

Everything inside the K-Cluster is known as Objects.

Green ones are known as deployment objects.

Oranges ones are known as service objects.

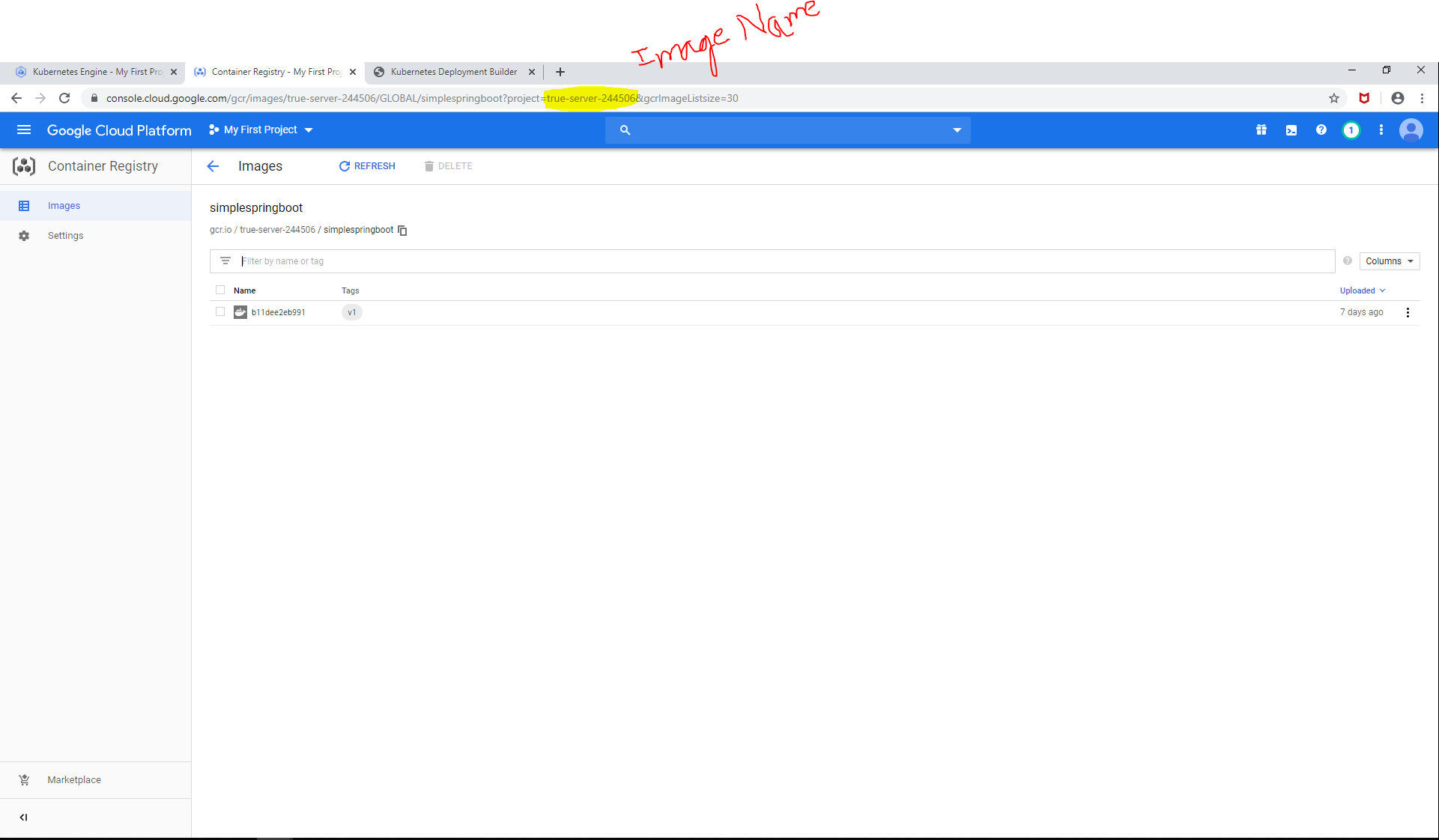
In order to create a yml file quickly, use the link below

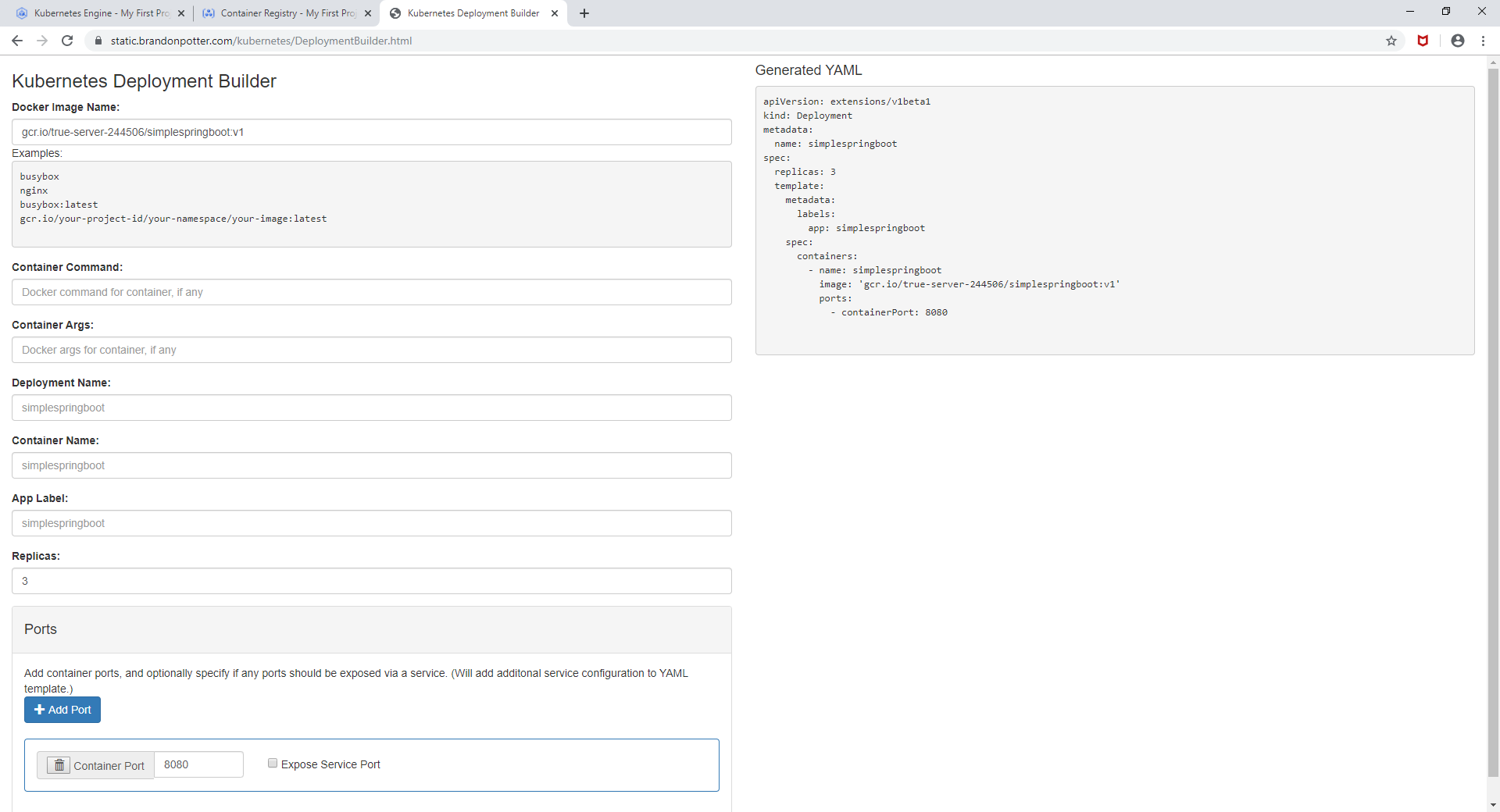
<https://static.brandonpotter.com/kubernetes/DeploymentBuilder.html>

image from the previous document

gcr.io/$GOOGLE\_CLOUD\_PROJECT/simplespringboot:v1 , Image name becomes

**gcr.io/true-server-244506/simplespringboot:v1**





apiVersion: extensions/v1beta1

kind: Deployment

metadata:

name: simplespringboot

spec:

replicas: 3

template:

metadata:

labels:

app: simplespringboot

spec:

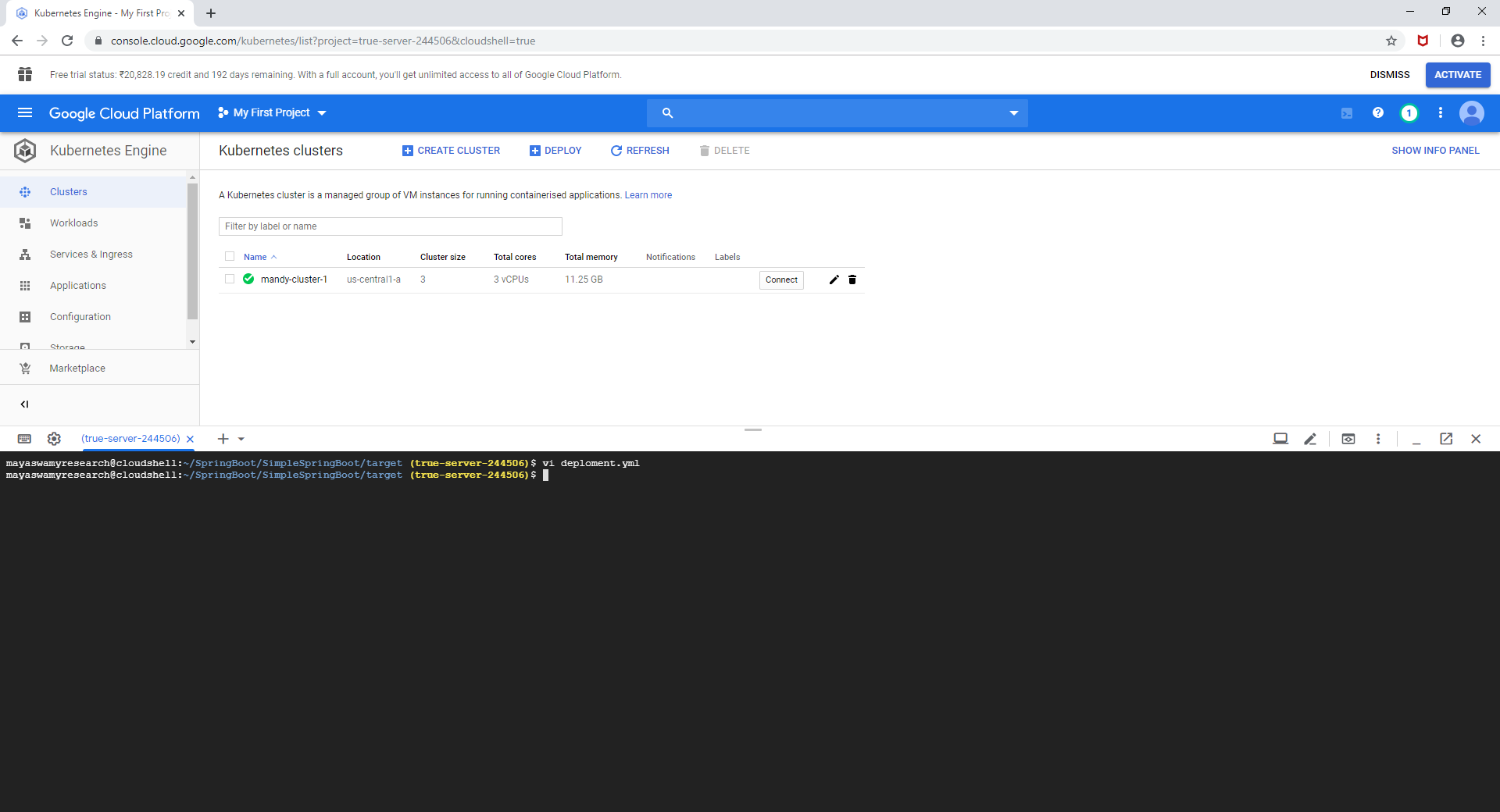
containers:

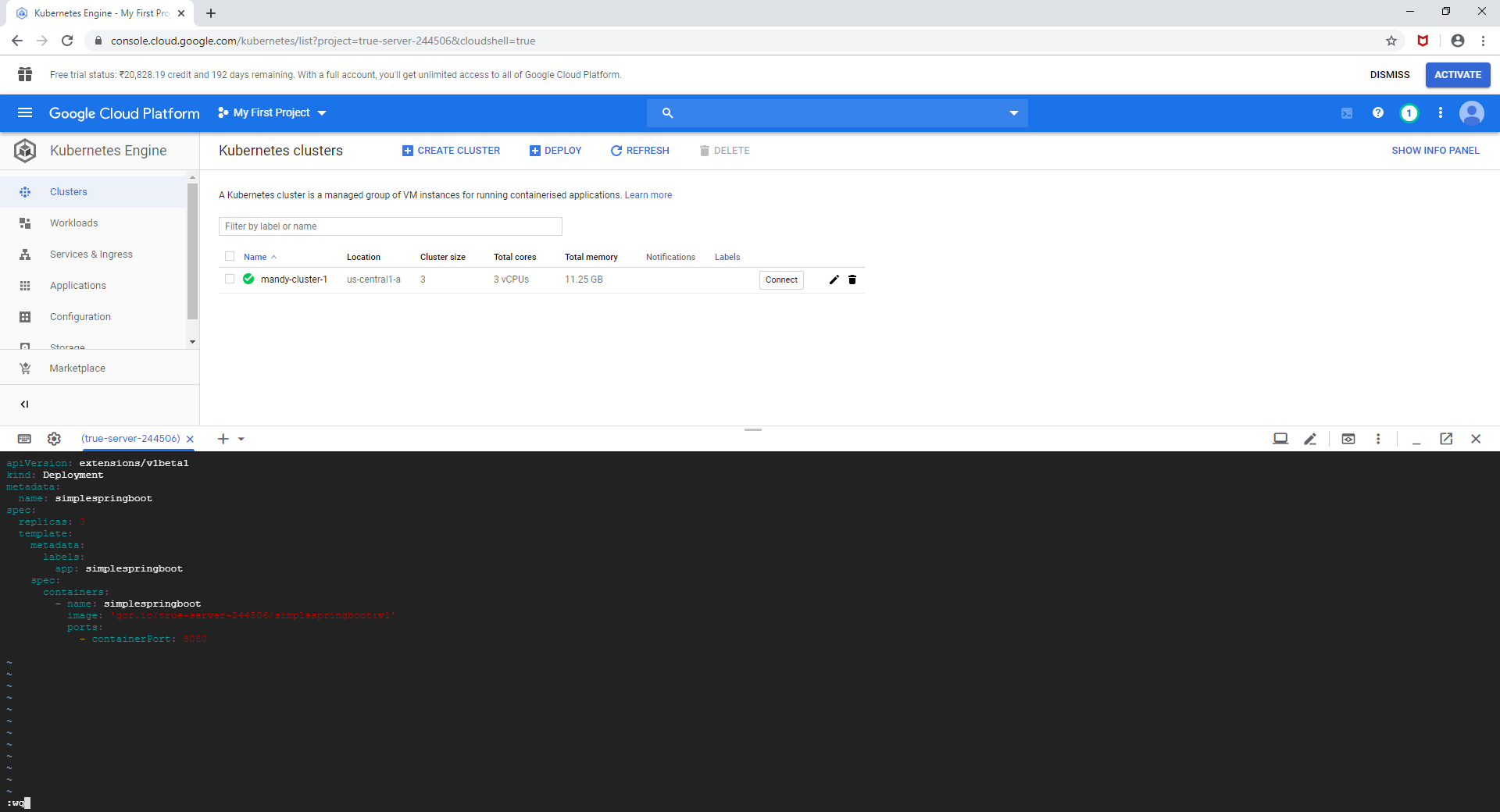
- name: simplespringboot

image: 'gcr.io/true-server-244506/simplespringboot:v1'

ports:

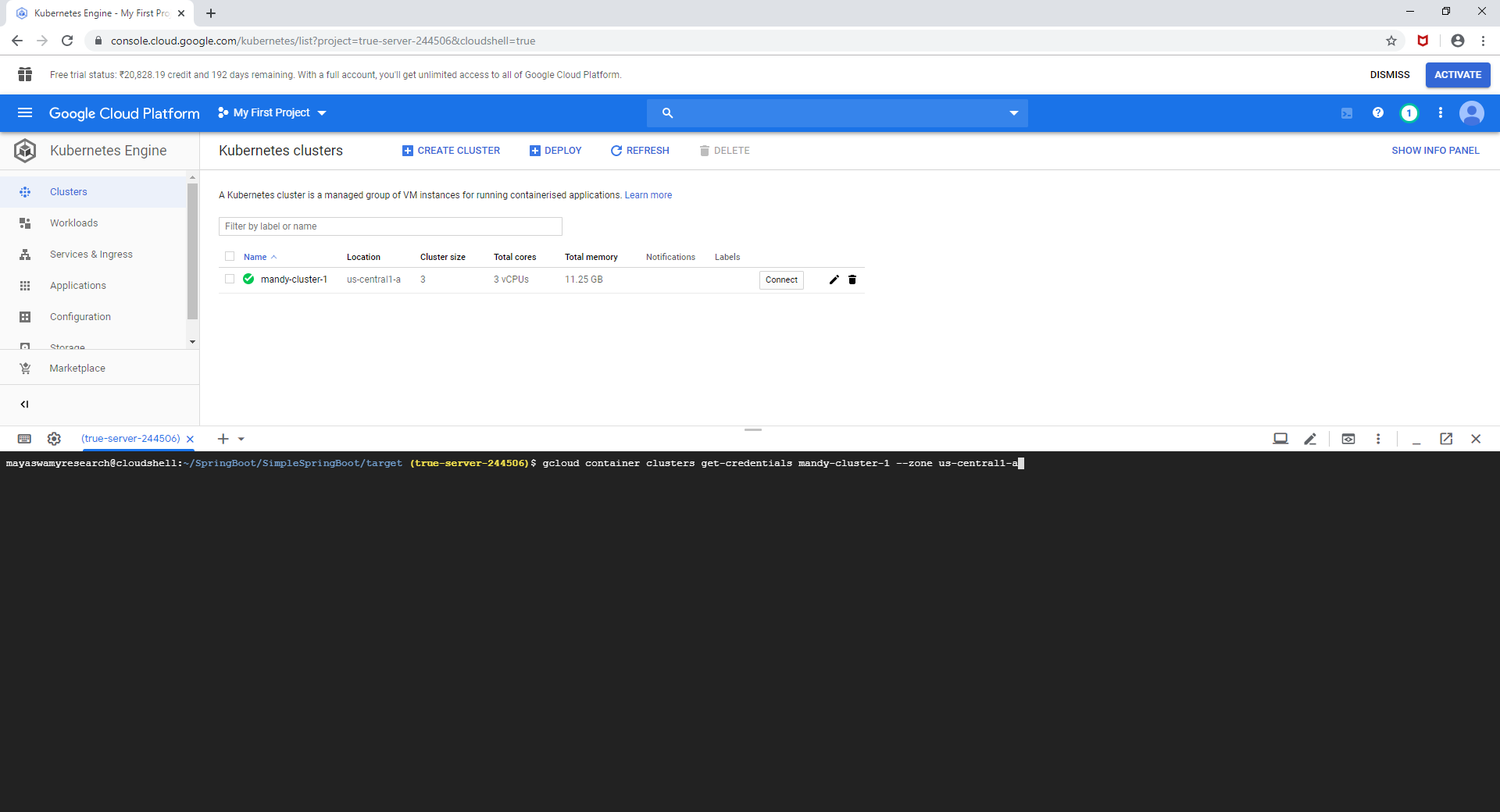
- containerPort: 8080

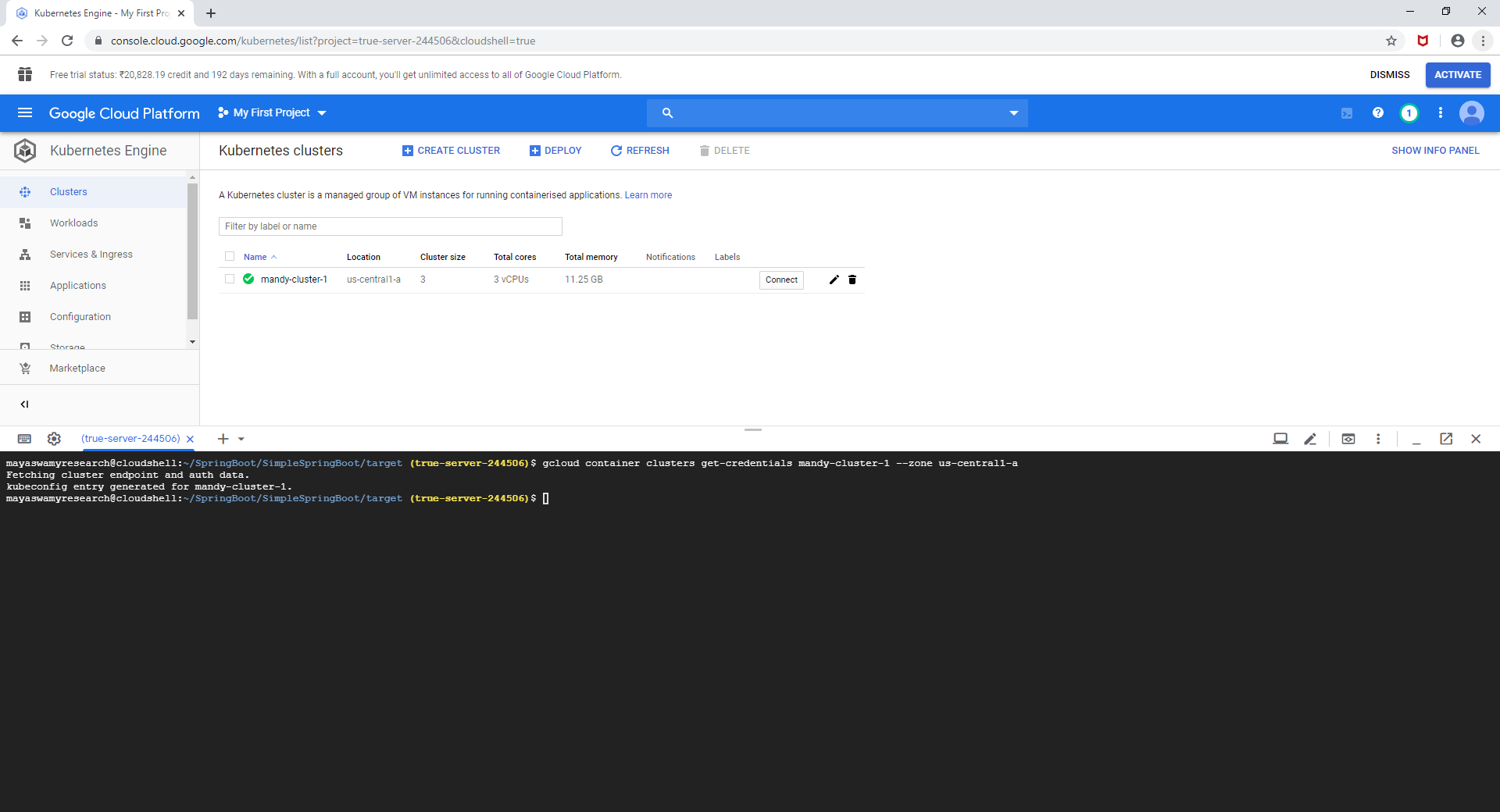




Now Login current terminal into my cluster.

**gcloud container clusters get-credentials mandy-cluster-1 --zone us-central1-a**

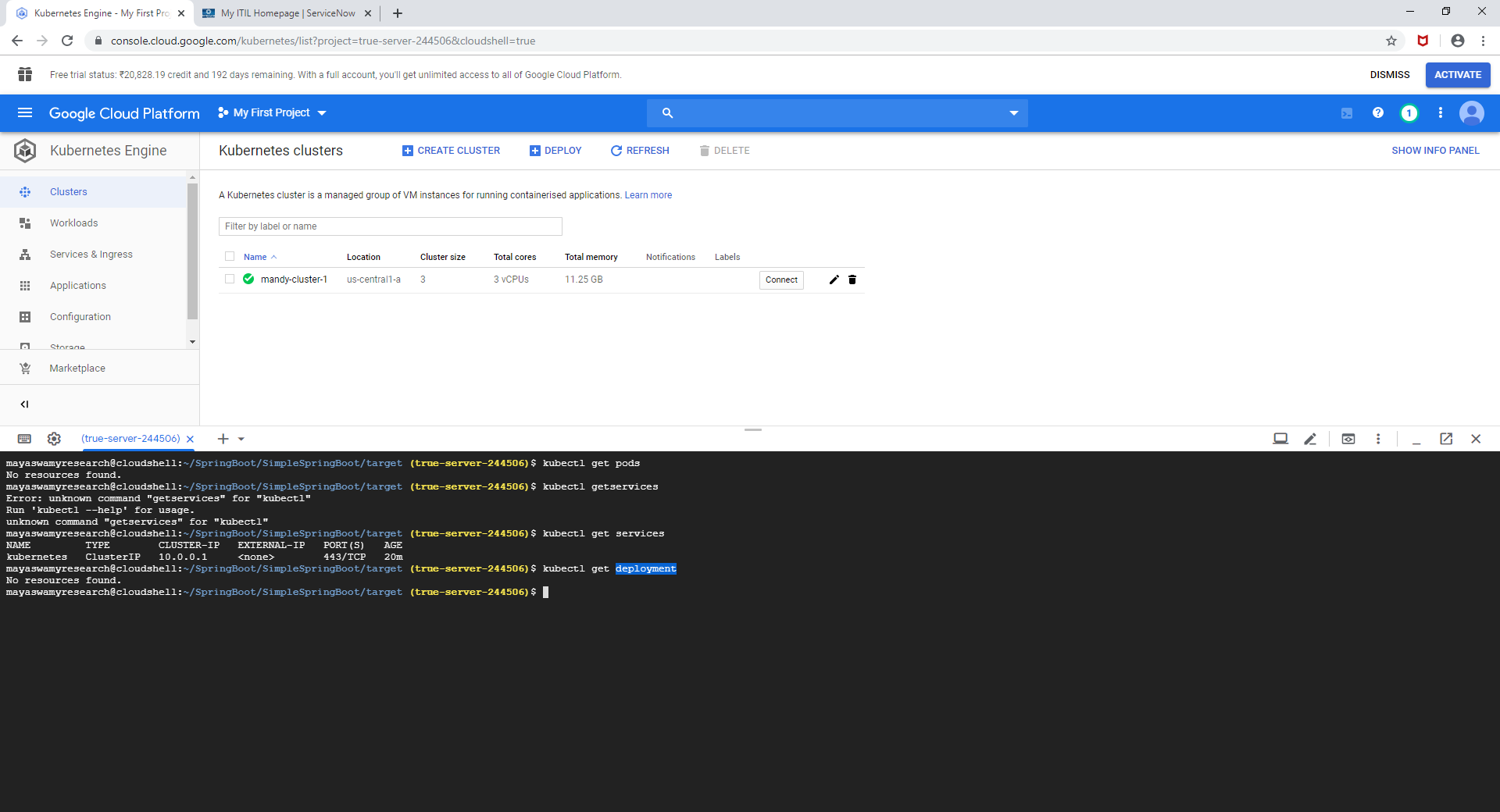




kubectl get pods

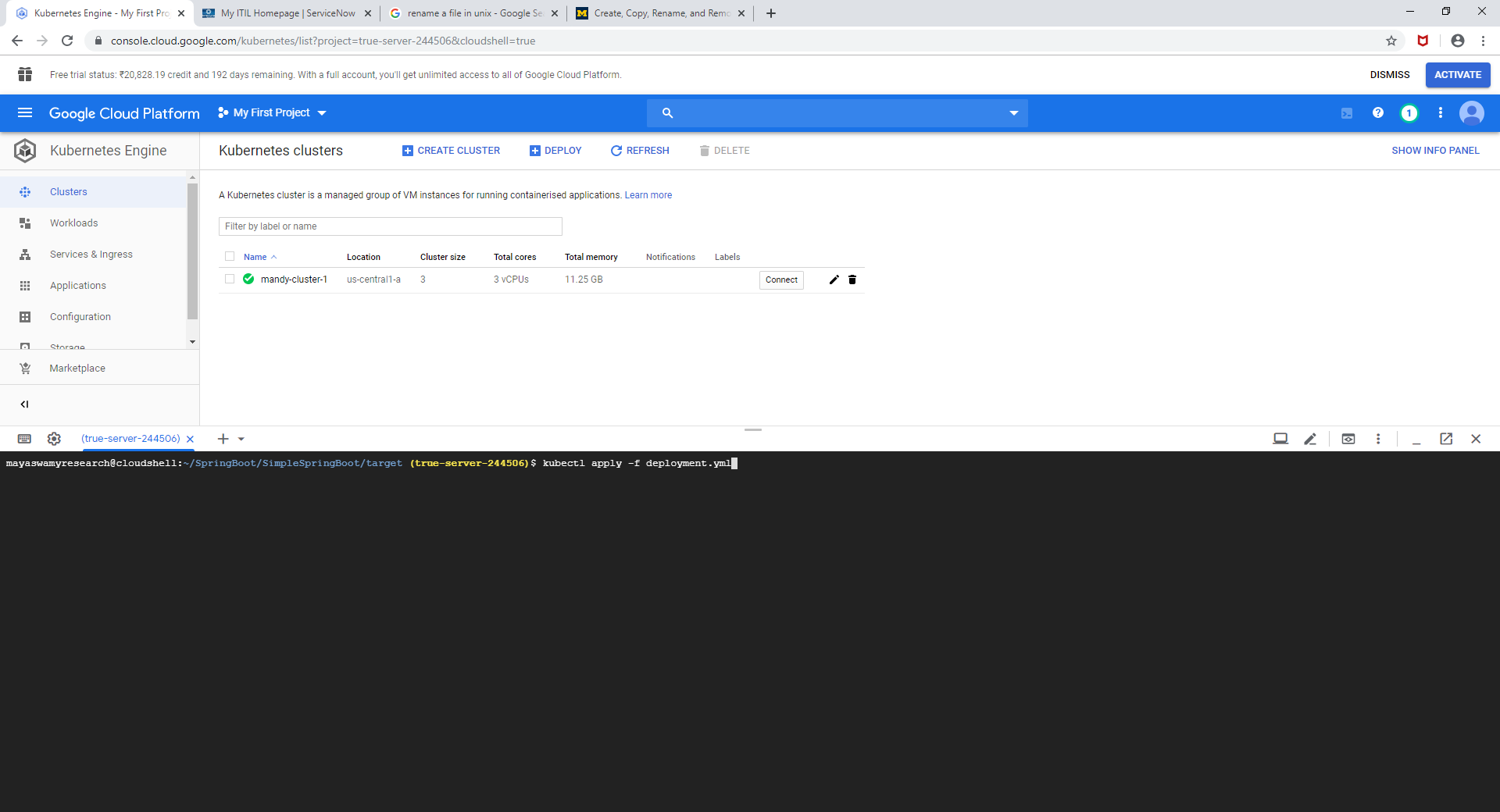
kubectl get services

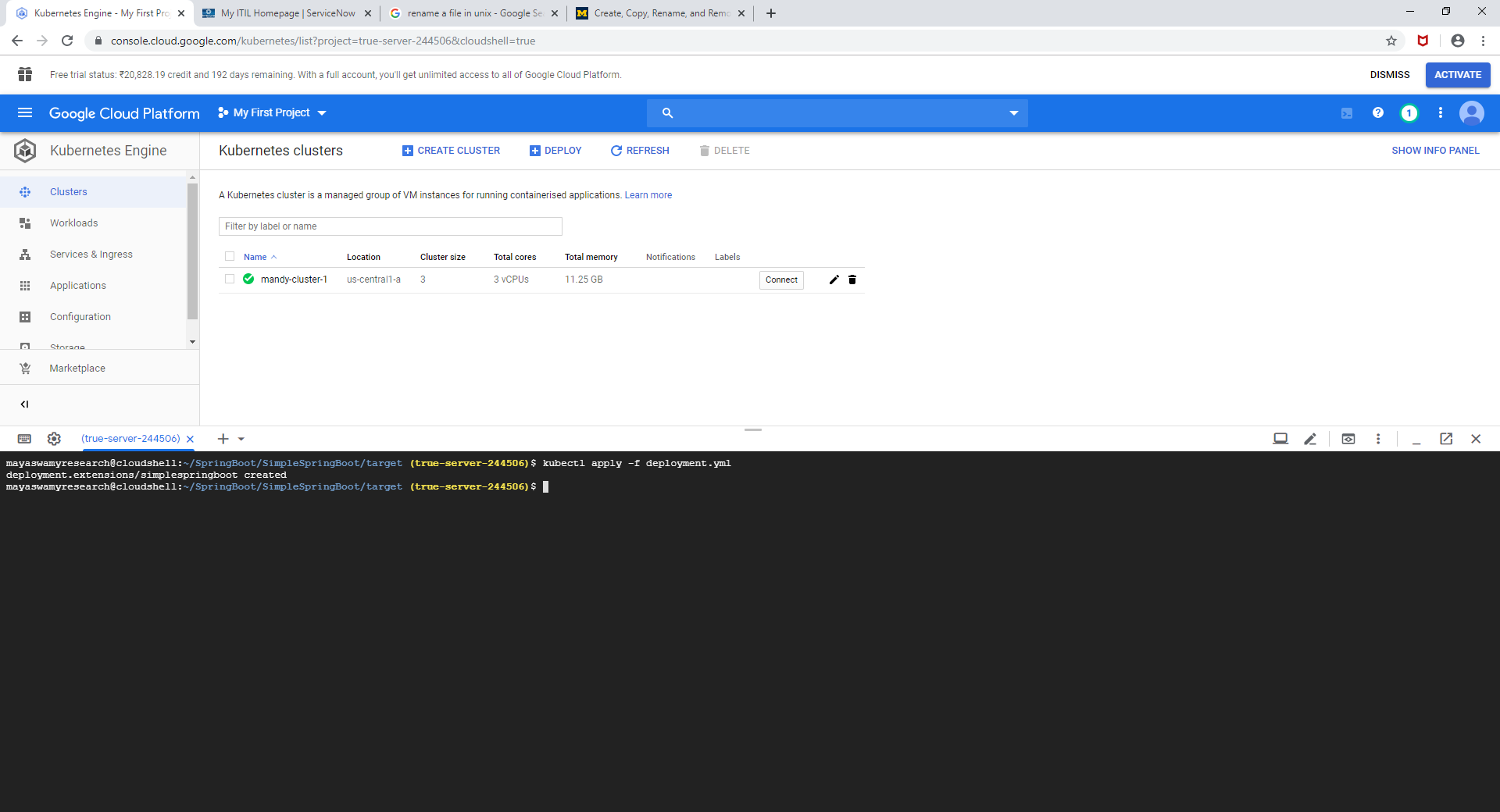
kubectl get deployment



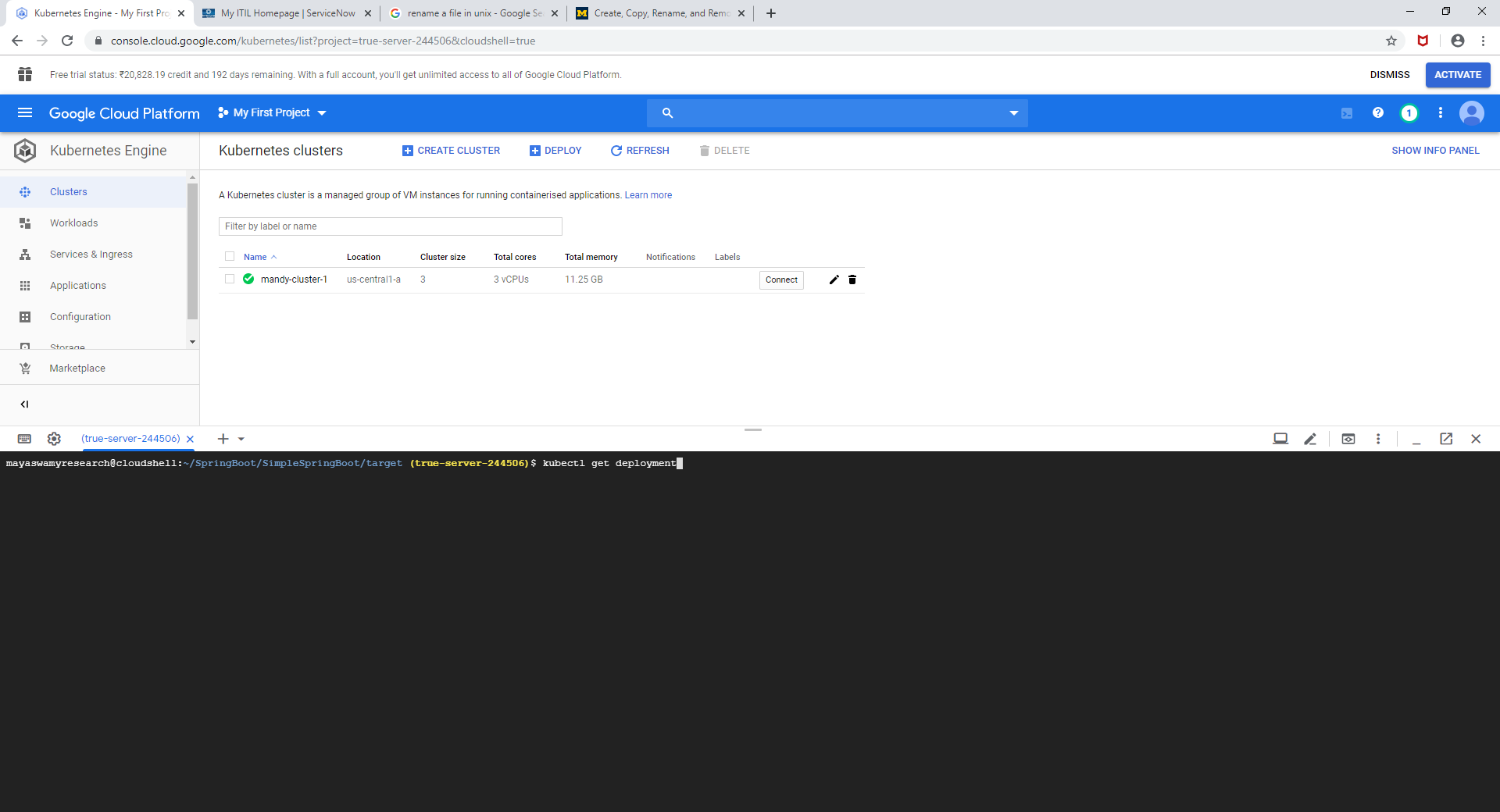
In order to create a deployment object using deployment.yml , use the command

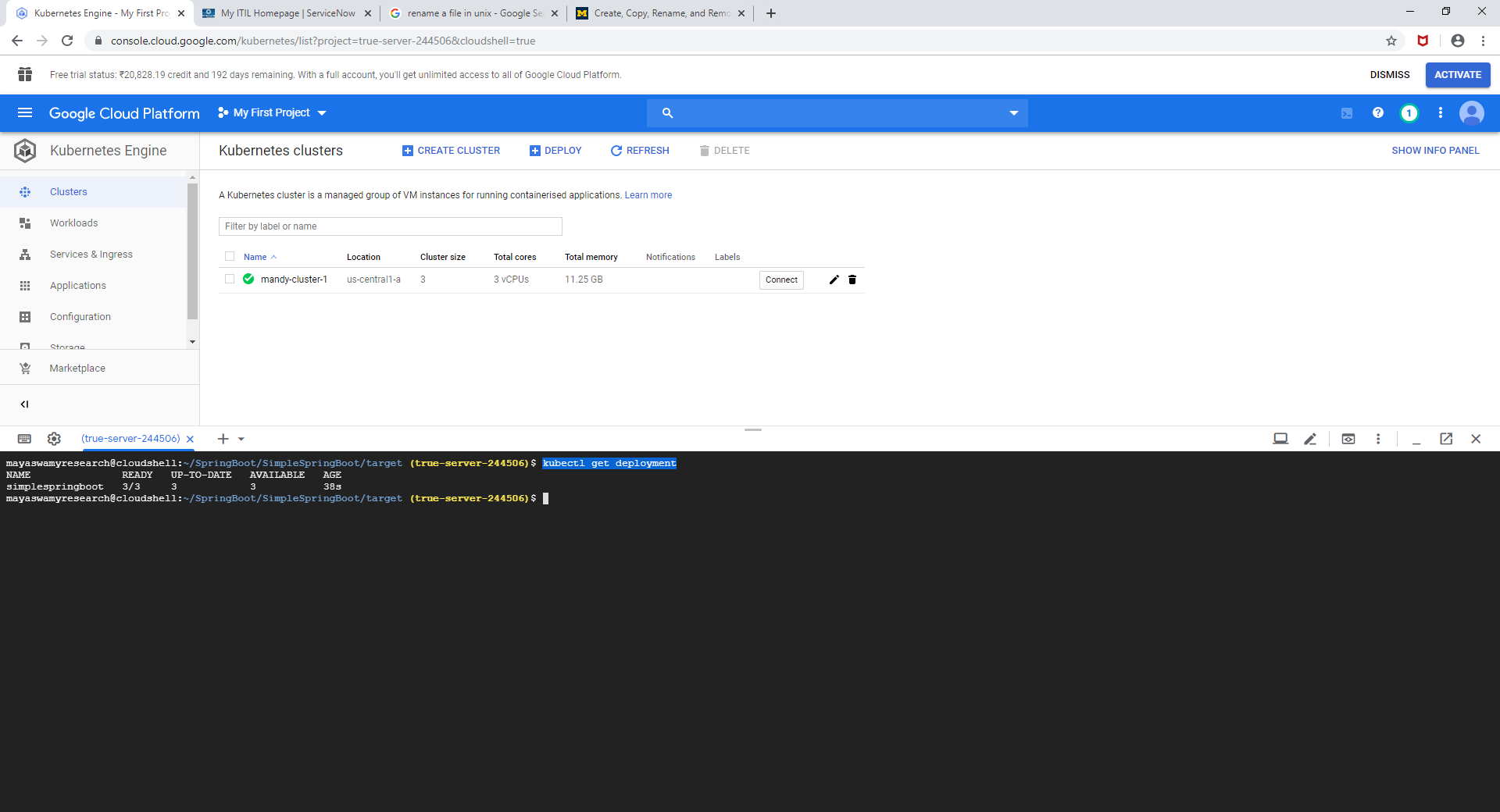
**kubectl apply -f deployment.yml**



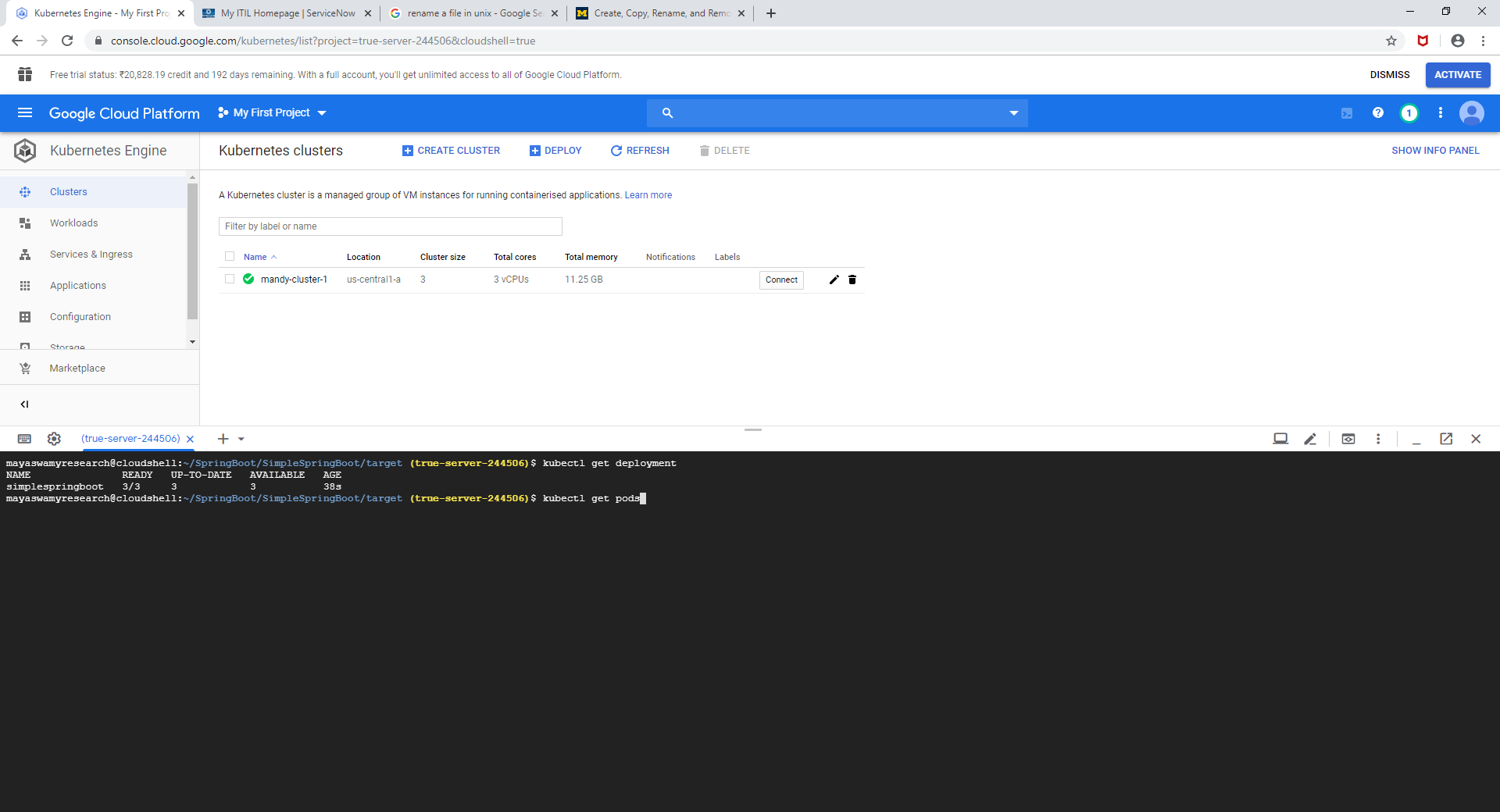


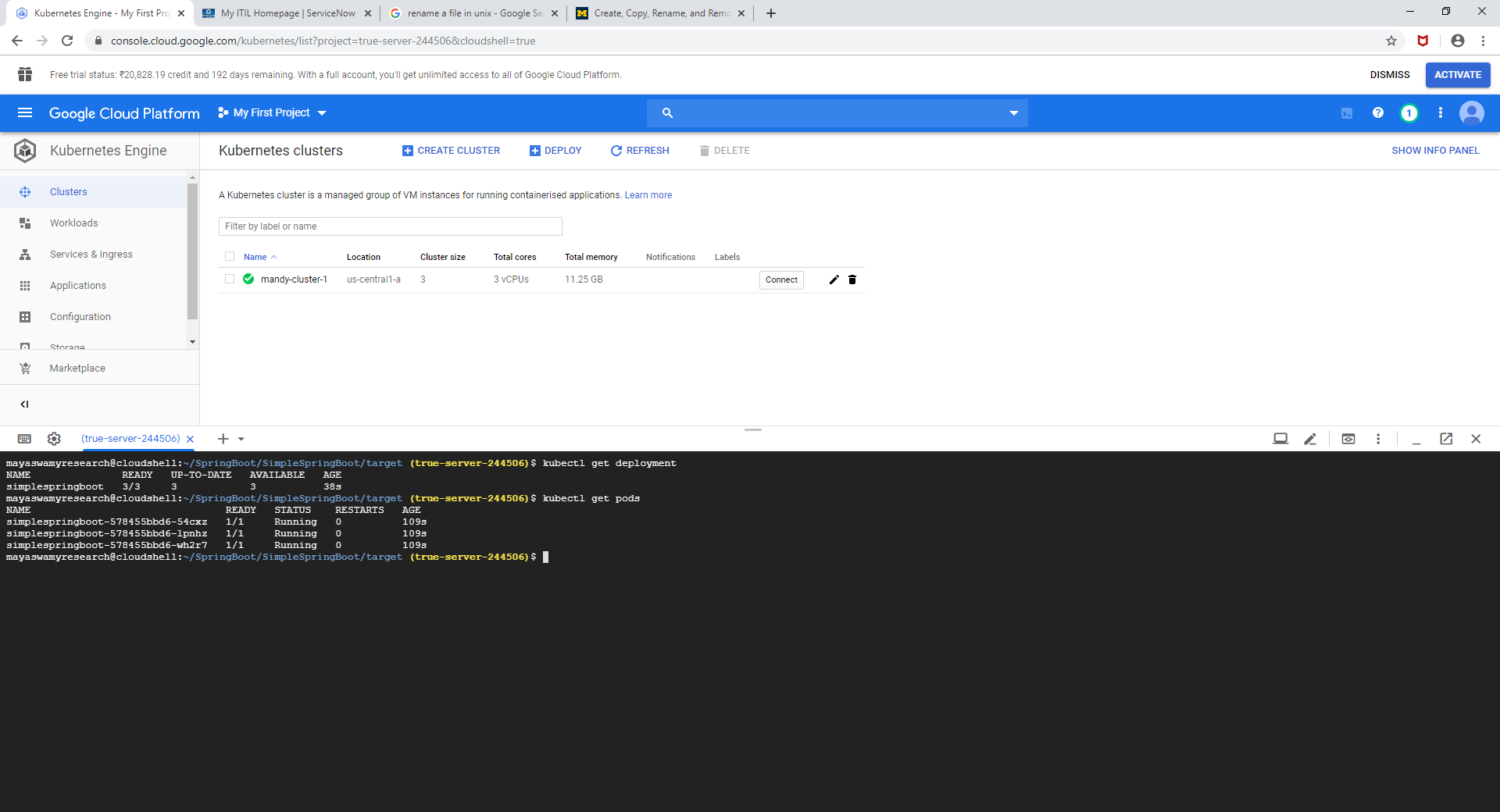
**kubectl get deployment**



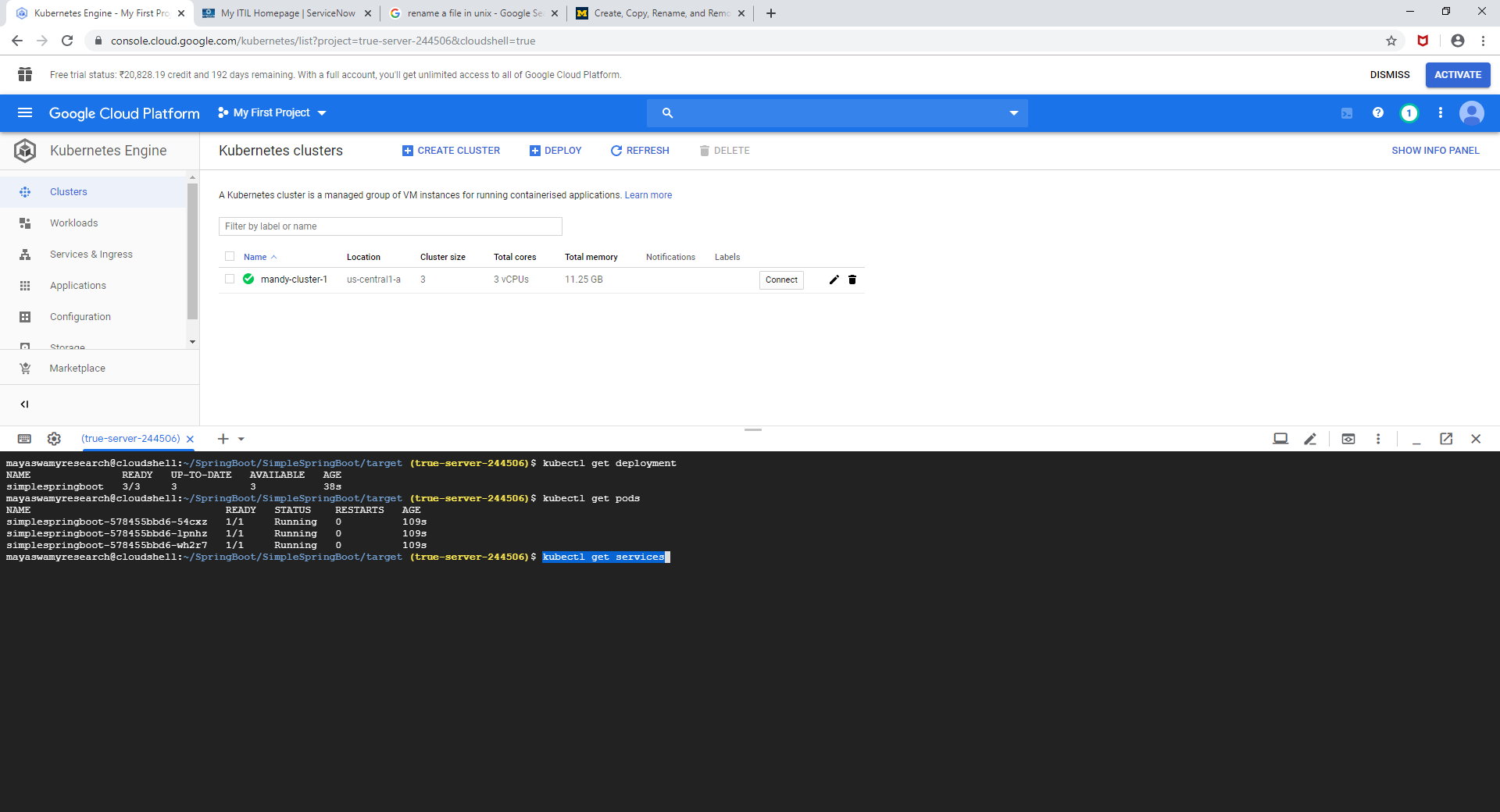


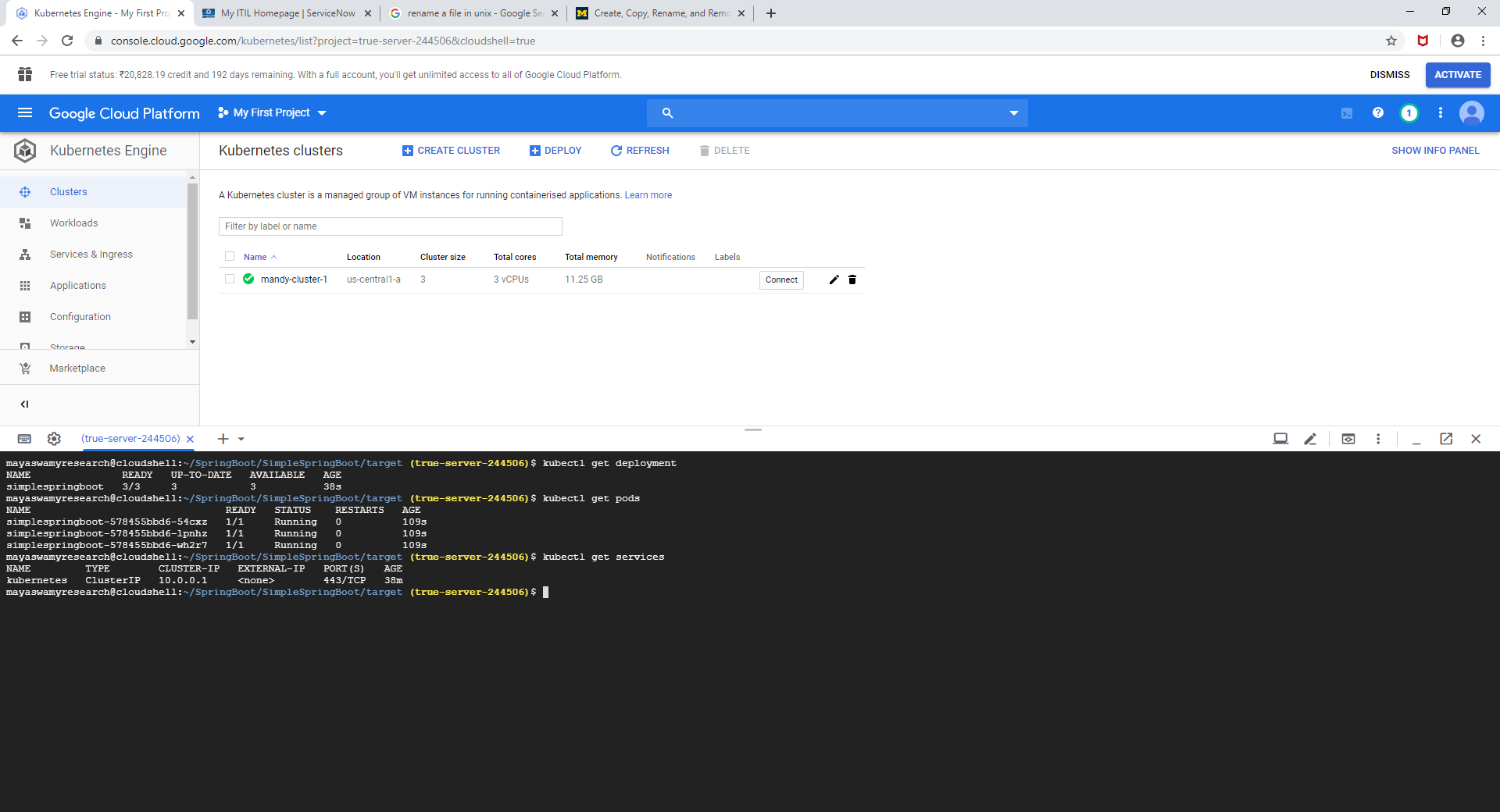
**kubectl get pods**





**kubectl get services**

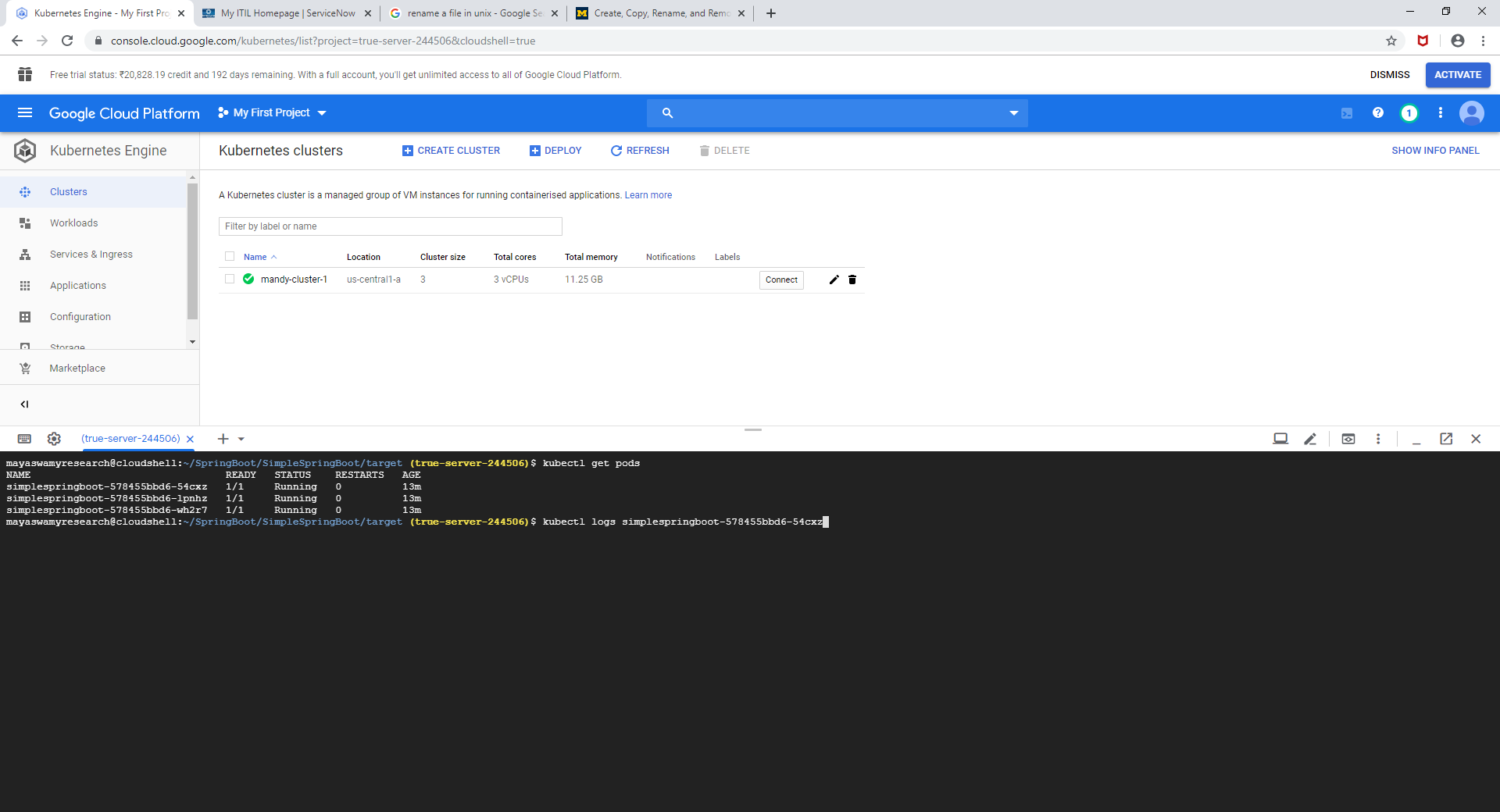


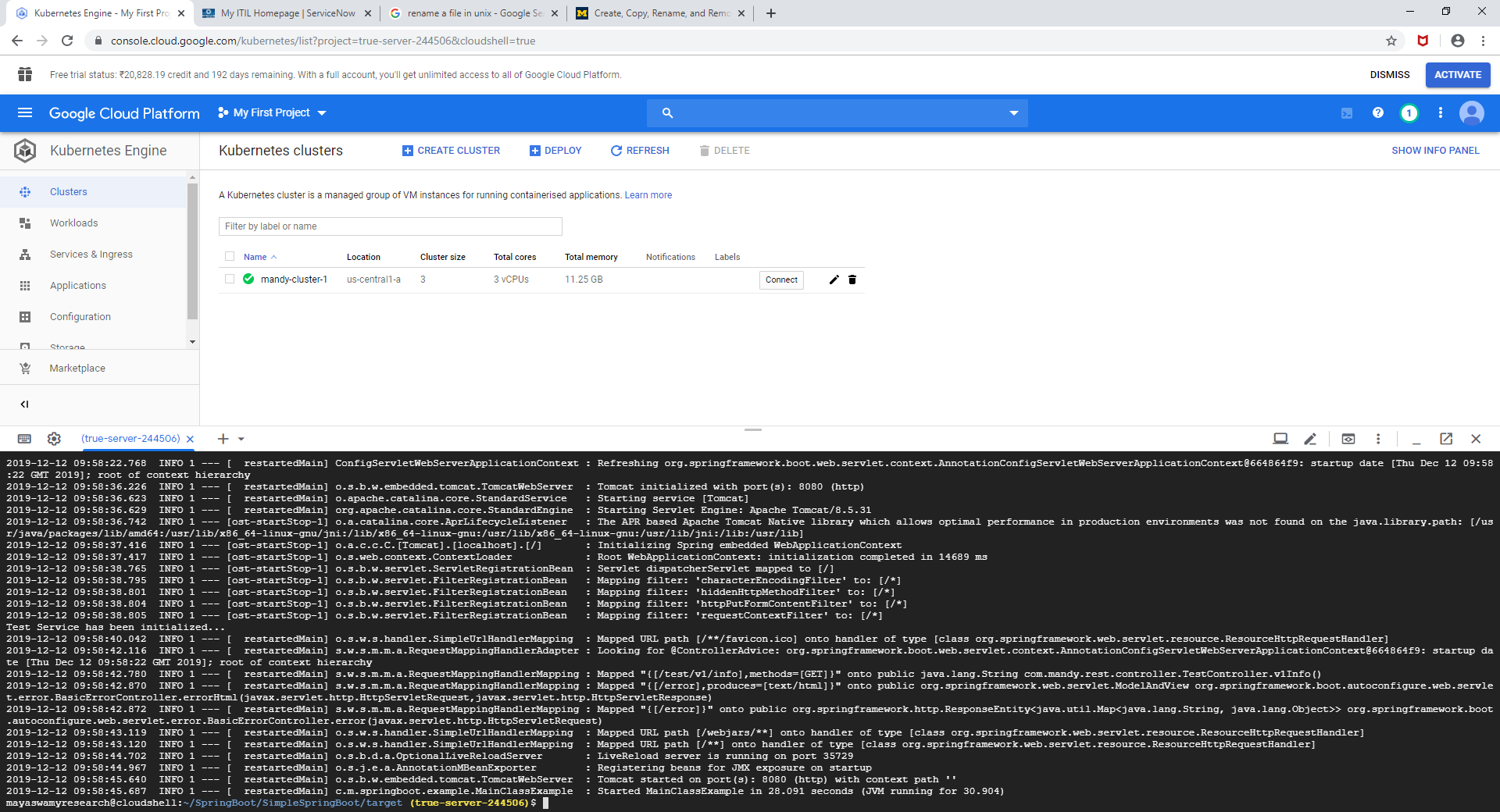


In order to check the logs on any of the deployments a command can be used ,

**Syntax: -** kubectl logs name\_of\_deployment

**Command: -** kubectl logs simplespringboot-578455bbd6-54cxz

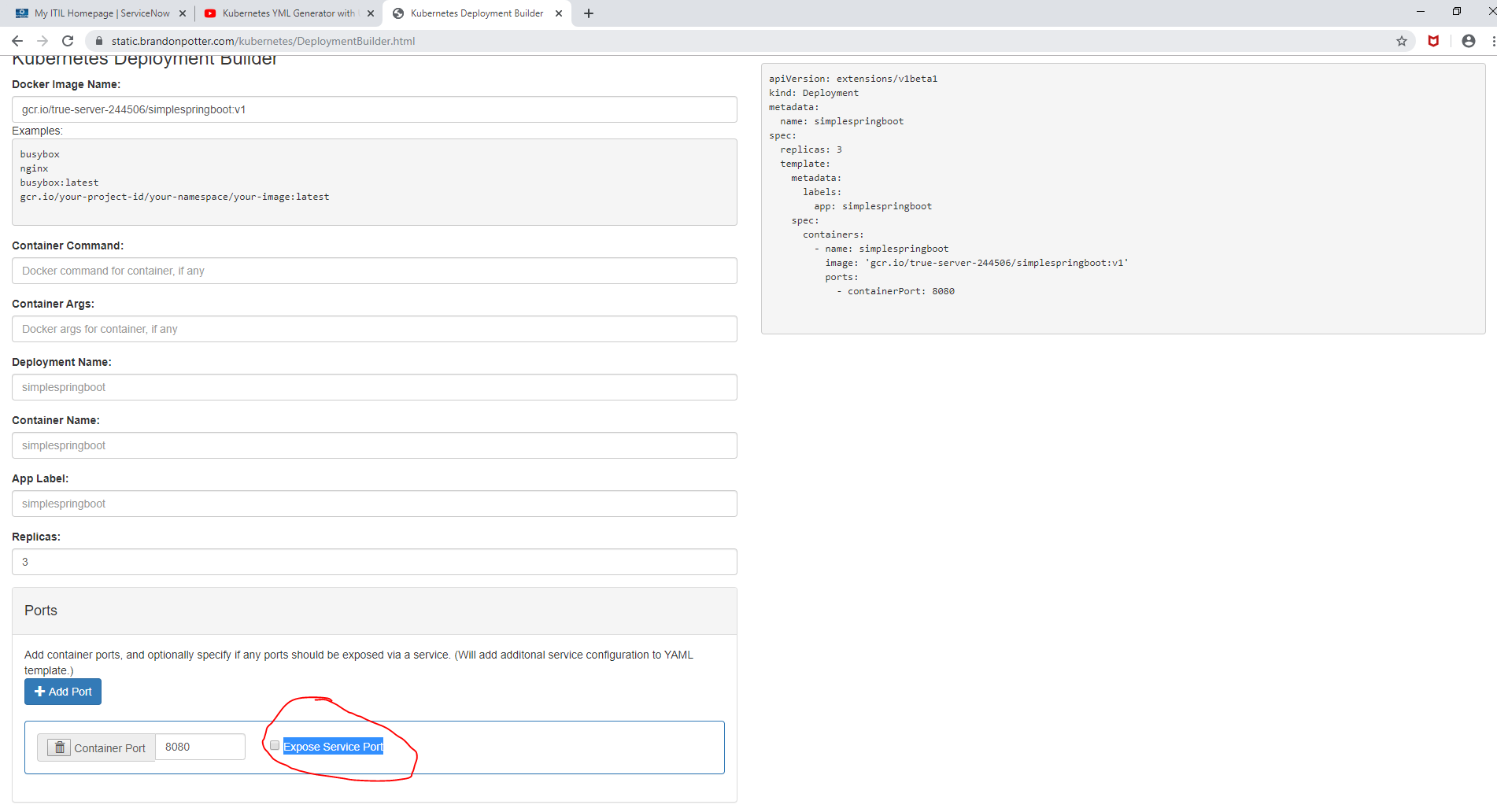


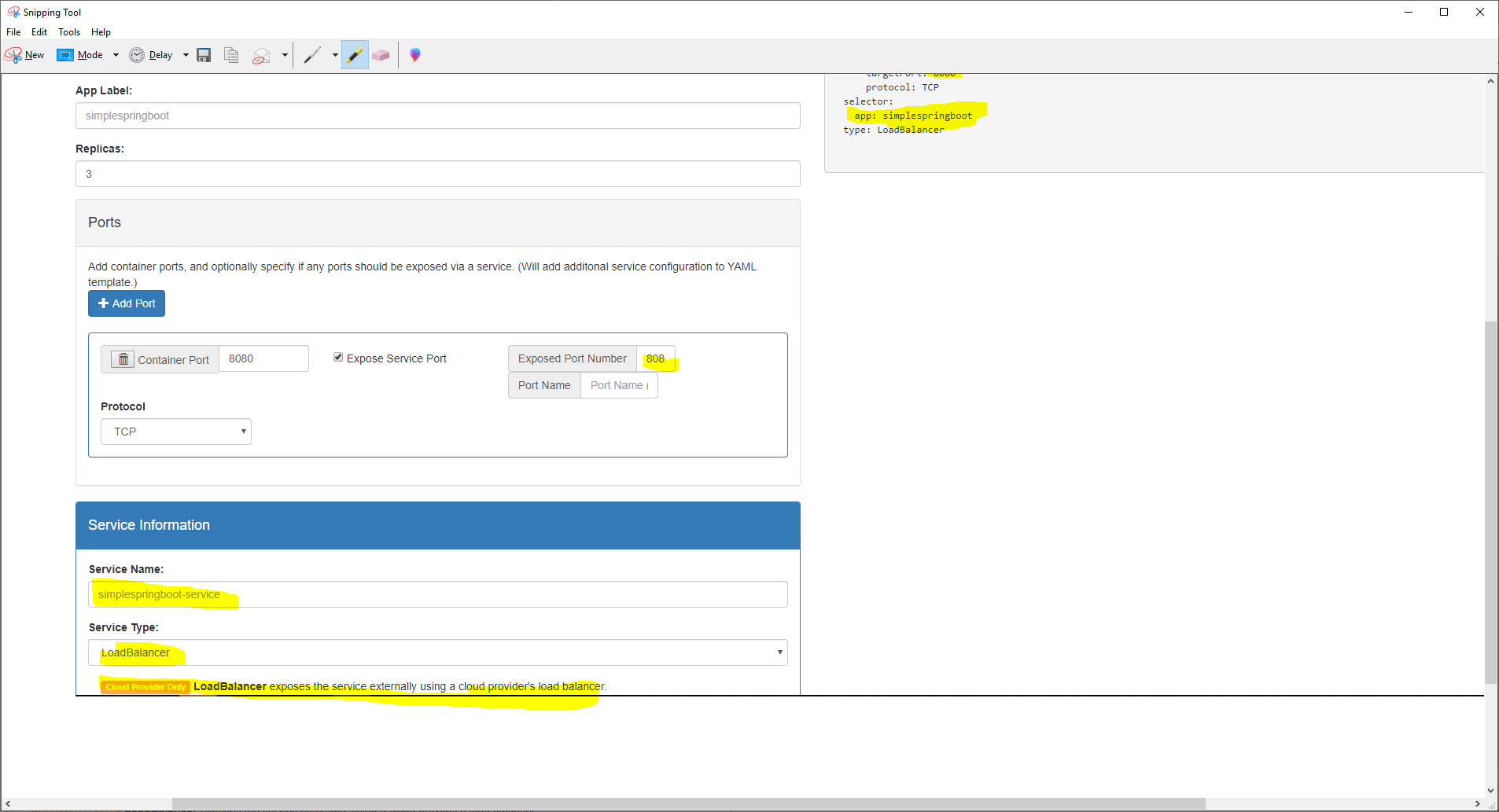


We now have to create a service in order to access the endpoint in the deployment.

The service will have a LoadBalancer which will redirect the request to one of the three running deployments.

To do that , we have to create a service.yml file just like we created for deployment(deployment.yml)





apiVersion: v1

kind: Service

metadata:

name: simplespringboot-service

labels:

name: simplespringboot-service

spec:

ports:

- port: 8080

targetPort: 8080

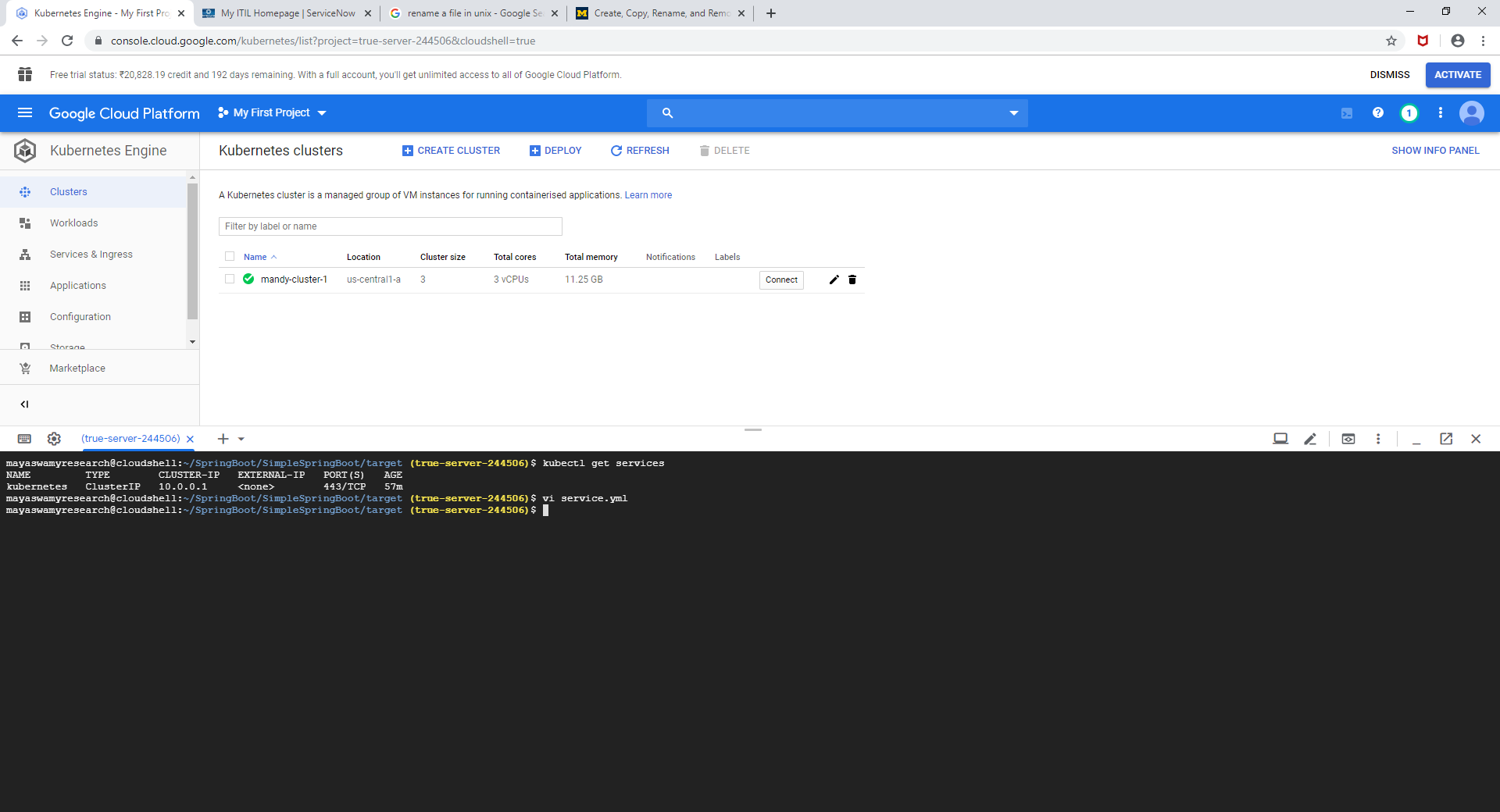
protocol: TCP

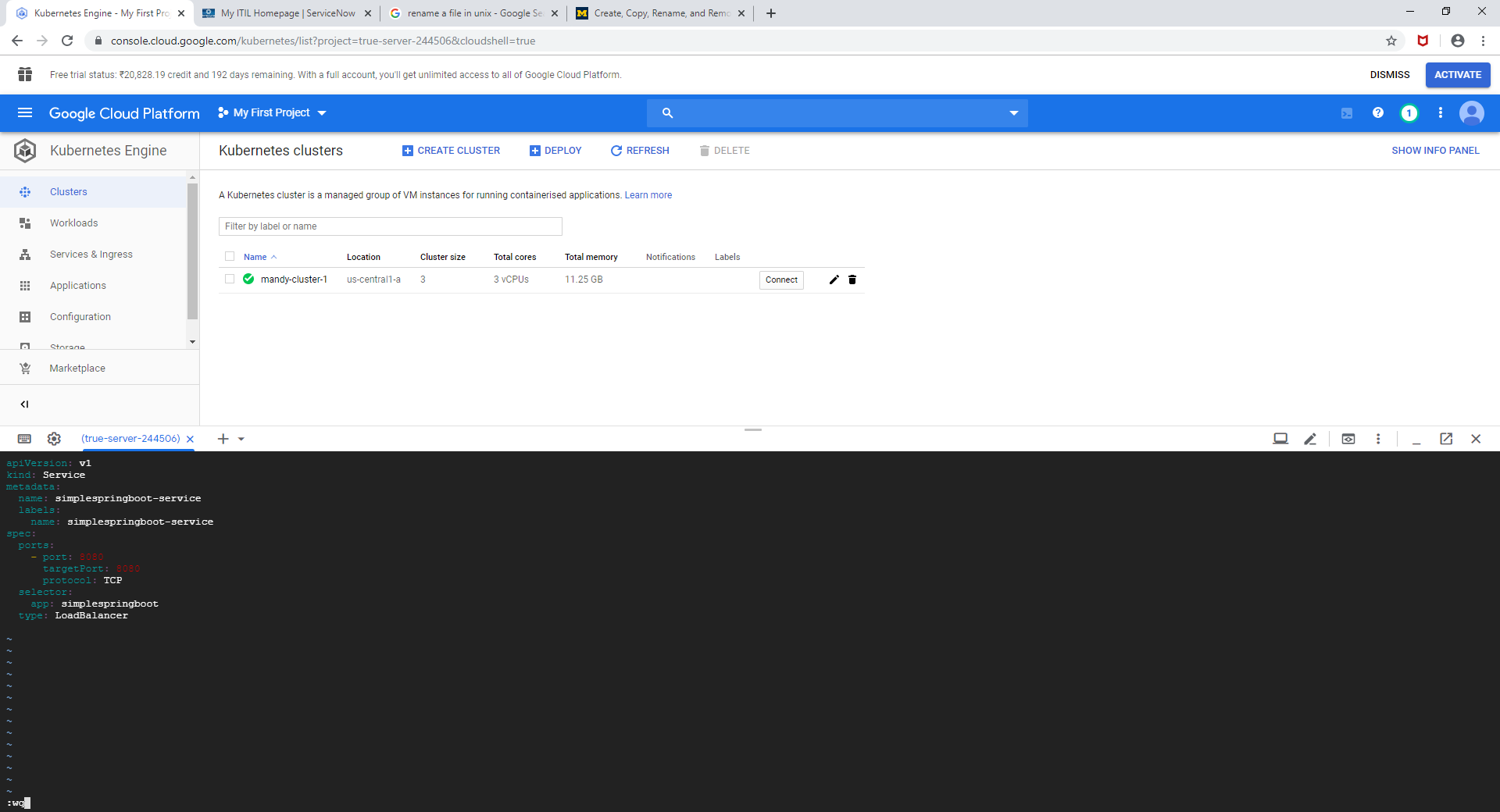
selector:

app: simplespringboot

type: LoadBalancer

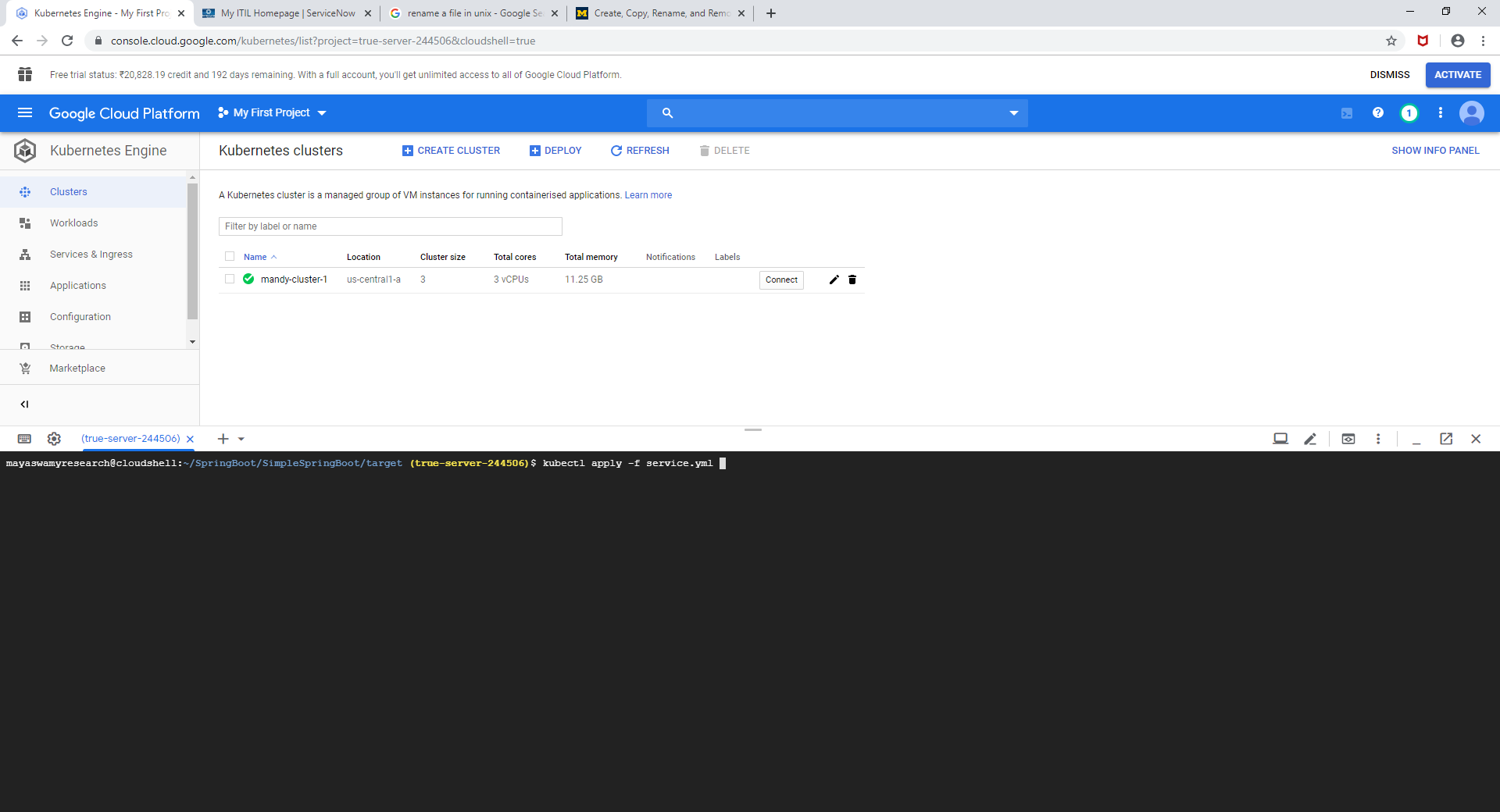
**vi service.yml**

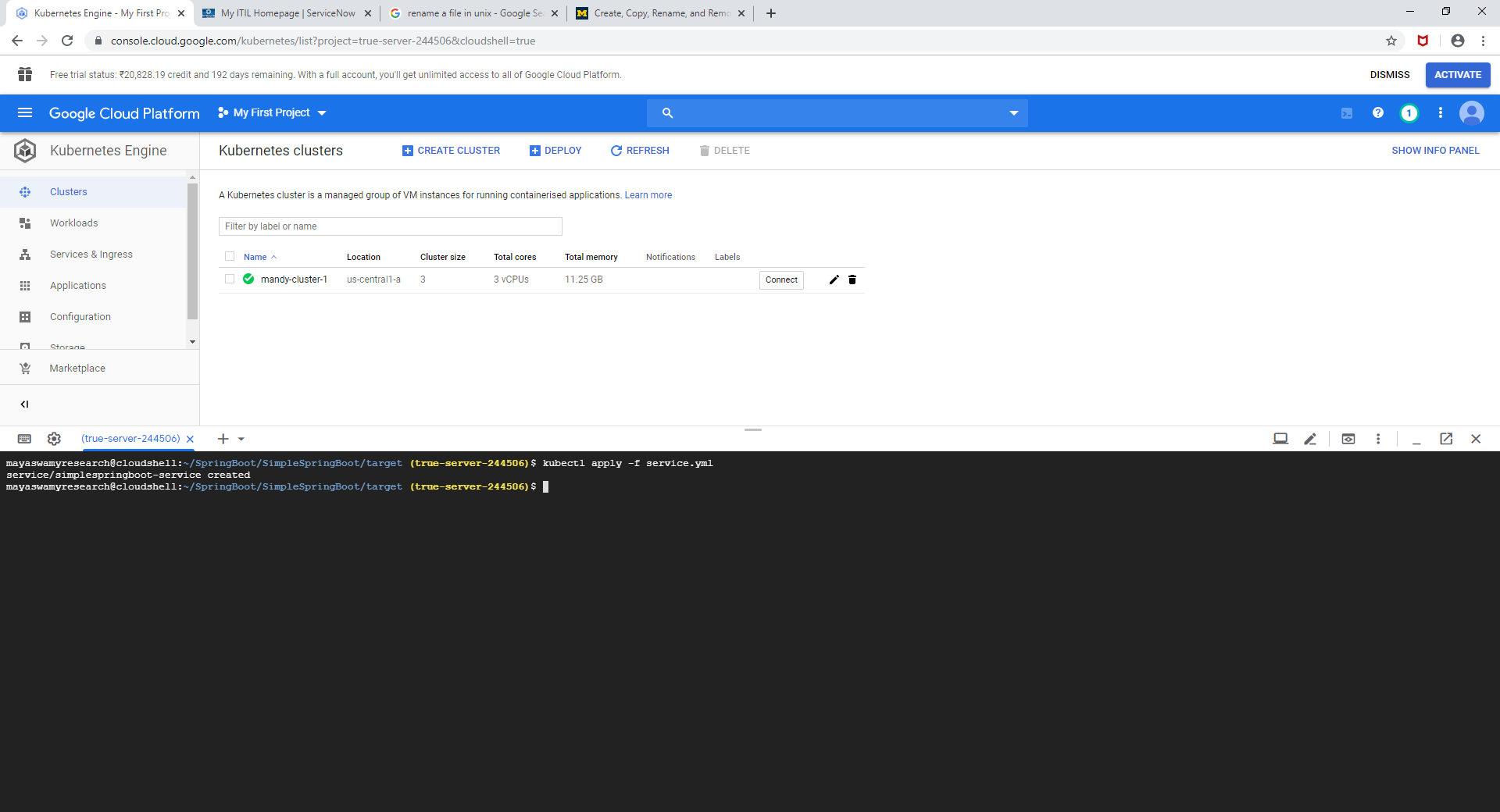




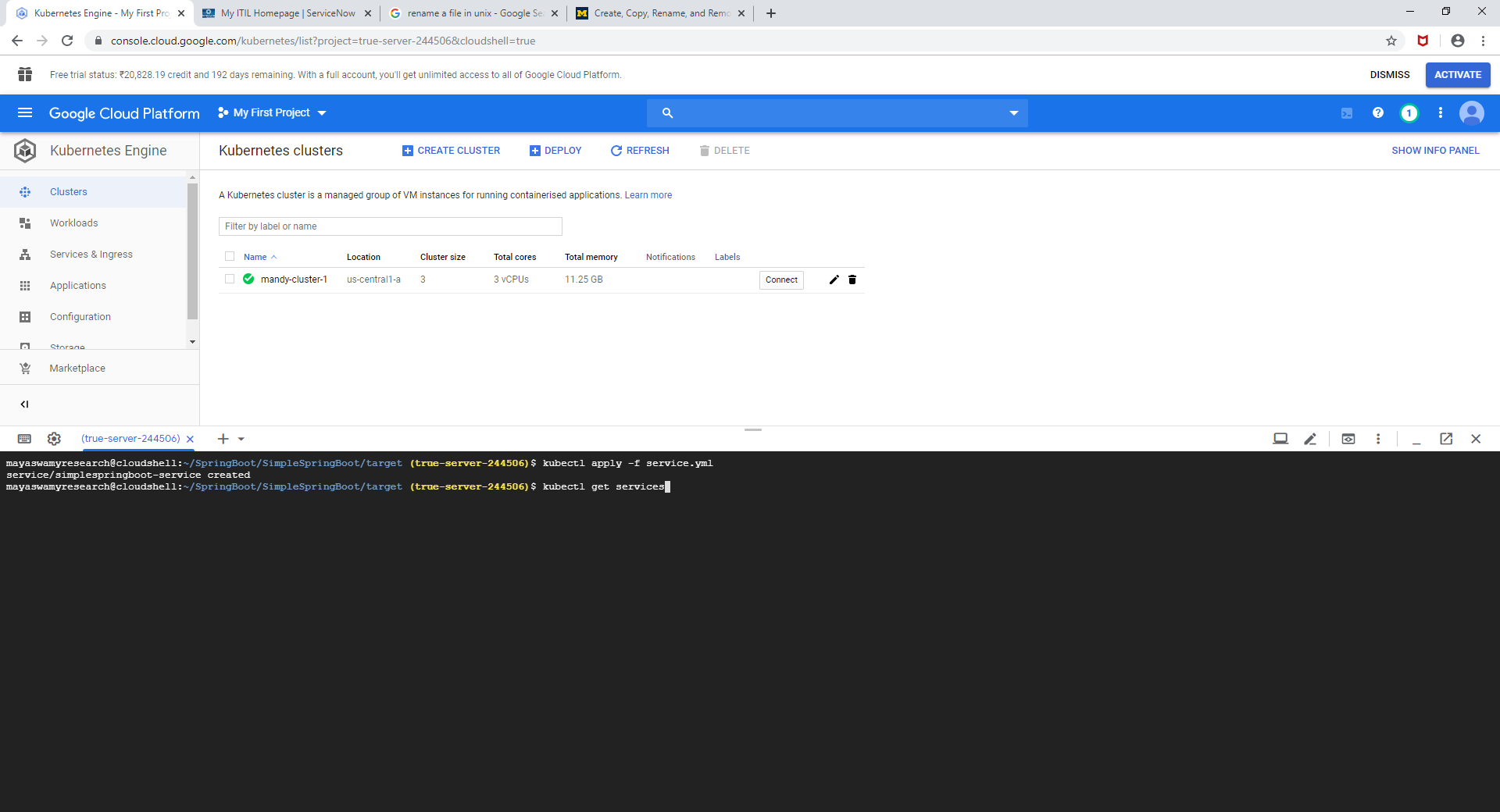
Command to deploy the service

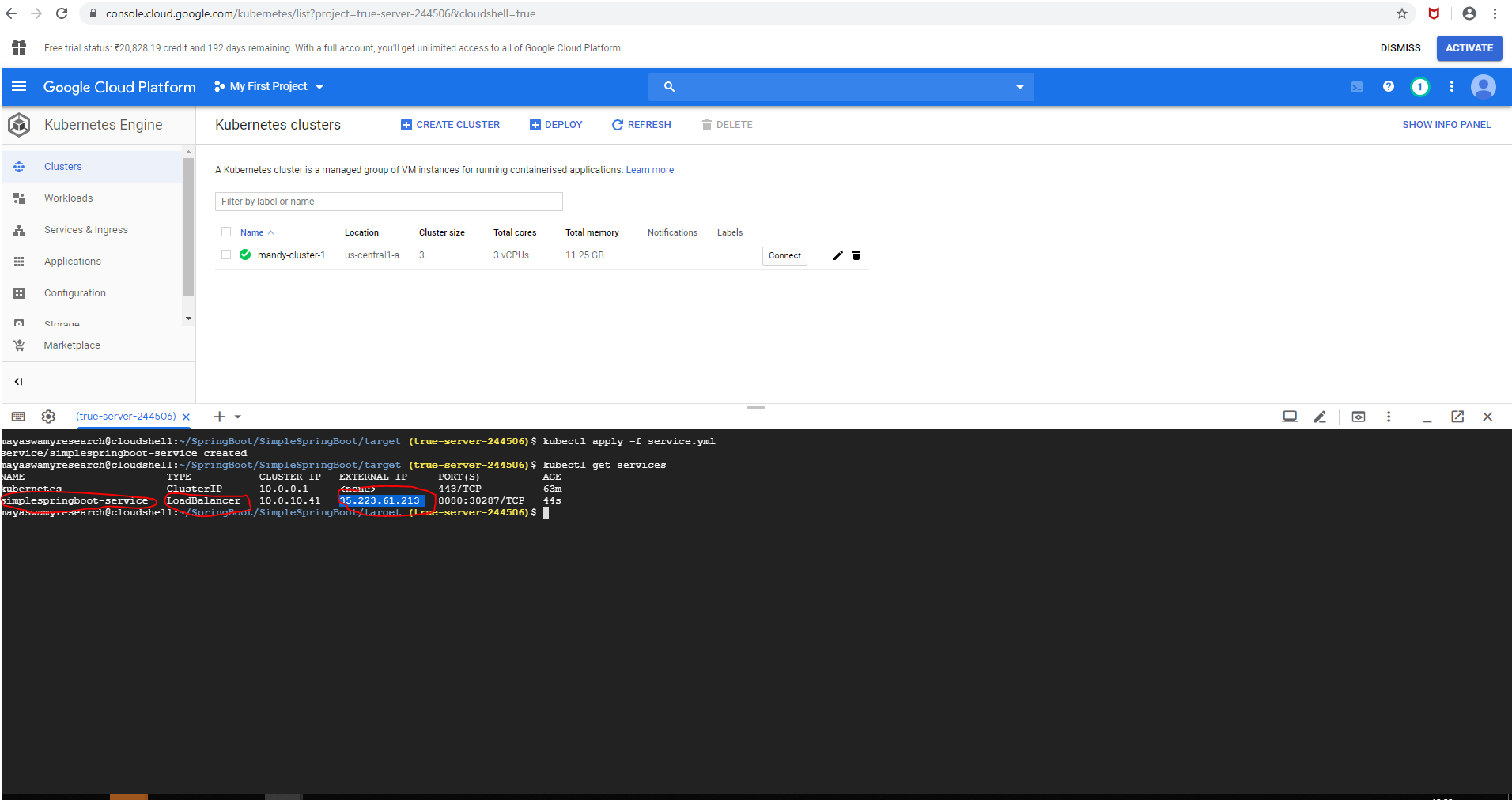
**kubectl apply -f service.yml**



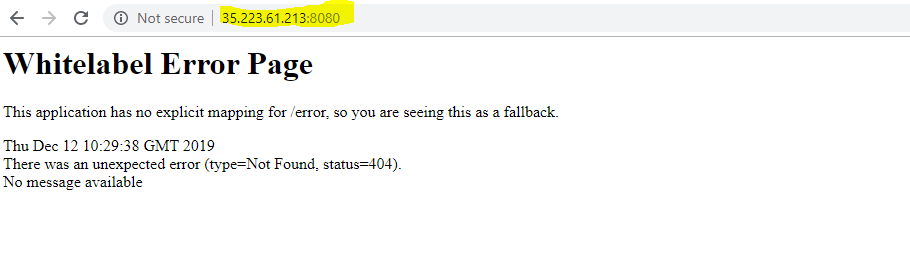


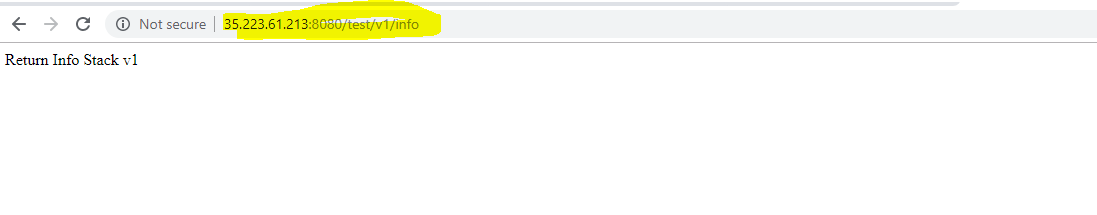
kubectl get services



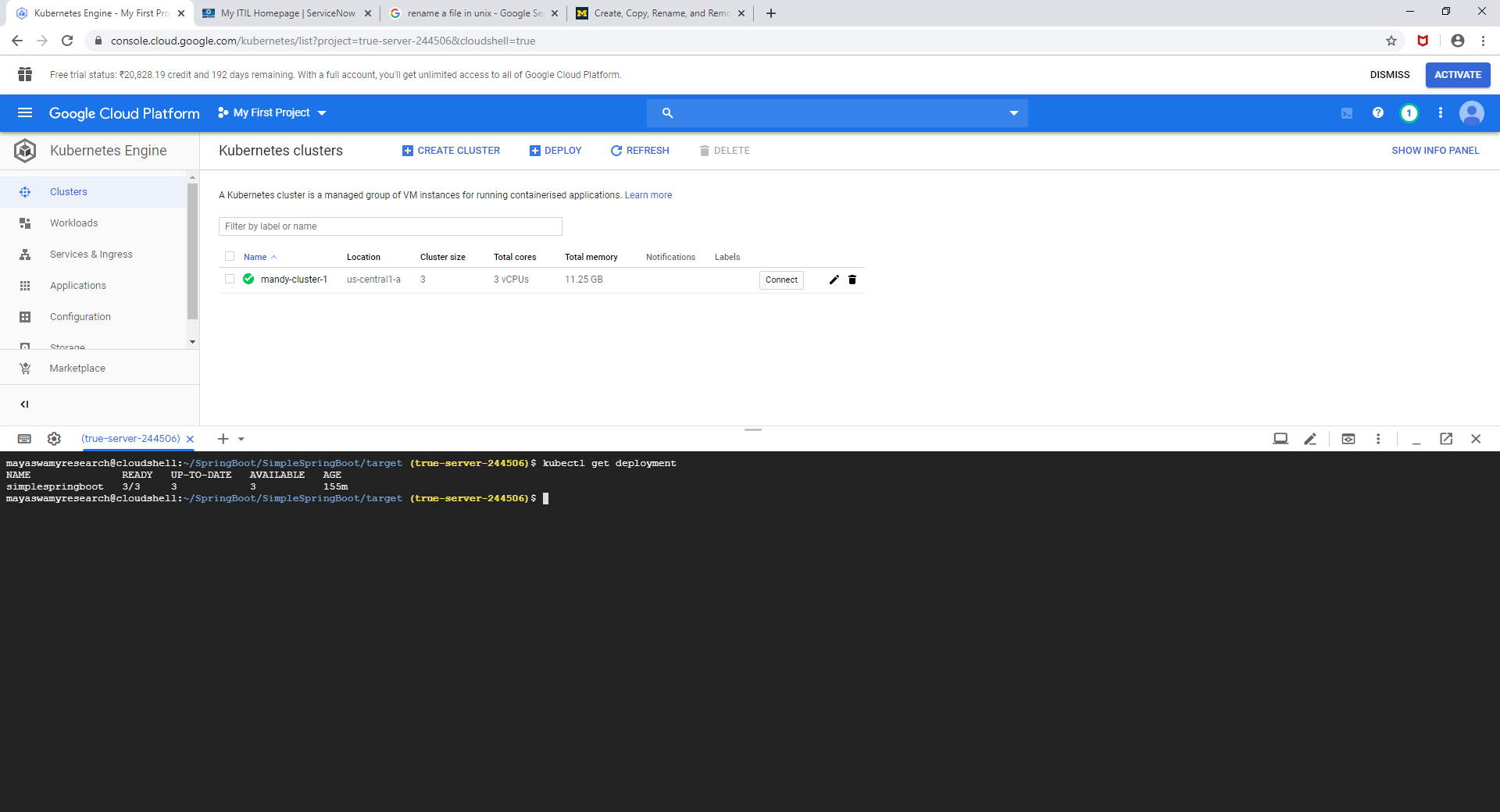


The external IP has been assigned to the service which can be used to hit the endpoint on the deployments.





kubectl get deployment

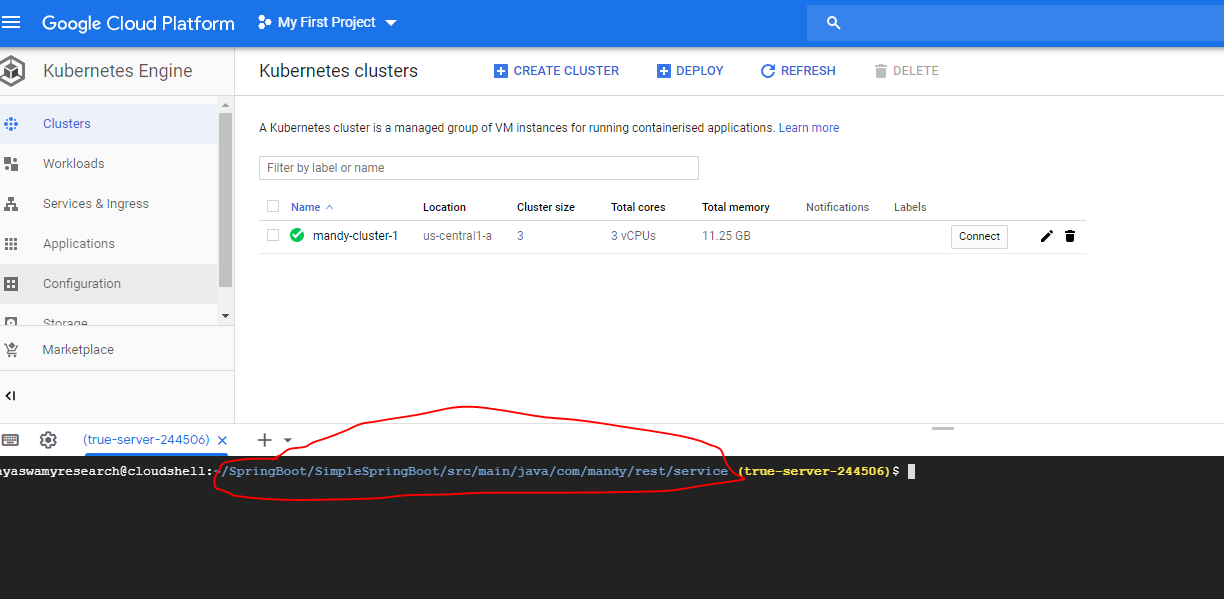


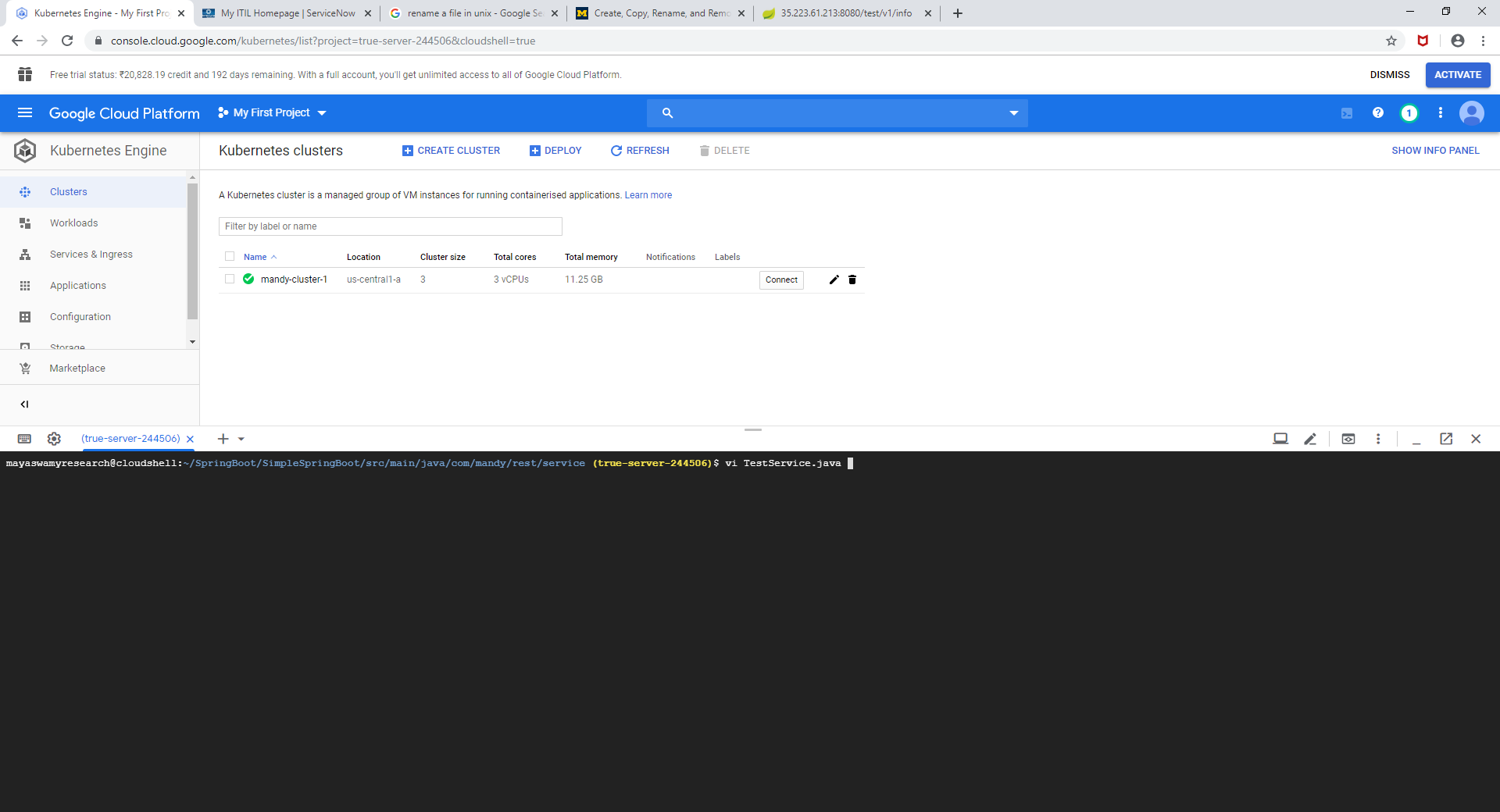
Right now there are 3 deployments and all are running , cluster will create a new deployment automatically if any one goes down

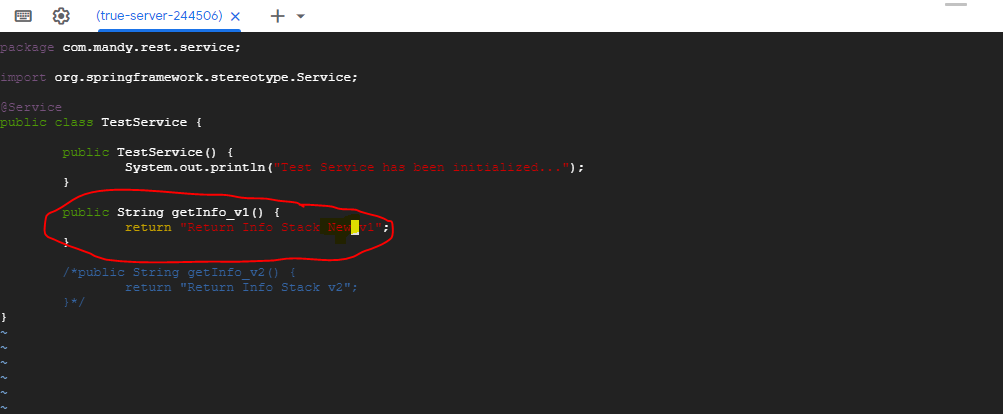
If we have new changes in the existing deployment or we have new image with some changes in the deployment we can apply the deployments ,

For example , we will change the response of the ,<http://35.223.61.213:8080/test/v1/info>

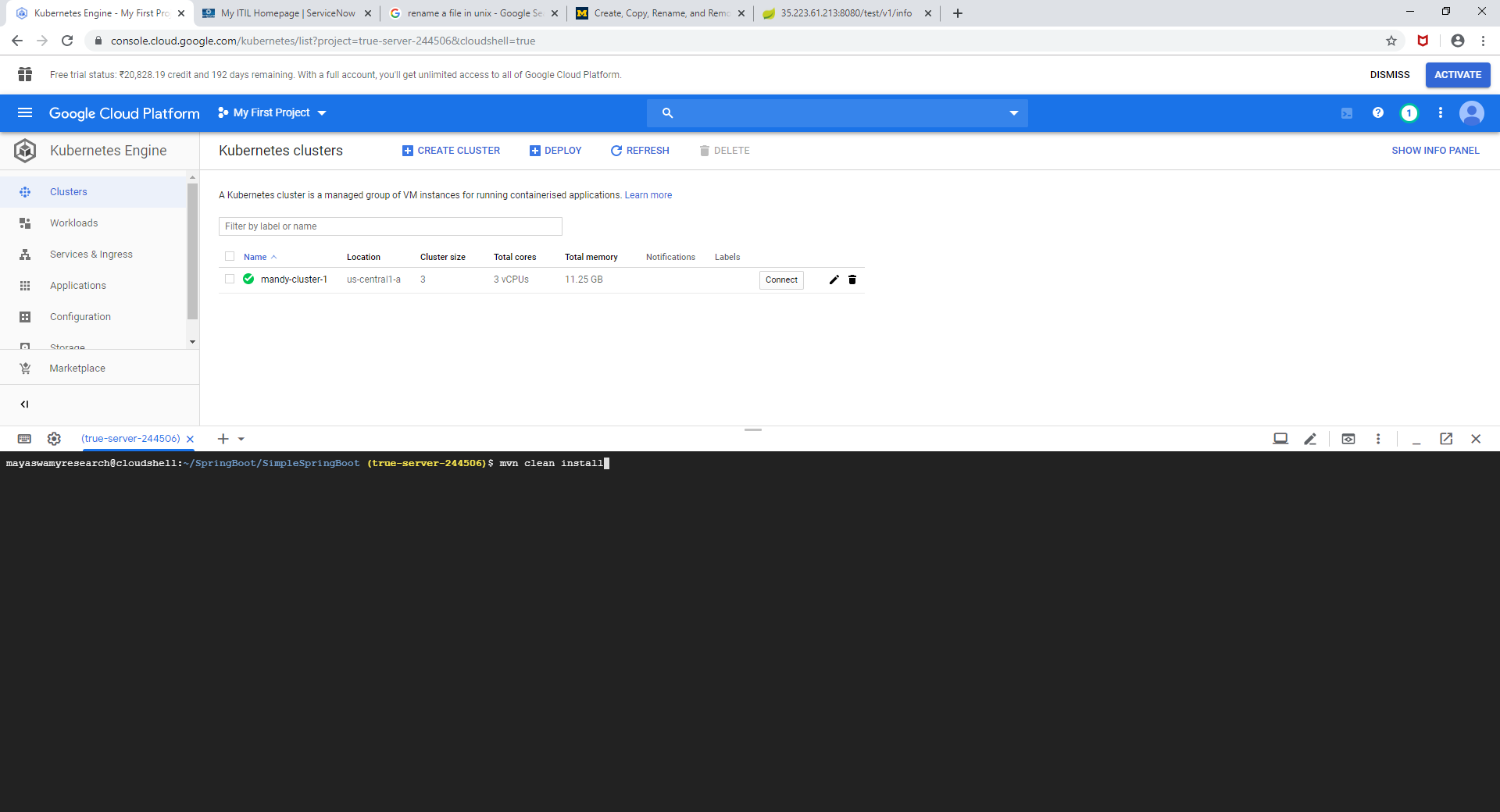
from **Return Info Stack v1** to **Return Info Stack new v1.**







Rebuild the jar file



Now in order to redeploy the new deployments, we will create a new yml file called **kube.yml in which we will have configuration for deployment and service object.**

apiVersion: extensions/v1beta1

kind: Deployment

metadata:

name: simplespringboot

spec:

replicas: 3

template:

metadata:

labels:

app: simplespringboot

spec:

containers:

- name: simplespringboot

image: 'gcr.io/true-server-244506/simplespringboot:v1'

ports:

- containerPort: 8080

---

apiVersion: v1

kind: Service

metadata:

name: simplespringboot-service

labels:

name: simplespringboot-service

spec:

ports:

- port: 8080

targetPort: 8080

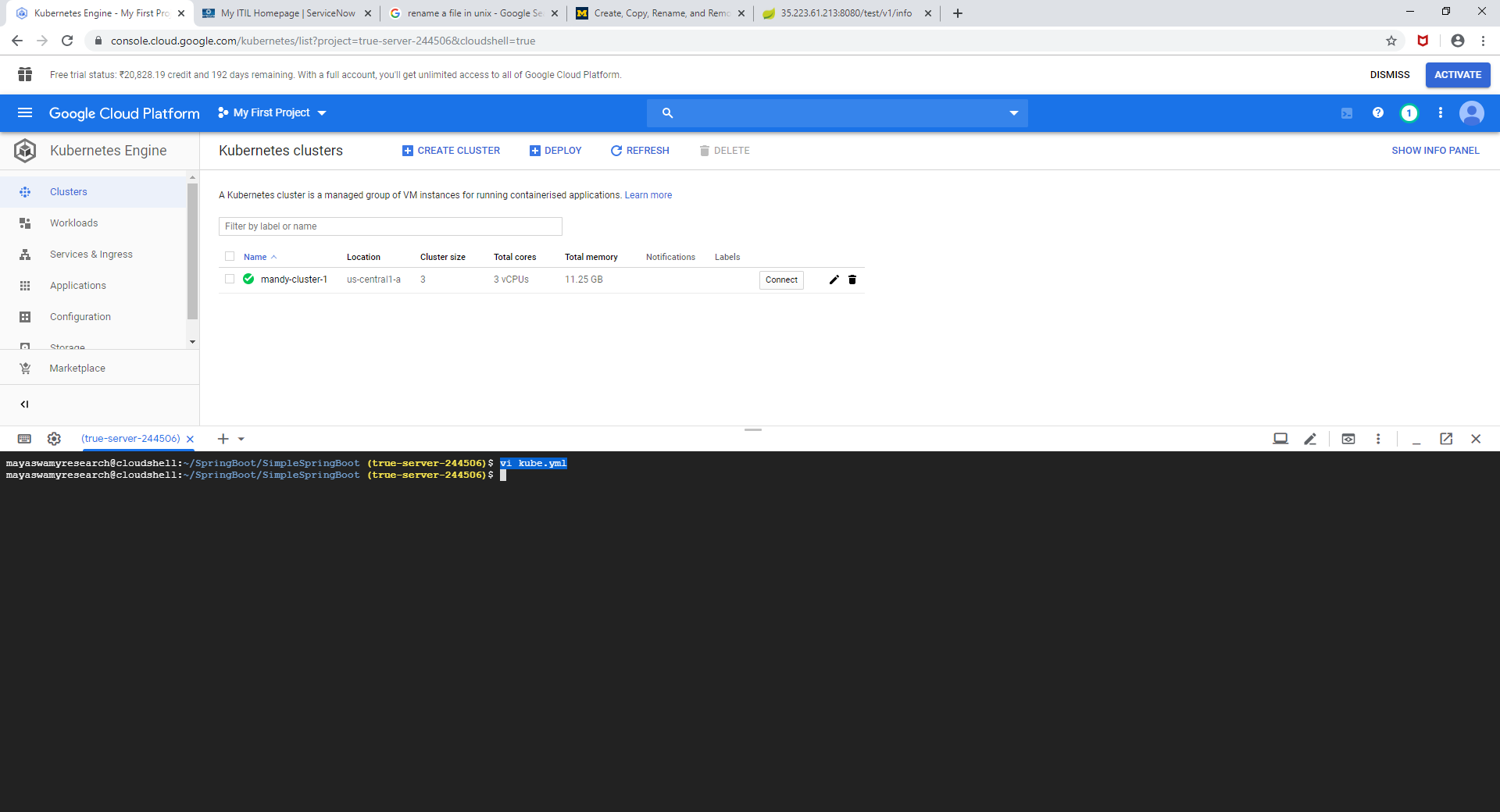
protocol: TCP

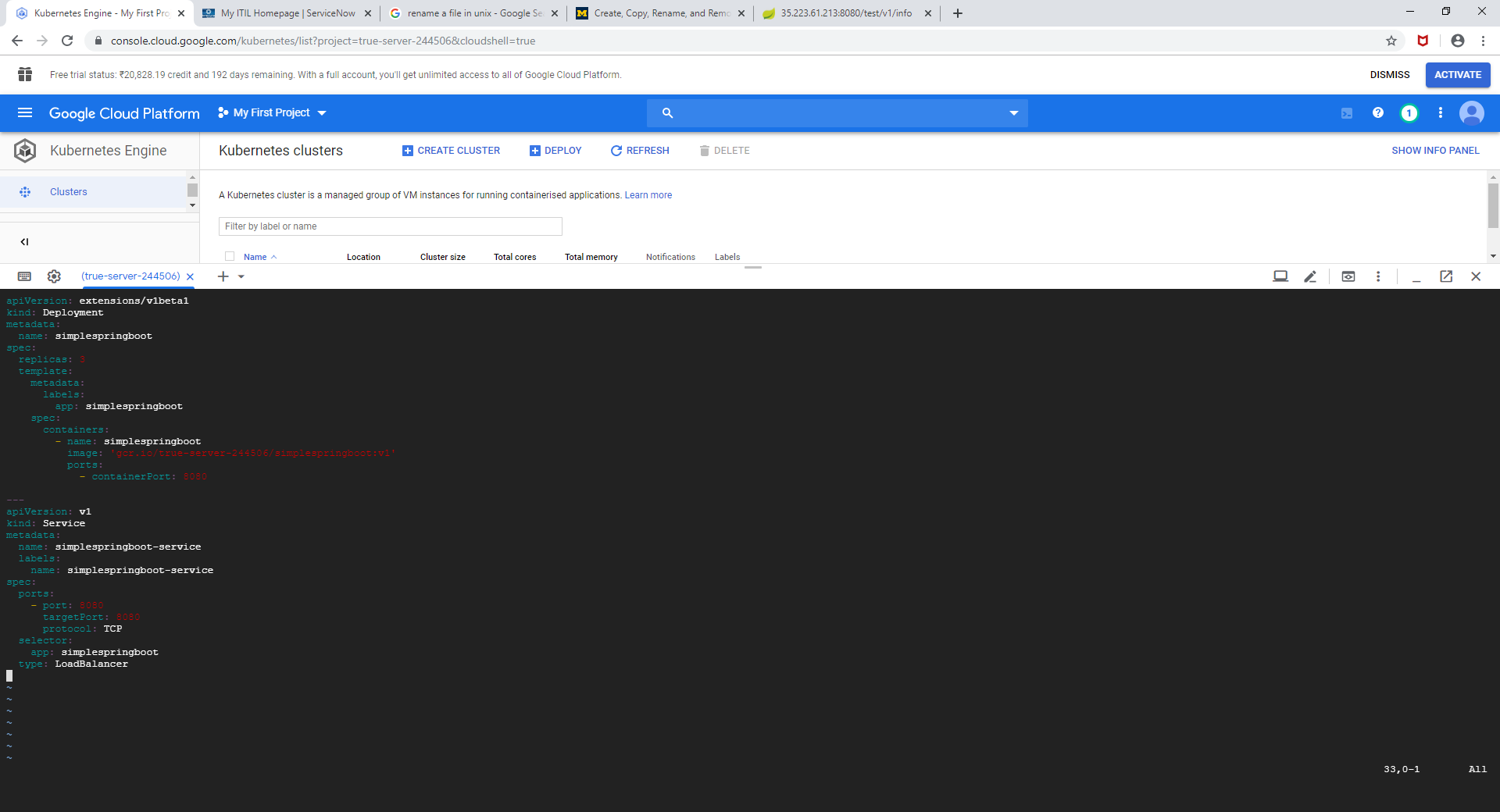
selector:

app: simplespringboot

type: LoadBalancer

vi kube.yml

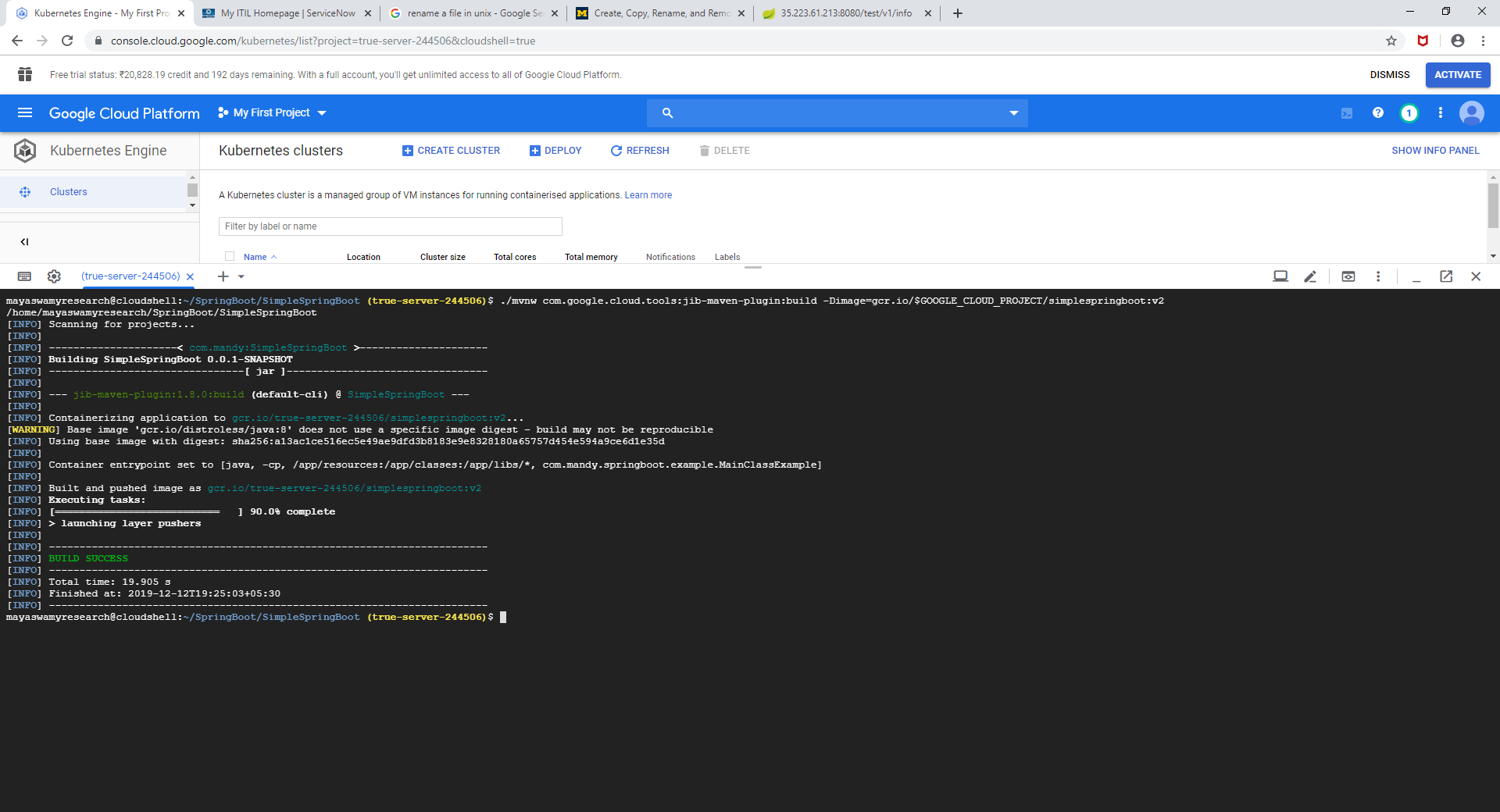




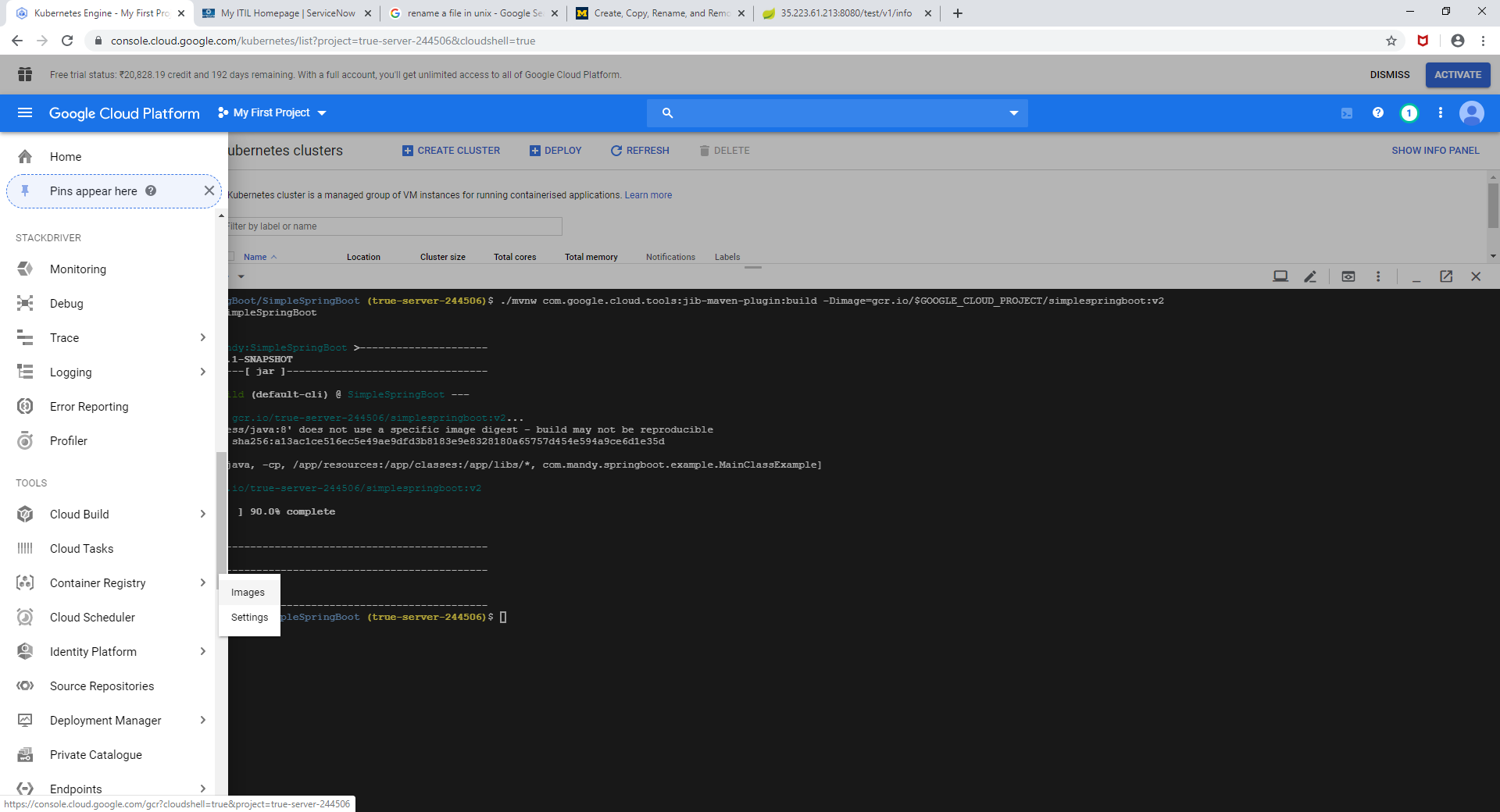
**Now Create a new image with v2 version**

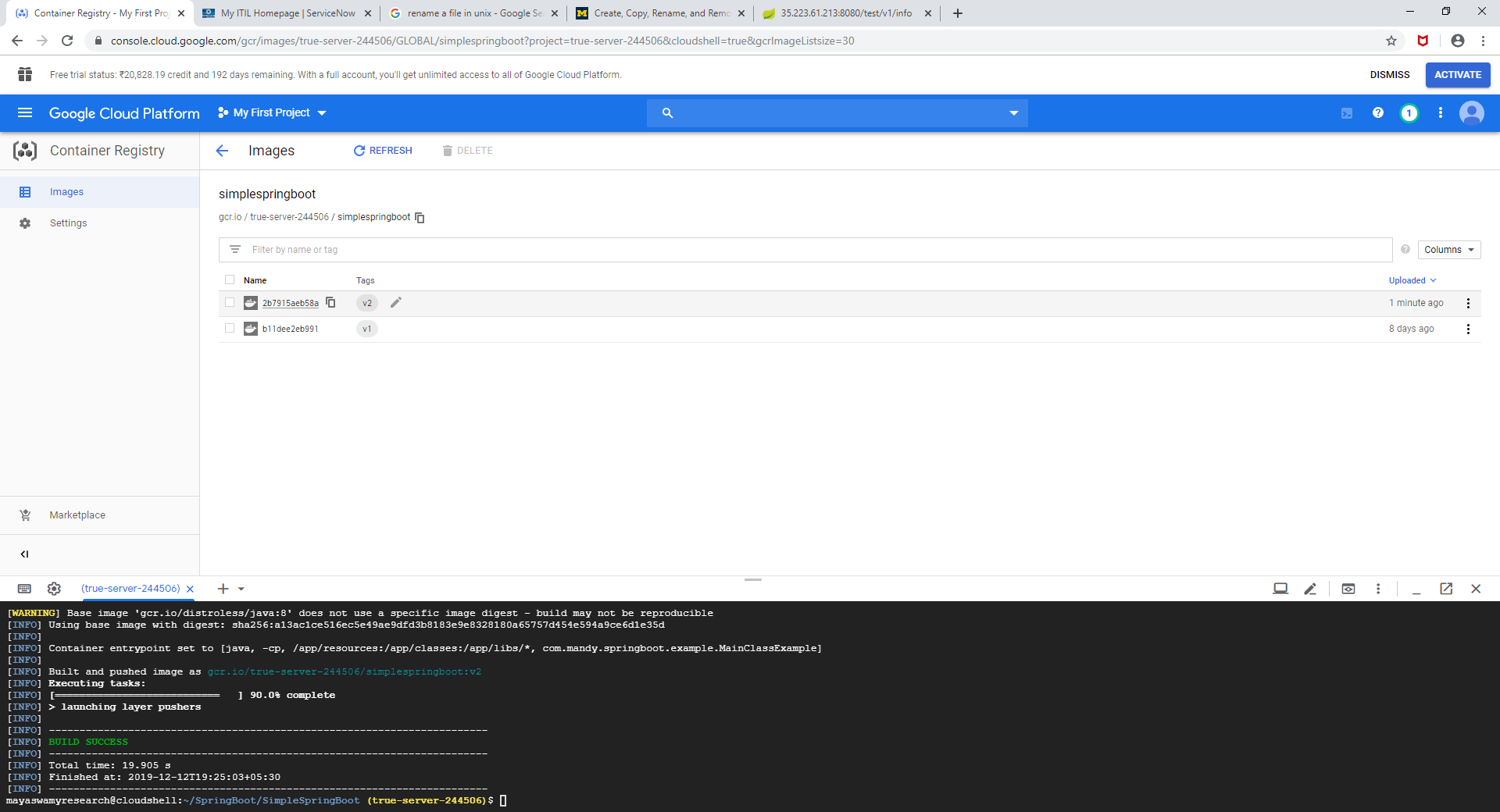
**./mvnw com.google.cloud.tools:jib-maven-plugin:build -Dimage=gcr.io/$GOOGLE\_CLOUD\_PROJECT/simplespringboot:v2**



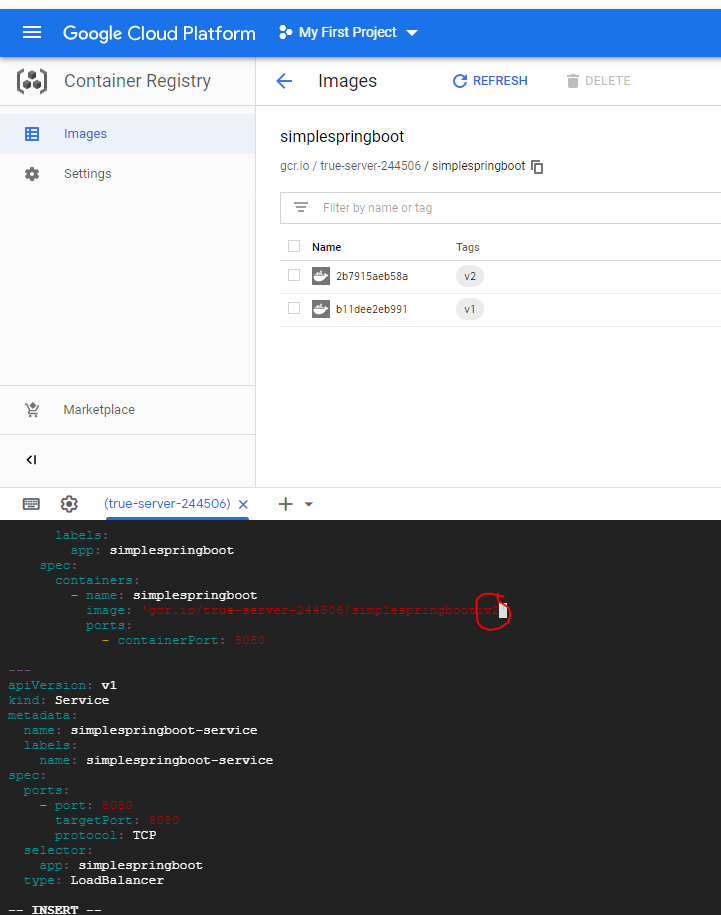


**Check if new image has been created**

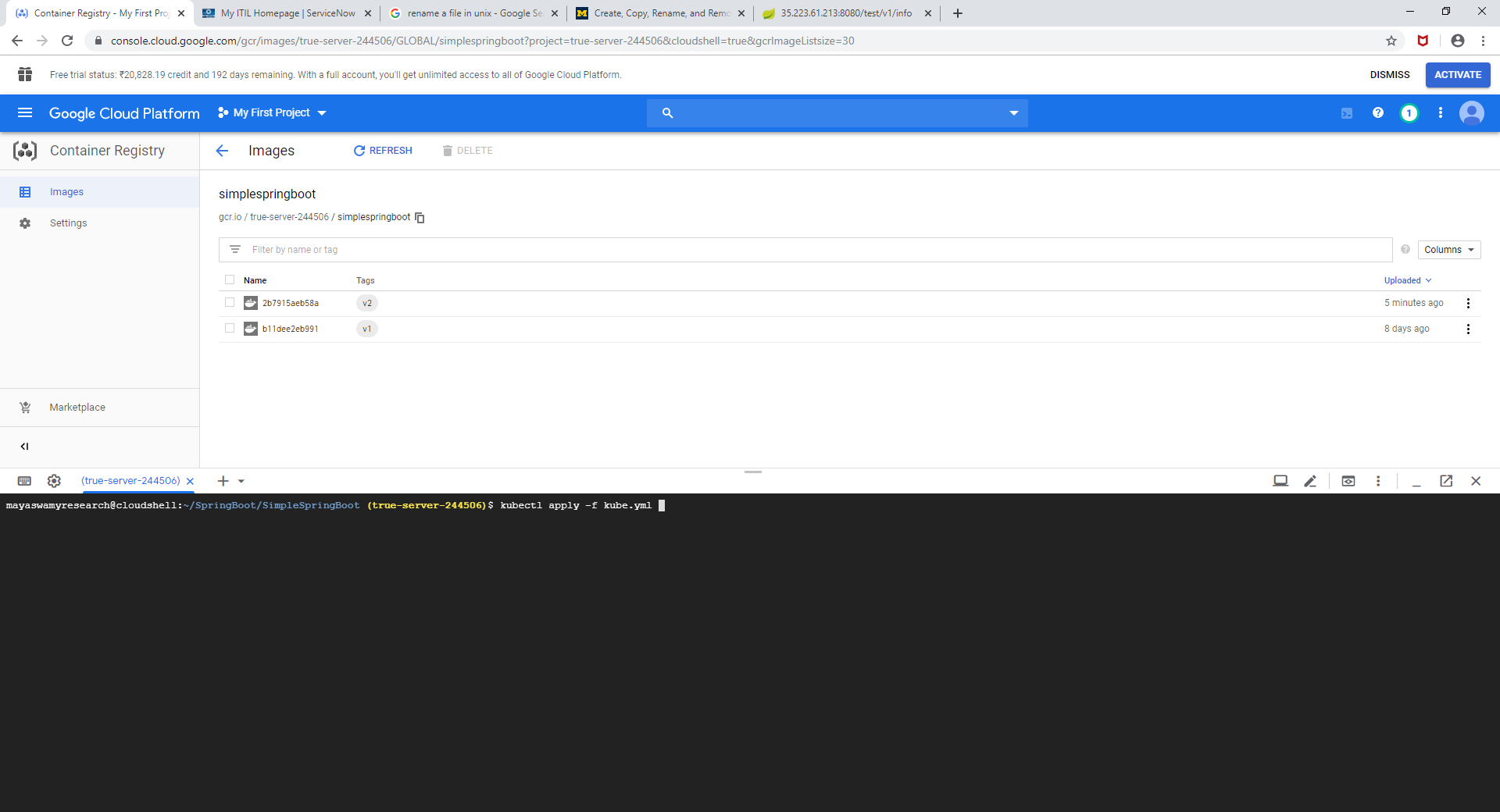


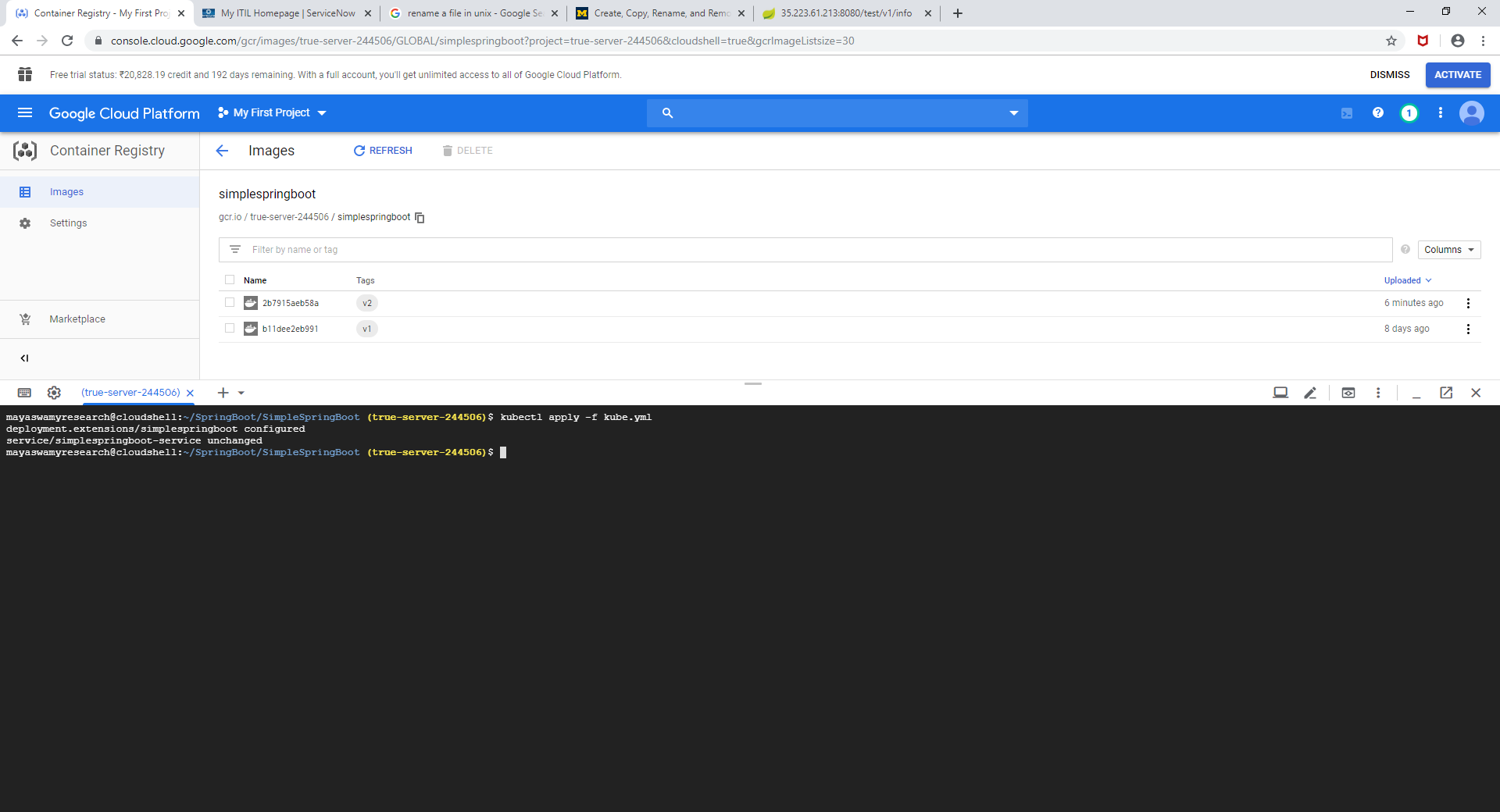


**Change the v1 verision of the image in kube.yml to v2**



**Apply new changes**





**Check the status**

**kubectl get pods**

**kubectl get services**

**kubectl get deployment**

