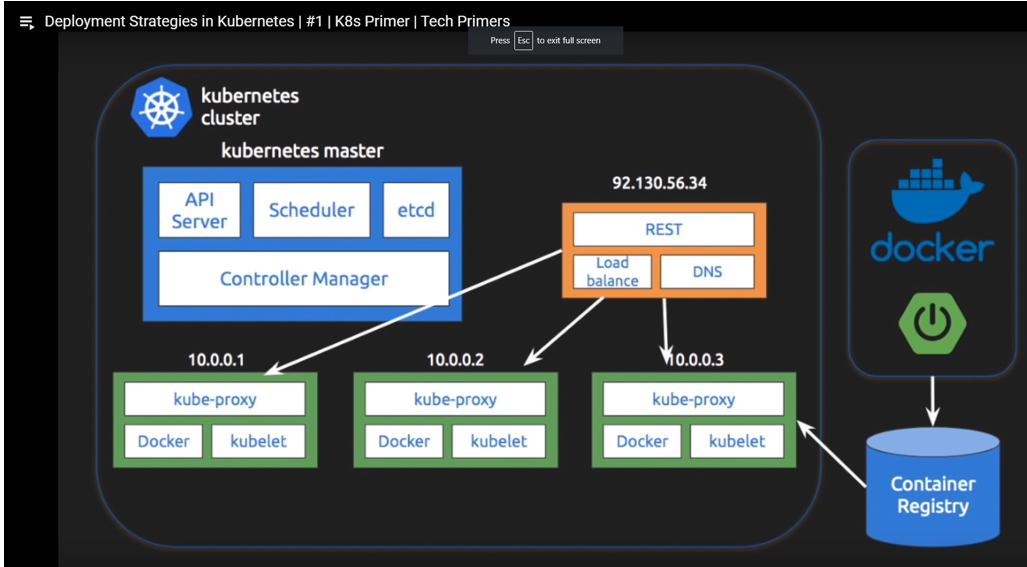
Deployment Strategies

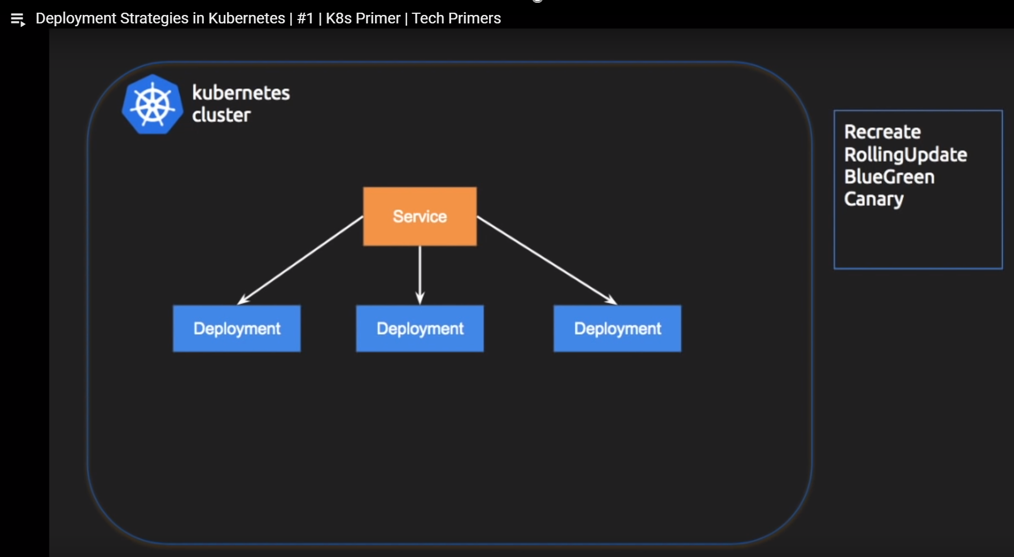
1). Recreate

2). RollingUpdate

3). BlueGreen

4). Canary

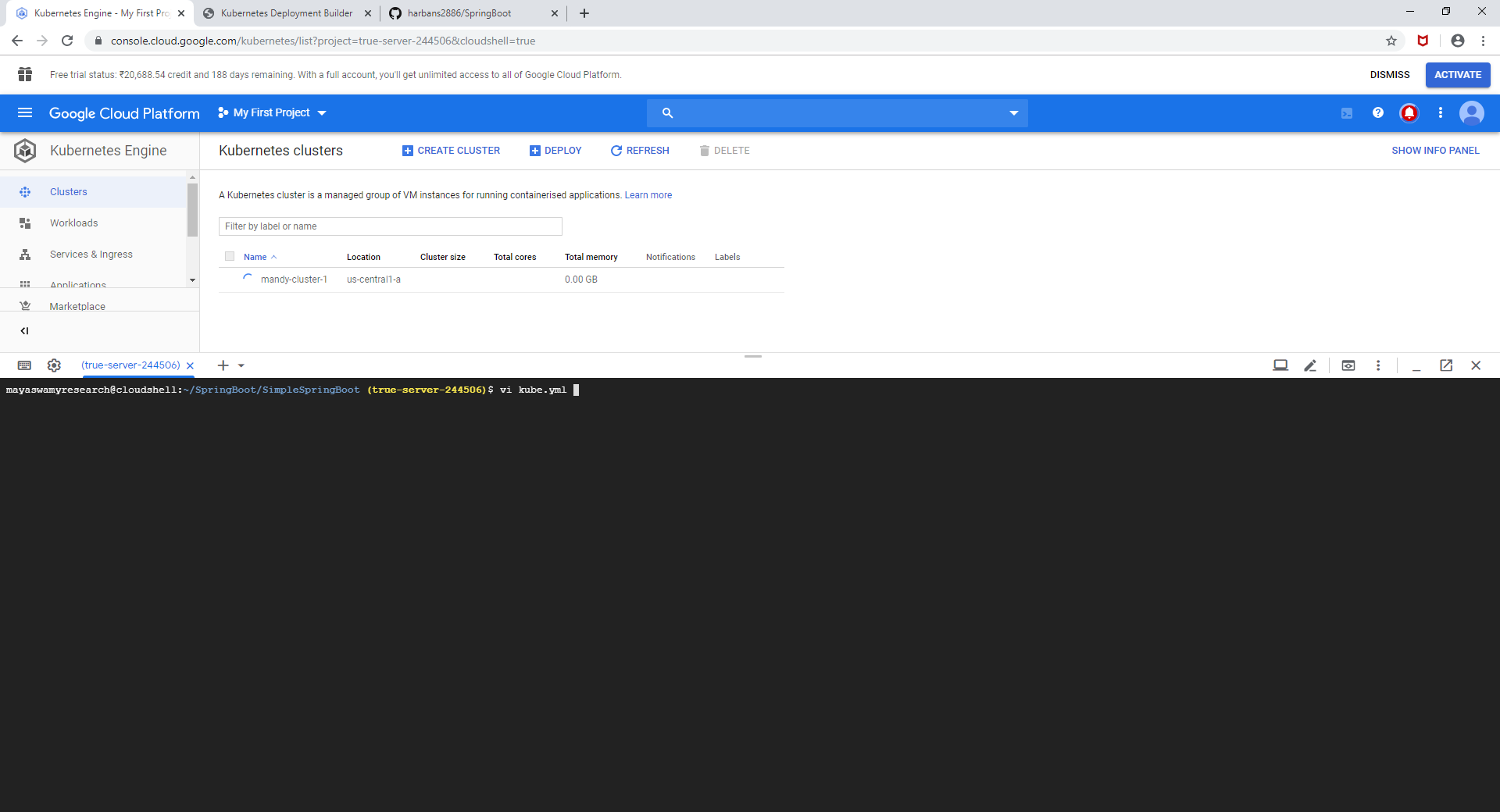


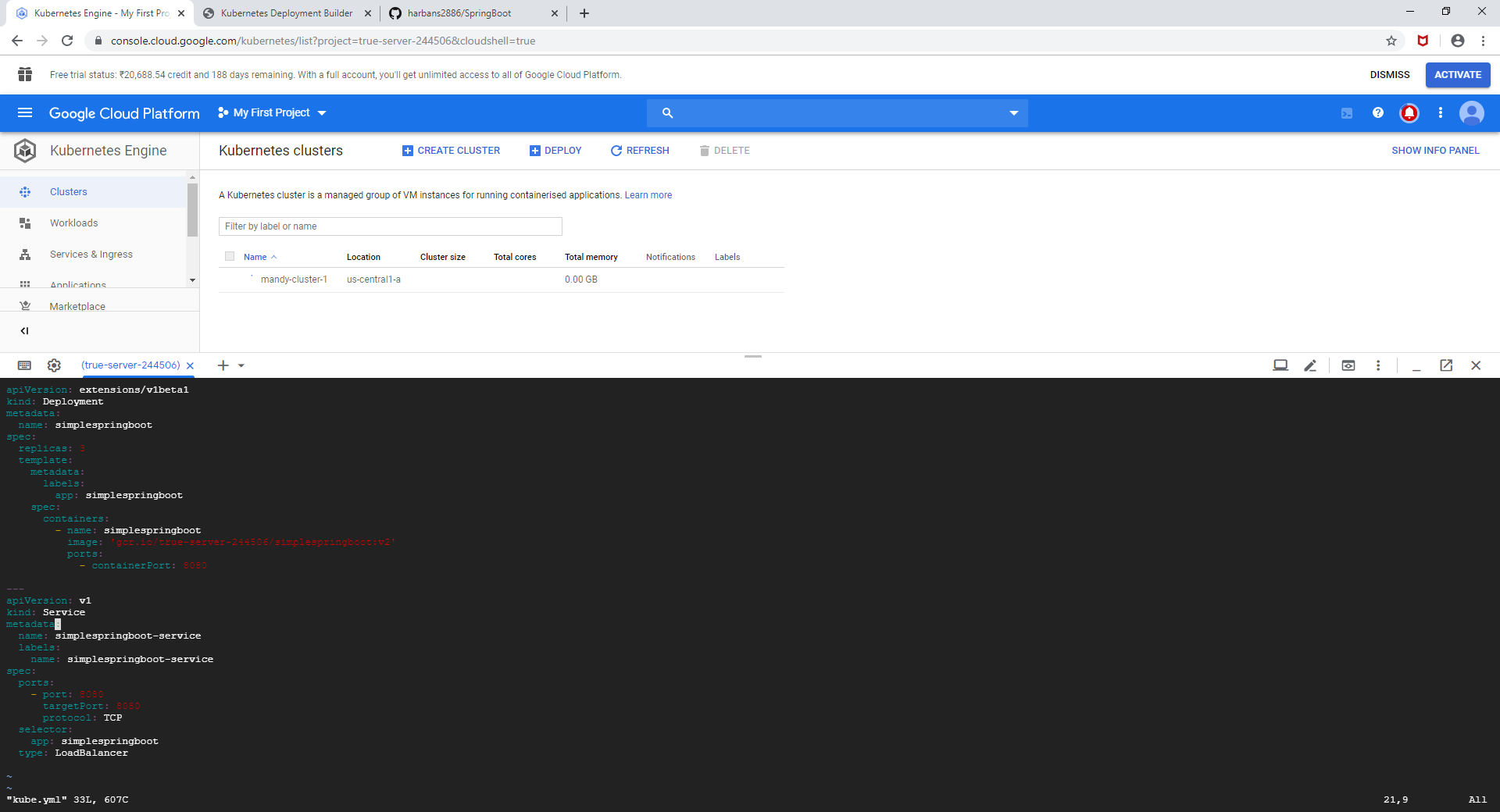


1). Recreate,

Terminate all the current deployments and re-create new ones.

We are going to use **kube.yml** from previous example

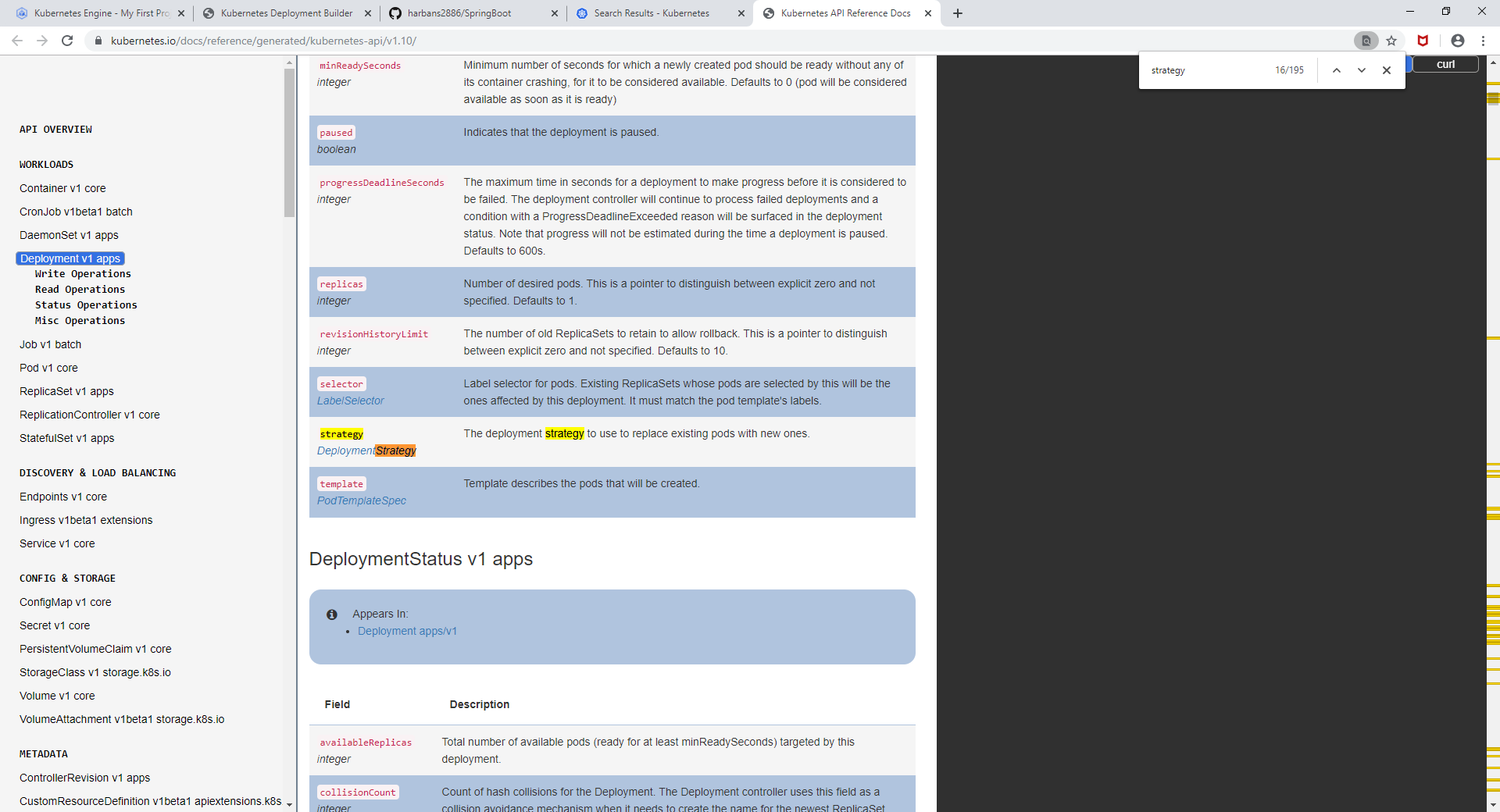


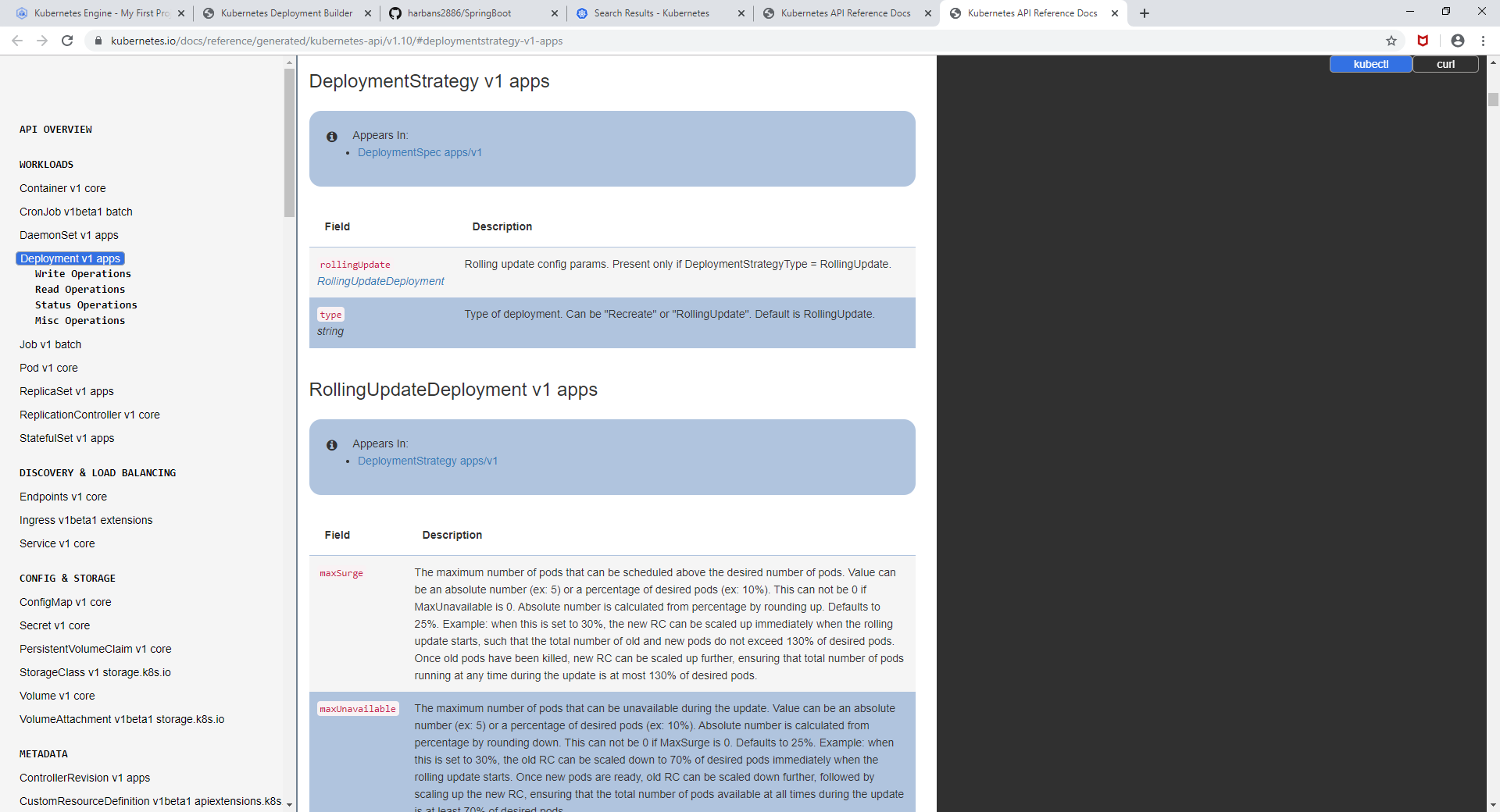


First part of the kube.yml is the configuration related to deployment object and the second part of the kube.yml is the service object.

In this example we are going to use deployment object only.

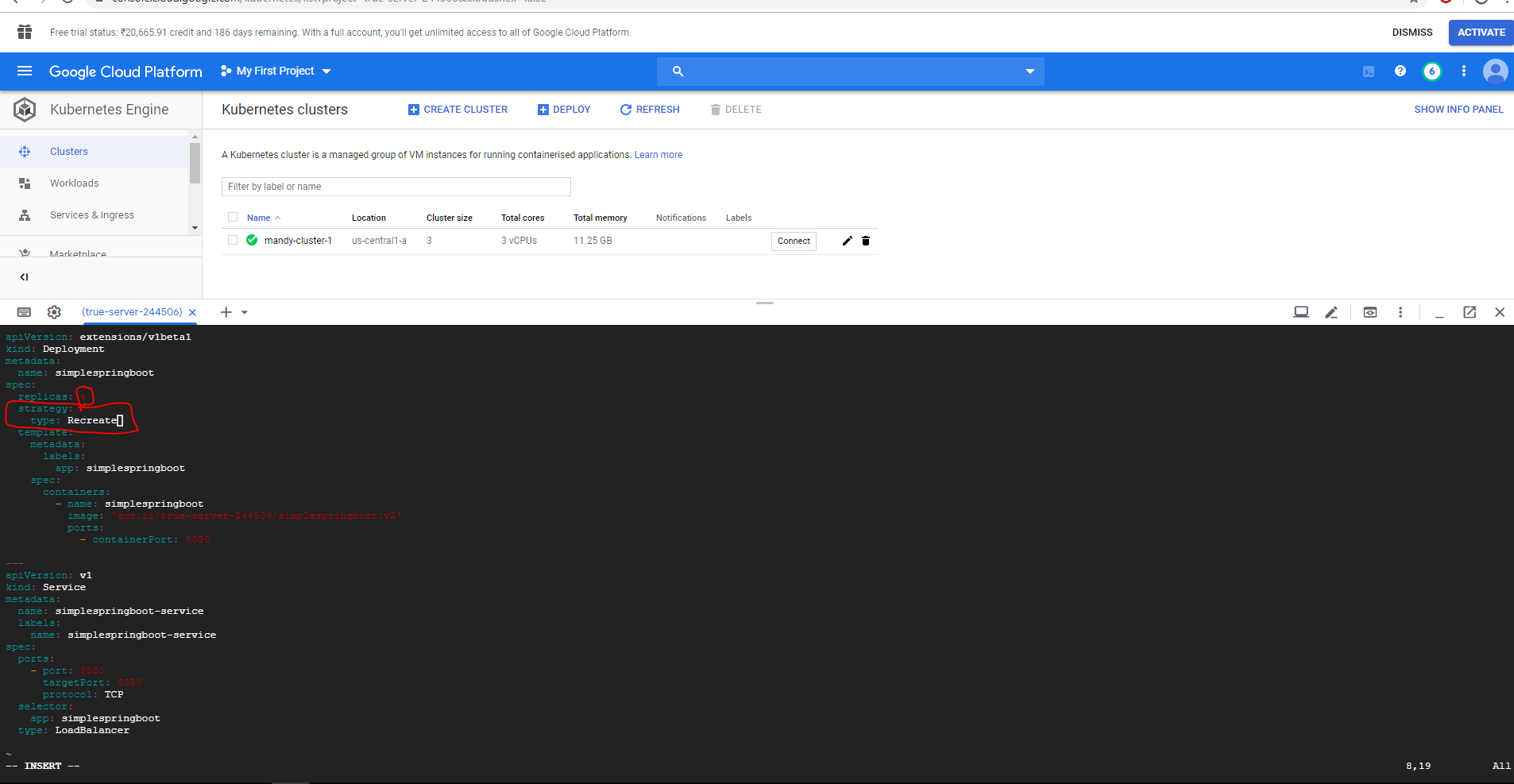
Documentation for deployment Strategy.





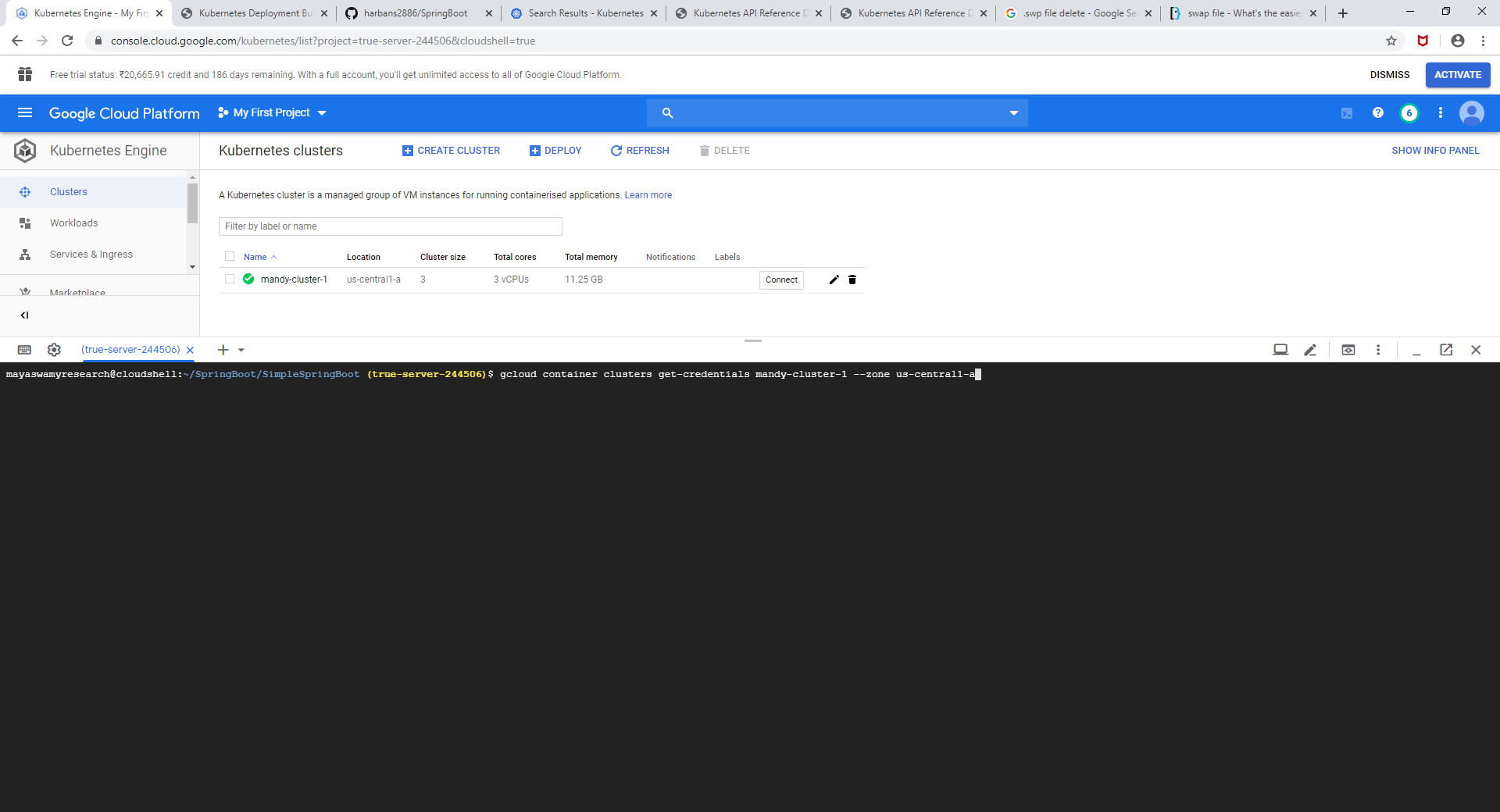
Add a Strategy in kube.yml

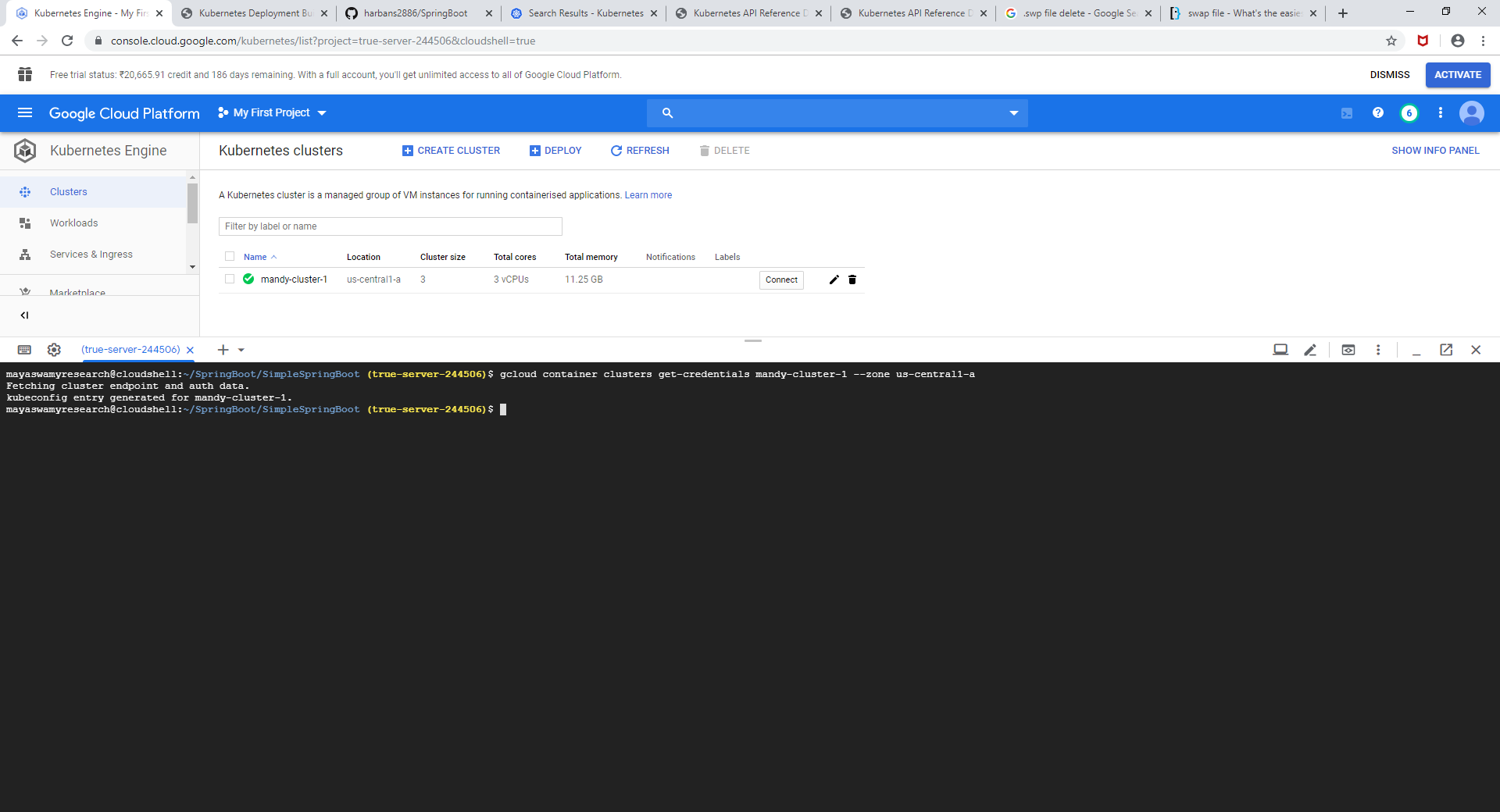
Not Relevant:- **(find . -type f -name "\*.sw[klmnop]" -delete) to delete swp file**



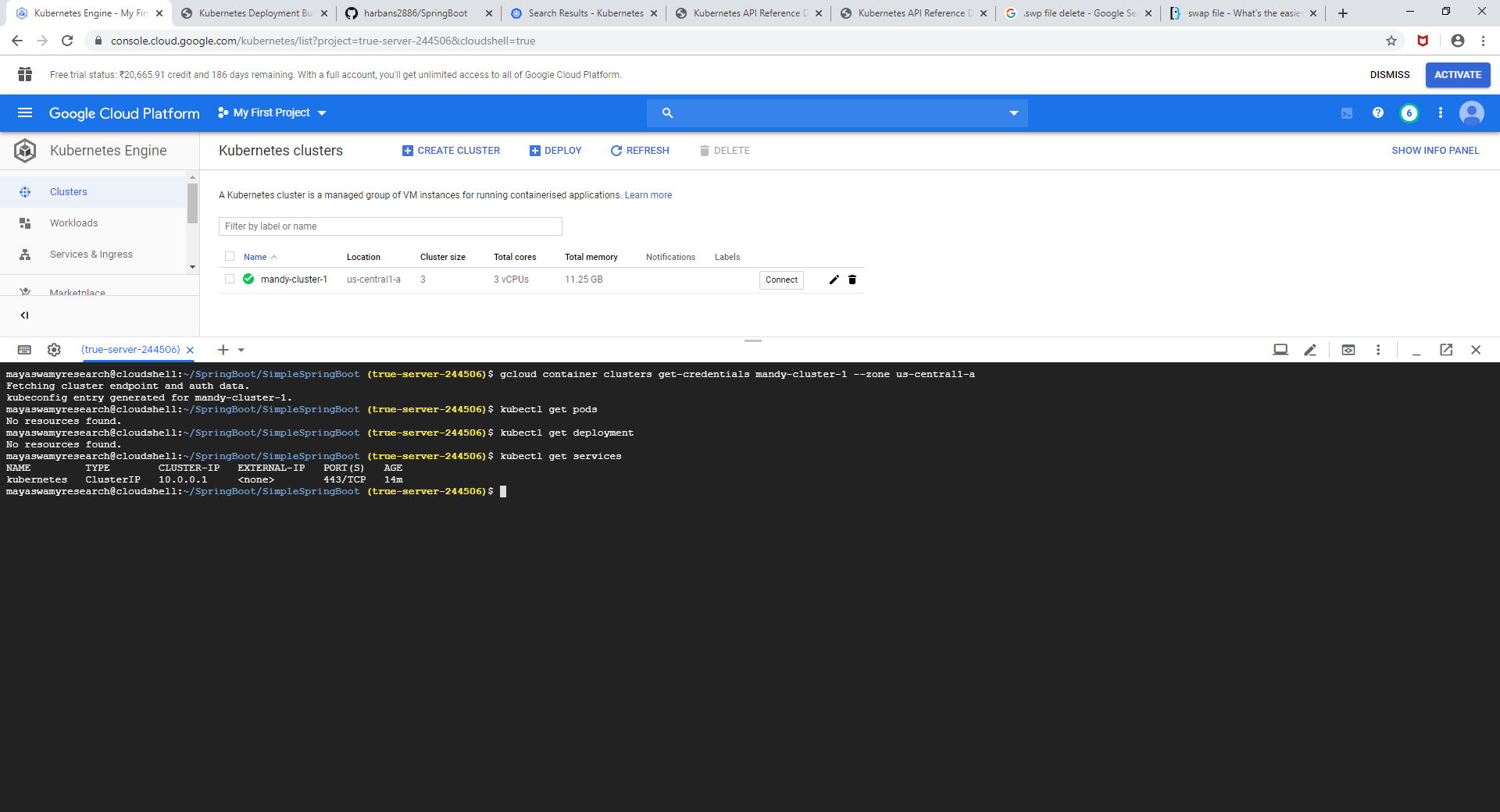
Now enter the cluster

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



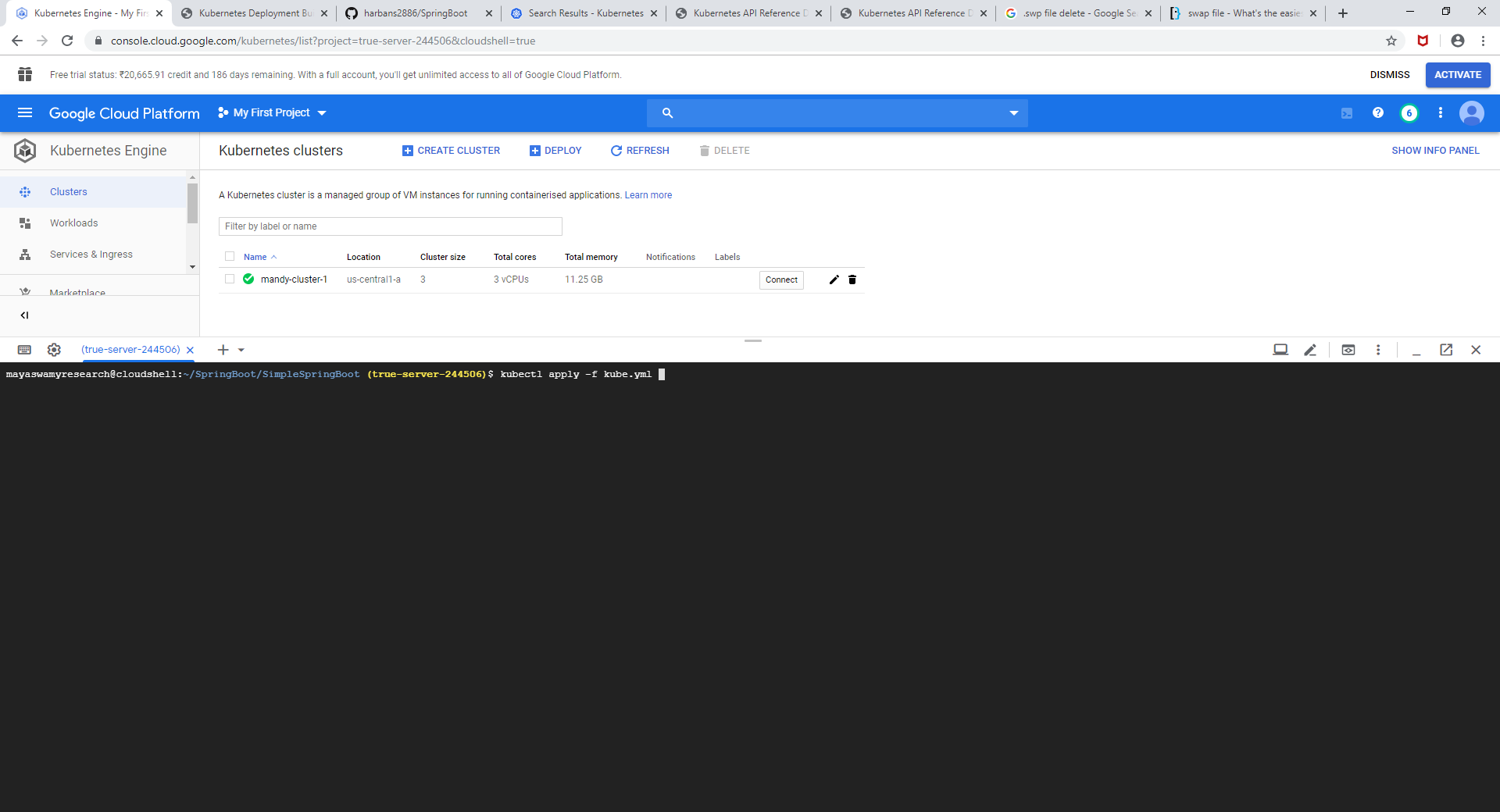


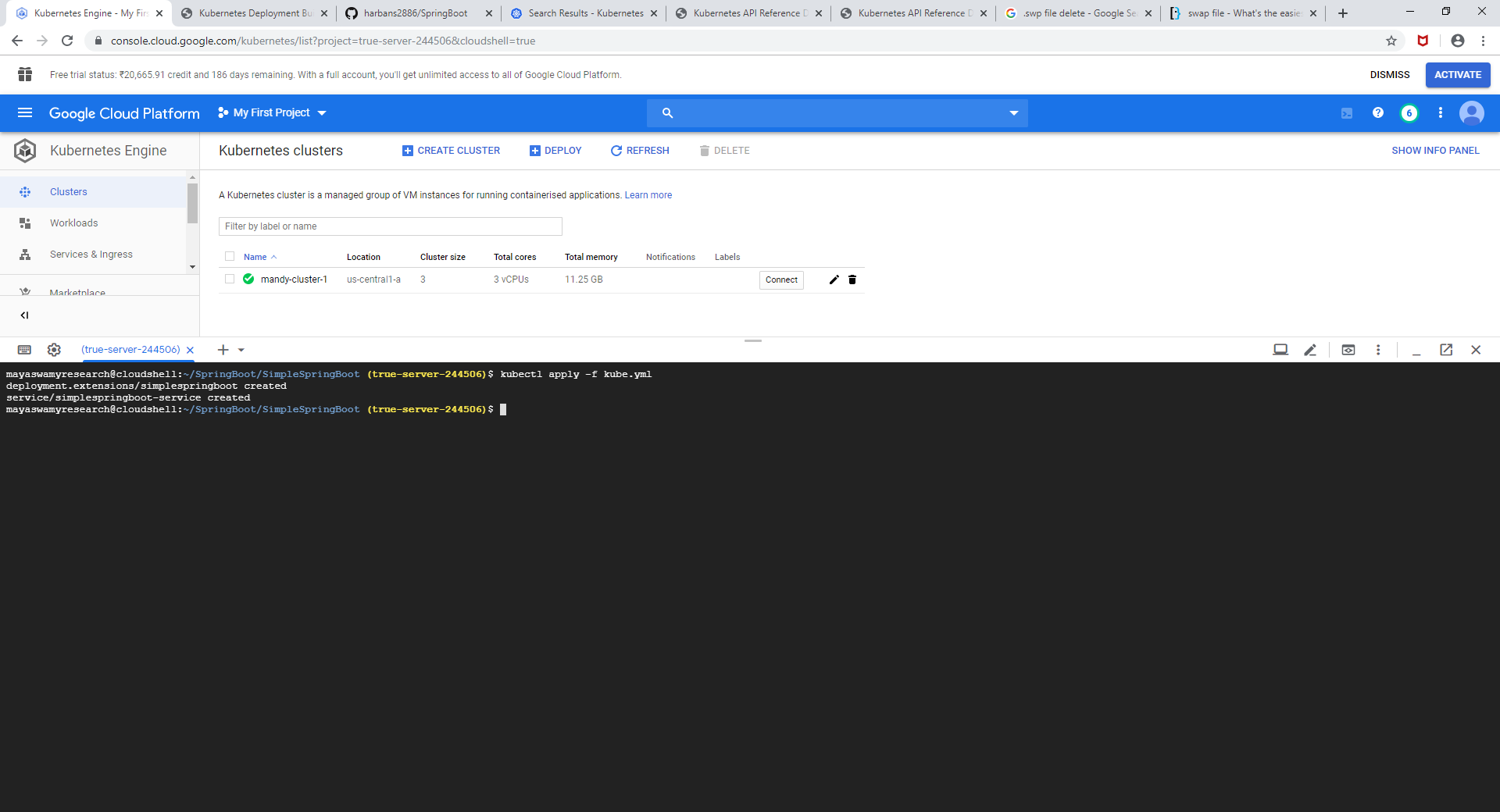
No Resources are running @ the moment

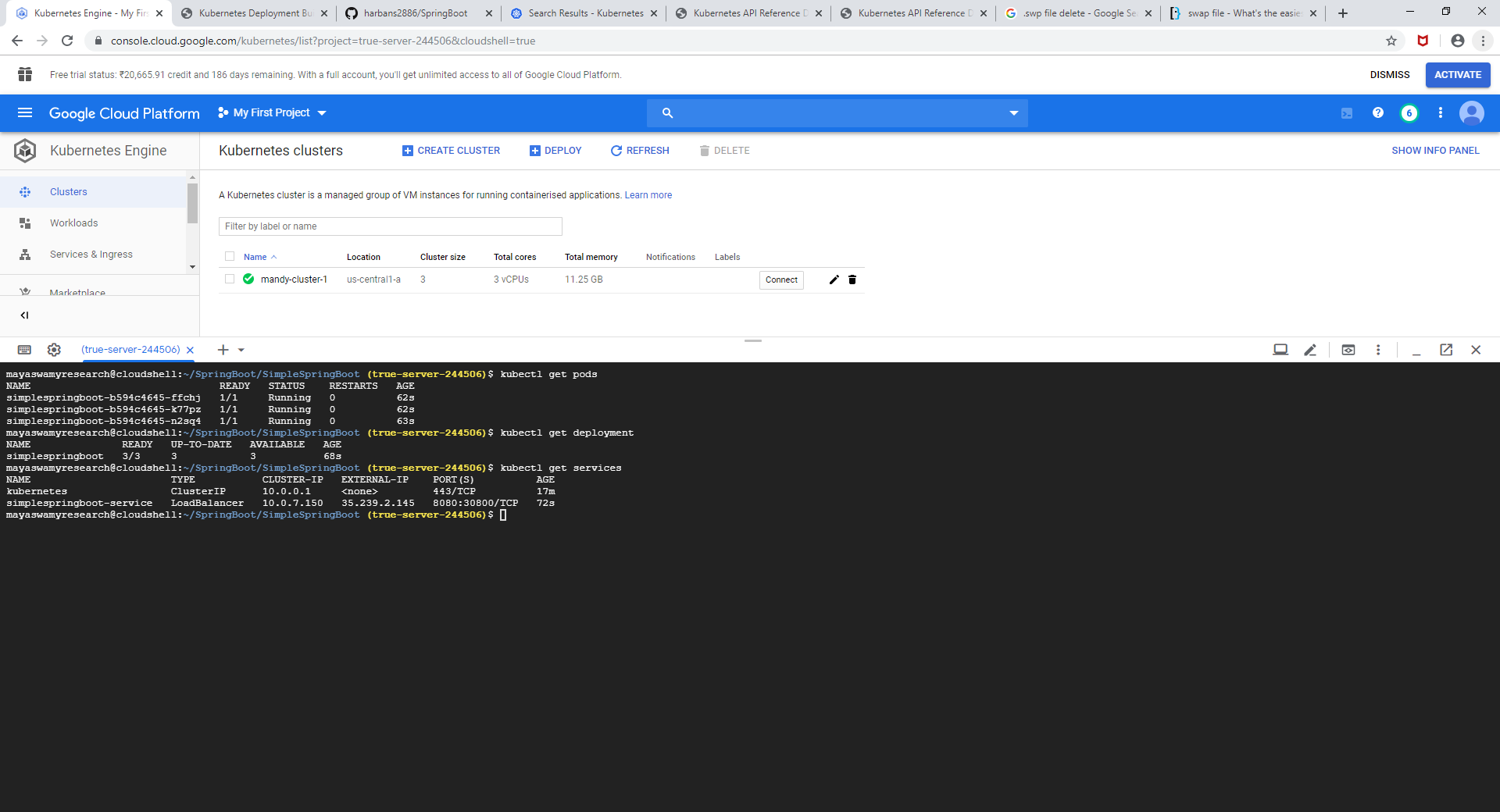


Apply the deployments

kubectl apply -f kube.yml

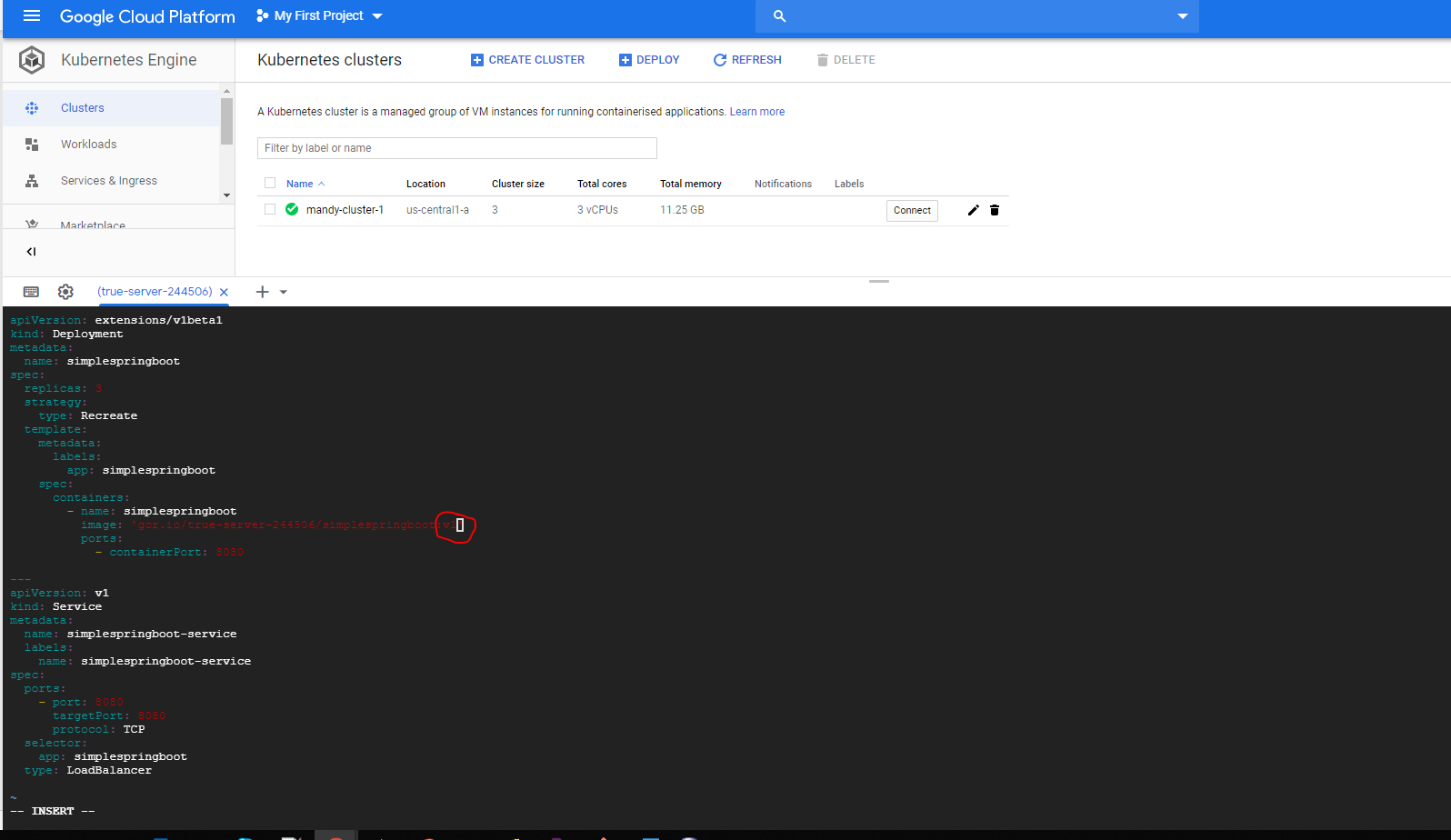






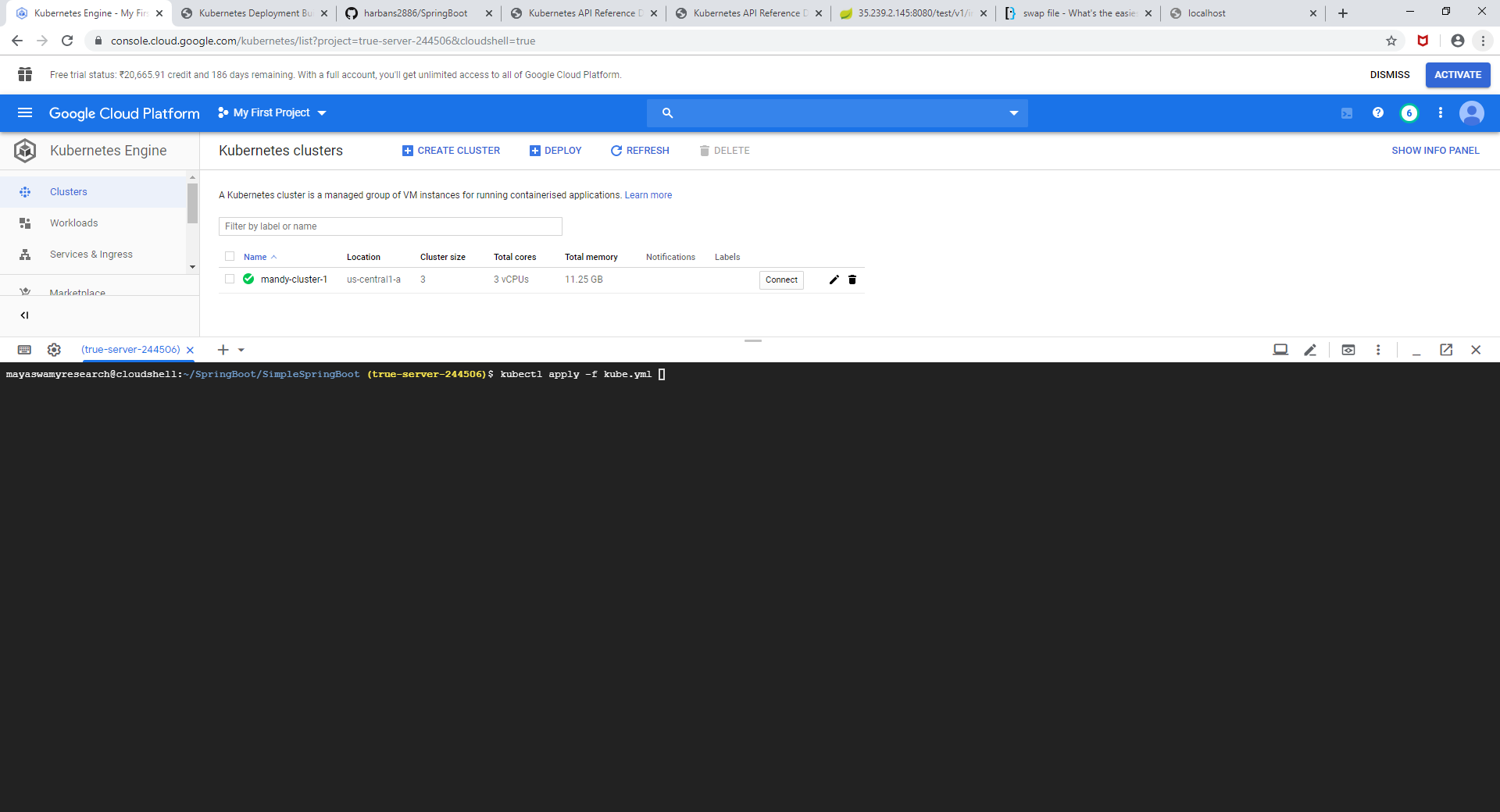


Now change the deployment from v2 to v1.

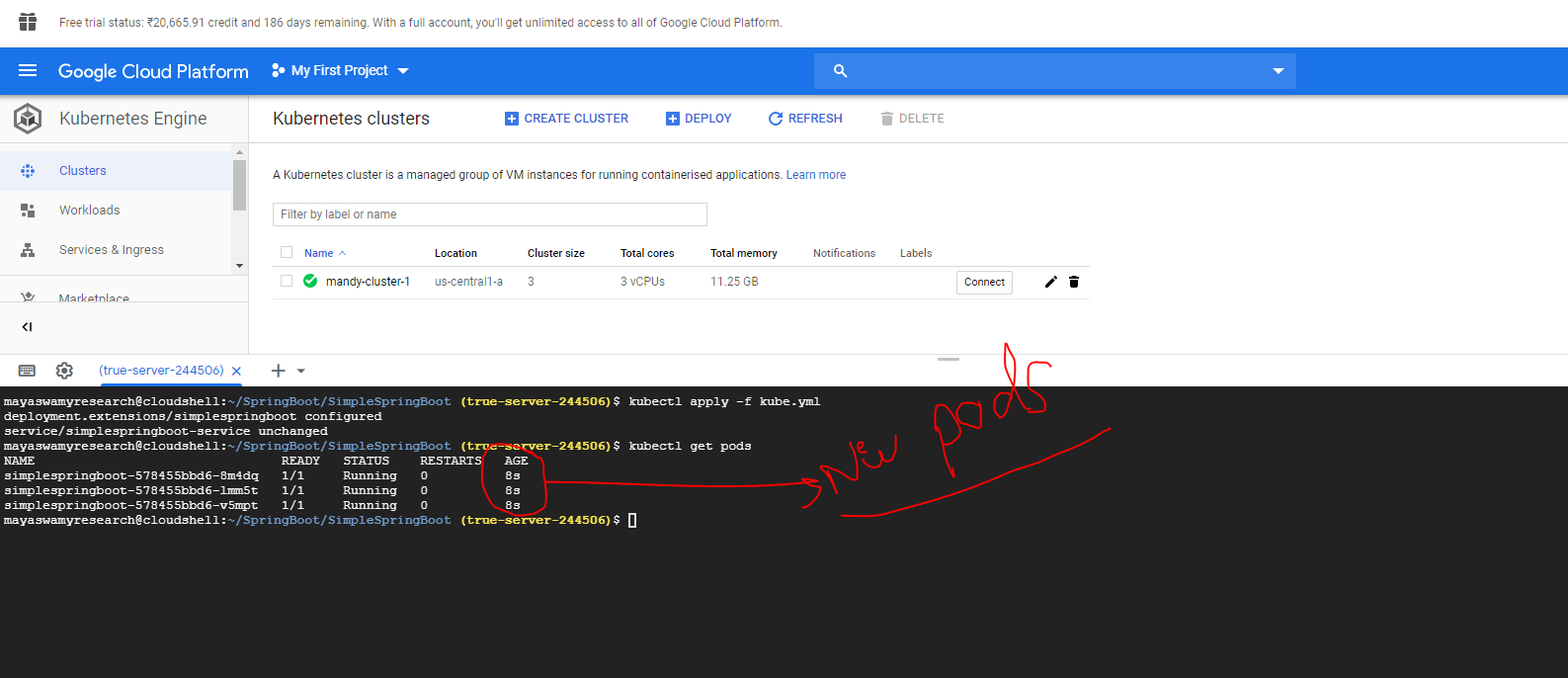


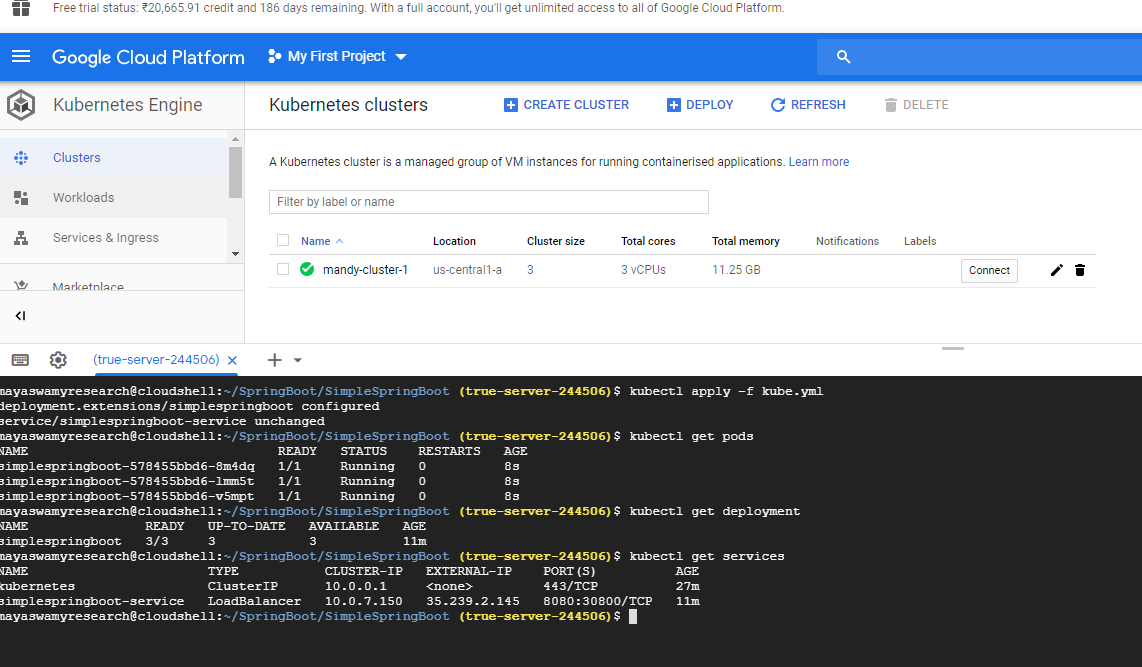
Apply the configuration

kubectl apply -f kube.yml

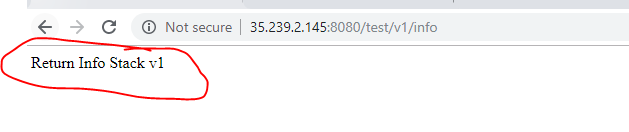


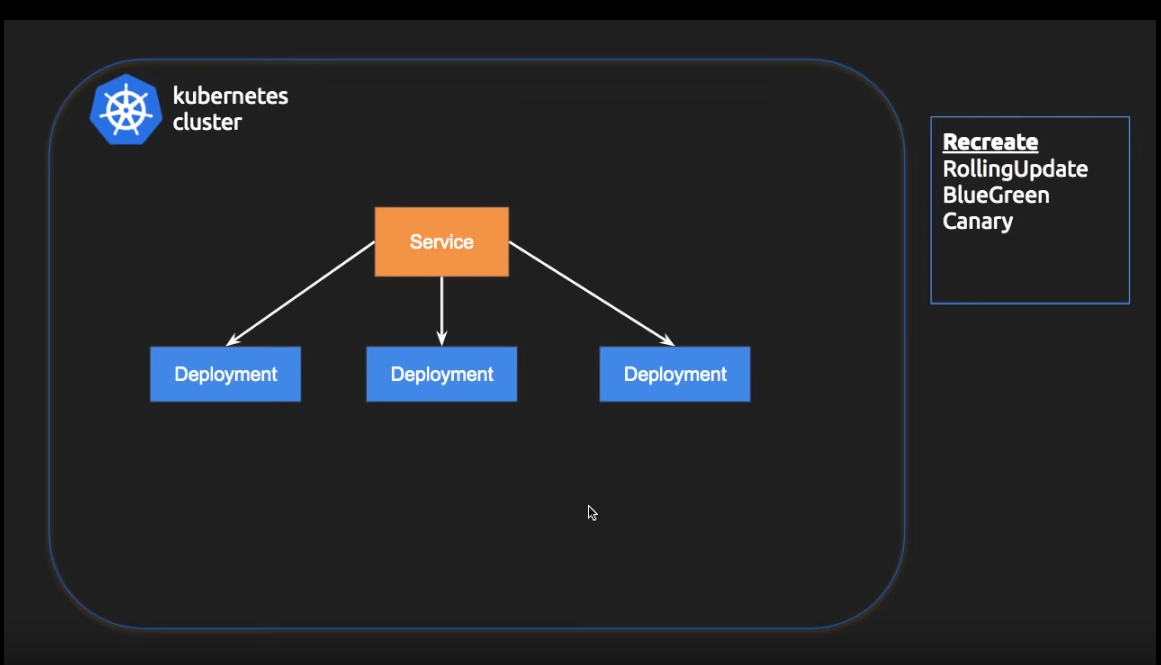


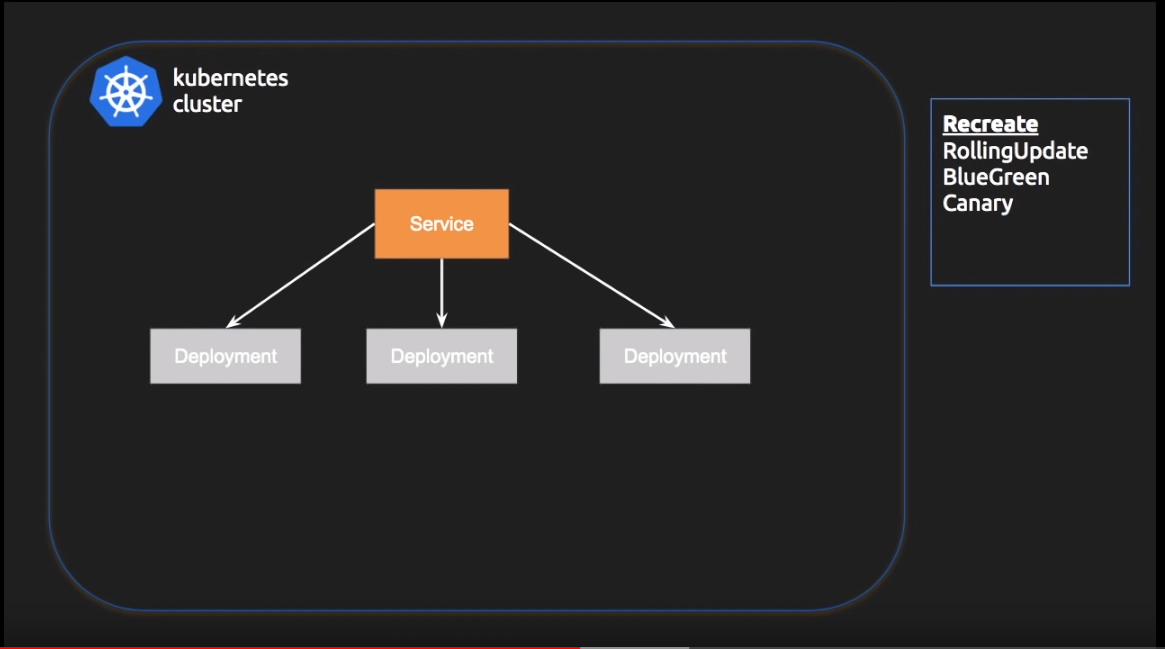










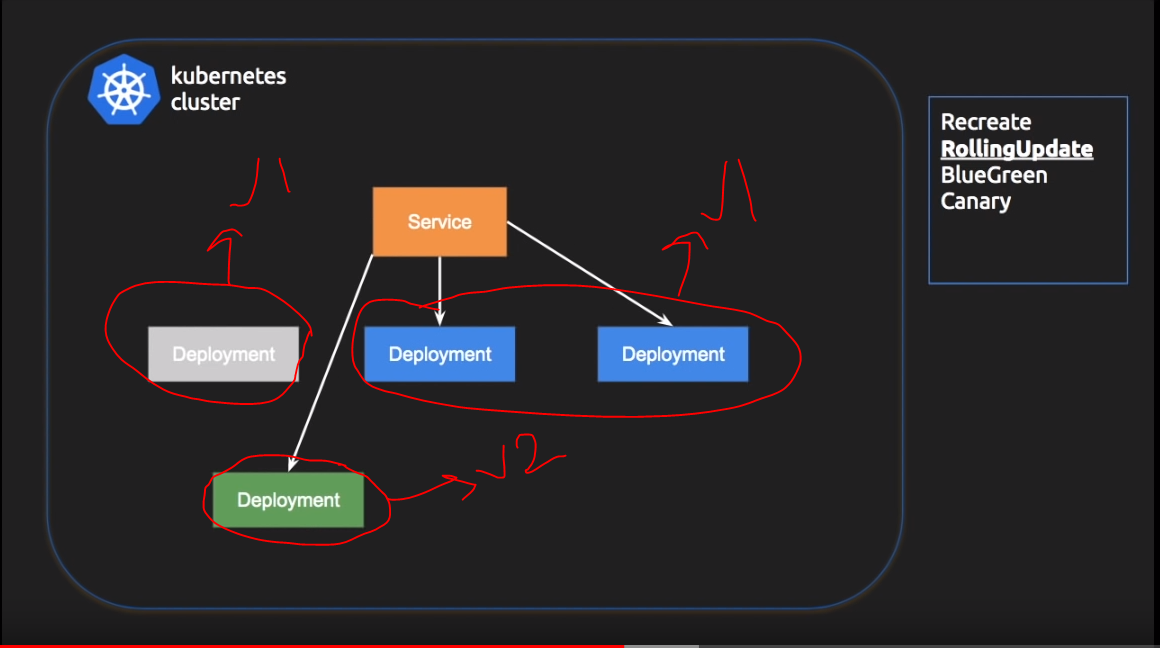




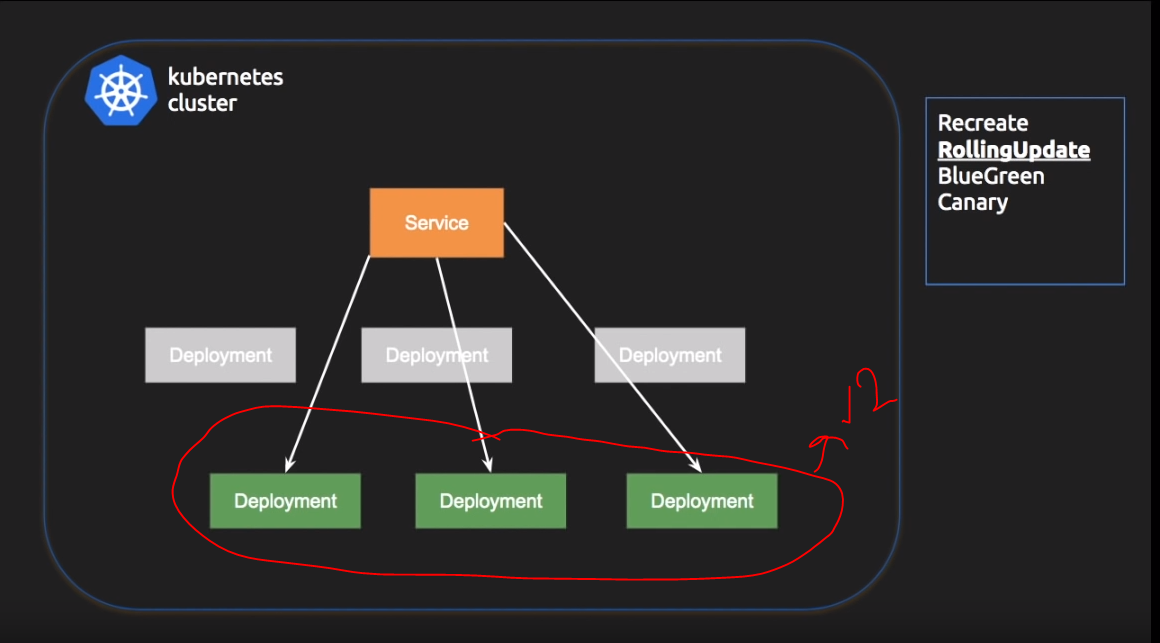
2). RollingUpdate,

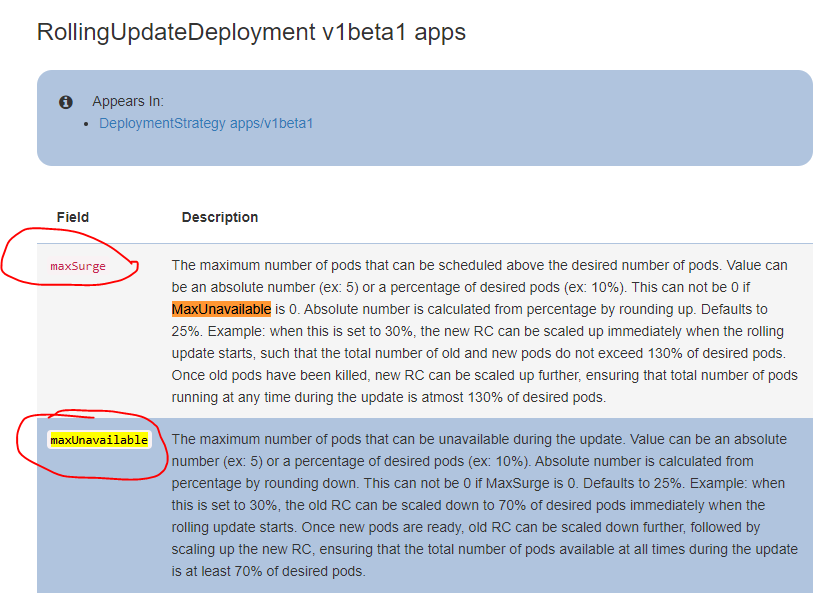
This strategy helps user to achieve **0 downtime** deployment.

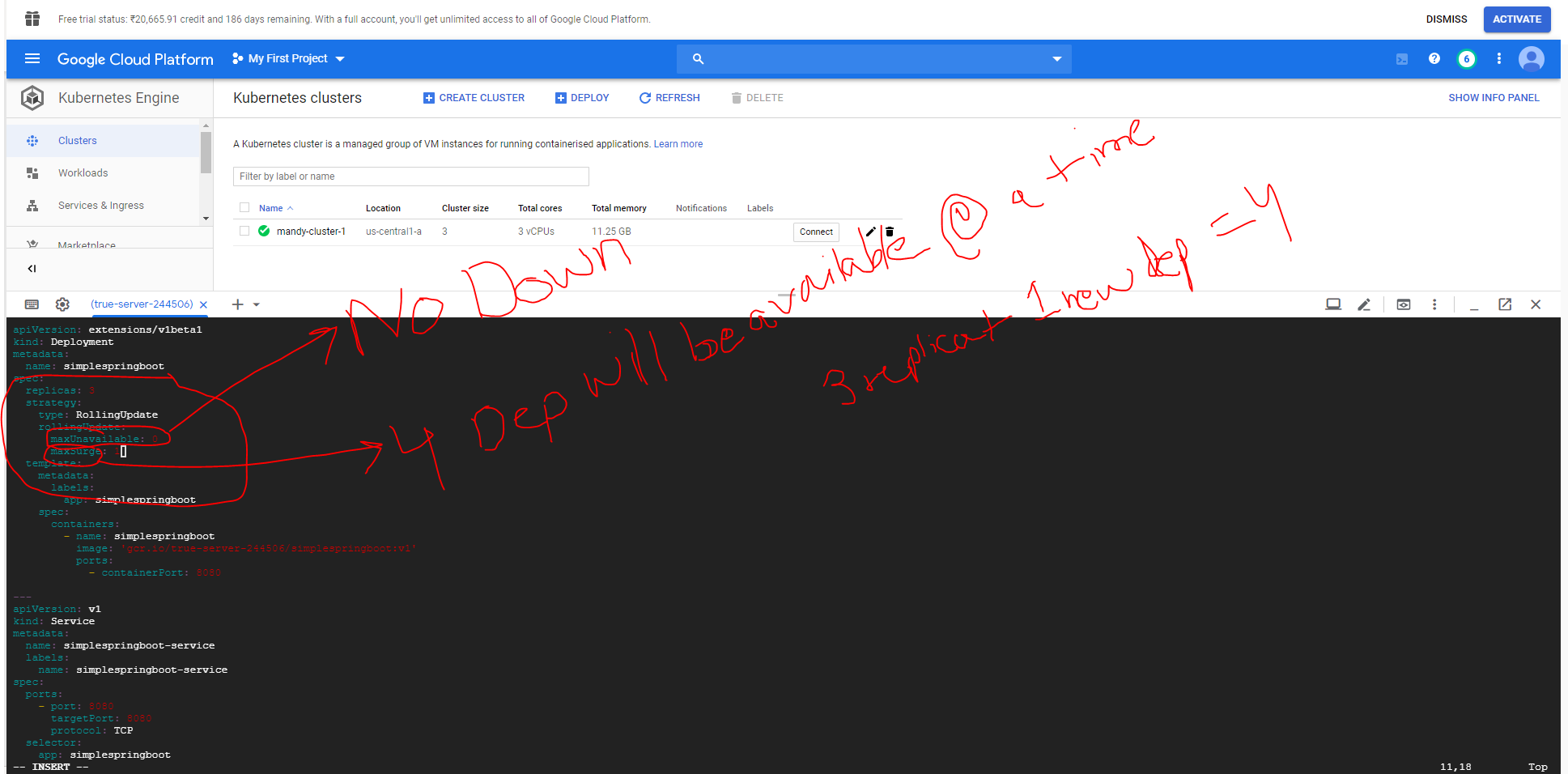
We are going to use **kube.yml** from previous example

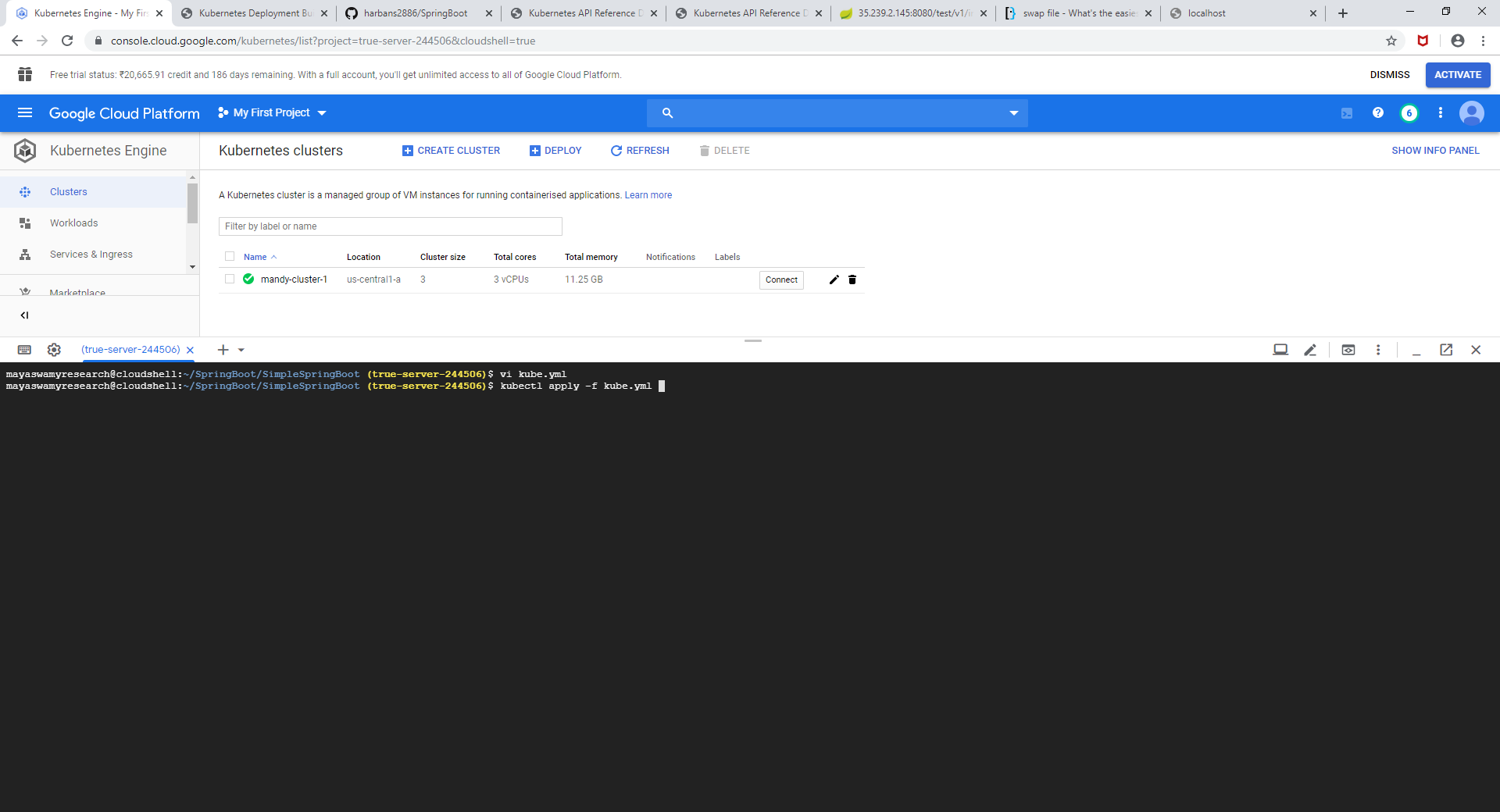


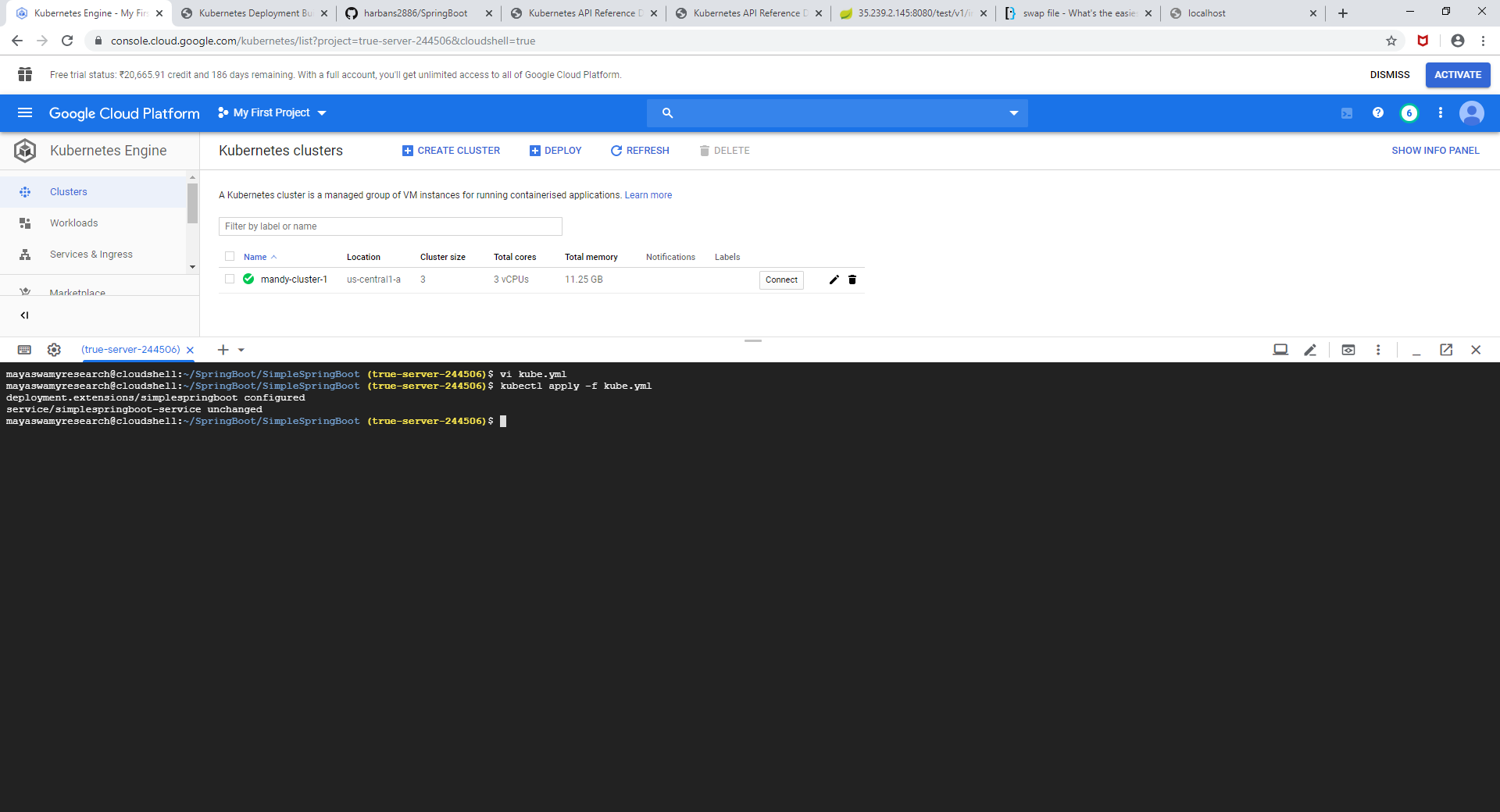
Final stage

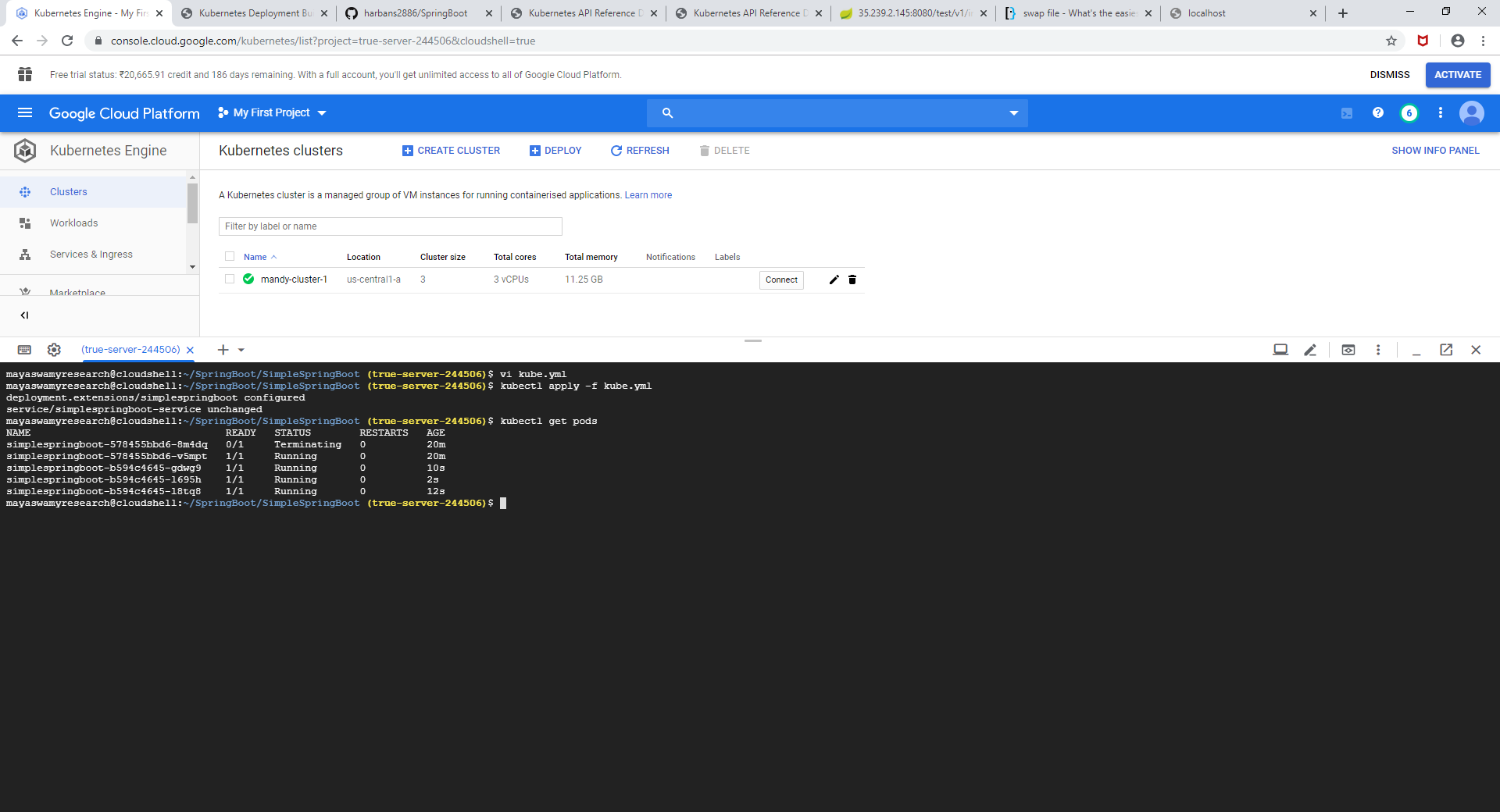


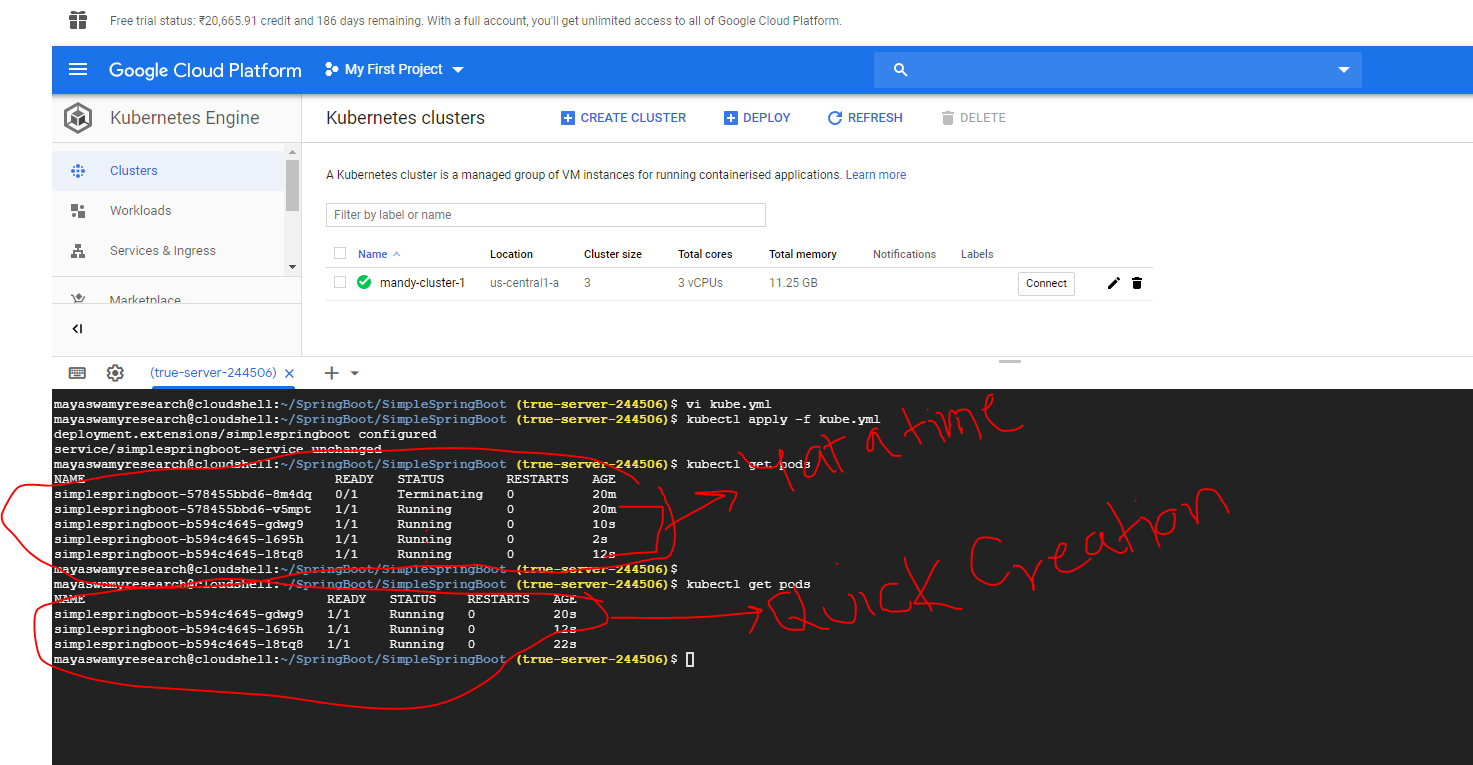






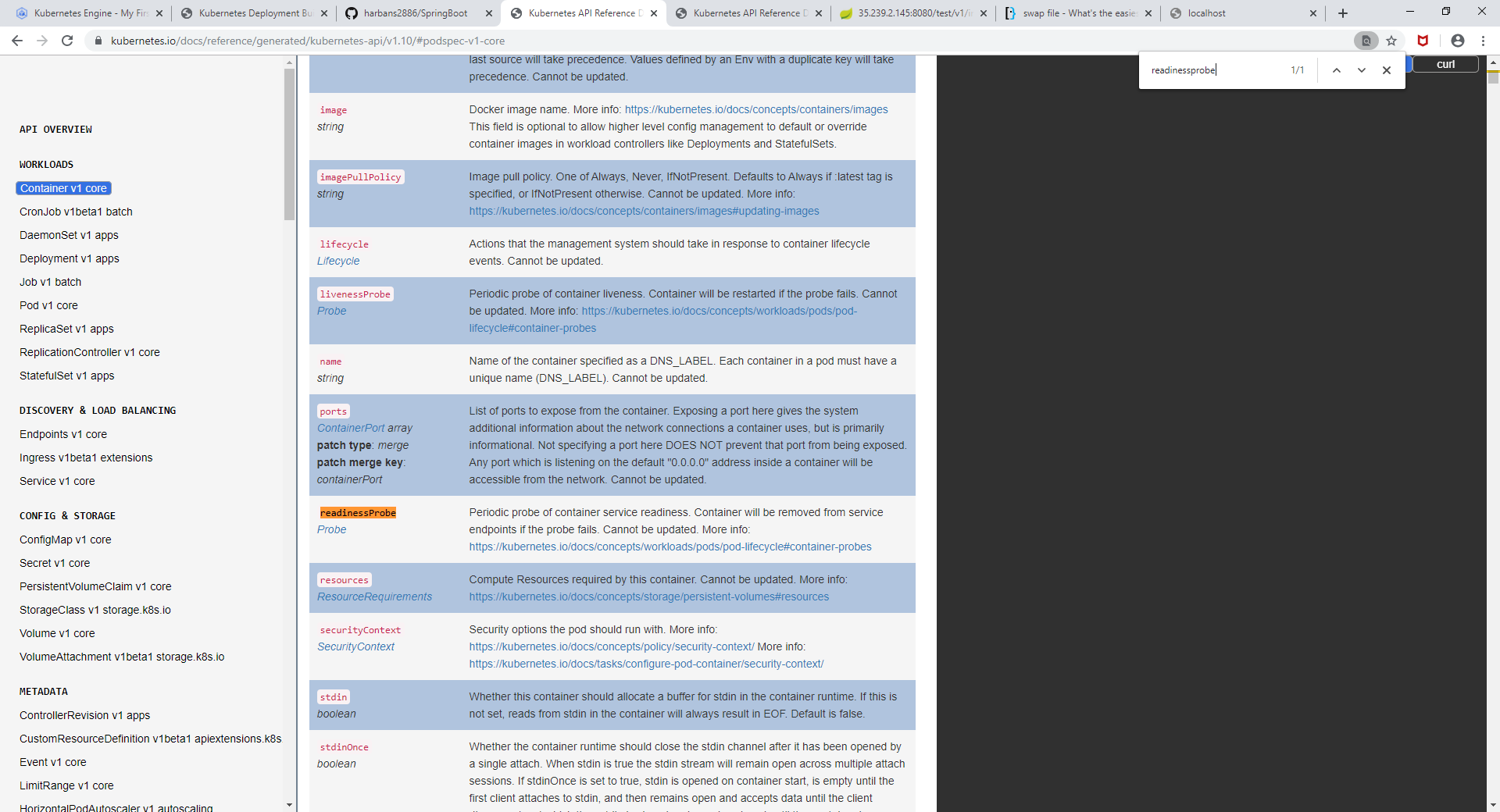


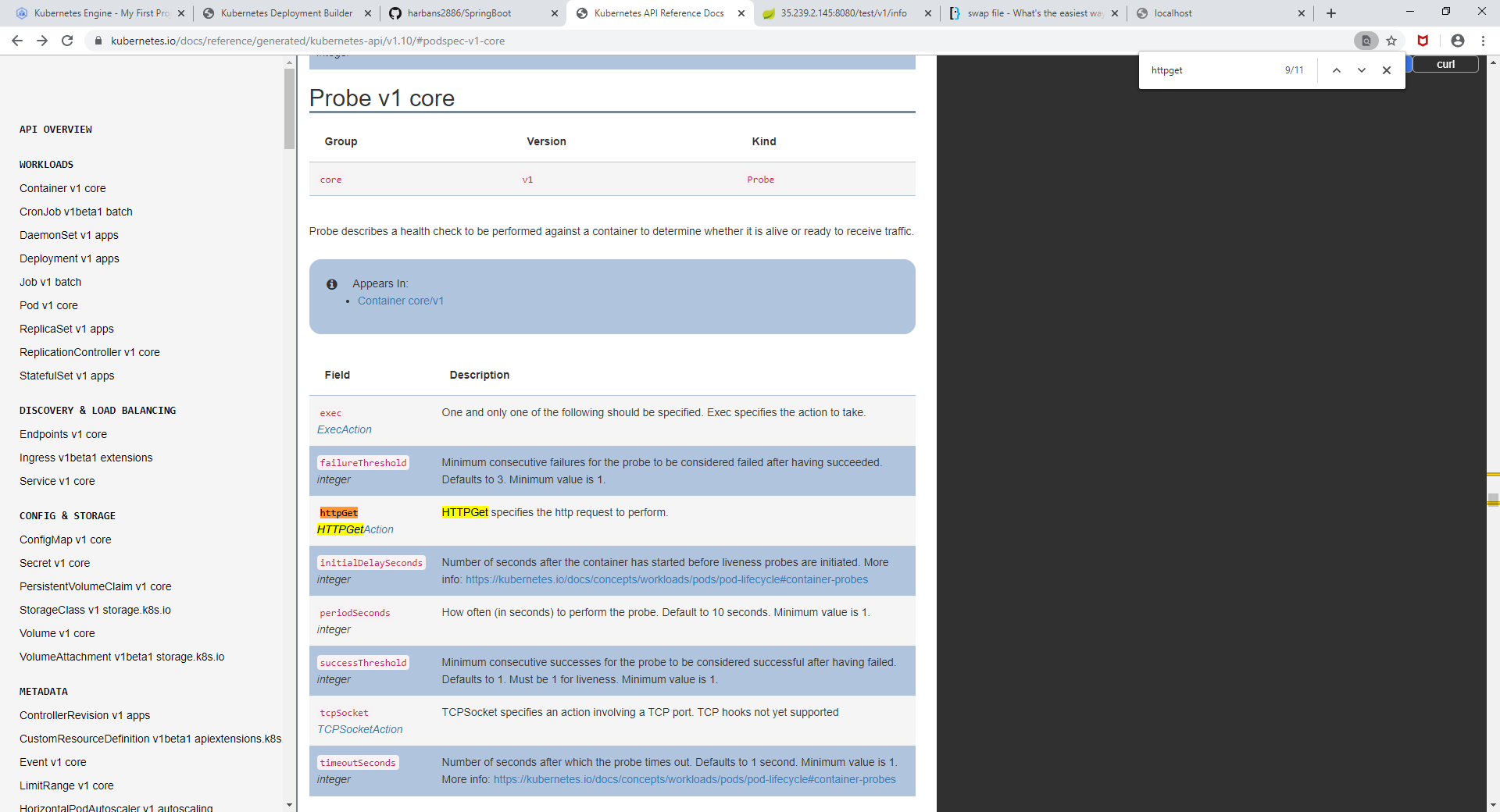




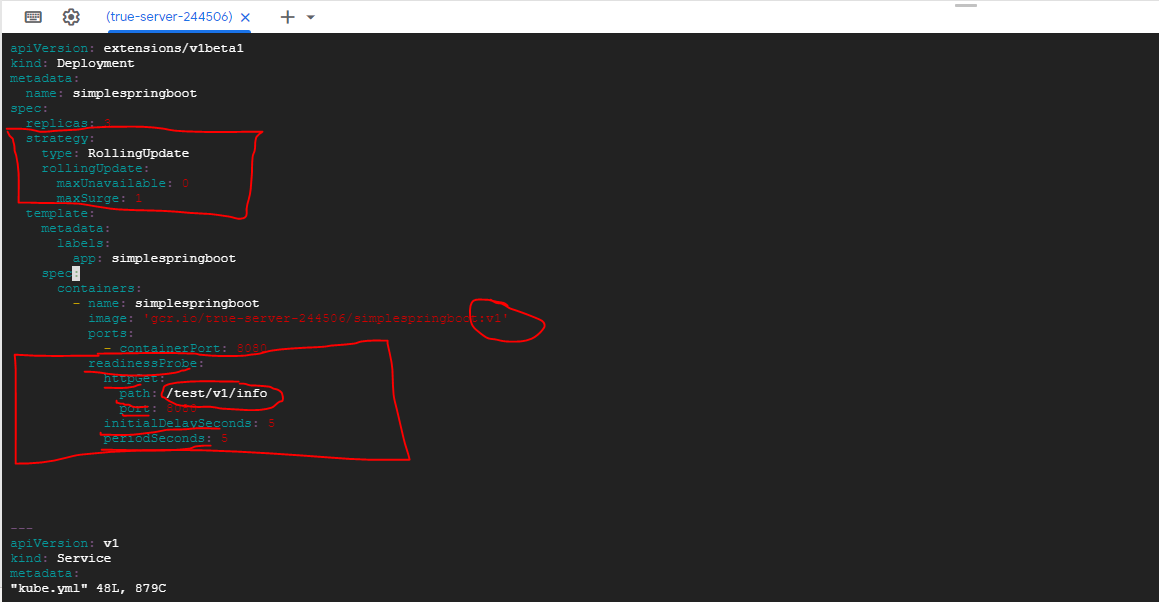


Add a delay to validate all the deployments are up and running

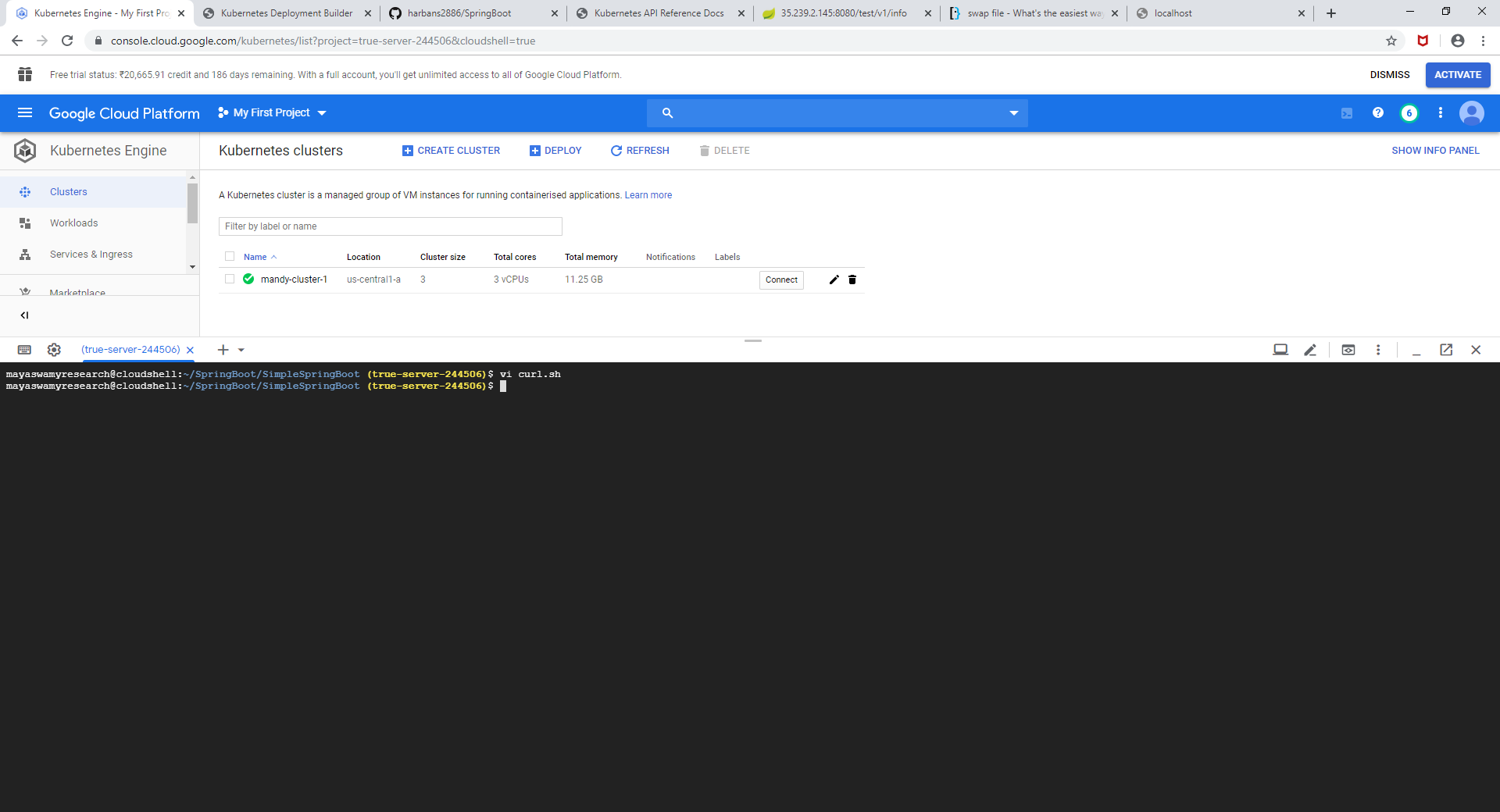




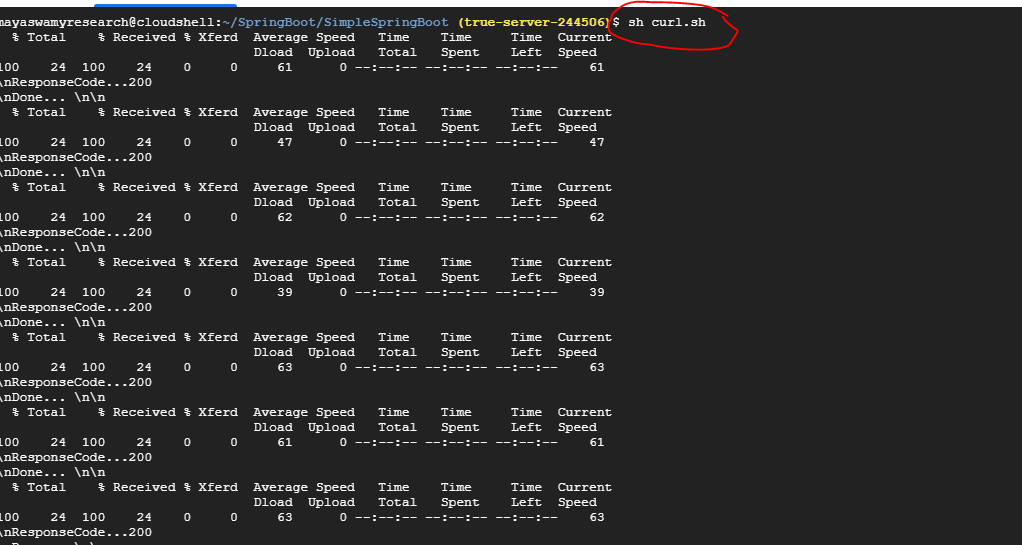
Change the version from v2 to v1 again

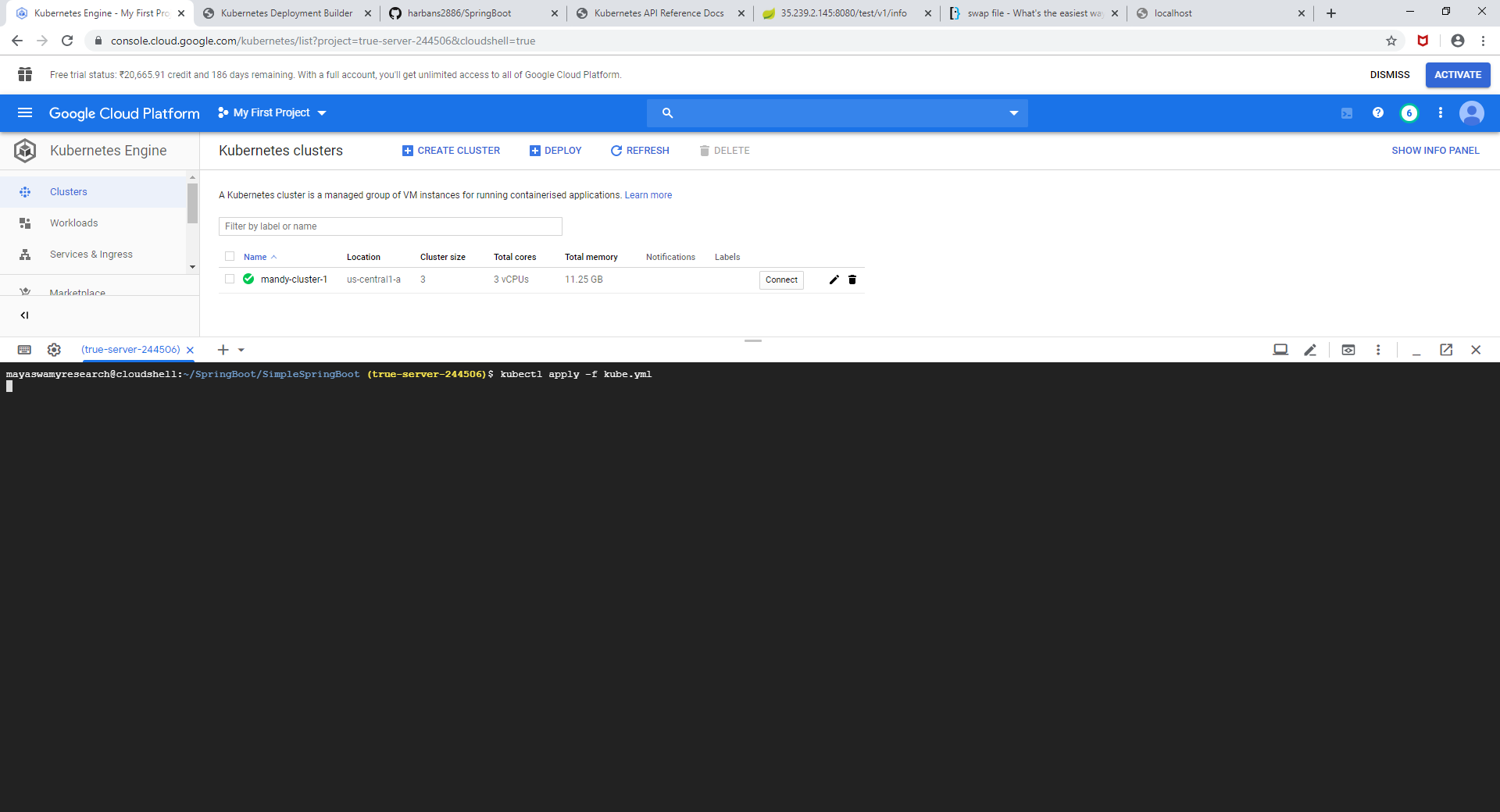


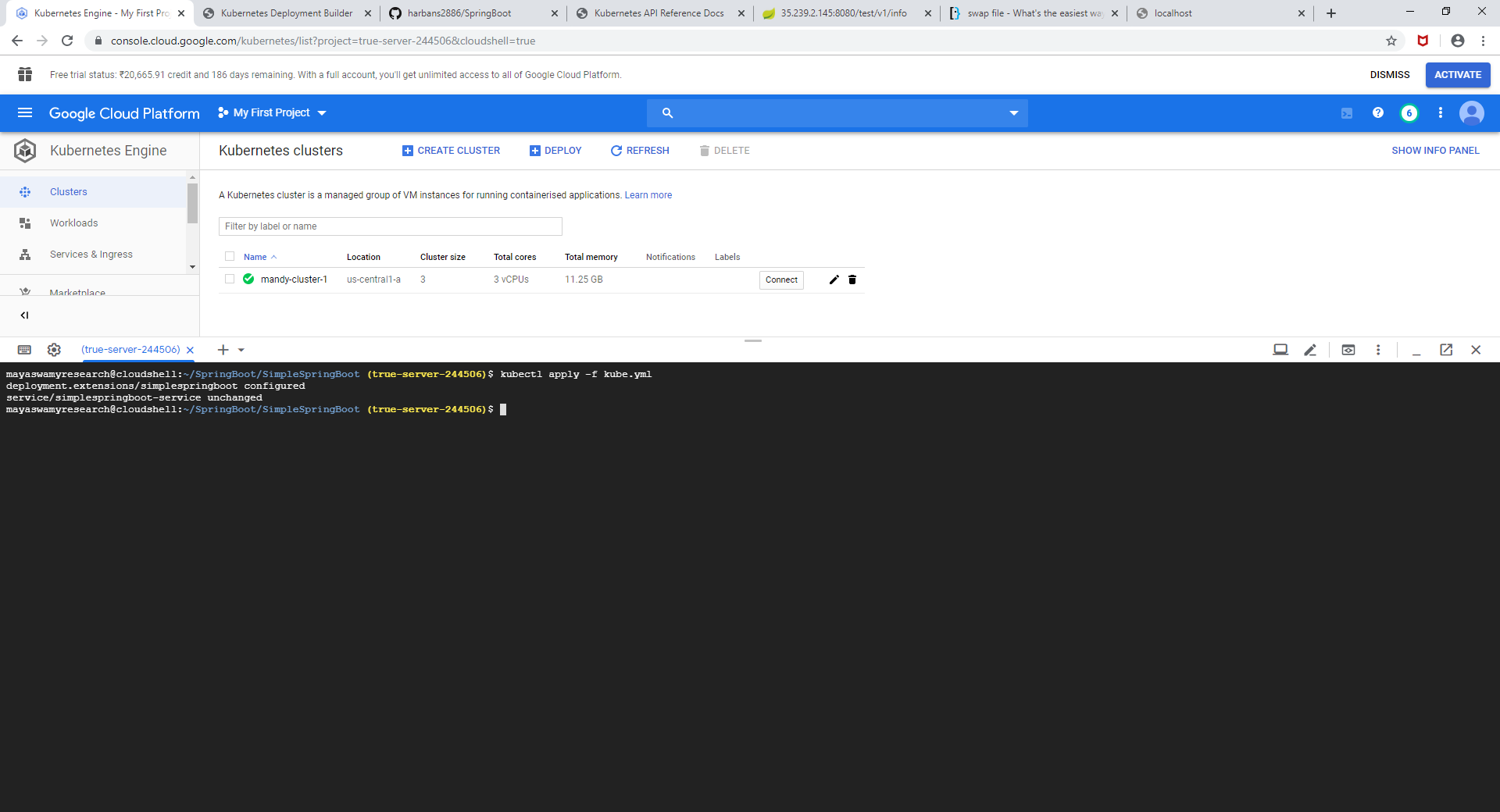
Create a file called curl.sh which will call our endpoint indefinitely to check if there is any down time.

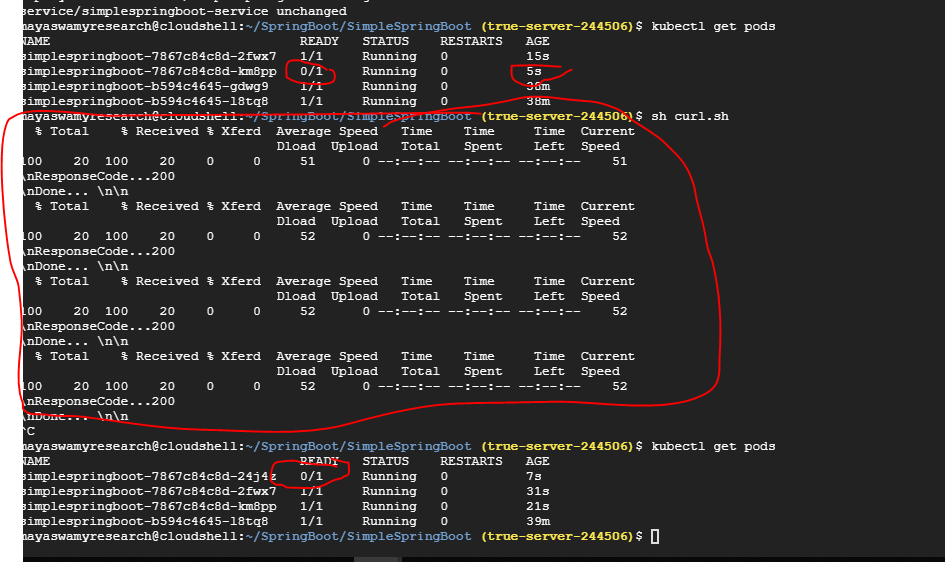




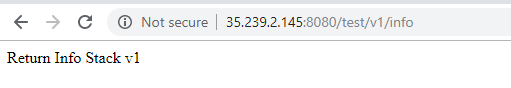








No down time even if new deployments are being prepared



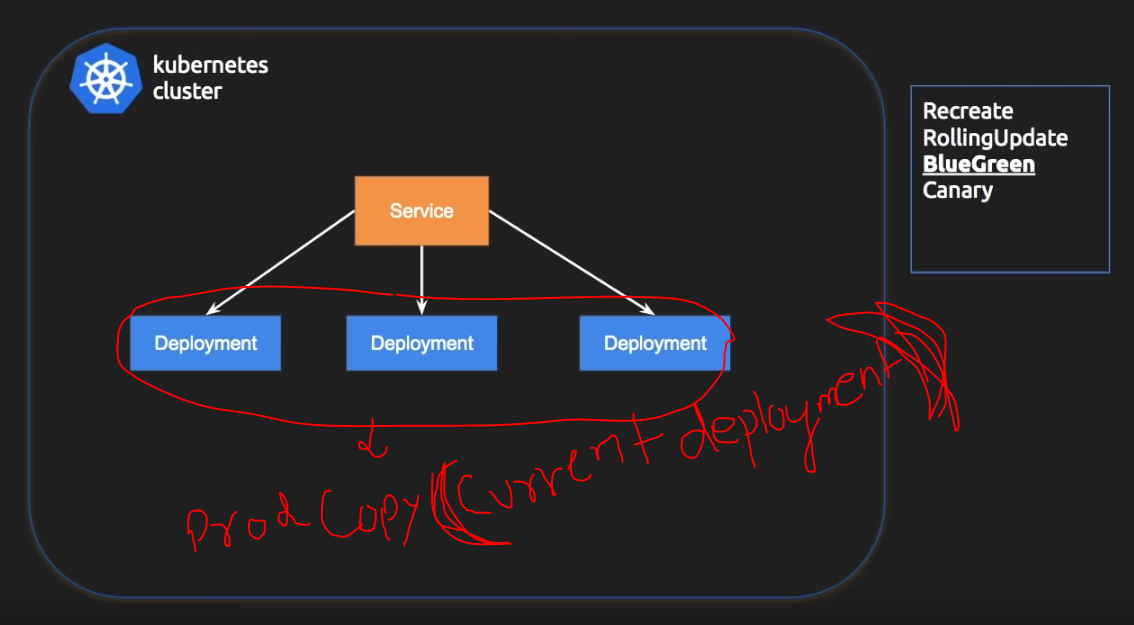
2). **BlueGreen**,

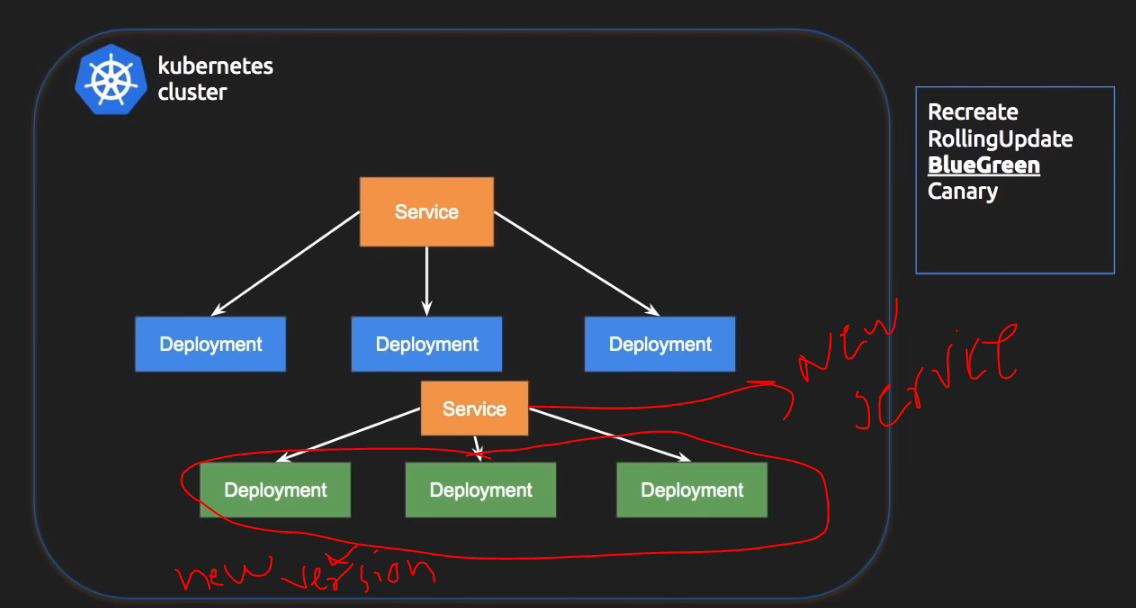
This strategy already running deployment objects are known are as **Blue** and new version of deployment objects are known as **Green**.

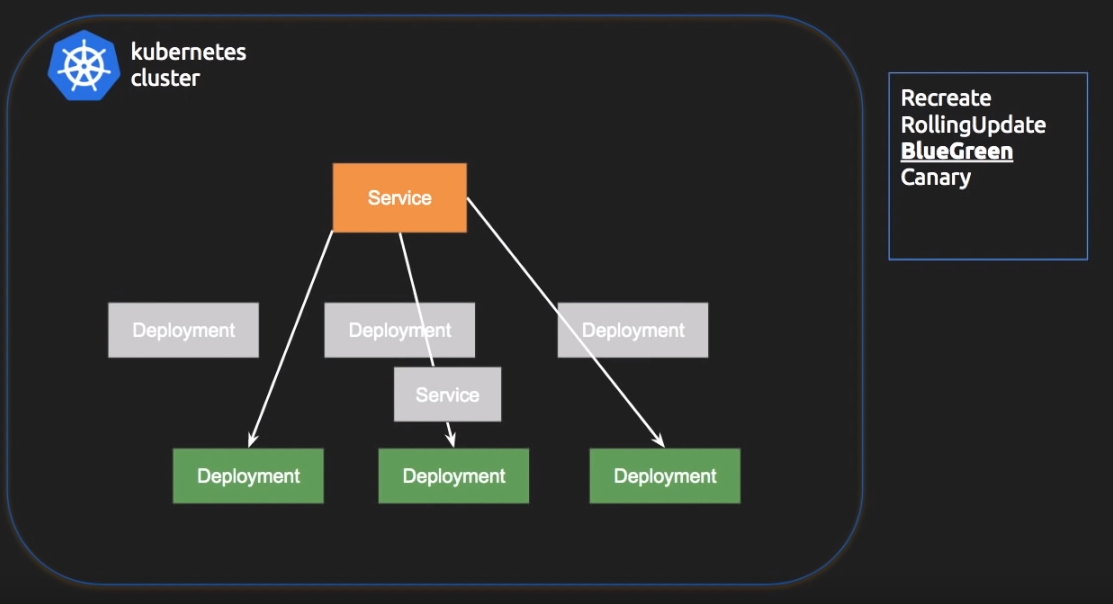
Green version deployment objects are grouped by a different Service to test that in prod environment without effecting the existing deployment.

Green version deployment objects will have different endpoint.

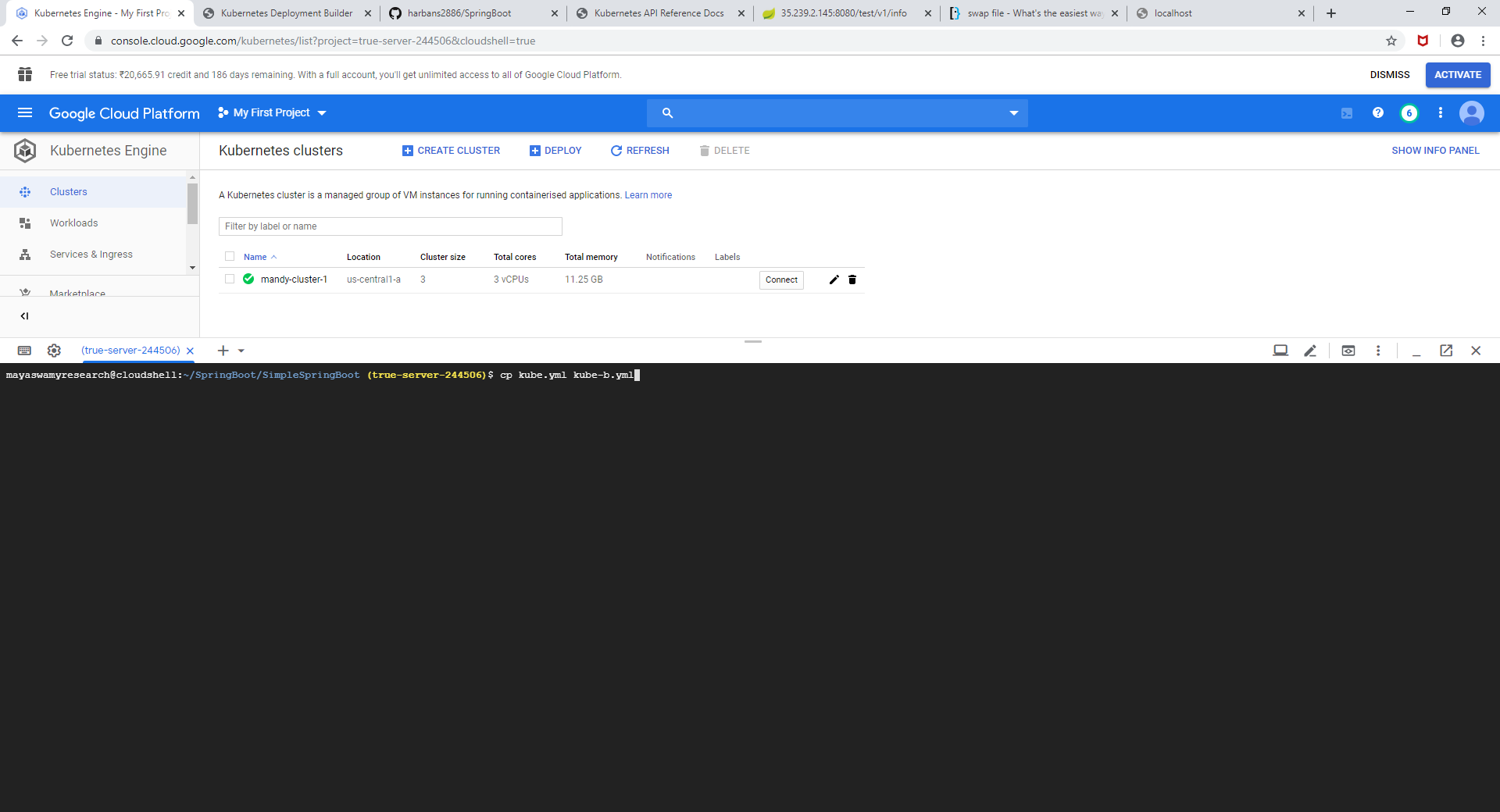
We are going to use **kube.yml** from previous example



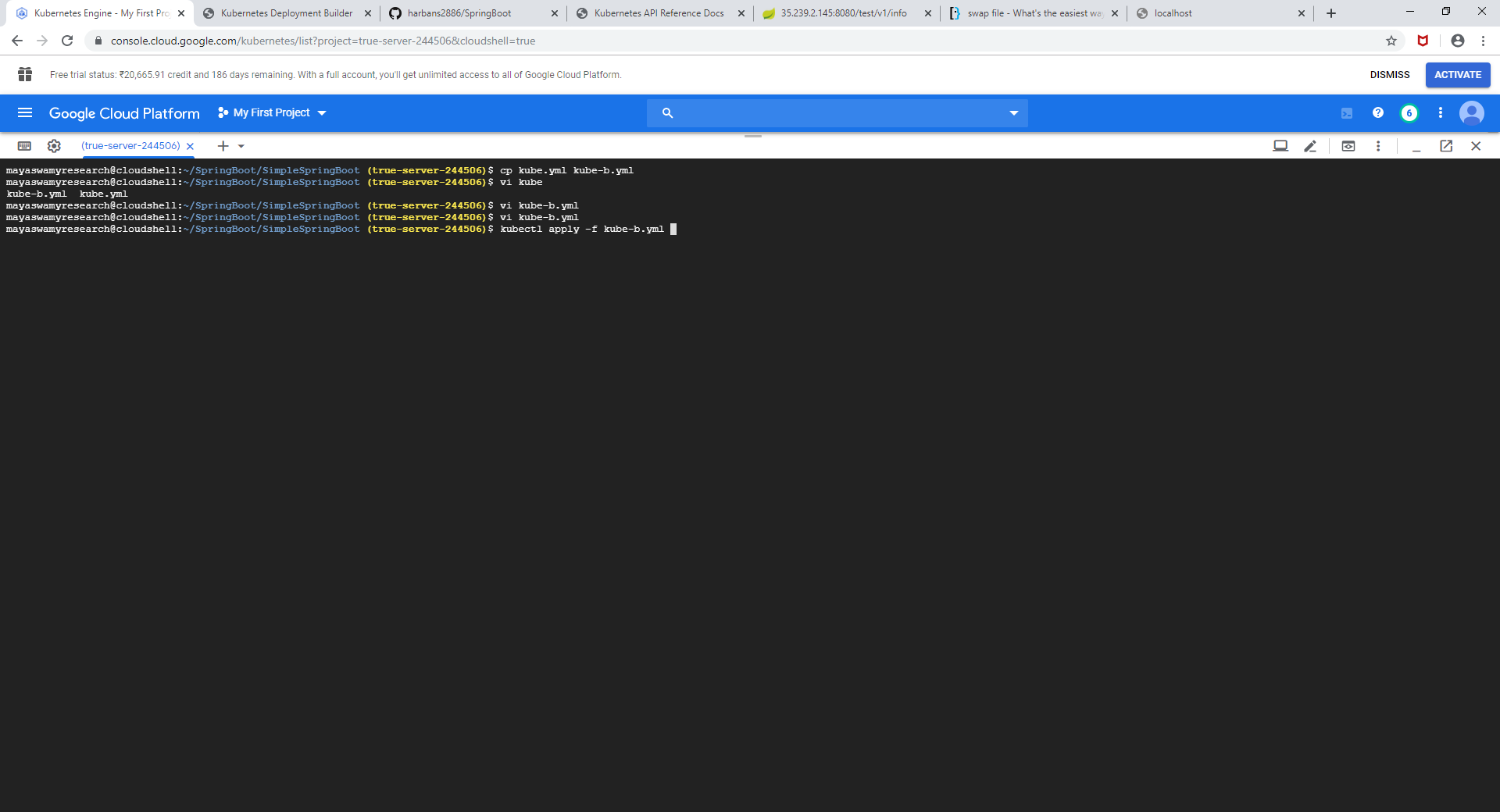


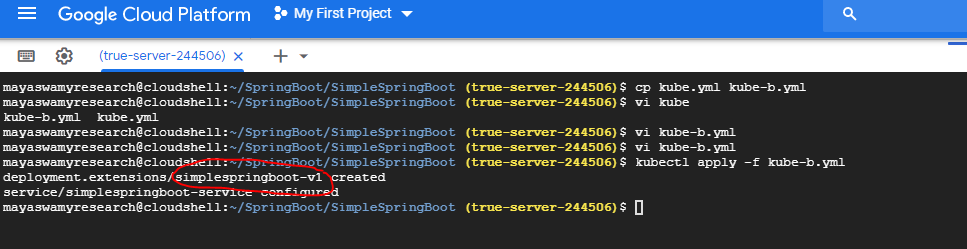


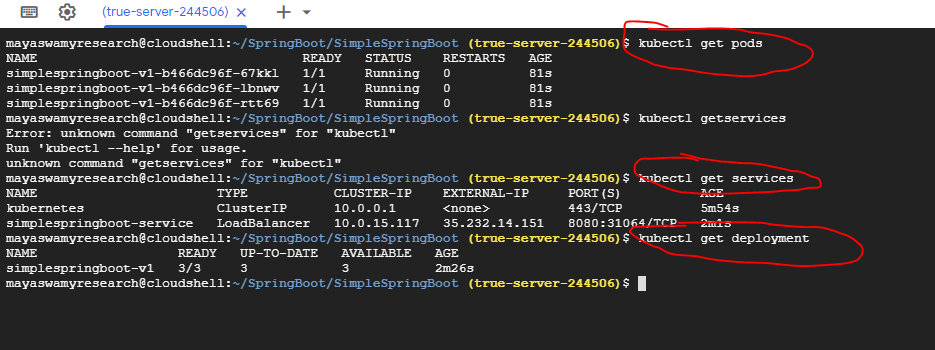
After the final testing, the Blue version gets destroyed and Green version becomes Blue this can be achieved by flipping\pointing the old rebalancer\Service to the new deployments.

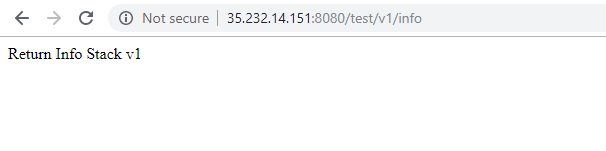
Copy the kube.yml contents into kube-b.yml(**Blue** version of deployment)





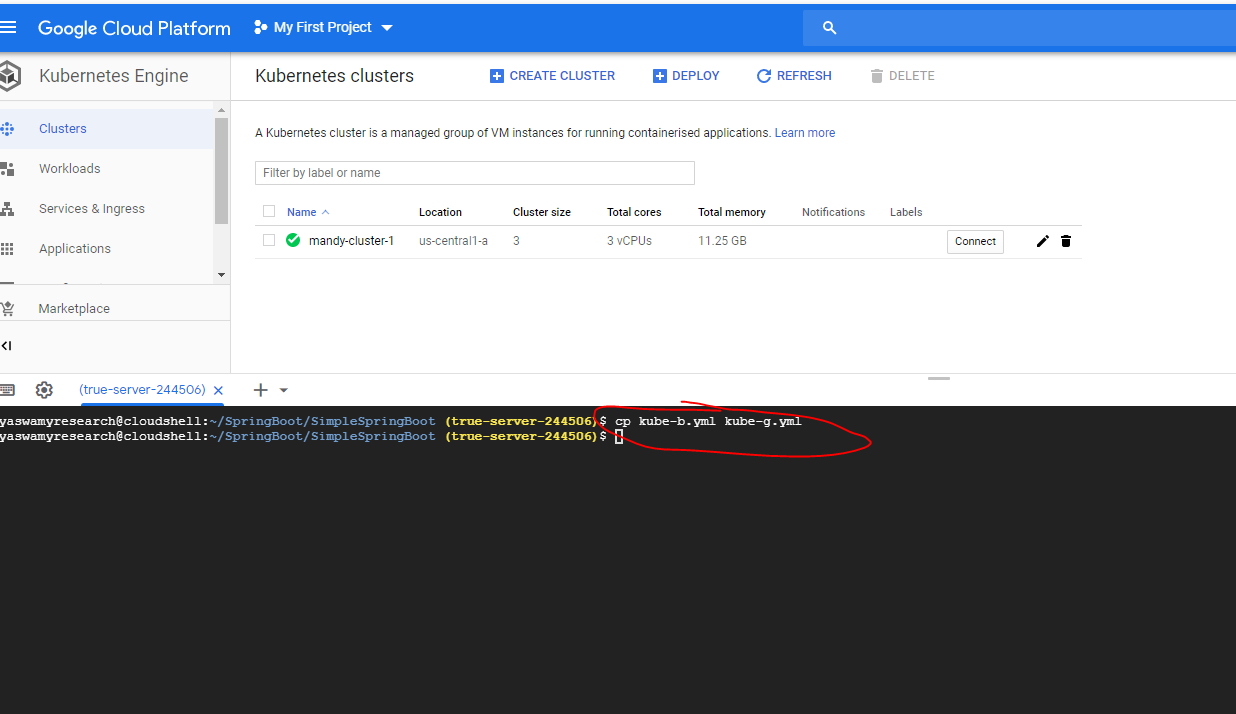


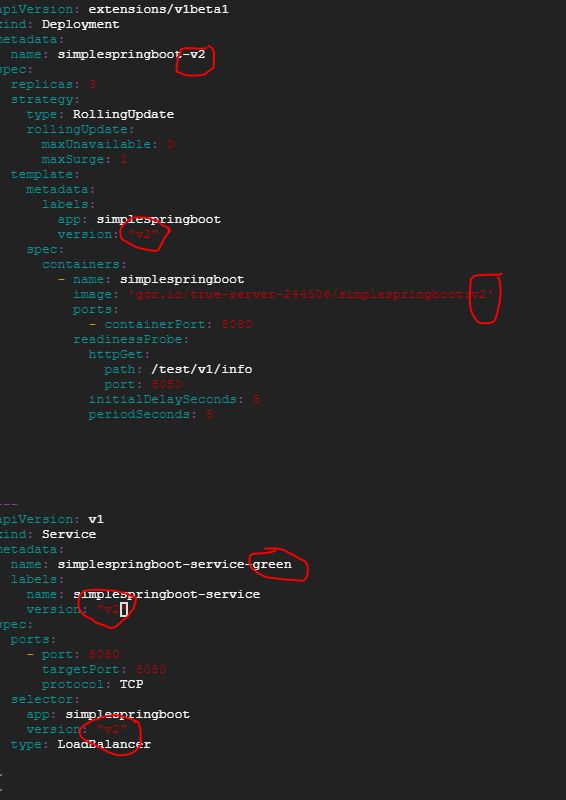


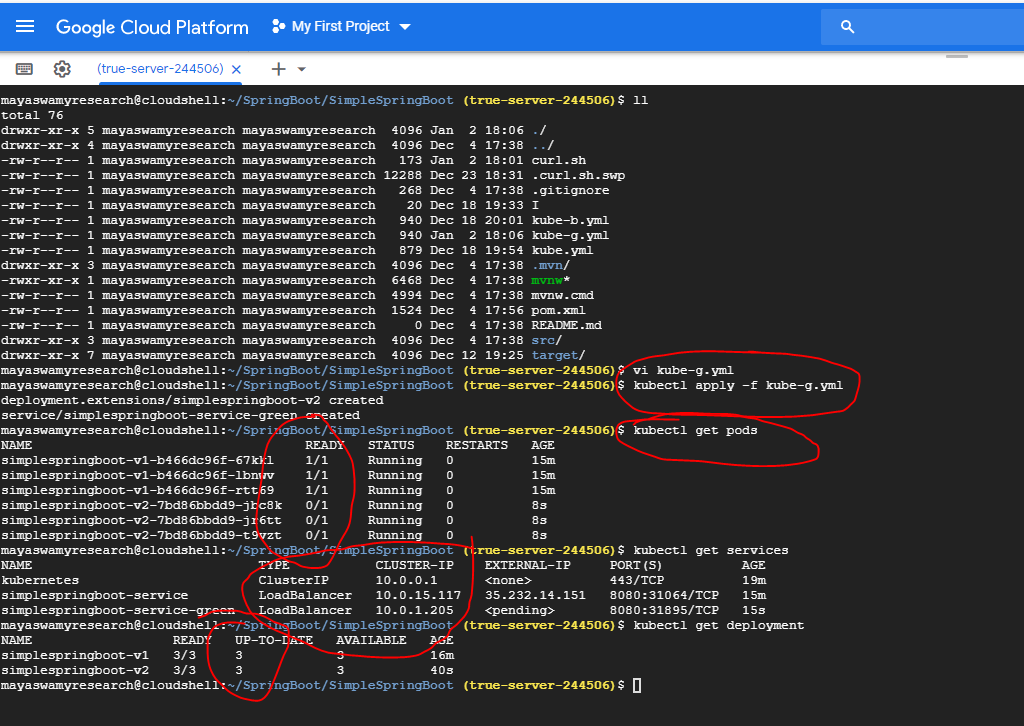


Now we have production version running already and we must deploy the new version which is **Green** version of the deployment.

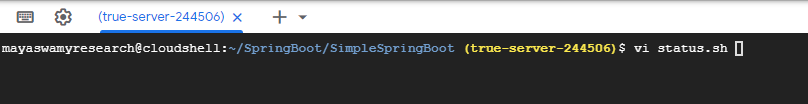
In order to do that we have to create a new deployment file kube-g.yml

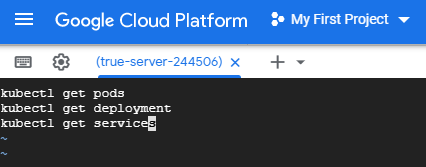


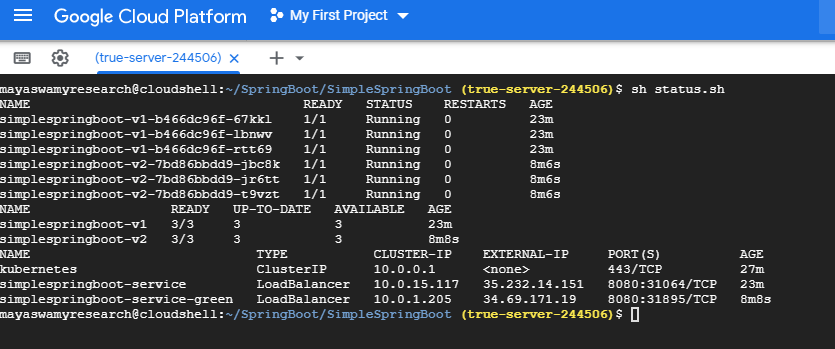




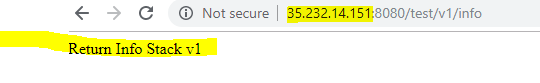
To get the status of services, deployments and pods I have created a file called status.sh

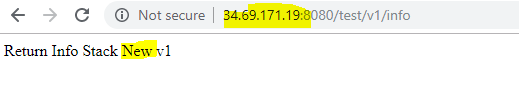




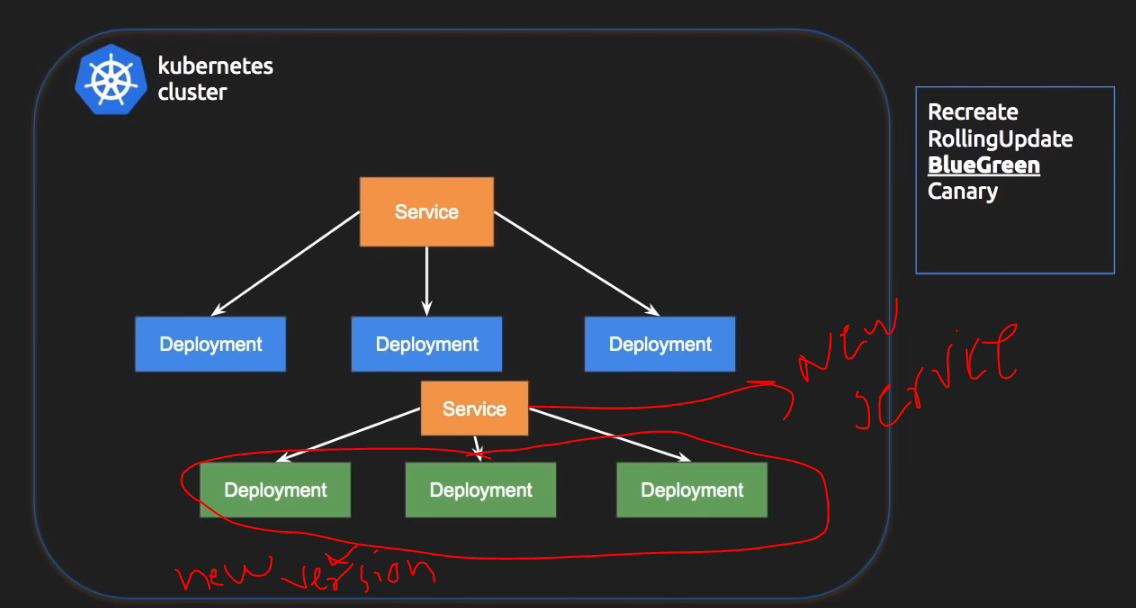


Now we can see there are 2 sets of pods with v1 and v2, 2 sets of deployments and 2 loadbalancer services.

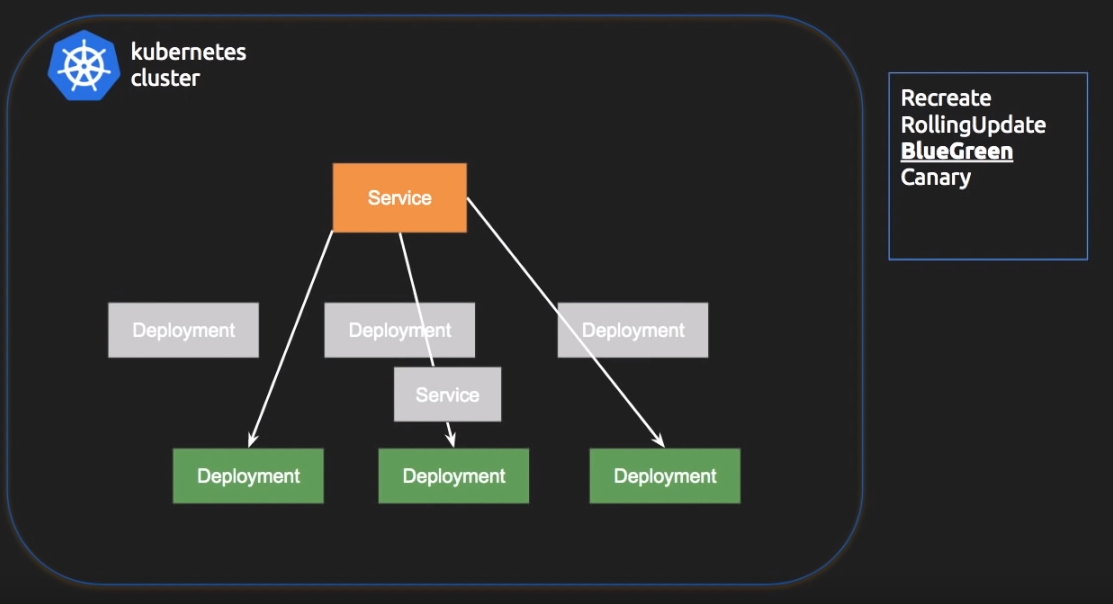




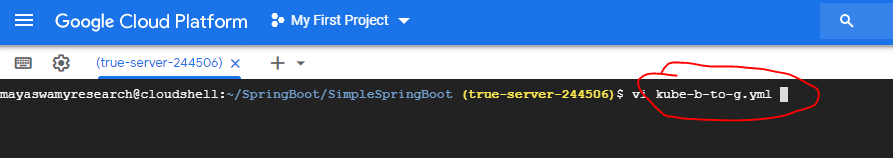
Now we have a created the scenario show in the screenshot below



Now we must convert the deployment from screenshot above to screenshot below.

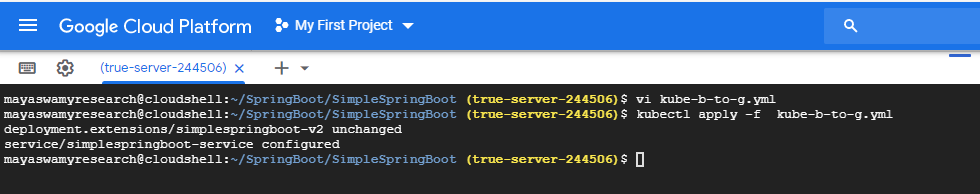


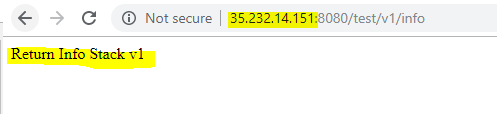
Create a .yml file to turn green deployments to blue. For that I have created a file called kube-b-to-g.yml .

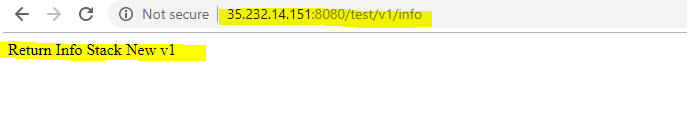


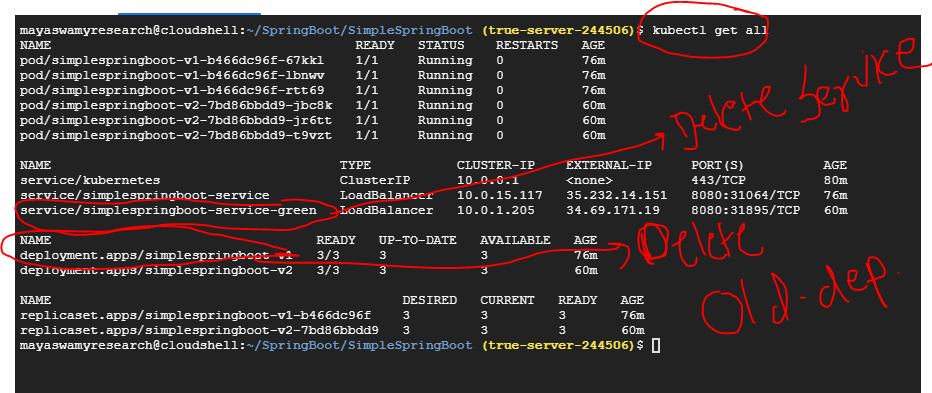
The file will remain the same as kube.g.yml but the service name will be the older one which will be pointing to the v2 deployments and pods.

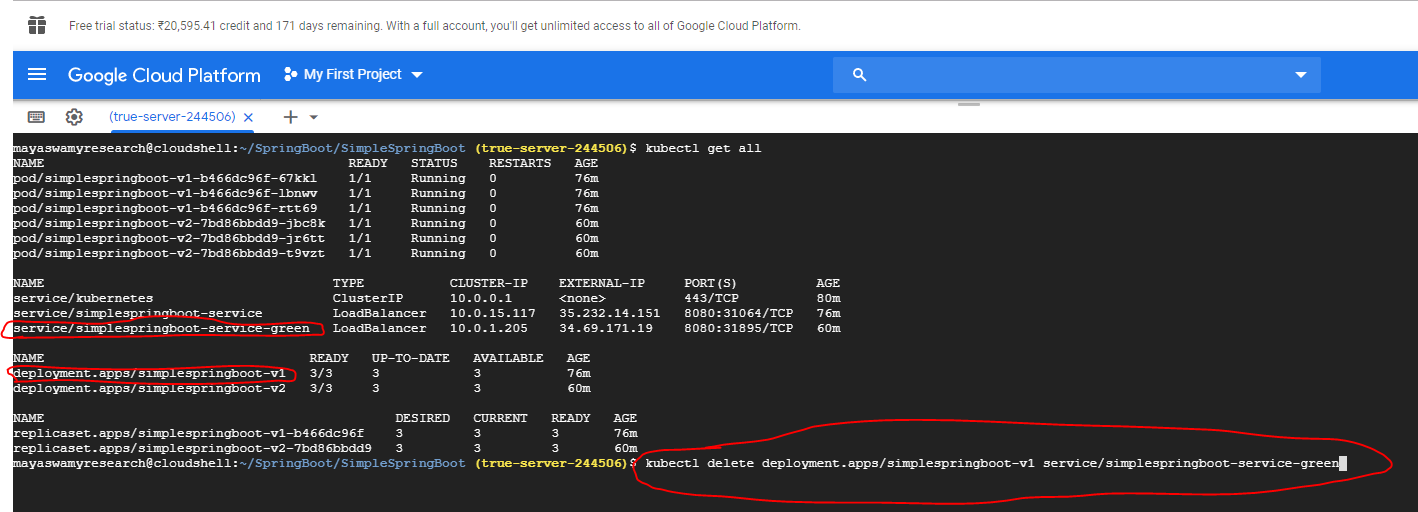












kubectl delete deployment.apps/simplespringboot-v1 service/simplespringboot-service-green

