Kubernetes 3 tier Project

- Deploying a webapp in Kubernetes cluster.
- For that we need image files on DockerHub which we will use in our manifest files.
- For the application create a Dockerfile.

```
FROM tomcat:9.0-slim
WORKDIR /opt
# Set environment variables (consider injecting from outside)
ENV APP_HOME=/usr/local/tomcat
ENV PORT=8080
# Copy application WAR
ADD https://webapp2-akashapp.s3.amazonaws.com/student.war $APP_HOME/webapps/
# Copy database connector
ADD https://webapp-akash.s3.amazonaws.com/mysql-connector-j-8.3.0.jar $APP_HOME/lib
# Copy configuration
COPY config /opt
RUN sed -i '20r /opt/config' /usr/local/tomcat/conf/context.xml
EXPOSE $PORT
CMD ["catalina.sh", "run"]
```

- Now we also need a config file.
- In the config file instead of container IP we are now using the service name.

```
<Resource name="jdbc/TestDB" auth="Container" type="javax.sql.DataSource"
maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="root"
password="1234" driverClassName="com.mysql.jdbc.Driver"
url="jdbc:mysql://backend-service:3306/studentapp"/>
```

- We also need a Dockerfile for our database.

```
FROM mysql
ENV MYSQL_ROOT_PASSWORD="1234"
ENV MYSQL_DATABASE="studentapp"
COPY init.sql /docker-entrypoint-initdb.d/
~
```

- We also need a init.sql file in which there is a mysql query to create a database and table.

```
CREATE DATABASE IF NOT EXISTS studentapp;
USE 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)
        );
```

- Now build the dockerfile which is created for the webapp.
- Hit command “docker build -t “webapp” . “
- This command will build the dockerfile and create a docker image.
- Now we have to run the image to create a container.
- Hit command “docker run -d webapp” this command will create a container.
- Hit command “docker ps” to check the running containers.

```
[cloudshell-user@ip-10-130-94-235 app]$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED              SIZE
webapp       latest    6bea404d8abb   About a minute ago   424MB
[cloudshell-user@ip-10-130-94-235 app]$ docker run -d webapp
1822447045c2e99e11881541d0e59c0715b407e23d1c811589f8b1903b331334
[cloudshell-user@ip-10-130-94-235 app]$ docker ps
CONTAINER ID   IMAGE     COMMAND             CREATED          STATUS         PORTS      NAMES
1822447045c2   webapp    "catalina.sh run"   10 seconds ago   Up 4 seconds   8080/tcp   affectionate_liskov
[cloudshell-user@ip-10-130-94-235 app]$
```

- Now we need to create a image from this container.
- Hit command “docker commit container_id image_name” to create a container from the image.
- Check the images if created.
- Now we need to give tag to the image to push it to the DockerHub.

```
[cloudshell-user@ip-10-130-94-235 app]$ docker run -d webapp
1822447045c2e99e11881541d0e59c0715b407e23d1c811589f8b1903b331334
[cloudshell-user@ip-10-130-94-235 app]$ docker ps
CONTAINER ID   IMAGE     COMMAND             CREATED          STATUS         PORTS      NAMES
1822447045c2   webapp    "catalina.sh run"   10 seconds ago   Up 4 seconds   8080/tcp   affectionate_liskov
[cloudshell-user@ip-10-130-94-235 app]$ docker commit 1822447045c2 studentapp
sha256:4b95e264a36bc2d816e79ca6acc18f70cff986b405a3783b58e65624d56fa461
[cloudshell-user@ip-10-130-94-235 app]$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
studentapp   latest    4b95e264a36b   30 seconds ago   424MB
webapp       latest    6bea404d8abb   5 minutes ago    424MB
[cloudshell-user@ip-10-130-94-235 app]$ docker tag studentapp aakashshinde09/project:studentapp
[cloudshell-user@ip-10-130-94-235 app]$ docker images
REPOSITORY                TAG          IMAGE ID       CREATED              SIZE
aakashshinde09/project    studentapp   4b95e264a36b   About a minute ago   424MB
studentapp                latest       4b95e264a36b   About a minute ago   424MB
webapp                    latest       6bea404d8abb   6 minutes ago        424MB
[cloudshell-user@ip-10-130-94-235 app]$
```

- Now we need to push this image to the DockerHub.
- Hit command “docker push aakashshinde09/project:studentapp”.

```
[cloudshell-user@ip-10-130-94-235 app]$ docker push aakashshinde09/project:studentapp
The push refers to repository [docker.io/aakashshinde09/project]
b874c8cd4c68: Pushed
03114570621d: Pushed
b74cf2ba0ccb: Pushed
741d275890d3: Pushed
94ab5c4fdcdc: 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
1f5ca080c3c6: Mounted from library/tomcat
c1c1112d601f: Mounted from library/tomcat
9ecd51c60835: Mounted from library/tomcat
cf5b3c6798f7: Mounted from library/tomcat
studentapp: digest: sha256:61895d528195cdefac40cd676707e8d3081db130ce79ec970abcee206b0021d4 size: 3459
[cloudshell-user@ip-10-130-94-235 app]$
```

Tags

This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
studentapp		Image	---	a few seconds ago

See all

- Now we need to build the dockerfile created for mysql.
- Hit command “docker build -t “mysql” .”
- Now hit command “docker images” to check the docker images.
- Now we need to run the image to create a container.
- Hit command “docker run -d mysql” to create a container.

```
[cloudshell-user@ip-10-138-184-155 mysql]$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
mysql        latest    b61bd43f9795   18 seconds ago   578MB
[cloudshell-user@ip-10-138-184-155 mysql]$ docker run -d -e MYSQL_ROOT_PASSWORD="1234" mysql
7bc4947638d78c5006cee5b428b6cd35e3ba5fa89919c3e878b79f209f69f2f9
[cloudshell-user@ip-10-138-184-155 mysql]$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS         PORTS                   NAMES
7bc4947638d7   mysql     "docker-entrypoint.s…"   17 seconds ago   Up 6 seconds   3306/tcp, 33060/tcp     unruffled_noether
[cloudshell-user@ip-10-138-184-155 mysql]$
```

- Now we need to create an image from container.
- Hit command “docker commit container_id image_name” to create image from container.
- Now check the docker images.
- Now we need to give tag to the image to push it.
- Hit command “docker tag database aakashshinde09/project:database” to add the tag.
- How hit “docker push aakashshinde09/project:database”.

```
[cloudshell-user@ip-10-138-184-155 mysql]$ docker commit 7bc4947638d7 database
sha256:772f727213d878b2096da08421a4b58f2ff0175bdf325bbcf8be09c8c80e3d7c
[cloudshell-user@ip-10-138-184-155 mysql]$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
database     latest    772f727213d8   4 seconds ago    578MB
mysql        latest    b61bd43f9795   2 minutes ago    578MB
[cloudshell-user@ip-10-138-184-155 mysql]$ docker tag database aakashshinde09/project:database
[cloudshell-user@ip-10-138-184-155 mysql]$ docker images
REPOSITORY                TAG        IMAGE ID       CREATED          SIZE
aakashshinde09/project    database   772f727213d8   27 seconds ago   578MB
database                  latest     772f727213d8   27 seconds ago   578MB
mysql                     latest     b61bd43f9795   2 minutes ago    578MB
[cloudshell-user@ip-10-138-184-155 mysql]$
```

- Now we need to push the image to the DockerHub.
- Hit command “docker push aakashshinde09/project:database” to push the image to the DockerHub.

```
[cloudshell-user@ip-10-138-184-155 mysql]$ docker push aakashshinde09/project:database
The push refers to repository [docker.io/aakashshinde09/project]
195cf7e69663: Pushed
1dc1426ee148: Pushed
3ea8a8e10f3b: Layer already exists
c49ae6e704bc: Layer already exists
7382813ae583: Layer already exists
32a82dd08411: Layer already exists
50a5c055665e: Layer already exists
69e69f50bf68: Layer already exists
1c1ba8184ba3: Layer already exists
4495253a80ca: Layer already exists
1f0d8422d768: Layer already exists
fb2eff4bbe4c: Layer already exists
database: digest: sha256:d240e8ad468c47b572f8b2eb19f28a76b5bfd2839c4cac0273aaf8068d68ef77 size: 2825
[cloudshell-user@ip-10-138-184-155 mysql]$
```

- Now we have both images on the DockerHub.
- Now we need a manifest file for frontend deployment and a service file to expose the deployment.
- The deployment file for the frontend is in the image below.
- In image we put our DockerHub address of the file we just pushed.

```yaml
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: aakashshinde09/project:studentapp
        ports:
        - name: tomcat
          containerPort: 8080
          protocol: TCP
```

- Now we need to apply the deployment file.
- Hit command “kubectl apply -f deployment.yaml”.

```
[cloudshell-user@ip-10-138-184-155 frontend]$ ls
config  deployment.yaml  Dockerfile  service.yaml
[cloudshell-user@ip-10-138-184-155 frontend]$ kubectl apply -f deployment.yaml
deployment.apps/frontend-app created
[cloudshell-user@ip-10-138-184-155 frontend]$ kubectl get deploy
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
frontend-app   2/2     2            2           6s
[cloudshell-user@ip-10-138-184-155 frontend]$ kubectl get po
NAME                            READY   STATUS    RESTARTS   AGE
frontend-app-65495c4c65-mp5tk   1/1     Running   0          13s
frontend-app-65495c4c65-qtpdm   1/1     Running   0          13s
[cloudshell-user@ip-10-138-184-155 frontend]$
```

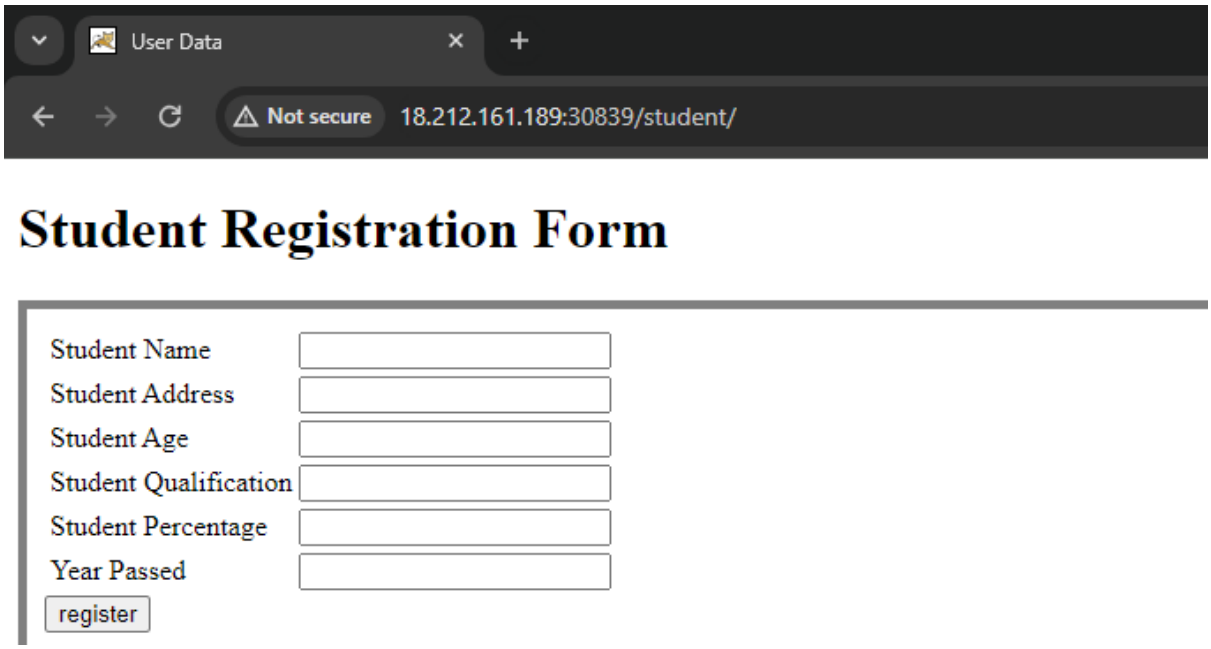
- Now we need a service file for our deployment to expose.

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

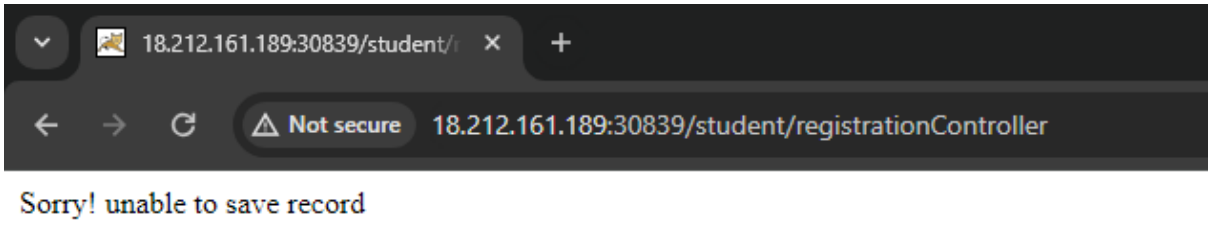
- Apply this file to expose our frontend deployment.
- Now hit command "kubectl get svc" to see the port on which our service is exposed.
- Now we need to set the inbound rule in the security groups to allow All TCP or you can specify the specific PORT which you want to allow.
- Now hit the Node IP or Instance IP with the Port Number and /student.

```
[cloudshell-user@ip-10-138-184-155 frontend]$ kubectl apply -f service.yaml
service/frontend-service created
[cloudshell-user@ip-10-138-184-155 frontend]$ kubectl get svc
NAME              TYPE           CLUSTER-IP      EXTERNAL-IP                                                              PORT(S)        AGE
frontend-service  LoadBalancer   10.100.208.58   afcebfbd073ab4e598a674b0d4903b1f-1767615844.us-east-1.elb.amazonaws.com   80:30839/TCP   8s
kubernetes        ClusterIP      10.100.0.1      <none>                                                                   443/TCP        2m47s
[cloudshell-user@ip-10-138-184-155 frontend]$ kubectl get no -o wide
NAME                           STATUS   ROLES    AGE   VERSION               INTERNAL-IP     EXTERNAL-IP     OS-IMAGE         KERNEL-VERSION
          CONTAINER-RUNTIME
ip-172-31-90-190.ec2.internal  Ready    <none>   10h   v1.29.3-eks-ae9a62a   172.31.90.190   18.212.161.189  Amazon Linux 2   5.10.215-203.850.amzn
2.x86_64   containerd://1.7.11
[cloudshell-user@ip-10-138-184-155 frontend]$
```

- Trying to hit the Instance IP with the port number and /student.
- Our page is successfully loading.



- Here we cannot save data as we have not deployed the MySQL pod for our database.



- Now we need a manifest file for our backend deployment.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: backend-app
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: backend-app
    spec:
      containers:
      - name: backend-app
        image: aakashshinde09/project:database
        ports:
        - name: java
          containerPort: 8080
          protocol: TCP
```

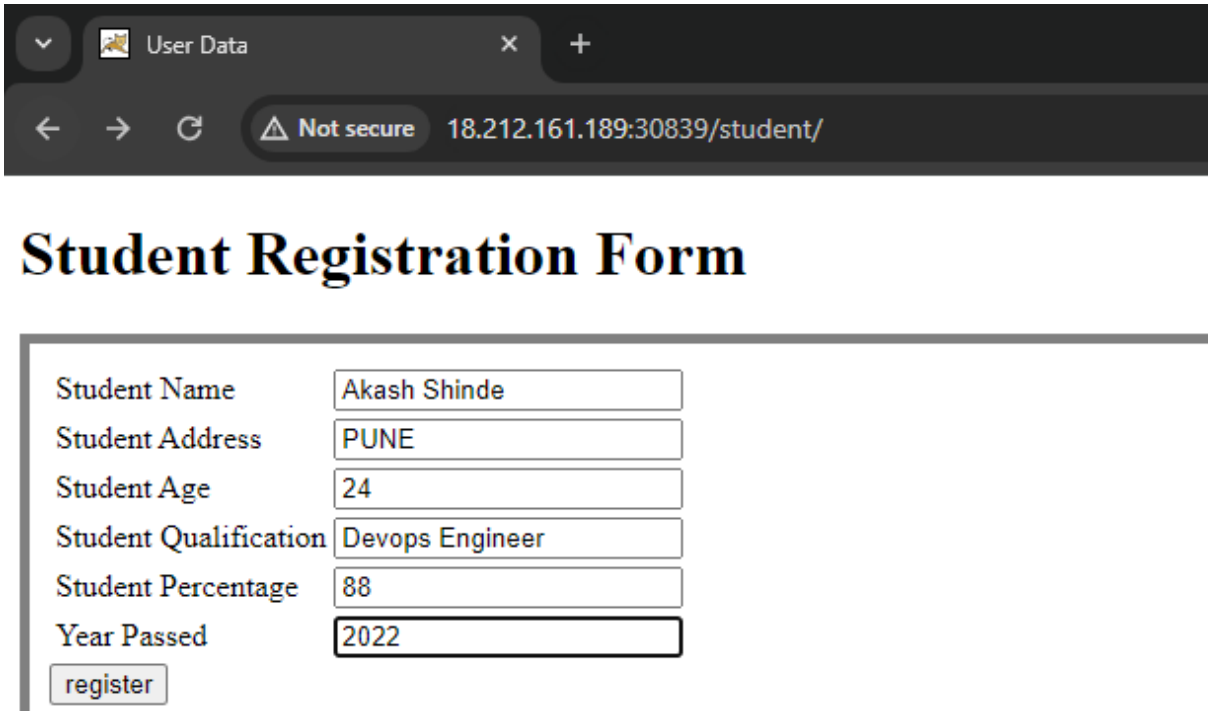
- Apply the file using command “kubectl apply -f deployment.yaml”.
- Also we need to expose the service.
- To expose the service we need a manifest.

```yaml
apiVersion: v1
kind: Service
metadata:
  name: backend-service
spec:
  selector:
    app: backend-app
  ports:
  - name: http
    targetPort: 3306
    port: 3306
  type: ClusterIP
```

- Apply the service file using command “kubectl apply -f service.yaml” this will expose the deployment on the ClusterIP.

```
[cloudshell-user@ip-10-138-184-155 backend]$ kubectl apply -f deployment.yaml
deployment.apps/backend-app created
[cloudshell-user@ip-10-138-184-155 backend]$ kubectl apply -f service.yaml
service/backend-service created
[cloudshell-user@ip-10-138-184-155 backend]$ kubectl get deploy
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
backend-app    2/2     2            2           13s
frontend-app   2/2     2            2           5m12s
[cloudshell-user@ip-10-138-184-155 backend]$ kubectl get po
NAME                            READY   STATUS    RESTARTS   AGE
backend-app-5bc9cdd96b-hcjt9    1/1     Running   0          19s
backend-app-5bc9cdd96b-lnksf    1/1     Running   0          19s
frontend-app-65495c4c65-mp5tk   1/1     Running   0          5m18s
frontend-app-65495c4c65-qtpdm   1/1     Running   0          5m18s
[cloudshell-user@ip-10-138-184-155 backend]$ kubectl get svc
NAME               TYPE           CLUSTER-IP      EXTERNAL-IP                                                              PORT(S)        AGE
backend-service    ClusterIP      10.100.94.140   <none>                                                                   3306/TCP       18s
frontend-service   LoadBalancer   10.100.208.58   afcebfbd073ab4e598a674b0d4903b1f-1767615844.us-east-1.elb.amazonaws.com   80:30839/TCP   4m33s
kubernetes         ClusterIP      10.100.0.1      <none>                                                                   443/TCP        7m12s
[cloudshell-user@ip-10-138-184-155 backend]$
```

- Now try to save the data.
- Fill the data in the form and click register.



- Our data is being sent to our database successfully.

Register Student

Students List

Student ID	StudentName	Student Addrs	Student Age	Student Qualification	Student Percentage	Student Year Passed	Edit	Delete
1	Akash Shinde	PUNE	24	Devops Engineer	88	2022	edit	delete

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

```
FROM nginx:latest

# Copy the custom configuration file to the NGINX directory
COPY nginx.conf /etc/nginx/nginx.conf

# Remove the default.conf file
RUN rm /etc/nginx/conf.d/default.conf

# Expose the necessary port (e.g., 80 for HTTP)
EXPOSE 80
```

- Nginx.conf file which we are copying.

```
user  nginx;

worker_processes  1;


error_log  /var/log/nginx/error.log warn;

pid        /var/run/nginx.pid;


events {

    worker_connections  1024;

}


http {

    include       /etc/nginx/mime.types;

    default_type  application/octet-stream;


    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '

                      '$status $body_bytes_sent "$http_referer" '

                      '"$http_user_agent" "$http_x_forwarded_for"';


    access_log  /var/log/nginx/access.log  main;
```

```
    sendfile        on;
    tcp_nopush      on;
    tcp_nodelay     on;
    keepalive_timeout  65;
    types_hash_max_size 2048;

    include /etc/nginx/conf.d/*.conf;

    server {
        listen 80;
        server_name localhost;

        location / {
            proxy_pass http://frontend-service/student/;

        }
    }
}
```

- Create a file named nginx.conf and copy the configuration to that file.
- Now we need to build the Dockerfile for the proxy to create an Docker Image.
- Hit command “Docker build -t “pro” .
- This command will create an image from the Dockerfile.
- Hit command “docker images” to check the images.
- Now we need to run the docker image to create a container.
- Hit command “docker run -d pro” to create a container.

```
[cloudshell-user@ip-10-138-184-155 proxy]$ docker images
REPOSITORY                TAG        IMAGE ID       CREATED          SIZE
pro                       latest     9acbc847a3c4   4 minutes ago    188MB
aakashshinde09/project    database   772f727213d8   34 minutes ago   578MB
database                  latest     772f727213d8   34 minutes ago   578MB
mysql                     latest     b61bd43f9795   36 minutes ago   578MB
[cloudshell-user@ip-10-138-184-155 proxy]$ docker run -d pro
5f26f627347f44dd05ad2f0d4568cdd6d2ea52e2446b5084ba84f8bbbf2768af
[cloudshell-user@ip-10-138-184-155 proxy]$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS                   NAMES
7bc4947638d7   mysql     "docker-entrypoint.s…"   35 minutes ago   Up 35 minutes   3306/tcp, 33060/tcp     unruffled_noether
[cloudshell-user@ip-10-138-184-155 proxy]$ docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS                      PORTS                   NAMES
5f26f627347f   pro       "/docker-entrypoint.…"   11 seconds ago   Exited (1) 7 seconds ago                            laughing_torvalds
7bc4947638d7   mysql     "docker-entrypoint.s…"   36 minutes ago   Up 35 minutes               3306/tcp, 33060/tcp     unruffled_noether
[cloudshell-user@ip-10-138-184-155 proxy]$
```

- Now we need to create a image from the container.
- Hit command “docker commit container_id image_name” to create an Image.
- Now we need to check if the image is created or not.
- Hit command “docker images”
- Now we need to give the tag to the image.
- Hit command “docker tag proxy aakashshinde09/project:proxy” to create a tag.

```
[cloudshell-user@ip-10-138-184-155 proxy]$ docker commit 5f26f627347f proxy
sha256:b24204b63b4c124b999f9167f2148b7b159468ae4e7b376b14627705c1473620
[cloudshell-user@ip-10-138-184-155 proxy]$ docker images
REPOSITORY                TAG        IMAGE ID       CREATED          SIZE
proxy                     latest     b24204b63b4c   8 seconds ago    188MB
pro                       latest     9acbc847a3c4   11 minutes ago   188MB
aakashshinde09/project    database   772f727213d8   40 minutes ago   578MB
database                  latest     772f727213d8   40 minutes ago   578MB
mysql                     latest     b61bd43f9795   43 minutes ago   578MB
[cloudshell-user@ip-10-138-184-155 proxy]$ docker tag proxy aakashshinde09/project:proxy
[cloudshell-user@ip-10-138-184-155 proxy]$ docker images
REPOSITORY                TAG        IMAGE ID       CREATED          SIZE
aakashshinde09/project    proxy      b24204b63b4c   59 seconds ago   188MB
proxy                     latest     b24204b63b4c   59 seconds ago   188MB
pro                       latest     9acbc847a3c4   11 minutes ago   188MB
aakashshinde09/project    database   772f727213d8   41 minutes ago   578MB
database                  latest     772f727213d8   41 minutes ago   578MB
mysql                     latest     b61bd43f9795   43 minutes ago   578MB
```

- Now we need to push the docker image to the DockerHub.
- Hit command “docker push aakashshinde09/project:proxy” to push the image.

```
[cloudshell-user@ip-10-138-184-155 proxy]$ docker push aakashshinde09/project:proxy
The push refers to repository [docker.io/aakashshinde09/project]
b291e12004b1: Pushed
800b7e18b297: Pushed
14773070094d: Mounted from library/nginx
7d2fd59c368c: Mounted from library/nginx
56f8fe6aedcd: Mounted from library/nginx
9f4d73e635f1: Mounted from library/nginx
747b290aeba8: Mounted from library/nginx
fc1cf9ca5139: Mounted from library/nginx
5d4427064ecc: Mounted from library/nginx
proxy: digest: sha256:8d03afa14844171d895f8ff08fed7e397533ca3df644dc04712305a3010c5689 size: 2192
[cloudshell-user@ip-10-138-184-155 proxy]$
```

- Our image is pushed to the DockerHub.

Tags

This repository contains 3 tag(s).

Tag	OS	Type	Pulled	Pushed
proxy		Image	---	a minute ago
database		Image	23 minutes ago	40 minutes ago
studentapp		Image	28 minutes ago	5 hours ago

- Now we need a manifest file for our proxy deployment.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: proxy-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: proxy-app
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: proxy-app
    spec:
      containers:
      - name: proxy-app
        image: aakashshinde09/project:proxy
        ports:
        - name: proxy
          containerPort: 80
          protocol: TCP
```

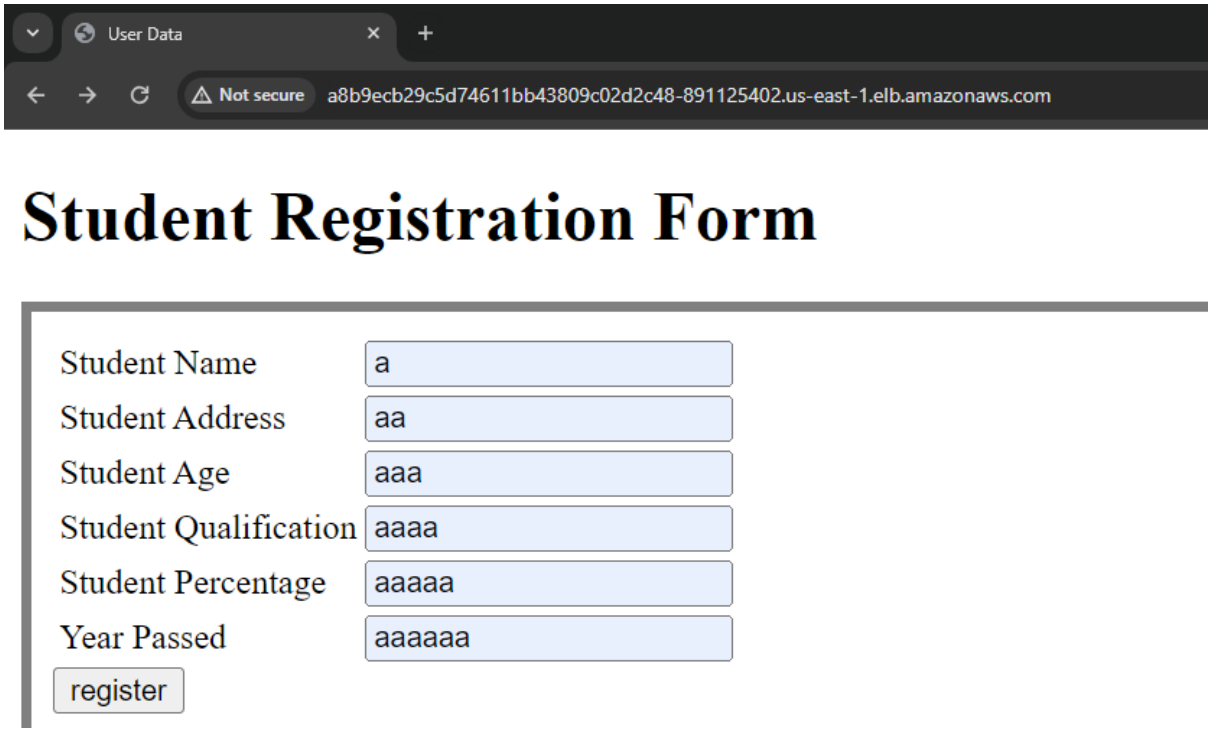
- We also need the service manifest file.

```yaml
apiVersion: v1
kind: Service
metadata:
  name: proxy-service
spec:
  selector:
    app: proxy-app
  ports:
  - name: http
    targetPort: 80
    port: 80
  type: LoadBalancer
```

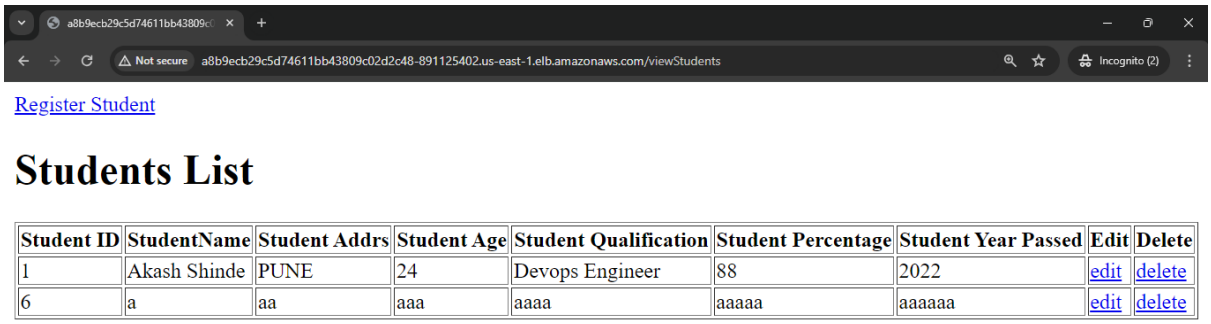
- Apply this deployment file.
- Hit command “kubectl apply -f deployment.yaml” to deploy the proxy.
- Now we need to expose the proxy service.
- Hit command “kubectl apply -f service.yaml” to expose the deployment.
- Now hit the command “kubectl get svc”.

```
[cloudshell-user@ip-10-138-184-155 proxy]$ kubectl apply -f deployment.yaml
deployment.apps/proxy-app created
[cloudshell-user@ip-10-138-184-155 proxy]$ kubectl apply -f service.yaml
service/proxy-service created
[cloudshell-user@ip-10-138-184-155 proxy]$ kubectl get svc
NAME              TYPE           CLUSTER-IP      EXTERNAL-IP                                                              PORT(S)        AGE
backend-service   ClusterIP      10.100.94.140   <none>                                                                   3306/TCP       31m
frontend-service  LoadBalancer   10.100.208.58   afcebfbd073ab4e598a674b0d4903b1f-1767615844.us-east-1.elb.amazonaws.com   80:30839/TCP   35m
kubernetes        ClusterIP      10.100.0.1      <none>                                                                   443/TCP        38m
proxy-service     LoadBalancer   10.100.59.128   a8b9ecb29c5d74611bb43809c02d2c48-891125402.us-east-1.elb.amazonaws.com   80:30328/TCP   10s
[cloudshell-user@ip-10-138-184-155 proxy]$
```

- Now hit the external IP of the service.
- Our webapp is loading successfully.



- Now try filling the data and hit register button.



Student ID	StudentName	Student Addrs	Student Age	Student Qualification	Student Percentage	Student Year Passed	Edit	Delete
1	Akash Shinde	PUNE	24	Devops Engineer	88	2022	edit	delete
6	a	aa	aaa	aaaa	aaaaa	aaaaaa	edit	delete

- Our 3 project is completed with frontend, backend and proxy.