

Requirements:

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

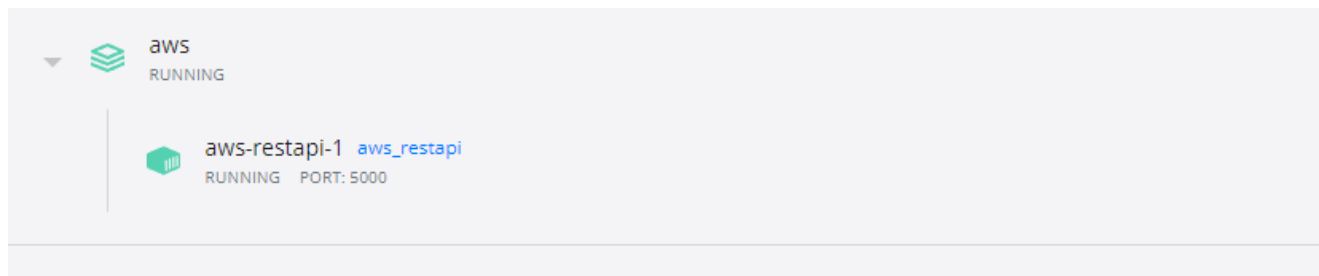
```

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



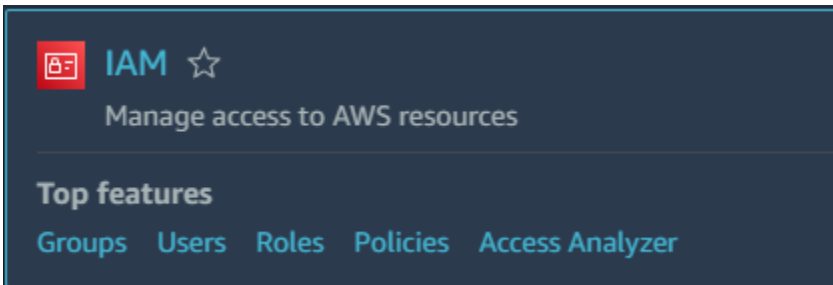
Now we need to publish the app in aws

<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

## Set user details

You can add multiple users at once with the same access type and permissions. [Learn more](#)

User name\*

[+ Add another user](#)

## Select AWS access type

Select how these users will primarily access AWS. If you choose only programmatic access, it does NOT prevent users from accessing the console using an assumed role. Access keys and autogenerated passwords are provided in the last step. [Learn more](#)

- Select AWS credential type\* ☒ **Access key - Programmatic access**  
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.
- ☒ **Password - AWS Management Console access**  
Enables a **password** that allows users to sign-in to the AWS Management Console.

- Console password\* ☒ Autogenerated password  
☐ Custom password

- Require password reset ☐ User must create a new password at next sign-in  
Users automatically get the [IAMUserChangePassword](#) policy to allow them to change their own password.

Next

Attach existing policies directly


Here we need access to


[AmazonEC2ContainerRegistryFullAccess](#) and [AWSAppRunnerFullAccess](#)


## Add user

1 2 3 4 5

### Set permissions



 Add user to group

 Copy permissions from existing user

 Attach existing policies directly

Create policy

Filter policies  Showing 2 results

	Policy name	Type	Used as
<input checked="" type="checkbox"/>	 AWSAppRunnerFullAccess	AWS managed	None
<input checked="" type="checkbox"/>	 AmazonEC2ContainerRegistryFullAccess	AWS managed	None

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

 **Success**  
You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.  
  
Users with AWS Management Console access can sign-in at: <https://623100158955.signin.aws.amazon.com/console>

 Download .csv

User	Access key ID	Secret access key
 restapi	AKIAZCE5J4PV2LHUR2XN 	***** <a href="#">Show</a>

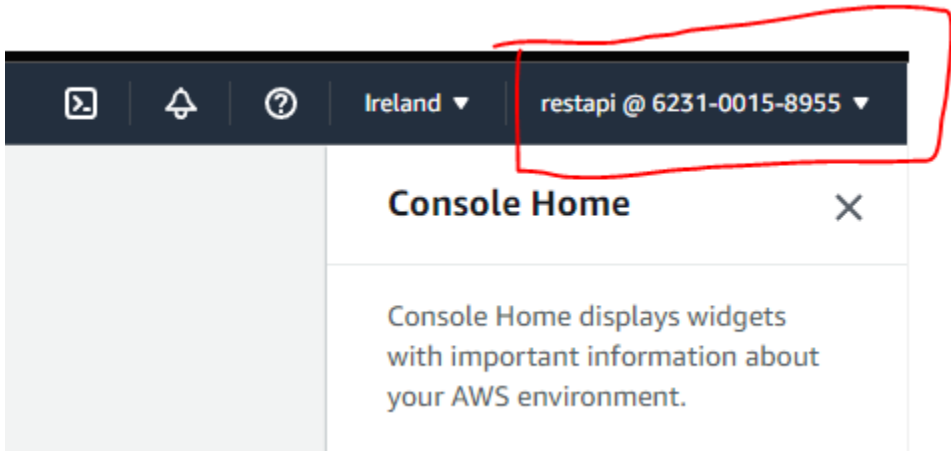
The CSV will contain all the required information

	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/u3uO9Nq5aJU2feaEYNmSWWuNBV	https://623100158955.signin.aws.amazon.com/console	
3						
4						

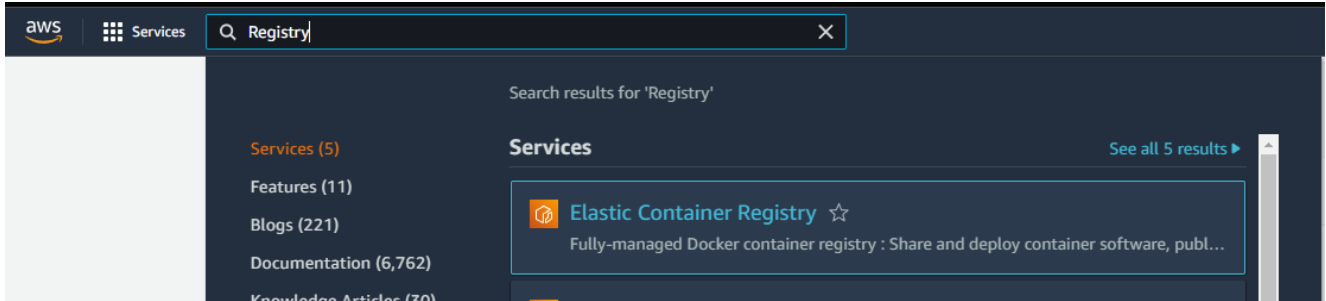
Accessing the console link to verify it is created

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

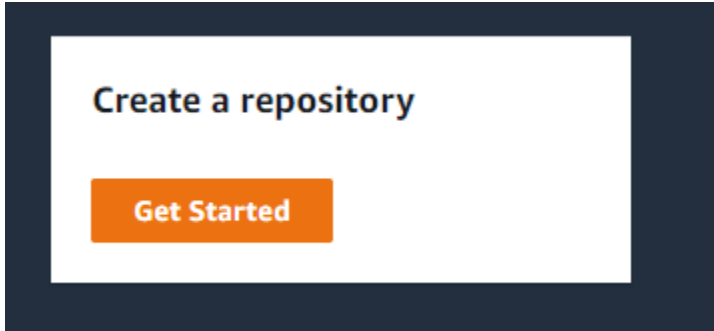
Switch to new console



Access Elastic Container Registry



Create a repository



## 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.

Repository name

Provide a concise name. A developer should be able to identify the repository contents by the name.

623100158955.dkr.ecr.eu-west-1.amazonaws.com/


restapi

7 out of 256 characters maximum (2 minimum). The name must start with a letter and can only contain lowercase letters, numbers, hyphens, underscores, and forward slashes.

Tag immutability [Info](#)

Enable tag immutability to prevent image tags from being overwritten by subsequent image pushes using the same tag. Disable tag immutability to allow image tags to be overwritten.

☐ Disabled

 Once a repository is created, the visibility setting of the repository can't be changed.

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

A new private repo will show

Private repositories (1)

View push commands

Delete

Edit

Create repository

< 1 >

⚙

	Repository name ▲	URI	Created at ▼	Tag immutability	Scan frequency	Encryption type	Pull through cache
○	restapi	623100158955.dkr.ecr.eu-west-1.amazonaws.com/restapi	February 22, 2022, 11:27:28 (UTC+02)	Disabled	Manual	AES-256	Inactive

Clicking it will show the docker images available

Images (0)

🔄

Delete

Scan

<

1

>

⚙️

	Image tag	Pushed at ▼	Size (MB) ▼	Image URI	Digest	Scan status	Vulnerabilities
No images							
No images to display							

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 your console

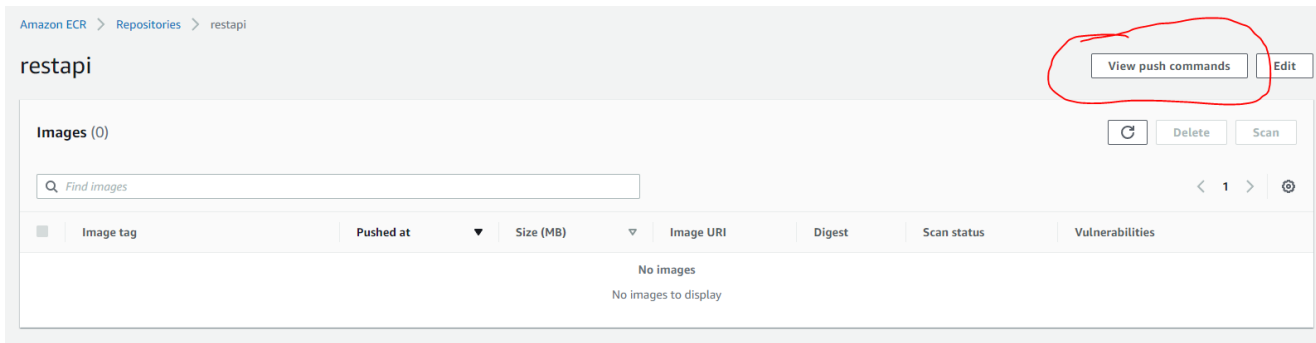
We will use eu-west-1 for this test as a region

Default output format is json

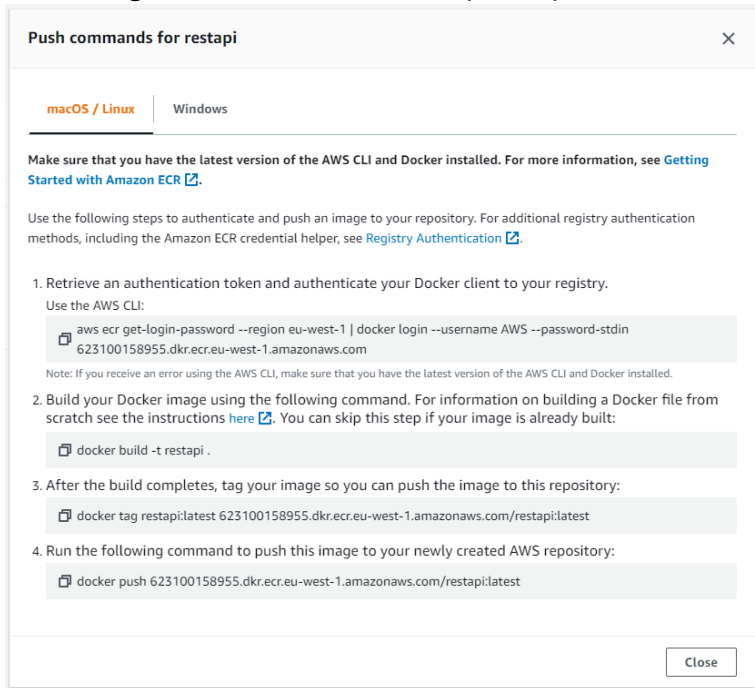
Now we need to log in

And push the current image on our pc

In the Images view of your repo you'll 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>
```

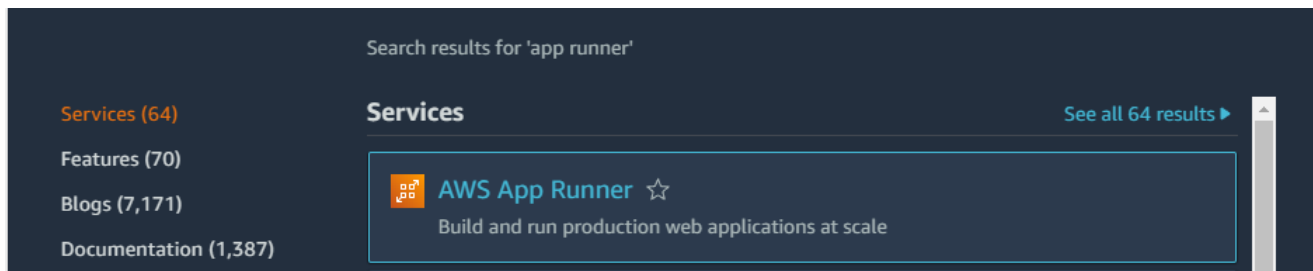
After the tag we need to push the image into aws

```
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
```

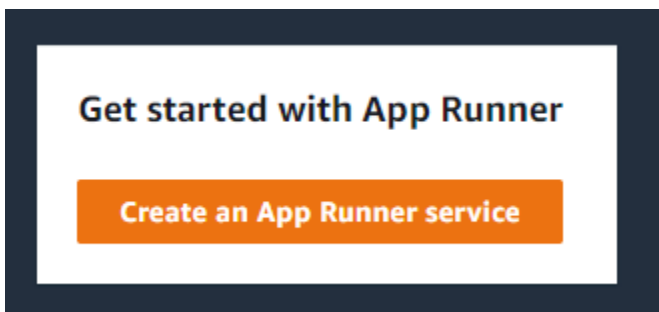
The image should be visible in the registry

restapi							View push commands	Edit
Images (1)								Delete Scan
<input type="text" value="Find images"/>							< 1 >	
<input type="checkbox"/>	Image tag	Pushed at	Size (MB)	Image URI	Digest	Scan status	Vulnerabilities	
<input type="checkbox"/>	latest	February 22, 2022, 11:41:16 (UTC+02)	88.91	Copy URI	sha256:1c49588cd610ee035d126b2b6e1d1...	-	-	

Now access the app runner



Create a new App runner Service



Browse your container image

**Source and deployment** [Info](#)

Choose the source for your App Runner service and the way it's deployed.

**Source**

Repository type

☒ **Container registry**  
Deploy your service from a container image stored in a container registry.

☐ **Source code repository**  
Deploy your service from code hosted in a source code repository.

Provider

☒ **Amazon ECR**

☐ **Amazon ECR Public**

Container image URI  
Enter a URI to an image you can access, or browse images in your Amazon ECR account.

**Deployment settings**

Deployment trigger

☒ **Manual**  
Start each deployment yourself using the App Runner console or AWS CLI.

☐ **Automatic**  
App Runner monitors your registry and deploys a new version of your service for each image push.

ECR access role [Info](#)  
This role gives App Runner permission to access ECR. To create a custom role, go to the [IAM console](#).

☒ **Create new service role**

☐ **Use existing service role**

Service role name  
The name of an IAM role that App Runner creates in your account with an attached managed policy for ECR access.

Add service configuration

## Configure service Info

### Service settings

Service name

Virtual CPU & memory

Environment variables — *optional*  
Key-value pairs that you can use to store custom configuration values.  
No environment variables have been configured.

[Add environment variable](#)

Port  
Your service uses this TCP port.

► **Additional configuration**

Add Name

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

Create and deploy

Create service in progress.

App Runner > Services > restapi

## restapi Info

### Service overview

Status	Service ARN
⌚ Operation in progress	am:aws:apprunner:eu-west-1:623100
Default domain	Source
<a href="https://jsv2eqwffp.eu-west-1.awsapprunner.com">https://jsv2eqwffp.eu-west-1.awsapprunner.com</a>	623100158955.dkr.ecr.eu-west-1.am

Now the service is being created

Wait until it finishes (5-10 minutes)

✔ Create service succeeded.

App Runner > Services > restapi

## restapi Info

### Service overview

Status  
✔ Running

Default domain  
<https://jsv2eqwffp.eu-west-1.awsapprunner.com>

Now Click deploy

Actions **Deploy**

Service ARN  
 `arn:aws:apprunner:eu-west-1:623100158955:service/restapi/3141fd996b744238ab260d6c091c5248`

Source  
 `623100158955.dkr.ecr.eu-west-1.amazonaws.com/restapi:latest`

Wait until it finishes deploying (5-10 minutes)

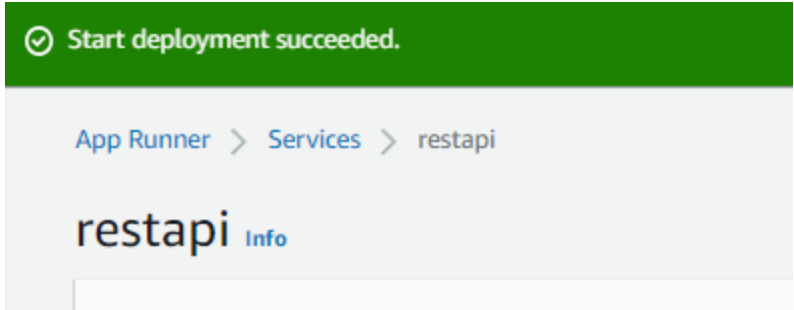
Start deployment in progress.

App Runner > Services > restapi

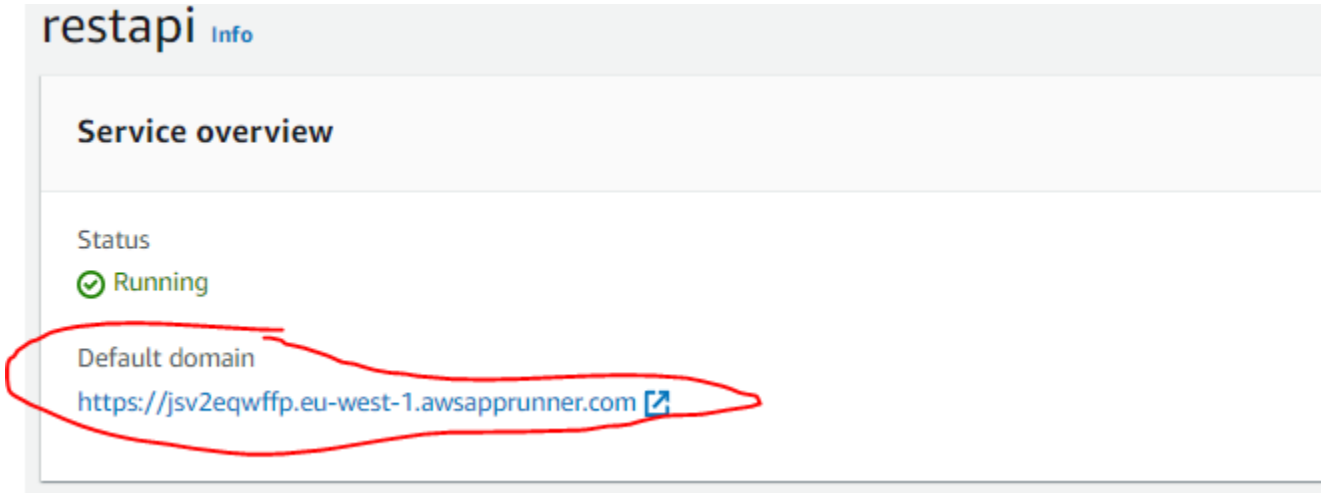
## restapi Info

### Service overview

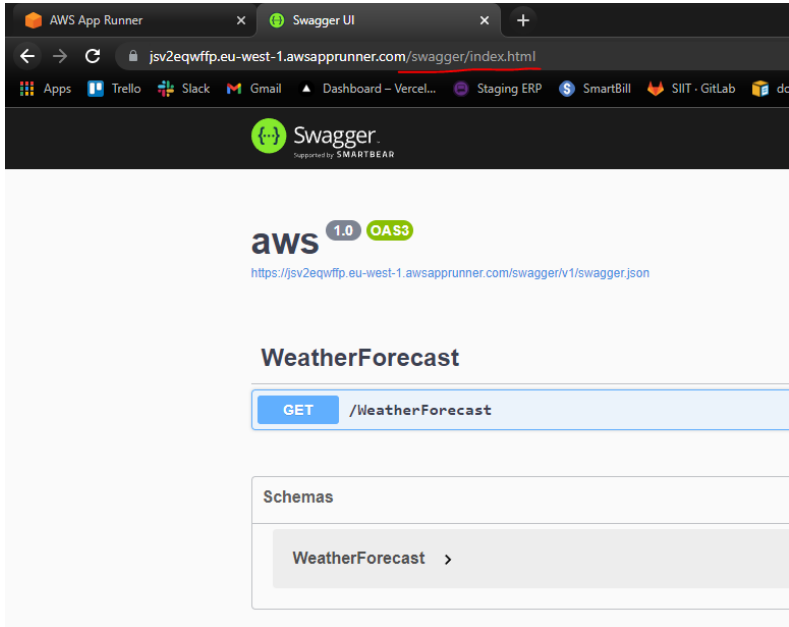
Status  
⋮ Operation in progress



Now you can access the link noted in the window



With /Swagger at the end



Your APP is officially published