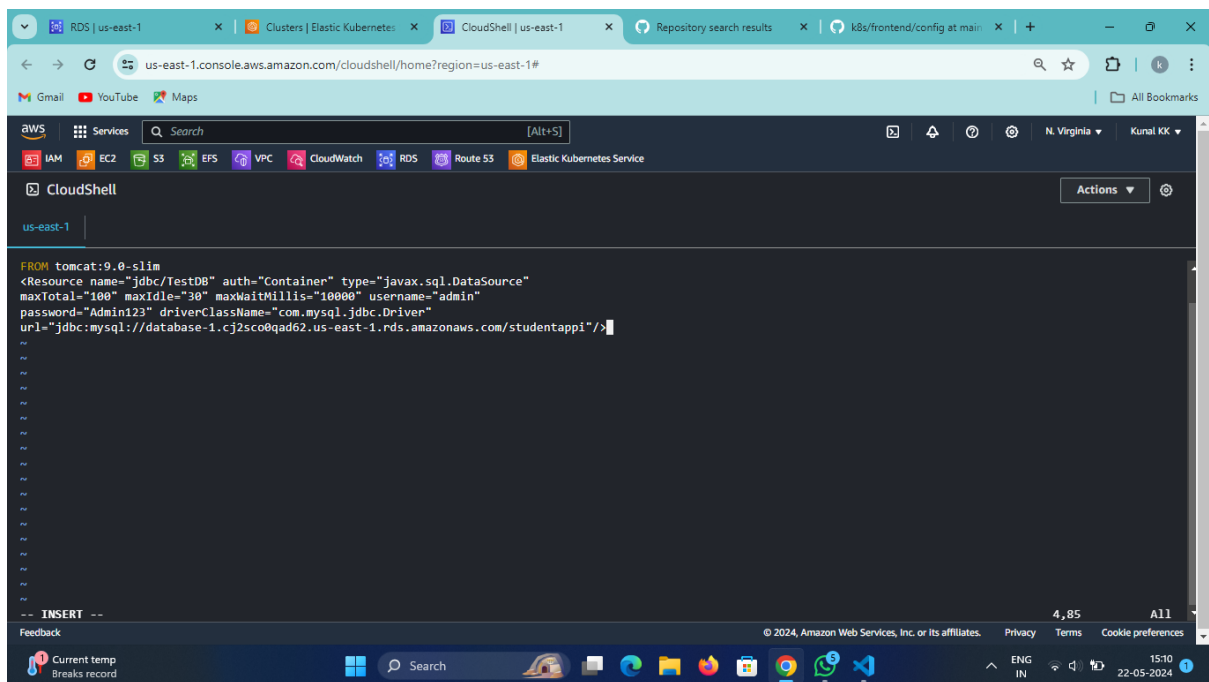
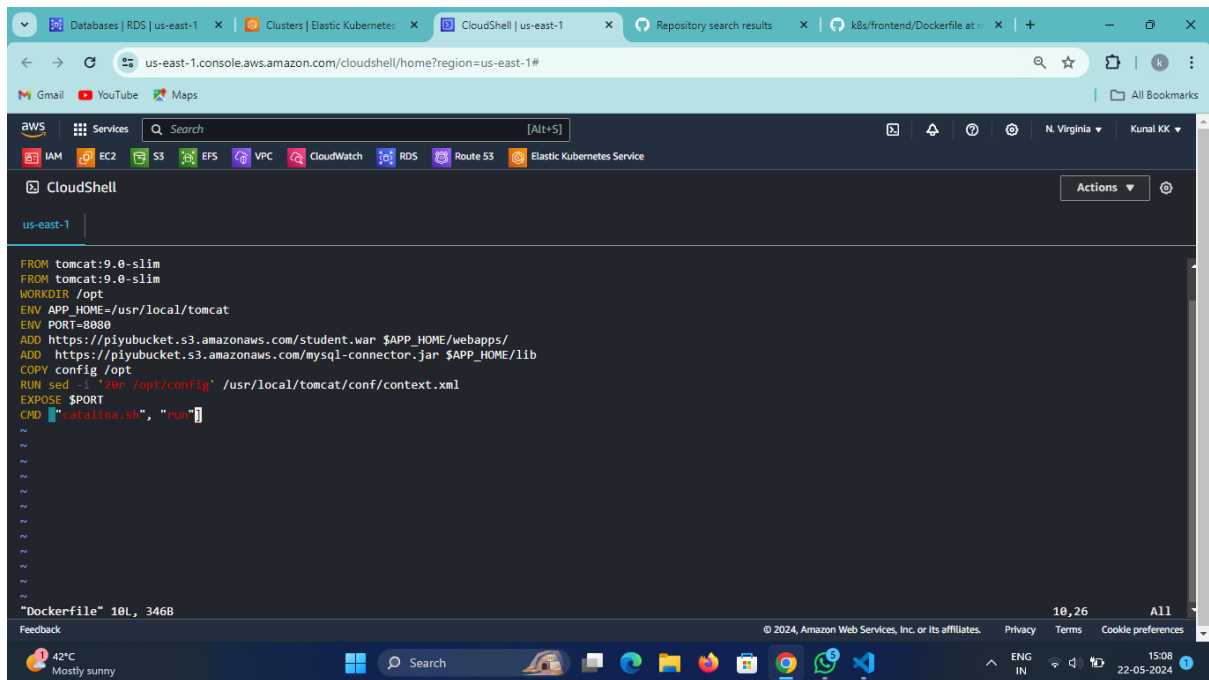


KUBERNETES 3 TIER PROJECT USING DB INSTANCE

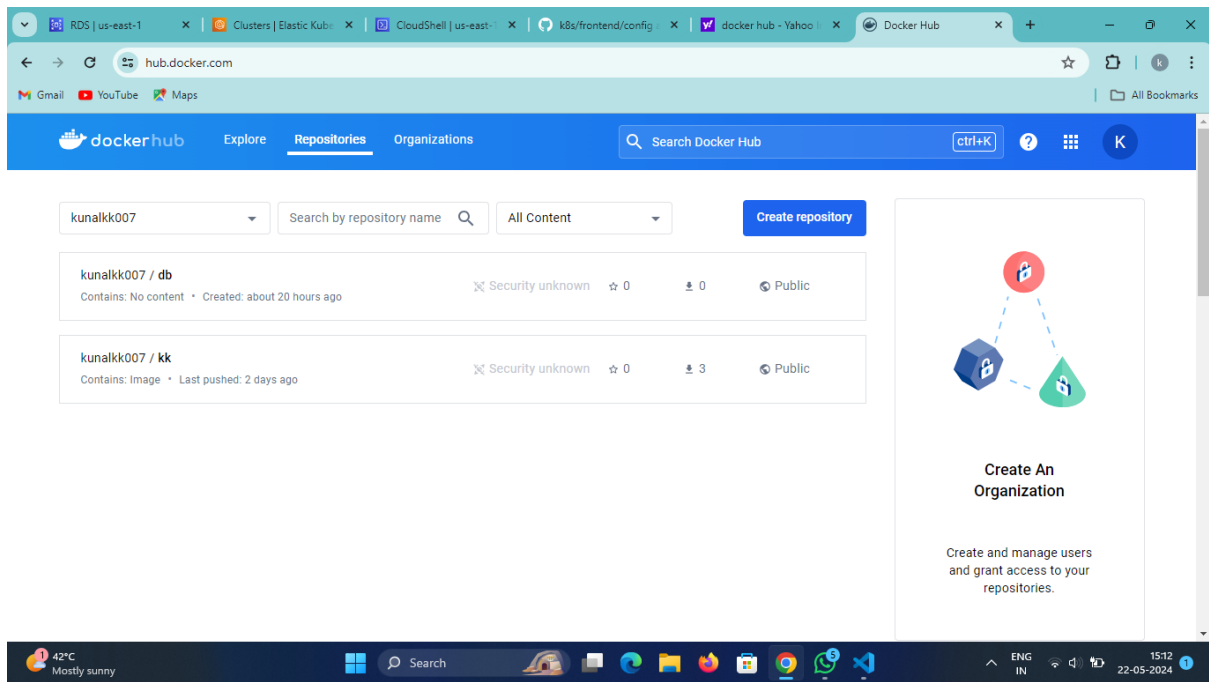
1. First create a cluster.
2. Then create an DB instance in RDS.



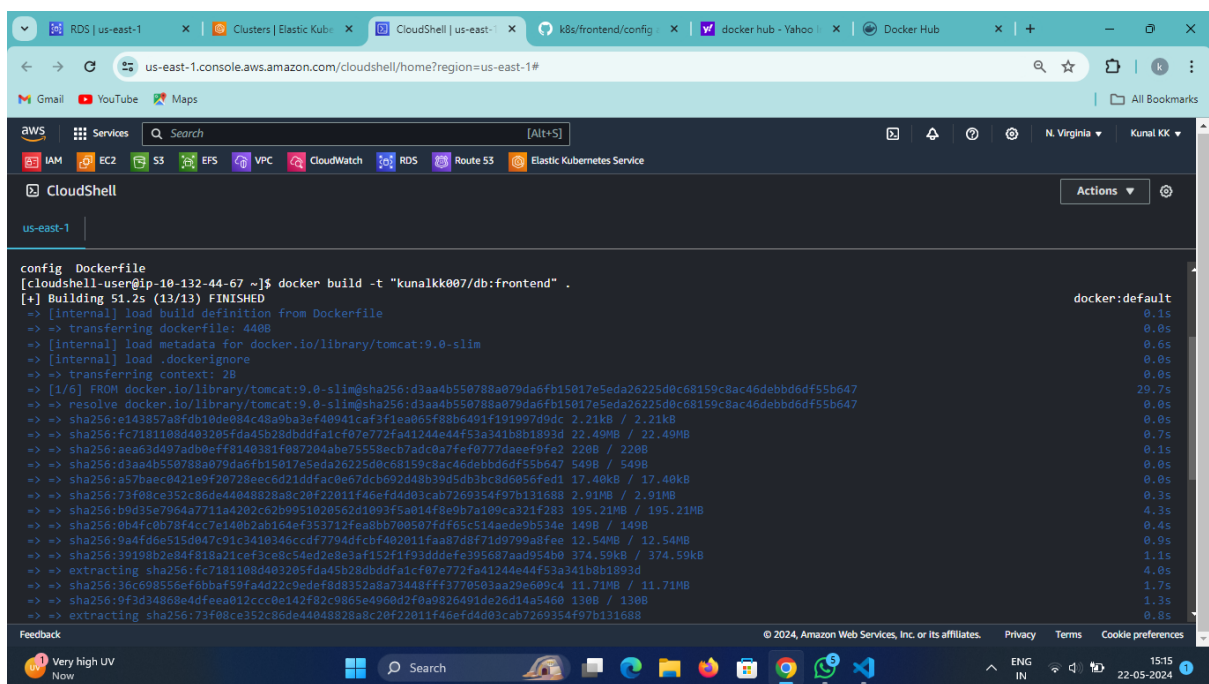
3. Then go on cloudshell and create a Dockerfile and Config file for frontend.
4. Just insert the endpoint of DB instance in Config file.



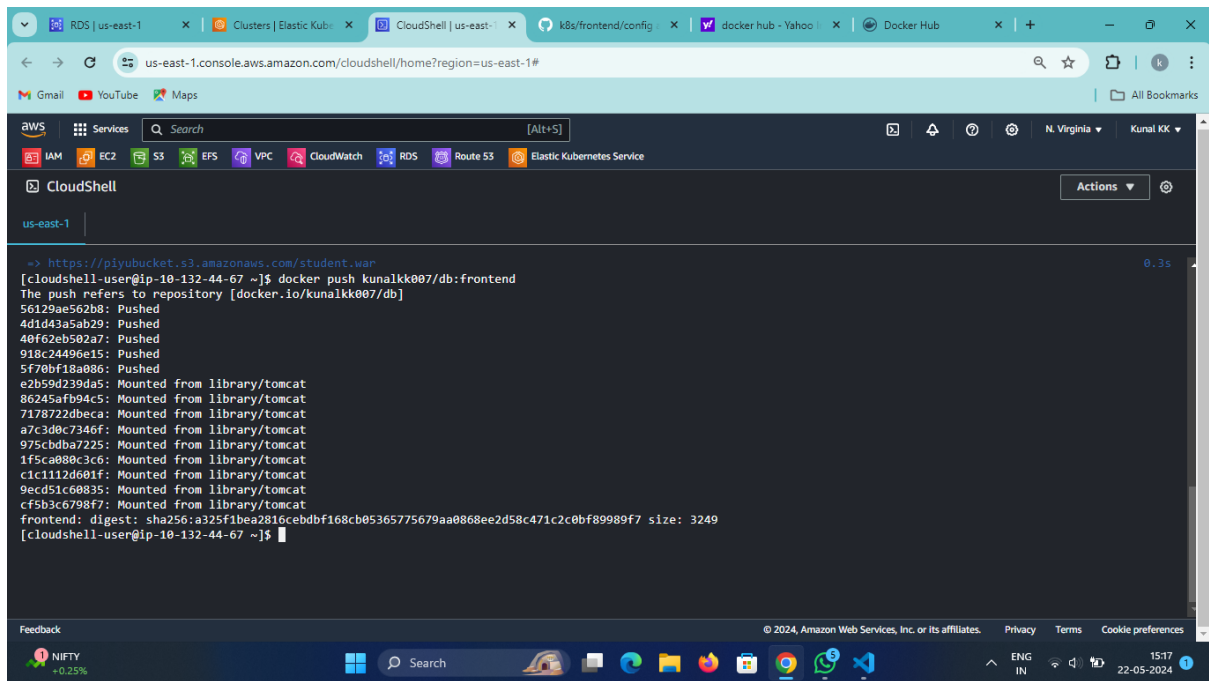
5. After that open docker hub and create a repository.



6. create the docker containers using Dockerfile using following command.
docker build -t "kunalkk007/db:frontend" .



7. Upload Docker containers in Docker hub
8. Note: we need to configure username and password with docker login command...
docker push kunalkk007/db:frontend



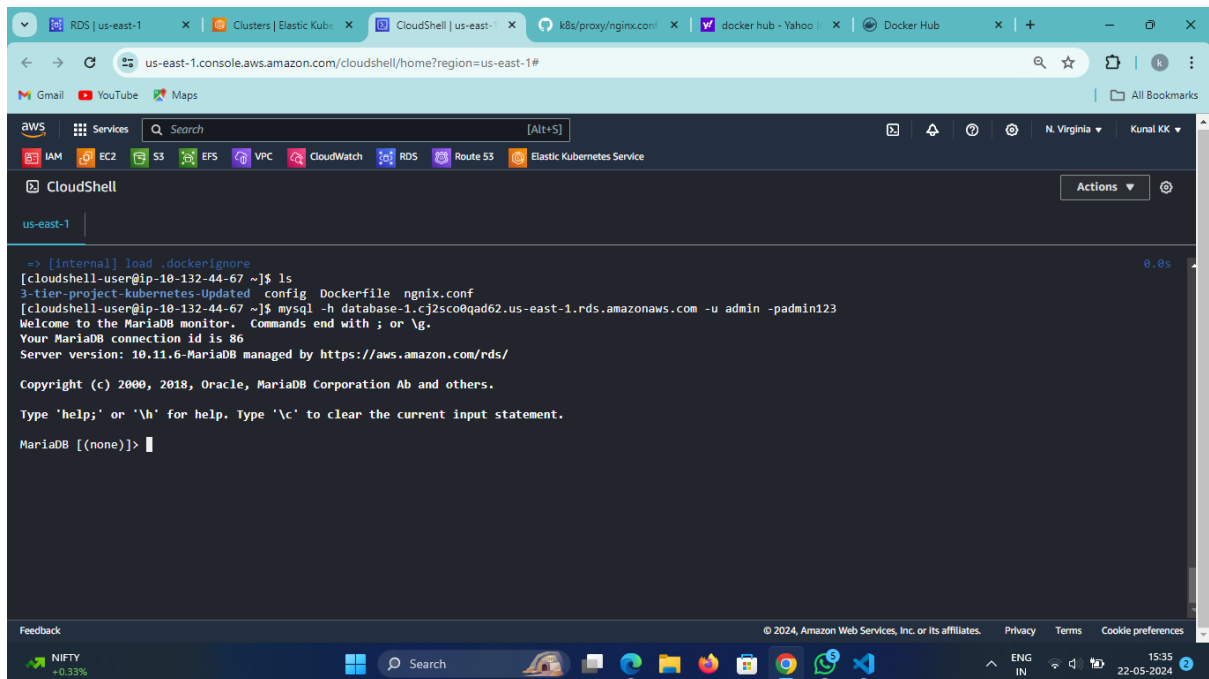
The screenshot shows the AWS CloudShell interface in a web browser. The terminal window displays the command `docker push kunalkk007/db:frontend` and its output, which includes a list of layers being pushed and their SHA256 hashes, followed by the digest and size of the final image. The terminal output is as follows:

```
-> https://piyubucket.s3.amazonaws.com/student.war
[cloudshell-user@ip-10-132-44-67 ~]$ docker push kunalkk007/db:frontend
The push refers to repository [docker.io/kunalkk007/db]
56129ae562b8: Pushed
4d1d43a5ab29: Pushed
40f62eb502a7: Pushed
918c24496e15: Pushed
5f70bf18a086: Pushed
e2b59d239da5: Mounted from library/tomcat
86245afb94c5: Mounted from library/tomcat
7178722dbeca: Mounted from library/tomcat
a7c3d0c7346f: Mounted from library/tomcat
975cbdba7225: Mounted from library/tomcat
1f5ca080c3c0: Mounted from library/tomcat
c1c112d081f: Mounted from library/tomcat
9ec451c60835: Mounted from library/tomcat
cf5b3c6798f7: Mounted from library/tomcat
frontend: digest: sha256:a325f1bea2816cebdf168cb05365775679aa0868ee2d58c471c2c0bf89989f7 size: 3249
[cloudshell-user@ip-10-132-44-67 ~]$
```

9. Create Schema in Amazon RDS database...

- Use below commands for connecting with RDS database

mysql -h <endpoint_of_dbinstanc> -u admin -padmin1234



The screenshot shows the AWS CloudShell interface in a web browser. The terminal window displays the command `mysql -h database-1.cj2sco8qad62.us-east-1.rds.amazonaws.com -u admin -padmin123` and its output, which shows the connection to the MariaDB database. The terminal output is as follows:

```
-> [internal] load .dockerignore
[cloudshell-user@ip-10-132-44-67 ~]$ ls
3-tier-project-kubernetes-Updated config Dockerfile nginx.conf
[cloudshell-user@ip-10-132-44-67 ~]$ mysql -h database-1.cj2sco8qad62.us-east-1.rds.amazonaws.com -u admin -padmin123
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 86
Server version: 10.11.6-MariaDB managed by https://aws.amazon.com/rds/

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

10. Add schema in database.

The screenshot shows the AWS CloudShell interface in the us-east-1 region. The terminal displays the following commands and output:

```
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 86
Server version: 10.11.6-MariaDB managed by https://aws.amazon.com/rds/

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE DATABASE IF NOT EXISTS studentapp;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]> USE studentapp;
Database changed
MariaDB [studentapp]> 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)
  -> );
```

The bottom of the screenshot shows a Windows taskbar with a weather widget indicating 42°C and a system clock showing 15:36 on 22-05-2024.

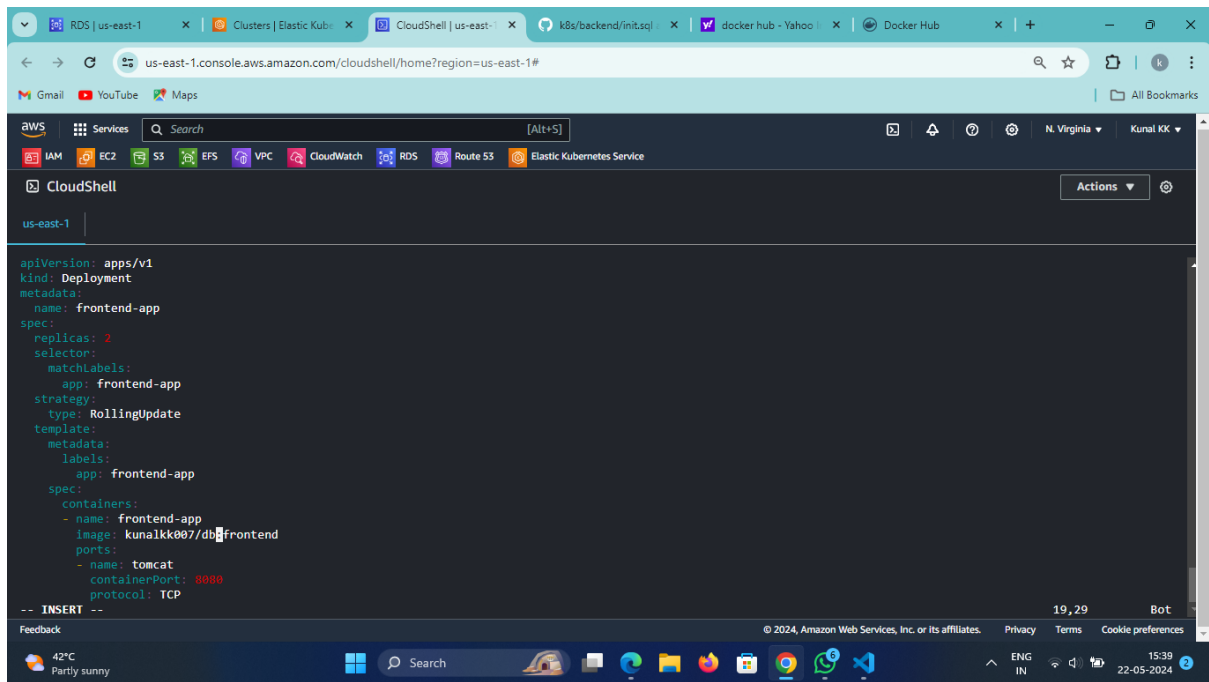
The screenshot shows the AWS CloudShell interface in the us-east-1 region. The terminal displays the following command and output:

```
MariaDB [studentapp]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| innodb |
| mysql |
| performance_schema |
| studentapp |
| sys |
+-----+
6 rows in set (0.001 sec)

MariaDB [studentapp]>
```

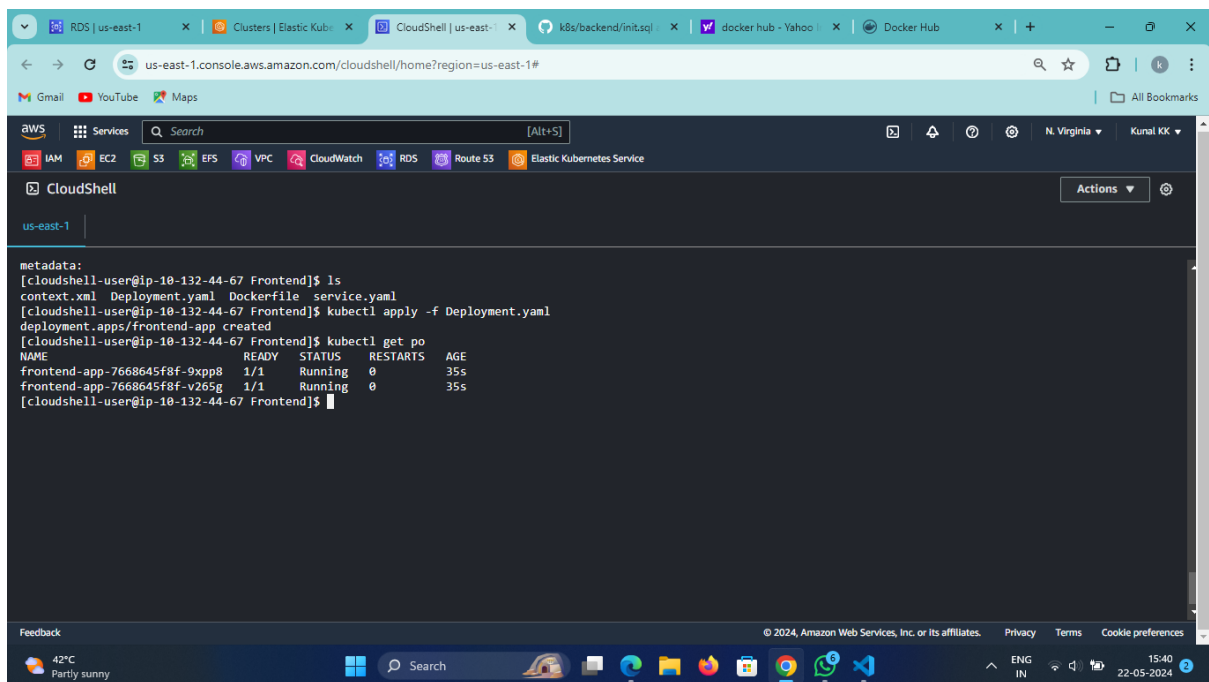
The bottom of the screenshot shows a Windows taskbar with a weather widget indicating 42°C and a system clock showing 15:37 on 22-05-2024.

11. Successfully created studentapp database.
12. Write a Deployment.yaml files for Deploying application in Kubernetes.
13. Deployment.yaml File for Frontend.



The screenshot shows the AWS CloudShell interface in a web browser. The terminal displays a Kubernetes Deployment manifest for 'frontend-app'. The manifest includes metadata, spec, replicas, selector, strategy, and template sections. The container is named 'frontend-app' and uses the image 'kunal007/dbfrontend'. The ports section shows 'tomcat' on 'containerPort: 8080' with 'protocol: TCP'. The terminal output is as follows:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: frontend-app
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: frontend-app
    spec:
      containers:
        - name: frontend-app
          image: kunalkk007/dbfrontend
          ports:
            - name: tomcat
              containerPort: 8080
              protocol: TCP
-- INSERT --
```



The screenshot shows the AWS CloudShell interface with the terminal displaying the execution of several commands. The user lists files in the current directory, applies the 'Deployment.yaml' file, and then checks the status of the pods. The terminal output is as follows:

```
[cloudshell-user@ip-10-132-44-67 Frontend]$ ls
context.xml  Deployment.yaml  Dockerfile  service.yaml
[cloudshell-user@ip-10-132-44-67 Frontend]$ kubectl apply -f Deployment.yaml
deployment.apps/frontend-app created
[cloudshell-user@ip-10-132-44-67 Frontend]$ kubectl get po
NAME                                READY   STATUS    RESTARTS   AGE
frontend-app-7668645f8f-9xpp8       1/1     Running   0           35s
frontend-app-7668645f8f-v265g       1/1     Running   0           35s
[cloudshell-user@ip-10-132-44-67 Frontend]$
```

14. Write service.yaml files for Exposing The application over Internet
Service.yaml file for frontend.

apiVersion: v1

kind: Service

metadata:

name: frontend-service

spec:

selector:

app: frontend-app

ports:

- **name: http**
targetPort: 8080
port: 80

type: ClusterIP

```
apiVersion: v1
kind: Service
metadata:
  name: frontend-service
spec:
  selector:
    app: frontend-app
  ports:
  - name: http
    targetPort: 8080
    port: 80
  type: ClusterIP
```

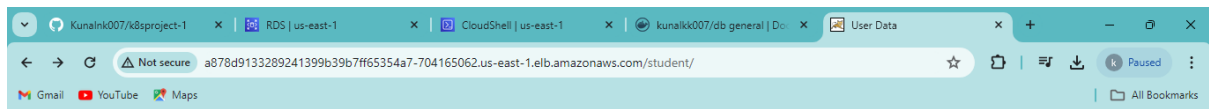
15. Deploy Deployment.yaml and Service.yaml files using below commands...

kubectl apply -f deployment.yaml (for frontend)

kubectl apply -f service.yaml (for frontend)

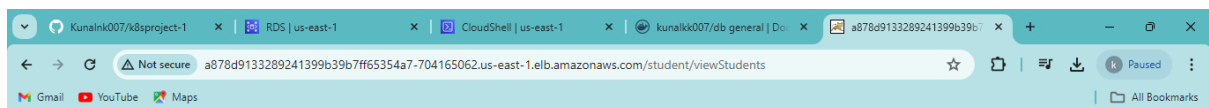
Note:- before Accessing the website with load-balancer we need to edit security group of following.....

- 1) Edit security group of RDS database (add port 3306 in inbound rule)
- 2) Edit security group of Load balancer (add port 80 in inbound rule)



Student Registration Form

Student Name	<input type="text" value="a"/>
Student Address	<input type="text" value="a"/>
Student Age	<input type="text" value="a"/>
Student Qualification	<input type="text" value="a"/>
Student Percentage	<input type="text" value="a"/>
Year Passed	<input type="text" value="a"/>
<input type="button" value="register"/>	



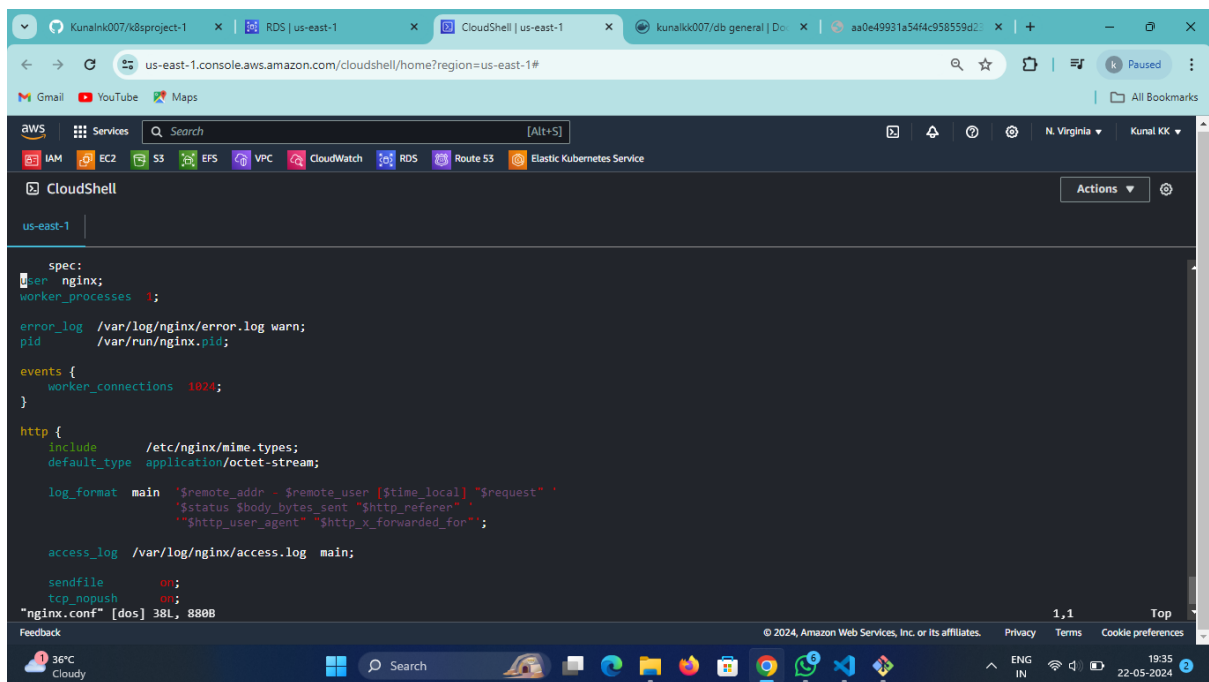
[Register Student](#)

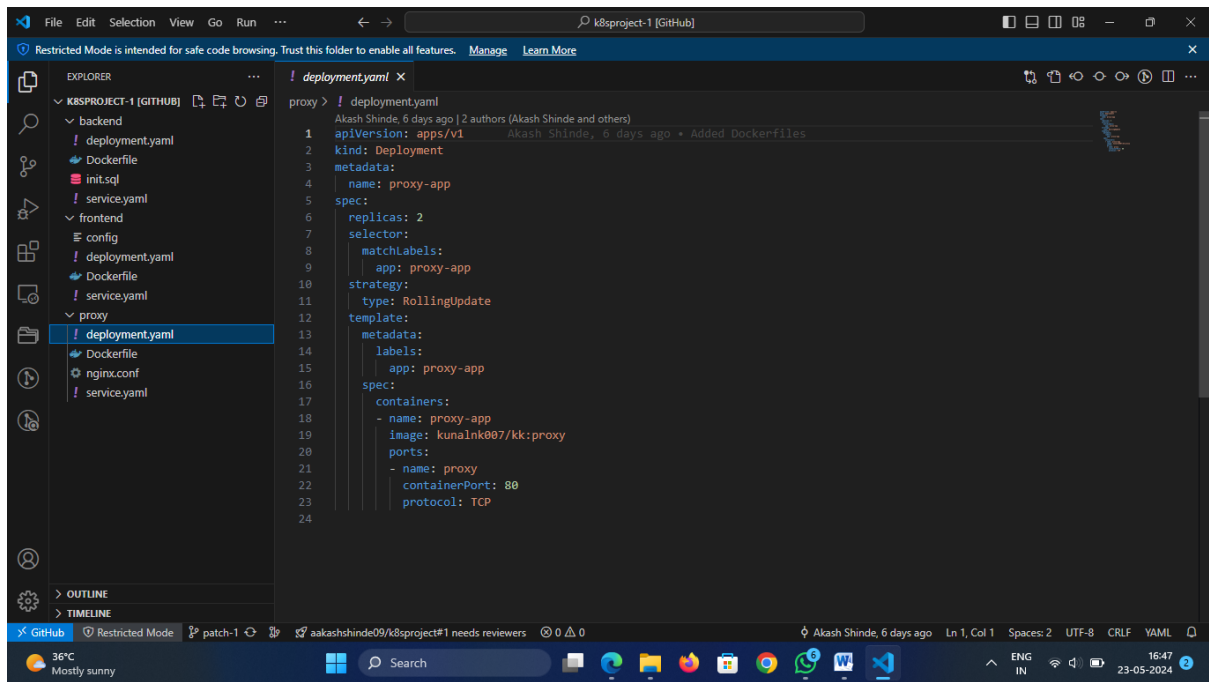
Students List

Student ID	StudentName	Student Addr	Student Age	Student Qualification	Student Percentage	Student Year Passed	Edit	Delete
1	a	a	a	a	a	a	edit	delete
2	a	a	a	a	a	a	edit	delete



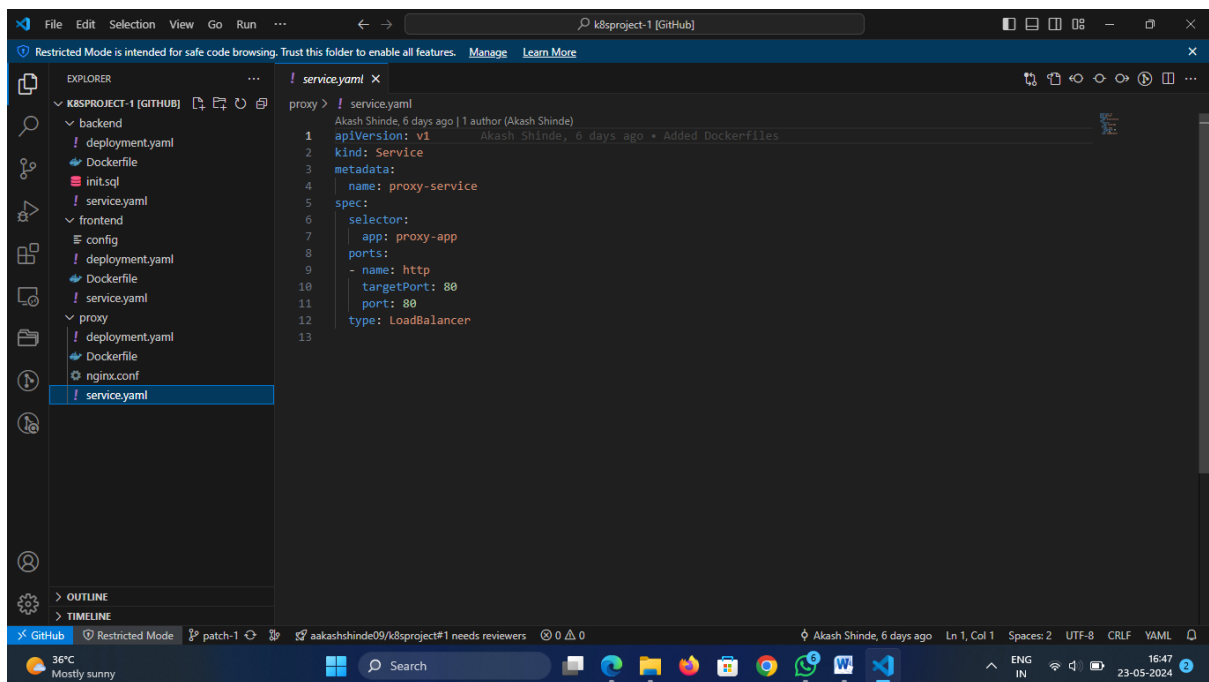
- Here our 2 tier project is completed.
- 16. Now we will try to pass the proxy.
- For that we need a Dockerfile and nginx.conf for our proxy.





The screenshot shows the Visual Studio Code editor with the file explorer on the left. The 'proxy' directory is expanded, showing 'deployment.yaml', 'Dockerfile', 'nginx.conf', and 'service.yaml'. The 'deployment.yaml' file is selected and its content is displayed in the main editor. The file is a Kubernetes Deployment manifest for a proxy application.

```
1 proxy > ! deployment.yaml
2 Akash Shinde, 6 days ago | 2 authors (Akash Shinde and others)
3 apiVersion: apps/v1
4 kind: Deployment
5 metadata:
6   name: proxy-app
7 spec:
8   replicas: 2
9   selector:
10    matchLabels:
11     app: proxy-app
12 strategy:
13   type: RollingUpdate
14 template:
15   metadata:
16     labels:
17     app: proxy-app
18   spec:
19     containers:
20     - name: proxy-app
21       image: kunalnk007/kk:proxy
22     ports:
23     - name: proxy
24       containerPort: 80
25       protocol: TCP
```



The screenshot shows the Visual Studio Code editor with the file explorer on the left. The 'proxy' directory is expanded, showing 'deployment.yaml', 'Dockerfile', 'nginx.conf', and 'service.yaml'. The 'service.yaml' file is selected and its content is displayed in the main editor. The file is a Kubernetes Service manifest for a proxy service.

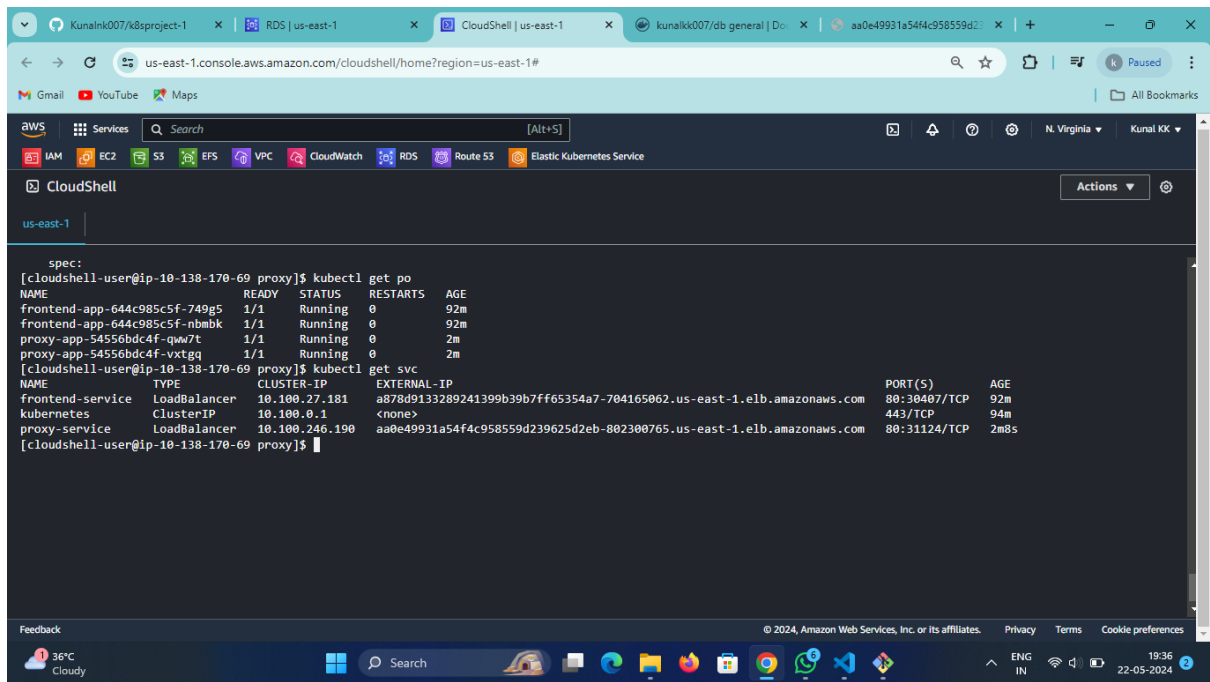
```
1 proxy > ! service.yaml
2 Akash Shinde, 6 days ago | 1 author (Akash Shinde)
3 apiVersion: v1
4 kind: Service
5 metadata:
6   name: proxy-service
7 spec:
8   selector:
9     app: proxy-app
10   ports:
11   - name: http
12     targetPort: 80
13     port: 80
14   type: LoadBalancer
```

17. Deploy Deployment.yaml and Service.yaml files using below commands...

kubectl apply -f deployment.yaml (for proxy)

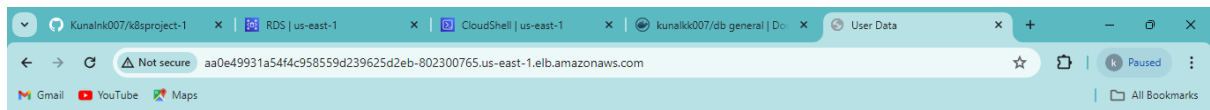
kubectl apply -f service.yaml (for proxy)

- Now hit command **# kubectl get svc** and copy the external ip of proxy-service and hit on web browser.



18. Our application hosted successfully.

- Output-1:-



Student Registration Form

Student Name	<input type="text" value="kunal"/>
Student Address	<input type="text" value="pune"/>
Student Age	<input type="text" value="20"/>
Student Qualification	<input type="text" value="12th"/>
Student Percentage	<input type="text" value="90"/>
Year Passed	<input type="text" value="2022"/>
<input type="button" value="register"/>	



Output-2:-

Successfully stored the data in RDS.