This assignment focuses on understanding, analyzing, and applying various methods to deploy a Go-based application to the cloud. Our recent project, the Go Weather API, which facilitates the weather of a city, with request on the URL, URL request could be GET or POST.  
We have now experimented with migrating our database tables to cloud platforms like Microsoft Azure and AWS.

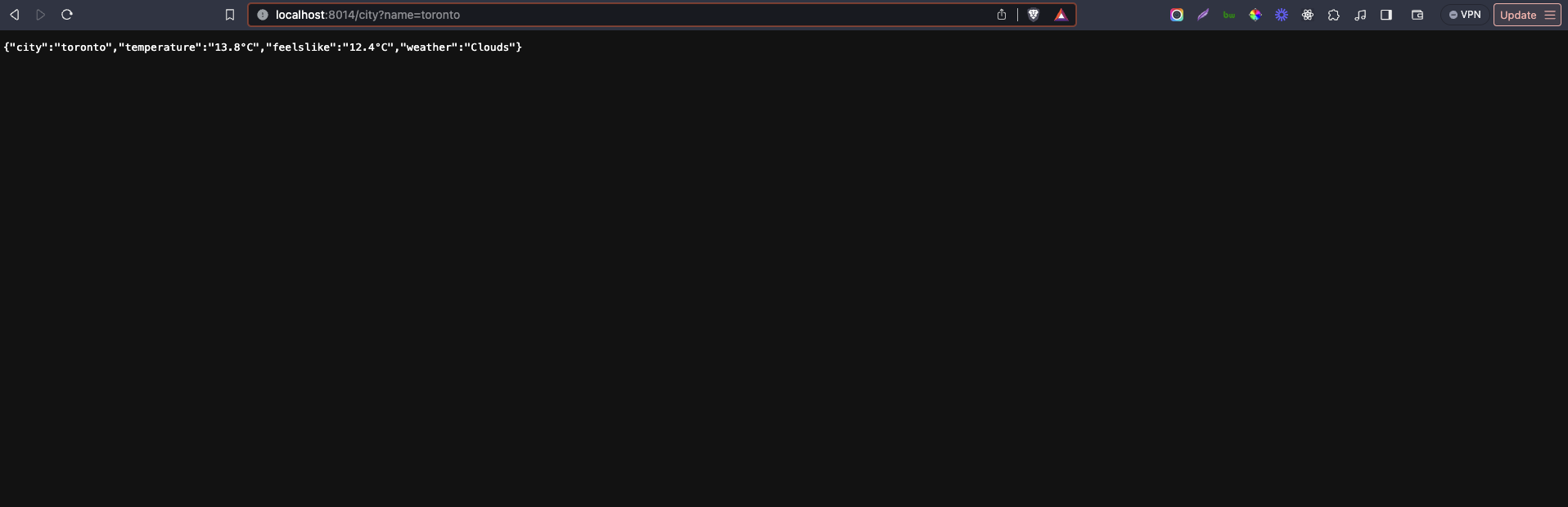
**Microsoft Azure** and **Amazon AWS** are two of the leading cloud computing platforms. Here's a simple breakdown of their differences:

1. **Services Offered:** Both offer a wide range of cloud services, but AWS has a larger selection due to its earlier start. Azure, however, is known for better integration with Microsoft's software products.
2. **Pricing**: Both platforms use a pay-as-you-go pricing model, but their structures are different. AWS charges per hour, while Azure charges per minute, which can offer more precise pricing options.
3. **Hybrid Cloud:** Azure provides stronger support for hybrid cloud configurations, making it easier for businesses to integrate their on-premises datacenters with the cloud. AWS has also expanded its hybrid capabilities, but Azure leads due to Microsoft's long-standing presence in enterprise IT.
4. **Global Reach:** AWS has a broader global footprint, with more data centers worldwide. This can be advantageous for serving users in different geographical locations.
5. **Open Source Support:** AWS is generally considered to have stronger support for Linux and open-source technologies. Azure has made significant strides in this area but is traditionally seen as more Windows-centric.

**Why We choose AWS**

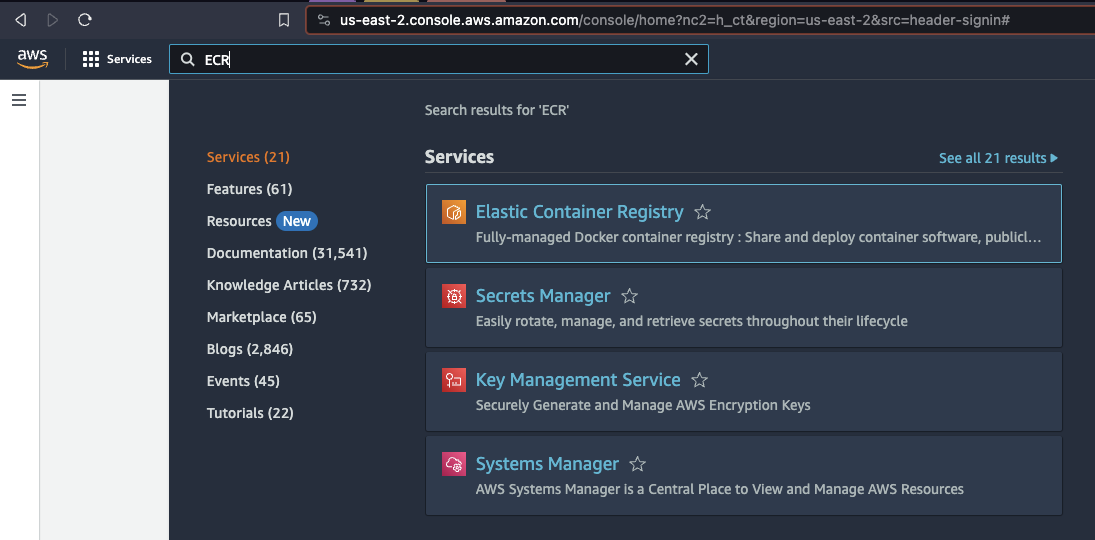
**Choosing AWS might be right for you due to its:   
1. Wide Range of Services:** A vast selection of services for computing, storage, databases, and more.   
**2. Global Reach:** A broad network of data centers worldwide for lower latency and regulatory compliance.  **3. Innovation:** Quick adoption of new technologies and services**.   
4. Flexibility in Pricing:** Pay-as-you-go model helps manage costs effectively.   
**5. Maturity and Reliability:** Proven track record with large enterprises and various industries.   
**6. Open Source Support:** Strong support for Linux and open-source technologies.  
**7. Ecosystem:** A large partner ecosystem provides additional resources and support.

**Using of our application on Localhost**

****

**Implementation on AWS**

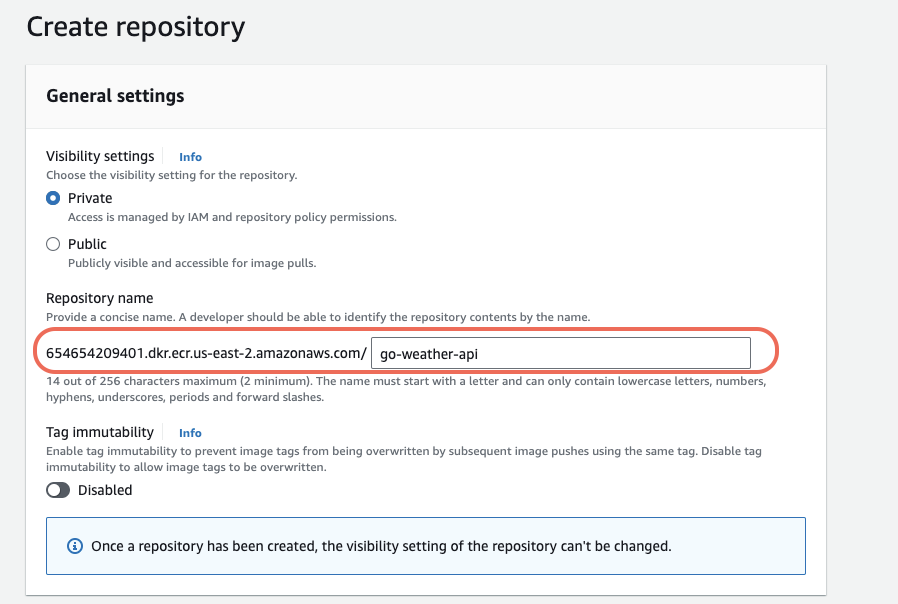
We will use Elastic Container Registry(ECR) to create a repositry, it is like a code hub like github

****

**STEP 1) PUSH our Image to ECR**

**i) Create a repositry**

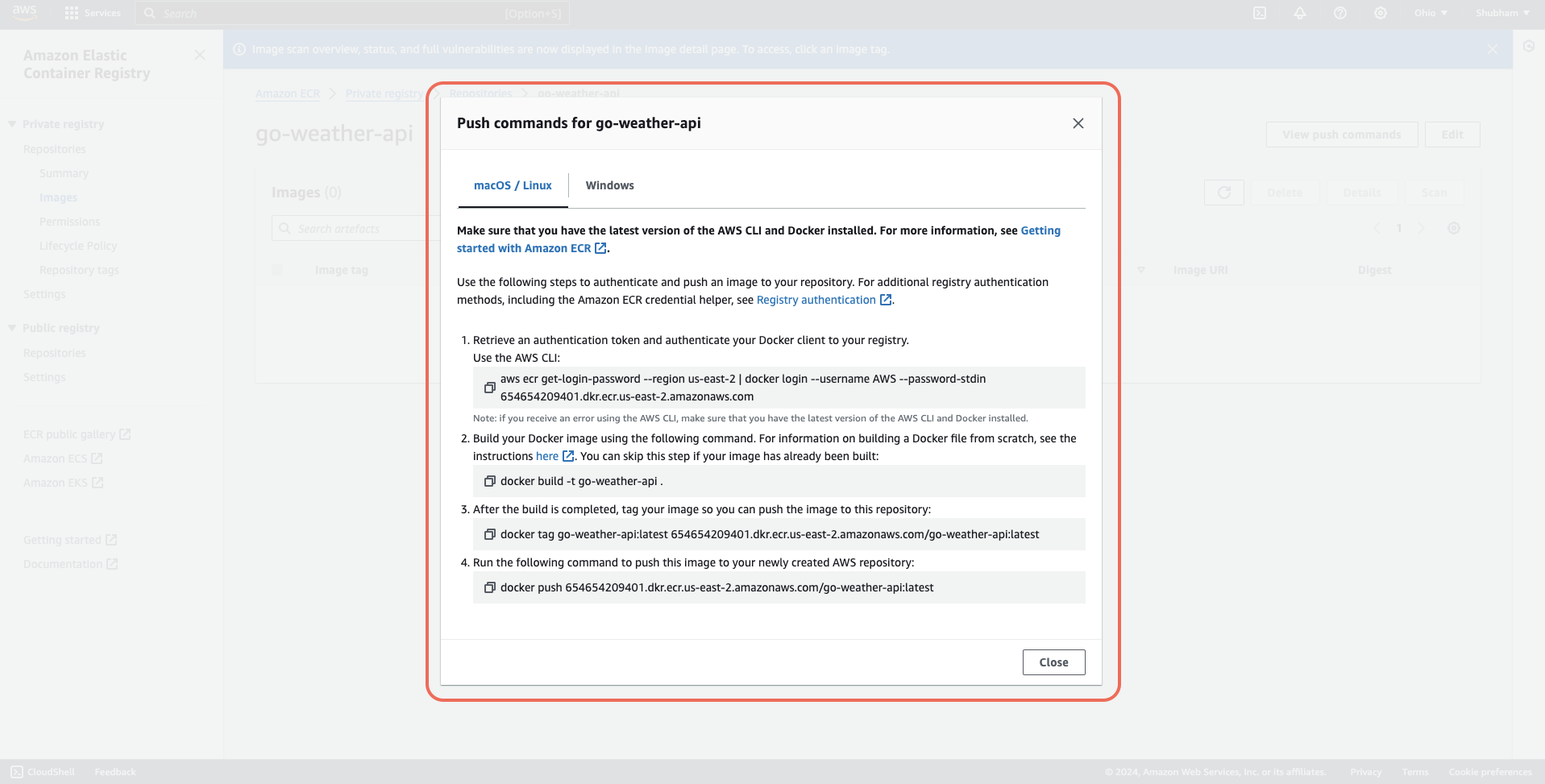
Also keeping it private so that other person can not pull it

****

**Our Repositry is created but it is empty, so we need to push our image to the repositry**

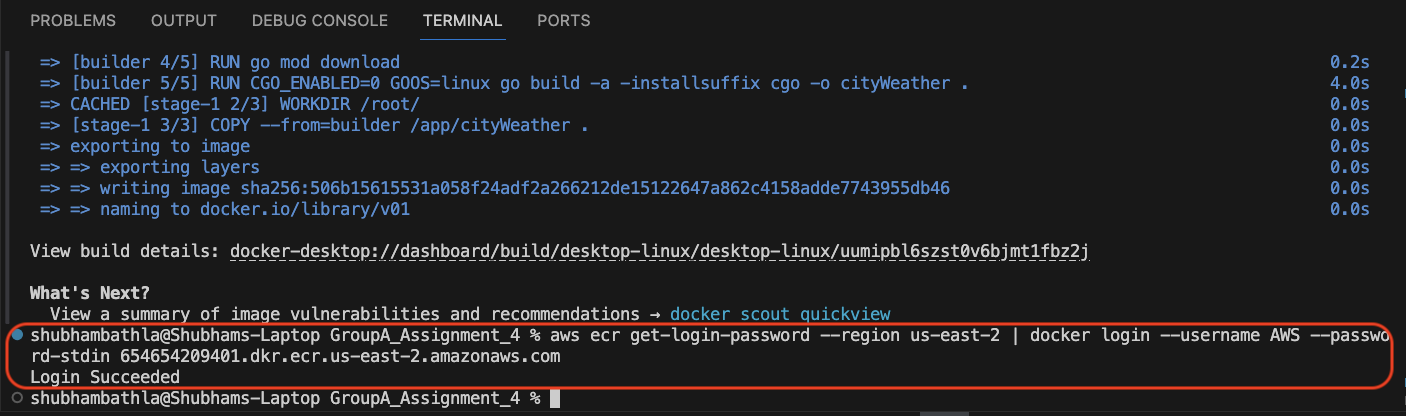
**ii) Push Image to the repository created on ECR**

We can below commands to push our image to the repositry

****

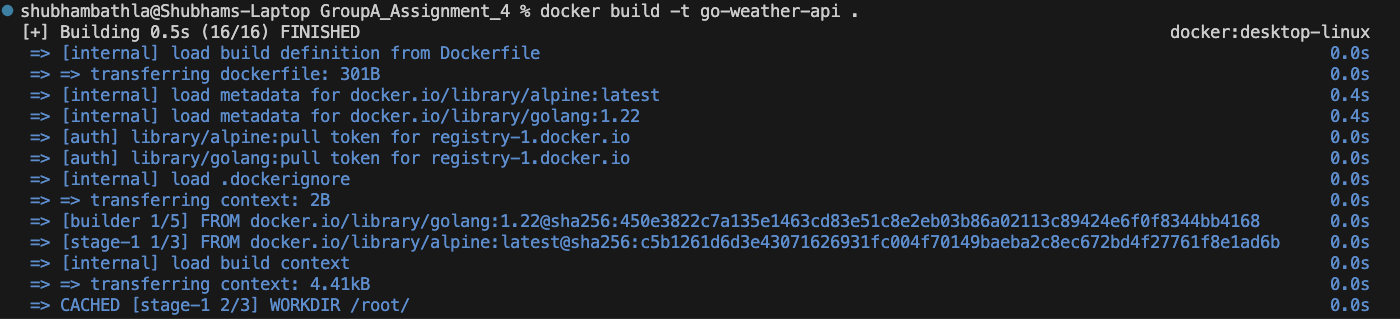
**iii) First Command is to login**

aws ecr get-login-password --region us-east-2 | docker login --username AWS --password-stdin 654654209401.dkr.ecr.us-east-2.amazonaws.com

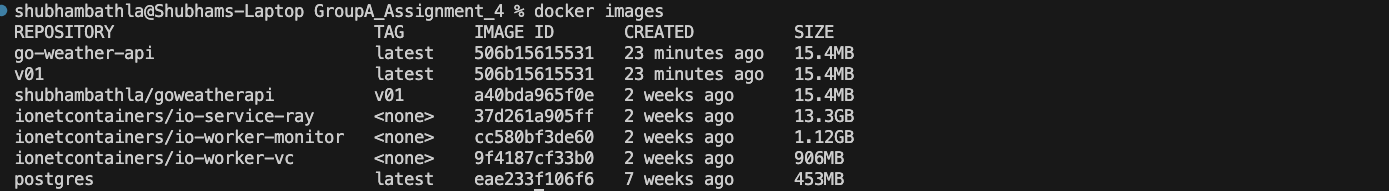
****

**iv) Build Command**

docker build -t go-weather-api .



Check if image is created using docker images command

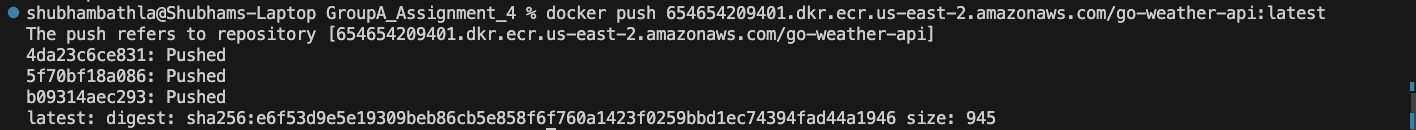


**v) Tag the Image**

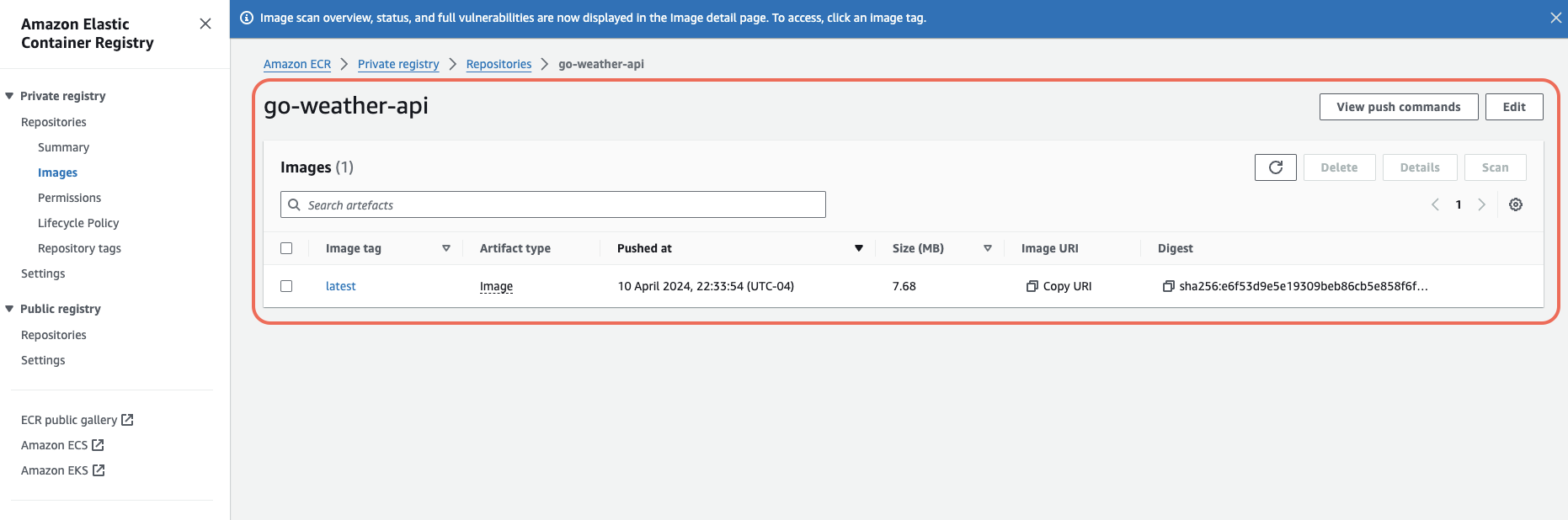
docker tag go-weather-api:latest [654654209401.dkr.ecr.us-east-2.amazonaws.com/go-weather-api:latest](http://654654209401.dkr.ecr.us-east-2.amazonaws.com/go-weather-api:latest)

**vi) Push Image to the Cloud (Registry)**

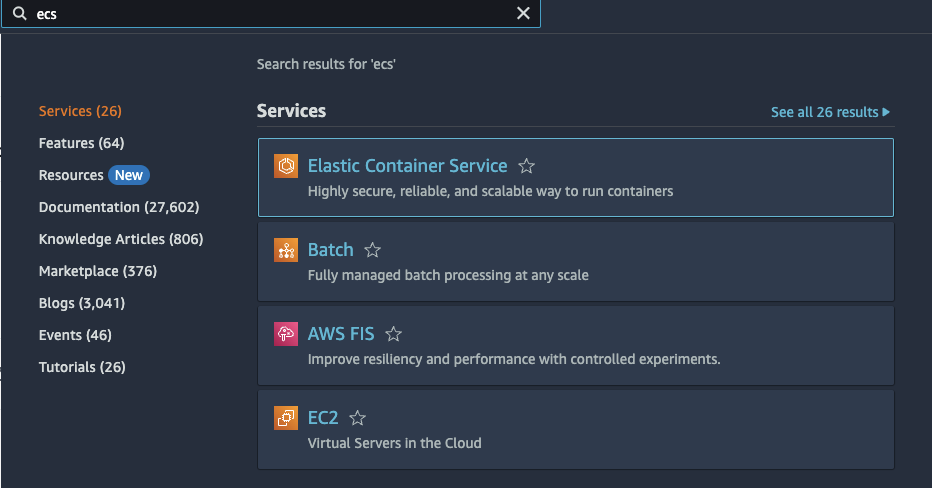
docker push [654654209401.dkr.ecr.us-east-2.amazonaws.com/go-weather-api:latest](http://654654209401.dkr.ecr.us-east-2.amazonaws.com/go-weather-api:latest)



**Now our Image is pushed on AWS ECR**



**Step 2) Deploy our container to Elastic Container Service (ECS)**

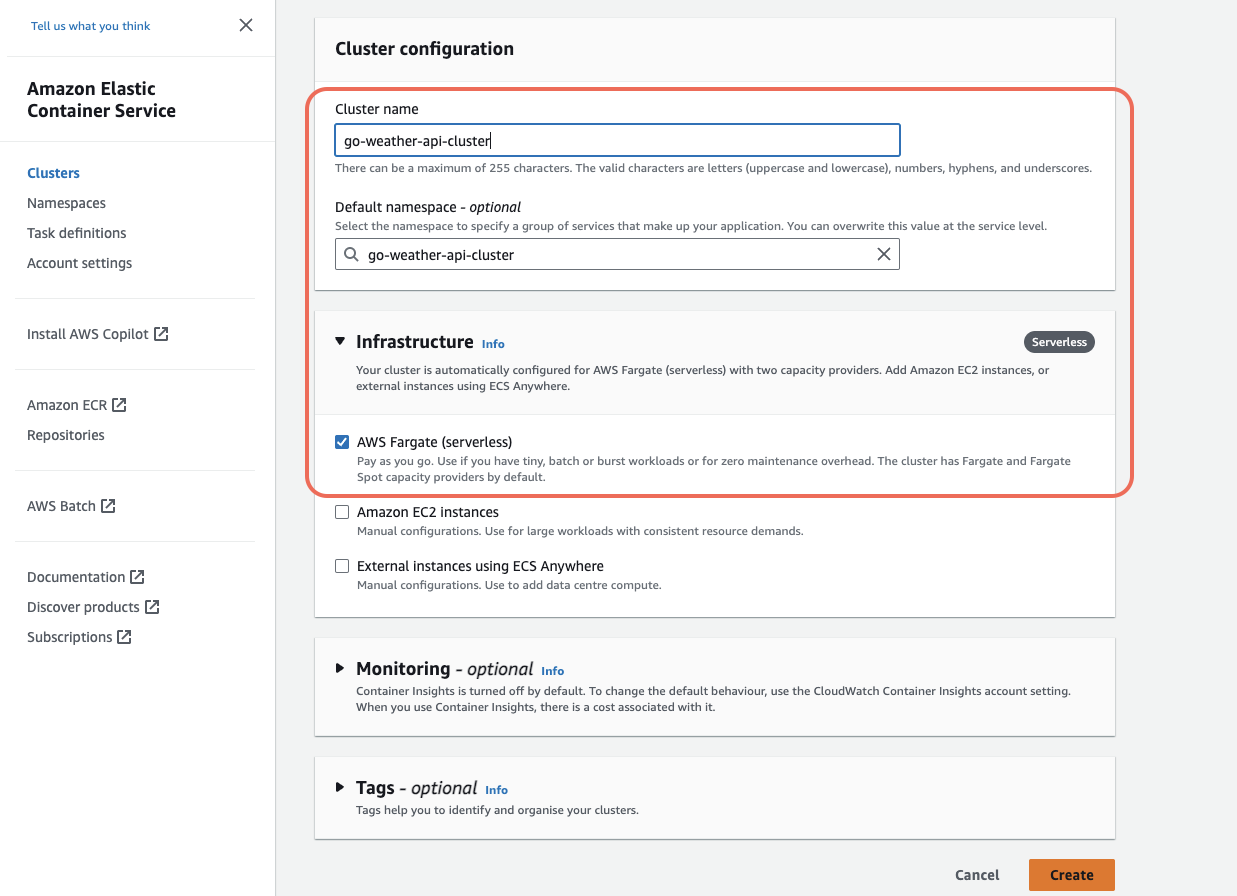
****

**i ) Create a Cluster**

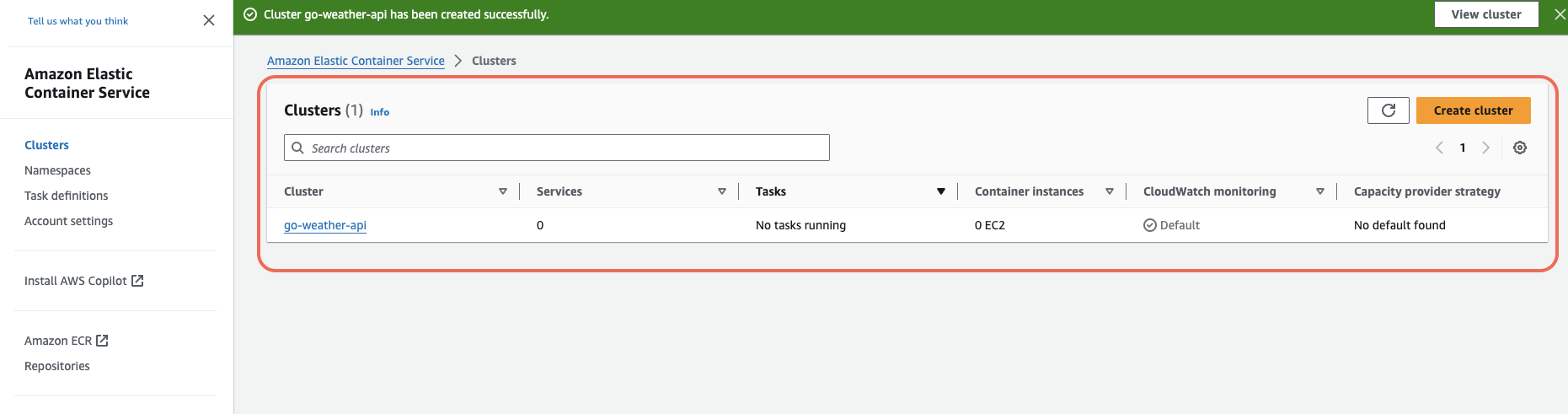
In a cluster we can have multiple services,



**Here we are going to use serless infrastructure, we can also use EC2 but that may cost more**

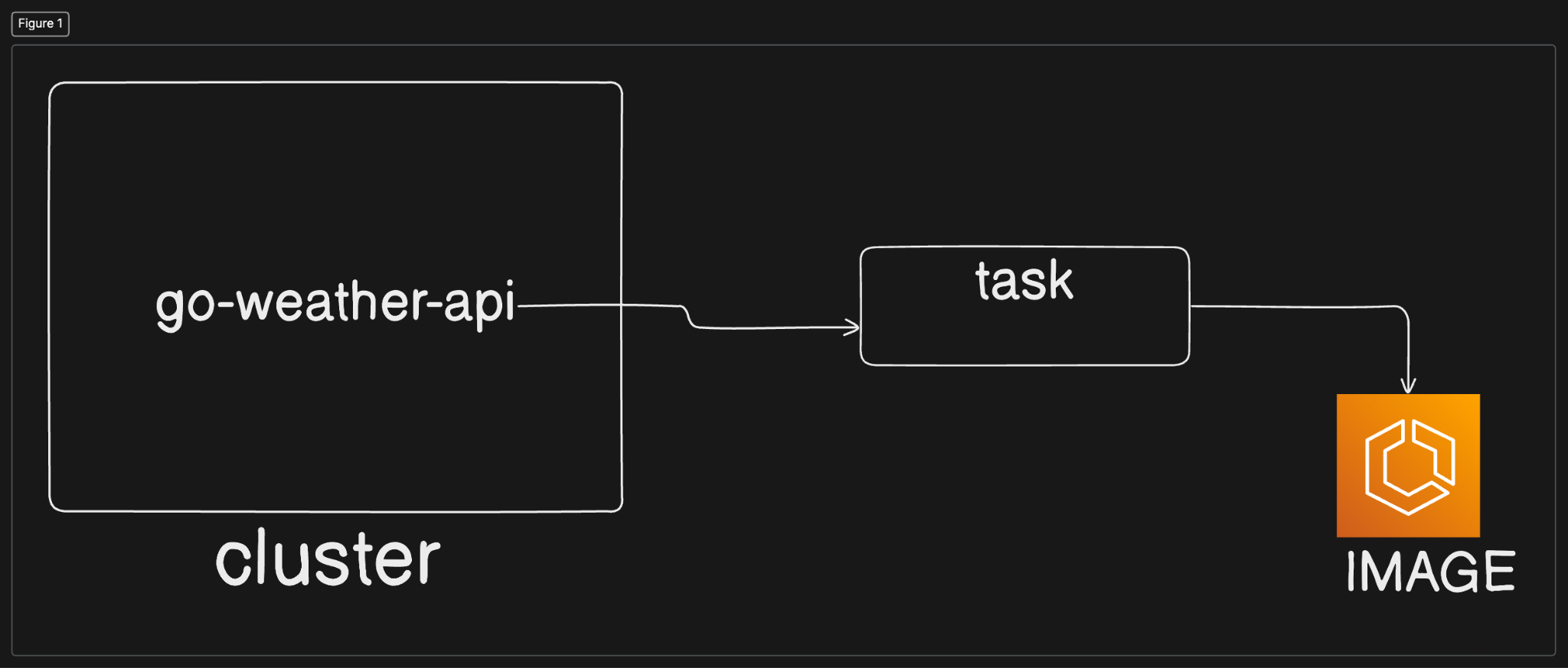
****

**Cluster Created:**

****

**Now we are going to create a task in ECS which would know how to access image and api can access Task**

**Overview of the above statement**

****