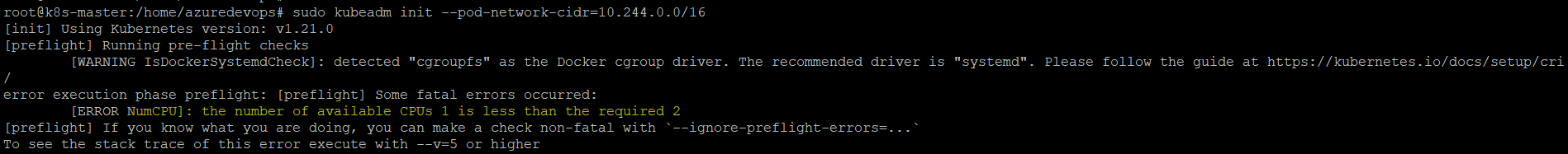
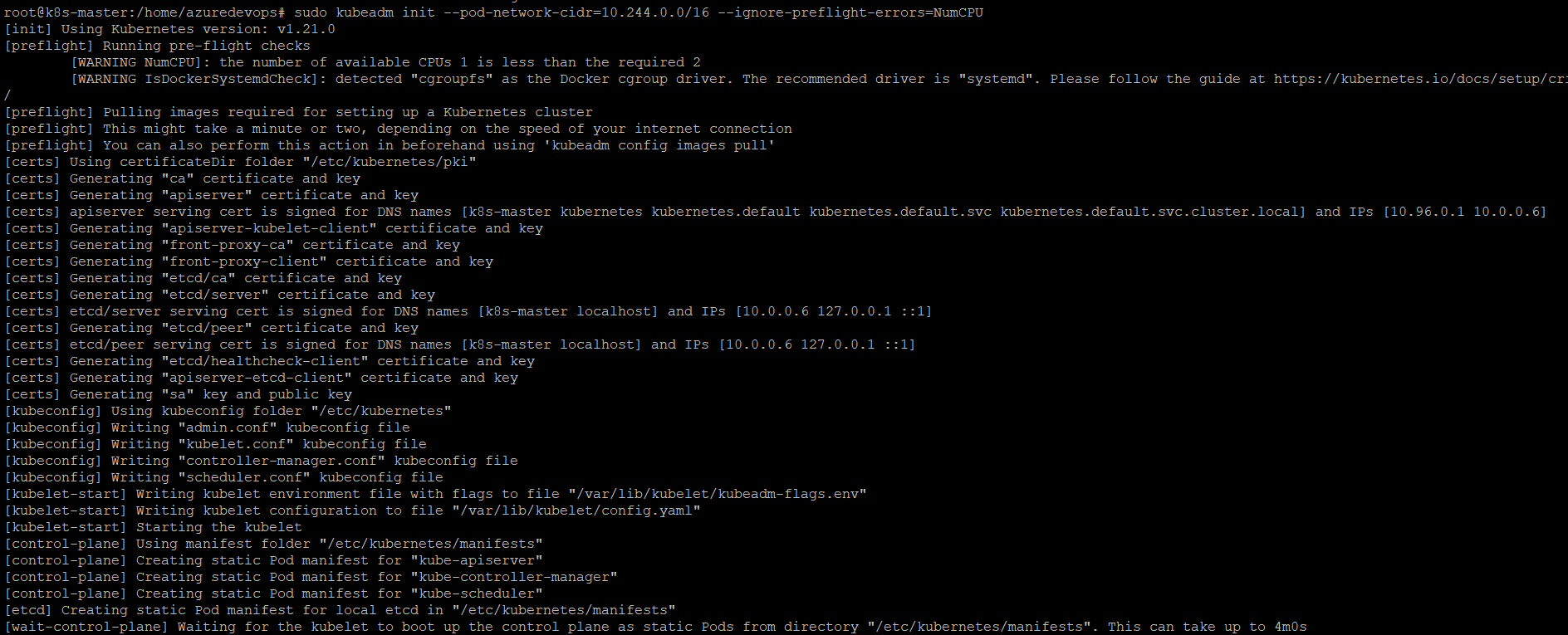
**Deploying SpringBoot Application with Nodejs on Kubernetes**

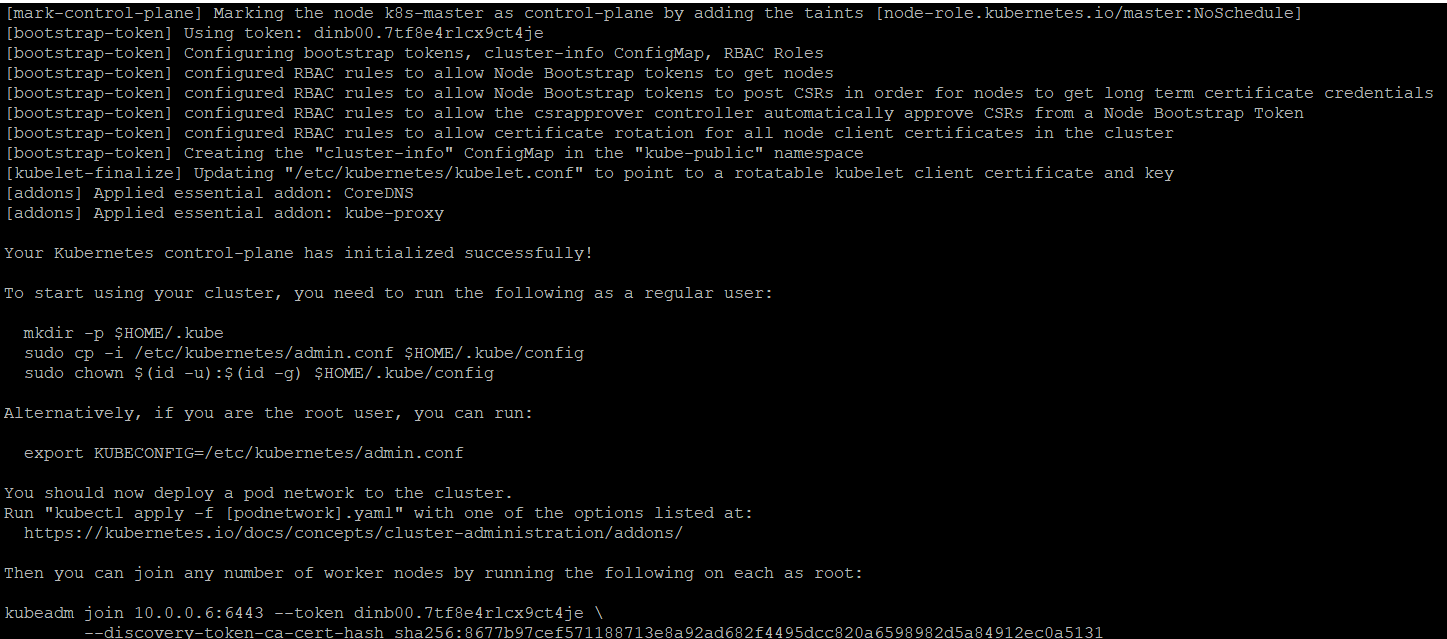
**Installation of Kubernetes on Ubuntu with single CPU**

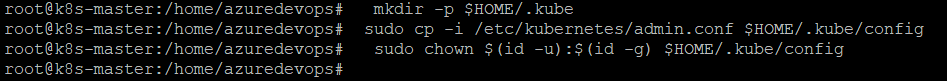
**$sudo kubeadm init --pod-network-cidr=10.244.0.0/16**

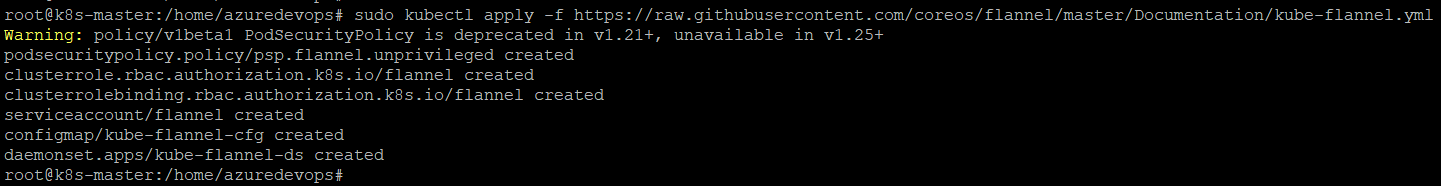


$**sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=NUMCPU**

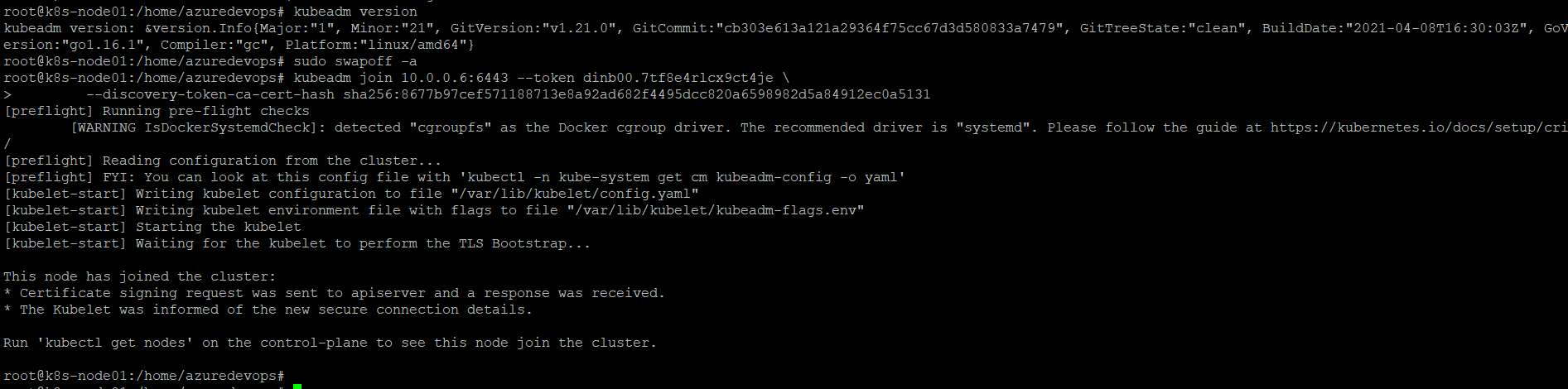




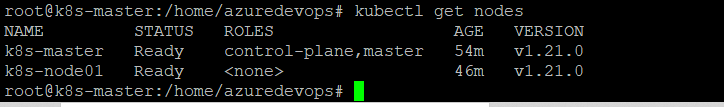


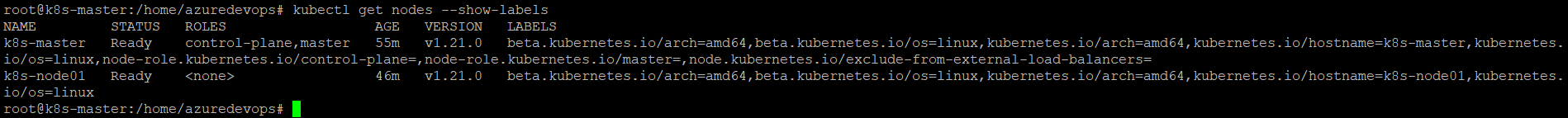
**$sudo kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml** 

**$kubeadm version**



**$kubectl get nodes**





**Deploying SpringBoot Application with Nodejs on Kubernetes**

Github Url:

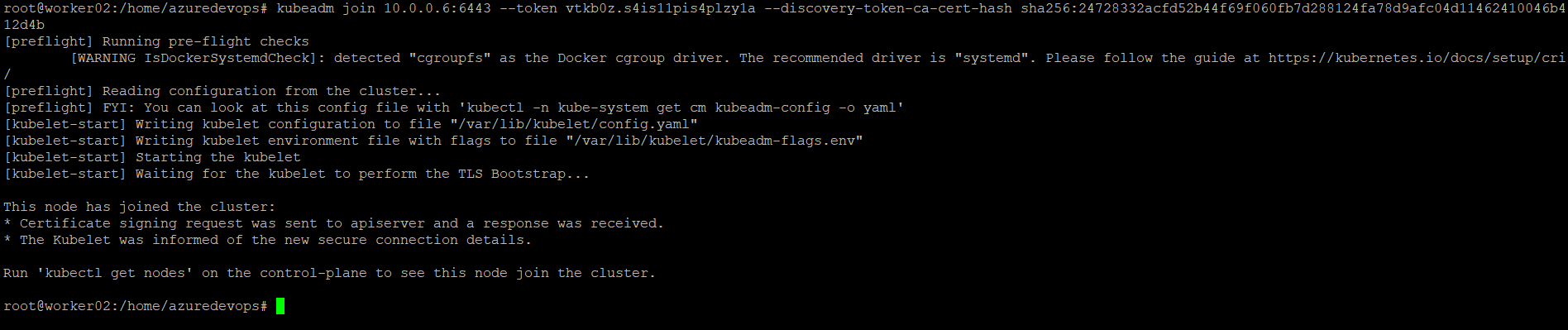
**Adding new Worker node to the cluster**

In Master node

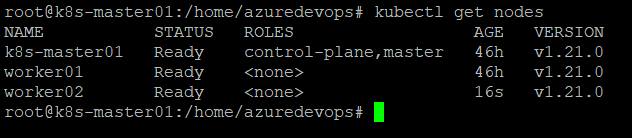
**$kubeadm token create --print-join-command**

****

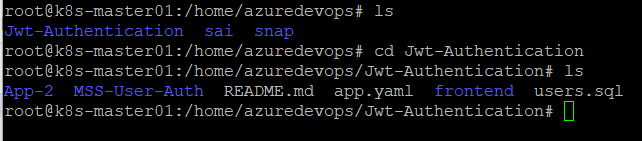
**In Worker node**

****

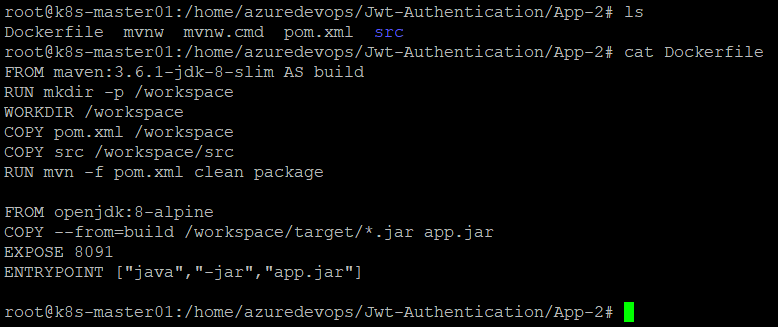
**$kubectl get nodes**

****

Listing all the files and directories in Jwt-Authentication

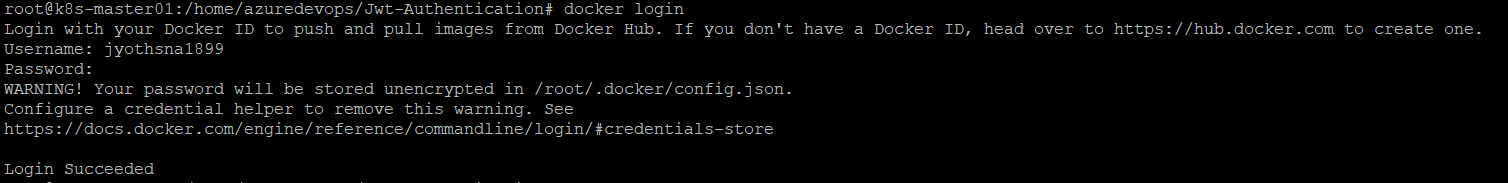
****

Changing current directory into **App-2** and list the files and directories then viewing the Dockerfile using cat command.

****

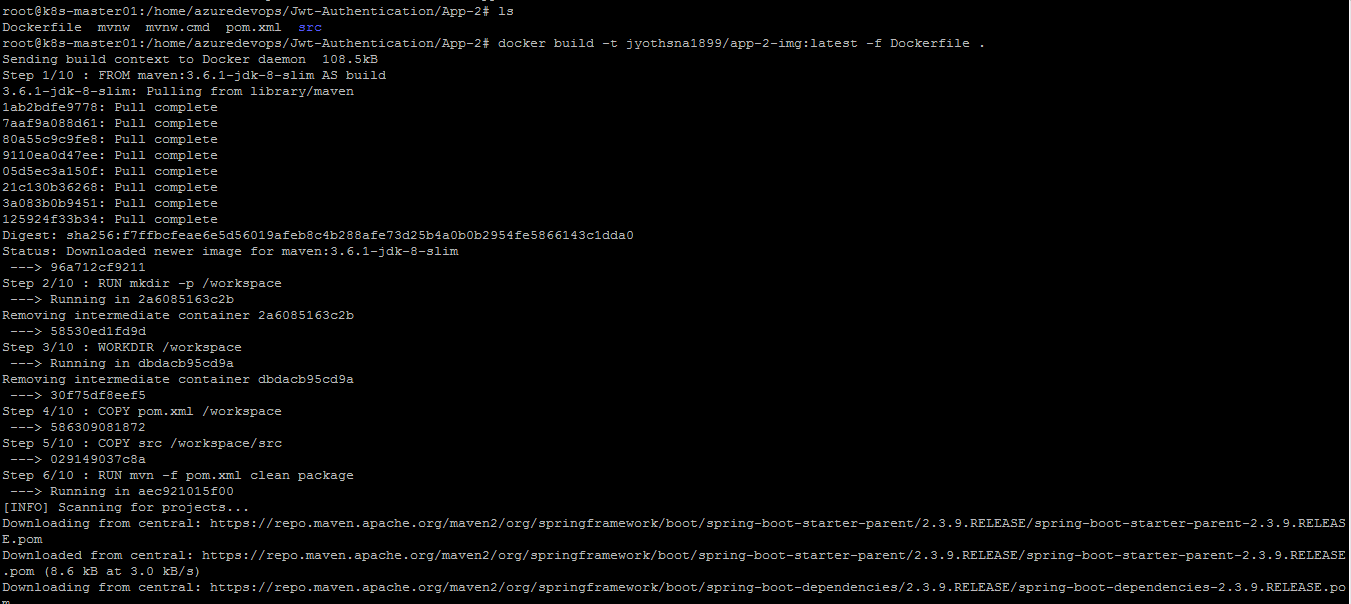
Docker login in Kubernetes:

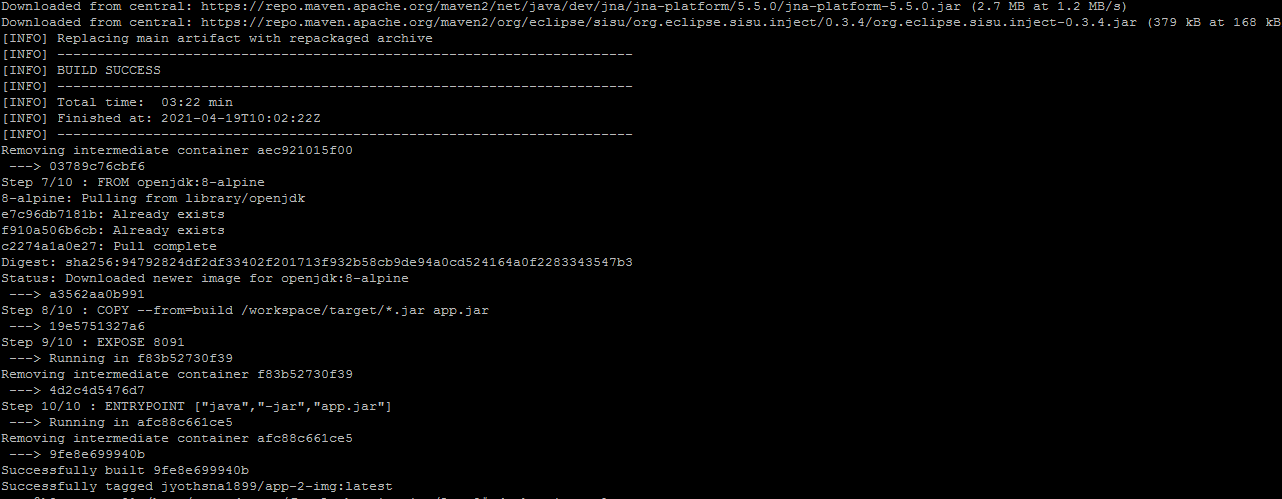
**$docker login**



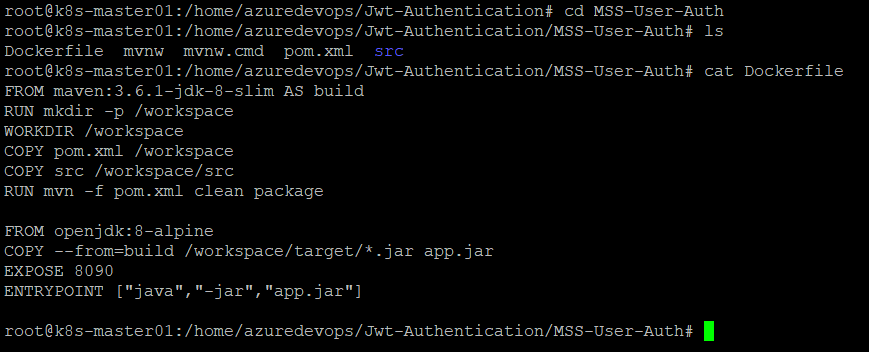
Building Docker image using Docker file:

$**docker build -t jyothsna1899/app-2-img:latest -f Dockerfile .**



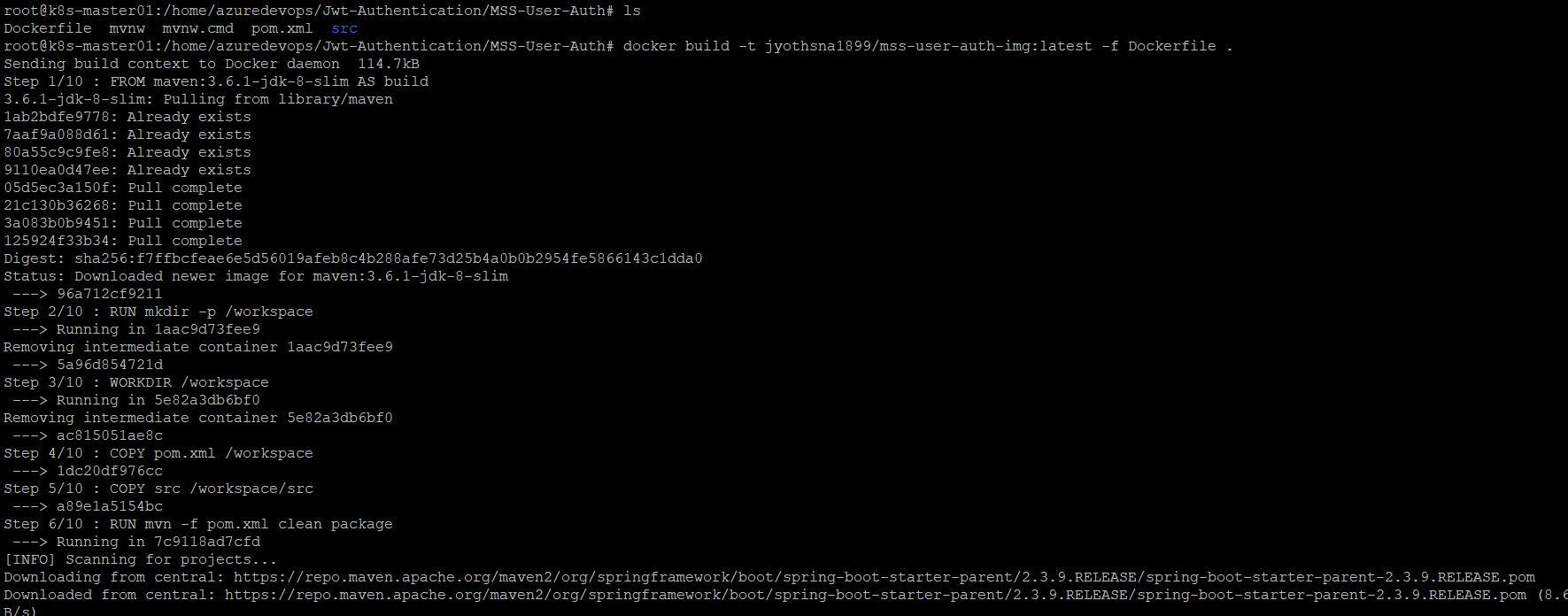


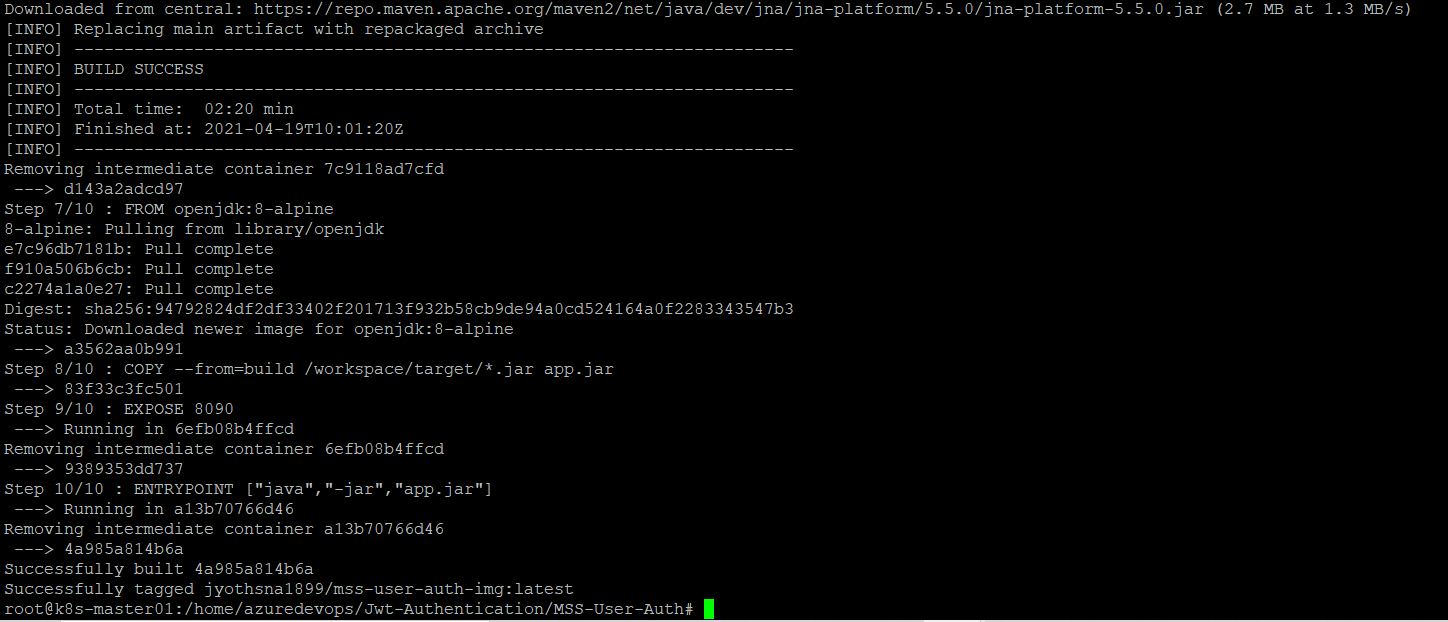
Changing current directory into MSS-User-Auth. List the files and directories and then viewing the content of Dockerfile



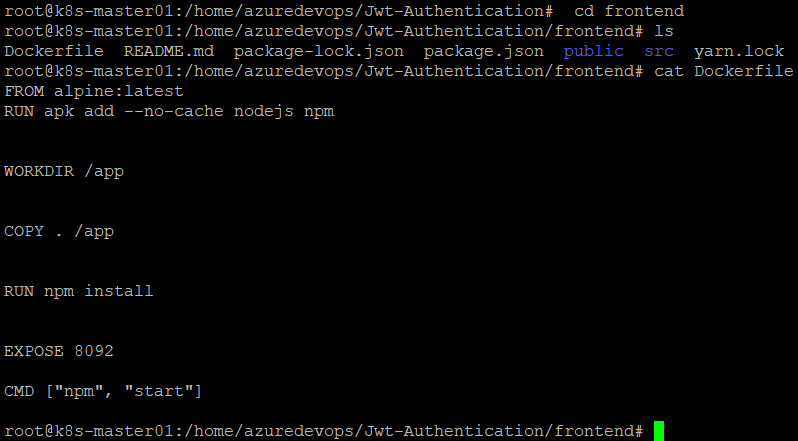
Building Docker image using Dockerfile:

**$docker build -t jyothsna1899/mss-user-auth-img:latest -f Dockerfile .**



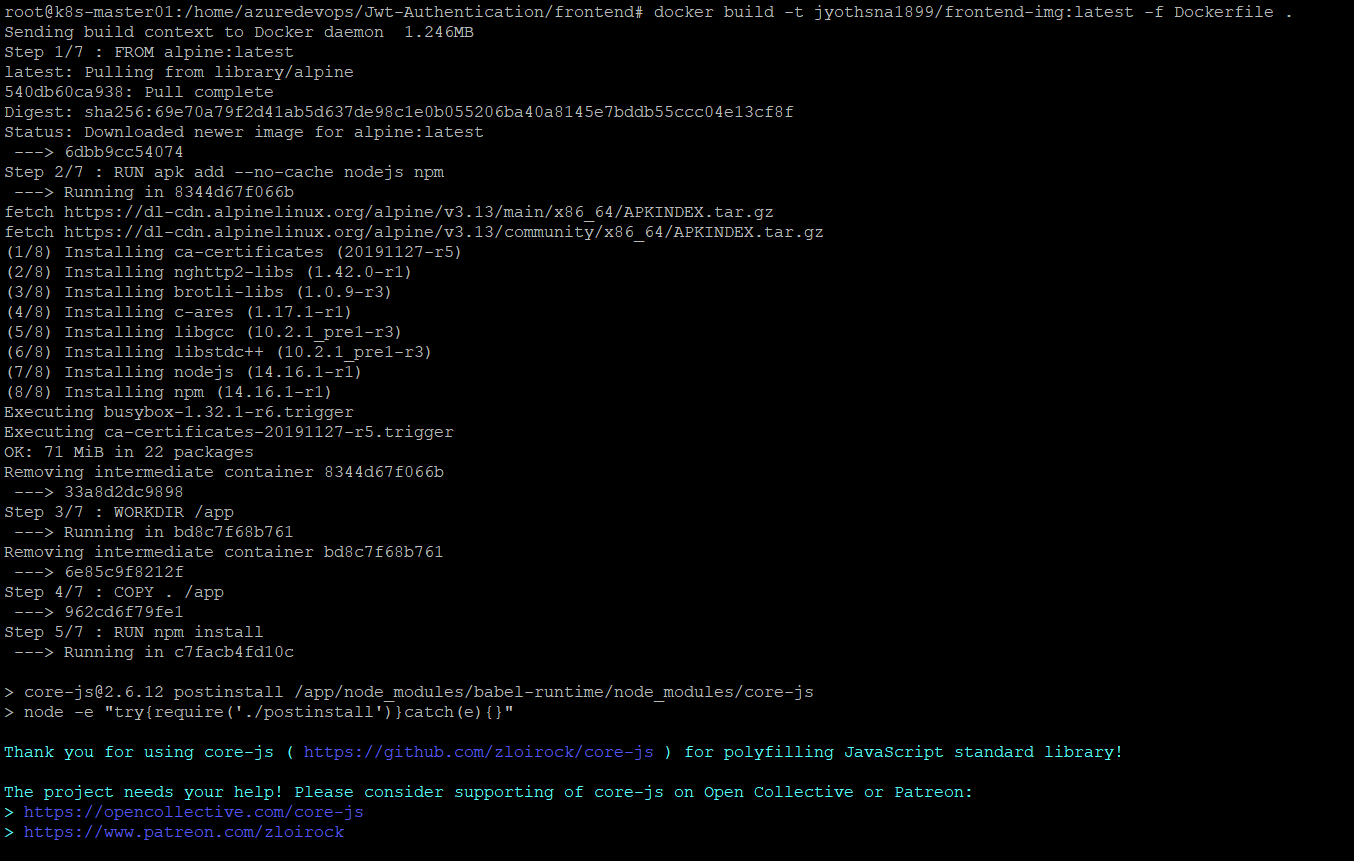


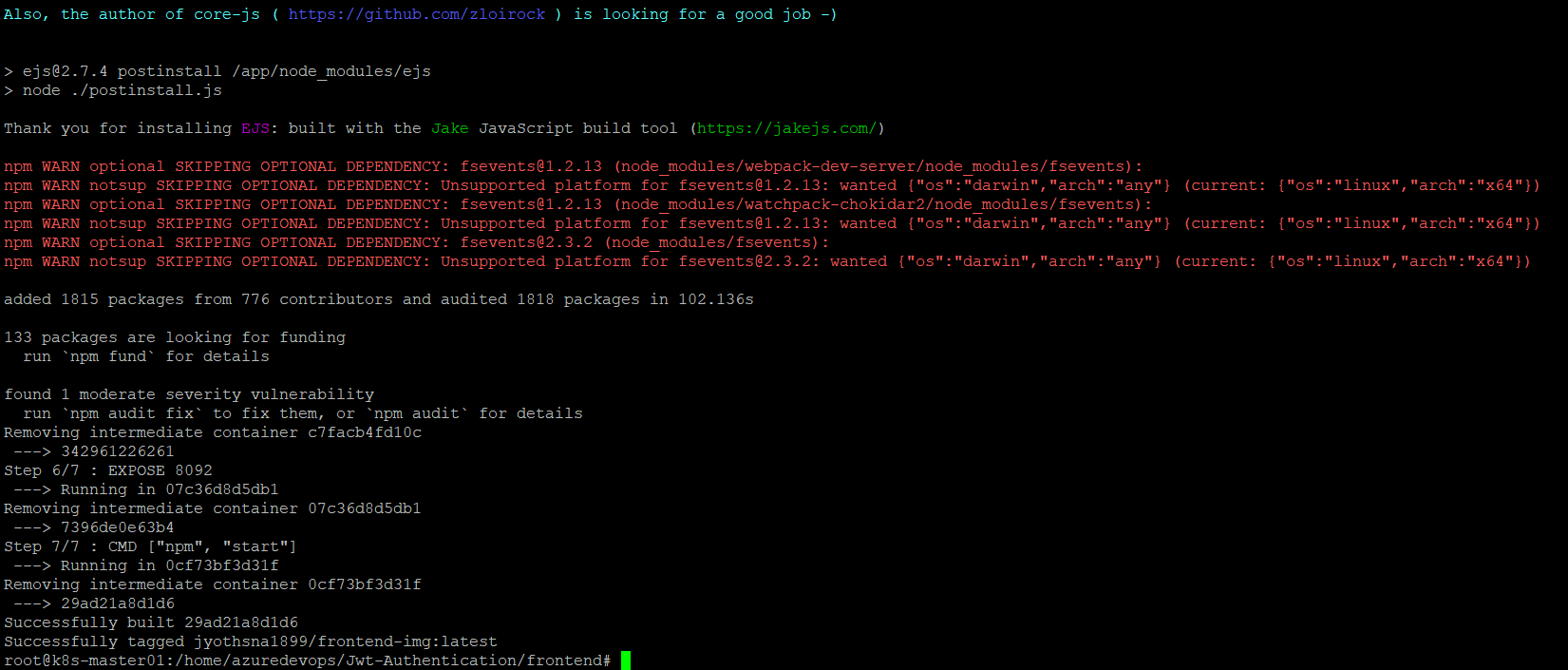
Changing current directory into frontend. List the files and directories and then viewing the content of Dockerfile



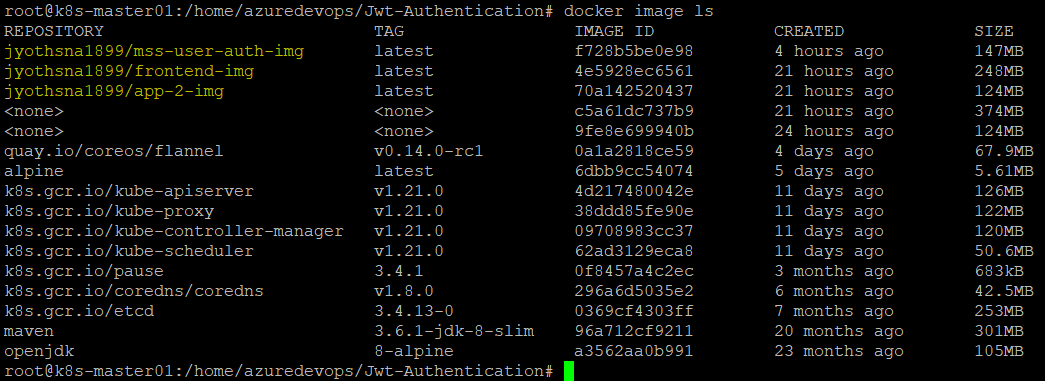
Building docker image using Dockerfile:

**$docker build -t jyothsna1899/frontend-img:latest -f Dockerfile .**





**$docker image ls**

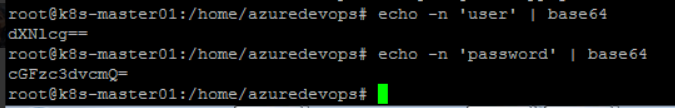
****

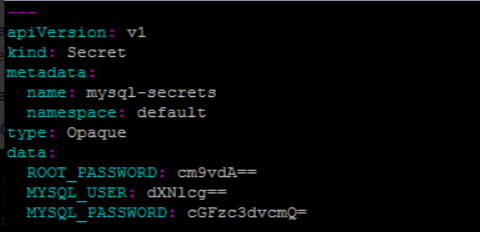
**Yaml files for Spring Boot Application:**

To deploy this application, we’ll use a few additional concepts in Kubernetes called PersistentVolumes, PersistentVolumeClaim, Secrets,Deployments,Services,Configmaps./

**Secrets:**

We’ll make use of Kubernetes secrets to store the Database credentials. A Secret is an object in Kubernetes that lets you store and manage sensitive information, such as passwords, tokens, ssh keys etc. The secrets are stored in Kubernetes backing store, etcd. You can enable encryption to store secrets in encrypted form in etcd.



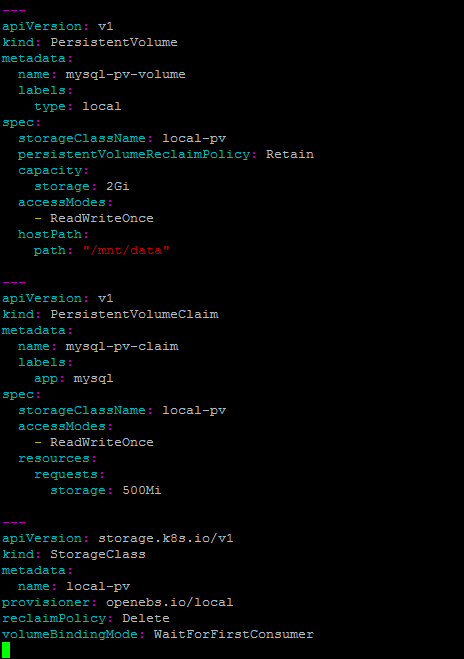


**Persistent Volumes , Persistent Volume Claim and Storage Class:**

We’ll use Kubernetes Persistent Volumes to deploy Mysql. A PersistentVolume (PV) is a piece of storage in the cluster. It is a resource in the cluster just like a node. The Persistent volume’s lifecycle is independent from Pod lifecycles. It preserves data through restarting, rescheduling, and even deleting Pods.

PersistentVolumes are consumed by something called a PersistentVolumeClaim (PVC). A PVC is a request for storage by a user. It is similar to a Pod. Pods consume node resources and PVCs consume PV resources. Pods can request specific levels of resources (CPU and Memory). PVCs can request specific size and access modes (e.g. read-write or read-only).

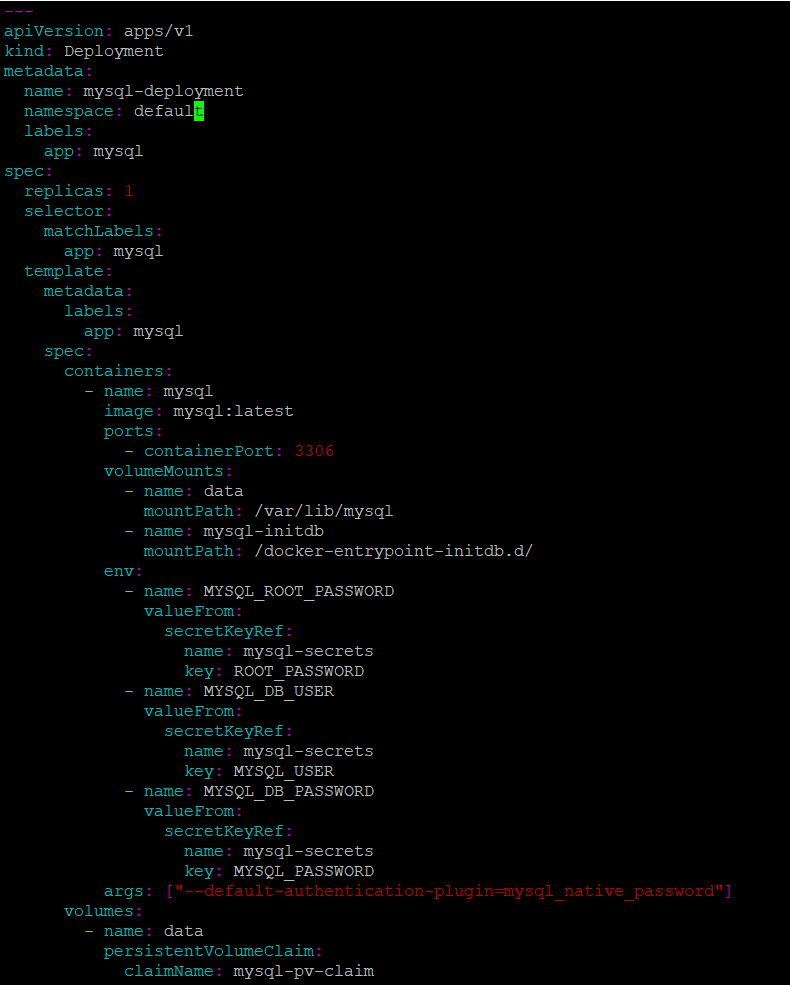
A **StorageClass** provides a way for administrators to describe the "classes" of storage they offer. Different classes might map to quality-of-service levels, or to backup policies, or to arbitrary policies determined by the cluster administrators. **Kubernetes** itself is opinionated about what classes represent.

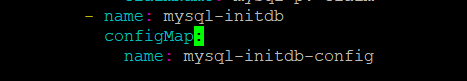


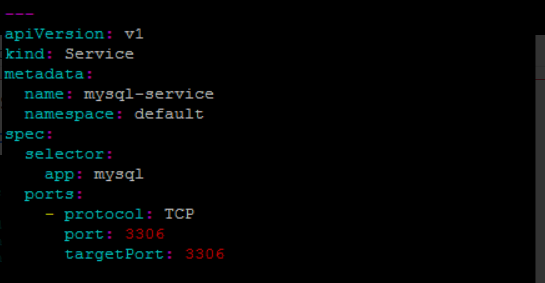
We used ConfigMap:

**$kubectl create configmap mysql-initdb-config --from-file=users.sql**

**Deploying Mysql on Kubernetes using PersistentVolume and Secrets**







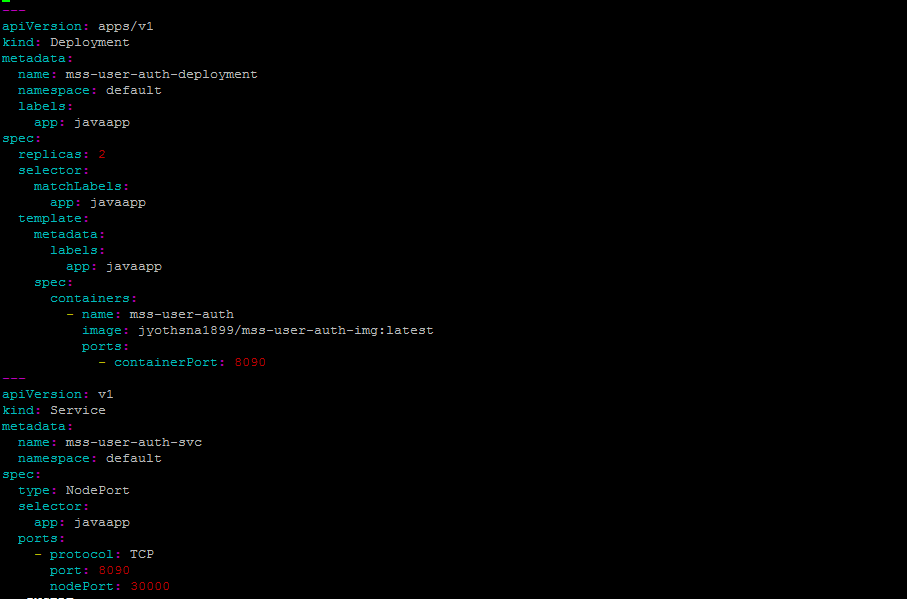
We’re creating four resources in the above manifest file. A PersistentVolume, a PersistentVolumeClaim for requesting access to the PersistentVolume resource, a service for having a static endpoint for the MySQL database, and a deployment for running and managing the MySQL pod.

The MySQL container reads database credentials from environment variables. The environment variables access these credentials from Kubernetes secrets.

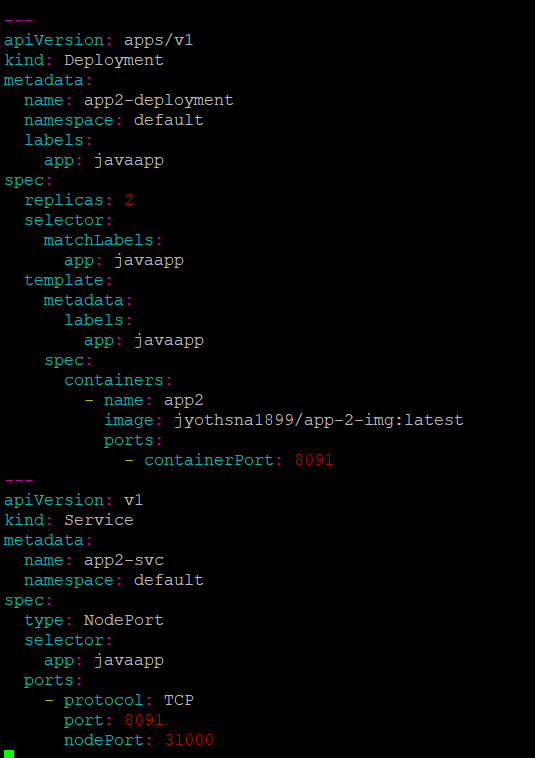
Yaml file for frontend application:

****

**Yaml file for Mss-user-auth:**

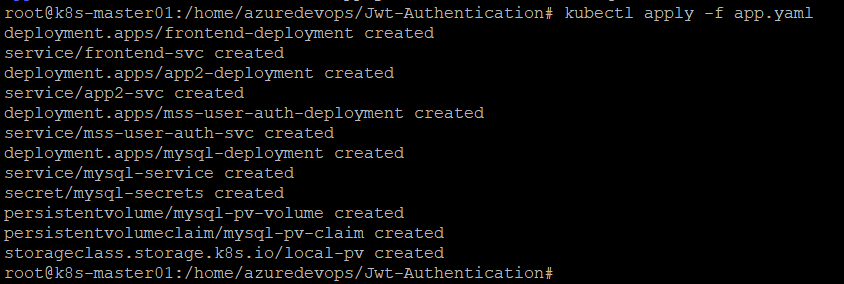
****

**Yaml file for app-2-deployment:**

****

Deploying the application using kubectl command

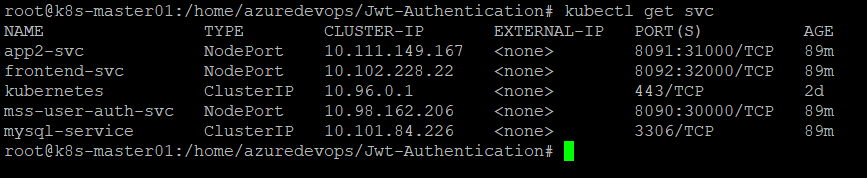
**$kubectl apply -f app.yaml**



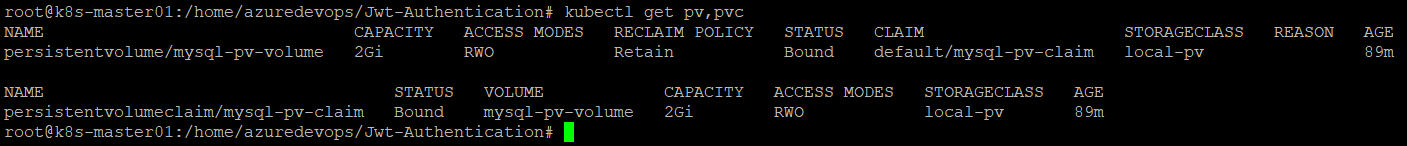
**$kubectl get pods**



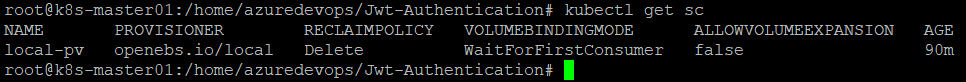
**$kubectl get svc**



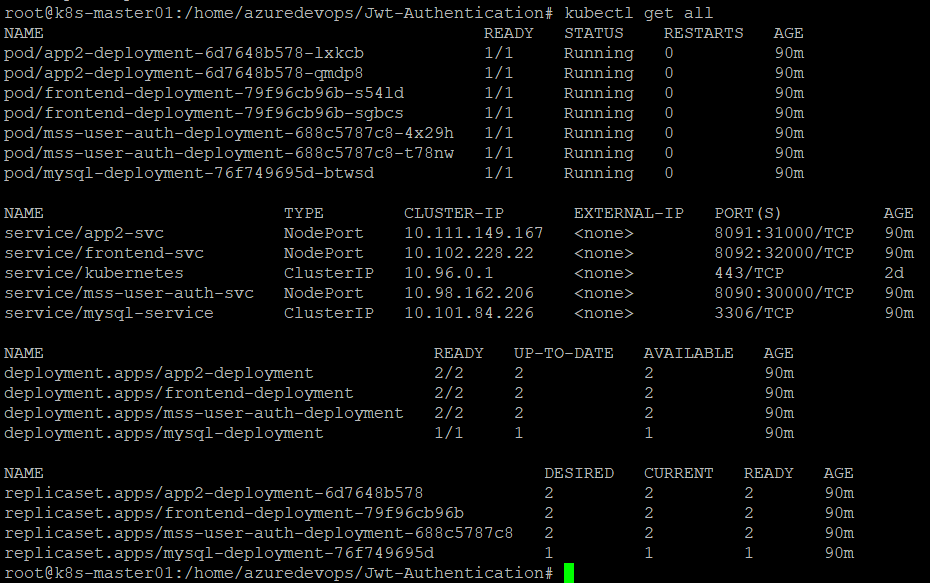
**$kubectl get pv,pvc**



**$kubectl get sc**

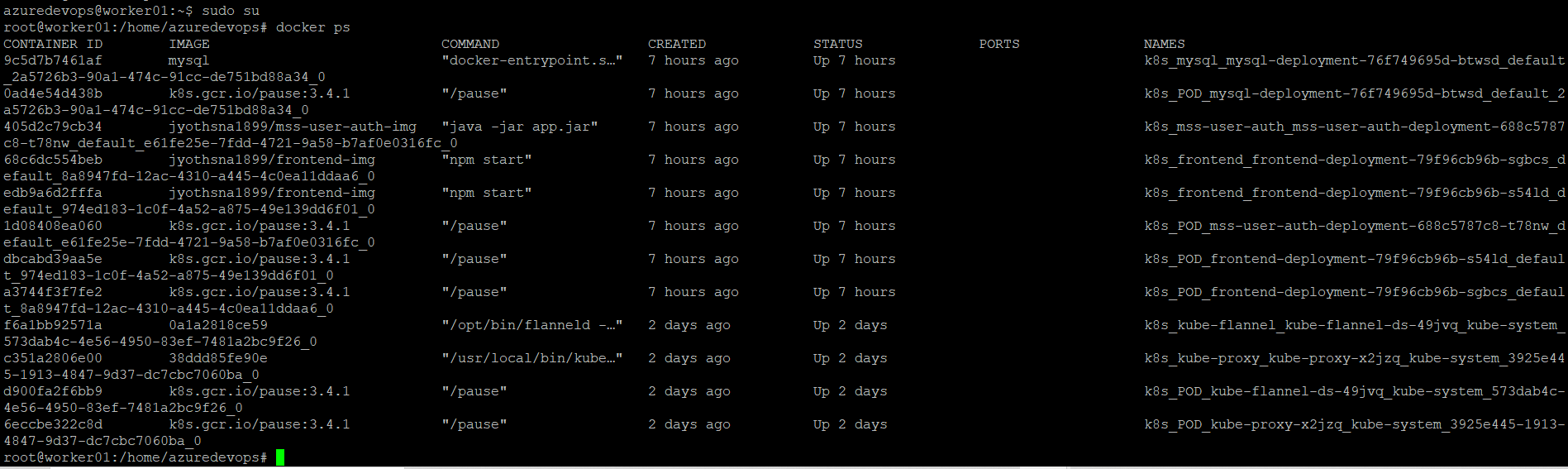
****

**$kubectl get all**



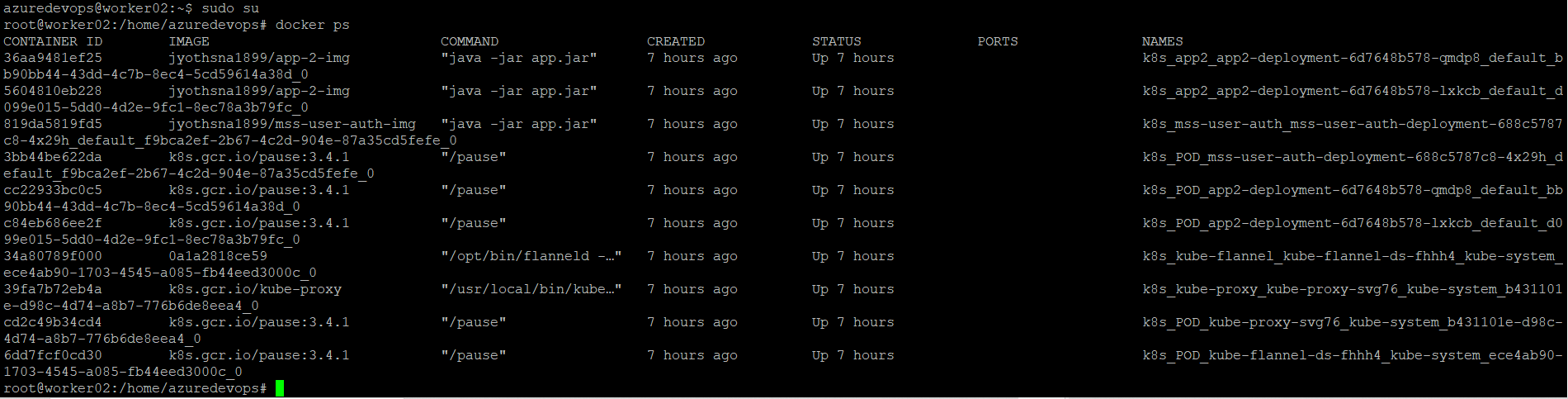
Containers running in worker node 1

**$docker ps**



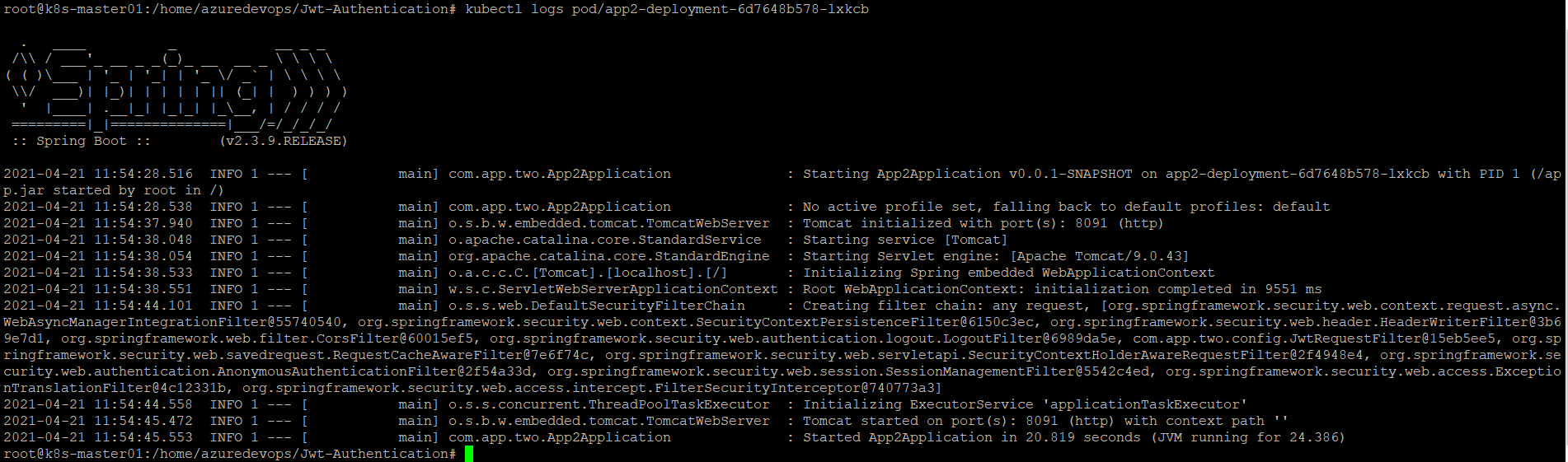
**Workernode 2**

**$docker ps**

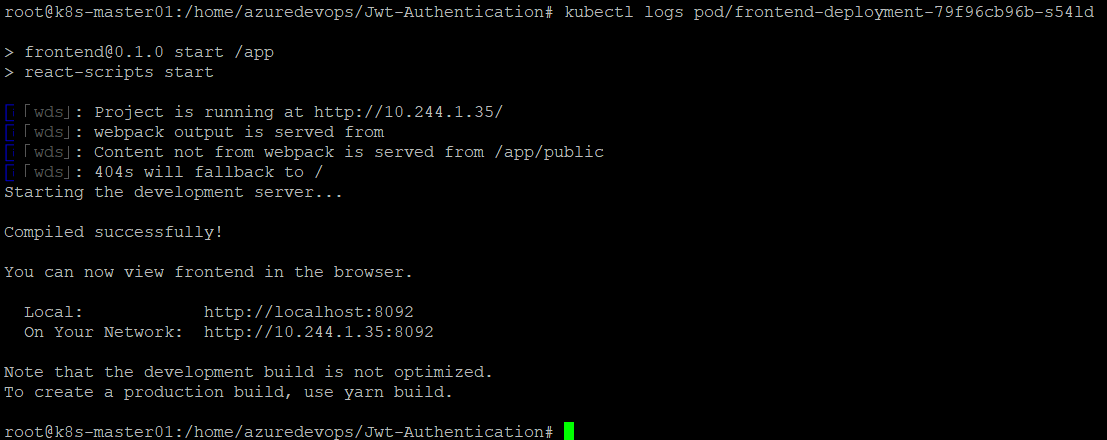
****

Displaying logs of pods

**$kubectl logs pod/app2-deployment-6d7648b578-lxkcb**

****

**$kubectl logs pod/frontend-deployment-79f96cb96b-s54ld**

****

**$kubectl describe pod mysql-deployment-76f749695d-btwsd**

