

Sicat, Sean Russel

IV – ACSAD

ASSIGNMENT # 5 ELEC 3

1. Hello Minikube

Steps:

a. Start Minikube

```
PS C:\Windows\system32> minikube start --driver=docker
* minikube v1.37.0 on Microsoft Windows 11 Pro 10.0.26100.7171 Build 26100.7171
* 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 v0.0.48 ...
  > gcr.io/k8s-minikube/kicbase...: 488.52 MiB / 488.52 MiB 100.00% 10.70 M
* Creating docker container (CPUs=2, Memory=8000MB) ...
! 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 v1.34.0 on Docker 28.4.0 ...
* 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
```

b. Create a First Deployment

```
PS C:\Windows\system32> 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
```

c. View the Deployment

```
PS C:\Windows\system32> kubectl get deployments
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
hello-node    1/1     1            1           38s
```

d. Check Pods

```
PS C:\Windows\system32> kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
hello-node-6c9b5f4b59-vrz5v        1/1     Running   0           65s
```

e. Create a Service

```
PS C:\Windows\system32> kubectl expose deployment hello-node --type=LoadBalancer --port=8080
service/hello-node exposed
```

f. Check the Service

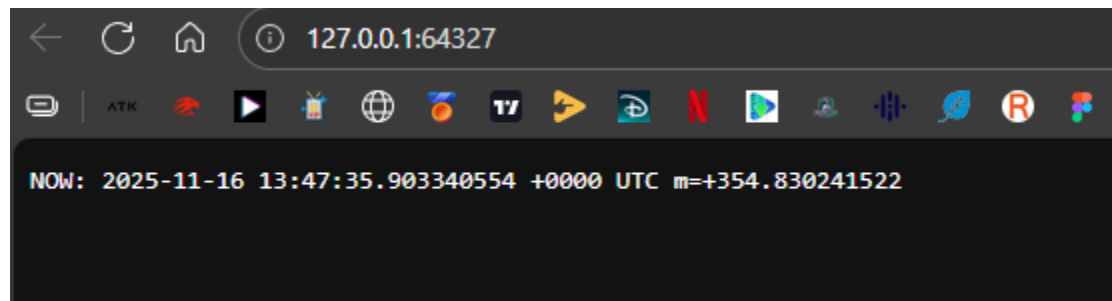
```
PS C:\Windows\system32> kubectl get services
NAME          TYPE          CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
hello-node    LoadBalancer 10.109.2.2    <pending>     8080:31801/TCP   39s
kubernetes    ClusterIP     10.96.0.1    <none>        443/TCP          6m7s
```

g. Check the app's response

```
PS C:\Windows\system32> minikube service hello-node
```

| NAMESPACE | NAME | TARGET PORT | URL |
|-----------|------------|-------------|---------------------------|
| default | hello-node | 8080 | http://192.168.49.2:31801 |

```
* 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.
```



2. Get a Shell to a Running Container

Steps

a. Create the Pod

```
PS C:\Windows\system32> kubectl apply -f https://k8s.io/examples/application/shell-demo.yaml
pod/shell-demo created
```

b. Verify that the container is running

```
PS C:\Windows\system32> kubectl get pod shell-demo
```

| NAME | READY | STATUS | RESTARTS | AGE |
|------------|-------|---------|----------|-------|
| shell-demo | 1/1 | Running | 0 | 6m41s |

c. Get a shell to the running container

```
PS C:\Windows\system32> kubectl exec --stdin --tty shell-demo -- /bin/bash
root@minikube:/#
```

d. Inside the shell run some commands

```
PS C:\Windows\system32> kubectl exec --stdin --tty shell-demo -- /bin/bash
root@minikube:/# ls
bin      dev      docker-entrypoint.sh  home  lib64  mnt  proc  run  srv  tap  var
root     docker-entrypoint.d  etc      lib   media  opt  root  shin sys  usr
root@minikube:/# cat /proc/mounts
overlay / overlay rw,relatime,lowerdir=/var/lib/docker/overlay2/l/ESBQ3HCLGLOKGFQKGBGZLKH4Q:/var/lib/docker/overlay2/l/LI3JSP87ALVSNVHOGVGOEING:/var/lib/docker/overlay2/l/3KEUL3AB42F72HUIJSMWGHORBP:/var/lib/docker/overlay2/l/CQ9KVP1Q
L4R0C3L3U3L3C3:/var/lib/docker/overlay2/l/1496CC8M1Jsh136r0894ZV08:/var/lib/docker/overlay2/l/7NL5R0dWJLFu0P76S380C9M6:/var/lib/docker/overlay2/l/AP6X3J5L1JC6W7LC24J50URJ:/var/lib/docker/overlay2/l/WC7Q9QL364BQ674RTEV4T8D46F_upp
endlr=/var/lib/docker/overlay2/3481935075ad11c8054f5dc94767080719fab017baa1d949484c73a/diff,workdir=/var/lib/docker/overlay2/3481935075ad11c8054f5dc94767080719fab017baa1d949484c73a/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 cgroup1 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=10170192k,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/i915 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/sched tmpfs ro,relatime 0 0
tmpfs /sys/firmware tmpfs ro,relatime 0 0
```

e. Writing root page page for nginx

```
root@minikube:/# echo "Hello shell demo" > /usr/share/nginx/html/index.html
root@minikube:/# curl http://localhost/
Hello shell demo
```

f. Running individual commands in a container

```
PS C:\Windows\system32> kubectl exec shell-demo -- ls /usr/share/nginx/html
index.html
PS C:\Windows\system32> kubectl exec shell-demo -- cat /usr/share/nginx/html/index.html
Hello shell demo
```

3. Deploying Wordpress and MySQL with persistent volumes

Steps

- Create a Folder
- Download the Wordpress and MySQL yaml files from <https://kubernetes.io/docs/tutorials/stateful-application/mysql-wordpress-persistent-volume/> and put it in the created folder
- Create a kustomization.yaml and put this:
secretGenerator: - name: mysql-pass
literals:
- password=YOUR_PASSWORD
resources: - mysql-deployment.yaml - wordpress-deployment.yaml
- Apply

```
PS C:\wordpressmysql> kubectl apply -k ./
secret/mysql-pass-hg8bgc6f55 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
```

e. Verify Secrets

```
PS C:\wordpressmysql> kubectl get secrets
NAME                                TYPE    DATA  AGE
mysql-pass-hg8bgc6f55              Opaque  1      3m17s
```

f. Verify PersistentVolume

```
PS C:\wordpressmysql> kubectl get pvc
NAME              STATUS    VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
mysql-pv-claim    Bound     pvc-7346a9d4-26e4-4bf9-8b34-f6e032496b10  20Gi       RWO            standard       <unset>              2m9s
wp-pv-claim       Bound     pvc-198439c7-0514-4ab7-8c69-4e45b4388336  20Gi       RWO            standard       <unset>              2m9s
```

g. Verify Pods

```
PS C:\wordpressmysql> kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
hello-node-6c9b5f4b59-spxc7        1/1     Running   0          37s
wordpress-7fcf756d5b-dpb5c         1/1     Running   0          8m29s
wordpress-mysql-d976ff876-fjkh7    1/1     Running   0          8m29s
```

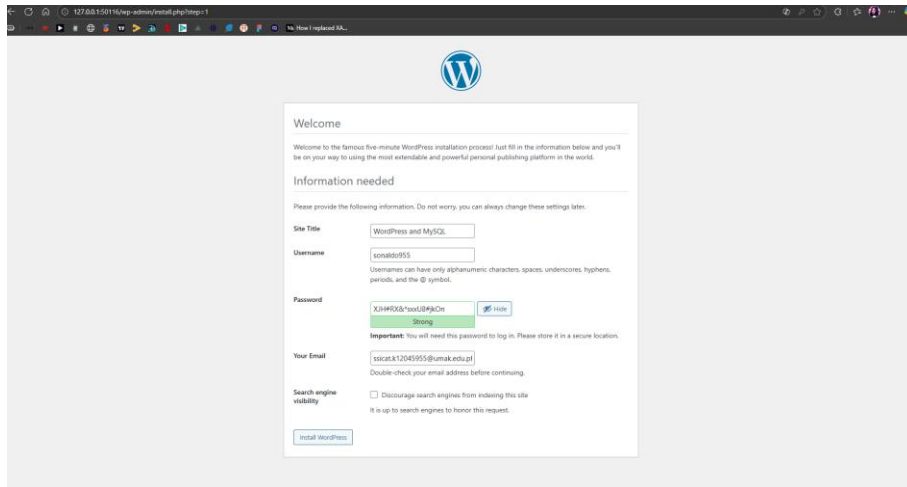
h. Check Services

```
PS C:\wordpressmysql> kubectl get services
NAME              TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
hello-node        LoadBalancer  10.109.2.2      <pending>     8080:31801/TCP   53m
kubernetes         ClusterIP      10.96.0.1       <none>        443/TCP          59m
wordpress         LoadBalancer  10.97.91.225    <pending>     80:32155/TCP     9m25s
wordpress-mysql   ClusterIP      None            <none>        3306/TCP         9m25s
```

i. Get the url for Wordpress

```
PS C:\wordpressmysql> minikube service wordpress --url
http://127.0.0.1:50116
! Because you are using a Docker driver on windows, the terminal needs to be open to run it.
```

j. Finish the WordPress Setup from the Browser:



The screenshot shows the WordPress installation 'Welcome' screen in a browser. The URL is 127.0.0.1:8016/wp-admin/. The page has a light gray background with a white central box containing the WordPress logo and the following text:

Welcome

Welcome to the famous five minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.

Information needed

Please provide the following information. Do not worry, you can always change these settings later.

Site Title
WordPress and MySQL

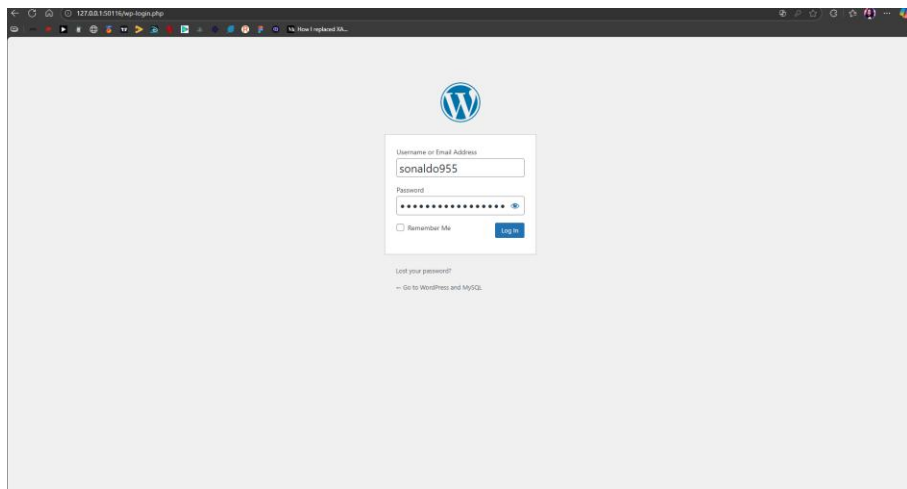
Username
sonald955
Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.

Password
XJH955Gh7mGdMjCm
Important: You will need this password to log in. Please store it in a secure location.

Your Email
sonald955@umak.edu.pl
Double-check your email address before continuing.

Search engine visibility
☐ Discourage search engines from indexing this site
It is up to search engines to honor this request.

[Install WordPress](#)



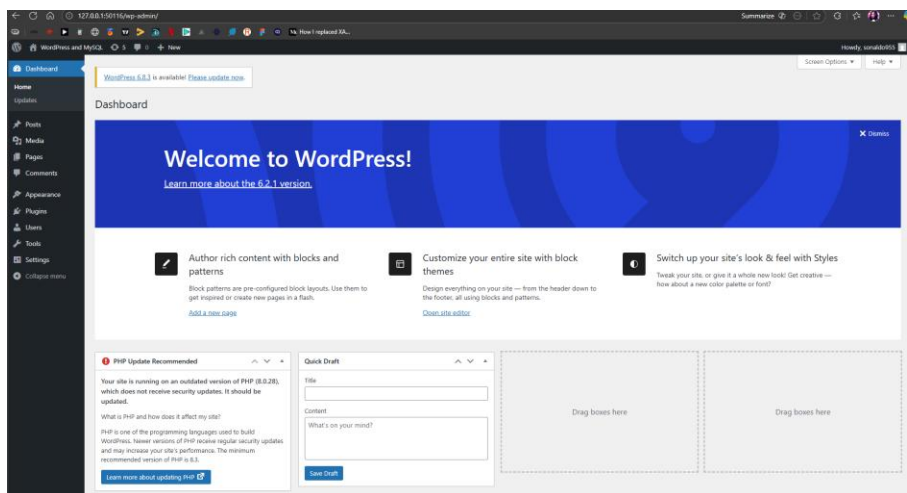
The screenshot shows the WordPress login screen in a browser. The URL is 127.0.0.1:8016/wp-admin/. The page has a light gray background with a white central box containing the WordPress logo and the following text:

Username or Email Address
sonald955

Password
[masked]
☐ Remember Me

[Log In](#)

Lost your password?
[Go to WordPress and MySQL](#)



The screenshot shows the WordPress dashboard in a browser. The URL is 127.0.0.1:8016/wp-admin/. The page has a dark blue sidebar with a menu containing: Dashboard, Home, Updates, Posts, Media, Pages, Comments, Appearance, Plugins, Users, Tools, Settings, and Customize themes. The main content area has a light gray background with a blue header banner that says "Welcome to WordPress! Learn more about the 6.2.1 version." Below the banner are three cards: "Author rich content with blocks and patterns", "Customize your entire site with block themes", and "Switch up your site's look & feel with Styles". At the bottom, there are three widgets: "PHP Update Recommended" (with a red icon), "Quick Draft" (with a text input field), and two empty "Drag boxes here" placeholders.

k. Delete the Resources

```
PS C:\wordpressmysql> kubectl delete -k ./
secret "mysql-pass-hg8bgc6f55" deleted from default namespace
service "wordpress" deleted from default namespace
service "wordpress-mysql" deleted from default namespace
persistentvolumeclaim "mysql-pv-claim" deleted from default namespace
persistentvolumeclaim "wp-pv-claim" deleted from default namespace
deployment.apps "wordpress" deleted from default namespace
deployment.apps "wordpress-mysql" deleted from default namespace
```