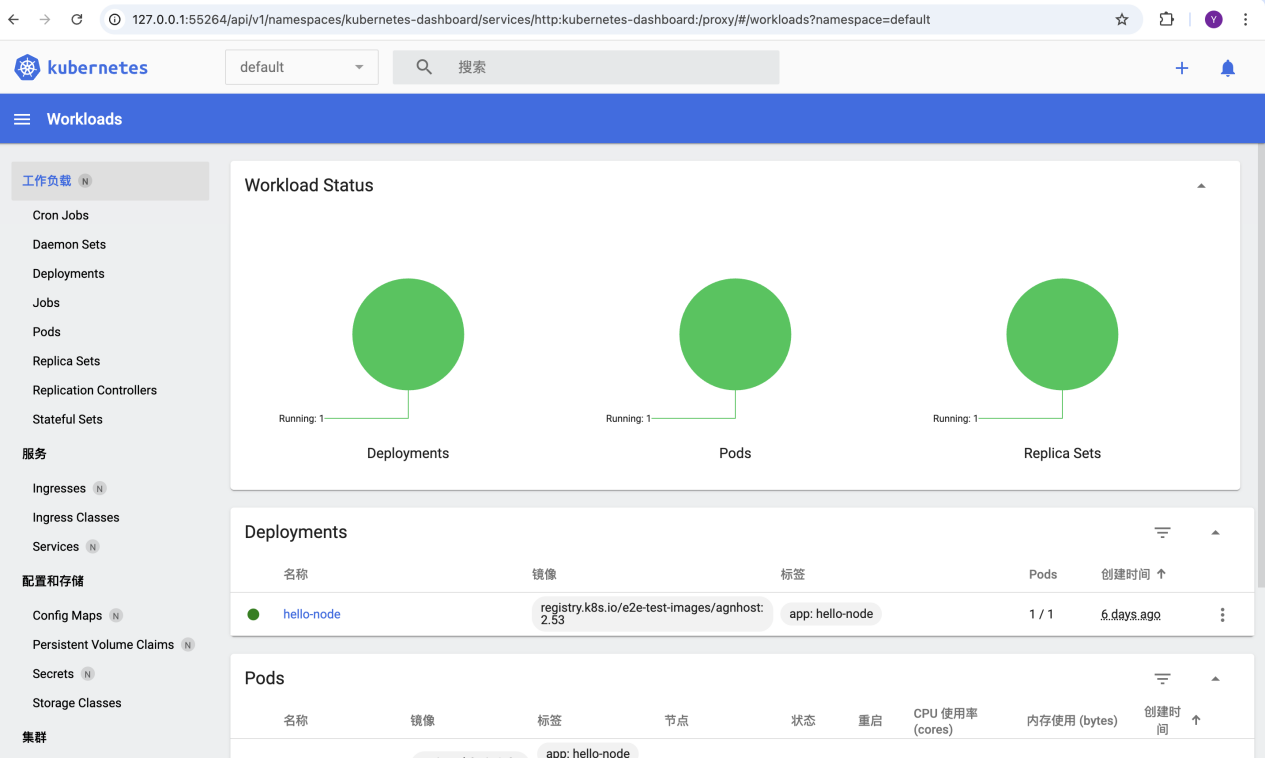
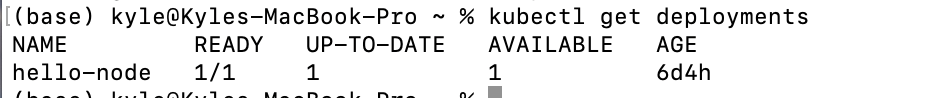
Open the Dashboard

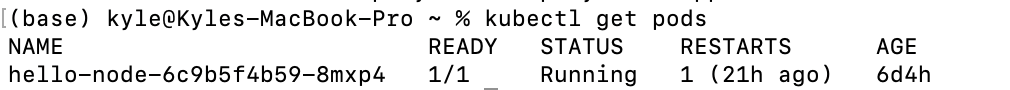


Create a Deployment

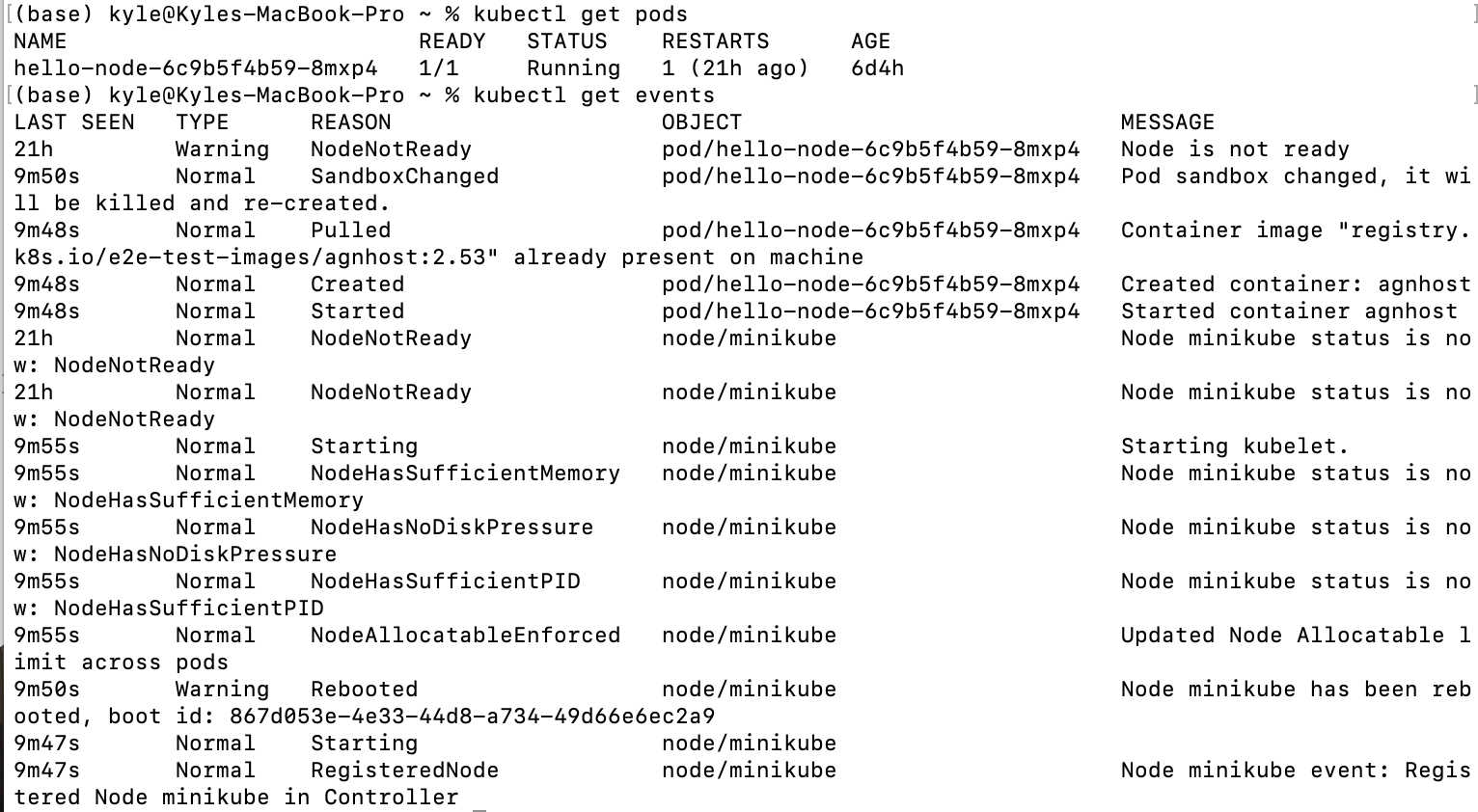
View the Deployment:



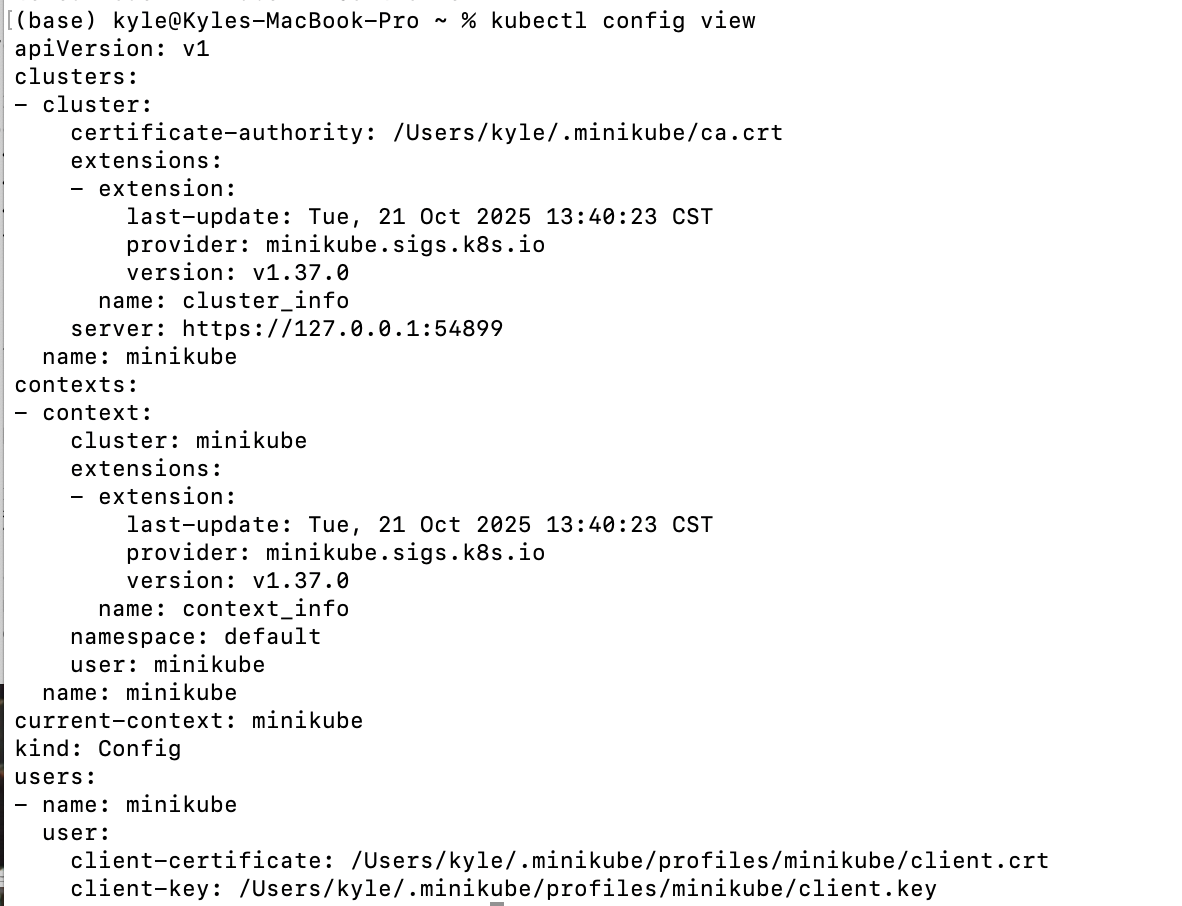
View the Pod:



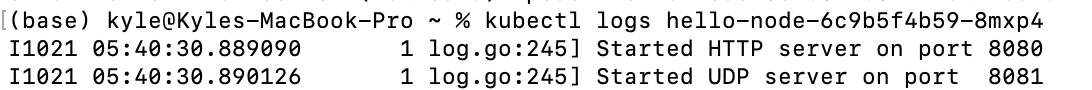
View cluster events:



View the kubectl configuration:



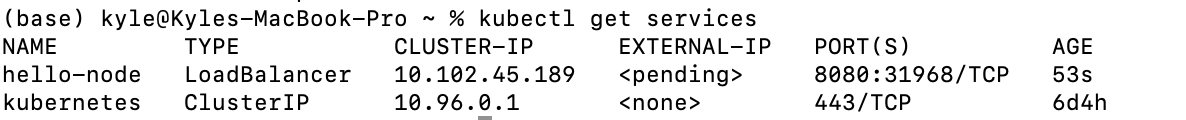
View application logs for a container in a pod (replace pod name with the one you got from kubectl get pods).



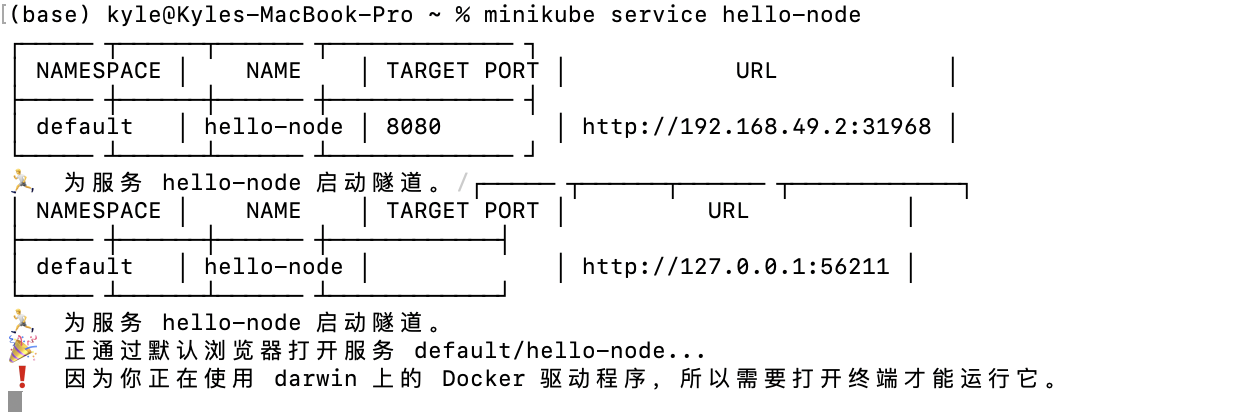
Expose the Pod to the public internet

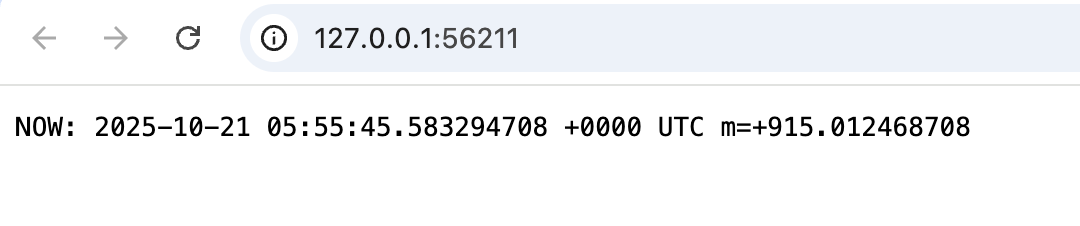
截屏2025-10-21 13.54.35

View the Service

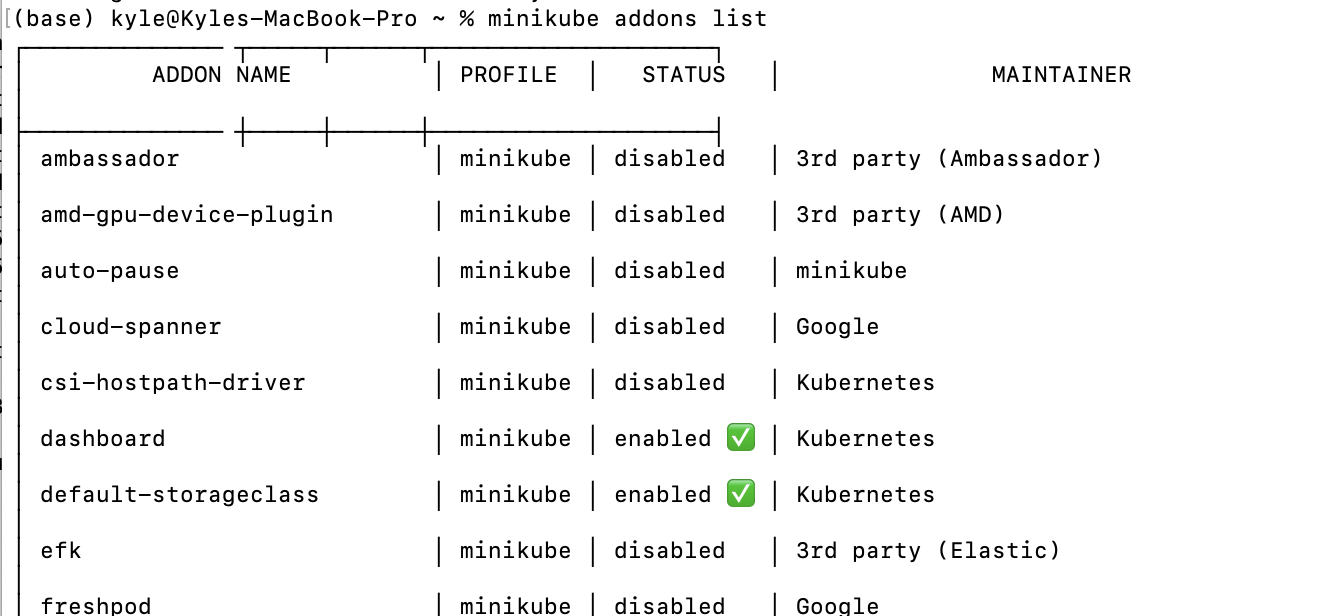


Open up a browser window that serves my app and shows the app's response.





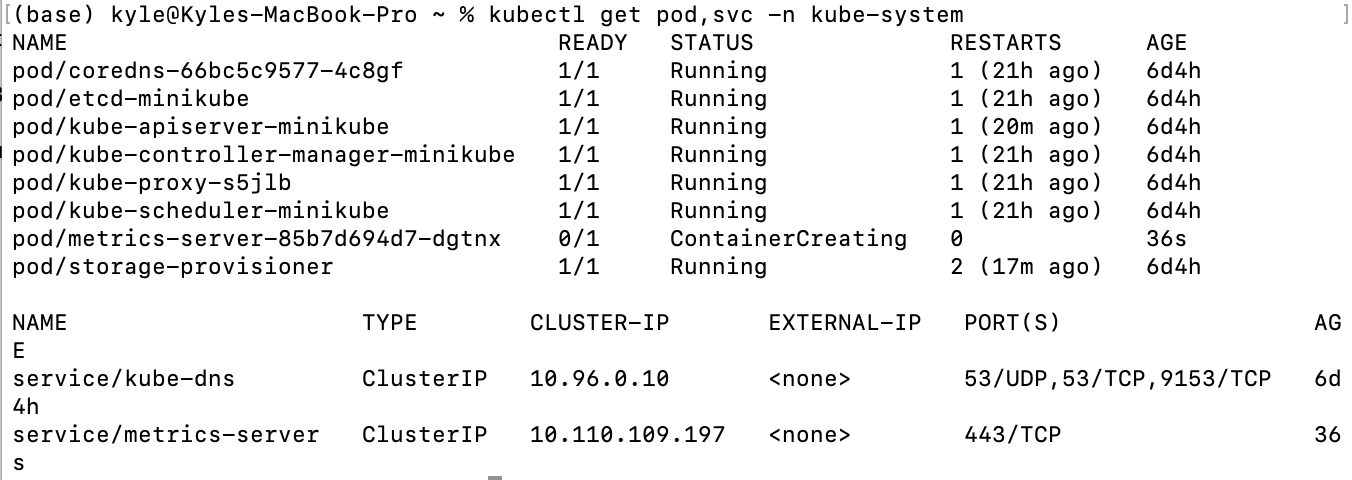
List the currently supported addons:



Enable an addon, for example, metrics-server:



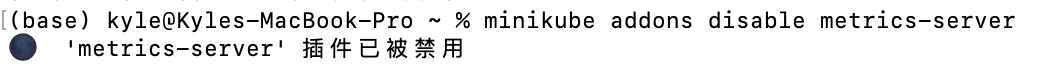
View the Pod and Service



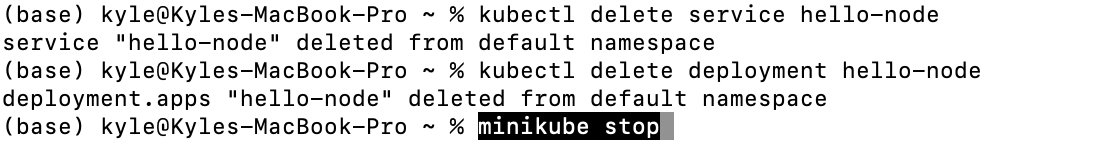
Check the output from metrics-server

截屏2025-10-21 13.59.32

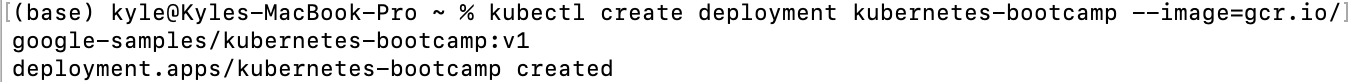
So i need to disable the metrics-server



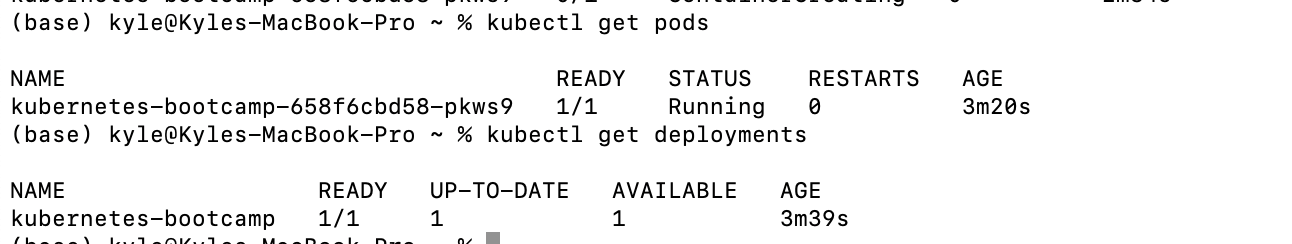
Clean up



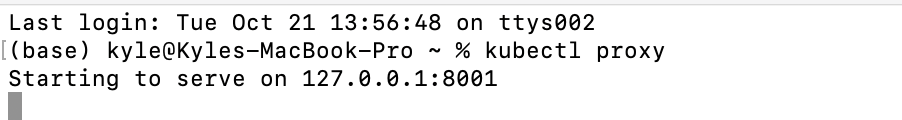
Deploy an app



To list deployments



open a second terminal window to run the proxy.

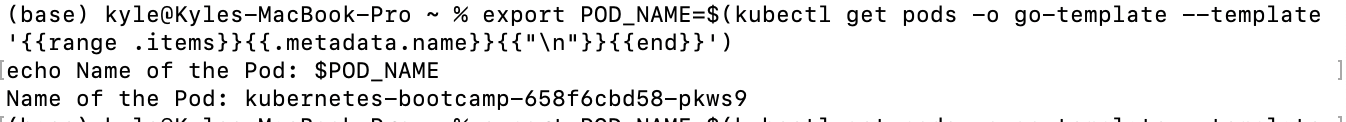




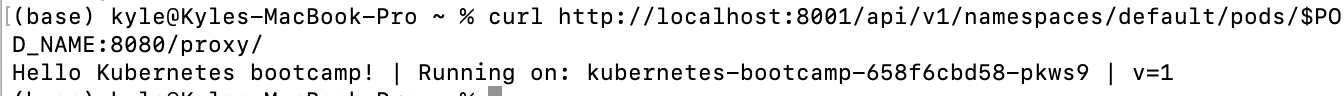
See all those APIs hosted through the proxy endpoint.



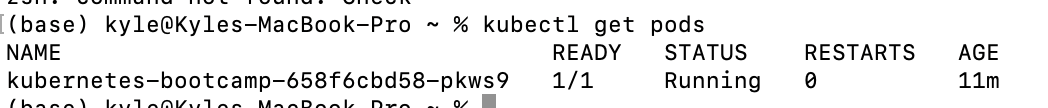
get the Pod name, and we'll store it in the environment variable POD\_NAME.



Access the Pod through the proxied API

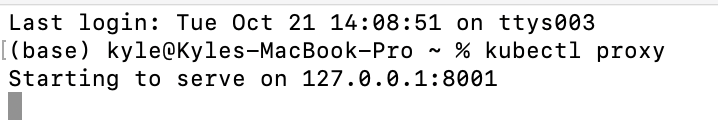


Check application configuration



View what containers are inside that Pod and what images are used to build those containers

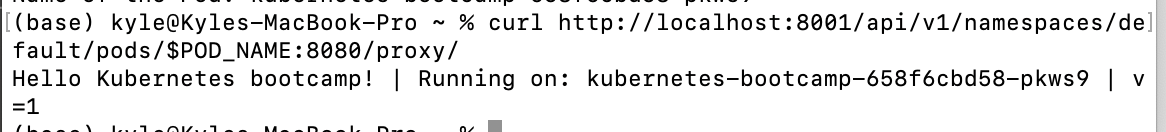




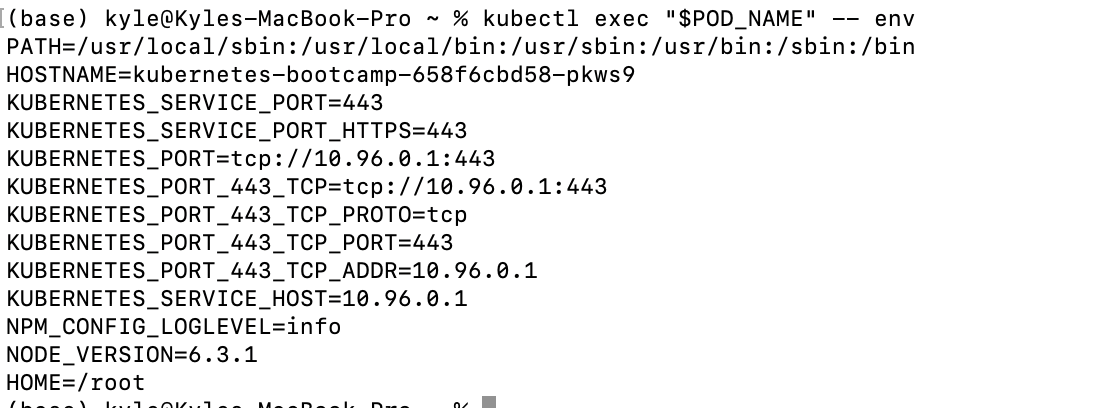
Now again, we'll get the Pod name and query that pod directly through the proxy. To get the Pod name and store it in the POD\_NAME



The URL is the route to the API of the Pod.



we use the exec subcommand and use the name of the Pod as a parameter.

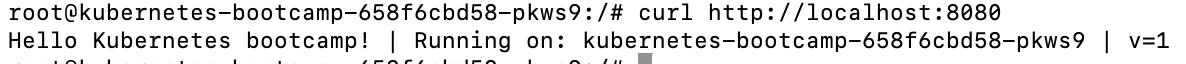


Start a bash session in the Pod’s container

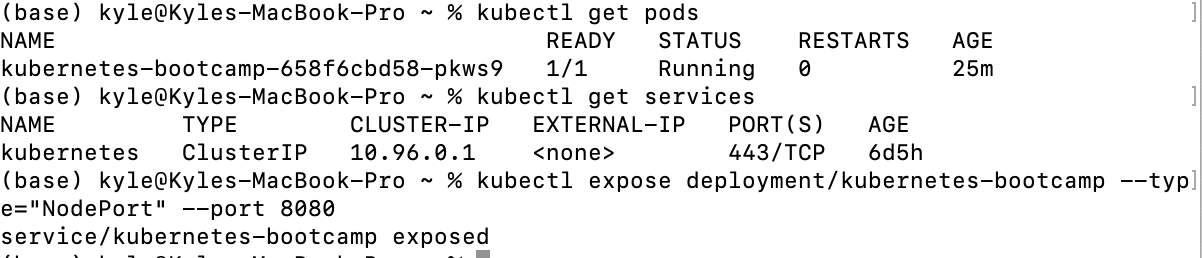
We have now an open console on the container where we run our NodeJS application. The source code of the app is in the server.js file



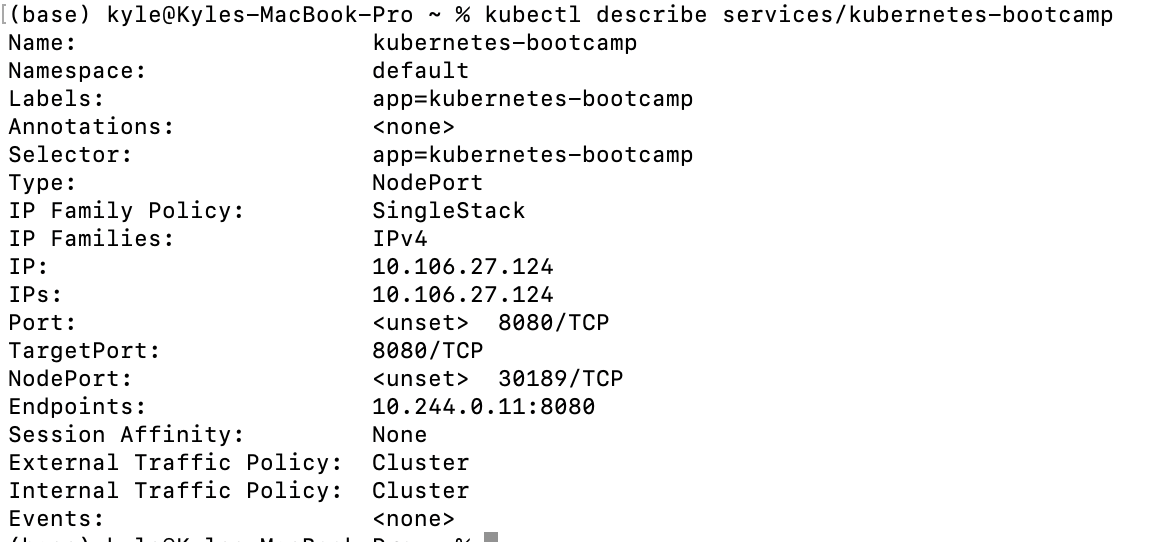
Check that the application is up



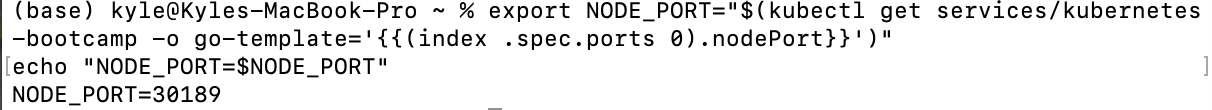
Step 1: Creating a new Service



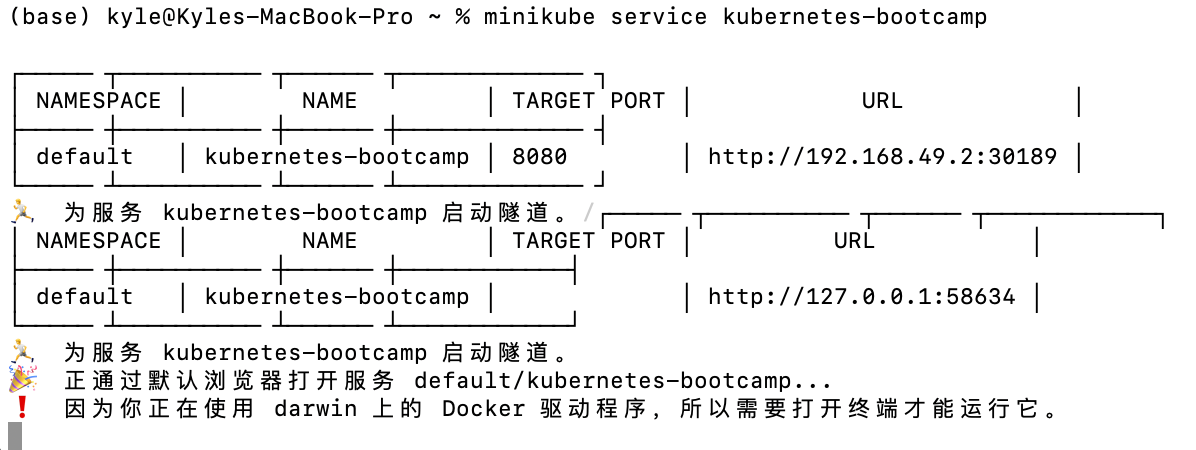
We have now a running Service called kubernetes-bootcamp. Here we see that the Service received a unique cluster-IP, an internal port and an external-IP (the IP of the Node).

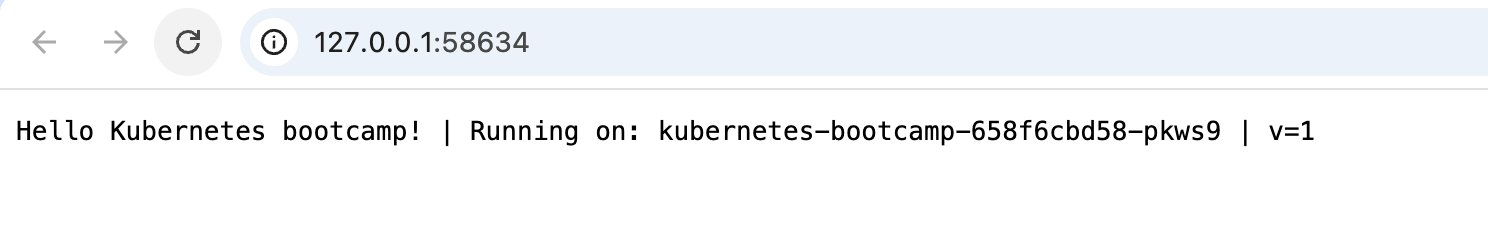


Create an environment variable called NODE\_PORT that has the value of the Node port assigned:

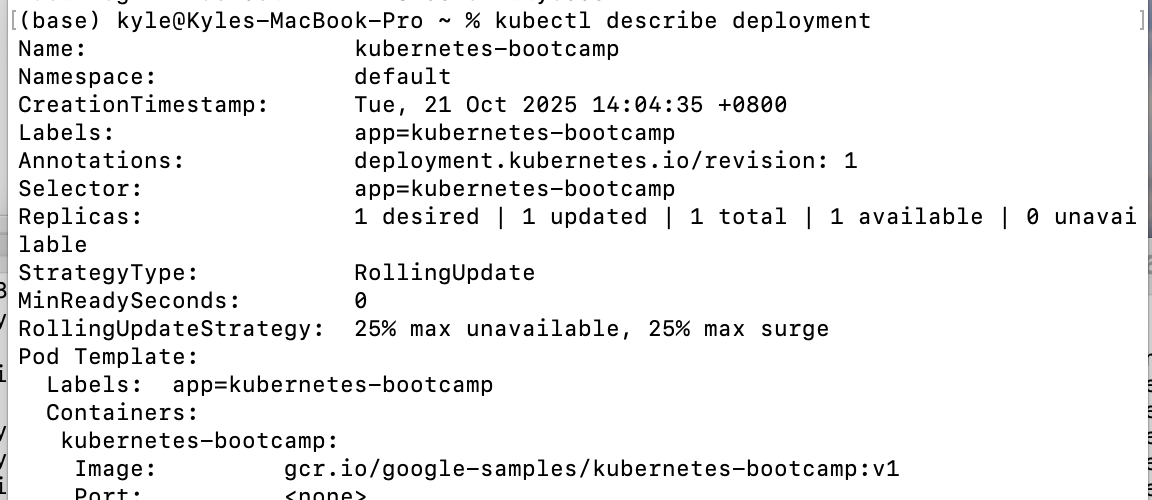


Now we can test that the app is exposed outside of the cluster using curl

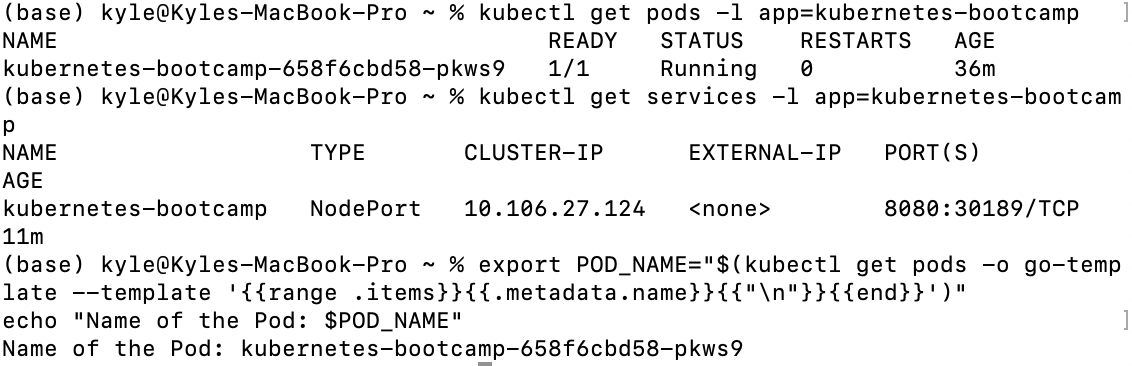




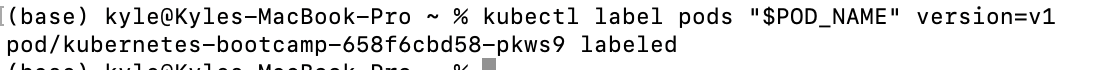
Step 2: Using labels



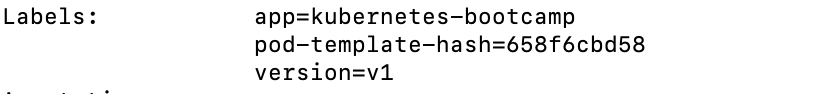
Let’s use this label to query our list of Pods,Services and POD\_NAME



To apply a new label we use the label subcommand followed by the object type, object name and the new label:

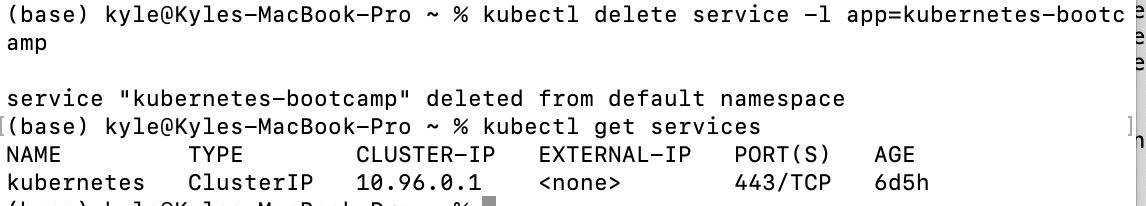


check it with the describe pod



Step 3: Deleting a service

To delete Services you can use the delete service and confirm that the Service is gone

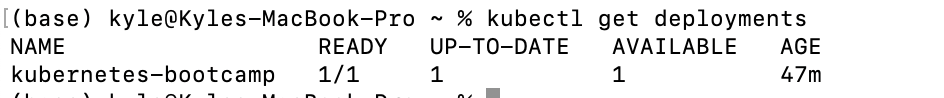


This proves that the application is not reachable anymore from outside of the cluster. You can confirm that the app is still running with a curl from inside the pod

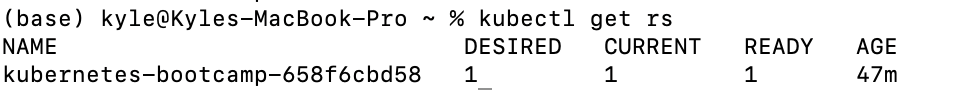


Scaling a Deployment

To list your Deployments, use the get deployments



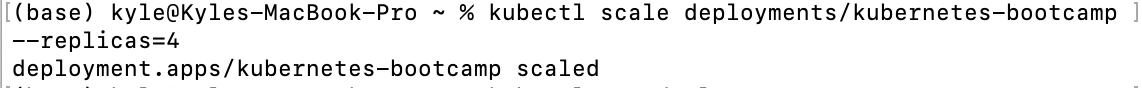
To see the ReplicaSet created



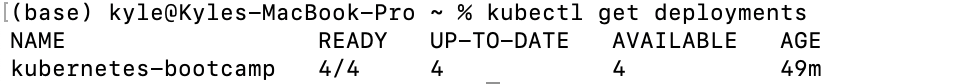
DESIRED displays the desired number of replicas of the application, which you define when you create the Deployment. This is the desired state.

CURRENT displays how many replicas are currently running.

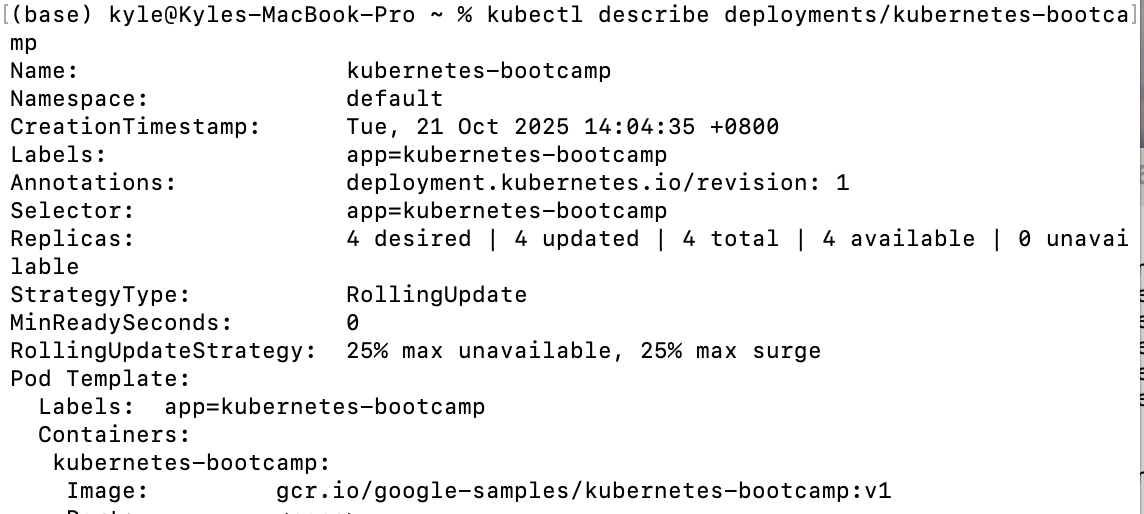
Next, let’s scale the Deployment to 4 replicas.



List Deployments once again,



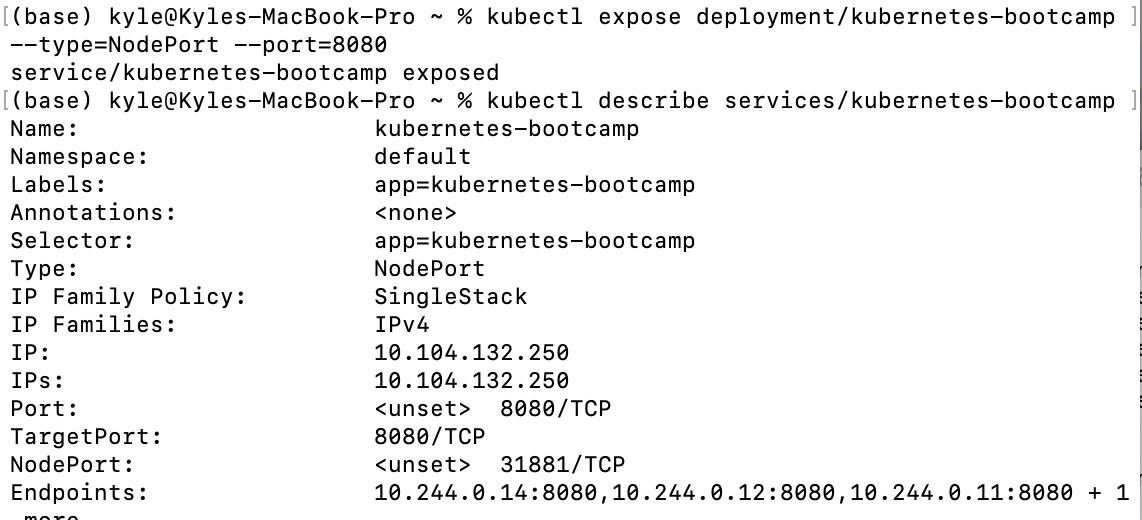
There are 4 Pods now, with different IP addresses. The change was registered in the Deployment events log.



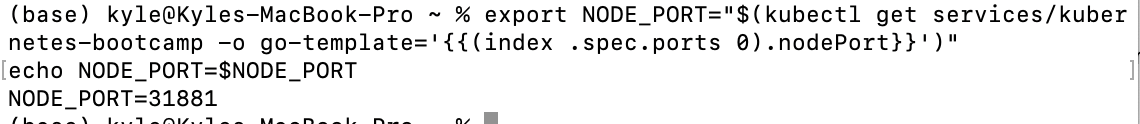
Load Balancing

Let's check that the Service is load-balancing the traffic.

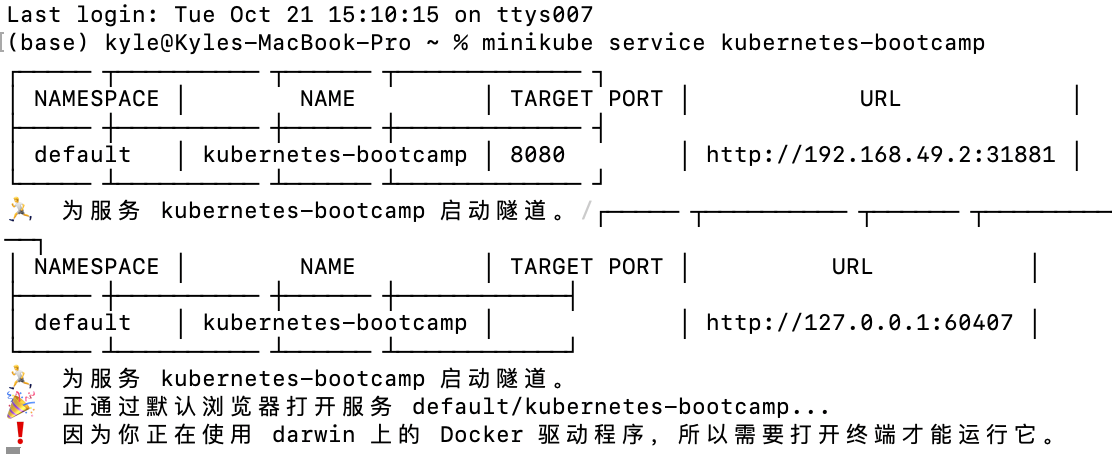
I need to restart service

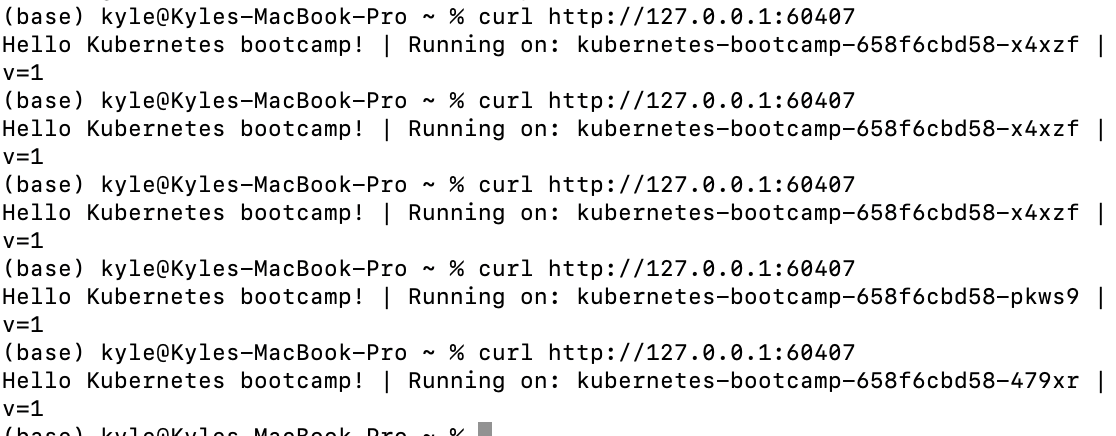


Create an environment variable called NODE\_PORT that has a value as the Node port:

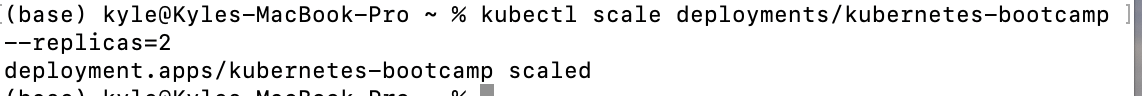


Next, do a curl to the exposed IP address and port.



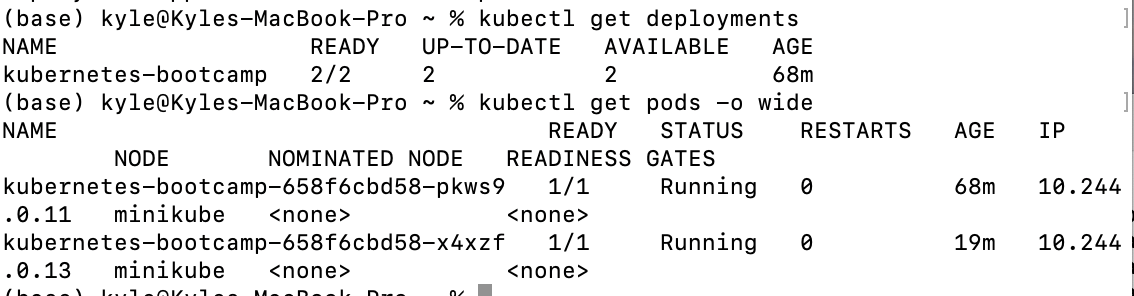


To scale down the Deployment to 2 replicas,



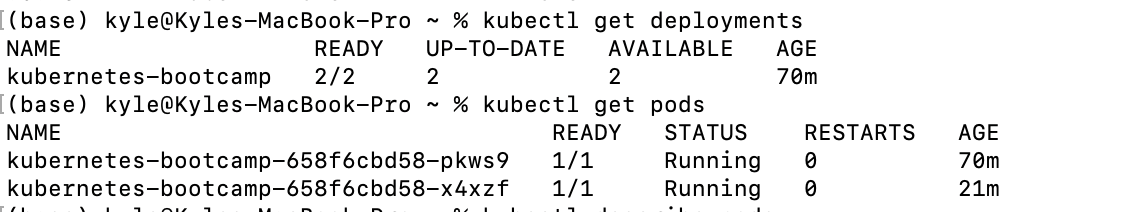
List the Deployments

The number of replicas decreased to 2. List the number of Pods, with get pods:

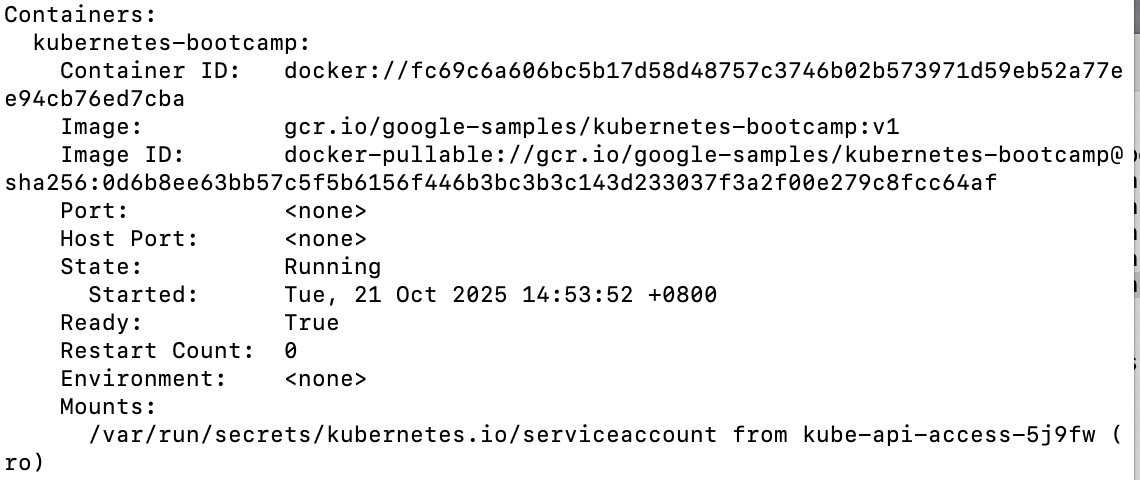


Update the version of the app

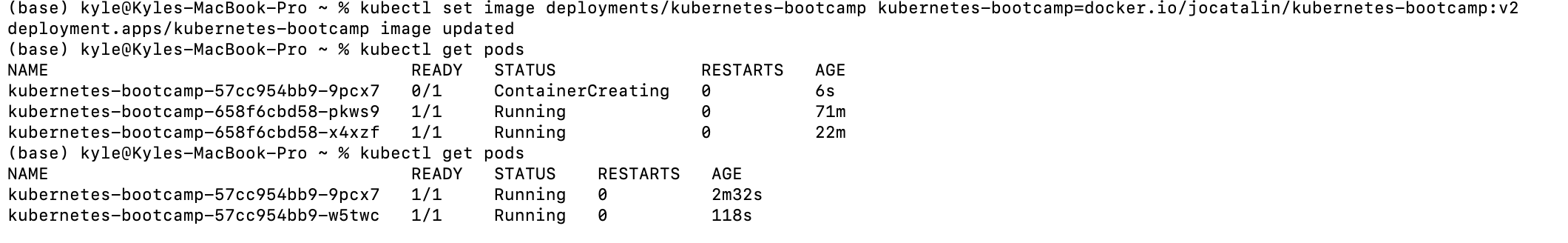
List developments and pods



To view the current image version of the app

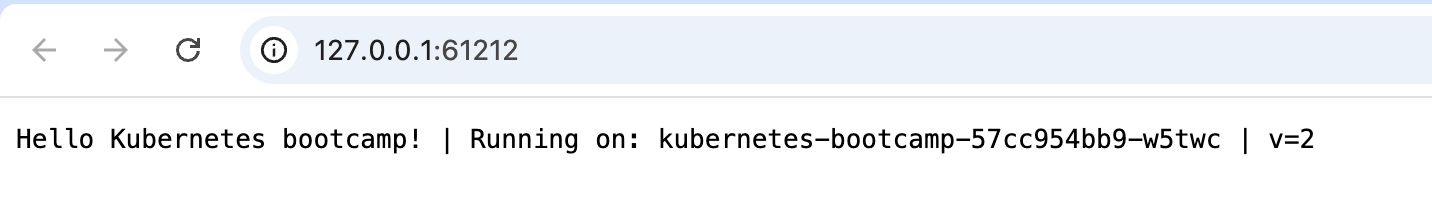


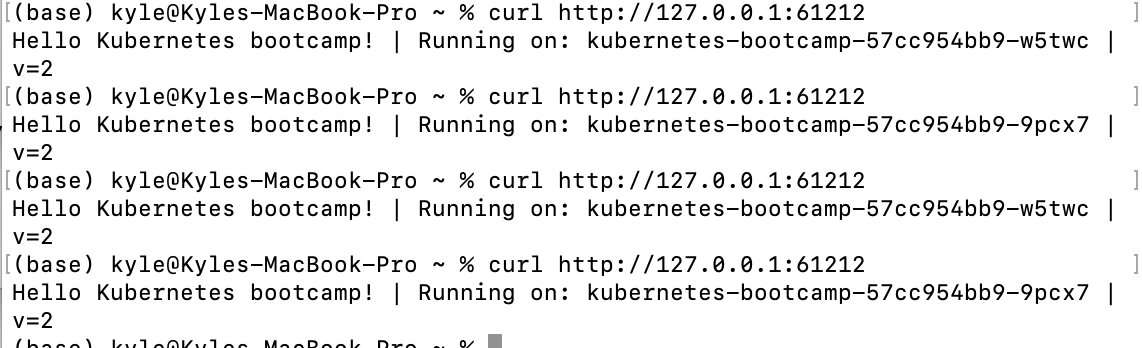
To update the image of the application to version 2



Verify an update

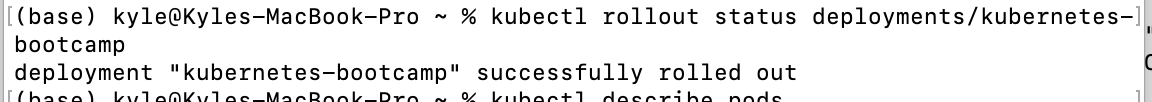




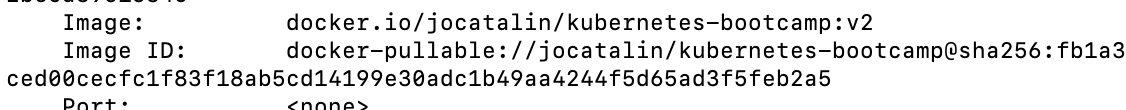


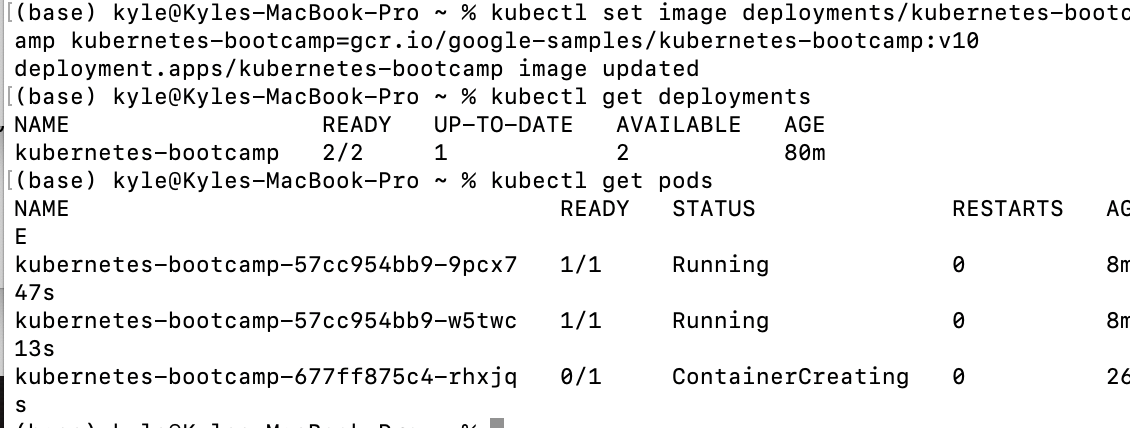
Every time you run the curl command, we will hit a different Pod.

To view the current image version of the app.



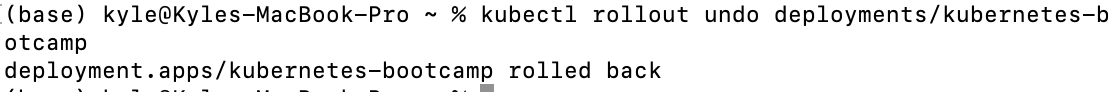
In the Image field of the output, verify that you are running the latest image version (v2).

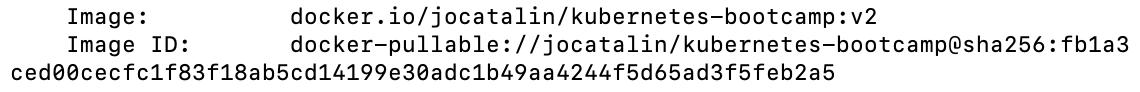




Notice that some of the Pods have a status of ImagePullBackOff

To roll back the deployment to your last working version





clean up the local cluster.

