Deploying a Kubernetes Cluster with Minikube

Introduction

The concept of kubernetes is to manage containers and hence , it is simply termed as a container orchestration tool. Kubernetes clusters (comprising of the control plane and worker nodes) are purposely designed to manage containers with the control plane acting as the brain of the cluster , managing the entire cluster.

Unlike kubernetes clusters such as Amazon Elastic container service for kubenetes which are mainly used in the production environmnet, minikube is a single node kubernetes cluster which is used for running local kubernetes clusters .When minikube is installed, and started with the required memory and CPU, it creates a kubernetes cluster comprising of one node and control plane needed to manage or orchestrate containers locally. With this, application can be tested using the containerised environment created by minikube which comes with a kubernetes cluster to manage the container locally.With minikube, you just focus on how to make the application in the containers achieve the desired state by defining the kubernetes deployments and configurations and the minikube provides the kubernetes cluster to manage the application including the container the application will be running on. Minikube makes you focus on developing and testing your application without worrying about container runtime.

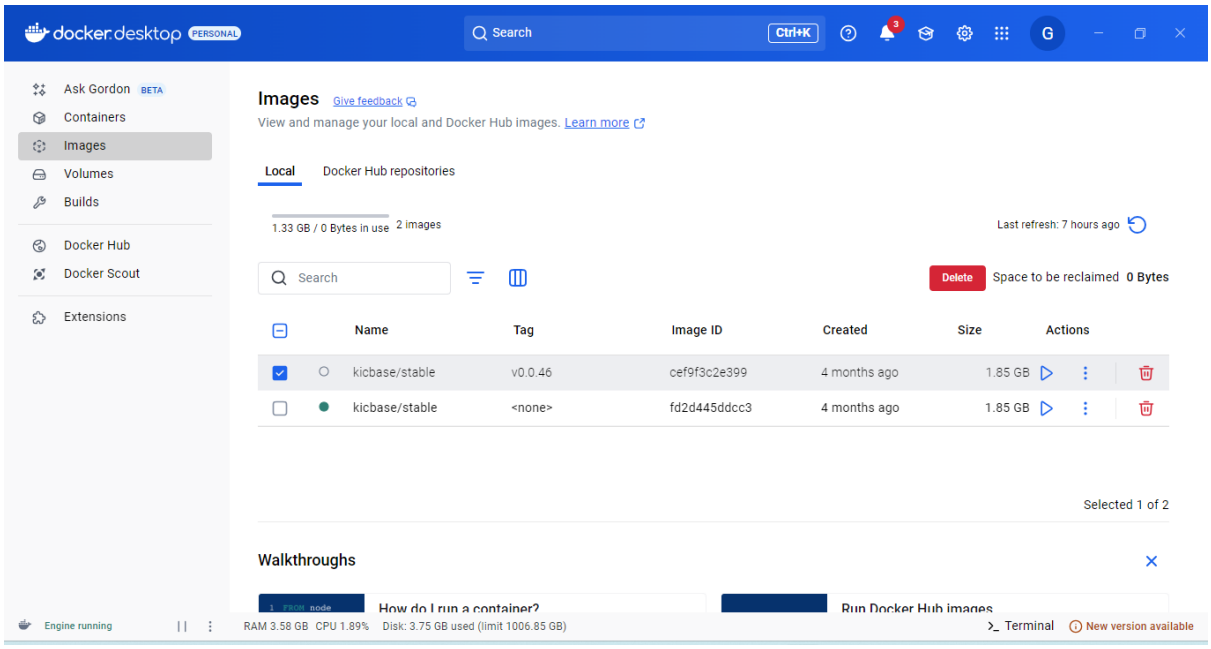
This lab focus on creating a one node kubernetes cluster using minikube to orchestrate containers in the deveopment and testing environment for applications locally.Minikube can run on virtualization platforms such as hyperV , virtual box and docker. However, this lab made use of docker to provide the virtialisation platform for minikube to run and provision kubernetes cluster as well as a container runtime for the application.

Demo

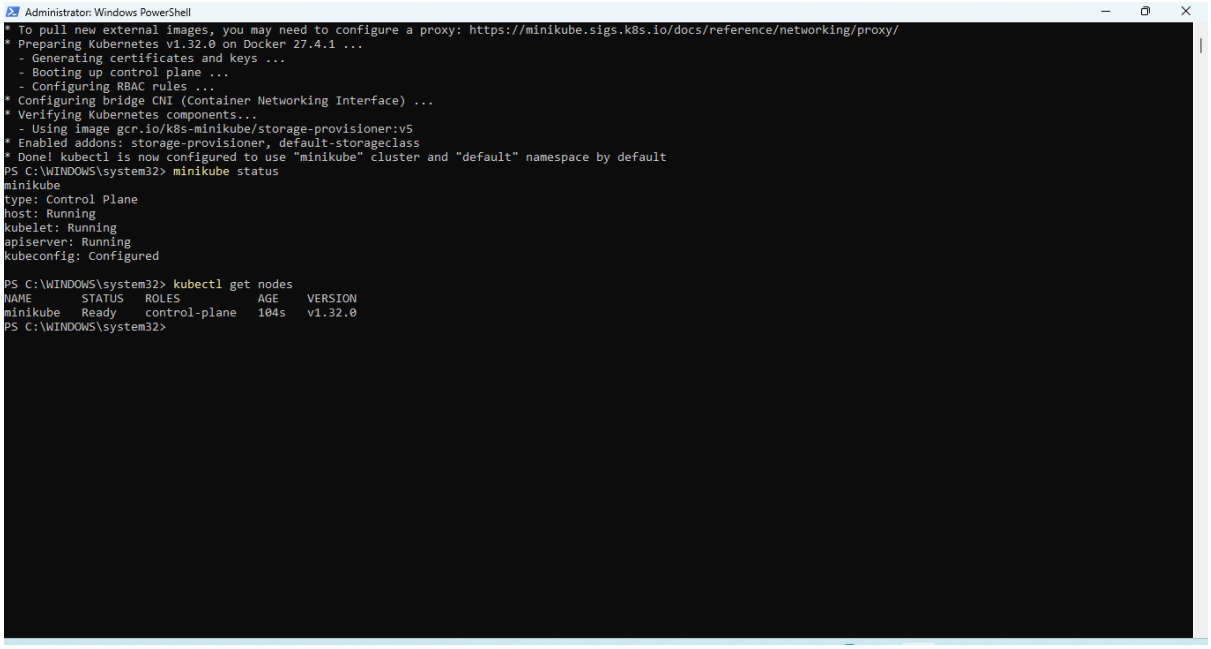
The minikube was startedsuccessfullly and running on the driver ,docker.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\WINDOWS\system32> minikube start --driver=docker --memory=3830 --cpus=2
* minikube v1.35.0 on Microsoft Windows 11 Pro Education 10.0.22631.5262 Build 22631.5262
* Using the docker driver based on user configuration
* Using Docker Desktop driver with root privileges
* Starting "minikube" primary control-plane node in "minikube" cluster
* Pulling base image v0.0.46 ...
! minikube was unable to download gcr.io/k8s-minikube/kicbase:v0.0.46, but successfully downloaded docker.io/kicbase/stable:v0.0.46@sha256:fd2d445ddcc33ebc5c6b68a17e621
9ea207ce63c005095ea1525296da2d1a279 as a fallback image
* Creating docker container (CPUs=2, Memory=3830MB) ...
! Failing to connect to https://registry.k8s.io/ from inside the minikube container
* To pull new external images, you may need to configure a proxy: https://minikube.sigs.k8s.io/docs/reference/networking/proxy/
* Preparing Kubernetes v1.32.0 on Docker 27.4.1 ...
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Configuring bridge CNI (Container Networking Interface) ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
PS C:\WINDOWS\system32>
```

An image of the minikube was created in the docker driver with a corresponding container as well.



The status of the minikube was verified and it was up and running.



Kube control interacted with the control plane to retrieve information about the nodes in the cluster and there was only one node which was ready.

```
Administrator: Windows PowerShell
! Failing to connect to https://registry.k8s.io/ from inside the minikube container
* To pull new external images, you may need to configure a proxy: https://minikube.sigs.k8s.io/docs/reference/networking/proxy/
* Preparing Kubernetes v1.32.0 on Docker 27.4.1 ...
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Configuring bridge CNI (Container Networking Interface) ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
PS C:\WINDOWS\system32> minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured

PS C:\WINDOWS\system32> kubectl get nodes
NAME       STATUS   ROLES           AGE    VERSION
minikube   Ready    control-plane   104s   v1.32.0
PS C:\WINDOWS\system32> kubectl create configmap nginx-index --from-file=C:\Users\gertr\Documents\index.html
configmap/nginx-index created
PS C:\WINDOWS\system32>
```

A config map was created in the cluster to store the index.html file.


```
Administrator: Windows PowerShell
! Failing to connect to https://registry.k8s.io/ from inside the minikube container
* To pull new external images, you may need to configure a proxy: https://minikube.sigs.k8s.io/docs/reference/networking/proxy/
* Preparing Kubernetes v1.32.0 on Docker 27.4.1 ...
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Configuring bridge CNI (Container Networking Interface) ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
PS C:\WINDOWS\system32> minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured

PS C:\WINDOWS\system32> kubectl get nodes
NAME       STATUS   ROLES           AGE    VERSION
minikube   Ready    control-plane   104s   v1.32.0
PS C:\WINDOWS\system32> kubectl create configmap nginx-index --from-file=C:\Users\gertr\Documents\index.html
configmap/nginx-index created
PS C:\WINDOWS\system32>
```

A deployment was made with the name “minikube-lab” yielding the creation of a pod.

```
Administrator: Windows PowerShell
  - Configuring RBAC rules ...
* Configuring bridge CNI (Container Networking Interface) ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
PS C:\WINDOWS\system32> minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured

PS C:\WINDOWS\system32> kubectl get nodes
NAME       STATUS   ROLES           AGE    VERSION
minikube   Ready    control-plane   104s   v1.32.0
PS C:\WINDOWS\system32> kubectl create configmap nginx-index --from-file=C:\Users\gertr\Documents\index.html
configmap/nginx-index created
PS C:\WINDOWS\system32>
>> kubectl create deployment minikube-lab --image=nginx:1.25-alpine
deployment.apps/minikube-lab created
PS C:\WINDOWS\system32>
```

```
PS C:\WINDOWS\system32> kubectl get pods -o wide
NAME                              READY   STATUS    RESTARTS   AGE     IP           NODE       NOMINATED NODE   READINESS GATES
minikube-lab-8596dd7d8d-hhwph     1/1     Running   0          9m11s   10.244.0.4   minikube   <none>           <none>
PS C:\WINDOWS\system32>
```

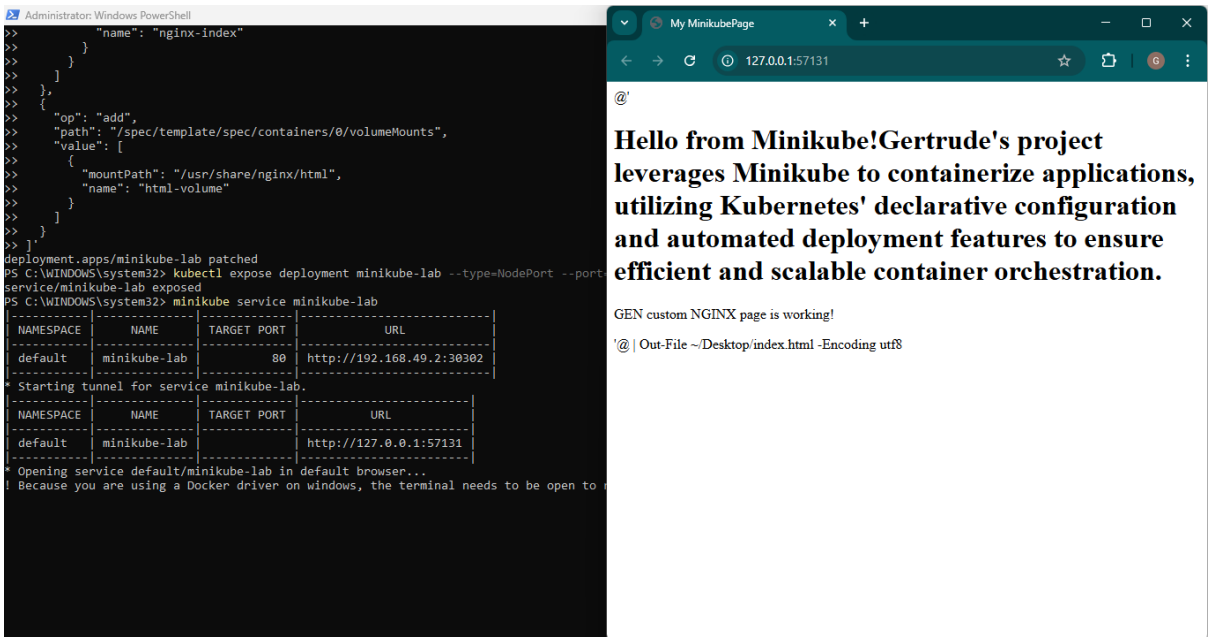
The deployment was patched and updated the pod template. A new volume named html-volume was added to the Pod template, which was sourced from the nginx-index configmap. Also, a new volume mount was added to the container, which mounts the html-volume volume at the path /usr/share/nginx/html.

```
Administrator: Windows PowerShell
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
PS C:\WINDOWS\system32> minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured

PS C:\WINDOWS\system32> kubectl get nodes
NAME       STATUS   ROLES           AGE    VERSION
minikube   Ready    control-plane   104s   v1.32.0
PS C:\WINDOWS\system32> kubectl create configmap nginx-index --from-file=C:\Users\gertr\Documents\index.html
configmap/nginx-index created
PS C:\WINDOWS\system32>
>> kubectl create deployment minikube-lab --image=nginx:1.25-alpine
deployment.apps/minikube-lab created
PS C:\WINDOWS\system32> kubectl patch deployment minikube-lab --type='json' -p='[
>>   {
>>     "op": "add",
>>     "path": "/spec/template/spec/volumes",
>>     "value": [
>>       {
>>         "name": "html-volume",
>>         "configMap": {
>>           "name": "nginx-index"
>>         }
>>       }
>>     ]
>>   },
>>   {
>>     "op": "add",
>>     "path": "/spec/template/spec/containers/0/volumeMounts",
>>     "value": [
>>       {
>>         "mountPath": "/usr/share/nginx/html",
>>         "name": "html-volume"
>>       }
>>     ]
>>   }
>> ]'
deployment.apps/minikube-lab patched
PS C:\WINDOWS\system32>
```

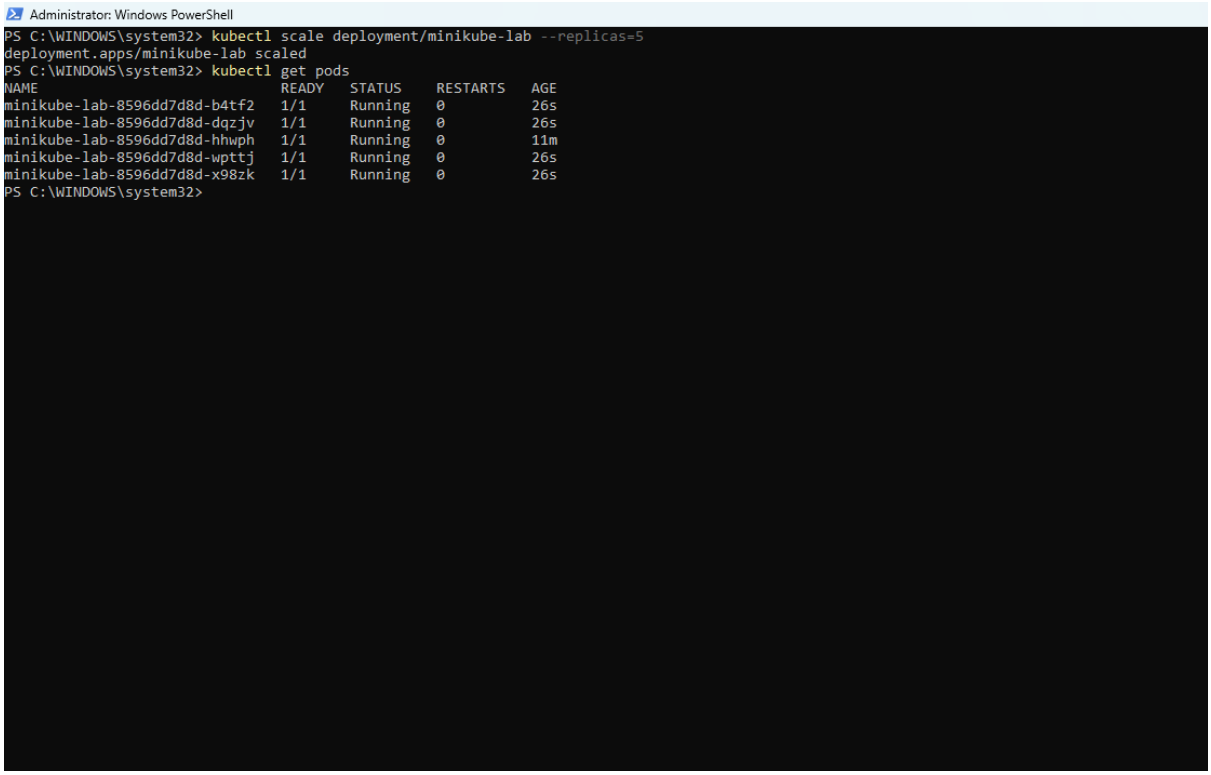
The pod was exposed to allow access to the Nginx server, which served the index.html content.

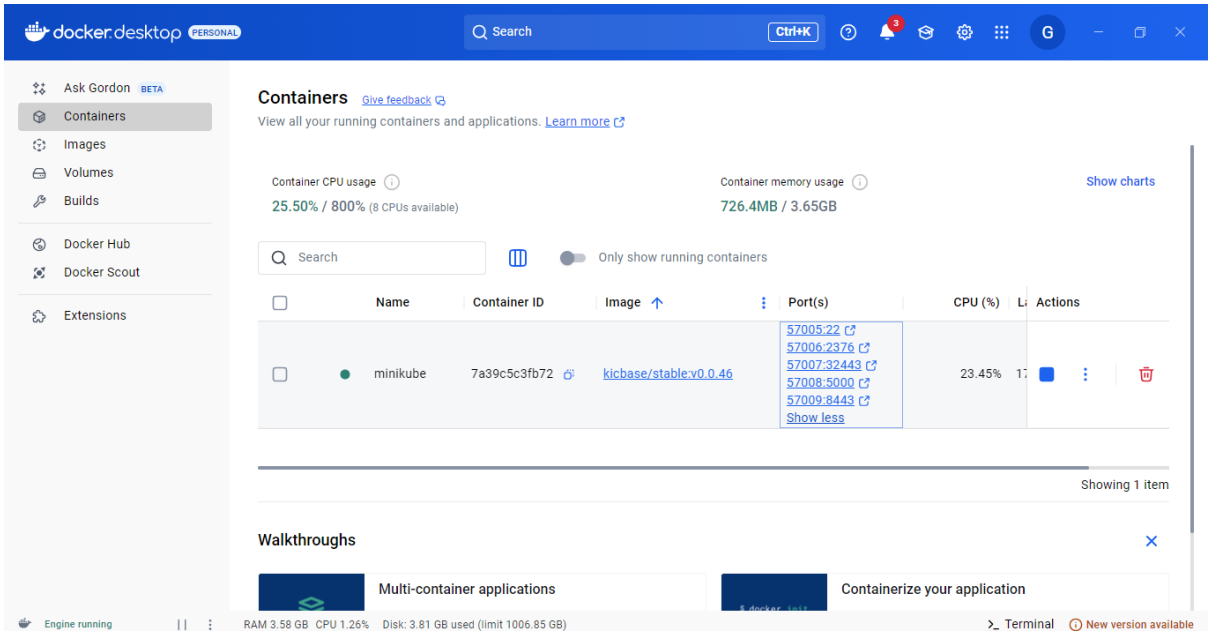
```
Administrator: Windows PowerShell
>>         "name": "html-volume",
>>         "configMap": {
>>           "name": "nginx-index"
>>         }
>>       }
>>     ]
>>   },
>>   {
>>     "op": "add",
>>     "path": "/spec/template/spec/containers/0/volumeMounts",
>>     "value": [
>>       {
>>         "mountPath": "/usr/share/nginx/html",
>>         "name": "html-volume"
>>       }
>>     ]
>>   }
>> ]'
deployment.apps/minikube-lab patched
PS C:\WINDOWS\system32> kubectl expose deployment minikube-lab --type=NodePort --port=80
service/minikube-lab exposed
PS C:\WINDOWS\system32>
```

A service command was entered to provide an endpoint to access the application. It acted as a load balancer, distributing traffic across the available Pod.



Finally, the number of pods was scaled up to 5 replicas to be able to withstand traffic demands.




docker desktop PERSONAL
Search
Ctrl+K
Ask Gordon BETA
Containers
Images
Volumes
Builds
Docker Hub
Docker Scout
Extensions
Containers
Give feedback
View all your running containers and applications. Learn more
Container CPU usage
25.50% / 800% (8 CPUs available)
Container memory usage
726.4MB / 3.65GB
Show charts
Search
Only show running containers
Name
Container ID
Image
Port(s)
CPU (%)
Actions
minikube
7a39c5c3fb72
kicbase/stable:v0.0.46
57005:22
57006:2376
57007:32443
57008:5000
57009:8443
Show less
23.45%
Showing 1 item
Walkthroughs
Multi-container applications
Containerize your application
Engine running
RAM 3.58 GB CPU 1.26%
Disk: 3.81 GB used (limit 1006.85 GB)
Terminal
New version available