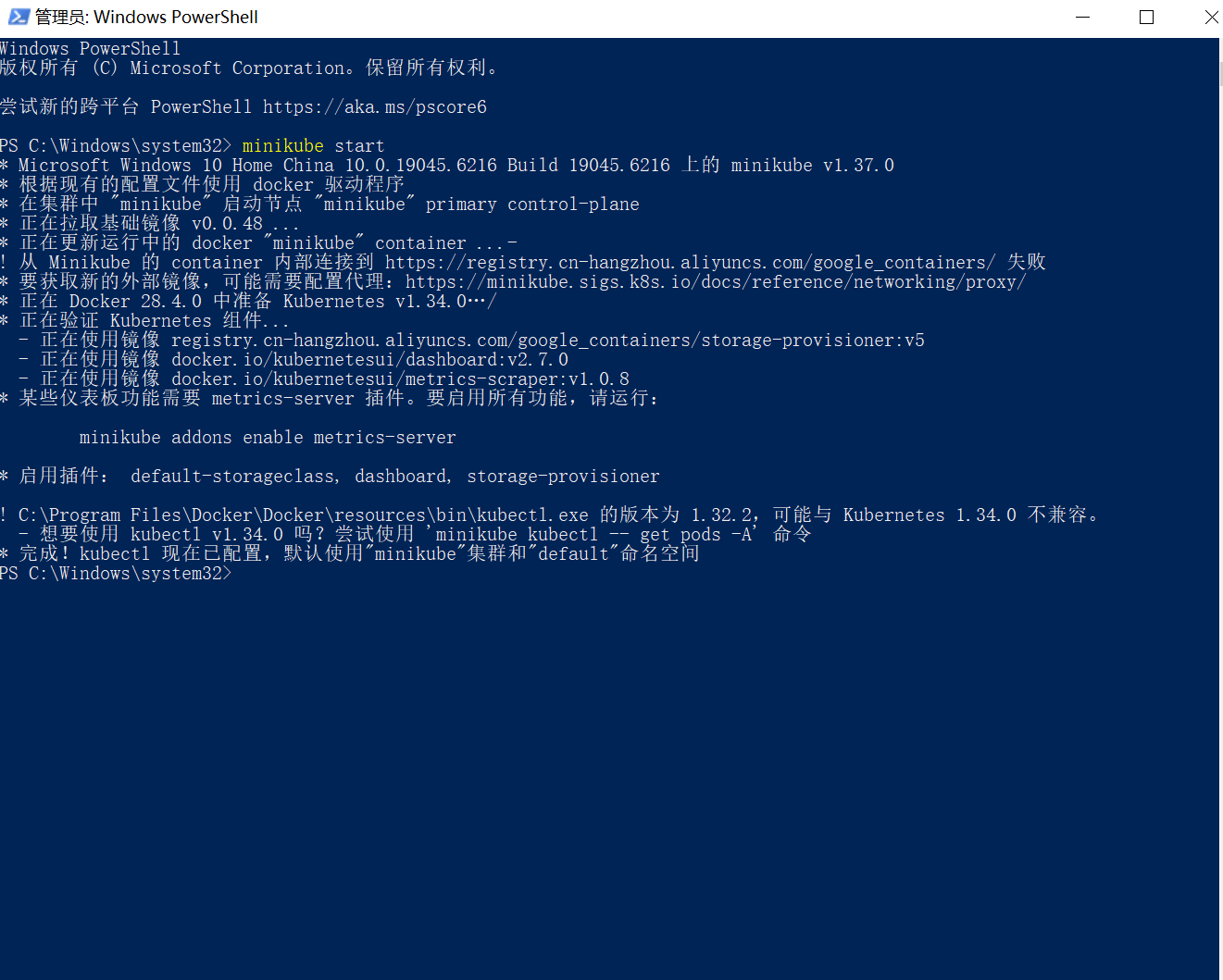
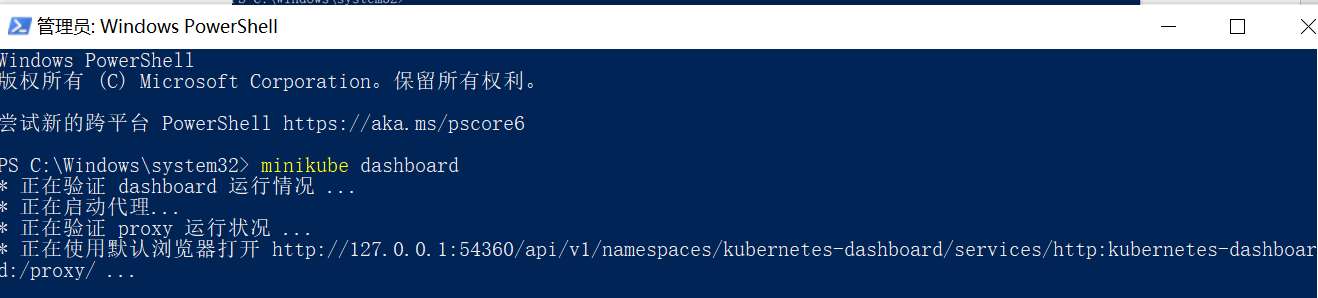
1. **Hello Minikube**

Run PowerShell as administrator and run minikube start



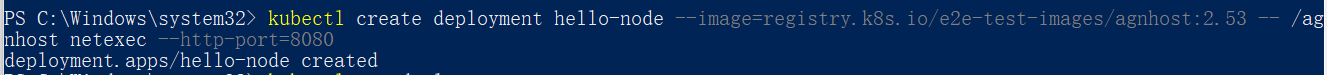
**Open the Dashboard**

a. Open the second admin powershell and run Minikube Dashboard

 图形用户界面, 文本, 应用程序, Word

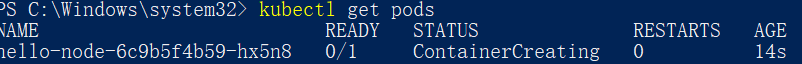
AI 生成的内容可能不正确。

**Create a Deployment**

a. Use the kubectl create command to create a Deployment that manages a Pod. The Pod runs a Container based on the provided Docker image. 

b. View the Deployment:



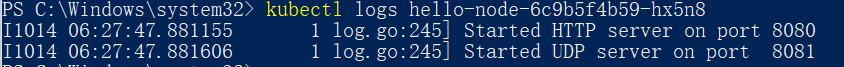
c. View the Pod: 

d. View cluster events: 图形用户界面, 文本

AI 生成的内容可能不正确。

e. View the kubectl configuration: 文本

AI 生成的内容可能不正确。

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

**Create a Service**

a. Expose the Pod to the public internet using the kubectl expose command: 

b. View the Service you created:

文本

AI 生成的内容可能不正确。

c. Run the following command: 日程表

AI 生成的内容可能不正确。 图形用户界面, 应用程序

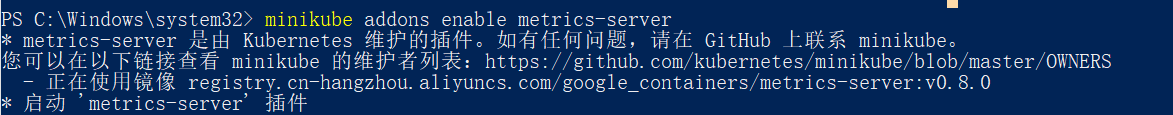
AI 生成的内容可能不正确。

**Enable addons**

1. List the currently supported addons:

图片包含 文本

AI 生成的内容可能不正确。

1. Enable an addon, for example, metrics-server:
2. View the Pod and Service you created by installing that addon: 电脑屏幕的照片上有文字

   AI 生成的内容可能不正确。
3. Check the output from metrics-server:



1. Disable metrics-server:



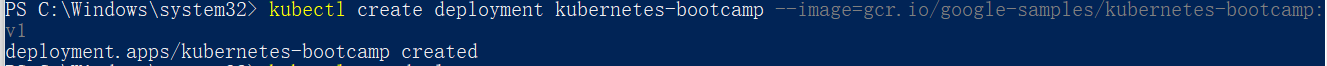
**Clean up**

文本

AI 生成的内容可能不正确。

1. **Using kubectl to Create a Deployment**

Deploy an app

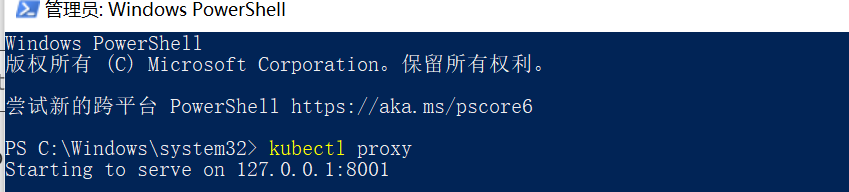
****

 List my deployments use the kubectl get deployments command:

文本

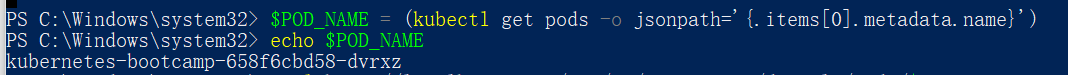
AI 生成的内容可能不正确。

Open a second terminal window to run the proxy.

 图形用户界面, 文本, 应用程序, 电子邮件

AI 生成的内容可能不正确。

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



We can access the Pod through the proxied API

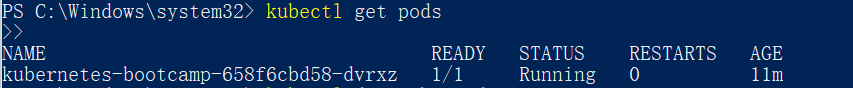
图形用户界面, 文本, 应用程序, Word

AI 生成的内容可能不正确。

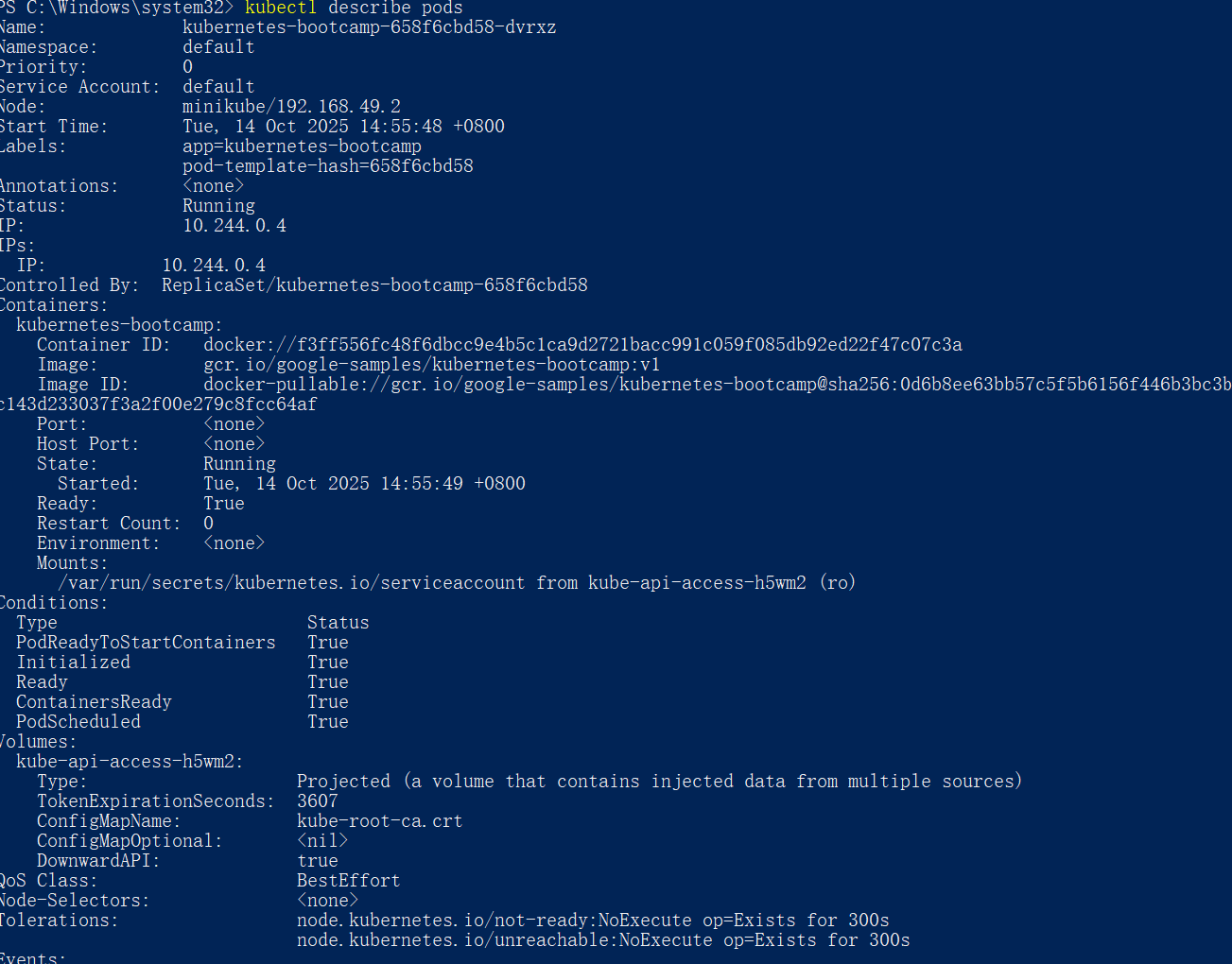
1. **Viewing Pods and Nodes**

Check application configuration

a. Let's verify that the application we deployed in the previous scenario is running. We'll use the kubectl get command and look for existing Pods:



b. Next, to view what containers are inside that Pod and what images are used to build those containers we run the kubectl describe pods command:



Show the app in the terminal

1. Open a new terminal window, and in that new terminal, run:

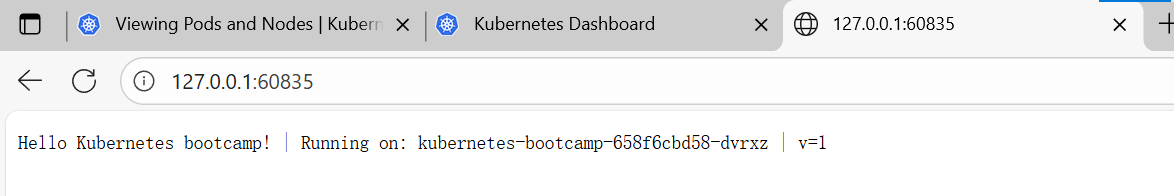
kubectl proxy

1. 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 environment variable:

export POD\_NAME="**$(**kubectl get pods -o go-template --template '{{range .items}}{{.metadata.name}}{{"\n"}}{{end}}'**)**"

echo Name of the Pod: $POD\_NAME

Then we can see:



**Executing commands on the container**

1. We can execute commands directly on the container once the Pod is up and running. For this, we use the exec subcommand and use the name of the Pod as a parameter. Let’s list the environment variables:

图形用户界面, 文本

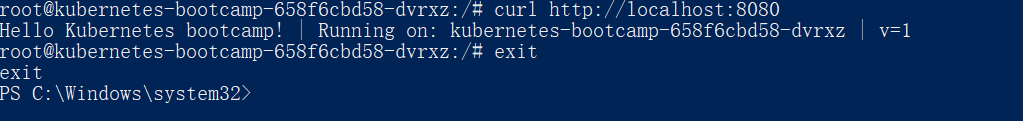
AI 生成的内容可能不正确。

1. Again, it's worth mentioning that the name of the container itself can be omitted since we only have a single container in the Pod.

Next let’s start a bash session in the Pod’s container:

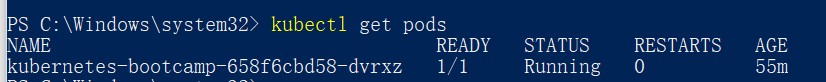


1. We have now an open console on the container where we run our NodeJS application. The source code of the app is in the server.js file: 文本

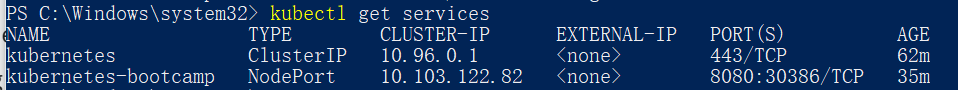
   AI 生成的内容可能不正确。
2. check that the application is up by running a curl command: 
3. **Using a Service to Expose Your App**

Creating a new Service

1. Let’s verify that our application is running. We’ll use the kubectl get command and look for existing Pods:



1. Next, let’s list the current Services from our cluster:



1. To expose the deployment to external traffic, we'll use the kubectl expose command with the --type=NodePort option:

kubectl expose deployment/kubernetes-bootcamp --type="NodePort" --port 8080

1. To find out what port was opened externally (for the type: NodePort Service) we’ll run the describe service subcommand:

文本

AI 生成的内容可能不正确。

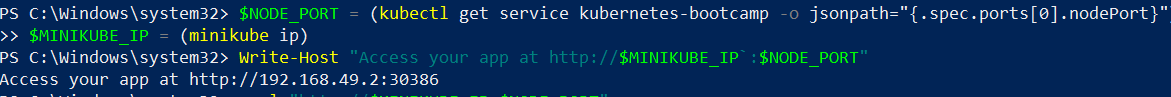
1. In PowerShell, you don't use export, just use variables:

$NODE\_PORT = (kubectl get service kubernetes-bootcamp -o jsonpath="{.spec.ports[0].nodePort}")

$MINIKUBE\_IP = (minikube ip)

Write-Host "Access your app at http://$MINIKUBE\_IP`:$NODE\_PORT"

minikube service kubernetes-bootcamp –url

 图形用户界面, 文本

AI 生成的内容可能不正确。 图形用户界面, 文本, 应用程序

AI 生成的内容可能不正确。

Using labels

1. View the Labels that are automatically created by the Deployment:

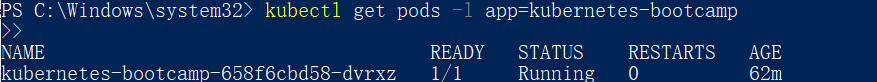
kubectl describe deployment kubernetes-bootcamp

图形用户界面, 文本

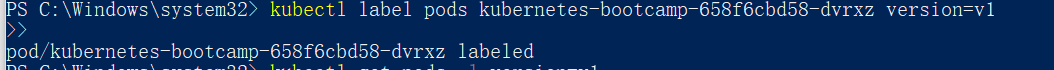
AI 生成的内容可能不正确。

1. Querying a pod with a label:

kubectl get pods -l app=kubernetes-bootcamp

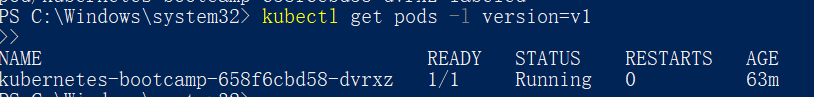


1. Add a new Label to the pod(Version information):

PS C:\Windows\system32> kubectl label pods kubernetes-bootcamp-658f6cbd58-dvrxz version=v1

1. Verify:

kubectl get pods -l version=v1



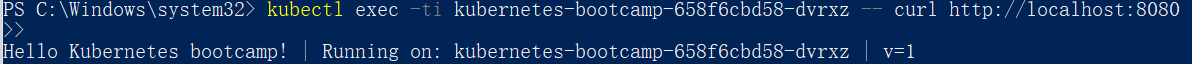
Deleting a service

kubectl delete service -l app=kubernetes-bootcamp

kubectl get services 文本

AI 生成的内容可能不正确。

Access to the app's external NodePort is invalid, but it is still accessible inside the pod:



**Running Multiple Instances of Your App**

1. Confirm the current status of the Deployment

文本

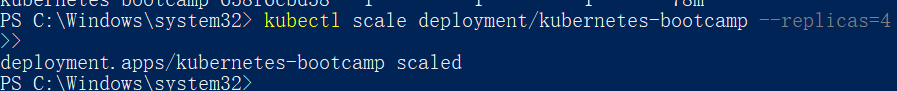
AI 生成的内容可能不正确。

1. View ReplicaSet

文本

AI 生成的内容可能不正确。

1. Expansion Deployment（Scale Out）



1. View the scaling results

电脑萤幕画面

AI 生成的内容可能不正确。

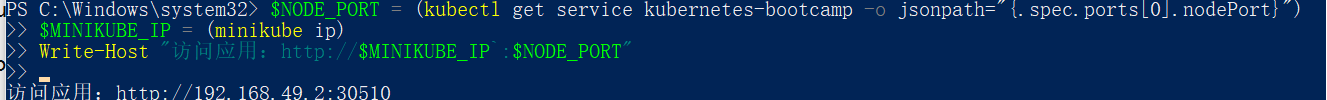
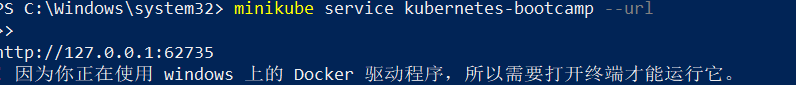
1. Service Load Balancing

Make sure the Deployment has been exposed by Service

文本

AI 生成的内容可能不正确。

Get the NodePort under PowerShell

  图形用户界面, 应用程序

AI 生成的内容可能不正确。

1. Shrinking Deployment（Scale Down）

图形用户界面, 文本

AI 生成的内容可能不正确。

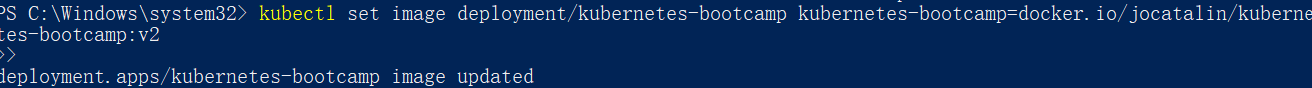
**Performing a Rolling Update**

1. Review the current Deployment and Pods

图形用户界面, 文本

AI 生成的内容可能不正确。

1. Perform rolling updates



Check the update status:

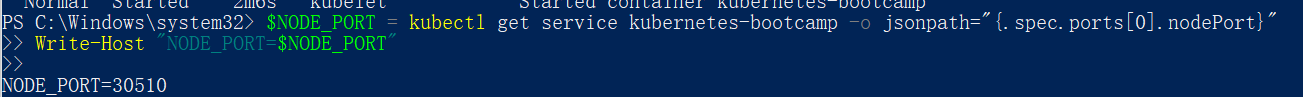
See if the new pod is already running:

图形用户界面, 文本

AI 生成的内容可能不正确。

1. Access the updated app through Service

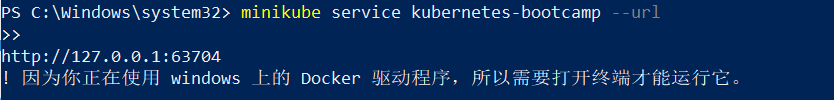
To get the NodePort:

fetch minikube IP：

文本

AI 生成的内容可能不正确。

Access the app:

 图形用户界面, 文本, 应用程序

AI 生成的内容可能不正确。

1. Rolling-back

图形用户界面, 文本

AI 生成的内容可能不正确。

1. Clean up resources

