LAB 5

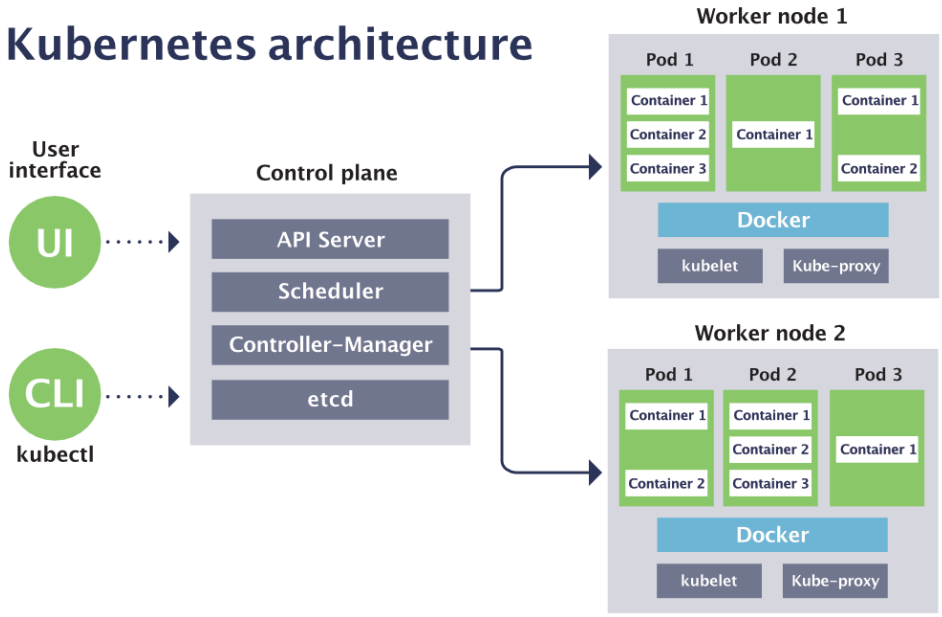
Containerized applications deployment and management using Kubernetes

Full name: Trương Đặng Trúc Lâm

Student ID: B2111933

- Note: screenshots need to be clear and good-looking; submissions must be in PDF format.

Kubernetes, also known as K8S, is an open-source system for automating deployment, scaling, and management of containerized applications. This 6-minutes video can help you quickly understand the concepts of Kubernetes. This lab provides a walkthrough of the basics of the Kubernetes cluster orchestration system



Containers: A containerized application is an application that has been packaged as one or more containers. **An image** is a static file that serves as a blueprint for creating a container. A container is a running instance of an image.

Pods: host and manage the containers that run containerized applications. A Pod can host a single container or multiple containers.

Nodes: physical or virtual machines that are used to run pods. The **master nodes** host the control plane, which is responsible for managing the state of a Kubernetes cluster. The

worker nodes are responsible for running containers. We never directly interact with the worker nodes. We send instructions to the control plane

Cluster: a Kubernetes cluster is a group of nodes used to run containerized applications

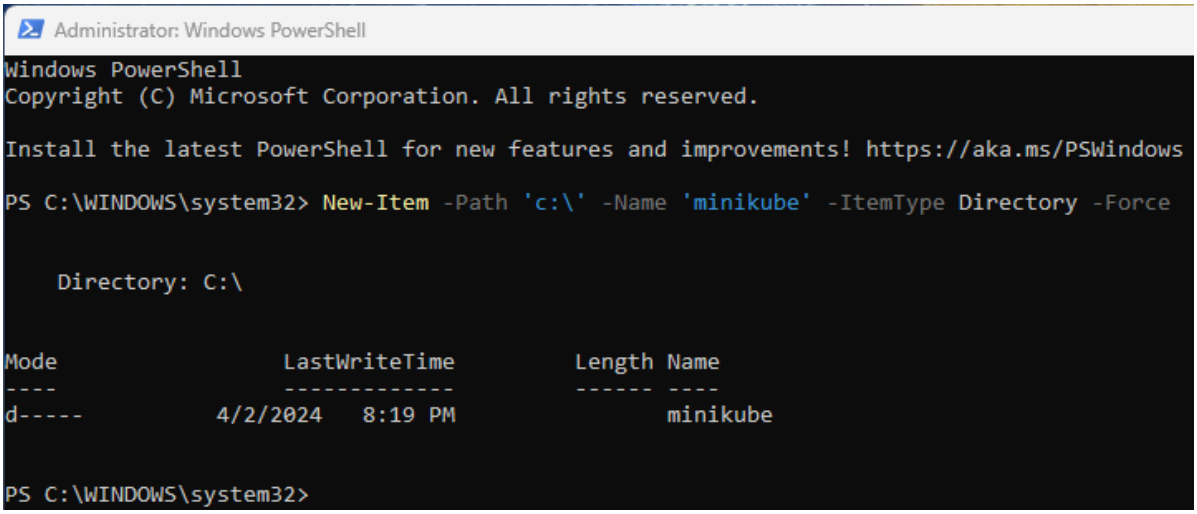
Services: A method for exposing an application that is running as one or more pods in cluster.

1. **Install minikube and kubectl on Windows OS**

minikube is local Kubernetes, focusing on making it easy to learn and develop for Kubernetes. kubectl is a command line tool for communicating with a Kubernetes cluster's control plane, using the Kubernetes API.

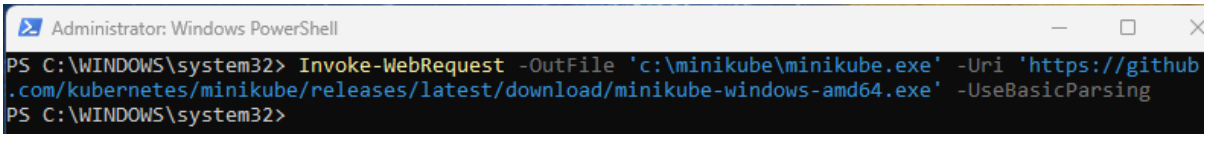
- Download minikube and kubectl using PowerShell:

```
New-Item -Path 'c:\' -Name 'minikube' -ItemType Directory -Force
```



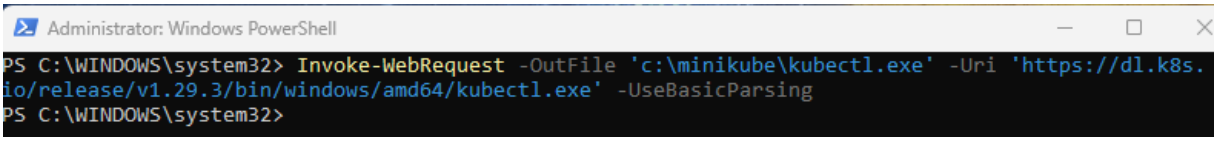
Create a directory named `minikube`

```
Invoke-WebRequest -OutFile 'c:\minikube\minikube.exe' -Uri 'https://github.com/kubernetes/minikube/releases/latest/download/minikube-windows-amd64.exe' -UseBasicParsing
```



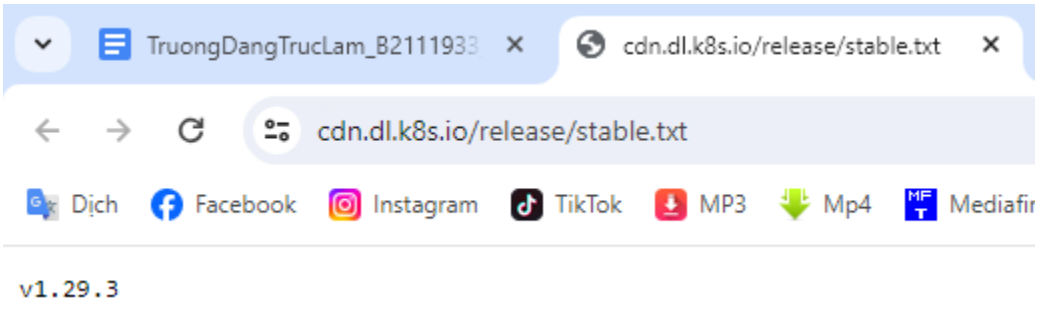
Download the latest Minikube installer for Windows (64-bit)

```
Invoke-WebRequest -OutFile 'c:\minikube\kubectl.exe' -Uri 'https://dl.k8s.io/release/v1.29.3/bin/windows/amd64/kubectl.exe' -UseBasicParsing
```



Downloads the latest `kubectl` executable for Windows (64-bit)

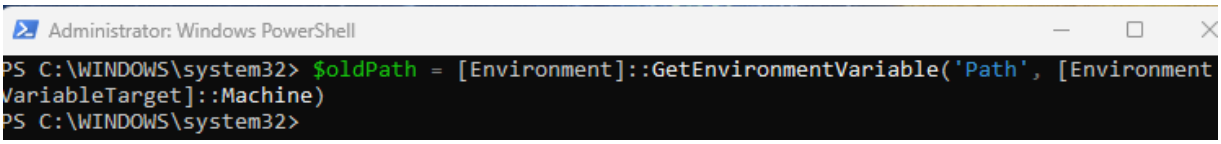
Note: To find out the latest stable version (for example, for scripting), take a look at https://dl.k8s.io/release/stable.txt.



`v1.29.3` is the latest stable version

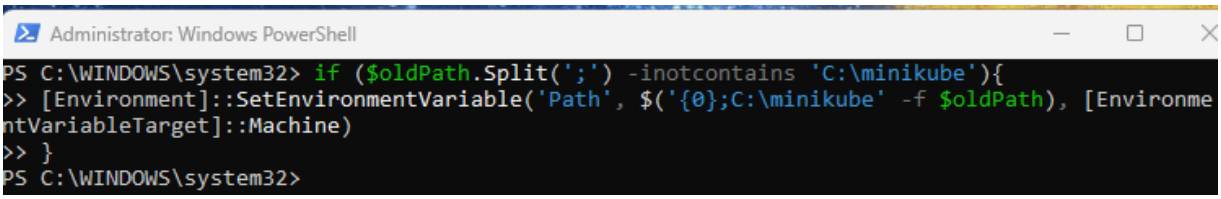
- Add the `minikube.exe` and `kubectl.exe` binary to your PATH (make sure to run PowerShell as Administrator):

```
$oldPath = [Environment]::GetEnvironmentVariable('Path', [EnvironmentVariableTarget]::Machine)
```



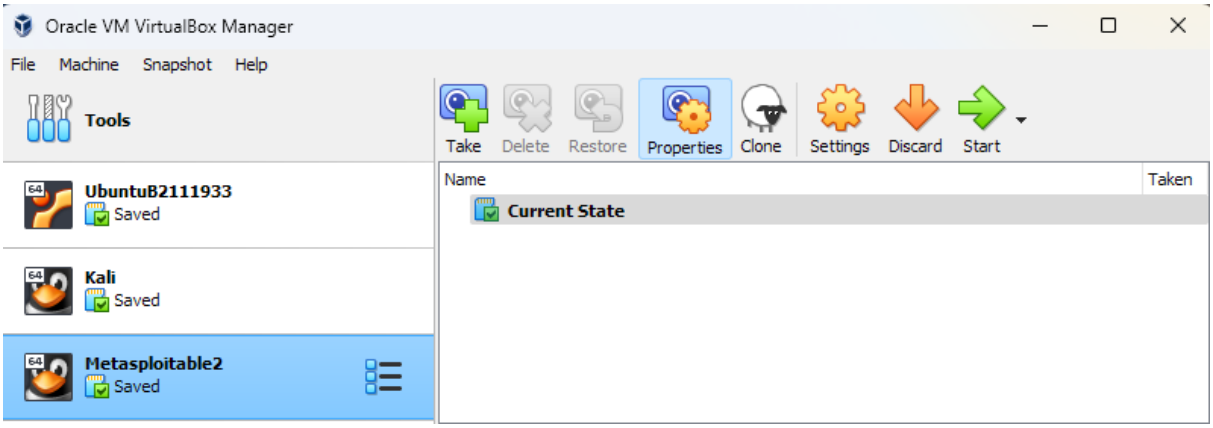
Add the `minikube.exe` and `kubectl.exe` binary to the PATH

```
if ($oldPath.Split(';') -inotcontains 'C:\minikube'){
[Environment]::SetEnvironmentVariable('Path', $('{0};C:\minikube' -f $oldPath), [EnvironmentVariableTarget]::Machine)
}
```



Check if the directory "`C:\minikube`" is already in the system PATH and adds it if not

- Download and install VirtualBox (if necessary)



Already installed

2. Learn Kubernetes Basics

2.1. Create a cluster

- Open Powershell with administrator privilege, using Minikube to create a cluster

```
minikube start
```

(take a screenshot)

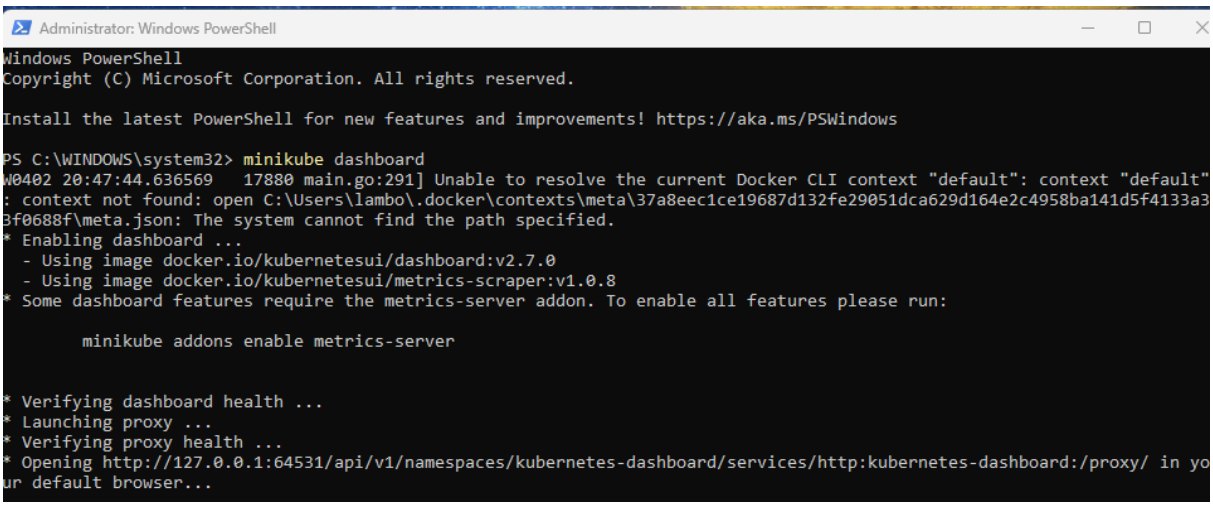
```
PS C:\WINDOWS\system32> minikube start
W0402 20:43:28.284741    7764 main.go:291] Unable to resolve the current Docker CLI context "default": context "defa
: context not found: open C:\Users\lambo\.docker\contexts\meta\37a8eec1ce19687d132fe29051dca629d164e2c4958ba141d5f41
3f0688f\meta.json: The system cannot find the path specified.
* minikube v1.32.0 on Microsoft Windows 11 Pro 10.0.22631.3296 Build 22631.3296
* Using the virtualbox driver based on existing profile
* Starting control plane node minikube in cluster minikube
* virtualbox "minikube" VM is missing, will recreate.
* Creating virtualbox VM (CPUs=2, Memory=4000MB, Disk=20000MB) ...
! This VM is having trouble accessing https://registry.k8s.io
* To pull new external images, you may need to configure a proxy: https://minikube.sigs.k8s.io/docs/reference/networ
/proxy/
* Preparing Kubernetes v1.28.3 on Docker 24.0.7 ...
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Configuring bridge CNI (Container Networking Interface) ...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Verifying Kubernetes components...
* Enabled addons: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
PS C:\WINDOWS\system32>
```

Using Minikube to create a cluster

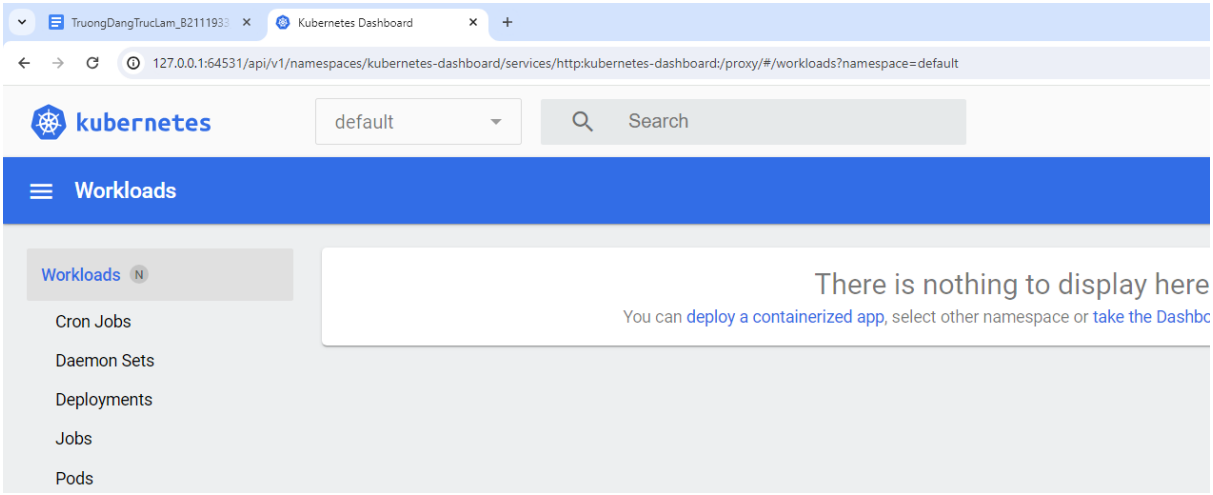
- Open the Dashboard

```
# Start a new terminal, and leave this running.
minikube dashboard
```

(take a screenshot)



Open the Dashboard

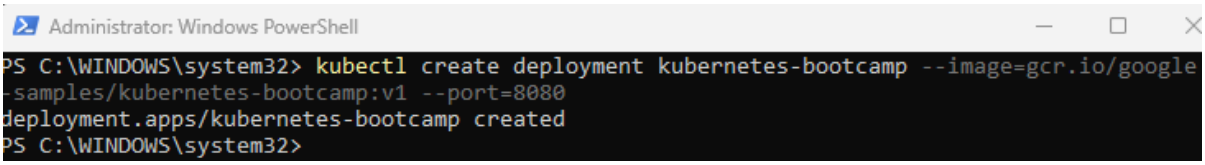


Interface of the Dashboard

2.2. Create a Deployment

- Switch back to the terminal where you ran `minikube start`
- Using kubectl to Create a Deployment

  `kubectl create deployment kubernetes-bootcamp --image=gcr.io/google-samples/kubernetes-bootcamp:v1`

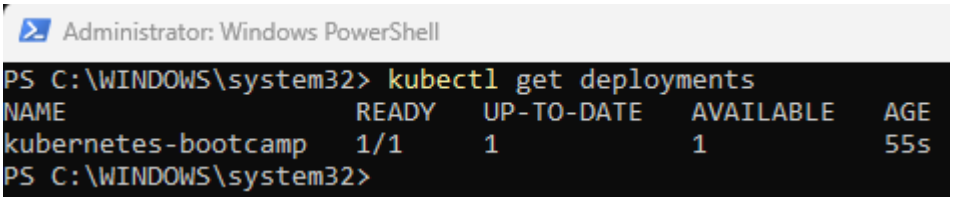


Using `kubectl` to create a deployment

- To list your deployments

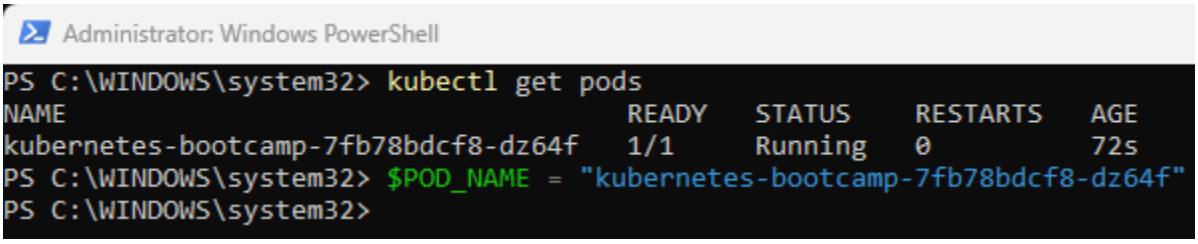
  `kubectl get deployments`

  (take a screenshot)



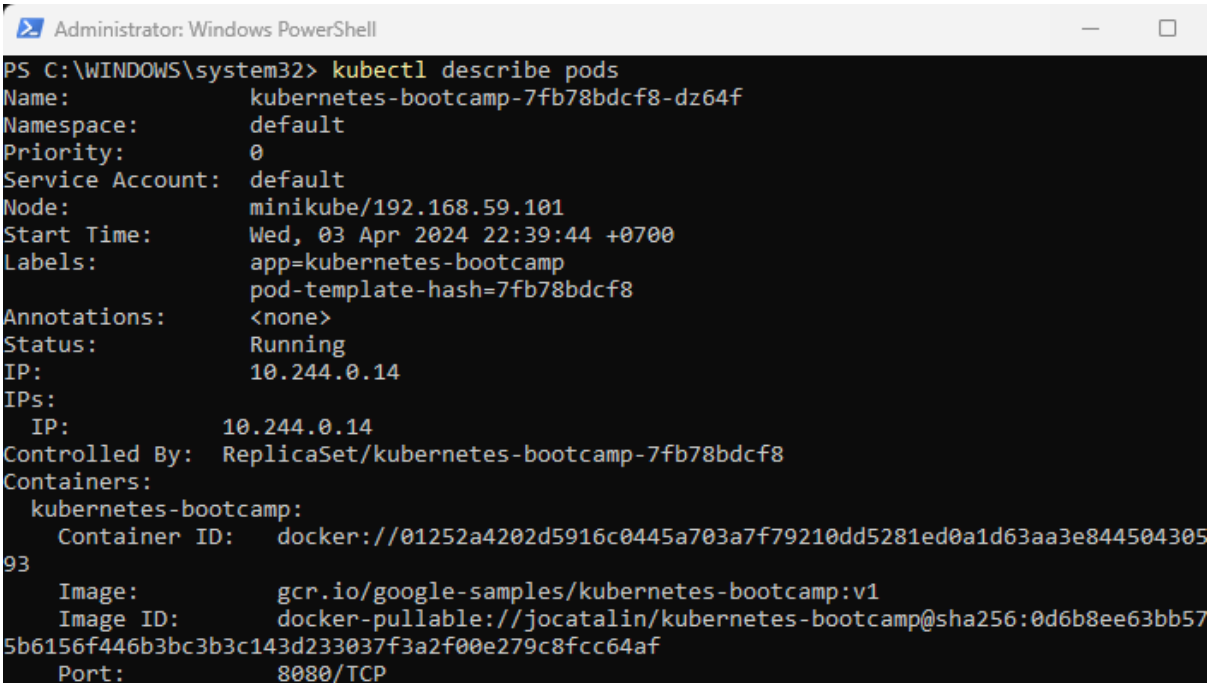
List deployments

- Looking for existing Pods:
  `kubectl get pods`
  `$POD_NAME = "<POD name>"`



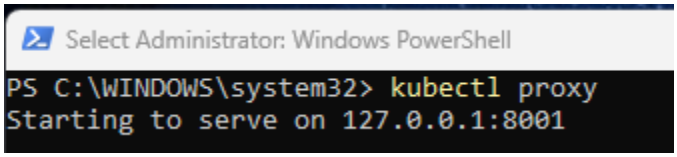
Looking for existing Pods and setup the Pod Name

- To see details about the Pod's container
  `kubectl describe pods`



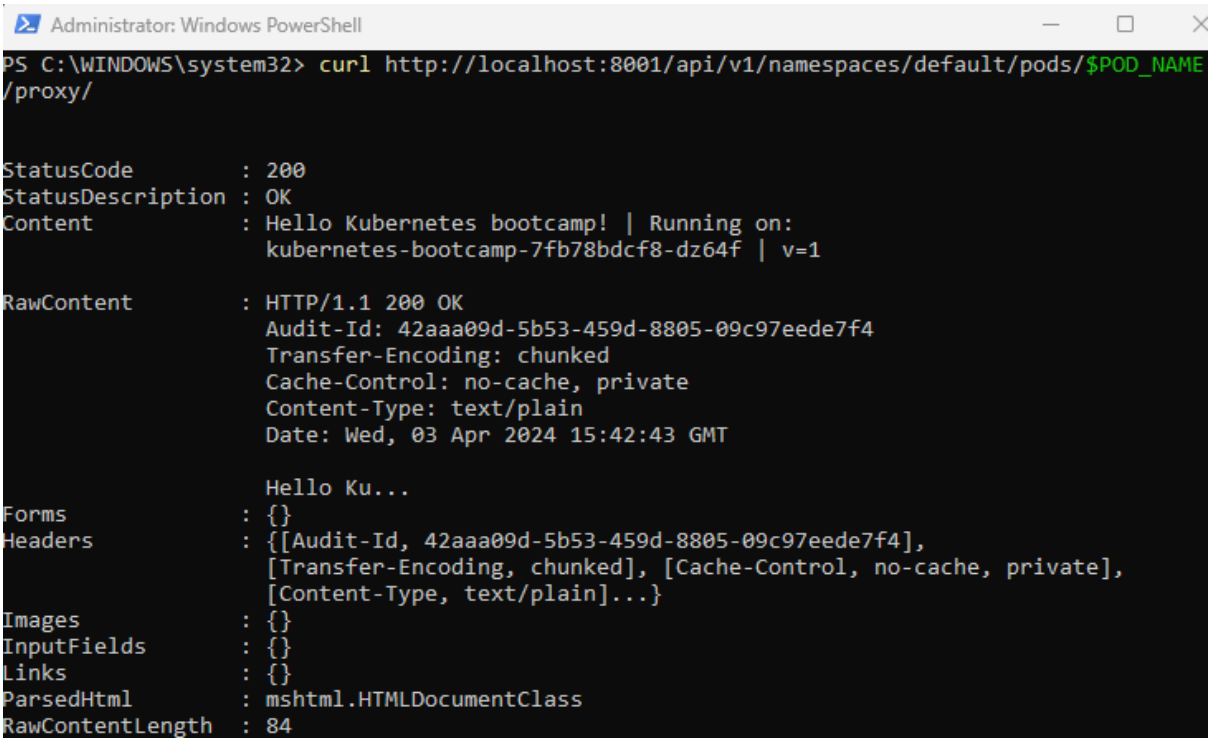
See details about the Pod's container

- Create a proxy that will forward communications into the cluster-wide private network
  `kubectl proxy`



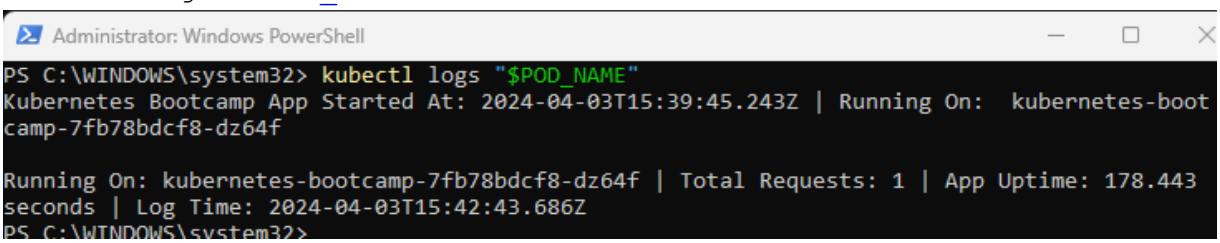
Create a proxy

- Show your app in the terminal (or a web browser)
  `curl http://localhost:8001/api/v1/namespaces/default/pods/$POD_NAME:8080/proxy/`



Display in the terminal

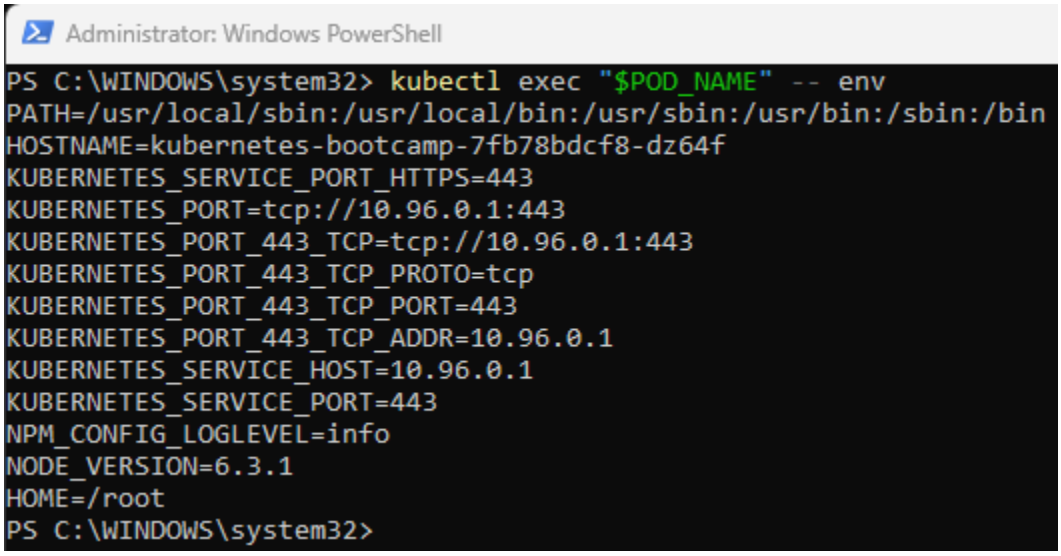
- View the container logs
  `kubectl logs "$POD_NAME"`



View the container logs

- Execute command on the container

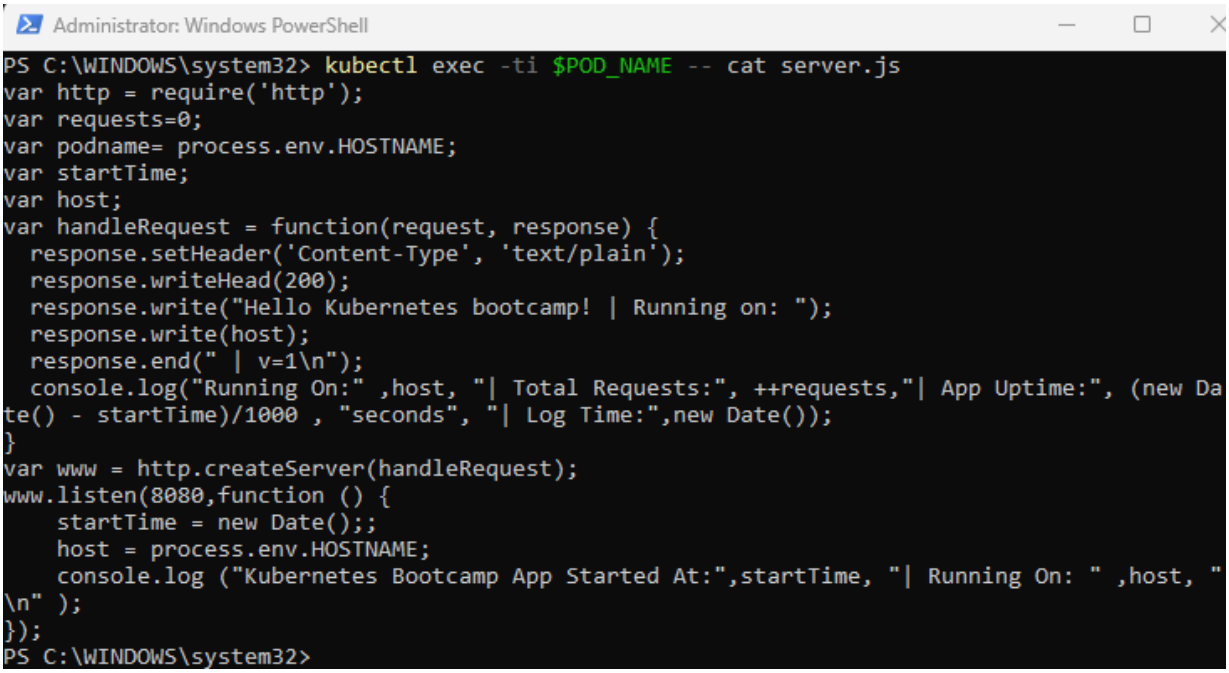
```
kubectl exec "$POD_NAME" -- env
```



Execute command on the container

- View the source code of the app is in the `server.js` file

```
kubectl exec -ti $POD_NAME -- cat server.js
```

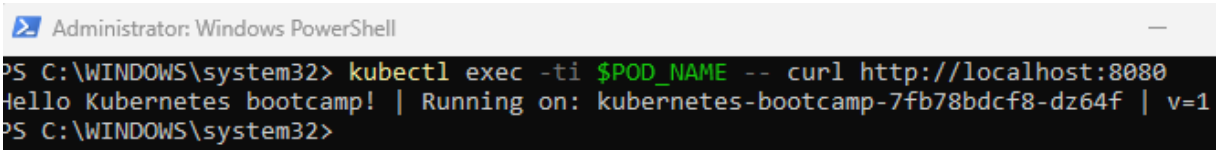


View the source code of the app is in the `server.js` file

- View the application status

```
kubectl exec -ti $POD_NAME -- curl http://localhost:8080
```

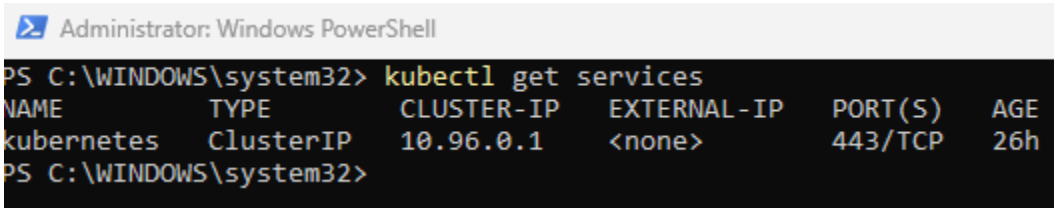
(take a screenshot)



View the application status

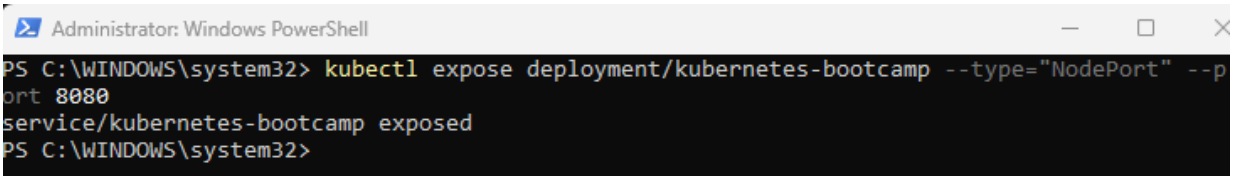
2.3. Expose Your Appicaton

- List the current services from our cluster:
  `kubectl get services`



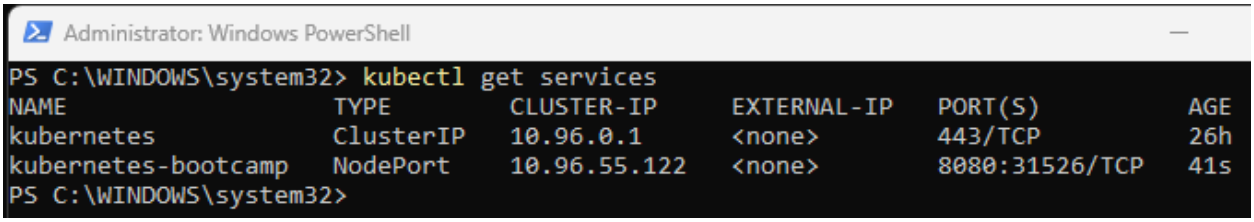
List the current services from our cluster

- Create a new service and expose it to external traffic
  `kubectl expose deployment/kubernetes-bootcamp --type="NodePort" --port 8080`
  Note: see types of services in Kubernetes



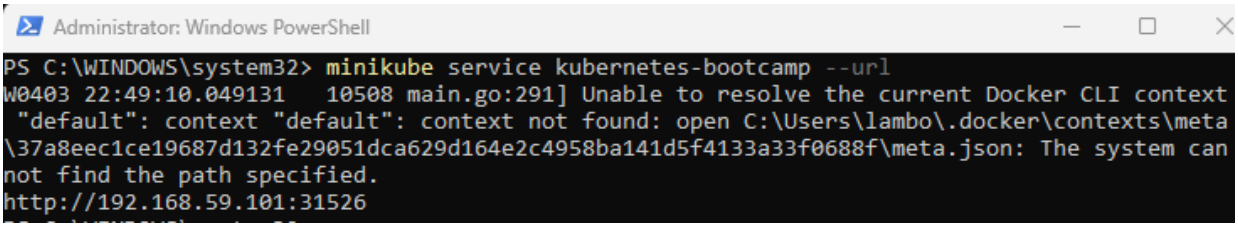
Create a new service and expose it to external traffic

- List the current services from our cluster again:
  `kubectl get services`
  (take a screenshot)



List the current services from our cluster again

- To see the service URL
  `minikube service kubernetes-bootcamp --url`

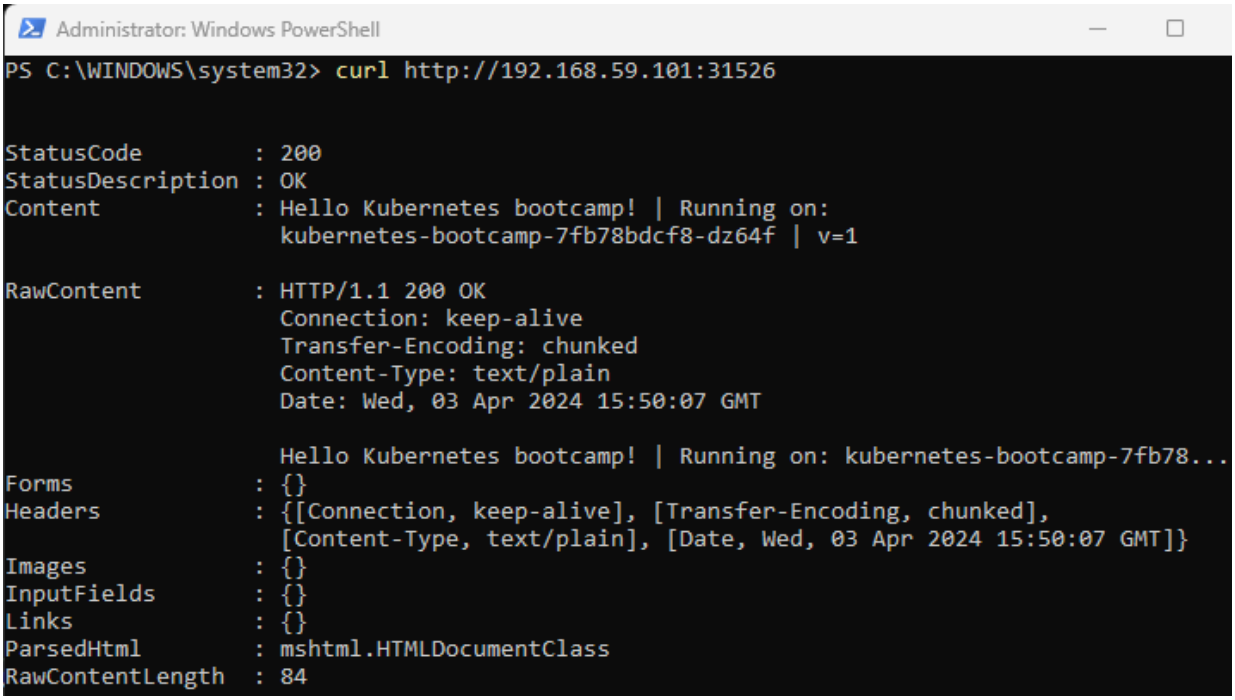


Display the service URL

- Access your service in the terminal (or a web browser)

  `curl <Service URL>`

  (take a screenshot)



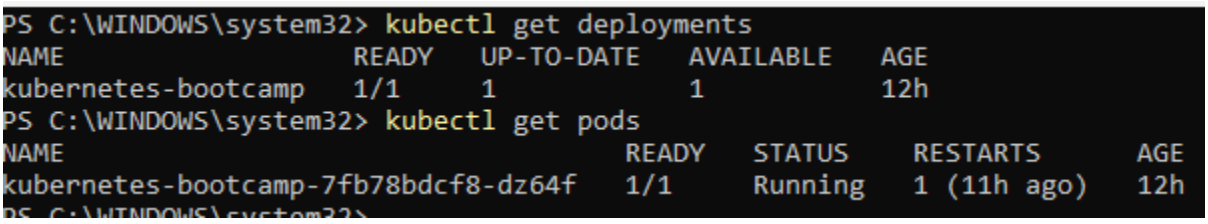
Access the service via the terminal

2.4. Scale your application

- To list your deployments

  `kubectl get deployments`

  `kubectl get pods`

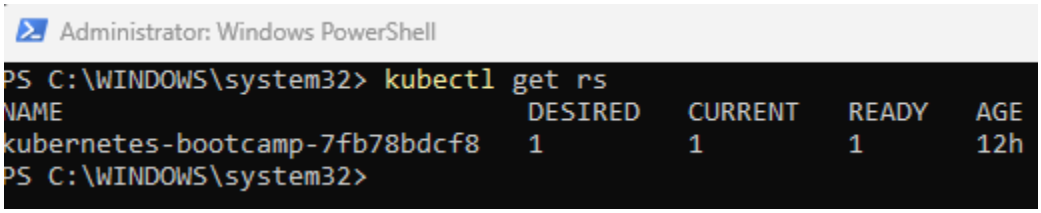


List resources

(take a screenshot)

- To see the ReplicaSet created by the Deployment

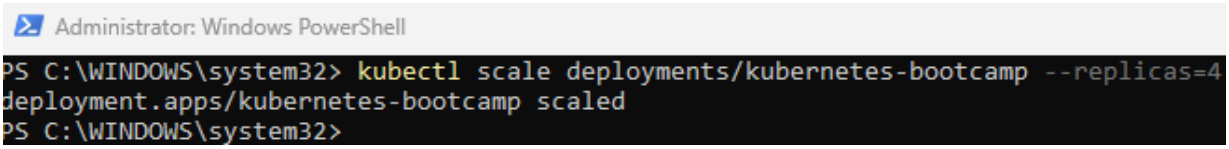
```
kubectl get rs
```



Display the ReplicaSet created by the Deployment

- Scale the Deployment to 4 replicas

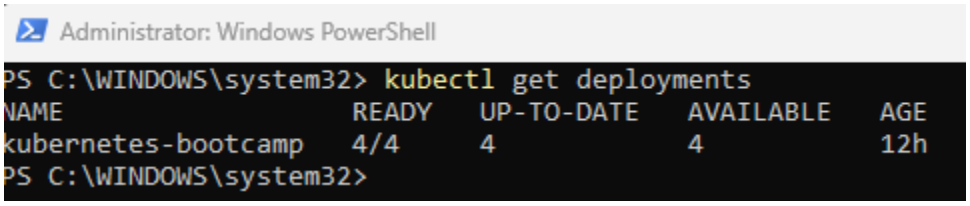
```
kubectl scale deployments/kubernetes-bootcamp --replicas=4
```



Scale the Deployment to 4 replicas

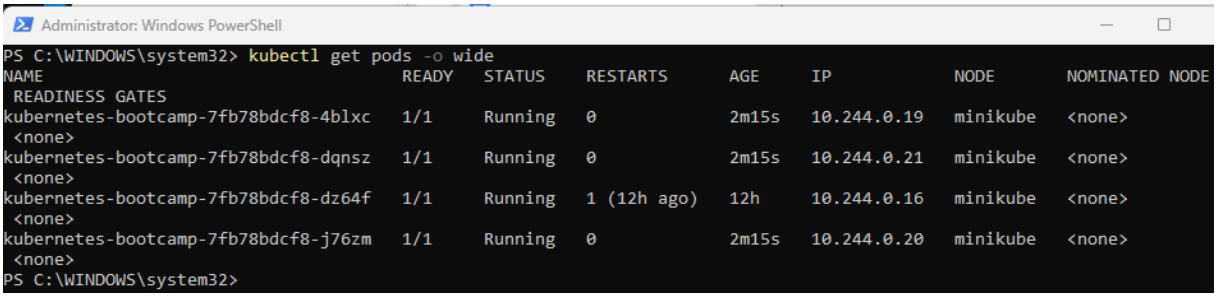
- View your Deployments once again

```
kubectl get deployments
```



Now it has 4 replicas

```
kubectl get pods -o wide
```

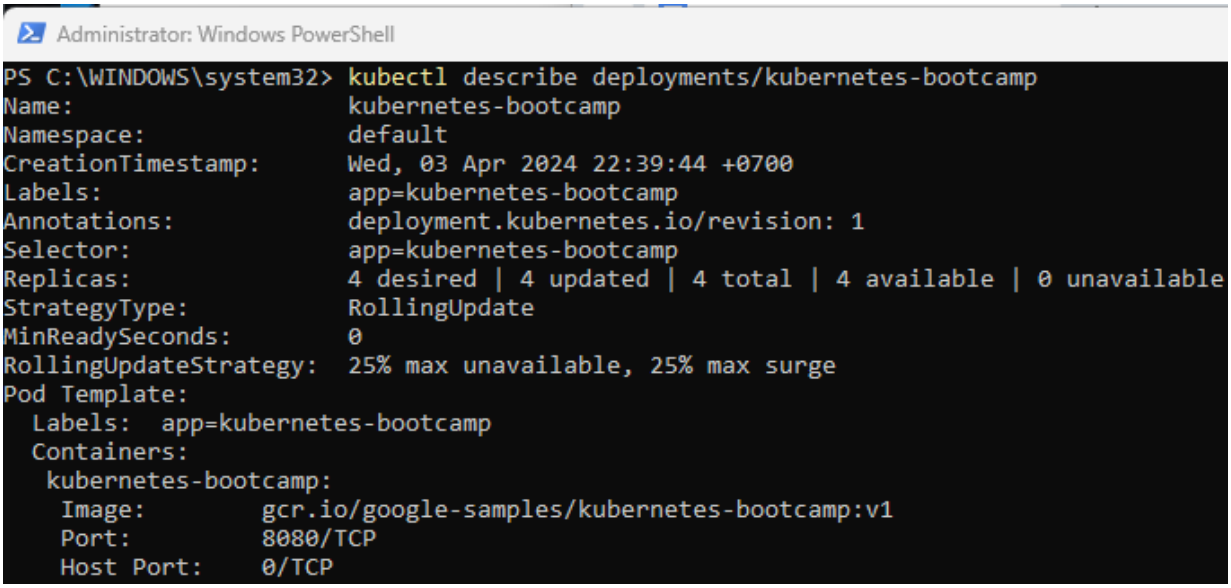


The same as

(take a screenshot)

`kubectl describe deployments/kubernetes-bootcamp`

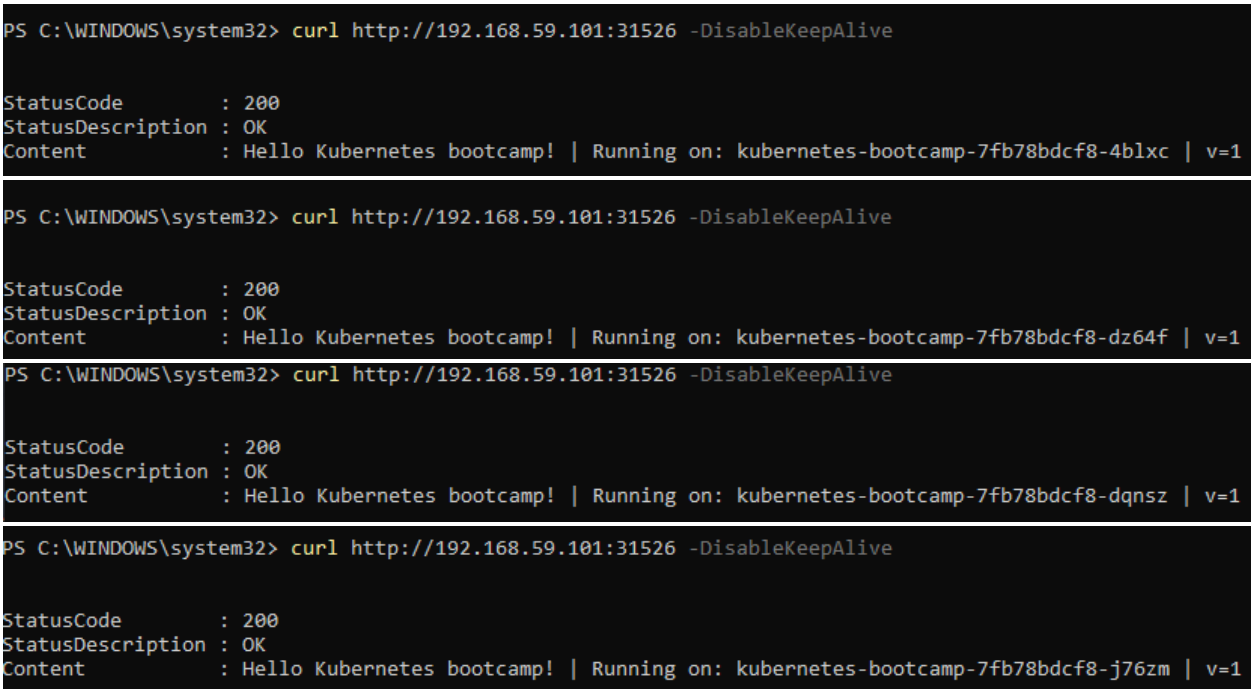
(take a screenshot)



Display the description again

- Access your service in the terminal (or a web browser) multiple times. We hit a different Pod with every request. This demonstrates that the load-balancing is working.

`curl <Service URL> -DisableKeepAlive`



We hit a different Pod with every request

2.5. Update your application

- To view the current image version of the app

  `kubectl get pods`

```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> kubectl get pods
NAME                                   READY   STATUS    RESTARTS      AGE
kubernetes-bootcamp-7fb78bdcf8-4blxc   1/1     Running   0             13m
kubernetes-bootcamp-7fb78bdcf8-dqnsz   1/1     Running   0             13m
kubernetes-bootcamp-7fb78bdcf8-dz64f   1/1     Running   1 (12h ago)   12h
kubernetes-bootcamp-7fb78bdcf8-j76zm   1/1     Running   0             13m
PS C:\WINDOWS\system32>
```

Display pods

  `kubectl describe pods`

```
PS C:\WINDOWS\system32> kubectl describe pods
Name:             kubernetes-bootcamp-7fb78bdcf8-4blxc
Namespace:        default
Priority:         0
Service Account:  default
Node:             minikube/192.168.59.101
Start Time:       Thu, 04 Apr 2024 10:52:03 +0700
Labels:           app=kubernetes-bootcamp
                  pod-template-hash=7fb78bdcf8
Annotations:      <none>
Status:           Running
IP:               10.244.0.19
IPs:
  IP:           10.244.0.19
Controlled By:  ReplicaSet/kubernetes-bootcamp-7fb78bdcf8

Name:             kubernetes-bootcamp-7fb78bdcf8-dqnsz
Namespace:        default
Priority:         0
Service Account:  default
Node:             minikube/192.168.59.101
Start Time:       Thu, 04 Apr 2024 10:52:03 +0700
Labels:           app=kubernetes-bootcamp
                  pod-template-hash=7fb78bdcf8
Annotations:      <none>
Status:           Running
IP:               10.244.0.21
IPs:
  IP:           10.244.0.21
Controlled By:  ReplicaSet/kubernetes-bootcamp-7fb78bdcf8
```

Display the description of pods again

- Update the image of the application to version 2

  `kubectl set image deployments/kubernetes-bootcamp kubernetes-bootcamp=docker.io/jocatalin/kubernetes-bootcamp:v2`

```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> kubectl set image deployments/kubernetes-bootcamp kubernetes-bootcamp=
docker.io/jocatalin/kubernetes-bootcamp:v2
deployment.apps/kubernetes-bootcamp image updated
PS C:\WINDOWS\system32>
```

`kubernetes-bootcamp` image updated

- Check the status of the new Pods, and view the old one terminating

  `kubectl get pods`

```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> kubectl get pods
NAME                                   READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-5f9578c84f-d9hql   1/1     Running   0          3m11s
kubernetes-bootcamp-5f9578c84f-m56vm   1/1     Running   0          3m12s
kubernetes-bootcamp-5f9578c84f-rf22j   1/1     Running   0          3m17s
kubernetes-bootcamp-5f9578c84f-sw8k6   1/1     Running   0          3m17s
PS C:\WINDOWS\system32>
```

Check the status of the new Pods

- Roll back the deployment to your last working version

  `kubectl rollout undo deployments/kubernetes-bootcamp`

```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> kubectl rollout undo deployments/kubernetes-bootcamp
deployment.apps/kubernetes-bootcamp rolled back
PS C:\WINDOWS\system32>
```

Roll back

- To view the current image version of the app

  `kubectl get pods`

```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> kubectl get pods
NAME                                   READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-7fb78bdcf8-bwkqh   1/1     Running   0          70s
kubernetes-bootcamp-7fb78bdcf8-j9mlj   1/1     Running   0          69s
kubernetes-bootcamp-7fb78bdcf8-khx9q   1/1     Running   0          70s
kubernetes-bootcamp-7fb78bdcf8-mjqcf   1/1     Running   0          69s
PS C:\WINDOWS\system32>
```

View the current image version of the app

`kubectl describe pods`

(take a screenshot)

```
PS C:\WINDOWS\system32> kubectl describe pods
Name:             kubernetes-bootcamp-7fb78bdcf8-bwkqh
Namespace:        default
Priority:         0
Service Account:  default
Node:             minikube/192.168.59.101
Start Time:       Thu, 04 Apr 2024 11:12:47 +0700
Labels:           app=kubernetes-bootcamp
                  pod-template-hash=7fb78bdcf8
Annotations:      <none>
Status:           Running
IP:               10.244.0.26
IPs:
  IP:           10.244.0.26
Controlled By:  ReplicaSet/kubernetes-bootcamp-7fb78bdcf8
Containers:
  kubernetes-bootcamp:
    Container ID:   docker://457b5e40401cc012b900e159ed8c45d52a75fb209aa6d8db11f2122487dfcb0d
Name:             kubernetes-bootcamp-7fb78bdcf8-j9mlj
Namespace:        default
Priority:         0
Service Account:  default
Node:             minikube/192.168.59.101
Start Time:       Thu, 04 Apr 2024 11:12:48 +0700
Labels:           app=kubernetes-bootcamp
                  pod-template-hash=7fb78bdcf8
Annotations:      <none>
Status:           Running
IP:               10.244.0.28
IPs:
  IP:           10.244.0.28
Controlled By:  ReplicaSet/kubernetes-bootcamp-7fb78bdcf8
Containers:
  kubernetes-bootcamp:
    Container ID:   docker://761278bfb94aa9844c22f2d7120a917dd36817933175d186958cbe30779e8e7c
```

Display the description of pods again

2.6. Cleaning up

- We can clean up the resources we created in the cluster:

`kubectl delete service kubernetes-bootcamp`

```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> kubectl delete service kubernetes-bootcamp
service "kubernetes-bootcamp" deleted
PS C:\WINDOWS\system32>
```

Delete the service

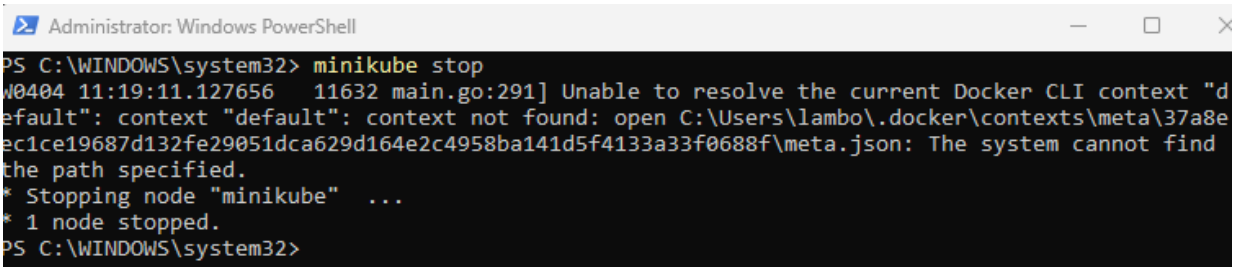
`kubectl delete deployment kubernetes-bootcamp`

```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> kubectl delete deployment kubernetes-bootcamp
deployment.apps "kubernetes-bootcamp" deleted
PS C:\WINDOWS\system32>
```

Delete deployments

- Stop the Minikube cluster

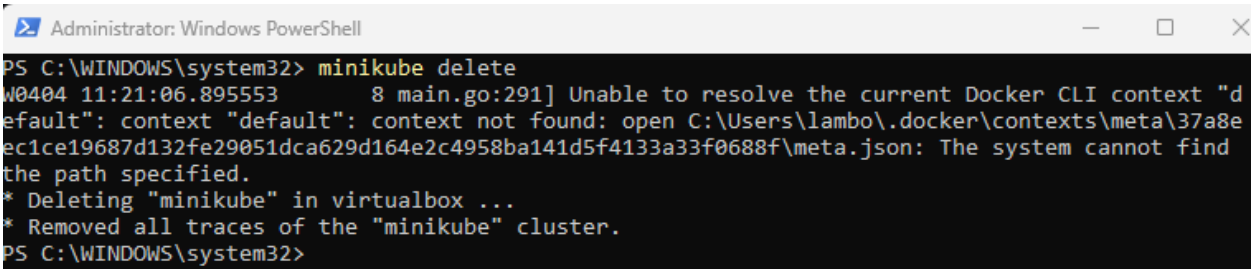
```
minikube stop
```



Stop the Minikube cluster

- Delete the Minikube VM (optional)

```
minikube delete
```

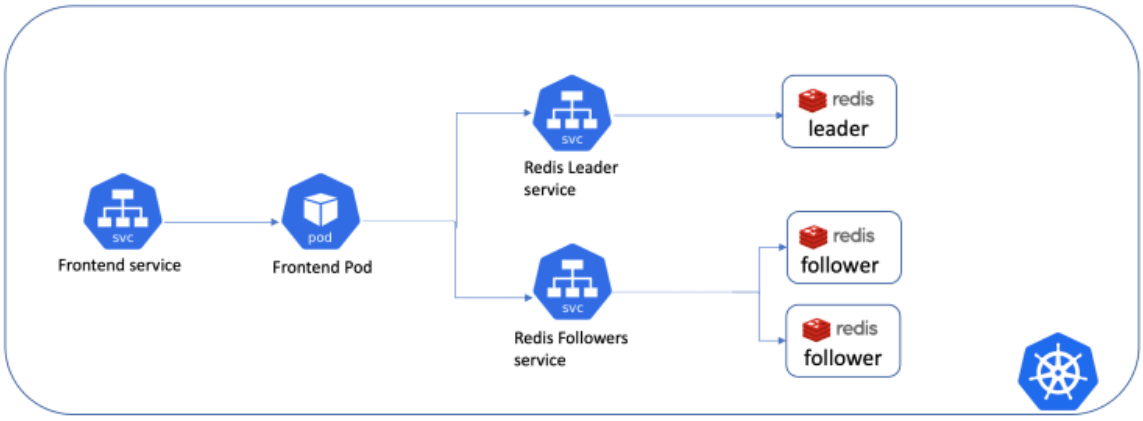


Delete the Minikube VM

3. Deploying PHP Guestbook application with Redis

This tutorial shows you how to build and deploy a simple (not production ready), multi-tier web application using Kubernetes and Docker. This example consists of the following components:

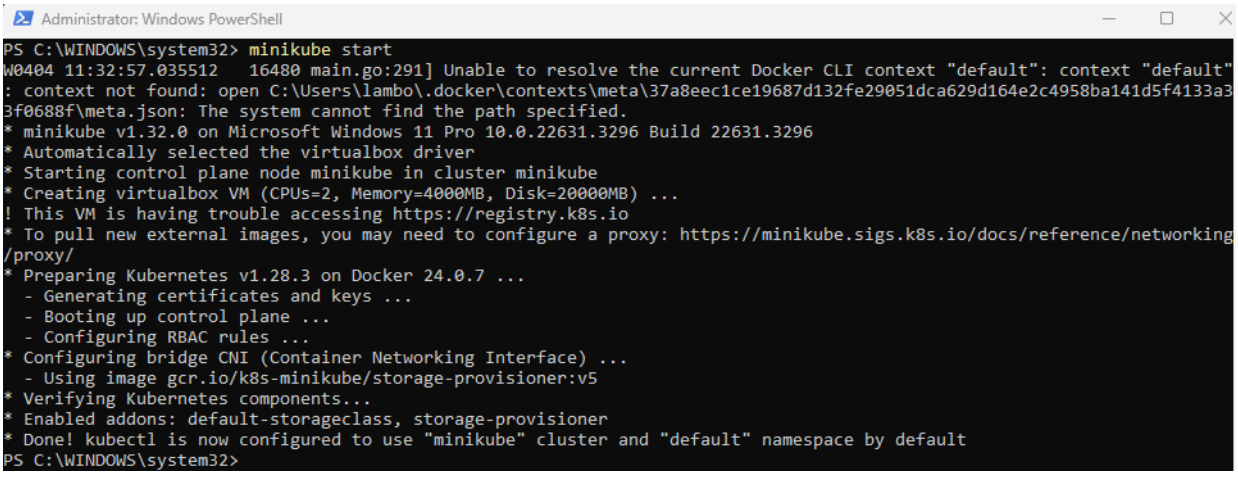
+ A single-instance Redis to store guestbook entries
+ Multiple web frontend instances



3.1. Start up the Redis Database

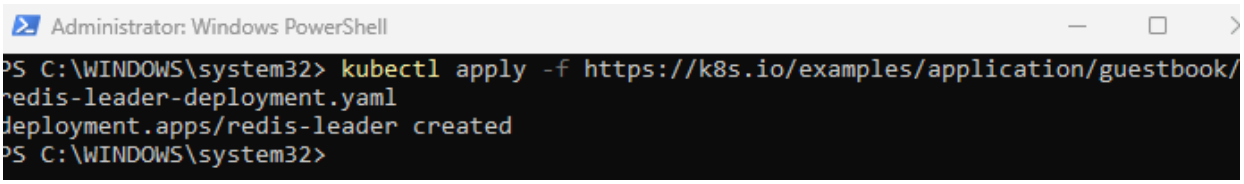
The guestbook application uses Redis to store its data.

a. Creating the Redis Deployment



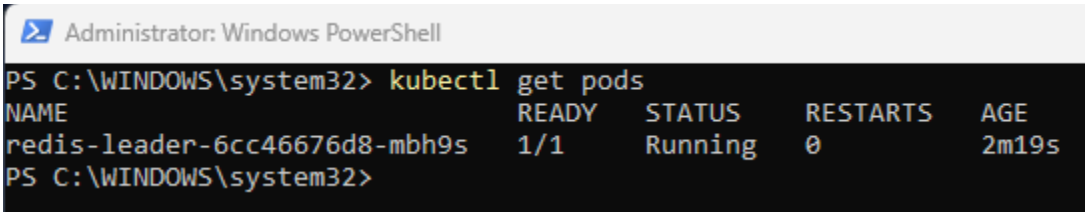
Create new cluster

- Apply the Redis Deployment from the redis-leader-deployment.yaml file:
  `\kubectl apply -f` https://k8s.io/examples/application/guestbook/redis-leader-deployment.yaml



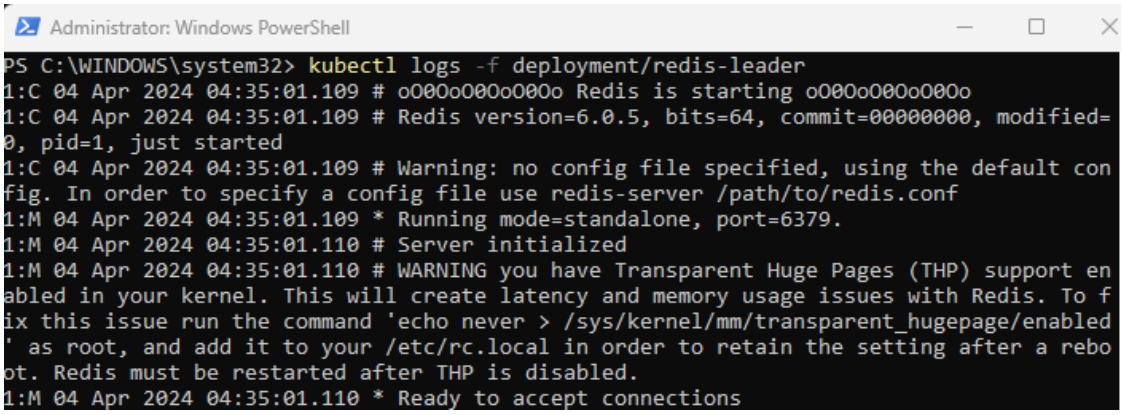
Apply the Redis Deployment from the redis-leader-deployment.yaml file

- Query the list of Pods to verify that the Redis Pod is running:
  `kubectl get pods`
  (take a screenshot)



Verify that the Redis Pod is running

- Run the following command to view the logs from the Redis leader Pod:
  `kubectl logs -f deployment/redis-leader`

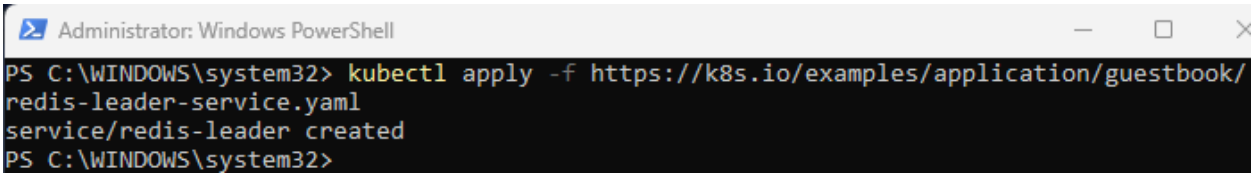


View the logs from the Redis leader Pod

b. Creating the Redis leader Service

- Apply the Redis Service from the following redis-leader-service.yaml file

  `kubectl apply -f` https://k8s.io/examples/application/guestbook/redis-leader-service.yaml

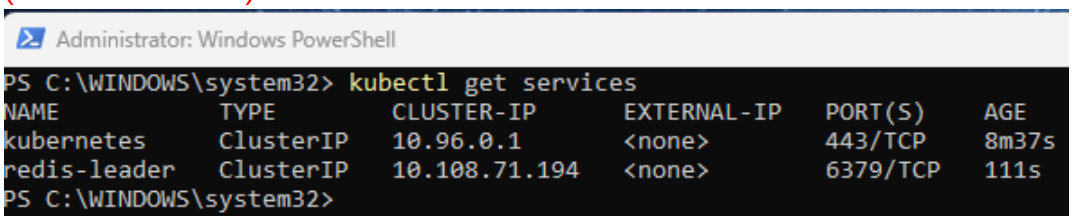


Apply the Redis Service from the following redis-leader-service.yaml file

- Query the list of Services to verify that the Redis Service is running:

  `kubectl get services`

  (take a screenshot)

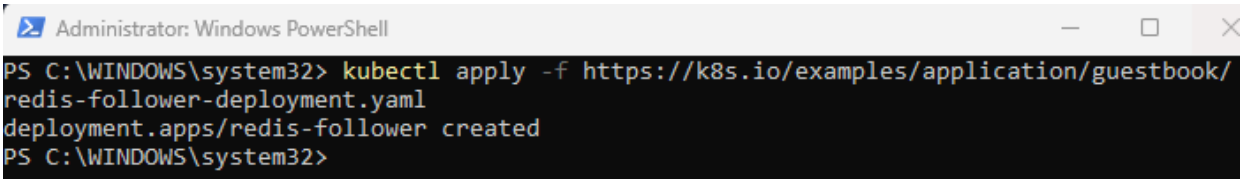


Query the list of Services to verify that the Redis Service is running

c. Set up Redis followers

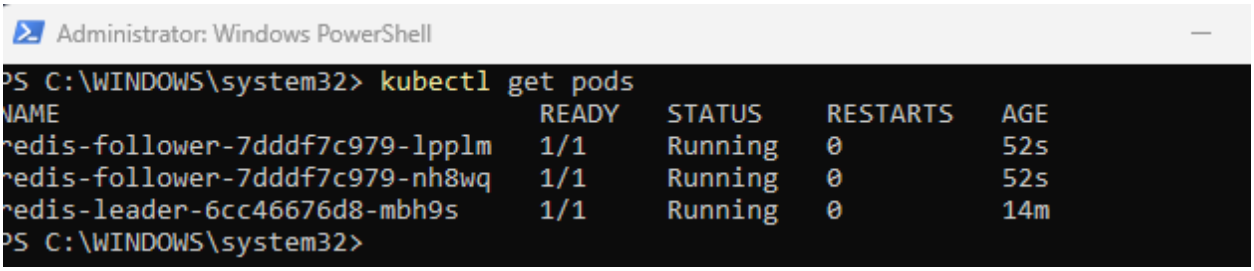
- Apply the Redis Deployment from the following redis-follower-deployment.yaml file:

  `kubectl apply -f` https://k8s.io/examples/application/guestbook/redis-follower-deployment.yaml



Apply the Redis Deployment from the following redis-follower-deployment.yaml file

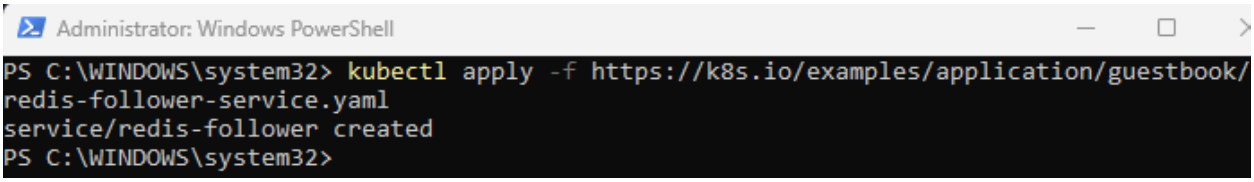
- Verify that the two Redis follower replicas are running by querying the list of Pods:
  `kubectl get pods`



Verify that the two Redis follower replicas are running by querying the list of Pods (take a screenshot)

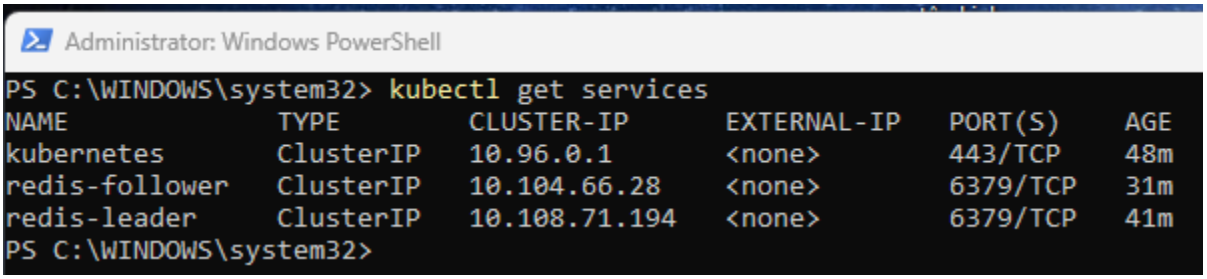
d. Creating the Redis follower service

- Apply the Redis Service from the following redis-follower-service.yaml file:
  `kubectl apply -f https://k8s.io/examples/application/guestbook/redis-follower-service.yaml`



Apply the Redis Service from the following redis-follower-service.yaml file

- Query the list of Services to verify that the Redis Service is running:
  `kubectl get services`
  (take a screenshot)

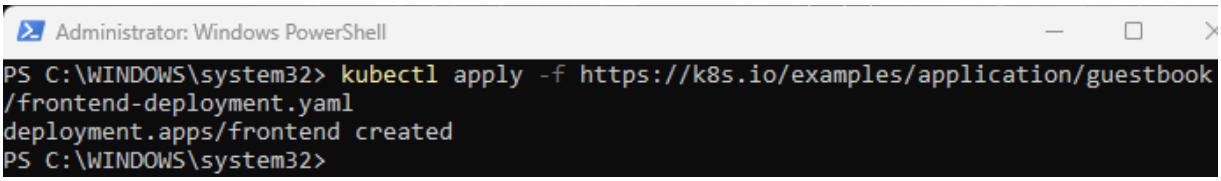


Query the list of Services to verify that the Redis Service is running

3.2. Set up and Expose the Guestbook Frontend

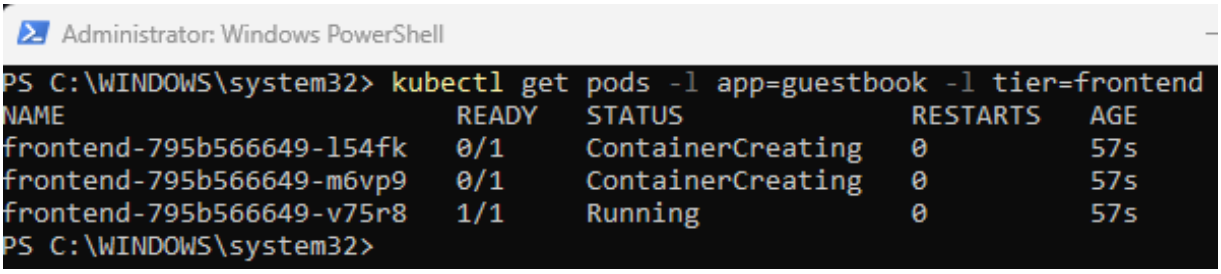
a. Creating the Guestbook Frontend Deployment

- Apply the frontend Deployment from the frontend-deployment.yaml file:
  `kubectl apply -f https://k8s.io/examples/application/guestbook/frontend-deployment.yaml`



Apply the frontend Deployment from the frontend-deployment.yaml file

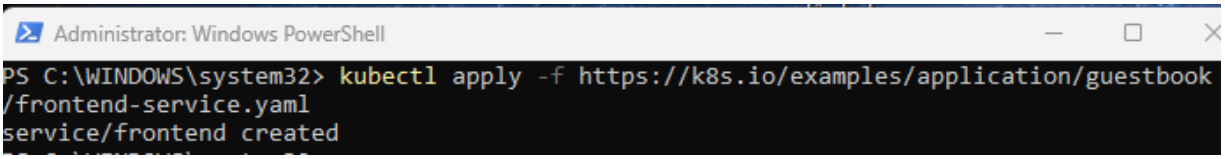
- Query the list of Pods to verify that the three frontend replicas are running:
  `kubectl get pods -l app=guestbook -l tier=frontend`



Query the list of Pods to verify that the three frontend replicas are running (take a screenshot)

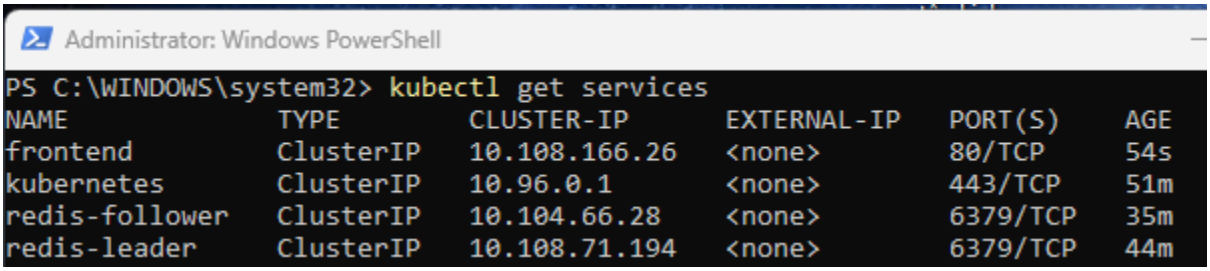
b. Creating the Frontend Service

- Apply the frontend Service from the frontend-service.yaml file:
  `kubectl apply -f https://k8s.io/examples/application/guestbook/frontend-service.yaml`



Apply the frontend Service from the frontend-service.yaml file

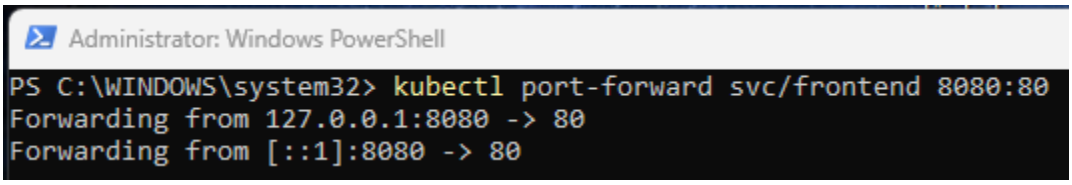
- Query the list of Services to verify that the frontend Service is running:
  `kubectl get services`



Query the list of Services to verify that the frontend Service is running (take a screenshot)

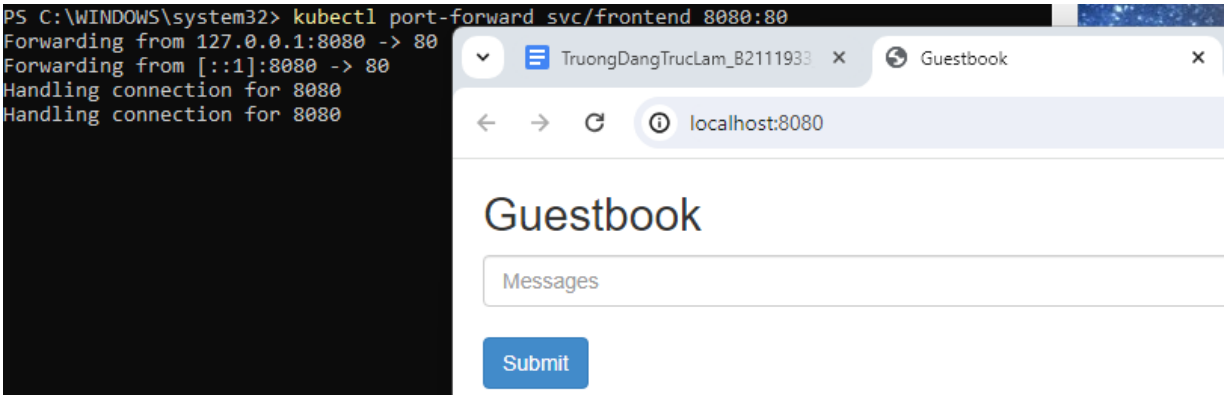
c. Viewing the Frontend Service via `kubectl port-forward`

- Run the following command to forward port 8080 on your local machine to port 80 on the service.
  `kubectl port-forward svc/frontend 8080:80`



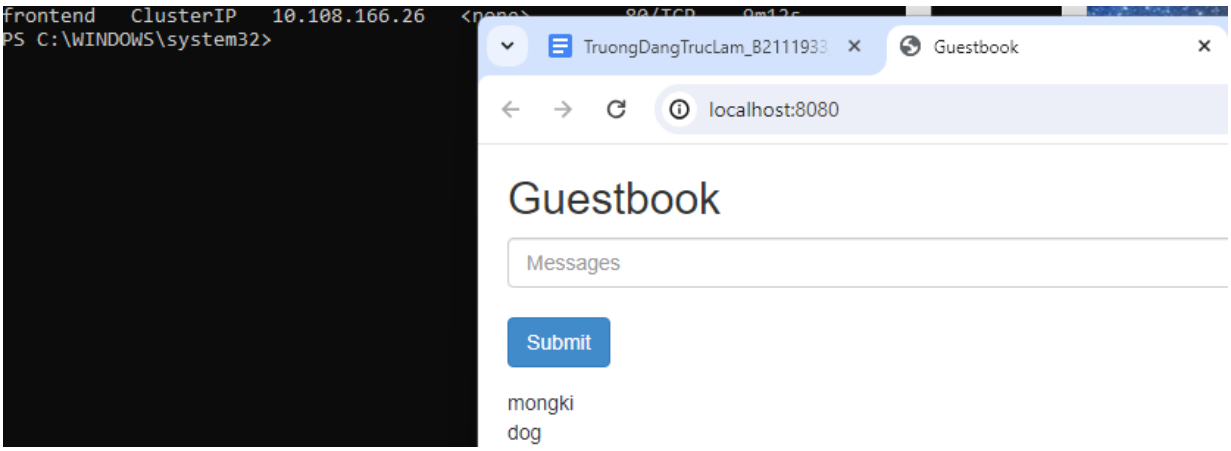
Forward port 8080 on your local machine to port 80 on the service

- Load the page http://localhost:8080 in your browser to view your guestbook.



Display the guestbook

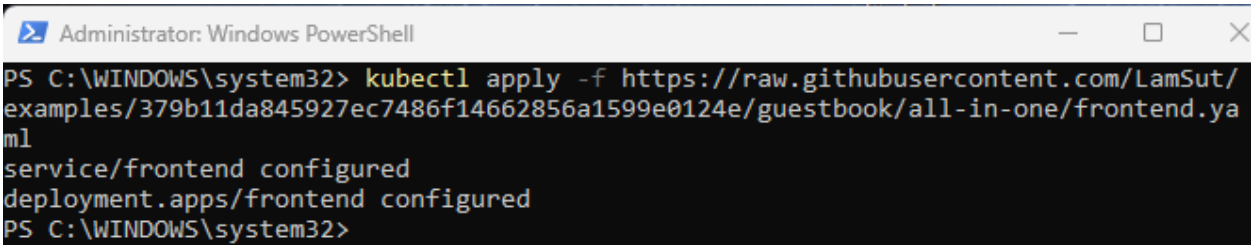
(take a screenshot)



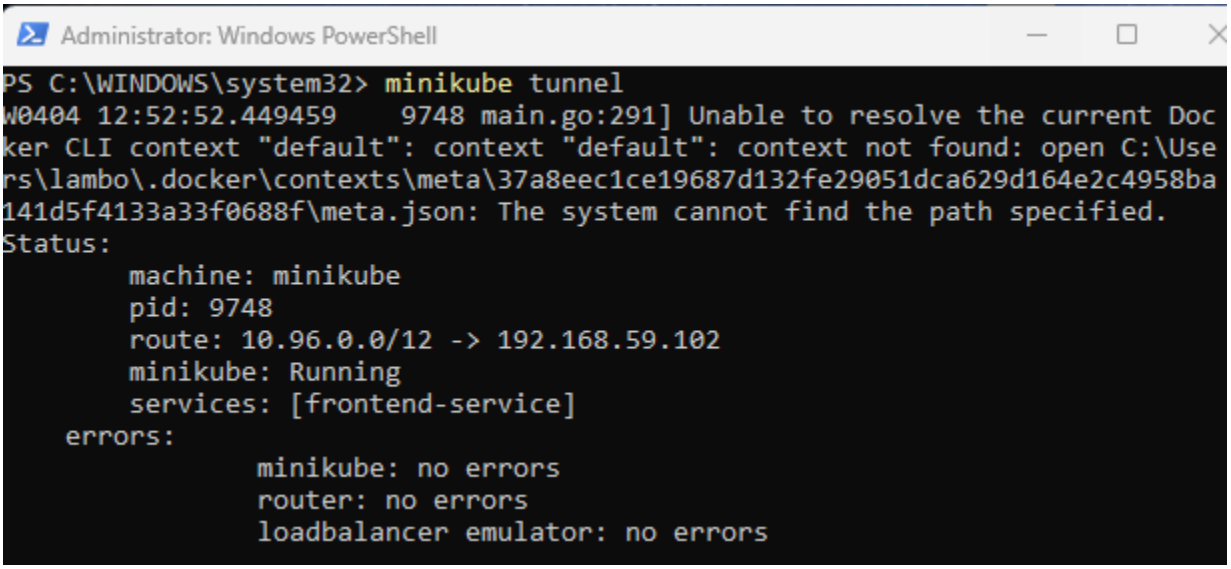
Try the Submit button

d. Viewing the Frontend Service via LoadBalancer

- If you deployed the `frontend-service.yaml` manifest with type: LoadBalancer you need to find the IP address to view your Guestbook.

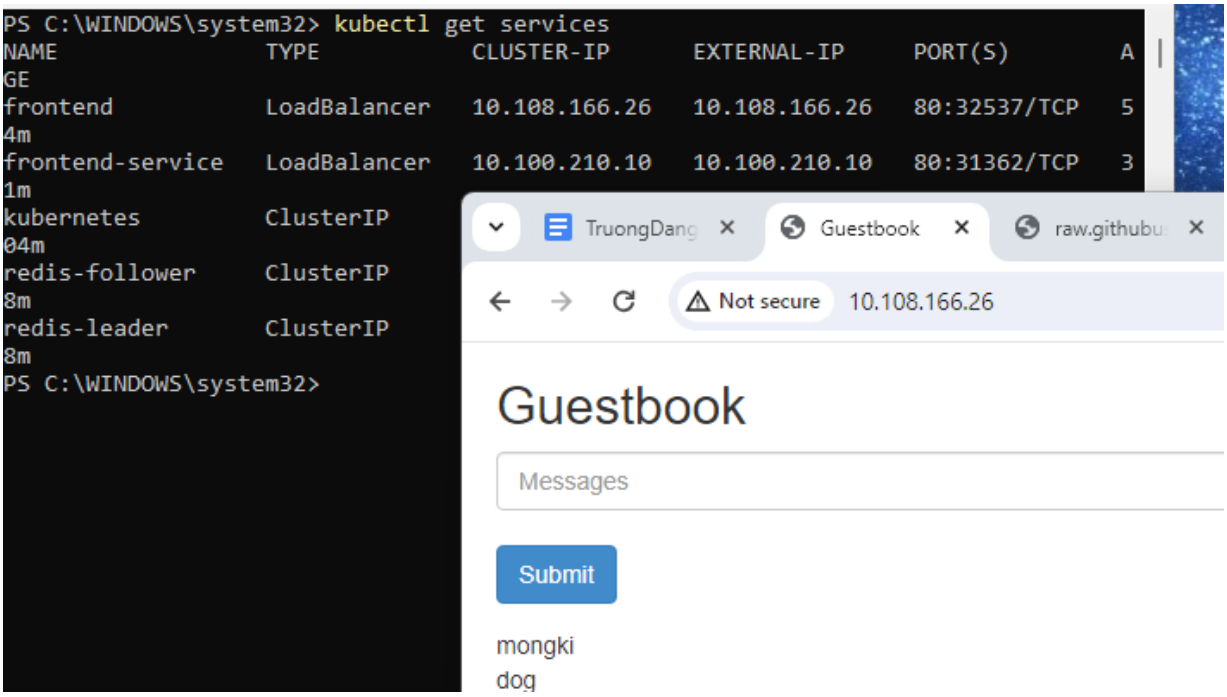


This is the link of `frontend-service.yaml` manifest with type: LoadBalancer



Expose the external IP directly to any program running on the host operating system

- Copy the external IP address, and load the page in your browser to view your guestbook.

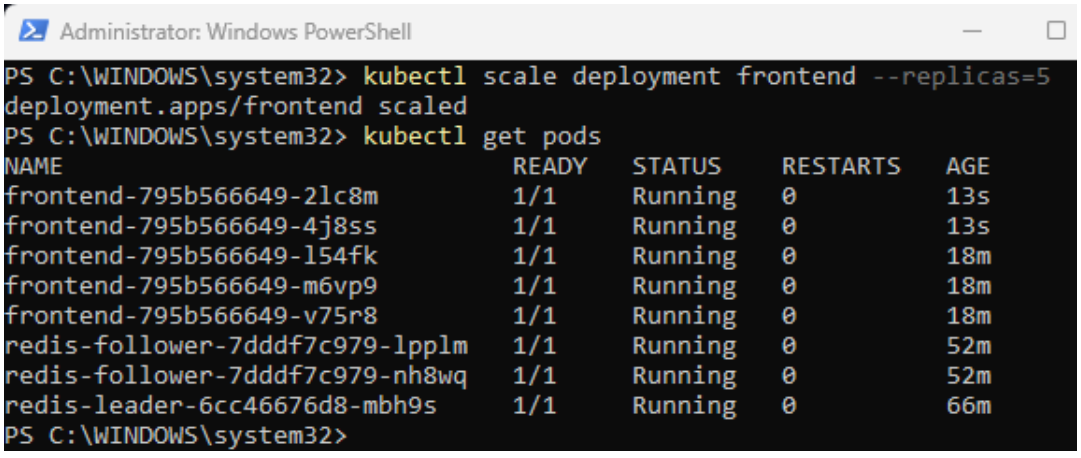


Display Guestbook via LoadBalancer

3.3. Scale the Web Frontend

- Run the following command to scale up the number of frontend Pods:

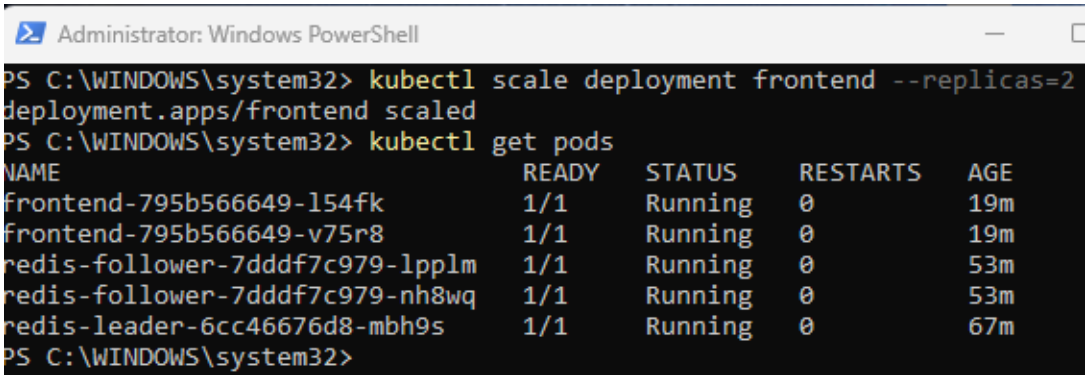
```
kubectl scale deployment frontend --replicas=5
kubectl get pods
```



Scale up the number of frontend Pods

- Run the following command to scale down the number of frontend Pods:

```
kubectl scale deployment frontend --replicas=2
kubectl get pods
```



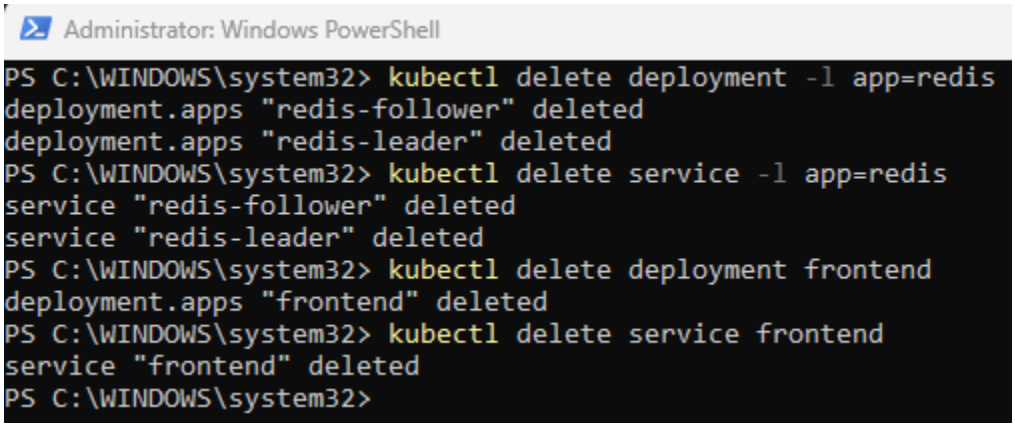
Scale down the number of frontend Pods

(take a screenshot)

3.4. Cleaning up

- Run the following commands to delete all Pods, Deployments, and Services.

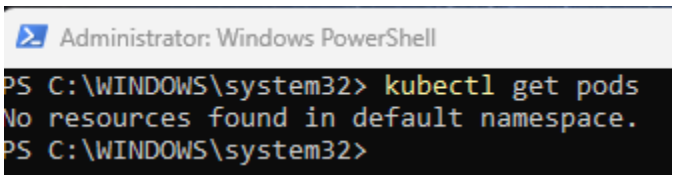
```
kubectl delete deployment -l app=redis
kubectl delete service -l app=redis
kubectl delete deployment frontend
kubectl delete service frontend
```



Delete all Pods, Deployments, and Services

- Query the list of Pods to verify that no Pods are running

```
kubectl get pods
```



There is no Pod left

(take a screenshot)

---END---