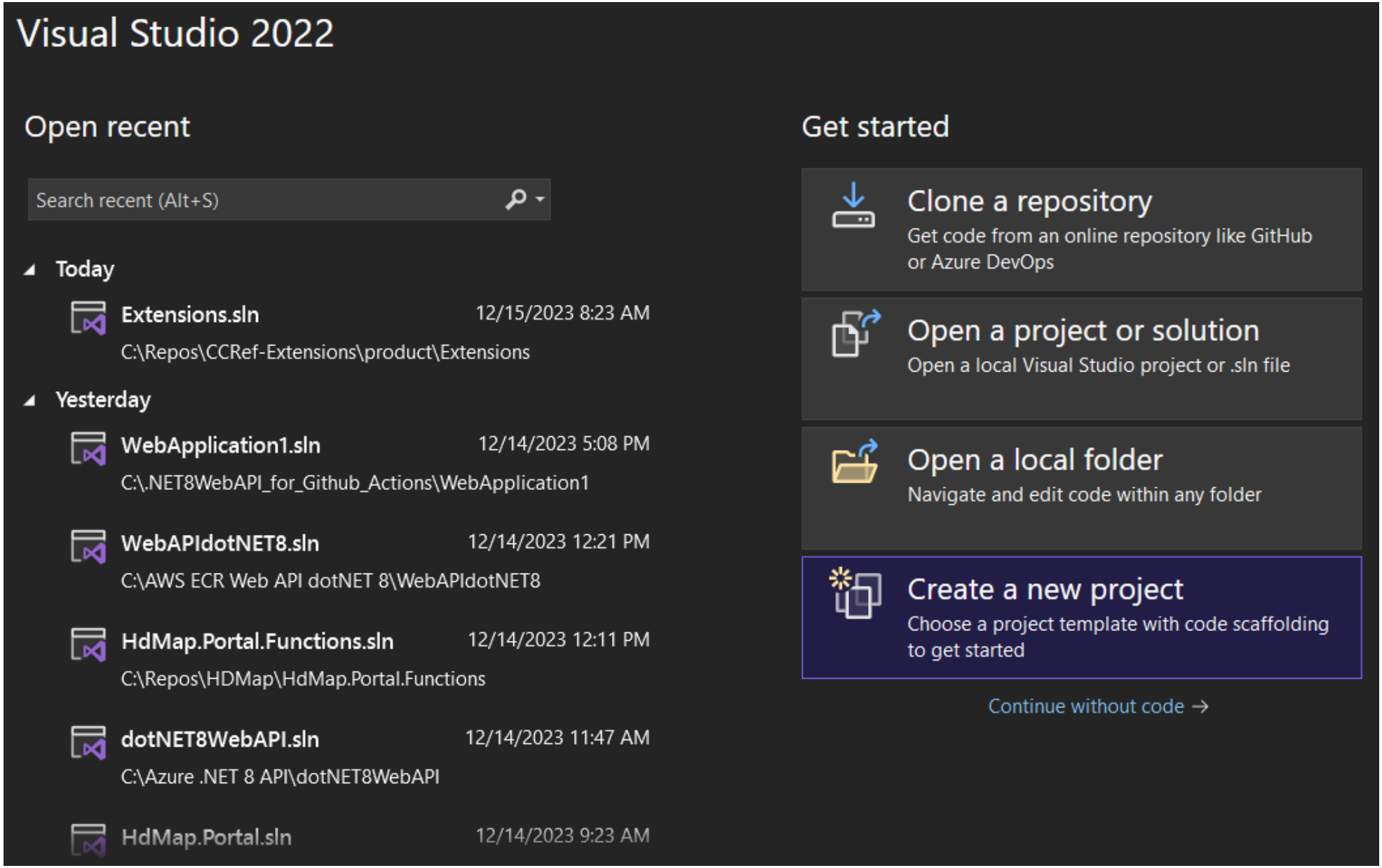
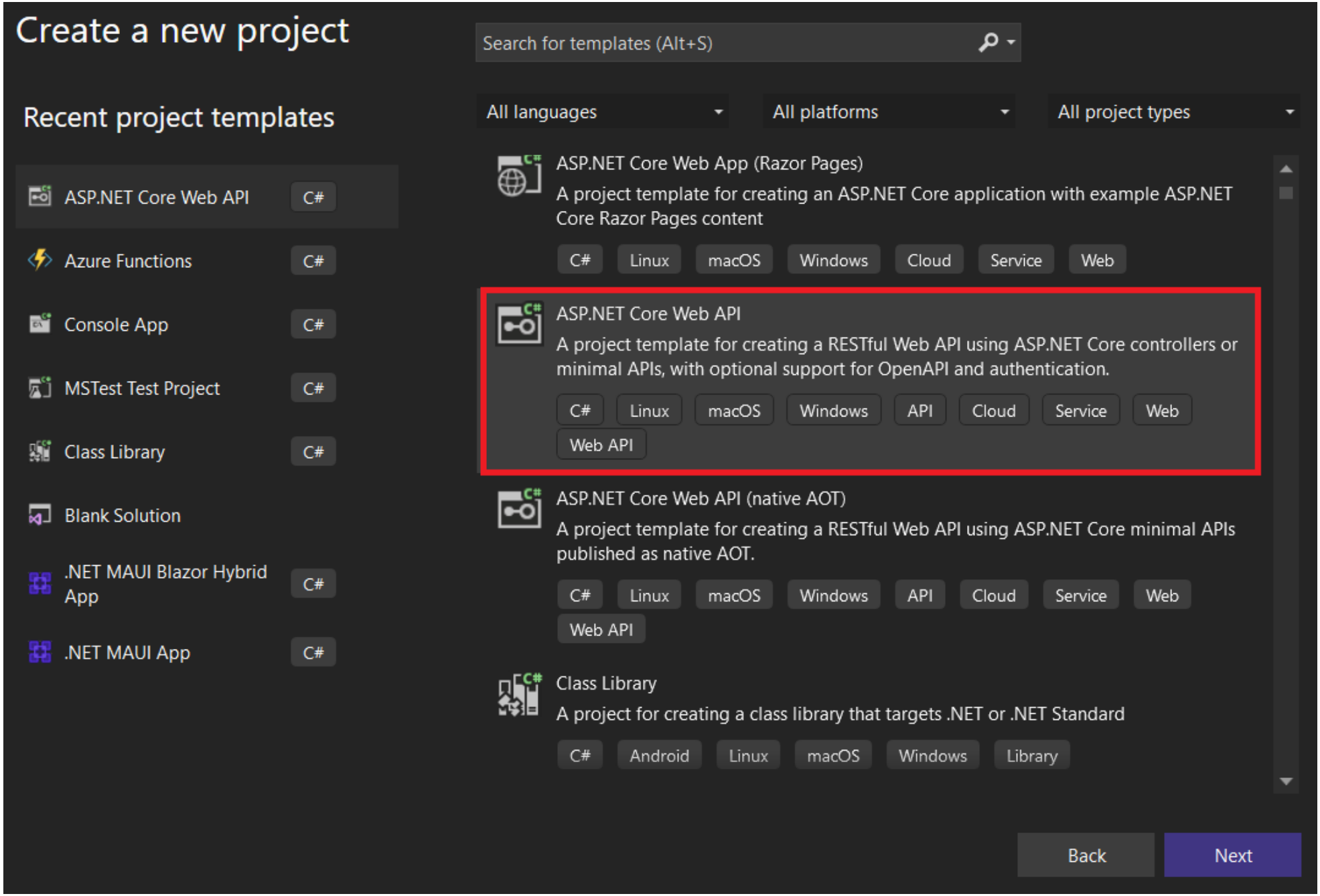
GithubActions: How to create .NET 8 Web API Docker image and Upload it to Azure Container Registry (ACR)

1. Create a .NET 8 Web API in Visual Studio Community Edition

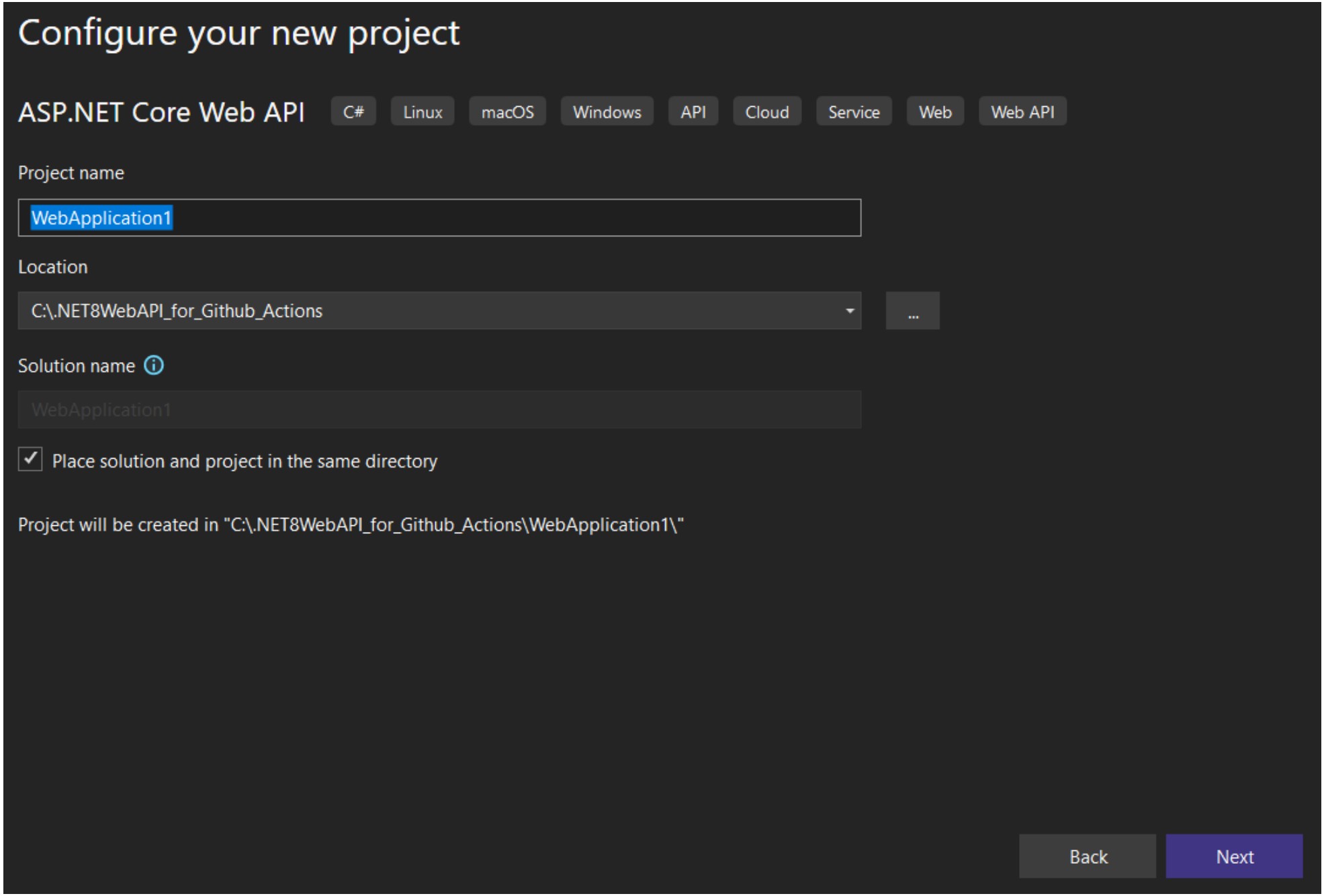
We run Visual Studio 2022 and we select the menu option "Create a new Project"



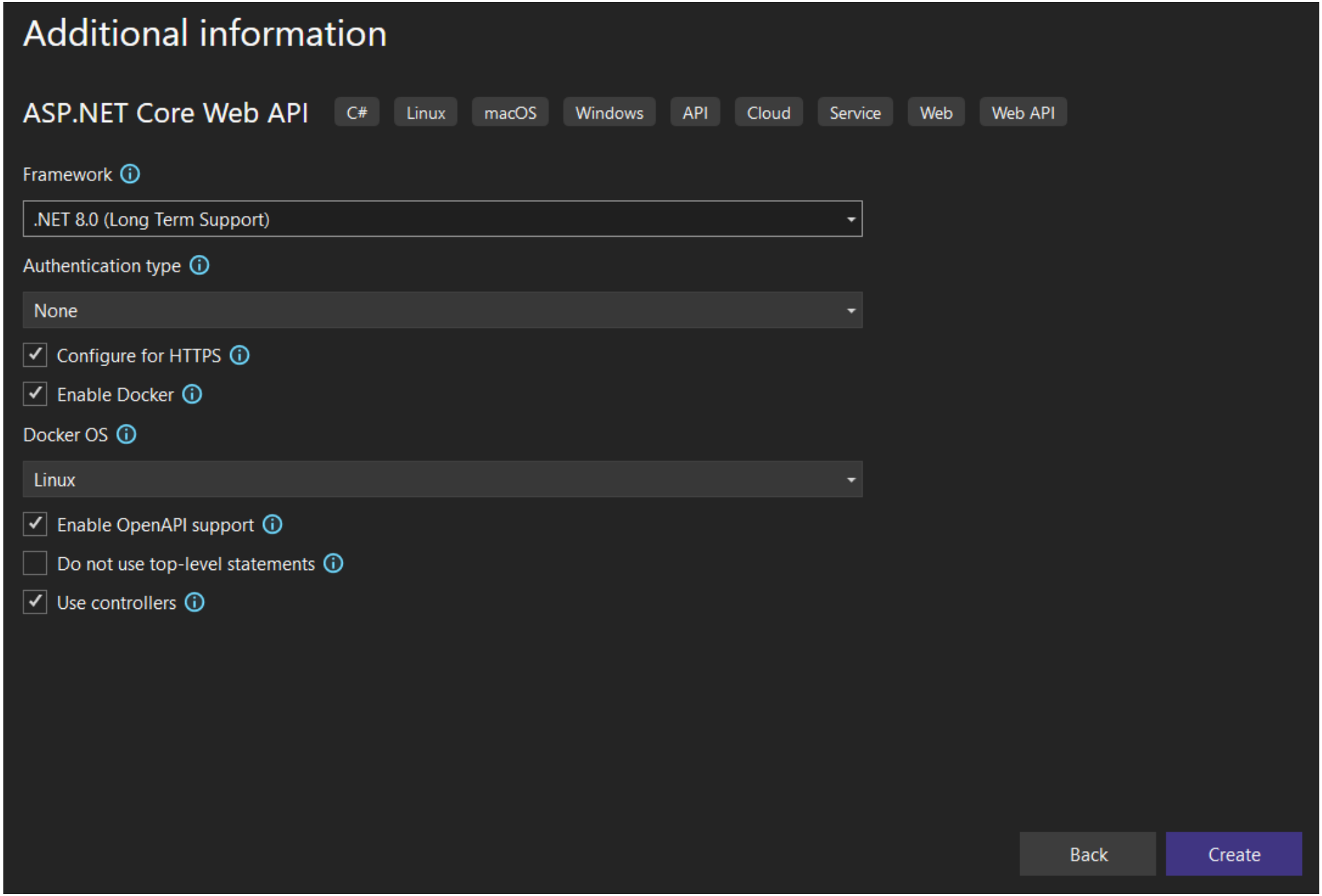
We select the ASP.NET Core Web API project template



We set the project name and location

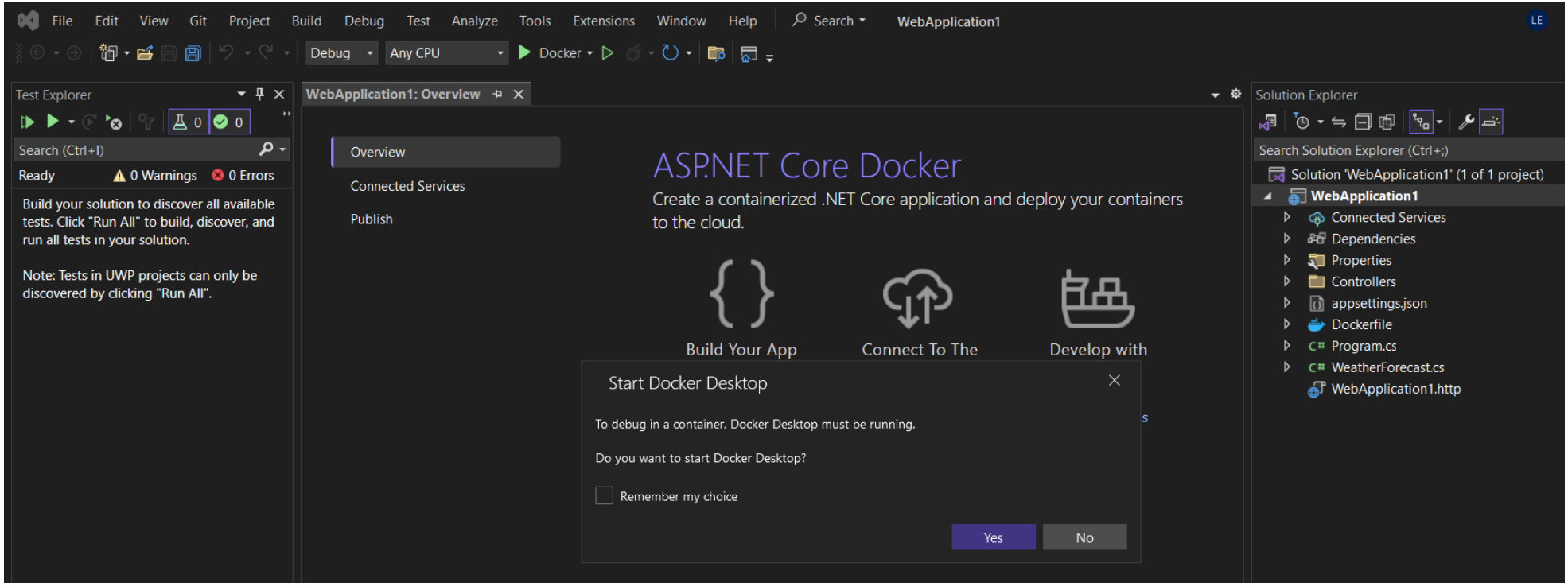


We select the project main features (.NET 8 Framework, Configure HTTPS, Enable Docker and Docker Operating System Linux)

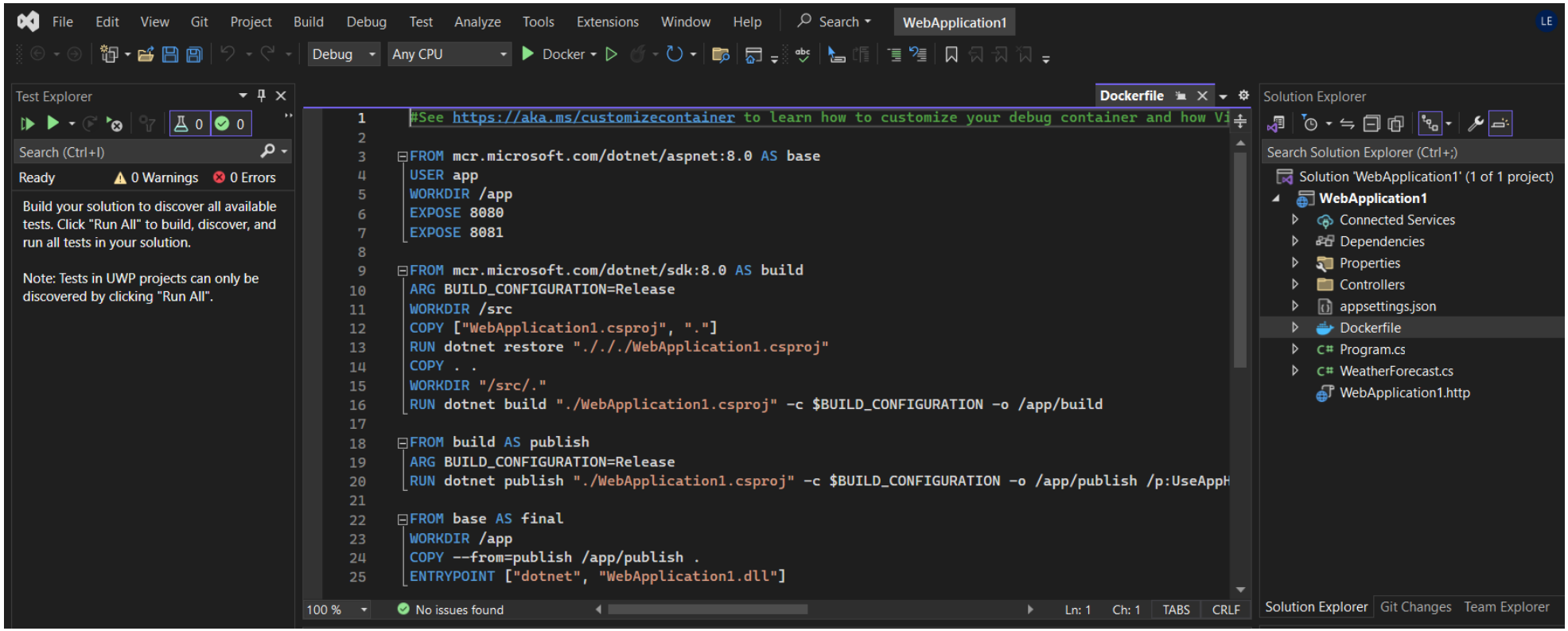


We press the create button

Here is the new .NET 8 Web API folders structure and start the **Docker Desktop** as requested in the message



As you can see in the following picture we created a **Dockerfile** when creating the application



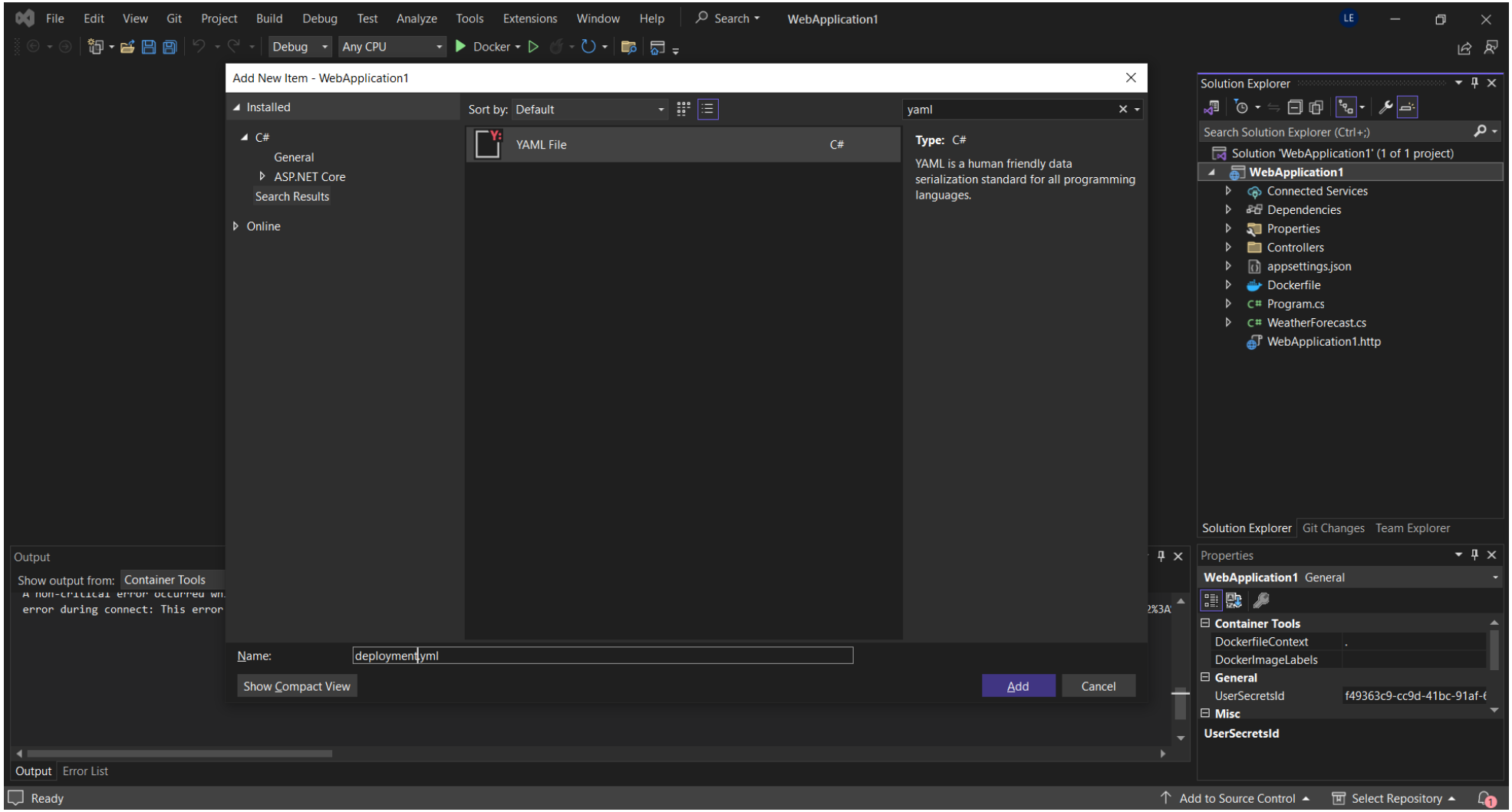
This is the Dockerfile source code

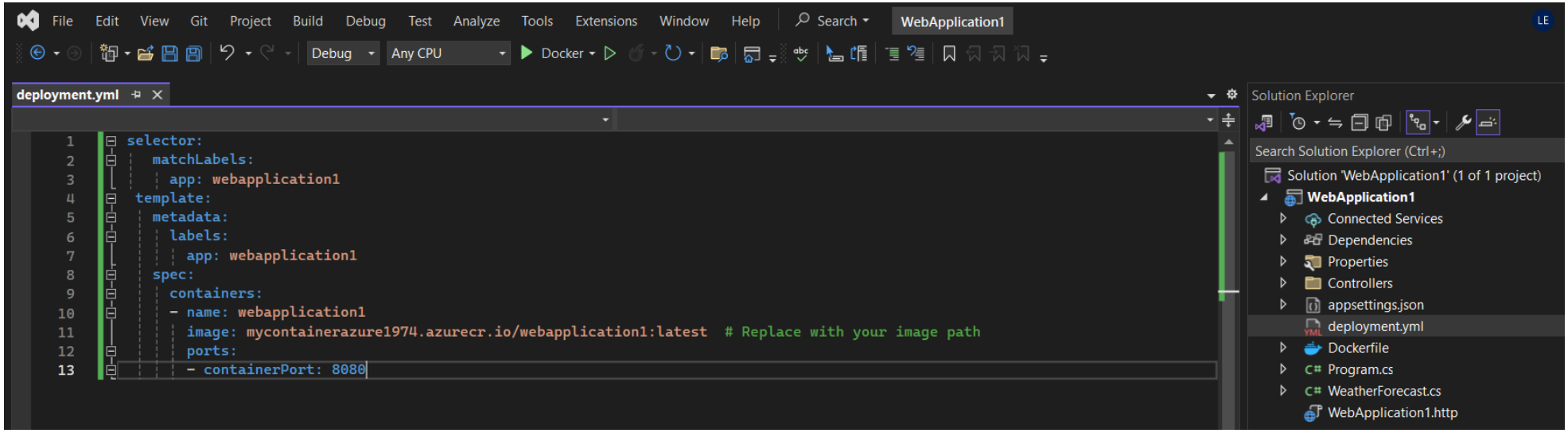
Dockerfile

```
#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 ["WebApplication1.csproj", "."]
RUN dotnet restore "./././WebApplication1.csproj"
COPY . .
WORKDIR "/src/."
RUN dotnet build "./WebApplication1.csproj" -c $BUILD_CONFIGURATION -o /app/build

FROM build AS publish
ARG BUILD_CONFIGURATION=Release
RUN dotnet publish "./WebApplication1.csproj" -c $BUILD_CONFIGURATION -o /app/publish /p:UseAppHost=false

FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "WebApplication1.dll"]
```

Do not forget to add the Kubernetes manifest files in your application: **deployment.yml** and **service.yml**

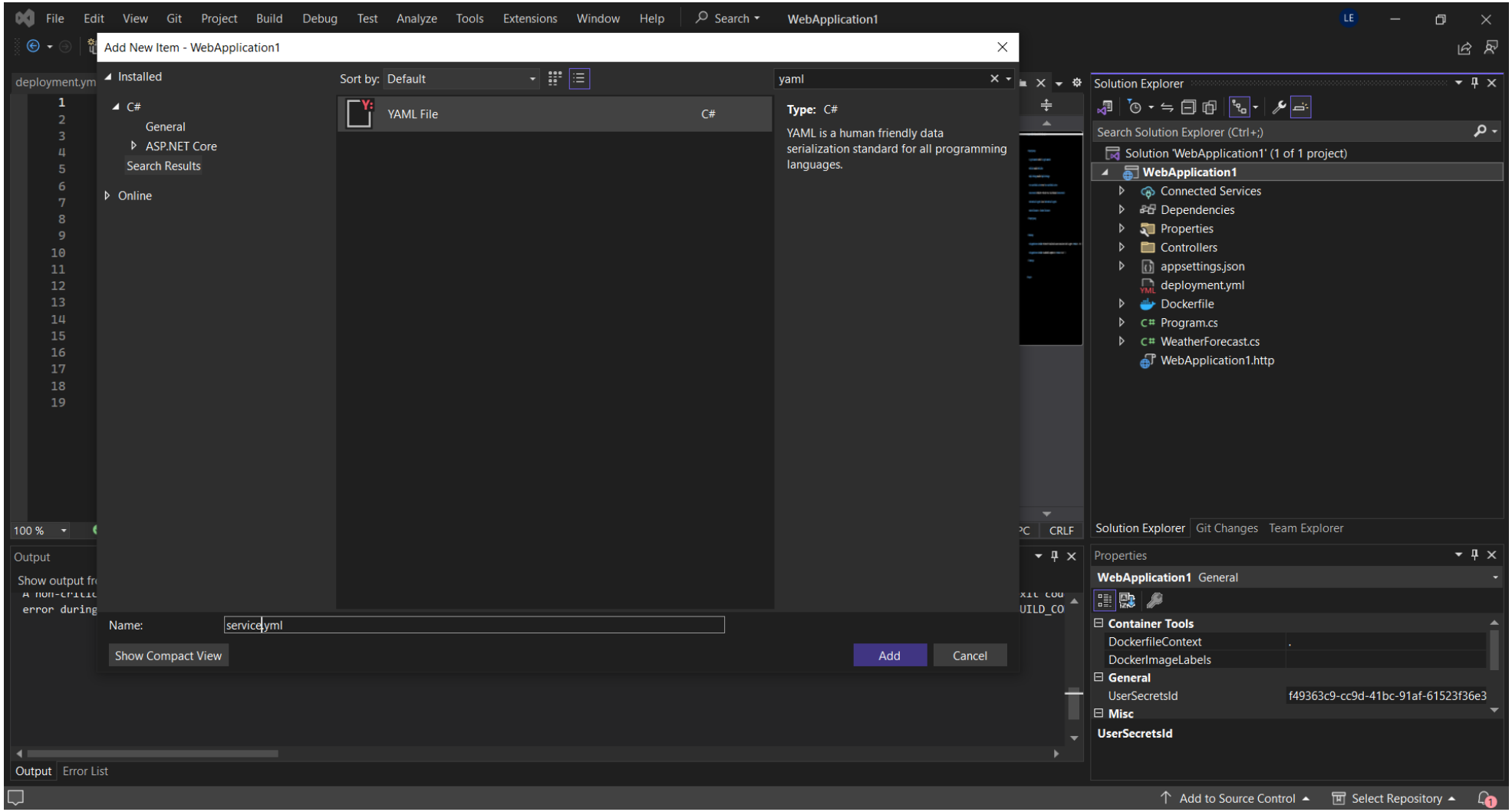

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search
WebApplication1
Debug
Any CPU
Docker
Add New Item - WebApplication1
Installed
C#
General
ASP.NET Core
Search Results
Online
Sort by: Default
YAML File
C#
yaml
Type: C#
YAML is a human friendly data serialization standard for all programming languages.
Name:
deployment.yml
Show Compact View
Add
Cancel
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'WebApplication1' (1 of 1 project)
WebApplication1
Connected Services
Dependencies
Properties
Controllers
appsettings.json
Dockerfile
Program.cs
WeatherForecast.cs
WebApplication1.http
Solution Explorer Git Changes Team Explorer
Output
Show output from: Container Tools
error during connect: This error
Output Error List
Properties
WebApplication1 General
Container Tools
DockerfileContext
DockerImageLabels
General
UserSecretsId
f49363c9-cc9d-41bc-91af-
Misc
UserSecretsId
Ready
Add to Source Control
Select Repository

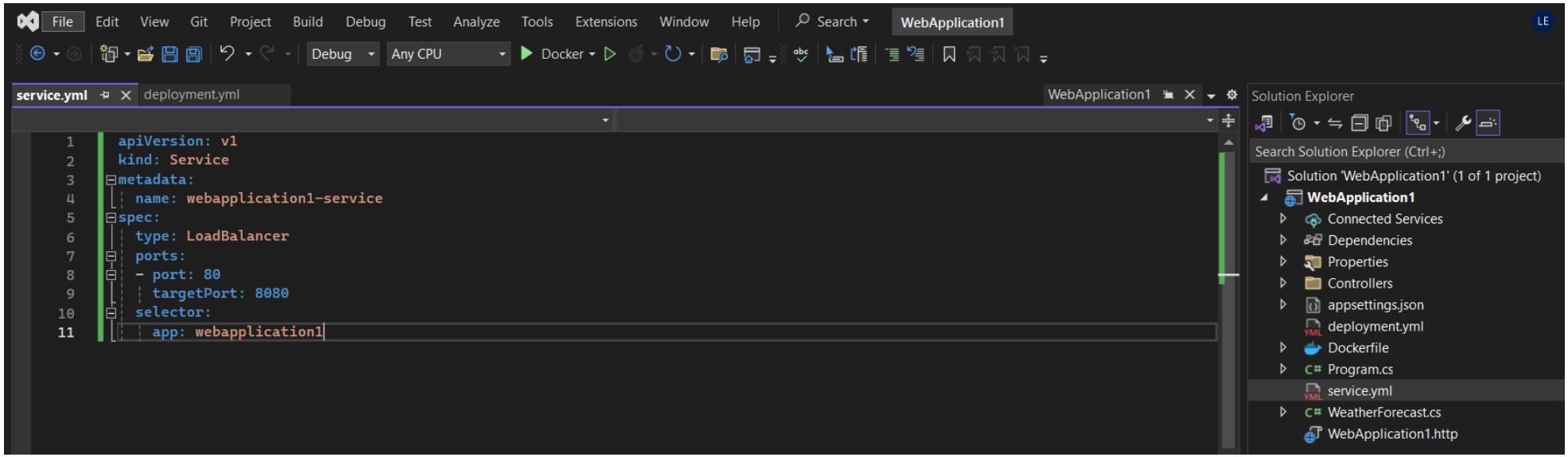


deployment.yml

```yaml
selector:
  matchLabels:
    app: webapplication1
template:
  metadata:
    labels:
      app: webapplication1
  spec:
    containers:
    - name: webapplication1
      image: mycontainerazure1974.azurecr.io/webapplication1:latest  # Replace with your image path
      ports:
      - containerPort: 8080
```

We also add the service.yml file

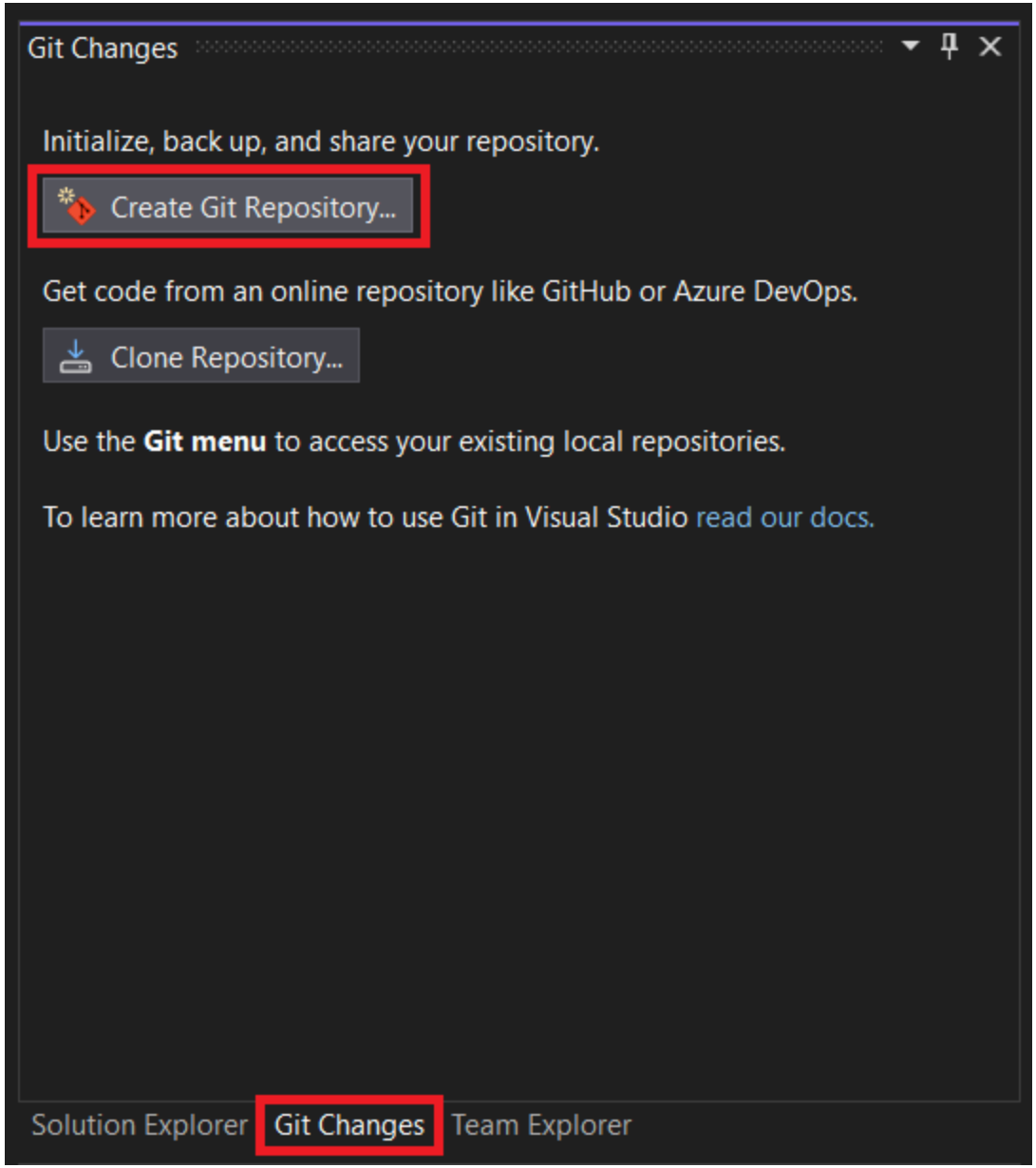

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search
WebApplication1
Add New Item - WebApplication1
Installed
C#
General
ASP.NET Core
Search Results
Online
Sort by: Default
YAML File
C#
yaml
Type: C#
YAML is a human friendly data serialization standard for all programming languages.
Name:
service.yml
Show Compact View
Add
Cancel
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'WebApplication1' (1 of 1 project)
WebApplication1
Connected Services
Dependencies
Properties
Controllers
appsettings.json
deployment.yml
Dockerfile
Program.cs
WeatherForecast.cs
WebApplication1.http
Solution Explorer Git Changes Team Explorer
Properties
WebApplication1 General
Container Tools
DockerfileContext
DockerImageLabels
General
UserSecretsId
f49363c9-cc9d-41bc-91af-61523f36e3
Misc
UserSecretsId
Output
Output Error List
Add to Source Control
Select Repository



sevice.yml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: webapplication1-service
spec:
  type: LoadBalancer
  ports:
  - port: 80
    targetPort: 8080
  selector:
    app: webapplication1
```

2. Create a Github repository in Visual Studio 2022 an upload the .NET 8 Web API source code


Git Changes
Initialize, back up, and share your repository.
Create Git Repository...
Get code from an online repository like GitHub or Azure DevOps.
Clone Repository...
Use the Git menu to access your existing local repositories.
To learn more about how to use Git in Visual Studio read our docs.
Solution Explorer
Git Changes
Team Explorer

Create a Git repository

Push to a new remote

- GitHub
- Azure DevOps

Other

- Existing remote
- Local only

Initialize a local Git repository

Local path: C:\.NET8WebAPI_for_Github_Actions\WebApplication1

.gitignore template: Default (VisualStudio)

License template: None

☑ Add a README.md

Create a new GitHub repository

Account: luiscoco (GitHub)

Re-enter your credentials

Owner: luiscoco

Repository name: GithubActions_Create_DockerImage_Upload_to_Azure_ACR_dotNET8WebAP

Description: Enter the description of the GitHub repository <Optional>

☐ Private repository

Push your code to GitHub

https://github.com/luiscoco/GithubActionsCreate_DockerImage_Upload_to_Azure_ACR_dotNET8WebAP

Create and Push | Cancel

3. Create Azure Container Registry ACR service for storing your Docker image

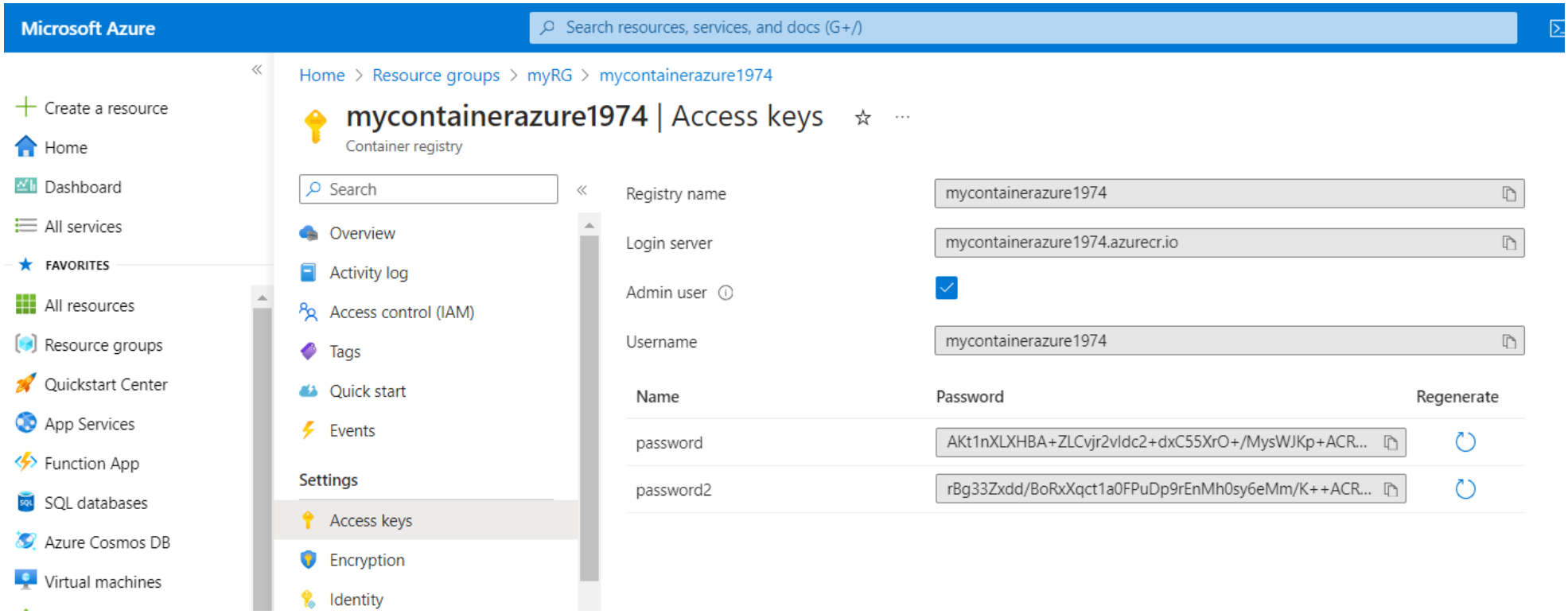
First, with Azure CLI, we create a new ResourceGroup in France Central region

```
az group create --name myRG --location francecentral
```

Then we create the new Azure Container Registry ACR named "mycontainerazure1974"

```
az acr create --name mycontainerazure1974 --resource-group myRG --sku Basic --location francecentral
```

We set Admin User in our new Azure ACR

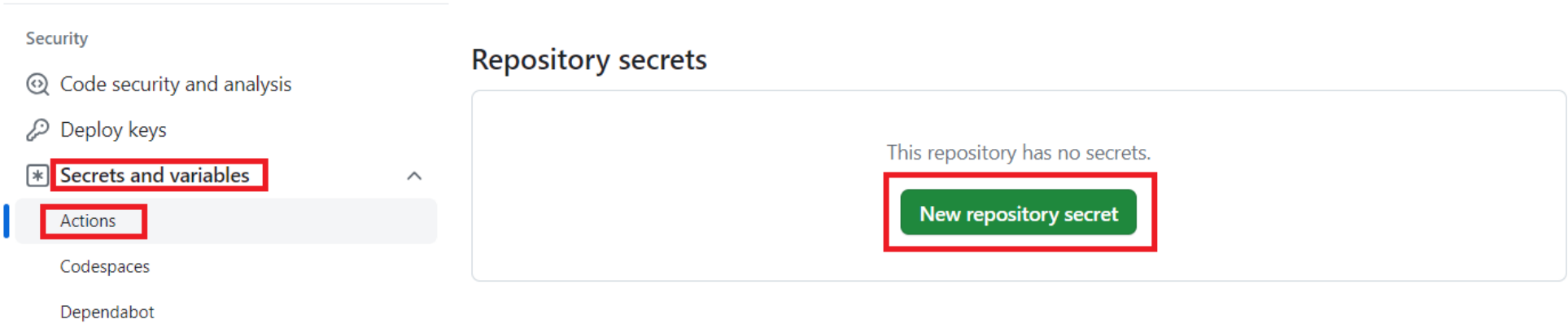


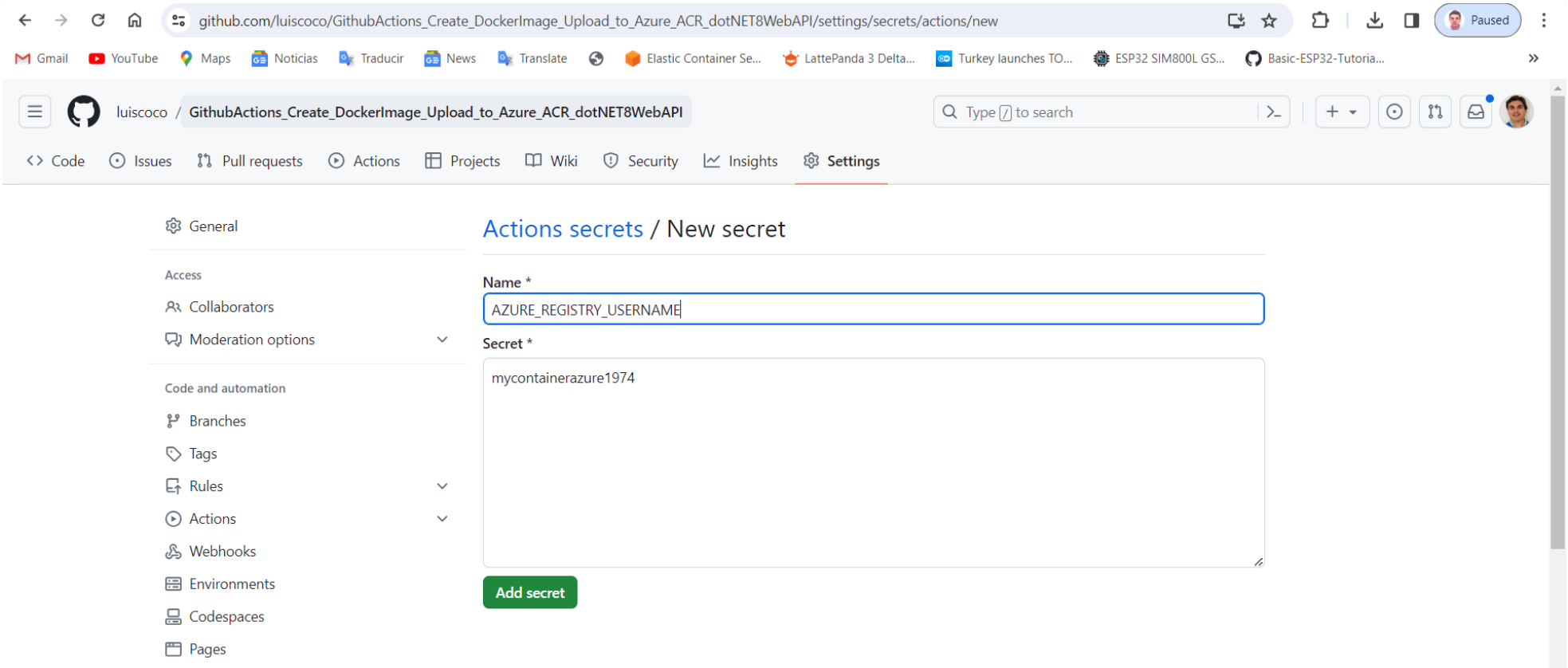
We copy the username and the password to store both values in Gihub repository secrets.

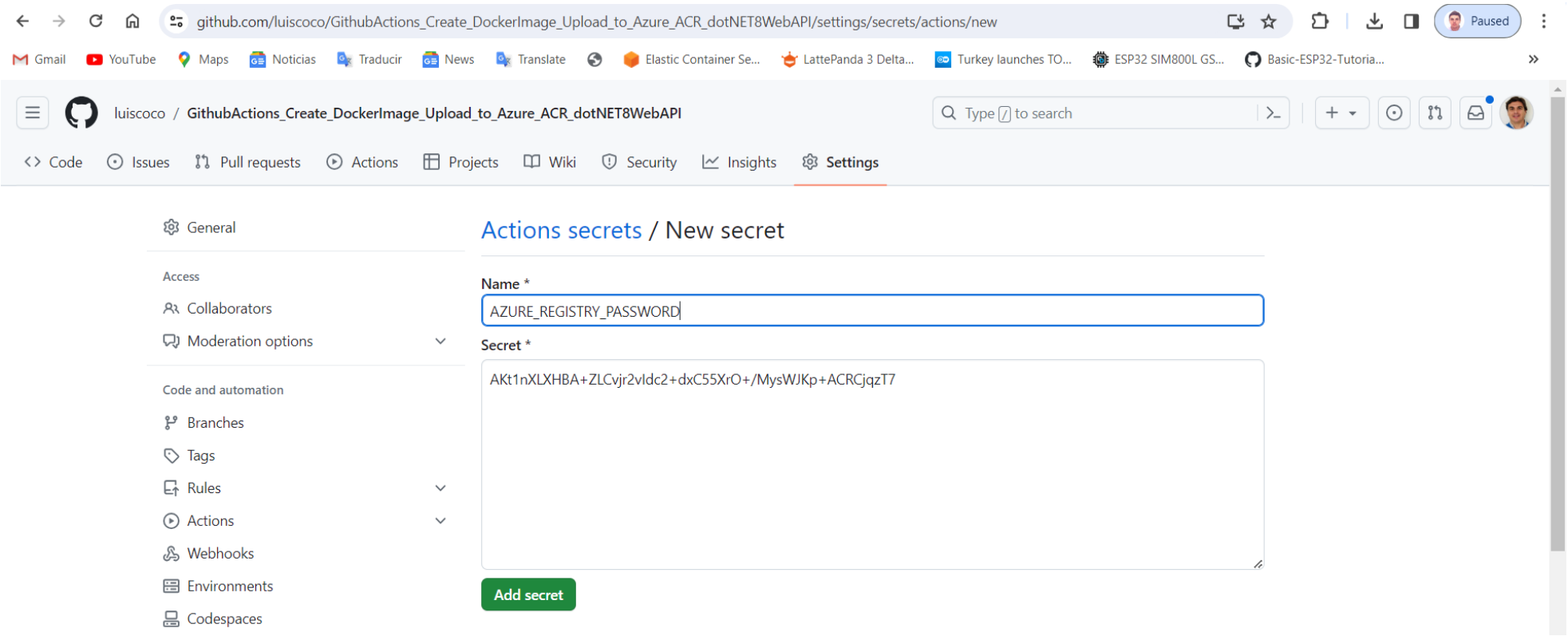
4. Create the Github repository secrets (Docker Hub username and password)

We nagivate to the Setting menu option in our Github repository

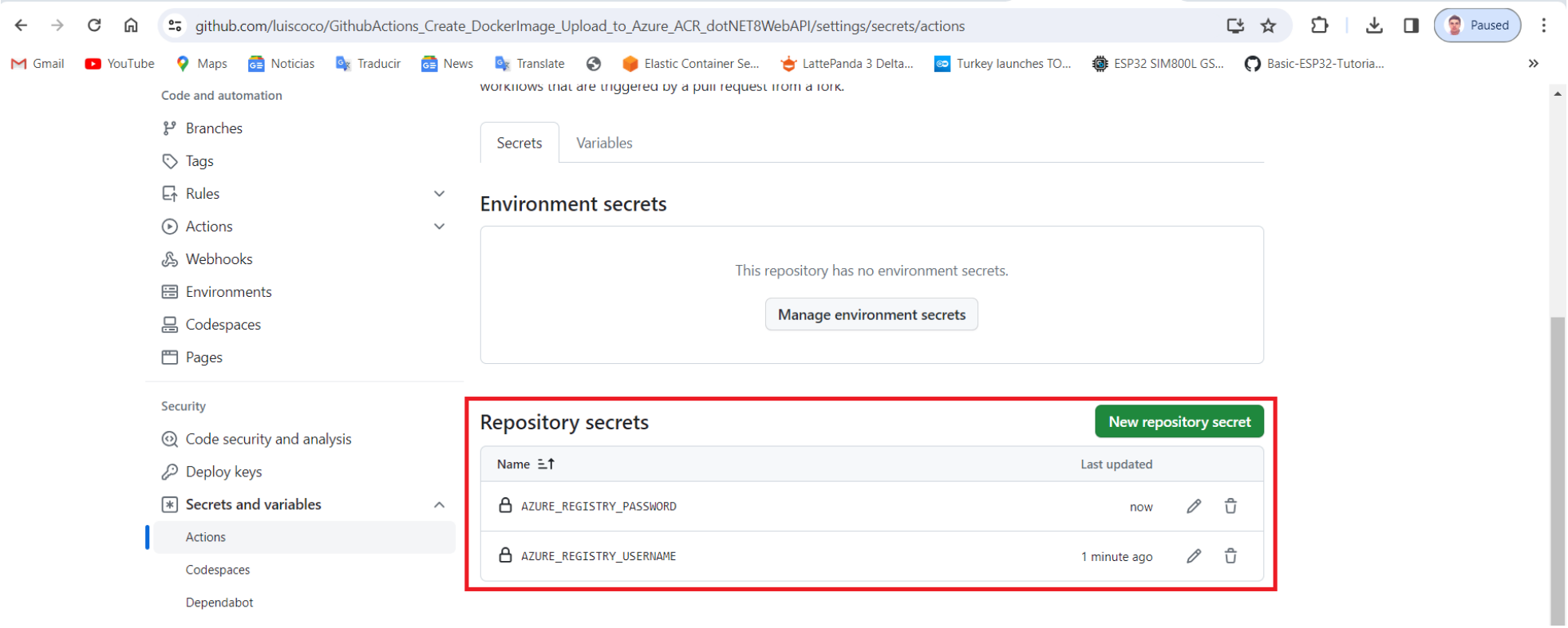
We select Settings->Secrets and variables->Actions->New repository secret and we create the secrets for storing the Azure ACR username and password




github.com/luiscoco/GithubActions_Create_DockerImage_Upload_to_Azure_ACR_dotNET8WebAPI/settings/secrets/actions/new
Paused
Gmail
YouTube
Maps
Noticias
Traducir
News
Translate
Elastic Container Se...
LattePanda 3 Delta...
Turkey launches TO...
ESP32 SIM800L GS...
Basic-ESP32-Tutoria...
luiscoco / GithubActions_Create_DockerImage_Upload_to_Azure_ACR_dotNET8WebAPI
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
General
Access
Collaborators
Moderation options
Code and automation
Branches
Tags
Rules
Actions
Webhooks
Environments
Codespaces
Pages
Actions secrets / New secret
Name *
AZURE_REGISTRY_USERNAME
Secret *
mycontainerazure1974
Add secret

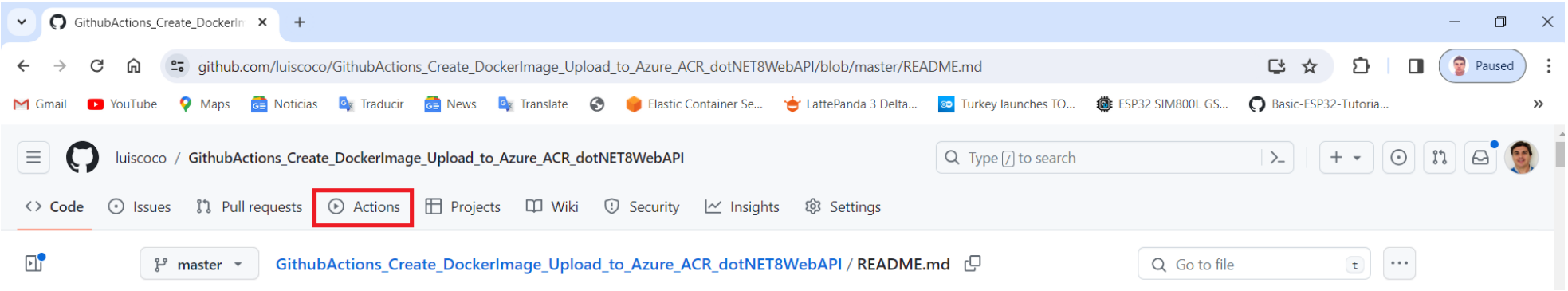


These are the two secrets stored in Github

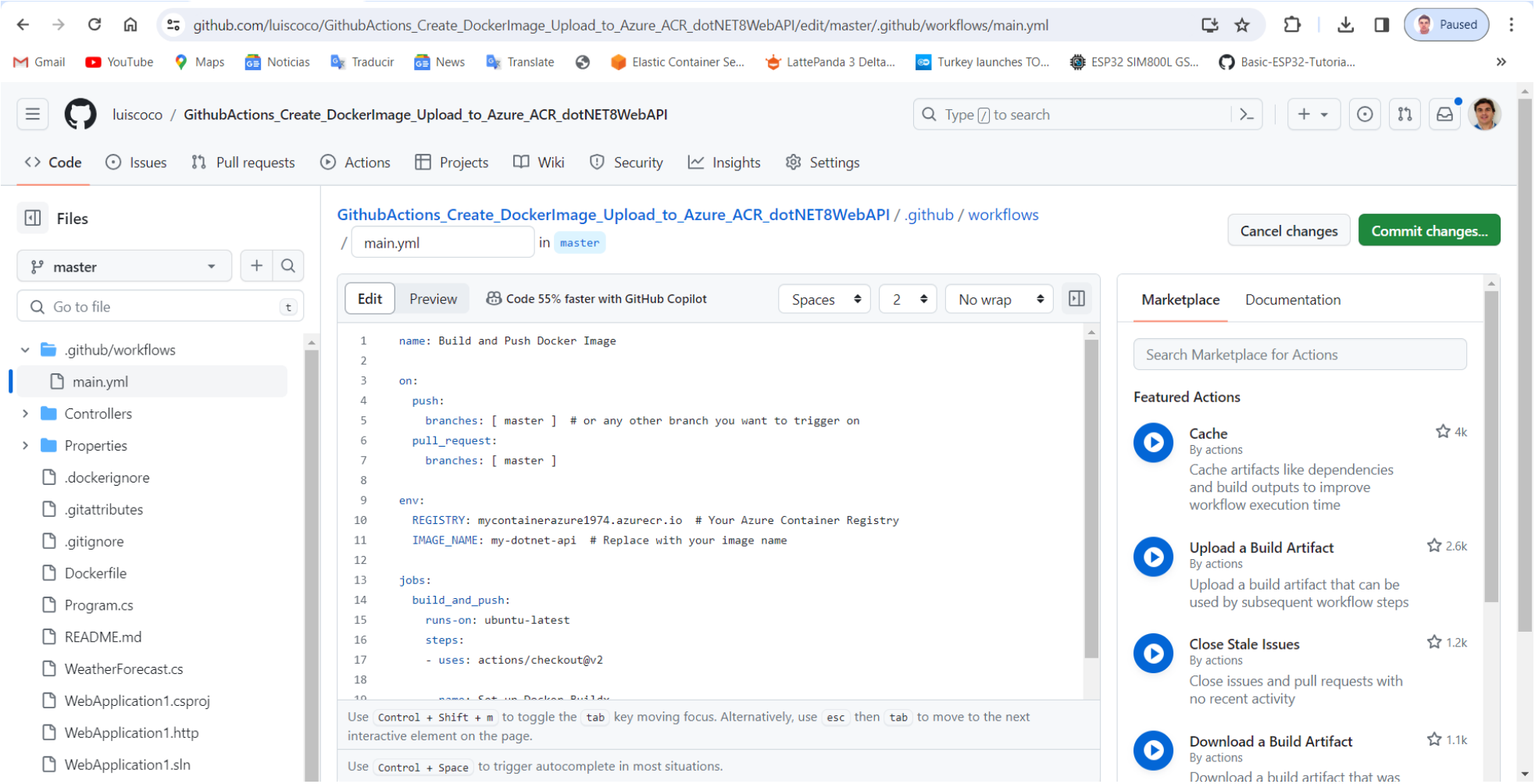


4. Create the Github Actions Workflow

In the Github repo we click on the "**Actions**" button to add a new workflow



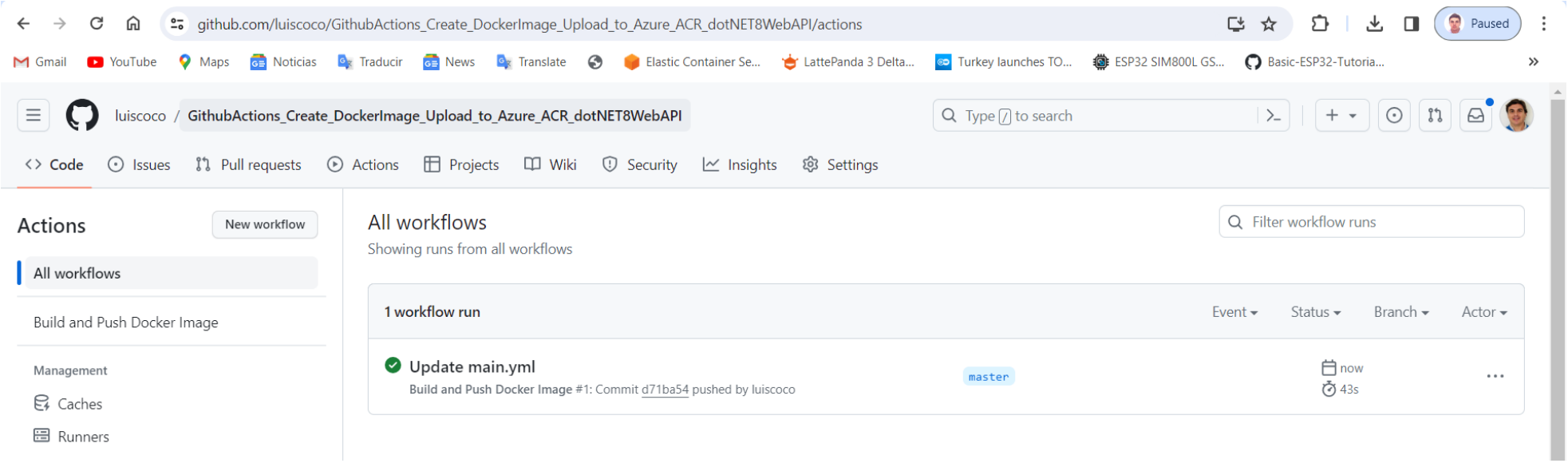
We input the workflow **main.yml** file source code and press the "**Commit changes...**"




```yaml
name: Build and Push Docker Image

on:
  push:
    branches: [ master ]  # or any other branch you want to trigger on
  pull_request:
    branches: [ master ]
```

```yaml
env:
  REGISTRY: mycontainerazure1974.azurecr.io  # Your Azure Container Registry
  IMAGE_NAME: webapplication1  # Replace with your image name

jobs:
  build_and_push:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2

    - name: Set up Docker Buildx
      uses: docker/setup-buildx-action@v1

    - name: Login to Azure Container Registry
      uses: docker/login-action@v1
      with:
        registry: ${{ env.REGISTRY }}
        username: ${{ secrets.AZURE_REGISTRY_USERNAME }}  # Set in GitHub secrets
        password: ${{ secrets.AZURE_REGISTRY_PASSWORD }}  # Set in GitHub secrets

    - name: Build and Push Image
      uses: docker/build-push-action@v2
      with:
        context: .
        file: ./Dockerfile  # Path to your Dockerfile
        push: true
        tags: ${{ env.REGISTRY }}/${{ env.IMAGE_NAME }}:latest

    # Additional steps for deployment or other actions can be added here.
```

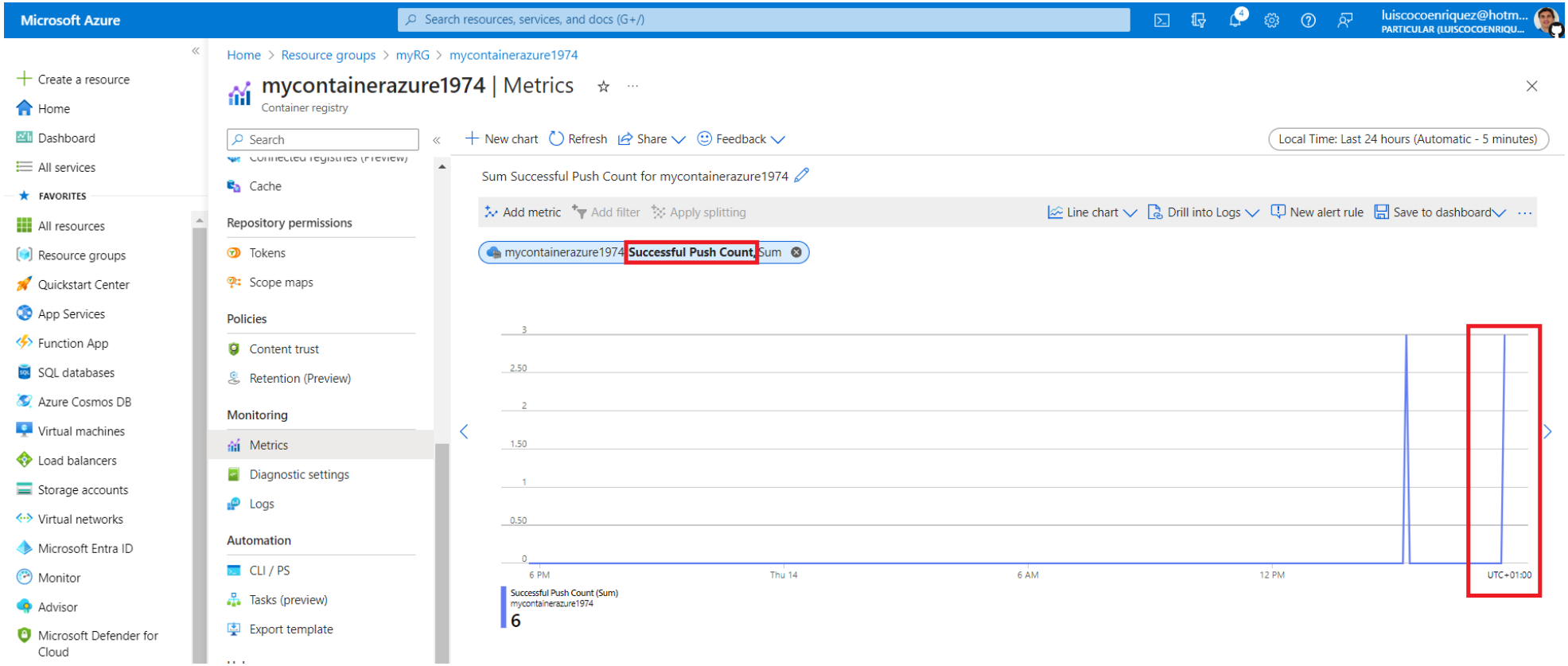
We verify the workflow is running ok



5. Verify in Azure Portal the docker image uploaded to Azure ACR

We log in Azure Portal and navigate to our new Azure ACR.

We select the Metrics option and we confirm we uploaded the docker image a few seconds ago



6. Deploy your application docker image in Azure Kubernetes AKS

We create a new Azure Kubernetes Cluster AKS with the following Azure CLI command:

```
az aks create --resource-group myRG --name mydotnet8webapiakscluster --location francecentral --node-count 1 --generate-ssh-keys
```

Then we attach my Container Registry ACR (called "mycontainerazure1974") to my Kubernetes Cluster AKS (called "mydotnet8webapiakscluster")

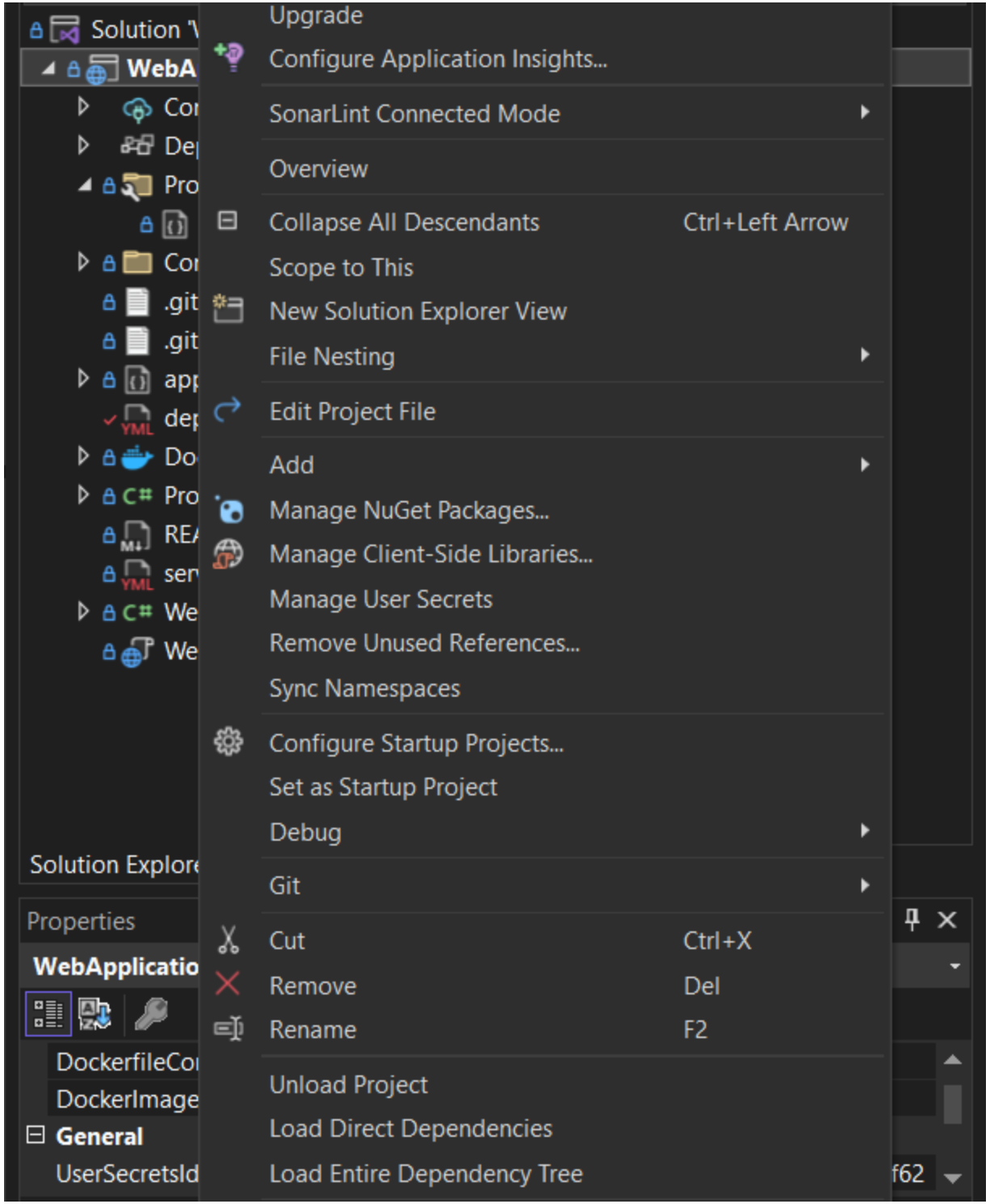
```
az aks update -n mydotnet8webapiakscluster -g myRG --attach-acr mycontainerazure1974
```

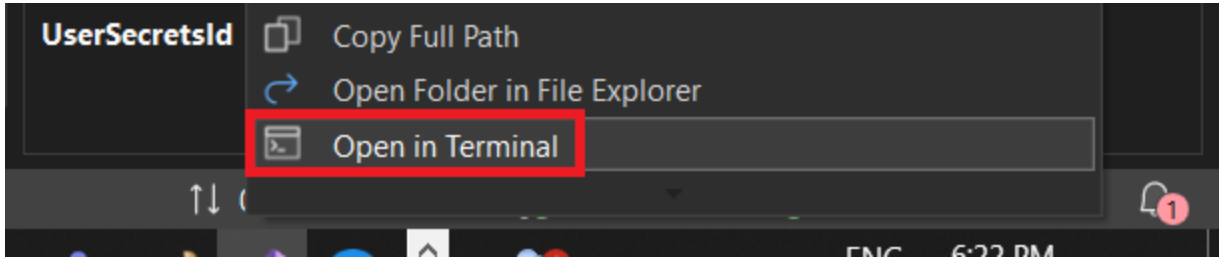
In I did not find any permission proble running the "az aks update" command.

If you don't have the necessary permissions, you might need to ask your Azure administrator to grant you the required roles.

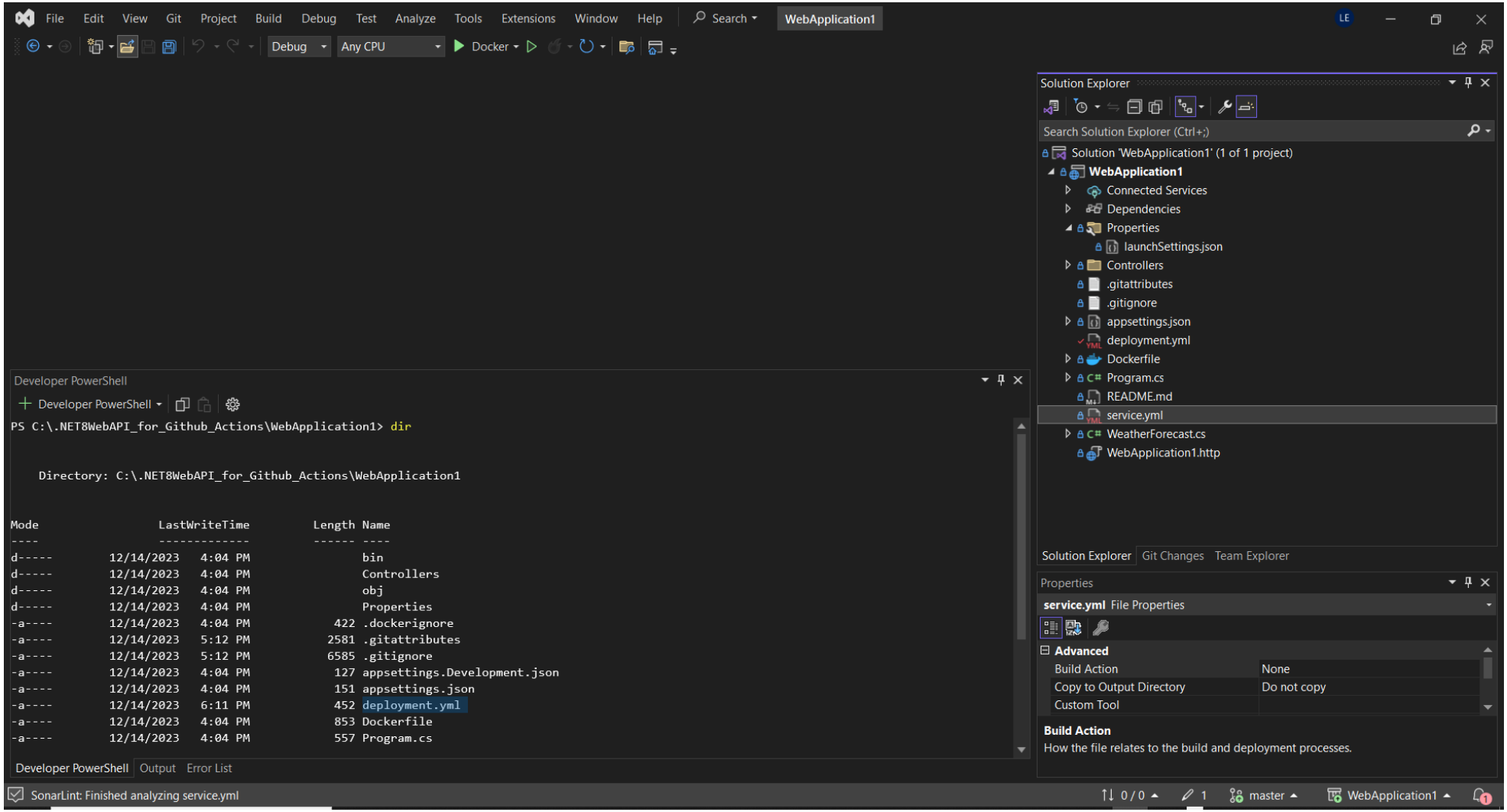
For instance, being assigned the "**Contributor**" role at the resource group or resource level would typically suffice for these operations.

We go to Visual Studio and we right click on the project and select the option **Open in Terminal**

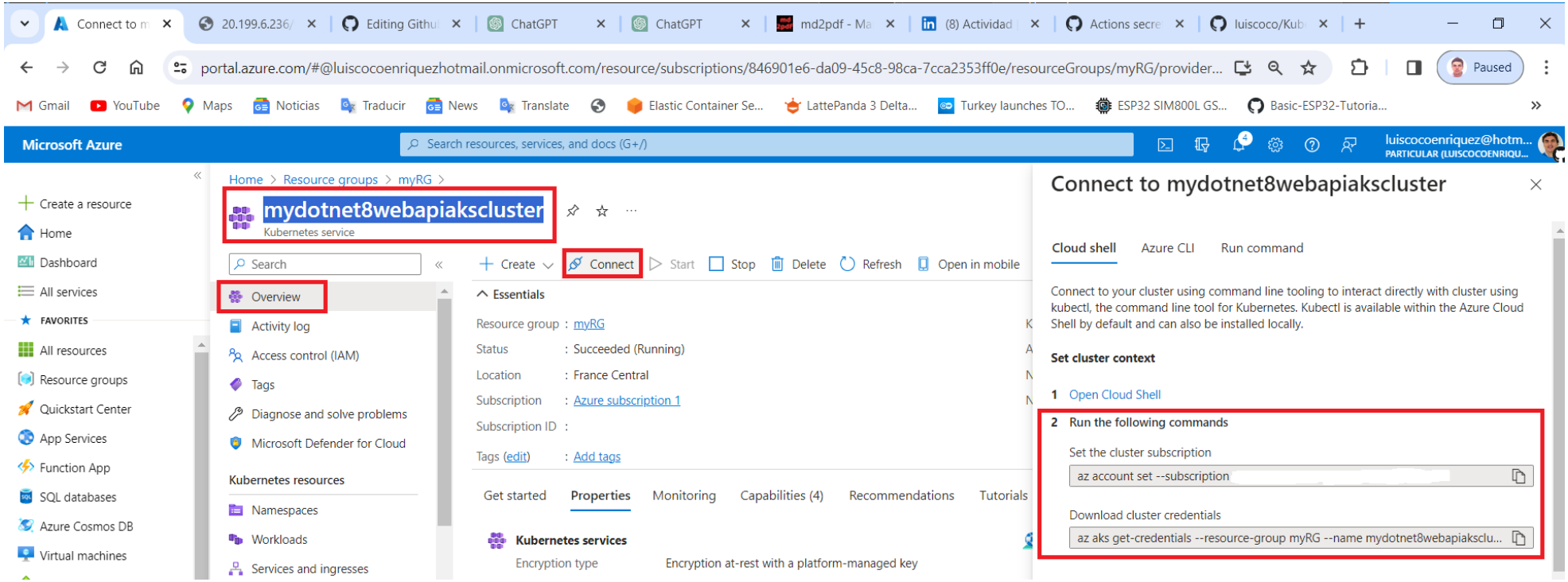

Upgrade
Configure Application Insights...
SonarLint Connected Mode
Overview
Collapse All Descendants Ctrl+Left Arrow
Scope to This
New Solution Explorer View
File Nesting
Edit Project File
Add
Manage NuGet Packages...
Manage Client-Side Libraries...
Manage User Secrets
Remove Unused References...
Sync Namespaces
Configure Startup Projects...
Set as Startup Project
Debug
Git
Cut Ctrl+X
Remove Del
Rename F2
Unload Project
Load Direct Dependencies
Load Entire Dependency Tree
Solution Explore
Properties
WebApplicatio
DockerfileCo
DockerImage
General
UserSecretsId



In the Terminal window we confirm we are in the folder where are located the Kubernetes manifest files (deployment.yml and service.yml)



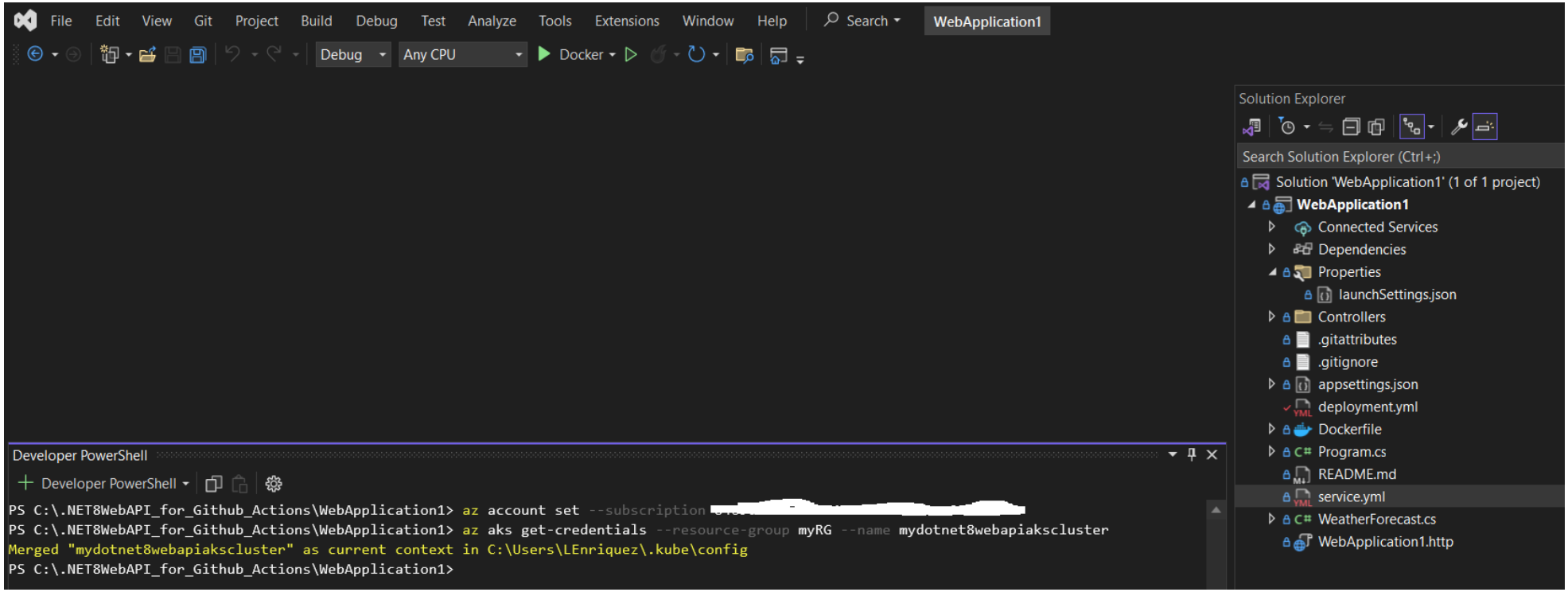
Then we open in another window Azure Portal and navigate to Azure AKS service and select the **Overview->Connect** menu option



Then we copy the commands and login in our Azure AKS:

```
az account set --subscription XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

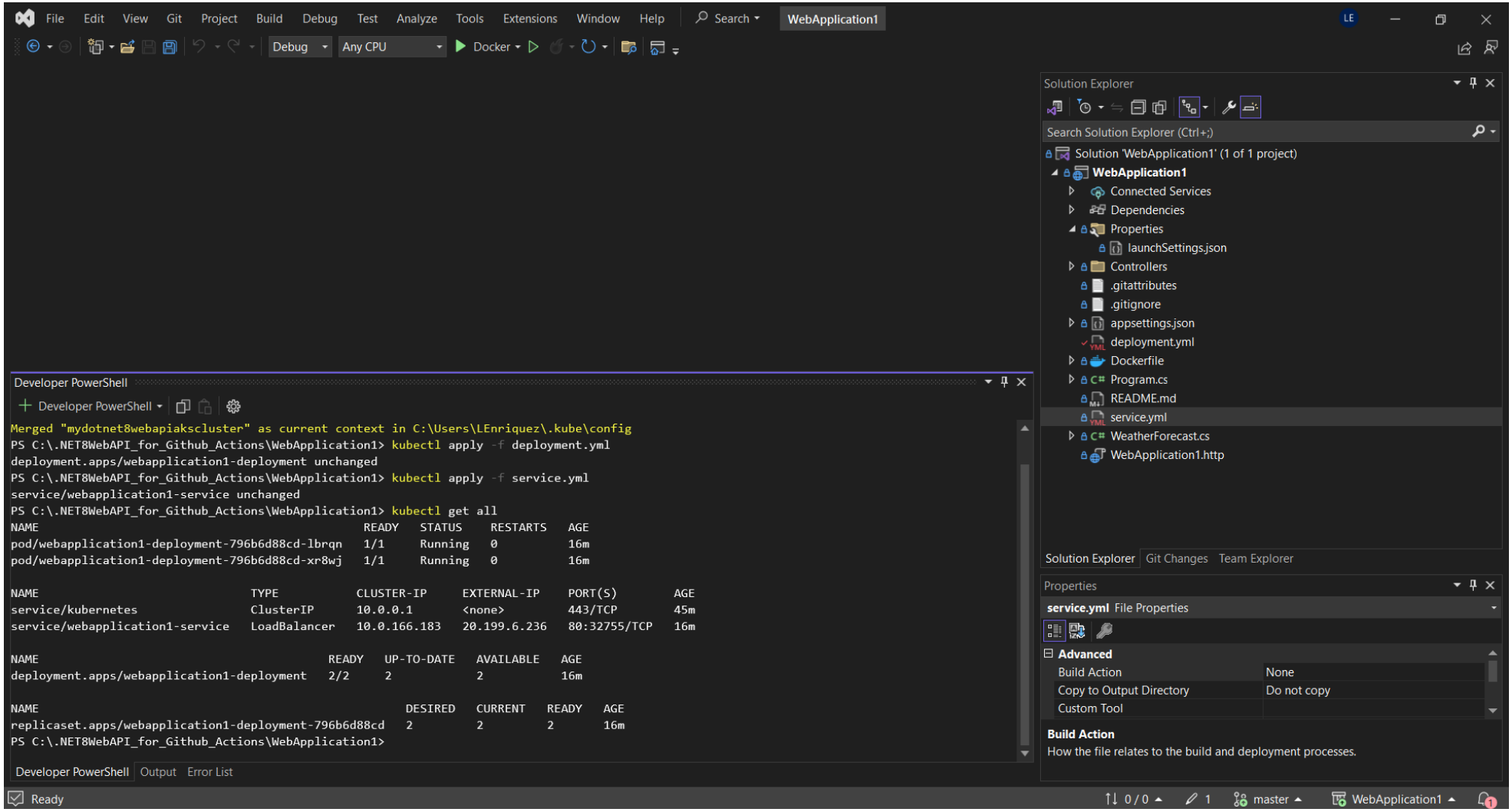
```
az aks get-credentials --resource-group myRG --name mydotnet8webapiakscluster
```



Now we apply the deployment to our AKS Cluster

```
kubectl apply -f deployment.yml
```

```
kubectl apply -f service.yml
```



We can see the Load Balancer is exposing the port 80 and the **external IP address is 20.199.6.236**

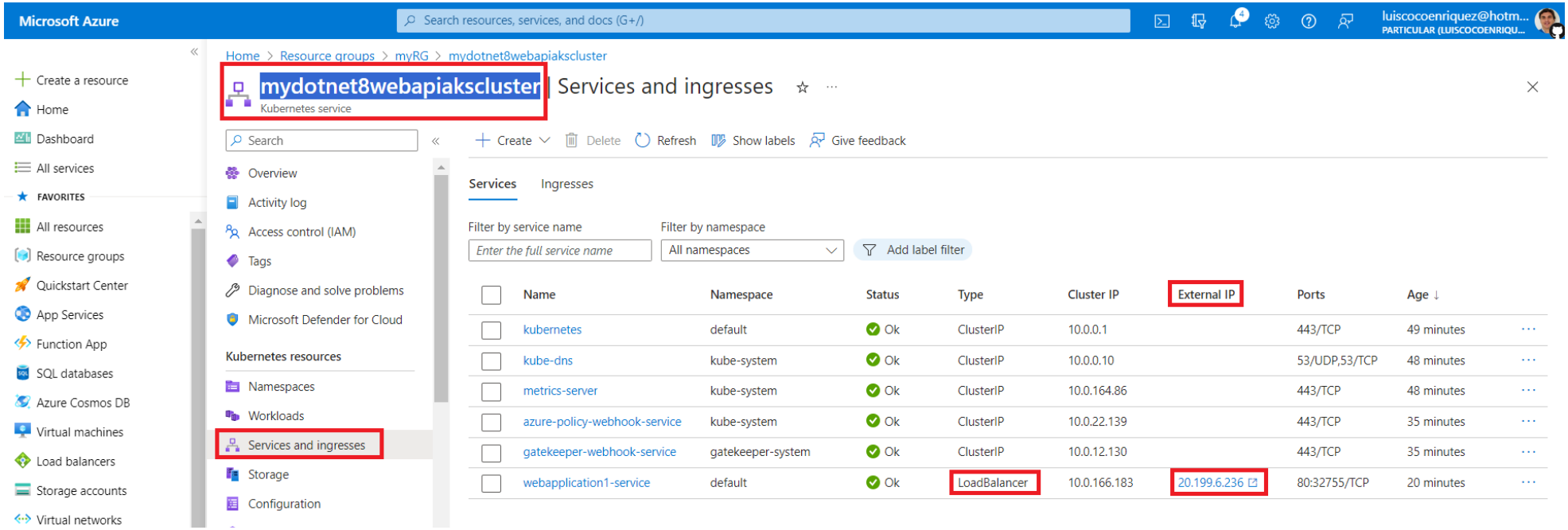
20.199.6.236/weatherforecast

Not secure | 20.199.6.236/weatherforecast


```
[
    {
        "date": "2023-12-15",
        "temperatureC": 51,
        "temperatureF": 123,
        "summary": "Sweltering"
    },
    {
        "date": "2023-12-16",
        "temperatureC": 15,
        "temperatureF": 58,
        "summary": "Freezing"
    },
    {
        "date": "2023-12-17",
        "temperatureC": 35,
        "temperatureF": 94,
        "summary": "Hot"
    },
    {
        "date": "2023-12-18",
        "temperatureC": 24,
        "temperatureF": 75,
        "summary": "Mild"
    },
    {
        "date": "2023-12-19",
        "temperatureC": 54,
        "temperatureF": 129,
        "summary": "Chilly"
    }
]
```


We can also confirm in Azure Portal the Load Balancer external IP

Navigate to the AKS and select the menu option **Services and Ingresses**


Microsoft Azure
Search resources, services, and docs (G+/)
luiscocoenriquez@hotm...
PARTICULAR (LUISCOCOENRIQU...
Create a resource
Home
Dashboard
All services
FAVORITES
All resources
Resource groups
Quickstart Center
App Services
Function App
SQL databases
Azure Cosmos DB
Virtual machines
Load balancers
Storage accounts
Virtual networks
Home > Resource groups > myRG > mydotnet8webapiakscluster
mydotnet8webapiakscluster | Services and ingresses
Kubernetes service
Search
Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Microsoft Defender for Cloud
Kubernetes resources
Namespaces
Workloads
Services and ingresses
Storage
Configuration
Create
Delete
Refresh
Show labels
Give feedback
Services
Ingresses
Filter by service name
Enter the full service name
Filter by namespace
All namespaces
Add label filter
Name
Namespace
Status
Type
Cluster IP
External IP
Ports
Age
kubernetes default Ok ClusterIP 10.0.0.1 443/TCP 49 minutes
kube-dns kube-system Ok ClusterIP 10.0.0.10 53/UDP,53/TCP 48 minutes
metrics-server kube-system Ok ClusterIP 10.0.164.86 443/TCP 48 minutes
azure-policy-webhook-service kube-system Ok ClusterIP 10.0.22.139 443/TCP 35 minutes
gatekeeper-webhook-service gatekeeper-system Ok ClusterIP 10.0.12.130 443/TCP 35 minutes
webapplication1-service default Ok LoadBalancer 10.0.166.183 20.199.6.236 80:32755/TCP 20 minutes