Requierments:

1. https://dotnet.microsoft.com/en-us/download/dotnet/6.0
2. https://desktop.docker.com/win/main/amd64/Docker%20Desktop%20Installer.exe
3. https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html
    a. `msiexec.exe /i https://awscli.amazonaws.com/AWSCLIV2.msi`
    b. `aws --version`

- dotnet new webapi – create a new webapi command
- dotnet run – for verification

```
PS C:\Users\Ionel\Desktop\aws> dotnet run
Building...
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: https://localhost:7219 => this link + /swagger
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5217
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\Users\Ionel\Desktop\aws\
```

- Remove the if block containing useSwagger in Program.cs – this is so our app will be deployed with swagger so we can see the endpoints

```
var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.
//if (app.Environment.IsDevelopment()) - THIS LINE
//{                       - THIS LINE
    app.UseSwagger();
    app.UseSwaggerUI();
//}- THIS LINE
```

- dotnet publish --configuration Release – build the Release

Create a docker file with the following content

```
FROM mcr.microsoft.com/dotnet/aspnet:6.0 AS base
WORKDIR /app
EXPOSE 5000

ENV ASPNETCORE_URLS="http://+:5000"
ENV ASPNETCORE_ENVIRONMENT: Production

COPY bin/Release/net6.0/publish .
ENTRYPOINT ["dotnet", "aws.dll"]
```

The app itself will run on the exposed port :5000 in our case

- Create a docker-compose.yml file with the content

```
version: '3.4'
services:
  restapi:
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - 5000:5000
```

Spacing is important here – 1Tab should be 2 spaces

Let's test the docker image/container

- Open Console and run docker-compose build

```
PS C:\Users\Ionel\Desktop\aws> docker-compose build
[+] Building 1.0s (8/8) FINISHED
 => [internal] load build definition from Dockerfile
0.1s
 => => transferring dockerfile: 265B
0.0s
 => [internal] load .dockerignore
0.0s
 => => transferring context: 2B
0.0s
 => [internal] load metadata for mcr.microsoft.com/dotnet/aspnet:6.0
0.5s
 => [1/3] FROM
mcr.microsoft.com/dotnet/aspnet:6.0@sha256:299dce0ad496ffd8ec987c420e5d66129422247b02ffa7c57
79c7a9911770d0a      0.0s
 => => resolve
mcr.microsoft.com/dotnet/aspnet:6.0@sha256:299dce0ad496ffd8ec987c420e5d66129422247b02ffa7c57
79c7a9911770d0a      0.0s
```

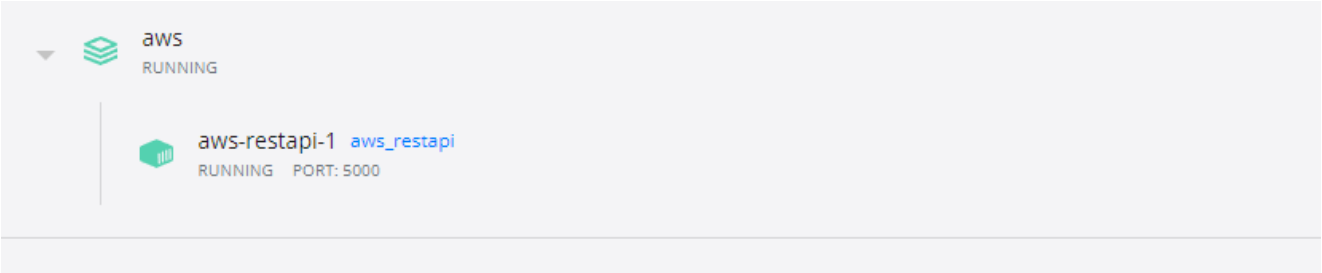
```
 => [internal] load build context
0.1s
 => => transferring context: 185.09kB
0.0s
 => CACHED [2/3] WORKDIR /app
0.0s
 => [3/3] COPY bin/Release/net6.0/publish .
0.2s
 => exporting to image
0.1s
 => => exporting layers
0.1s
 => => writing image sha256:3945f94f2ba293f25d91a98b500508e62ac676f8790ed0aafdcc3c3ce18b4973
0.0s
 => => naming to docker.io/library/aws_restapi
0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to
fix them:
```

- Run docker-compose up

```
PS C:\Users\Ionel\Desktop\aws> docker-compose up
[+] Running 2/2
 - Network aws_default      Created
0.7s
 - Container aws-restapi-1  Created
0.2s
Attaching to aws-restapi-1
```

Now if you access localhost:5000/swagger you should see the app running

And a container was created



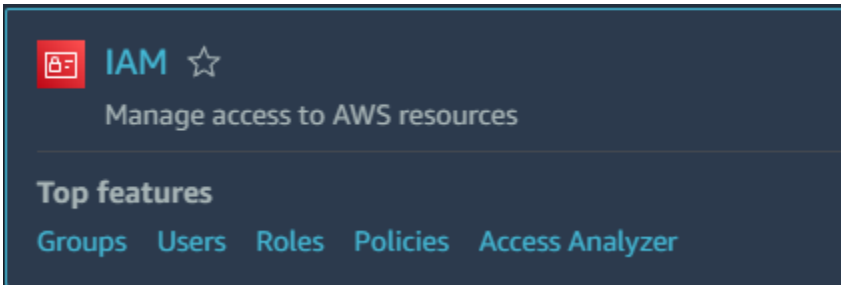
Now we need to publish the app in aws

https://console.aws.amazon.com/

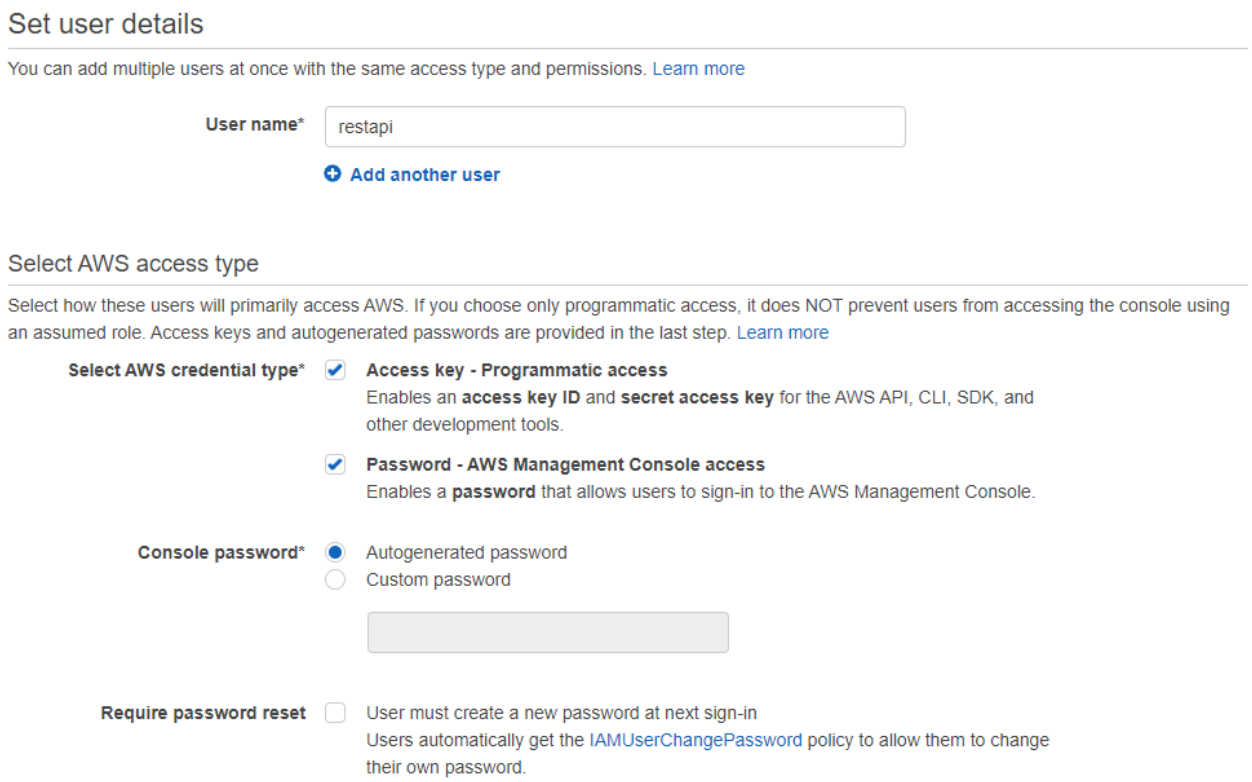
For this we need to setup aws on our system

- For learning purposes we will create a new user with an access key and right to deploy in apprunner

- Access IAM in the aws console search



- Access Users -> Add New User enable both access key and password

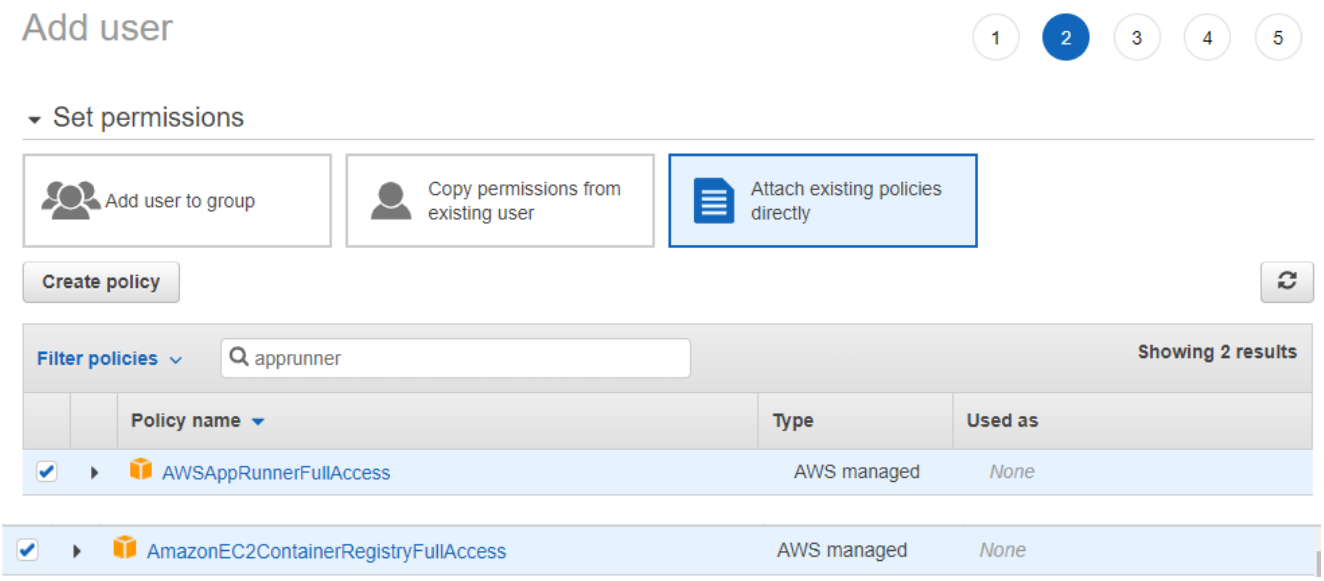


Next

Attach existing policies directly

Here we need access to

AmazonEC2ContainerRegistryFullAccess and AWSAppRunnerFullAccess



IAMFULLACCESS NEEDED TO CREATE APPRUNNER SERVICES.

Next

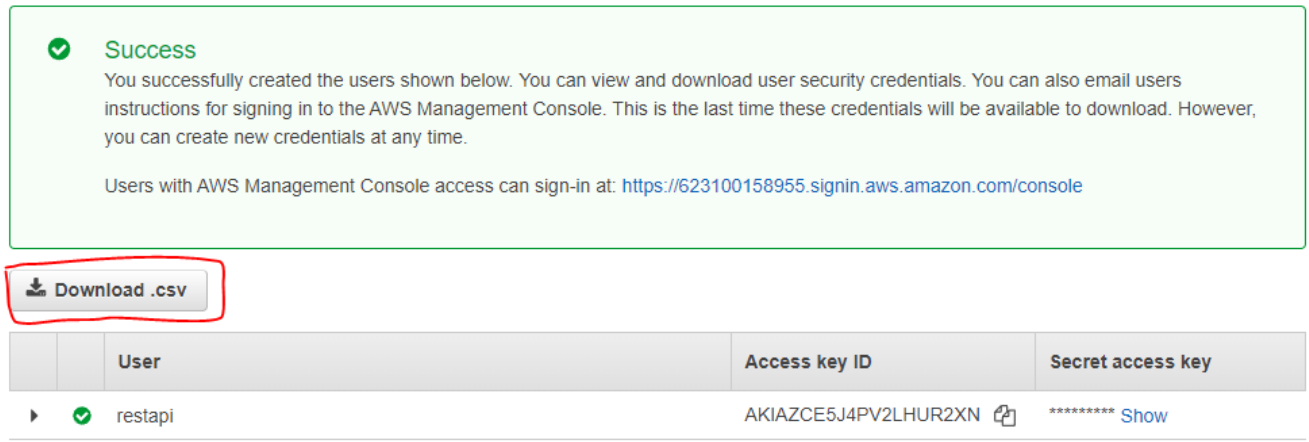
This step is Optional

These Tags are used to organize and track the access of this user

Create User

An access key will be given

You can download the file containing all the information



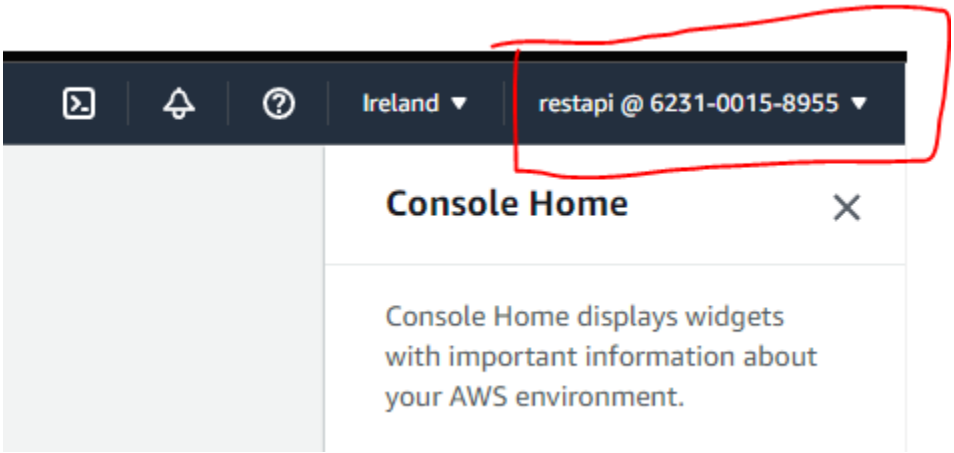
The CSV will contain all the required information

	A	B	C	D	E	F
1	User name	Password	Access key ID	Secret access key	Console login link	
2	restapi	eFo4&=y1ZOCW^qa	AKIAZCE5J4PV66L5INWR	V4LMTEiwfMT1/u3uO9Nq5aJUZ2feaEYNmSWWuNBV	https://623100158955.signin.aws.amazon.com/console	
3						
4						

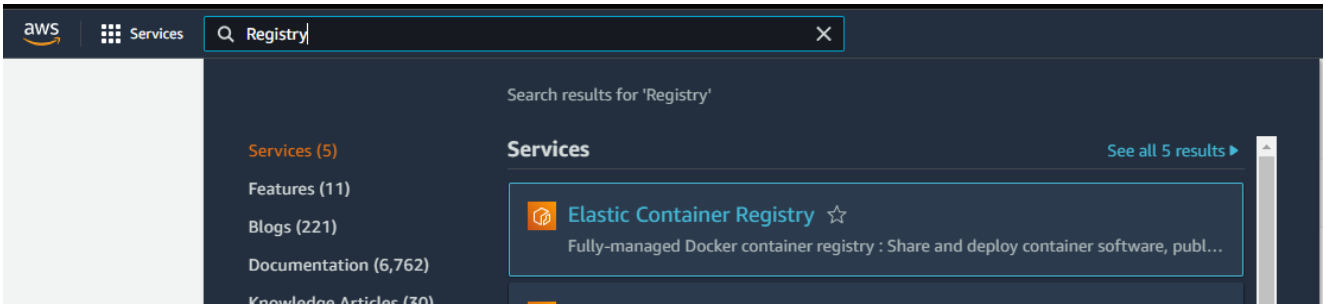
Accessing the console link to verify it is created

You will log in using the username and password from the file

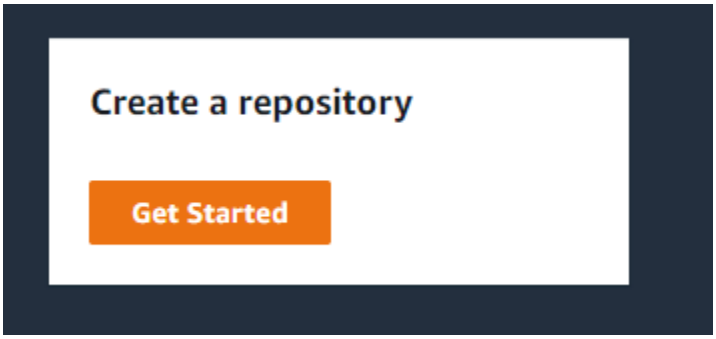
Switch to new console

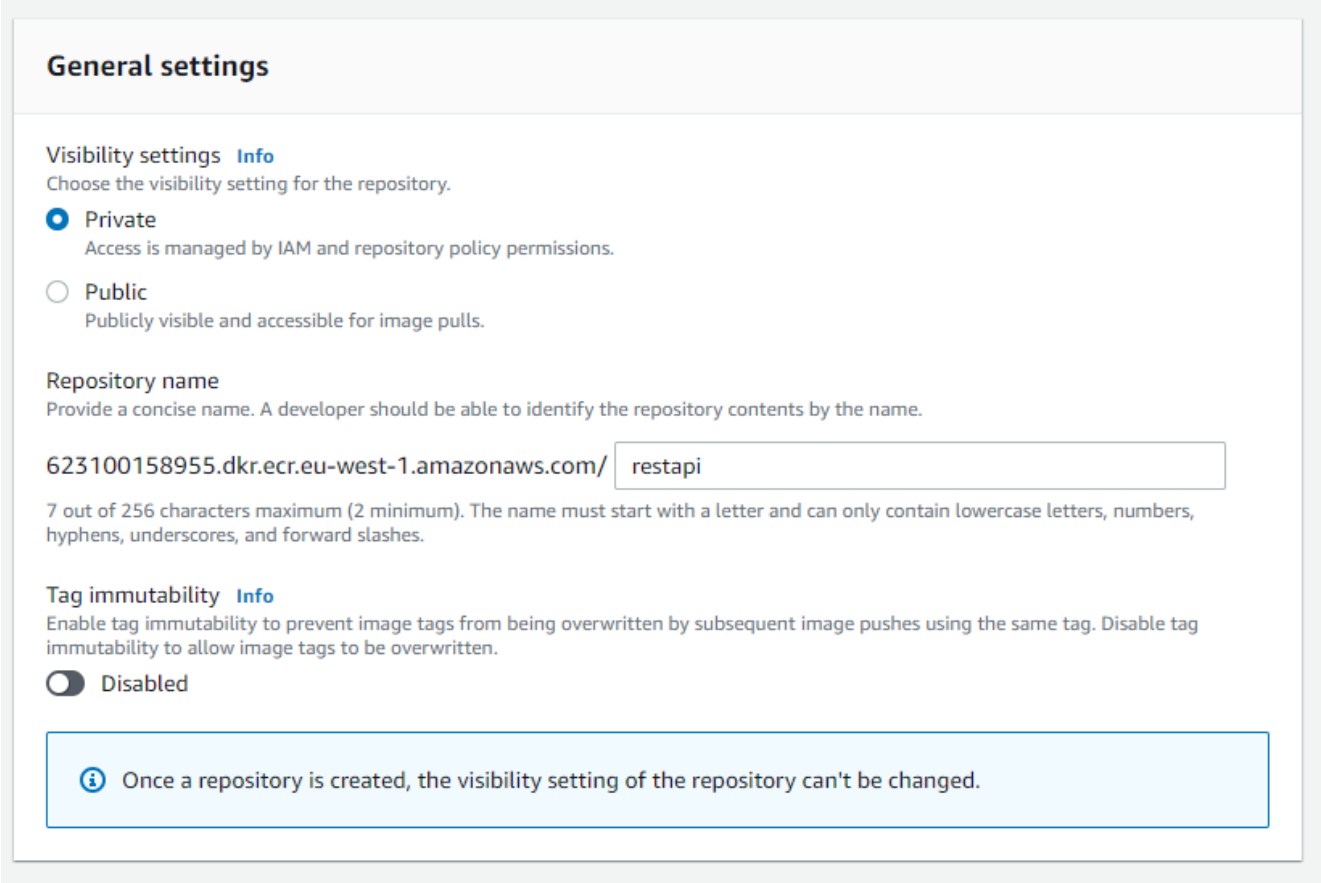


Access Elastic Container Registry



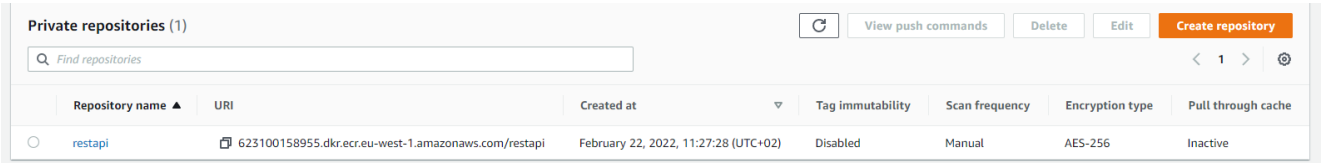
Create a repository



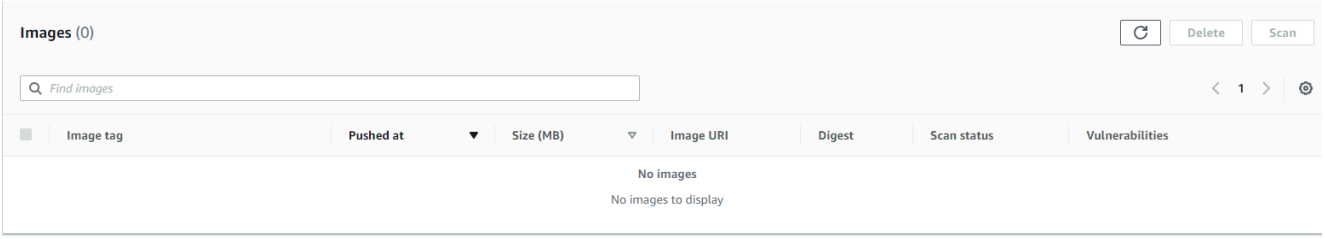


We will leave the rest of the items unchanged in the Create repo form

A new private repo will show



Clicking it will show the docker images available



Now in the CMD/PowerShell on your PC

Run aws configure

Add the access keys requested

You can copy paste them from the file into tour console

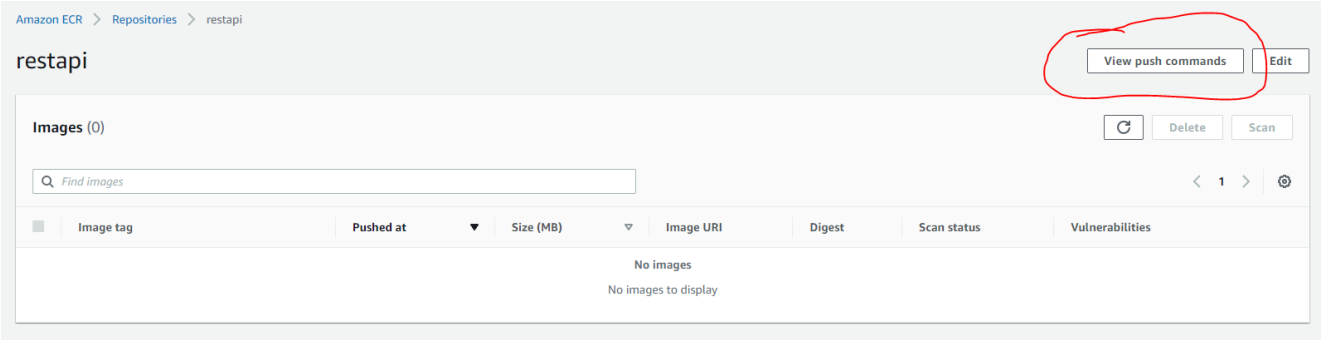
We will use eu-west-1 for this test as a region

Default output format is json

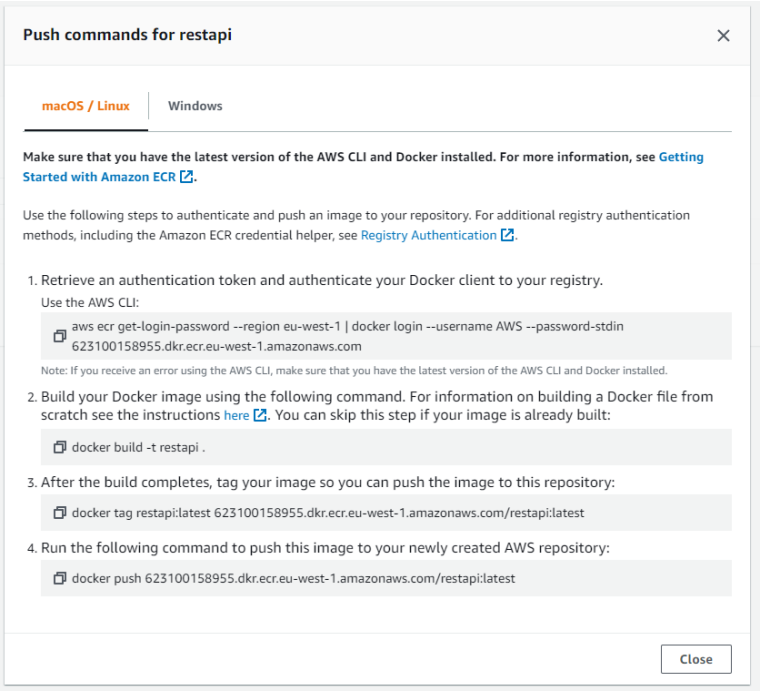
Now we need to log in

And push the current image on our pc

In the Images view of your repo youll find a view push commands button



Accessing it will show the steps required and the commands you need to use based on your use



First steps is to request an auth token for your registry

```
PS C:\Users\Ionel\Desktop\aws> aws ecr get-login-password --region eu-west-1 | docker login
--username AWS --password-stdin 623100158955.dkr.ecr.eu-west-1.amazonaws.com
Login Succeeded
```

We already built the image on our local system using docker-compose build

And the image name is aws_restapi

Now we need to tag the image

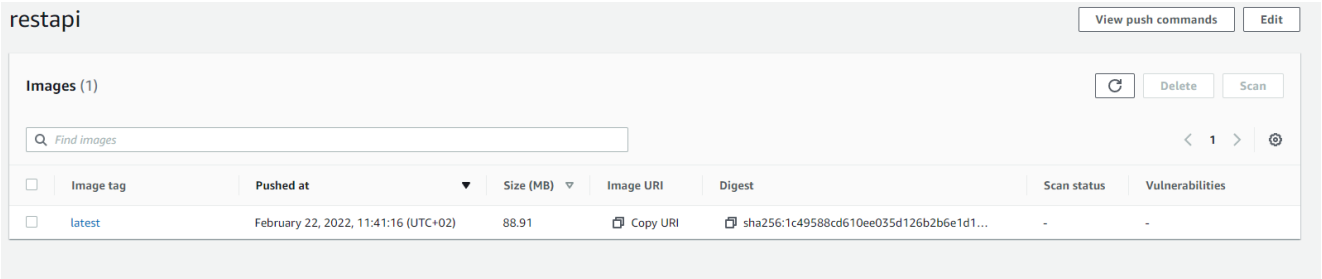
```
PS C:\Users\Ionel\Desktop\aws> docker tag aws_restapi 623100158955.dkr.ecr.eu-west-
1.amazonaws.com/restapi:latest
```

```
PS C:\Users\Ionel\Desktop\aws>
```

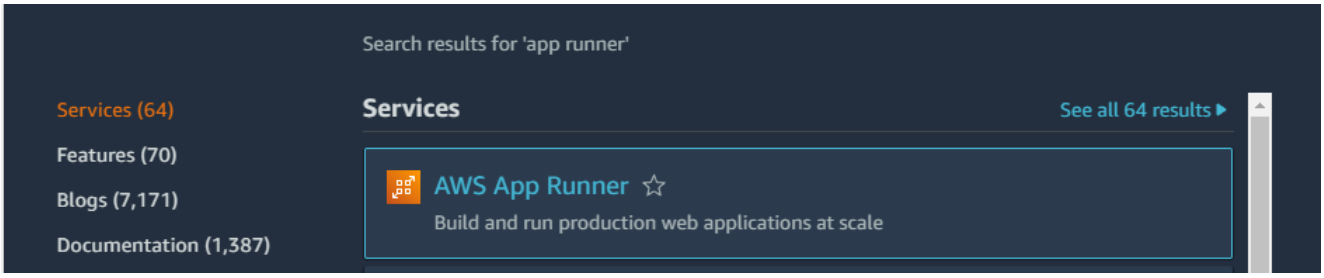
After the tag we need to push the image into aws

```
PS C:\Users\Ionel\Desktop\aws> docker push 623100158955.dkr.ecr.eu-west-
1.amazonaws.com/restapi:latest
The push refers to repository [623100158955.dkr.ecr.eu-west-1.amazonaws.com/restapi]
0846bd788fb7: Pushed
4cdaab58c367: Pushed
448a02ae706e: Pushed
a38656eccfc9: Pushed
e28de40436ef: Pushed
40bd18193dde: Pushed
7d0ebbe3f5d2: Pushed
latest: digest: sha256:1c49588cd610ee035d126b2b6e1d1e2c22b99c4a7bb854710c8c6579d5e919fc
size: 1789
```

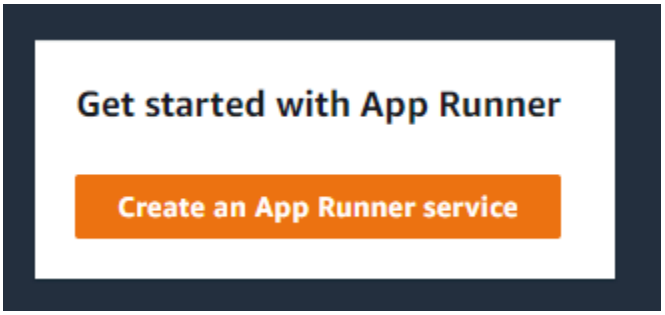
The image should be visible in the registry



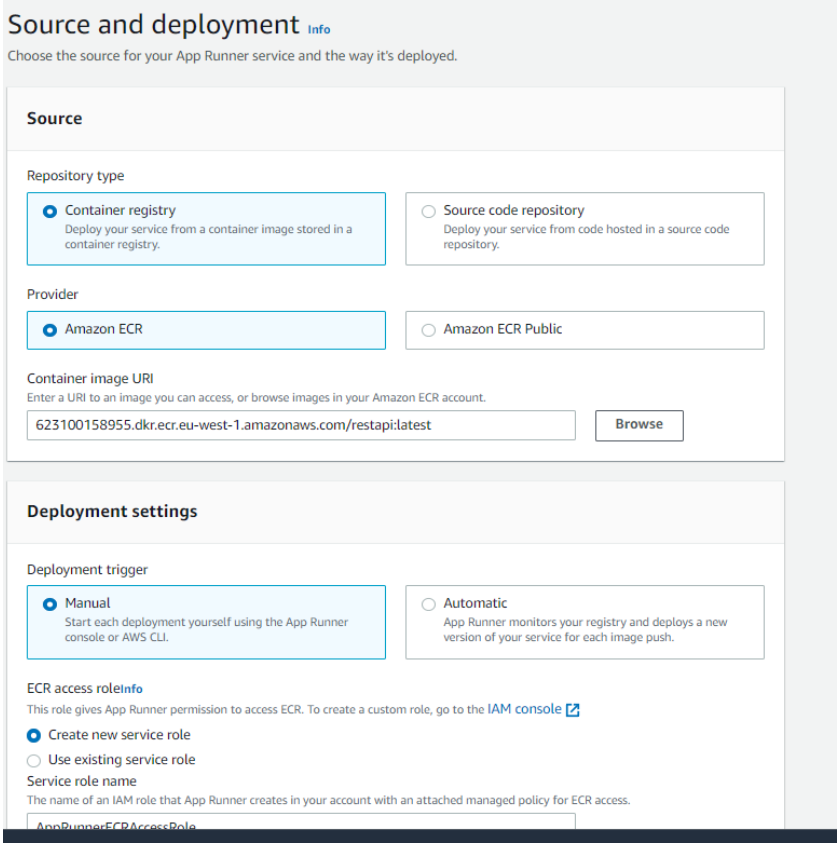
Now access the app runner



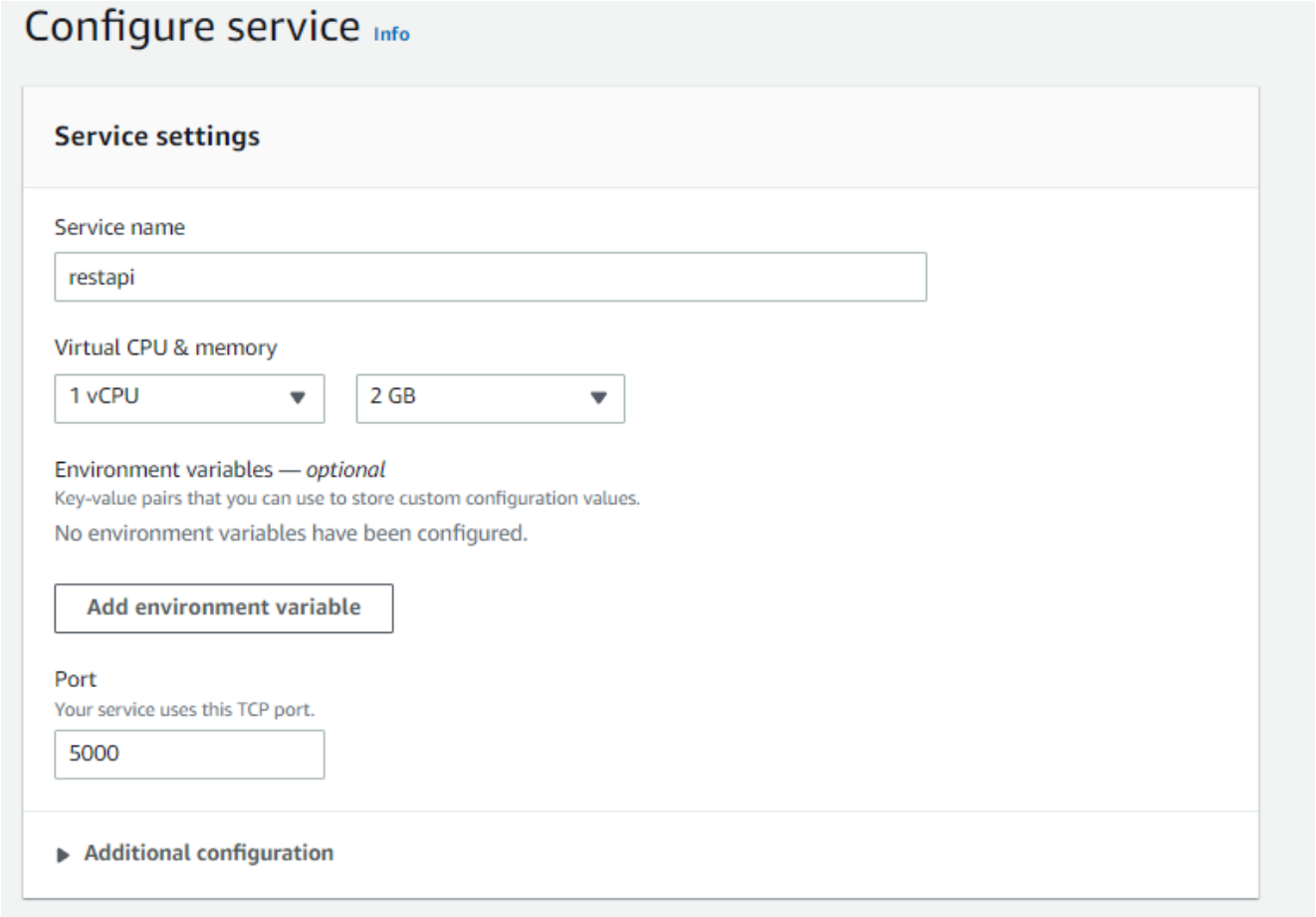
Create a new App runner Service



Browse your container image



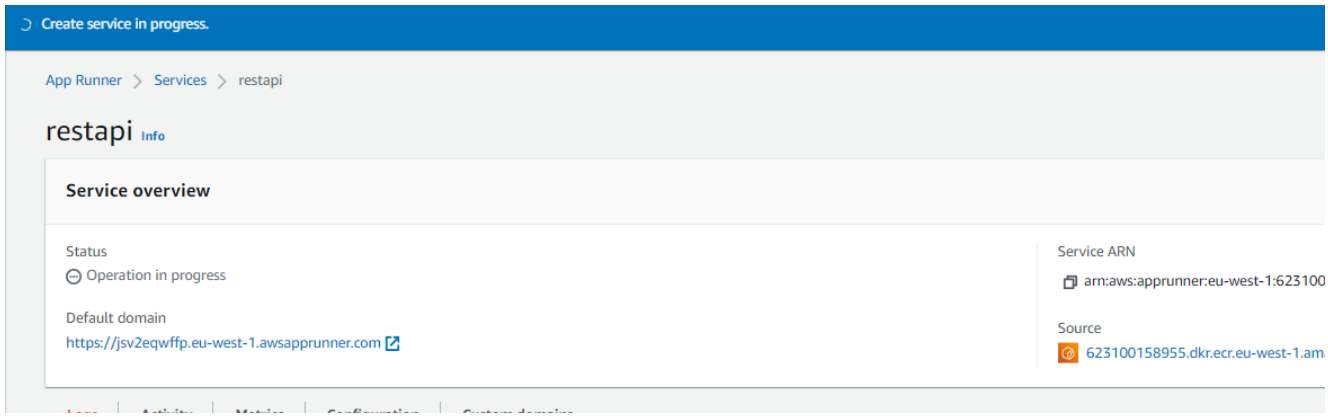
Add service configuration



Add Name

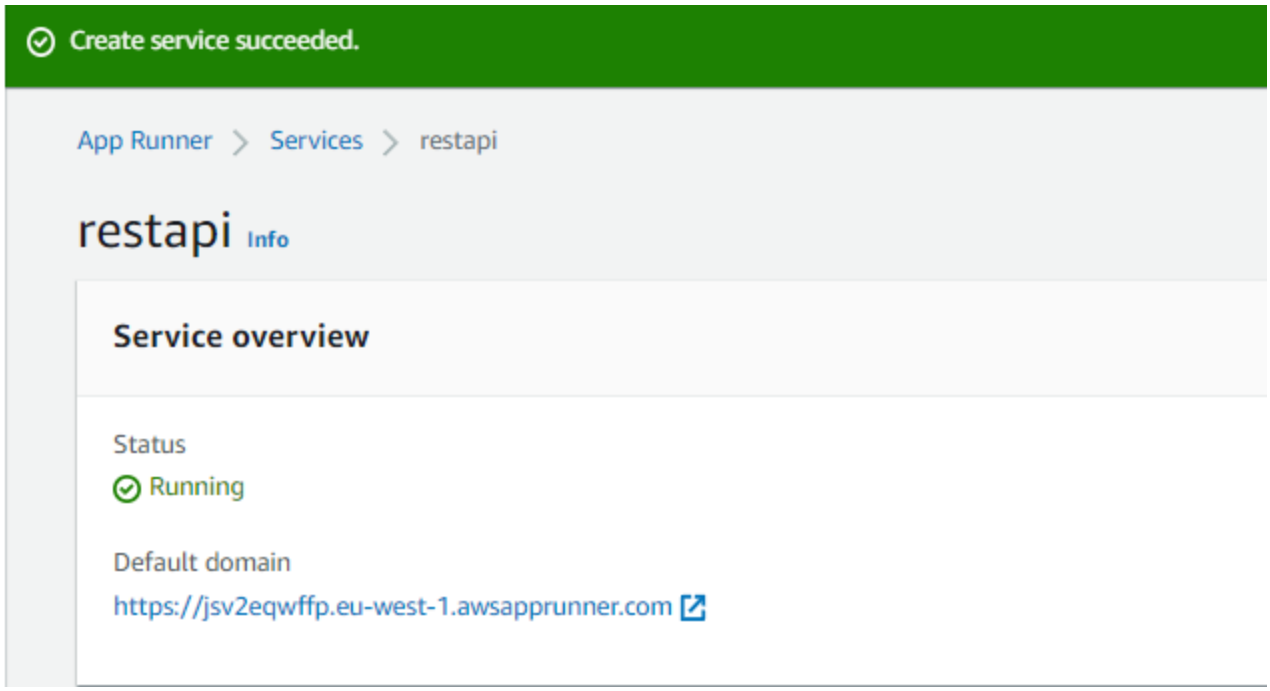
Add the port you run your app on (5000 in our case)

Create and deploy

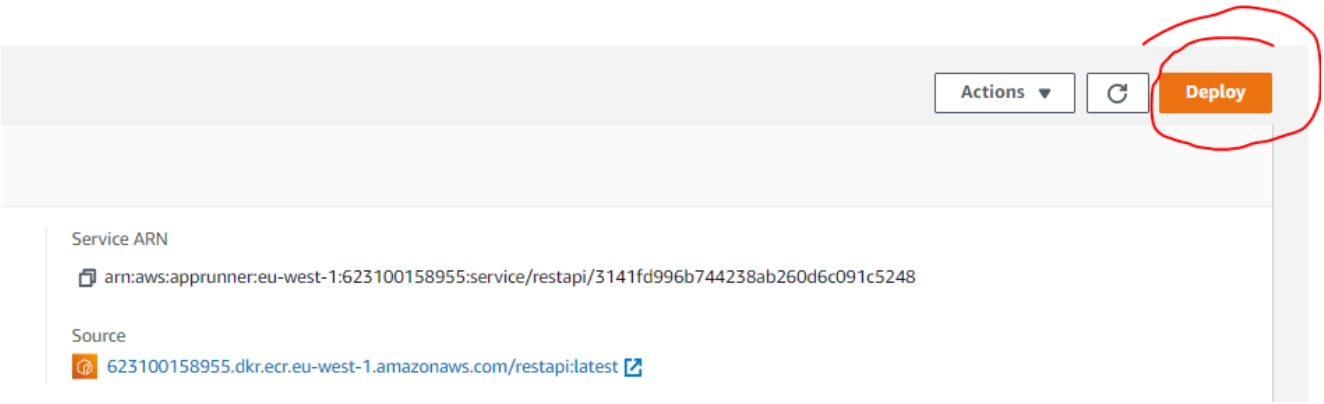


Now the service is being created

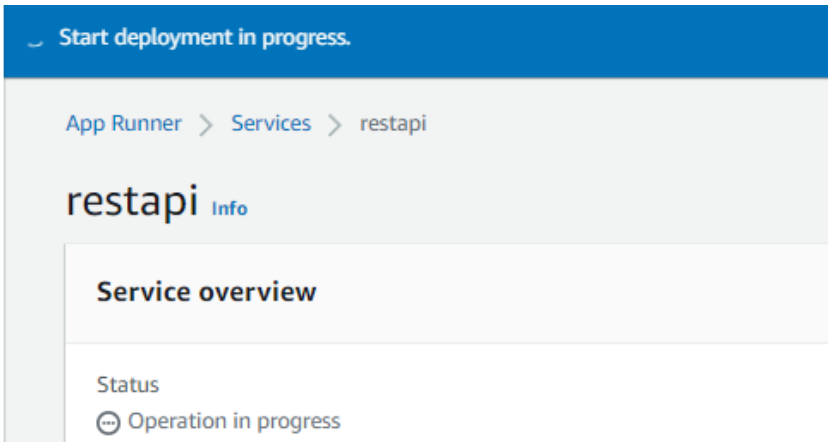
Wait until it finishes (5-10 minutes)

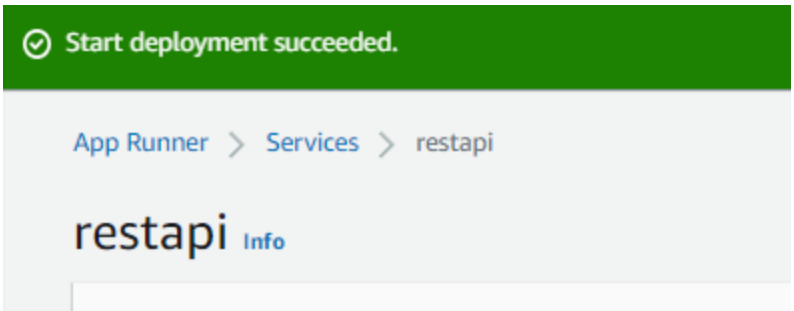


Now Click deploy

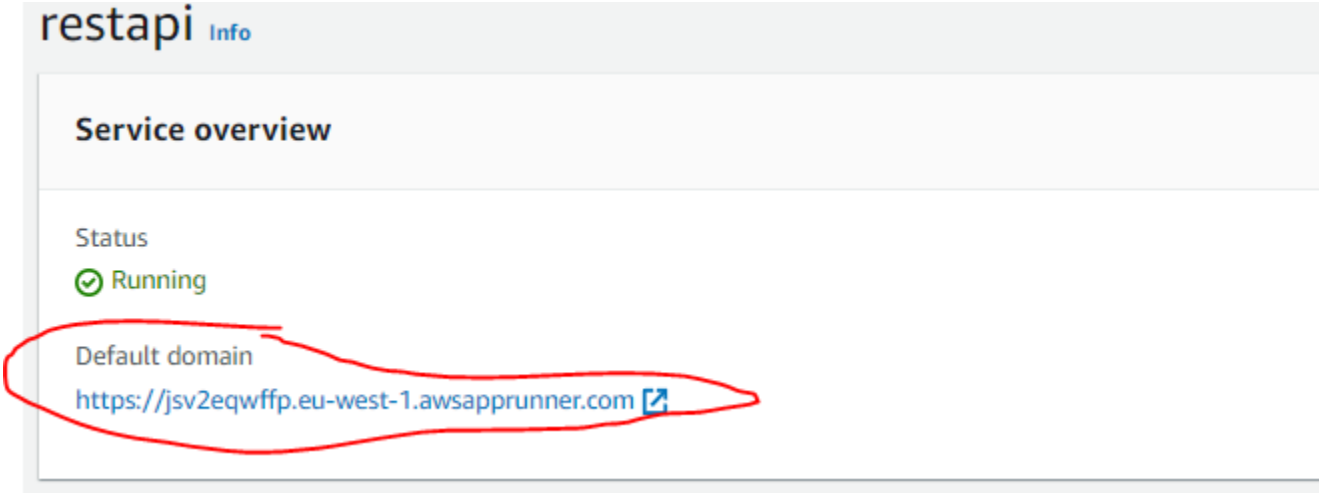


Wait until it finishes deploying (5-10 minutes)

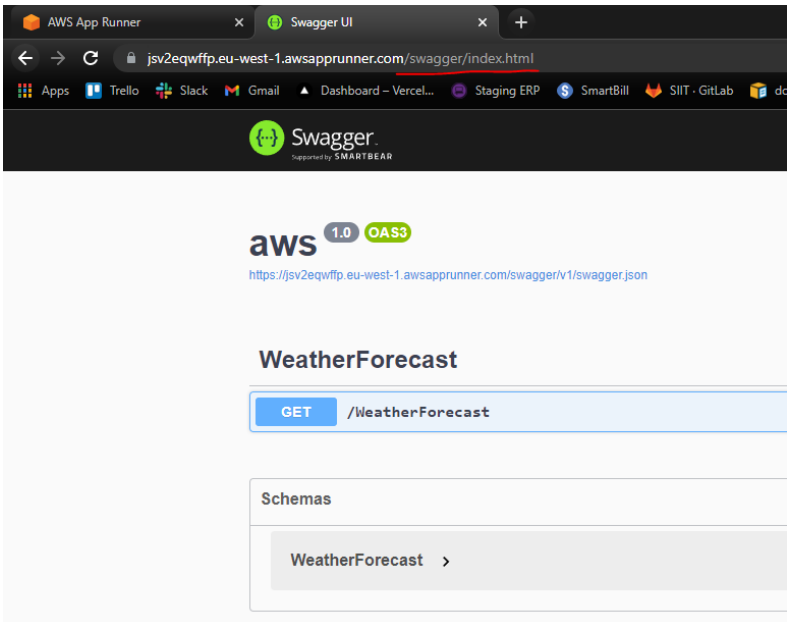




Now you can access the link noted in the window



With /Swagger at the end



Your APP is officially published