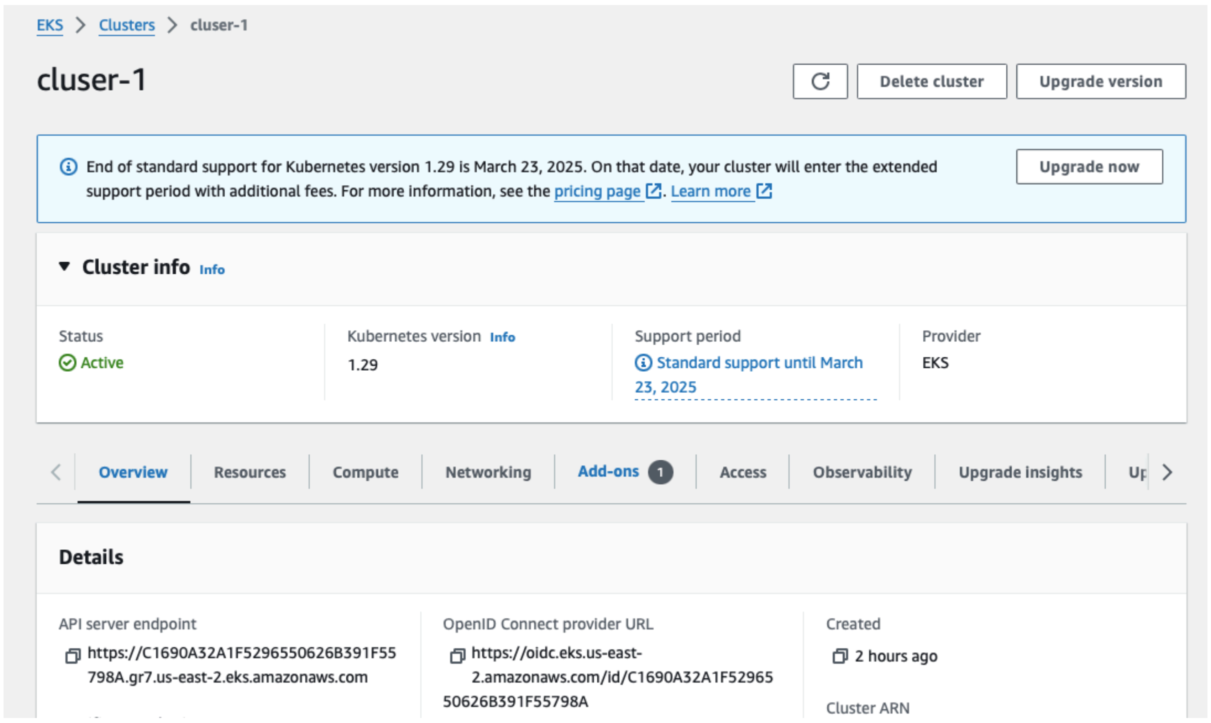
Friday, 24 May 2024

3 Tier application on Kubernetes

For 3 tier application on Kubernetes we need Kubernetes Cluster with node group and RDS database,

Here Cluster “cluser-1” is created using AWS Elastic Kubernetes Service EKS which is a managed Kubernetes service to run Kubernetes in the AWS cloud.



cluser-1

End of standard support for Kubernetes version 1.29 is March 23, 2025. On that date, your cluster will enter the extended support period with additional fees. For more information, see the pricing page. Learn more

Cluster info Info

Status	Kubernetes version Info	Support period	Provider
Active	1.29	Standard support until March 23, 2025	EKS

Details

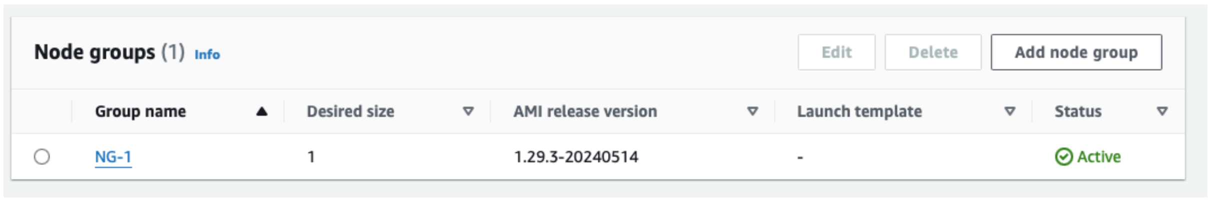
API server endpoint
https://C1690A32A1F5296550626B391F55798A.gr7.us-east-2.eks.amazonaws.com

OpenID Connect provider URL
https://oidc.eks.us-east-2.amazonaws.com/id/C1690A32A1F5296550626B391F55798A

Created
2 hours ago

Cluster ARN

In cluster we need to create a Node Group to deploy application. “NG-1” is created. Node Groups automate the provisioning and lifecycle management of nodes (Amazon EC2 instances) for Amazon EKS Kubernetes clusters.

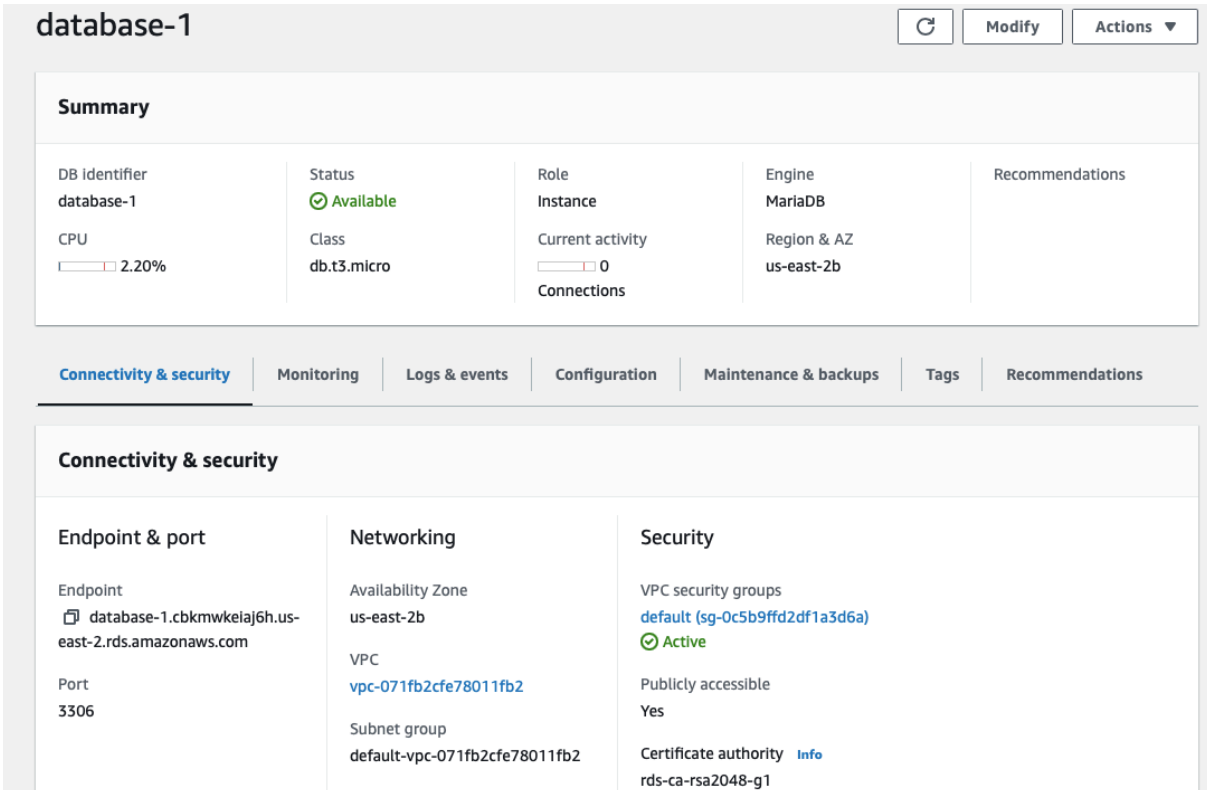


Node groups (1) Info

Group name	Desired size	AMI release version	Launch template	Status
NG-1	1	1.29.3-20240514	-	Active

For database database-1 is created by using AWS Relational Database Service RDS which is a web service that makes it easier to set up, operate, and scale a relational database in the AWS Cloud.

In this application we need to keep database publicly accessible and and set user name and password, in this case username is “admin” and password is “12345678”. Also we need to add port 3306 in security group for SQL.



database-1

Summary

DB identifier	Status	Role	Engine	Recommendations
database-1	Available	Instance	MariaDB	
CPU	Class	Current activity	Region & AZ	
2.20%	db.t3.micro	0 Connections	us-east-2b	

Connectivity & security

Endpoint & port

Endpoint
database-1.cbkmwkeiaj6h.us-east-2.rds.amazonaws.com

Port
3306

Networking

Availability Zone
us-east-2b

VPC
vpc-071fb2cfe78011fb2

Subnet group
default-vpc-071fb2cfe78011fb2

Security

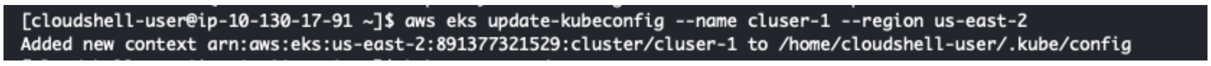
VPC security groups
default (sg-0c5b9ffd2df1a3d6a)
Active

Publicly accessible
Yes

Certificate authority Info
rds-ca-rsa2048-g1

After creating cluster and node group we need to connect to our cluster, we are connecting using AWS Cloud Shell.

#aws eks update-kubeconfig --name cluser-1 -region us-east-2



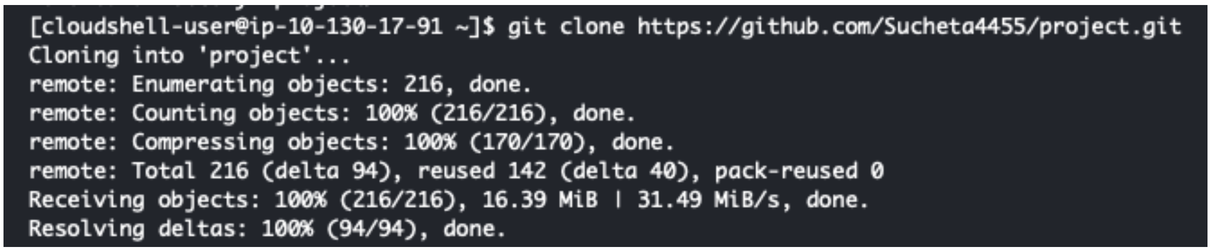
```
[cloudshell-user@ip-10-130-17-91 ~]$ aws eks update-kubeconfig --name cluser-1 --region us-east-2
Added new context arn:aws:eks:us-east-2:891377321529:cluster/cluser-1 to /home/cloudshell-user/.kube/config
```

In our application we have Frontend, Backend and Database.

we need to write Dockerfile which is a text file that contains all commands, in order, needed to build a given image, for Frontend as well as Backend. And also we need to write Kubernetes manifests are essentially files in YAML or JSON format that describe the desired state of Kubernetes API objects within the cluster. Manifest file for deployment and service.

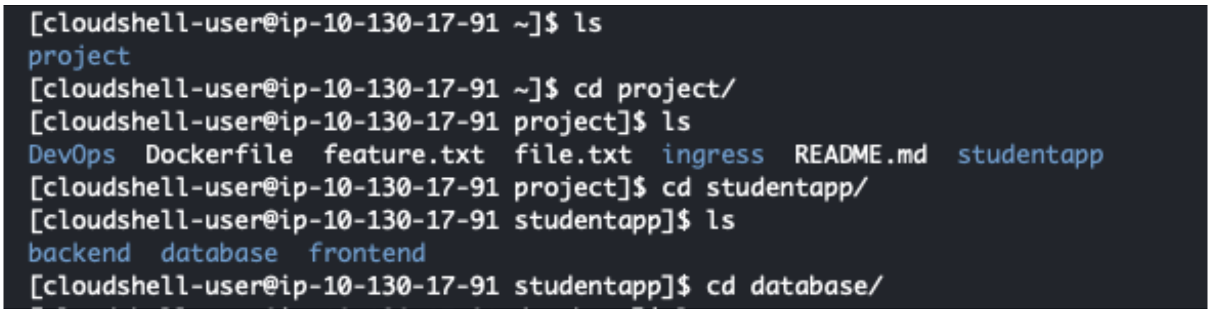
First Clone the GitHub repository with all source code and Dockerfile and Manifest file.

#git clone https://github.com/Sucheta4455/project.git



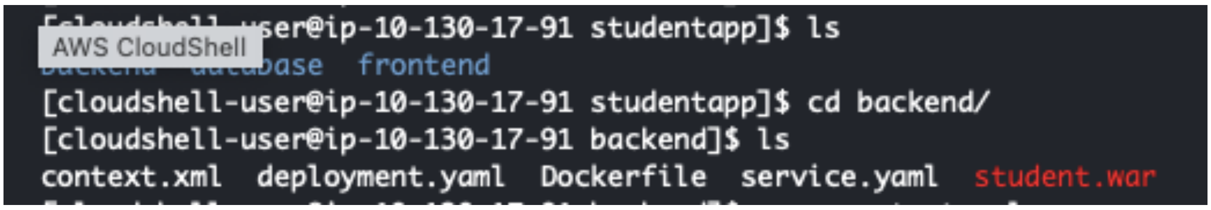
```
[cloudshell-user@ip-10-130-17-91 ~]$ git clone https://github.com/Sucheta4455/project.git
Cloning into 'project'...
remote: Enumerating objects: 216, done.
remote: Counting objects: 100% (216/216), done.
remote: Compressing objects: 100% (170/170), done.
remote: Total 216 (delta 94), reused 142 (delta 40), pack-reused 0
Receiving objects: 100% (216/216), 16.39 MiB | 31.49 MiB/s, done.
Resolving deltas: 100% (94/94), done.
```

In this Repo we have our studentapp with frontend, backend.

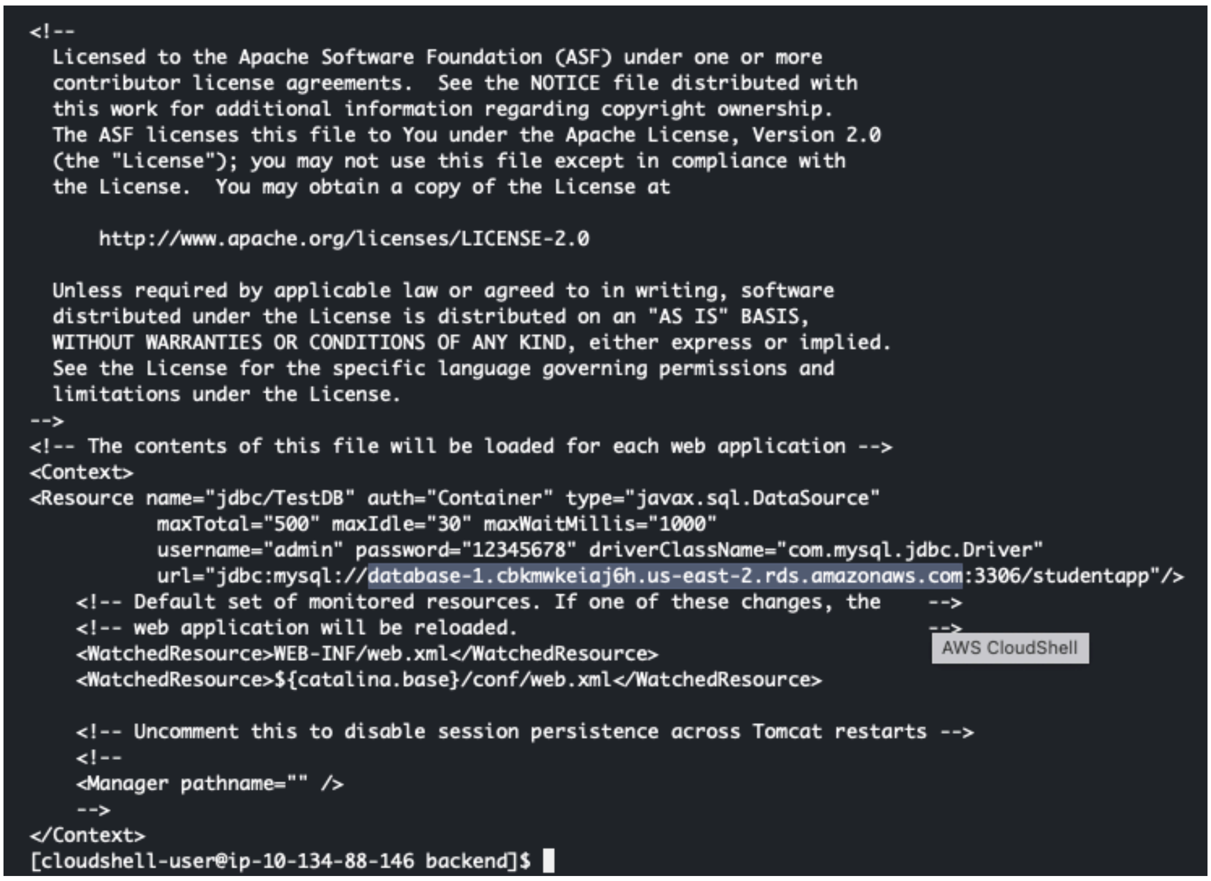


```
[cloudshell-user@ip-10-130-17-91 ~]$ ls
project
[cloudshell-user@ip-10-130-17-91 ~]$ cd project/
[cloudshell-user@ip-10-130-17-91 project]$ ls
DevOps  Dockerfile  feature.txt  file.txt  ingress  README.md  studentapp
[cloudshell-user@ip-10-130-17-91 project]$ cd studentapp/
[cloudshell-user@ip-10-130-17-91 studentapp]$ ls
backend  database  frontend
[cloudshell-user@ip-10-130-17-91 studentapp]$ cd database/
```

First we will edit context.xml file which will connect our database with backend. For this we will add end point of our RDS database in this file along with username and password.



```
[cloudshell-user@ip-10-130-17-91 studentapp]$ ls
backend  database  frontend
[cloudshell-user@ip-10-130-17-91 studentapp]$ cd backend/
[cloudshell-user@ip-10-130-17-91 backend]$ ls
context.xml  deployment.yaml  Dockerfile  service.yaml  student.war
```



```
<!--
  Licensed to the Apache Software Foundation (ASF) under one or more
  contributor license agreements.  See the NOTICE file distributed with
  this work for additional information regarding copyright ownership.
  The ASF licenses this file to You under the Apache License, Version 2.0
  (the "License"); you may not use this file except in compliance with
  the License.  You may obtain a copy of the License at

      http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.
-->
<!-- The contents of this file will be loaded for each web application -->
<Context>
<Resource name="jdbc/TestDB" auth="Container" type="javax.sql.DataSource"
           maxTotal="500" maxIdle="30" maxWaitMillis="1000"
           username="admin" password="12345678" driverClassName="com.mysql.jdbc.Driver"
           url="jdbc:mysql://database-1.cbkmwkeiaj6h.us-east-2.rds.amazonaws.com:3306/studentapp"/>
    <!-- Default set of monitored resources. If one of these changes, the    -->
    <!-- web application will be reloaded.                                   -->
    <WatchedResource>WEB-INF/web.xml</WatchedResource>
    <WatchedResource>${catalina.base}/conf/web.xml</WatchedResource>

    <!-- Uncomment this to disable session persistence across Tomcat restarts -->
    <!--
    <Manager pathname="" />
    -->
</Context>
[cloudshell-user@ip-10-134-88-146 backend]$ 
```

Then we need to configure database,

We can connect to our database using below command.

#mysql -h database-1.cbkmwkeiaj6h.us-east-2.rds.amazonaws.com -u admin -p12345678

after connecting we need to add database and come sql query.

# show databases;

#create database studentapp;

#use studentapp;

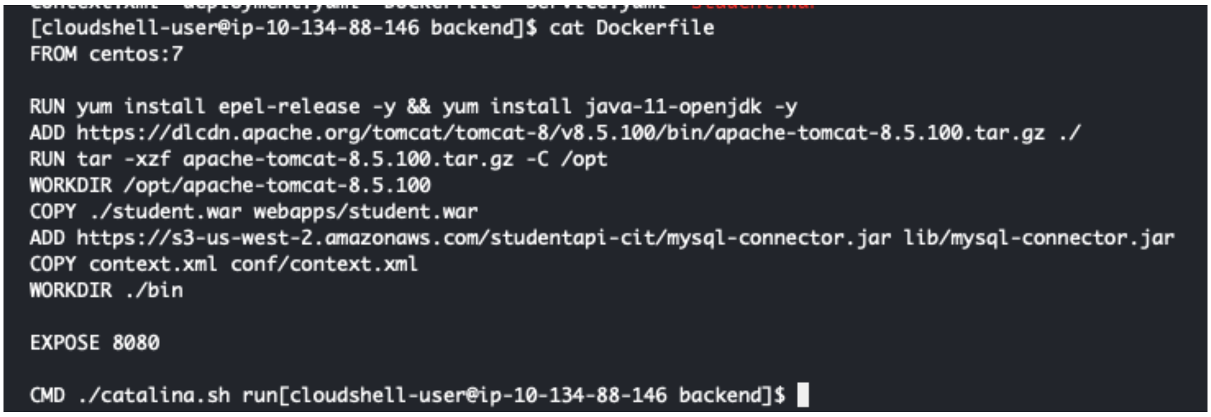
#CREATE TABLE if not exists students(student_id INT NOT NULL AUTO_INCREMENT, student_name VARCHAR(100) NOT NULL, student_addr VARCHAR(100) NOT NULL, student_age VARCHAR(3) NOT NULL, student_qual VARCHAR(20) NOT NULL, student_percent VARCHAR(10) NOT NULL, student_year_passed VARCHAR(10) NOT NULL, PRIMARY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

#describe students;

#exit

Now our database is ready.

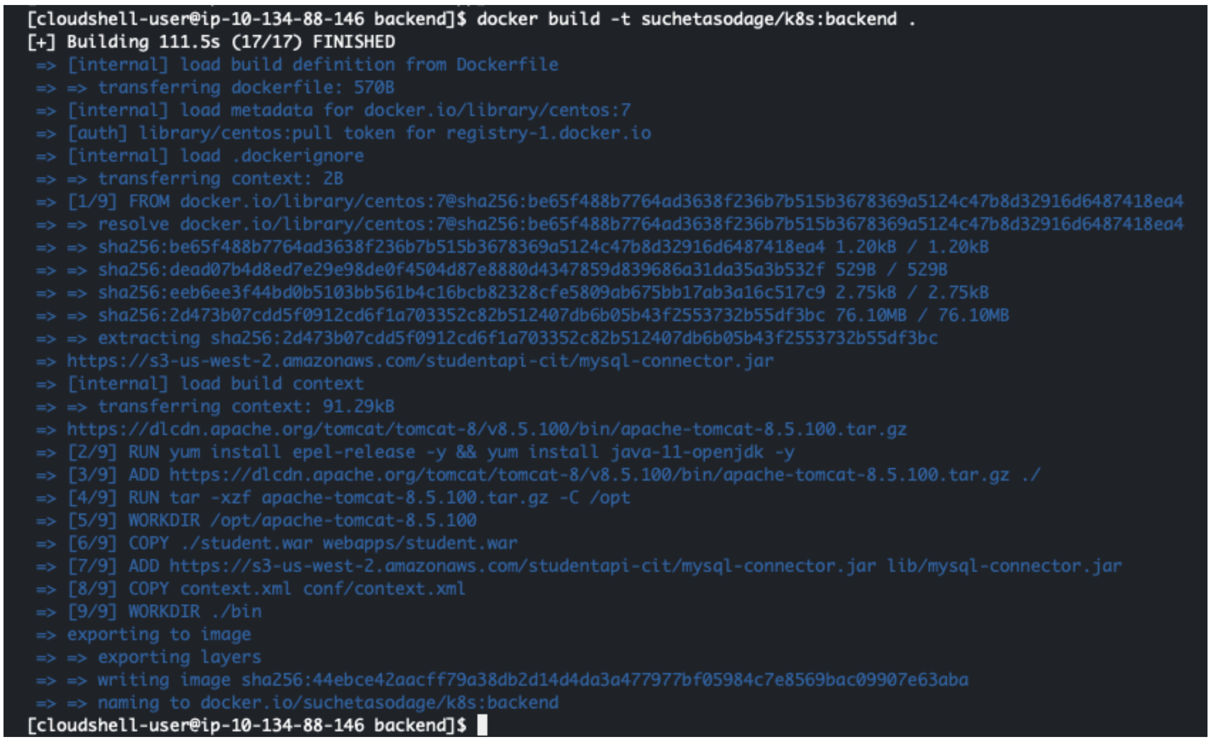
Then we will work on backend. We need first write our Dockerfile.



```
[cloudshell-user@ip-10-134-88-146 backend]$ cat Dockerfile
FROM centos:7

RUN yum install epel-release -y && yum install java-11-openjdk -y
ADD https://dlcdn.apache.org/tomcat/tomcat-8/v8.5.100/bin/apache-tomcat-8.5.100.tar.gz ./
RUN tar -xzf apache-tomcat-8.5.100.tar.gz -C /opt
WORKDIR /opt/apache-tomcat-8.5.100
COPY ./student.war webapps/student.war
ADD https://s3-us-west-2.amazonaws.com/studentapi-cit/mysql-connector.jar lib/mysql-connector.jar
COPY context.xml conf/context.xml
WORKDIR ./bin

EXPOSE 8080

CMD ./catalina.sh run[cloudshell-user@ip-10-134-88-146 backend]$
```

Then build docker image using below command.

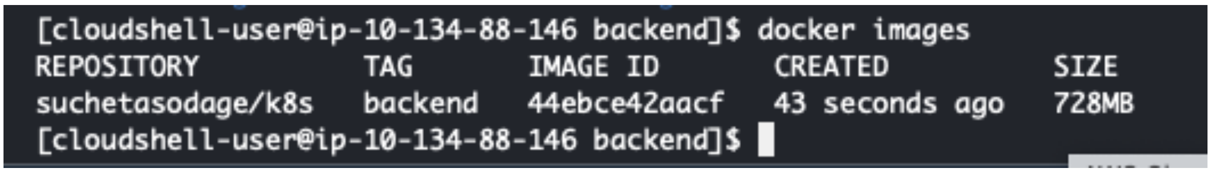
#docker build -t suchetasodage/k8s: backend .



```
[cloudshell-user@ip-10-134-88-146 backend]$ docker build -t suchetasodage/k8s:backend .
[+] Building 111.5s (17/17) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 570B
 => [internal] load metadata for docker.io/library/centos:7
 => [auth] library/centos:pull token for registry-1.docker.io
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/9] FROM docker.io/library/centos:7@sha256:be65f488b7764ad3638f236b7b515b3678369a5124c47b8d32916d6487418ea4
 => => resolve docker.io/library/centos:7@sha256:be65f488b7764ad3638f236b7b515b3678369a5124c47b8d32916d6487418ea4
 => => sha256:be65f488b7764ad3638f236b7b515b3678369a5124c47b8d32916d6487418ea4 1.20kB / 1.20kB
 => => sha256:dead07b4d8ed7e29e98de0f4504d87e8880d4347859d839686a31da35a3b532f 529B / 529B
 => => sha256:eeb6ee3f44bd0b5103bb561b4c16bcb82328cfe5809ab675bb17ab3a16c517c9 2.75kB / 2.75kB
 => => sha256:2d473b07cdd5f0912cd6f1a703352c82b512407db6b05b43f2553732b55df3bc 76.10MB / 76.10MB
 => => extracting sha256:2d473b07cdd5f0912cd6f1a703352c82b512407db6b05b43f2553732b55df3bc
 => https://s3-us-west-2.amazonaws.com/studentapi-cit/mysql-connector.jar
 => [internal] load build context
 => => transferring context: 91.29kB
 => https://dlcdn.apache.org/tomcat/tomcat-8/v8.5.100/bin/apache-tomcat-8.5.100.tar.gz
 => [2/9] RUN yum install epel-release -y && yum install java-11-openjdk -y
 => [3/9] ADD https://dlcdn.apache.org/tomcat/tomcat-8/v8.5.100/bin/apache-tomcat-8.5.100.tar.gz ./
 => [4/9] RUN tar -xzf apache-tomcat-8.5.100.tar.gz -C /opt
 => [5/9] WORKDIR /opt/apache-tomcat-8.5.100
 => [6/9] COPY ./student.war webapps/student.war
 => [7/9] ADD https://s3-us-west-2.amazonaws.com/studentapi-cit/mysql-connector.jar lib/mysql-connector.jar
 => [8/9] COPY context.xml conf/context.xml
 => [9/9] WORKDIR ./bin
 => exporting to image
 => => exporting layers
 => => writing image sha256:44ebce42aacff79a38db2d14d4da3a477977bf05984c7e8569bac09907e63aba
 => => naming to docker.io/suchetasodage/k8s:backend
[cloudshell-user@ip-10-134-88-146 backend]$
```

we can our images by using;

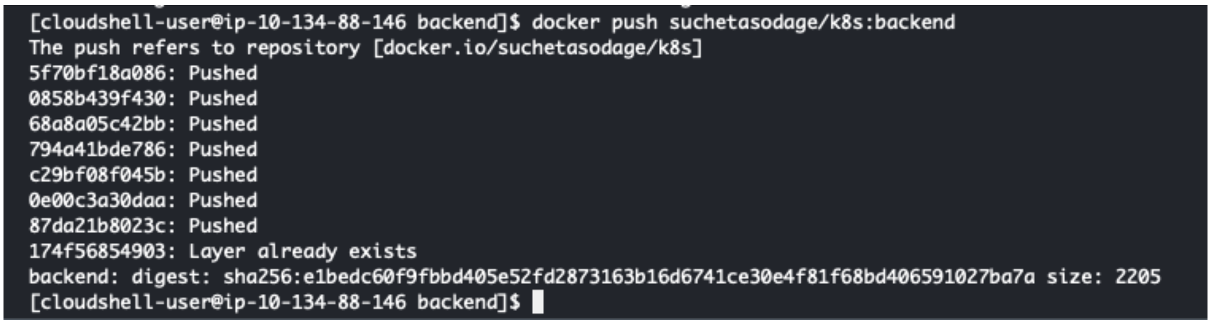
#docker images



```
[cloudshell-user@ip-10-134-88-146 backend]$ docker images
REPOSITORY          TAG       IMAGE ID       CREATED          SIZE
suchetasodage/k8s   backend   44ebce42aacf   43 seconds ago   728MB
[cloudshell-user@ip-10-134-88-146 backend]$
```

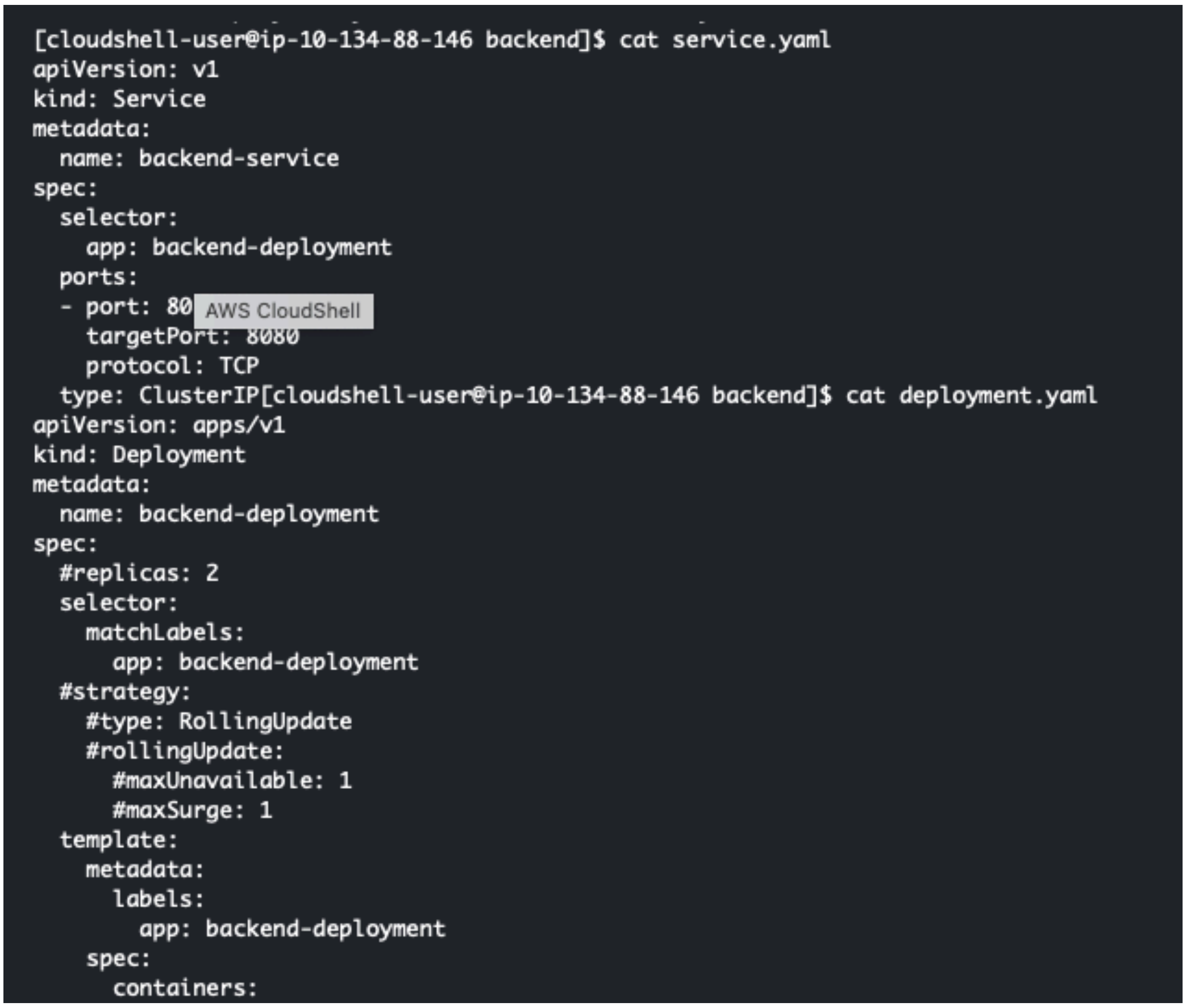
After build we will push the same image to DockerHub registry.

#docker push suchetasodage/k8s: backend



```
[cloudshell-user@ip-10-134-88-146 backend]$ docker push suchetasodage/k8s:backend
The push refers to repository [docker.io/suchetasodage/k8s]
5f70bf18a086: Pushed
0858b439f430: Pushed
68a8a05c42bb: Pushed
794a41bde786: Pushed
c29bf08f045b: Pushed
0e00c3a30daa: Pushed
87da21b8023c: Pushed
174f56854903: Layer already exists
backend: digest: sha256:e1bedc60f9fbbd405e52fd2873163b16d6741ce30e4f81f68bd406591027ba7a size: 2205
[cloudshell-user@ip-10-134-88-146 backend]$ 
```

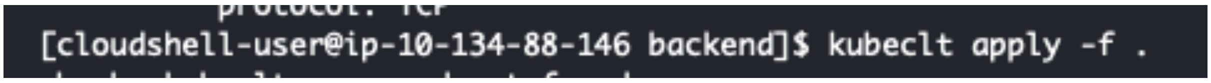
Then we will deploy our backend with the help of deployment by using docker image we have build and pushed to dockerhub along with that we need to create a service ClusterIP to internally communication with backend.



```
[cloudshell-user@ip-10-134-88-146 backend]$ cat service.yaml
apiVersion: v1
kind: Service
metadata:
  name: backend-service
spec:
  selector:
    app: backend-deployment
  ports:
  - port: 80
    targetPort: 8080
    protocol: TCP
  type: ClusterIP[cloudshell-user@ip-10-134-88-146 backend]$ cat deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend-deployment
spec:
  #replicas: 2
  selector:
    matchLabels:
      app: backend-deployment
  #strategy:
    #type: RollingUpdate
    #rollingUpdate:
      #maxUnavailable: 1
      #maxSurge: 1
  template:
    metadata:
      labels:
        app: backend-deployment
    spec:
      containers:
```

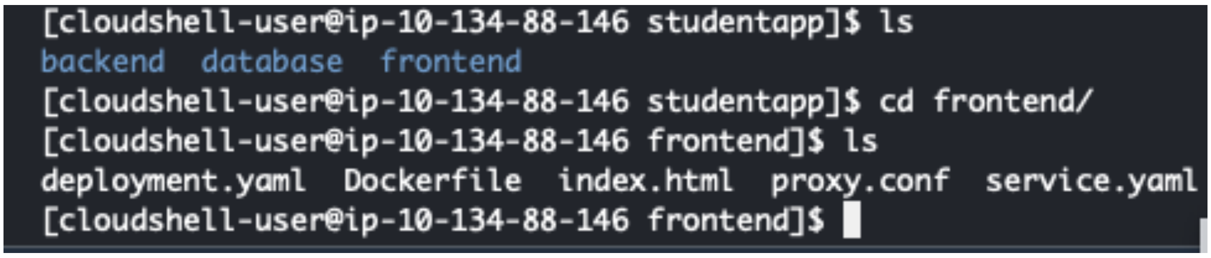
Then we will apply this deployment and service using kubectl command.

#kubeclt apply -f



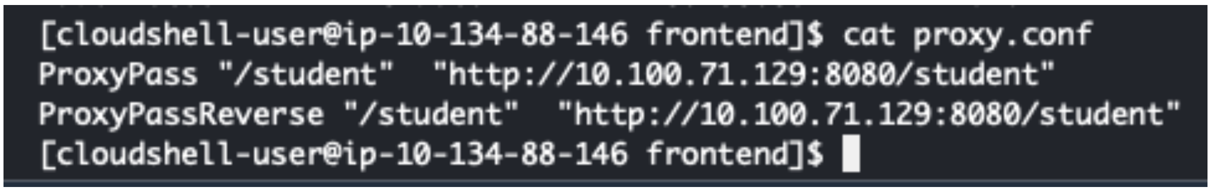
```
[cloudshell-user@ip-10-134-88-146 backend]$ kubeclt apply -f .
```

Now we will move to frontend.



```
[cloudshell-user@ip-10-134-88-146 studentapp]$ ls
backend  database  frontend
[cloudshell-user@ip-10-134-88-146 studentapp]$ cd frontend/
[cloudshell-user@ip-10-134-88-146 frontend]$ ls
deployment.yaml  Dockerfile  index.html  proxy.conf  service.yaml
[cloudshell-user@ip-10-134-88-146 frontend]$
```

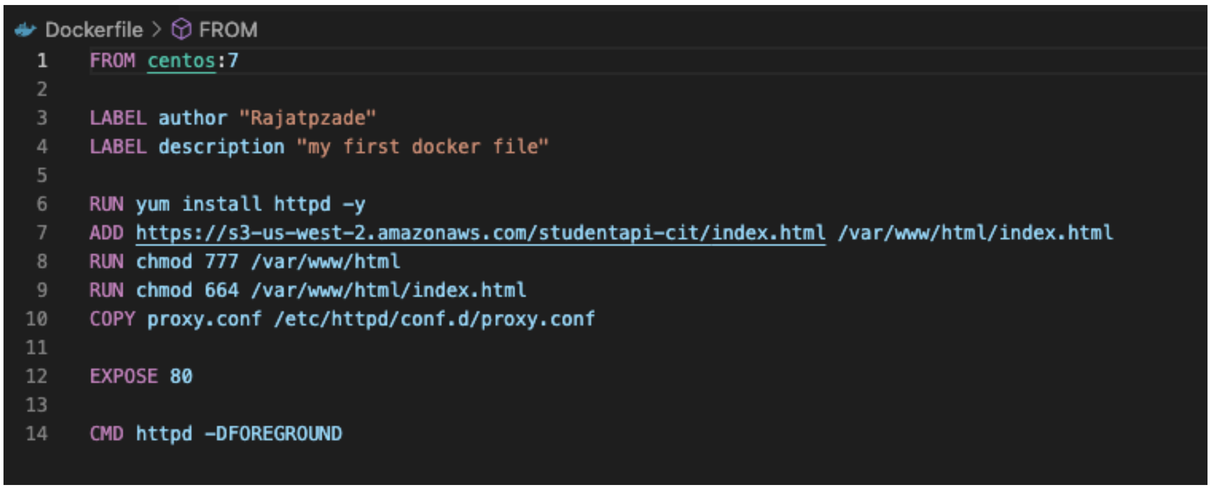
We will get IP of our backend service. that need to mention in proxy.conf file of frontend. To redirect our frontend page to backend.



```
[cloudshell-user@ip-10-134-88-146 frontend]$ cat proxy.conf
ProxyPass "/student"  "http://10.100.71.129:8080/student"
ProxyPassReverse "/student"  "http://10.100.71.129:8080/student"
[cloudshell-user@ip-10-134-88-146 frontend]$
```

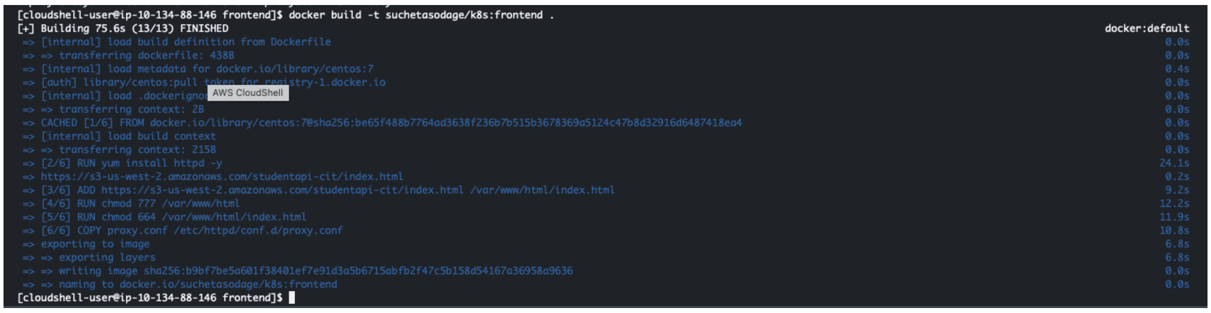
We will build our docker image.

#docker build -t suchetasodage/k8s:frontend .



Dockerfile > FROM

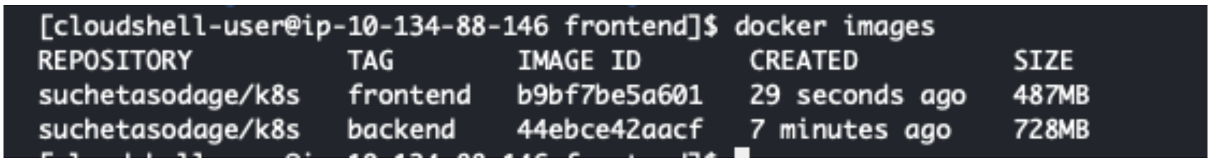
```
FROM centos:7

LABEL author "Rajatpzade"
LABEL description "my first docker file"

RUN yum install httpd -y
ADD https://s3-us-west-2.amazonaws.com/studentapi-cit/index.html /var/www/html/index.html
RUN chmod 777 /var/www/html
RUN chmod 664 /var/www/html/index.html
COPY proxy.conf /etc/httpd/conf.d/proxy.conf

EXPOSE 80

CMD httpd -DFOREGROUND
```



```
[cloudshell-user@ip-10-134-88-146 frontend]$ docker build -t suchetasodage/k8s:frontend .
[+] Building 75.6s (13/13) FINISHED                                                                docker:default
 => [internal] load build definition from Dockerfile                                                         0.0s
 => => transferring dockerfile: 438B                                                                         0.0s
 => [internal] load metadata for docker.io/library/centos:7                                                  0.4s
 => [auth] library/centos:pull token for registry-1.docker.io                                                0.0s
 => [internal] load .dockerignor                                                                             0.0s
 => => transferring context: 2B                                                                              0.0s
 => CACHED [1/6] FROM docker.io/library/centos:7@sha256:be65f488b7764ad3638f236b7b515b3678369a5124c47b8d32916d6487418ea4  0.0s
 => [internal] load build context                                                                            0.0s
 => => transferring context: 215B                                                                            0.0s
 => [2/6] RUN yum install httpd -y                                                                          24.1s
 => https://s3-us-west-2.amazonaws.com/studentapi-cit/index.html                                             0.2s
 => [3/6] ADD https://s3-us-west-2.amazonaws.com/studentapi-cit/index.html /var/www/html/index.html          9.2s
 => [4/6] RUN chmod 777 /var/www/html                                                                       12.2s
 => [5/6] RUN chmod 664 /var/www/html/index.html                                                            11.9s
 => [6/6] COPY proxy.conf /etc/httpd/conf.d/proxy.conf                                                      10.8s
 => exporting to image                                                                                       6.8s
 => => exporting layers                                                                                      6.8s
 => => writing image sha256:b9bf7be5a601f38401ef7e91d3a5b6715abfb2f47c5b158d54167a36958a9636                 0.0s
 => => naming to docker.io/suchetasodage/k8s:frontend                                                        0.0s
[cloudshell-user@ip-10-134-88-146 frontend]$
```

We can check docker images using

#docker images



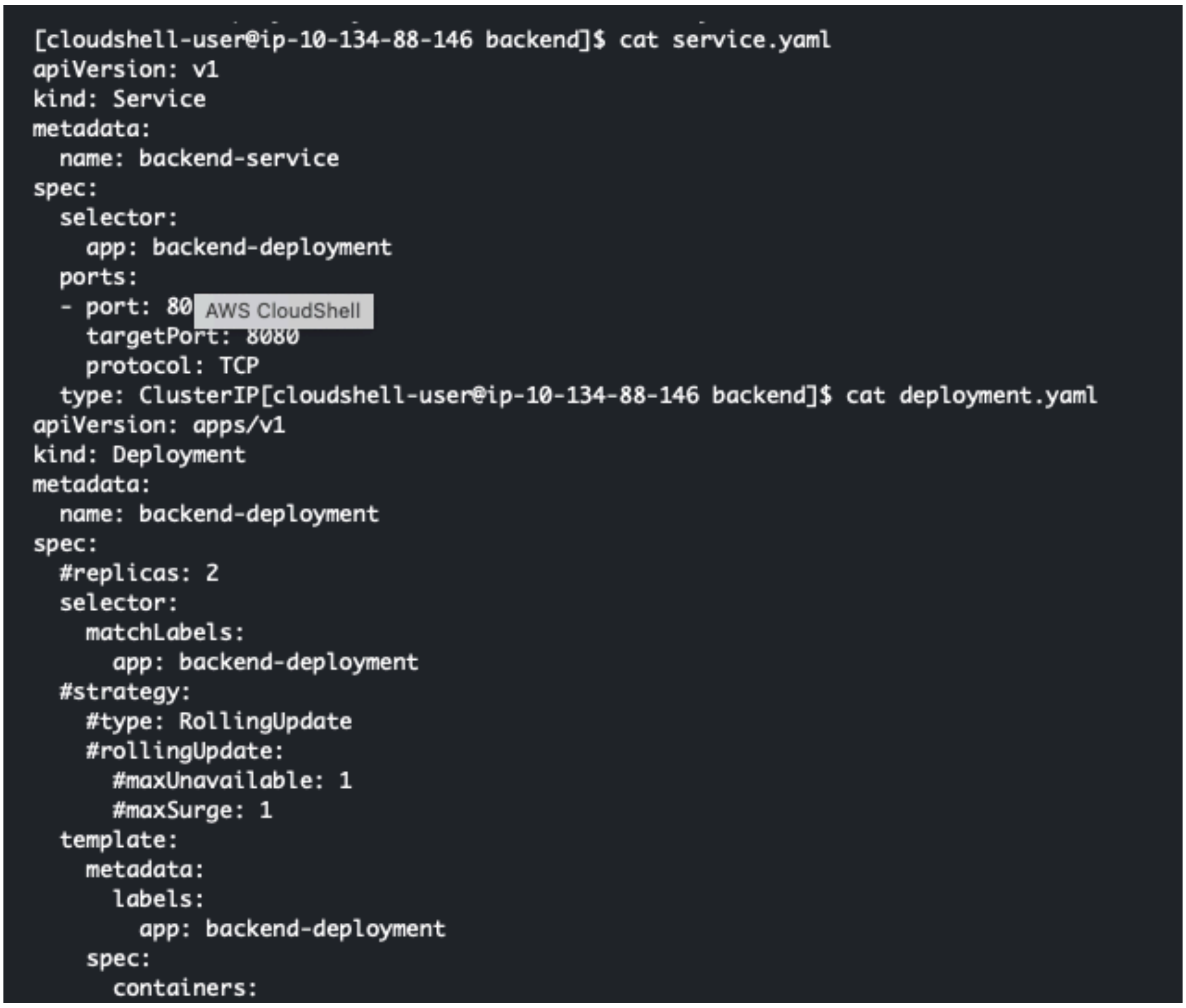
```
[cloudshell-user@ip-10-134-88-146 frontend]$ docker images
REPOSITORY          TAG        IMAGE ID       CREATED          SIZE
suchetasodage/k8s   frontend   b9bf7be5a601   29 seconds ago   487MB
suchetasodage/k8s   backend    44ebce42aacf   7 minutes ago    728MB
```

Push docker image to DockerHub

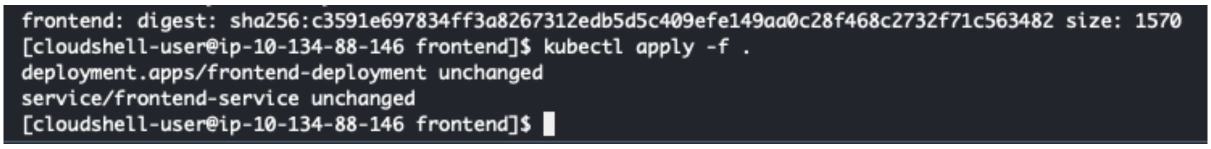
#docker push suchetasodage/k8s:frontend

Next we will apply deployment and service Load Balancer to communicate with outside world using kubectl command.

#kubectl apply -f .



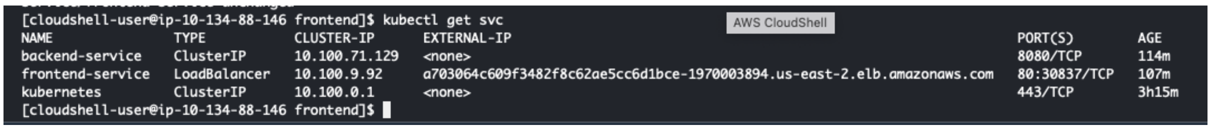
```
[cloudshell-user@ip-10-134-88-146 backend]$ cat service.yaml
apiVersion: v1
kind: Service
metadata:
  name: backend-service
spec:
  selector:
    app: backend-deployment
  ports:
  - port: 80
    targetPort: 8080
    protocol: TCP
  type: ClusterIP[cloudshell-user@ip-10-134-88-146 backend]$ cat deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend-deployment
spec:
  #replicas: 2
  selector:
    matchLabels:
      app: backend-deployment
  #strategy:
    #type: RollingUpdate
    #rollingUpdate:
      #maxUnavailable: 1
      #maxSurge: 1
  template:
    metadata:
      labels:
        app: backend-deployment
    spec:
      containers:
```



```
frontend: digest: sha256:c3591e697834ff3a8267312edb5d5c409efe149aa0c28f468c2732f71c563482 size: 1570
[cloudshell-user@ip-10-134-88-146 frontend]$ kubectl apply -f .
deployment.apps/frontend-deployment unchanged
service/frontend-service unchanged
[cloudshell-user@ip-10-134-88-146 frontend]$
```

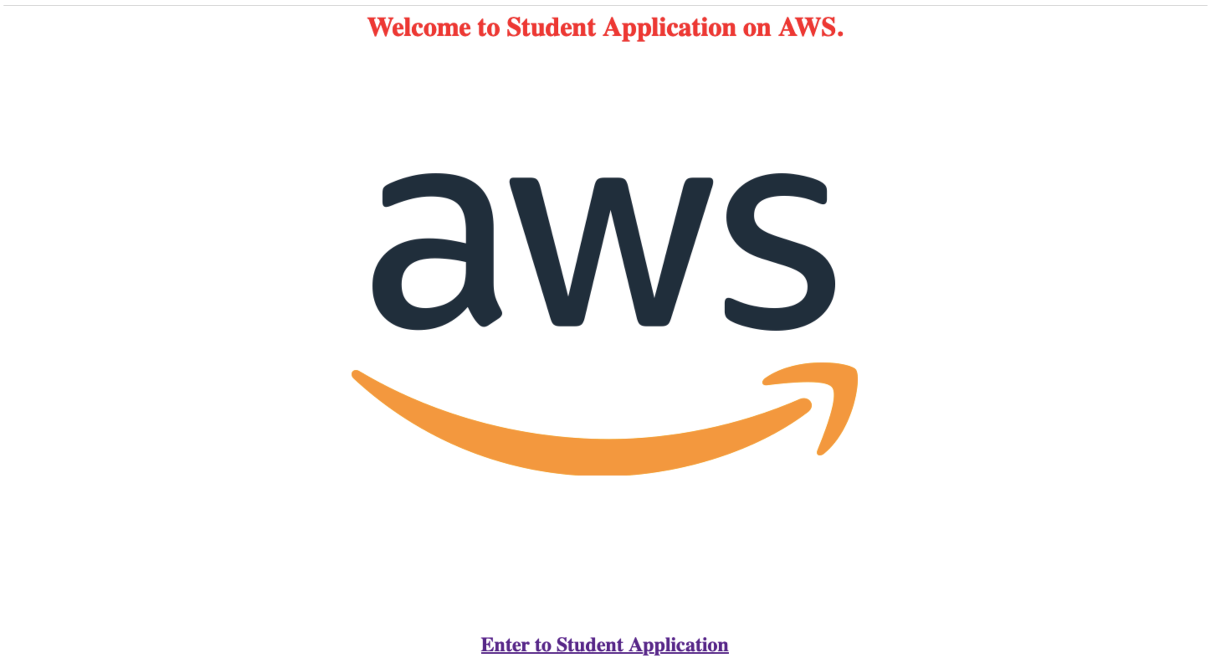
We can check Kubernetes service by

#kubectl get svc



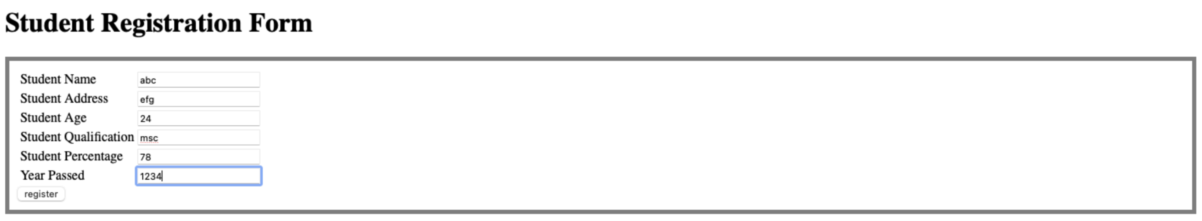
```
[cloudshell-user@ip-10-134-88-146 frontend]$ kubectl get svc
NAME               TYPE           CLUSTER-IP      EXTERNAL-IP                                                                PORT(S)        AGE
backend-service    ClusterIP      10.100.71.129   <none>                                                                     8080/TCP       114m
frontend-service   LoadBalancer   10.100.9.92     a703064c609f3482f8c62ae5cc6d1bce-1970003894.us-east-2.elb.amazonaws.com   80:30837/TCP   107m
kubernetes         ClusterIP      10.100.0.1      <none>                                                                     443/TCP        3h15m
[cloudshell-user@ip-10-134-88-146 frontend]$
```

We can access our application using LoadBalancer end point.



Welcome to Student Application on AWS.

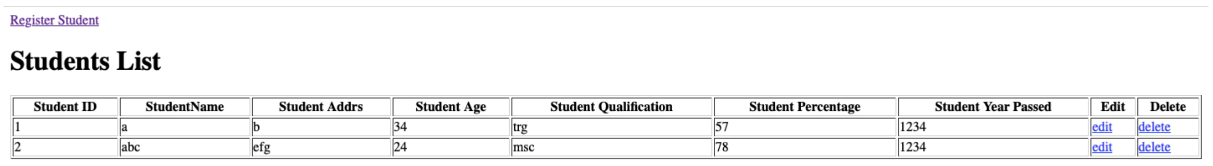
Enter to Student Application



Student Registration Form

Student Name	abc
Student Address	efg
Student Age	24
Student Qualification	msc
Student Percentage	78
Year Passed	1234

register



Register Student

Students List

Student ID	StudentName	Student Addrs	Student Age	Student Qualification	Student Percentage	Student Year Passed	Edit	Delete
1	a	b	34	trg	57	1234	edit	delete
2	abc	efg	24	msc	78	1234	edit	delete