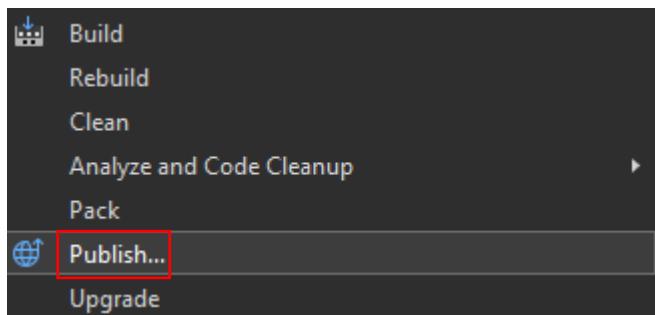
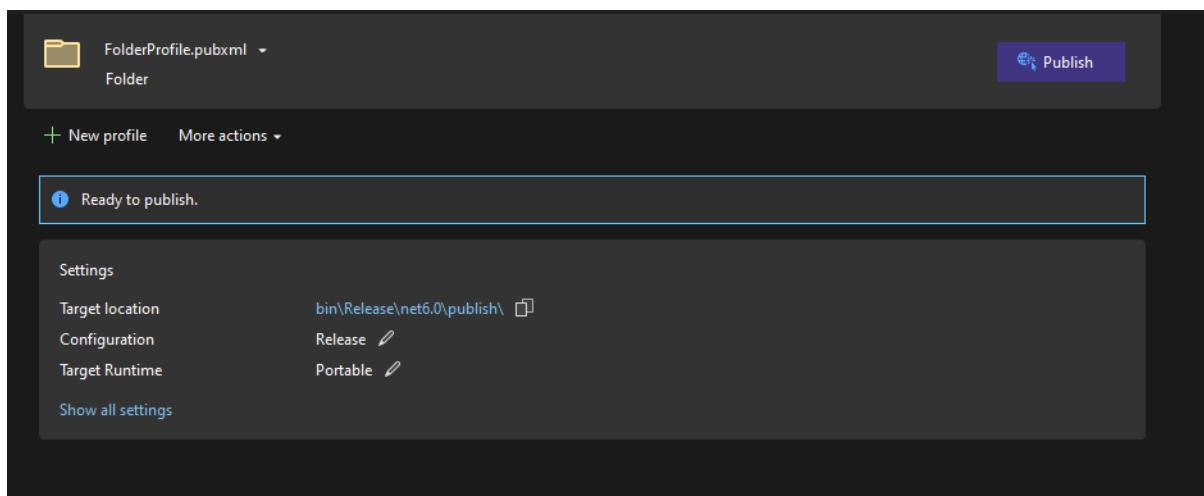


Deploying Docker

1. For this lab you are required to download two files from GitHub, first is Docker file and other one is Send Messages zip file.
2. After downloading them go to AWS Console and start building the setup.
3. First you need to navigate to EC2 and create an instance based on Ubuntu OS.
4. And you also have a SQS queue ready in order to send and receive messages.
5. Once you have both in place, then you need to extract the Send messages zip file and then open it in visual studio.
6. In visual studio from the top right corner, you need to right click on your project, then click on publish.



7. Then it will ask you where you want to store it just click on folder then you need to select the folder.
8. After that you will be on this screen shown below, here just need to click on publish it will create something known as publish profile.
9. What this does is it's going to make a runnable copy of your program that can be run on a machine that has .net in place.



10. Now you need to connect to your instance either with FileZilla or WinSCP and then you need to transfer files from your local machine to your instance.
11. In FileZilla or WinSCP whatever you are using, I am using FileZilla, so in there you need to go to the location where your Publish folder is stored.

12. You need to drag this publish folder and drop it into the instance which will be copied directly to your instance.

Filename	Filesize	Filetype	Last modified	Permissions	Owner/Group
..					
.cache		File folder	15-02-2024 09:...	drwx-----	ubuntu ubu...
.ssh		File folder	14-02-2024 20:...	drwx-----	ubuntu ubu...
publish		File folder	15-02-2024 09:...	drwxrwxr-x	ubuntu ubu...
.bash_logout	220	Bash Logo...	06-01-2022 21:...	-rw-r--r--	ubuntu ubu...
.bashrc	3,771	Bash RC So...	06-01-2022 21:...	-rw-r--r--	ubuntu ubu...
.profile	807	Profile Sou...	06-01-2022 21:...	-rw-r--r--	ubuntu ubu...

13. Once this is done SSH to your instance, and there you need to install Docker on your machine. You can visit to this website mentioned below, this will take you to the documentation page to install docker.

14. Use this particular website because the steps to install docker keeps changing as they are pushing updates time to time.

<https://docs.docker.com/engine/install/ubuntu/>

15. Now use this documentation page to install docker on your Instance. So, below are the commands which I used to install docker. You should cross check these commands on the docker documentation page they keep changing those.

```
# Add Docker's official GPG key:  
sudo apt-get update  
sudo apt-get install ca-certificates curl  
sudo install -m 0755 -d /etc/apt/keyrings  
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o  
/etc/apt/keyrings/docker.asc  
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

Add the repository to Apt sources:

```

echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \
$(./etc/os-release && echo "$VERSION_CODENAME") stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update

#Installing Docker
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin

```

16. Once you docker is installed. Similarly, we have to create something known as a Docker image. That Docker image should have everything that would be required to run an application along with the binary, you know the binary files for the application. Everything should be bundled as this image and then we create or run basically a container out of this image. You will have your application running within the container.
17. Now in the start I told you to download a File named Dockerfile. Now you need to copy that file in the publish folder with in the instance.

Remote site: /home/ubuntu/publish

The screenshot shows a file manager interface. The top section displays the directory structure:

```

/ 
  +-- home
    +-- ubuntu
      +-- .cache
      +-- .ssh
      +-- publish

```

The bottom section shows a detailed file list:

Filename	Filesize	Filetype	Last modified	Permissions	Owner/Group
..					
AWSSDK.Core.dll	1,871,520	Application	15-02-2024 09:...	-rw-rw-r--	ubuntu ubu...
AWSSDK.SQS.dll	139,424	Application	15-02-2024 09:...	-rw-rw-r--	ubuntu ubu...
Dockerfile	109	File	15-02-2024 11:...	-rw-rw-r--	ubuntu ubu...
Newtonsoft.Json.dll	712,464	Application	15-02-2024 09:...	-rw-rw-r--	ubuntu ubu...
SendMessages.deps.js...	2,220	JSON Sour...	15-02-2024 09:...	-rw-rw-r--	ubuntu ubu...
SendMessages.dll	8,192	Application	15-02-2024 09:...	-rw-rw-r--	ubuntu ubu...
SendMessages.exe	149,504	Application	15-02-2024 09:...	-rw-rw-r--	ubuntu ubu...
SendMessages.pdb	11,036	Program D...	15-02-2024 09:...	-rw-rw-r--	ubuntu ubu...
SendMessages.runtim...	253	JSON Sour...	15-02-2024 09:...	-rw-rw-r--	ubuntu ubu...

18. And then we're going to run some commands to go ahead and create that Docker image.

- Inside your instance first go to the publish folder then run the below command to build the image out of that docker file.
- The Dot indicates that the Docker file, along with all of the contents for that image, is available in the current publish folder.

```
sudo docker build -t sendmessages .
```

```
ubuntu@ip-172-31-0-26:~/publish$ cd publish
ubuntu@ip-172-31-0-26:~/publish$ sudo docker build -t sendmessages .
```

```
ubuntu@ip-172-31-0-26:~/publish$ sudo docker build -t sendmessages .
[+] Building 6.4s (8/8) FINISHED
--> [internal] load build definition from Dockerfile
--> => transferring dockerfile: 146B
--> [internal] load metadata for mcr.microsoft.com/dotnet/runtime:6.0
--> [internal] load .dockerrcignore
--> => transferring context: 2B
--> [1/3] FROM mcr.microsoft.com/dotnet/runtime:6.0@sha256:782d1ef1341eed06a3b19f205d1f9274f1cf0e4542c95af13721f1 5.4s
--> => resolve mcr.microsoft.com/dotnet/runtime:6.0@sha256:782d1ef1341eed06a3b19f205d1f9274f1cf0e4542c95af13721f1 0.0s
--> => sha256:782d1ef1341eed06a3b19f205d1f9274f1cf0e4542c95af13721f1097fc9782 1.79kB / 1.79kB 0.0s
--> => sha256:26495b5ac2d20e8017e72a4529671c34afc85169b80f9eb05a1ace45579f0238 1.16kB / 1.16kB 0.0s
--> => sha256:7bffa6240f9d50abd6a36b67bdad6f616b2fb7af2bed328f0e82413a882b4b3 1.92kB / 1.92kB 0.0s
--> => sha256:5d0aaeccef7eeb53c3f853fb229ea7fd13a5a56f4ba371ca48f04774938046b782 31.42MB / 31.42MB 0.9s
--> => sha256:7c2bfda75264da4277b3ac99bde1258e08cb153c5281e10643a577e536a1f7c 15.17MB / 15.17MB 0.8s
--> => sha256:950196e58fe38d437496132b6e5e1335c768e968a856157cf8fa1d963c0ceae 31.66MB / 31.66MB 1.1s
--> => sha256:ecf3c05ee2f667bb83b7782b0e554d187d8abe2beead0f9a4f19c0243c0f27fb 1548 / 1548 0.9s
--> => extracting sha256:5d0aaeccef7eeb53c3f853fb229ea7fd13a5a56f4ba371ca48f04774938046b782 2.1s
--> => extracting sha256:7c2bfda75264da4277b3ac09b0e1258e48c8153c5281e10643a577e536a1f7c 0.6s
--> => extracting sha256:950196e58fe38d437496132b6e5e1335c768e968a856157cf8fa1d963c0ceae 1.3s
--> => extracting sha256:ecf3c05ee2f667bb83b7782b0e554d187d8abe2beead0f9a4f19c0243c0f27fb 0.0s
--> [internal] load build context
--> => transferring context: 2.90MB 0.1s
--> [2/3] WORKDIR /app 0.2s
--> [3/3] COPY . 0.1s
--> => exporting to image 0.1s
--> => exporting layers 0.1s
--> => writing image sha256:56969356d782633bfbc72b7954bfc7434bd87719f6e8141713215b20b01b1c7c 0.0s
--> => naming to docker.io/library/sendmessages 0.0s
ubuntu@ip-172-31-0-26:~/publish$ |
```

- Out of this image now you can run a container.
- Now if you will run this command, you will face an error because the EC2 instance don't have the required permission.

```
sudo docker run --name sendmessages-sqs sendmessages
```

```
ubuntu@ip-172-31-0-26:~/publish$ sudo docker run --name sendmessages-sqs sendmessages
Unhandled exception. Amazon.Runtime.AmazonServiceException: Unable to get IAM security credentials from EC2 Instance Metadata Service.
at Amazon.Runtime.DefaultInstanceProfileRoleCredentials.FetchMetadataAsync()
at Amazon.Runtime.DefaultInstanceProfileRoleCredentials.GetCredentials()
at Amazon.Runtime.DefaultInstanceProfileRoleCredentials.GetAsync()
at Amazon.Runtime.Internal.CredentialsRetriever.InvokeAsync[T](IExecutionContext executionContext)
at Amazon.Runtime.Internal.RetryHandler.InvokeAsync[T](IExecutionContext executionContext)
at Amazon.Runtime.Internal.RetryHandler.InvokeAsync[T](IExecutionContext executionContext)
at Amazon.Runtime.Internal.CallbackHandler.InvokeAsync[T](IExecutionContext executionContext)
at Amazon.Runtime.Internal.CallbackHandler.InvokeAsync[T](IExecutionContext executionContext)
at Amazon.Runtime.Internal.ErrorCallbackHandler.InvokeAsync[T](IExecutionContext executionContext)
at Amazon.Runtime.Internal.MetricsHandler.InvokeAsync[T](IExecutionContext executionContext)
at Program.<Main>()>_SendMessage[0,0](IAmazonSQS sqsClient, String queueUrl, String messageBody) in C:\Users\PULKIT\Downloads\SendMessages\SendMessages\SendMessages\Program.cs:line 22
at Program.<Main>$(String[] args) in C:\Users\PULKIT\Downloads\SendMessages\SendMessages\SendMessages\Program.cs:line 16
at Program.<Main>(String[] args)
ubuntu@ip-172-31-0-26:~/publish$ |
```

- Now you have to go to IAM roles and create a role for EC2 instance. Add this permission to your role.
- Then you need to attach this role with your EC2 instance.

Filter by type		All types	3 matches	< 1 >	⋮
Policy name	Type	Description			
<input checked="" type="checkbox"/> AmazonSQSFullAccess	AWS managed	Provides full access to Amazon SQS via t...			
<input type="checkbox"/> AmazonSQSReadOnlyAccess	AWS managed	Provides read only access to Amazon SQ...			
<input type="checkbox"/> AWSLambdaSQSQueueExecutionRole	AWS managed	Provides receive message, delete messag...			

25. Once you have attached your role with EC2 then rerun the command.
26. But you will face an error and it will say that the container name is already in use.
27. For that you will need to remove the container ID only.

```
sudo docker ps -a
sudo docker rm CONATINER ID
```

```
ubuntu@ip-172-31-0-16:/publish$ sudo docker run --name sendmessages-sqs sendmessages
docker: Error response from daemon: Conflict: The container name "/sendmessages-sqs" is already in use by container "0e0079bf5438d70c45e49e32b817a96b363e723a93c679d7b84dc16c1c845287". You have to remove (or re
name) that container to be able to reuse that name.
See 'docker run --help'.
ubuntu@ip-172-31-0-26:/publish$ sudo docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
0e0079bf5438        sendmessages        "dotnet SendMessages..."   4 minutes ago     Exited (139) 4 minutes ago          sendMessages-sqs
ubuntu@ip-172-31-0-26:/publish$ sudo docker rm 0e0079bf5438
0e0079bf5438
```

28. Once the container ID is removed now you need to run the docker command to send messages.

```
ubuntu@ip-172-31-0-26:~/publish$ sudo docker run --name sendmessages-sqs sendmessages
Message added to queue
ubuntu@ip-172-31-0-26:~/publish$
```

29. Then you need to come to SQS queue and click on start poll.
30. You will see the pending messages there which have been send through the container.

Receive messages		Info		Edit poll settings		Stop polling	Poll for messages
Messages available	5	Polling duration	30	Maximum message count	10	Polling progress	50%
						1.7 receives/second	
Messages (5)		View details		Delete			
<input type="checkbox"/> Search messages							
ID	Sent	Size	Receive count				
48dc0a8f-e289-44de-a2e2-60205137f42f	2024-02-15T11:26+05:30	51 bytes	1				
729d9361-f203-45c6-87ee-99701a3eba52	2024-02-15T11:26+05:30	52 bytes	1				
c0d402e7-7aec-4063-9708-e0f74a153eaf	2024-02-15T11:26+05:30	52 bytes	1				
7ed0ae8a-75be-4fc1-a45b-b5092b9b832c	2024-02-15T11:26+05:30	52 bytes	1				
f26fb3bf-17b9-4ada-97d8-B2ca03e100f8	2024-02-15T11:26+05:30	52 bytes	1				