LAB 6

Containerized applications deployment and management using Kubernetes

(Part II)

Full name: Tran Dang Khoa

Student ID: B2014926

- Note: screenshots need to be clear and good-looking; submissions must be in PDF format.

Before you begin this lab, you should familiarize yourself with the following Kubernetes concepts in Lab 05:

1. Deploying WordPress and MySQL with Persistent Volumes

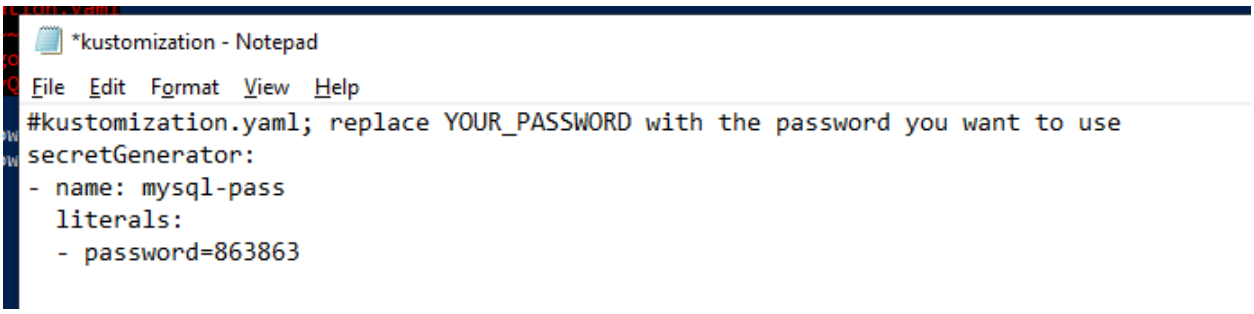
This exercise shows you how to deploy a WordPress site and a MySQL database using Minikube. Both applications use *PersistentVolumes* and *PersistentVolumeClaims* to store data.

A PersistentVolume (PV) is a piece of storage in the cluster that has been manually provisioned by an administrator, or dynamically provisioned by Kubernetes using a StorageClass. A PersistentVolumeClaim (PVC) is a request for storage by a user that can be fulfilled by a PV. PersistentVolumes and PersistentVolumeClaims are independent from Pod lifecycles and preserve data through restarting, rescheduling, and even deleting Pods.

1.1. Create a kustomization.yaml

- Add a Secret generator: a secret is an object that stores a piece of sensitive data like a password or key. Since 1.14, `kubectl` has supported the management of Kubernetes objects using a kustomization file. You can create a Secret by generators in `kustomization.yaml`

```
notepad.exe kustomization.yaml
#kustomization.yaml; replace YOUR_PASSWORD with the password you want to use
secretGenerator:
- name: mysql-pass
  literals:
  - password=YOUR_PASSWORD
```



1.2. Add resource configs for MySQL and WordPress

- Download the MySQL deployment configuration file
  `curl -o mysql-deployment.yaml https://k8s.io/examples/application/wordpress/mysql-deployment.yaml`
- Download the WordPress configuration file.
  `curl -o wordpress-deployment.yaml https://k8s.io/examples/application/wordpress/wordpress-deployment.yaml`
- Add them to `kustomization.yaml` file.

```
resources:
  - mysql-deployment.yaml
  - wordpress-deployment.yaml
```

```
#kustomization.yaml; replace YOUR_PASSWORD with the password you want to use
secretGenerator:
- name: mysql-pass
  literals:
  - password=863863
resources:
  - mysql-deployment.yaml
  - wordpress-deployment.yaml
```

1.3. Apply and Verify

- The `kustomization.yaml` contains all the resources for deploying a WordPress site and a MySQL database. You can apply the directory by
  `kubectl apply -k ./`

```
PS C:\Windows\system32> kubectl apply -k ./
secret/mysql-pass-74gh6d2467 created
service/wordpress created
service/wordpress-mysql created
persistentvolumeclaim/mysql-pv-claim created
persistentvolumeclaim/wp-pv-claim created
deployment.apps/wordpress created
deployment.apps/wordpress-mysql created
PS C:\Windows\system32>
```

- Verify that the Secret exists by running the following command
  `kubectl get secrets`

```
PS C:\Windows\system32> kubectl get secrets
NAME                  TYPE     DATA   AGE
mysql-pass-74gh6d2467   Opaque   1      39s
PS C:\Windows\system32>
```

(take a screenshot)

- Verify that a PersistentVolume got dynamically provisioned
  `kubectl get pvc`

```
S C:\Windows\system32> kubectl get pvc
AME             STATUS   VOLUME                                     CAPACITY   ACCESS MODES
 STORAGECLASS   AGE
ysql-pv-claim    Bound    pvc-12622ae4-e759-4453-ab28-fd86d30c7a4e   20Gi       RWO
 standard       89s
p-pv-claim       Bound    pvc-4da5875b-b6c7-4b09-9a01-938a37c13443   20Gi       RWO
 standard       89s
S C:\Windows\system32>
```

(take a screenshot)

- Verify that the Pod is running by running the following command:
  `kubectl get pods`

```
PS C:\Windows\system32> kubectl get pods
NAME                               READY   STATUS              RESTARTS   AGE
wordpress-57cbfbb74b-dvdtk         0/1     ContainerCreating   0          110s
wordpress-mysql-7985c7fc77-xf5x6   0/1     ContainerCreating   0          110s
PS C:\Windows\system32>
```

(take a screenshot)

- Verify that the Service is running by running the following command:
  `kubectl get services wordpress`

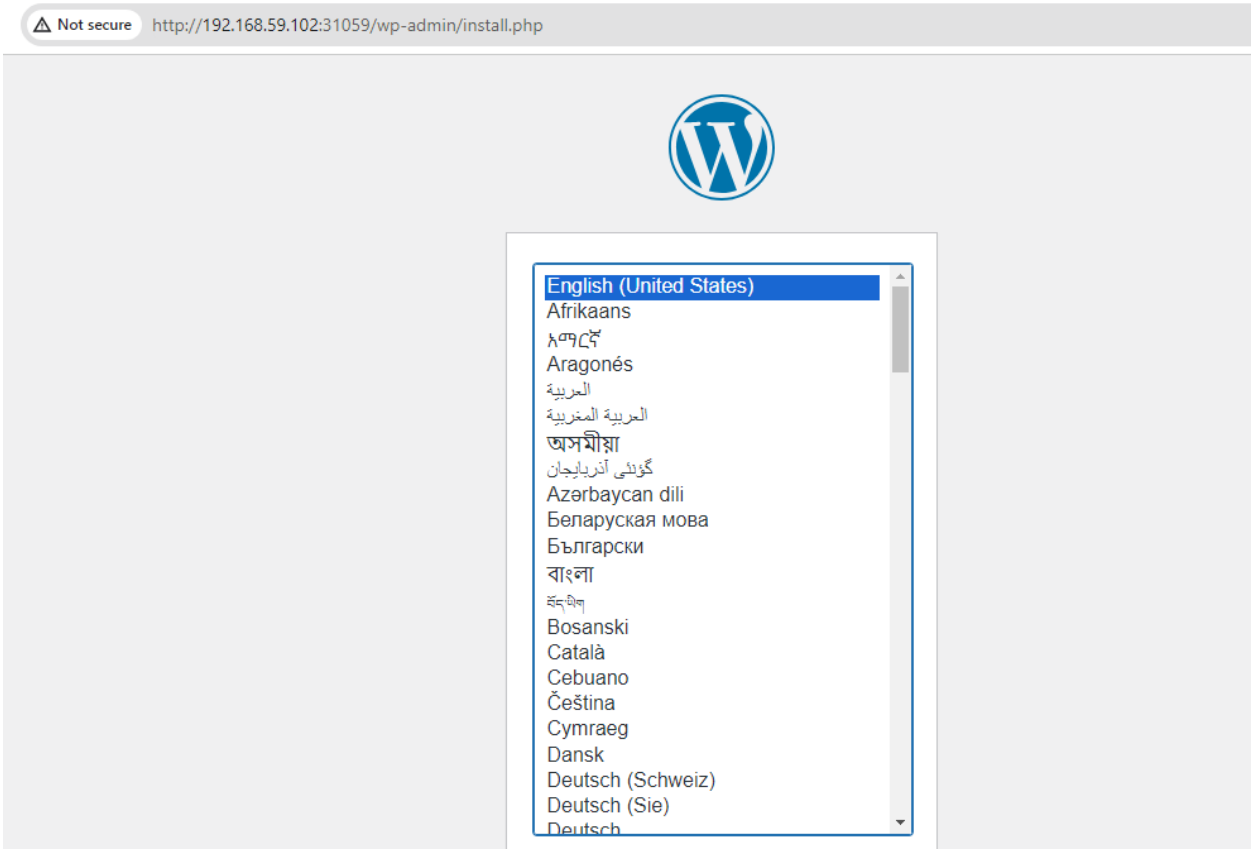
```
PS C:\Windows\system32> kubectl get services wordpress
NAME        TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
wordpress   LoadBalancer   10.109.128.227   <pending>     80:31059/TCP   2m49s
PS C:\Windows\system32>
```

(take a screenshot)

- Run the following command to get the IP Address for the WordPress Service:
  `minikube service wordpress –url`

```
PS C:\Windows\system32> minikube service wordpress --url
http://192.168.59.102:31059
```

- Copy the IP address, and load the page in your browser to view your site.
  (take a screenshot)



1.4. Cleaning up

- Run the following command to delete your Secret, Deployments, Services and PersistentVolumeClaims:

```
kubectl delete -k ./
```

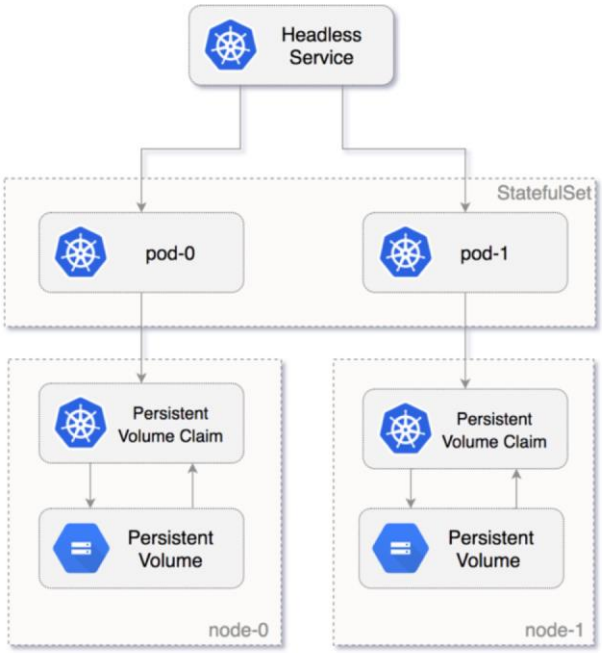
```
PS C:\Windows\system32> kubectl delete -k ./
secret "mysql-pass-74gh6d2467" deleted
service "wordpress" deleted
service "wordpress-mysql" deleted
persistentvolumeclaim "mysql-pv-claim" deleted
persistentvolumeclaim "wp-pv-claim" deleted
deployment.apps "wordpress" deleted
deployment.apps "wordpress-mysql" deleted
PS C:\Windows\system32>
```

2. Kubernetes StatefulSet

StatefulSets and Deployments are two Kubernetes API objects used to manage sets of Pods. The difference between *StatefulSets* and *Deployments* reflects the divide between *stateful* and *stateless* systems. As their name suggests, StatefulSets are designed to run stateful components, while Deployments are used for stateless ones.

Features	**StatefulSet**	**Deployment**
Stateful/Stateless	Stateful	Stateless

Pod identities	Pods are assigned a persistent identifier, derived from the StatefulSet's name and their ordinal creation index.	Pods are assigned random identifiers, derived from the Deployment's name and a unique random string.
Pod interchangeability	Pods in a StatefulSet are not interchangeable. It's expected that each Pod has a specific role, such as always running as a primary or read-only replica for a database application.	All Pods are identical, so they're interchangeable and can be replaced at any time.
Rollout ordering	Pods are guaranteed to be created and removed in sequence. When you scale down the StatefulSet, Kubernetes will terminate the most recently created Pod.	No ordering is supported. When you scale down the Deployment, Kubernetes will terminate a random Pod.
Storage access	Each Pod in the StatefulSet is assigned its own Persistent Volume (PV) and Persistent Volume Claim (PVC)	All Pods share the same PV and PVC



2.1. Creating a StatefulSet

- We will need to use at least two terminal windows. In the first terminal, use `kubectl get` to watch the creation of the StatefulSet's Pods.

```
kubectl get pods --watch -l app=nginx
```

```
PS C:\Windows\system32> kubectl get pods --watch -l app=nginx
```

```
# use this terminal to run commands that specify --watch
# end this watch when you are asked to start a new watch
```

- In the second terminal, use `kubectl apply` to create the headless Service and StatefulSet:

```
kubectl apply -f https://k8s.io/examples/application/web/web.yaml
```

```
PS C:\Windows\system32> kubectl apply -f https://k8s.io/examples/application/web/web.yaml
service/nginx created
statefulset.apps/web created
```

- Examining the Pod's ordinal index

```
kubectl get pods -l app=nginx
```

```
PS C:\Windows\system32> kubectl get pods -l app=nginx
NAME    READY   STATUS              RESTARTS   AGE
web-0   0/1     ContainerCreating   0          29s
PS C:\Windows\system32>
```

- Each Pod has a stable hostname based on its ordinal index.

```
kubectl exec web-0 -- sh -c 'hostname'
```

```
PS C:\Windows\system32> kubectl exec web-0 -- sh -c 'hostname'
web-0
PS C:\Windows\system32>
```

```
kubectl exec web-1 -- sh -c 'hostname'
```

```
PS C:\Windows\system32> kubectl exec web-1 -- sh -c 'hostname'
web-1
```

2.2. Writing to stable storage

- Get the PersistentVolumeClaims for web-0 and web-1:

```
kubectl get pvc -l app=nginx
```

```
PS C:\Windows\system32> kubectl get pvc -l app=nginx
NAME        STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
www-web-0   Bound    pvc-97553502-729c-43b9-af56-740fcf6e0ec8   1Gi        RWO            standard       2m31s
www-web-1   Bound    pvc-8f48e2d8-e8b6-4617-a49a-0b2341fa263c   1Gi        RWO            standard       2m2s
PS C:\Windows\system32>
```

- Write the Pods' hostnames to their index.html files and verify that the NGINX webservers serve the hostnames:

```
kubectl exec web-0 -- sh -c 'echo "$(hostname)" > /usr/share/nginx/html/index.html'
kubectl exec web-1 -- sh -c 'echo "$(hostname)" > /usr/share/nginx/html/index.html'
kubectl exec -i -t web-0 -- curl http://localhost/
kubectl exec -i -t web-1 -- curl http://localhost/
```

(take a screenshot)

```
PS C:\Windows\system32> kubectl exec web-1 -- sh -c 'echo "$(hostname)" > /usr/share/nginx/html/index.html'
PS C:\Windows\system32> kubectl exec web-0 -- sh -c 'echo "$(hostname)" > /usr/share/nginx/html/index.html'
PS C:\Windows\system32> kubectl exec -i -t web-0 -- curl http://localhost/
web-0
PS C:\Windows\system32> kubectl exec -i -t web-1 -- curl http://localhost/
web-1
PS C:\Windows\system32>
```

2.3. Scaling a StatefulSet

- Scale up the number of replicas to 5

```
kubectl scale sts web --replicas=5
```

```
PS C:\Windows\system32> kubectl scale sts web --replicas=5
statefulset.apps/web scaled
PS C:\Windows\system32>
```

- In another terminal window, watch the Pods in the StatefulSet:

```
kubectl get pod --watch -l app=nginx
```

```
PS C:\Windows\system32> kubectl get pod --watch -l app=nginx
NAME    READY   STATUS    RESTARTS   AGE
web-0   1/1     Running   0          4m41s
web-1   1/1     Running   0          4m12s
web-2   1/1     Running   0          22s
web-3   1/1     Running   0          18s
web-4   1/1     Running   0          14s
```

(take a screenshot)

- Scale down the number of replicas to 3

```
kubectl scale sts web --replicas=3
```

```
PS C:\Windows\system32> kubectl scale sts web --replicas=3
statefulset.apps/web scaled
PS C:\Windows\system32>
```

(take a screenshot)

- Get the StatefulSet's Pods and PersistentVolumeClaims

```
kubectl get pods -l app=nginx
kubectl get pvc -l app=nginx
```

```
PS C:\Windows\system32> kubectl get pods -l app=nginx
NAME    READY   STATUS    RESTARTS   AGE
web-0   1/1     Running   0          5m35s
web-1   1/1     Running   0          5m6s
web-2   1/1     Running   0          76s
PS C:\Windows\system32> kubectl get pvc -l app=nginx
NAME        STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
www-web-0   Bound    pvc-97553502-729c-43b9-af56-740fcf6e0ec8   1Gi        RWO            standard       5m48s
www-web-1   Bound    pvc-8f48e2d8-e8b6-4617-a49a-0b2341fa263c   1Gi        RWO            standard       5m19s
www-web-2   Bound    pvc-6032d806-675d-4105-af40-98e03d48dfda   1Gi        RWO            standard       89s
www-web-3   Bound    pvc-af827551-f5ca-425b-96d6-f7e3e4da51b1   1Gi        RWO            standard       85s
www-web-4   Bound    pvc-7dfff7f8-6222-4a9e-808f-4c8cde3c3ea1   1Gi        RWO            standard       81s
PS C:\Windows\system32>
```

Note: the PersistentVolumes mounted to the Pods of a StatefulSet are not deleted when the StatefulSet's Pods are deleted
(take a screenshot)

2.4. Cleaning up

- Run the following command to delete your StatefulSet, Services and PersistentVolumeClaims:

```
kubectl delete sts web
kubectl delete svc nginx
kubectl delete pvc www-web-0 www-web-1 www-web-2 www-web-3 www-web-4
```

```
PS C:\Windows\system32> kubectl delete sts web
statefulset.apps "web" deleted
PS C:\Windows\system32> kubectl delete svc nginx
service "nginx" deleted
PS C:\Windows\system32> kubectl delete pvc www-web-0 www-web-1 www-web-2 www-web-3 www-web-4
persistentvolumeclaim "www-web-0" deleted
persistentvolumeclaim "www-web-1" deleted
persistentvolumeclaim "www-web-2" deleted
persistentvolumeclaim "www-web-3" deleted
persistentvolumeclaim "www-web-4" deleted
PS C:\Windows\system32>
```

3. Run Kubernetes on the public cloud (optional)

Google Kubernetes Engine (GKE)
Amazon Elastic Kubernetes Service (Amazon EKS)
Managed Kubernetes Service (AKS)

---END---