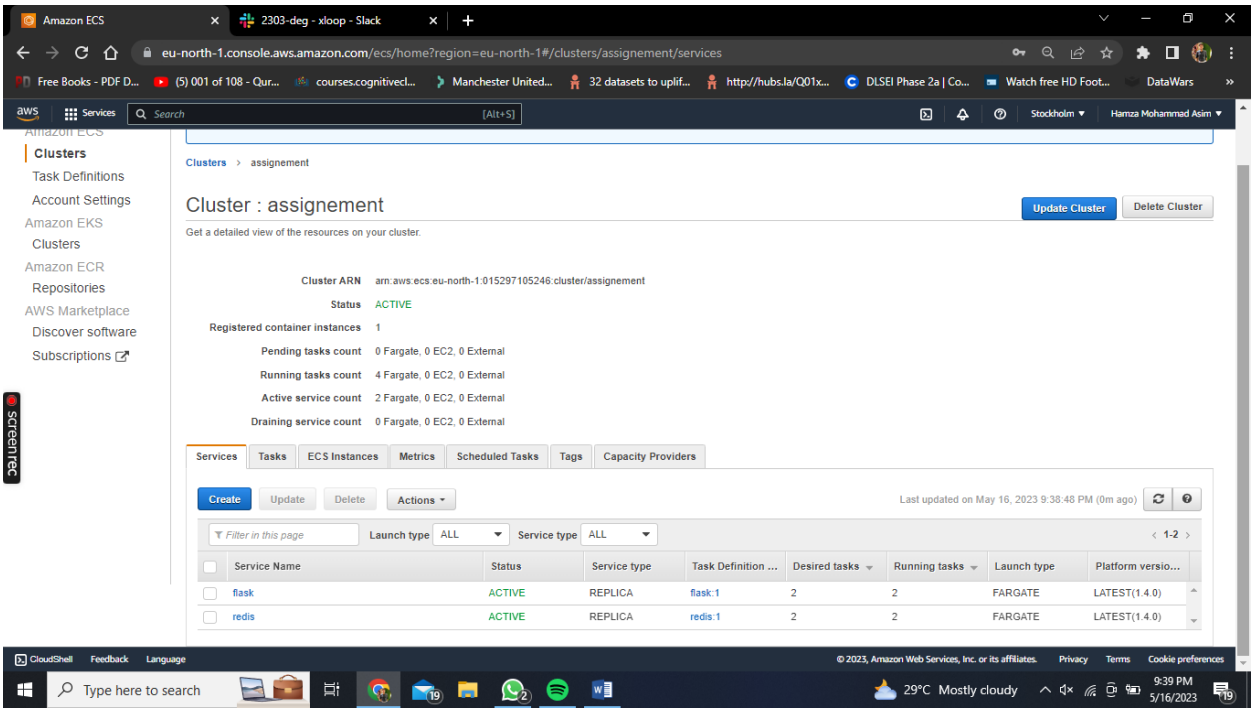
Graded Assignment 4.5

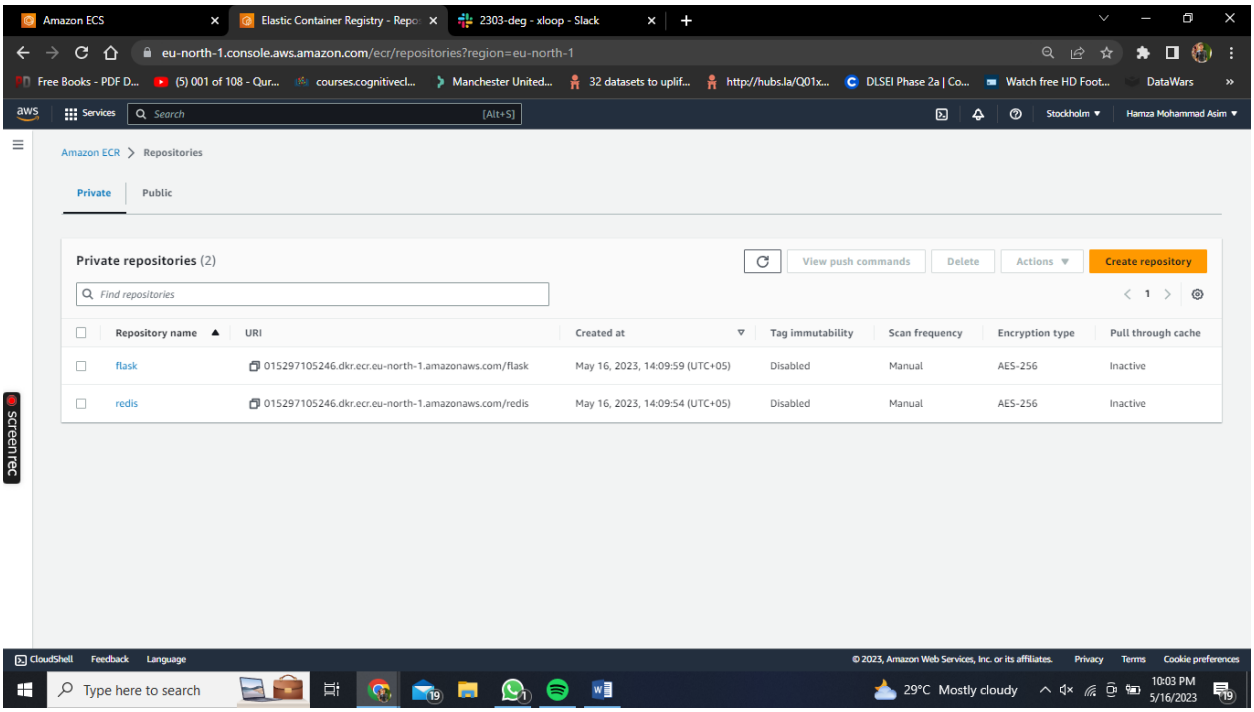
Name: Saad Sameer Khan
Employee#: 2303.KHI.DEG.034
Collaborated with: Mohammad Hamza Asim (2303.KHI.DEG.014)



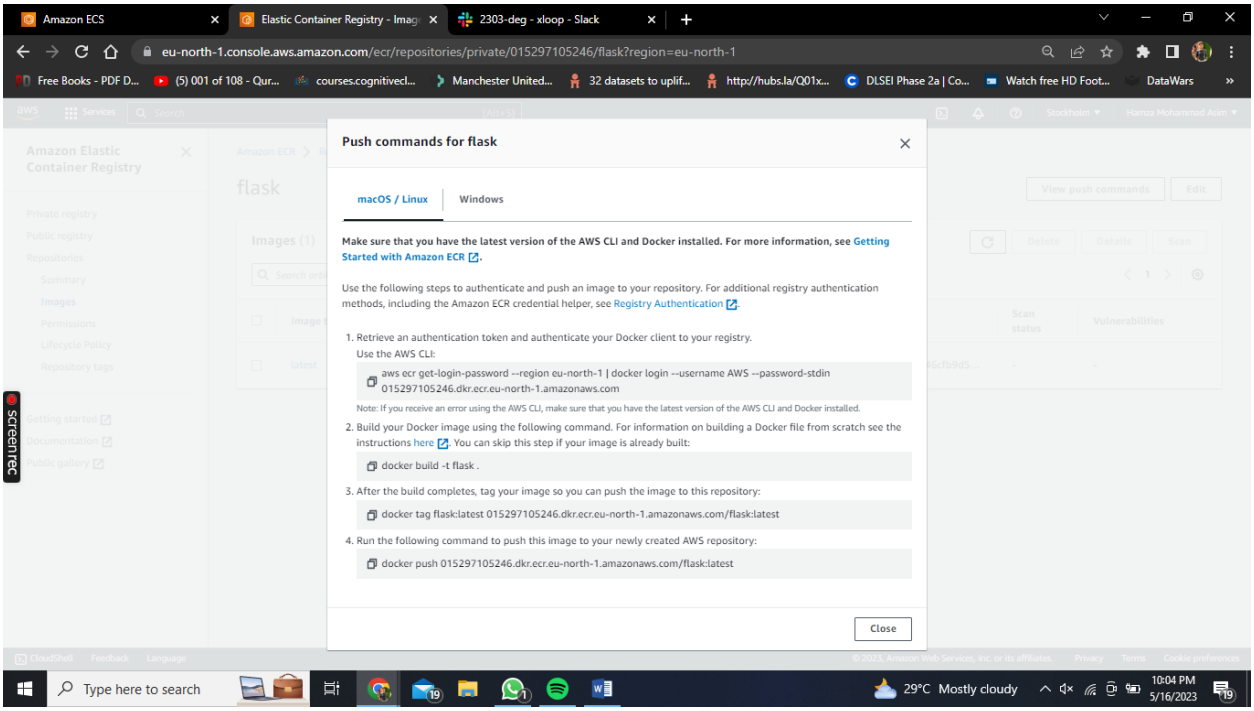
Created a cluster named ‘assignment’ and added 2 services in it. Clusters are basically a group where you are capable of managing and deploying the containers. Services are taking care of the containers where you can see whether it’s in active or inactive state and provide a replica service type in case of failure, which version of Task Definition do you want to run on (which is mentioned later) and so on

The two services added were:

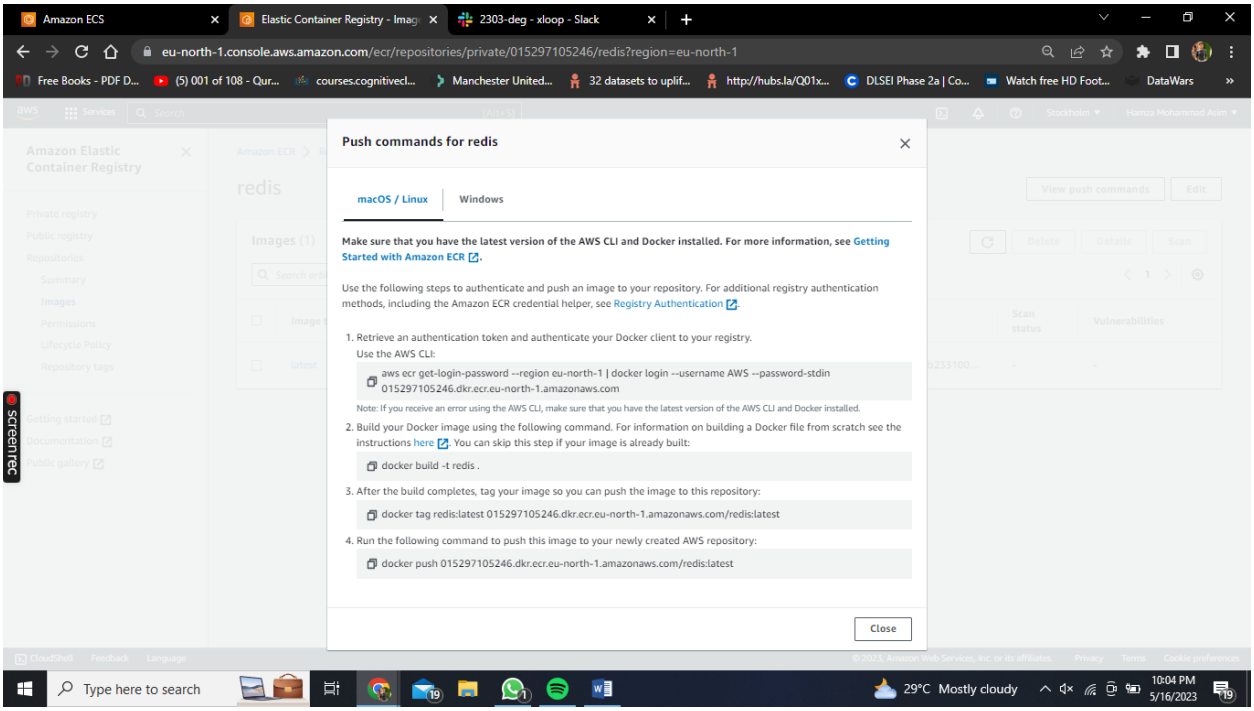
1. ‘flask’: the flask app
2. ‘redis’: redis image



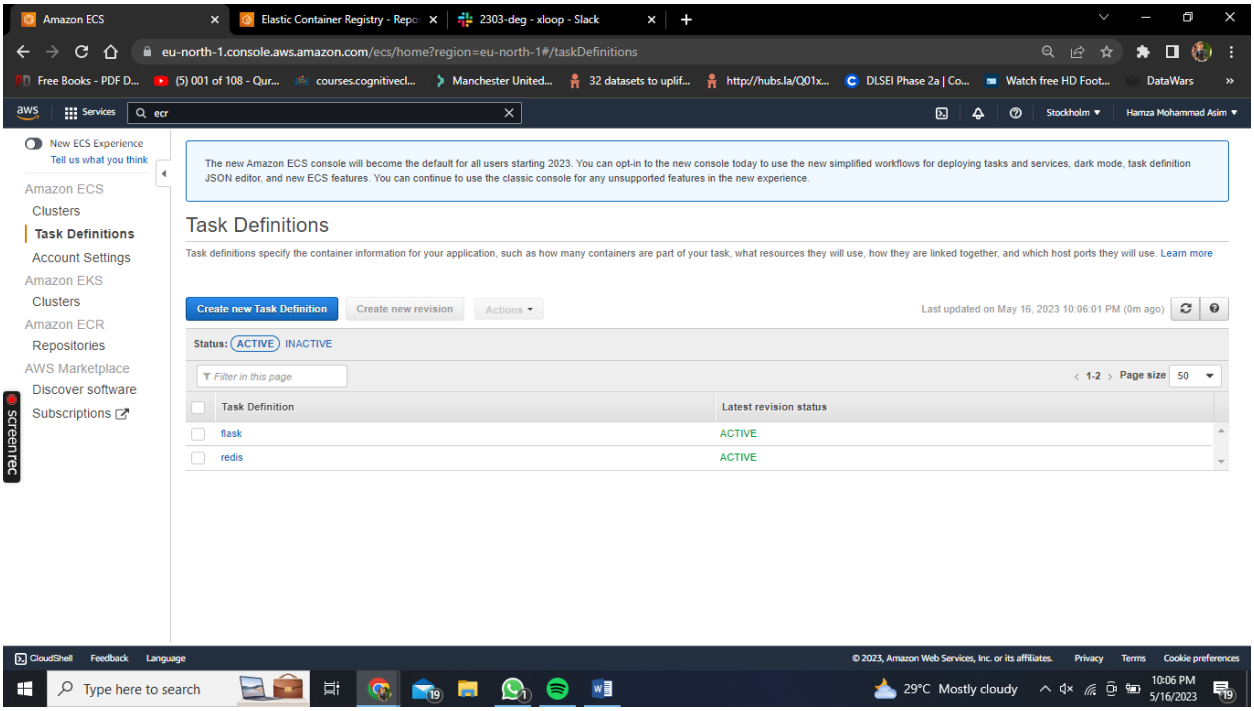
This is ECR (Elastic Container Registry) where we have a repository which allows us to push our image in this AWS registry for various use and purposes. As seen in the picture, two are added one is flask app and other is redis.



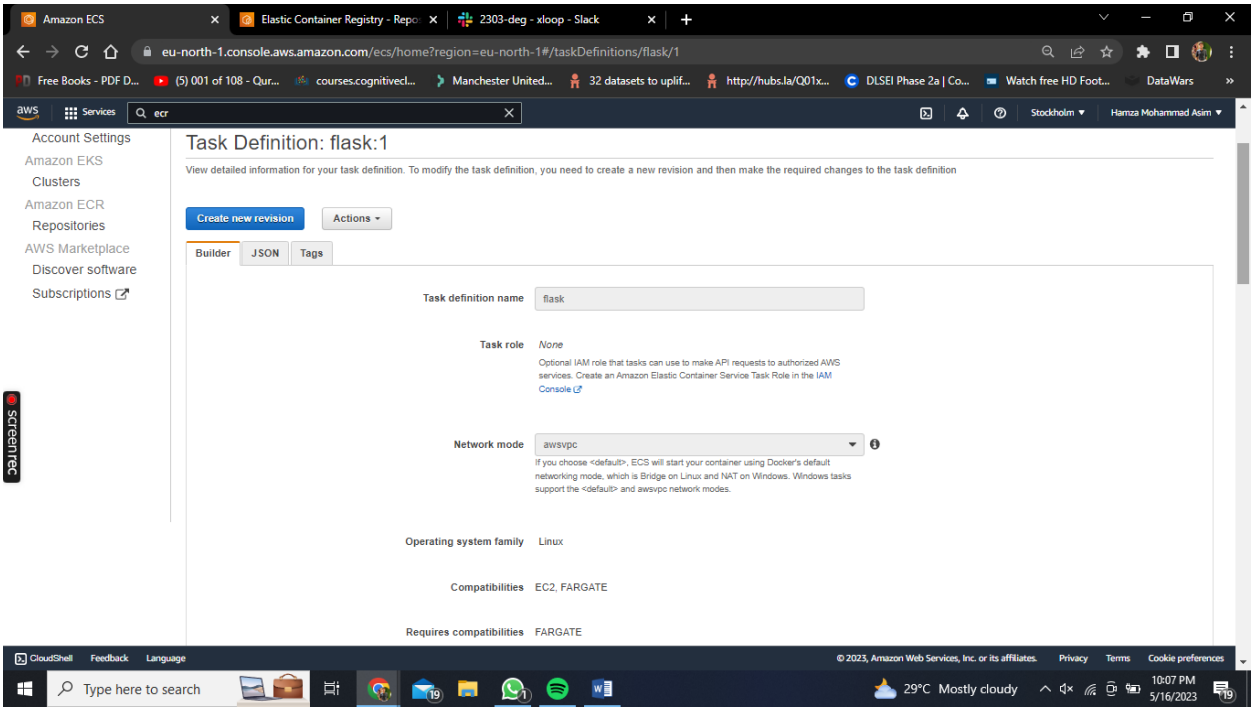
These commands were run in the same directory where the dockerfile was present along with the app.py and requirement.txt for pushing the flask app into the registry ECR



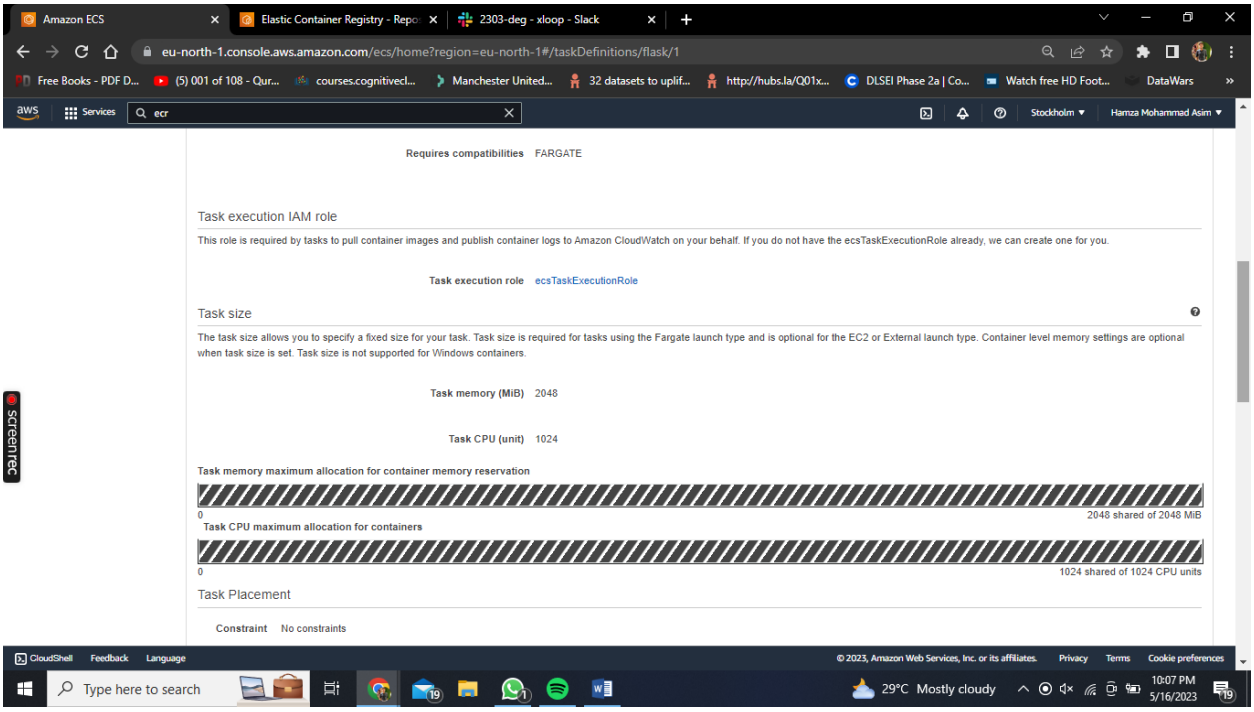
Likewise, same steps were performed for Redis image but the only slight difference in this was that instead of build command we ran ‘docker pull redis’ which allowed us to pull the redis image from dockerhub and then we simply push it in the ECR



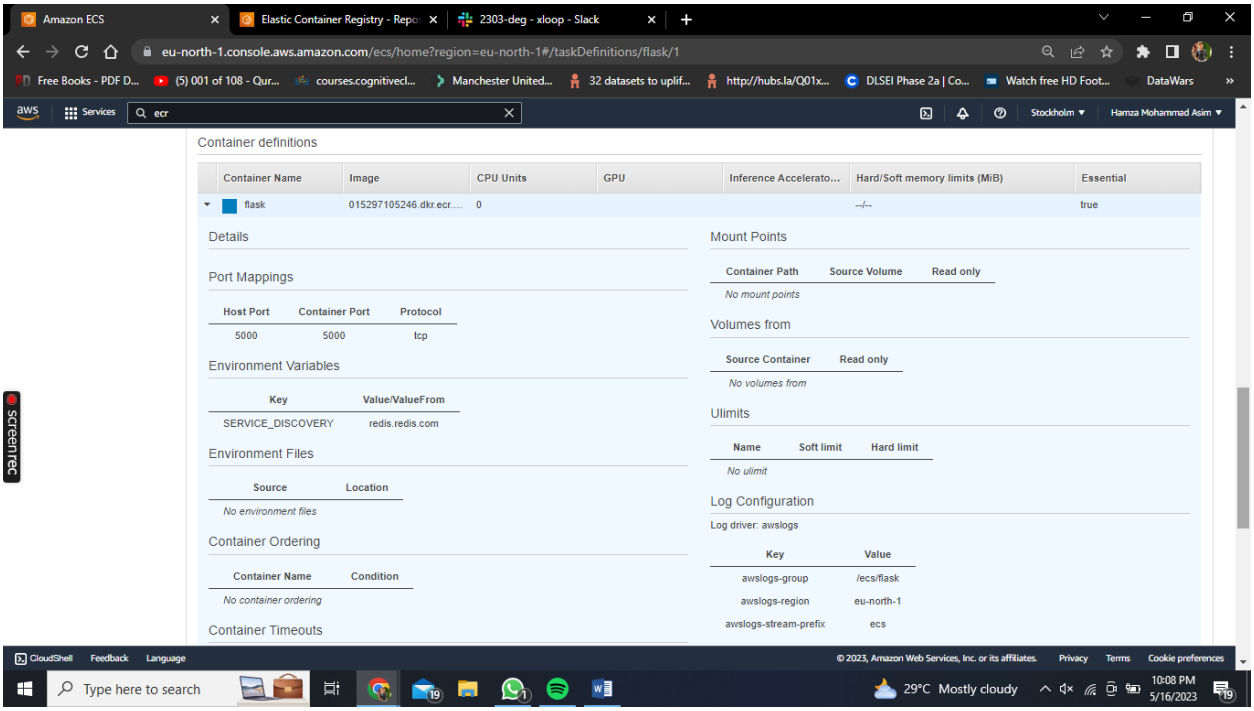
Here we have created the Task Definitions. This section of ECS is very crucial and should be carefully looked at as we move on to specify important details such as the container image, CPU and memory requirements, networking settings. It's like a blueprint that guides ECS on how to create and manage your containers effectively.



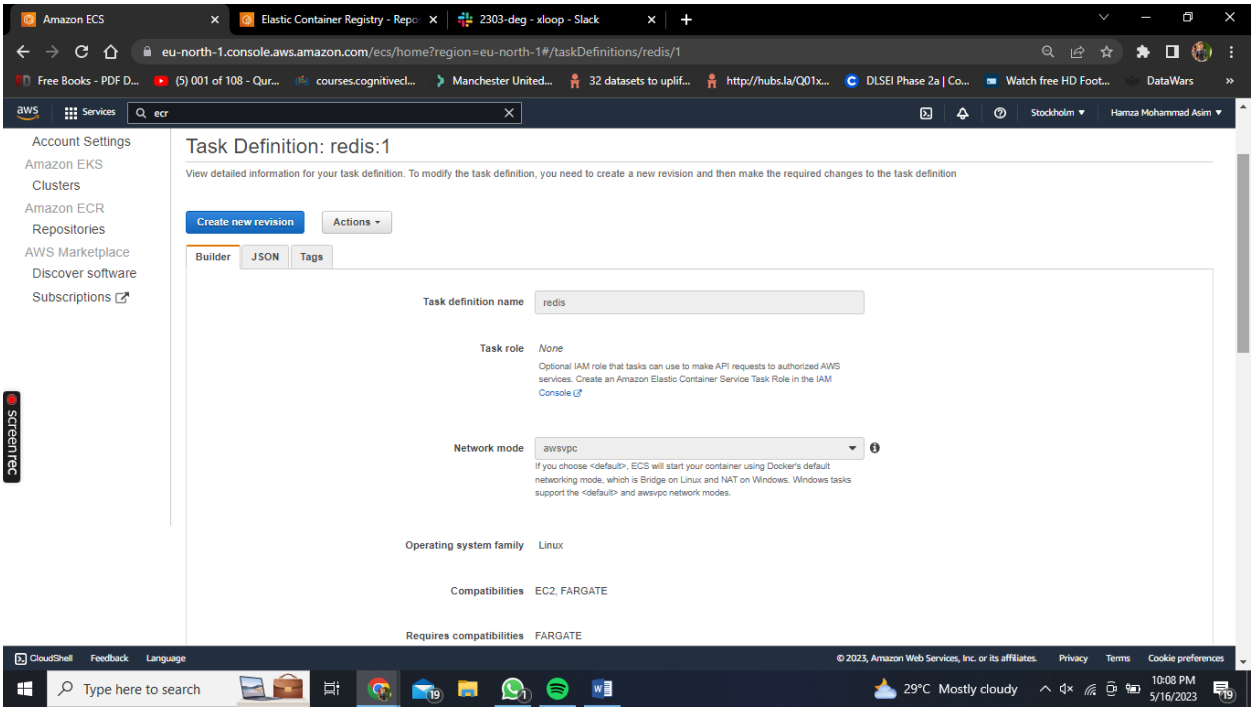
Here is the task definition of flask. Fargate is chosen for compatibility as it makes our lives much easier when it comes to management of underlying compute resources which are required to run a container



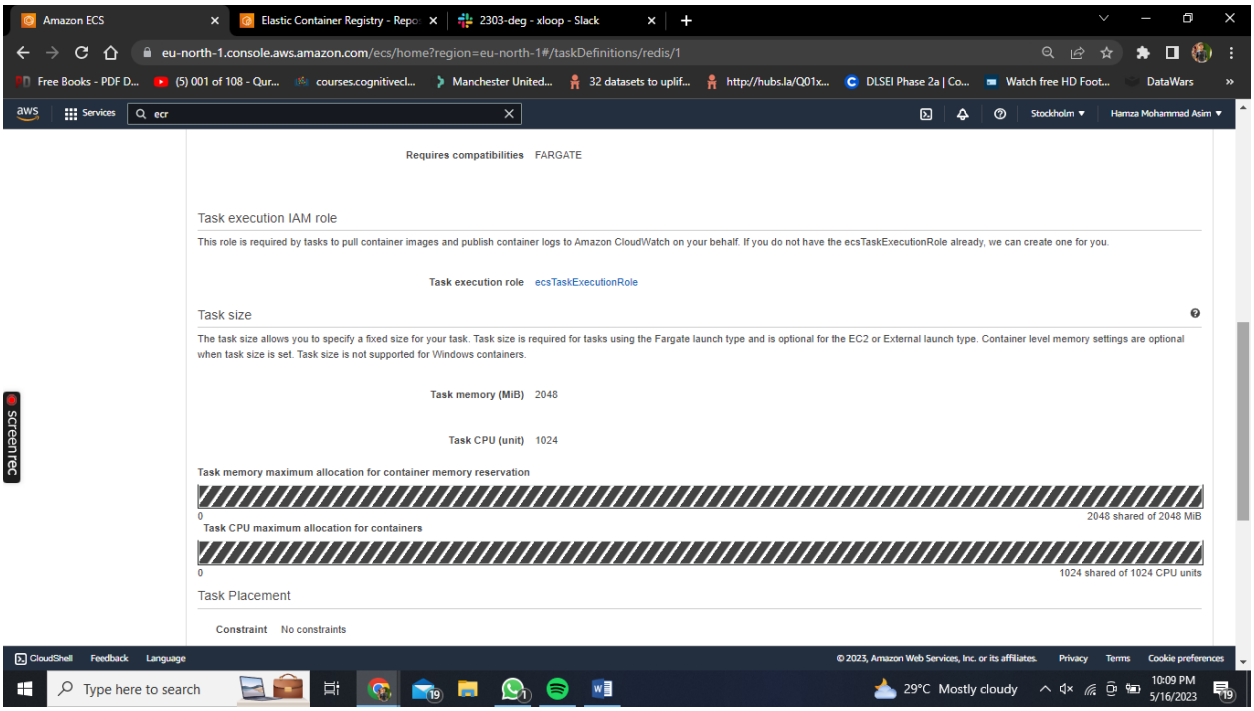
Resource allocation was provided of 2GB RAM and 1vCPU

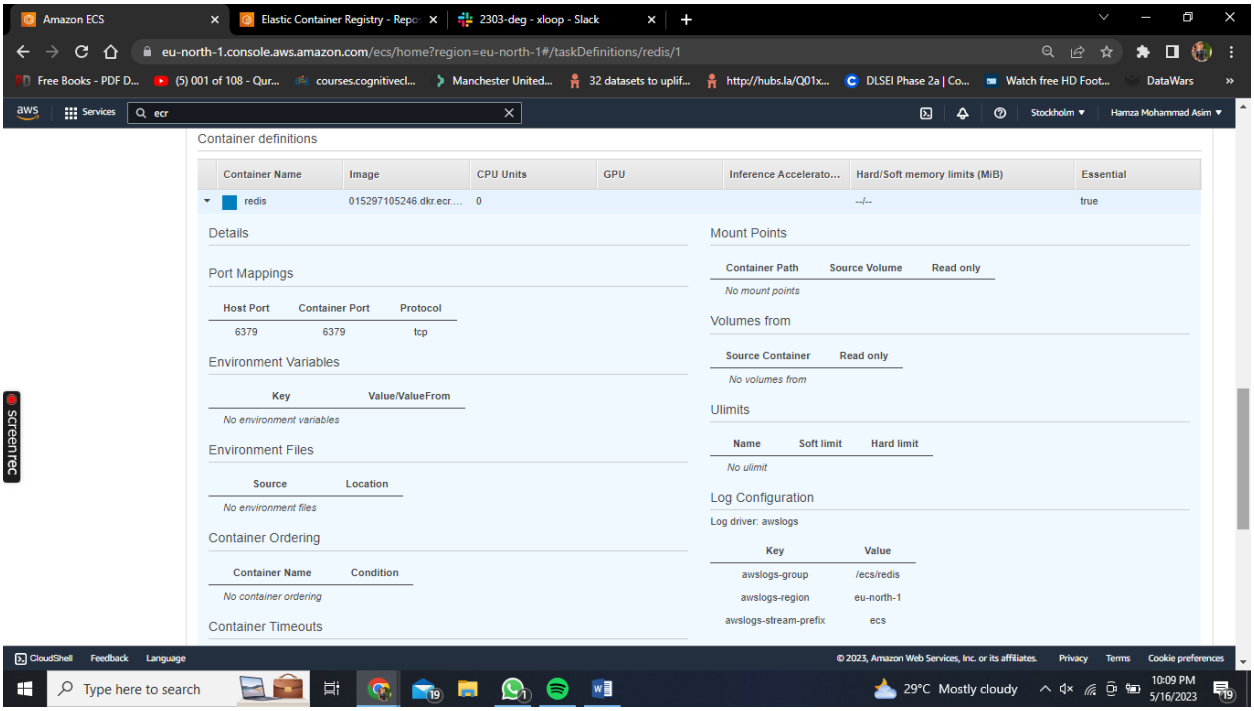


This is the container which is added and the image url was provided the same from ECR where we recently pushed our app. Port was provided 5000. Environment Variables were provided with key ‘SERVICE_DISCOVERY’ which allows us to host our flask and easily locate the redis container.

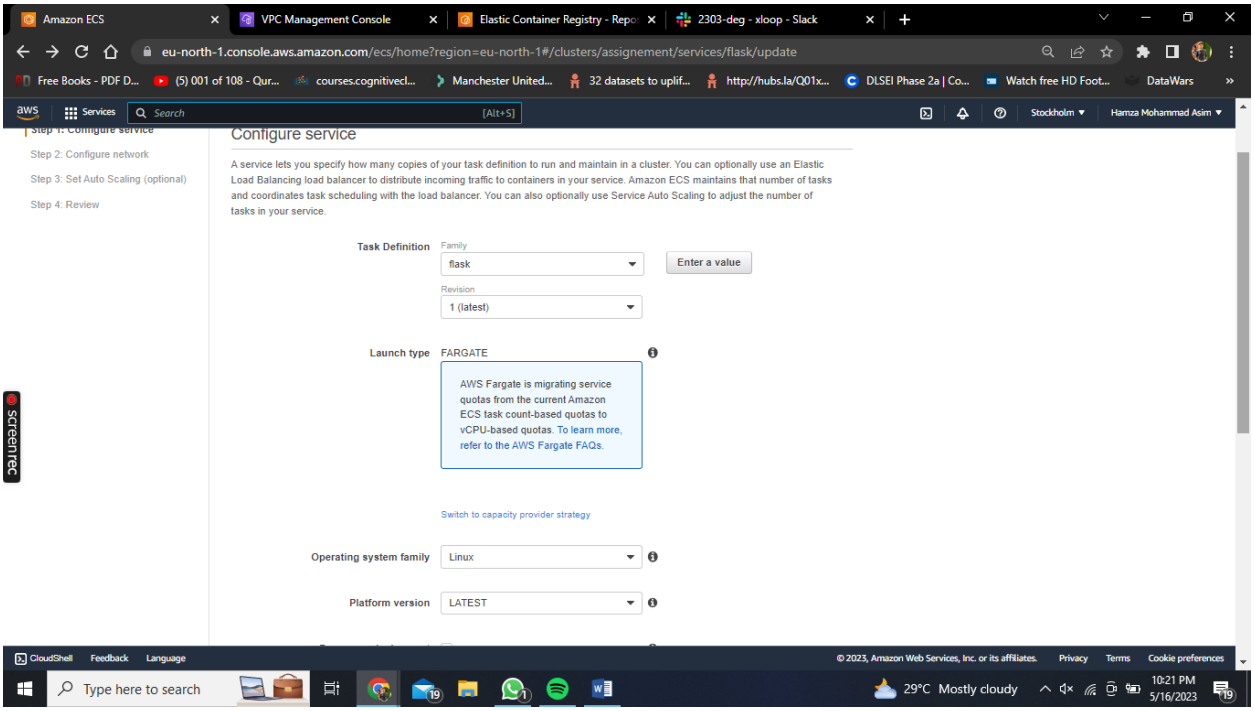


Likewise, same settings were provided to Redis Task definition as well.

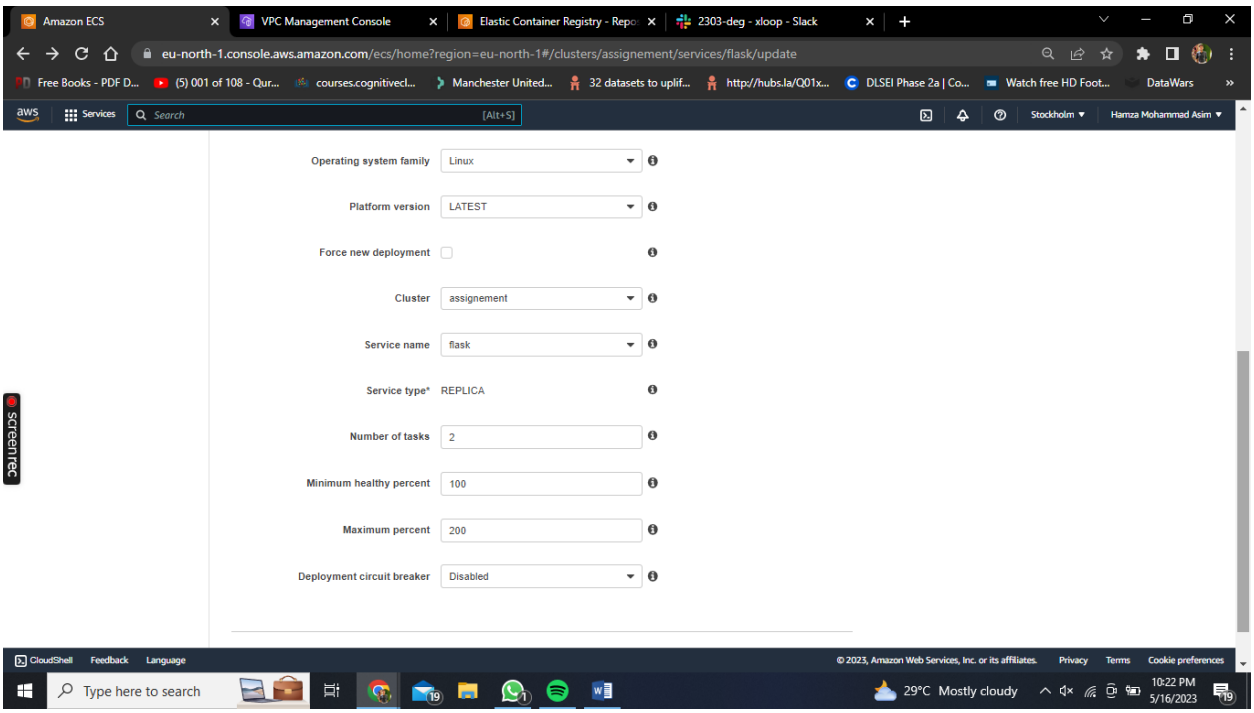


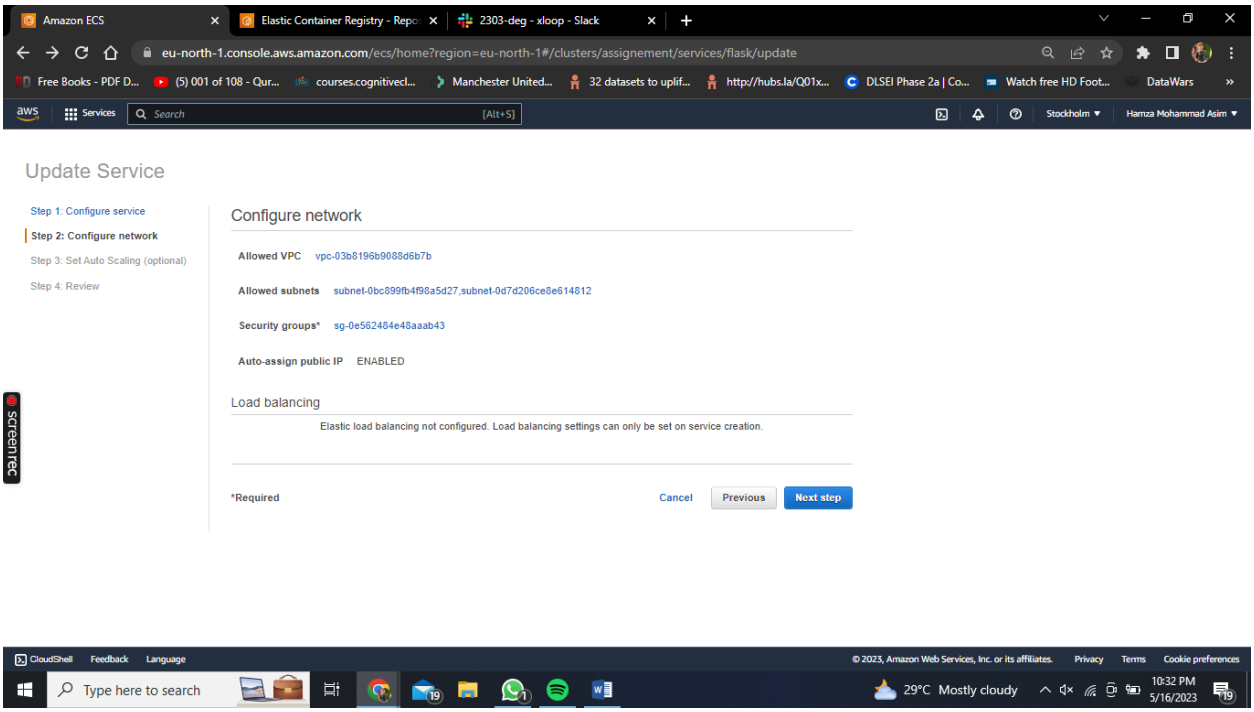


Port mentioned above: 6379

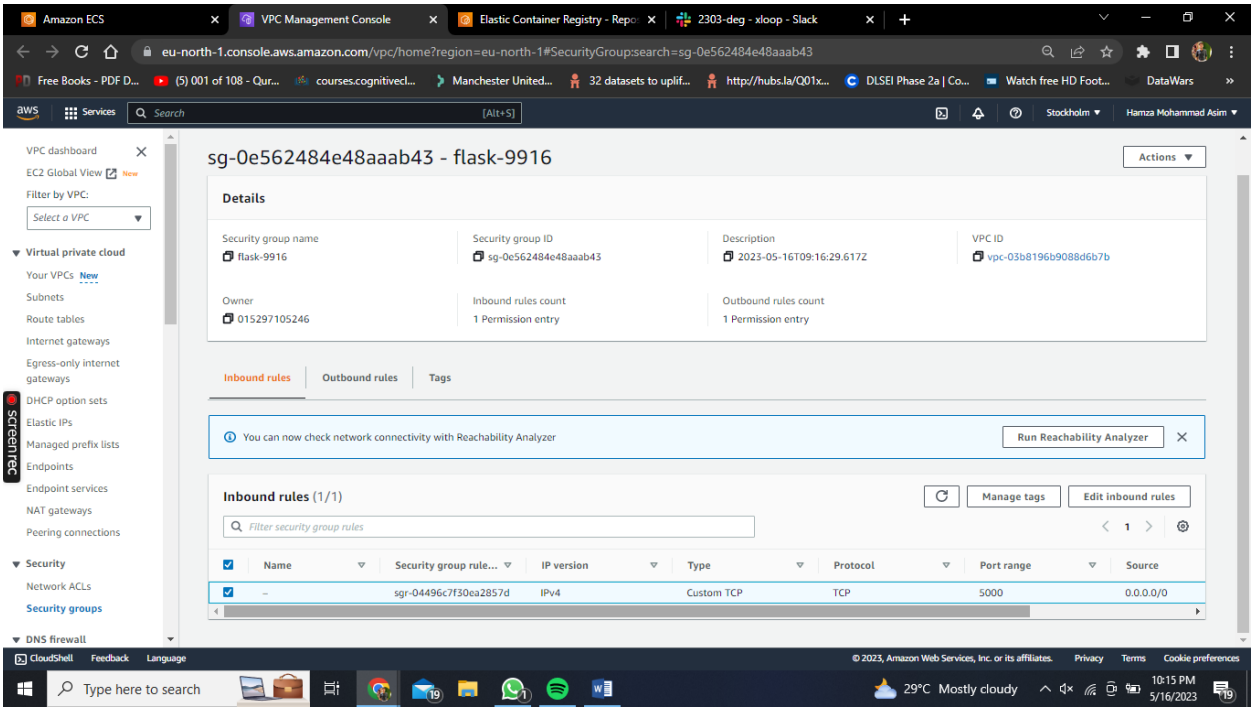


Now we're back in ECS to configure our services after creating Definition Tasks. The service that we created recently of both flask and redis, we'll specify those definition tasks while configuring service in both services respectively.

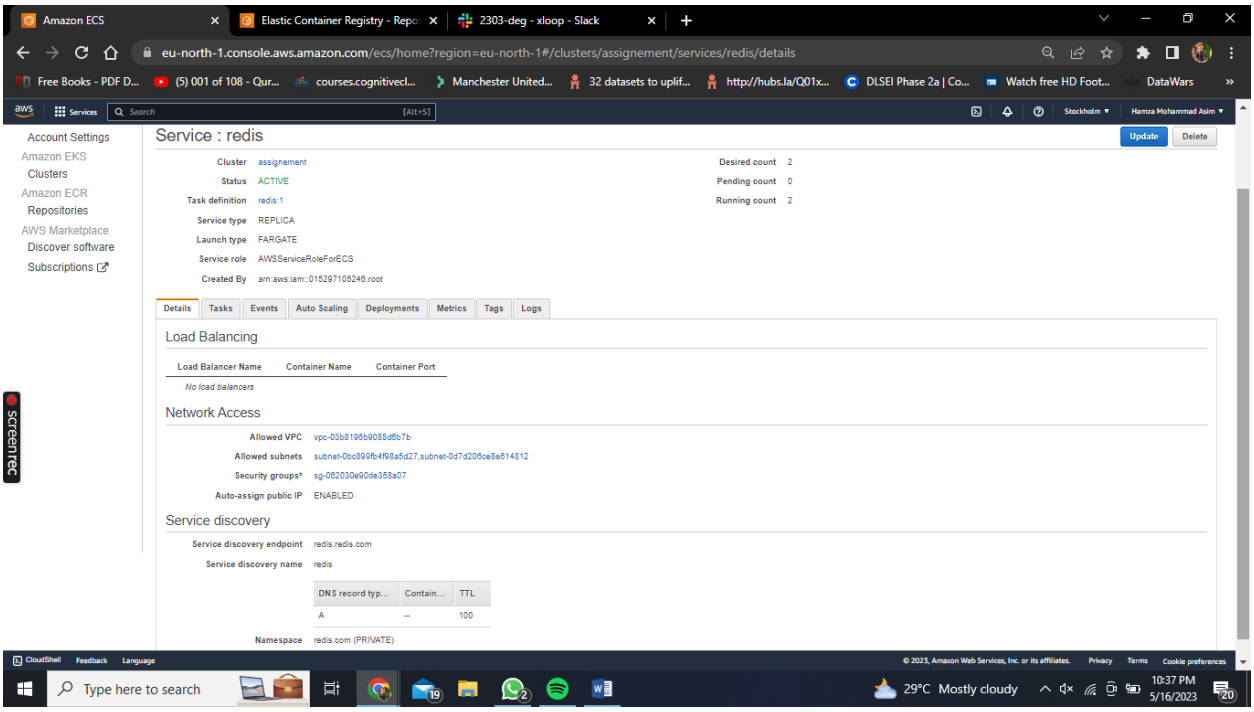




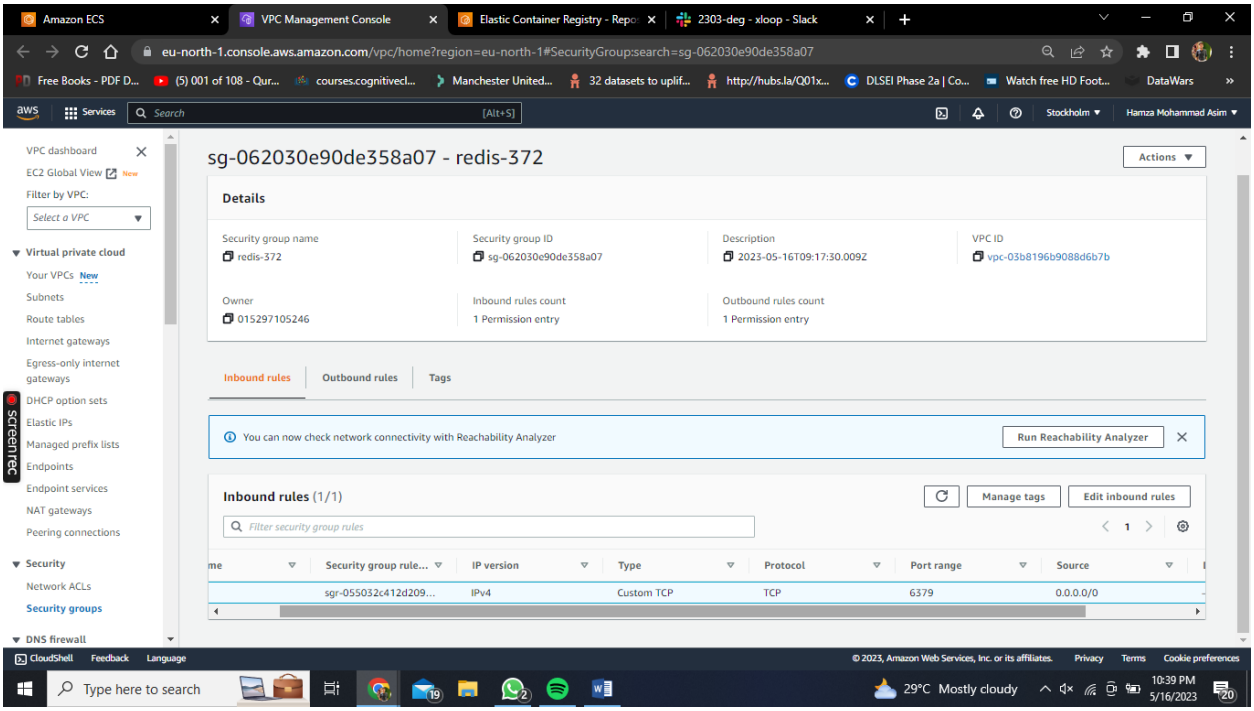
VPC is chosen same as cluster's. we chose all available subnets.



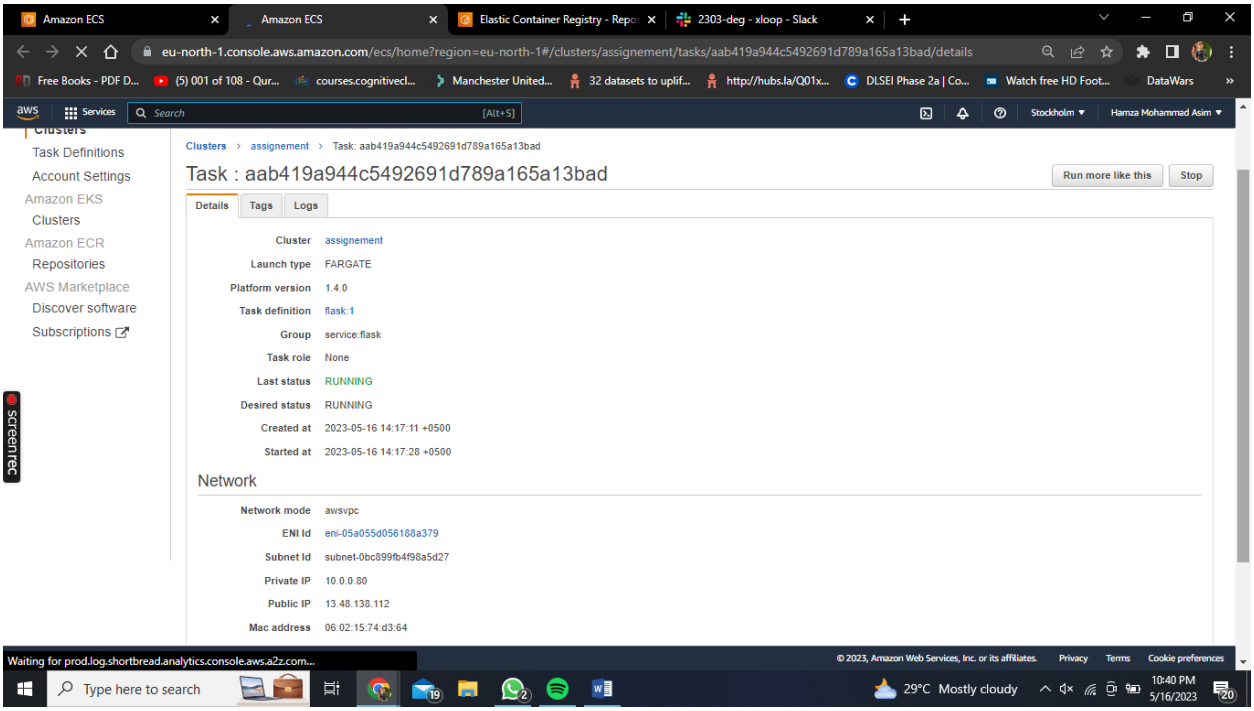
Security group inbound rule was added with port range 5000



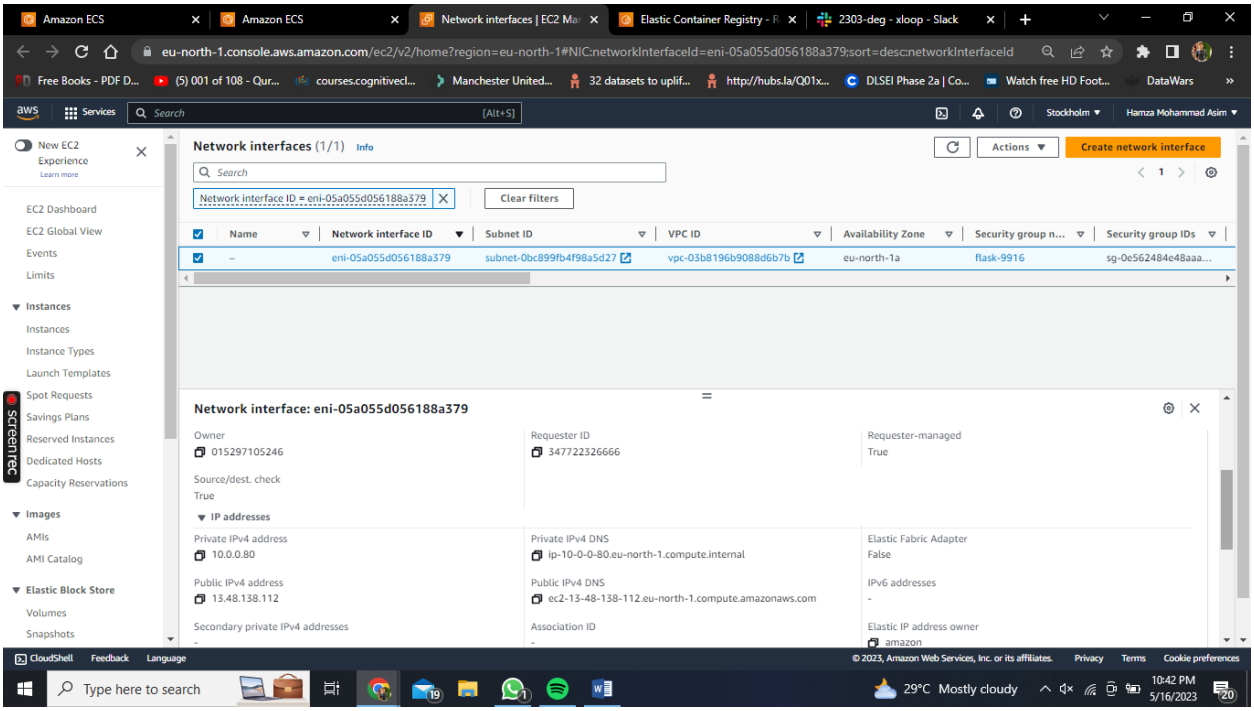
Likewise, same done for redis. Note that namespace was mentioned over here which was created in AWS cloudmap named : redis.com and service name : redis.
Service discovery endpoint: redis.redis.com



Security group inbound rule was added with port range 6379

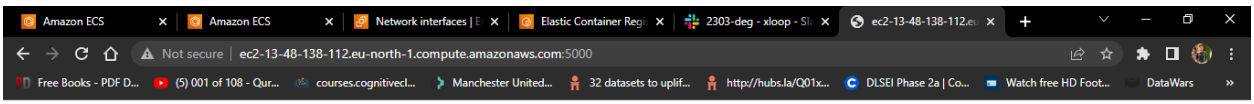


After setting up everything, In this step we will go to our cluster and in flask service. We'll open up a task and click on ENI ID

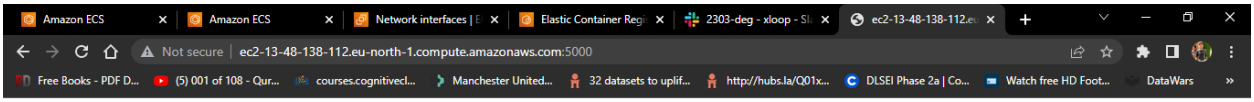


After ticking the Network interface above, a box will open up where we can copy our Public IPv4 DNS and copy on a new tab along with the port 5000. Results are attached below.

‘ http://ec2-13-48-138-112.eu-north-1.compute.amazonaws.com:5000/ ’



Hello World! I have been seen 4 times.



Hello World! I have been seen 4 times.