**Problem Statement:**

1. Create a CD pipeline for given code and deploy it on your local.

you should be able to show same output which was shown during demo of the sample code run.

(localhost:8787 this endpoint shows you database data from table employee in database demo, localhost:8787/hello this endpoint returns system date)

1. While creating image you might need to consider modules/libraries used in project
2. (optional) Deploy project on cloud
3. Get an email to your mail id when this deployment is successful
4. (optional) use some monitoring tool to display any data related to this project.

Steps for this Project

1. Create your webservice.js file which will connect to mysql DB, in connection pool use environment variables instead of hardcoding the server’s name.

A screen shot of a computer program

Description automatically generated

1. Create a Packages.Json and mention all dependencies which are mentioned in webserver.js file.

A computer screen shot of a program code

Description automatically generated

1. Create a Dockerfile and mention all required step like below.

A screenshot of a computer program

Description automatically generated

1. (Optional)Build the docker file, tag the image, push to docker hub. This is just for testing.

|  |
| --- |
| docker build -t capstone-image devops-capstone-project .  docker tag capstone-image satyabrataswain/devops-capstone-project:v1  docker push satyabrataswain/devops-capstone-project:v1 |

1. Validate from Docker hub, if your image is pushed or not.

A screenshot of a computer

Description automatically generated

1. Create a docker-compose file to define webservice.js and mysql db

A screen shot of a computer

Description automatically generated

1. (Optional to test in local)Run the docker compose to validate the site in local

|  |
| --- |
| docker-compose up -d |

1. Login to mysql container to create the DB, table and insert records.

|  |
| --- |
| docker exec -it mysqlserver bash  mysql -u root -p  create database demo;  use demo;  CREATE TABLE IF NOT EXISTS employee (      empid int NOT NULL,      ename VARCHAR(255) NOT NULL,      salary int NOT NULL  );  INSERT INTO employee (empid, ename, salary) VALUES      (1,'John P', 99000),      (2,'Jonathan Smith', 75000),      (3, 'Emily Johnson', 50000); |

1. Validate both containers are running or not

|  |
| --- |
| docker ps -a |

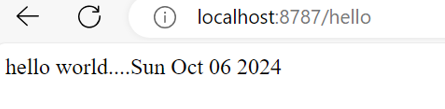
1. Browse the site

|  |
| --- |
| http://localhost:8787/  http://localhost:8787/hello |

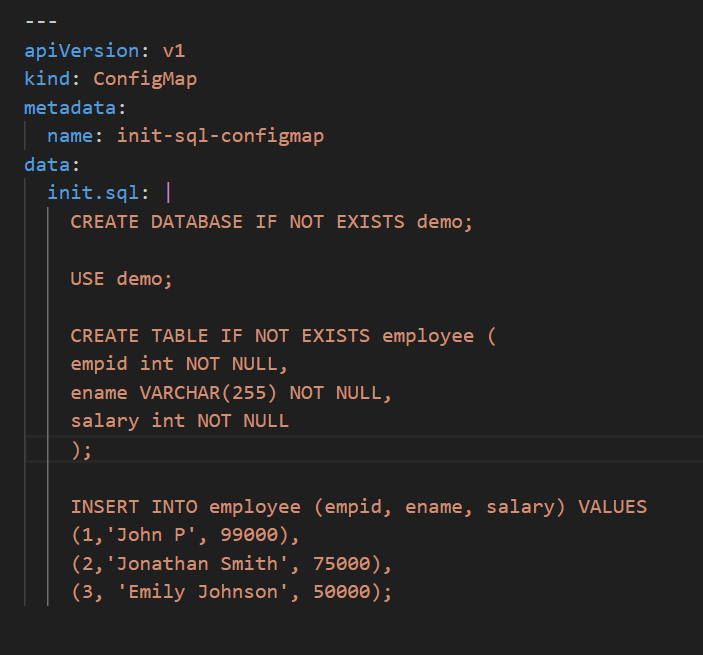
1. We can see details like below.

A screenshot of a computer code

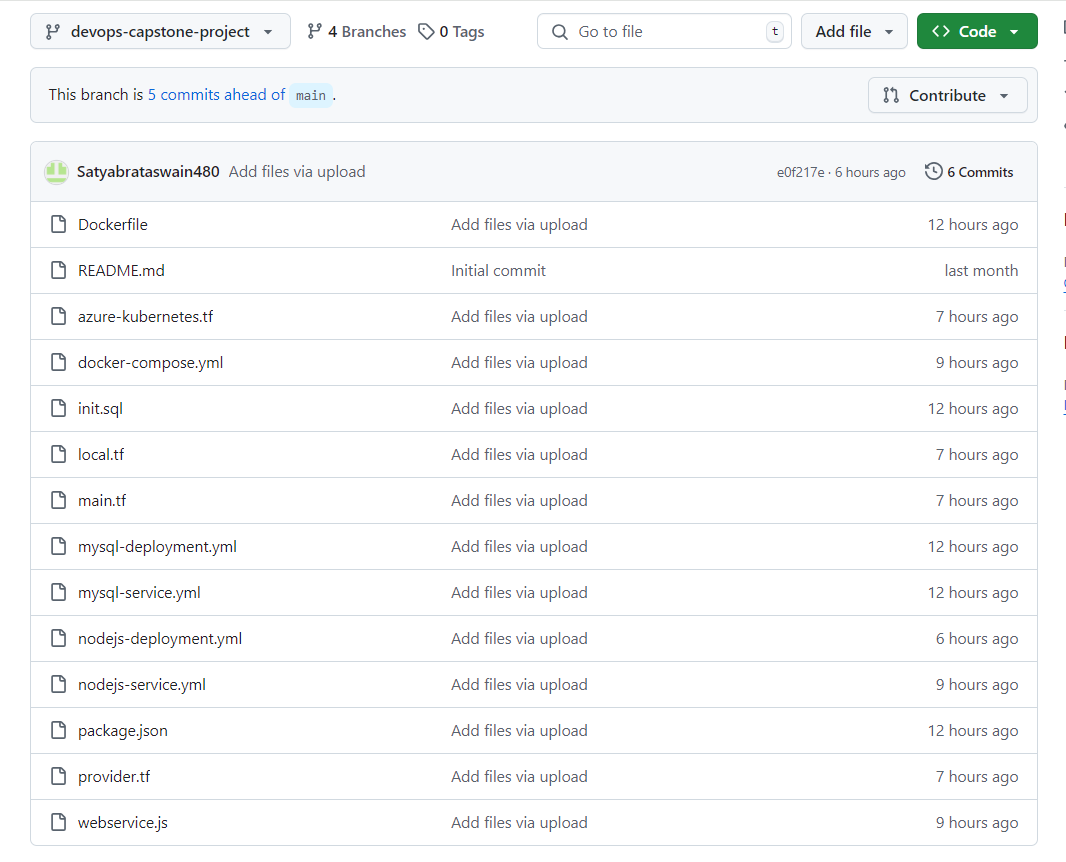
Description automatically generated



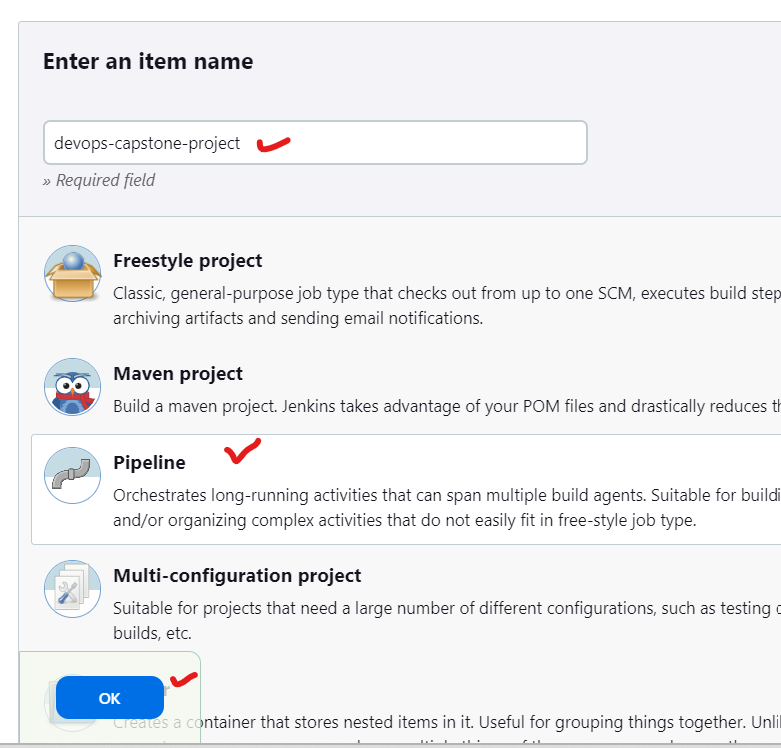
1. Create 2 deployment and 2 service files for nodejs app and mysql db. Here mysql db is using ClusterIP and nodejs app is using load balancer service to access from outside.
2. Create a init.sql file and add that as config map in mysql-deployment.yml. Here we are creating the db->table->insert records to table.



1. Create all .tf files to automate our infrastruacture. Here we are creating resource group and a Kubernetes cluster in Azure.
2. Now commit and push all your changes to Github



1. Open Jenkins-> Refer <https://github.com/Satyabrataswain480/skilluponlinetraining/blob/jenkins/JenkinsInstallationSteps.txt> for installation of jenkins.
2. Create a item type as pipeline.



1. Define all the stages
2. Checkout from Github
3. Build docker image
4. Push Docker image to Dockerhub
5. Install Azure CLI for IAC deployment
6. Provision infrastructure with terraform
7. Deploy to AKS
8. Validate Pod and Service
9. Fetch the URL from load balancer and sent a alert in email.

A screenshot of a computer

Description automatically generated

1. Instead of hardcoding all details declare environment variables like below and use that in all stages of pipeline.

A close-up of a computer code

Description automatically generated

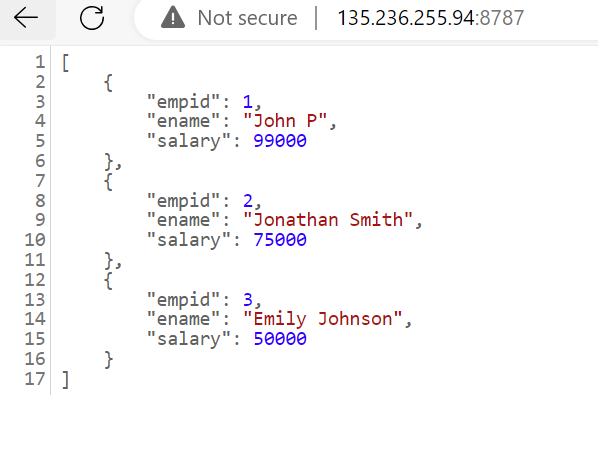
1. For login to docker hub and aks from jenkins, create credentials like below.

A screenshot of a computer

Description automatically generated

1. Once Pipeline is succeeded, browse the application via load balancer URL

|  |
| --- |
| <http://135.236.255.94:8787/>  <http://135.236.255.94:8787/hello> |

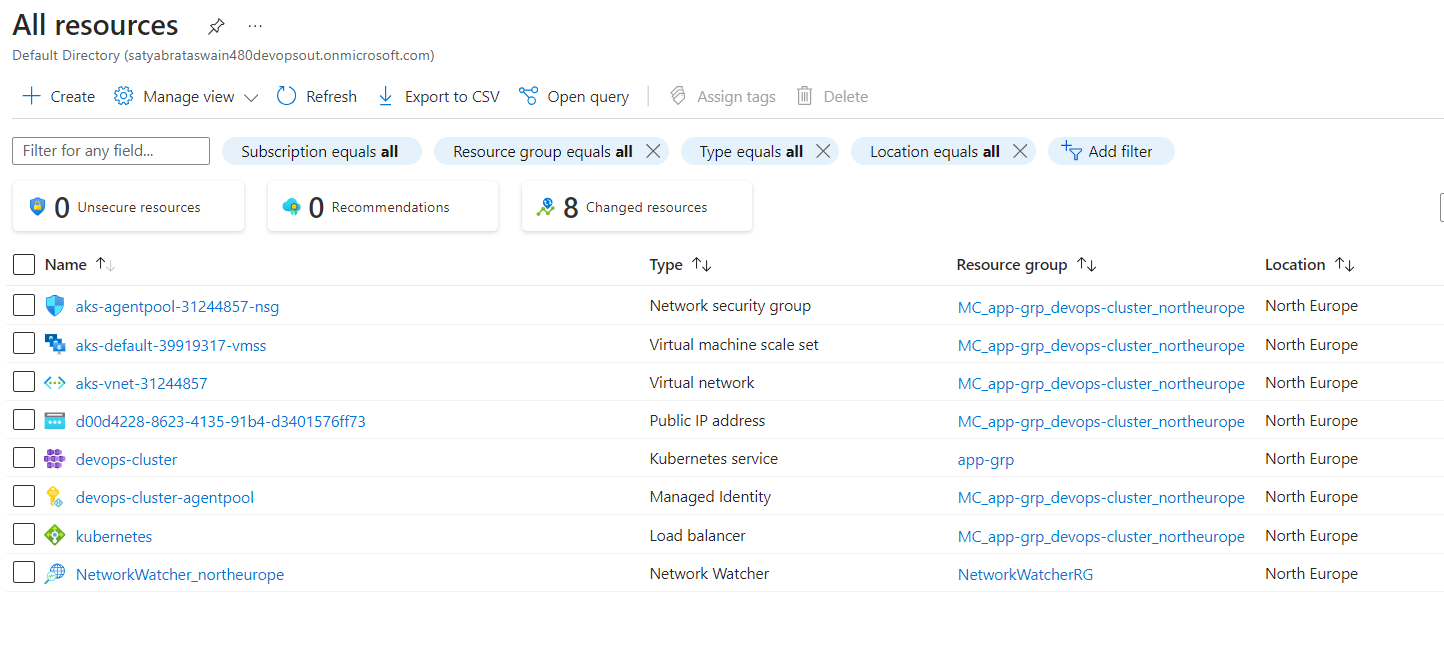


A screenshot of a computer

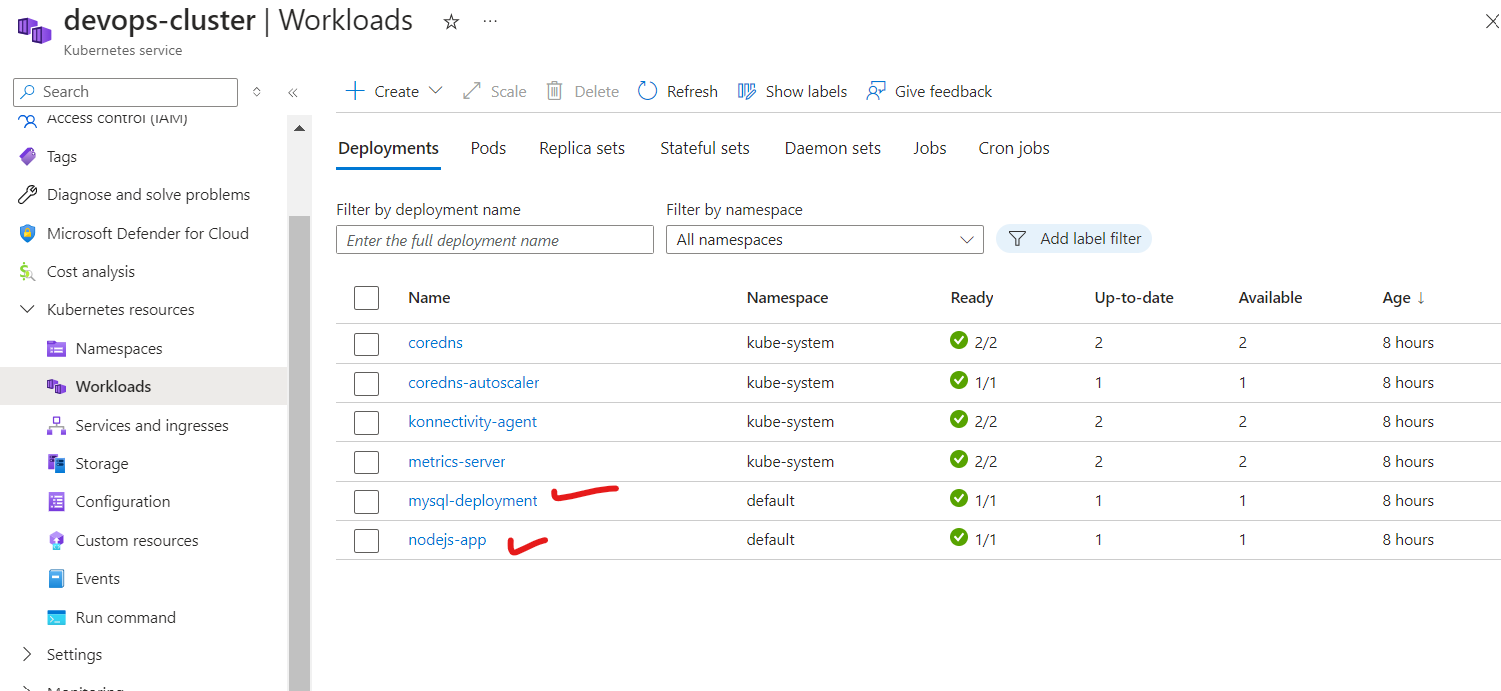
Description automatically generated

1. Azure resources created via terraform

**All Resouces**



**Kubernetes Deployment**



**Kubernetes PODS**

A screenshot of a computer

Description automatically generated

**Kubernetes ReplicaSets**

A screenshot of a computer

Description automatically generated

**Kubernetes ConfigMaps**

A screenshot of a computer

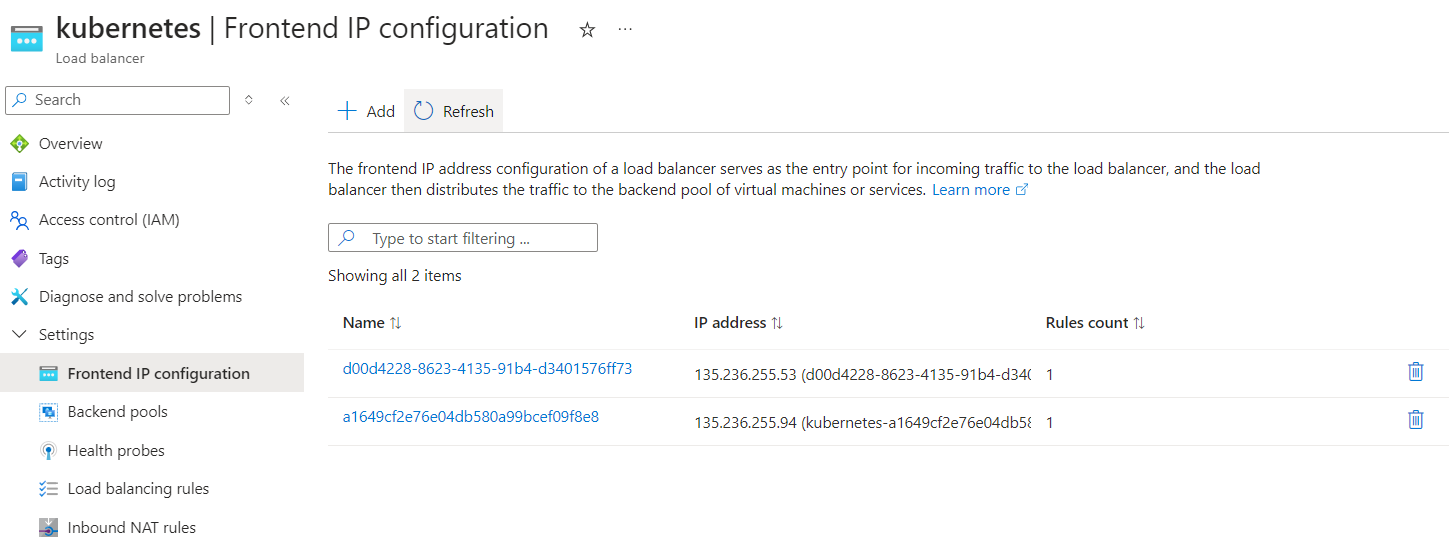
Description automatically generated

**Kubernetes Service**

A screenshot of a computer

Description automatically generated

**Kubernetes Load Balancer**



**Commands and Troubleshooting:**

* kubectl apply -f mysql-deployment.yml
* kubectl apply -f mysql-service.yml
* kubectl apply -f nodejs-deployment.yml
* kubectl apply -f nodejs-service.yml
* kubectl get all
* kubectl describe pod nodejs-app-789bbd97dc-whdh4
* az aks get-credentials --resource-group app-grp --name devops-cluster --file ~/.kube/config

**For more info refer my Github account:**

<https://github.com/Satyabrataswain480/SkillupProjects/tree/devops-capstone-project>