Deployment with Docker and AWS

Updated Documentation Spring 2024

# Prerequisites

* Create a Docker account at [https://hub.docker.com](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

# A screenshot of a computer program Description automatically generatedPrep 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

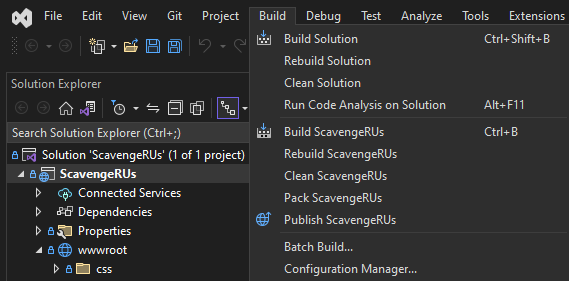
1. In Visual Studio, while in your project navigate to the top bar and click “Build” and then “Publish <Project name>”
   * If you do not see a publish with Docker option, make sure you open the project by double-clicking the *.sln* or *.csproj* file (opens project as a solution)

A screenshot of a computer

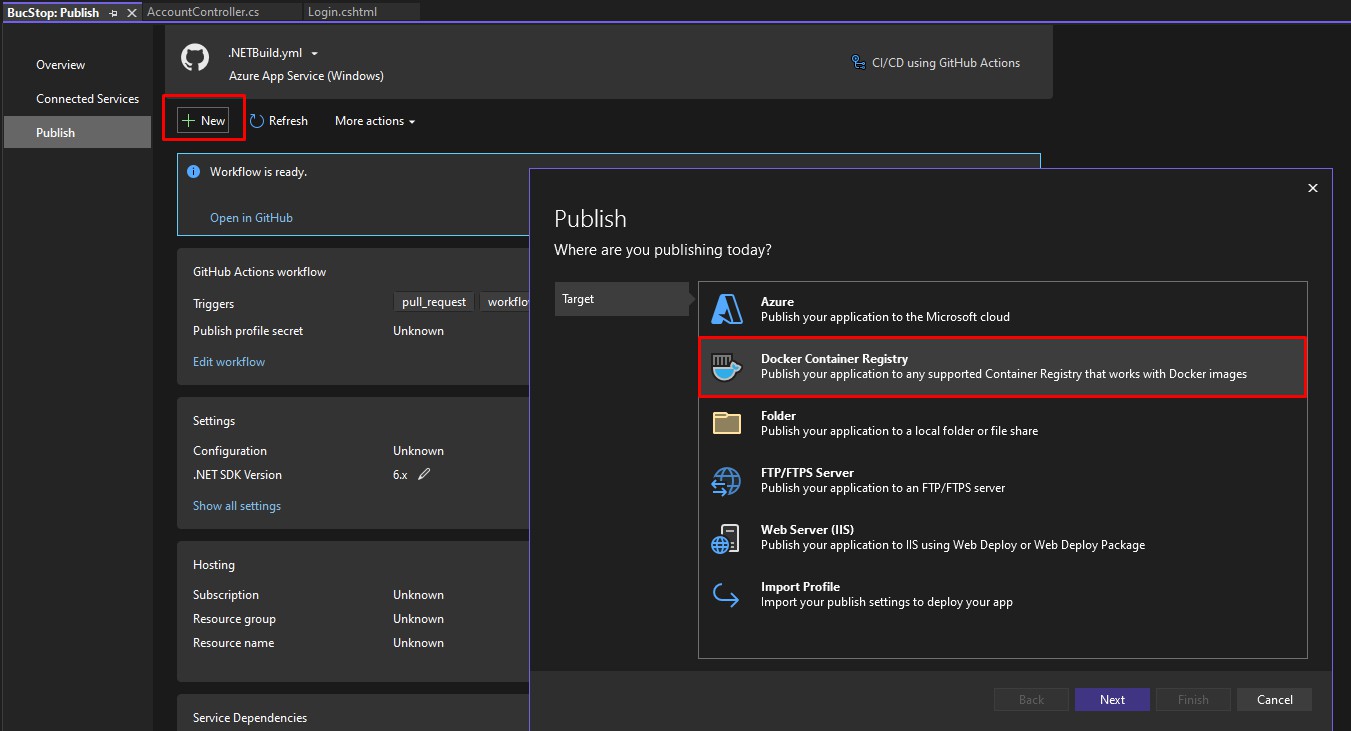
Description automatically generated

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

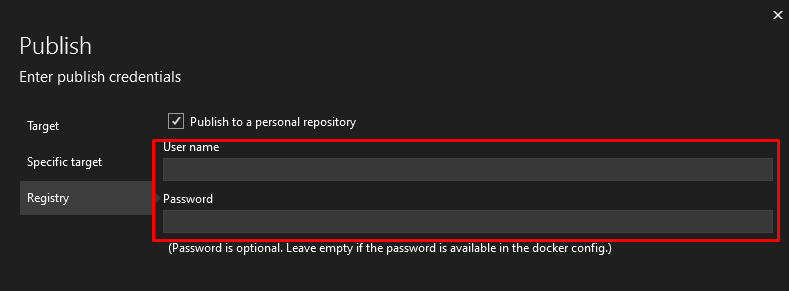


1. If this is the first time, you have to press ***New*** and then choose your target on where you want to publish, in this case it will be ***Docker***. Once you press next, you will have three options to choose from, choose the one that says ***Docker Hub***

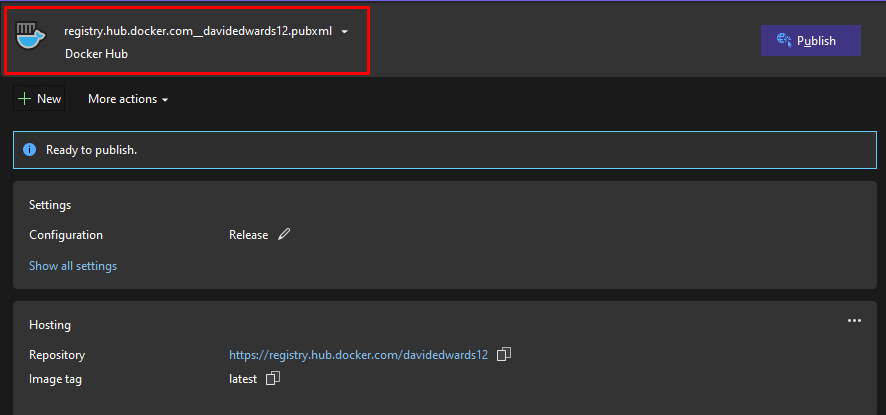


Note: You might have to install some packages in your VS if you do not have the required packages. If this is the case, go ahead and install them and then continue.

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

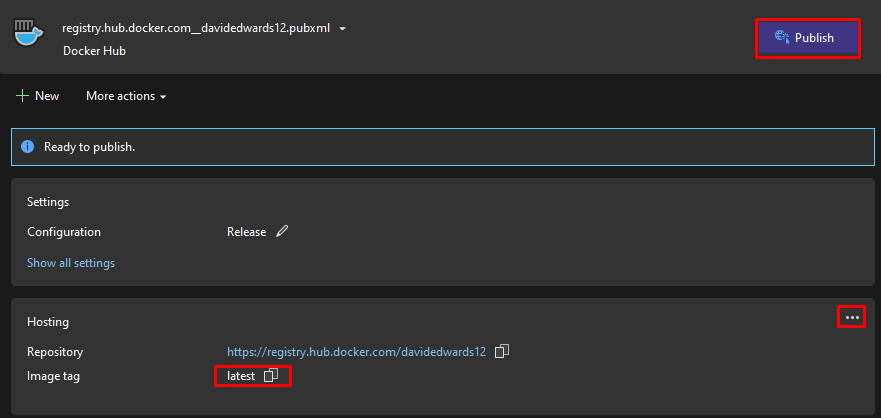


1. Back in the publish tab, you should see that you are now logged into your Docker Hub account and can publish to your Docker Hub

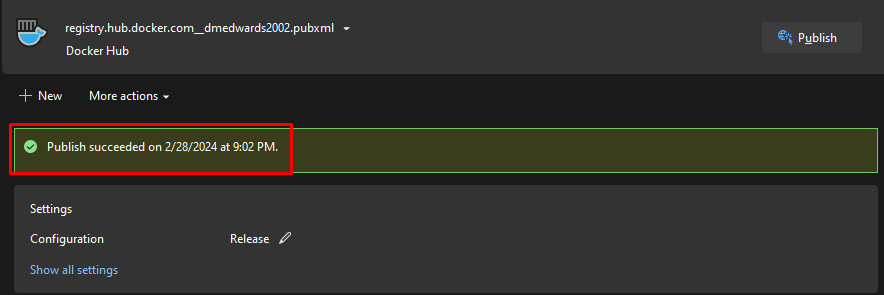


1. Now we can publish the project! Here, you can change the image tag by pressing the three little dots to the side (I think of the image tag as the version of the project, i.e.,

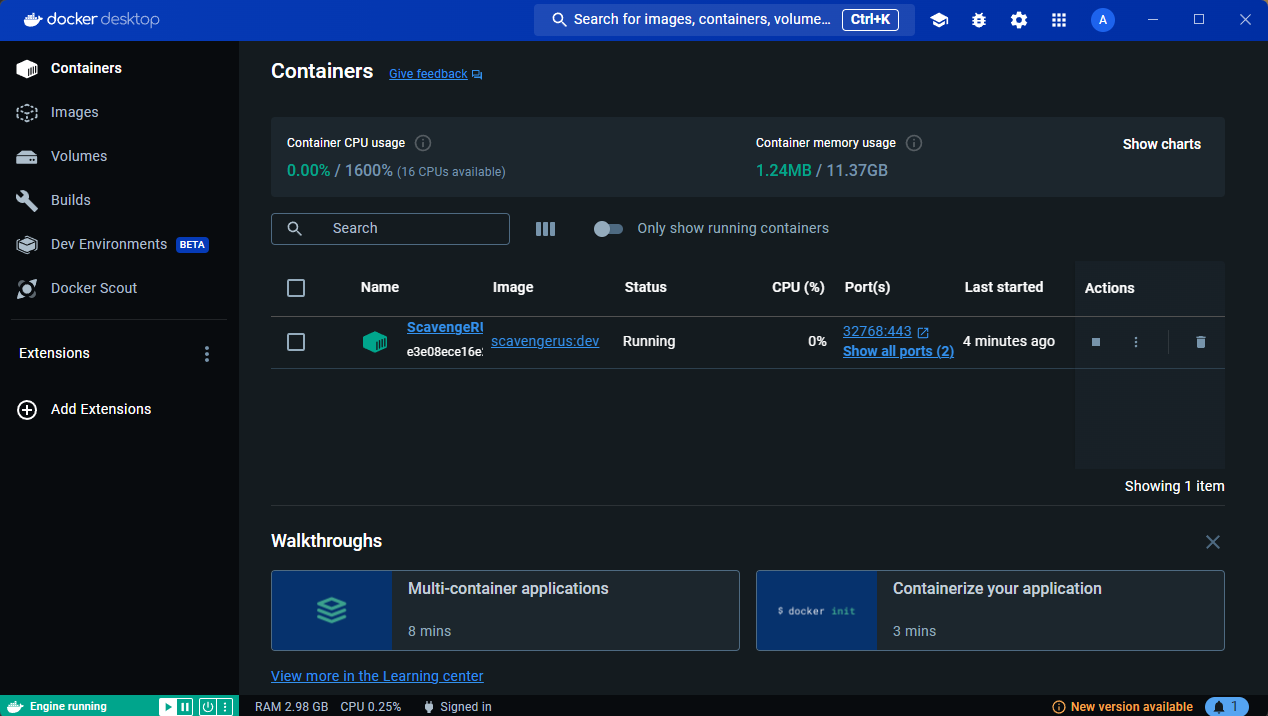
1.0.0, 1.0.1, etc..). You can leave it latest if you want to, just know that it replaces the last latest you had. Once you have the image tag set to whatever you want, you can now press the “Publish button” at the top (if you have not downloaded Docker Desktop by this point, you will need to now as VS requires it to publish)

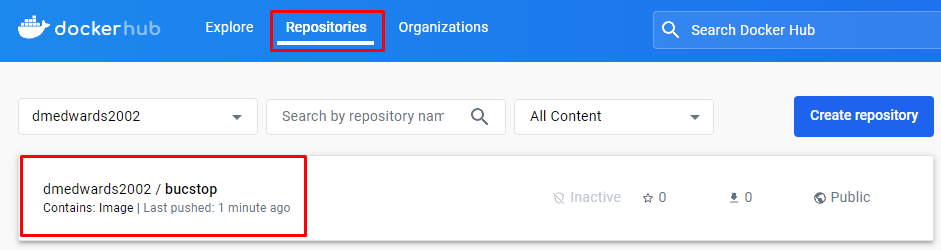


1. You will know you have published if you see this



1. 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



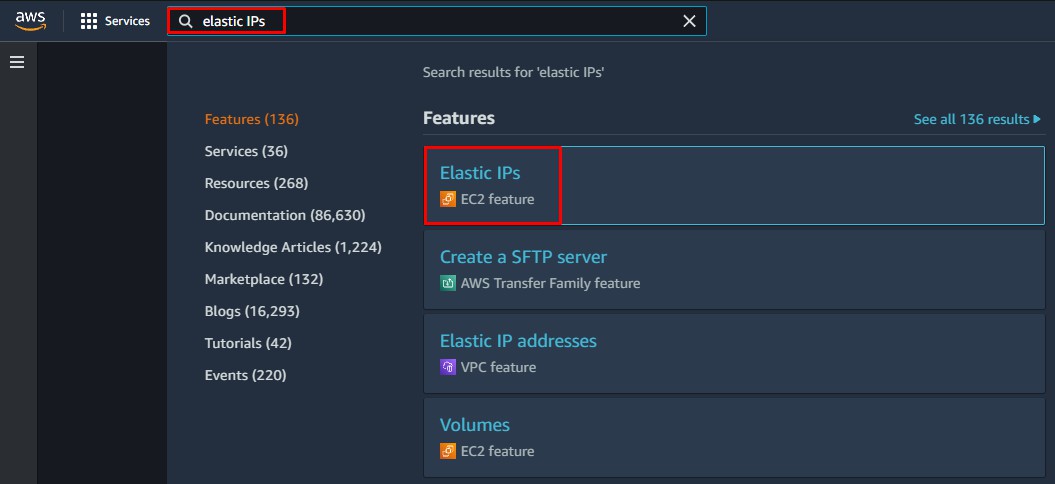


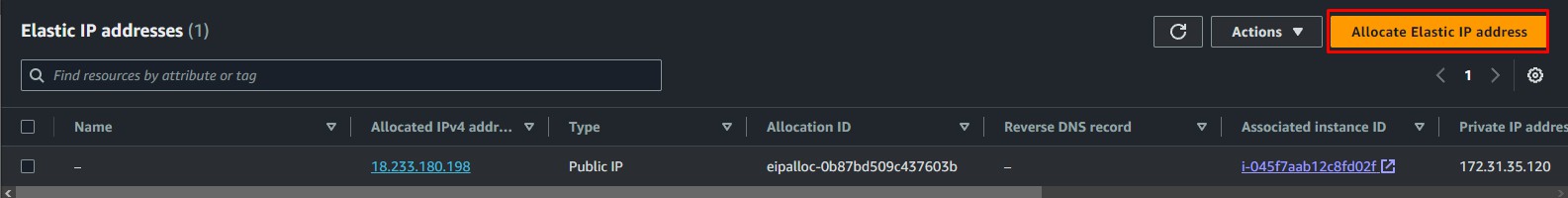
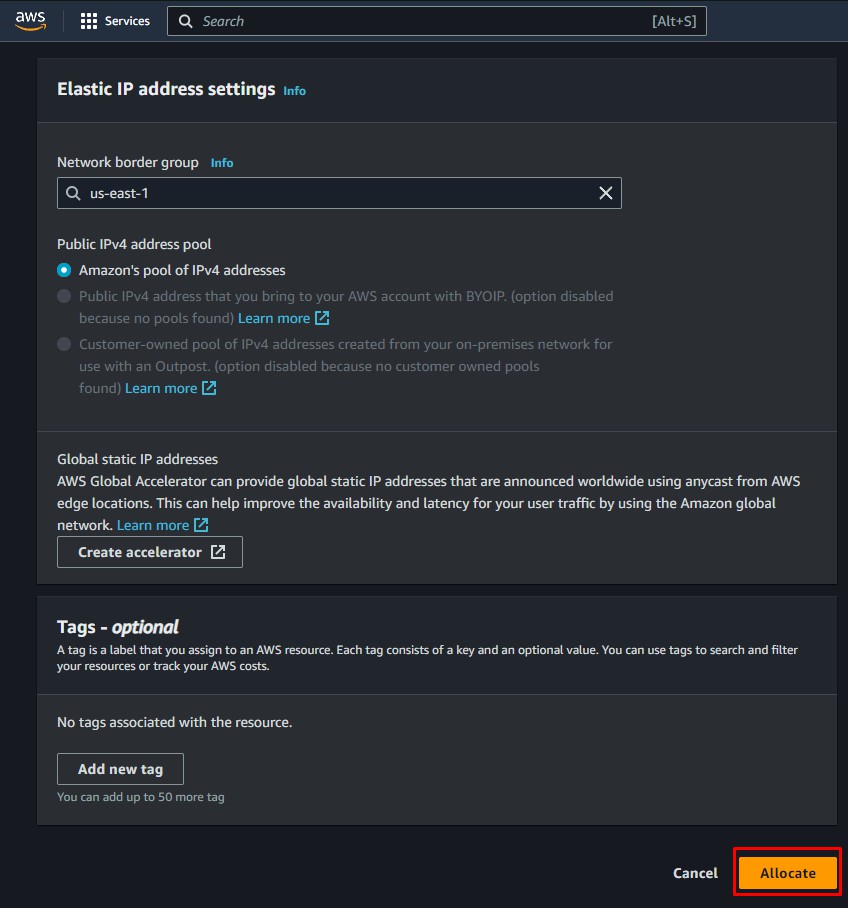
You can now deploy two different ways

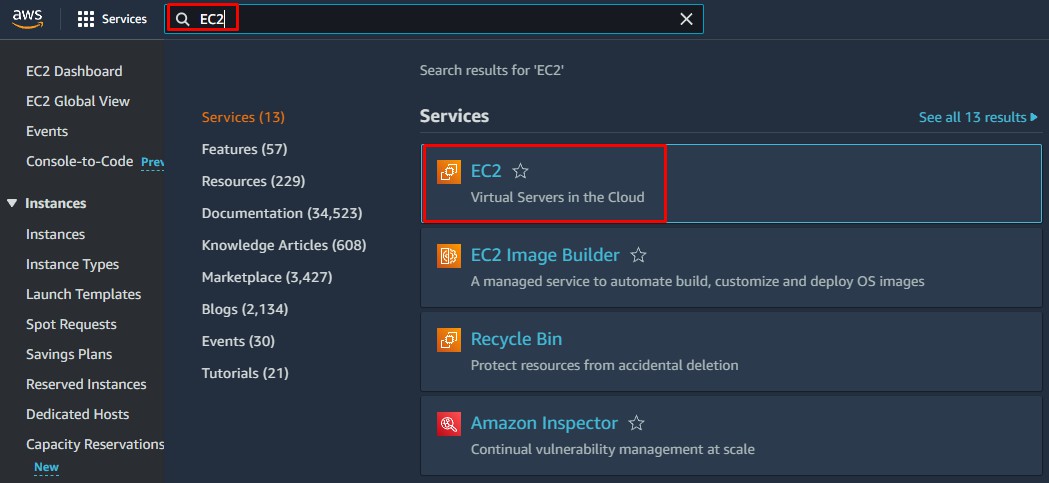
* Locally through the Docker Desktop application that we downloaded earlier
* Go through a public IP address with AWS. For this demo, we will be using AWS and an elastic IP address (more on that later).

# Setting Up AWS

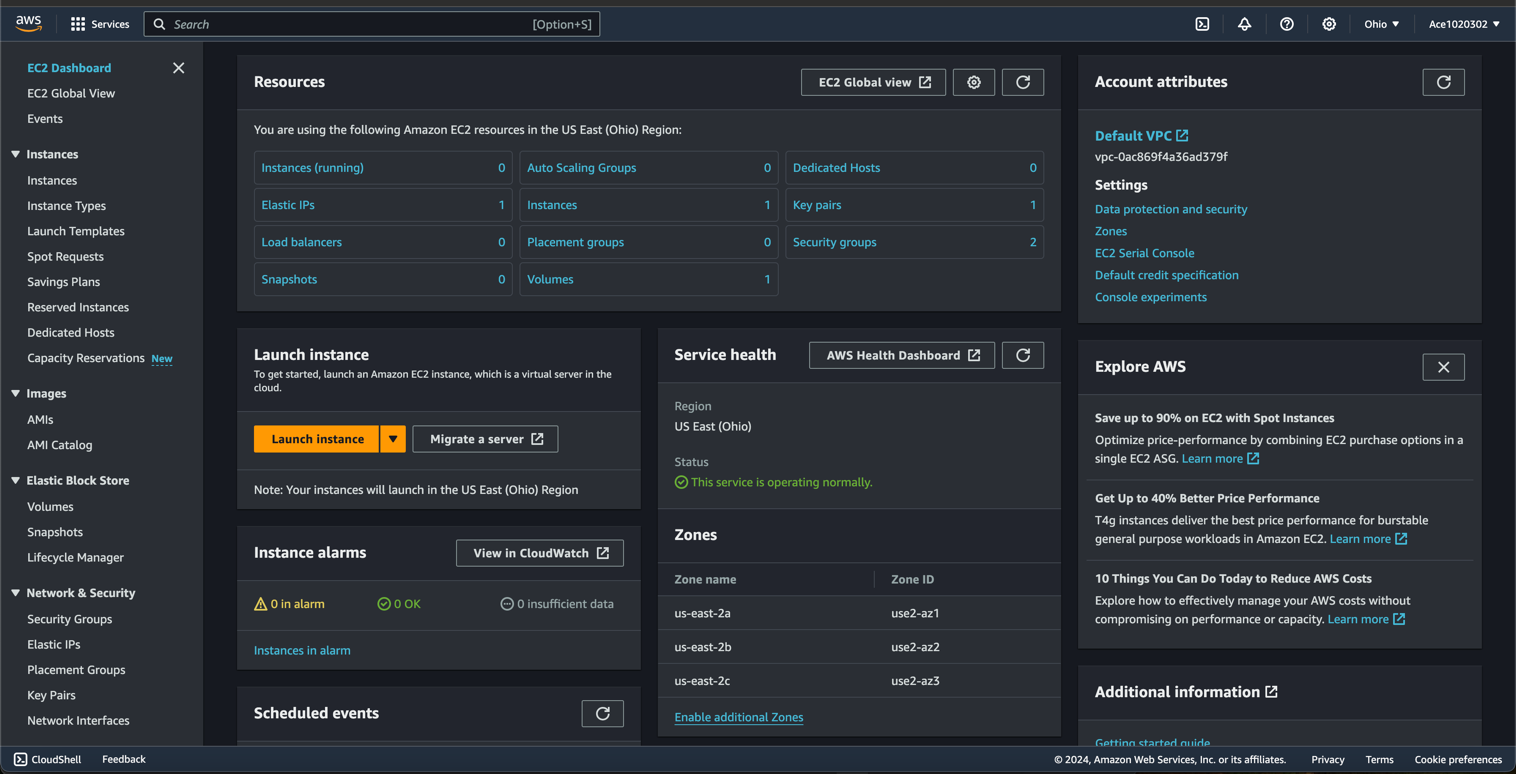
1. Assuming you have already made an AWS account, head over to the search bar at the top and search for “Elastic IPs”, we will use this later.



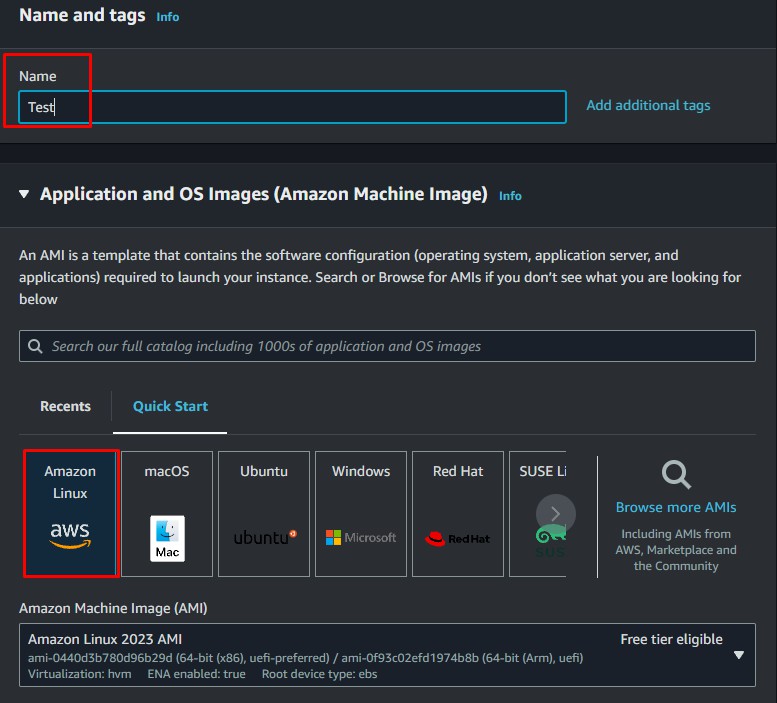
1. In the top right, click on **Allocate Elastic IP address**
2. I recommend leaving everything as it is and pressing **Allocate** at the button. You will be given an IP address to use when we set up our virtual server
3. Next, we search for “EC2.” This is where we set up our virtual server



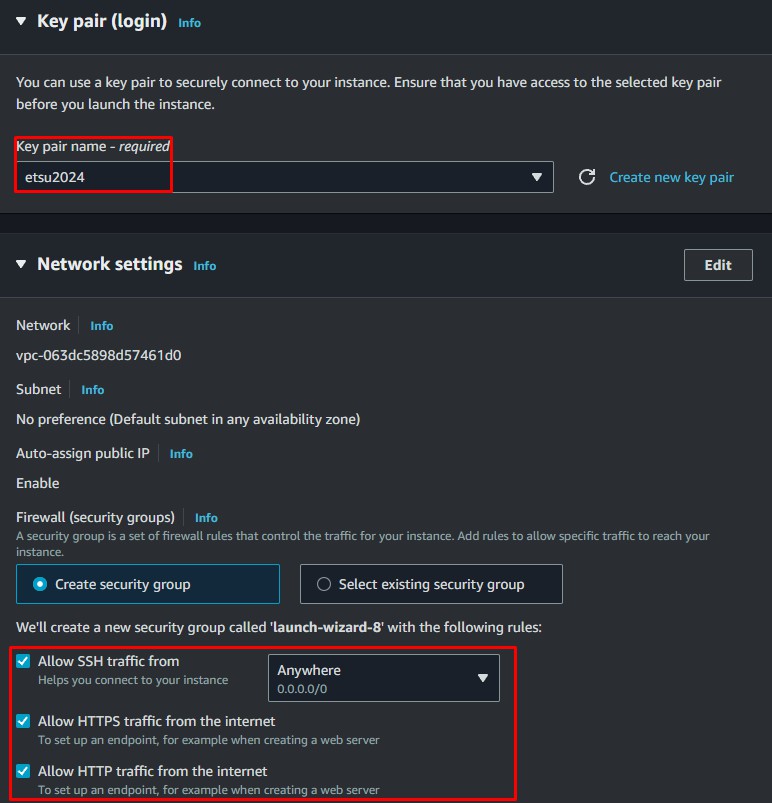
1. On that page, you will want to press **Launch instance**



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

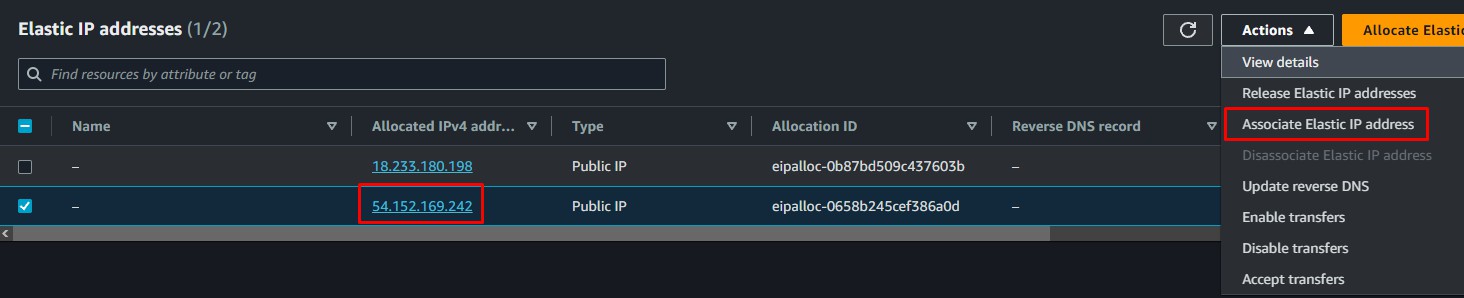


1. (Optional) Scrolling down, you can use a Key Pair to secure the instance more. It is not necessary but recommended. Then, in the network settings, check all the boxes. This will produce a certificate to download and hold on to.

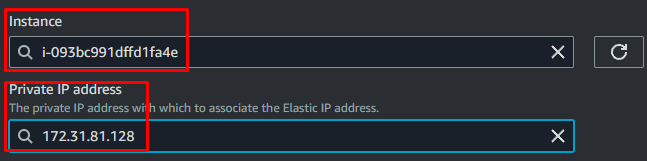


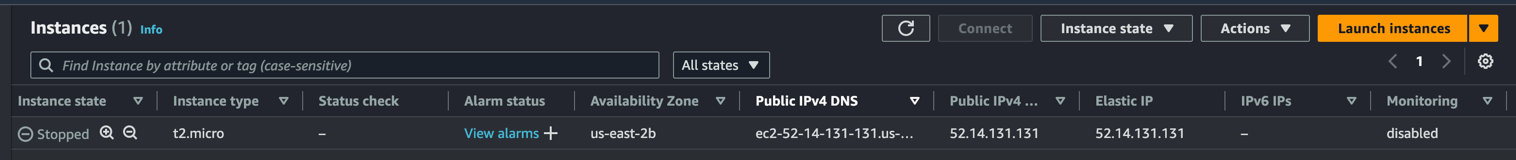
1. A screenshot of a computer

   Description automatically generatedFinally, press **Launch Instance** on the side.
2. 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”



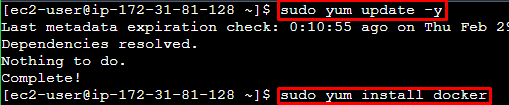
1. Choose the instance that you just created and select the provided private IP, then press Associate.



1. When you go back to your instances, you should now see that the Elastic IP is now associated with the instance that you selected
2. Connect to your instance with all the settings that are provided
3. 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

1. Run the command

sudo service docker start

To start the Docker service

1. Now run the command

sudo su

To get into the root account. You will run docker commands from this path

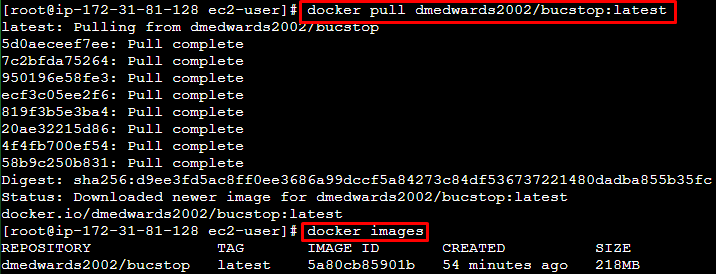
1. We now pull in our Docker image that we published using VS.

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 images you have in the instance

1. 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



1. (Optional) To make sure that the image is running run:

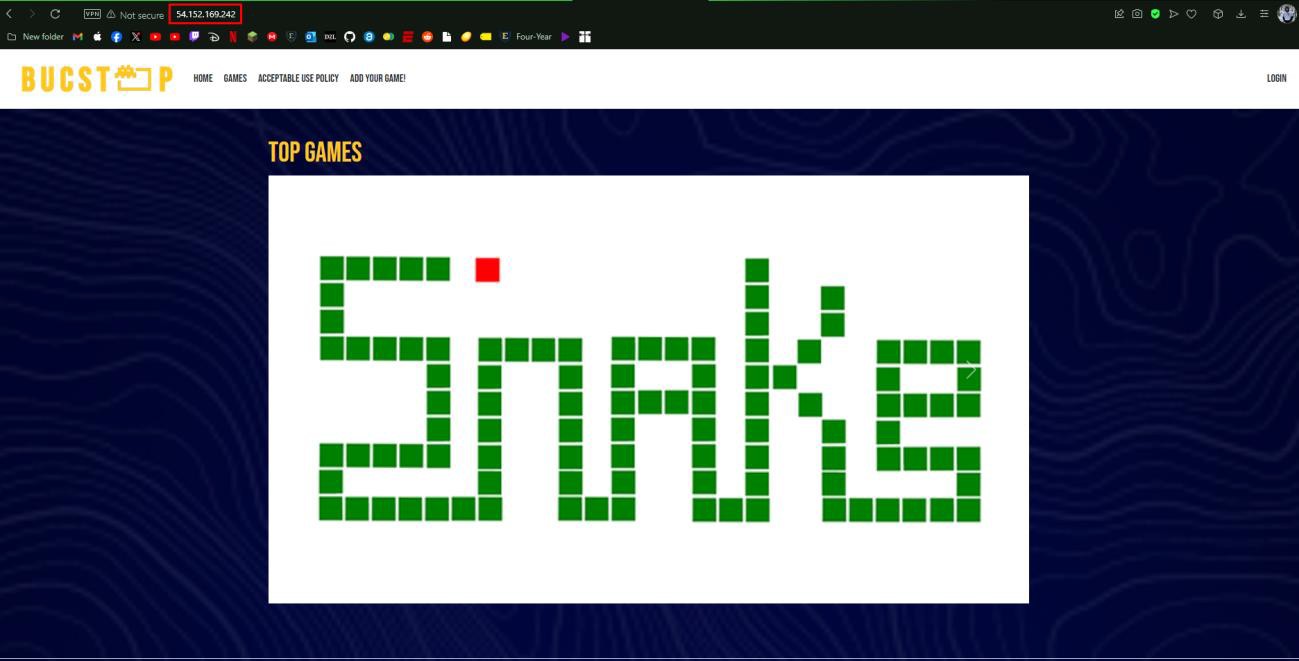
docker container ls

This will list all of your active containers

1. 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



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



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.

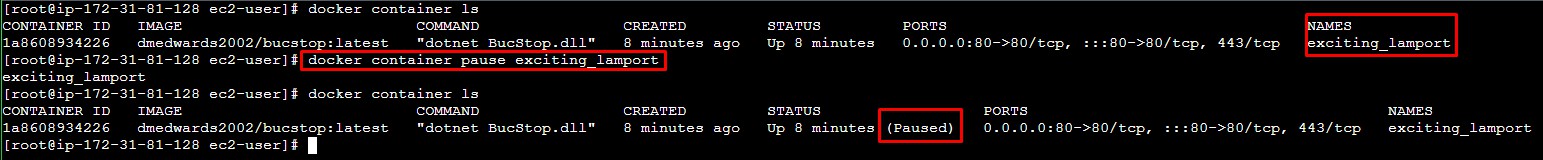
To do this, enter the command

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!