Deployment with Docker and AWS

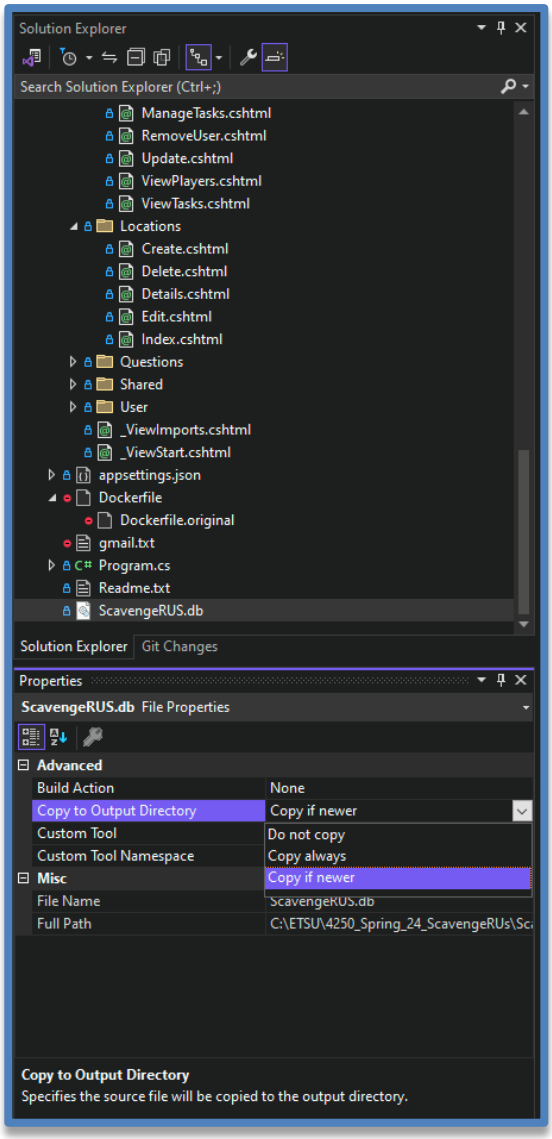
Updated Documentation Fall 2024

Prerequisites

- Create a Docker account at https://hub.docker.com
- Download Docker Desktop https://www.docker.com/products/docker-desktop/
    - You will have to log in
- Create an AWS account at https://portal.aws.amazon.com/billing/signup
    - You will have to provide credit card info. Free tier provides an allowance of time before charging. Make sure to stop your instance and this should not be a problem.
- (Optional) This is for Windows machines, steps will differ for Mac/Linux

Prep Database for Docker (For Development)

This technique will ensure that the database file already in the project files will retain its data when published to Docker as an image. This is useful to Developers as it can be used to quickly test database transactions with pre-populated data. However, there is a caveat: each time the image is deployed to a container, it will only contain the data from when the image was originally published



1. In the solution explorer, click the ScavengeRUS.db (database) file
2. Look at the option Copy to Output Directory and set it to *Copy if newer* or *Copy always*

Visual Studio to Docker

1. In Visual Studio, while in your project navigate to the top bar and click “Build” and then “Publish <Project name>”
    - If you do not see a publish with Docker option, make sure you open the project by double-clicking the *.sln* or *.csproj* file (opens project as a solution)

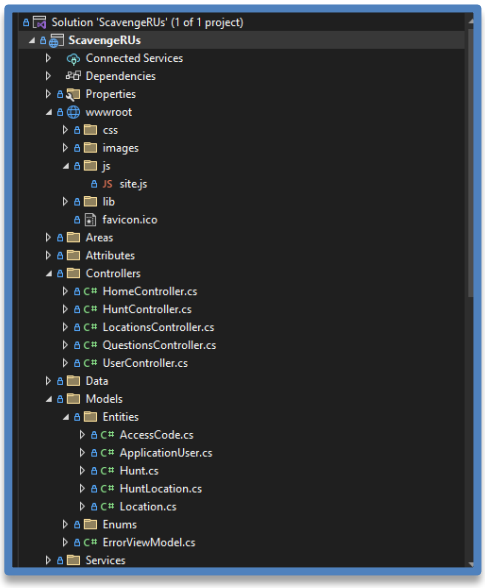
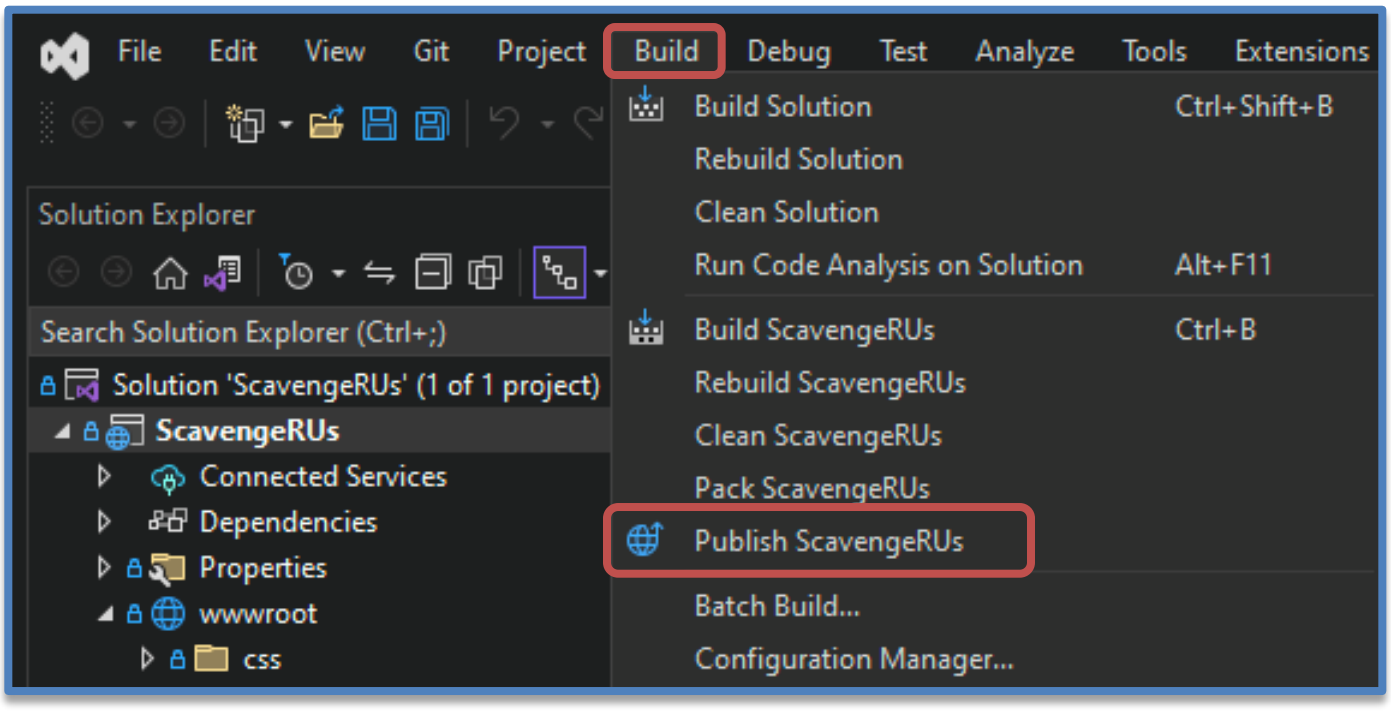
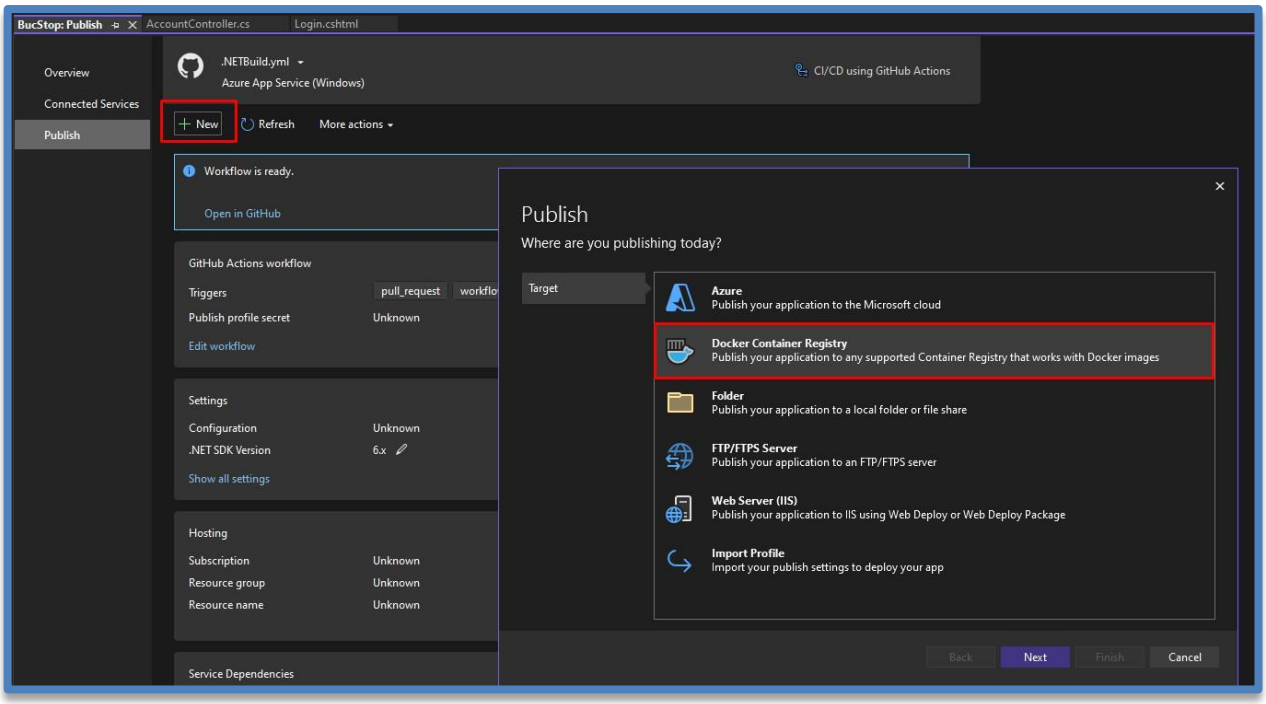


Figure 1 - You should see something like this if you opened it correctly

    - If you are on Mac/Linux, Visual Studio support is sparse or no-longer supported. You can right click to add Docker Support, but you will not be able to deploy to AWS or to your Docker account

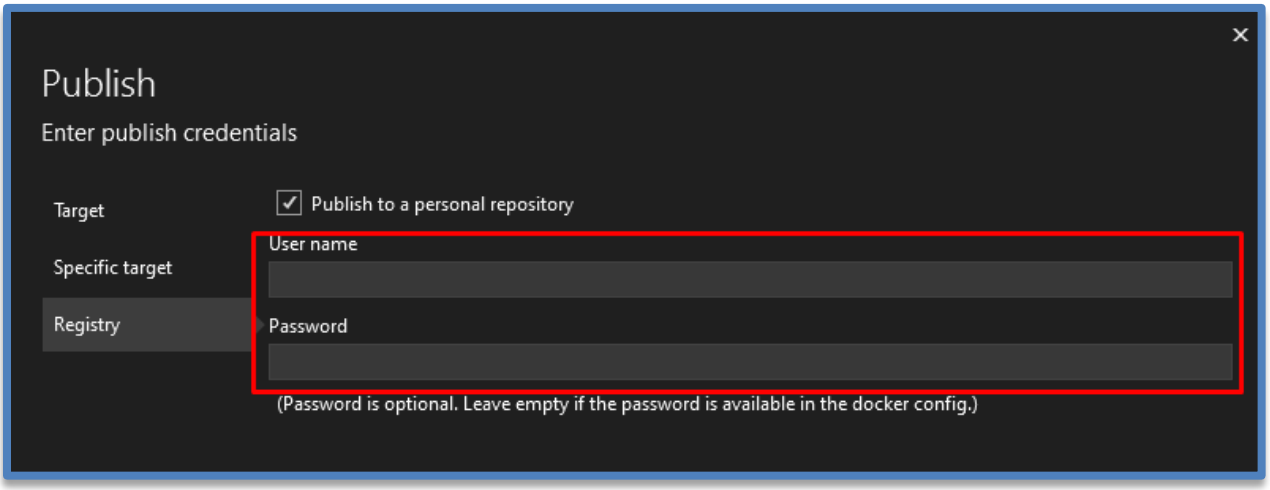


2. If this is the first time, you have to press ***New*** and then choose your target on where you want to publish, in this case it will be ***Docker***. Once you press next, you will have three options to choose from, choose the one that says ***Docker Hub***

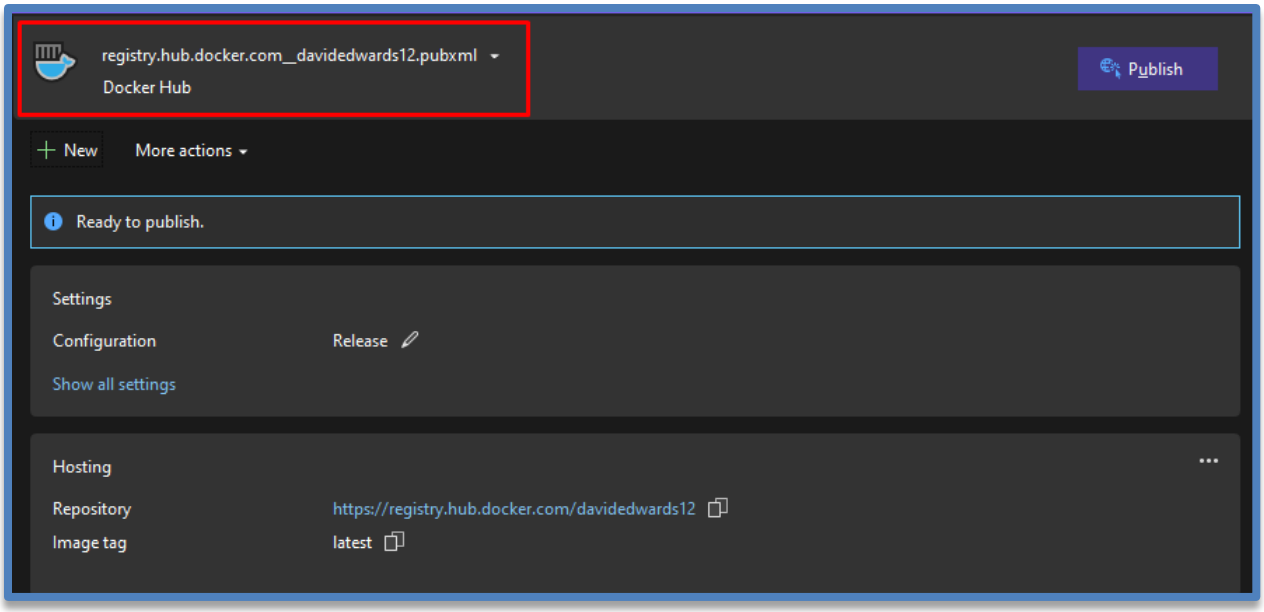


Note: You might have to install some packages in your VS if you do not have the required packages. If this is the case, go ahead and install them and then continue.

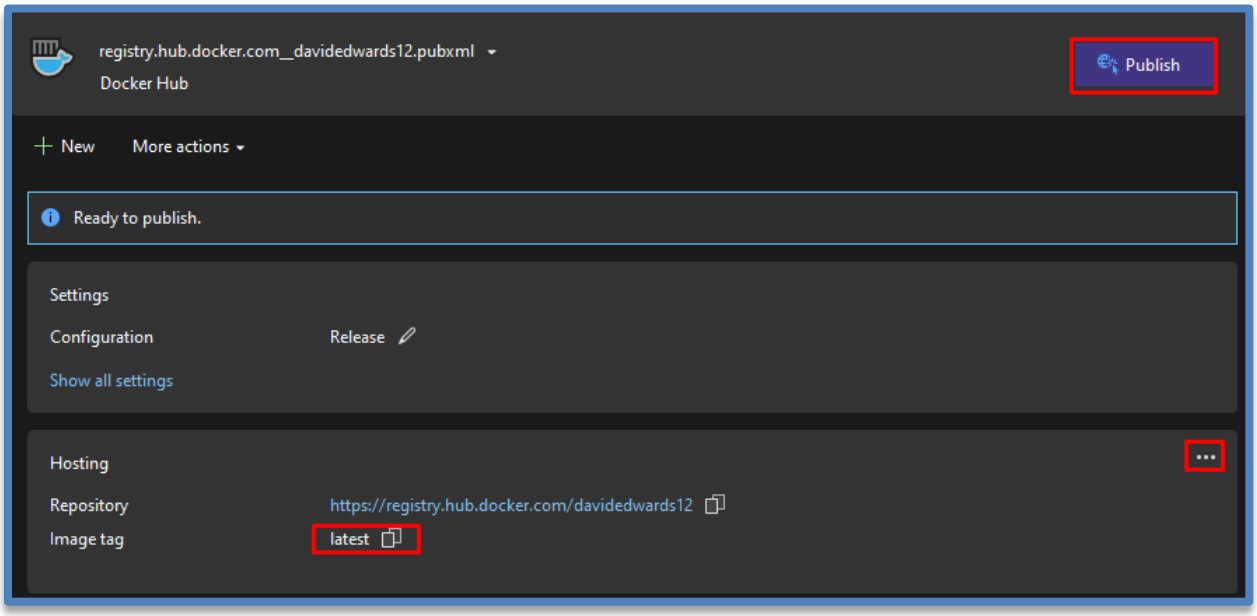
3. Next, log in into your Docker Hub account that you created at the start. Once you log in, you can press **Close**



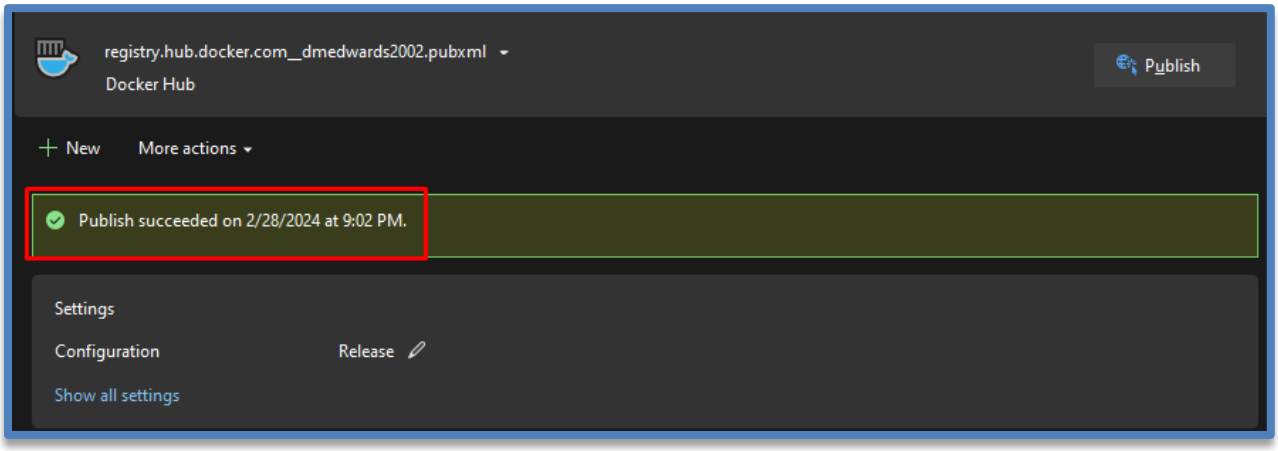
4. Back in the publish tab, you should see that you are now logged into your Docker Hub account and can publish to your Docker Hub



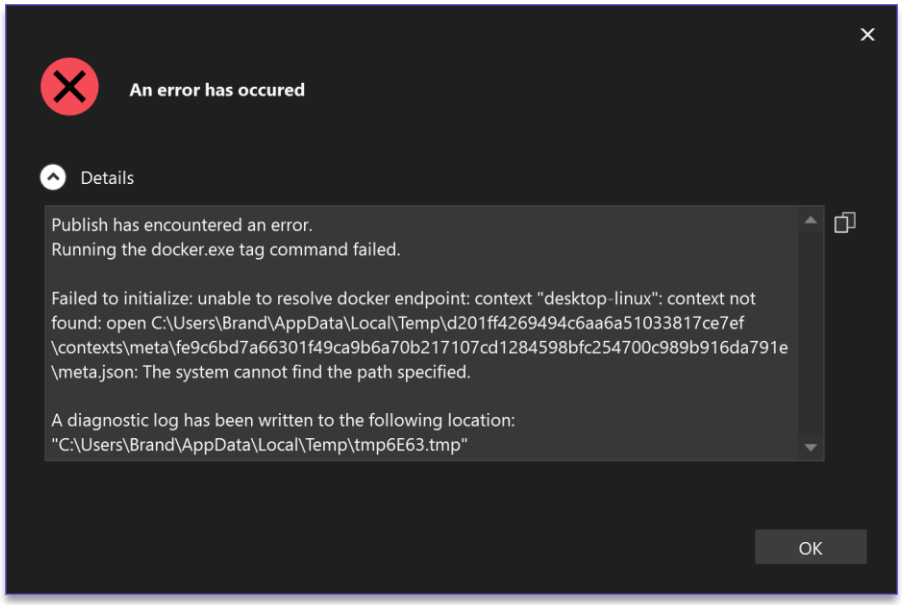
5. Now we can publish the project! Here, you can change the image tag by pressing the three little dots to the side (I think of the image tag as the version of the project, i.e., 1.0.0, 1.0.1, etc..). You can leave it latest if you want to, just know that it replaces the last latest you had. Once you have the image tag set to whatever you want, you can now press the "Publish button" at the top (if you have not downloaded Docker Desktop by this point, you will need to now as VS requires it to publish)



6. You will know you have published if you see this

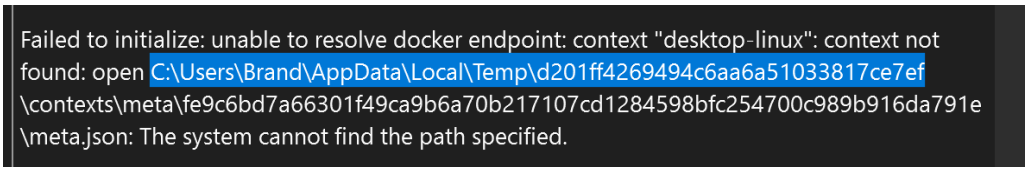


7. If the build has failed with the following error, we can fix this by copying the “context” folder into the location needed.

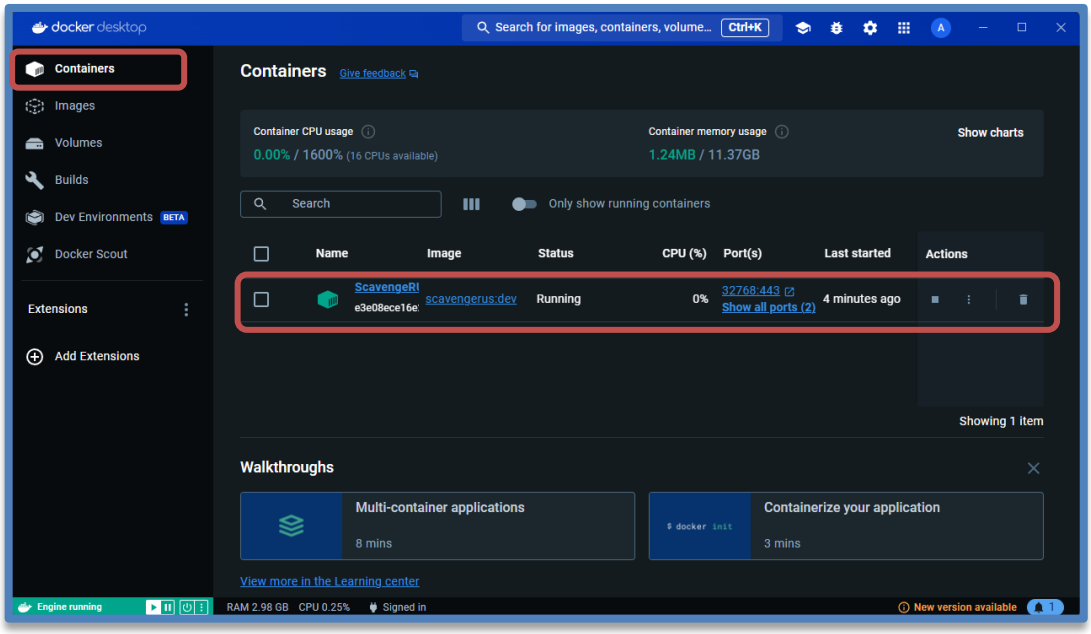


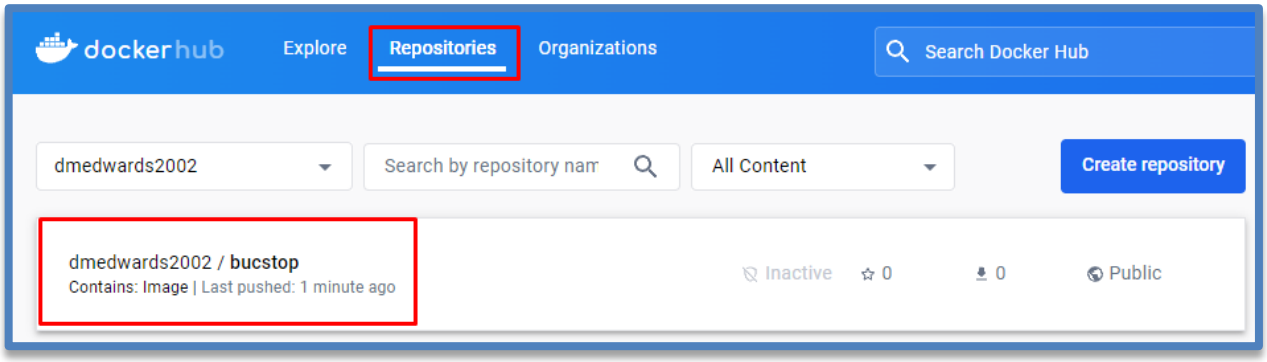
8. Locate your context folder in the .docker folder.
   Should be “C:\Users\{YOUR USERNAME}\.docker.

9. Copy the context folder into the path of the error code prior to the \contexts\meta…



10. Once you have that, you can go to your Docker Hub account and under repositories you should see your newly published project. You should be able to see this in the Docker Desktop as well
    - If you do not have both of these, try running it again from the log-in step onward



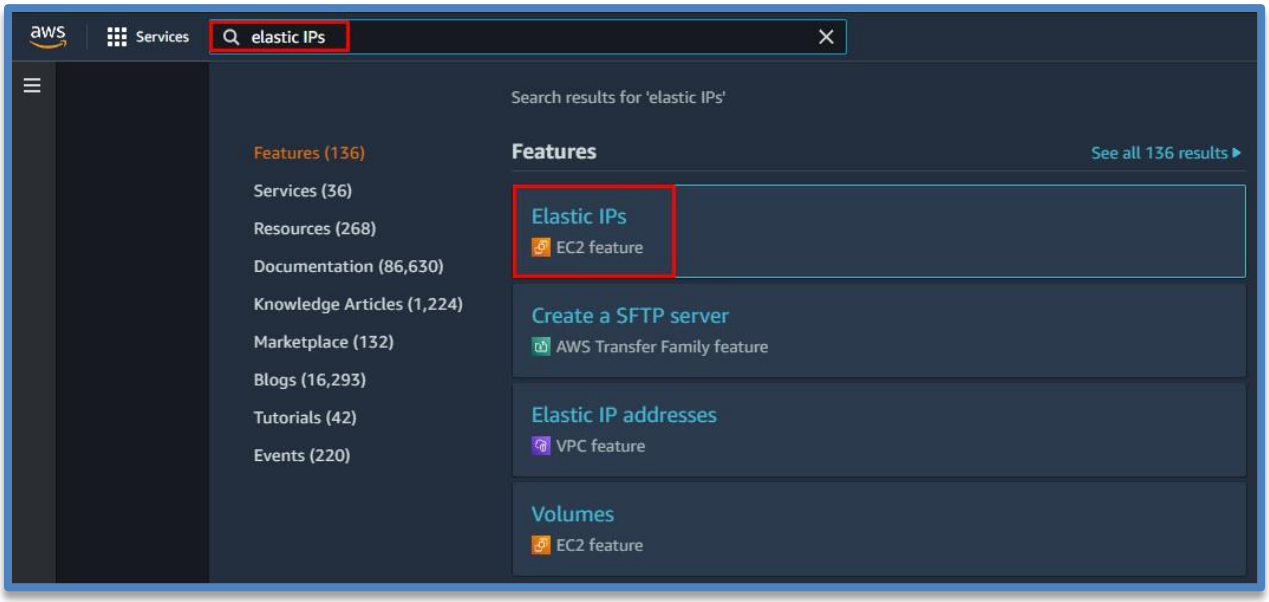


You can now deploy two different ways

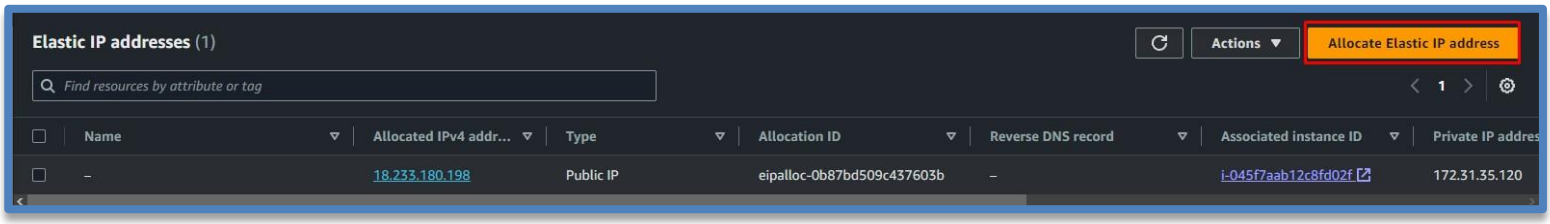
- Locally through the Docker Desktop application that we downloaded earlier
- Go through a public IP address with AWS. For this demo, we will be using AWS and an elastic IP address (more on that later).

Setting Up AWS

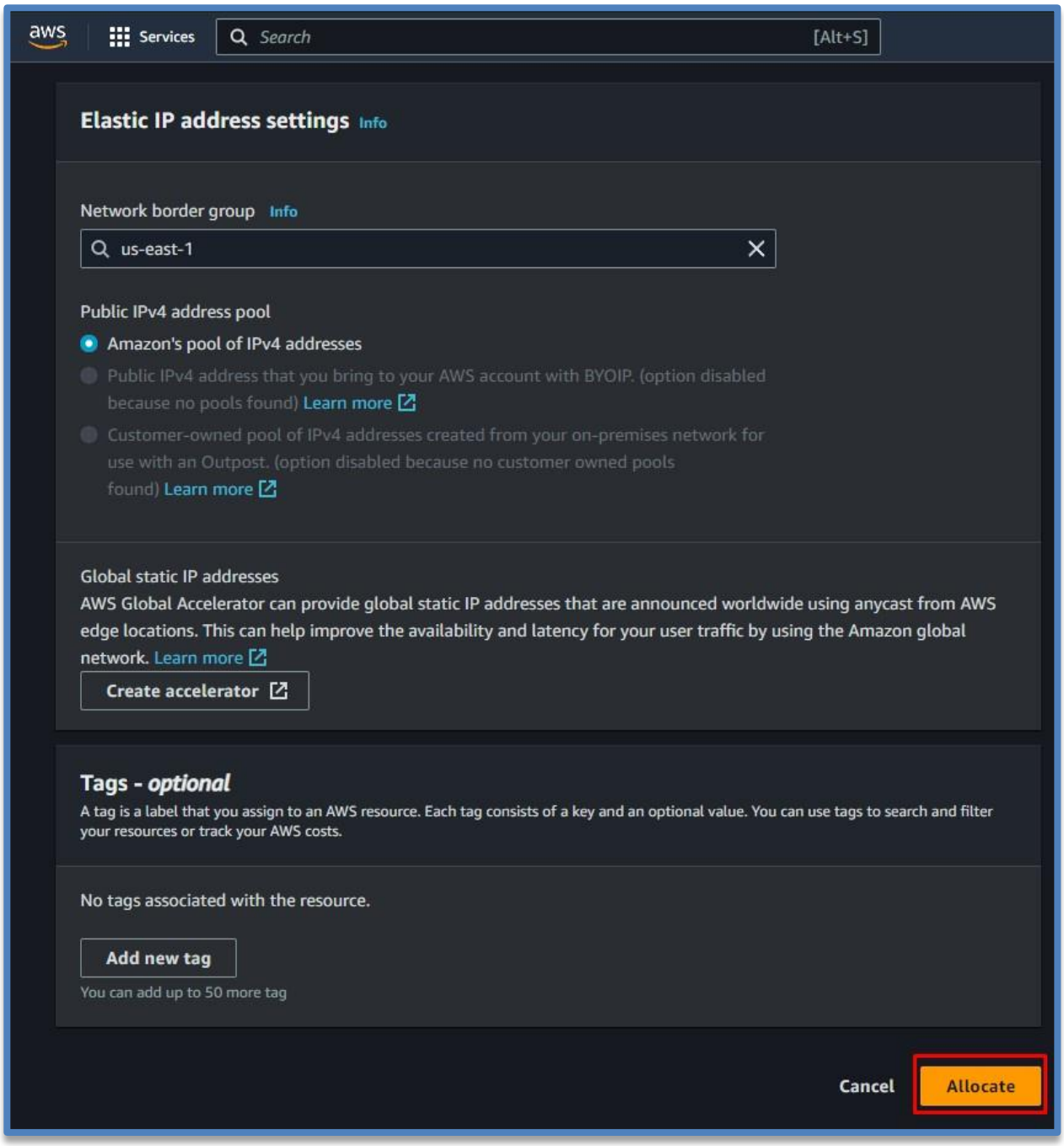
1. Assuming you have already made an AWS account, head over to the search bar at the top and search for “Elastic IPs”, we will use this later.



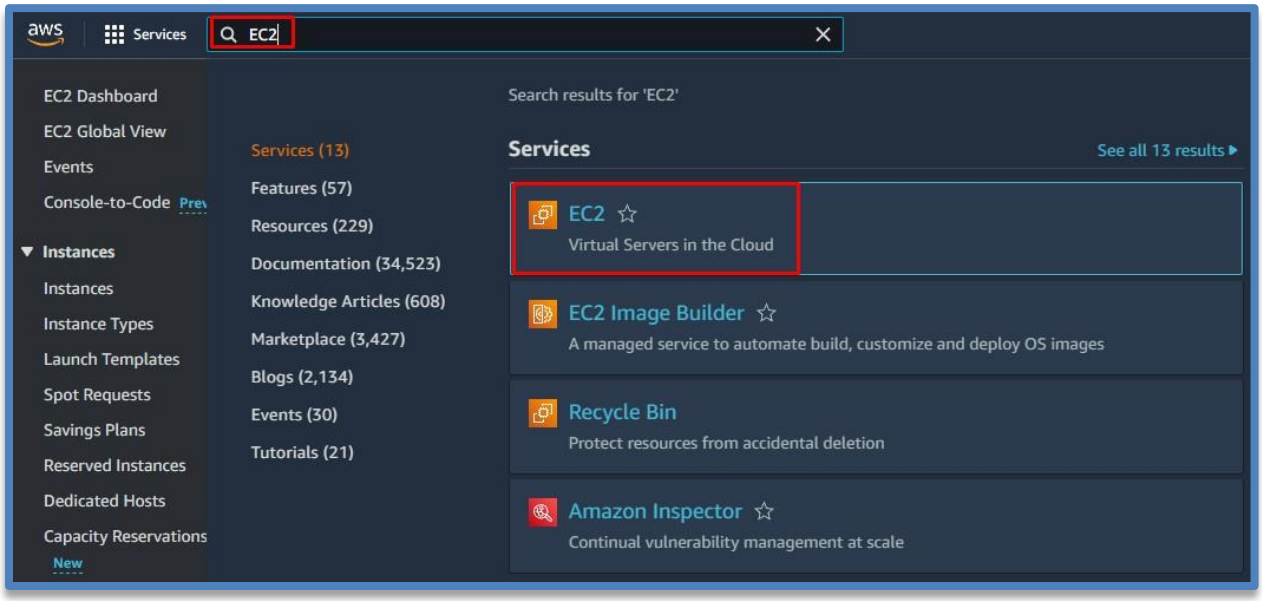
2. In the top right, click on **Allocate Elastic IP address**



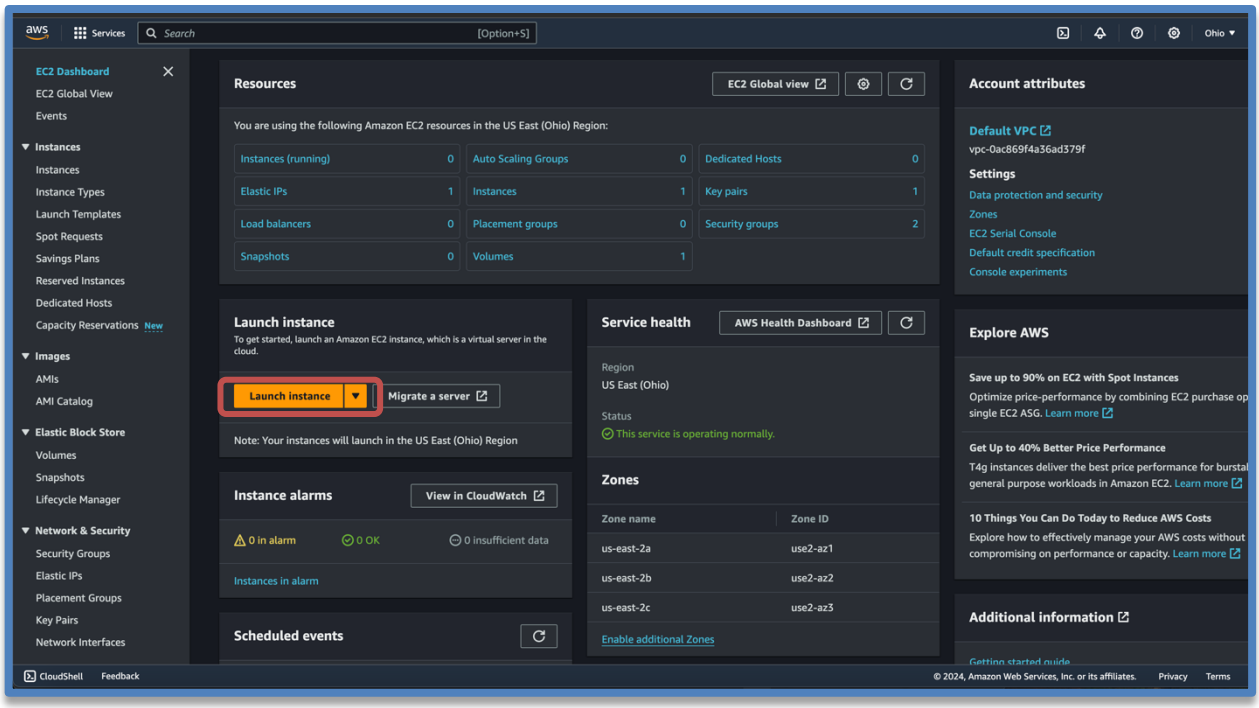
3. I recommend leaving everything as it is and pressing **Allocate** at the button. You will be given an IP address to use when we set up our virtual server



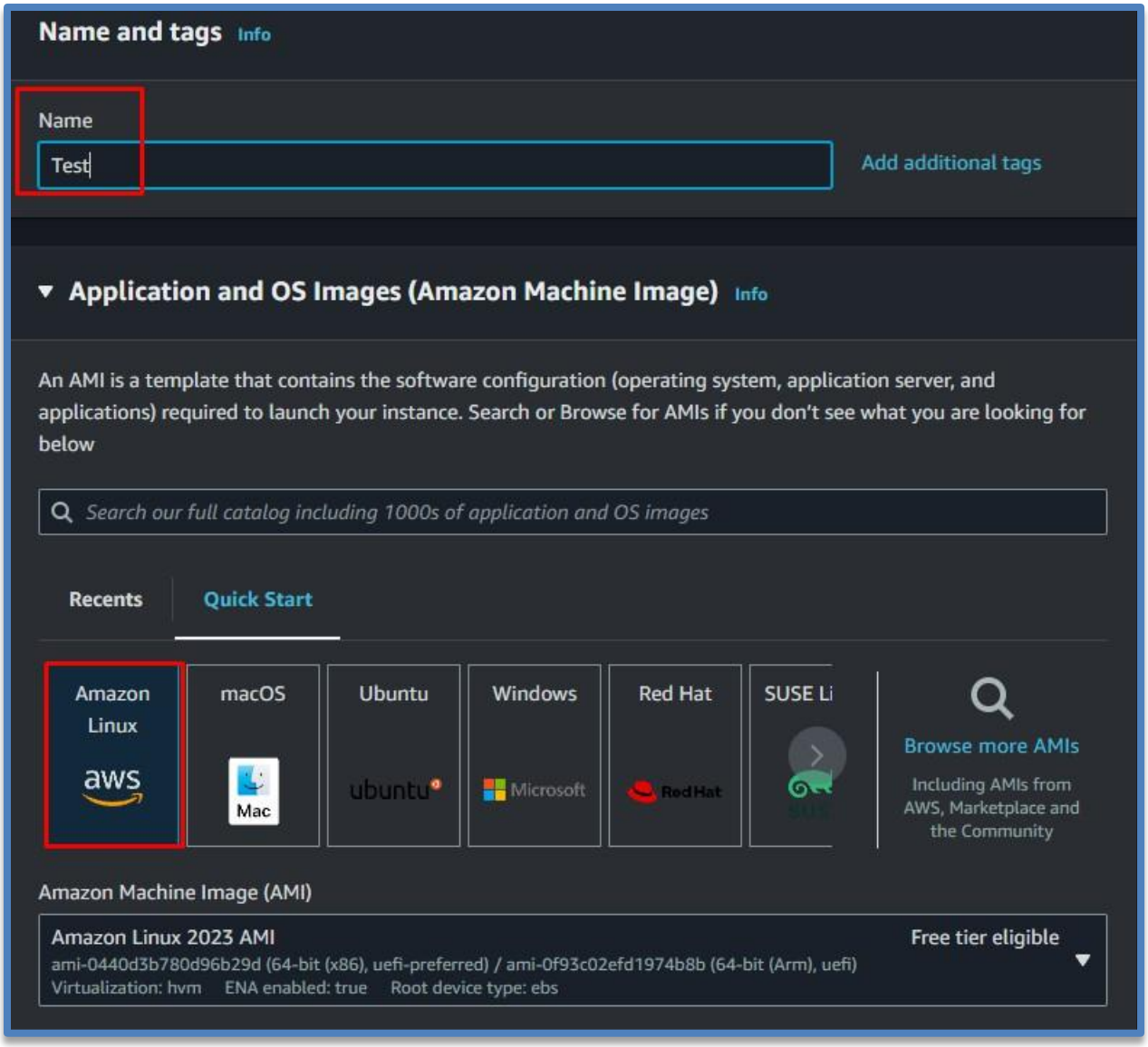
4. Next, we search for “EC2.” This is where we set up our virtual server



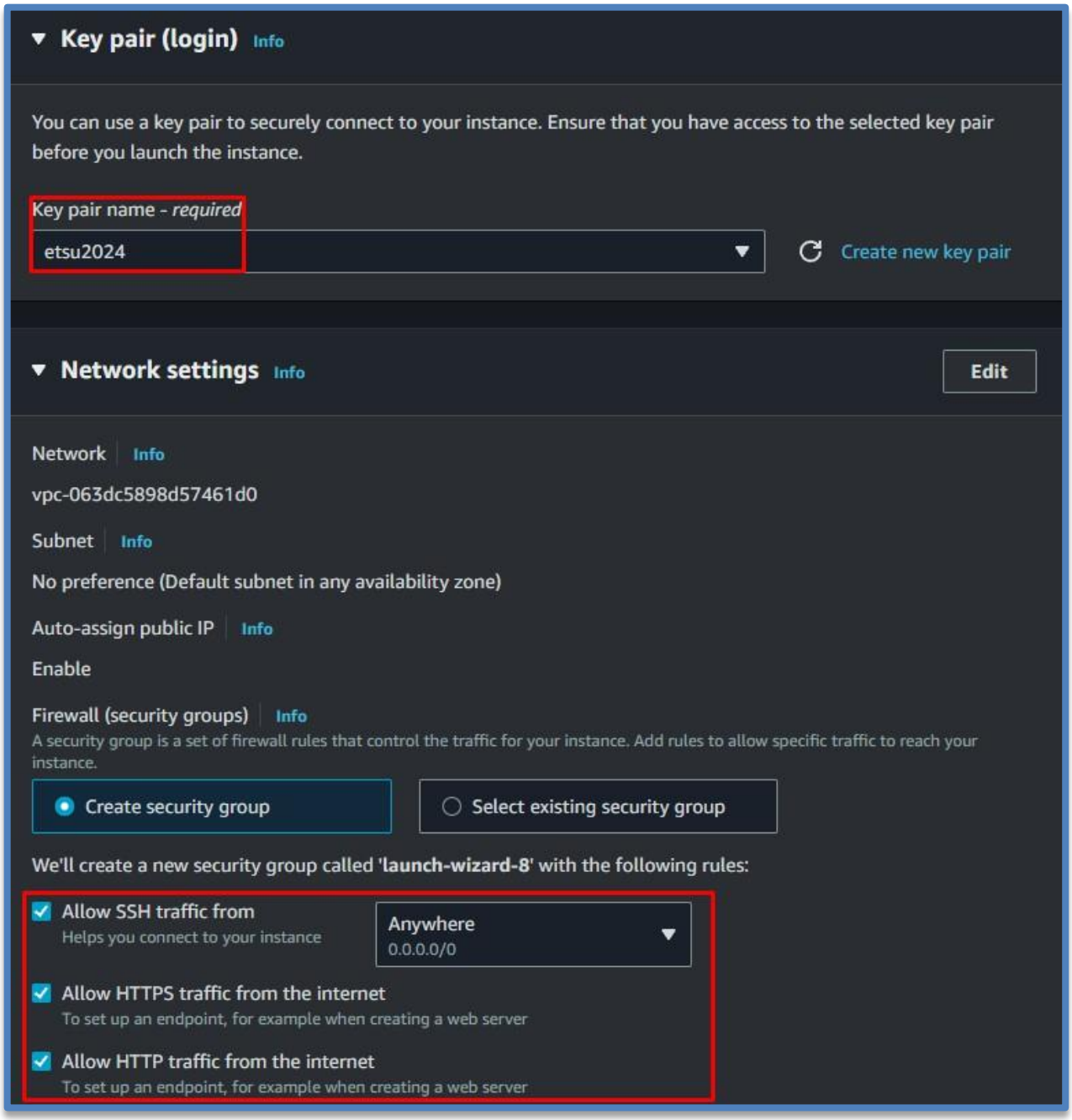
5. On that page, you will want to press **Launch instance**



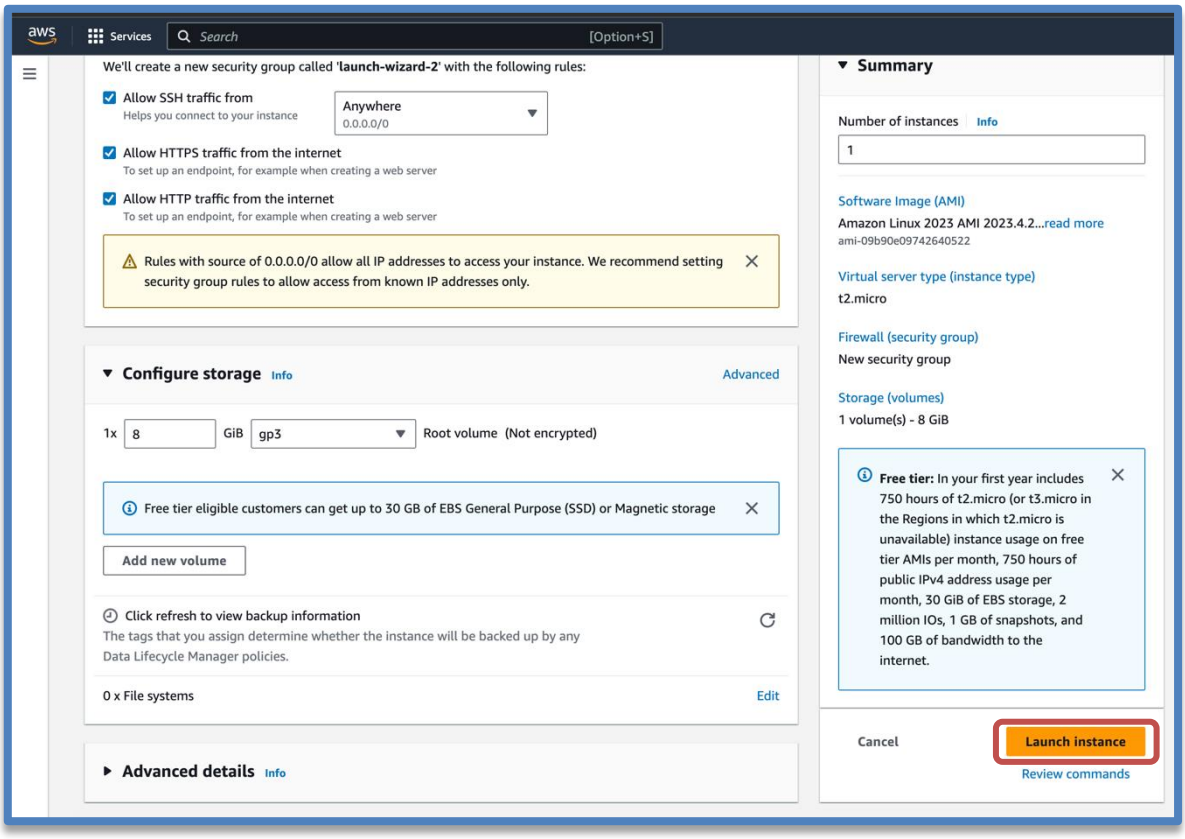
6. Now we are setting up the instance. Name it and define an image. For this we will be using Amazon Linux



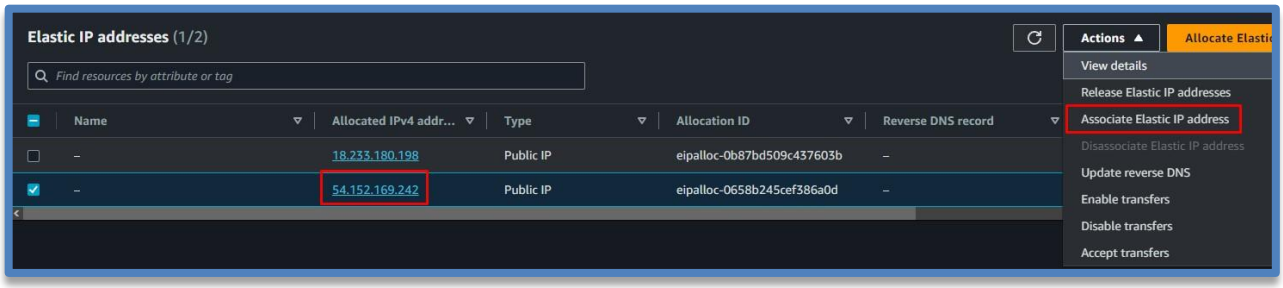
7. (Optional) Scrolling down, you can use a Key Pair to secure the instance more. It is not necessary but recommended. Then, in the network settings, check all the boxes. This will produce a certificate to download and hold on to.


Key pair (login) Info
You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.
Key pair name - required
etsu2024
Create new key pair
Network settings Info
Edit
Network Info
vpc-063dc5898d57461d0
Subnet Info
No preference (Default subnet in any availability zone)
Auto-assign public IP Info
Enable
Firewall (security groups) Info
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.
Create security group
Select existing security group
We'll create a new security group called 'launch-wizard-8' with the following rules:
Allow SSH traffic from
Helps you connect to your instance
Anywhere
0.0.0.0/0
Allow HTTPS traffic from the internet
To set up an endpoint, for example when creating a web server
Allow HTTP traffic from the internet
To set up an endpoint, for example when creating a web server

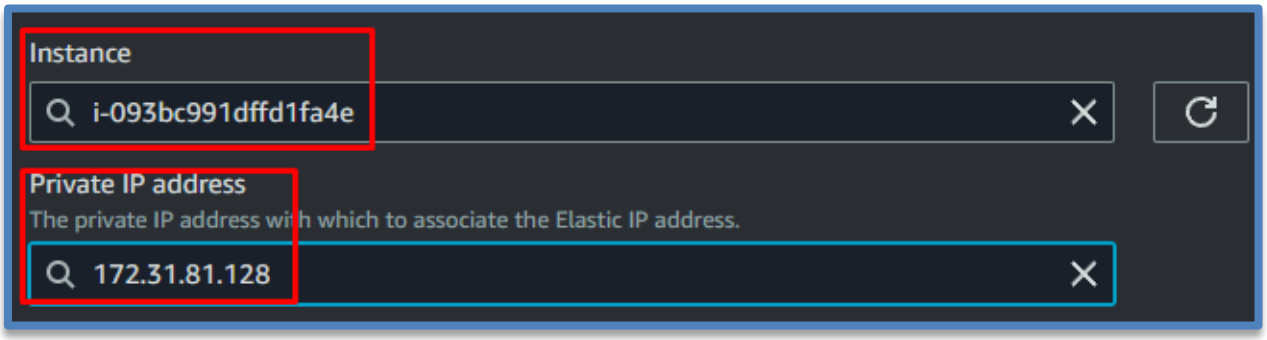
8. Finally, press **Launch Instance** on the side.



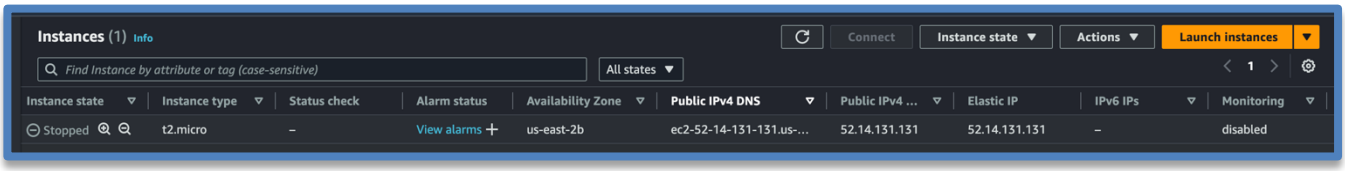
9. Before you connect to the instance, add the elastic IP to the instance. Go back to the Elastic IPs and select the one you got and press “Actions” and then “Associate Elastic IP Address”



10. Choose the instance that you just created and select the provided private IP, then press Associate.



11. When you go back to your instances, you should now see that the Elastic IP is now associated with the instance that you selected



12. Connect to your instance with all the settings that are provided

Steps 13-20 will show you how to deploy a single page without any connecting applications or microservices. Skip to step 21 for connecting two applications.

Single Application

13. Now that we are connected to the instance, we have to install docker on the instance to be able to run the commands. We use commands
    sudo yum update -y"
    to make sure everything is up to date and then use
    sudo yum install docker

```
[ec2-user@ip-172-31-81-128 ~]$ sudo yum update -y
Last metadata expiration check: 0:10:55 ago on Thu Feb 2
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-81-128 ~]$ sudo yum install docker
```

14. Run the command
    sudo service docker start
    To start the Docker service

15. Now run the command
    sudo su
    To get into the root account. You will run docker commands from this path

16. We now pull in our Docker image that we published using VS.
    To do this, we run the command
    docker pull $username/<project-name>:<tag>
    in my case, this will be
    docker pull ace1020302/scavengerus:dev
    To make sure you got the correct one, you can run "docker images" and it will show all the

```
[root@ip-172-31-81-128 ec2-user]# docker pull dmedwards2002/bucstop:latest
latest: Pulling from dmedwards2002/bucstop
5d0aeceef7ee: Pull complete
7c2bfda75264: Pull complete
950196e58fe3: Pull complete
ecf3c05ee2f6: Pull complete
819f3b5e3ba4: Pull complete
20ae32215d86: Pull complete
4f4fb700ef54: Pull complete
58b9c250b831: Pull complete
Digest: sha256:d9ee3fd5ac8ff0ee3686a99dccf5a84273c84df536737221480dadba855b35fc
Status: Downloaded newer image for dmedwards2002/bucstop:latest
docker.io/dmedwards2002/bucstop:latest
[root@ip-172-31-81-128 ec2-user]# docker images
REPOSITORY              TAG       IMAGE ID       CREATED          SIZE
dmedwards2002/bucstop   latest    5a80cb85901b   54 minutes ago   218MB
```

images you have in the instance

17. Next, we run the image using
    docker run -d -p<port>:<port> $username/<project-name>:<tag>
    for me it will look like
    docker run -d -p80:80 ace1020302/scavengerus:dev

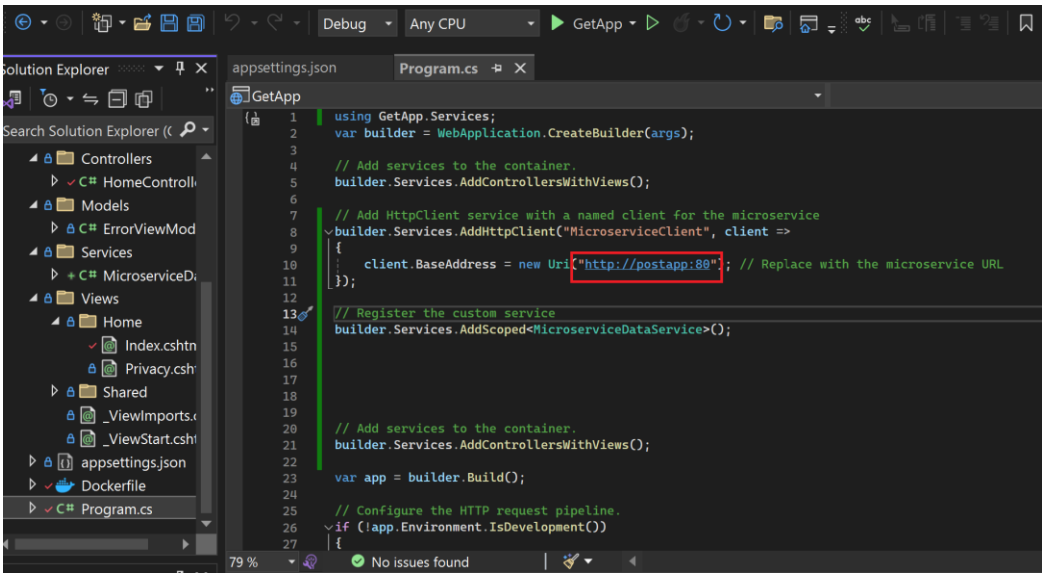
18. (Optional) To make sure that the image is running run:
    docker container ls
    This will list all of your active containers

19. Now, to make sure it all worked, you can copy and paste that Elastic IP address into your address bar and press enter! If you do not remember what the IP is, then at the bottom of the instance, it will tell you what your public IP address is

20. And now you can give this IP to anyone while you have the container running, and they will be able to see your project!

Multiple Applications

NOTE: This requires you to set the correct URI's in the source code so the two applications can talk. For example, the basic apps in this tutorial are the GET app and POST app. Below is the code I needed to change in the GET app in order for it to communicate with the POST app.

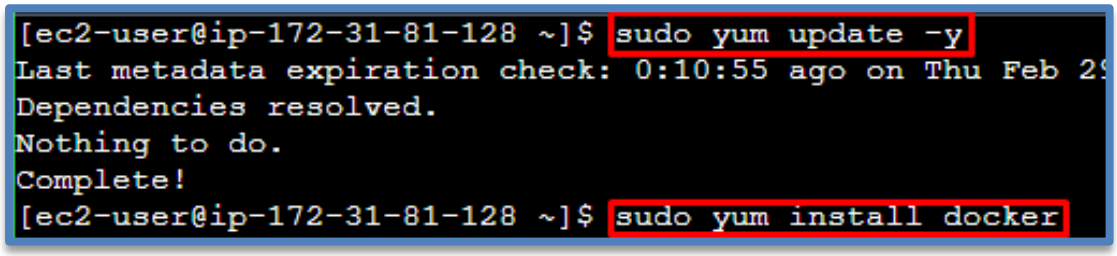


21. Now that we are connected to the instance, we have to install docker on the instance to be able to run the commands. We use commands

    sudo yum update -y"

    To make sure everything is up to date and then use

    sudo yum install docker



22. Run the command to start the Docker service

    sudo service docker start

23. Now run the command to get into the root account. You will run docker commands from this path.

    sudo su

24. We now pull in our Docker images that we published using VS.

    To do this, we run the command

    docker pull $username/<project-name>:<tag>

in this case it will be

docker pull brandonriddle828/getapp:latest

docker pull brandonriddle828/postapp:latest

```
[root@ip-172-31-1-219 ec2-user]# docker pull brandonriddle828/postapp:latest
latest: Pulling from brandonriddle828/postapp
6dce3b49cfe6: Already exists
a4561b904f51: Already exists
e85481dfd30f: Already exists
1c3b6f9e7b03: Already exists
a352695c487b: Already exists
32fe0028ff85: Already exists
4f4fb700ef54: Already exists
e288d641c7a6: Pull complete
Digest: sha256:64c335b71192c5787b716d341d9094ae2b465ca71a6fc9caea9034d97b16536e
Status: Downloaded newer image for brandonriddle828/postapp:latest
docker.io/brandonriddle828/postapp:latest
[root@ip-172-31-1-219 ec2-user]# docker pull brandonriddle828/getapp:latest
latest: Pulling from brandonriddle828/getapp
Digest: sha256:c683a0f162b4a3c8a565ded3ade027b5b852fc339f14eade727fae74156db001
Status: Image is up to date for brandonriddle828/getapp:latest
docker.io/brandonriddle828/getapp:latest
[root@ip-172-31-1-219 ec2-user]#
```

25. To make sure you got the correct one, you can run docker image to show all the images you have in the instance

    docker images

```
[root@ip-172-31-1-219 ec2-user]# docker images
REPOSITORY                  TAG       IMAGE ID       CREATED       SIZE
brandonriddle828/getapp     latest    b480539dfaea   2 weeks ago   217MB
brandonriddle828/postapp    latest    2df7b8af56c8   2 weeks ago   212MB
[root@ip-172-31-1-219 ec2-user]#
```

26. Now we need to create the network that both these images will use to communicate with each other using

    docker network create {name the network}

```
[root@ip-172-31-1-219 ec2-user]# docker network create get-post-network
3aabedb0e4016bff59a0580e5dfbea488b3b597357991d6b3550460f2c7b1930
[root@ip-172-31-1-219 ec2-user]#
```

27. Now we need to name and run these apps on the network we just created using the following command:

    docker run -d --name {name of the image} --network {your network name} -p80:80 {docker image and location}

NOTE: You must configure the ports differently on each app. The last number is where they will be connected so that must be the same. The first number, however, must be different!

This instance will be using:
docker run -d --name getapp --network get-post-network -p80:80 brandonriddle828/getapp:latest
AND
docker run -d --name postapp --network get-post-network -p8001:80 brandonriddle828/postapp:latest

```
[root@ip-172-31-1-219 ec2-user]# docker run -d --name getapp --network get-post-network
 -p80:80 brandonriddle828/getapp:latest
d48ea1f7231db670f0d8938ad5686bef2f43741eb0dc66430585e8b134c29e3f
[root@ip-172-31-1-219 ec2-user]# docker run -d --name postapp --network get-post-network -p8001:80
brandonriddle828/postapp:latest
24dcdc487b6dc3057accc4f2841757ce70247019c3a61075b22ee1e93d0d2373
```

28. Now, to make sure it all worked, you can copy and paste that Elastic IP address into your address bar and press enter! If you do not remember what the IP is, then at the bottom of the instance, it will tell you what your public IP address is

NOTE: If this is not working and you need to change the source code in your project, there are a few commands that you need to know.

- First you need to stop the image with: **docker stop {appname}**
- Second you need to remove the image with: **docker rm {appname}**
- Lastly, pull the update down with: **docker pull {user/appname}**

```
[root@ip-172-31-27-142 ec2-user]# docker stop postapp
postapp
[root@ip-172-31-27-142 ec2-user]# docker rm postapp
postapp
[root@ip-172-31-27-142 ec2-user]# docker pull brandonriddle828/postapp:latest
latest: Pulling from brandonriddle828/postapp
Digest: sha256:64c335b71192c5787b716d341d9094ae2b465ca71a6fc9caea9034d97b16536e
Status: Image is up to date for brandonriddle828/postapp:latest
docker.io/brandonriddle828/postapp:latest
[root@ip-172-31-27-142 ec2-user]#
```

After this you can rerun your image on the network and try it out!

Note: The container will continue running and that uses resources, resources that are limited due to AWS having a free version and if you go over the limits then you start paying. To stop this from happening, be sure to pause the container whenever you are done running it. You can also stop the container, but that gets rid of it entirely so pausing does the job just fine.

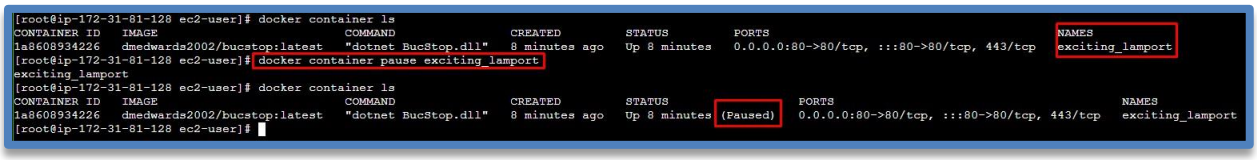
To do this, enter the command

docker container pause <container name>

for me it looks like

docker container pause exciting_lamport

AWS comes up with crazy names



That is how you utilize AWS and Docker to deploy your project. Hope this helps you in figuring it all!