How to deploy SpringBoot WebAPI to AWS ECS

1. Create Spring Boot Web API application with VSCode

See this repo: https://github.com/luiscoco/SpringBoot_Sample2-created-WebAPI-with-VSCode

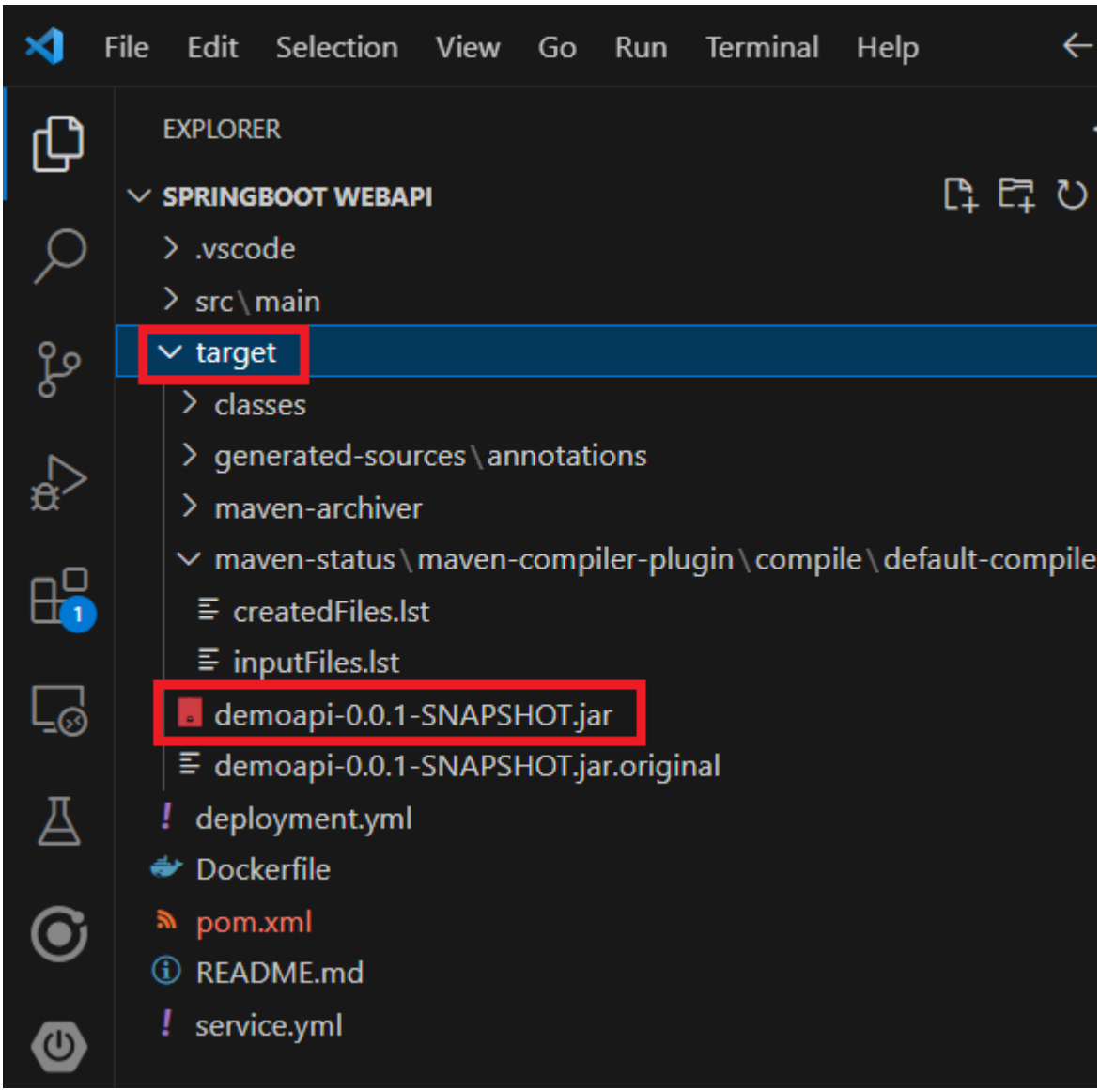
We set, in the **application.properties** file, the application port to 80

```
# Server Configuration
server.port=80

# Logging
logging.level.org.springframework.web=INFO
logging.level.org.hibernate=ERROR

# Actuator Endpoints
management.endpoints.web.exposure.include=*
management.endpoint.health.show-details=always

# Actuator Info
info.app.name=My Spring Boot Application
info.app.description=A simple demo application
info.app.version=1.0.0
```

Before creating the docker image we should create the **JAR** file withe the following command:

```
mvn clean install
```

We confirm we created the **target** folder



After creating the **JAR** file we run the application with this command:

```
java -jar .\target\demoapi-0.0.1-SNAPSHOT.jar
```

We verify the application endpoints:

http://localhost:80/hello

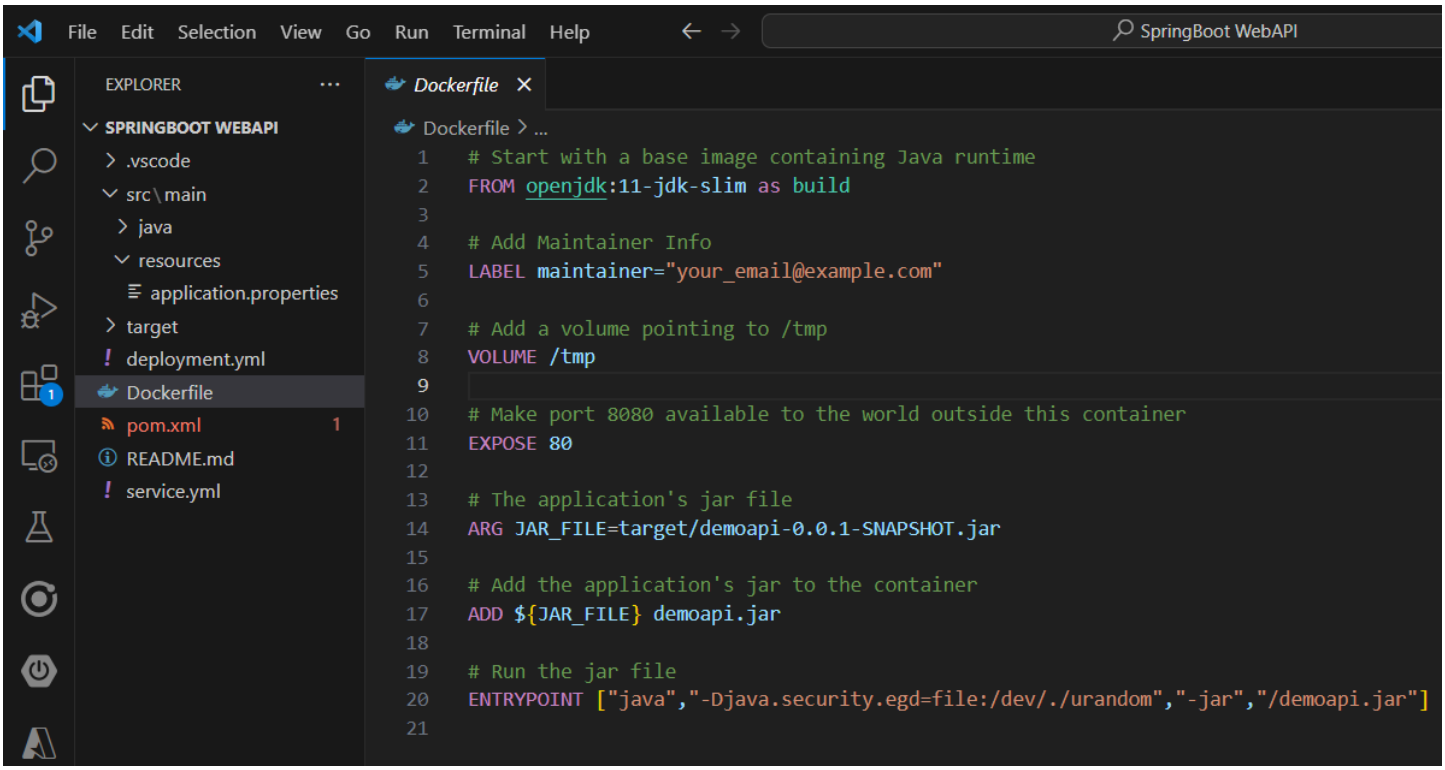
http://localhost:80/actuator/health

2. Create a Docker file

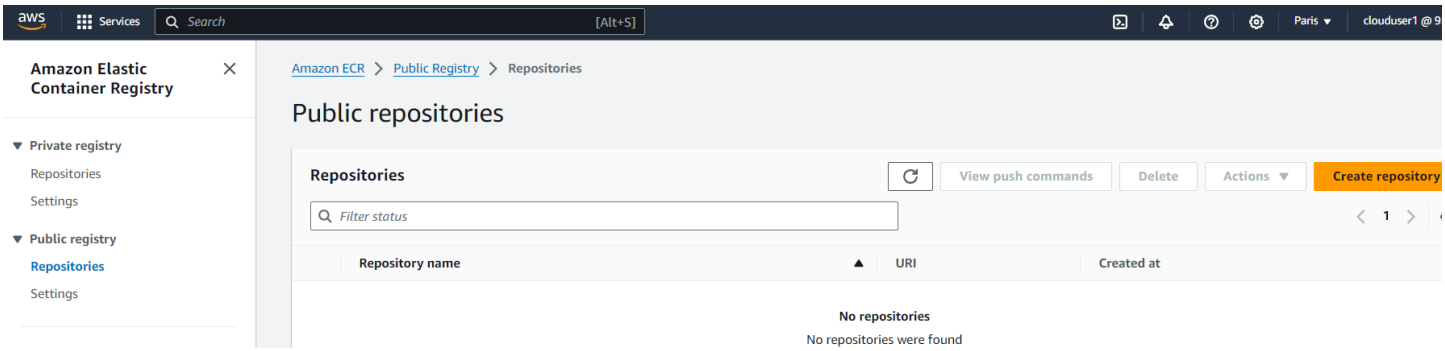
Create a Dockerfile and copy this content inside:

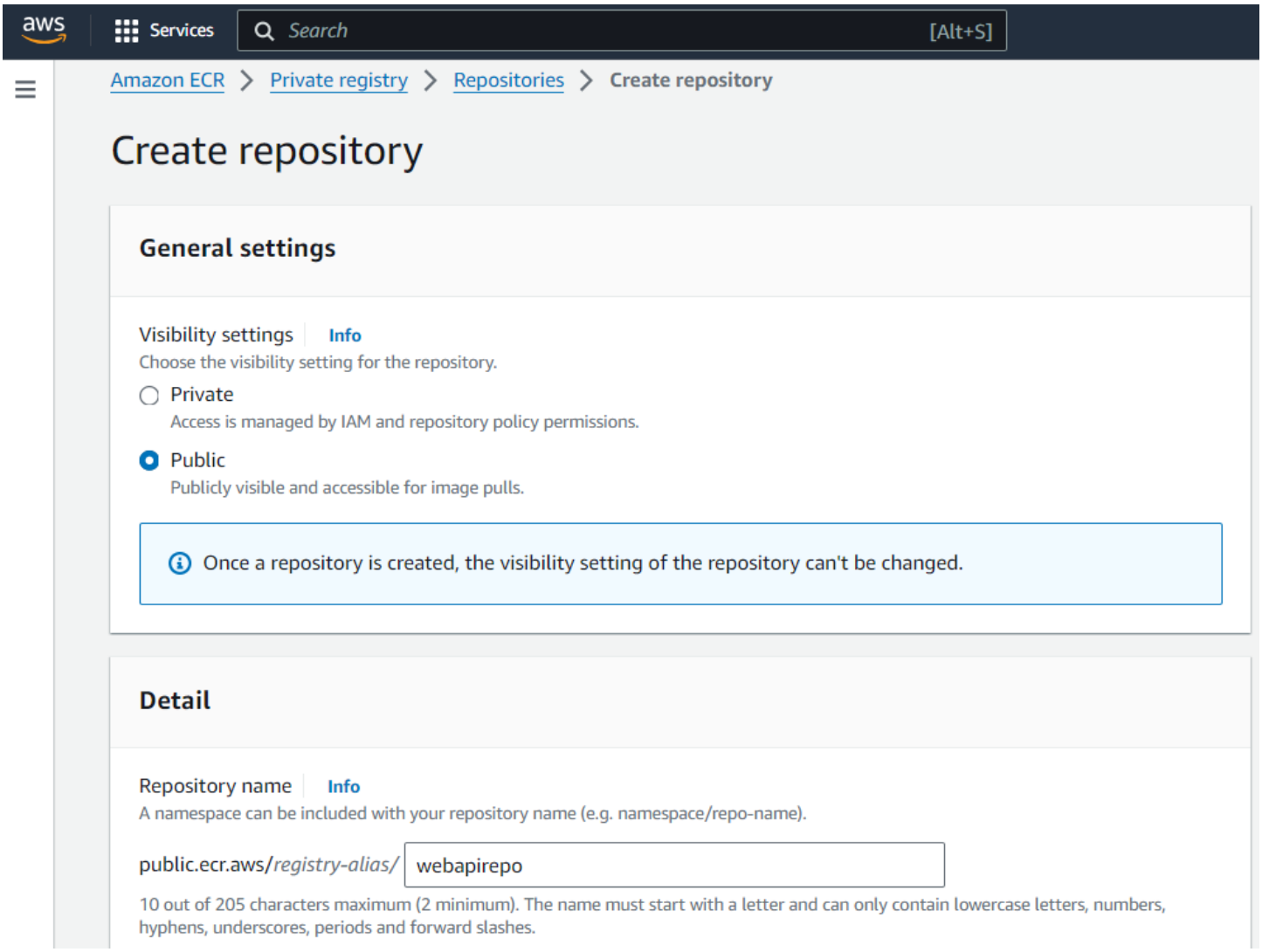
```
# Start with a base image containing Java runtime
FROM openjdk:11-jdk-slim as build

# Add Maintainer Info
LABEL maintainer="your_email@example.com"
```

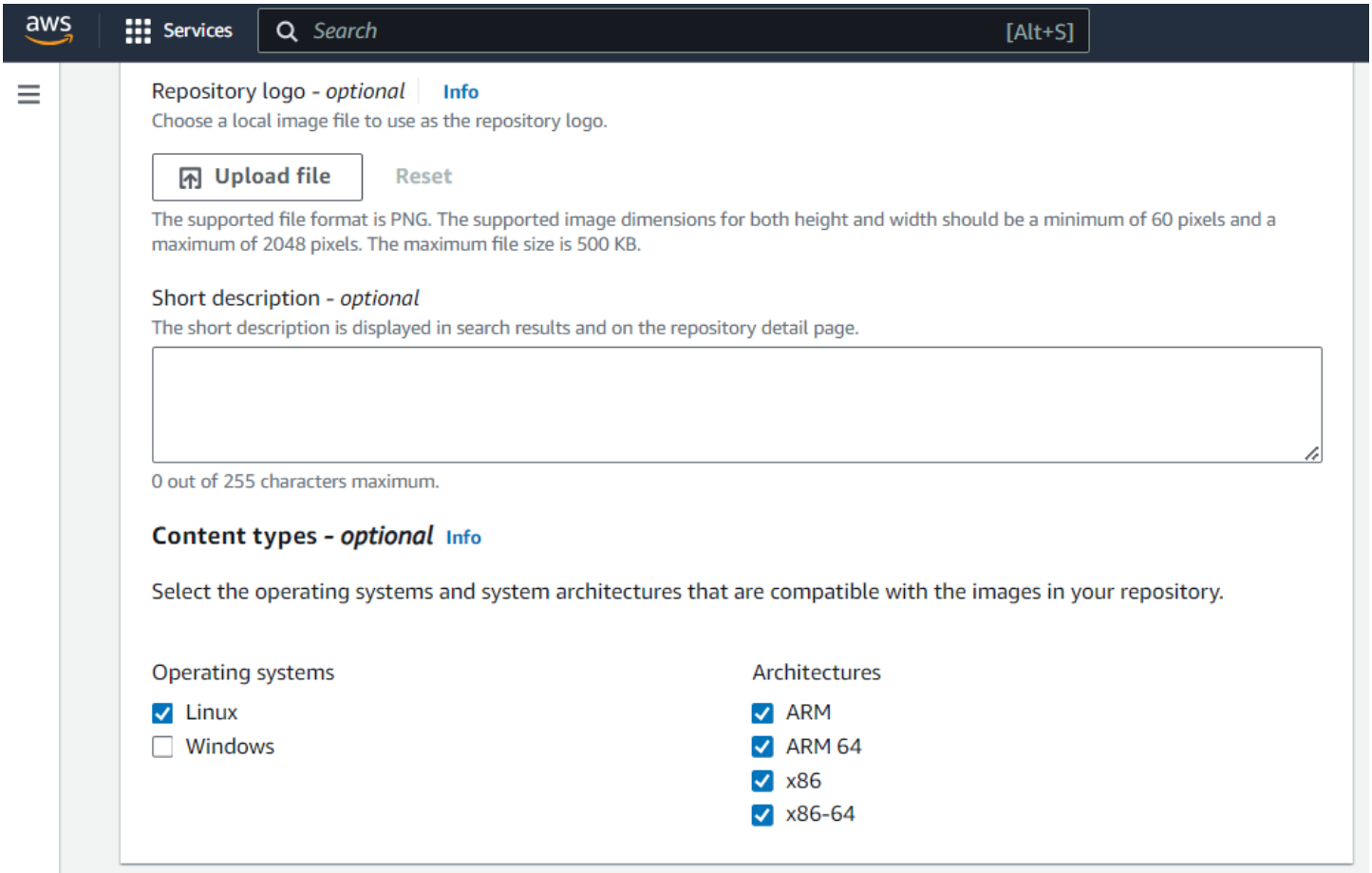
```
# Add a volume pointing to /tmp
VOLUME /tmp

# Make port 80 available to the world outside this container
EXPOSE 80

# The application's jar file
ARG JAR_FILE=target/demoapi-0.0.1-SNAPSHOT.jar

# Add the application's jar to the container
ADD ${JAR_FILE} demoapi.jar

# Run the jar file
ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/demoapi.jar"]
```

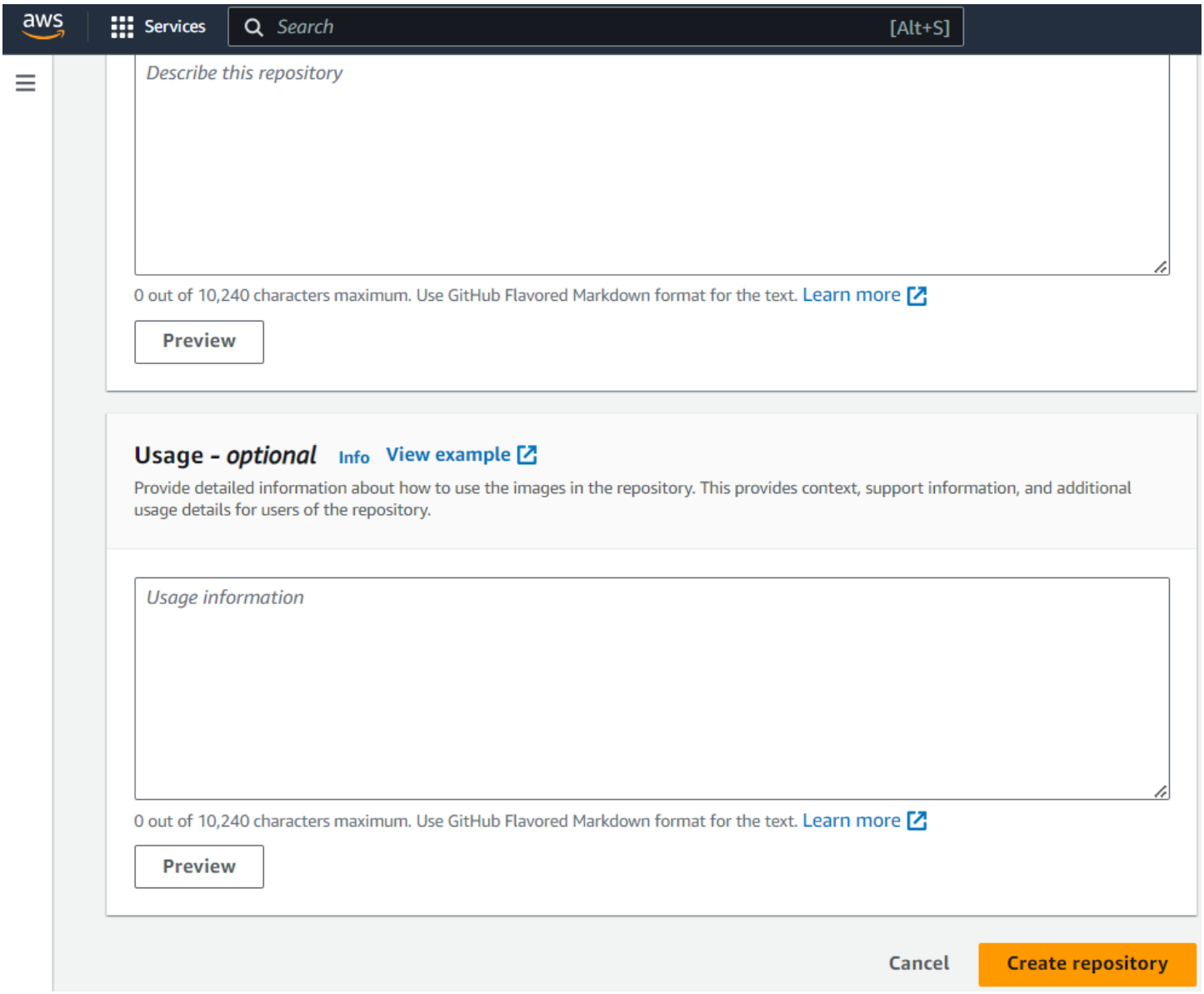


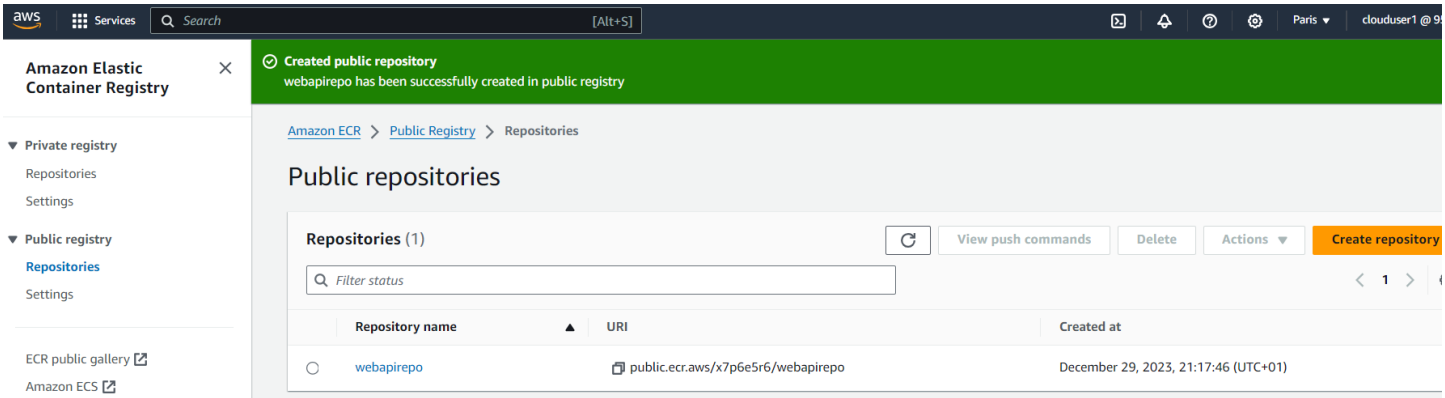
3. Create a AWS Elastic Container Registry ECR Public repo

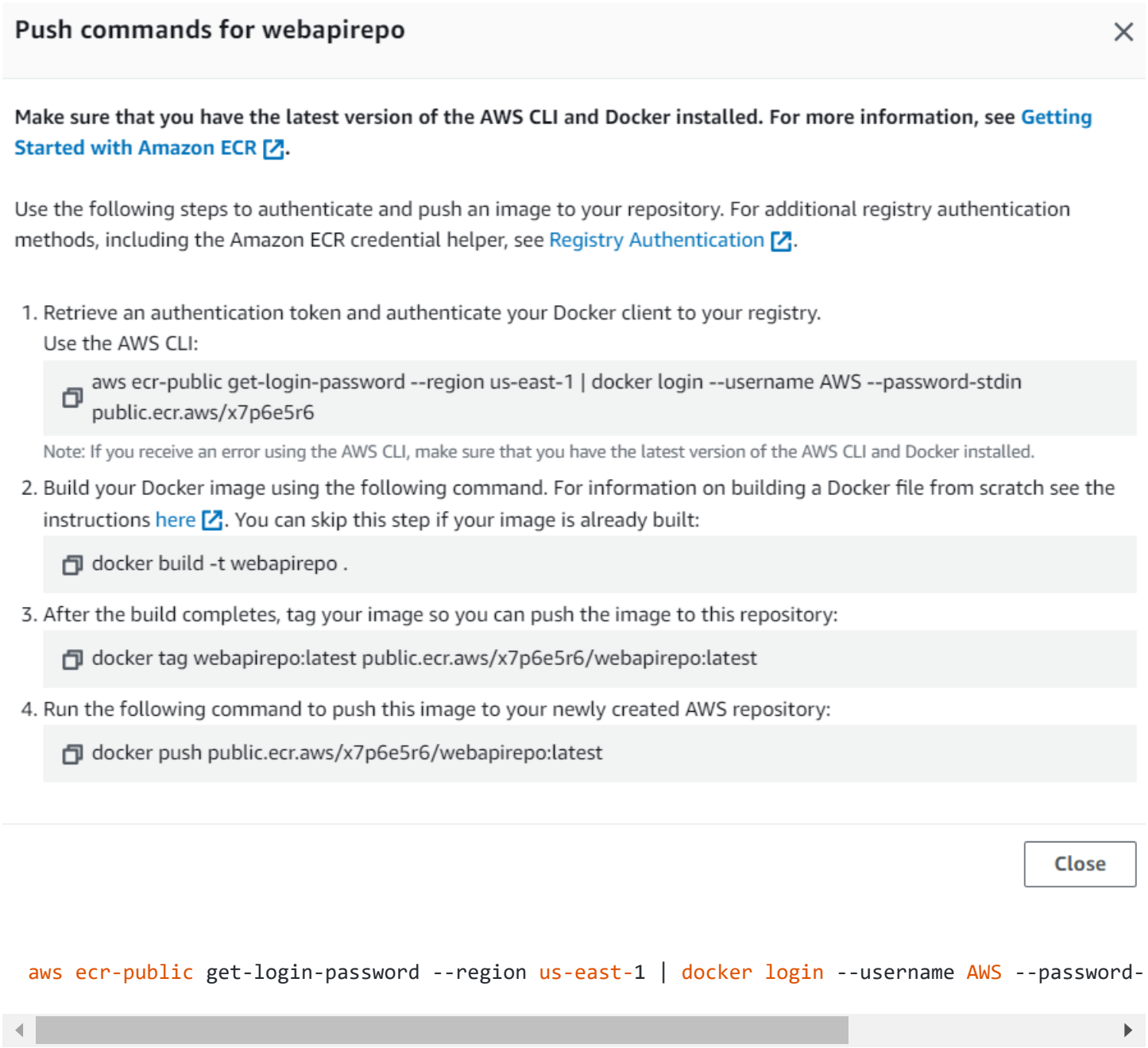



aws
Services
Search
[Alt+S]
Amazon ECR > Private registry > Repositories > Create repository
Create repository
General settings
Visibility settings Info
Choose the visibility setting for the repository.
Private
Access is managed by IAM and repository policy permissions.
Public
Publicly visible and accessible for image pulls.
Once a repository is created, the visibility setting of the repository can't be changed.
Detail
Repository name Info
A namespace can be included with your repository name (e.g. namespace/repo-name).
public.ecr.aws/registry-alias/ webapirepo
10 out of 205 characters maximum (2 minimum). The name must start with a letter and can only contain lowercase letters, numbers, hyphens, underscores, periods and forward slashes.


aws
Services
Search
[Alt+S]
Repository logo - optional Info
Choose a local image file to use as the repository logo.
Upload file
Reset
The supported file format is PNG. The supported image dimensions for both height and width should be a minimum of 60 pixels and a maximum of 2048 pixels. The maximum file size is 500 KB.
Short description - optional
The short description is displayed in search results and on the repository detail page.
0 out of 255 characters maximum.
Content types - optional Info
Select the operating systems and system architectures that are compatible with the images in your repository.
Operating systems
Linux
Windows
Architectures
ARM
ARM 64
x86
x86-64


Services
Search
[Alt+S]
Describe this repository
0 out of 10,240 characters maximum. Use GitHub Flavored Markdown format for the text. Learn more
Preview
Usage - optional Info View example
Provide detailed information about how to use the images in the repository. This provides context, support information, and additional usage details for users of the repository.
Usage information
0 out of 10,240 characters maximum. Use GitHub Flavored Markdown format for the text. Learn more
Preview
Cancel
Create repository

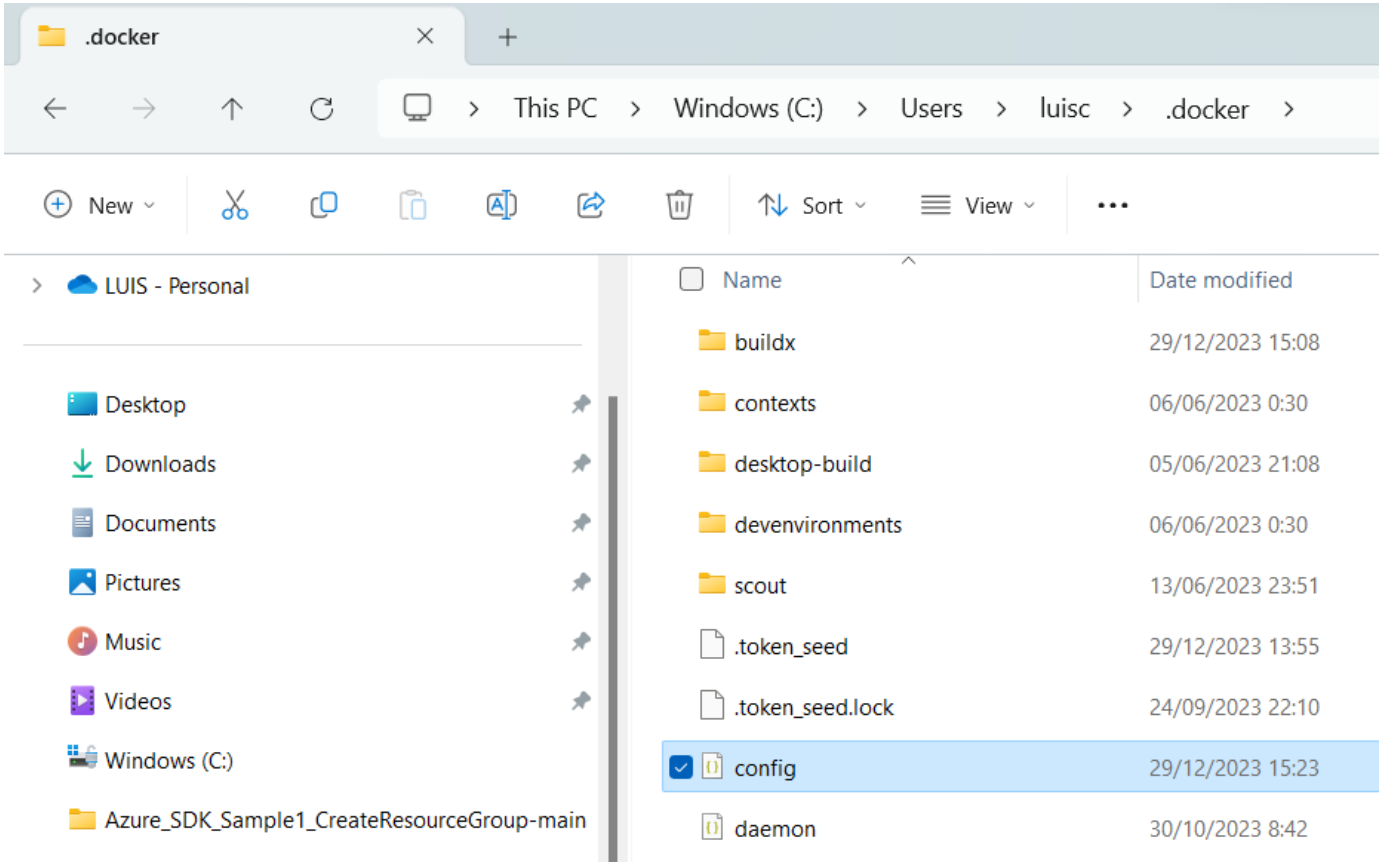

Services
Search
[Alt+S]
Paris
Amazon Elastic Container Registry
Created public repository
webapirepo has been successfully created in public registry
Private registry
Repositories
Settings
Public registry
Repositories
Settings
ECR public gallery
Amazon ECS
Amazon ECR > Public Registry > Repositories
Public repositories
Repositories (1)
View push commands
Delete
Actions
Create repository
Filter status
Repository name
URI
Created at
webapirepo
public.ecr.aws/x7p6e5r6/webapirepo
December 29, 2023, 21:17:46 (UTC+01)




```
aws ecr-public get-login-password --region us-east-1 | docker login --username AWS --password-
```

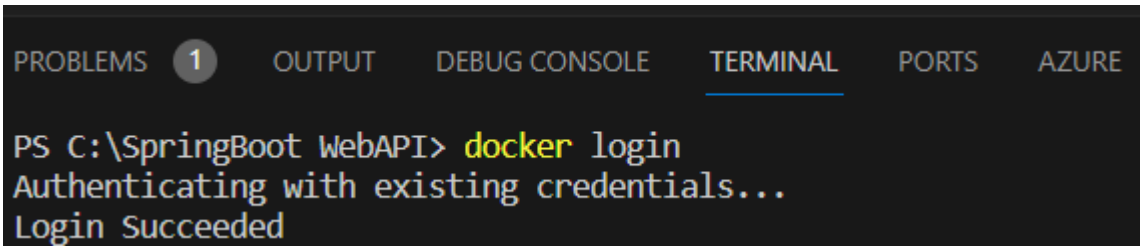
IMPORTANT: if you cannot enter in the AWS ECR follow these steps

- Delete **config.json** file in the following location:

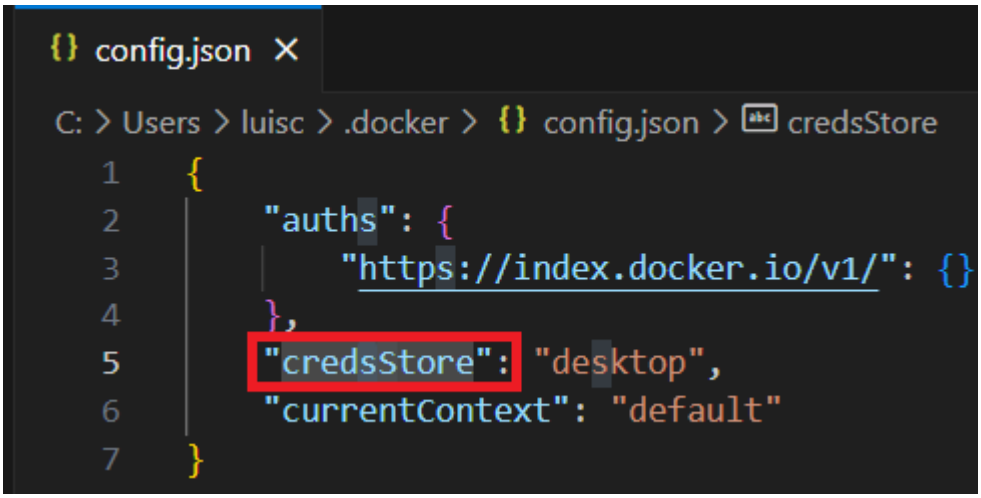


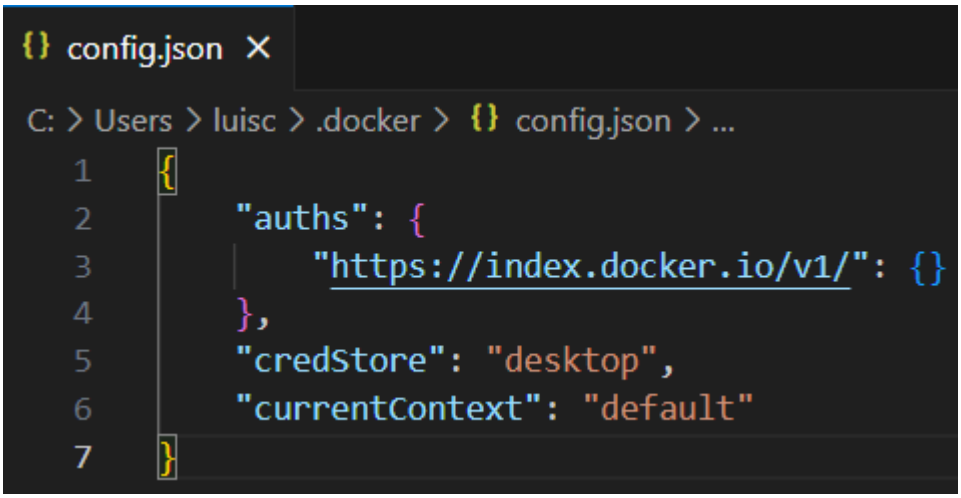
- Type the command:

```
docker login
```

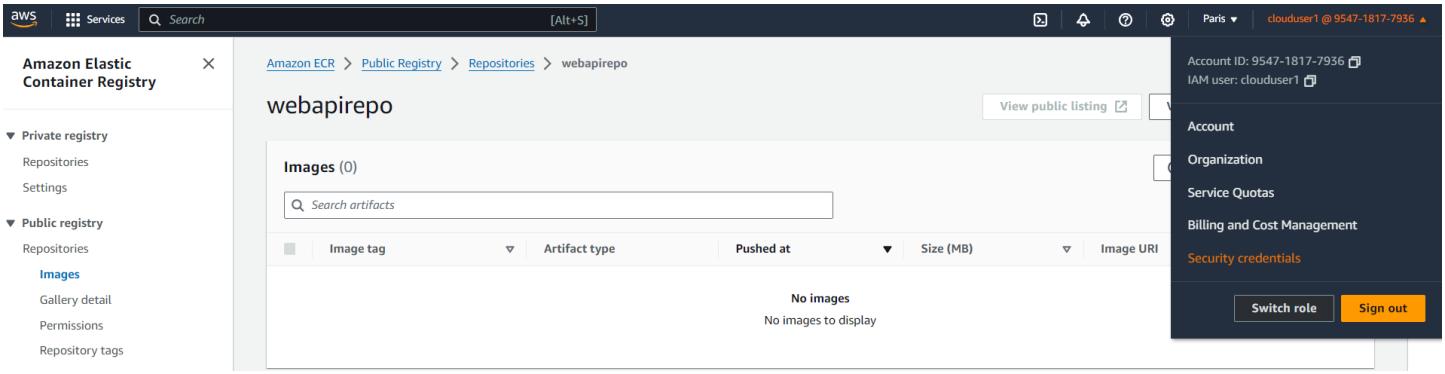


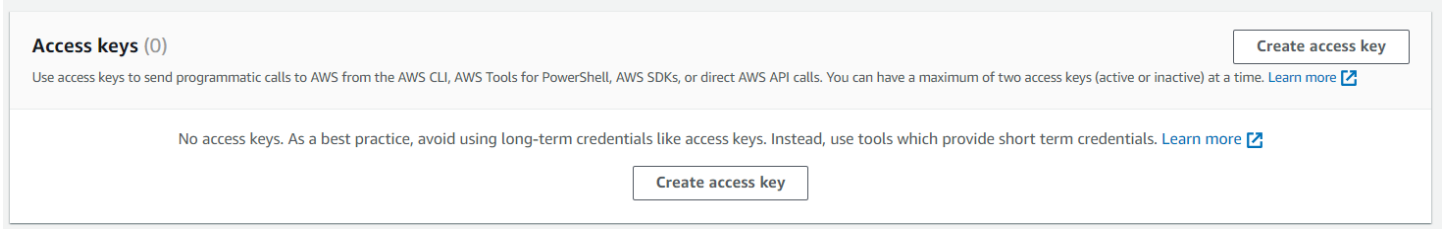
- Delete the letter "s" the "credsStore": "desktop" the result is this word: "credStore": "desktop"

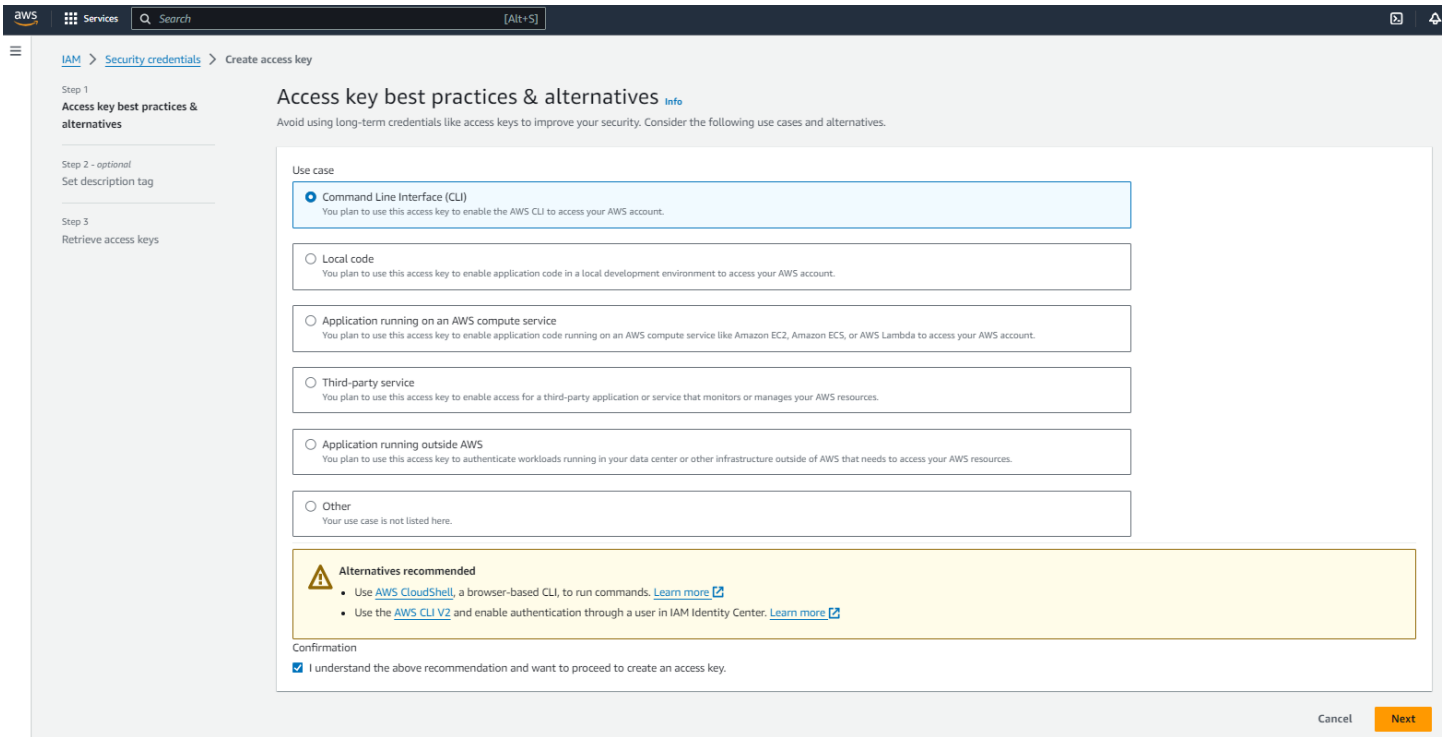


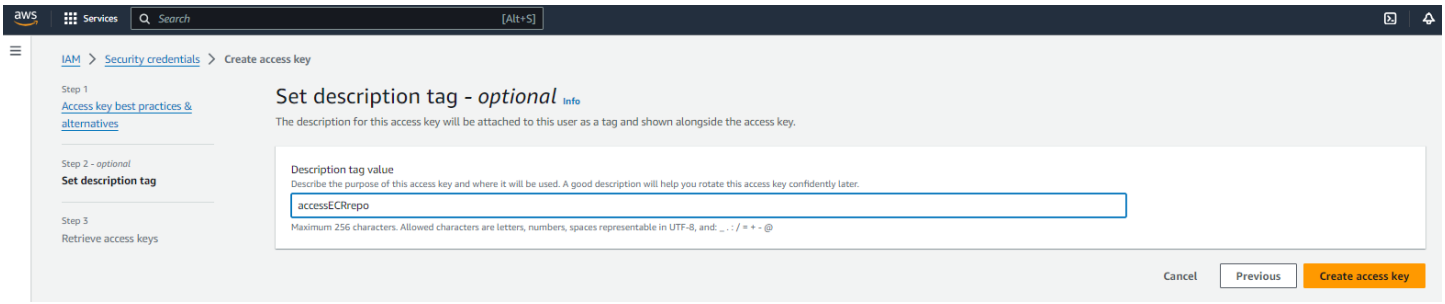


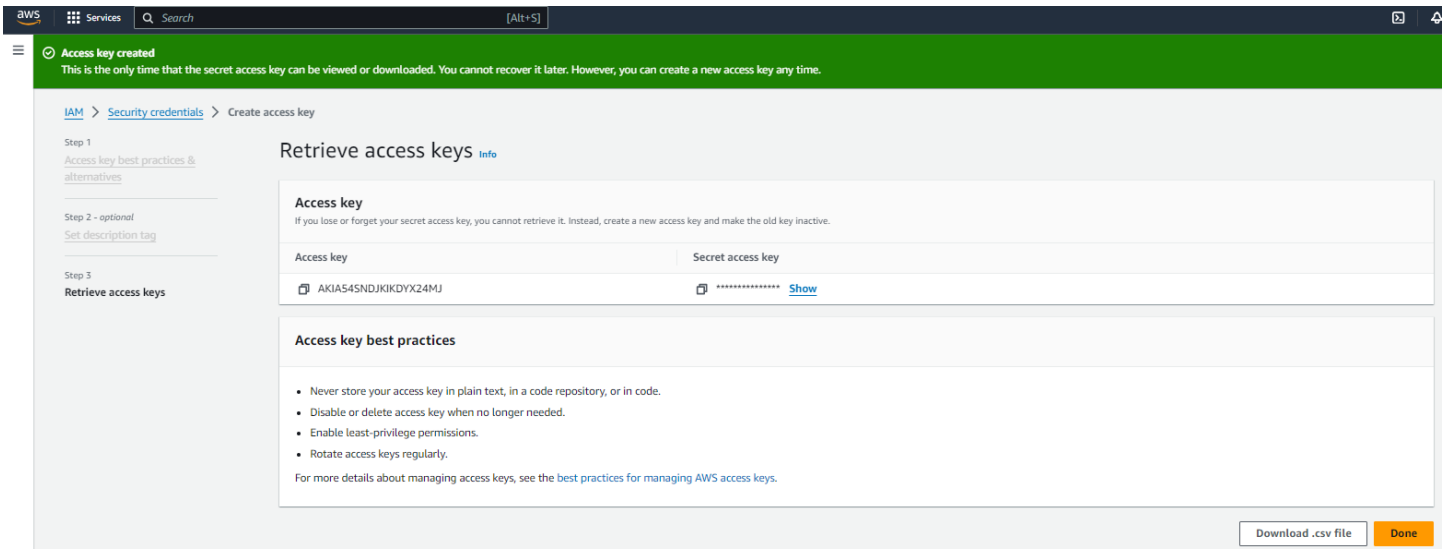
- Configure AWS account with access key and secret key

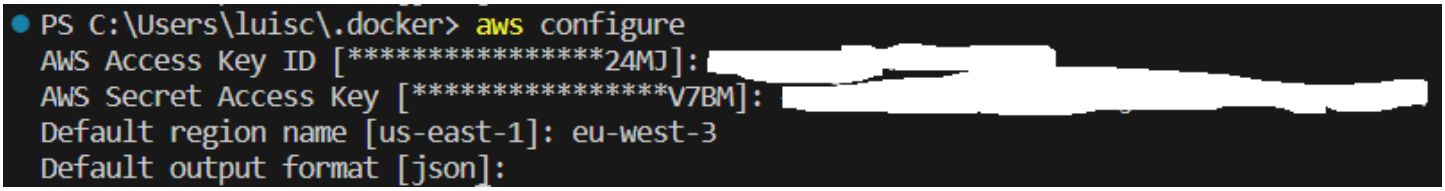






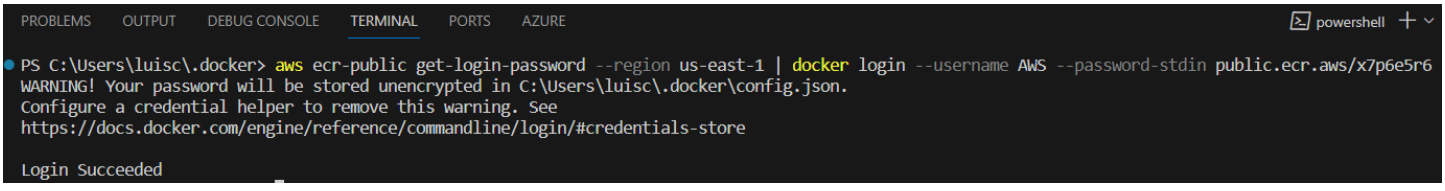






We try to enter now again in the AWS ECR with the following command:

```
aws ecr-public get-login-password --region us-east-1 | docker login --username AWS --password-
```



- Now we can create the SpringBoot WebAPI Docker image with this command

```
docker build -t webapirepo .
```

- Rename the Docker image for pushing it to the AWE ECR repo

```
docker tag webapirepo:latest public.ecr.aws/x7p6e5r6/webapirepo:latest
```

- Also we can verify the application docker image running it in our local Docker Desktop with this command:

```
docker run -p 80:80 --name myapp-container webapirepo:latest
```

- Verify the application endpoints:

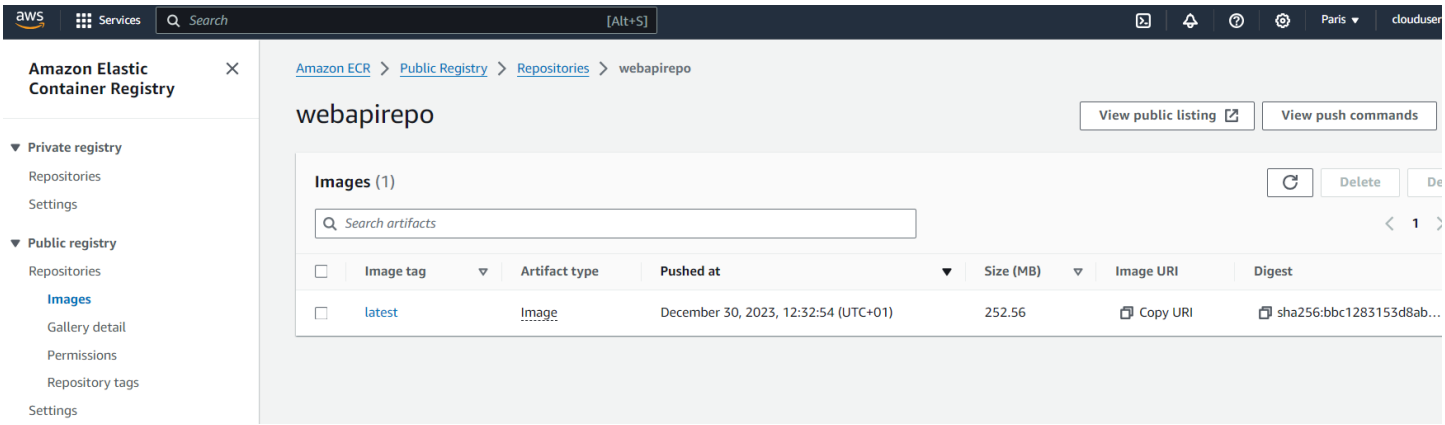
http://localhost:80/hello

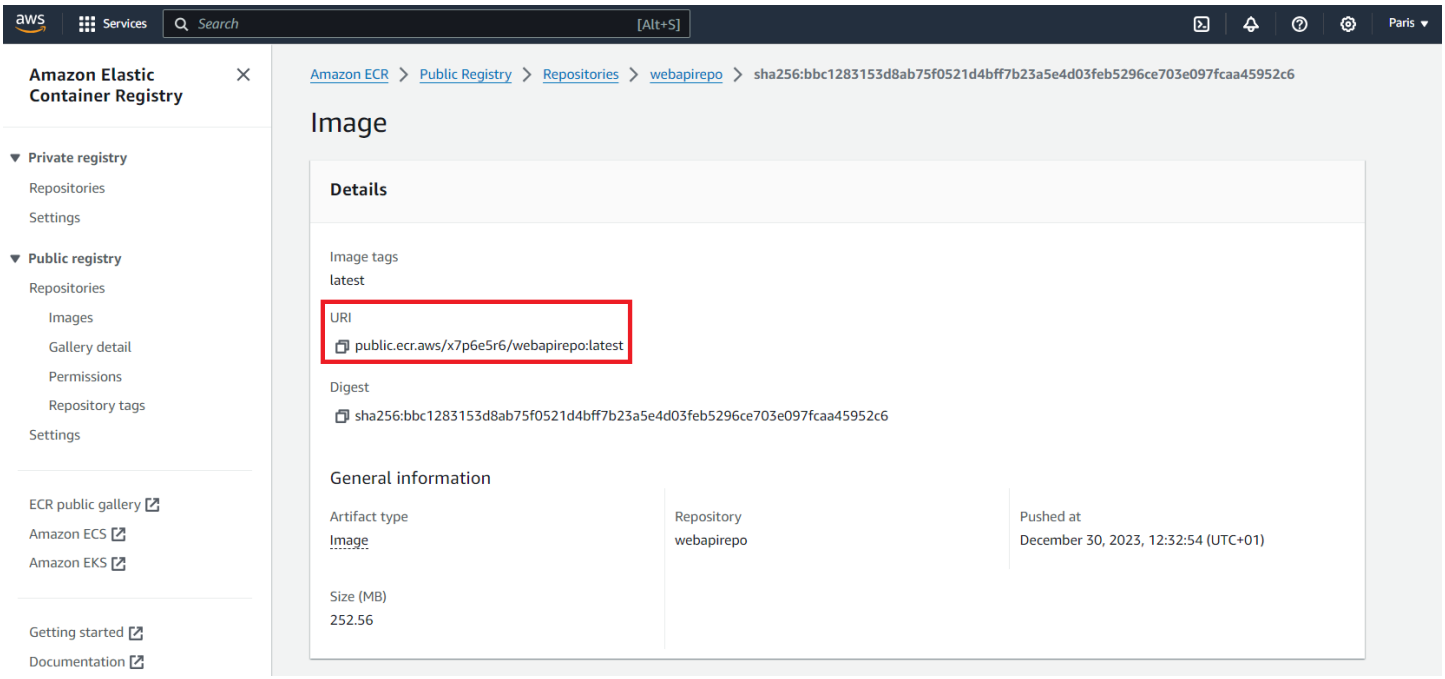
http://localhost:80/actuator/health

4. Push the Docker image to AWS ECR

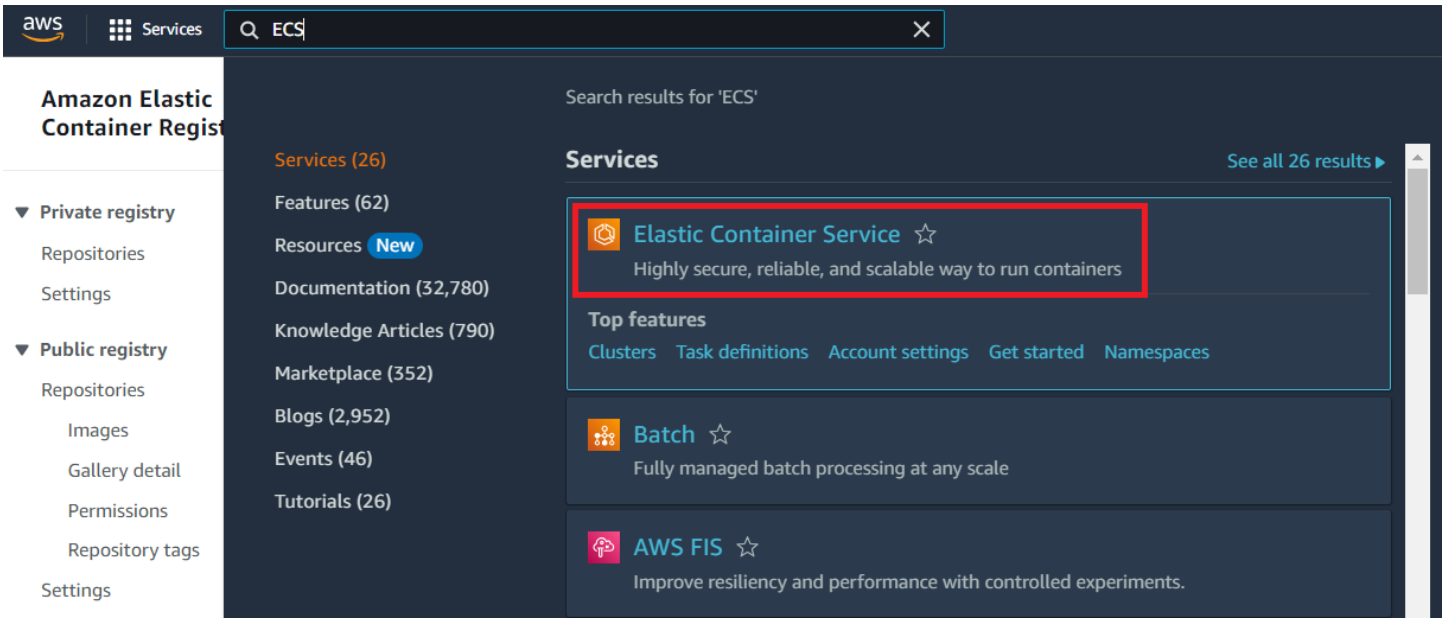
Push the Docker image to AWS ECR

```
docker push public.ecr.aws/x7p6e5r6/webapirepo:latest
```



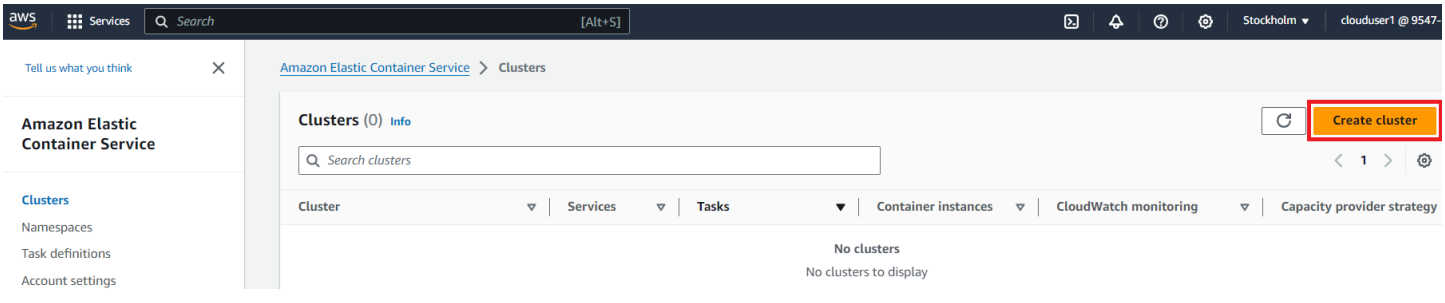


5. Deploy Docker image from AWS ECR to AWS ECS

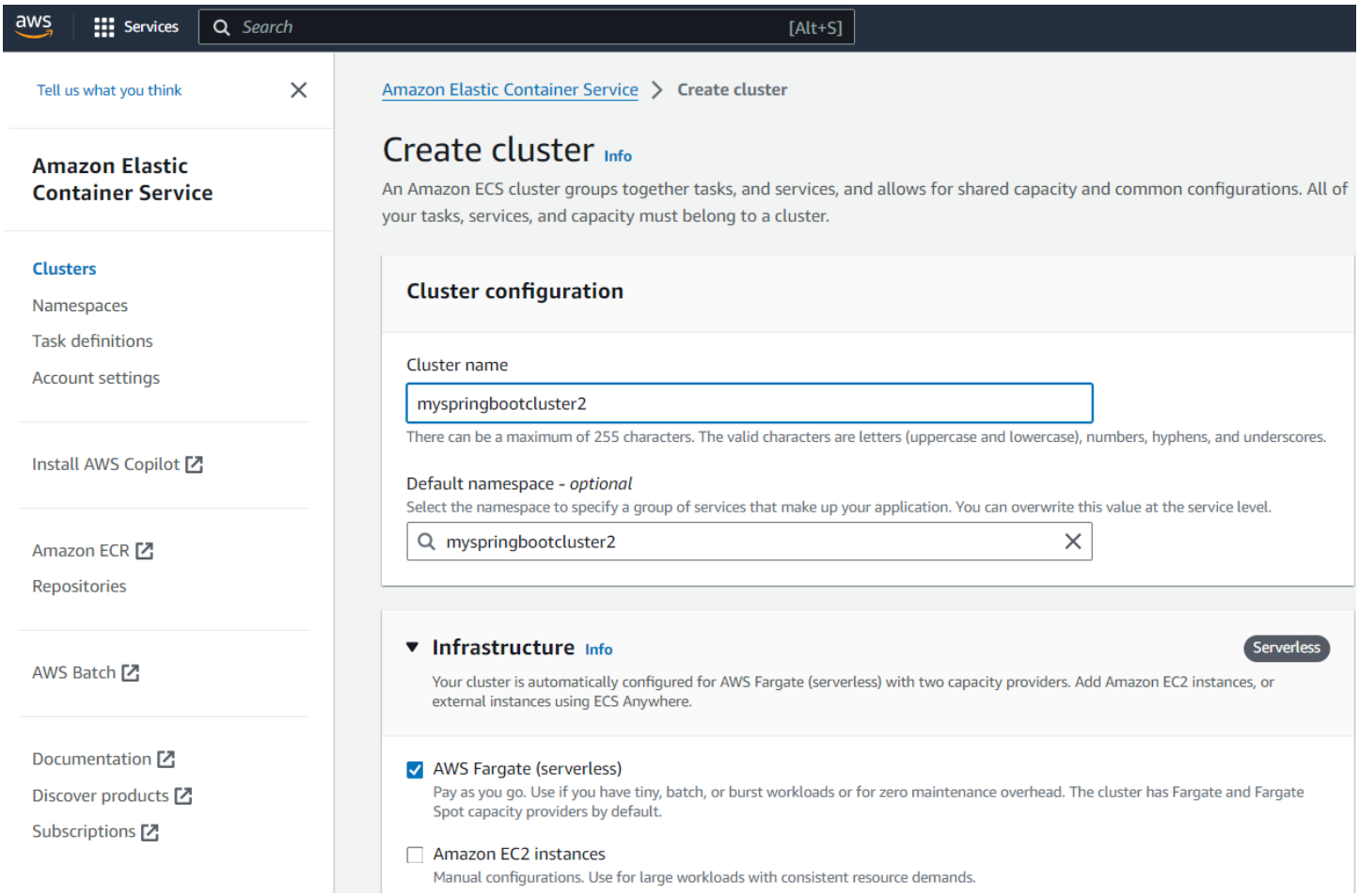


5.1. Create a AWS ECS cluster

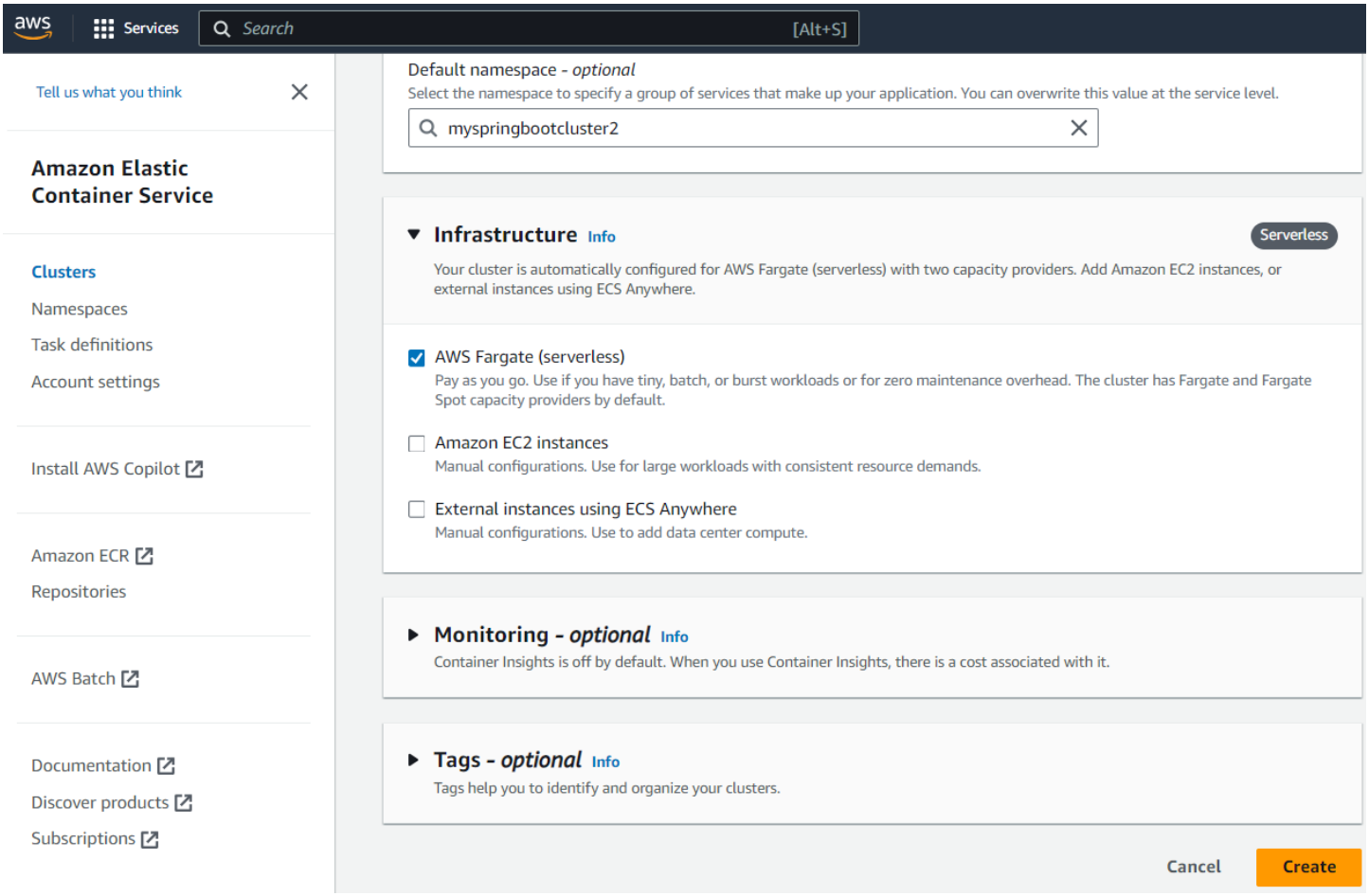
We press in the **Create cluster** button:

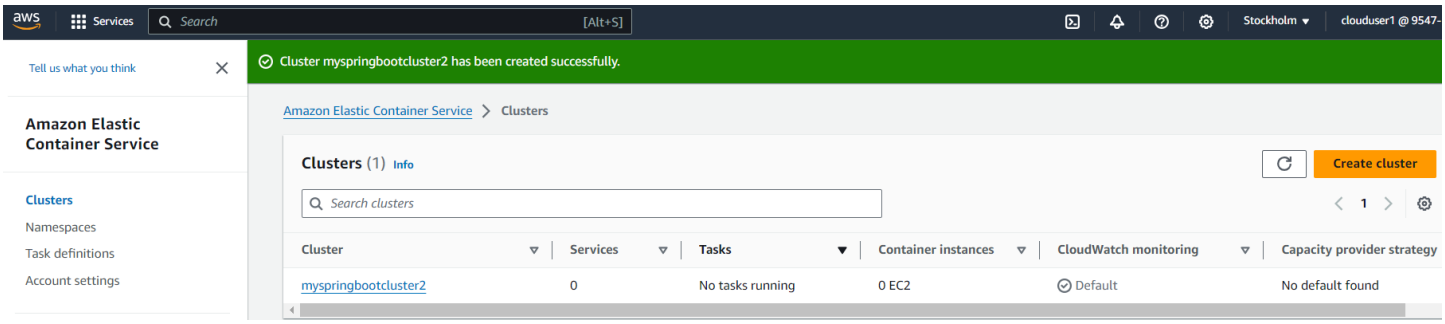


We set the AWS Cluster name



We leave the **Fargate** option checked and we press the **Create** button

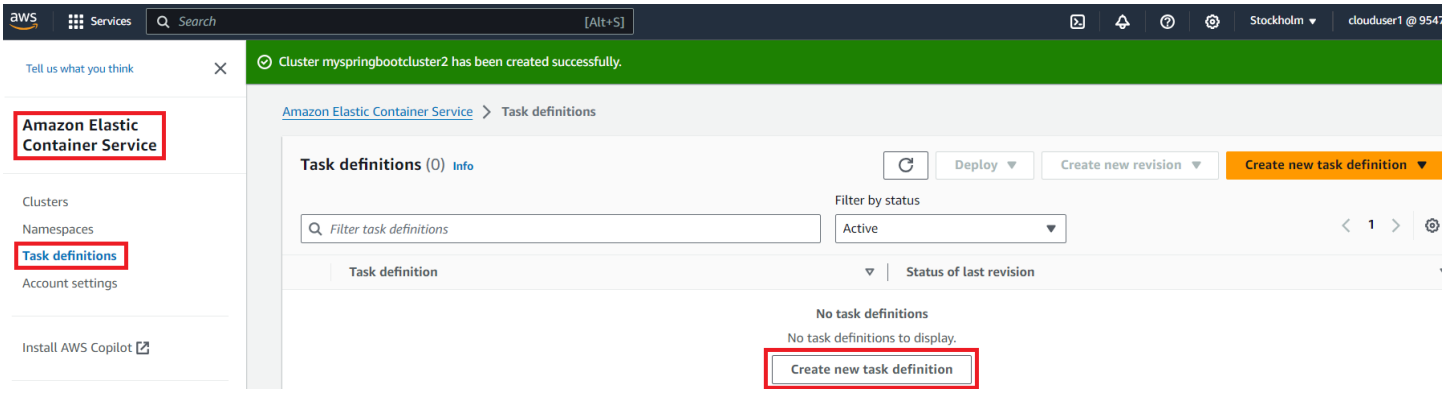




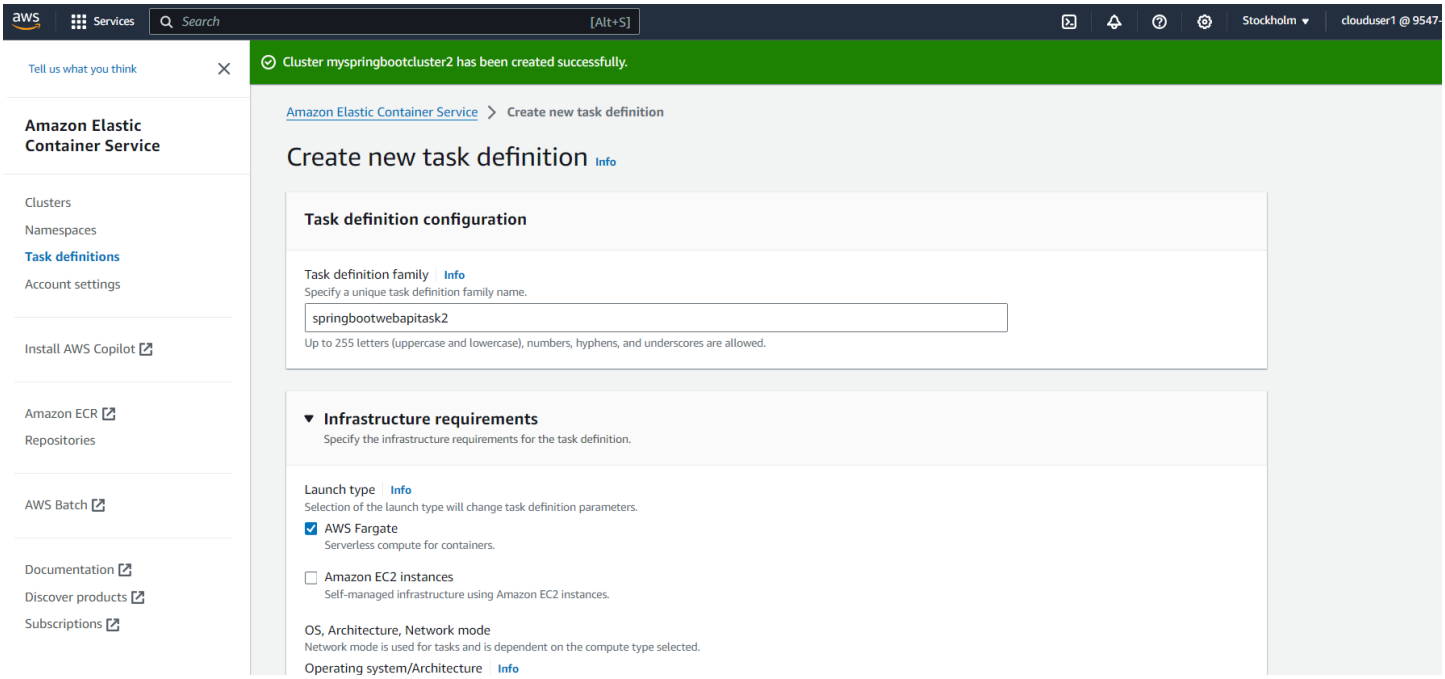
Note: we selected the eu-north-1 region but the more advisable solution is select eu-west-3 because it is the nearest to our location in Spain. But today I had some technical problems creating cluster in Paris region and I decided to slect the Stockholm one.

5.2. Create a Task Definition

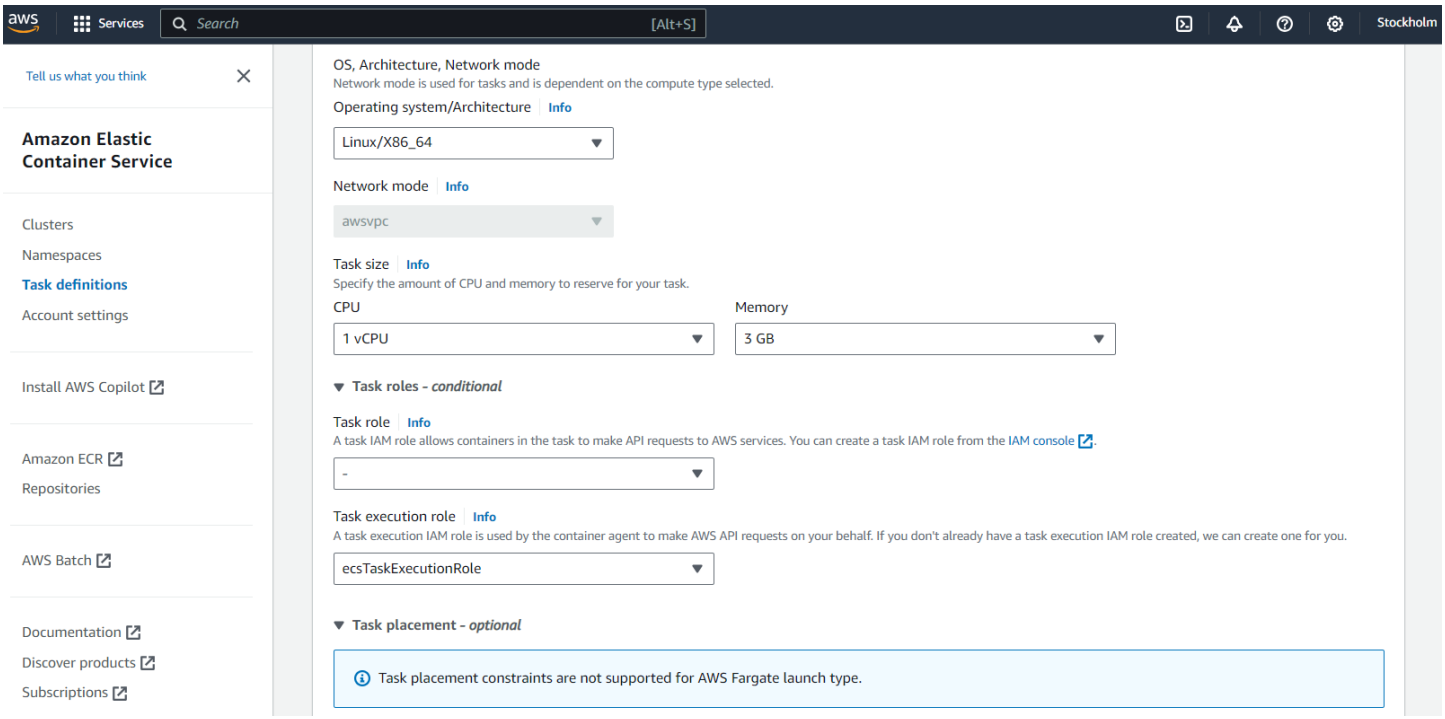
We press the **Create new task definition** button



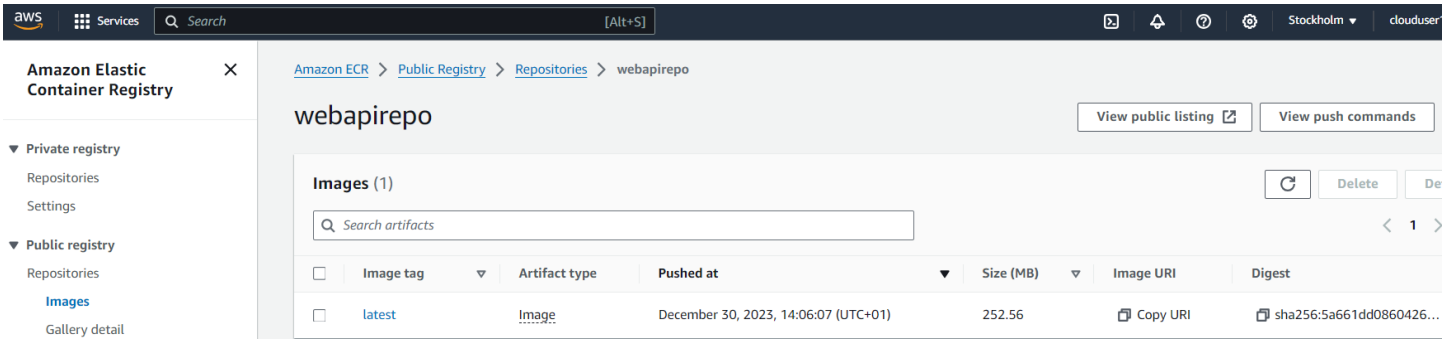
We select the infrastructure option **Fargate**

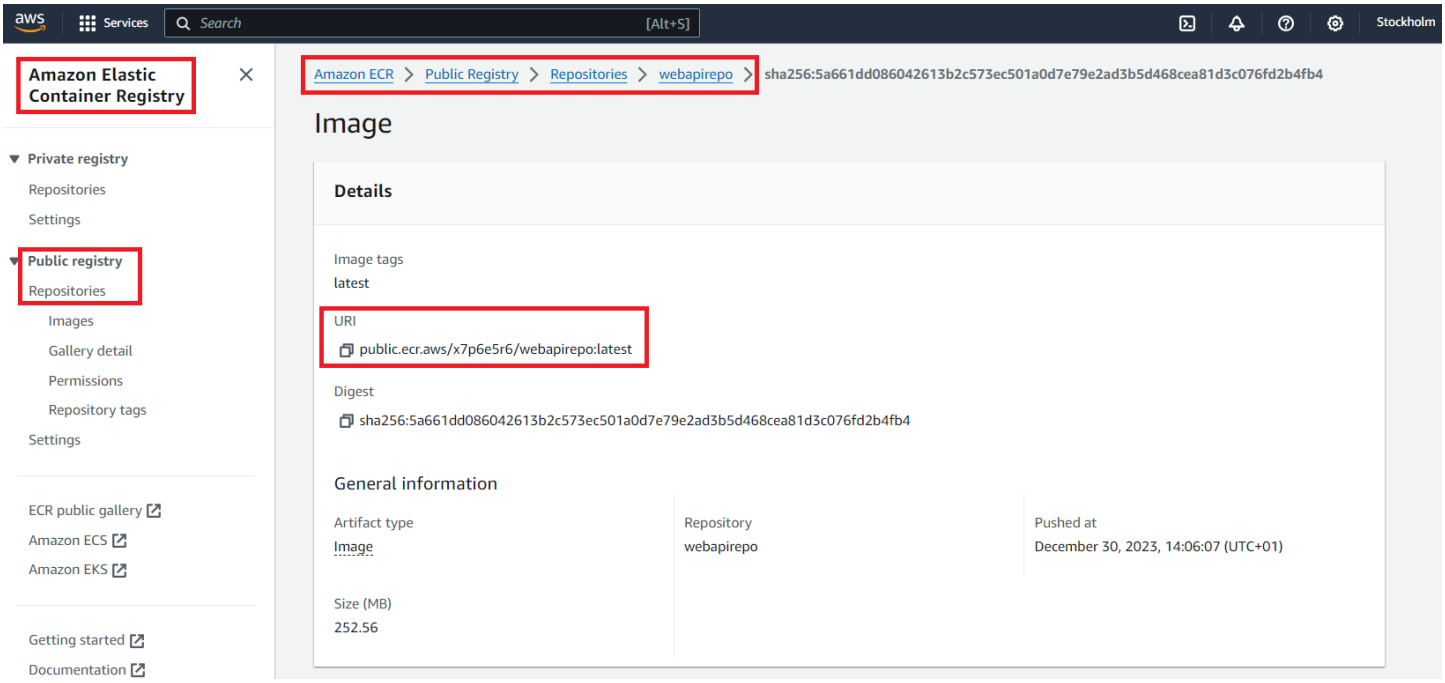


We leave the rest of options as they are by default

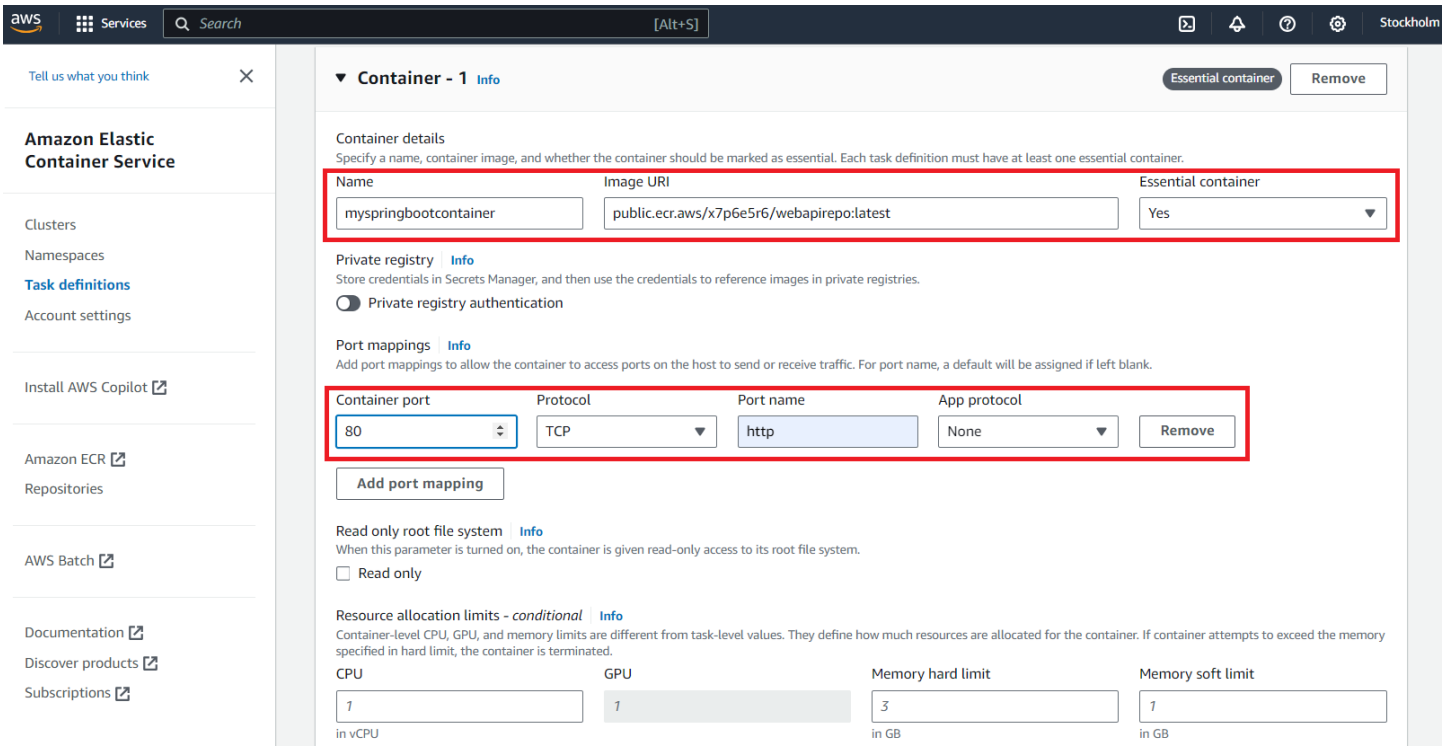


We have to set container image name

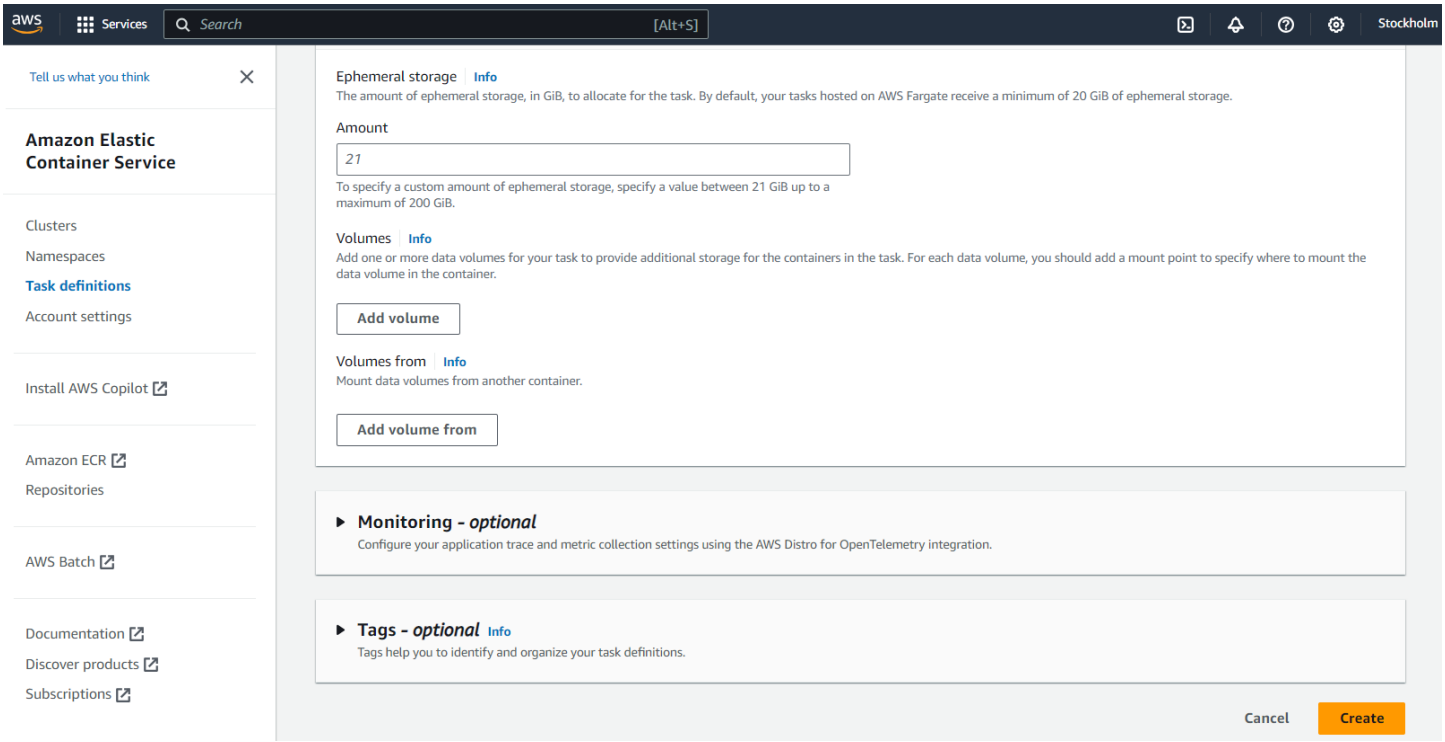


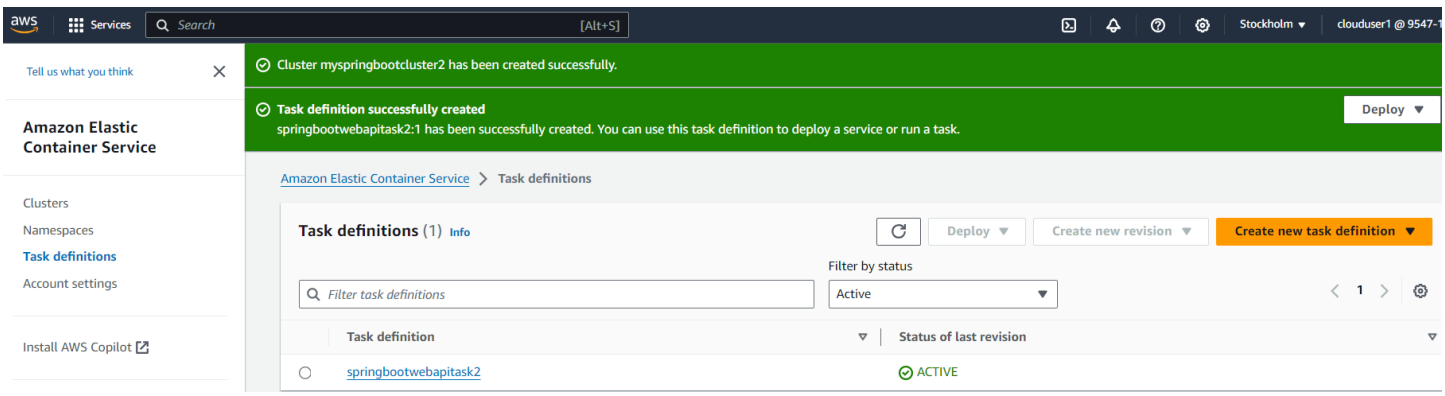


We set the container URL and the port mapping

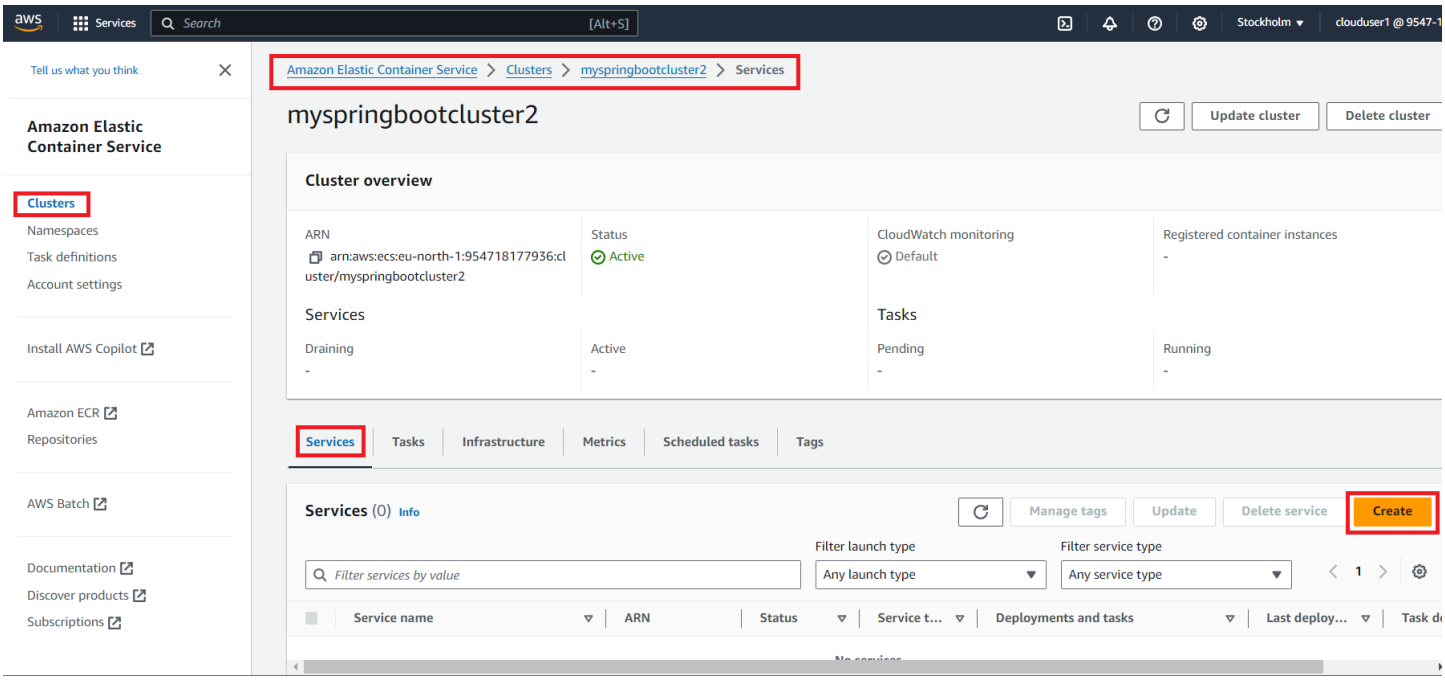


We leave the rest of options as they are by default and we press the ****

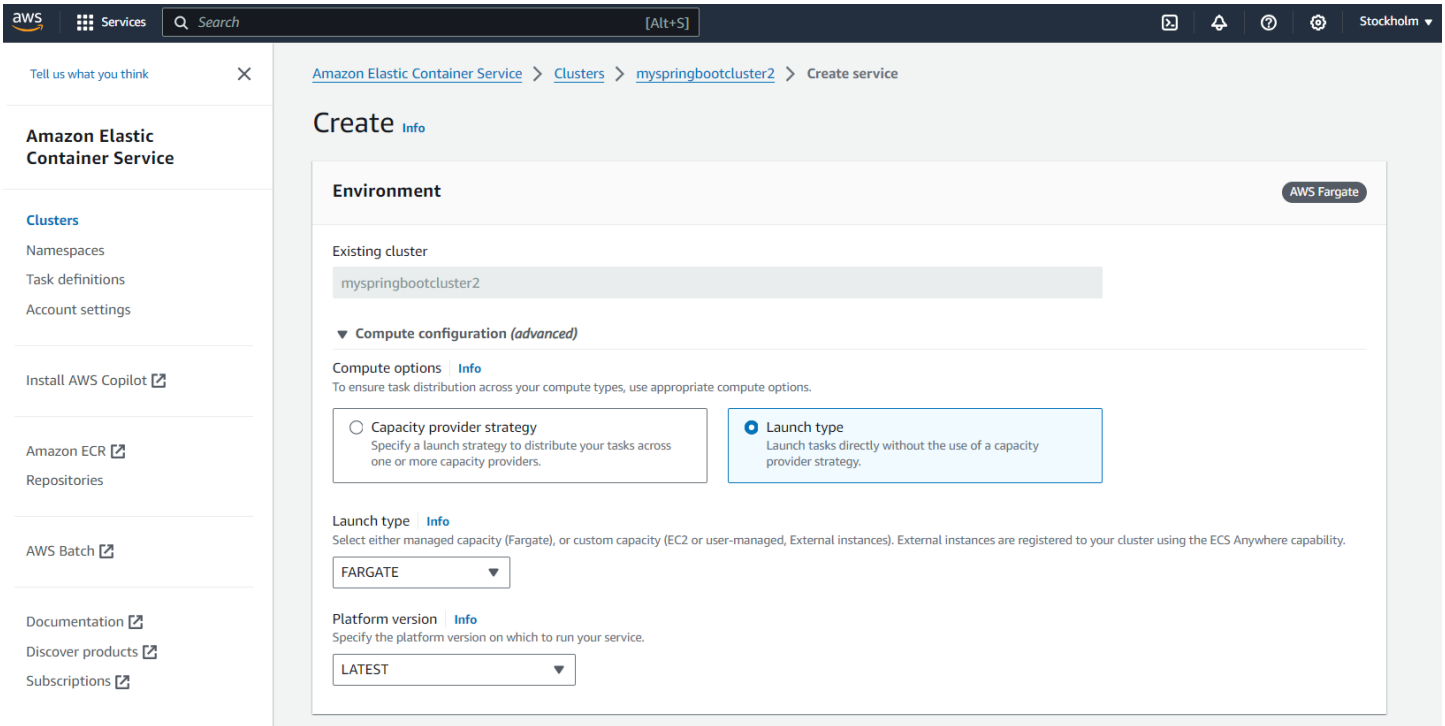


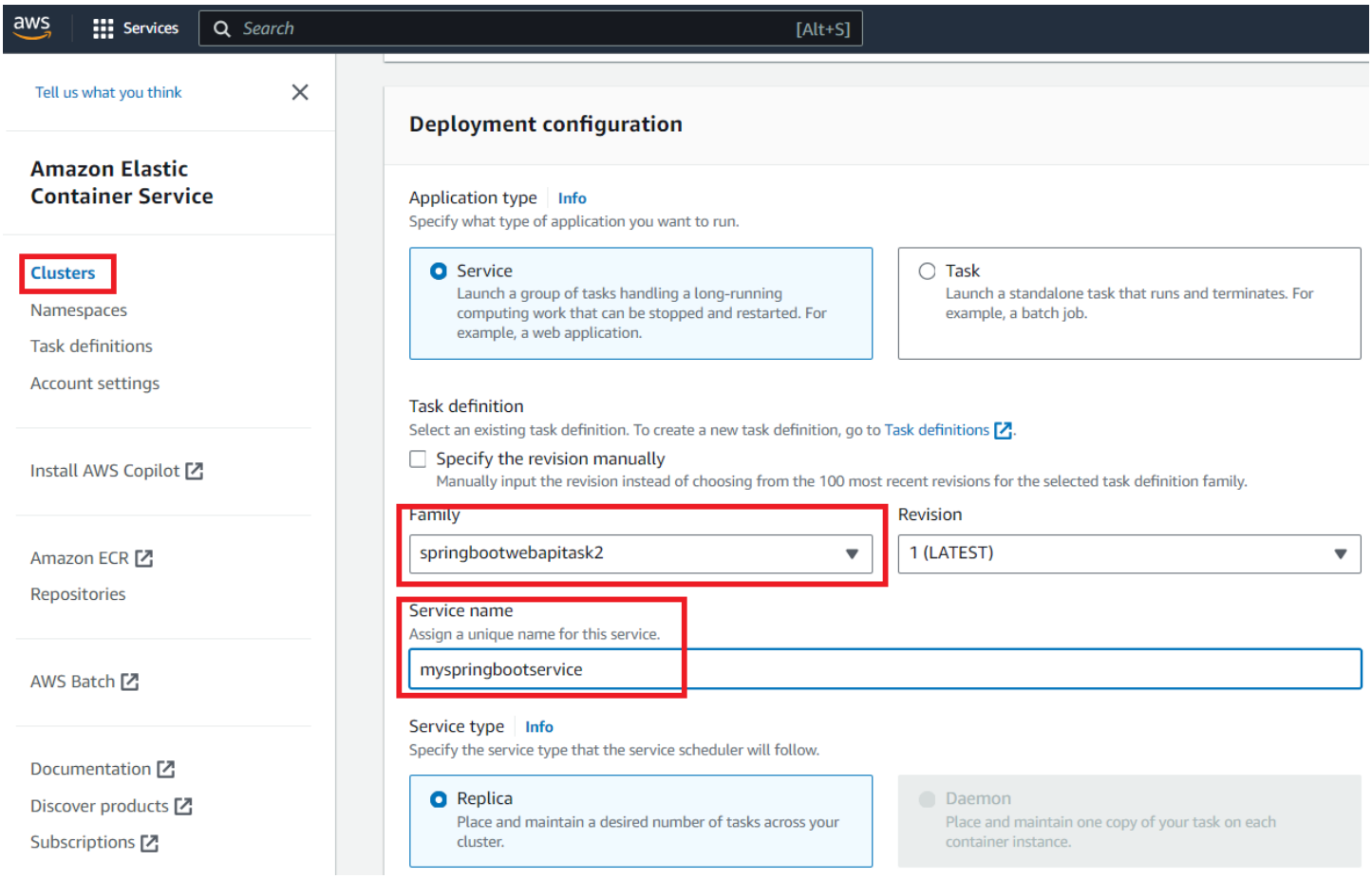


5.3. Create a Service

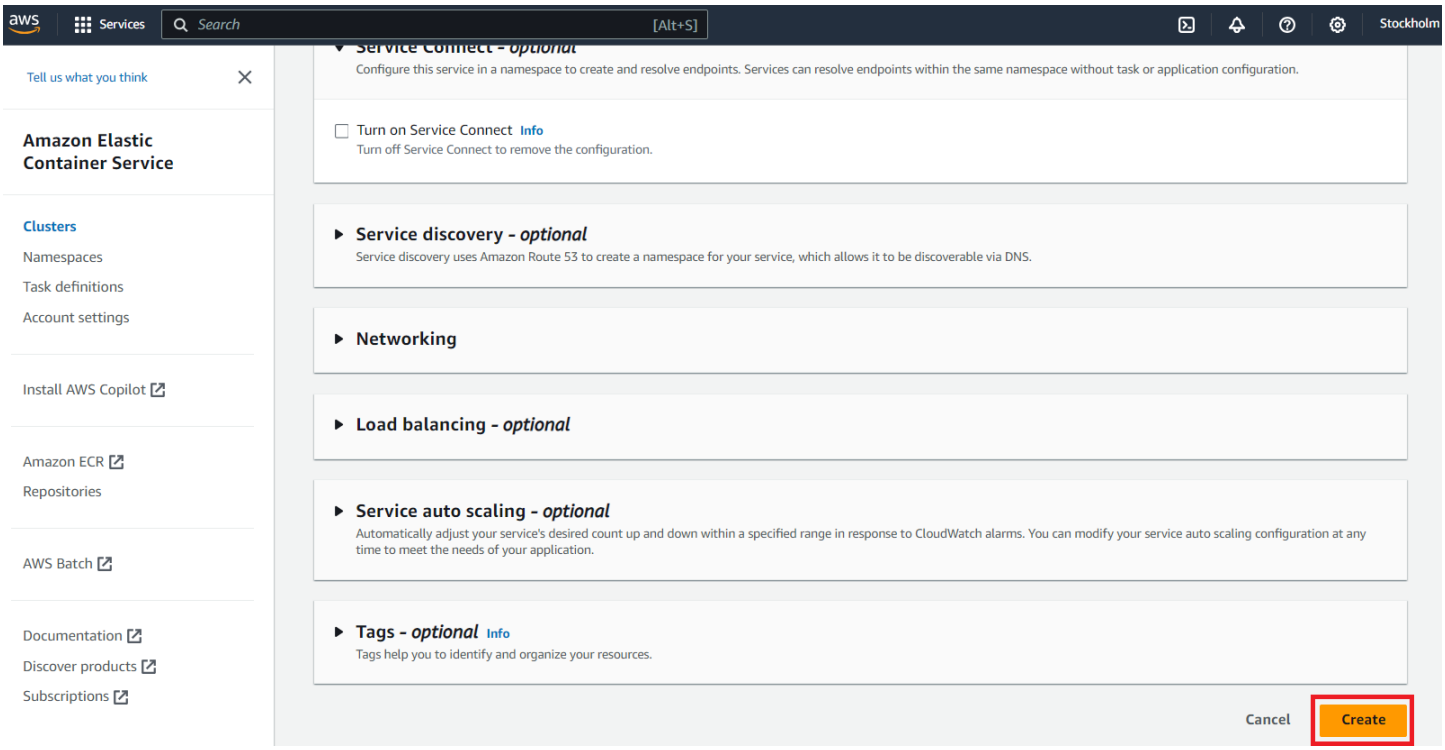


We select the launch type **Fargate**

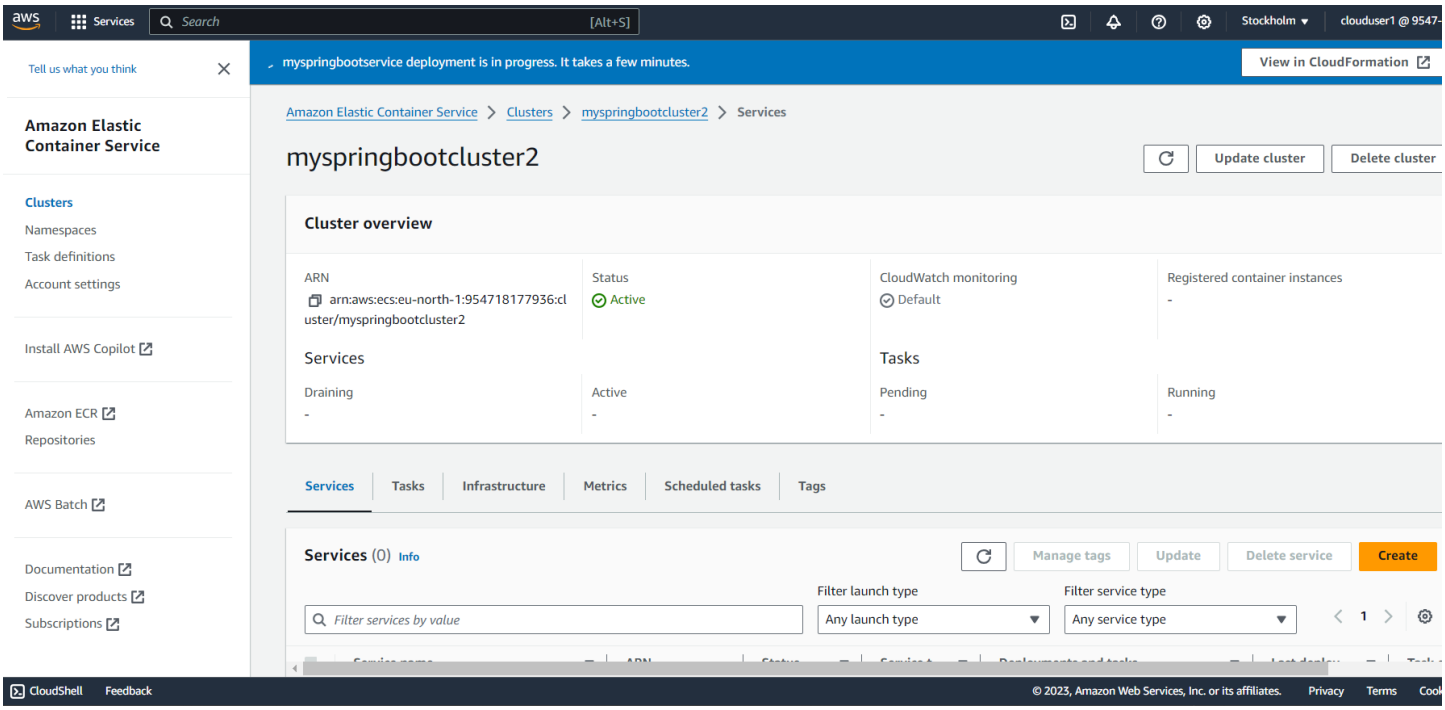


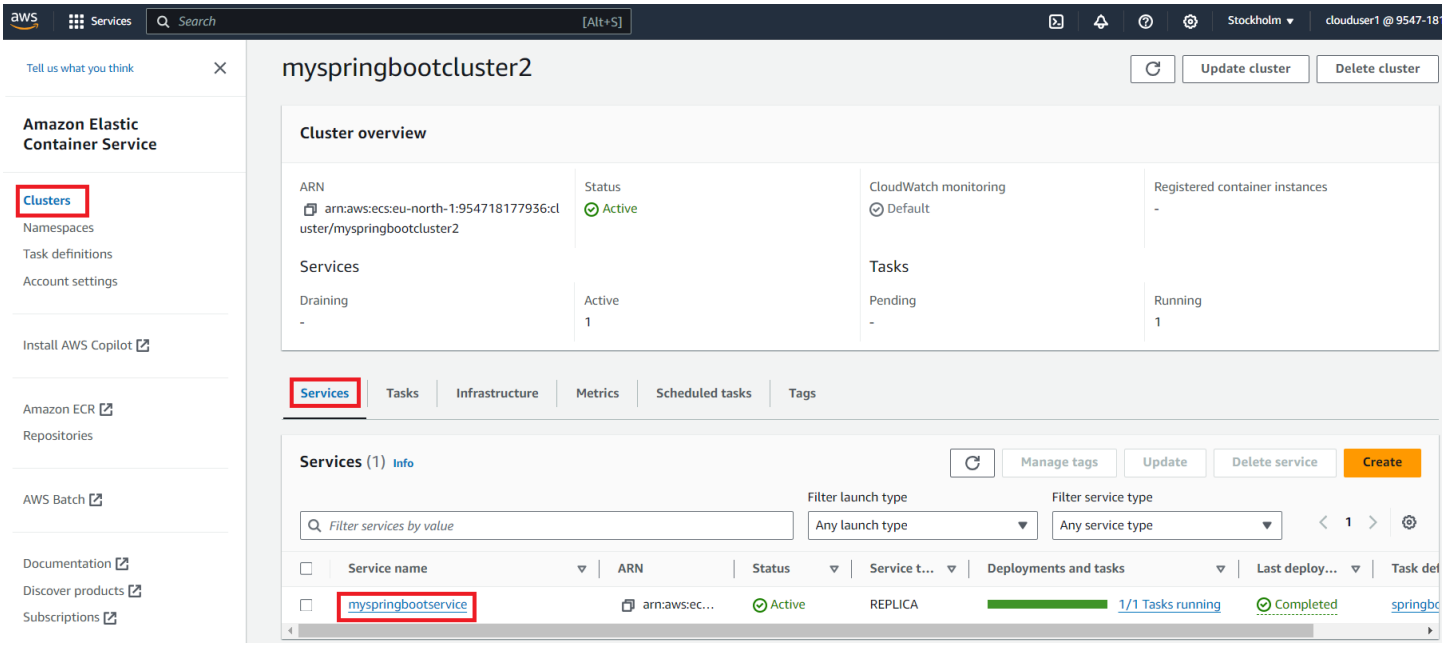


We press the **Create** button

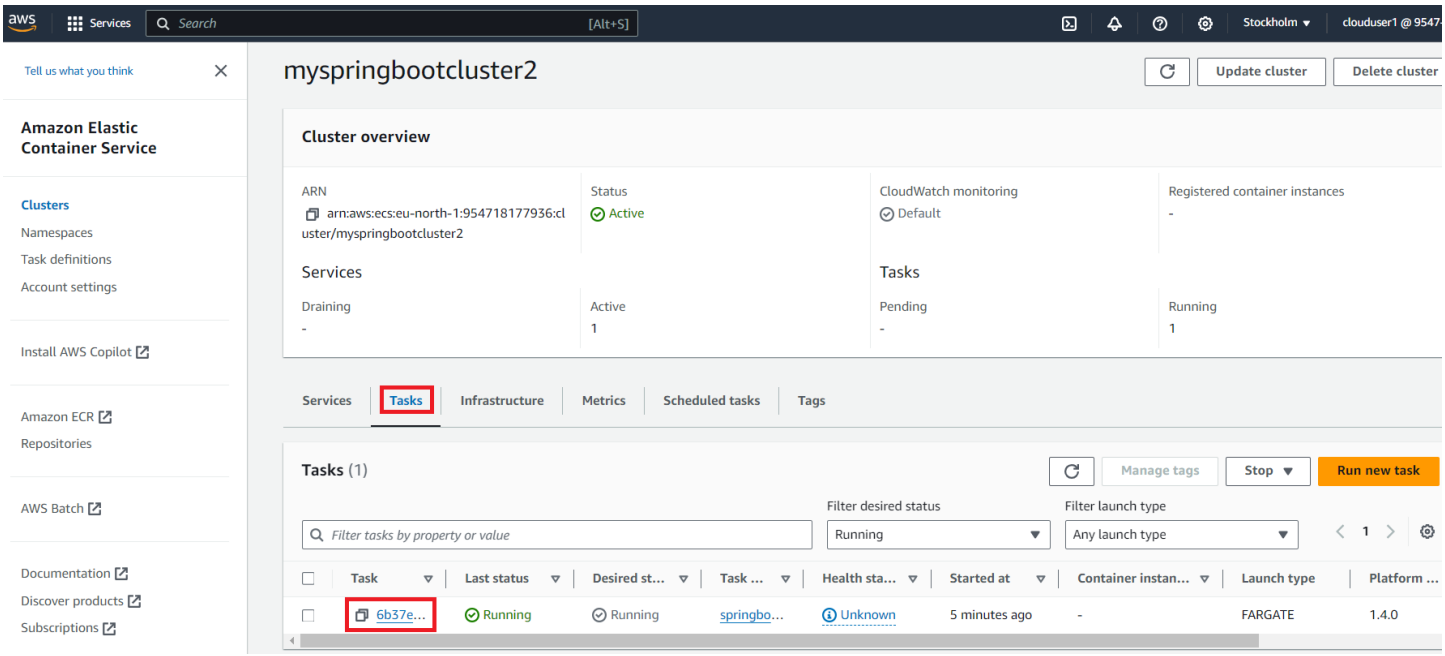


It takes few minutes to create the new Service



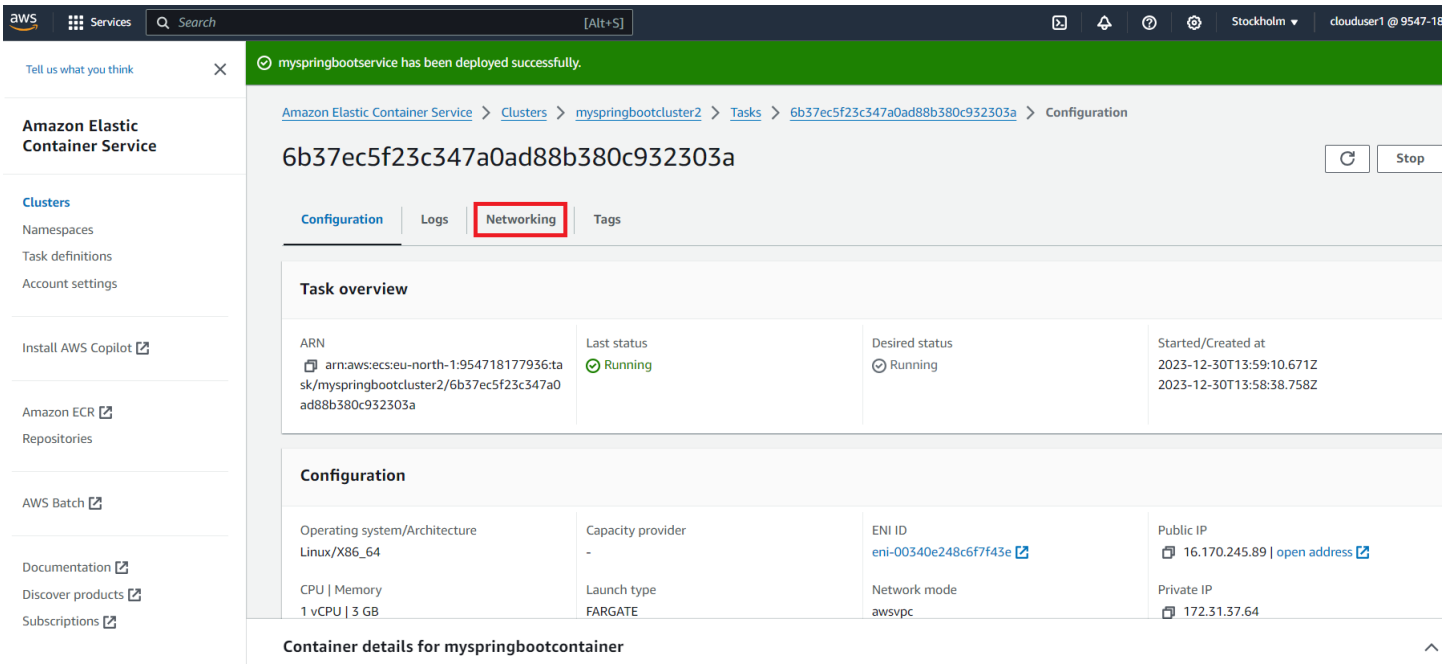


Now we select the **Tasks** tab and we press on the task

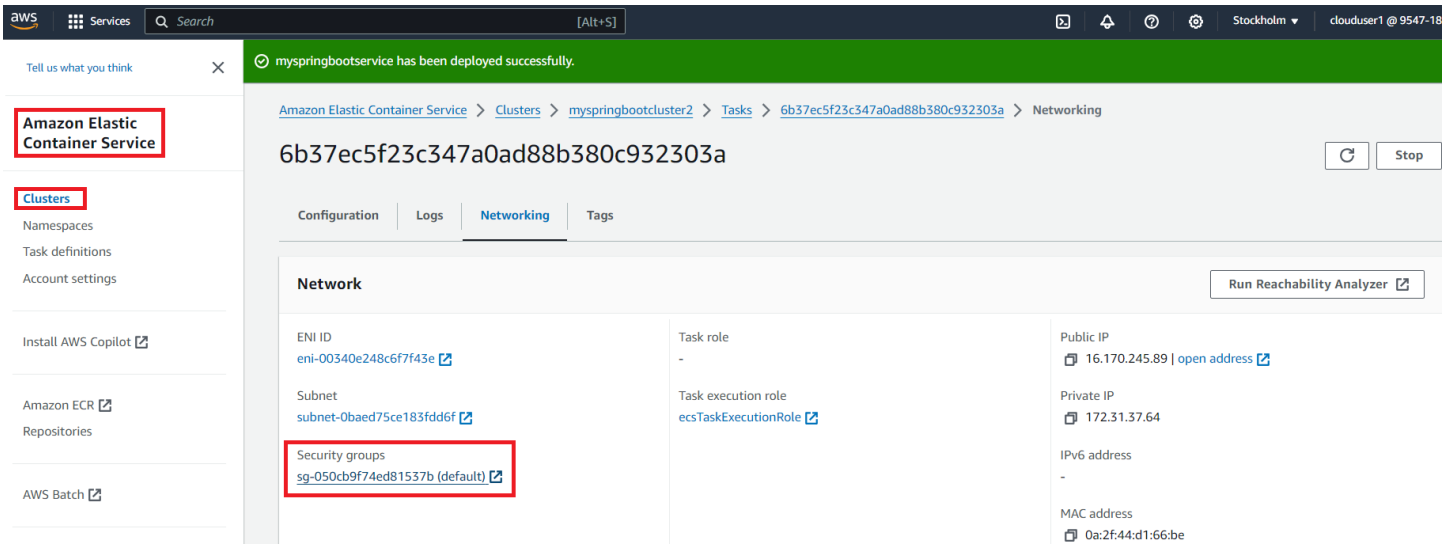


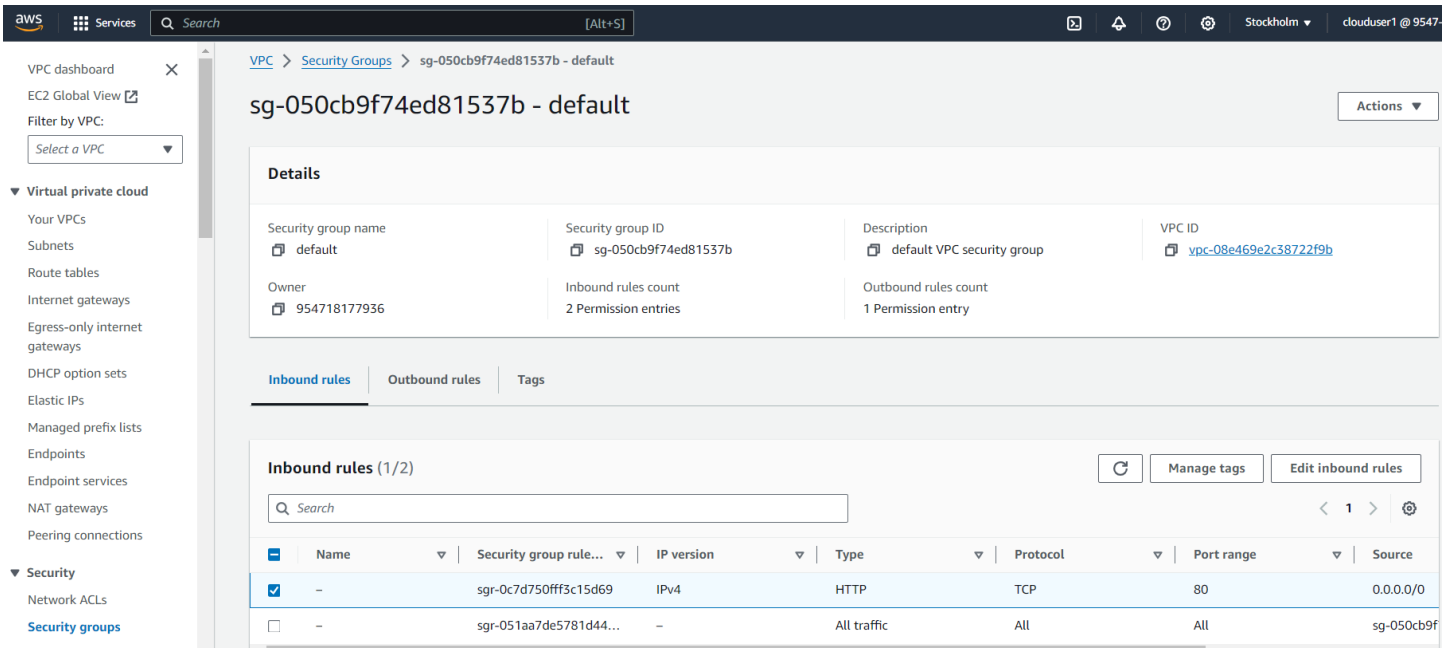
5.4. How to create a new inbound rule to allow traffic on port 80

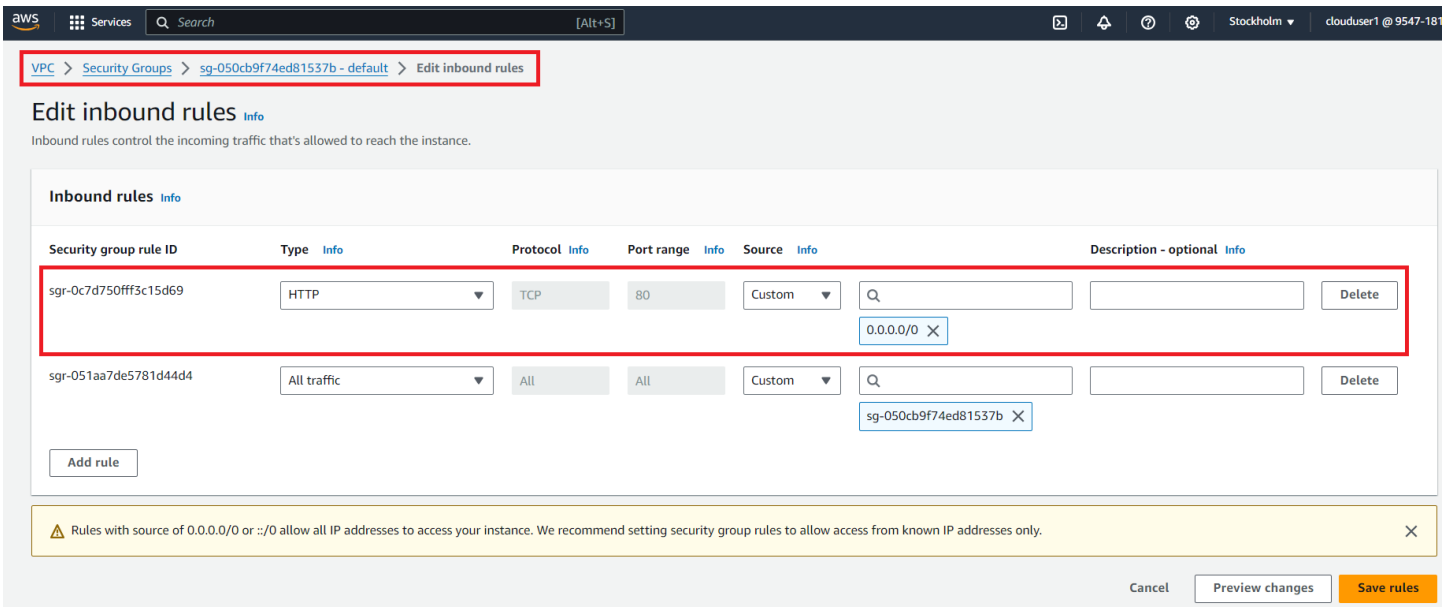
We press in the **Networking** tab



We select the Security Group

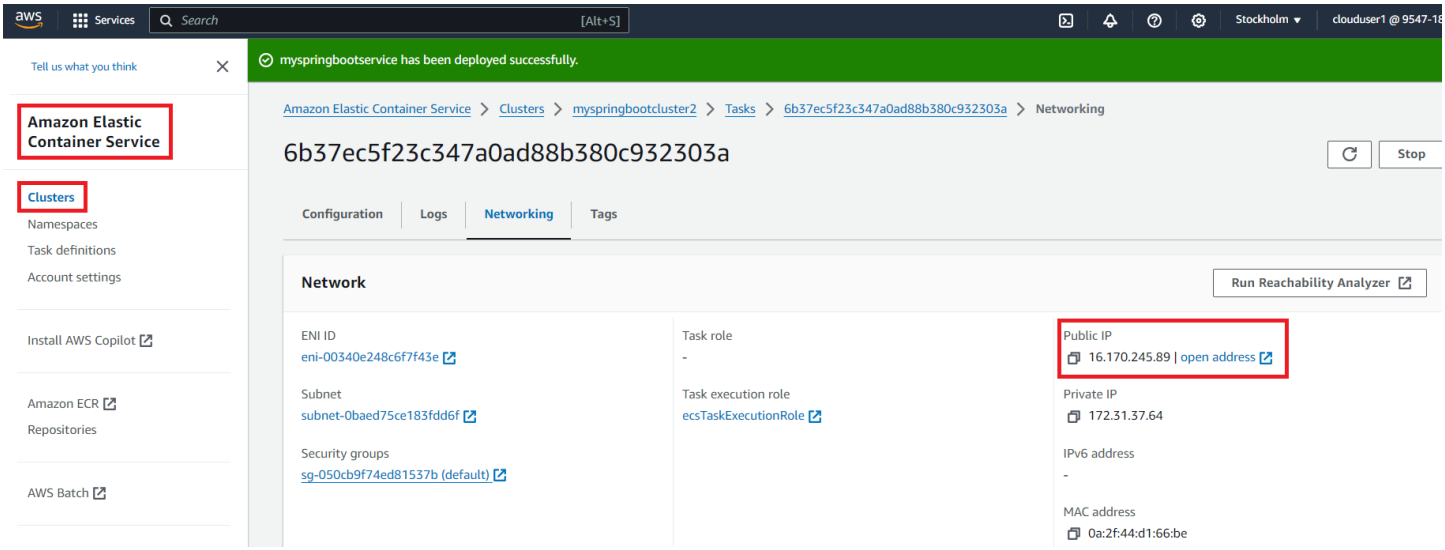






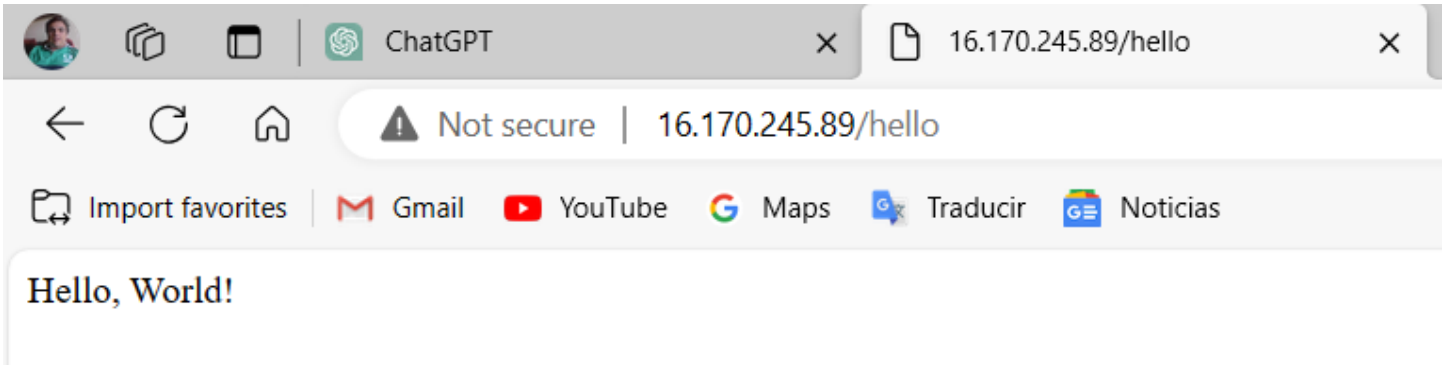
5.5. Verify the application endpoints

We copy the Public IP address and we check the application endpoints



These are the application endpoints:

http://16.170.245.89/hello



http://16.170.245.89/actuator/health