

😊 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.

The screenshot shows the Azure portal interface for managing container instances. The top navigation bar includes 'Home', 'Microsoft.ContainerInstances-20250305124234 | Overview', and other account options. Below the navigation is a search bar and a toolbar with 'Start', 'Stop', 'Delete', 'Refresh', and 'Give feedback' buttons. The main area displays a container instance named 'sqlinstance121' under the 'Container instances' section. On the left, there's a sidebar with links for 'Overview', 'Activity log', 'Access control (IAM)', 'Tags', and 'Resource visualizer'. A central modal dialog box titled 'Delete container instances' asks 'Do you want to delete all container instances in container group 'sqlinstance121'?'. It contains two buttons: 'Yes' (highlighted in blue) and 'No'. At the bottom of the page, there are details about the instance: 'Location: North Europe', 'Subscription (move): MSDN Platforms Subscription', 'IP address (Public)', and 'FQDN'.

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.

The screenshot shows the Visual Studio IDE. The code editor displays the 'ActivityService.cs' file, which contains C# code for an ASP.NET application. The code includes methods for connecting to a MySQL database and retrieving activity data. The Solution Explorer on the right shows the project structure for 'sqlapp', including files like 'ActivityService.cs', 'appsettings.json', and 'Program.cs'.

```
ActivityService.cs
7
8     2 references
9     public class ActivityService
10    {
11        1 reference
12        private MySqlConnection _connection
13        {
14            1 reference
15            return new MySqlConnection("Server=localhost;UserId = root;Password=Azure123;Database=appdb");
16        }
17        1 reference
18        public List<Activity> GetActivities()
19        {
20            List<Activity> _activity_lst = new List<Activity>();
21            string _statement = "SELECT Id,Operationname,Status,Eventcategory,Resourcetype,Resource from log";
22            MySqlConnection _connection = GetConnection();
23
24            _connection.Open();
25
26            MySqlCommand _sqlcommand = new MySqlCommand(_statement, _connection);
27            using (MySqlDataReader _reader = _sqlcommand.ExecuteReader())
28            {
29                while (_reader.Read())
30                {
31                    Activity _activity = new Activity();
32                    _activity.Id = _reader.GetInt32("Id");
33                    _activity.Operationname = _reader.GetString("Operationname");
34                    _activity.Status = _reader.GetString("Status");
35                    _activity.Eventcategory = _reader.GetString("Eventcategory");
36                    _activity.Resourcetype = _reader.GetString("Resourcetype");
37                    _activity.Resource = _reader.GetString("Resource");
38
39                    _activity_lst.Add(_activity);
40                }
41            }
42            return _activity_lst;
43        }
44    }
```

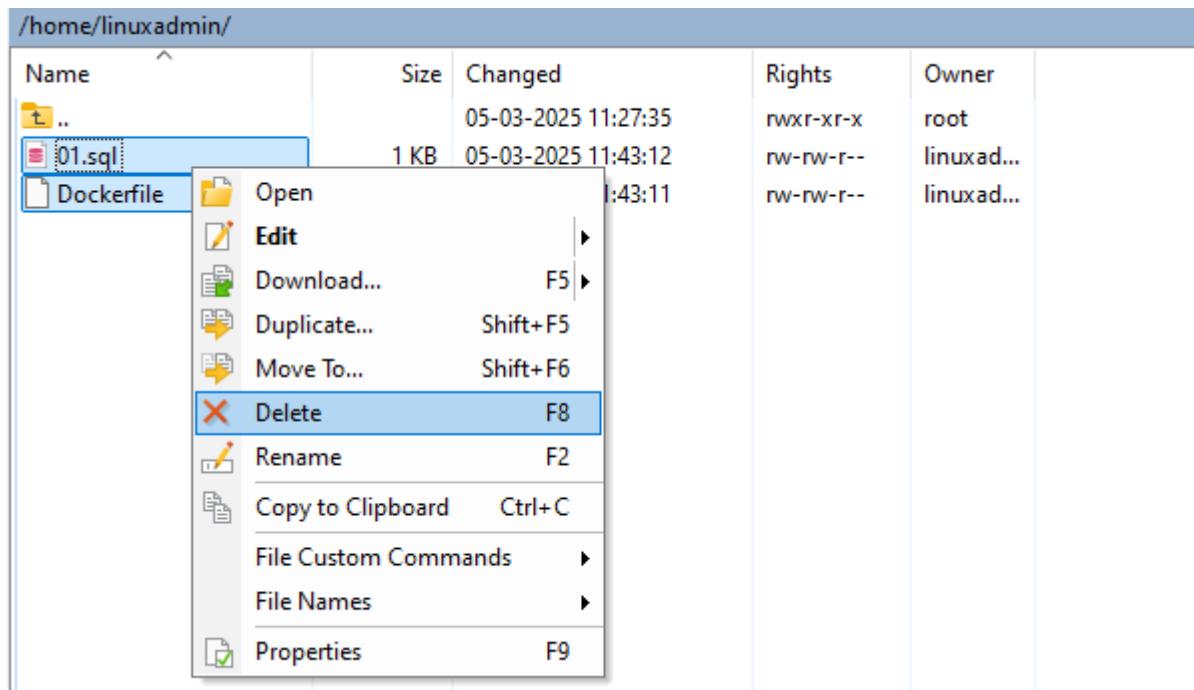
4. I have the Docker file in here. It's a very simple file. I'm using the base image of ASP.NET that can be used for running my application within a container. I'm also exposing port 80 for the application on the container itself.

```

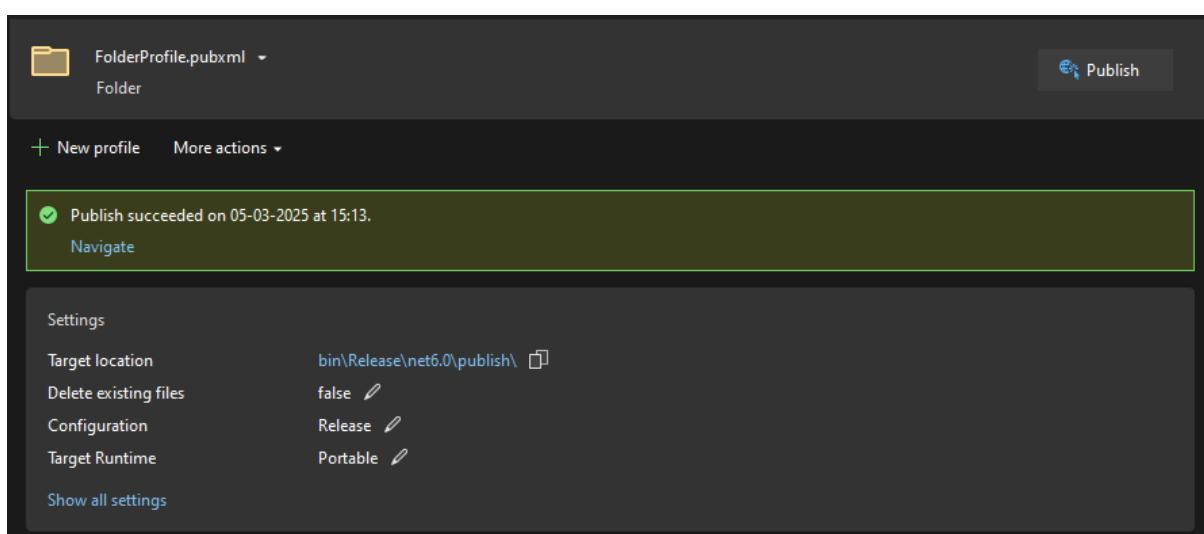
FROM mcr.microsoft.com/dotnet/aspnet:6.0
WORKDIR /app
COPY . .
EXPOSE 80
ENTRYPOINT ["dotnet", "sqlapp.dll"]

```

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.

C:\tmp2*.*				/home/linuxadmin/publish/					
Name	Size	Type	Changed	Name	Size	Changed	Rights	Owner	
t..		Parent directory	05-03-2025 1	t..		05-03-2025 15:14:53	rwxr-x---	linuxad...	
publish		File folder	05-03-2025 1	runtimes		05-03-2025 15:15:10	rwxrwxr-x	linuxad...	
Dockerfile	1 KB	File	05-03-2025 1	wwwroot		05-03-2025 15:16:07	rwxrwxr-x	linuxad...	
				appsettings.Develop...	1 KB	05-03-2025 10:46:38	rw-rw---	linuxad...	
				appsettings.json	1 KB	05-03-2025 10:46:38	rw-rw---	linuxad...	
				BouncyCastle.Crypto...	3,241 KB	19-10-2021 17:23:34	rw-rw---	linuxad...	
				Dockerfile	1 KB	05-03-2025 15:09:01	rw-rw---	linuxad...	
				Google.Protobuf.dll	400 KB	26-10-2022 19:16:18	rw-rw---	linuxad...	
				K4os.Compression.LZ...	66 KB	06-01-2023 22:46:12	rw-rw---	linuxad...	
				K4os.Compression.LZ...	79 KB	06-01-2023 22:46:12	rw-rw---	linuxad...	
				K4os.Hash.xxHash.dll	13 KB	08-11-2022 23:38:14	rw-rw---	linuxad...	
				Microsoft.Win32.Syst...	23 KB	15-11-2019 14:06:56	rw-rw---	linuxad...	
				MySQL.Data.dll	1,167 KB	13-04-2023 06:01:20	rw-rw---	linuxad...	
				sqlapp.deps.json	21 KB	05-03-2025 15:13:25	rw-rw---	linuxad...	
				sqlapp.dll	46 KB	05-03-2025 15:13:24	rw-rw---	linuxad...	
				sqlapp.exe	148 KB	05-03-2025 15:13:25	rw-rw---	linuxad...	
				sqlapp.pdb	34 KB	05-03-2025 15:13:24	rw-rw---	linuxad...	
				sqlapp.runtimeconfig...	1 KB	05-03-2025 15:13:25	rw-rw---	linuxad...	
				sqlapp.staticwebasset...	129 KB	05-03-2025 15:13:26	rw-rw---	linuxad...	
				System.Configuration...	372 KB	20-11-2017 23:39:04	rw-rw---	linuxad...	
				System.Drawing.Com...	141 KB	19-09-2018 01:08:14	rw-rw---	linuxad...	
				System.IO.Pipelines.dll	78 KB	13-04-2022 23:19:36	rw-rw---	linuxad...	
				System.Security.Crypt...	25 KB	19-07-2017 15:31:34	rw-rw---	linuxad...	
				System.Security.Perms...	91 KB	15-11-2019 14:26:44	rw-rw---	linuxad...	

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
1 sudo docker build -t sqlapp .
2
3
4 Push this to Azure Container Registry
5
6 sudo docker login appregistry1212.azurecr.io -u appregistry1212 -p plvAXQngcXXH60SmTtKNXmVexr+7EY/WsJNaox4QkK+ACRBDCzZ3
7
8 sudo docker tag sqlapp appregistry1212.azurecr.io/sqlapp
9
10 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 plvAXQngcXXH60SmTtKNXmVexr+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]
9babccb43760: 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.

```
Y Deployment.yml
1  apiVersion: 2019-12-01
2  location: northeurope
3  name: SQLAppGroup
4  properties:
5    containers:
6      - name: db
7        properties:
8          image: appregistry1212.azurecr.io/appsqlimage:latest
9          resources:
10            requests:
11              cpu: 1
12              memoryInGb: 1.5
13            ports:
14              - port: 3306
15      - name: web
16        properties:
17          image: appregistry1212.azurecr.io/sqlapp:latest
18          resources:
19            requests:
20              cpu: 1
21              memoryInGb: 1.5
22            ports:
23              - port: 80
24    osType: Linux
25    ipAddress:
26      type: Public
27      ports:
28        - protocol: tcp
29        |  port: 80
30        - protocol: tcp
31        |  port: 3306
32    imageRegistryCredentials:
33      - server: appregistry1212.azurecr.io
34      |  username: appregistry1212
35      |  password: plvAXQngcXXH6OSmTtKNXmVexr+7EY/WsJNaox4QkK+ACRBDCzz3
36    type: Microsoft.ContainerInstance/containerGroups
37
```

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
```

Switch to Bash Restart Manage files

```
PS /home/fabricuser> ls
deployment.yml  Microsoft
PS /home/fabricuser>
```

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

```
PS /home/fabricuser> az container create --resource-group new-grp --file deployment.yml
|| Running ..
```

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.

The screenshot shows two main sections of the Azure portal:

Resource Group Overview: The left sidebar shows the 'new-grp' resource group with various navigation options like Overview, Activity log, Access control (IAM), Tags, Resource visualizer, Events, Settings, Cost Management, and Monitoring. The main area displays the 'Essentials' section with a table for Resources. One row is visible for 'SQLAppGroup' with columns: Name (db), Image (appregistry1212.azurecr.io/appsqlima...), State (Running), Previous state (-), Start time (2025-03-05T10:13:38.728Z), and Restart count (0).

Container Instances Details: This section is titled 'SQLAppGroup | Containers'. It shows 2 containers and 0 init containers. A table lists the 'db' and 'web' containers with their respective details. Below this, there is a 'Containers' blade with tabs for Events, Properties, Logs, and Connect. The Events tab shows a table of events with columns: Name, Type, First timestamp, Last timestamp, Message, and Count. The events listed are: Started (Normal, 5/3/2025, 03:43:38 pm IST, Started container, 1), Pulled (Normal, 5/3/2025, 03:43:29 pm IST, Successfully pulled image "appr...", 1), and Pulling (Normal, 5/3/2025, 03:43:07 pm IST, pulling image "appregistry1212...", 1).

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.

The screenshot shows a web browser window with the address bar displaying 'Not secure 40.127.253.247'. The page title is 'Information in the Activity Log'. Below the title is a table header with columns: Id, Operation name, Status, Event category, Resource type, and Resource. There is no data in the table body.

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.

```
1  using MySql.Data.MySqlClient;
2  using System.Data;
3  using System.Data.SqlClient;
4  using System.Reflection.Metadata;
5
6  string strFilePath = "C:\\\\tmp1\\\\Log.csv";
7  StreamReader? logReader = null;
8  int Id = 1;
9  MySqlConnection appdbConnection =
10 [new MySqlConnection("Server=40.127.253.247;UserID = root;Password=Azure123;Database=appdb");
11
12  MySqlParameter paramId = new MySqlParameter();
13  paramId.ParameterName = "@Id";
14
```

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

The screenshot shows a terminal window with the path 'C:\Users\PULKIT\Downloads\' at the top. The window contains the text: 'Written Record 1', 'Written Record 2', 'Written Record 3', 'Written Record 4', 'Written Record 5', 'Written Record 6', and 'Written Record 7'.

Not secure 40.127.253.247

sqlapp Home Privacy

Information in the Activity Log

Id	Operation name	Status	Event category	Resource type	Resource
1	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Update SQL database	Succeeded	Administrative	Informational
2	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Create Deployment	Started	Administrative	Informational
3	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Create Deployment	Accepted	Administrative	Informational
4	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Registers the Microsoft SQL Database Resource Provider	Started	Administrative	Informational
5	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Registers the Microsoft SQL Database Resource Provider	Succeeded	Administrative	Informational
6	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Update SQL server	Started	Administrative	Informational
7	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	'audit' Policy action.	Succeeded	Policy	Warning
8	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	'auditIfNotExists' Policy action.	Started	Policy	Informational
9	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Update SQL server	Accepted	Administrative	Informational
10	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Update SQL server firewall rules	Started	Administrative	Informational
11	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Update SQL server firewall rules	Started	Administrative	Informational
12	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Update Server Connection Policy Create	Started	Administrative	Informational
13	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Update SQL database	Started	Administrative	Informational
14	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Update Server Connection Policy Create	Succeeded	Administrative	Informational
15	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Update SQL server	Succeeded	Administrative	Informational
16	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	'auditIfNotExists' Policy action.	Started	Policy	Informational
17	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Update SQL database	Accepted	Administrative	Informational
18	99fe9c3a-e36e-44e0-acd4-58272ab10c7e	Update SQL server firewall rules	Succeeded	Administrative	Informational

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.