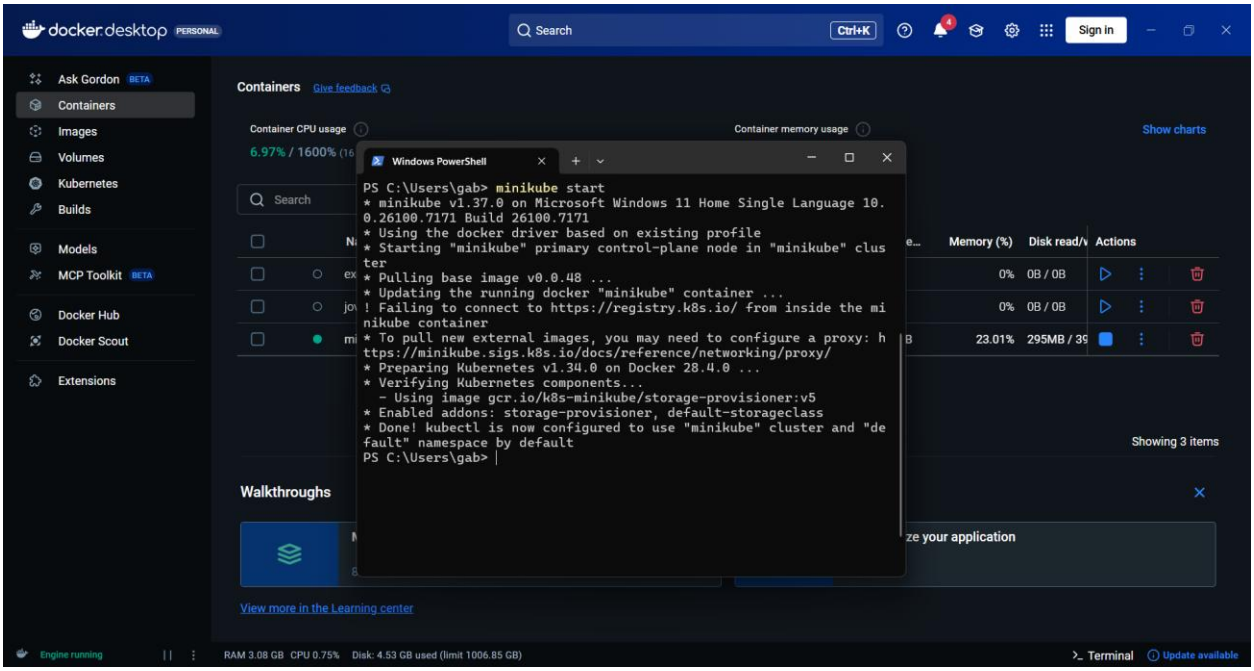
Faith Gabrielle Gamboa
IV – BCSAD

Assignment # 5
November 26, 2025

# Hello Minikube

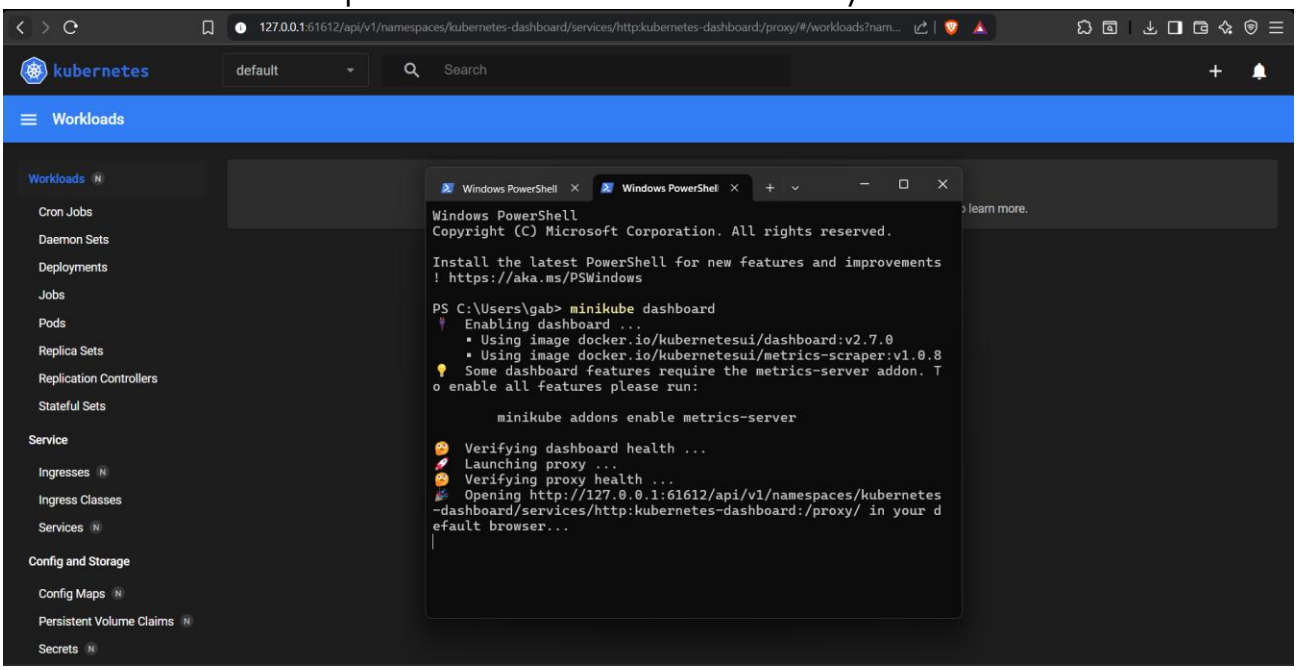
**Step 1: Create a minikube cluster**

a. Launch Docker Desktop and wait until it shows “Docker is running.”
b. Open Windows PowerShell (or PowerShell as Administrator if needed).
c. Run Minikube with:
    i. `minikube start`



**Step 2: Open the dashboard**

a. On a new terminal, run:
   i. `minikube dashboard`
b. Minikube will generate a link in the terminal.
c. Click the link to open the Kubernetes Dashboard in your browser.




**Step 3: Create a Deployment**

a. In your terminal, create a deployment by running:
   i. `kubectl create deployment hello-node --image=registry.k8s.io/e2e-test-images/agnhost:2.53 -- /agnhost netexec --http-port=8080`

```
PS C:\Users\gab> 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
```

b. To check that your deployment was created successfully, run:
   i. `kubectl get deployments`
c. It should display this output:

```
PS C:\Users\gab> kubectl get deployments
NAME         READY   UP-TO-DATE   AVAILABLE   AGE
hello-node   1/1     1            1           6m14s
```

d. To see the pod created by your deployment, run:
   i. `kubectl get pods`
e. It should display this output:

```
PS C:\Users\gab> kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
hello-node-6c9b5f4b59-lh5tj   1/1     Running   0          8m16s
```

f. To view the cluster events, run:
   i. `kubectl get events`

g. It should display this output:

```
PS C:\Users\gab> kubectl get events
LAST SEEN   TYPE     REASON                   OBJECT                                  MESSAGE
8m21s       Normal   Scheduled                pod/hello-node-6c9b5f4b59-lh5tj   Successfully assigned default/hello-node-6c9b5f4b59-lh5tj to minikube
8m21s       Normal   Pulling                  pod/hello-node-6c9b5f4b59-lh5tj   Pulling image "registry.k8s.io/e2e-test-images/agnhost:2.53"
8m1s        Normal   Pulled                   pod/hello-node-6c9b5f4b59-lh5tj   Successfully pulled image "registry.k8s.io/e2e-test-images/agnhost:2.53" i
n 20.125s (20.125s including waiting). Image size: 139374622 bytes.
8m1s        Normal   Created                  pod/hello-node-6c9b5f4b59-lh5tj   Created container: agnhost
8m          Normal   Started                  pod/hello-node-6c9b5f4b59-lh5tj   Started container agnhost
8m22s       Normal   SuccessfulCreate         replicaset/hello-node-6c9b5f4b59   Created pod: hello-node-6c9b5f4b59-lh5tj
8m22s       Normal   ScalingReplicaSet        deployment/hello-node             Scaled up replica set hello-node-6c9b5f4b59 from 0 to 1
40m         Normal   Starting                 node/minikube                     Starting kubelet.
40m         Normal   NodeAllocatableEnforced  node/minikube                     Updated Node Allocatable limit across pods
40m         Normal   NodeHasSufficientMemory  node/minikube                     Node minikube status is now: NodeHasSufficientMemory
40m         Normal   NodeHasNoDiskPressure    node/minikube                     Node minikube status is now: NodeHasNoDiskPressure
40m         Normal   NodeHasSufficientPID     node/minikube                     Node minikube status is now: NodeHasSufficientPID
40m         Normal   RegisteredNode           node/minikube                     Node minikube event: Registered Node minikube in Controller
40m         Normal   Starting                 node/minikube
30m         Normal   Starting                 node/minikube
30m         Normal   RegisteredNode           node/minikube                     Node minikube event: Registered Node minikube in Controller
```

h. To view the kubectl configuration, run:
   i. `kubectl config view`

i. It should display this output:

```
PS C:\Users\gab> kubectl config view
apiVersion: v1
clusters:
- cluster:
    certificate-authority: C:\Users\gab\.minikube\ca.crt
    extensions:
    - extension:
        last-update: Wed, 26 Nov 2025 10:56:21 +08
        provider: minikube.sigs.k8s.io
        version: v1.37.0
      name: cluster_info
    server: https://127.0.0.1:52589
  name: minikube
contexts:
- context:
    cluster: minikube
    extensions:
    - extension:
        last-update: Wed, 26 Nov 2025 10:56:21 +08
        provider: minikube.sigs.k8s.io
        version: v1.37.0
      name: context_info
    namespace: default
    user: minikube
  name: minikube
current-context: minikube
kind: Config
users:
- name: minikube
  user:
    client-certificate: C:\Users\gab\.minikube\profiles\minikube\client.crt
    client-key: C:\Users\gab\.minikube\profiles\minikube\client.key
```

j. To view application logs for a container in a pod, run this command:
   i. `kubectl logs hello-node-6c9b5f4b59-lh5tj`

k. Make sure to replace '`hello-node-6c9b5f4b59-lh5tj`' with the actual pod name from the '`kubectl get pods`' command output.

```
PS C:\Users\gab> kubectl logs hello-node-6c9b5f4b59-lh5tj
I1126 03:18:47.122808       1 log.go:245] Started HTTP server on port 8080
I1126 03:18:47.124534       1 log.go:245] Started UDP server on port  8081
```

**Step 4: Create a Service**

a. To expose the Pod to the public internet, run this command:
   i. `kubectl expose deployment hello-node --type=LoadBalancer --port=8080`
   ii. `--type=LoadBalancer` → Exposes the Pod outside the cluster (works with Minikube's internal LoadBalancer setup).
   iii. `--port=8080` → Maps to the container's port.

b. `It should display this output:`

```
PS C:\Users\gab> kubectl expose deployment hello-node --type=LoadBalancer --port=8080
service/hello-node exposed
```

c. To view the service created, run this command:
   i. `kubectl get services`

d. It should display this output:

```
PS C:\Users\gab> kubectl get services
NAME         TYPE           CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
hello-node   LoadBalancer   10.102.34.63   <pending>     8080:32390/TCP   9s
kubernetes   ClusterIP      10.96.0.1      <none>        443/TCP          72m
```

e. Lastly, run this command:
   i. `minikube service hello-node`

```
PS C:\Users\gab> minikube service hello-node
|-----------|------------|-------------|---------------------------|
| NAMESPACE |    NAME    | TARGET PORT |            URL            |
|-----------|------------|-------------|---------------------------|
| default   | hello-node | 8080        | http://192.168.49.2:32390 |
|-----------|------------|-------------|---------------------------|
🏃  Starting tunnel for service hello-node./
|-----------|------------|-------------|------------------------|
| NAMESPACE |    NAME    | TARGET PORT |          URL           |
|-----------|------------|-------------|------------------------|
| default   | hello-node |             | http://127.0.0.1:61721 |
|-----------|------------|-------------|------------------------|
🏃  Starting tunnel for service hello-node.
🎉  Opening service default/hello-node in default browser...
❗  Because you are using a Docker driver on windows, the terminal needs to be open to run it.
```

f. It would also open a browser window that serves as the response after running the command.


g. It is important to note to only do this in a local Minikube cluster for testing. Do not expose `agnhost` with the `/shell` endpoint on a public or production cluster.

**Step 5: Enable addons**

a. To view all the currently supported addons, run this command:
   i. `minikube addons list`
b. It should display this output

```
PS C:\Users\gab> minikube addons list
```

| ADDON NAME | PROFILE | STATUS | MAINTAINER |
|---|---|---|---|
| ambassador | minikube | disabled | 3rd party (Ambassador) |
| amd-gpu-device-plugin | minikube | disabled | 3rd party (AMD) |
| auto-pause | minikube | disabled | minikube |
| cloud-spanner | minikube | disabled | Google |
| csi-hostpath-driver | minikube | disabled | Kubernetes |
| dashboard | minikube | enabled ✅ | Kubernetes |
| default-storageclass | minikube | enabled ✅ | Kubernetes |
| efk | minikube | disabled | 3rd party (Elastic) |
| freshpod | minikube | disabled | Google |
| gcp-auth | minikube | disabled | Google |
| gvisor | minikube | disabled | minikube |
| headlamp | minikube | disabled | 3rd party (kinvolk.io) |
| inaccel | minikube | disabled | 3rd party (InAccel [info@inaccel.com]) |
| ingress | minikube | disabled | Kubernetes |
| ingress-dns | minikube | disabled | minikube |
| inspektor-gadget | minikube | disabled | 3rd party (inspektor-gadget.io) |
| istio | minikube | disabled | 3rd party (Istio) |
| istio-provisioner | minikube | disabled | 3rd party (Istio) |
| kong | minikube | disabled | 3rd party (Kong HQ) |
| kubeflow | minikube | disabled | 3rd party |
| kubetail | minikube | disabled | 3rd party (kubetail.com) |
| kubevirt | minikube | disabled | 3rd party (KubeVirt) |
| logviewer | minikube | disabled | 3rd party (unknown) |
| metallb | minikube | disabled | 3rd party (MetalLB) |
| metrics-server | minikube | disabled | Kubernetes |
| nvidia-device-plugin | minikube | disabled | 3rd party (NVIDIA) |
| nvidia-driver-installer | minikube | disabled | 3rd party (NVIDIA) |
| nvidia-gpu-device-plugin | minikube | disabled | 3rd party (NVIDIA) |
| olm | minikube | disabled | 3rd party (Operator Framework) |
| pod-security-policy | minikube | disabled | 3rd party (unknown) |
| portainer | minikube | disabled | 3rd party (Portainer.io) |
| registry | minikube | disabled | minikube |
| registry-aliases | minikube | disabled | 3rd party (unknown) |
| registry-creds | minikube | disabled | 3rd party (UPMC Enterprises) |
| storage-provisioner | minikube | enabled ✅ | minikube |
| storage-provisioner-gluster | minikube | disabled | 3rd party (Gluster) |
| storage-provisioner-rancher | minikube | disabled | 3rd party (Rancher) |
| volcano | minikube | disabled | third-party (volcano) |
| volumesnapshots | minikube | disabled | Kubernetes |
| yakd | minikube | disabled | 3rd party (marcnuri.com) |

c. To enable an addon, try running this command:
   i. `minikube addons enable metrics-server`
d. This starts the metrics-server Pod in the kube-system namespace.
e. It allows you to view CPU and memory usage for your Pods using commands like:
   i. `kubectl top pods`

```
PS C:\Users\gab> minikube addons enable metrics-server
💡  metrics-server is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
    ▪ Using image registry.k8s.io/metrics-server/metrics-server:v0.8.0
🌟  The 'metrics-server' addon is enabled
```

d. To View the Pod and Service you created by installing that addon:
    i. `kubectl get pod,svc -n kube-system`
e. It should display this output:

```
PS C:\Users\gab> kubectl get pod,svc -n kube-system
NAME                                   READY   STATUS    RESTARTS      AGE
pod/coredns-66bc5c9577-p6dtg           1/1     Running   1 (85m ago)   95m
pod/etcd-minikube                      1/1     Running   1 (85m ago)   95m
pod/kube-apiserver-minikube            1/1     Running   1 (85m ago)   95m
pod/kube-controller-manager-minikube   1/1     Running   1 (85m ago)   95m
pod/kube-proxy-k48qf                   1/1     Running   1 (85m ago)   95m
pod/kube-scheduler-minikube            1/1     Running   1 (85m ago)   95m
pod/metrics-server-85b7d694d7-2bdd4    1/1     Running   0             107s
pod/storage-provisioner                1/1     Running   2 (85m ago)   95m

NAME                     TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)                  AGE
service/kube-dns         ClusterIP   10.96.0.10      <none>        53/UDP,53/TCP,9153/TCP   95m
service/metrics-server   ClusterIP   10.101.245.27   <none>        443/TCP                  107s
```

f. To check the output from metrics-server, run this command:
    i. `kubectl top pods`
g. It should display this output

```
PS C:\Users\gab> kubectl top pods
NAME                          CPU(cores)   MEMORY(bytes)
hello-node-6c9b5f4b59-lh5tj   1m           6Mi
```

h. If it displays 'error: Metrics API not available,' wait for a while, and try again.
i. To disable the metrics-server, run this command:
    i. `minikube addons disable metrics-server`
j. It should display this output:

```
PS C:\Users\gab> minikube addons disable metrics-server
    "The 'metrics-server' addon is disabled
```

## Get a Shell to a Running Container

**Step 1: Getting a shell to a container**

a. Create the pod by running this command:
    i. `kubectl apply -f` https://k8s.io/examples/application/shell-demo.yaml

```
PS C:\Users\gab> kubectl apply -f https://k8s.io/examples/application/shell-demo.yaml
pod/shell-demo created
```

b. To verify if the container is running, run this command:
    i. `kubectl get pod shell-demo`

```
PS C:\Users\gab> kubectl get pod shell-demo
NAME         READY   STATUS    RESTARTS   AGE
shell-demo   1/1     Running   0          27s
```

c. To get a shell into a running container, run this command:
    i. `kubectl exec --stdin --tty shell-demo -- /bin/bash`

```
PS C:\Users\gab> kubectl exec --stdin --tty shell-demo -- /bin/bash
root@minikube:/# ls/
```

d. The `--stdin (or -i)` keeps stdin open, and `--tty (or -t)` allocates a pseudo-terminal. The double dash `--` separates kubectl arguments from the command you want to run inside the container.

e. Explore inside the container. Once in the shell, run commands like:

   i. `ls /` - list the root directory

   ii. `cat /proc/mounts` - view mounted filesystems

   iii. `ps aux` - see running processes

```
root@minikube:/# ls /
bin   dev                docker-entrypoint.sh  home  lib64  mnt  proc  run   srv  tmp  var
boot  docker-entrypoint.d  etc                  lib   media  opt  root  sbin  sys  usr
```

**Step 2: Writing the root page for nginx**

a. Try the nginx demo by creating a test HTML file by running this command:

   i. `echo 'Hello shell demo' > /usr/share/nginx/html/index.html`

```
root@minikube:/# echo 'Hello shell demo' > /usr/share/nginx/html/index.html
```

b. To set a GET request to the nginx server, run this command:

   i. `apt-get update`
   `apt-get install curl`
   `curl http://localhost/`

c. It should display the "Hello shell demo"

```
root@minikube:/# apt-get update
apt-get install curl
curl http://localhost/
Get:1 http://deb.debian.org/debian trixie InRelease [140 kB]
Get:2 http://deb.debian.org/debian trixie-updates InRelease [47.3 kB]
Get:3 http://deb.debian.org/debian-security trixie-security InRelease [43.4 kB]
Get:4 http://deb.debian.org/debian trixie/main amd64 Packages [9670 kB]
Get:5 http://deb.debian.org/debian trixie-updates/main amd64 Packages [5412 B]
Get:6 http://deb.debian.org/debian-security trixie-security/main amd64 Packages [73.1 kB]
Fetched 9979 kB in 4s (2533 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
curl is already the newest version (8.14.1-2+deb13u2).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Hello shell demo
```

d. Once done with shell, run the `'exit'` command

```
root@minikube:/# exit
exit
```

**Step 3: Running individual commands in a container**

a. In an ordinary command window, not shell, list the environment variables in the running container by running this command:
   i. `kubectl exec shell-demo – env`

```
C:\Users\gab>kubectl exec shell-demo -- env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=minikube
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_PORT_443_TCP_PORT=443
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
HELLO_NODE_PORT_8080_TCP_PORT=8080
HELLO_NODE_PORT=tcp://10.102.34.63:8080
HELLO_NODE_PORT_8080_TCP=tcp://10.102.34.63:8080
HELLO_NODE_PORT_8080_TCP_ADDR=10.102.34.63
KUBERNETES_SERVICE_HOST=10.96.0.1
HELLO_NODE_SERVICE_PORT=8080
KUBERNETES_SERVICE_PORT=443
HELLO_NODE_SERVICE_HOST=10.102.34.63
HELLO_NODE_PORT_8080_TCP_PROTO=tcp
NGINX_VERSION=1.29.3
NJS_VERSION=0.9.4
NJS_RELEASE=1~trixie
PKG_RELEASE=1~trixie
DYNPKG_RELEASE=1~trixie
HOME=/root
```

b. Experiment with running other commands such as:
   i. `kubectl exec shell-demo -- ps aux`

```
C:\Users\gab>kubectl exec shell-demo -- ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.2  14764  8832 ?        Ss   04:11   0:00 nginx: master process nginx -g daemon off;
nginx         29  0.0  0.0  15224  3848 ?        S    04:11   0:00 nginx: worker process
nginx         30  0.0  0.0  15224  3592 ?        S    04:11   0:00 nginx: worker process
nginx         31  0.0  0.0  15224  3592 ?        S    04:11   0:00 nginx: worker process
nginx         32  0.0  0.0  15224  3464 ?        S    04:11   0:00 nginx: worker process
nginx         33  0.0  0.0  15224  3592 ?        S    04:11   0:00 nginx: worker process
nginx         34  0.0  0.0  15224  3464 ?        S    04:11   0:00 nginx: worker process
nginx         35  0.0  0.0  15224  3464 ?        S    04:11   0:00 nginx: worker process
nginx         36  0.0  0.0  15224  3464 ?        S    04:11   0:00 nginx: worker process
nginx         37  0.0  0.0  15224  3464 ?        S    04:11   0:00 nginx: worker process
nginx         38  0.0  0.0  15224  3464 ?        S    04:11   0:00 nginx: worker process
nginx         39  0.0  0.0  15224  3464 ?        S    04:11   0:00 nginx: worker process
nginx         40  0.0  0.0  15224  3464 ?        S    04:11   0:00 nginx: worker process
nginx         41  0.0  0.0  15224  3464 ?        S    04:11   0:00 nginx: worker process
nginx         42  0.0  0.0  15224  3592 ?        S    04:11   0:00 nginx: worker process
nginx         43  0.0  0.0  15224  3464 ?        S    04:11   0:00 nginx: worker process
nginx         44  0.0  0.0  15224  3464 ?        S    04:11   0:00 nginx: worker process
root         111  0.0  0.0   4332  3456 pts/0    Ss+  05:02   0:00 /bin/bash
root         237 25.0  0.0   6396  3584 ?        Rs   05:02   0:00 ps aux
```

ii. `kubectl exec shell-demo -- ls /`

```
C:\Users\gab>kubectl exec shell-demo -- ls /
bin
boot
dev
docker-entrypoint.d
docker-entrypoint.sh
etc
home
lib
lib64
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
```

iii. `kubectl exec shell-demo -- cat /proc/1/mounts`

```
C:\Users\gab>kubectl exec shell-demo -- cat /proc/1/mounts
overlay / overlay rw,relatime,lowerdir=/var/lib/docker/overlay2/l/73LNMHFS7BH5AVURIU5JXA65LG:/var/lib/docker/overlay2/l/63TQMVCEEOHA2PMMIQPXK2E2X7:/var/lib/
docker/overlay2/l/VVXGQBIDVETL3T2QE7UHVF33P4:/var/lib/docker/overlay2/l/UQ3LA4UJSFRXGUC3YM5XYLHAH3:/var/lib/docker/overlay2/l/2K5W4LDSVW6ZL4XHW6TUBWYJGP:/va
r/lib/docker/overlay2/l/GUUTESQDRSG7K4JRCJLHWDJ5IW:/var/lib/docker/overlay2/l/AFBGUSYIZACGWHYCQGJBCPZ56G:/var/lib/docker/overlay2/l/NLHRNTW7Q7AZEFVSPXQL3ZMI
4H,upperdir=/var/lib/docker/overlay2/3c440b9509408221c9a123692fbeaaedeae81da271c5e6735e3ce156c59ae733/diff,workdir=/var/lib/docker/overlay2/3c440b9509408221
c9a123692fbeaaedeae81da271c5e6735e3ce156c59ae733/work 0 0
proc /proc proc rw,nosuid,nodev,noexec,relatime 0 0
tmpfs /dev tmpfs rw,nosuid,size=65536k,mode=755 0 0
devpts /dev/pts devpts rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=666 0 0
sysfs /sys sysfs ro,nosuid,nodev,noexec,relatime 0 0
cgroup /sys/fs/cgroup cgroup2 ro,nosuid,nodev,noexec,relatime 0 0
mqueue /dev/mqueue mqueue rw,nosuid,nodev,noexec,relatime 0 0
shm /dev/shm tmpfs rw,nosuid,nodev,noexec,relatime,size=65536k 0 0
/dev/sde /dev/termination-log ext4 rw,relatime 0 0
/dev/sde /etc/resolv.conf ext4 rw,relatime 0 0
/dev/sde /etc/hostname ext4 rw,relatime 0 0
/dev/sde /etc/hosts ext4 rw,relatime 0 0
/dev/sde /usr/share/nginx/html ext4 rw,relatime 0 0
tmpfs /run/secrets/kubernetes.io/serviceaccount tmpfs ro,relatime,size=3858624k,noswap 0 0
proc /proc/bus proc ro,nosuid,nodev,noexec,relatime 0 0
proc /proc/fs proc ro,nosuid,nodev,noexec,relatime 0 0
proc /proc/irq proc ro,nosuid,nodev,noexec,relatime 0 0
proc /proc/sys proc ro,nosuid,nodev,noexec,relatime 0 0
proc /proc/sysrq-trigger proc ro,nosuid,nodev,noexec,relatime 0 0
tmpfs /proc/acpi tmpfs ro,relatime 0 0
tmpfs /proc/interrupts tmpfs rw,nosuid,size=65536k,mode=755 0 0
tmpfs /proc/kcore tmpfs rw,nosuid,size=65536k,mode=755 0 0
tmpfs /proc/keys tmpfs rw,nosuid,size=65536k,mode=755 0 0
tmpfs /proc/latency_stats tmpfs rw,nosuid,size=65536k,mode=755 0 0
tmpfs /proc/timer_list tmpfs rw,nosuid,size=65536k,mode=755 0 0
tmpfs /proc/scsi tmpfs ro,relatime 0 0
tmpfs /sys/firmware tmpfs ro,relatime 0 0
```

c. When a Pod contains multiple containers, the --container (or -c) flag must be provided with the kubectl exec command to specify which container to access.
   i. For example, consider a Pod named my-pod with two containers: main-app and helper-app.
   ii. The following command opens a shell inside the main-app container:
   iii. `kubectl exec -i -t my-pod --container main-app -- /bin/bash`

# Deploying WordPress and MySQL with Persistent Volumes

**Step 1: Create a working directory**

a. Create a new directory named wordpress-k8s and move into it:
   i. `mkdir wordpress-k8s`
   ii. `cd wordpress-k8s`

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

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

PS C:\Users\gab> mkdir wordpress-k8s


    Directory: C:\Users\gab


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        27/11/2025   1:11 pm                wordpress-k8s


PS C:\Users\gab> cd wordpress-k8s
```

**Step 2: Create a kustomization.yaml**

a. There are two ways to create the kustomization.yaml. First, you can create through notepad and paste this.
   i.
```
cat <<EOF >./kustomization.yaml
secretGenerator:
- name: mysql-pass
  literals:
  - password=YOUR_PASSWORD
EOF
```

b. Another method to create this file, is through window power shell by pasting this line:
   i.
```
@"
secretGenerator:
- name: mysql-pass
  literals:
  - password=mypass123
resources:
  - mysql-deployment.yaml
  - wordpress-deployment.yaml
"@ | Out-File -FilePath kustomization.yaml -Encoding utf8
```

```
PS C:\Users\gab\wordpress-k8s> @"
>> secretGenerator:
>> - name: mysql-pass
>>   literals:
>>   - password=mypass123
>> resources:
>>   - mysql-deployment.yaml
>>   - wordpress-deployment.yaml
>> "@ | Out-File -FilePath kustomization.yaml -Encoding utf8
```

**Step 3: Add resource configs for MySQL and WordPress**

a. Download the MySQL deployment configuration file by running this command in window power shell:
   i. `Invoke-WebRequest -Uri https://k8s.io/examples/application/wordpress/mysql-deployment.yaml -OutFile mysql-deployment.yaml`

b. Download the WordPress configuration file by running this command in window power shell:
   i. `Invoke-WebRequest -Uri https://k8s.io/examples/application/wordpress/wordpress-deployment.yaml -OutFile wordpress-deployment.`

```
PS C:\Users\gab\wordpress-k8s> Invoke-WebRequest -Uri https://k8s.io/examples/application/wordpress/mysql-deployment.yaml -OutFile mysql-deployment.yaml
PS C:\Users\gab\wordpress-k8s> Invoke-WebRequest -Uri https://k8s.io/examples/application/wordpress/wordpress-deployment.yaml -OutFile wordpress-deployment.
yaml
```

**Step 4: Verify all 3 files**

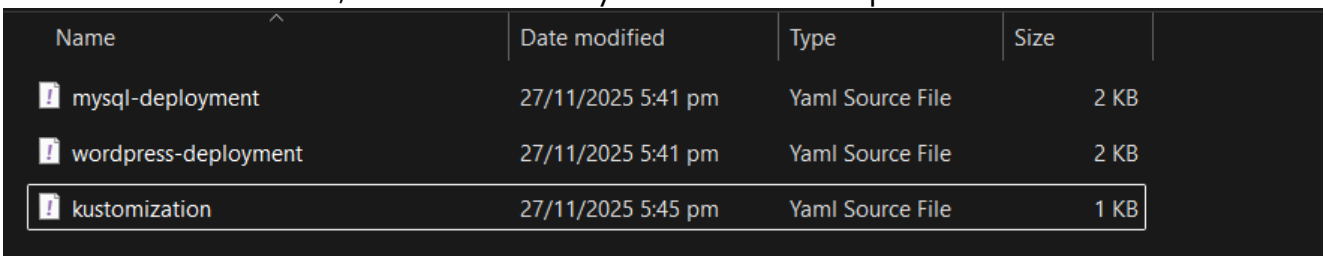
a. To verify that all 3 files exist, run the 'dir' command, and it should display all 3 files.

```
PS C:\Users\gab\wordpress-k8s> dir


    Directory: C:\Users\gab\wordpress-k8s


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        27/11/2025   5:45 pm            142 kustomization.yaml
-a----        27/11/2025   5:41 pm           1442 mysql-deployment.yaml
-a----        27/11/2025   5:41 pm           1341 wordpress-deployment.yaml
```

b. To double check, view the directory itself in the file explorer.



**Step 5: Make sure Minikube is running**

a. Run the following command to check the Minikube status:
   i. `minikube status`
b. Look for the line that shows the host status:
   i. If it displays `'host: Running'`, continue to Step 6.
   ii. If it displays `'host: Stopped'`, or any error message, start Minikube with:
      i. `minikube start`
c. Wait for Minikube to finish starting (typically 1–2 minutes).
d. Minikube must be running in order to create and manage the Kubernetes resources used for the WordPress deployment.

```
PS C:\Users\gab\wordpress-k8s> minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
```

**Step 6: Deploy WordPress and MySQL, and Verify**

a. Apply the WordPress and MySQL configuration by running this command:
   i. `kubectl apply -k ./`
b. It should display this output:

```
PS C:\Users\gab\wordpress-k8s> kubectl apply -k ./
secret/mysql-pass-782846ffc9 created
service/wordpress created
Warning: spec.SessionAffinity is ignored for headless services
service/wordpress-mysql created
persistentvolumeclaim/mysql-pv-claim created
persistentvolumeclaim/wp-pv-claim created
deployment.apps/wordpress created
deployment.apps/wordpress-mysql created
```

c. To verify that the secret was created, run this command:
   i. `kubectl get secrets`
d. It should display this output:

```
PS C:\Users\gab\wordpress-k8s> kubectl get secrets
NAME                    TYPE     DATA   AGE
mysql-pass-782846ffc9   Opaque   1      10s
```

e. To verify that PersistentVolumeClaims were provisioned, check the PVCs by running this command:
   i. `kubectl get pvc`

```
PS C:\Users\gab\wordpress-k8s> kubectl get pvc
NAME           STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
mysql-pv-claim   Bound    pvc-a56bb915-98a0-45a8-9c92-7bae94570f08   20Gi       RWO            standard       <unset>                 60s
wp-pv-claim      Bound    pvc-4aff2191-d5a2-4529-b641-2d97aaed7a47   20Gi       RWO            standard       <unset>                 60s
```

f. Dynamic provisioning may take several minutes before the PVCs reach the Bound state.
g. To verify that the pods are running, check the pod status by running this command:
    i. `kubectl get pods`
h. Both wordpress must say "Running" under STATUS and "1/1" under READY. Hence, it should display this output:

```
PS C:\Users\gab\wordpress-k8s> kubectl get pods
NAME                              READY   STATUS    RESTARTS      AGE
hello-node-6c9b5f4b59-lh5tj       1/1     Running   1 (9h ago)    30h
shell-demo                        1/1     Running   0             5h43m
wordpress-59dfd8d54d-rpkh2        1/1     Running   0             6m8s
wordpress-mysql-5cc66b8b8-gn5gw   1/1     Running   0             6m8s
```

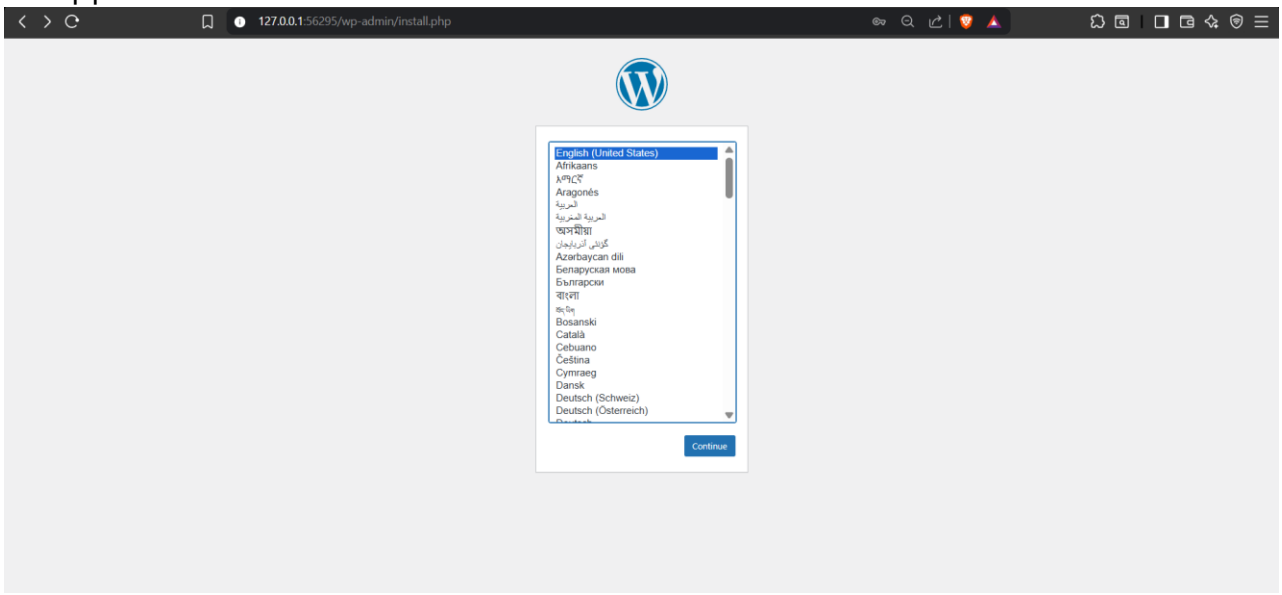
i. Pod creation may take a few minutes before reaching the Running state.
j. To verify that the WordPress Service is active, check the service by running this command:
    i. `kubectl get services wordpress`
k. It should display this output:

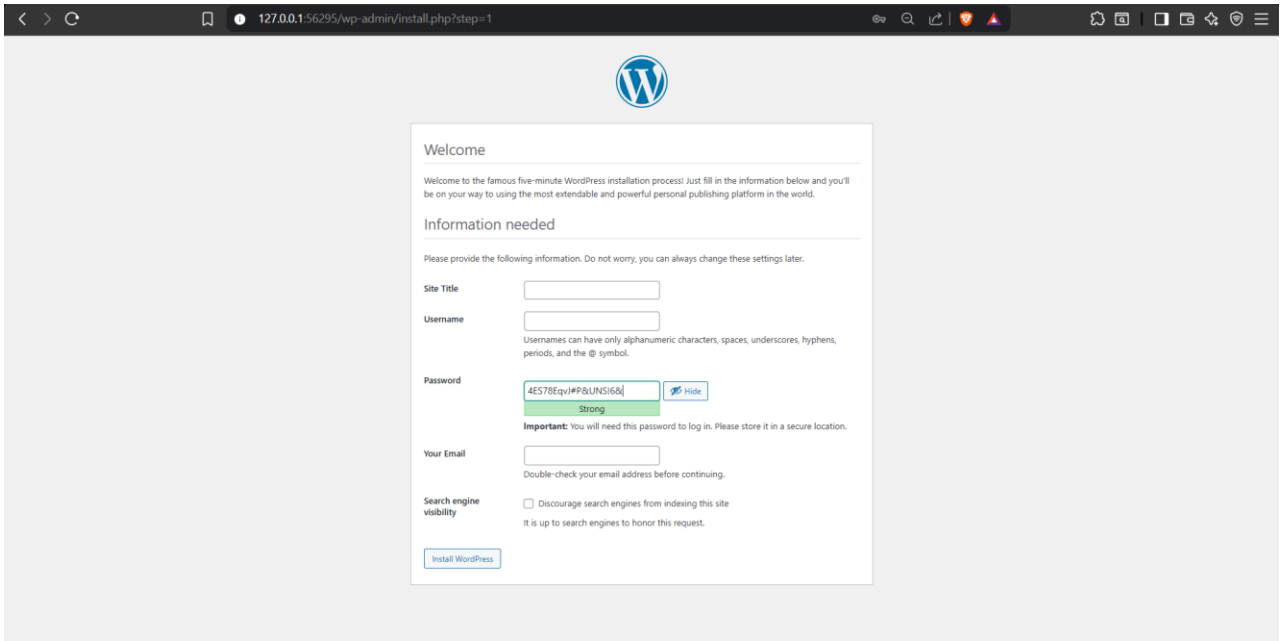
```
PS C:\Users\gab\wordpress-k8s> kubectl get services wordpress
NAME        TYPE           CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
wordpress   LoadBalancer   10.96.43.155   <pending>     80:30860/TCP   6m24s
```

l. In Minikube, LoadBalancer Services do not receive an external IP. The EXTERNAL-IP field will remain in a pending state.
m. To retrieve the WordPress access URL, use Minikube to obtain the Service URL by running this command:
    i. minikube service wordpress –url
n. It should display the url:

```
PS C:\Users\gab\wordpress-k8s> minikube service wordpress --url
http://127.0.0.1:56295
```

o. Access the WordPress setup page by copying the URL from the previous command and opening it in a web browser. The WordPress setup screen should appear.




**Step 7: Cleaning up**

a. To remove all resources created by the kustomization—including the Secret, Deployments, Services, and PersistentVolumeClaims—run:

   i. `kubectl delete -k ./`

b. This command deletes every object defined in the kustomization directory.

c. This prevents unused Pods, Services, Secrets, and PersistentVolumeClaims from remaining in the cluster. Leaving these resources behind can consume system resources, create unnecessary storage usage, and interfere with future deployments or testing.