# Name: Neha Sharma MID: M1040187

# 

# Kubernetes 201

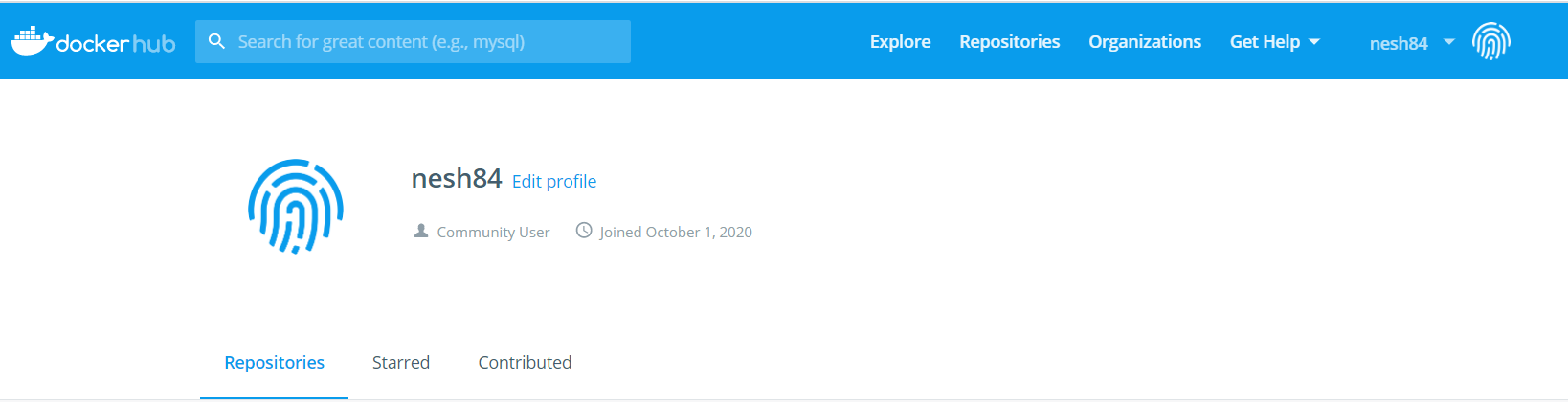
Problem Statement:

Build a 2-tier java application. Write and push the image to Docker hub. Containerize the application by creating yml file for defining the application service structure and run the same using Kubernetes.

Deployments / Replication sets. Bring in Auto Healing & Auto Scaling capabilities to the app. Ensure data persistence by mounting the data outside of the containers.

Account Creation:

Create a Docker account hub official site ([https://hub.docker.com/).](https://hub.docker.com/).The) The created account is used to create a repository and push the Docker Images. Now we must create a repository.

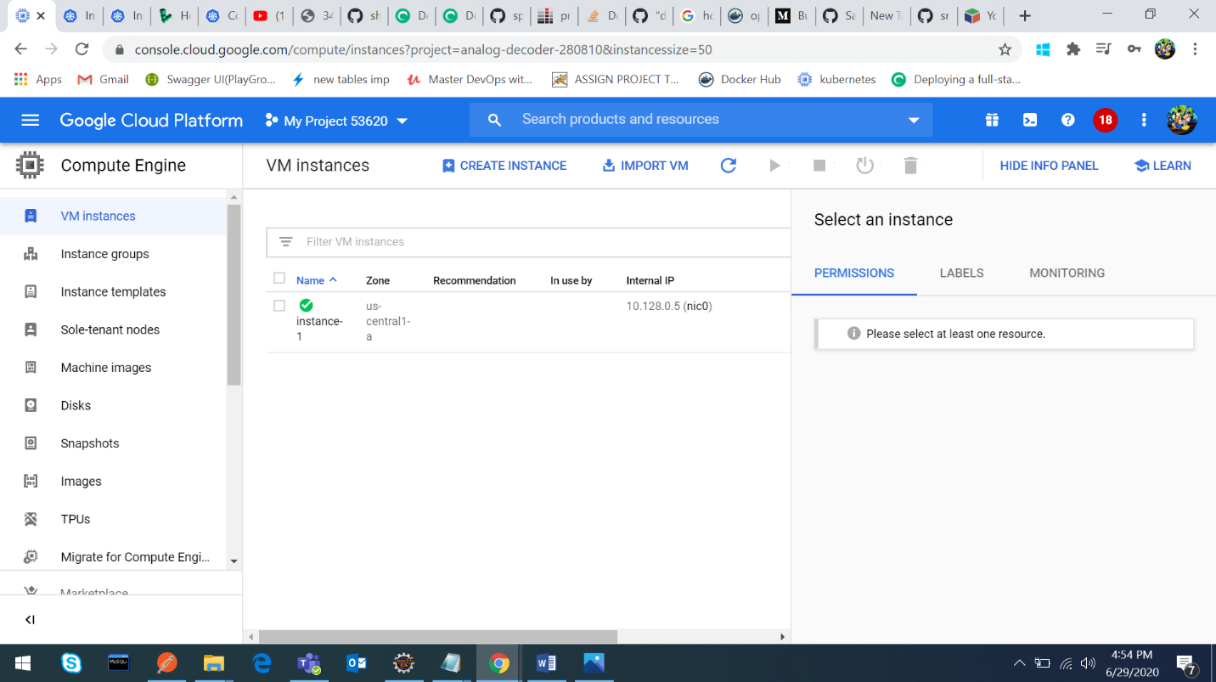


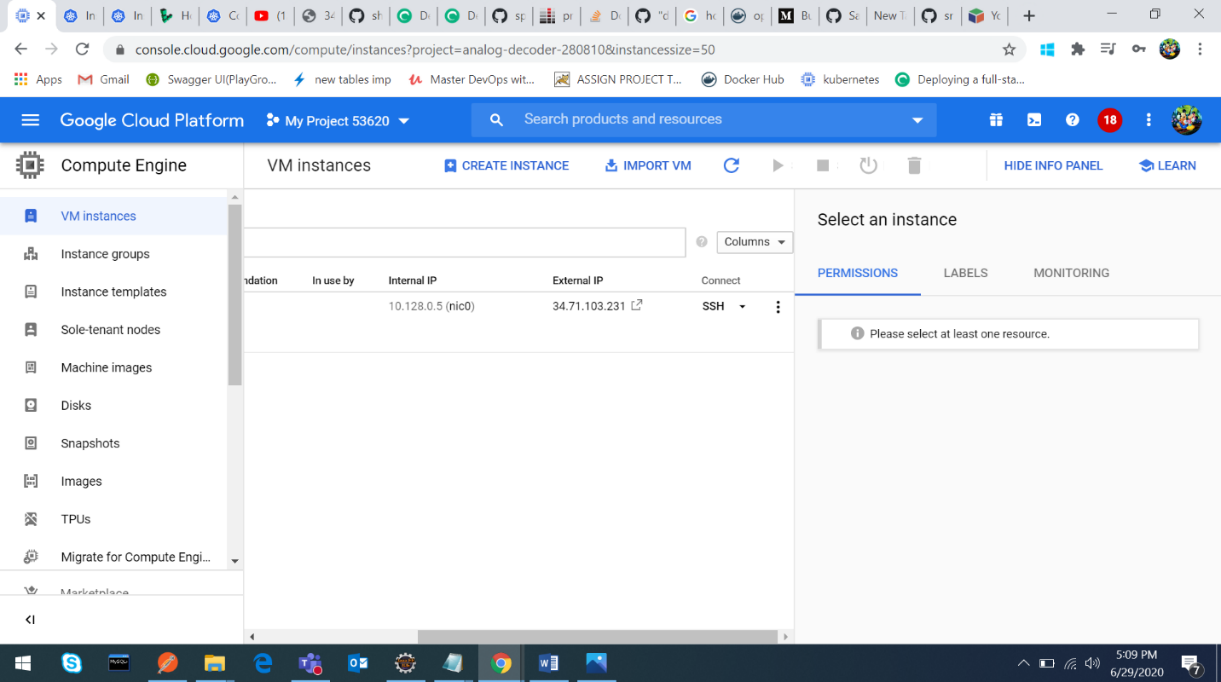
Cloud platform (GKE):

Create an account in google cloud platform and Login with your details.

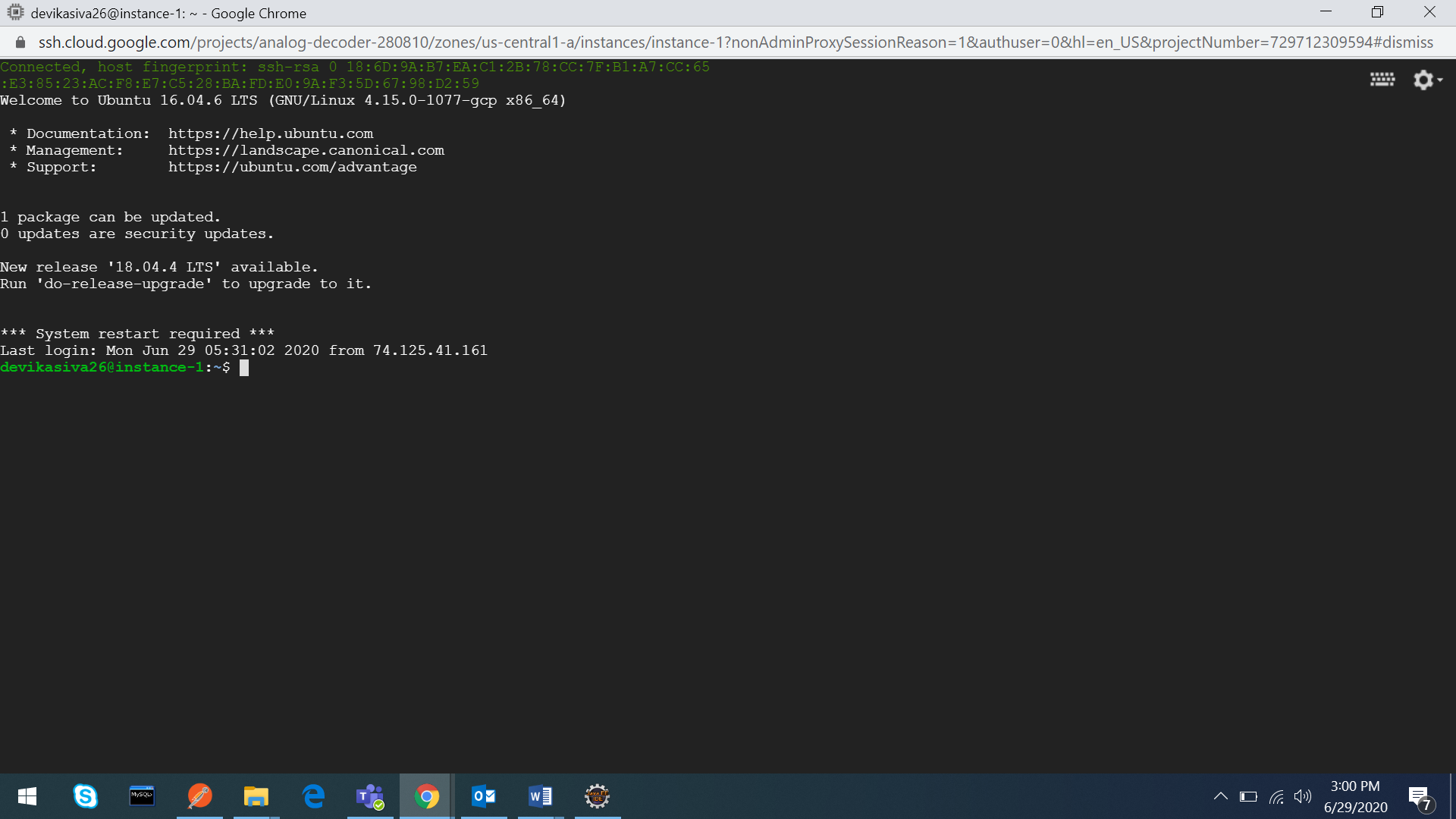
<https://cloud.google.com/kubernetes-engine>

Once the Kubernetes pages opens, we have to make cluster first by clicking on Create cluster.





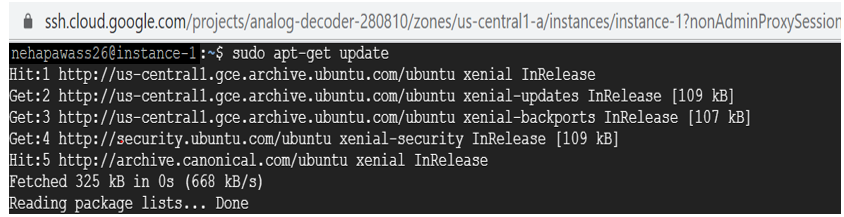
Use SSH to login into VM:

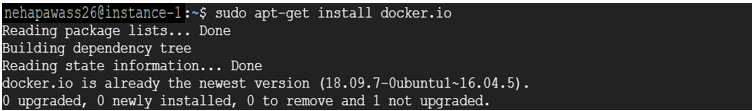


Docker Installation:

Commands: sudo apt-get update

sudo apt-get install docker.io





Validate with Command: docker version

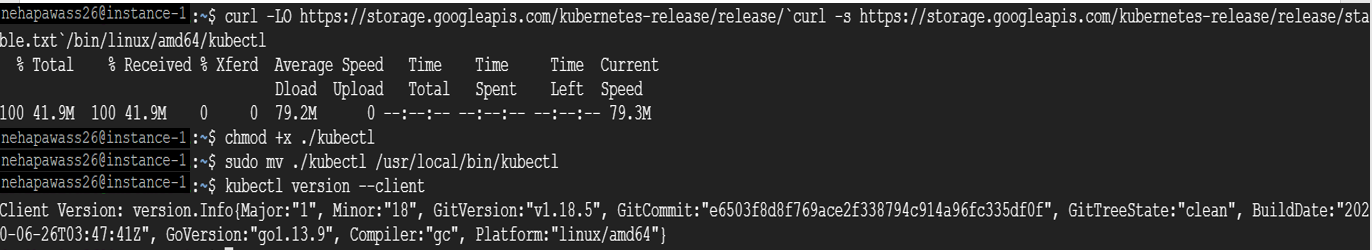
Now install **Kubectl** command:

curl -LO https://storage.googleapis.com/kubernetes-release/release/`curl -s [https://storage.googleapis.com/kubernetes-release/release/stable.txt`/bin/linux/amd64/kubectl](https://storage.googleapis.com/kubernetes-release/release/stable.txt%60/bin/linux/amd64/kubectl)

Make kubectl binary executable by running: chmod +x ./kubectl

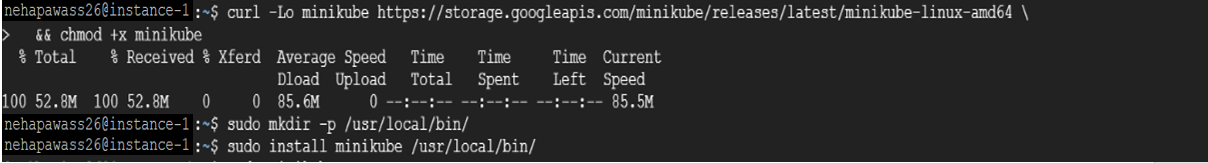
Move binay into customized path by running: sudo mv ./kubectl /usr/local/bin/kubectl

Check version by running: kubectl version --client



Minikube Installation:

Run: curl -Lo minikube https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64 **\** && chmod +x minikube



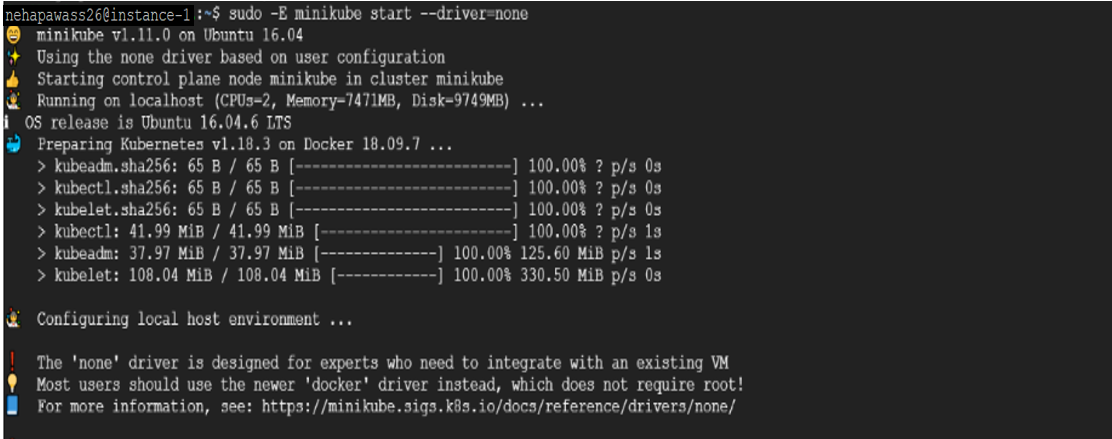
Add minikube executable to the path

Commands: sudo mkdir -p /usr/local/bin/

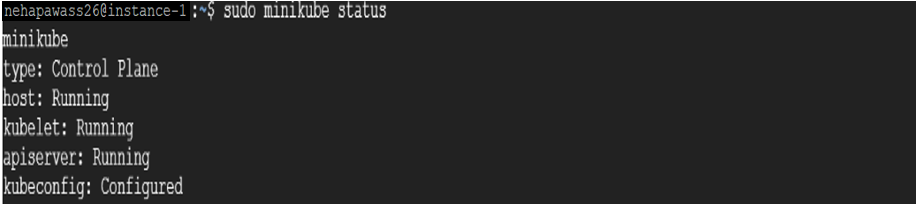
sudo install minikube /usr/local/bin/

Start minikube

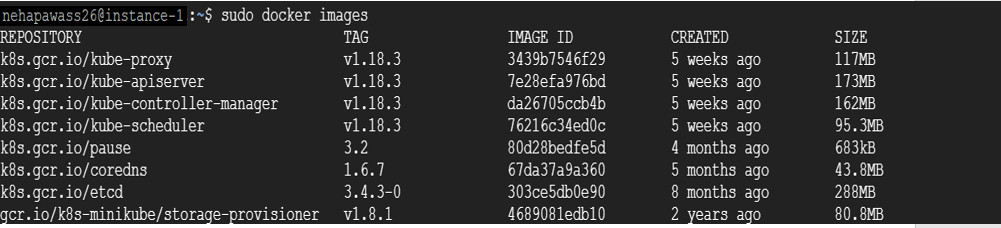
Command: sudo –E minikube start –driver=none



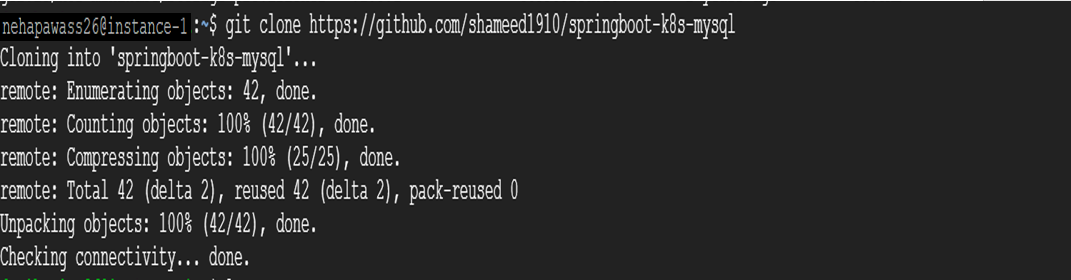
Check minikube status: sudo minikube status



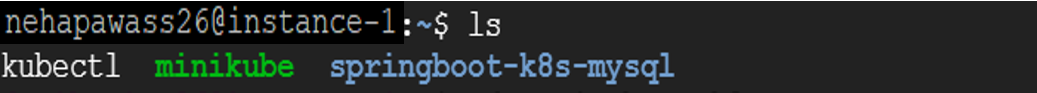
Check docker images: sudo docker images



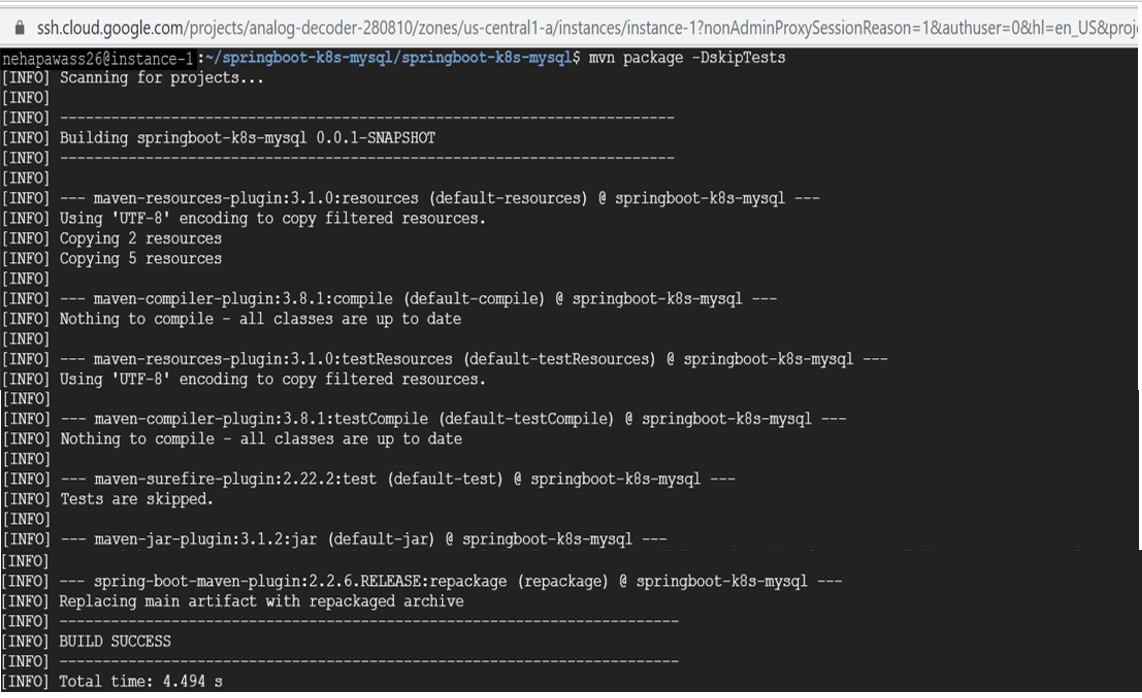
Cloning existing project using command: git clone <git-url>



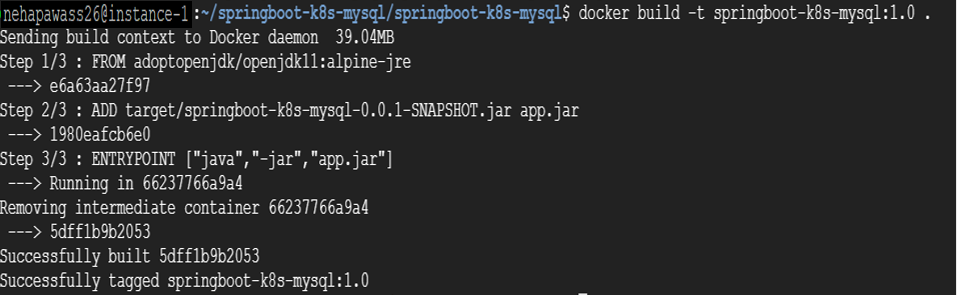
Now we have everything to start:



To generate jar run: mvn package –DskipTests



Build project run: docker build –t springboot-k8s-mysql:1.0 .



Make sure all yml file should present in resources folder.

Location: springboot-k8s-mysql/springboot-k8s-mysql/scr/main/resources

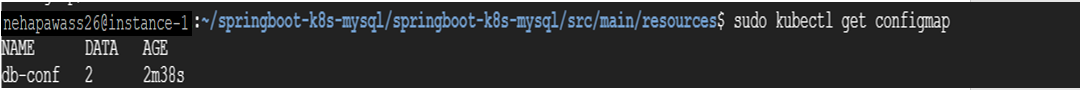


Creating ConfigMap:

Command: sudo kubectl apply –f mysql-configmap.yml

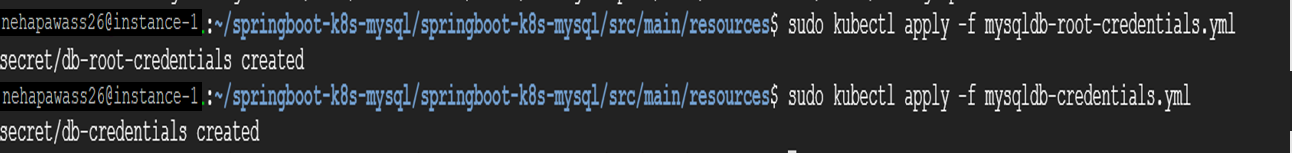


Check the created config map: sudo kubectl get configmap

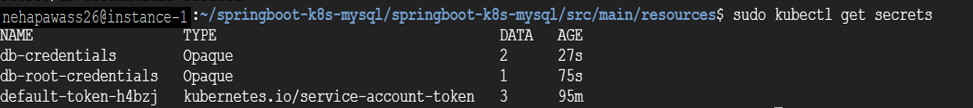


Creating Secrets:

DB root secrets Run: sudo kubectl apply –f mysqldb-root-credentials.yml

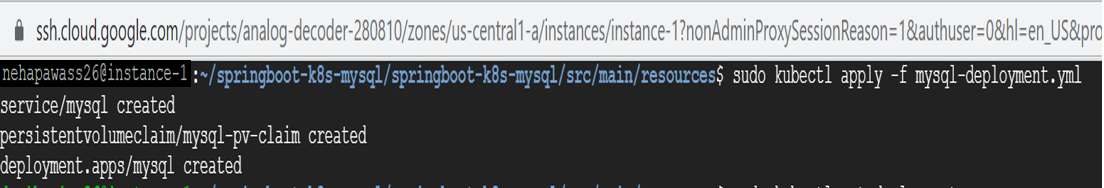


Look secrets created run: sudo kubectl get secrets



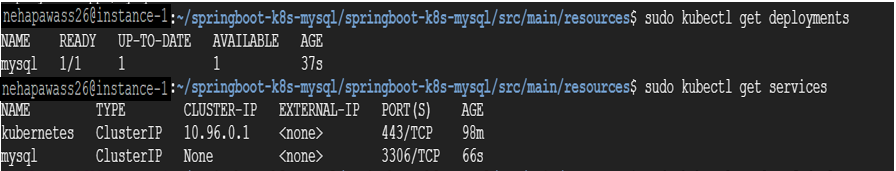
Creating Deployments File:

Create mysql deployment by running: sudo kubectl apply –f mysql-deployment.yml

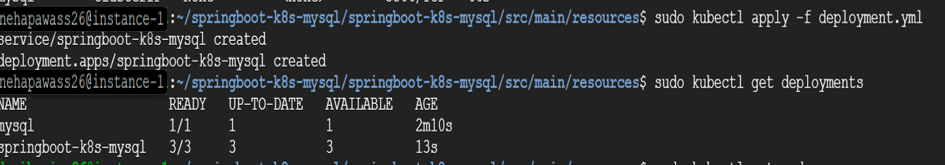


For deployment file run: sudo kubectl get deployments

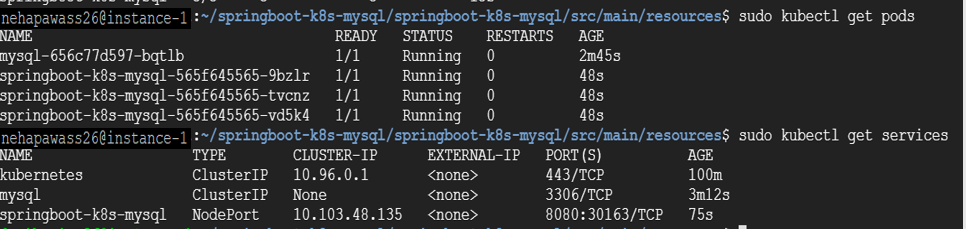
For services run: sudo kubectl get services



Spring boot deployment and services object run: sudo kubectl apply –f deployment.yml

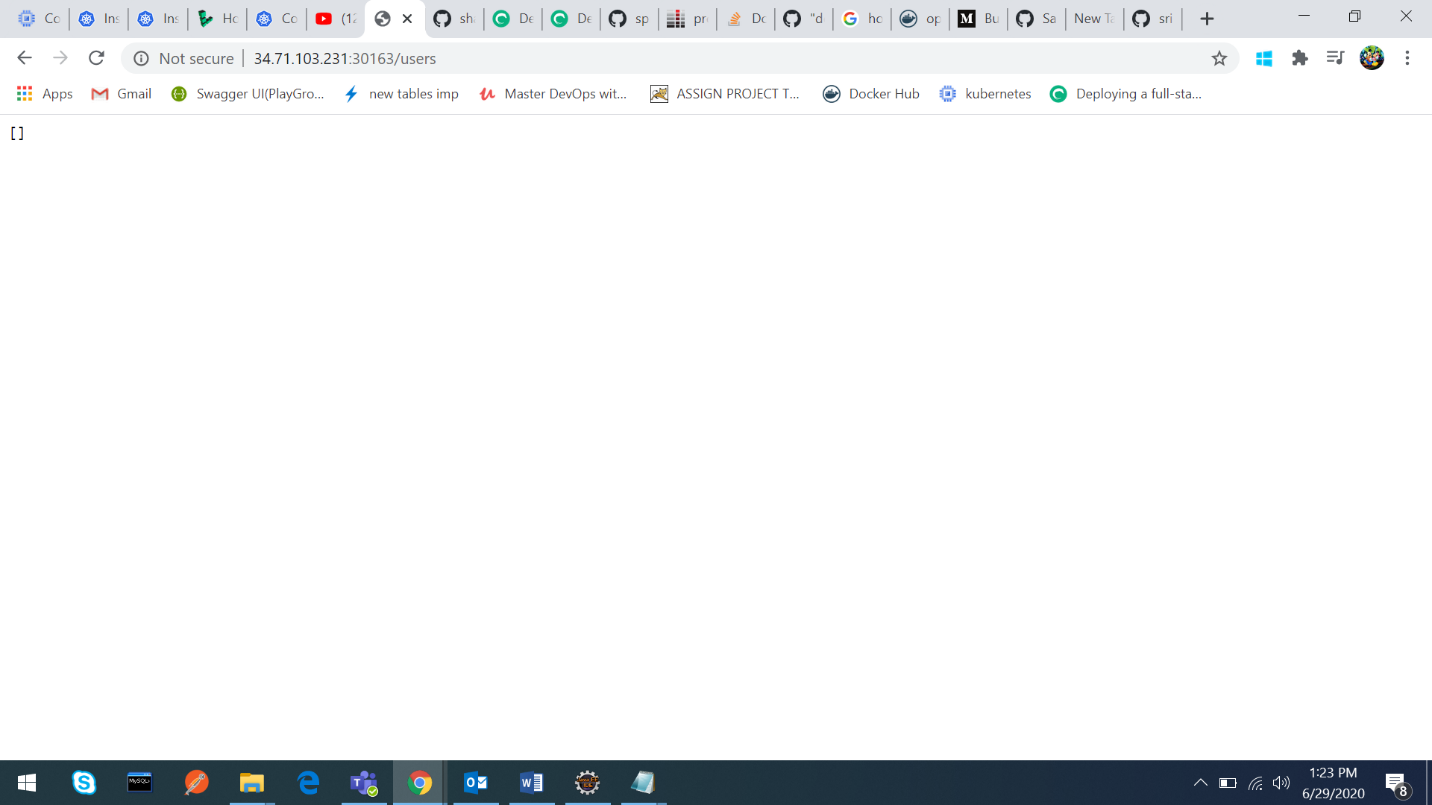


Look pods are running: sudo kubectl get pods



Running Application:

Application running in <public\_ip\_of\_host> means, VM External IP

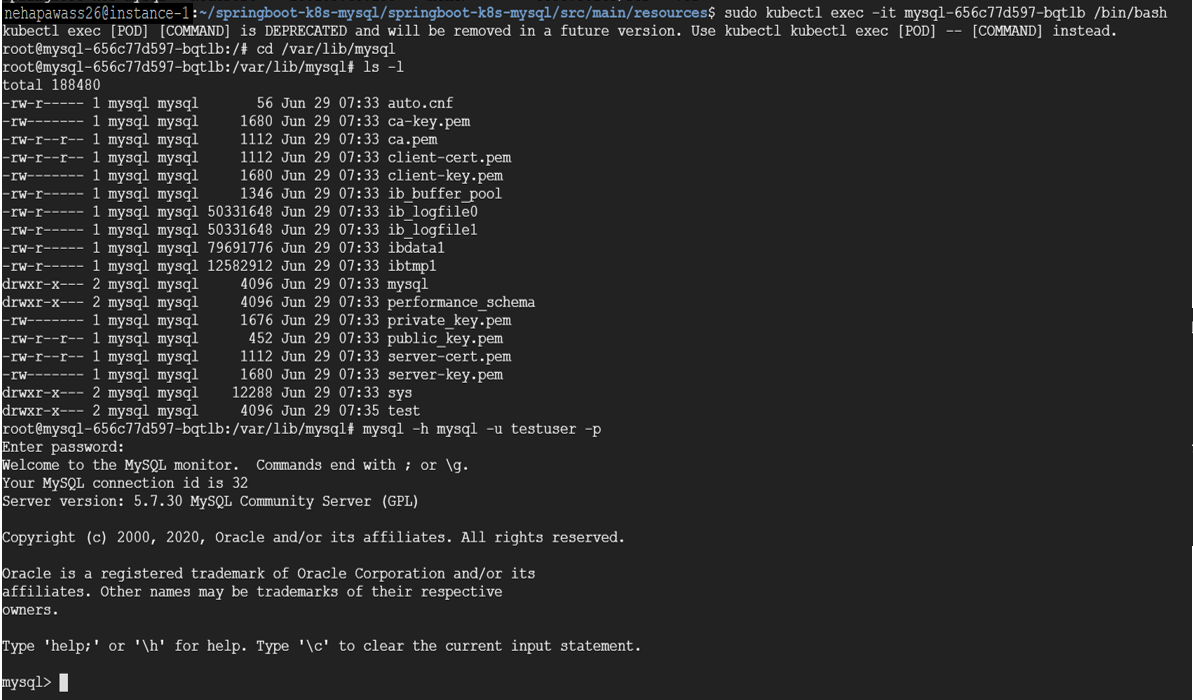


Ah: No data is present, let’s add some

Accessing MySql:

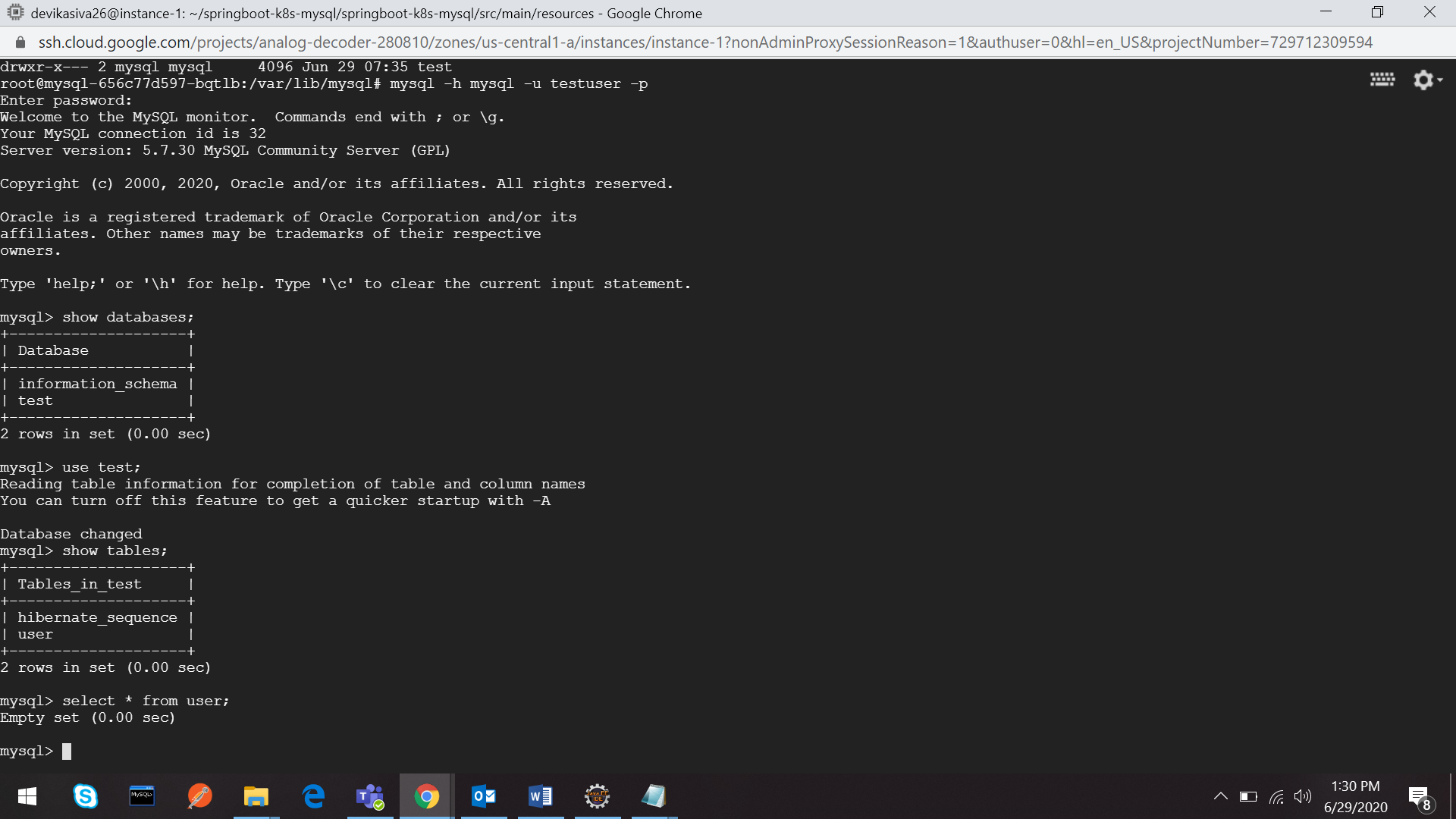
Mysql container: sudo kubectl exec –it <mysqlId> /bin/bash

Login: mysql –h mysql –u <user\_name> –p<password>



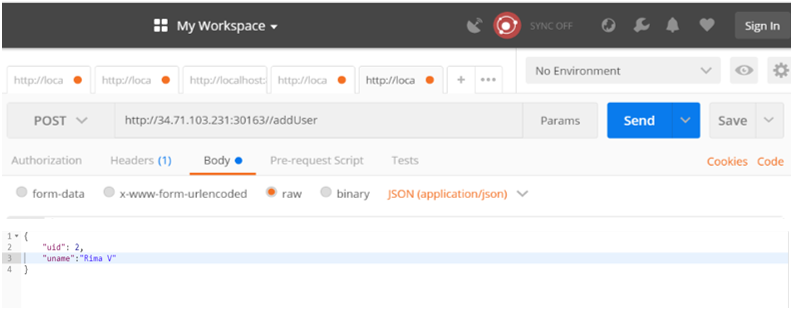
Look databases, user table is empty.

Command: show databases;

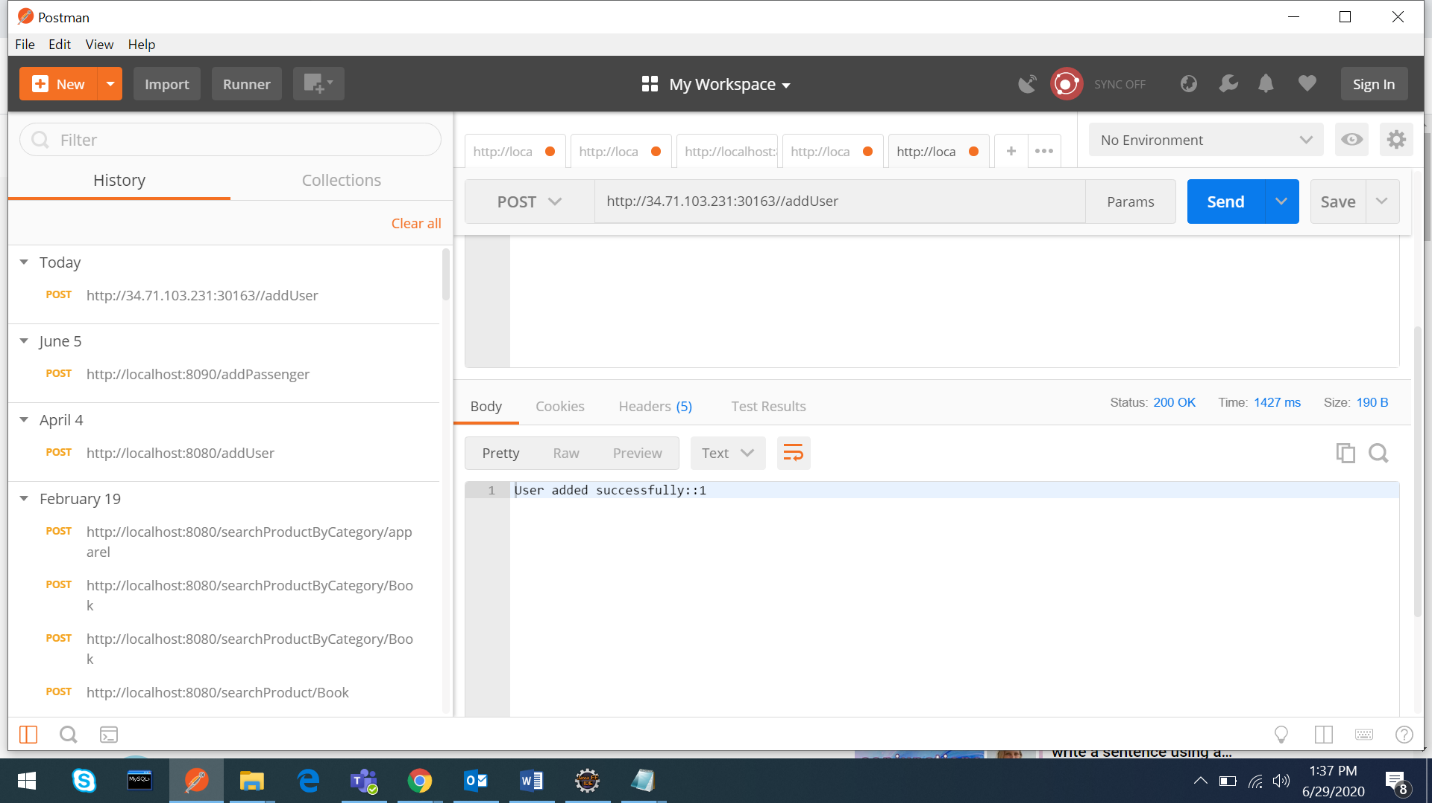


Run application:

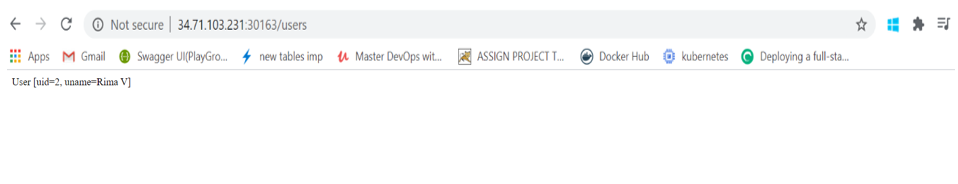
Add-



Successful-



View-

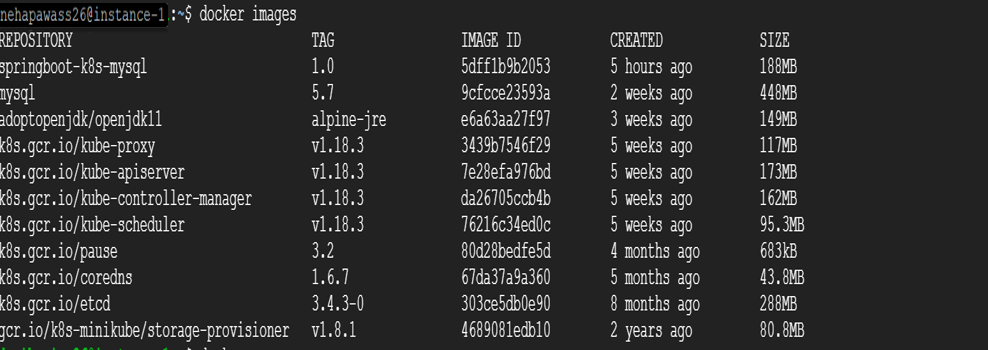


Even we can validate thru database:

Command: select \* from user;

Check existing docker image:

Look all images exist in the repository run: docker images

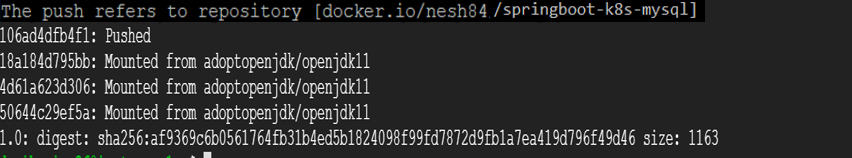


Push Docker image:

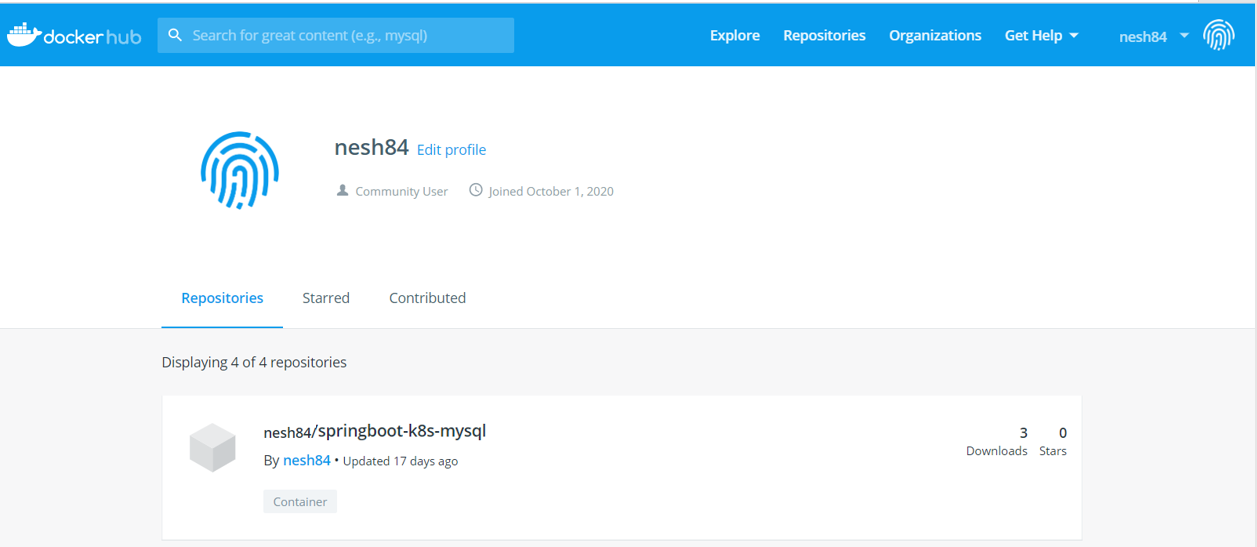
Login Docker hub using command: docker login

Now push docker image into **Docker Hub** repository.

Command: docker push nesh84/springboot-k8s-mysql:1.0



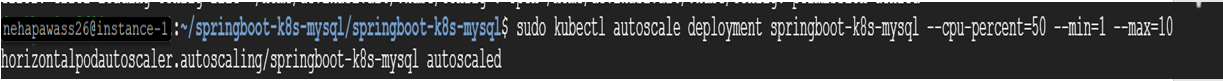
Validate:



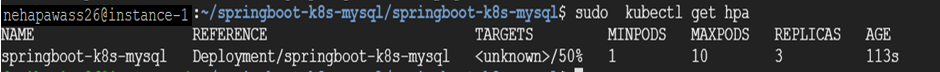
Auto Healing & Auto Scaling:

For Auto Healing & Auto scaling capabilities run:

sudo kubectl autoscale deployment springboot-k8s-mysql --cpu-percent=50 --min=1 --max=10

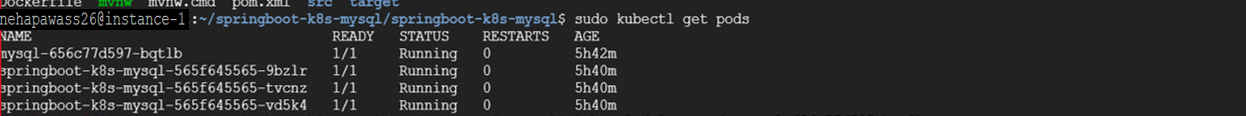


Check status run: sudo kubectl get hpa

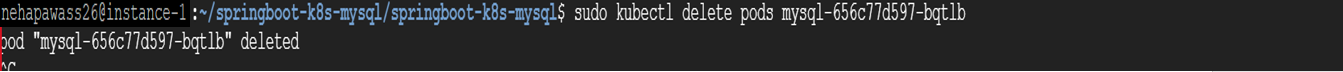


Check for data persistence:

Get all the pods by running: sudo kubectl get pods

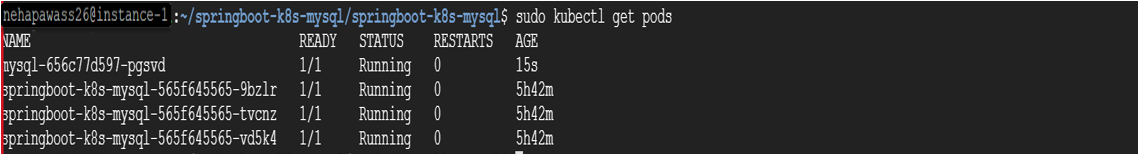


Delete the mysqlpod: sudo kubectl delete pods <pod>



Again check list of pods: sudo kubectl get pods

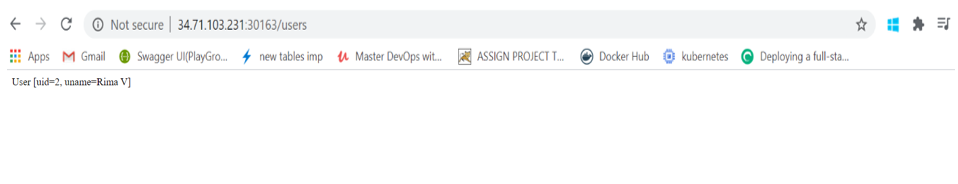
New MySQL pod is created with different pod Name



Now run the app again and check whether the data is present or not.

Here we can see by creating new pod with the same data, data persisted even though pod is deleted.

Result:



Data persisted!

Great ☺