COMP7305 Cluster and Cloud Computing 2023-24

Submission Deadline: July 19, 2024 23:59

Workshop Exercise 3: Deploying MySQL + PHP to AKS

In this exercise, we try deploying the MySQL database with PHP and Web applications on the Azure Kubernetes Service (AKS) environment. We shall use the **Azure Blob** as a persistent storage to store the **database data**, as well as the **Web applications' PHP scripts and HTML files** so that the MySQL data can be stored permanently, and we can **run any PHP-based Web applications on this platform**. Like Exercise 2, we also include PHPMyAdmin to connect to the MySQL database for administration.

Stage 1: Creating the Storage Account and Blob Containers

As we shall use **static provisioning** with Azure Blob to create and mount the persistent storage for use by the pods in the AKS cluster, the first thing we are going to do is to **create the Azure Blob Storage Account** and the blob container**s** in Azure Portal. Please follow steps 1-8 in the Section "Static Provisioning" of 08-Workshop-AKS.pdf to **create one Azure Blob Storage Account** and **two** blob containers inside the storage account. One blob container will be mounted as the MySQL database data directory (which will be created by the MySQL pod in AKS during its initialization), and the other is to store the PHP scripts and other HTML-related files for our Web applications. Note: Make sure that NFS V3 is enabled when creating the container (Step 4), and that the firewall has been set properly (Step 7). Otherwise, the AKS cluster will not be able to mount the blob containers for use.

After creating the two blob containers, make sure the first container is empty, because MySQL is very strict about the database directory when it starts, if there are some invalid and unrelated files inside the database directory, MySQL will report an error and refuse to start. For the second container, you upload the following files to the second storage container using Azure Portal:

- index.php
- mtr.html
- lines-v1.6.json

You can download these files from the course's Moodle site under the Assignment Three block. (Note: index.php is the PHP program we used in Exercise 2 with **minor changes**.)

Stage 2: Provisioning the Blob Containers

The next stage is to provision the blob containers as Persistent Volumes (PV) and Persistent Volume Claims (PVC) so that they can be used by the pods in the AKS. Follow Steps 9-12 in the Section "Static Provisioning" of 08-Workshop-AKS.pdf to create the .yaml files for configuring the PV and PVC. Since we have two blob containers, the .yaml file configuring the PV should have two sections, each creating one PV resource. Similarly, the .yaml file configuring the PVC should also have two sections creating two PVCs. Use a line with three consecutive hyphens (---) to separate the two sections.

Stage 3: Creating the ConfigMap and Secret .yaml Files

Like in Minikube, we shall create a ConfigMap file and a Secret file to store the database name, username, and passwords, as well as the service name, so that PHPMyAdmin and other PHP scripts can use them to connect to the MySQL database successfully. This part is **exactly** the same as that in Minikube and you may simply **reuse** the ConfigMap and Secret .yaml files.

Stage 4: Creating the .yaml File for the MySQL+PHPMyAdmin Deployment and Services

Next, we shall prepare the .yaml file for configuring the MySQL+PHPMyAdmin deployment and services. This part should be similar to the .yaml file we created for Exercise 2. The only difference is that we need to **use the PVC** created in Stage 2 and mount the first blob container to the MySQL container. Please refer to the code in Steps 9-11 in the Section "Dynamic Provisioning" of 08-Workshop-AKS.pdf about what to add to use the PVC and specify the mount point of the PVC. For your information, the mount point for the MySQL data directory should be **/var/lib/mysql**.

The PHPMyAdmin configuration is basically the same setting as for Exercise 2 except that you need to change the Service section, where LoadBalancer should be used instead of NodePort as the type of service (as this helps create the external IP automatically). Also, remove the port number for NodePort as it is no longer used in the AKS setting.

Stage 5: Creating the .yaml File for the Apache + PHP Deployment and Service

The last stage is to create the .yaml file for configuring the Apache + PHP deployment and service. Once again, this part is very similar to the .yaml file we created for Exercise 2. Two main differences are:

- We need to use the PVC created in Stage 2 and mount the second blob container (that is, the one you have uploaded the PHP and HTML files) to the Apache pod. Please refer to the code in Steps 9-11 in the Section "Dynamic Provisioning" of 08-Workshop-AKS.pdf about what to add to use the PVC and specify the mount point of the PVC. For your information, the mount point for the default homepage directory should be **/var/www/html**. (Note: We are reusing the same image you prepared for Exercise 2. Although that image already contains the index.php script, when the system successfully mounts the blob container to /var/www/html, the version in the blob container will be used instead as the mounted directory will cover the original /var/www/html in the image. You will find that the heading of the PHP output shows "COMP7305 Exercise 3 (AKS) " instead of just " COMP7305 Exercise 2".)
- In the Service section, LoadBalancer should be used instead of NodePort as the type of service. Also, remove the port number for NodePort as it is no longer used in AKS.

To test whether the deployments work properly, after creating the .yaml files and applying them in order, you can perform the following:

1. In the Azure Shell, type "kubectl get services" to see the list of the services (you should see 4 services, 3 of them are created in this exercise, namely one for MySQL, one for PHPMyAdmin, and one for Apache + PHP), the other one is the default Kubernetes service that should always exist. One sample output is like this:

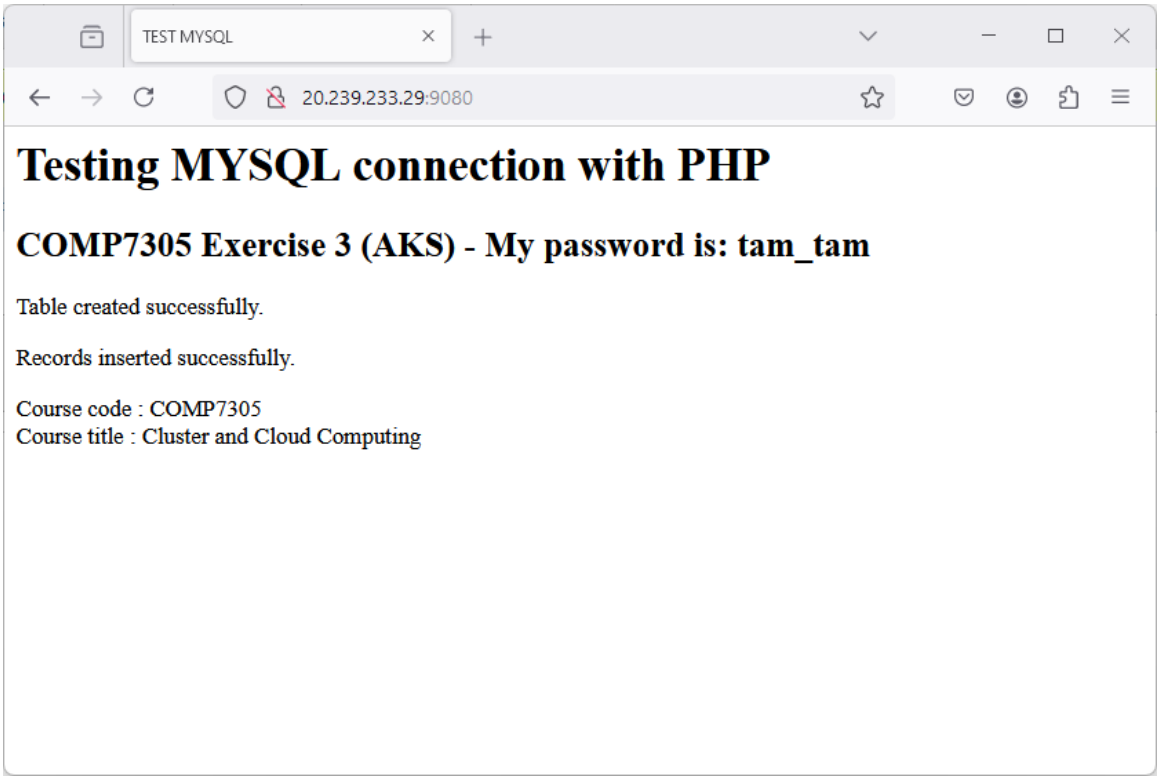
```
NAME                TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)          AGE
apache-php-service  LoadBalancer   10.0.148.238   20.239.233.29   9080:32455/TCP   10m
kubernetes          ClusterIP      10.0.0.1       <none>          443/TCP          22h
mysql-service       ClusterIP      10.0.100.15    <none>          3306/TCP         16m
phpadmin-service    LoadBalancer   10.0.138.247   4.144.33.168    9000:31432/TCP   16m
```

   To test whether PHPMyAdmin works properly, open a web browser, and type this URL: http://<External_IP:Port> (in the above example, the URL will be http://4.144.33.168:9000). Note that sometimes it may take a while for the PHPMyAdmin login webpage to appear.

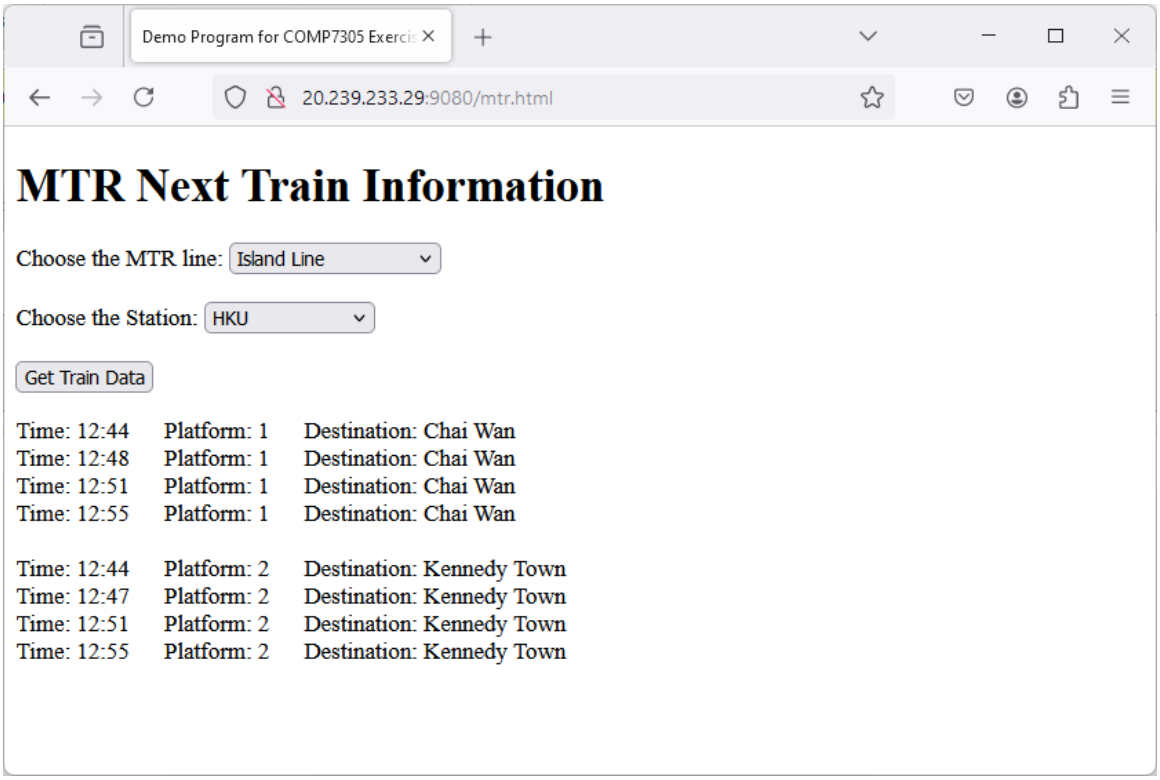
2. Enter the MySQL account username (sql7305) and the corresponding password (the one that matches your group). If the .yaml files are written correctly, you should be able to log in.
3. To test whether Apache + PHP works properly, you can do the same thing as Step 2 (e.g., http:// 20.239.233.29:9080) to access the index.php program. However, you will find a "Permission Denied" error when you try this the first time. The reason is that when you upload the index.php script, the permission is set as 640 (i.e. readable and writable by the owner, readable by users in the same group as the owner, but none for other users), while webpages and PHP scripts require the permission 644 (i.e. also readable by other users).
4. Hence, we need to log in to the Apache + PHP pod to change the file permission. In the Azure Shell, type "kubectl exec -it <pod_name> -- /bin/bash" to log in to the Apache pod. Replace <pod_name> with the pod name of your Apache + PHP pod (you may find the name by using "kubectl get pods"). If you successfully log in to the pod, you will see the prompt changes to something like "root@<pod_name>:/var/www/html#".
5. Type the following commands:
   - chmod 644 *
   - exit

```
D:\> kubectl exec pod/apache-php-5949c854f-kw9rh -- ls -l
total 10
-rw-r----- 1 root root 1491 Jun 21 07:43 index.php
-rw-r----- 1 root root 3321 Jun 21 07:43 lines-v1.6.json
-rw-r----- 1 root root 5001 Jun 21 07:43 mtr.html
D:\> kubectl exec pod/apache-php-5949c854f-kw9rh -it -- /bin/bash
root@apache-php-5949c854f-kw9rh:/var/www/html# chmod 0644 *
root@apache-php-5949c854f-kw9rh:/var/www/html# ls -l
total 10
-rw-r--r-- 1 root root 1491 Jun 21 07:43 index.php
-rw-r--r-- 1 root root 3321 Jun 21 07:43 lines-v1.6.json
-rw-r--r-- 1 root root 5001 Jun 21 07:43 mtr.html
root@apache-php-5949c854f-kw9rh:/var/www/html# exit
exit
D:\>
```

6. Reload the web browser tab, this time the PHP script should be able to run:



7. Open another tab to access the mtr.html program (http://<external_ip:port>/mtr.html). You should be able to access this webpage and run JavaScript to get the real-time MTR next train information.



8. Use the "kubectl get pod" command to find the ID of the MySQL pod and then use the "kubectl delete pod <mysql_pod>" command to kill the MySQL pod.
9. Run the "kubectl get pod" command to show that a new MySQL pod is created.
10. Access the index.php page again to show that the PHP program can access the MySQL pod and add a new entry to the MySQL server. The page should show more than one set of the Course code and Course title information.
11. Finally, remember to delete all resources and stop the AKS after you fully test the system to save cost.

Submission:

- Deadline: July 19, 2024, 23:59
- Assume AKS, Storage Account, and the two containers are ready. Use Zoom to record a live demonstration of:
    - using Azure Portal to upload the three files – index.php, mtr.html, & lines-v1.6.json to the storage container for the Apache pod;
    - using Azure Portal to access the storage container for the MySQL pod and show that the container is empty;
    - using kubectl to create the configMap object and the secret object;
    - using kubectl to create the PV objects and the PVC objects;
    - running the "kubectl get pv" and "kubectl get pvc" commands to show the status of the PV and PVC objects;
    - using kubectl to create the MySQL+PHPMyAdmin deployment and services;
    - running the "kubectl get all" command a few times to show the status of the deployment and services until the MySQL pod is ready;
    - using kubectl to create the Apache+PHP deployment and service;
    - running the "kubectl get all" command a few times to show the status of the deployments and services until the Apache pod is ready;
    - identify the external IP address of the PHYMyAdmin service and use a web browser to access the MyAdmin page;
    - showing that you can successfully log in to the MySQL server with the sql7305 or root account;
    - apply the permission setting as described in steps 4-5 above;

- identify the external IP address of the Apache service and use a web browser to access the index.php program; show the rendering output of the browser;
- access the mtr.html program and select an MTR line and a station for showing the next train information;
- using kubectl to terminate the MySQL pod and show that a new MySQL pod has created;
- finally, use a web browser to access the index.php page again and show that an additional set of Course code + title has added to the database.

Make sure you store the recording on the cloud and submit the recording link for the tutors to check your work.

- Submit all the .yaml files for this exercise. You can compress them to one single zip file for submission.

Grading Policy

Points	Criteria
1.0	• Successfully create a Storage account with two containers and have the three files (index.php, mtr.html, & lines-v1.6.json) uploaded to one container
2.0	• Successfully create the two PVs and PVCs as reported by "kubectl get pv" and "kubectl get pvc"
1.5	• Successfully deployed the MySQL and PHPMyAdmin pods and the two services as reported by "kubectl get all"
1.5	• Successfully deployed the Apache pod and its service as reported by "kubectl get all"
1.0	• Successfully connected to the PHPMyAdmin web interface using the specific password of the sql7305 account to login
1.5	• Successfully connected a web browser to show the index.php web page (with correct content) and the mtr.html page with MTR information
1.5	• The MySQL database can access persistent storage and continue to provide its services even after a restart
-4.0	• No submission of the video recording
-10.0	• No submission of the YAML files