

# **Introduction to Cloud Computing Report**

## **Lab 6: Kubernetes Basics Tutorial**

Name: Yin Yuang

Student ID: 202383930027

Class: Software Engineering Class 1

Introduction to Cloud Computing  
(Autumn,2025)

Nanjing University of Information Science and Technology, China  
Waterford Institute

# I. Object

The objective of this experiment is to understand and practice the fundamental operations of Kubernetes in a local environment using Minikube. Through hands-on experimentation, this lab focuses on deploying containerized applications, managing Pods and Deployments, exposing applications through Kubernetes Services, scaling applications to multiple replicas, and performing rolling updates and rollback operations. By completing this experiment, students gain practical experience with core Kubernetes concepts, including container orchestration, service discovery, scalability, and fault recovery, as well as proficiency in using the `kubectl` command-line tool.

## II. Content

This experiment mainly includes the following tasks:

- Initializing a local Kubernetes cluster and dashboard using Minikube
- Creating and managing application Deployments using `kubectl`
- Inspecting Pods, Nodes, and application logs to verify runtime status
- Accessing applications through `kubectl proxy` and Kubernetes Services
- Exposing applications externally using NodePort and LoadBalancer Services
- Scaling applications to multiple replicas and verifying service load balancing
- Performing rolling updates, analyzing update failures, and executing rollback operations

## III. Results

This section presents the experimental results obtained from deploying and managing applications on a local Kubernetes cluster using Minikube. The results are organized according to the main experimental objectives, including cluster initialization, application deployment, service exposure, scaling, and rolling updates.

### 1. Hello Minikube

The experiment began with initializing a local Kubernetes cluster using Minikube to provide a controlled environment for subsequent deployment and management tasks.

#### 1.1 Starting the Minikube Cluster

The local Kubernetes cluster was initialized using the following command:

```
minikube start
```

After execution, Minikube successfully started the control-plane node and configured `kubectl` to use the Minikube context.

```

C:\Users\yin>minikube start
* Microsoft Windows 11 Pro 10.0.22631.6199 Build 22631.6199 上的 minikube v1.37.0
! 指定的 Kubernetes 版本 1.34.1 较新, 比支持的最新版本 v1.34.0 还要新。请使用 'minikube config defaults kubernetes-version'
! 在 Kubernetes 版本列表中找到指定的 Kubernetes 版本 1.34.1
* 在互联网上搜索 Kubernetes 版本...
* 在 GitHub 版本列表找到了 Kubernetes 版本 1.34.1。
* 根据现有的配置文件使用 docker 驱动程序
* 在集群中 "minikube" 启动节点 "minikube" primary control-plane
* 正在拉取基础镜像 v0.0.48 ...
* 正在更新运行中的 docker "minikube" container .../
! 从 Minikube 的 container 内部连接到 https://registry.k8s.io/ 失败
* 要获取新的外部镜像, 可能需要配置代理: https://minikube.sigs.k8s.io/docs/reference/networking/proxy/
! The image 'registry.k8s.io/k8s-minikube/storage-provisioner:v5' was not found; unable to add it to cache.
* 正在 Docker 28.4.0 中准备 Kubernetes v1.34.1...
X 无法加载缓存的镜像: LoadCachedImages: removing image: remove image docker: docker rmi registry.k8s.io/kube-controller-man
Process exited with status 1
stdout:

stderr:
Error response from daemon: conflict: unable to remove repository reference "registry.k8s.io/kube-controller-manager:v1.34.1" - container 9f0df895a97f is using its referenced image c80c8dbafe7d

* 正在验证 Kubernetes 组件...
- 使用镜像 registry.k8s.io/k8s-minikube/storage-provisioner:v5 (全局镜像仓库)
* 启用插件: storage-provisioner, default-storageclass
* 完成! kubectl 现在已配置, 默认使用"minikube"集群和"default"命名空间

```

Figure 1: Successful startup of the Minikube cluster

## 1.2 Opening the Kubernetes Dashboard

To visualize and manage Kubernetes resources, the Kubernetes Dashboard was launched in a new terminal:

```
minikube dashboard
```

This command enabled the dashboard addon and created a temporary proxy, allowing access to the dashboard from a web browser.

```

C:\Users\yin>minikube dashboard
* 正在验证 dashboard 运行情况 ...
* 正在启动代理 ...
* 正在验证 proxy 运行状况 ...

```

Figure 2: Kubernetes Dashboard accessed through Minikube

## 1.3 Creating a Deployment

A Deployment named hello-node was created using a test container image that provides a web server on port 8080:

```
kubectl create deployment hello-node --image=registry.k8s.io/e2e-test-images/agnhost:2.53 --
/agnhost netexec --http-port=8080
```

## 1.4 Verifying Deployment and Pod Status

The status of the Deployment and Pod was verified using the following commands:

```
kubectl get deployments
```

```
kubectl get pods
```

The output confirmed that the Pod was in the Running state.

```

C:\Users\yin>kubectl get deployments
NAME          READY  UP-TO-DATE  AVAILABLE  AGE
hello-node    1/1    1           1          2m13s

C:\Users\yin>kubectl get pods
NAME                                READY  STATUS    RESTARTS  AGE
hello-node-6c9b5f4b59-7nj4r        1/1    Running   0          2m15s

```

Figure 3: Deployment and Pod status

## 1.5 Viewing Application Logs

The logs of the running container were obtained using:

```
kubectl logs hello-node-<pod-name>
```

The logs showed that the HTTP server started successfully on port 8080.

```
C:\Users\yin>kubectl logs hello-node-6c9b5f4b59-7nj4r
I1215 11:10:11.177990      1 log.go:245] Started HTTP server on port 8080
I1215 11:10:11.178574      1 log.go:245] Started UDP server on port 8081
```

Figure 4: Application logs of the hello-node Pod

## 1.6 Exposing the Application as a Service

To allow external access, the Deployment was exposed as a Service of type LoadBalancer:

```
kubectl expose deployment hello-node --type=LoadBalancer --port=8080
```

The Service was accessed using:

```
minikube service hello-node
```

This command opened the application in the default web browser.

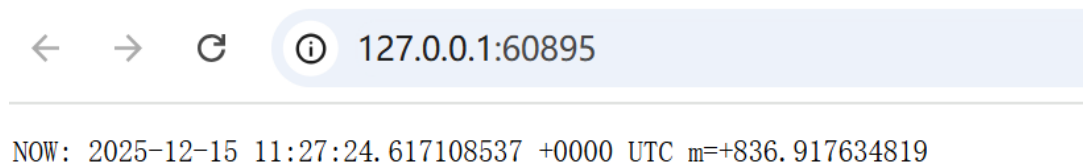


Figure 5: Hello Minikube application accessed in browser

## 2. Using kubectl to Create a Deployment

### 2.1 Cluster Verification

After starting the cluster, the status of the Kubernetes node was verified using the following command:

```
kubectl get nodes
```

The output confirms that the Minikube node is in the Ready state, indicating that the cluster is operational.

```
C:\Users\yin>kubectl get nodes
NAME        STATUS    ROLES    AGE   VERSION
minikube    Ready     control-plane  9h   v1.34.1
```

Figure 6: Kubernetes node status

### 2.2 Application Deployment

A Kubernetes Deployment named `kubernetes-bootcamp` was created using the following command:

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

This command instructs Kubernetes to create a Deployment using the specified container image. Kubernetes automatically schedules the Pod on the available node and ensures its availability.

```
C:\Users\yin>kubectl create deployment hello-node --image=registry.k8s.io/e2e-test-images/agnhost:2.53 -- /
agnhost netexec --http-port=8080
deployment.apps/hello-node created
```

Figure 7: Creating the Kubernetes Deployment

The Deployment status was verified using:

```
kubectl get deployments
```

```
C:\Users\yin>kubectl get deployments
NAME                READY  UP-TO-DATE  AVAILABLE  AGE
kubernetes-bootcamp  1/1    1           1          2m27s
```

Figure 8: Deployment status showing one running replica

### 2.3 Accessing the Application Using kubectl Proxy

By default, Pods run in an internal Kubernetes network and are not accessible externally. To access the application, the `kubectl proxy` command was used to create a temporary proxy between the local machine and the Kubernetes API server.

```
kubectl proxy
```

This proxy enables access to Kubernetes services and Pods via localhost.

```
C:\Users\yin>kubectl proxy
Starting to serve on 127.0.0.1:8001
```

Figure 9: kubectl proxy running

### 2.4 Pod Access and Verification

The name of the running Pod was obtained using:

```
kubectl get pods
```

The application was then accessed through the Kubernetes API proxy using the following command:

```
curl http://localhost:8001/api/v1/namespaces/default/pods/
<POD_NAME>:8080/proxy/
```

The successful response confirms that the application is running correctly inside the Pod.

```

PS C:\Users\yin> $env:POD_NAME="kubernetes-bootcamp-658f6cbd58-pmdmz"
PS C:\Users\yin> curl http://localhost:8001/api/v1/namespaces/default/pods/($env:POD_NAME):8080/proxy/

StatusCode      : 200
StatusDescription : OK
Content         : Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-658f6cbd58-pmdmz | v=1

RawContent      : HTTP/1.1 200 OK
                  Audit-Id: ba89a5ad-d71e-467c-b343-e87bf6bc9a5c
                  Transfer-Encoding: chunked
                  Cache-Control: no-cache, private
                  Content-Type: text/plain
                  Date: Mon, 15 Dec 2025 12:22:48 GMT

                  Hello Ku...
Forms           : {}
Headers        : {[Audit-Id, ba89a5ad-d71e-467c-b343-e87bf6bc9a5c], [Transfer-Encoding, chunked], [Cache-Control, no
                  -cache, private], [Content-Type, text/plain]...}
Images         : {}
InputFields    : {}
Links          : {}
ParsedHtml     : System.__ComObject
RawContentLength : 84

```

Figure 10: Successful access to the application through API proxy

### 3.Viewing Pods and Nodes

To verify that the application Pod is running, the following command was executed:

```
kubectl get pods
```

The output shows that the Pod is in the Running state, indicating that the application container has been successfully started.

```

C:\Users\yin>kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-658f6cbd58-pmdmz 1/1     Running   0          82m

```

Figure 11: Viewing Running Pods

To view detailed information about the Pod, including its IP address, container image, and events, the following command was used:

```
kubectl describe pod <pod-name>
```

```

C:\Users\yin>kubectl describe pod kubernetes-bootcamp-658f6cbd58-pdmz
Name:          kubernetes-bootcamp-658f6cbd58-pdmz
Namespace:     default
Priority:       0
Service Account: default
Node:          minikube/192.168.49.2
Start Time:    Mon, 15 Dec 2025 19:43:33 +0800
Labels:        app=kubernetes-bootcamp
               pod-template-hash=658f6cbd58
Annotations:   <none>
Status:        Running
IP:            10.244.0.8
IPs:           IP: 10.244.0.8
Controlled By: ReplicaSet/kubernetes-bootcamp-658f6cbd58
Containers:
  kubernetes-bootcamp:
    Container ID:  docker://15c8f1f42feb866d5a0be7dec1d144a94e495dc820d6d766d5ee9dc60bf610eb
    Image:         gcr.io/google-samples/kubernetes-bootcamp:v1
    Image ID:      docker-pullable://gcr.io/google-samples/kubernetes-bootcamp@sha256:0d6b8ee63bb57c5f5b6156f446b3bc3b3c143d2336
    Port:          <none>
    Host Port:     <none>
    State:         Running
      Started:     Mon, 15 Dec 2025 19:43:58 +0800
    Ready:         True
    Restart Count: 0
    Environment:   <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-n5fvw (ro)
Conditions:
  Type                                Status
  PodReadyToStartContainers           True
  Initialized                         True
  Ready                              True
  ContainersReady                     True
  PodScheduled                        True
Volumes:
  kube-api-access-n5fvw:
    Type:              Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds: 3607
    ConfigMapName:      kube-root-ca.crt
    Optional:           false
    DownwardAPI:        true
QoS Class:               BestEffort
Node-Selectors:          <none>
Tolerations:             node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                        node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:                   <none>

```

Figure 12: Detailed Pod Information

To inspect the Nodes in the cluster, the following command was executed:

```
kubectl get nodes
```

This confirms that the Pod is scheduled on the Minikube Node.

```

C:\Users\yin>kubectl get nodes
NAME        STATUS    ROLES    AGE   VERSION
minikube    Ready     control-plane  10h   v1.34.1

```

Figure 13: Viewing Cluster Nodes

### 3.1 Accessing the Pod Using kubectl Proxy

Pods run in an isolated internal network and are not directly accessible from outside the Kubernetes cluster. To access the application for debugging purposes, a proxy was created using the following command in a separate terminal:

```
kubectl proxy
```

The Pod name was retrieved and stored in an environment variable, and the application was accessed through the Kubernetes API using a curl request.

```
curl http://localhost:8001/api/v1/namespaces/default/pods/<pod-name>:8080/proxy/
```

```

PS C:\Users\yin> $env:POD_NAME="kubernetes-bootcamp-658f6cbd58-pmdmz"
PS C:\Users\yin> curl http://localhost:8001/api/v1/namespaces/default/pods/($env:POD_NAME):8080/proxy/

StatusCode      : 200
StatusDescription : OK
Content         : Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-658f6cbd58-pmdmz | v=1
RawContent      : HTTP/1.1 200 OK
                  Audit-Id: ba89a5ad-d71e-467c-b343-e87bf6bc9a5c
                  Transfer-Encoding: chunked
                  Cache-Control: no-cache, private
                  Content-Type: text/plain
                  Date: Mon, 15 Dec 2025 12:22:48 GMT
                  Hello Ku...
Forms           : {}
Headers         : {[Audit-Id, ba89a5ad-d71e-467c-b343-e87bf6bc9a5c], [Transfer-Encoding, chunked], [Cache-Control, no
                  -cache, private], [Content-Type, text/plain]...}
Images          : {}
InputFields     : {}
Links           : {}
ParsedHtml      : System.__ComObject
RawContentLength : 84

```

Figure 14: Accessing Pod via kubectl Proxy

### 3.2 Executing Commands Inside the Pod

To further inspect the running container, commands were executed directly inside the Pod using:

```
kubectl exec -it <pod-name> -- bash
```

Inside the container, the application was tested by sending a request to the local service:

```
curl http://localhost:8080
```

This confirmed that the application was running correctly inside the Pod.

```

C:\Users\yin>kubectl exec -it kubernetes-bootcamp-658f6cbd58-pmdmz -- bash
root@kubernetes-bootcamp-658f6cbd58-pmdmz:/# curl http://localhost:8080
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-658f6cbd58-pmdmz | v=1

```

Figure 15: Executing Commands Inside the Pod

## 4. Using a Service to Expose Your App

### 4.1 Checking Running Pods

First, the running Pods in the cluster were verified to ensure that the application was successfully deployed.

```
kubectl get pods
```

```

C:\Users\yin>kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-658f6cbd58-pmdmz 1/1     Running   0           82m

```

Figure 16: Running Pods in the Cluster

### 4.2 Creating a Service

To expose the application to external traffic, a NodePort Service was created using the following command:

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

This command created a Service that forwards external traffic to the application Pod.

```

C:\Users\yin>kubectl expose deployment/kubernetes-bootcamp --type=NodePort --port 8080
service/kubernetes-bootcamp exposed

```

Figure 17: Creating a NodePort Service



### 4.3 Inspecting the Service

The details of the Service were examined to obtain the assigned NodePort:

```
kubectl describe services/kubernetes-bootcamp
```

```
C:\Users\yin>kubectl describe services/kubernetes-bootcamp
Name:                kubernetes-bootcamp
Namespace:           default
Labels:              app=kubernetes-bootcamp
Annotations:         <none>
Selector:            app=kubernetes-bootcamp
Type:               NodePort
IP Family Policy:    SingleStack
IP Families:         IPv4
IP:                 10.111.141.233
IPs:                10.111.141.233
Port:               <unset> 8080/TCP
TargetPort:         8080/TCP
NodePort:           <unset> 30369/TCP
Endpoints:          10.244.0.8:8080
Session Affinity:    None
External Traffic Policy: Cluster
Internal Traffic Policy: Cluster
Events:             <none>
```

Figure 18: Service Details and NodePort

### 4.4 Accessing the Application

Since the experiment was performed using Minikube, the following command was used to obtain the service access URL:

```
minikube service kubernetes-bootcamp --url
```

The application was successfully accessed through the returned URL, confirming that the Service was working correctly.

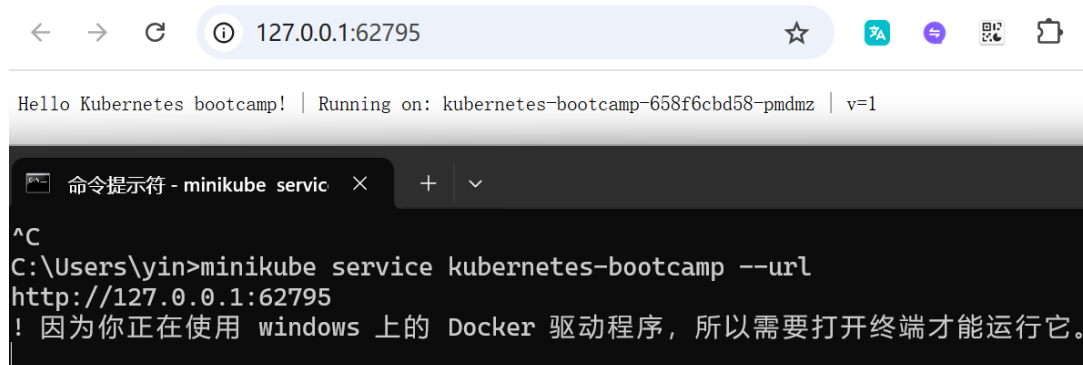


Figure 19: Accessing the Application via Service

## 5. Running Multiple Instances of Your App

### 5.1 Initial Deployment State

First, the current state of the Deployment is checked:

```
kubectl get deployments
```

The output shows that the application is running with only one replica.

```
C:\Users\yin>kubectl get deployments
NAME                READY    UP-TO-DATE    AVAILABLE    AGE
kubernetes-bootcamp 1/1      1             1            124m
```

Figure 20: Initial Deployment with a Single Replica

The ReplicaSet created by the Deployment is inspected using:

```
kubectl get rs
```

```
C:\Users\yin>kubectl get rs
NAME                                DESIRED    CURRENT    READY    AGE
kubernetes-bootcamp-658f6cbd58      1          1          1        127m
```

Figure 21: ReplicaSet Created by the Deployment

## 5.2 Scaling the Deployment

To handle increased workload, the Deployment is scaled to four replicas:

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

```
C:\Users\yin>kubectl scale deployment/kubernetes-bootcamp --replicas=4
deployment.apps/kubernetes-bootcamp scaled

C:\Users\yin>kubectl get deployments
NAME                READY    UP-TO-DATE    AVAILABLE    AGE
kubernetes-bootcamp 4/4      4             4            127m
```

Figure 22: Scaling the Deployment to Four Replicas

The updated Pod status is verified:

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

```
C:\Users\yin>kubectl get pods -o wide
NAME                                READY    STATUS    RESTARTS    AGE    IP             NODE       NOMINATED NODE    READINESS GATES
kubernetes-bootcamp-658f6cbd58-2wcb9 1/1      Running   0           12s    10.244.0.10    minikube   <none>             <none>
kubernetes-bootcamp-658f6cbd58-gn4qt 1/1      Running   0           12s    10.244.0.9     minikube   <none>             <none>
kubernetes-bootcamp-658f6cbd58-pmdmz 1/1      Running   0           127m   10.244.0.8     minikube   <none>             <none>
kubernetes-bootcamp-658f6cbd58-zmdrh 1/1      Running   0           12s    10.244.0.11    minikube   <none>             <none>
```

Figure 23: Four Running Pods After Scaling

Each Pod has a unique IP address, confirming that new Pods were successfully created.

## 5.3 Service Load Balancing Verification

The Service configuration is examined using:

```
kubectl describe service kubernetes-bootcamp
```

```
C:\Users\yin>kubectl describe services kubernetes-bootcamp
Name: kubernetes-bootcamp
Namespace: default
Labels: app=kubernetes-bootcamp
Annotations: <none>
Selector: app=kubernetes-bootcamp
Type: LoadBalancer
IP Family Policy: SingleStack
IP Families: IPv4
IP: 10.111.204.123
IPs: 10.111.204.123
Port: <unset> 8080/TCP
TargetPort: 8080/TCP
NodePort: <unset> 32300/TCP
Endpoints: 10.244.0.8:8080,10.244.0.11:8080,10.244.0.10:8080 + 1 more...
Session Affinity: None
External Traffic Policy: Cluster
Internal Traffic Policy: Cluster
Events: <none>
```

Figure 24: Service Configuration for kubernetes-bootcamp

Since Minikube is running with the Docker driver on Windows, the Service is accessed through a temporary local URL:

```
minikube service kubernetes-bootcamp --url
```

Multiple HTTP requests are sent to the application:

```
curl http://127.0.0.1:<PORT>
```

```
C:\Users\yin>curl http://127.0.0.1:65480/
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-658f6cbd58-gn4qt | v=1

C:\Users\yin>curl http://127.0.0.1:65480/
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-658f6cbd58-zmdrh | v=1

C:\Users\yin>curl http://127.0.0.1:65480/
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-658f6cbd58-pmdmz | v=1

C:\Users\yin>curl http://127.0.0.1:65480/
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-658f6cbd58-pmdmz | v=1

C:\Users\yin>curl http://127.0.0.1:65480/
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-658f6cbd58-2wcb9 | v=1

C:\Users\yin>curl http://127.0.0.1:65480/
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-658f6cbd58-pmdmz | v=1

C:\Users\yin>curl http://127.0.0.1:65480/
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-658f6cbd58-gn4qt | v=1

C:\Users\yin>curl http://127.0.0.1:65480/
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-658f6cbd58-pmdmz | v=1
```

Figure 25: Load Balancing Across Different Pods

The responses show different Pod names, demonstrating that the Service correctly distributes traffic among multiple Pods.

## 5.4 Scaling Down the Deployment

After verification, the Deployment is scaled down to two replicas:

```
kubectl scale deployments/kubernetes-bootcamp --replicas=2
```

The Pod list is checked again:

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

```

C:\Users\yin>kubectl get deployments
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp 2/2     2             2           134m

C:\Users\yin>kubectl get pods -o wide
NAME                READY   STATUS    RESTARTS   AGE   IP            NODE           NOMINATED NODE   REA
DINESS GATES
kubernetes-bootcamp-658f6cbd58-2wcb9 1/1     Running   0           7m8s  10.244.0.10   minikube      <none>           <no
ne>
kubernetes-bootcamp-658f6cbd58-gn4qt 1/1     Terminating 0           7m8s  10.244.0.9    minikube      <none>           <no
ne>
kubernetes-bootcamp-658f6cbd58-pmdmz 1/1     Running   0           134m  10.244.0.8    minikube      <none>           <no
ne>
kubernetes-bootcamp-658f6cbd58-zmdrh 1/1     Terminating 0           7m8s  10.244.0.11   minikube      <none>           <no
ne>

```

Figure 26: Two Pods Remaining After Scaling Down

This confirms that Kubernetes successfully terminated excess Pods.

## 6.Performing a Rolling Update

### 6.1 Checking the Current Deployment Status

The existing Deployment and Pods were first inspected to verify that the application was running correctly.

```

C:\Users\yin>kubectl get deployments
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp 2/2     2             2           166m

C:\Users\yin>kubectl get deployments
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp 2/2     2             2           168m

C:\Users\yin>kubectl get pods
NAME                READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-658f6cbd58-2wcb9 1/1     Running   0           41m
kubernetes-bootcamp-658f6cbd58-pmdmz 1/1     Running   0           168m

```

Figure 27: Deployment status before rolling update

```

C:\Users\yin>kubectl get pods
NAME                READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-658f6cbd58-2wcb9 1/1     Running   0           41m
kubernetes-bootcamp-658f6cbd58-pmdmz 1/1     Running   0           168m

```

Figure 28: Pods running before rolling update (version v2)

### 6.2 Verifying the Current Application Version

The image version of the running Pods was verified using the `kubectl describe pods` command.

```

C:\Users\yin>kubectl describe pods
Name:         kubernetec-bootcamp-658f6cbd58-2wcb9
Namespace:    default
Priority:      0
Service Account: default
Node:         minikube/192.168.49.2
Start Time:   Mon, 15 Dec 2025 21:51:05 +0800
Labels:       app=kubernetec-bootcamp
              pod-template-hash=658f6cbd58
Annotations:  <none>
Status:       Running
IP:           10.244.0.10
IPs:          IP: 10.244.0.10
Controlled By: ReplicaSet/kubernetec-bootcamp-658f6cbd58
Containers:
  kubernetec-bootcamp:
    Container ID:  docker://d8a962880f4c79f9a6193f18f4be017151fe4c4d9872d178df9b9689cfba0fd1
    Image:         gcr.io/google-samples/kubernetec-bootcamp:v1
    Image ID:      docker-pullable://gcr.io/google-samples/kubernetec-bootcamp@sha256:0d6b8ee63bb57c5f5b6156f446b3bc3c143d233037f3a2f00e279c8fcc64af
    Port:          <none>
    Host Port:     <none>
    State:         Running
      Started:     Mon, 15 Dec 2025 21:51:07 +0800
    Ready:         True
    Restart Count: 0
    Environment:   <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-vnkc (ro)
Conditions:
  Type              Status
  PodReadyToStartContainers  True
  Initialized        True
  Ready              True
  ContainersReady    True
  PodScheduled       True
Volumes:
  kube-api-access-vnkc:
    Type:              Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds: 3607
    ConfigMapName:       kube-root-ca.crt
    Optional:            false
    DownwardAPI:        true
    QoS Class:           BestEffort
  Node-Selectors:       <none>
  Tolerations:          node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                      node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason      Age   From      Message
  ----    -
  Normal  Scheduled   41m   default-scheduler   Successfully assigned default/kubernetec-bootcamp-658f6cbd58-2wcb9 to minikube
  Normal  Pulled      41m   kubelet    Container image "gcr.io/google-samples/kubernetec-bootcamp:v1" already present on machine
  Normal  Created     41m   kubelet    Created container: kubernetec-bootcamp
  Normal  Started     41m   kubelet    Started container kubernetec-bootcamp

Name:         kubernetec-bootcamp-658f6cbd58-pmdmz
Namespace:    default
Priority:      0
Service Account: default
Node:         minikube/192.168.49.2
Start Time:   Mon, 15 Dec 2025 19:43:33 +0800
Labels:       app=kubernetec-bootcamp
              pod-template-hash=658f6cbd58
Annotations:  <none>
Status:       Running
IP:           10.244.0.8
IPs:          IP: 10.244.0.8
Controlled By: ReplicaSet/kubernetec-bootcamp-658f6cbd58
Containers:
  kubernetec-bootcamp:
    Container ID:  docker://15c8f1f42feb866d5a0bc7dec1d104a204a495dc820d6d766d5ee9dc60bf610eb
    Image:         gcr.io/google-samples/kubernetec-bootcamp:v1
    Image ID:      docker-pullable://gcr.io/google-samples/kubernetec-bootcamp@sha256:0d6b8ee63bb57c5f5b6156f446b3bc3c143d233037f3a2f00e279c8fcc64af
    Port:          <none>
    Host Port:     <none>
    State:         Running
      Started:     Mon, 15 Dec 2025 19:43:58 +0800
    Ready:         True
    Restart Count: 0
    Environment:   <none>
    Mounts:

```

Figure 29: Pod image version before update (v2)

### 6.3 Updating the Application Version

The Deployment was updated to use a new image version using the following command:

```

kubectl set image deployments/kubernetec-bootcamp
kubernetec-bootcamp=gcr.io/google-samples/kubernetec-bootcamp:v10

```

```

C:\Users\yin>kubectl set image deployments/kubernetec-bootcamp kubernetec-bootcamp=docker.io/jocatalin/kubernetec-bootcamp:v2
deployment.apps/kubernetec-bootcamp image updated

C:\Users\yin>kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
kubernetec-bootcamp-57cc954bb9-8bnrl 1/1     Running   0           10s
kubernetec-bootcamp-57cc954bb9-rz2cz 1/1     Running   0           3s
kubernetec-bootcamp-658f6cbd58-2wcb9 1/1     Terminating 0           42m
kubernetec-bootcamp-658f6cbd58-pmdmz 1/1     Terminating 0           169m

C:\Users\yin>kubectl rollout status deployments/kubernetec-bootcamp
deployment "kubernetec-bootcamp" successfully rolled out

```

Figure 30: Rolling update process after updating the image version

### 6.4 Updating the Application to an Invalid Version

The Deployment was updated to use a non-existent image version (v10) in order to observe Kubernetes rolling update behavior under failure conditions.

```

PS C:\Users\yin> kubectl set image deployments/kubernetes-bootcamp `
>> kubernetes-bootcamp=gcr.io/google-samples/kubernetes-bootcamp:v10
deployment.apps/kubernetes-bootcamp image updated
PS C:\Users\yin> kubectl get deployments
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp 2/2     1            2           177m
PS C:\Users\yin> kubectl get deployments
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp 2/2     1            2           177m
PS C:\Users\yin> kubectl get pods
NAME                READY   STATUS              RESTARTS   AGE
kubernetes-bootcamp-57cc954bb9-8bnrl 1/1     Running            0          8m15s
kubernetes-bootcamp-57cc954bb9-rz2cz 1/1     Running            0          8m8s
kubernetes-bootcamp-677ff875c4-ghch5 0/1     ImagePullBackOff   0          24s

```

Figure 31: Pods status after updating to invalid image version (v10)

## 6.5 Analyzing the Update Failure

Detailed information about the failure was obtained using the `kubectl describe pod` command. The output showed that the image could not be pulled from the container registry.

```

PS C:\Users\yin> kubectl get pods
NAME                READY   STATUS              RESTARTS   AGE
kubernetes-bootcamp-57cc954bb9-8bnrl 1/1     Running            0          8m15s
kubernetes-bootcamp-57cc954bb9-rz2cz 1/1     Running            0          8m8s
kubernetes-bootcamp-677ff875c4-ghch5 0/1     ImagePullBackOff   0          24s

```

Figure 32: Pod events showing ImagePullBackOff error

## 6.6 Rolling Back to the Previous Stable Version

The Deployment was rolled back to the last known stable version using the `kubectl rollout undo` command.

```

PS C:\Users\yin> kubectl rollout undo deployments/kubernetes-bootcamp
deployment.apps/kubernetes-bootcamp rolled back

```

Figure 33: Rolling back the Deployment

## 6.7 Verifying the Rollback Result

After the rollback, the Pods were inspected again to ensure that the application returned to a stable state.

```

PS C:\Users\yin> kubectl get pods
NAME                READY   STATUS   RESTARTS   AGE
kubernetes-bootcamp-57cc954bb9-8bnrl 1/1     Running  0          8m41s
kubernetes-bootcamp-57cc954bb9-rz2cz 1/1     Running  0          8m34s

```

Figure 34: Pods running after rollback

```

PS C:\Users\yjin> kubectl describe pods
Name:          kubernetescamp-57cc954bb9-8bnrl
Namespace:     default
Priority:       0
Service Account: default
Node:          minikube/192.168.49.2
Start Time:    Mon, 15 Dec 2025 22:33:12 +0800
Labels:        app=kubernetescamp
               pod-template-hash=57cc954bb9
Annotations:   <none>
Status:        Running
IP:           10.244.0.12
IPs:          <none>
Controlled By: ReplicaSet/kubernetescamp-57cc954bb9
Containers:
  kubernetescamp:
    Container ID:  docker://046fca9806160efc55d4335c76d63476a112f23da31006fc1cc085263189b6
    Image:          docker.io/jocatalin/kubernetescamp:v2
    Image ID:       docker-pullable://jocatalin/kubernetescamp@sha256:fb1a3ced080ccf1f83f18ab5cd14199a38adc1b49aa4244f5d65ad3f5feb2a5
    Port:          <none>
    Host Port:     <none>
    State:         Running
      Started:      Mon, 15 Dec 2025 22:33:19 +0800
    Ready:         True
    Restart Count:  0
    Environment:   <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-vn2xs (ro)
Conditions:
  Type              Status
  PodReadyToStartContainers  True
  Initialized        True
  Ready              True
  ContainersReady    True
  PodScheduled       True
Volumes:
  kube-api-access-vn2xs:
    Type:          Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds: 3607
    ConfigMapName: kube-root-ca.crt
    Optional:      false
    DownwardAPI:   true
QoS Class:        BestEffort
Node-Selectors:   <none>
Tolerations:      node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                  node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason      Age   From      Message
  ----     -
  Normal   Scheduled   8m47s  default-scheduler  Successfully assigned default/kubernetescamp-57cc954bb9-8bnrl to minikube
  Normal   Pulling    8m40s  kubelet      Pulling image "docker.io/jocatalin/kubernetescamp:v2"
  Normal   Pulled     8m40s  kubelet      Successfully pulled image "docker.io/jocatalin/kubernetescamp:v2" in 5.858s (5.858s including waiting). Image size: 211336459 bytes.
  Normal   Created    8m40s  kubelet      Created container: kubernetescamp
  Normal   Started    8m40s  kubelet      Started container kubernetescamp

Name:          kubernetescamp-57cc954bb9-rz2cz
Namespace:     default
Priority:       0
Service Account: default
Node:          minikube/192.168.49.2
Start Time:    Mon, 15 Dec 2025 22:33:19 +0800
Labels:        app=kubernetescamp
               pod-template-hash=57cc954bb9
Annotations:   <none>
Status:        Running
IP:           10.244.0.13
IPs:          <none>
Controlled By: ReplicaSet/kubernetescamp-57cc954bb9
Containers:
  kubernetescamp:
    Container ID:  docker://af4b540c601e29da8a6b0274af0d1bb6ba97ccf1c0b10cc794d45d12b317d132
    Image:          docker.io/jocatalin/kubernetescamp:v2
    Image ID:       docker-pullable://jocatalin/kubernetescamp@sha256:fb1a3ced080ccf1f83f18ab5cd14199a38adc1b49aa4244f5d65ad3f5feb2a5
    Port:          <none>
    Host Port:     <none>
    State:         Running
      Started:      Mon, 15 Dec 2025 22:33:20 +0800
    Ready:         True
    Restart Count:  0
    Environment:   <none>

```

Figure 35: Pod image version after rollback (v2)

## IV. Conclusion

In this experiment, a local Kubernetes cluster was successfully deployed and managed using Minikube. Through a series of practical operations, containerized applications were deployed using Kubernetes Deployments, and their runtime status was verified by inspecting Pods, Nodes, and application logs. The experiment demonstrated how applications can be accessed internally via kubectl proxy and externally through Kubernetes Services, including NodePort and LoadBalancer types.

Furthermore, application scalability was validated by dynamically adjusting the number of replicas and observing service-level load balancing across multiple Pods. Rolling update and rollback mechanisms were also tested, including failure scenarios caused by invalid image versions, which highlighted Kubernetes' ability to automatically detect errors and recover to a stable state.

Overall, this experiment provided hands-on experience with core Kubernetes features such as deployment management, service exposure, scalability, and fault tolerance, strengthening the understanding of container orchestration concepts and practical cluster operations.