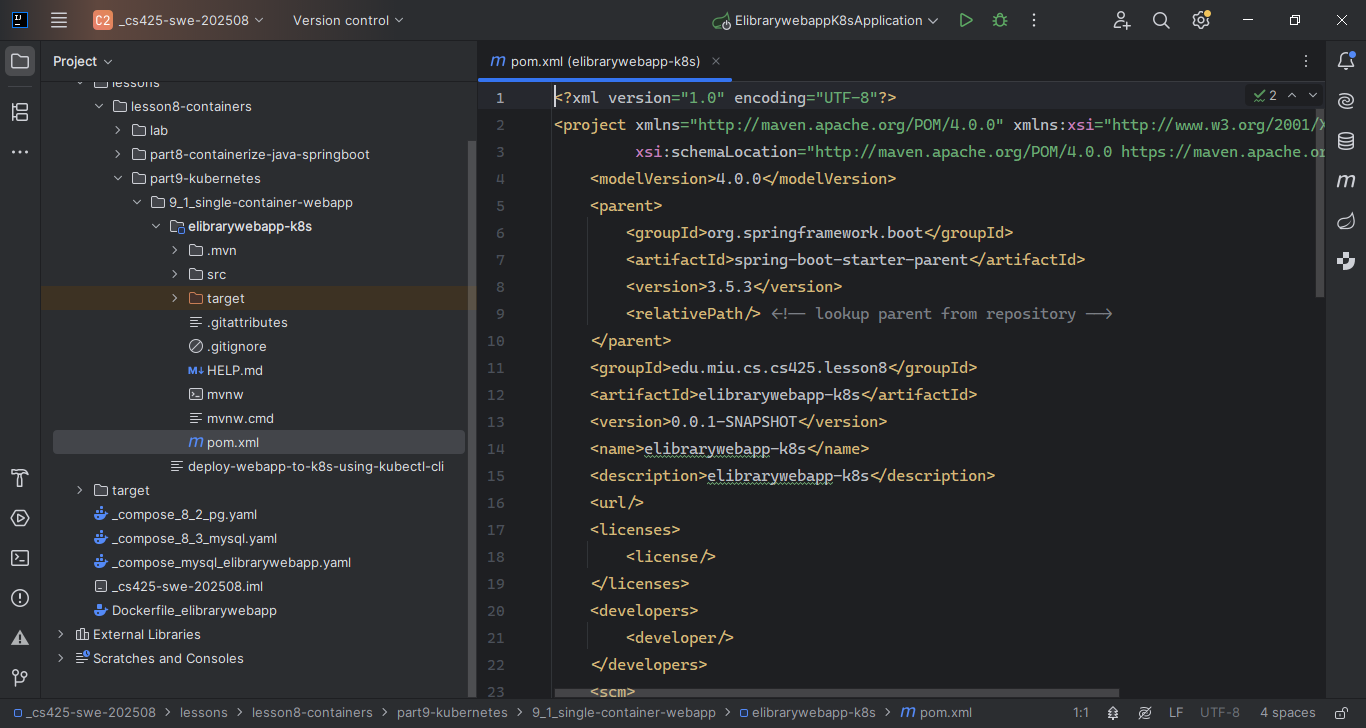
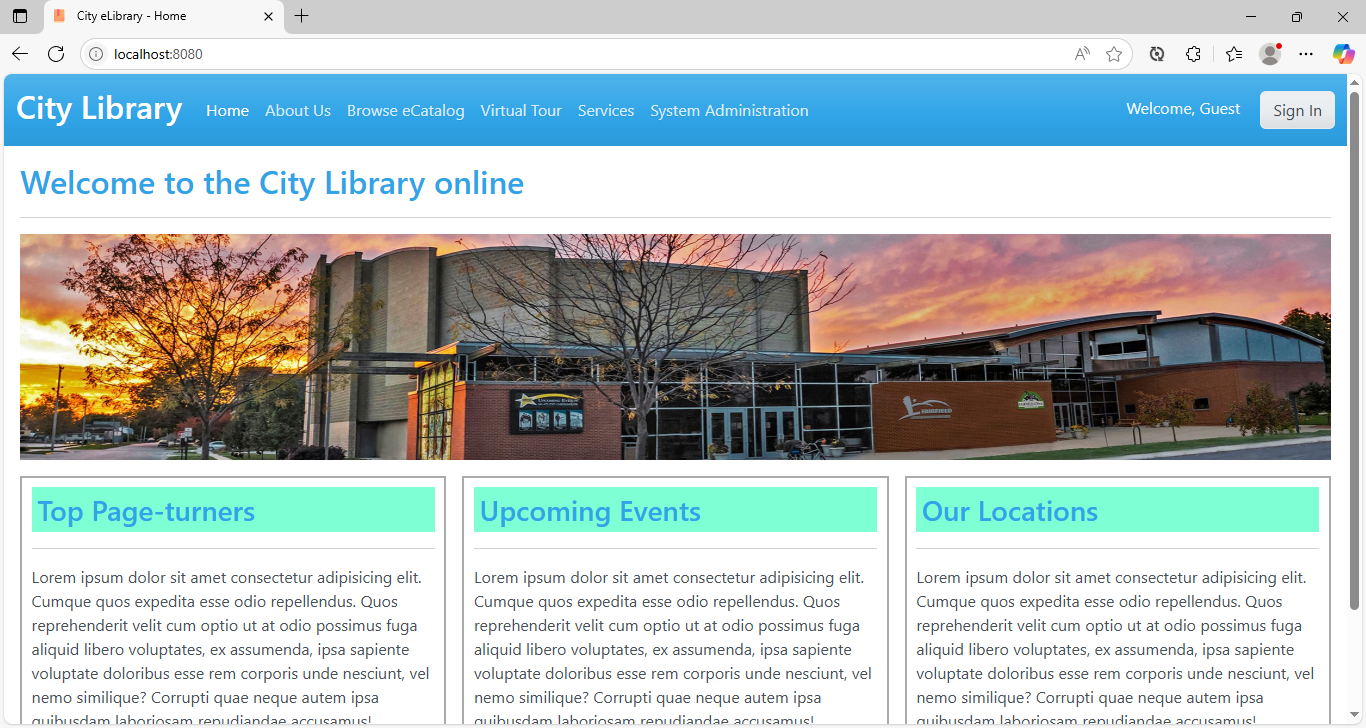
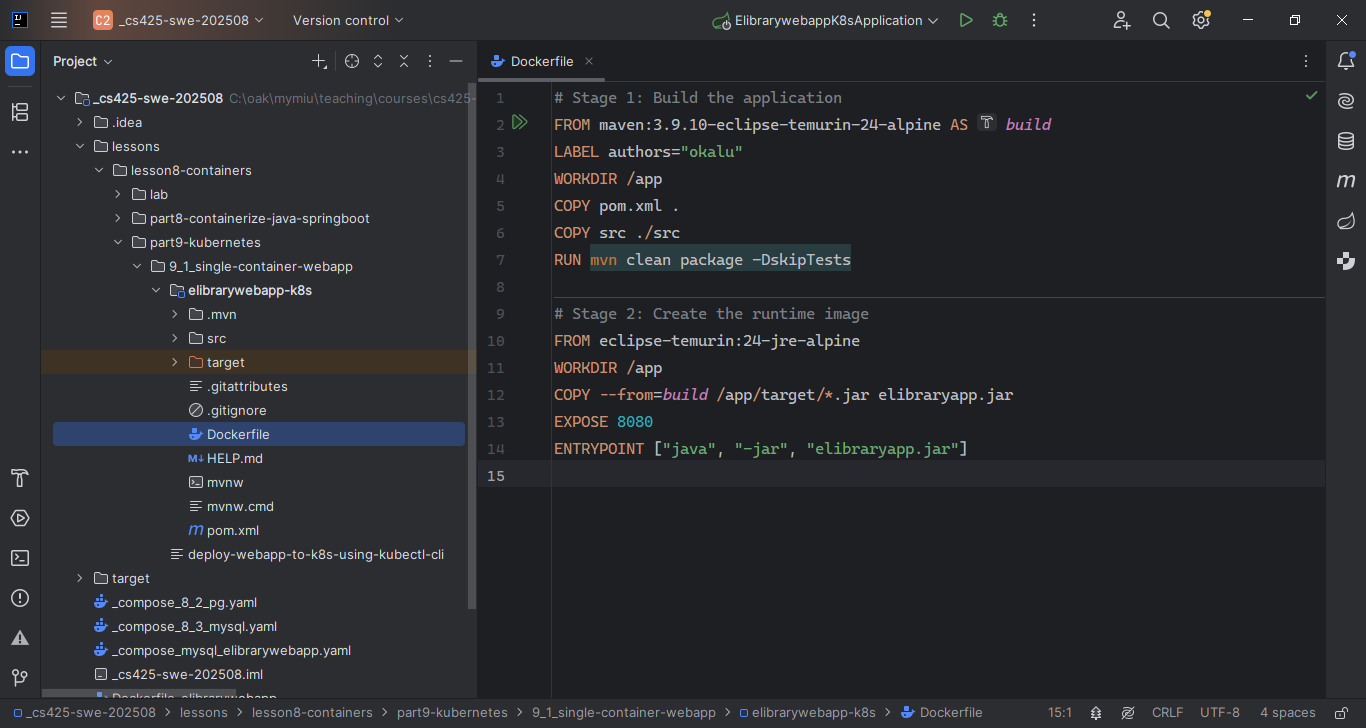
**Build and Deploy a WebApp to Local Kubernetes using Minikube and kubectl CLI**

1. Create the eLibrarywebapp

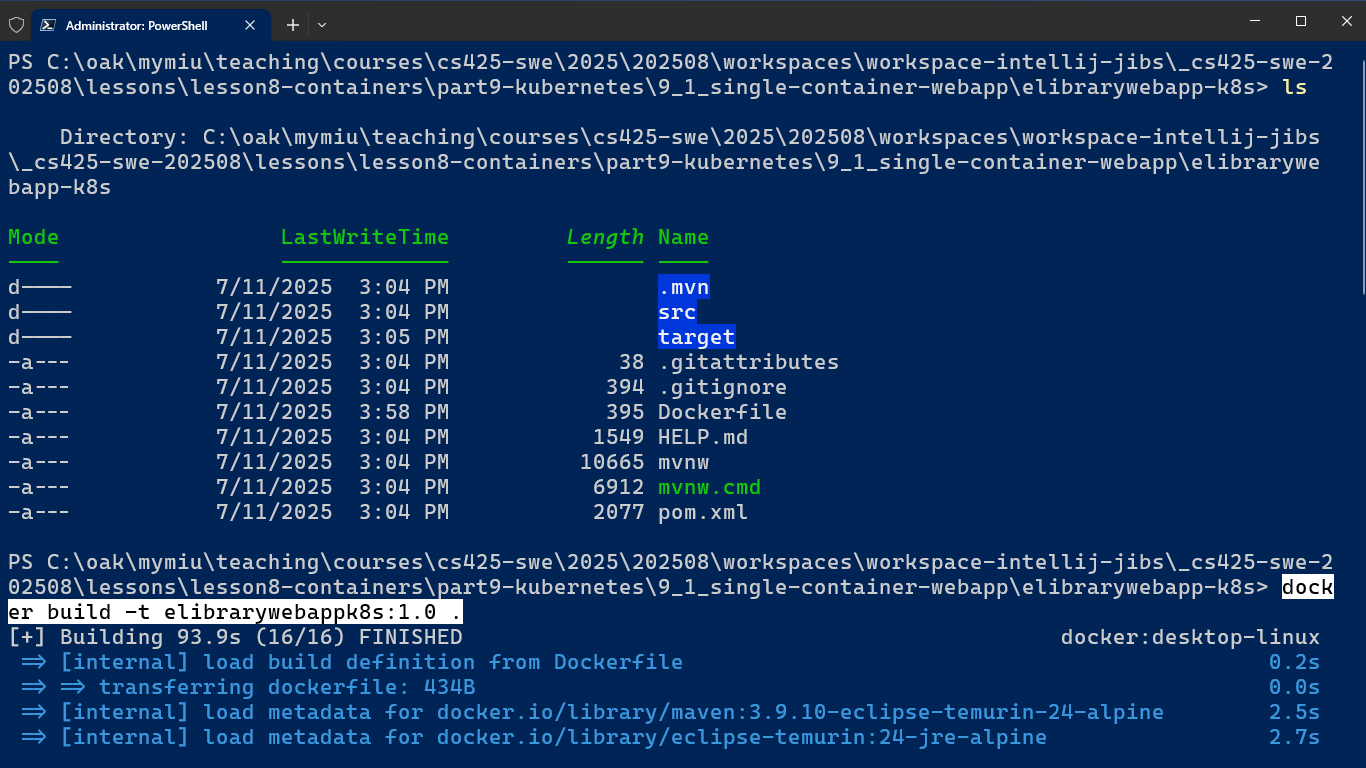


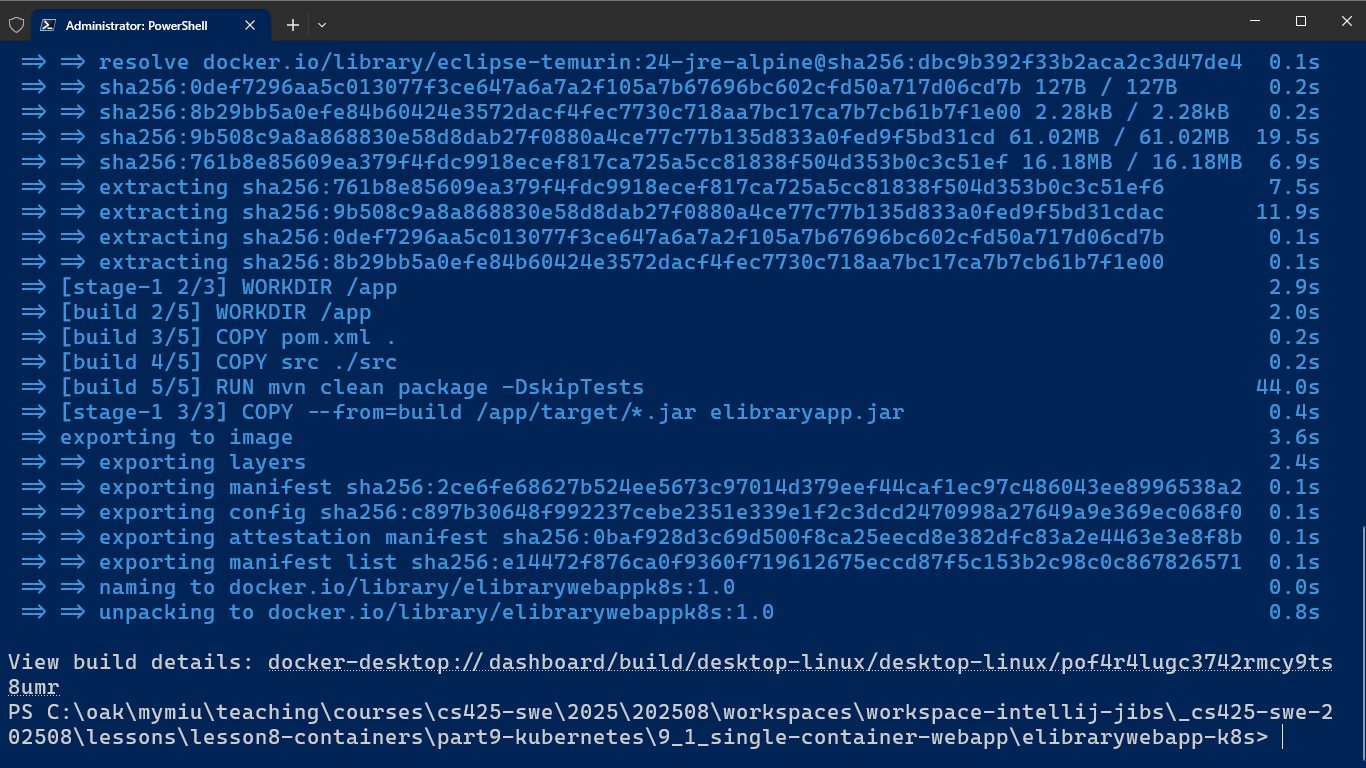


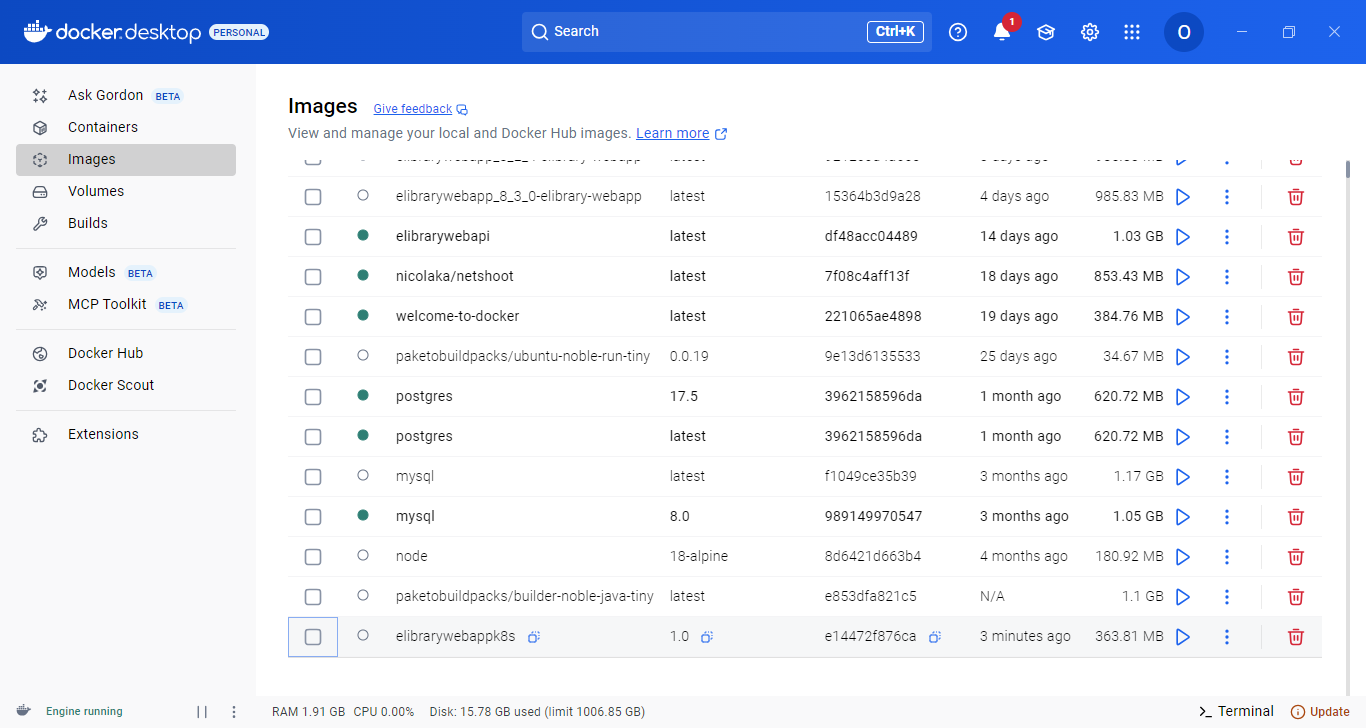
1. Add the Dockerfile (Note: Uses multi-stage build)



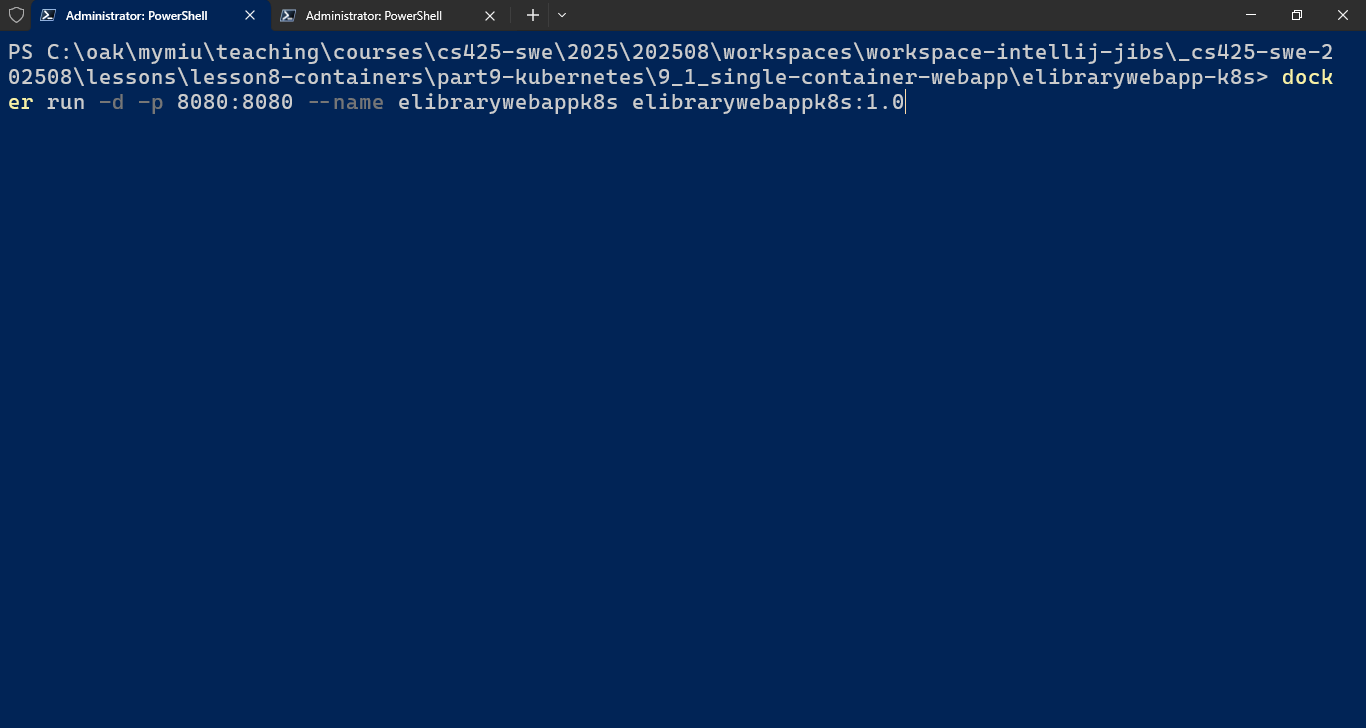
1. Execute “docker build ...” to create the image



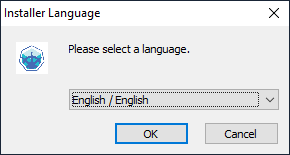


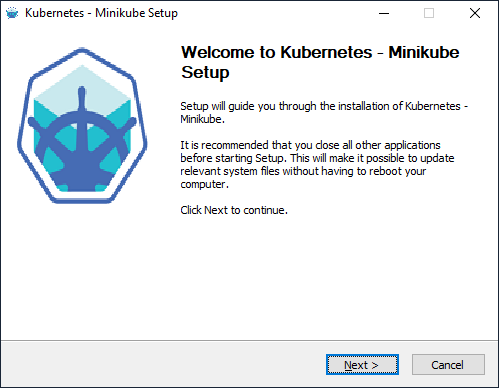


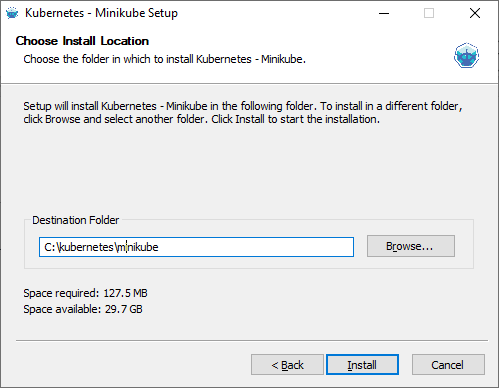
1. Test-run the image by running its docker container, via executing: “docker run ...”



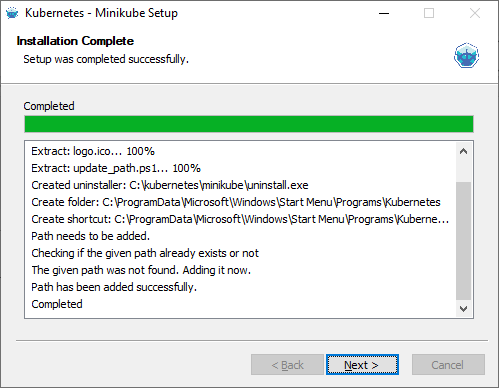
1. View the webapp UI in a browser, just to check it out, that it works/runs ok
2. Now, delete both the container and its image
3. Obtain/download the latest release of Minikube installer for Windows, from: <https://storage.googleapis.com/minikube/releases/latest/minikube-installer.exe> and execute it to install minikube

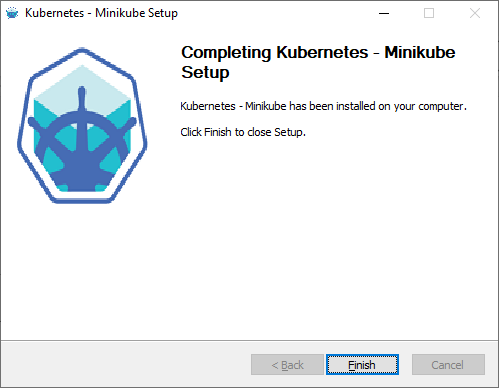




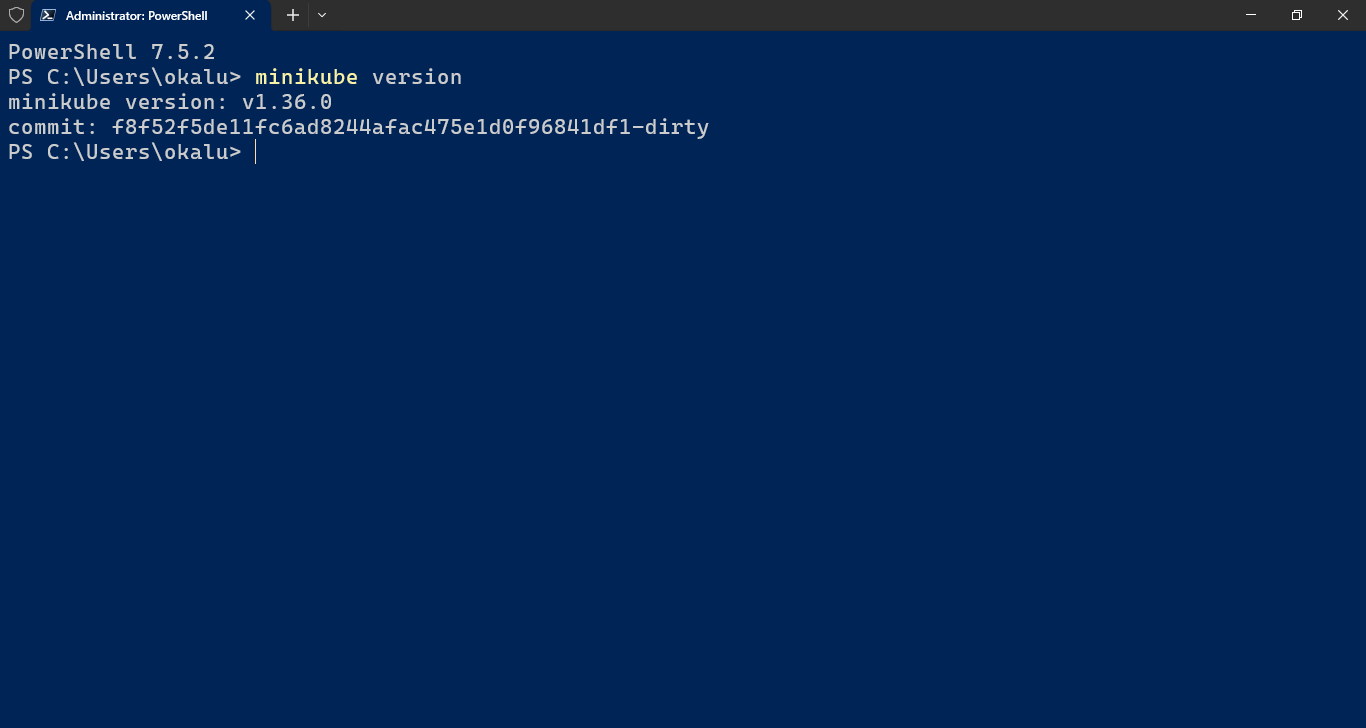






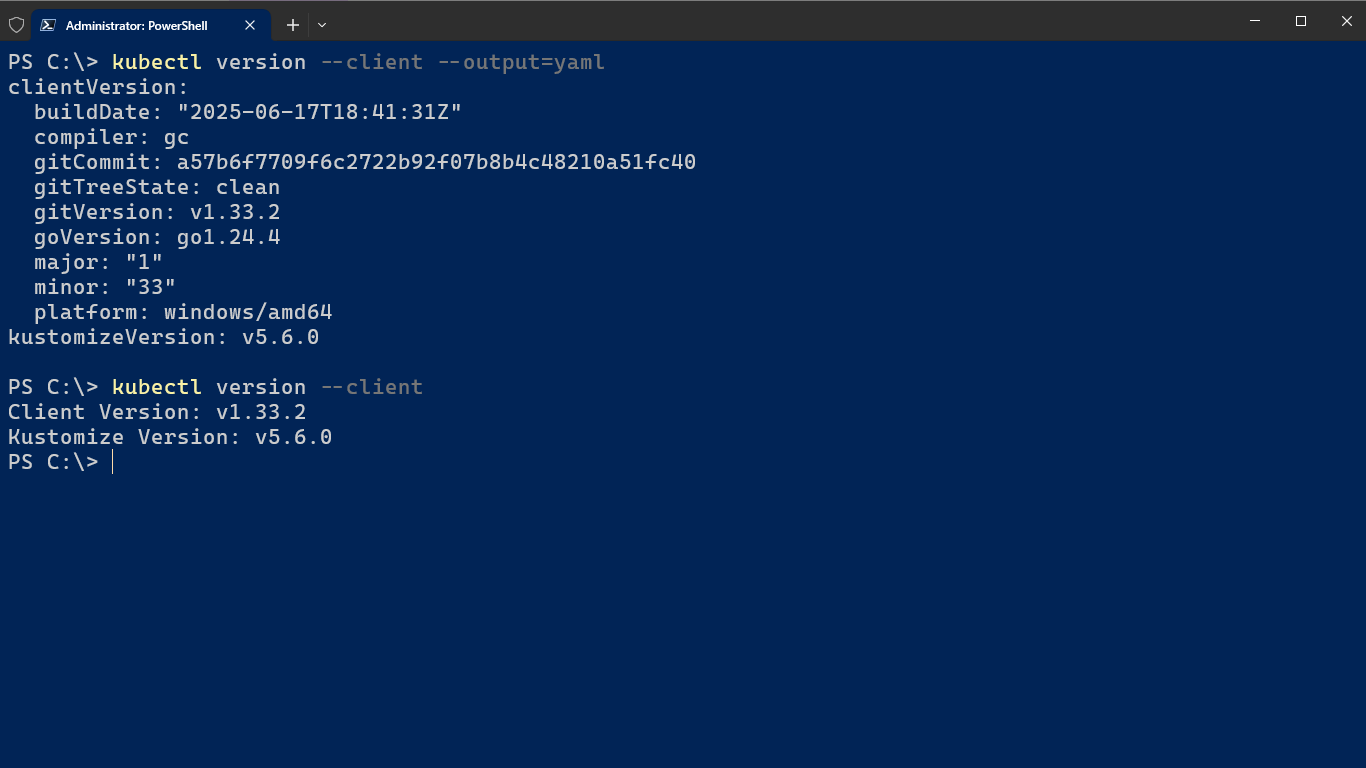


1. Execute the cmd, “minikube version”, to see the version information.



1. Obtain/download and setup Kubectl.exe and verify its version.

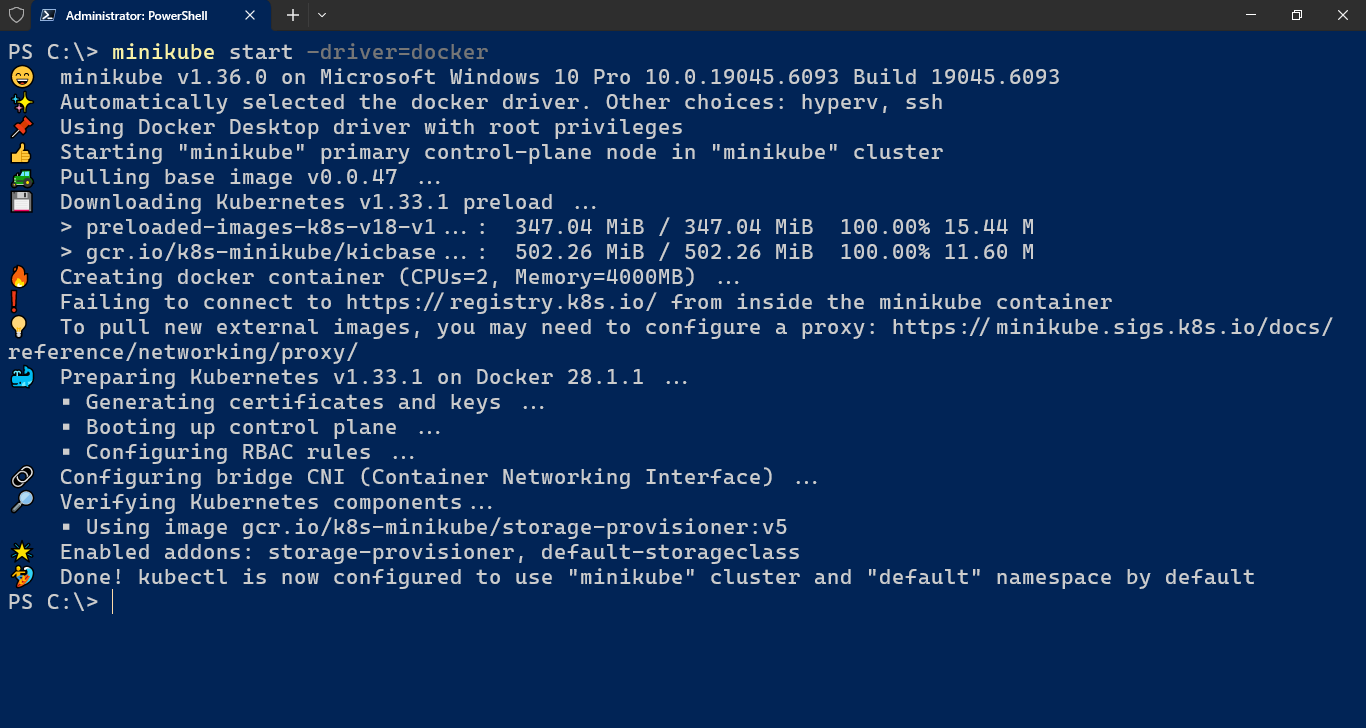
<https://kubernetes.io/releases/download/#binaries>

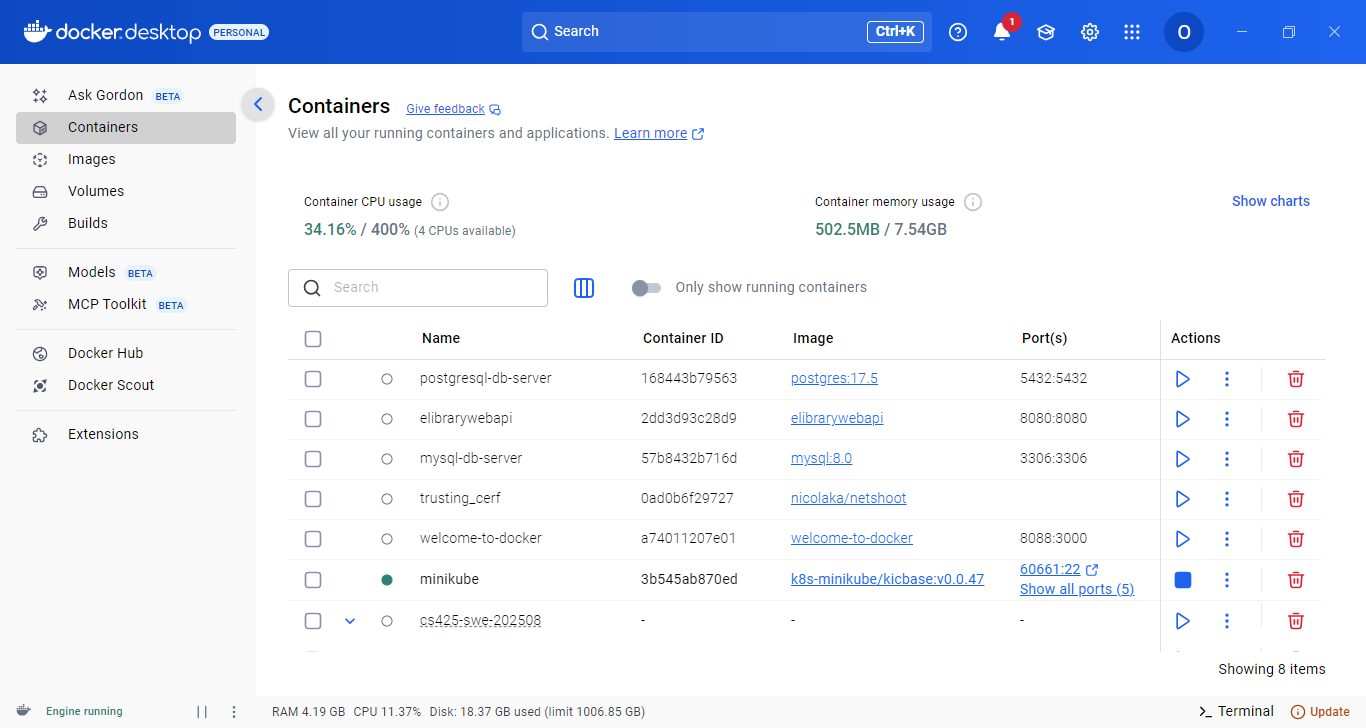


Note: Also run, minikube help, to see the minikube commands. Run > minikube [cmd] --help to see options for a given cmd. Use "minikube options" for a list of global command-line options (this applies to all commands).

1. Make sure to add both minikube.exe and kubectl.exe to the PATH environment variable.
2. Start minikube - > *minikube start -driver=docker*

(Note: This could take a few minutes... so patience)

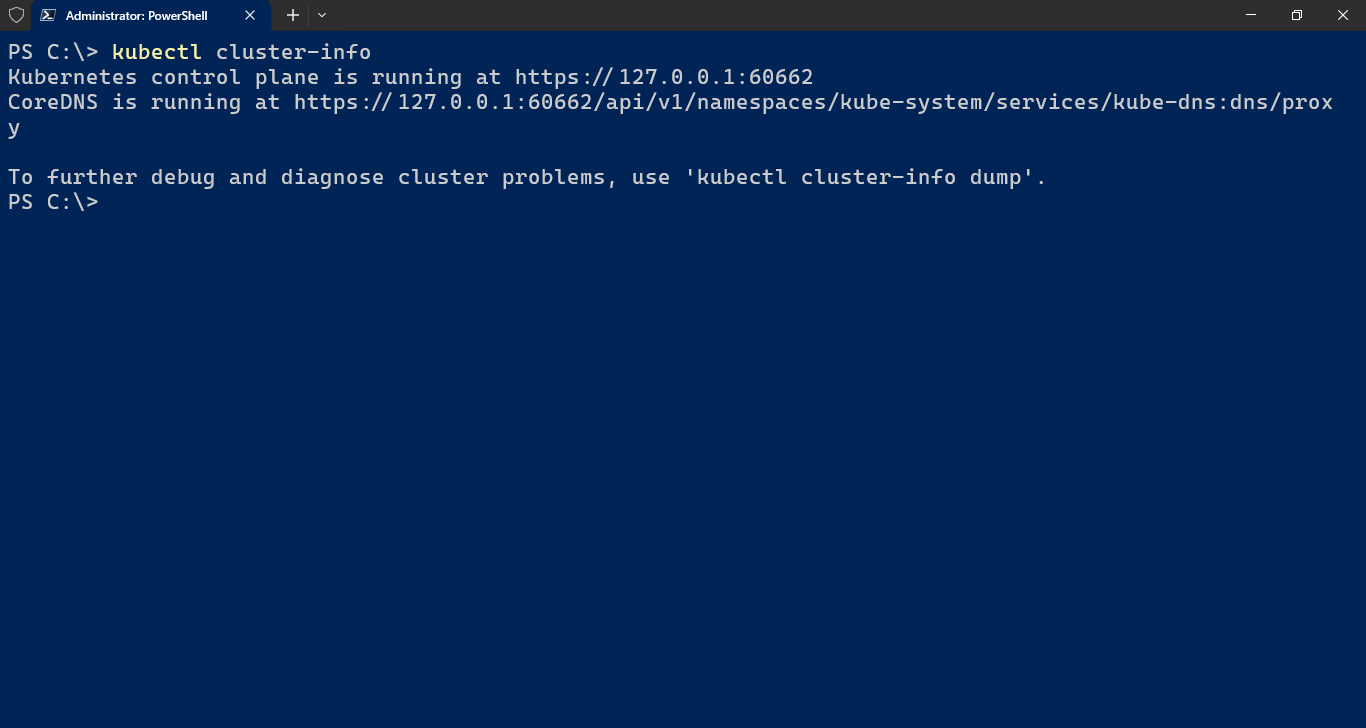


Minikube runs as a container in docker.

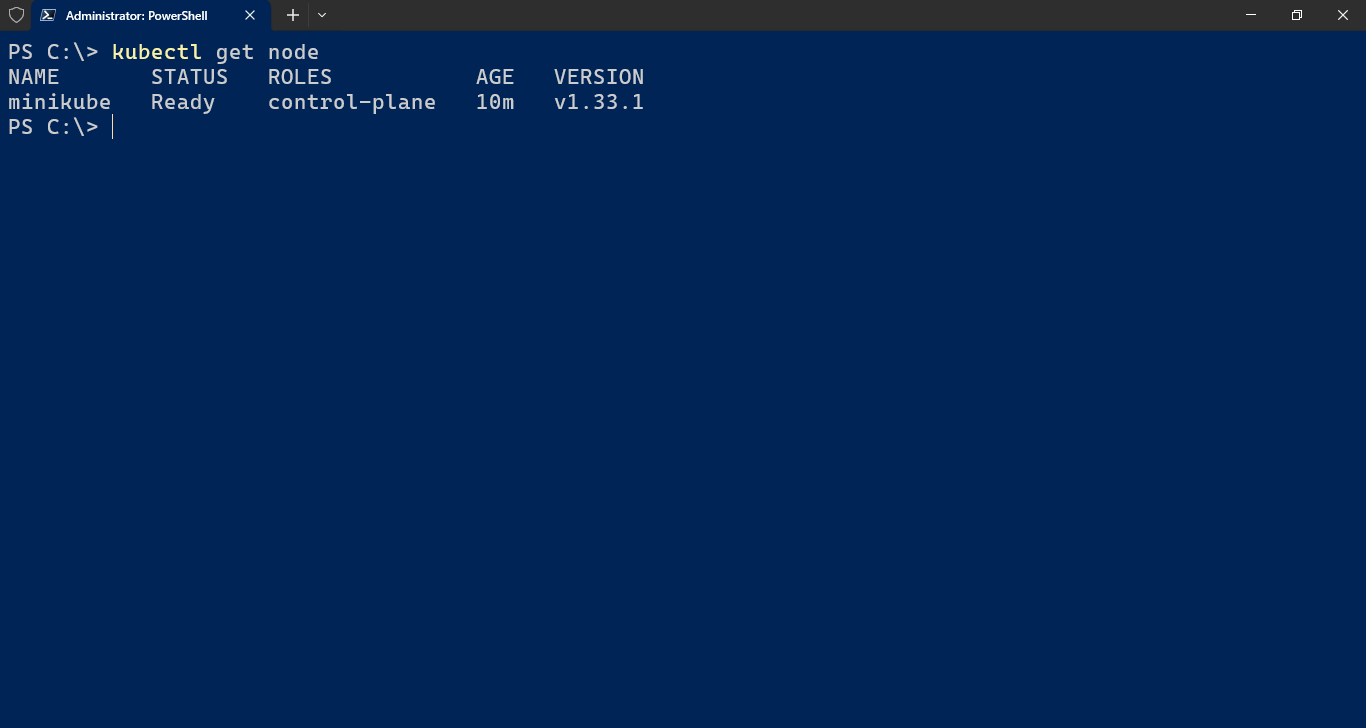
1. Verify the Minikube run status -> *minikube status*



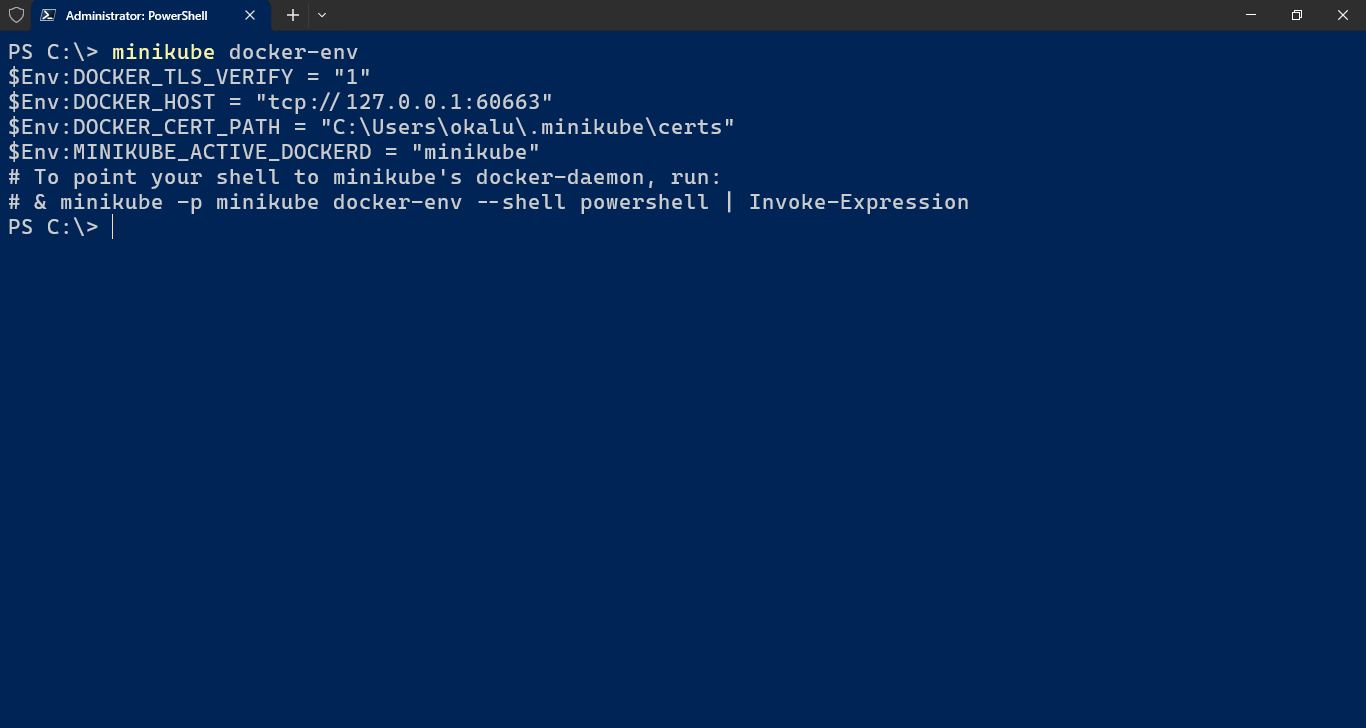
1. Verify that minikube has created a kubernetes cluster, by checking the cluster information using kubectl -> *kubectl cluster-info*

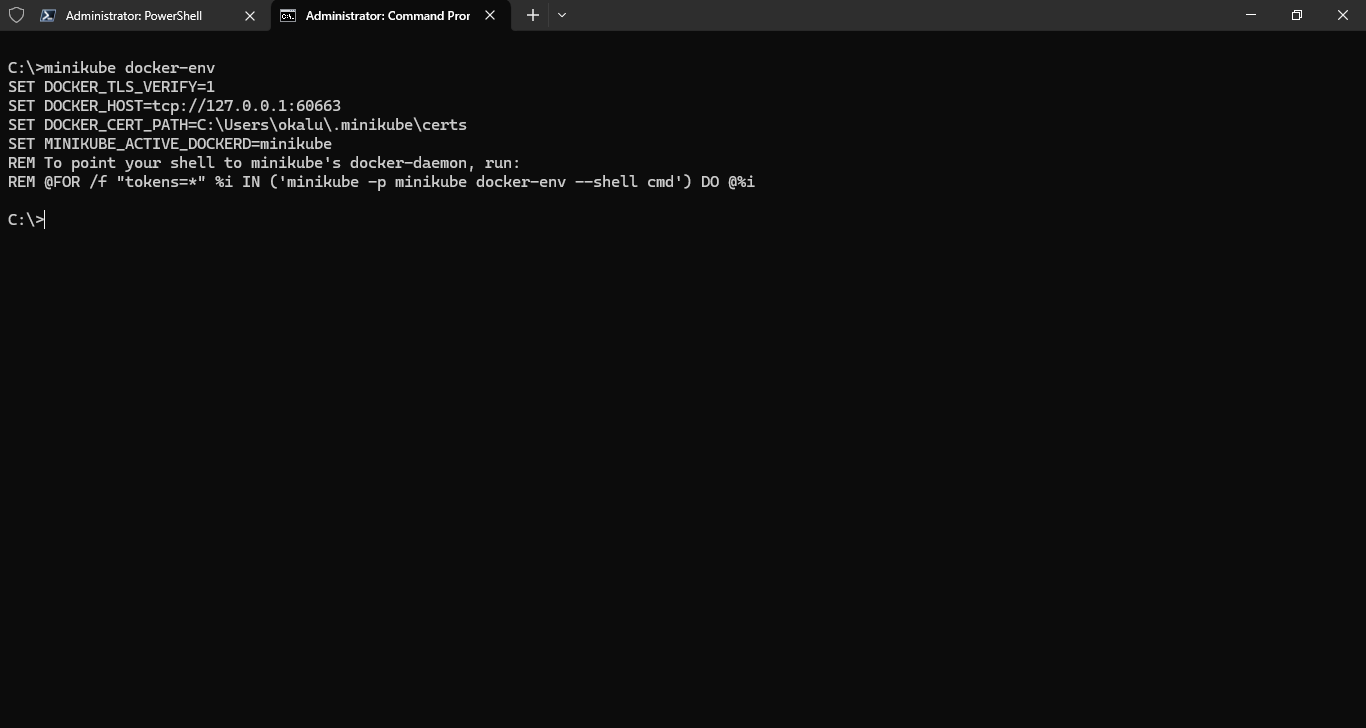


1. Check that a single Node was created on the cluster -> *kubectl get node*



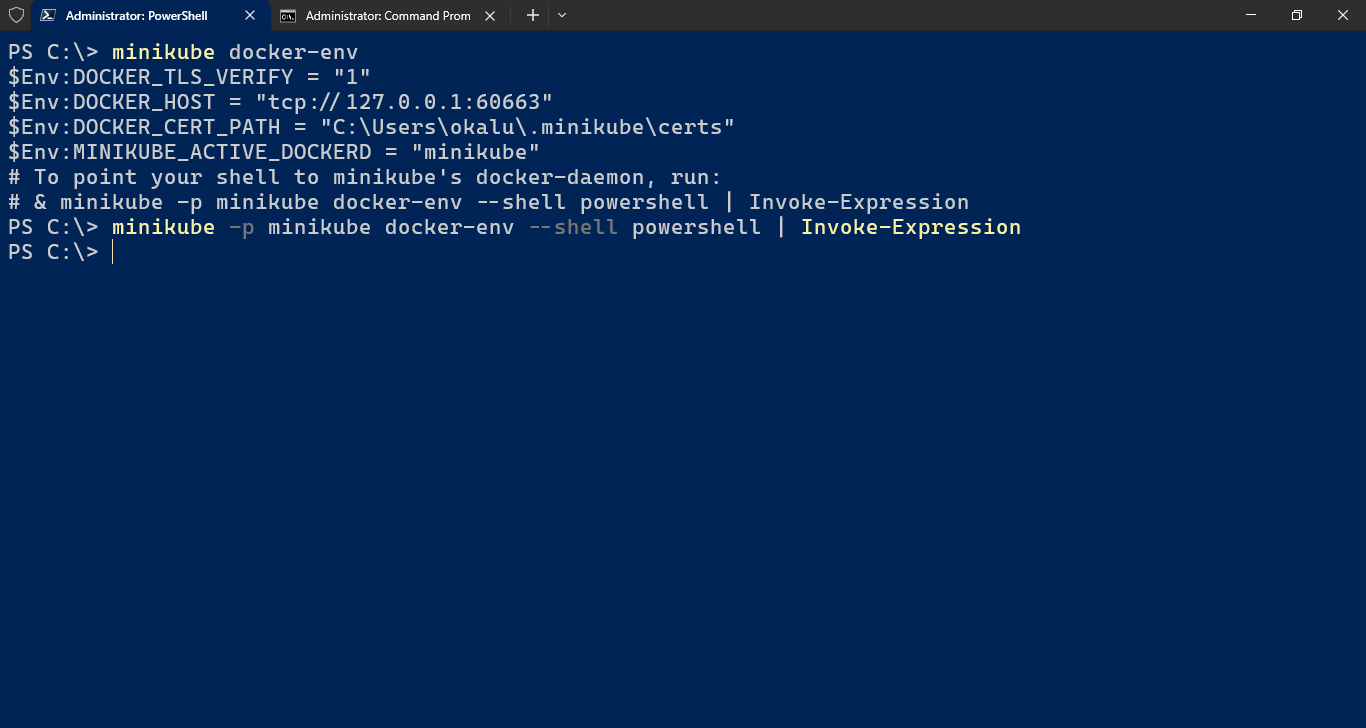
1. In order to enable deployment of containerized applications to kubernetes cluster hosted in minikube, minikube needs to be given access to operate the containers deployed to the docker environment. To do that, perform the following:
   1. Execute > *minikube docker-env*

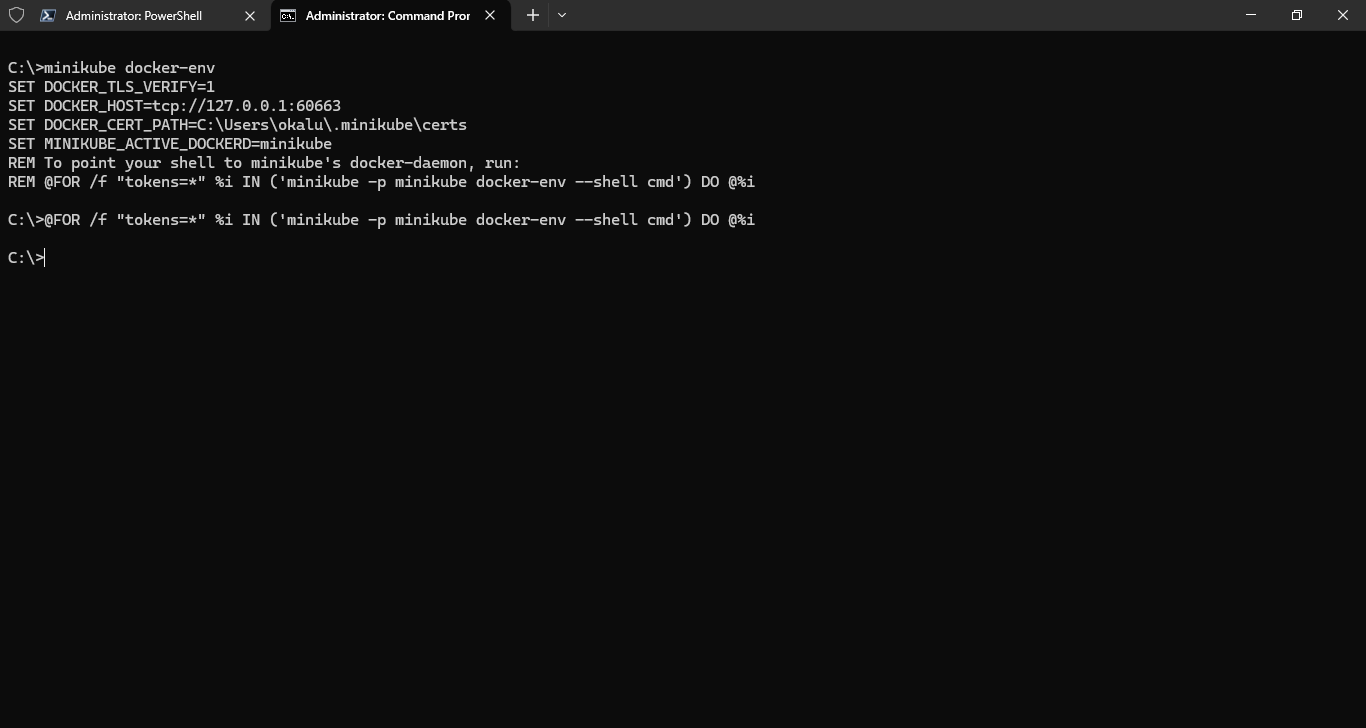
Or, if using the cmd shell



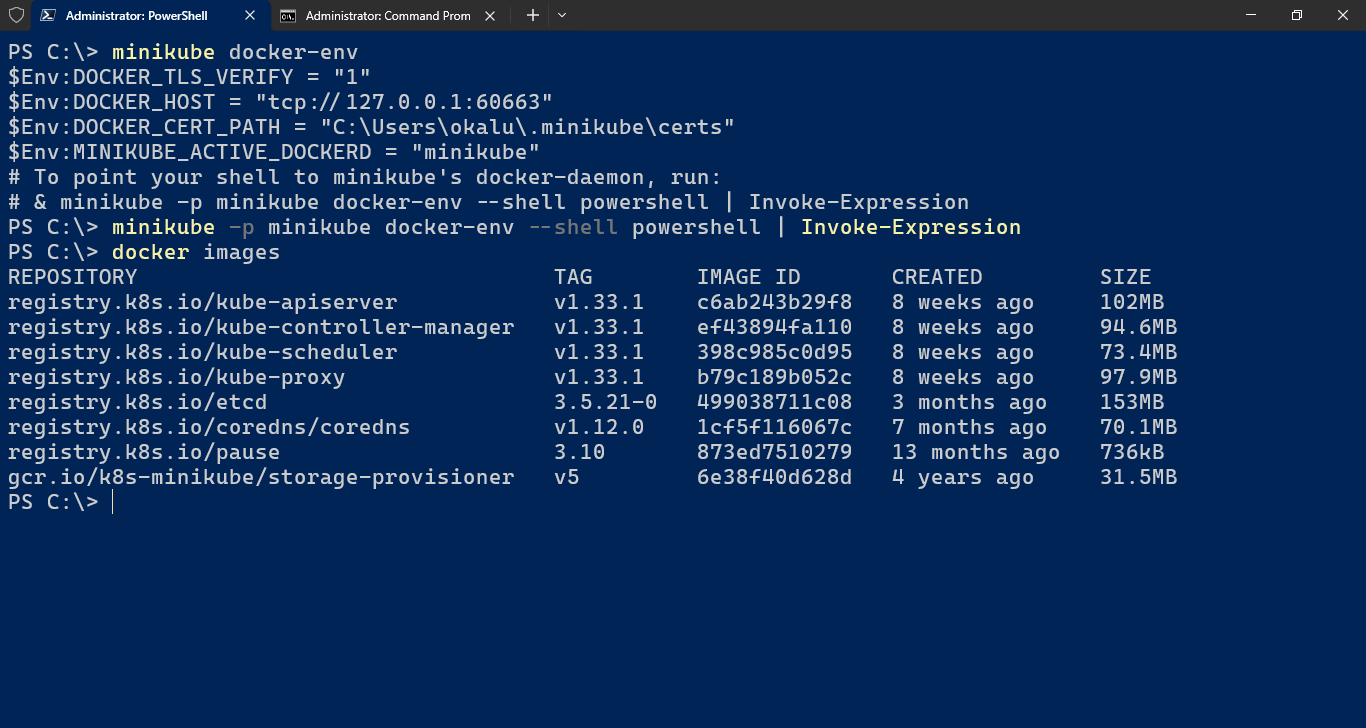
1. To point the shell to minikube’s docker-daemon, execute the following script:

*minikube -p minikube docker-env --shell powershell | Invoke-Expression*

Or, if using cmd.exe shell

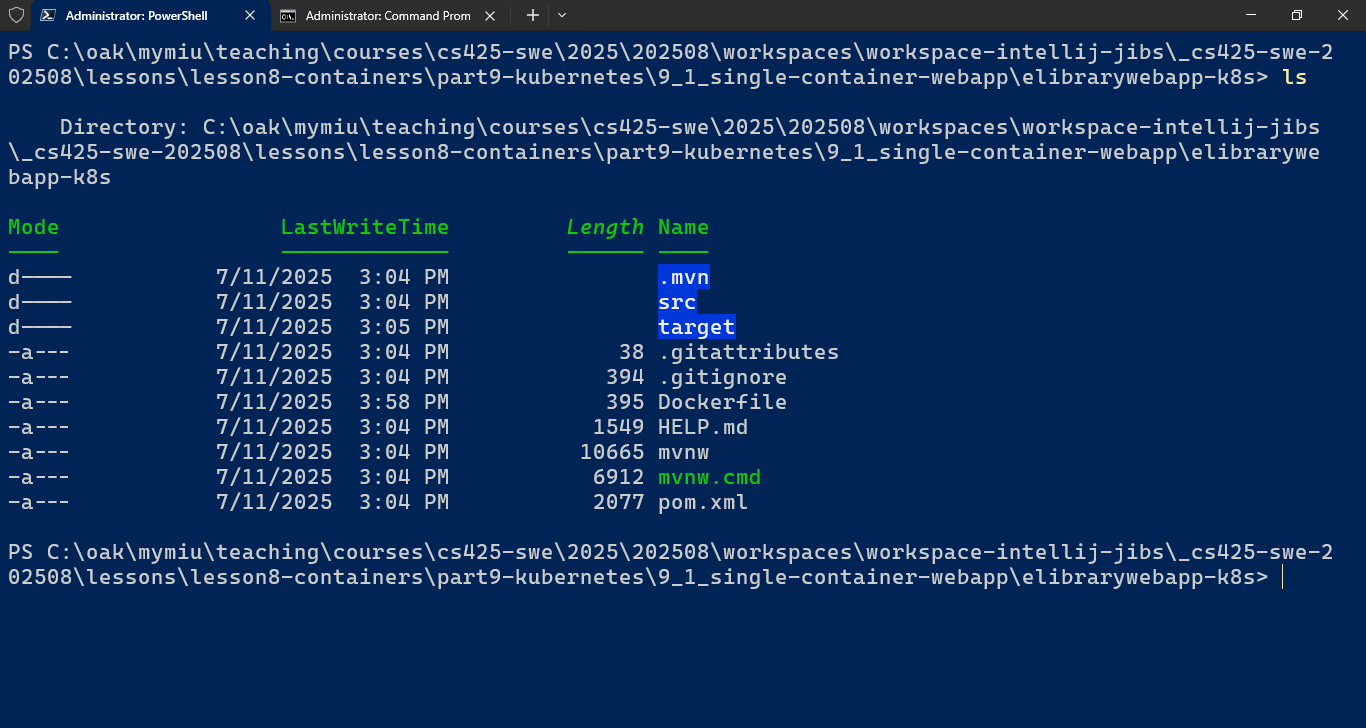


1. > docker images

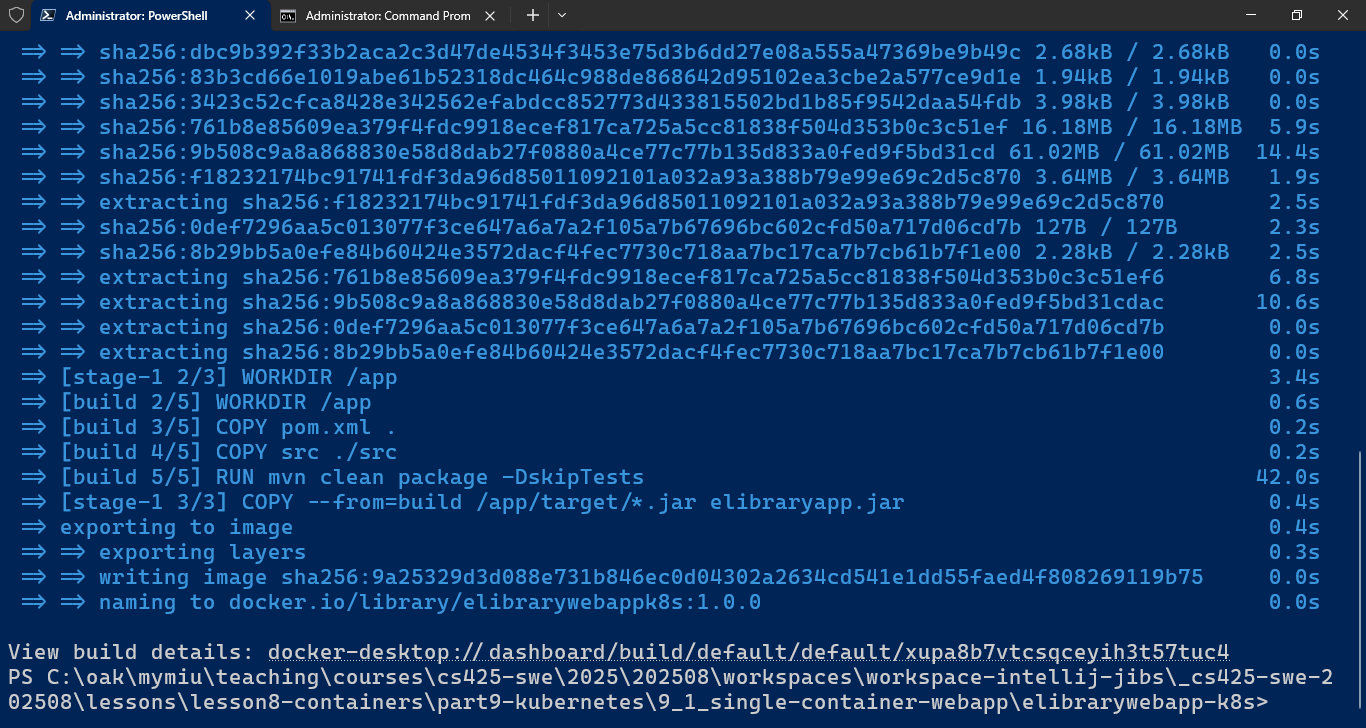
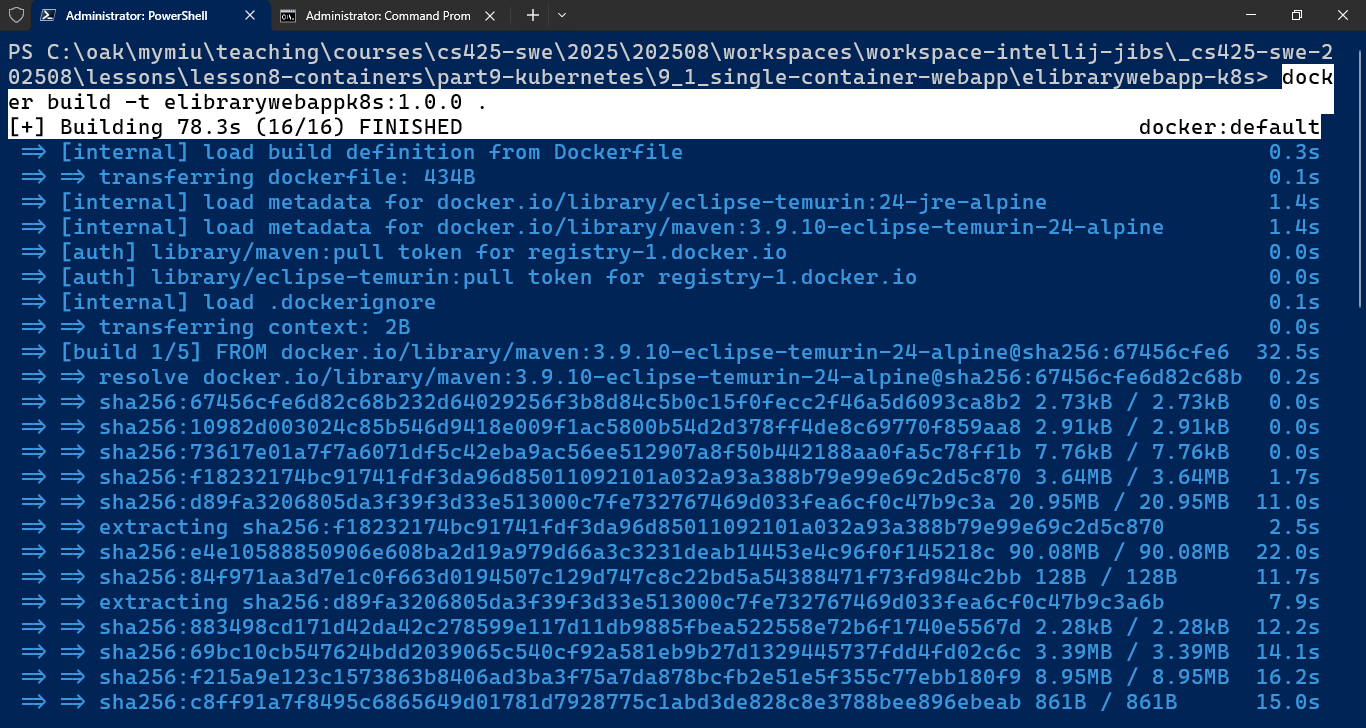


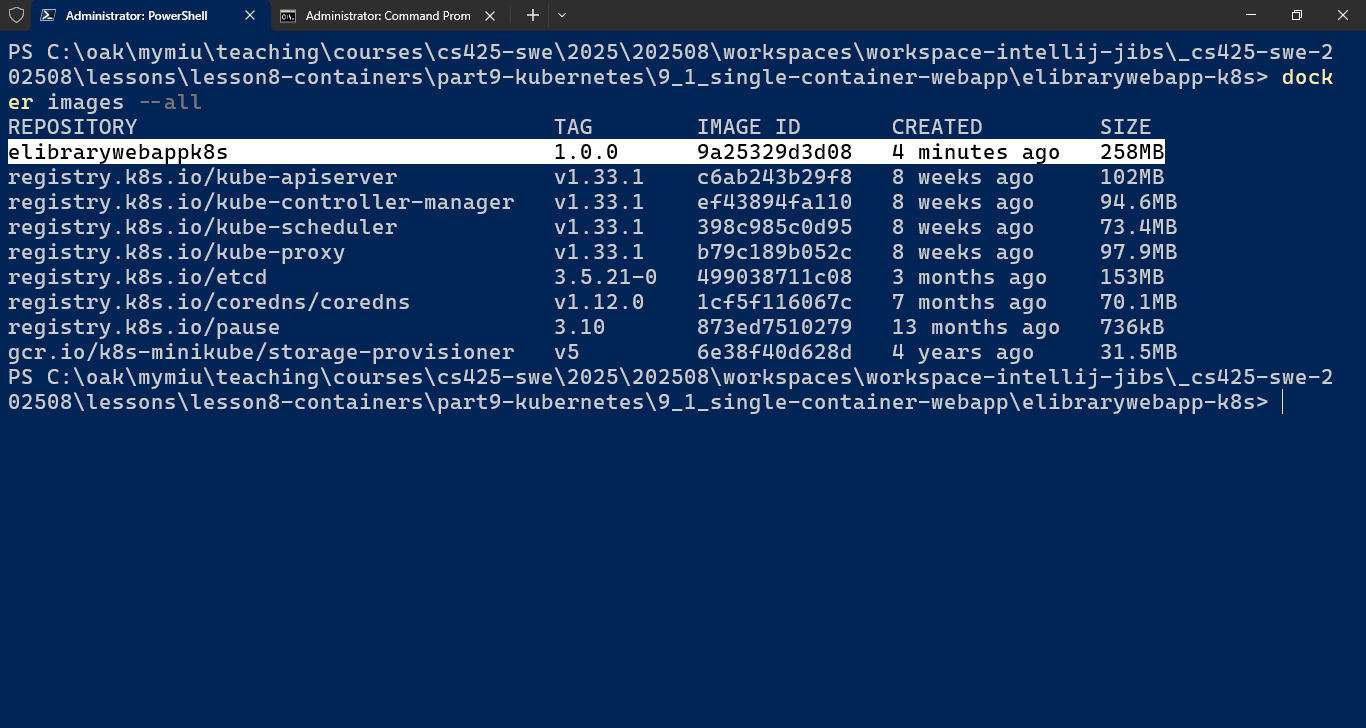
Our local single-node Kubernetes cluster is setup and running, ready for deployments!

1. Next, we proceed with deployment of the containerized eLibraryWebApp.
2. CD to the app’s root directory

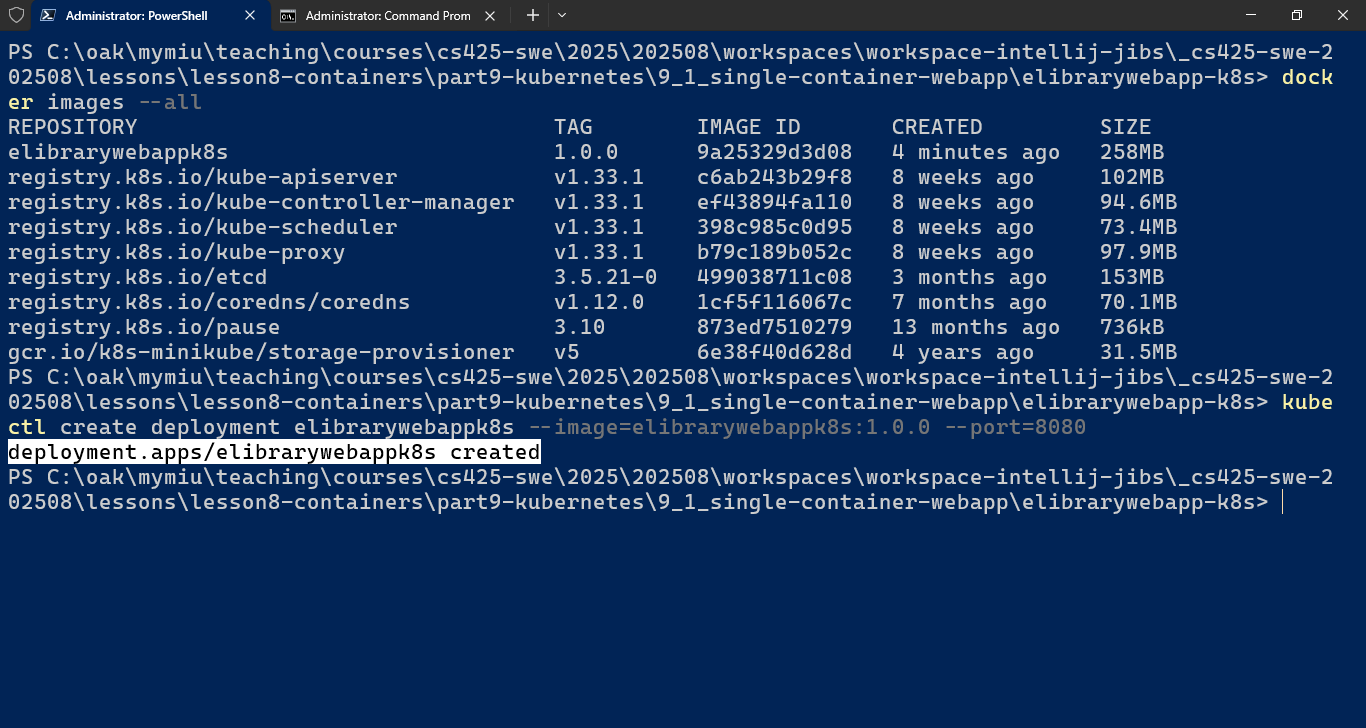


1. Execute > docker build –t elibrarywebappk8s:1.0.0

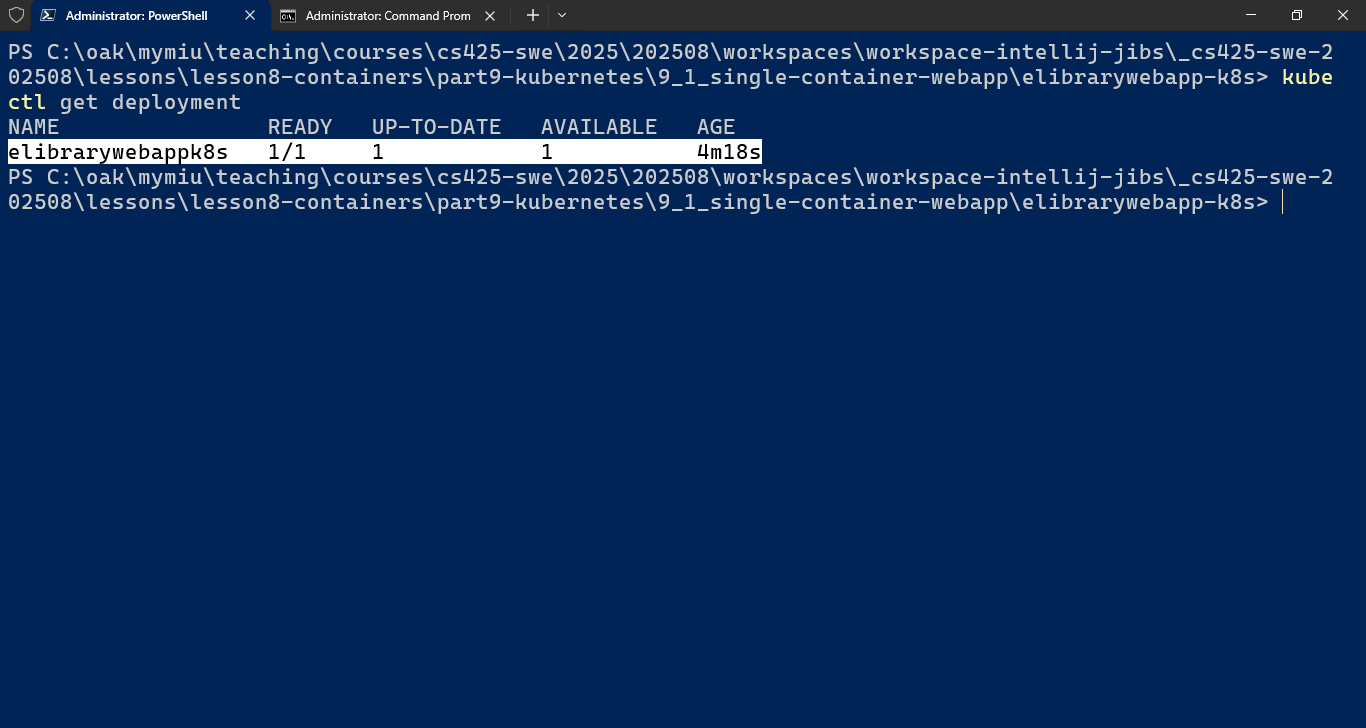




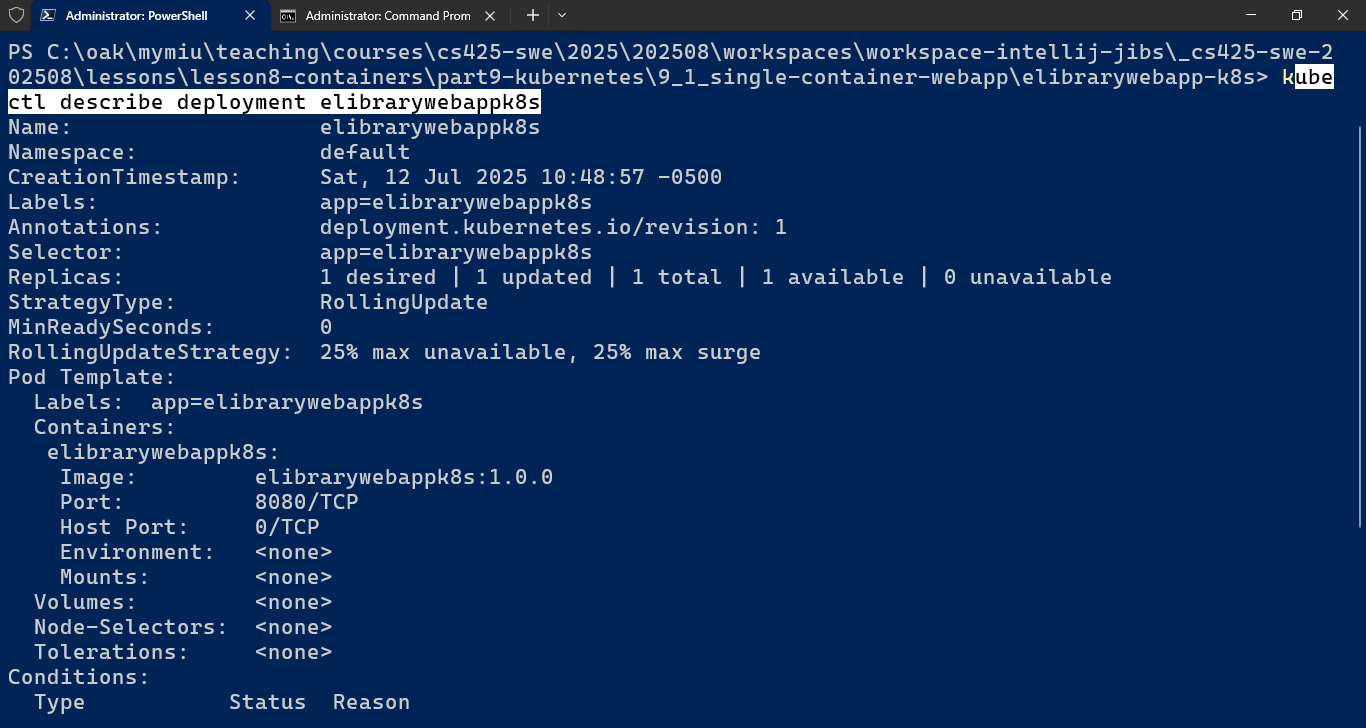
1. Next, we deploy the app. Two options are:
   1. Declaratively by Creating and executing/applying a deployment.yaml file (TBD in next demo)
   2. Interactively via CLI by executing the kubectl cmd: > *kubectl create deployment elibrarywebappk8s --image=elibrarywebappk8s:1.0.0 --port=8080*



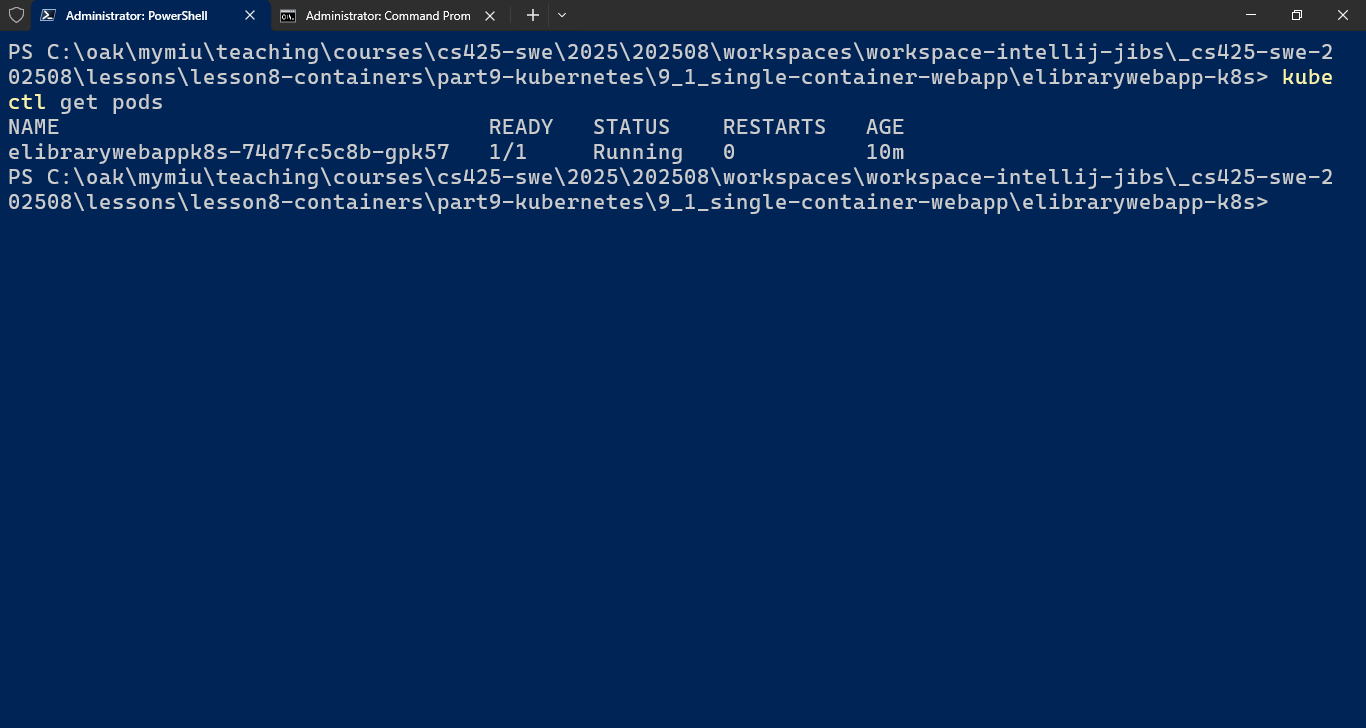
1. Verify that the deployment was created. > kubectl get deployments



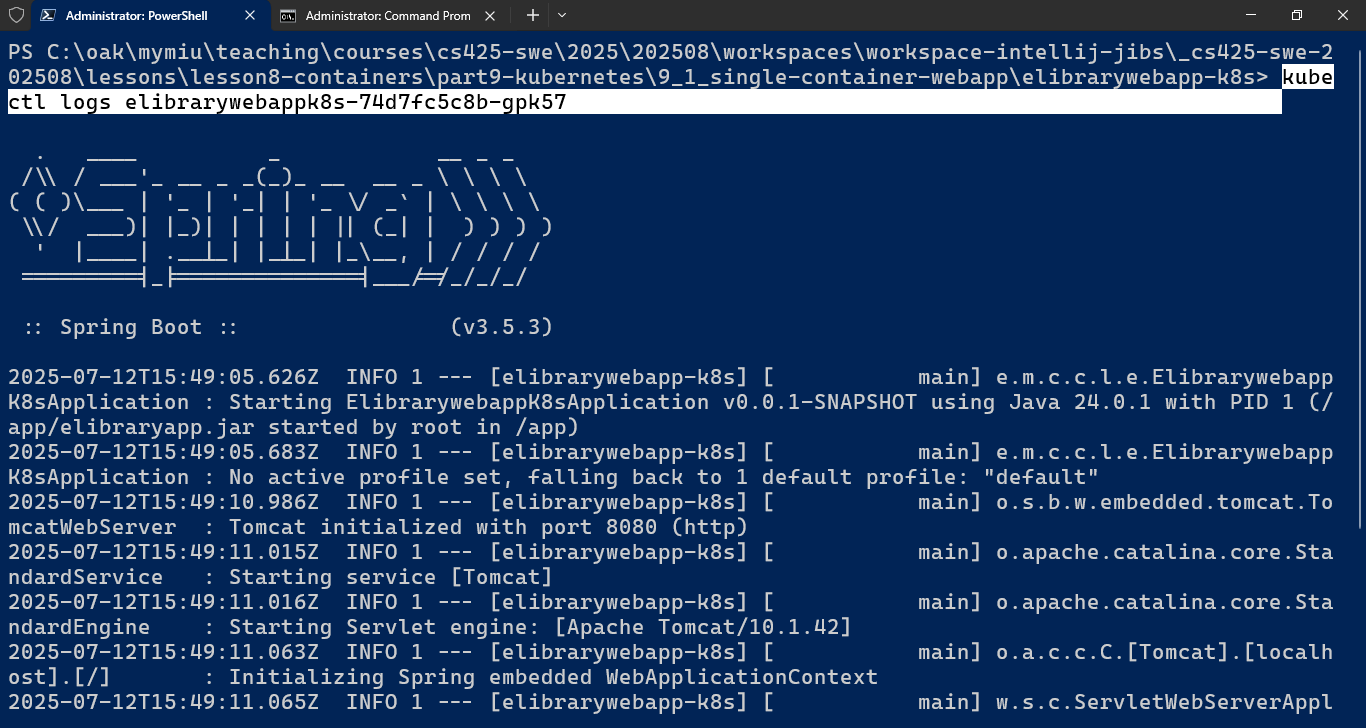
1. View full detailed information about the deployment: > kubectl describe deployment [depl-name]



1. Check the pod(s) in which the containerized application has been deployed in. > *kubectl get pods*

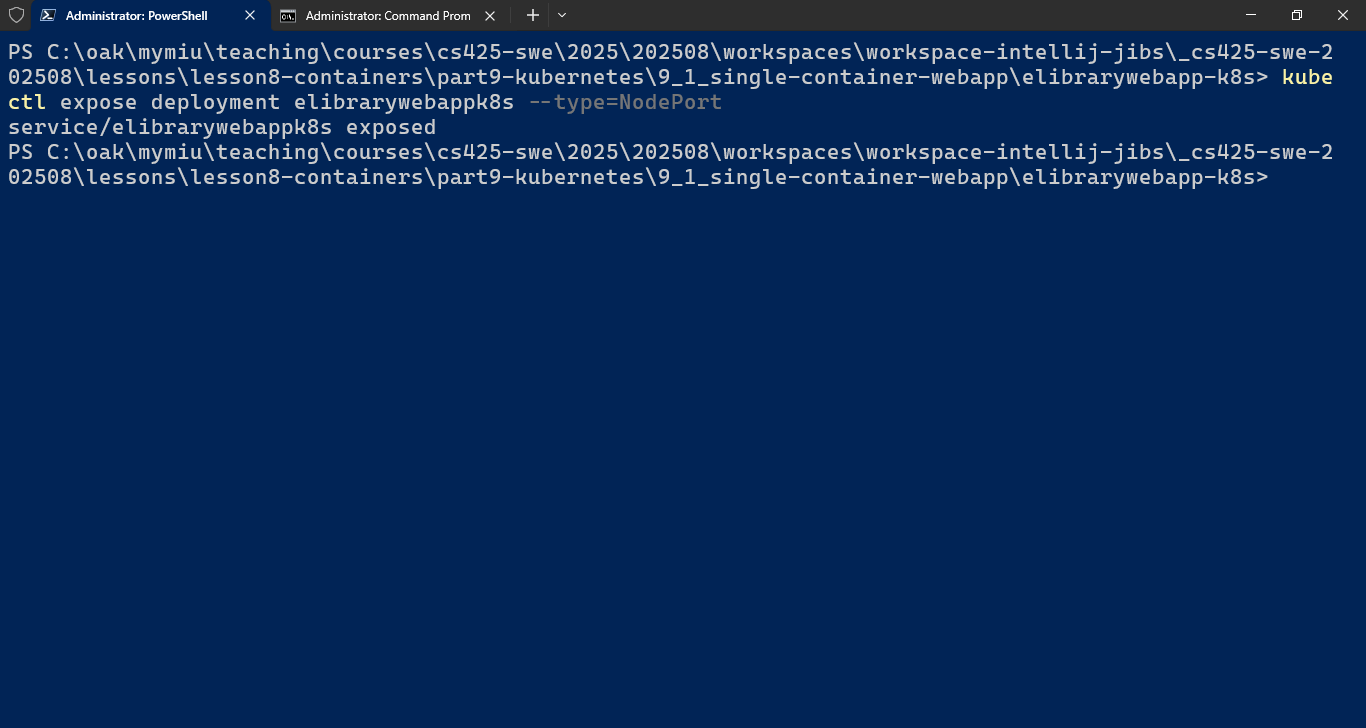


1. View the console log of the pod running the containerized application > kubectl logs [pod-name]



1. Next, to make the application accessible from the outside, a Service object needs to be created for the deployment. I.e. The Deployment needs to be exposed as a service of type, NodePort

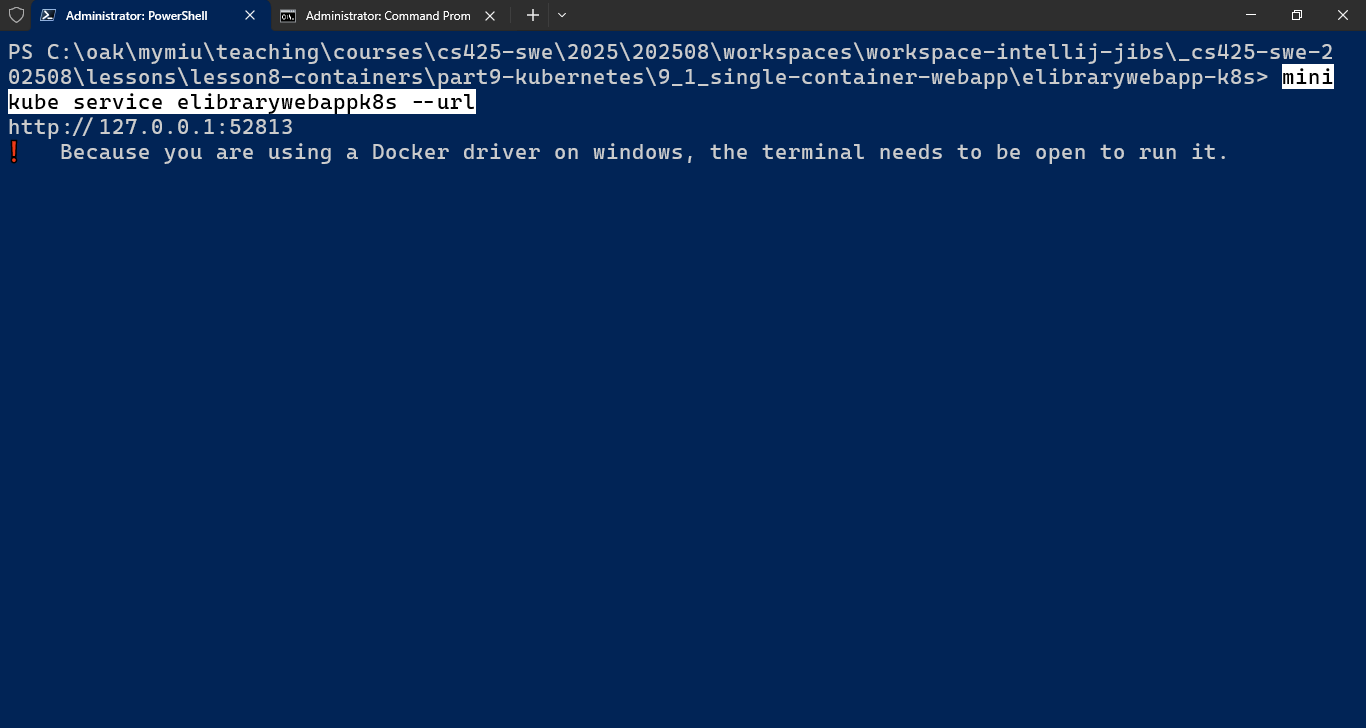
> *kubectl expose deployment [depl-name] --type=NodePort*

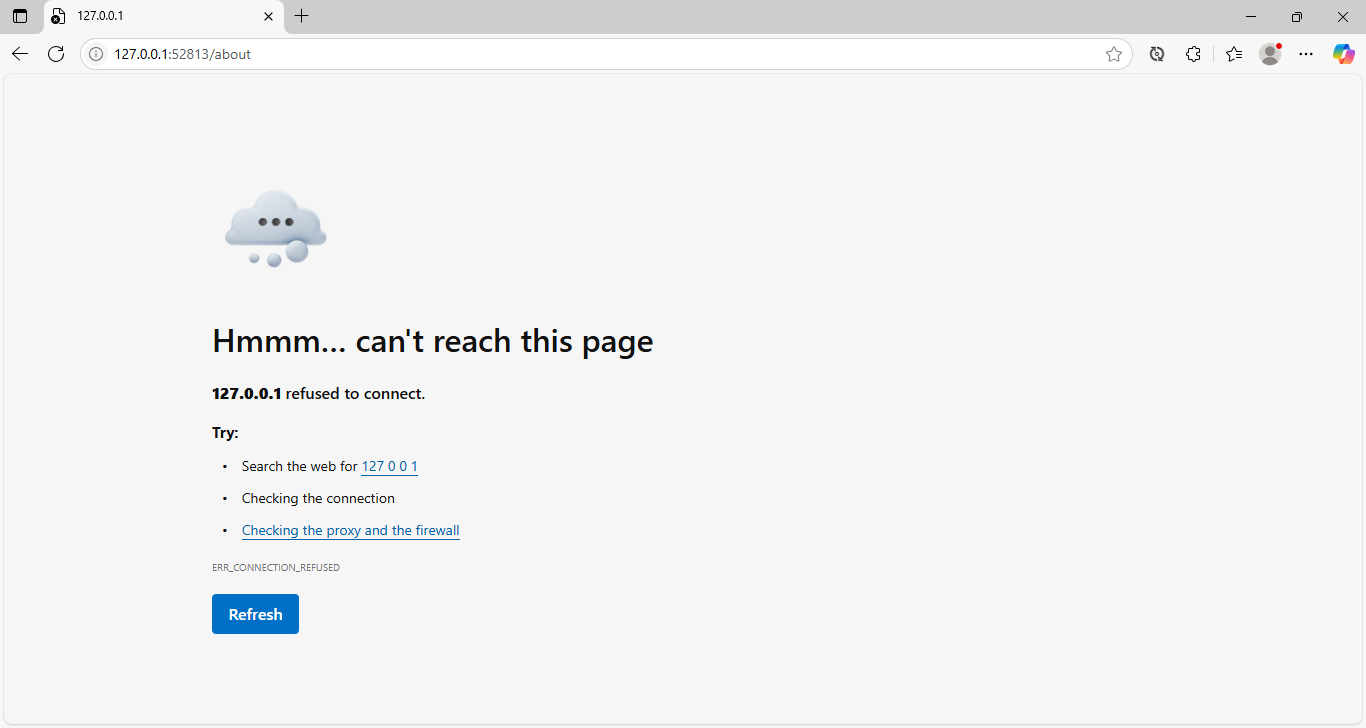


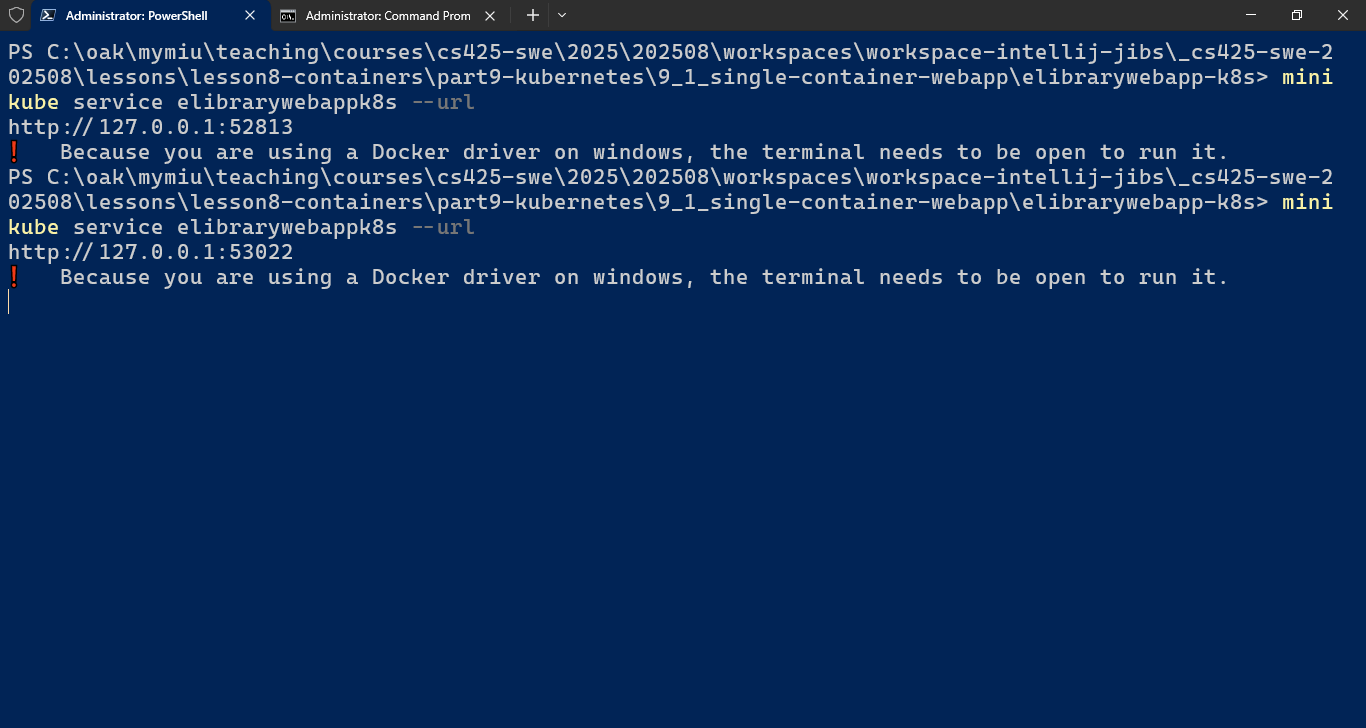
1. Verify the deployment is exposed as a service > kubectl get service or svc

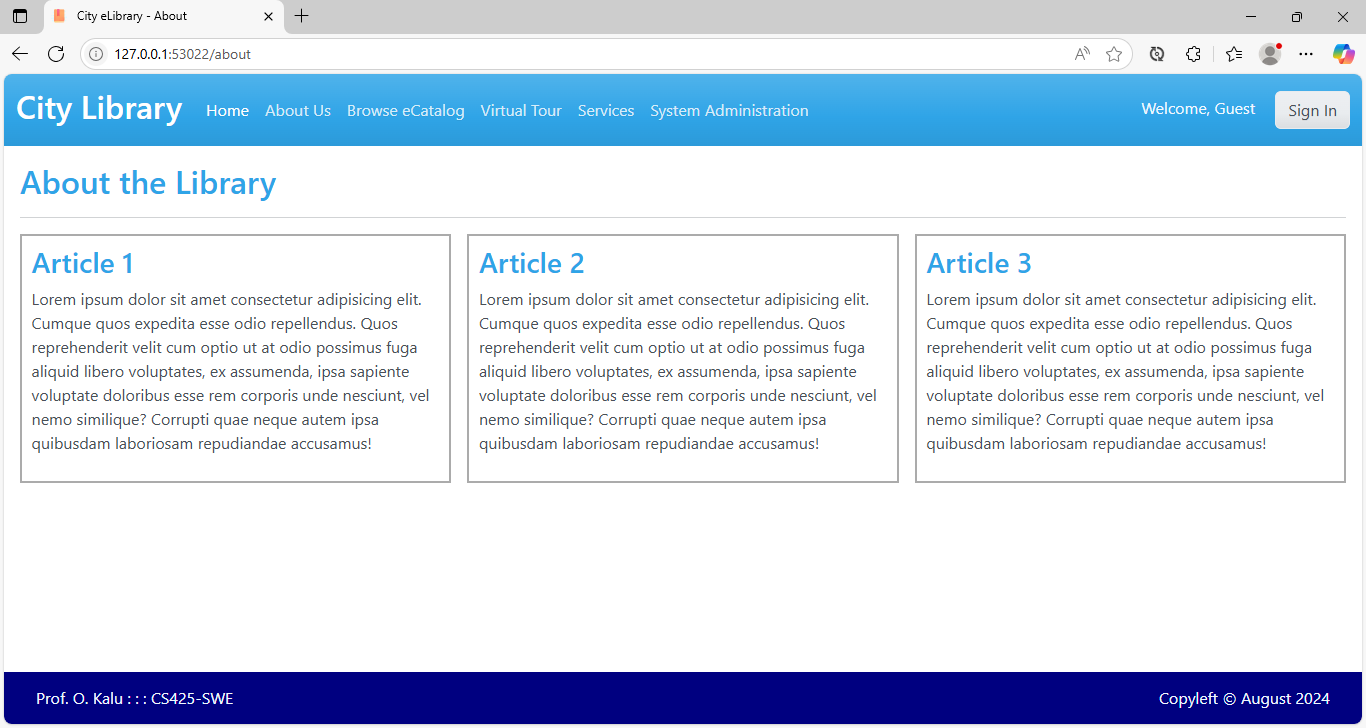


1. Next, to obtain the URL for accessing the webapp > minikube service [svc-name –elibrarywebappk8s] --url

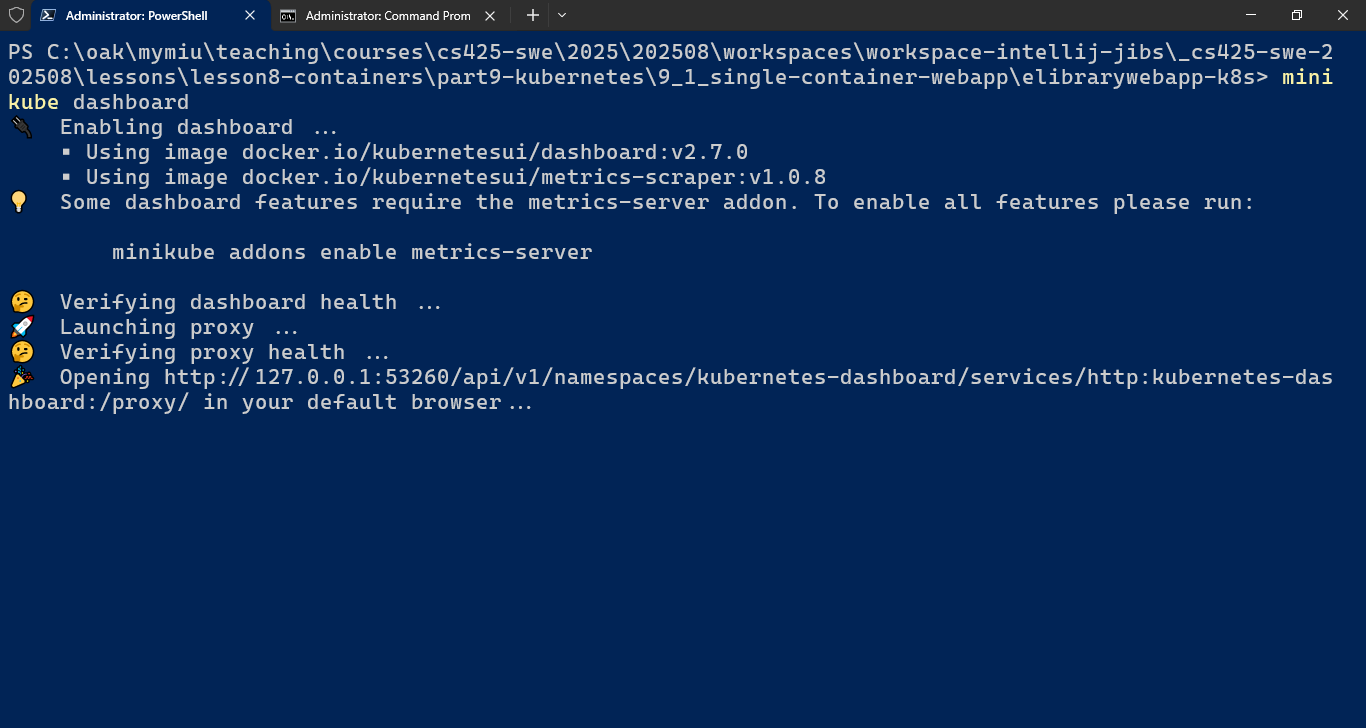
Note: The terminal/shell must be open/running to enable accessing the webapp http endpoint(s). If the shell cmd is terminated (e.g. by pressing Ctrl+C, access to the webapp NodePort service is stopped)

The cmd: > *minikube service [svc-name] --url* simply starts a network tunnel for the service, generating a URL + dynamically assigned PORT Number, for accessing the webapp.



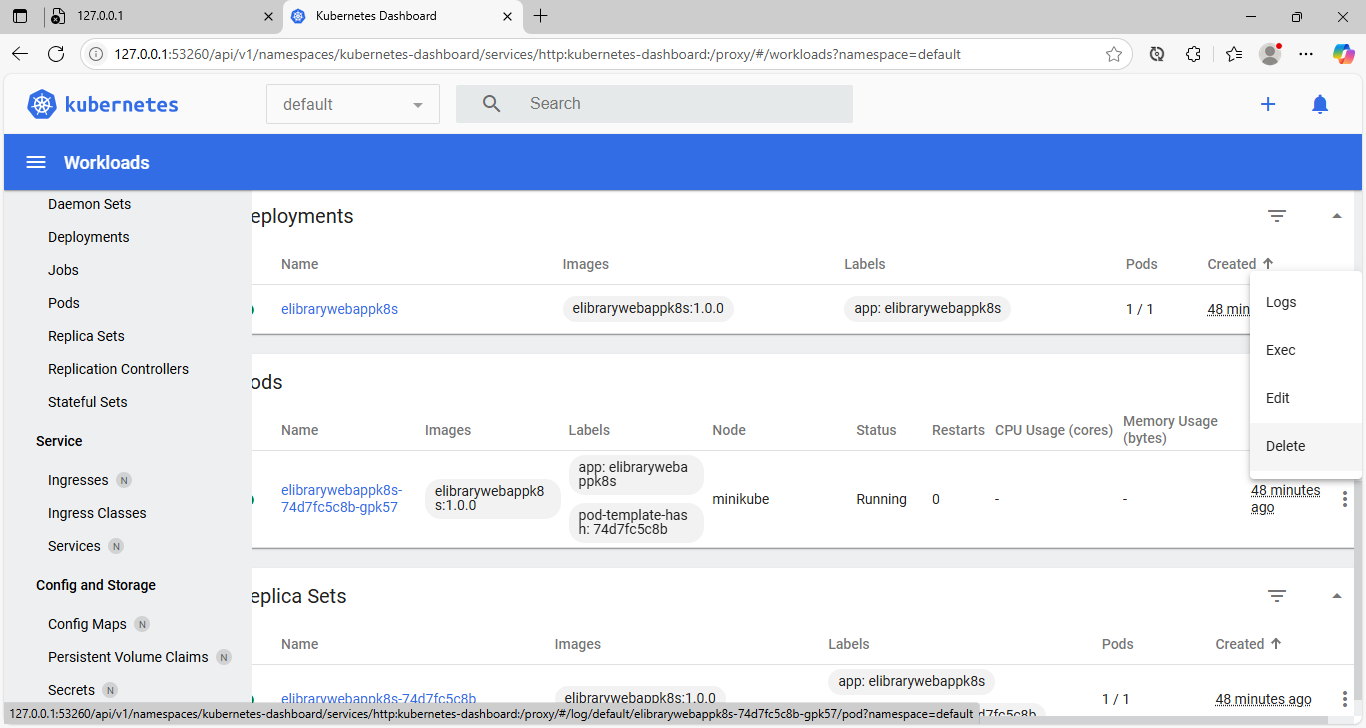


1. Next, we can view the health and status of the deployment, nodes and pods in K8s cluster. Minikube provide a GUI dashboard (a.k.a kubernetesui), which can be accessed via the cmd > *minikube dashboard*



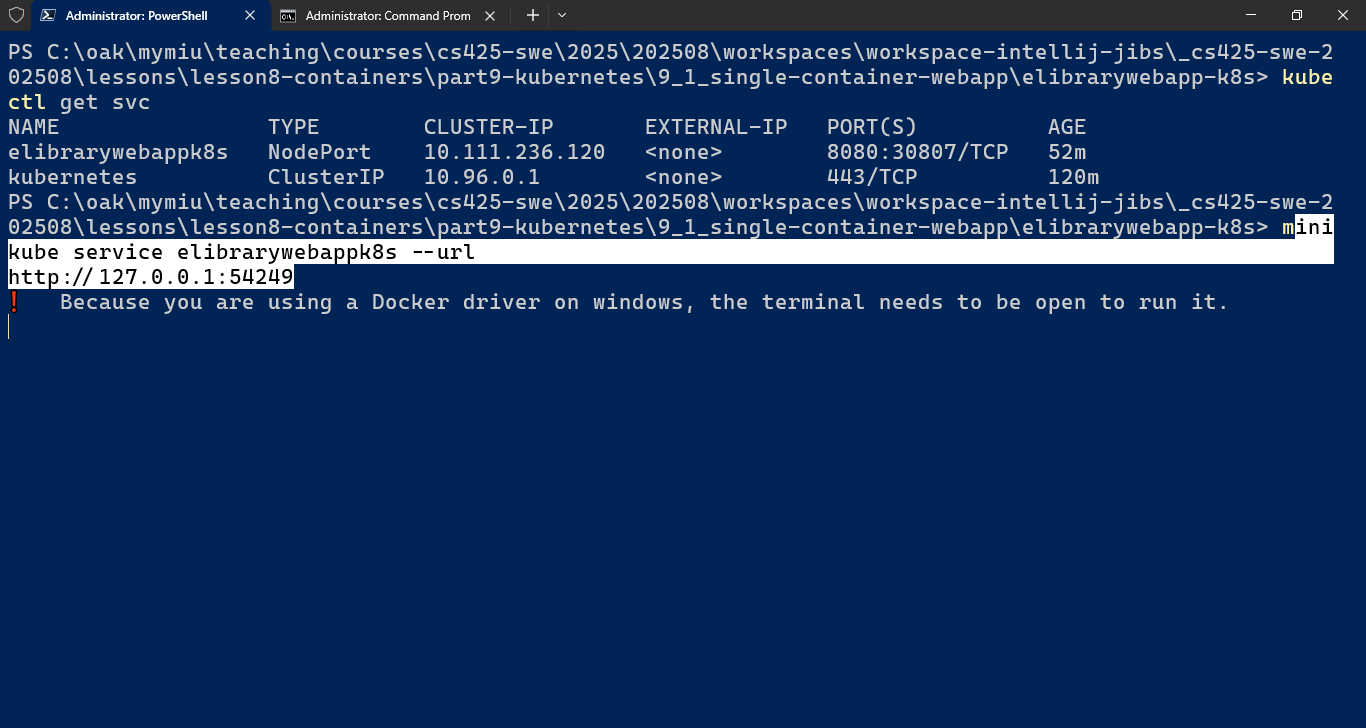


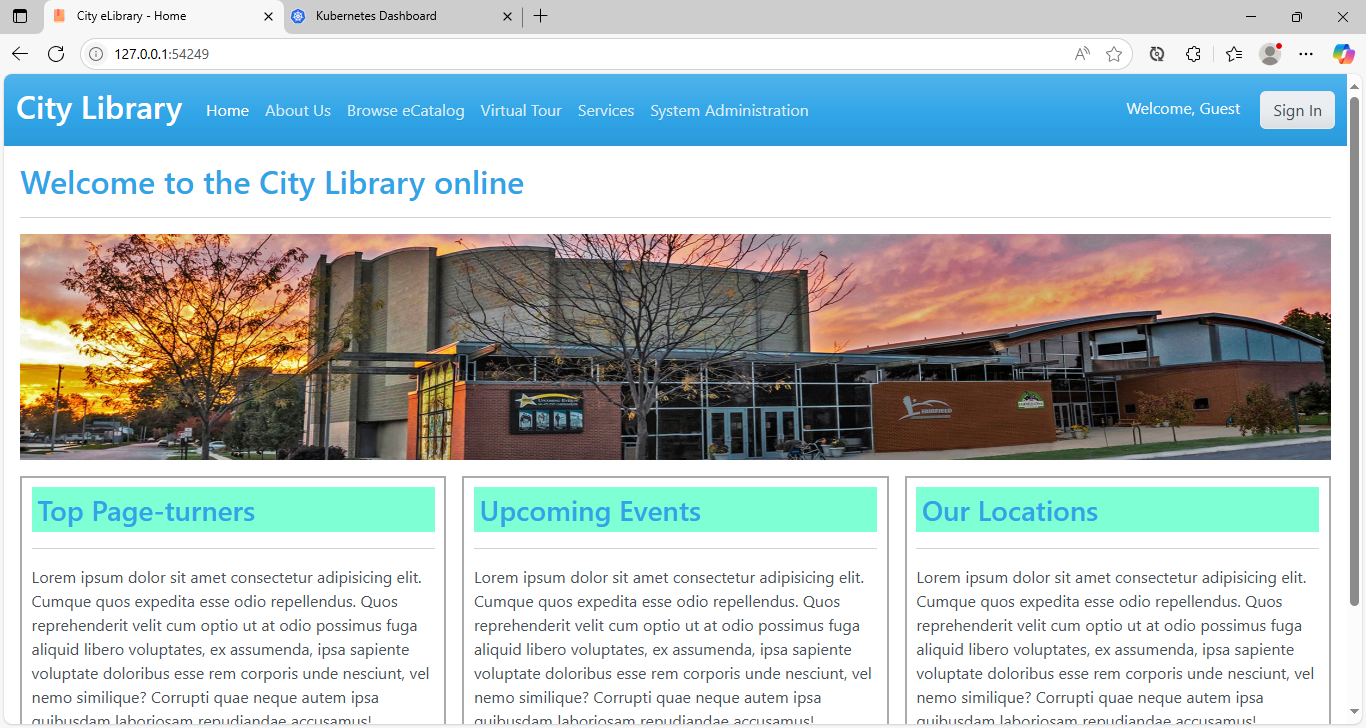
1. Demonstrate “Self-Healing” feature of Kubernetes, by deleting the pod and see that it gets recreated/replaced automatically.



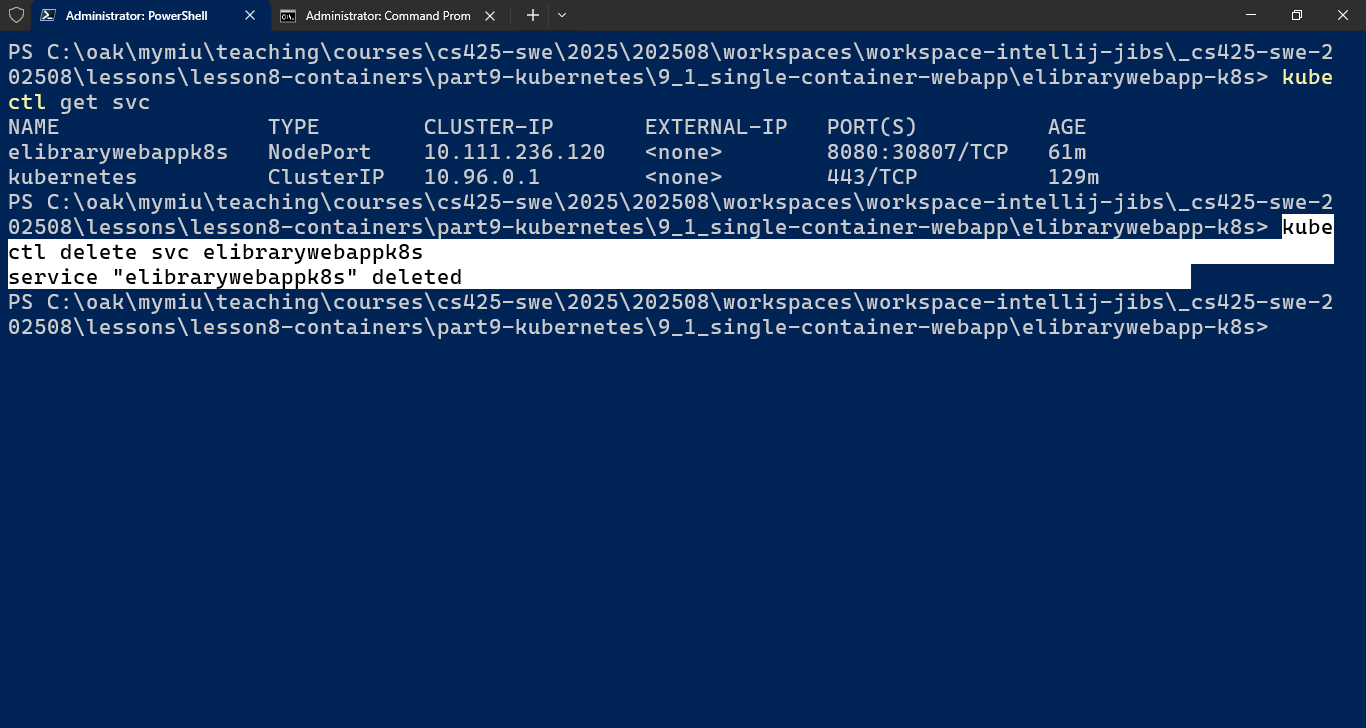
1. Once again, generate the webapp URL and view it in browser:

> minikube service [svc-name] --url

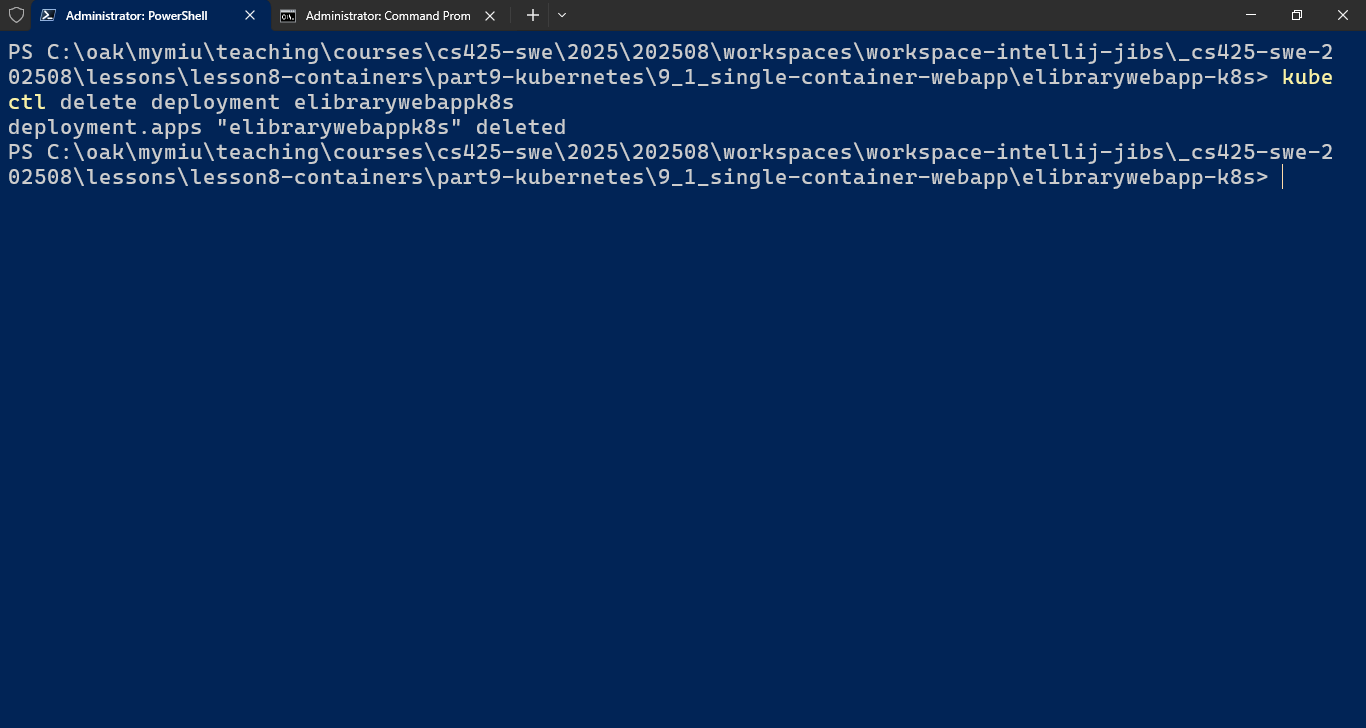




1. So, the WebApp has been successfully deployed to and ran on local kubernetes.
2. Finally, let’s clean up the environment (to avoid consuming and unnecessarily holding on to the amount of compute and storage resources it took on the PC)
   1. Delete the Service > *kubectl delete svc [svc-name-elibrarywebappk8s]*



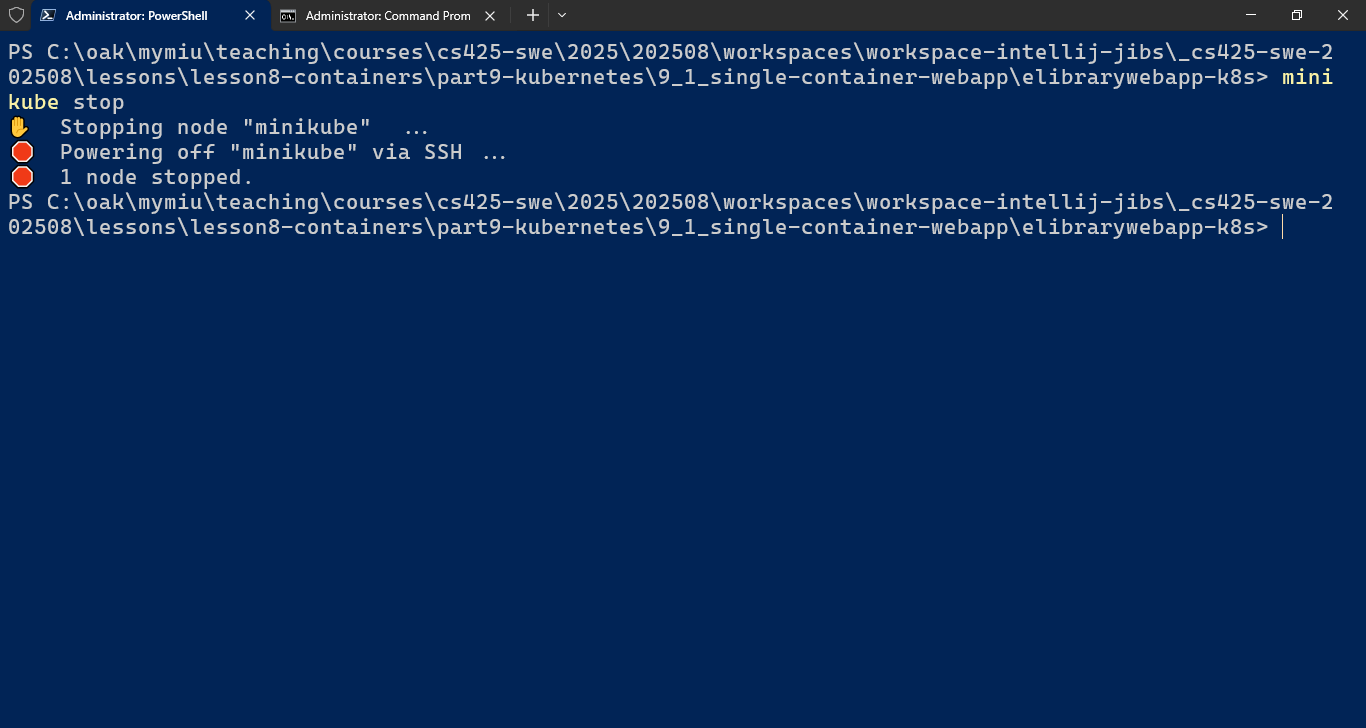
* 1. *Delete the deployment > kubectl delete deployment [depl-name]*



* 1. *Verify the deletion of the components*

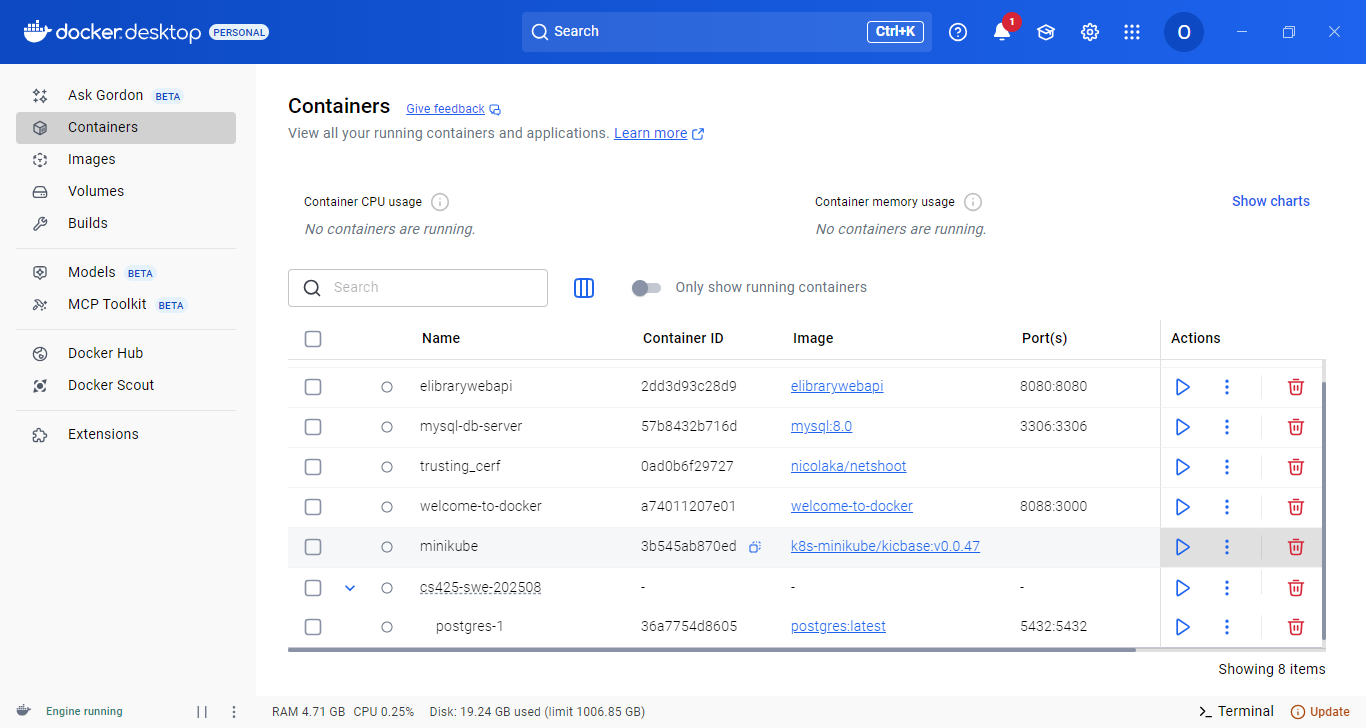


* 1. Stop the minikube Kubernetes cluster

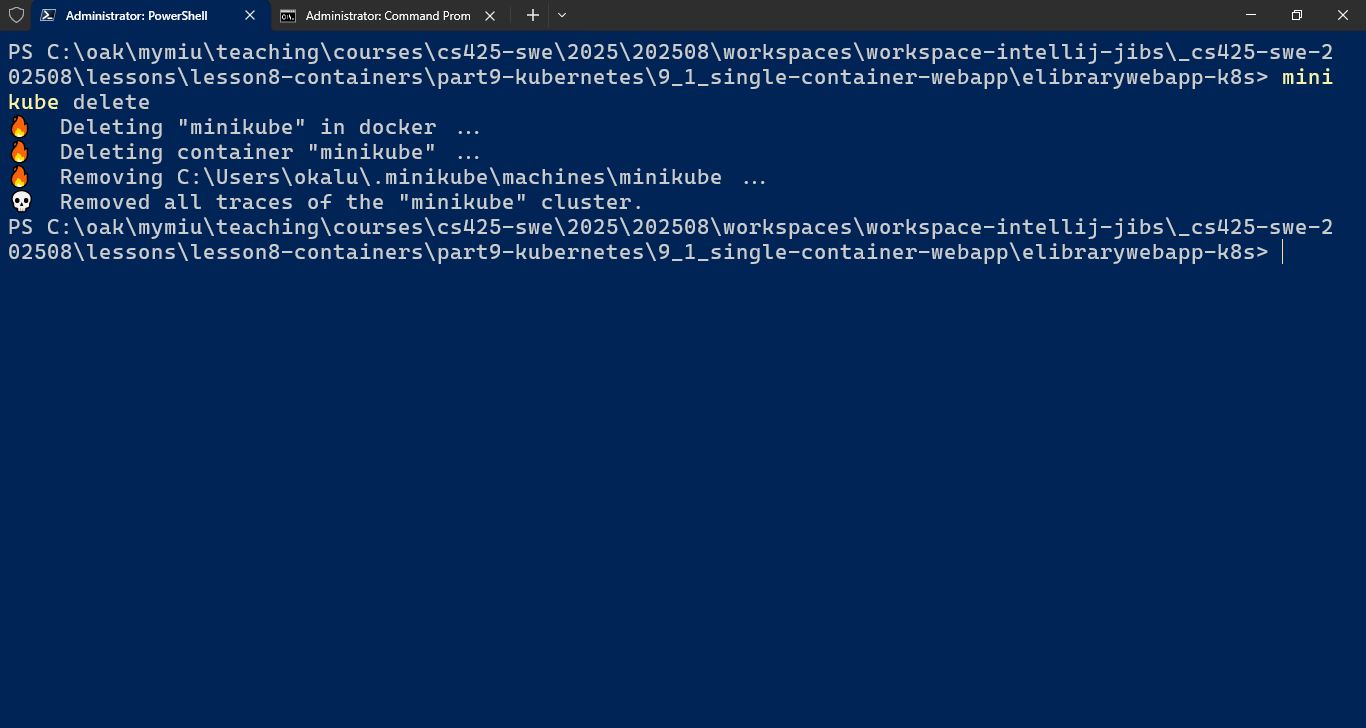


* 1. Verify minikube is stopped and the cluster has been removed.

However, the minikube container still exists.



* 1. Delete Minikube - > minikube delete



1. ... The End!!!