**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. **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

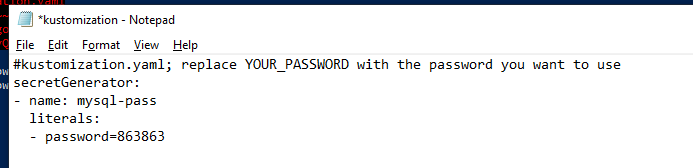
#kustomization.yaml; replace YOUR\_PASSWORD with the password you want to use

secretGenerator:

- name: mysql-pass

literals:

- password=YOUR\_PASSWORD



* 1. **Add resource configs for MySQL and WordPress**
* Download the MySQL deployment [configuration file](https://raw.githubusercontent.com/kubernetes/website/main/content/en/examples/application/wordpress/mysql-deployment.yaml)

curl -o mysql-deployment.yaml https://k8s.io/examples/application/wordpress/mysql-deployment.yaml

* Download the WordPress [configuration file.](https://raw.githubusercontent.com/kubernetes/website/main/content/en/examples/application/wordpress/wordpress-deployment.yaml)

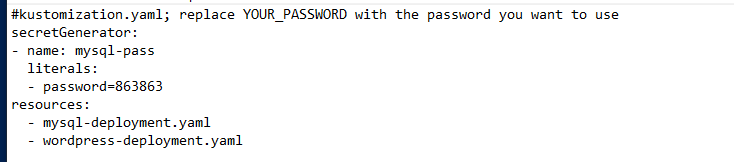
curl -o wordpress-deployment.yaml https://k8s.io/examples/application/wordpress/wordpress-deployment.yaml

* Add them to kustomization.yaml file.

resources:

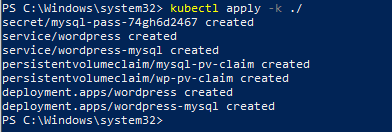
- mysql-deployment.yaml

- wordpress-deployment.yaml



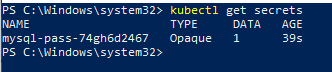
* 1. **Apply and Verify**
* The kustomization.yaml contains all the resources for deploying a WordPress site and a MySQL database. You can apply the directory by

kubectl apply -k ./



* Verify that the Secret exists by running the following command

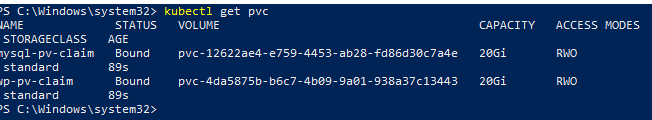
kubectl get secrets



(take a screenshot)

* Verify that a PersistentVolume got dynamically provisioned

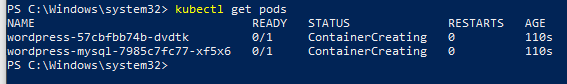
kubectl get pvc



(take a screenshot)

* Verify that the Pod is running by running the following command:

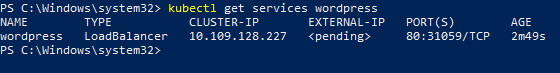
kubectl get pods



(take a screenshot)

* Verify that the Service is running by running the following command:

kubectl get services wordpress



(take a screenshot)

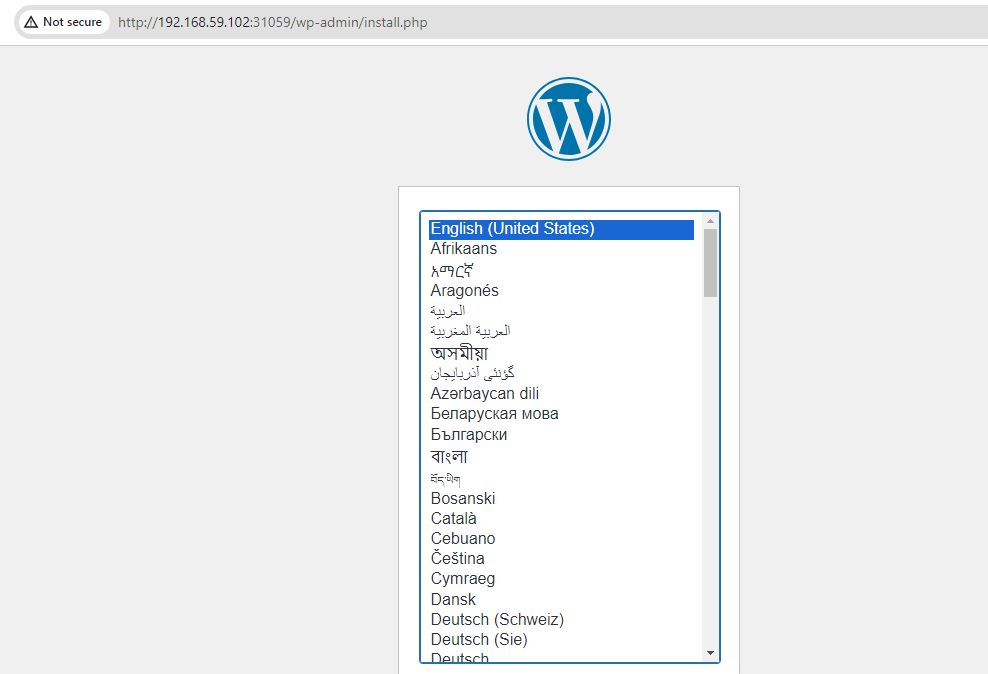
* Run the following command to get the IP Address for the WordPress Service:

minikube service wordpress –url



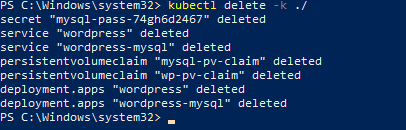
* Copy the IP address, and load the page in your browser to view your site.

(take a screenshot)



* 1. **Cleaning up**
* Run the following command to delete your Secret, Deployments, Services and PersistentVolumeClaims:

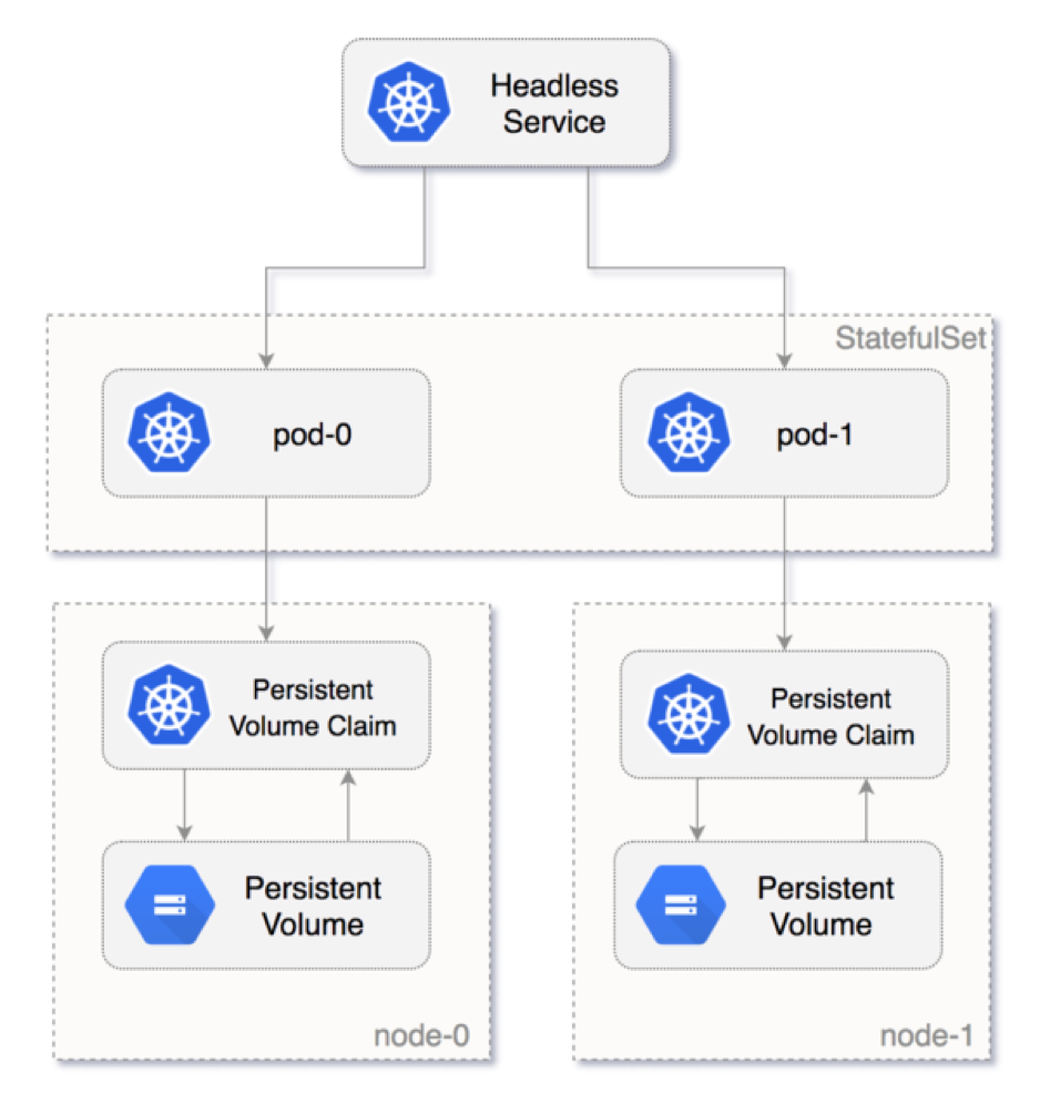
kubectl delete -k ./



1. **Kubernetes StatefulSet**

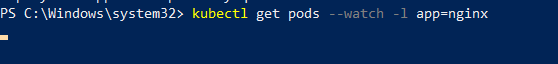
StatefulSets and Deployments are two Kubernetes API objects used to manage sets of Pods. The difference between *StatefulSets* and *Deployments* reflects the divide between *stateful* and *stateless* systems. As their name suggests, StatefulSets are designed to run stateful components, while Deployments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

****

* 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



# use this terminal to run commands that specify --watch

# end this watch when you are asked to start a new watch

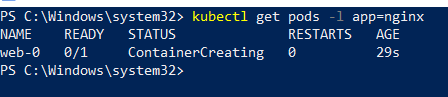
* In the second terminal, use kubectl apply to create the headless Service and StatefulSet:

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



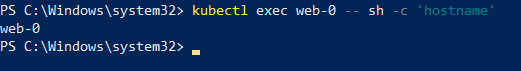
* Examining the Pod's ordinal index

kubectl get pods -l app=nginx



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

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

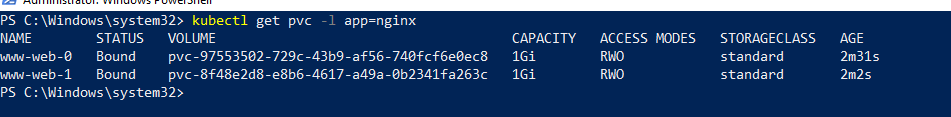


kubectl exec web-1 -- sh -c 'hostname'



* 1. **Writing to stable storage**
* Get the PersistentVolumeClaims for web-0 and web-1:

kubectl get pvc -l app=nginx



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

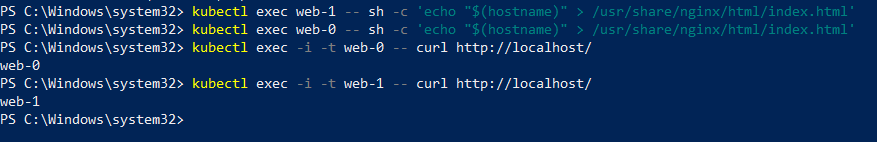
kubectl exec web-0 -- sh -c 'echo "$(hostname)" > /usr/share/nginx/html/index.html'

kubectl exec web-1 -- sh -c 'echo "$(hostname)" > /usr/share/nginx/html/index.html'

kubectl exec -i -t web-0 -- curl <http://localhost/>

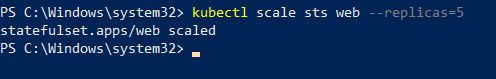
kubectl exec -i -t web-1 -- curl <http://localhost/>

(take a screenshot)



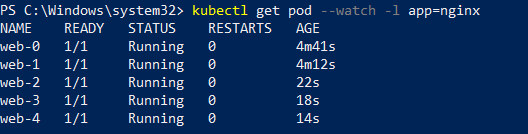
* 1. **Scaling a StatefulSet**
* Scale up the number of replicas to 5

kubectl scale sts web --replicas=5



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

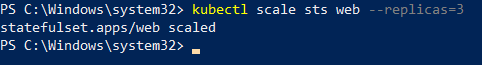
kubectl get pod --watch -l app=nginx



(take a screenshot)

* Scale down the number of replicas to 3

kubectl scale sts web --replicas=3

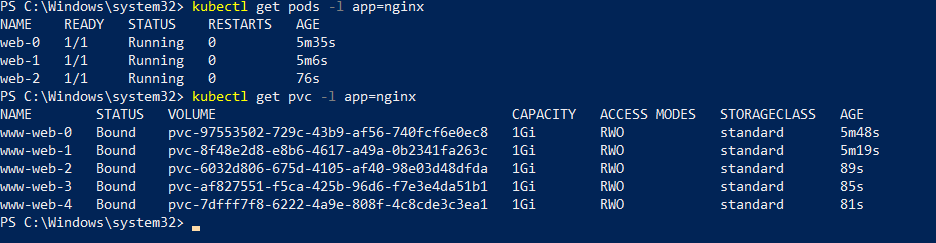


(take a screenshot)

* Get the StatefulSet's Pods and PersistentVolumeClaims

kubectl get pods -l app=nginx

kubectl get pvc -l app=nginx



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

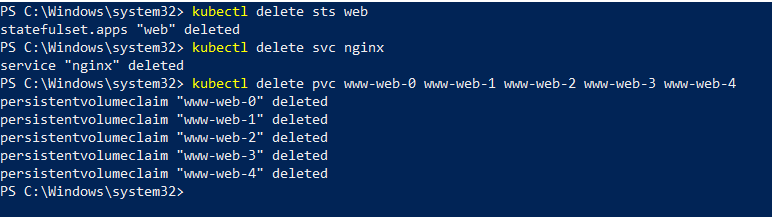
(take a screenshot)

* 1. **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



1. **Run Kubernetes on the public cloud (optional)**

[Google Kubernetes Engine (GKE)](https://cloud.google.com/kubernetes-engine)

[Amazon Elastic Kubernetes Service (Amazon EKS)](https://aws.amazon.com/eks/)

[Managed Kubernetes Service (AKS)](https://azure.microsoft.com/en-us/products/kubernetes-service)

---END---