GEORGIA INSTITUTE OF TECHNOLOGY
SCHOOL OF COMPUTER SCIENCE

ECE4150 Spring 2024
Lab 4: Container Orchestration with Kubernetes

Professor: Dr. Vijay Madisetti

References:

[1] A. Bahga, V. Madisetti, “Cloud Computing Solutions Architect: A Hands-On Approach”, ISBN: 978-0996025591

[2] https://docs.docker.com/

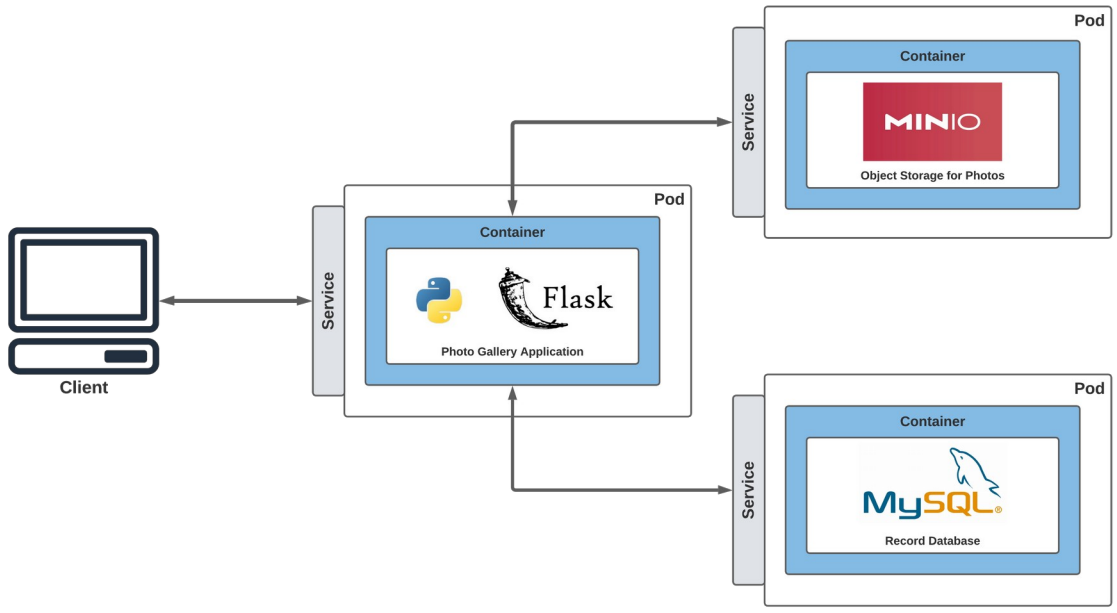
[3] https://docs.min.io/docs/minio-docker-quickstart-guide

[4] https://kubernetes.io/docs/concepts/

Due Date:

The lab report will be **due on March 20, 2024.**

The purpose of this lab is to deploy the containerized Photo Gallery application developed in Kubernetes.



Let us look at the main concepts of Kubernetes that are essential for deploying the application on a Kubernetes cluster:

- **Pod**: Pods are the smallest deployable units of computing that you can create and manage in Kubernetes. A Pod is a group of one or more containers with shared storage and network resources and a specification for how to run the containers.

- **Deployment**: A Deployment is a set of instructions provided to the master on how to create and update your application. With these instructions, the master will schedule and run your application on individual worker nodes. The master continuously monitors the deployment. If one of the instances of your applications goes down, it will be automatically replaced by a new instance.

- **Service**: Service is an abstract way to expose an application running on a set of Pods as a network service. A service defines a logical set of pods and policies to access them. This is necessary as pods can go down and be restarted (e.g., if a worker node is deleted or crashes). A service routes traffic across a set of pods and allows pods to die and replicate without impacting your application. A service can be of type: ClusterIP, NodePort, LoadBalancer, or ExternalName. By default, Kubernetes creates a ClusterIP service, which makes your Service only accessible from inside the cluster. A NodePort service exposes the Service on each Node's IP at a static port (the NodePort). NodePort is used when you want to access the Service from outside the cluster. Finally, a LoadBalancer service exposes the Service externally using a cloud provider's load balancer.

- **PersistentVolume**: A PersistentVolume (PV) is a piece of storage in the cluster that has been provisioned by an administrator or dynamically provisioned using Storage Classes. It is a resource in the cluster, just like a node is a cluster resource. A persistent volume is a storage resource with a lifecycle independent of a Pod. This means that the storage will persist if a pod goes down. In this example, the persistent volume can be a directory on your local filesystem.

- **PersistentVolumeClaim**: Pods are the smallest deployable units of computing that you can create and manage in Kubernetes. A Pod is a group of one or more containers with shared storage and network resources and a specification for how to run the containers.

- **PersistentVolumeClaim**: A PersistentVolumeClaim (PVC) is a request for storage by a user. It is similar to a Pod. However, pods consume node resources, and PVCs consume PV resources.

For deploying the photo gallery application on the Kubernetes cluster, we will create the following resources:

1. PersistentVolume and PersistentVolumeClaim for MySQL database
2. Deployment and Service for MySQL database
3. PersistentVolume and PersistentVolumeClaim for MinIO storage
4. Deployment and Service for MinIO storage
5. Deployment and Service for photo gallery Flask application

Follow the steps below to set up the Photo Gallery application.

1. Setup Minikube

- Minikube quickly sets up a local Kubernetes cluster on **Linux**, macOS, and Windows. Follow the installation instructions at: **https://minikube.sigs.k8s.io/docs/start/**
- Start minikube cluster with the following command:

```
$ minikube start --memory=4096
```

2. Setup kubectl

- You can interact with your minikube cluster using kubectl, just like any other Kubernetes cluster. To install kubectl follow the instructions here: **https://kubernetes.io/docs/tasks/ tools/**

3. Modify code to add credentials

- Navigate to the `files` directory and add the username and password created for the MinIO, and the Docker Hub username on the following files:
  - `configuration/`**`4-minio.yaml`**
  - `configuration/`**`5-app.yaml`**
    - ✴ Replace the image name with the image you generated in Lab-3 (specific to your Docker Hub account)

4. Setup MySQL Database

- Setup a PersistentVolume(PV) and PersistentVolumeClaim(PVC) for the MySQL database.

```
$ kubectl apply -f configuration/1-mysql-vol.yaml
```

- You can view the PV using `kubectl` as shown below:

```
~/De CS 2/release/files/configuration > kubectl get pv
NAME              CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                    STORAGECLASS   REASON   AGE
minio-pv-volume   1Gi        RWX            Retain           Bound    default/minio-pv-claim   manual                  36m
mysql-pv-volume   1Gi        RWO            Retain           Bound    default/mysql-pv-claim   manual                  36m
```

- Next, setup a Kubernetes Deployment and a Service for MySQL database.

```
$ kubectl apply -f configuration/2-mysql.yml
```

- You can view the pods, services and deployments using `kubectl` as shown below: **Ensure the status is in READY mode (1/1) before moving on to the next step.** (you have to wait and rerun the command to check)

```
~/Desktop/CS4083/2-kubernetes/release/files/configuration > kubectl get all
NAME                         READY   STATUS    RESTARTS   AGE
pod/mysql-7cbc469b79-x9rvz   1/1     Running   0          2s

NAME                 TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
service/db           LoadBalancer   10.110.205.197   <pending>     3306:30038/TCP   2s
service/kubernetes   ClusterIP      10.96.0.1        <none>        443/TCP          114m

NAME                    READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mysql   1/1     1            1           2s

NAME                               DESIRED   CURRENT   READY   AGE
replicaset.apps/mysql-7cbc469b79   1         1         1       2s
```

- Get the URL of the database using the command:

```
$ minikube service --all
```

```
~/Desktop/CS4083/2-kubernetes/release/files/configuration > minikube service --all
|-----------|------|-------------|---------------------------|
| NAMESPACE | NAME | TARGET PORT |            URL            |
|-----------|------|-------------|---------------------------|
| default   | db   | db/3306     | http://192.168.64.2:30038 |
|-----------|------|-------------|---------------------------|
|-----------|------------|-------------|--------------|
| NAMESPACE |    NAME    | TARGET PORT |     URL      |
|-----------|------------|-------------|--------------|
| default   | kubernetes |             | No node port |
|-----------|------------|-------------|--------------|
😿  service default/kubernetes has no node port
🎉  Opening service default/db in default browser...
```

- For this environment, the URL for the MySQL database (with the target port 3306) is http://192.168.64.2:30038 (**The URL may be different for your machine**).

- Modify the `utils/`**`photo-table.py`** and the add the **DB_HOSTNAME** (value with **192.168.64.2;** IP address of your db container) to create a new database table to store the photo records.
- Run the the to `photo-table.py` create the table

```
# Run the flask application (remember to activate venv)
$ python3 utils/photo-table.py
```

5. Setup MinIO Storage

- Setup a PersistentVolume(PV) and PersistentVolumeClaim(PVC) for the MinIO storage.

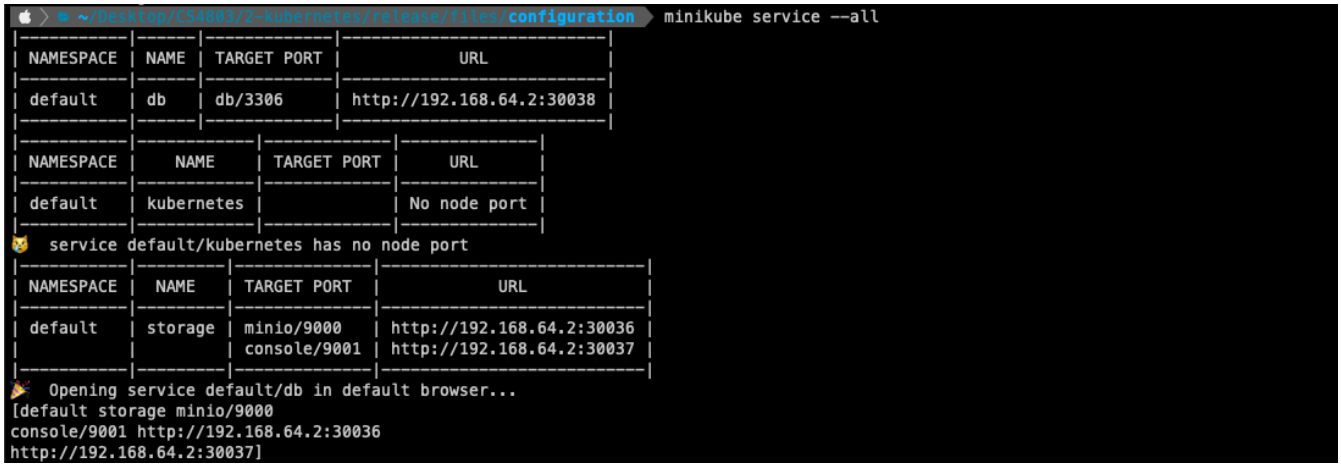
```
$ kubectl apply -f configuration/3-minio-vol.yaml
```

- You can view the PV using `kubectl` as shown below:

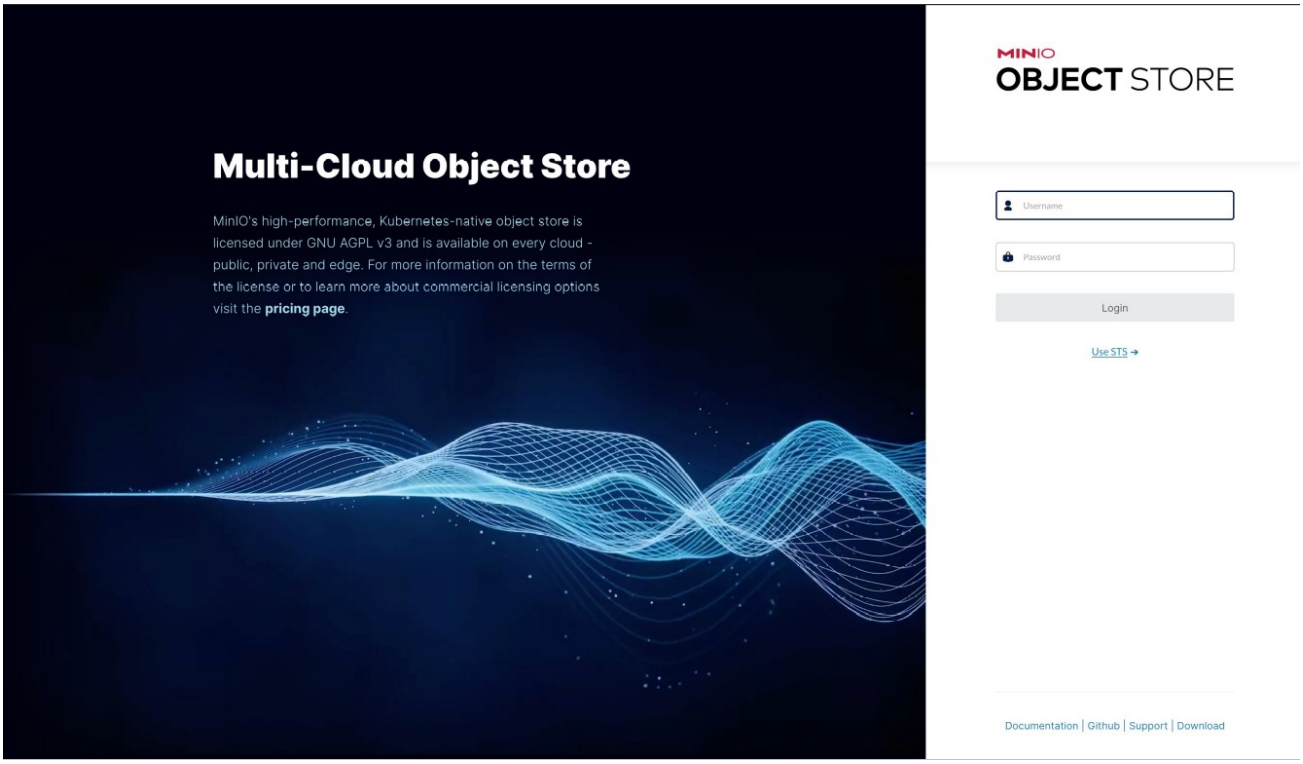
- Next, setup a Kubernetes Deployment and a Service for MinIO storage.

```
$ kubectl apply -f configuration/4-minio.yaml
```

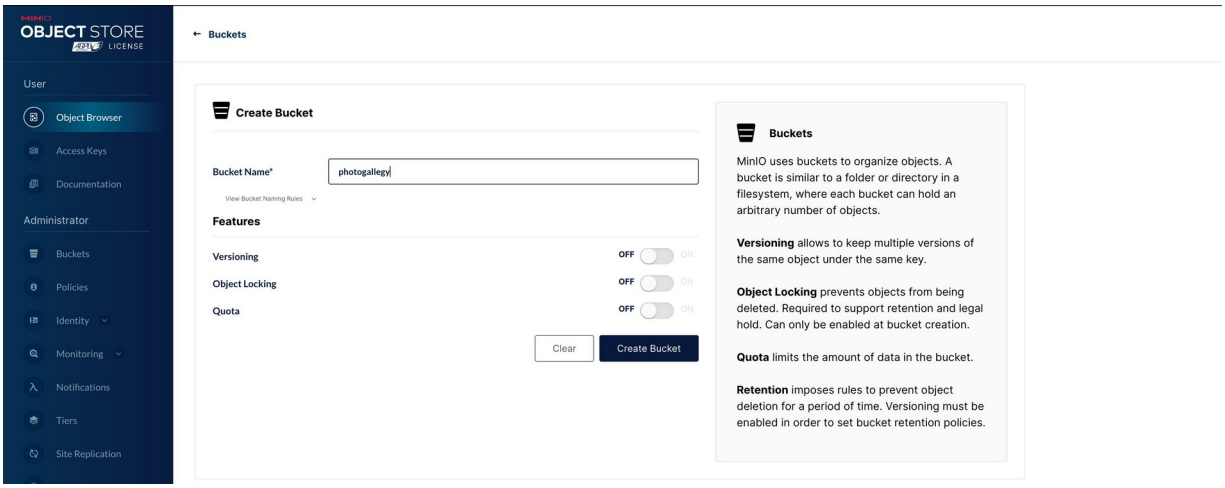
- Get the URL of the MinIO storage console using the command:



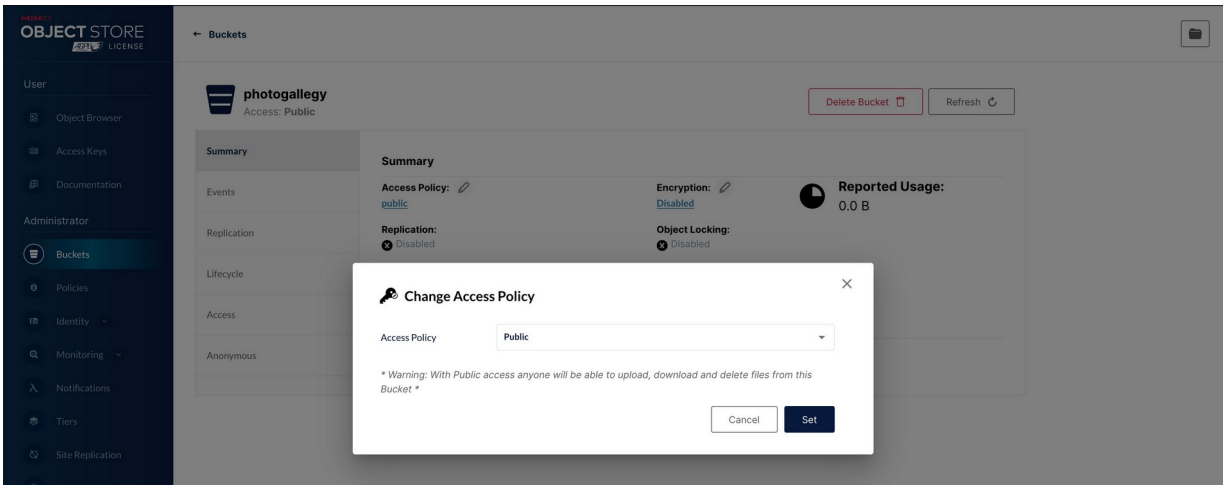
- For this environment, the URL for the MinIO console (with the target port 9001) is http:// 192.168.64.2:30037 (**The URL may be difference for your machine**).
- Browse to URL: http://192.168.64.2:30037 to access the MinIO console. (**refer the last page for troubleshooting if you can't access it**)
- Use the same username and password created for the MinIO on the following files.
  - Username: **JOHNDOEECE4150**
  - Password: **ECE4150JOHNDOE**



- From the MinIO console create a bucket named "**photogallery**".



- Go to the Buckets page and under the **Summary** section, edit the "**Access Policy**" by clicking the pencil icon. Change the access policy of the bucket to from private to **public**.



6. Setup Photo Gallery Flask application

- Next, set up a Kubernetes Deployment and a Service for the photo gallery Flask application. Apply the YAML file with `kubectl` as follows:

```
$ kubectl apply -f configuration/5-app.yaml
```

- You can view the pods, services and deployments using `kubectl` as shown below:

```
~/Desktop/CS4083/z-kubernetes/release/files/configuration > kubectl get all
NAME                                 READY   STATUS    RESTARTS   AGE
pod/mysql-7cbc469b79-2k29v           1/1     Running   0          23m
pod/photogallery-5fb65d796d-8knnh    1/1     Running   0          15s

NAME                            TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
service/db                      LoadBalancer   10.99.133.187   <pending>     3306:30038/TCP   23m
service/kubernetes              ClusterIP      10.96.0.1       <none>        443/TCP          152m
service/photogallery-service    LoadBalancer   10.103.7.20     <pending>     6000:30934/TCP   15s

NAME                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mysql         1/1     1            1           23m
deployment.apps/photogallery  1/1     1            1           15s

NAME                                     DESIRED   CURRENT   READY   AGE
replicaset.apps/mysql-7cbc469b79         1         1         1       23m
replicaset.apps/photogallery-5fb65d796d  1         1         1       15s
```

- Get the URL of the photo gallery service console using the command:

```
~/Desktop/CS4843/2-kubernetes/release/files/configuration > minikube service --all
|-----------|------|-------------|---------------------------|
| NAMESPACE | NAME | TARGET PORT |            URL            |
|-----------|------|-------------|---------------------------|
| default   | db   | db/3306     | http://192.168.64.2:30038 |
|-----------|------|-------------|---------------------------|
|-----------|------------|-------------|--------------|
| NAMESPACE |    NAME    | TARGET PORT |     URL      |
|-----------|------------|-------------|--------------|
| default   | kubernetes |             | No node port |
|-----------|------------|-------------|--------------|
😿  service default/kubernetes has no node port
|-----------|----------------------|-------------|---------------------------|
| NAMESPACE |         NAME         | TARGET PORT |            URL            |
|-----------|----------------------|-------------|---------------------------|
| default   | photogallery-service |        6000 | http://192.168.64.2:32291 |
|-----------|----------------------|-------------|---------------------------|
|-----------|---------|-------------|---------------------------|
| NAMESPACE |  NAME   | TARGET PORT |            URL            |
|-----------|---------|-------------|---------------------------|
| default   | storage | minio/9000  | http://192.168.64.2:30036 |
|           |         | console/9001 | http://192.168.64.2:30037 |
|-----------|---------|-------------|---------------------------|
🎉  Opening service default/db in default browser...
🎉  Opening service default/photogallery-service in default browser...
[default storage minio/9000
console/9001 http://192.168.64.2:30036
http://192.168.64.2:30037]
```

- For this environment, the URL for the MinIO console (with the target port 6000) is http:// 192.168.64.2:32291 (**The URL may be different for your machine, and port number might change each time you run** minikube service --all).
- Add an entry to the **/etc/hosts** file of your **local machine (host machine)** running the minikube cluster by specifying the IP address of the minikube, which is 192.168.64.2 in this case (as noted above). The /etc/hosts file will look as follows:

```
::1             localhost
127.0.0.1       let-local.nonprod.wmsports.io
192.168.64.2    minikube
# End of section
# Added by Docker Desktop
# To allow the same kube context to work on the host and the container:
127.0.0.1 kubernetes.docker.internal
# End of section
```

- This step is needed because, within the Flask application code, we generate URLs for photos uploaded to MinIO using the externally accessible hostname: minikube. Alternatively, we could have hardcoded the URL of the MinIO storage service (http:// 192.168.64.2:30036) as noted above within the Flask application code, which is not a recommended practice.
- Browse to the URL of the photo gallery service as noted above. You will see the photo gallery application. (**If you encounter a problem where you can't open the link, follow the troubleshooting step.**)

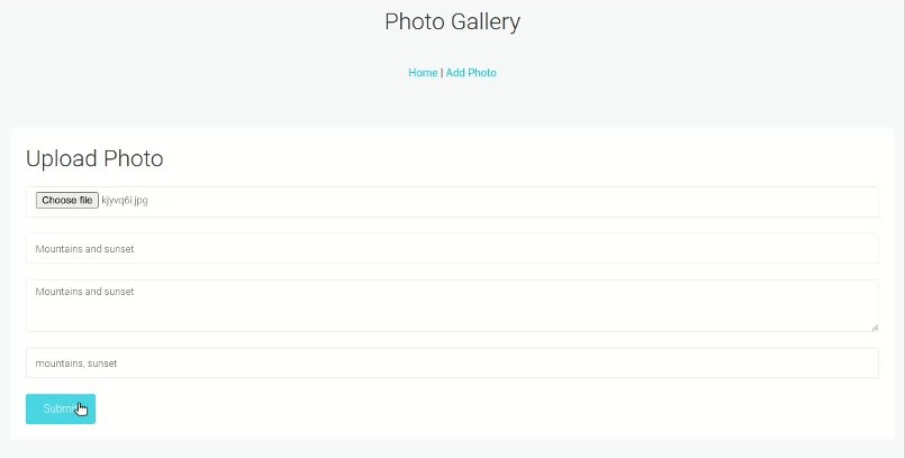
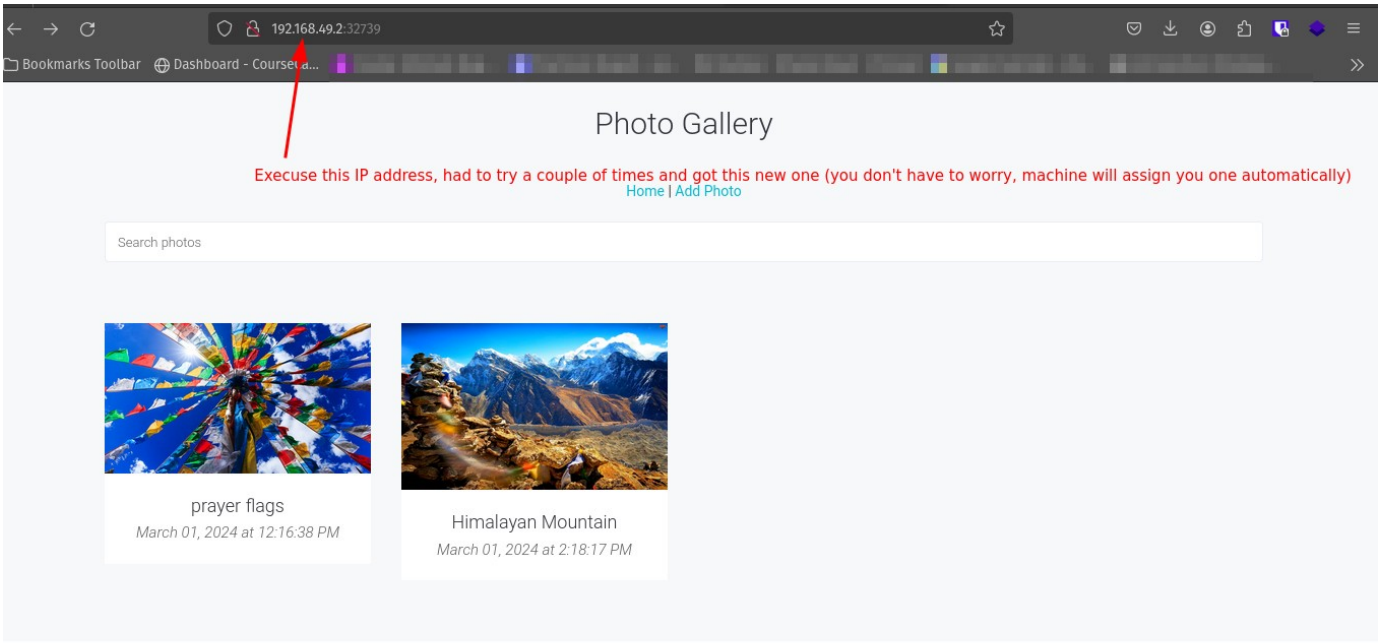


Photo Gallery
Home | Add Photo
Upload Photo
Choose file
kjyvq6l.jpg
Mountains and sunset
Mountains and sunset
mountains, sunset
Submit


192.168.49.2:32739
Bookmarks Toolbar
Dashboard - Coursera...
Photo Gallery
Execuse this IP address, had to try a couple of times and got this new one (you don't have to worry, machine will assign you one automatically)
Home | Add Photo
Search photos
prayer flags
March 01, 2024 at 12:16:38 PM
Himalayan Mountain
March 01, 2024 at 2:18:17 PM

7. Clean up

- Cleanup all Kubernetes resources and delete the minikube cluster:

```
$ kubectl delete -f 5-app.yaml
$ kubectl delete -f 4-minio.yaml
$ kubectl delete -f 3-minio-vol.yaml
$ kubectl delete -f 2-mysql.yml
$ kubectl delete -f 1-mysql-vol.yaml
$ minikube delete
```

Congratulation! You successfully completed this lab.

Deliverables:

A proof video that shows the deployment/shipment of the containers to Kubernetes and the functionalities of your website: upload a picture, view the photo and search for it using the search bar.

Please ensure that the video does not exceed **12 minutes** in duration, as every minute beyond this limit will result in a **deduction of 5 points**.

Sample Submission: https://youtu.be/-eiEoputGUE

Troubleshooting Section

When you can't access/browser the url from your host machine:

1. Mac / Linux Machine
- https://www.youtube.com/watch?v=adMdtWcFQhE

2. Window Machine
- https://www.youtube.com/watch?v=5z3uXrFxN1k