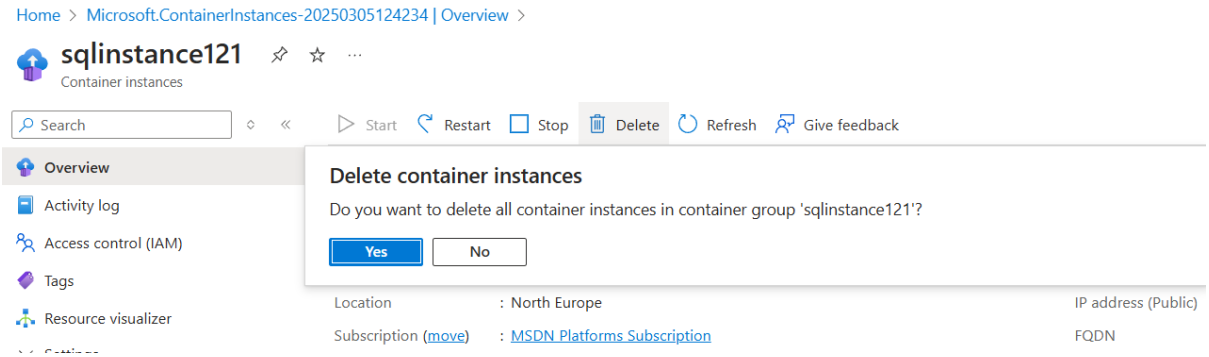
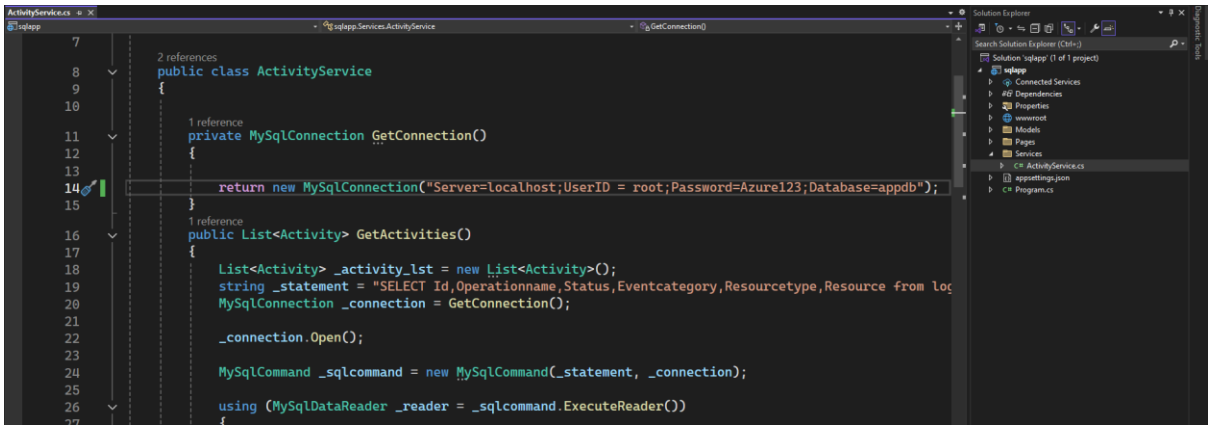
😊 Build our Application Image

1. We are going to continue with the previous lab, so far what we've done is, we have built a custom MySQL image. We have it in on Azure Container Registry, and we know it works because we deployed it on an Azure Container Instance. It's now time to go ahead and take our application and containerize it. I now want to create an image of our ASP.NET application, so that it can be deployed as a container.
2. First, we will go to our Azure container instance and delete it because we don't require it anymore.



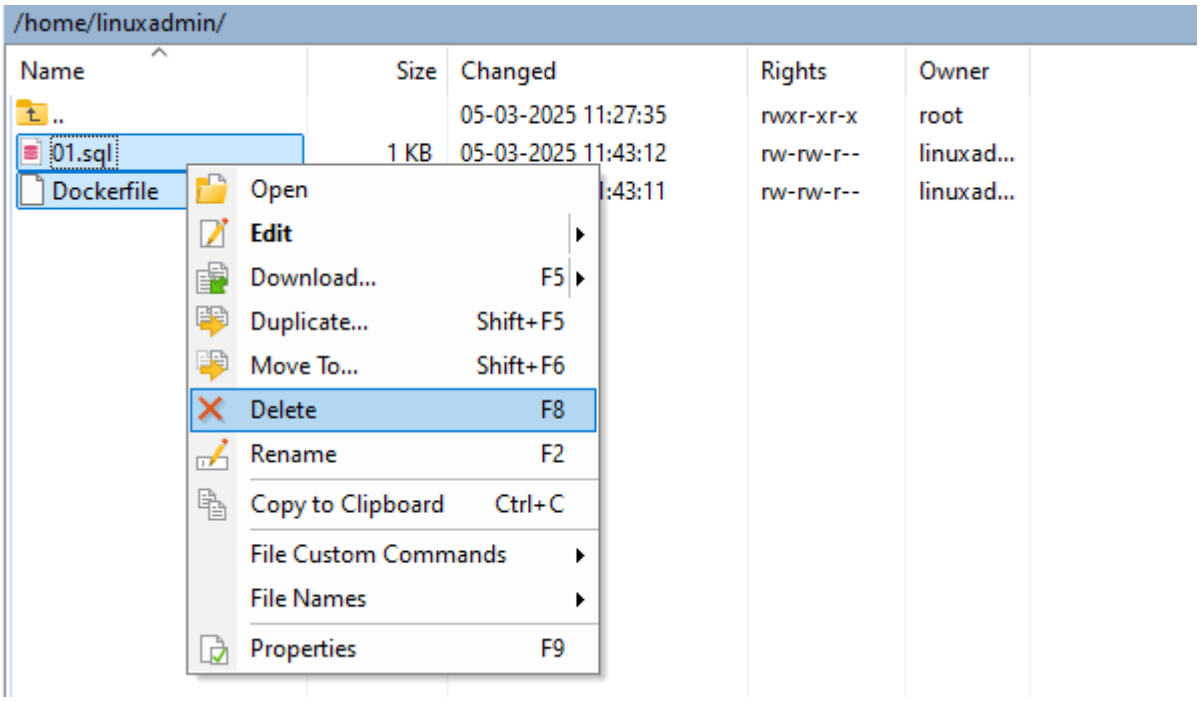
3. We'll go on to the program, which is used for displaying the data within the table in MySQL. For the server, I'm going to put it as localhost. I'm first going to be deploying our application and our database again onto the Azure Container Instance. And this time when it comes onto the connection for the application, we can mention it as localhost. The user ID and the password remain the same. We now need to containerize this application on our Linux machine. So this will be done by the use of a docker file.



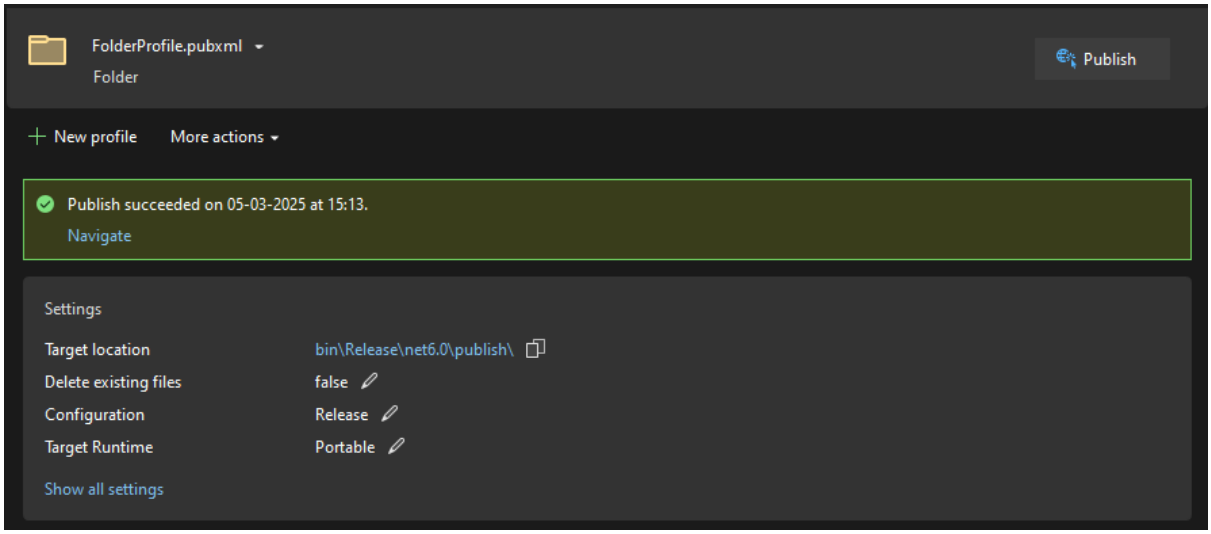
4. I have the Docker file in here. It's a very simple file. I'm using the base image of ASP.NET that can be used for running my application within a container. I'm also exposing port 80 for the application on the container itself.

```
FROM mcr.microsoft.com/dotnet/aspnet:6.0
WORKDIR /app
COPY  . .
EXPOSE 80
ENTRYPOINT ["dotnet", "sqlapp.dll"]
```

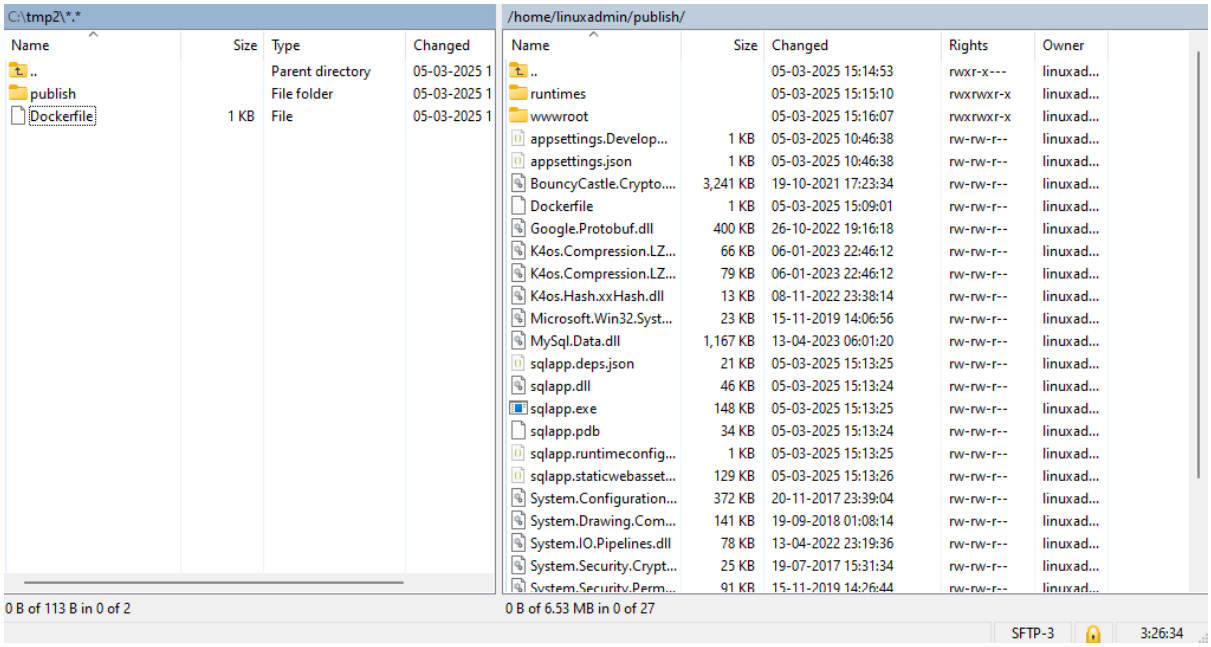
5. So, we need to copy this Docker file and the publish profile onto our Linux VM. We need to create the publish profile of our program.
6. But first we need to delete the files that we currently have inside our VM.



7. Below you can see that we have created the publish profile onto the folder in our laptop.



8. Now we need to open the Win SCP tool and then we need to copy the publish folder first then inside the publish folder we have to paste the docker file.



9. Now we are going to use the command to build our image and push it onto the container registry. You can see that the commands are the same as before just the name has been changed. The name of the container registry remains the same. You get the password from the access keys.

commands 2.txt

```
sudo docker build -t sqlapp .


Push this to Azure Container Registry

sudo docker login appregistry1212.azurecr.io -u appregistry1212 -p plvAXQngcXXH6OSmTtKNXmVexr+7EY/WsJNaox4QkK+ACRBDCzZ3

sudo docker tag sqlapp appregistry1212.azurecr.io/sqlapp

sudo docker push appregistry1212.azurecr.io/sqlapp
```

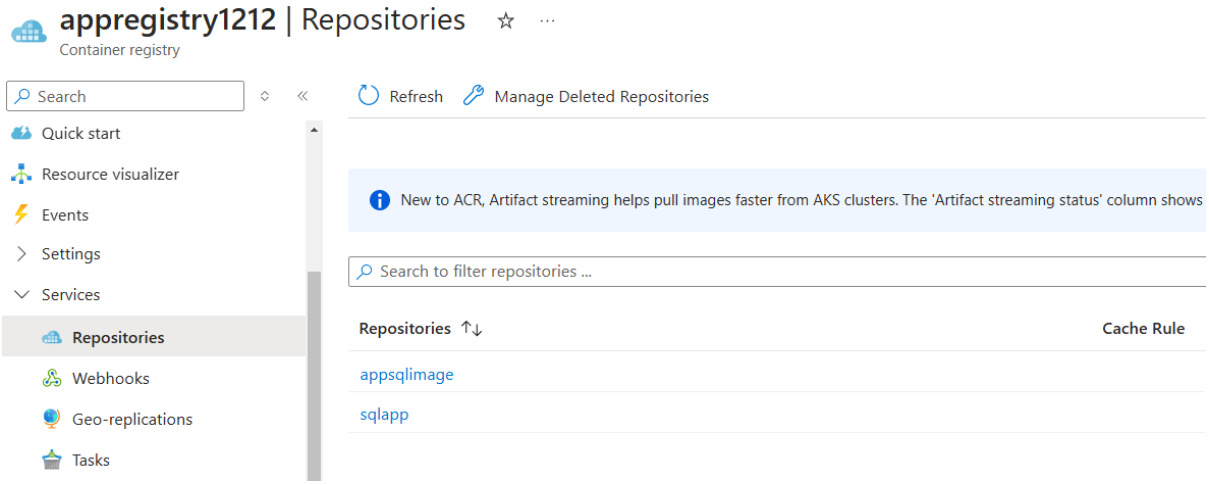
10. Now open up the terminal connect your VM and run all these commands. First, go inside the publish folder and run the build command.

```
linuxadmin@VM1:~$ ls
publish
linuxadmin@VM1:~$ cd publish/
linuxadmin@VM1:~/publish$ sudo docker build -t sqlapp .
[+] Building 7.2s (4/7)
```

11. Then run the rest of the commands to push the image onto the container registry.

```
linuxadmin@VM1:~/publish$ sudo docker login appregistry1212.azurecr.io -u appregistry1212 -p plvAXQngcXXH6OSmTtKNXmVexr+7EY/WsJNaox4QkK+A
CRBDCzZ3
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
Login Succeeded
linuxadmin@VM1:~/publish$ sudo docker tag sqlapp appregistry1212.azurecr.io/sqlapp
linuxadmin@VM1:~/publish$ sudo docker push appregistry1212.azurecr.io/sqlapp
Using default tag: latest
The push refers to repository [appregistry1212.azurecr.io/sqlapp]
9babbcb43760: Pushed
70f3da3aba93: Pushed
7f7140d2a118: Pushed
59361b3806d7: Pushed
a190071c1c72: Pushing [==================================================>]  70.82MB
14c1dc5943c0: Pushing [====================>                              ]  15.25MB/36.22MB
9b7261696dcc: Pushing [===>                                               ]  5.381MB/80.65MB
```

12. In the container registry you can see your images in the repository.



Now we're going to set up and run two containers—one for a MySQL database and one for your web application—using **Azure Container Instances (ACI)**. Your web application's container image is stored in **Azure Container Registry (ACR)**.

To deploy both containers together, you're using a **YAML configuration file**. This file defines the container group, specifying:

- **Two containers**:
    - A database container (**named "DB"**) running MySQL
    - A web application container (**named "web"**)
- **Resource allocation**: Each container is assigned a specific amount of CPU and memory.
- **Platform**: The containers run on **Linux**.
- **Exposed ports**: For the demo, both **port 80 (for the web app)** and **port 3306 (for MySQL)** are open to the public internet. However, in a real-world setup, the **database should not be exposed publicly**.
- **Container registry details**: The YAML file includes authentication and access details to pull the web application's image from **Azure Container Registry (ACR)**.

This setup ensures that both your database and application run together inside **a single container group** in Azure.

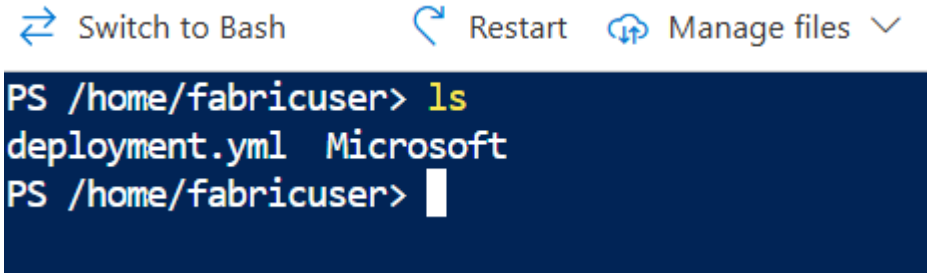
13. So, below is the deployment file which you can find on GitHub, and in this file you need to change the name of the container registry and the password as well.

Deployment.yml

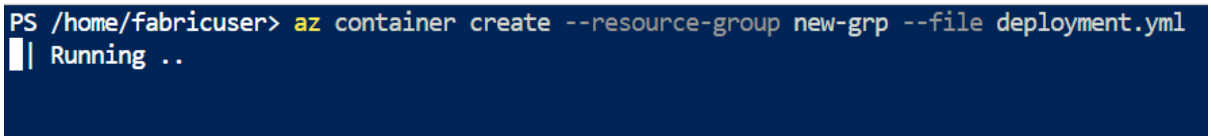
```yaml
apiVersion: 2019-12-01
location: northeurope
name: SQLAppGroup
properties:
  containers:
    - name: db
      properties:
        image: appregistry1212.azurecr.io/appsqlimage:latest
        resources:
          requests:
            cpu: 1
            memoryInGb: 1.5
        ports:
          - port: 3306
    - name: web
      properties:
        image: appregistry1212.azurecr.io/sqlapp:latest
        resources:
          requests:
            cpu: 1
            memoryInGb: 1.5
        ports:
          - port: 80
  osType: Linux
  ipAddress:
    type: Public
    ports:
      - protocol: tcp
        port: 80
      - protocol: tcp
        port: 3306
  imageRegistryCredentials:
    - server: appregistry1212.azurecr.io
      username: appregistry1212
      password: plvAXQngcXXH6OSmTtKNXmVexr+7EY/WsJNaox4QkK+ACRBDCzZ3
type: Microsoft.ContainerInstance/containerGroups
```

14. After that open Azure Cloud Shell and in the cloud shell upload the deployment file then run the below command to execute this file.

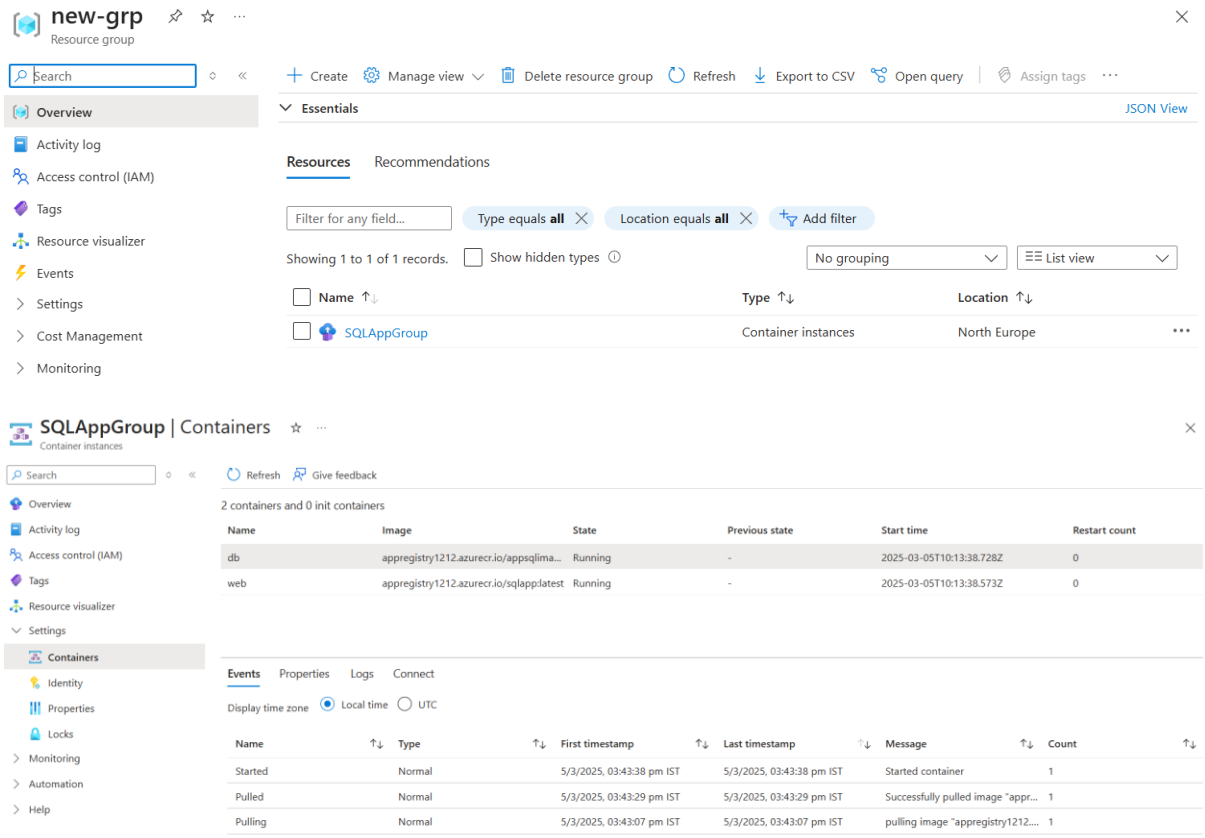
az container create --resource-group new-grp --file deployment.yml



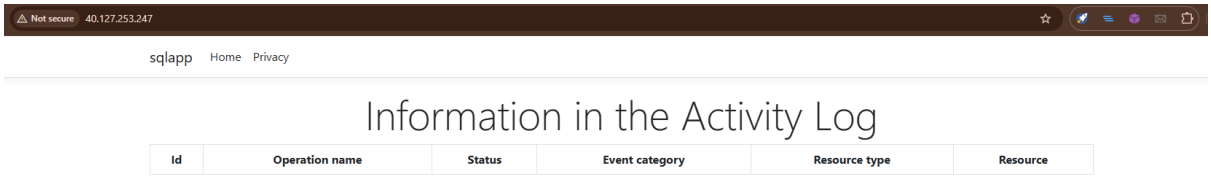
15. Below you can see that our file is running let's wait until it gets completed.



16. Here you can see that our container instance has been created. Open this and you can see that both of our containers are running.



17. As we know we have enabled port 3306 and port 80 we can use the public IP address of our container instance and paste it into a new tab we can see the table in place but it does not contain any data.



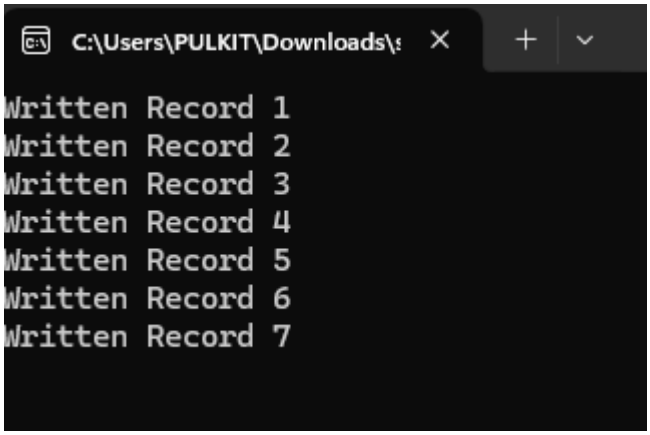
18. To upload data to our table we can again use the application, we just need to change the IP address and run the program.

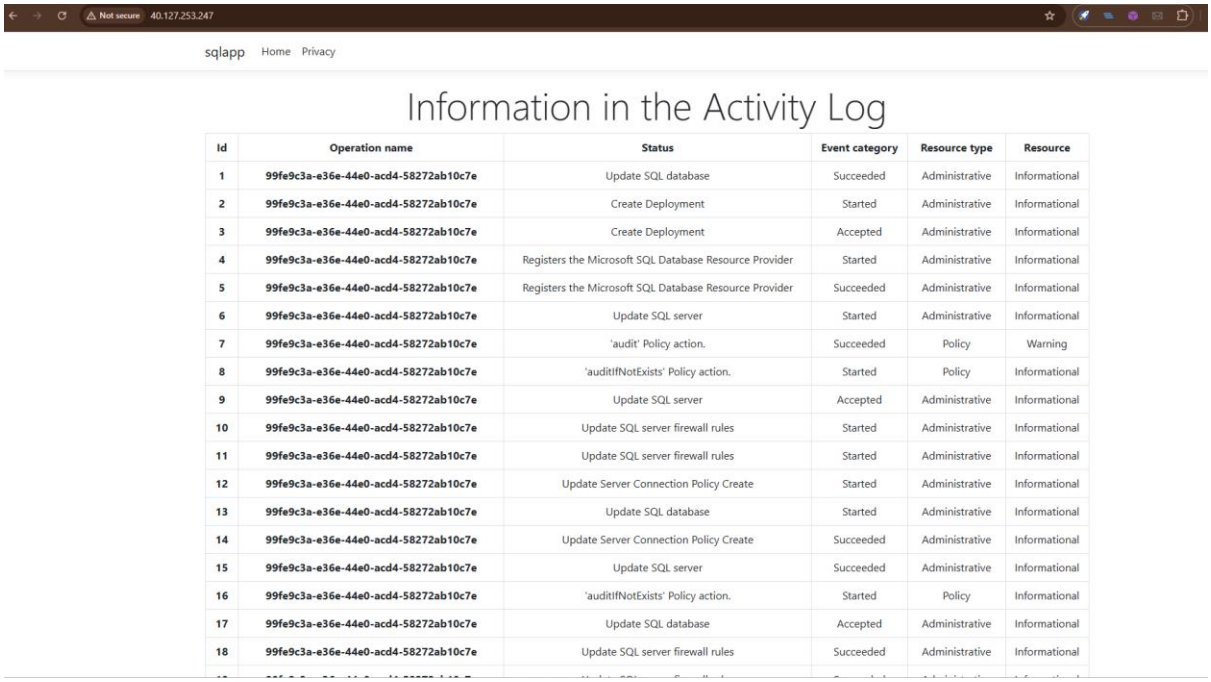
```
using MySql.Data.MySqlClient;
using System.Data;
using System.Data.SqlClient;
using System.Reflection.Metadata;

string strFilePath = "C:\\tmp1\\Log.csv";
StreamReader? logReader =null;
int Id = 1;
MySqlConnection appdbConnection =
    new MySqlConnection("Server=40.127.253.247;UserID = root;Password=Azure123;Database=appdb");

MySqlParameter paramId = new MySqlParameter();
paramId.ParameterName = "@Id";
```

19. Below you can see that the records are being written and if we refresh the page we can see that data.





20. This confirms that our web application is now running as a container in the Azure Container Instance service.
21. Once you are done delete your container instance.