Containerize a .NET 8 Web API application with Docker and upload/download the image to/from Docker Hub

0. Prerequisites

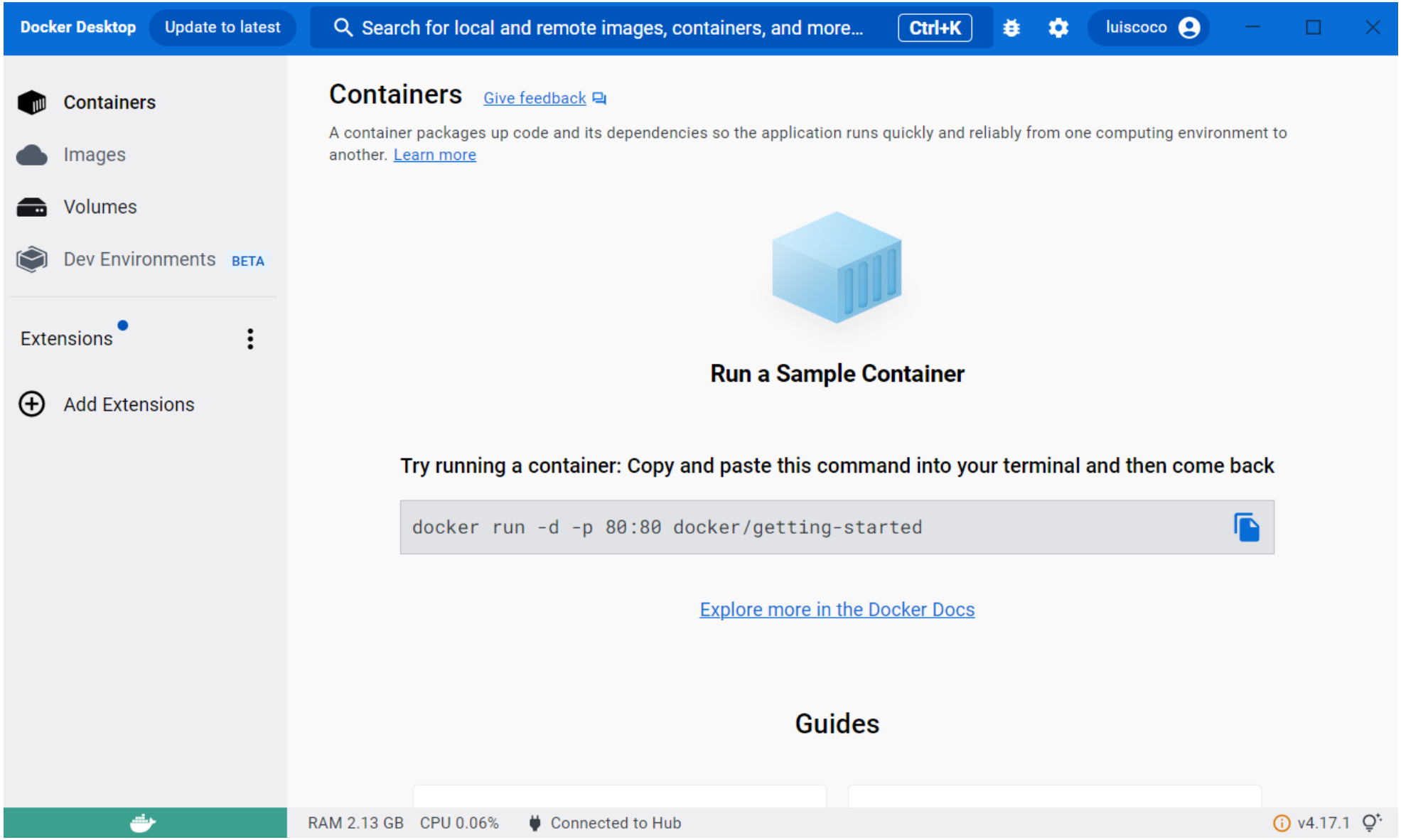
Install Docker Desktop

Create a new account in Docker Hub

Install Visual Studio 2022 Community edition

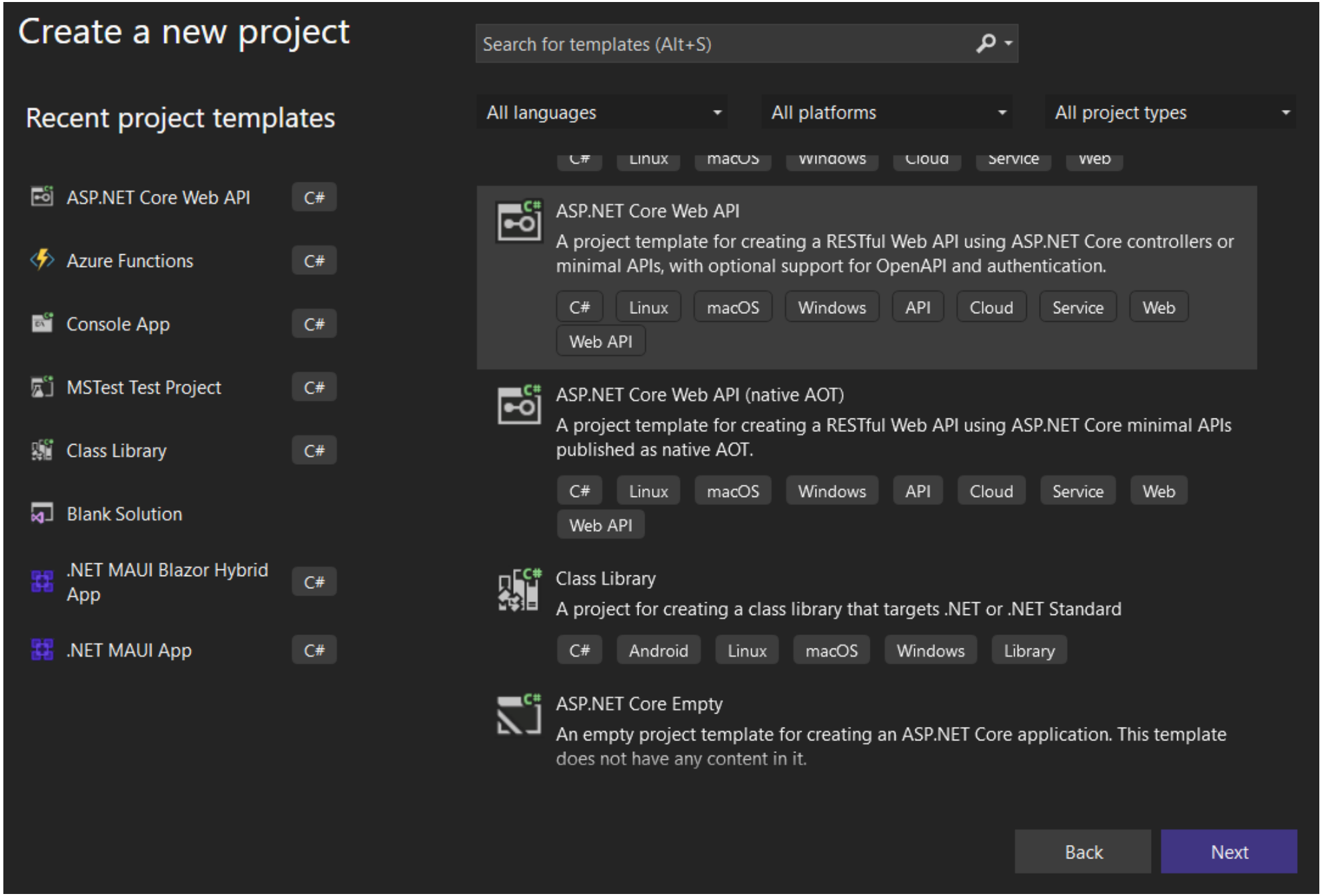
1. Create a .NET 8 Web API in Visual Studio 2022 community

IMPORTANT! Install and **Run Docker Desktop** before starting creating the application in Visual Studio 2022

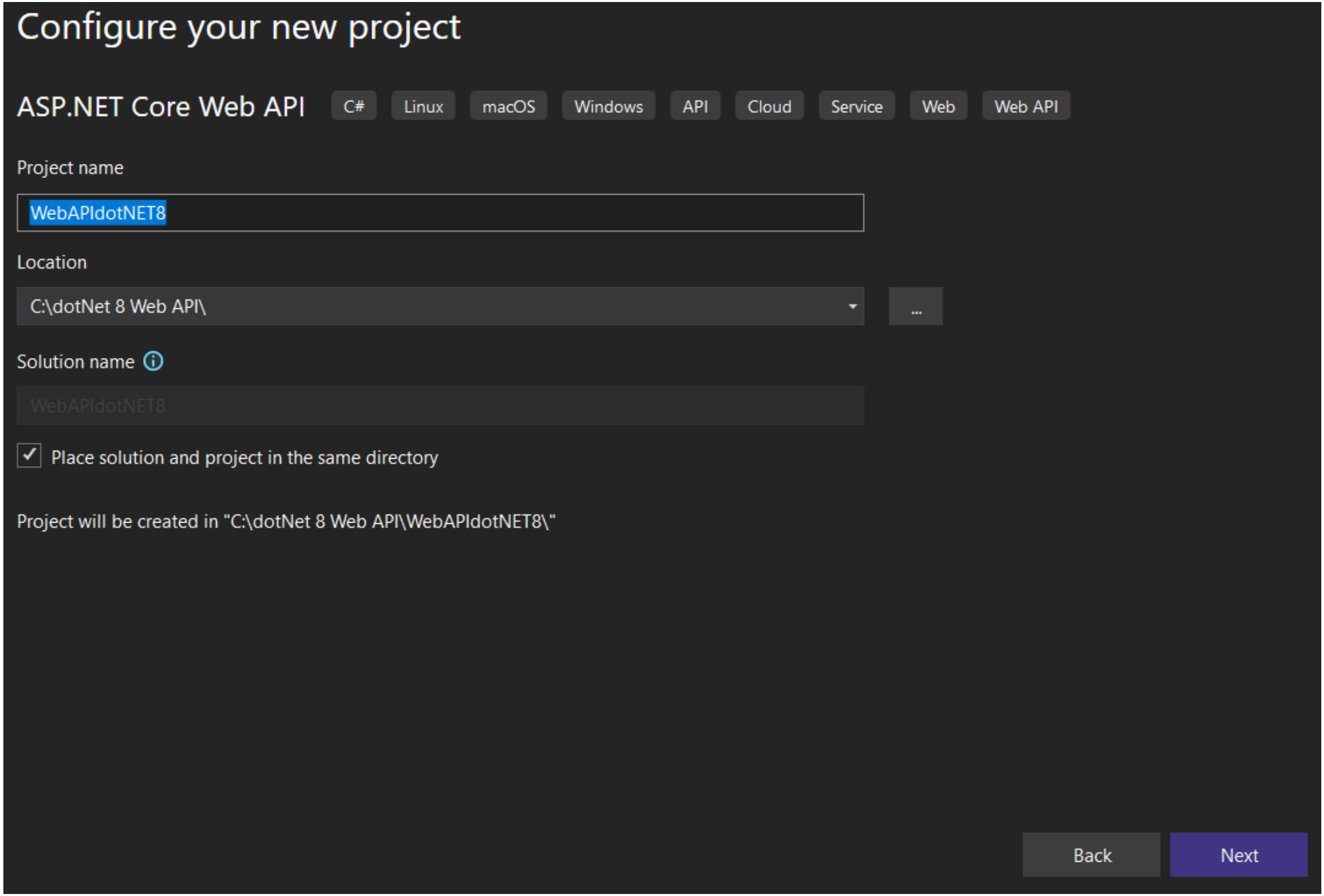


Run Visual Studio 2022 Community Edition and create a new project

Select the .NET Web API template



Set the project name and location



Select the .NET 8 framework, also select **Enable Docker** for creating a dockerfile automatically

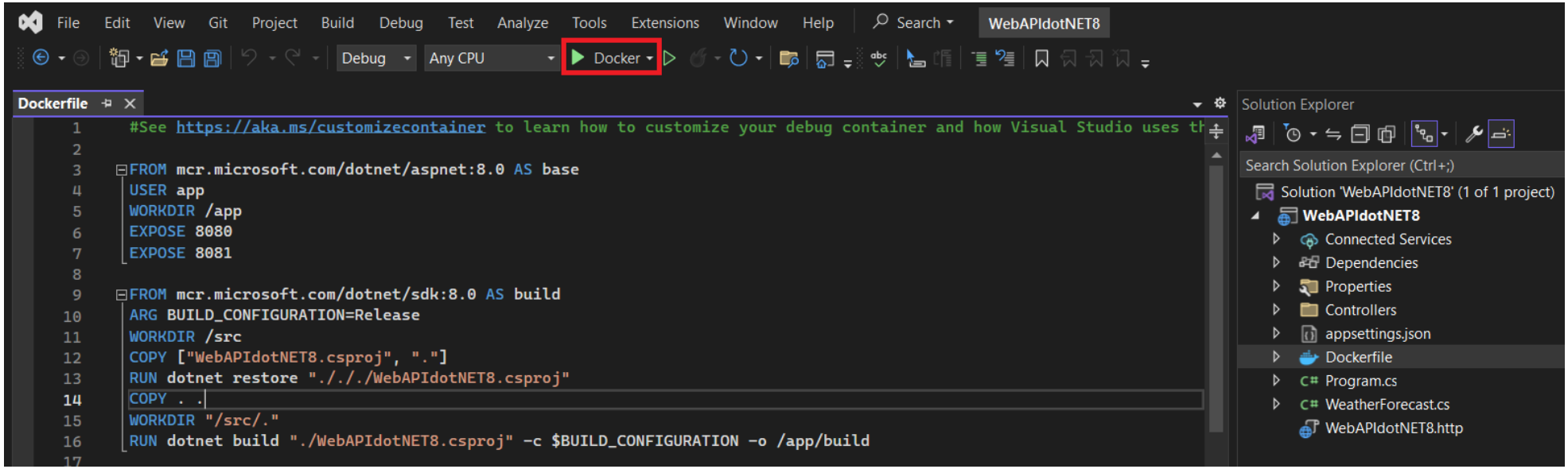
Click on the dockerfile to see the content

This is the dockefile source code

```
#See https://aka.ms/customizecontainer to learn how to customize your debug container and how Visual Studio uses this Dockerfile

FROM mcr.microsoft.com/dotnet/aspnet:8.0 AS base
USER app
WORKDIR /app
EXPOSE 8080
EXPOSE 8081

FROM mcr.microsoft.com/dotnet/sdk:8.0 AS build
ARG BUILD_CONFIGURATION=Release
WORKDIR /src
COPY ["WebAPIdotNET8.csproj", "."]
RUN dotnet restore "./././WebAPIdotNET8.csproj"
COPY . .
WORKDIR "/src/."
RUN dotnet build "./WebAPIdotNET8.csproj" -c $BUILD_CONFIGURATION -o /app/build

FROM build AS publish
ARG BUILD_CONFIGURATION=Release
RUN dotnet publish "./WebAPIdotNET8.csproj" -c $BUILD_CONFIGURATION -o /app/publish /p:UseAppHost=false

FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "WebAPIdotNET8.dll"]
```

2. Create and run the Docker image

We can automatically create and run the application Docker Image pressing the **Docker** button in Visual Studio 2022



See the docker image in Docker Desktop

See the docker running container in Docker Desktop

Also we can see the image with the command prompt command:

```
docker image
```

And we can see the running containers with the command:

```
docker ps
```

Finally see the Web API running container endpoints


Swagger UI
https://localhost:32768/swagger/index.html
Swagger
Supported by SMARTBEAR
Select a definition
WebAPIdotNET8 v1
WebAPIdotNET8
1.0
OAS3
https://localhost:32768/swagger/v1/swagger.json

Swagger UI localhost:32768/weatherforecast

https://localhost:32768/weatherforecast


```
[
    {
        "date": "2023-12-14",
        "temperatureC": 14,
        "temperatureF": 57,
        "summary": "Chilly"
    },
    {
        "date": "2023-12-15",
        "temperatureC": 9,
        "temperatureF": 48,
        "summary": "Bracing"
    },
    {
        "date": "2023-12-16",
        "temperatureC": 0,
        "temperatureF": 32,
        "summary": "Cool"
    },
    {
        "date": "2023-12-17",
        "temperatureC": 15,
        "temperatureF": 58,
        "summary": "Mild"
    },
    {
        "date": "2023-12-18",
        "temperatureC": 45,
        "temperatureF": 112,
        "summary": "Cool"
    }
]
```

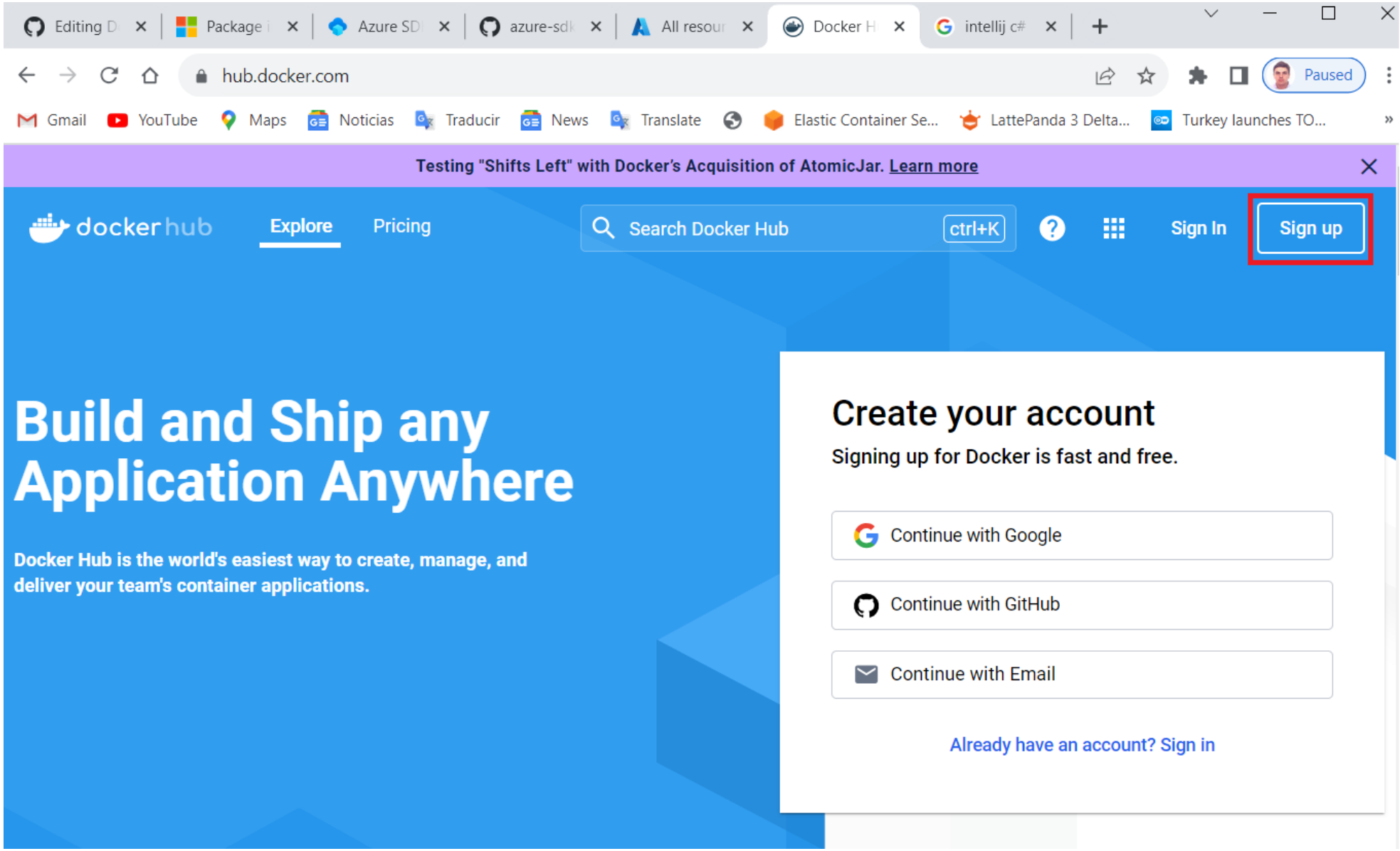

IMPORTANT NOTE:

Another option for creating automatically a dockerfile is to add "docker support" in our application

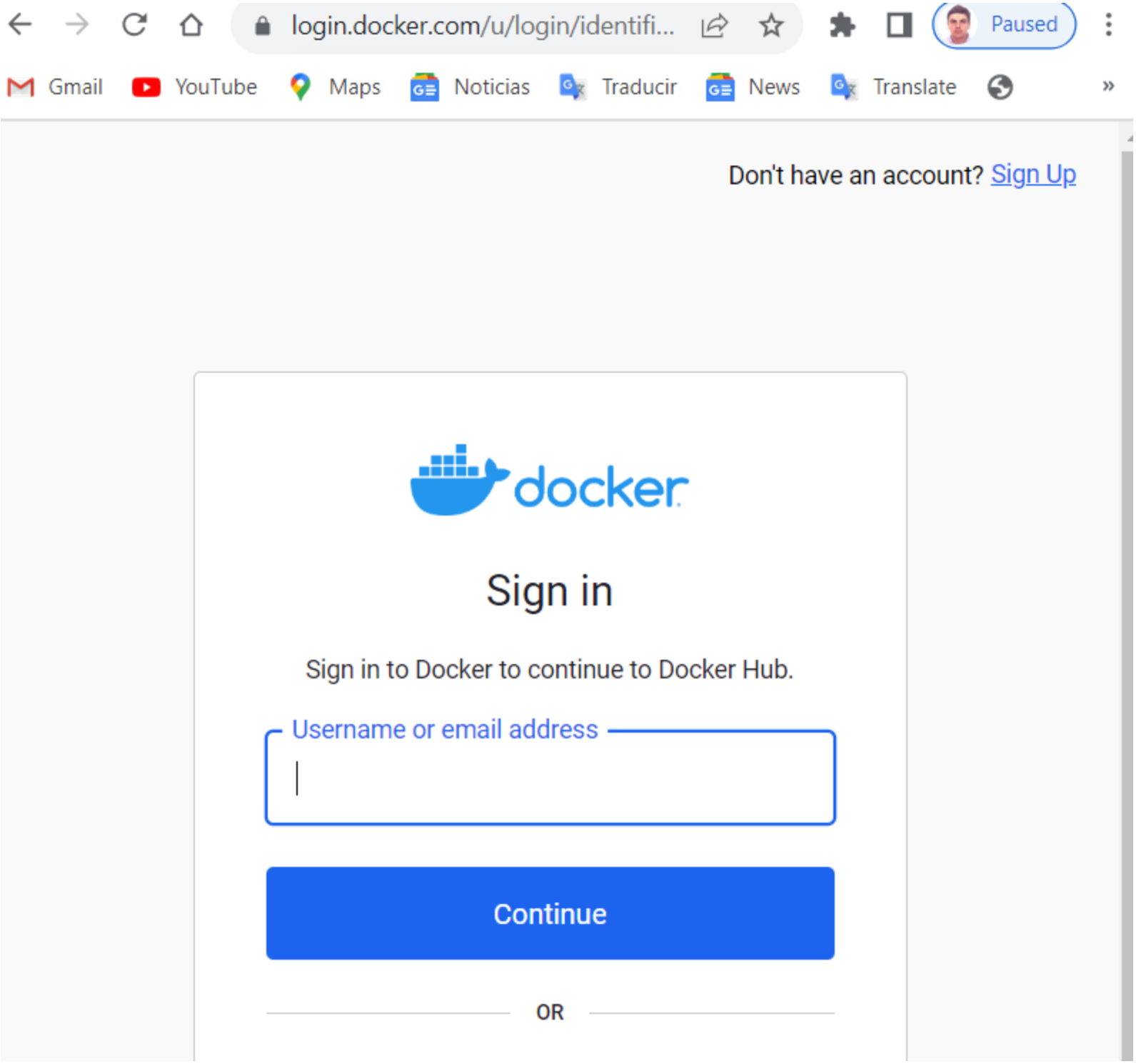
Then we select the docker virtual machine operating system, in our case we select "**Linux**"

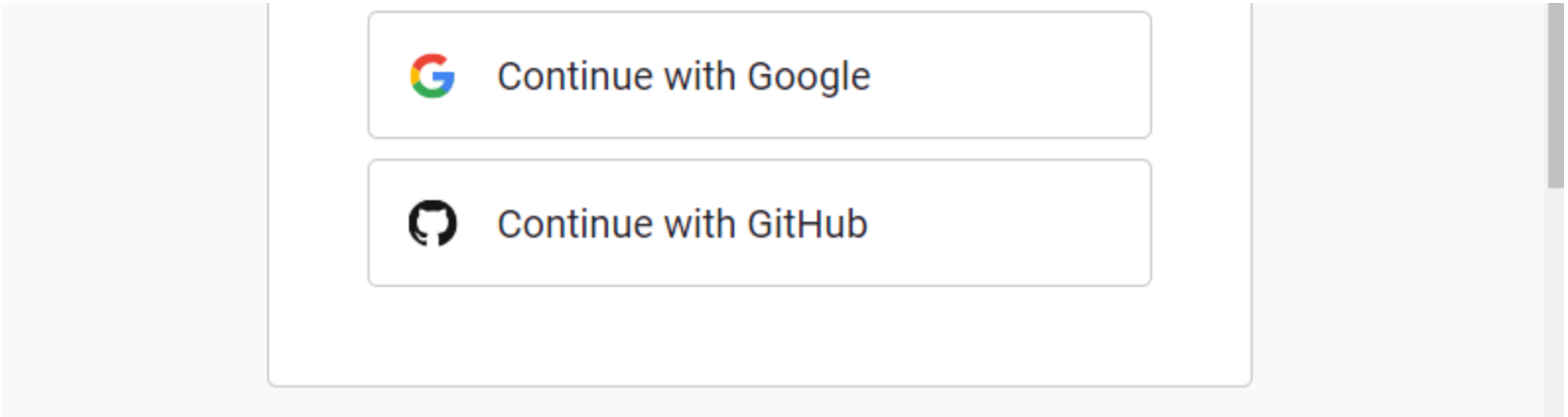
3. Upload/download the Docker image to/from the Docker Hub

Login in Docker Hub and create a new account (Sign Up)

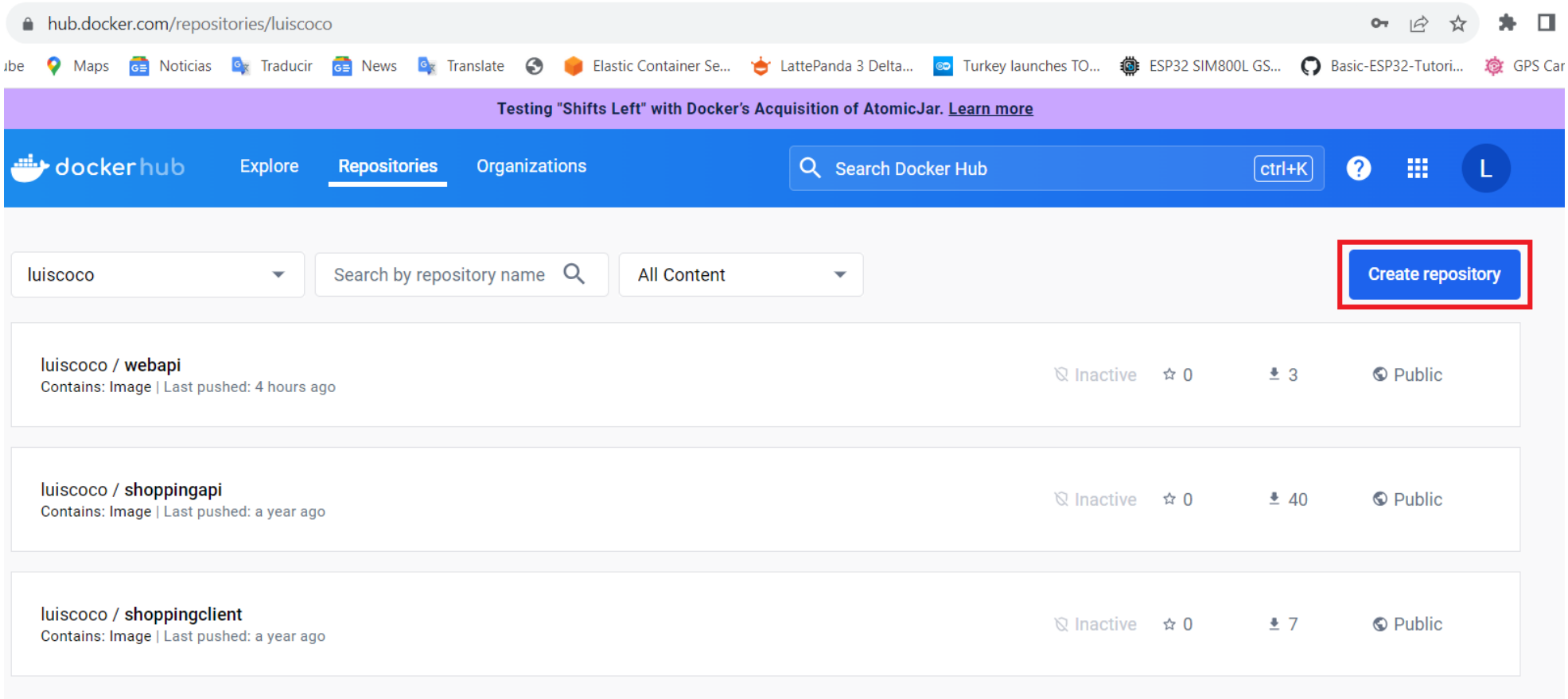


After creating an account we Sing in

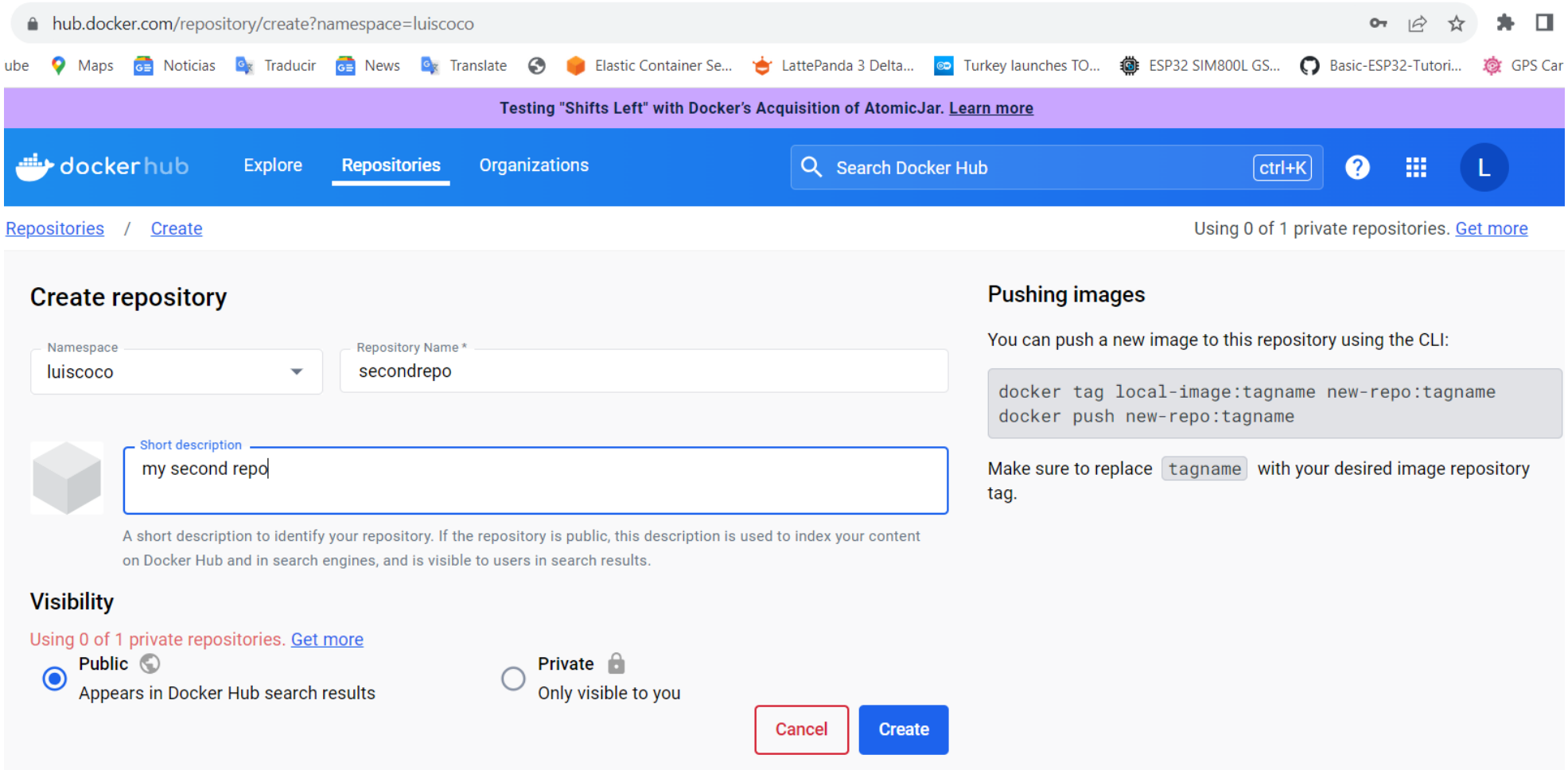

login.docker.com/u/login/identifi...
Paused
Gmail
YouTube
Maps
Noticias
Traducir
News
Translate
Don't have an account? Sign Up
docker
Sign in
Sign in to Docker to continue to Docker Hub.
Username or email address
Continue
OR



Press the "**Create Repository**" button



We set the new repo name, we select public or private repo and we include a repo description.



We can Push images to this repo with the following commands:

```
docker tag local-image:tagname new-repo:tagname
docker push new-repo:tagname
```

We can see the repositories list

4. Create the docker image and upload to the new repo in Docker Hub

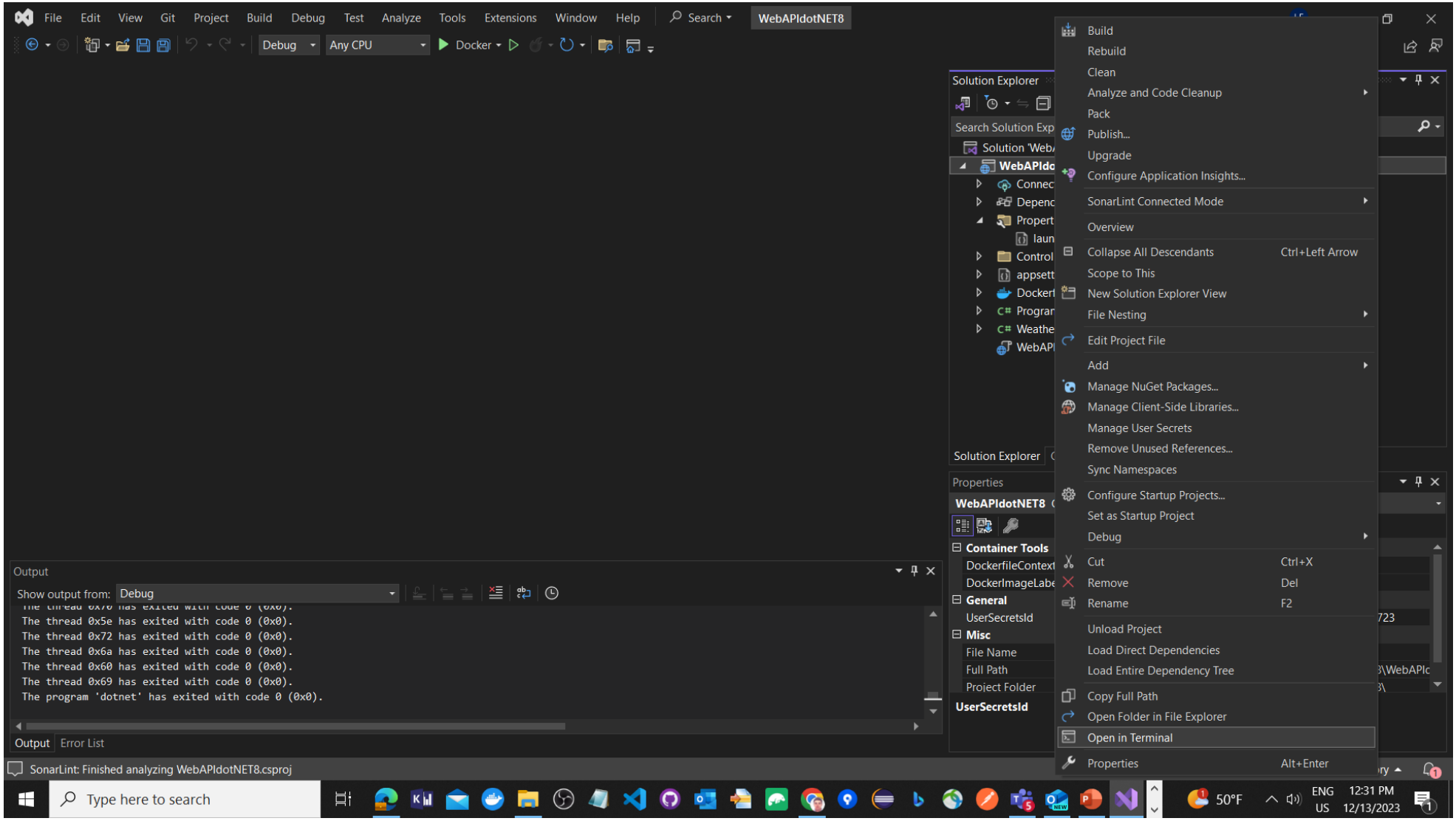
This is the general syntax for creating a new image that we would like to upload to the Docker Hub repo.

Pay attention we have to include the docker hub repo name then the docker image name and finally the docker image tag.

```
docker build -t dockerreponame/dockerimagename:tag .
```

For a real example, we right click on the project name and we select the menu option "**Open in Terminal**" then we run the following command:

```
docker build -t luiscoco/webapidotnet8:latest .
```


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search
WebAPIdotNET8
Debug
Any CPU
Docker
Solution Explorer
Search Solution Exp
Solution 'Weba
WebAPIdo
Connec
Depenc
Propert
laun
Control
appsett
Dockerf
Prograr
Weathe
WebAPI
Solution Explorer
Properties
WebAPIdotNET8
Container Tools
DockerfileContext
DockerImageLabe
General
UserSecretsId
Misc
File Name
Full Path
Project Folder
UserSecretsId
Build
Rebuild
Clean
Analyze and Code Cleanup
Pack
Publish...
Upgrade
Configure Application Insights...
SonarLint Connected Mode
Overview
Collapse All Descendants Ctrl+Left Arrow
Scope to This
New Solution Explorer View
File Nesting
Edit Project File
Add
Manage NuGet Packages...
Manage Client-Side Libraries...
Manage User Secrets
Remove Unused References...
Sync Namespaces
Configure Startup Projects...
Set as Startup Project
Debug
Cut Ctrl+X
Remove Del
Rename F2
Unload Project
Load Direct Dependencies
Load Entire Dependency Tree
Copy Full Path
Open Folder in File Explorer
Open in Terminal
Properties Alt+Enter
Output
Show output from: Debug
The thread 0x5e has exited with code 0 (0x0).
The thread 0x72 has exited with code 0 (0x0).
The thread 0x6a has exited with code 0 (0x0).
The thread 0x60 has exited with code 0 (0x0).
The thread 0x69 has exited with code 0 (0x0).
The program 'dotnet' has exited with code 0 (0x0).
Output Error List
SonarLint: Finished analyzing WebAPIdotNET8.csproj
Type here to search
50°F
ENG US
12:31 PM 12/13/2023

```
PS C:\dotNet 8 Web API\WebAPIdotNET8> docker build -t luiscoco/webapidotnet8:latest .
[+] Building 10.4s (12/17)
 => [internal] load build definition from Dockerfile                                                    0.0s
 => => transferring dockerfile: 882B                                                                    0.0s
 => [internal] load .dockerignore                                                                       0.0s
 => => transferring context: 464B                                                                       0.0s
 => [internal] load metadata for mcr.microsoft.com/dotnet/sdk:8.0                                       0.4s
 => [internal] load metadata for mcr.microsoft.com/dotnet/aspnet:8.0                                    0.3s
 => [build 1/7] FROM mcr.microsoft.com/dotnet/sdk:8.0@sha256:bb65e39b662be0265f780afae9cdbfcaa315ef63edb245ad9fb2aa1aabca0b6b    0.0s
 => [base 1/2] FROM mcr.microsoft.com/dotnet/aspnet:8.0@sha256:f4cb736b1220b1013e7a0f77e49487c5cc1fe643bf41a750306c4a1caa7faa97  0.0s
 => [internal] load build context                                                                       0.1s
 => => transferring context: 6.14kB                                                                     0.0s
 => CACHED [build 2/7] WORKDIR /src                                                                     0.0s
 => [build 3/7] COPY [WebAPIdotNET8.csproj, .]                                                          0.0s
```

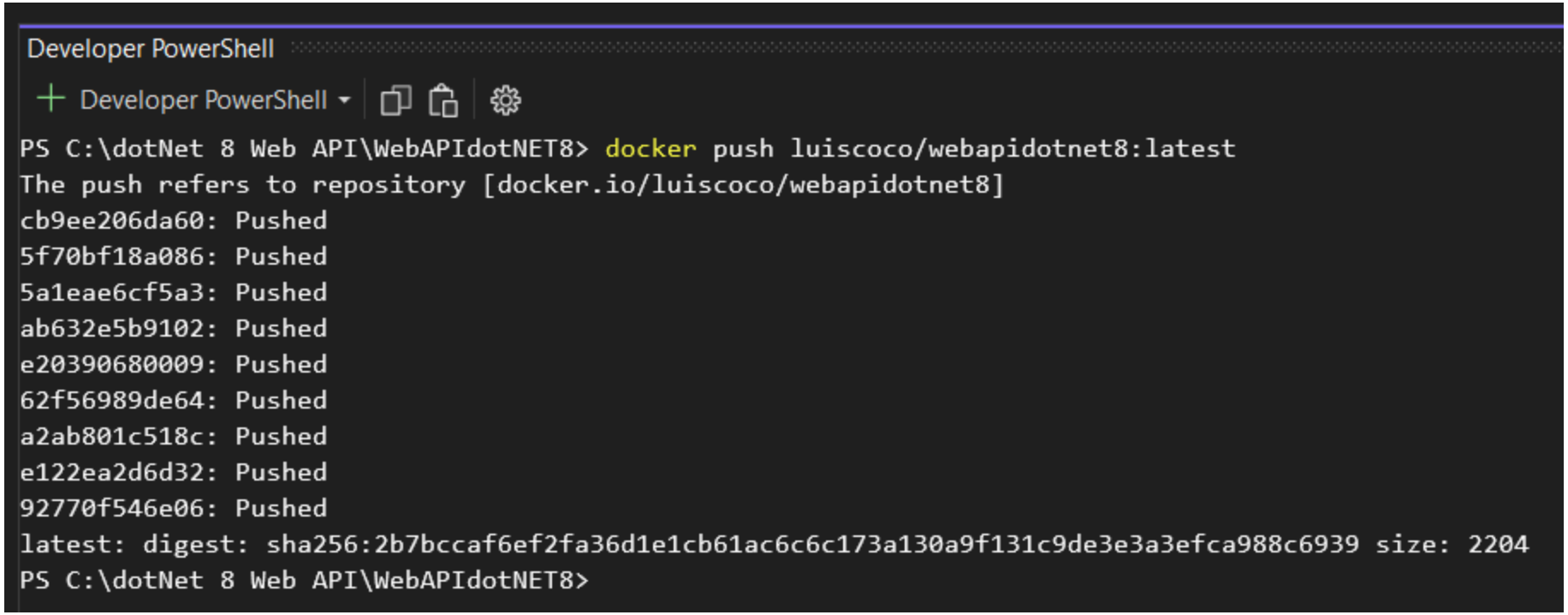
We check we create the docker image and we run the docker container

Then we use the docker push command to upload the image to the Docker Hub repository:

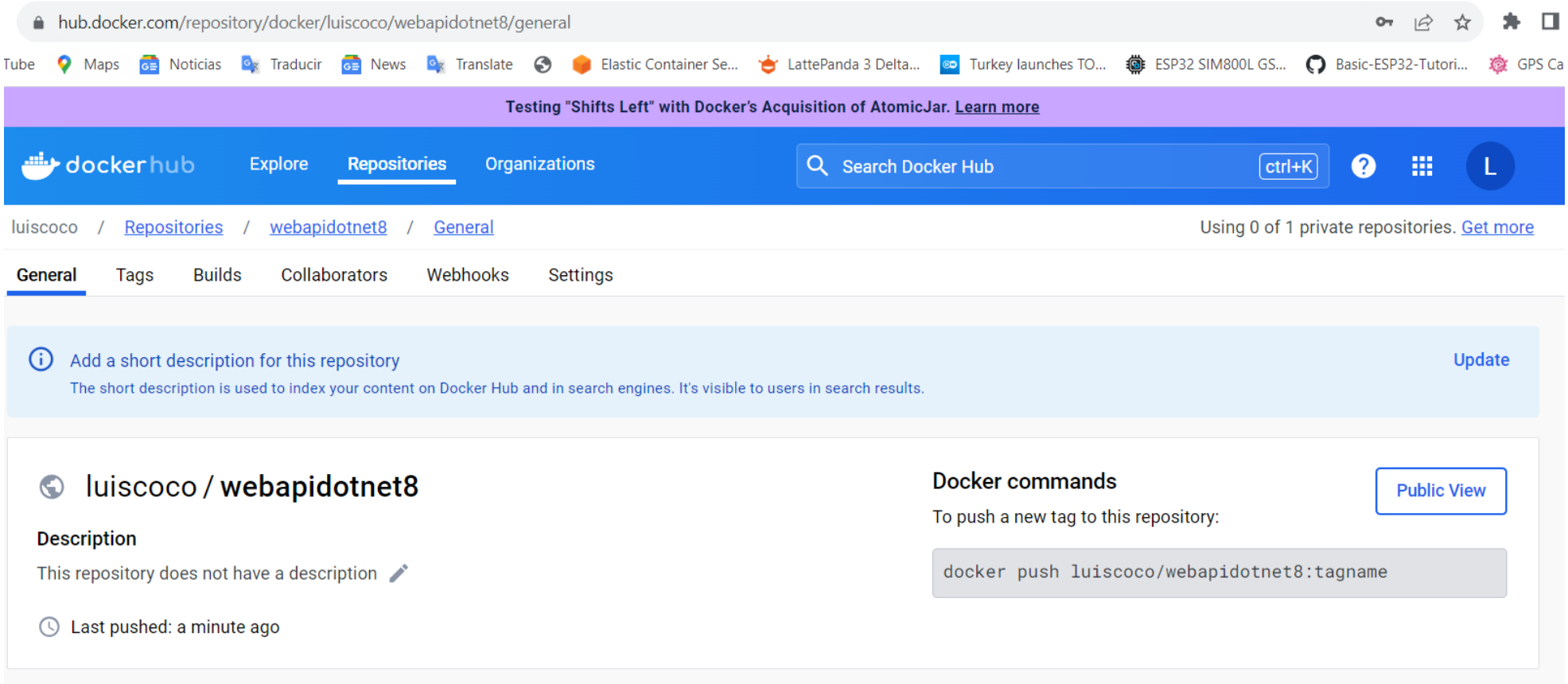
```
docker push luiscoco/webapidotnet8:latest
```

Also we can pull the docker image from the Docker Hub repository with the following command:

```
docker pull luiscoco/webapidotnet8
```



See the new docker image in the repo



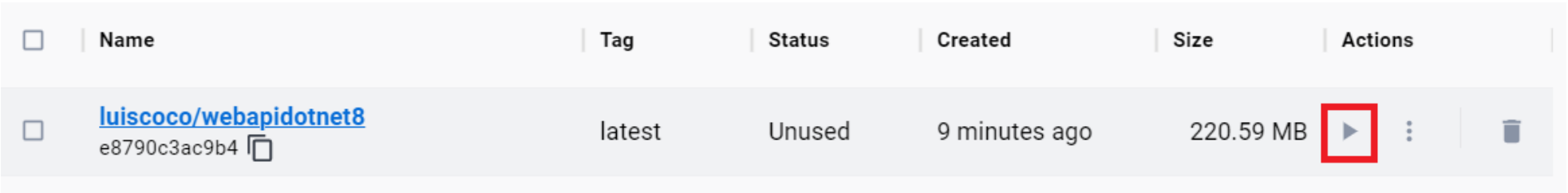
5. We pull the docker image from Docker hub and we run in Docker Desktop

We pull the image with the command:

```
docker pull luiscoco/webapidotnet8
```

Then we open Docker Desktop and we confirm the docker image was downloaded but it is not yet running in a container

To run the docker image in a container, we press the run button (see the following image), or we type this command



Or we can type the following command, taking into account we map the launchSetting.json http port with the 8080 port in dockerfile

```
docker run -d -p 5269:8080 luiscoco/webapidotnet8:latest
```

And also we can see the docker logs, see this picture

```
Developer PowerShell
+ Developer PowerShell
PS C:\dotNet 8 Web API\WebAPIdotNET8> docker run -d -p 5269:8080 luiscoco/webapidotnet8:latest
1910b41deb65d7f032a08a27396a914b36f77db98756f83e6aa8f67bceb8931e
PS C:\dotNet 8 Web API\WebAPIdotNET8> docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED         STATUS         PORTS                              NAMES
1910b41deb65   luiscoco/webapidotnet8:latest  "dotnet WebAPIdotNET…"   3 seconds ago   Up 3 seconds   8081/tcp, 0.0.0.0:5269->8080/tcp   festive_khorana
PS C:\dotNet 8 Web API\WebAPIdotNET8> docker logs festive_khorana
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://[::]:8080
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Production
info: Microsoft.Hosting.Lifetime[0]
      Content root path: /app
PS C:\dotNet 8 Web API\WebAPIdotNET8>
```

Now in Docker Desktop we can see the docker container is running

We can also navigate to the Web API endpoints:

http://localhost:5269/weatherforecast

Work items - Boards

localhost:5269/weatherforecast


```
[
    {
        "date": "2023-12-14",
        "temperatureC": 5,
        "temperatureF": 40,
        "summary": "Freezing"
    },
    {
        "date": "2023-12-15",
        "temperatureC": 30,
        "temperatureF": 85,
        "summary": "Hot"
    },
    {
        "date": "2023-12-16",
        "temperatureC": 26,
        "temperatureF": 78,
        "summary": "Freezing"
    },
    {
        "date": "2023-12-17",
        "temperatureC": -8,
        "temperatureF": 18,
        "summary": "Sweltering"
    },
    {
        "date": "2023-12-18",
        "temperatureC": 53,
        "temperatureF": 127,
        "summary": "Bracing"
    }
]
```