LAB 6

Containerized applications deployment and management using Kubernetes

(Part II)

Full name: Trương Đặng Trúc Lâm

Student ID: B2111933

- 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

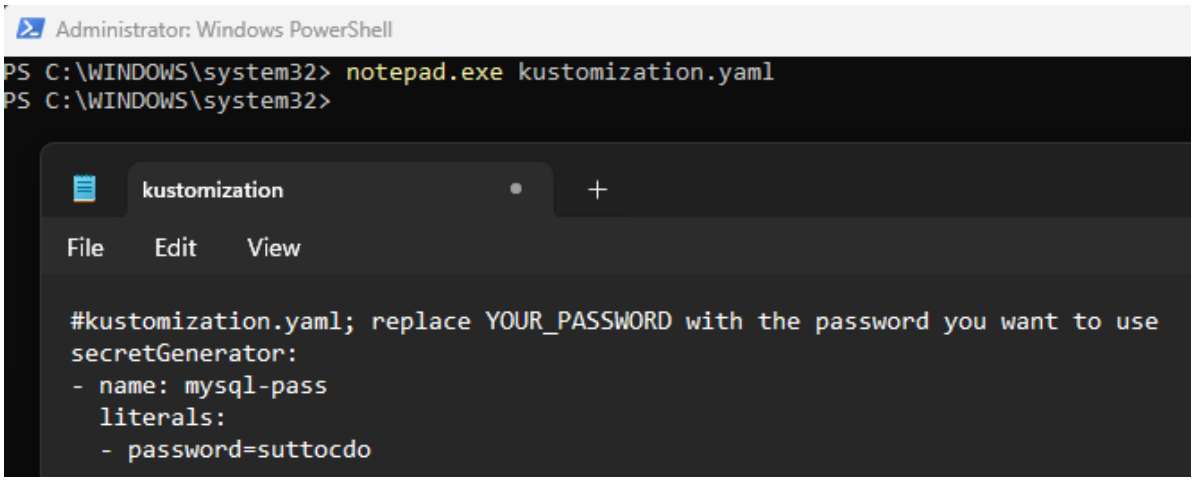
This exercise shows you how to deploy a WordPress site and a MySQL database using Minikube. Both applications use *`PersistentVolumes`* and *`PersistentVolumeClaims`* to store data.

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
```

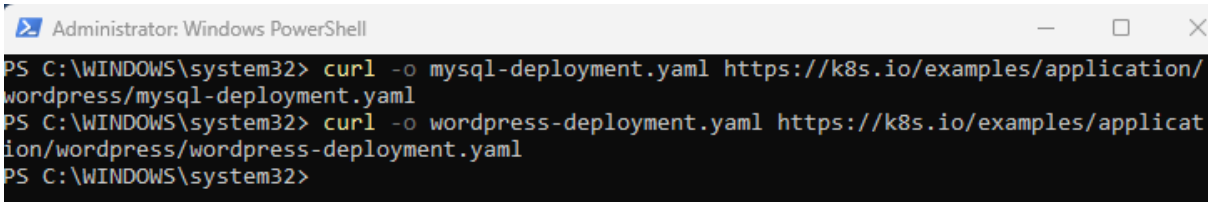


Add a Secret generator

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`



Download the MySQL deployment configuration file and WordPress configuration file.

- Add them to `kustomization.yaml` file.

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

```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> notepad.exe kustomization.yaml
PS C:\WINDOWS\system32>

kustomization
File   Edit   View

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

Add those configuration files into `kustomization.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 ./`

```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> kubectl apply -k ./
secret/mysql-pass-kd4cfd4f87 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>
```

Apply the directory

- Verify that the Secret exists by running the following command
  `kubectl get secrets`
  (take a screenshot)

```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> kubectl get secrets
NAME                    TYPE     DATA   AGE
mysql-pass-kd4cfd4f87   Opaque   1      81s
PS C:\WINDOWS\system32>
```

Verify that the Secret exists

- Verify that a PersistentVolume got dynamically provisioned

  `kubectl get pvc`

  (take a screenshot)

```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> kubectl get pvc
NAME             STATUS   VOLUME                                     CAPACITY   ACCESS MOD
ES   STORAGECLASS   AGE
mysql-pv-claim   Bound    pvc-0590c808-c067-4bfe-a3ce-a03a8be69b50   20Gi       RWO
     standard       2m45s
wp-pv-claim      Bound    pvc-d02b019e-1415-48a6-9ec3-a9872411c471   20Gi       RWO
     standard       2m45s
PS C:\WINDOWS\system32>
```

Verify that a PersistentVolume got dynamically provisioned

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

  `kubectl get pods`

  (take a screenshot)

```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
wordpress-594448cb8b-cjcm8        1/1     Running   0          3m29s
wordpress-mysql-6fd658999c-gf9z8  1/1     Running   0          3m29s
PS C:\WINDOWS\system32>
```

Verify that the Pod is running

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

  `kubectl get services wordpress`

  (take a screenshot)

```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> kubectl get services wordpress
NAME        TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
wordpress   LoadBalancer   10.101.177.70   <pending>     80:30959/TCP   4m22s
PS C:\WINDOWS\system32>
```

Verify that the Service is running

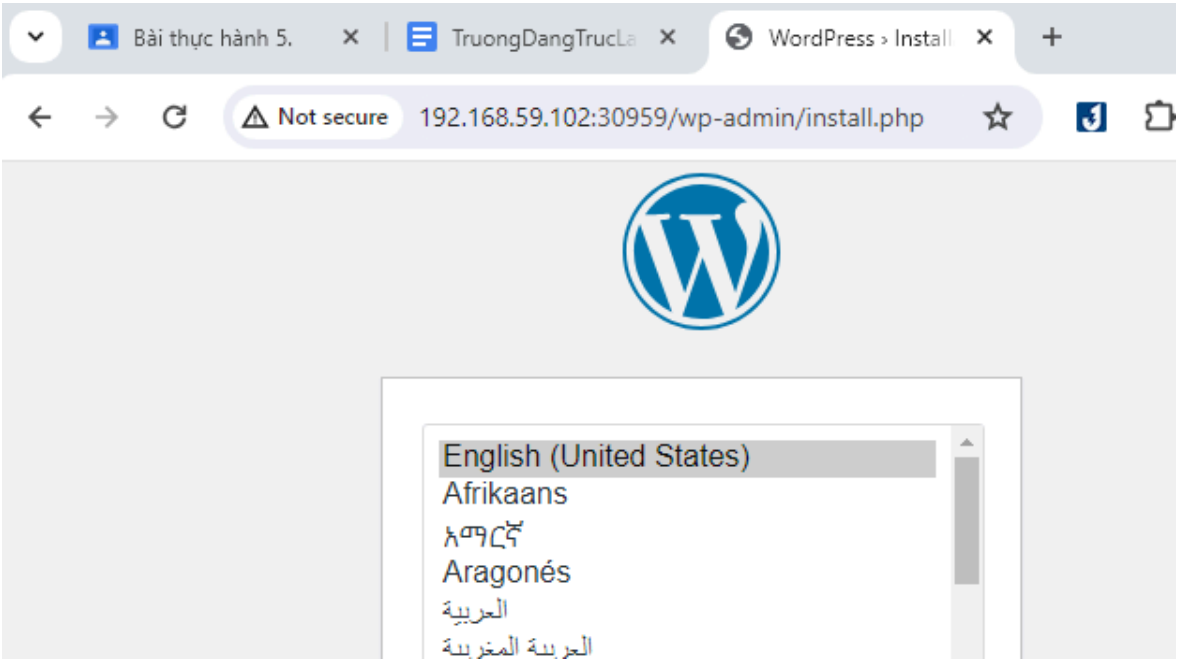
- Run the following command to get the IP Address for the WordPress Service:

  `minikube service wordpress --url`

```
PS C:\WINDOWS\system32> minikube service wordpress --url
W0410 09:51:14.473258   10732 main.go:291] Unable to resolve the current Docker CLI contex
t "default": context "default": context not found: open C:\Users\lambo\.docker\contexts\me
ta\37a8eec1ce19687d132fe29051dca629d164e2c4958ba141d5f4133a33f0688f\meta.json: The system
cannot find the path specified.
http://192.168.59.102:30959
PS C:\WINDOWS\system32>
```

192.168.59.102:30959 is the IP Address of the WordPress Service

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



This is the wordpress site

1.4. Cleaning up

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

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

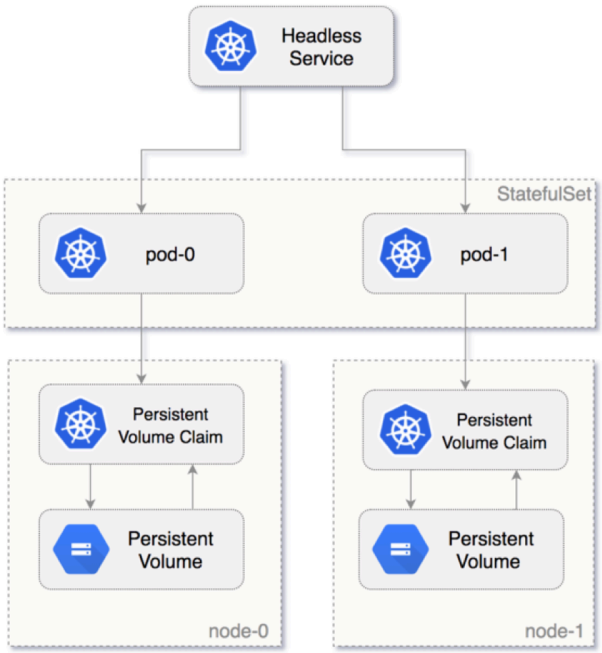
```
PS C:\WINDOWS\system32> kubectl delete -k ./
secret "mysql-pass-kd4cfd4f87" deleted
service "wordpress" deleted
service "wordpress-mysql" deleted
persistentvolumeclaim "mysql-pv-claim" deleted
persistentvolumeclaim "wp-pv-claim" deleted
deployment.apps "wordpress" deleted
deployment.apps "wordpress-mysql" deleted
PS C:\WINDOWS\system32>
```

Delete your Secret, Deployments, Services and PersistentVolumeClaims

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.

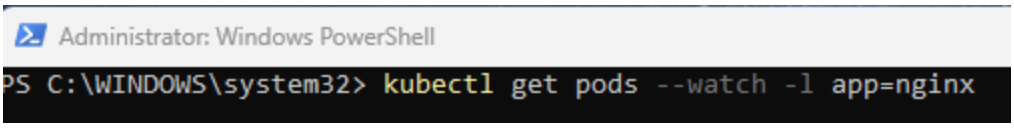
Features	StatefulSet	Deployment
Stateful/Stateless	Stateful	Stateless
Pod identities	Pods are assigned a persistent identifier, derived from the StatefulSet's name and their ordinal creation index.	Pods are assigned random identifiers, derived from the Deployment's name and a unique random string.
Pod 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
# use this terminal to run commands that specify --watch
# end this watch when you are asked to start a new watch
```



Watch the creation of the StatefulSet's Pods in the first terminal

- 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
```

Create the headless Service and StatefulSet in the second terminal

- Examining the Pod's ordinal index

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

```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> kubectl get pods -l app=nginx
NAME     READY   STATUS    RESTARTS   AGE
web-0    1/1     Running   0          91s
web-1    1/1     Running   0          79s
PS C:\WINDOWS\system32>
```

Examining the Pod's ordinal index

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

```
kubectl exec web-1 -- sh -c 'hostname'
kubectl exec web-2 -- sh -c 'hostname'
```

```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> kubectl exec web-1 -- sh -c 'hostname'
web-1
PS C:\WINDOWS\system32> kubectl exec web-0 -- sh -c 'hostname'
web-0
PS C:\WINDOWS\system32>
```

Each Pod has a stable hostname based on its ordinal index

2.2. Writing to stable storage

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

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

```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> kubectl get pvc -l app=nginx
NAME        STATUS   VOLUME                                     CAPACITY   ACCESS MODES
STORAGECLASS   AGE
www-web-0   Bound    pvc-42877e1c-617f-44a2-9f0f-087f644a7c4d   1Gi        RWO
standard       4m2s
www-web-1   Bound    pvc-1e312ea3-fd97-4f04-bf36-5830ac9466c0   1Gi        RWO
standard       3m50s
PS C:\WINDOWS\system32>
```

Get the PersistentVolumeClaims for web-0 and web-1

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

```
kubectl exec web-1 -- sh -c 'echo "$(hostname)" > /usr/share/nginx/html/index.html'
kubectl exec web-2 -- sh -c 'echo "$(hostname)" > /usr/share/nginx/html/index.html'
```

```
kubectl exec -i -t web-1 -- curl http://localhost/
kubectl exec -i -t web-2 -- curl http://localhost/
```

(take a screenshot)

```
Administrator: Windows PowerShell
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-1 -- curl http://localhost/
web-1
PS C:\WINDOWS\system32> kubectl exec -i -t web-0 -- curl http://localhost/
web-0
PS C:\WINDOWS\system32>
```

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

2.3. Scaling a StatefulSet

- Scale up the number of replicas to 5

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

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

Scale up the number of replicas to 5

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

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

(take a screenshot)

```
PS C:\WINDOWS\system32> kubectl get pods --watch -l app=nginx
NAME    READY   STATUS    RESTARTS   AGE
web-0   0/1     Pending   0          0s
web-0   0/1     Pending   0          0s
web-0   0/1     Pending   0          1s
web-0   0/1     ContainerCreating   0          1s
web-0   1/1     Running             0          12s
web-1   0/1     Pending             0          0s
web-1   0/1     Pending             0          0s
web-1   0/1     Pending             0          1s
web-1   0/1     ContainerCreating   0          1s
web-1   1/1     Running             0          3s
web-2   0/1     Pending             0          0s
web-2   0/1     Pending             0          0s
web-2   0/1     Pending             0          2s
web-2   0/1     ContainerCreating   0          2s
web-2   1/1     Running             0          3s
web-3   0/1     Pending             0          0s
web-3   0/1     Pending             0          0s
```

```
web-3   0/1     Pending             0          1s
web-3   0/1     ContainerCreating   0          1s
web-3   1/1     Running             0          2s
web-4   0/1     Pending             0          0s
web-4   0/1     Pending             0          0s
web-4   0/1     Pending             0          1s
web-4   0/1     ContainerCreating   0          1s
web-4   1/1     Running             0          2s
```

In the first terminal, watch the Pods in the StatefulSet

- Scale down the number of replicas to 3

  `kubectl scale sts web --replicas=3`

  (take a screenshot)

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

Scale down the number of replicas to 3

- Get the StatefulSet's Pods and PersistentVolumeClaims

  `kubectl get pods -l app=nginx`

```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> kubectl get pods --watch -l app=nginx
NAME    READY   STATUS    RESTARTS   AGE
web-0   1/1     Running   0          18m
web-1   1/1     Running   0          17m
web-2   1/1     Running   0          7m1s
```

Get the StatefulSet's Pods

`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-42877e1c-617f-44a2-9f0f-087f644a7c4d   1Gi        RWO
standard       19m
www-web-1   Bound    pvc-1e312ea3-fd97-4f04-bf36-5830ac9466c0   1Gi        RWO
standard       18m
www-web-2   Bound    pvc-84c17e6d-f54f-407d-ba75-94df65f5fcab   1Gi        RWO
standard       7m59s
www-web-3   Bound    pvc-b07047e7-1cc8-411f-9f16-47549db44d58   1Gi        RWO
standard       7m56s
www-web-4   Bound    pvc-0c696ac2-14d5-46ba-83c7-9958f42bce1d   1Gi        RWO
standard       7m54s
PS C:\WINDOWS\system32>
```

Get the PersistentVolumeClaims

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

(take a screenshot)

```
PS C:\WINDOWS\system32> kubectl get pvc -l app=nginx
NAME        STATUS   VOLUME                                     CAPACITY   ACCESS MODES
STORAGECLASS   AGE
www-web-0   Bound    pvc-42877e1c-617f-44a2-9f0f-087f644a7c4d   1Gi        RWO
standard       19m
www-web-1   Bound    pvc-1e312ea3-fd97-4f04-bf36-5830ac9466c0   1Gi        RWO
standard       18m
www-web-2   Bound    pvc-84c17e6d-f54f-407d-ba75-94df65f5fcab   1Gi        RWO
standard       7m59s
www-web-3   Bound    pvc-b07047e7-1cc8-411f-9f16-47549db44d58   1Gi        RWO
standard       7m56s
www-web-4   Bound    pvc-0c696ac2-14d5-46ba-83c7-9958f42bce1d   1Gi        RWO
standard       7m54s
PS C:\WINDOWS\system32> kubectl get pods --watch -l app=nginx
NAME    READY   STATUS    RESTARTS   AGE
web-0   1/1     Running   0          20m
web-1   1/1     Running   0          20m
web-2   1/1     Running   0          9m17s
```

Comparison

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
```

```
Administrator: Windows PowerShell
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>
```

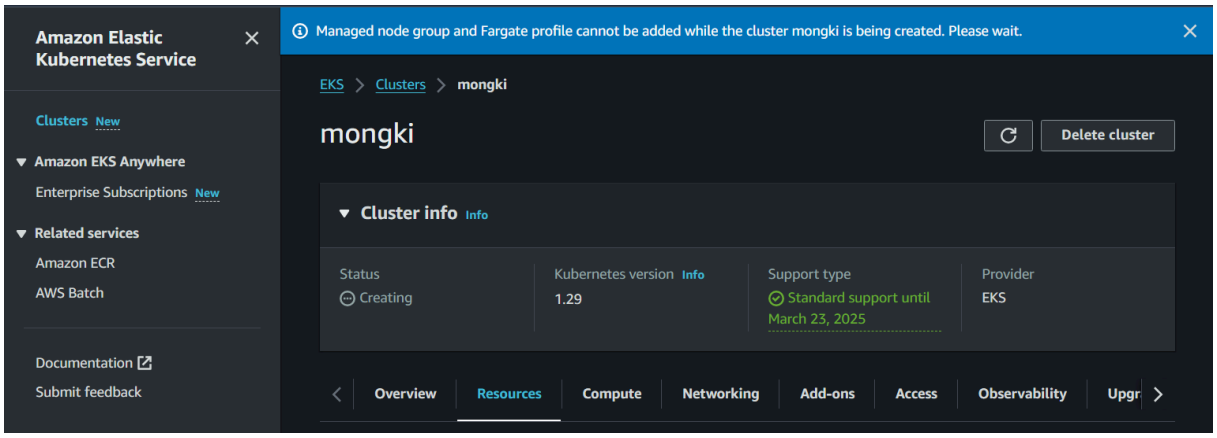
Delete StatefulSet, Services and PersistentVolumeClaims

3. Run Kubernetes on the public cloud (optional)

Google Kubernetes Engine (GKE)

Amazon Elastic Kubernetes Service (Amazon EKS)

Managed Kubernetes Service (AKS)



An example for Amazon Elastic Kubernetes Service (Amazon EKS)

---END---