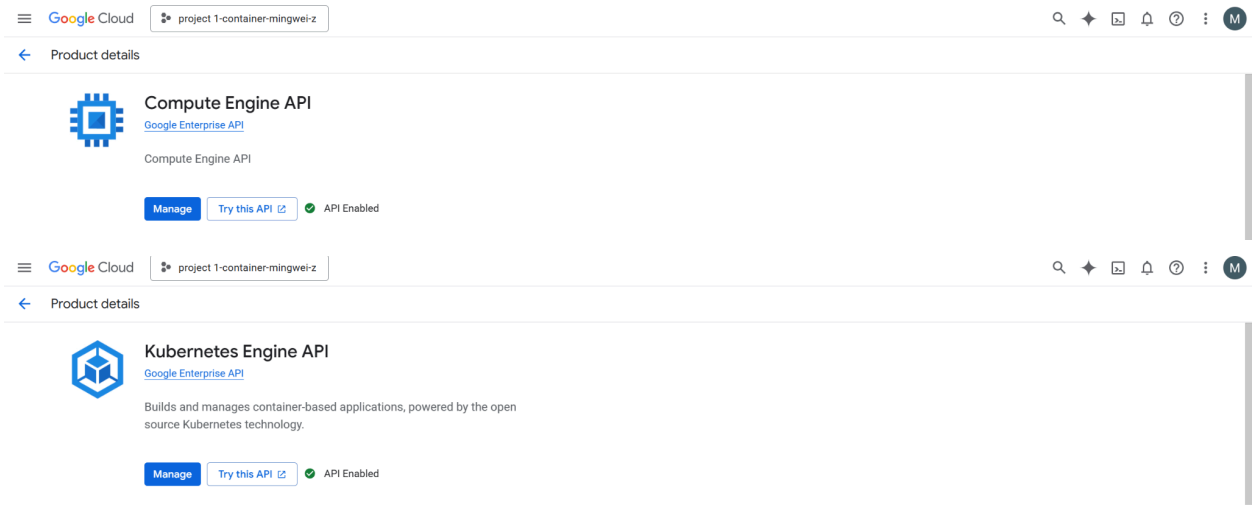
1. Set Up:

- I chose Google Cloud Platform (GCP) as my cloud provider and used Google Kubernetes Engine (GKE) to create a Kubernetes cluster with one master and two worker nodes. I accessed the environment through Google Cloud Shell, which comes with pre-installed kubectl and gcloud tools.

- After that, I created a new project called “project 1-container-mingwei-z”. Once the project was created, I enabled both the Kubernetes Engine API and the Compute Engine API. This was required for GCP to create and manage the virtual machines that would become my master and worker nodes in the cluster.



- Next, I set up the Cloud Shell Editor. In the editor’s terminal, I first verified that I was in the correct project using the following commands:

  gcloud config set project project-1-container-mingwei-z
  gcloud config get-value project

  Output:
  mingw_zhang123@cloudshell:~ (marine-clarity-475016-m8)$ gcloud config set project project-1-container-mingwei-z
  Updated property [core/project].
  mingw_zhang123@cloudshell:~ (project-1-container-mingwei-z)$ gcloud config get-value project
  Your active configuration is: [cloudshell-29216]
  project-1-container-mingwei-z

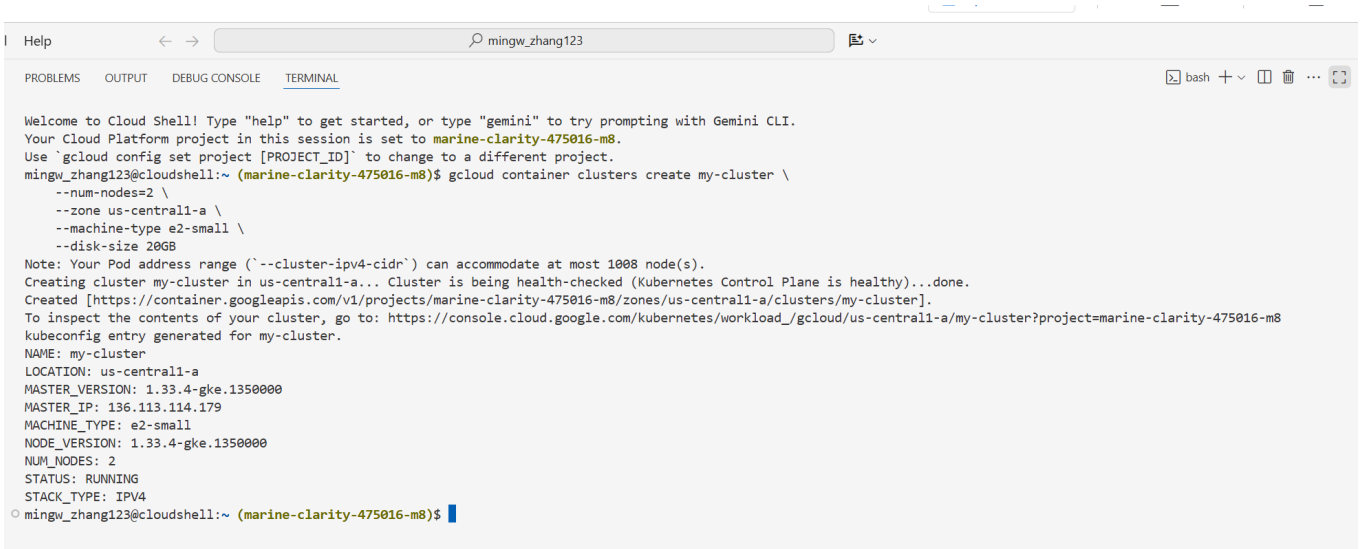
Purpose:
It sets project-1-container-mingwei-z as the active project for all future commands and checks and confirms which project is currently active. The output confirms that the active project was successfully set to project-1-container-mingwei-z, meaning all future commands will be executed under the correct project.

- I used this command to create my first Kubernetes cluster on Google Kubernetes Engine:
  gcloud container clusters create my-cluster \
  --num-nodes=2 \
  --zone us-central1-a \
  --machine-type e2-small \
  --disk-size 20GB

  Output:
  mingw_zhang123@cloudshell:~ (marine-clarity-475016-m8)$ gcloud container clusters create my-cluster \
  --num-nodes=2 \
  --zone us-central1-a \
  --machine-type e2-small \
  --disk-size 20GB
  Note: Your Pod address range (`--cluster-ipv4-cidr`) can accommodate at most 1008 node(s).
  Creating cluster my-cluster in us-central1-a... Cluster is being health-checked (Kubernetes Control Plane is healthy)...done.
  Created [https://container.googleapis.com/v1/projects/marine-clarity-475016-m8/zones/us-central1-a/clusters/my-cluster].
  To inspect the contents of your cluster, go to: https://console.cloud.google.com/kubernetes/workload_/gcloud/us-central1-a/my-cluster?project=marine-clarity-475016-m8

kubeconfig entry generated for my-cluster.
NAME: my-cluster
LOCATION: us-central1-a
MASTER_VERSION: 1.33.4-gke.1350000
MASTER_IP: 136.113.114.179
MACHINE_TYPE: e2-small
NODE_VERSION: 1.33.4-gke.1350000
NUM_NODES: 2
STATUS: RUNNING
STACK_TYPE: IPV4

Purpose:
The command sets up small virtual machines with 20 GB of disk space, which is enough for testing and running basic containerized applications. This step is to establish the main environment where my deployments and services will run. The output shows that the cluster was created successfully and is now running. It lists details such as the cluster name (my-cluster), location (us-central1-a), master and node versions, machine type (e2-small), number of nodes (2), and status (RUNNING). These details confirm that the Kubernetes control plane and worker nodes were set up correctly.



- Checking worker node and cluster information:

kubectl get nodes
kubectl cluster-info

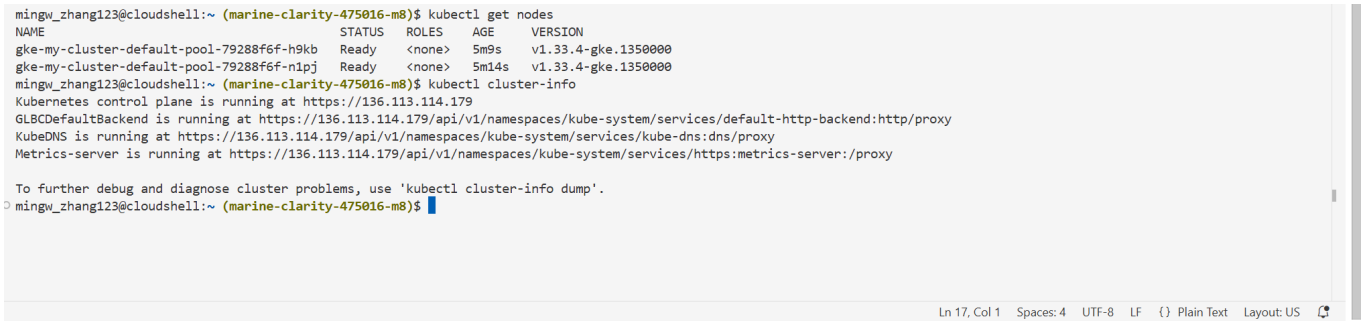
Output:

```
mingw_zhang123@cloudshell:~ (marine-clarity-475016-m8)$ kubectl get nodes
NAME                                     STATUS   ROLES    AGE     VERSION
gke-my-cluster-default-pool-79288f6f-h9kb   Ready    <none>   5m9s    v1.33.4-gke.1350000
gke-my-cluster-default-pool-79288f6f-n1pj   Ready    <none>   5m14s   v1.33.4-gke.1350000
mingw_zhang123@cloudshell:~ (marine-clarity-475016-m8)$ kubectl cluster-info
Kubernetes control plane is running at https://136.113.114.179
GLBCDefaultBackend is running at https://136.113.114.179/api/v1/namespaces/kube-system/services/default-http-backend:http/proxy
KubeDNS is running at https://136.113.114.179/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
Metrics-server is running at https://136.113.114.179/api/v1/namespaces/kube-system/services/https:metrics-server:/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

Purpose:

Inputs kubectl get nodes and kubectl cluster-infocheck the health of the cluster. kubectl get nodes shows the status of all worker nodes, and kubectl cluster-info displays details about the control plane to confirm the cluster is running correctly. The output confirms that both worker nodes (gke-my-cluster-default-pool-79288f6f-h9kb and gke-my-cluster-default-pool-79288f6f-n1pj) are in Ready status, running Kubernetes version v1.33.4-gke.1350000. It also shows that the Kubernetes control plane is active at IP 136.113.114.179, and key services such as GLBCDefaultBackend, KubeDNS, and Metrics-server are running properly, confirming the cluster is fully operational.



(Launch a Kubernetes cluster complete)

- After creating the deployment.yaml file, I deployed the application to the cluster using the command:

  kubectl apply -f deployment.yaml

  Purpose:
  Apply the configuration defined in the deployment.yaml file to the Kubernetes cluster. It tells Kubernetes to create the specified resources, which in this case are a Deployment running multiple Nginx containers and a Service that exposes them to the internet. This step launches the actual application inside the cluster.

2. Next Step:

- Created a file named deployment.yaml with the following configuration details:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.20
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: web-service
spec:
  type: LoadBalancer
```

```
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

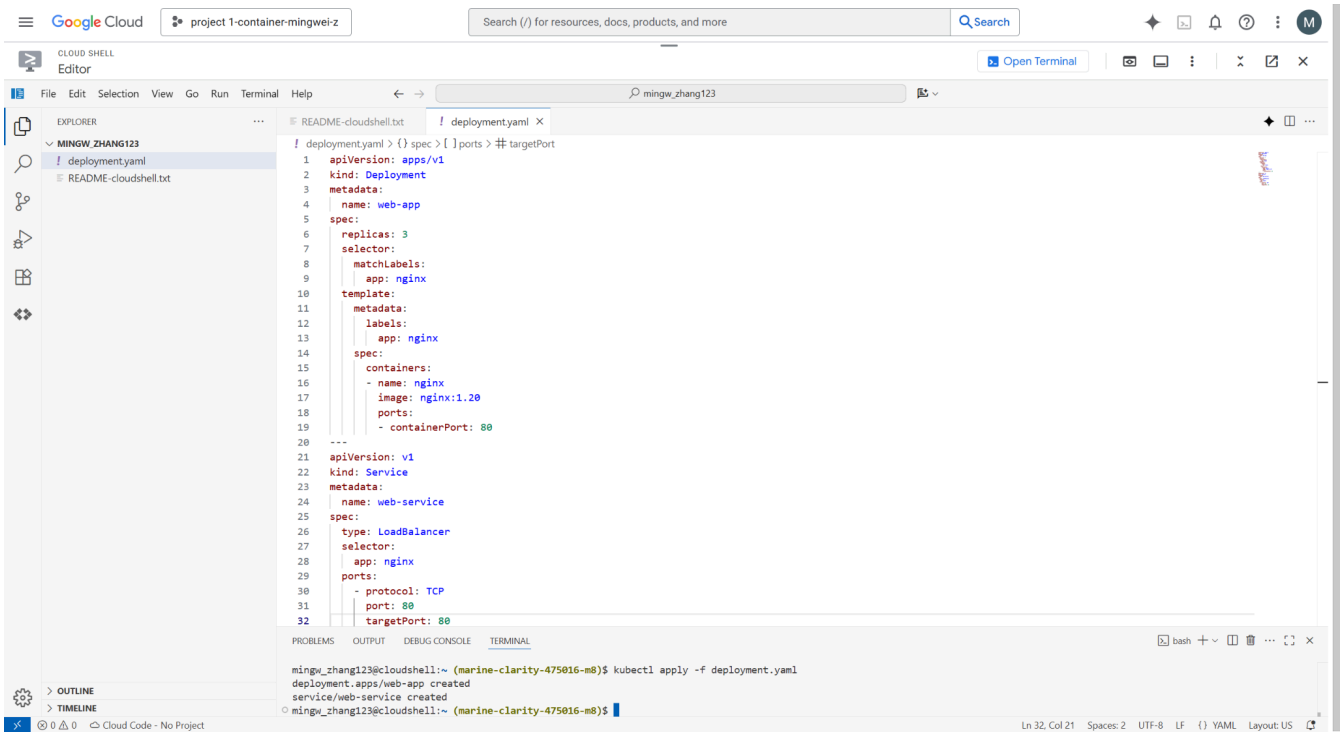
Purpose:
The purpose of creating the deployment.yaml file was to define and deploy an Nginx web application on the Kubernetes cluster. The output confirmed that both the deployment and service were successfully created, meaning the three Nginx pods were launched and made accessible externally through an assigned load balancer IP.

- Then, deployed the application by running:

  kubectl apply -f deployment.yaml

  Output:
  mingw_zhang123@cloudshell:~ (marine-clarity-475016-m8)$ kubectl apply -f deployment.yaml
  deployment.apps/web-app created
  service/web-service created

  Purpose:
  The input command was used to deploy the application defined in the deployment.yaml file to the Kubernetes cluster. The output confirms that both the Deployment (web-app) and Service (web-service) were successfully created, meaning the pods were launched, and the application was exposed through a load balancer for external access.



- Verified that everything was running by checking all Kubernetes resources and monitoring the pods until they were fully active:
  kubectl get all
  kubectl get pods --watch

  Output:

```
mingw_zhang123@cloudshell:~ (marine-clarity-475016-m8)$ kubectl get all
NAME                         READY   STATUS    RESTARTS   AGE
pod/web-app-7b9677b44c-5d8xj   1/1     Running   0          2m17s
pod/web-app-7b9677b44c-g4wxx   1/1     Running   0          2m17s
pod/web-app-7b9677b44c-lcwnp   1/1     Running   0          2m17s

NAME                  TYPE           CLUSTER-IP      EXTERNAL-IP    PORT(S)        AGE
service/kubernetes    ClusterIP      34.118.224.1    <none>         443/TCP        25m
service/web-service   LoadBalancer   34.118.234.168  34.68.59.122   80:31095/TCP   2m18s

NAME                     READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/web-app  3/3     3            3           2m19s

NAME                               DESIRED   CURRENT   READY   AGE
replicaset.apps/web-app-7b9677b44c   3         3         3       2m18s
```

```
mingw_zhang123@cloudshell:~ (marine-clarity-475016-m8)$ kubectl get pods --watch
NAME                   READY   STATUS    RESTARTS   AGE
web-app-7b9677b44c-5d8xj   1/1     Running   0          2m21s
web-app-7b9677b44c-g4wxx   1/1     Running   0          2m21s
web-app-7b9677b44c-lcwnp   1/1     Running   0          2m21s
```

Purpose:
To confirm that the deployment was working correctly and all components were running as expected. By using kubectl get all and kubectl get pods --watch, I checked that all three Nginx pods were in the Running state, the web-service load balancer was active with an external IP address, and the deployment had successfully created and maintained the desired number of replicas.

App can be viewed in the browser (which shows default Nginx Welcome Page and confirms confirms the app is live on Google Kubernetes Engine):
http://34.68.59.122

3. Scale the Application

- Scale from 3 to 5 replicas and verify the scaling:
  kubectl scale deployment web-app --replicas=5

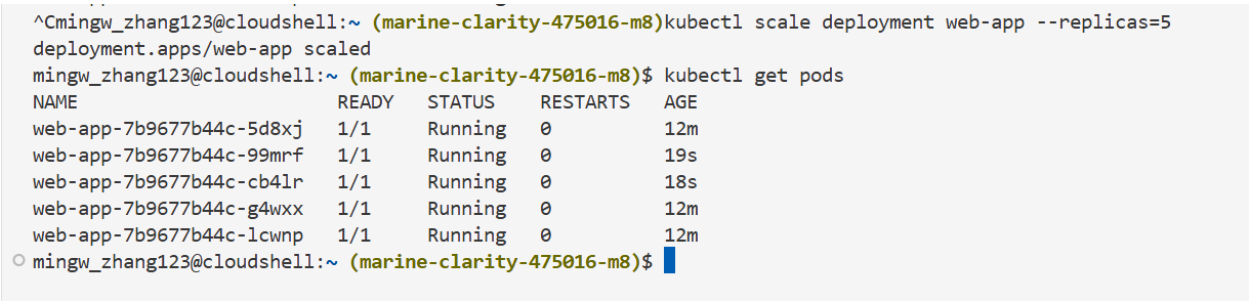
kubectl get pods

Output:

^Cmingw_zhang123@cloudshell:~ (marine-clarity-475016-m8)kubectl scale deployment web-app --replicas=5
deployment.apps/web-app scaled
mingw_zhang123@cloudshell:~ (marine-clarity-475016-m8)$ kubectl get pods
NAME READY STATUS RESTARTS AGE
web-app-7b9677b44c-5d8xj 1/1 Running 0 12m
web-app-7b9677b44c-99mrf 1/1 Running 0 19s
web-app-7b9677b44c-cb4lr 1/1 Running 0 18s
web-app-7b9677b44c-g4wxx 1/1 Running 0 12m
web-app-7b9677b44c-lcwnp 1/1 Running 0 12m

Purpose:

To scale the deployment from three to five replicas to test Kubernetes' ability to handle scaling. After scaling, I ran kubectl get pods to verify that the new pods were created successfully. The output shows five pods, all in the Running state, confirming that Kubernetes handled the scaling process correctly and the application continued running without interruption.



- Check deployment status:

kubectl describe deployment web-app

Output:

mingw_zhang123@cloudshell:~ (marine-clarity-475016-m8)$ kubectl describe deployment web-app
Name: web-app
Namespace: default
CreationTimestamp: Mon, 13 Oct 2025 17:28:40 +0000
Labels: <none>

```
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=nginx
Replicas:               5 desired | 5 updated | 5 total | 5 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=nginx
  Containers:
   nginx:
    Image:        nginx:1.20
    Port:         80/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
  Node-Selectors:  <none>
  Tolerations:     <none>
Conditions:
  Type          Status  Reason
  ----          ------  ------
  Progressing   True    NewReplicaSetAvailable
  Available     True    MinimumReplicasAvailable
OldReplicaSets:  <none>
NewReplicaSet:   web-app-7b9677b44c (5/5 replicas created)
Events:
  Type    Reason             Age    From                   Message
  ----    ------             ----   ----                   -------
  Normal  ScalingReplicaSet  14m    deployment-controller  Scaled up replica set
web-app-7b9677b44c from 0 to 3
  Normal  ScalingReplicaSet  2m17s  deployment-controller  Scaled up replica set
web-app-7b9677b44c from 3 to 5
```

Purpose:

I used the command kubectl describe deployment web-app to get detailed information about the current deployment. This command shows the configuration, replica count, container image, and recent scaling events. The output indicates that the deployment named web-app is running five replicas, all available and up to date. It also confirms that the deployment strategy is RollingUpdate, meaning updates happen gradually to avoid downtime. The events section shows that Kubernetes successfully scaled the replica set

from 3 to 5, verifying that the scaling operation worked as expected and the deployment is stable.

```
mingw_zhang123@cloudshell:~ (marine-clarity-475016-m8)$ kubectl describe deployment web-app
Name:                   web-app
Namespace:              default
CreationTimestamp:      Mon, 13 Oct 2025 17:28:40 +0000
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=nginx
Replicas:               5 desired | 5 updated | 5 total | 5 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=nginx
  Containers:
   nginx:
    Image:         nginx:1.20
    Port:          80/TCP
    Host Port:     0/TCP
    Environment:   <none>
    Mounts:        <none>
  Volumes:         <none>
  Node-Selectors:  <none>
  Tolerations:     <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Progressing    True    NewReplicaSetAvailable
  Available      True    MinimumReplicasAvailable
OldReplicaSets:  <none>
NewReplicaSet:   web-app-7b9677b44c (5/5 replicas created)
Events:
  Type    Reason             Age    From                   Message
  ----    ------             ----   ----                   -------
  Normal  ScalingReplicaSet  14m    deployment-controller  Scaled up replica set web-app-7b9677b44c from 0 to 3
  Normal  ScalingReplicaSet  2m17s  deployment-controller  Scaled up replica set web-app-7b9677b44c from 3 to 5
mingw_zhang123@cloudshell:~ (marine-clarity-475016-m8)$
```

4. Update the Application Version

- Update to nginx 1.23 (This shows how Kubernetes performs a rolling update by replacing old pods gradually without downtime):

  kubectl set image deployment/web-app nginx=nginx:1.27.0

  Output:
  mingw_zhang123@cloudshell:~ (marine-clarity-475016-m8)$ kubectl set image deployment/web-app nginx=nginx:1.27.0 deployment.apps/web-app image updated

- Monitor the rollout:
  kubectl rollout status deployment/web-app

  Output:
  mingw_zhang123@cloudshell:~ (marine-clarity-475016-m8)$ kubectl rollout status deployment/web-app

deployment "web-app" successfully rolled out

- Verify the new pod:
  kubectl get pods

  Output:
  mingw_zhang123@cloudshell:~ (marine-clarity-475016-m8)$ kubectl get pods
  NAME READY STATUS RESTARTS AGE
  web-app-8bdf8f46b-5qjv4 1/1 Running 0 2m7s
  web-app-8bdf8f46b-769hf 1/1 Running 0 118s
  web-app-8bdf8f46b-8mfgc 1/1 Running 0 2m
  web-app-8bdf8f46b-j7fkv 1/1 Running 0 2m6s
  web-app-8bdf8f46b-x7lfg 1/1 Running 0 2m7s

- Verify the new version:
  kubectl get pods -o jsonpath='{.items[*].spec.containers[*].image}'

  Output:
  mingw_zhang123@cloudshell:~ (marine-clarity-475016-m8)$ kubectl get pods -o jsonpath='{.items[*].spec.containers[*].image}'
  nginx:1.27.0 nginx:1.27.0 nginx:1.27.0 nginx:1.27.0 nginx:1.27.0mingw_zhang123@cloudshell:~ (marine-clarity-475016-m8)$

  Purpose:
  I used the command kubectl set image deployment/web-app nginx=nginx:1.27.0 to update the container image from nginx:1.20 to nginx:1.27.0. This input tells Kubernetes to roll out the new version gradually without downtime. The output confirmed that the deployment image was updated successfully. Next, I ran kubectl rollout status deployment/web-app to monitor the update process, and the output showed "deployment 'web-app' successfully rolled out", meaning the new version was applied to all pods. I then verified the running pods with kubectl get pods, which listed five pods in the Running state, confirming the update was stable. Lastly, I used kubectl get pods -o jsonpath='{.items[*].spec.containers[*].image}', and the output showed all pods using nginx:1.27.0, proving the version update was completed successfully.

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

Welcome to Cloud Shell! Type "help" to get started, or type "gemini" to try prompting with Gemini CLI.
Your Cloud Platform project in this session is set to marine-clarity-475016-m8.
mingw_zhang123@cloudshell:~ (marine-clarity-475016-m8)$ kubectl set image deployment/web-app nginx-container=nginx:1.23
error: unable to find container named "nginx-container"
mingw_zhang123@cloudshell:~ (marine-clarity-475016-m8)$ kubectl set image deployment/web-app nginx=nginx:1.27.0
deployment.apps/web-app image updated
mingw_zhang123@cloudshell:~ (marine-clarity-475016-m8)$ kubectl rollout status deployment/web-app
deployment "web-app" successfully rolled out
mingw_zhang123@cloudshell:~ (marine-clarity-475016-m8)$ kubectl get pods
NAME                       READY   STATUS    RESTARTS   AGE
web-app-8bdf8f46b-5qjv4    1/1     Running   0          2m7s
web-app-8bdf8f46b-769hf    1/1     Running   0          118s
web-app-8bdf8f46b-8mfgc    1/1     Running   0          2m
web-app-8bdf8f46b-j7fkv    1/1     Running   0          2m6s
web-app-8bdf8f46b-x7lfg    1/1     Running   0          2m7s
mingw_zhang123@cloudshell:~ (marine-clarity-475016-m8)$ kubectl get pods -o jsonpath='{.items[*].spec.containers[*].image}'
mingw_zhang123@cloudshell:~ (marine-clarity-475016-m8)$ kubectl get pods -o jsonpath='{.items[*].spec.containers[*].image}'
nginx:1.27.0 nginx:1.27.0 nginx:1.27.0 nginx:1.27.0 nginx:1.27.0mingw_zhang123@cloudshell:~ (marine-clarity-475016-m8)$
```

5: Delete Application and Stop Cluster

- Delete the application deployment and service:
  kubectl delete -f deployment.yaml

  Output:
  mingw_zhang123@cloudshell:~ (marine-clarity-475016-m8)$ kubectl delete -f deployment.yaml
  deployment.apps "web-app" deleted
  service "web-service" deleted

- Verify:
  kubectl get all

  Output:
  mingw_zhang123@cloudshell:~ (marine-clarity-475016-m8)$ kubectl get all
  NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
  service/kubernetes ClusterIP 34.118.224.1 <none> 443/TCP 107m

- Delete entire cluster:
  gcloud container clusters delete my-cluster --zone us-central1-a

  Output:
  mingw_zhang123@cloudshell:~ (marine-clarity-475016-m8)$ gcloud container clusters delete my-cluster --zone us-central1-a
  The following clusters will be deleted.
  - [my-cluster] in [us-central1-a]

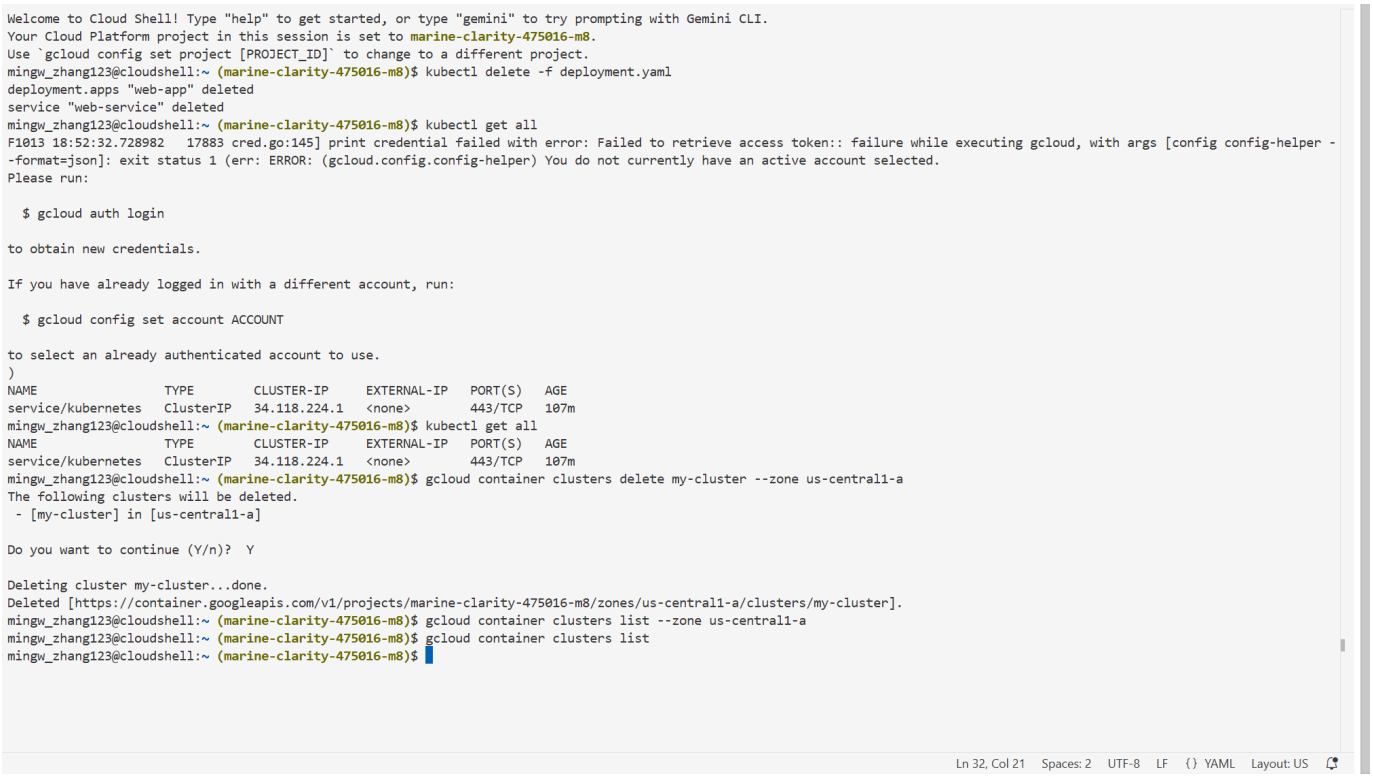
Do you want to continue (Y/n)? Y

Deleting cluster my-cluster...done.
Deleted
[https://container.googleapis.com/v1/projects/marine-clarity-475016-m8/zones/us-central1-a/clusters/my-cluster].

Verify Deletion:
gcloud container clusters list --zone us-central1-a

Output:
[Nothing]
(If output is nothing, then it means the cluster list is empty).

Purpose:
I started by deleting the deployed application and its service using kubectl delete -f deployment.yaml. This command removes all resources defined in the deployment file. The output confirmed successful deletion of both web-app and web-service. To verify, I ran kubectl get all, which showed only the default Kubernetes service remaining, confirming that the deployment and pods were removed. Next, I deleted the entire cluster using gcloud container clusters delete my-cluster --zone us-central1-a, which permanently shut down the GKE cluster. The command asked for confirmation, and after typing Y, the output showed that the cluster was deleted successfully. Last, I verified by running gcloud container clusters list --zone us-central1-a, and since there was no output, it confirmed the cluster list was empty, meaning the environment was fully cleaned up.



```
Welcome to Cloud Shell! Type "help" to get started, or type "gemini" to try prompting with Gemini CLI.
Your Cloud Platform project in this session is set to marine-clarity-475016-m8.
Use `gcloud config set project [PROJECT_ID]` to change to a different project.
mingw_zhang123@cloudshell:~ (marine-clarity-475016-m8)$ kubectl delete -f deployment.yaml
deployment.apps "web-app" deleted
service "web-service" deleted
mingw_zhang123@cloudshell:~ (marine-clarity-475016-m8)$ kubectl get all
F1013 18:52:32.728982   17883 cred.go:145] print credential failed with error: Failed to retrieve access token:: failure while executing gcloud, with args [config config-helper --format=json]: exit status 1 (err: ERROR: (gcloud.config.config-helper) You do not currently have an active account selected.
Please run:

  $ gcloud auth login

to obtain new credentials.

If you have already logged in with a different account, run:

  $ gcloud config set account ACCOUNT

to select an already authenticated account to use.
)
NAME                 TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   34.118.224.1   <none>        443/TCP   107m
mingw_zhang123@cloudshell:~ (marine-clarity-475016-m8)$ kubectl get all
NAME                 TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   34.118.224.1   <none>        443/TCP   107m
mingw_zhang123@cloudshell:~ (marine-clarity-475016-m8)$ gcloud container clusters delete my-cluster --zone us-central1-a
The following clusters will be deleted.
 - [my-cluster] in [us-central1-a]

Do you want to continue (Y/n)?  Y

Deleting cluster my-cluster...done.
Deleted [https://container.googleapis.com/v1/projects/marine-clarity-475016-m8/zones/us-central1-a/clusters/my-cluster].
mingw_zhang123@cloudshell:~ (marine-clarity-475016-m8)$ gcloud container clusters list --zone us-central1-a
mingw_zhang123@cloudshell:~ (marine-clarity-475016-m8)$ gcloud container clusters list
mingw_zhang123@cloudshell:~ (marine-clarity-475016-m8)$
```


- Final Check (Kubernetes cluster is gone and kubectl no longer has an active connection to any cluster):
  kubectl get nodes

  Output:
  mingw_zhang123@cloudshell:~ (marine-clarity-475016-m8)$ kubectl get nodes
  E1013 19:06:28.941822   19078 memcache.go:265] "Unhandled Error" err="couldn't get current server API group list: Get \"http://localhost:8080/api?timeout=32s\": dial tcp 127.0.0.1:8080: connect: connection refused"
  E1013 19:06:28.944376   19078 memcache.go:265] "Unhandled Error" err="couldn't get current server API group list: Get \"http://localhost:8080/api?timeout=32s\": dial tcp 127.0.0.1:8080: connect: connection refused"
  E1013 19:06:28.946327   19078 memcache.go:265] "Unhandled Error" err="couldn't get current server API group list: Get \"http://localhost:8080/api?timeout=32s\": dial tcp 127.0.0.1:8080: connect: connection refused"
  E1013 19:06:28.951614   19078 memcache.go:265] "Unhandled Error" err="couldn't get current server API group list: Get \"http://localhost:8080/api?timeout=32s\": dial tcp 127.0.0.1:8080: connect: connection refused"

E1013 19:06:28.953814 19078 memcache.go:265] "Unhandled Error" err="couldn't get current server API group list: Get \"http://localhost:8080/api?timeout=32s\": dial tcp 127.0.0.1:8080: connect: connection refused"
The connection to the server localhost:8080 was refused - did you specify the right host or port?

Purpose:
To make sure everything was completely deleted, I ran kubectl get nodes to check if there were any remaining cluster connections. Since the cluster was already deleted, the command failed to connect, showing multiple connection refused errors. These errors mean that kubectl no longer has an active API server to communicate with because the Kubernetes cluster no longer exists. The final message, "The connection to the server localhost:8080 was refused", confirms that there are no nodes or active clusters left, indicating a full cleanup and successful teardown of the entire environment.

Error Encountered:

- Permission error:
  mingw_zhang123@cloudshell:~ (project-1-container-mingwei-z)$ gcloud container clusters create my-cluster \
  --num-nodes=2 \
  --zone us-central1-a \
  --machine-type e2-small \
  --disk-size 20GB
  Note: Your Pod address range (`--cluster-ipv4-cidr`) can accommodate at most 1008 node(s).
  ERROR: (gcloud.container.clusters.create) ResponseError: code=403, message=Permission denied on resource project project-1-container-mingwei-z. This command is authenticated as mingw.zhang123@gmail.com which is the active account specified by the [core/account] property.

  Cause: Free trial not activated - no billing account linked to project
  Solution: Activated $300 Google Cloud free trial and

- Error 2: File Not Found:
  error: the path "deployment.yaml" does not exist

  Cause: Created file in different Cloud Shell tab/session
  Fix: Recreated deployment.yaml in current session using cat command

- Error 3: Idle Too Long:

print credential failed with error: Failed to retrieve access token:: failure while executing gcloud
ERROR: (gcloud.config.config-helper) You do not currently have an active account selected.

Cause: Cloud Shell session loses its temporary authentication token after a long idle period.
Fix: Refreshing or reopening Cloud Shell automatically restores the session.