CONTAINER ORCHESTRATION AND INFRASTRUCTURE AUTOMATION

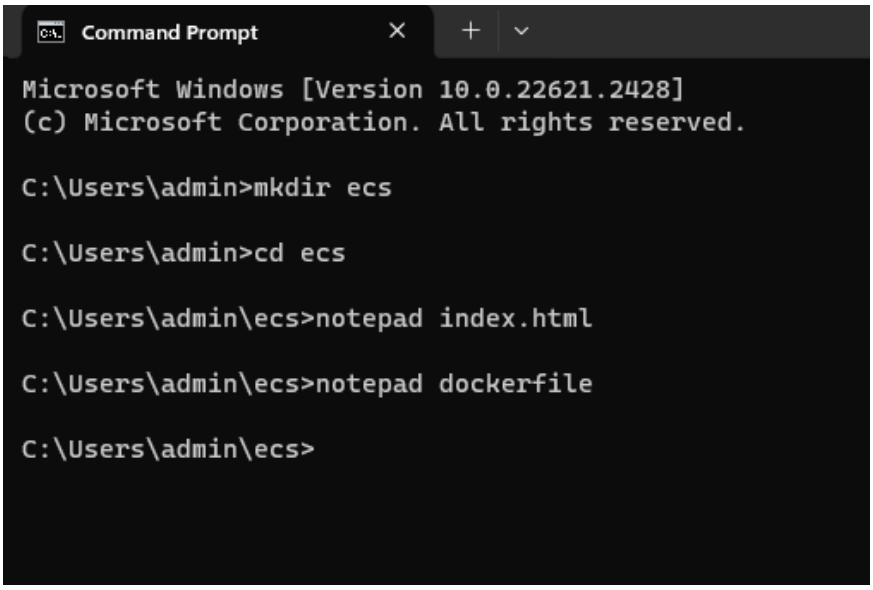
NAME- Shubhi Dixit

BATCH- 05

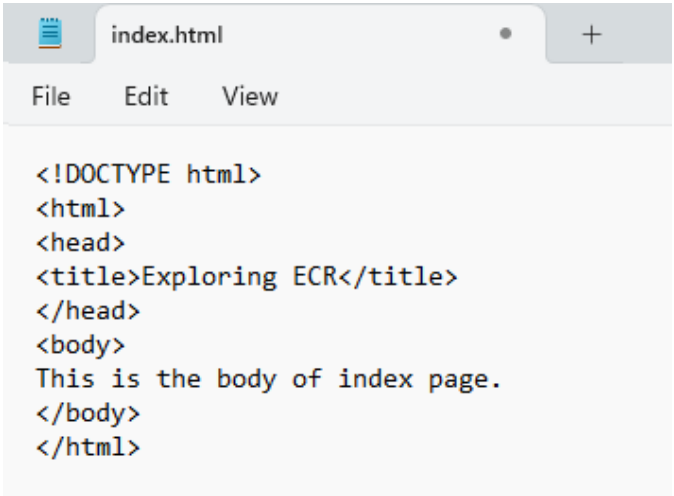
SAP ID- 500094571

ECS Experiment

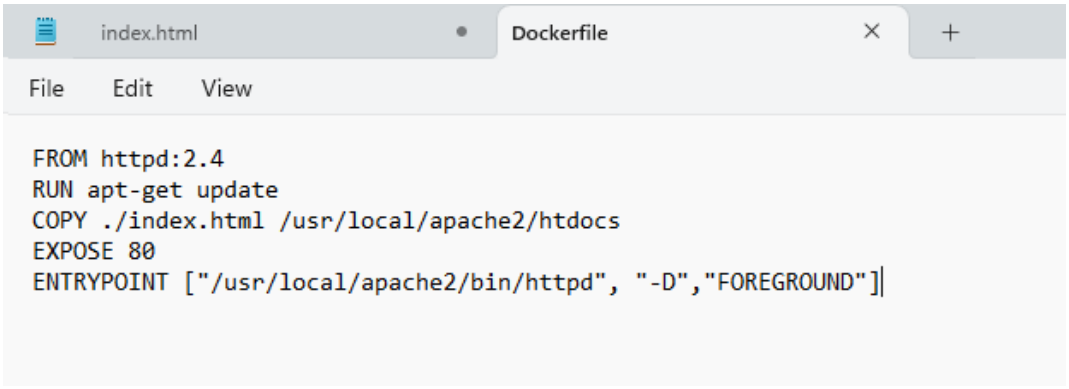
Step: 1- Firstly using the dockerfile, build the image of the setup.



index.html



dockerfile




```
C:\Users\admin\ecs>docker build -t htmlimg .
[+] Building 25.1s (8/8) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 193B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/library/httpd:2.4
 => [1/3] FROM docker.io/library/httpd:2.4@sha256:7b8098fbbdaa8d12c3ae2208b249e7adbe12376fe8d86ef2aae27ce1f13b7167
 => => resolve docker.io/library/httpd:2.4@sha256:7b8098fbbdaa8d12c3ae2208b249e7adbe12376fe8d86ef2aae27ce1f13b7167
 => => sha256:7b8098fbbdaa8d12c3ae2208b249e7adbe12376fe8d86ef2aae27ce1f13b7167 1.86kB / 1.86kB
 => => sha256:d70861224a52d0175c4b485ef7b70f56125f2c74d8a489a9a8ca10e3656929c9 1.37kB / 1.37kB
 => => sha256:7f6a969e81a54d01e2fc08fd45025badf8f2418d3530a3a484b77ab826a1b4a7 9.09kB / 9.09kB
 => => sha256:578acb154839e9d0034432e8f53756d6f53ba62cf8c7ea5218a2476bf5b58fc9 29.15MB / 29.15MB
 => => sha256:c1a8c8567b78fc222c60d83831081d5578fd80e6c30ee0bcf8a79019824301a1 178B / 178B
 => => sha256:10b9ab03bf458e12adc435ae235b1f58f473fea23c2cdf6d88407d92a4a51c41 4.20MB / 4.20MB
 => => sha256:74dbedf7ddc012b3398b2c645974f1f6e30a9592fef541d02faf456e40b67bc4 31.40MB / 31.40MB
 => => sha256:6a3b76b70f7385a99e72877fac4d1e00f4466859ed3ea50891bab80908e22c0b 296B / 296B
 => => extracting sha256:578acb154839e9d0034432e8f53756d6f53ba62cf8c7ea5218a2476bf5b58fc9
 => => extracting sha256:c1a8c8567b78fc222c60d83831081d5578fd80e6c30ee0bcf8a79019824301a1
 => => extracting sha256:10b9ab03bf458e12adc435ae235b1f58f473fea23c2cdf6d88407d92a4a51c41
 => => extracting sha256:74dbedf7ddc012b3398b2c645974f1f6e30a9592fef541d02faf456e40b67bc4
```

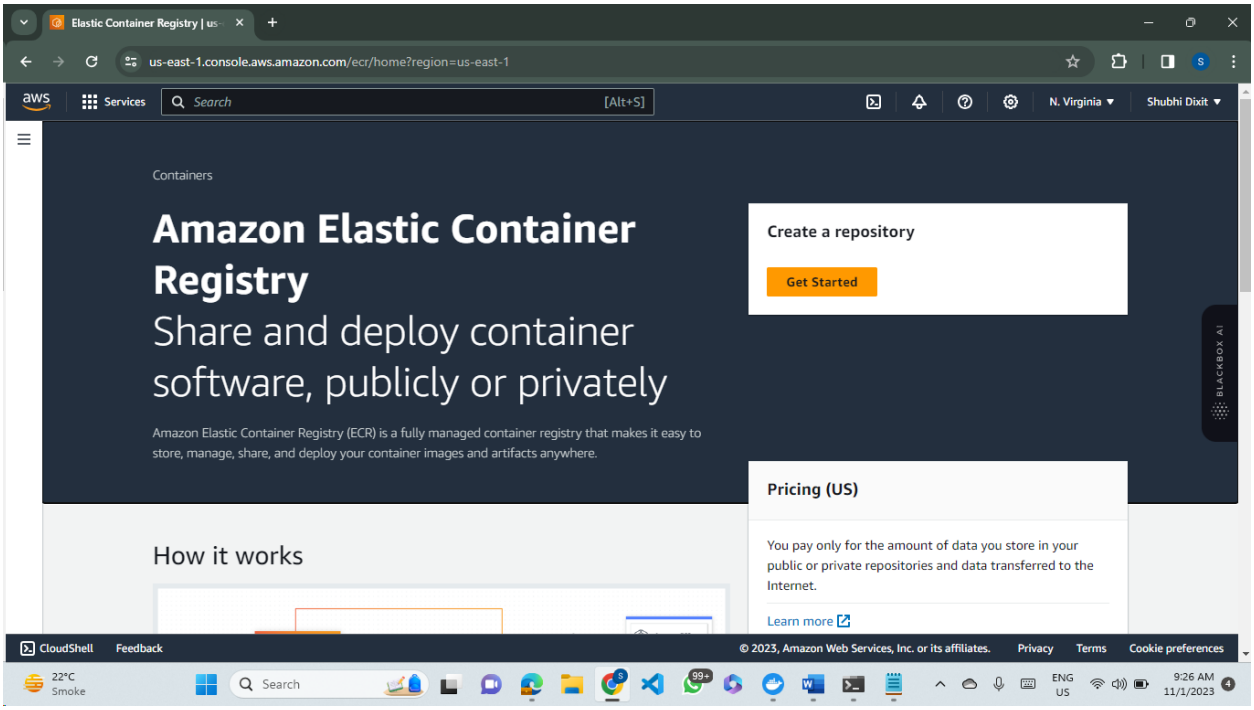
Step-2: After successfully building the image from the container, access the application at localhost: 3000

```
C:\Users\admin>docker run -p 3000:80 htmlimg
```

If this is working fine, then follow the next steps.

Steps:

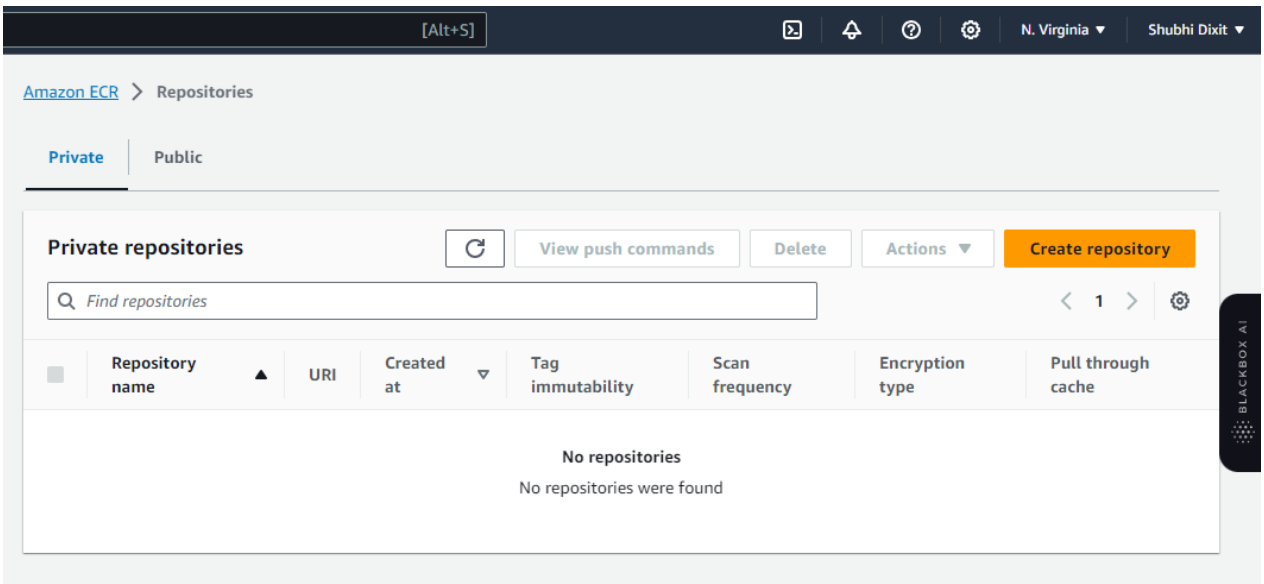
1) Now log in to your aws account and search for ECS



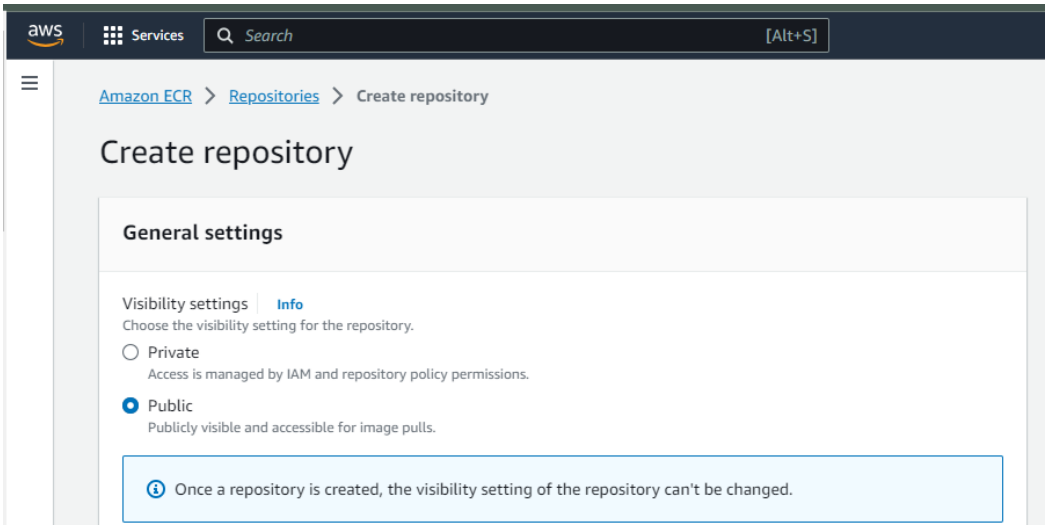
2) First thing that we need to do is to get that image that we created locally using dockerfile into AWS.

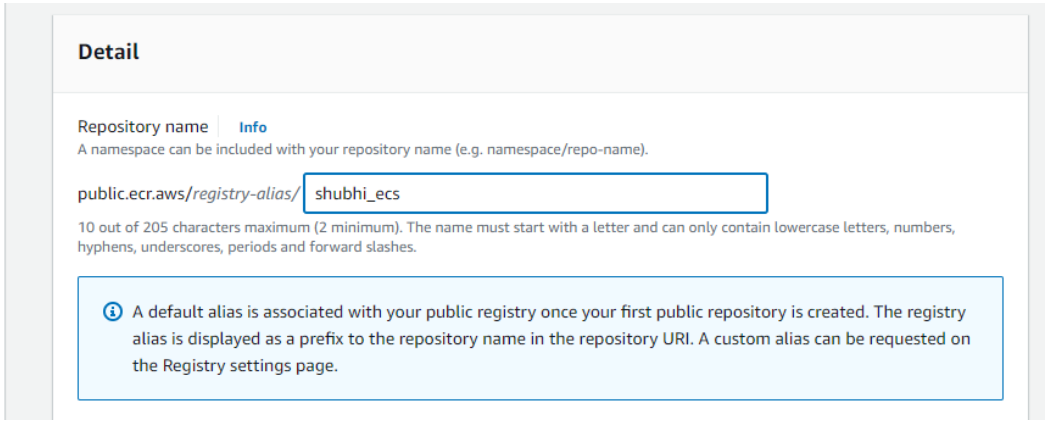
3) It will be done through Amazon ECR (Elastic Container Registry) where we register all our images. This is similar to something like docker hub where we store our images.

4) Go to “Repositories” and click on “Create repository”.

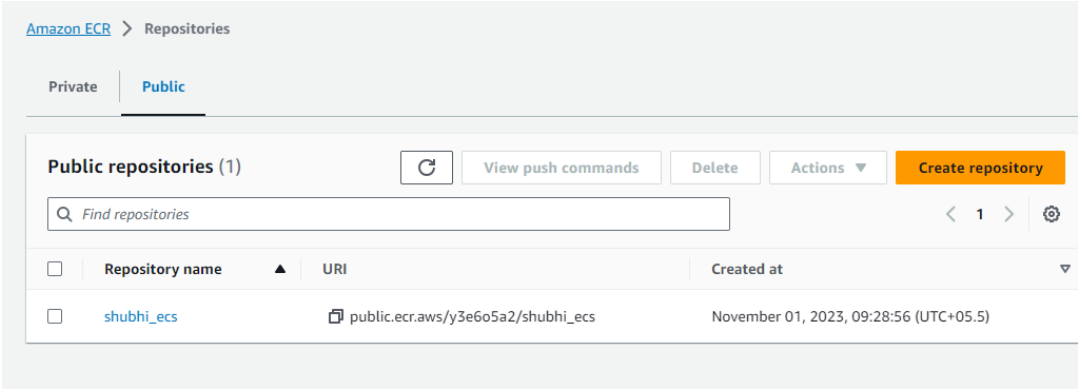


5) Make it a “Public repository” and give your repository a name



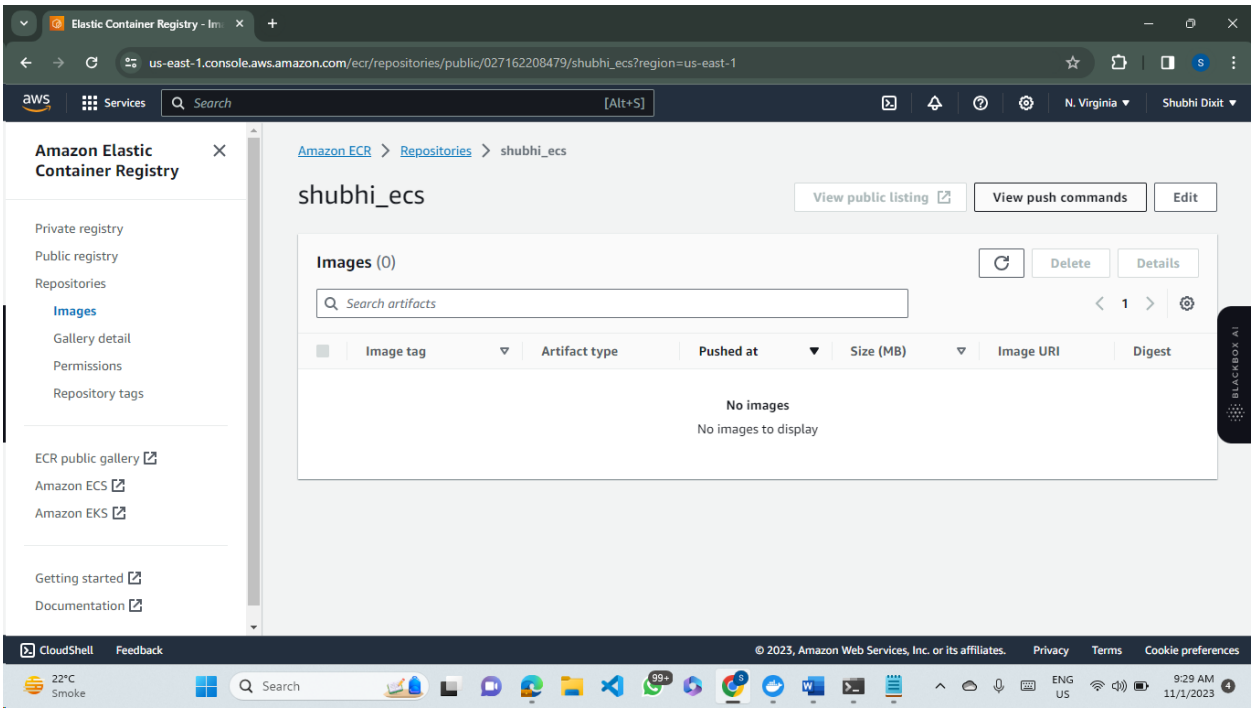


6) Then click on create repository and proceed. Repository is created.

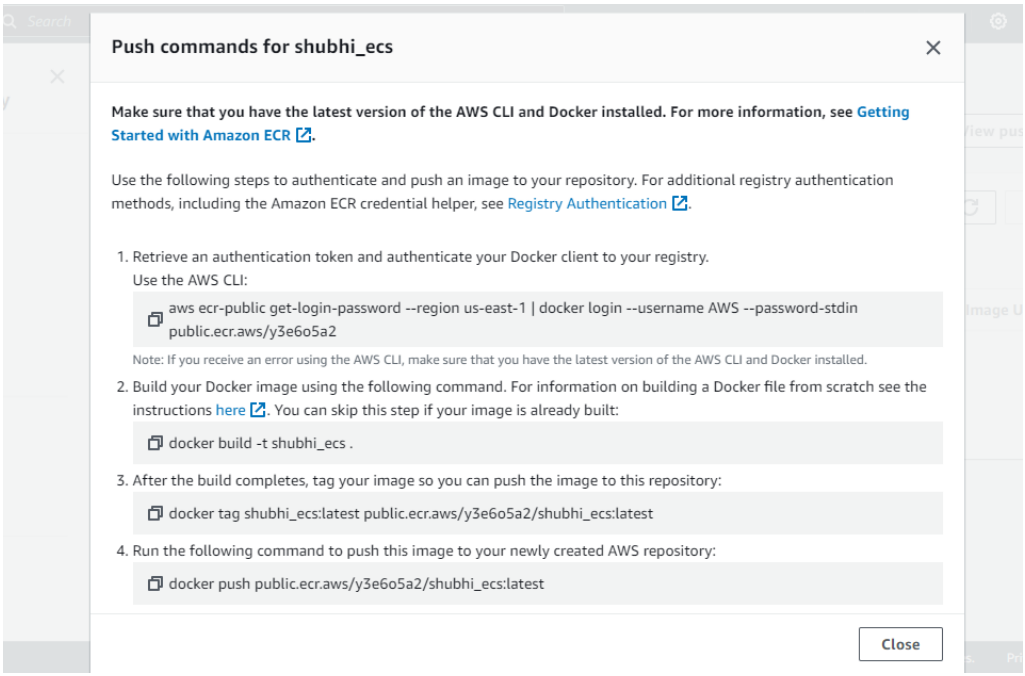


7) This is just the repository. Now what we need to do is to push our docker image into the repository just we just created.

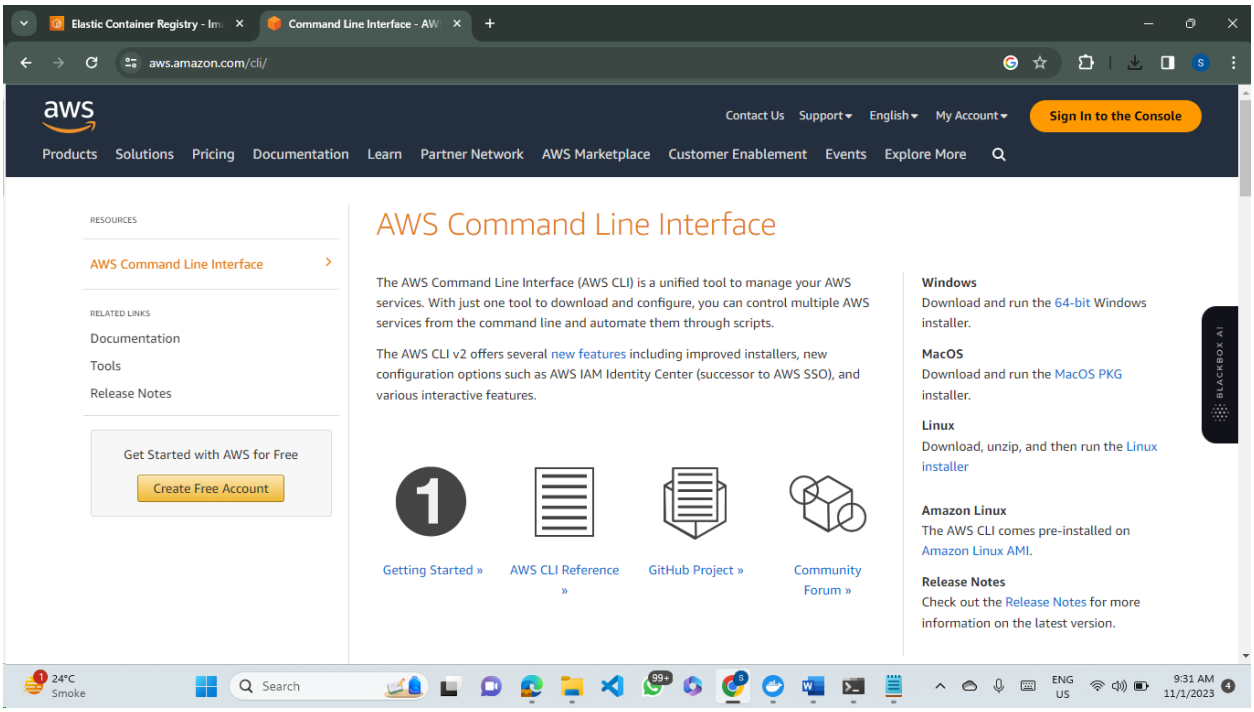
8) Click on repository that you just created.

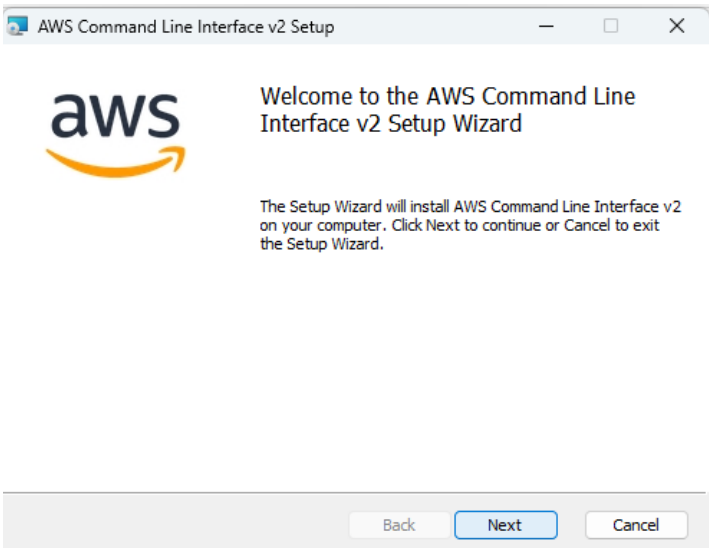


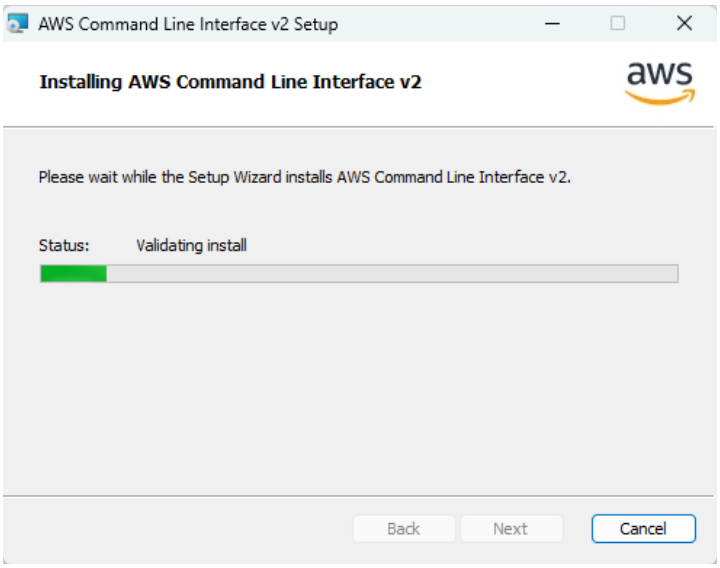
9) Click on view push commands on top right corner. This contains all the commands in order to push the local image into this repository on AWS.



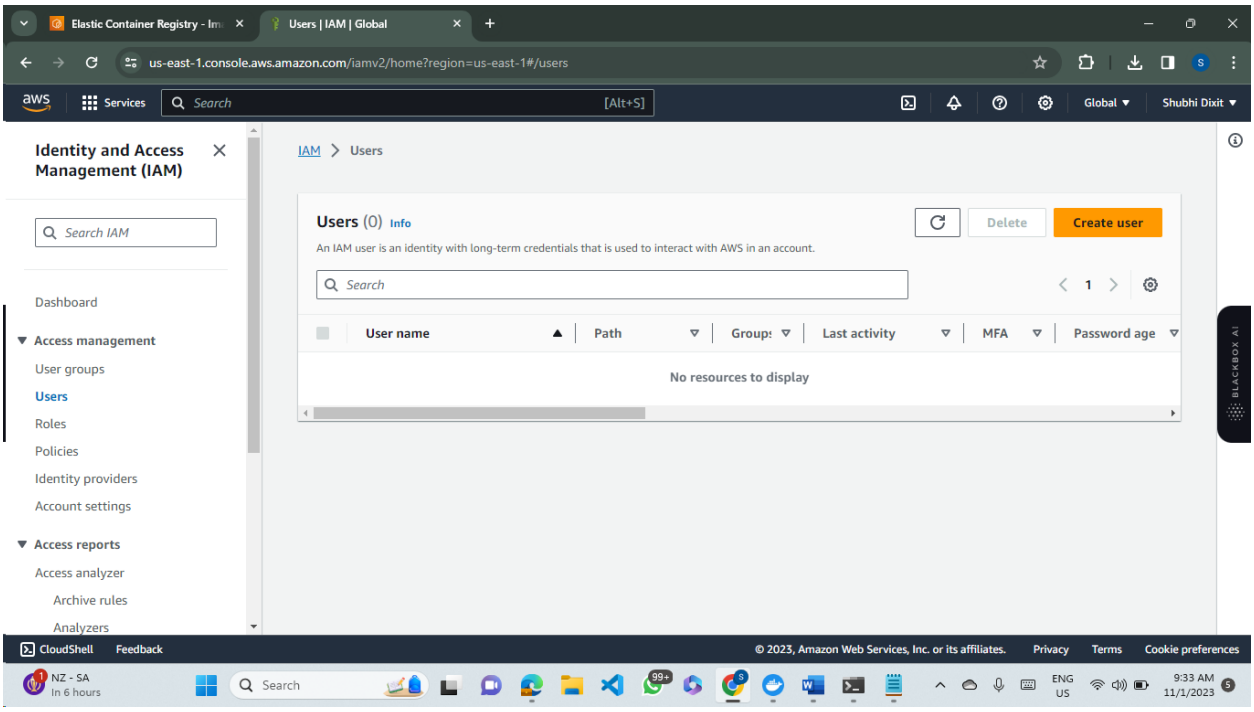
10) Before running these commands, you need to install AWS CLI and configure it if you haven't already: https://aws.amazon.com/cli/



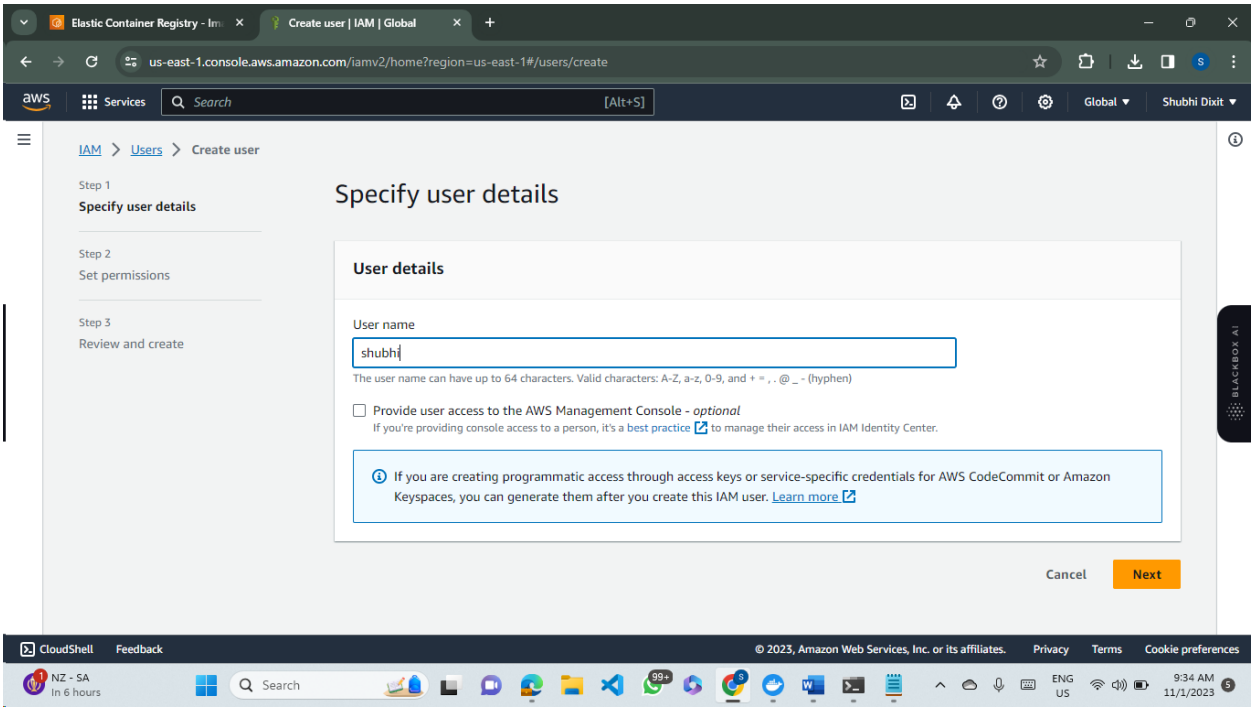




11) In order to configure AWS CLI Go to AWS console -> IAM -> Users -> Add users



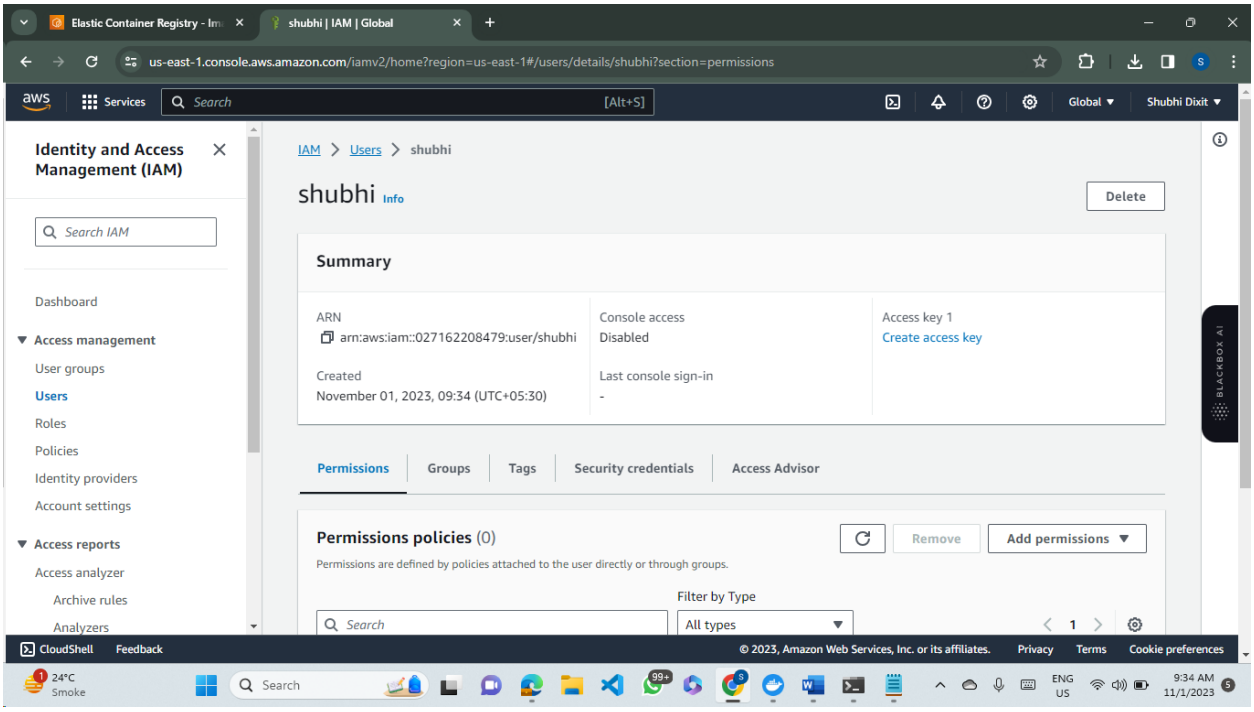
12) Specify the user details, add a user name and proceed with the following check boxes given in the image



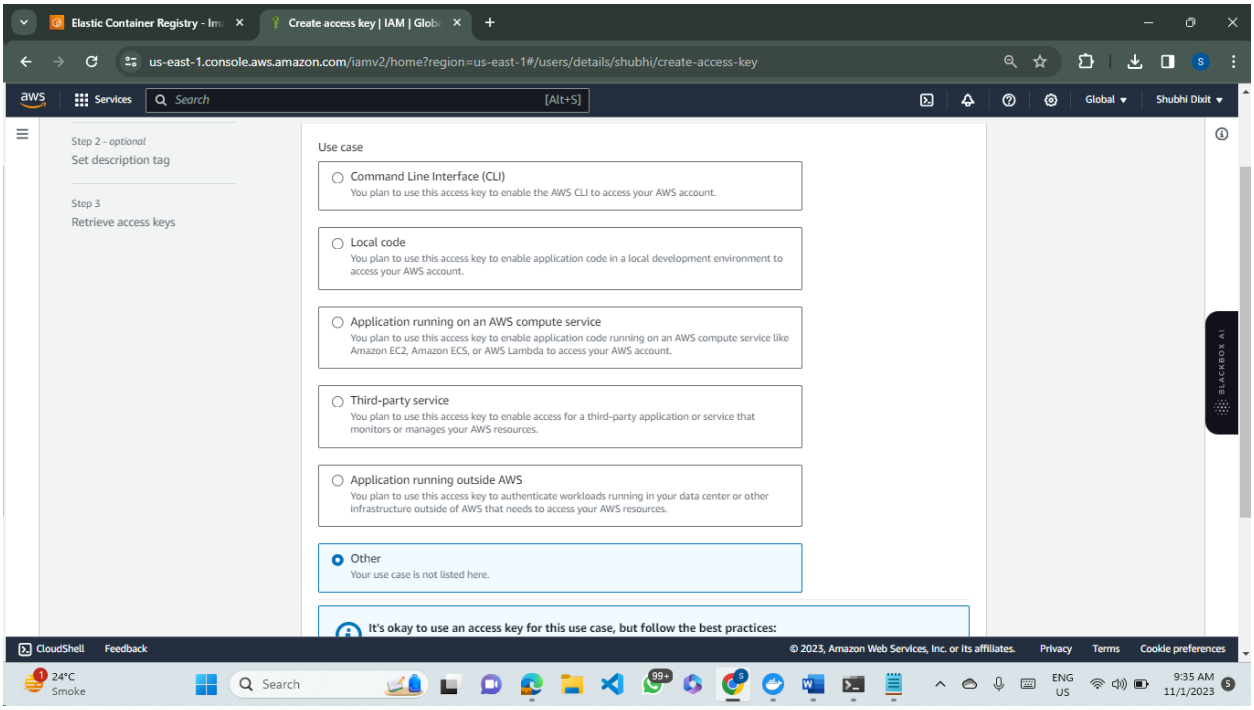
13)Click on next- next and “Create user”.

14) Select the user.

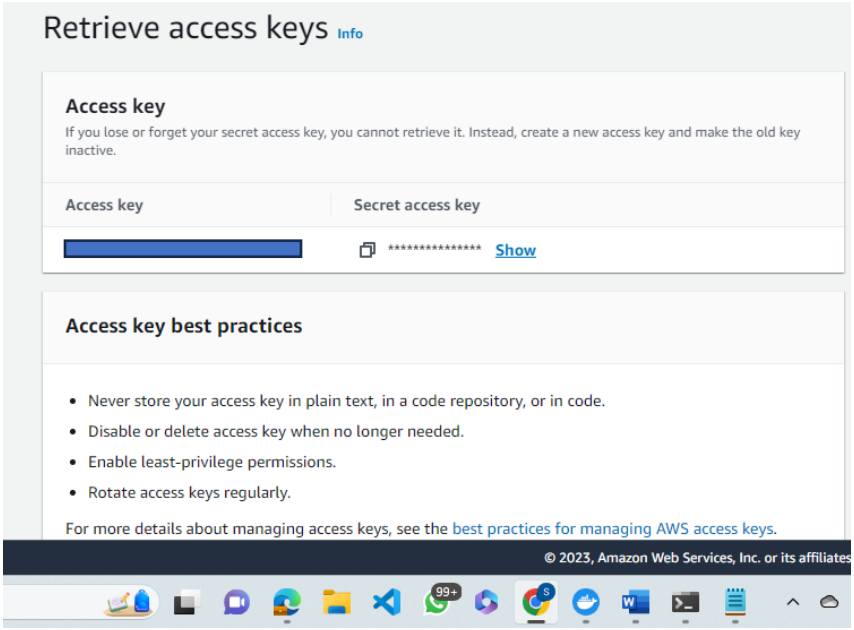
15)Click on “Create access key”

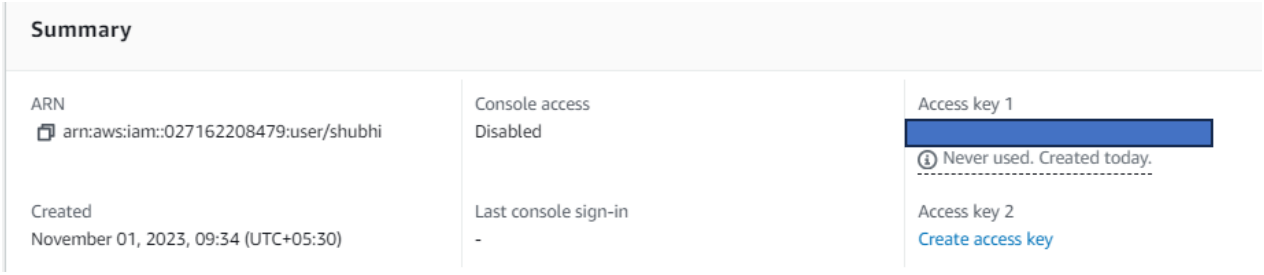


16) Select “Other” and proceed



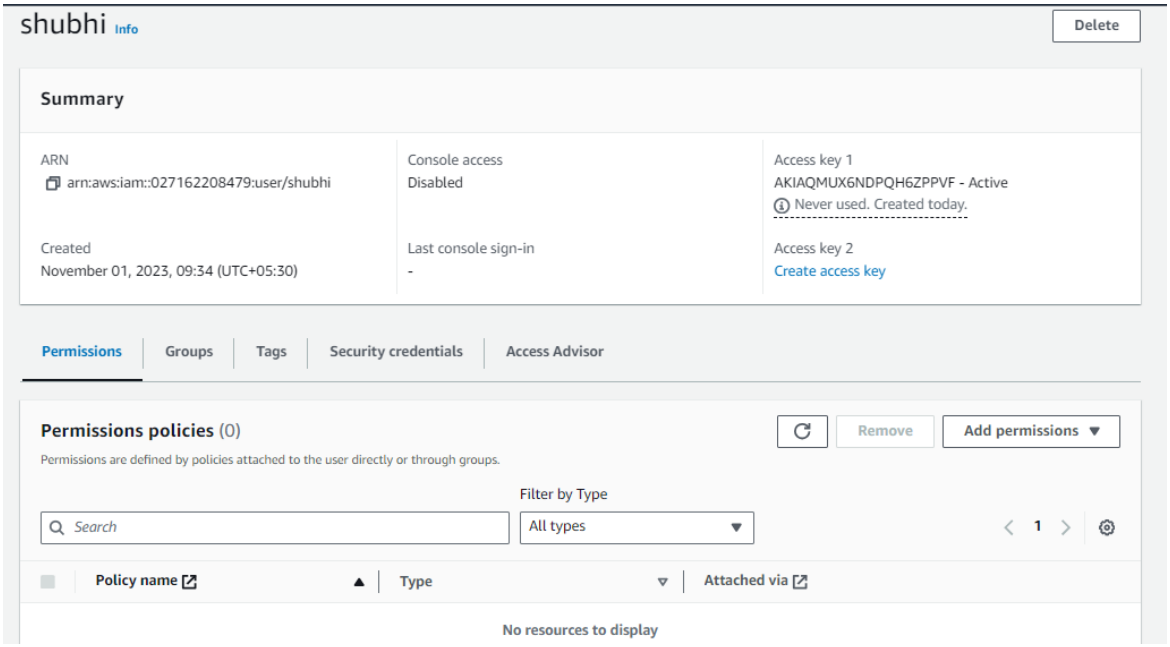
17) Note these Access key and Secret access key.





Access key added

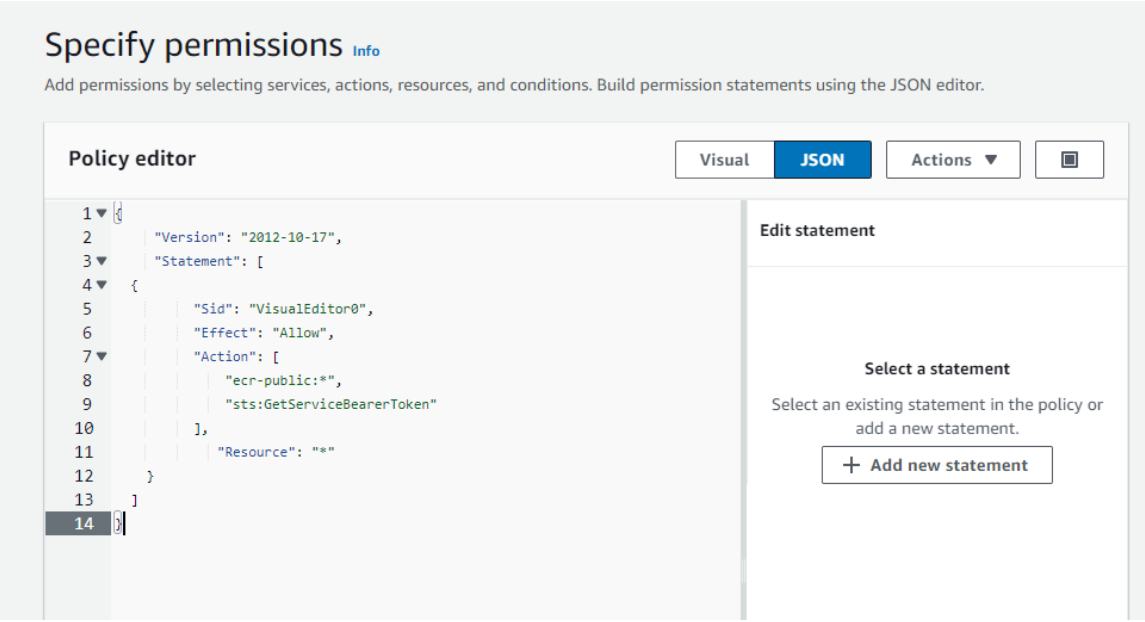
18) Now add permissions to the user. Go to “Add permissions”. Click on “Create inline policy



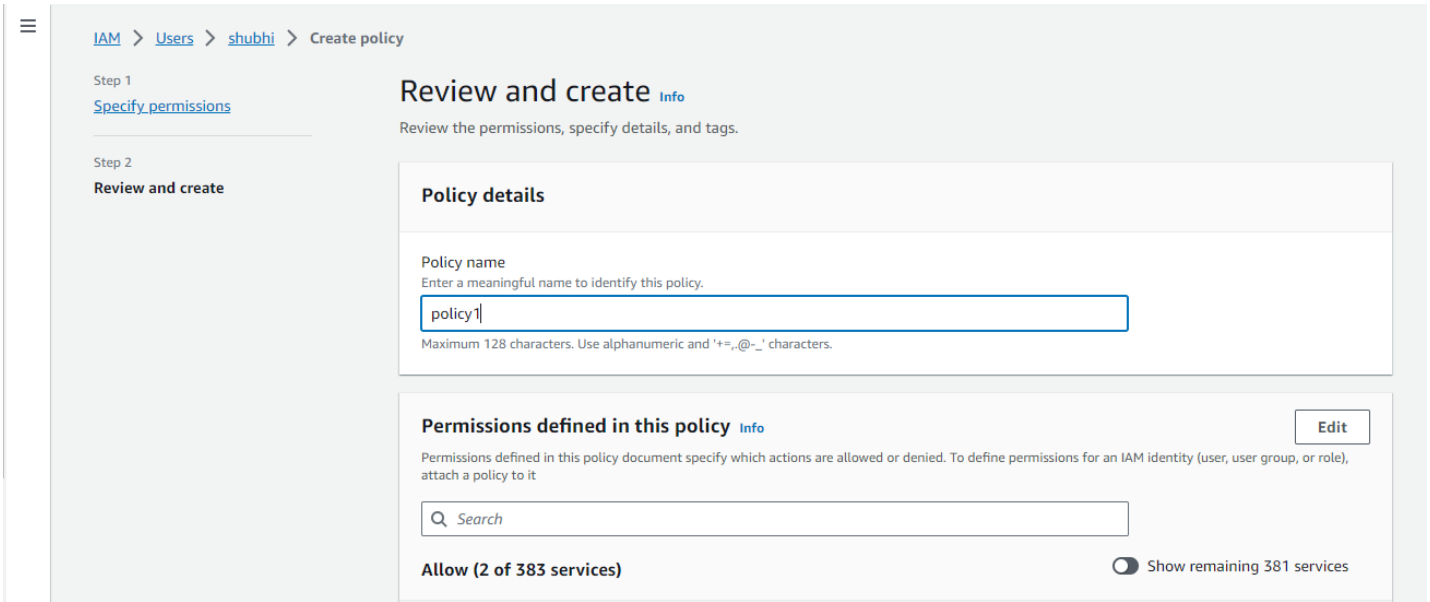
19) Add the following json in the policy and save.

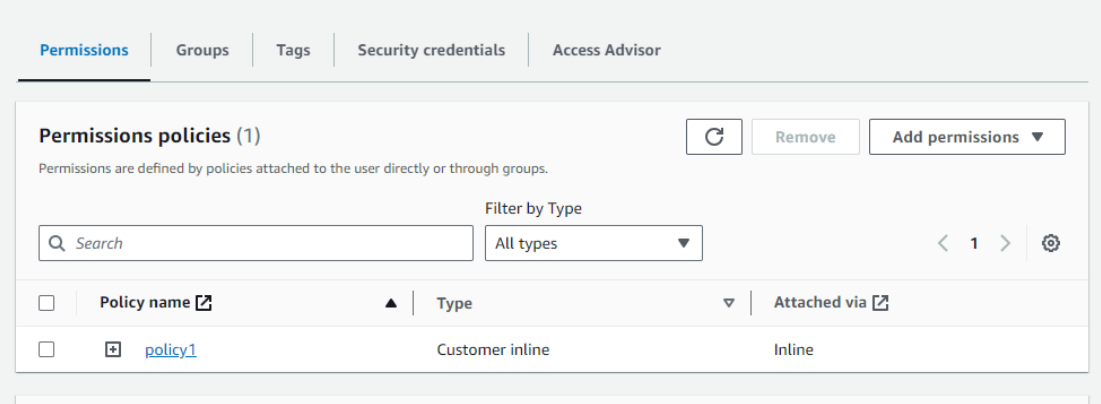
```
{
        "Version": "2012-10-17",
        "Statement": [
    {
          "Sid": "VisualEditor0",
           "Effect": "Allow",
            "Action": [
```

```
                "ecr-public:*",
                "sts:GetServiceBearerToken"
                ],
            "Resource": "*"
        }
    ]
}
```



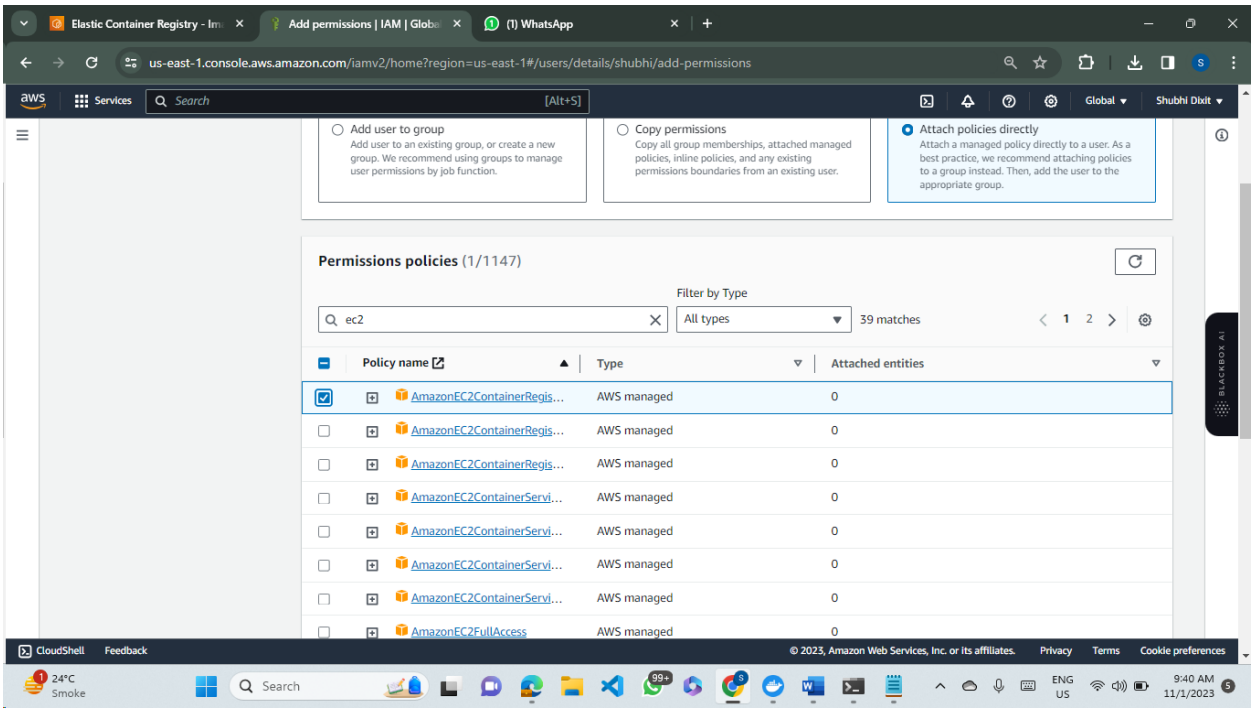
20) Name the Policy and click on create.



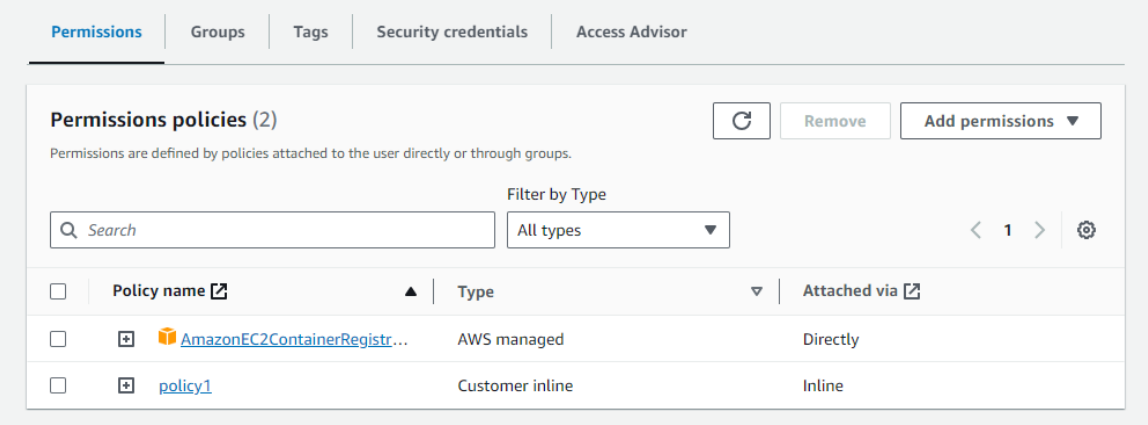


Policy created successfully.

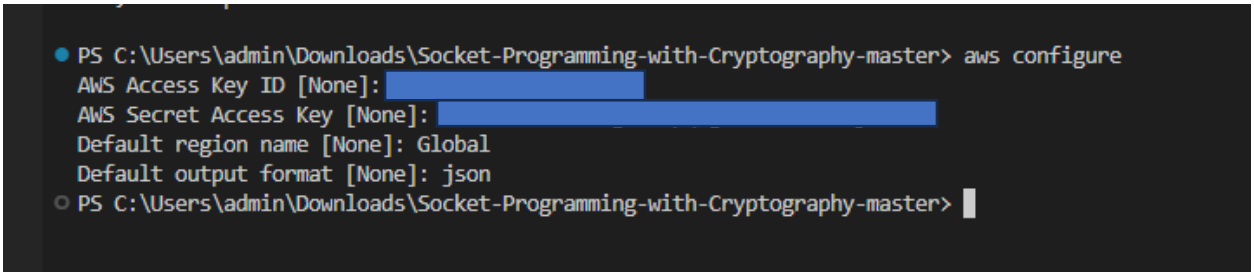
21) Now attach other permissions. Click on “Attach Policy directly”.



22) Make sure the permissions are attached.

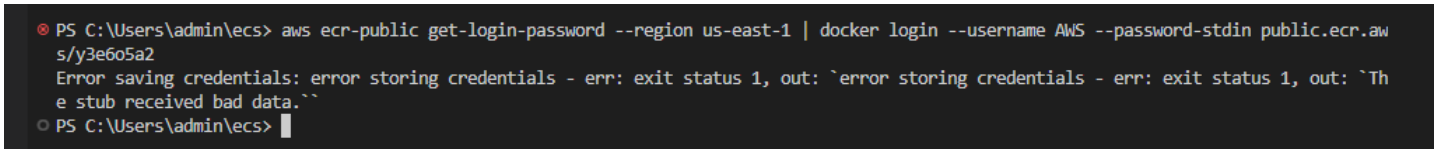


23) Now configure AWS in your IDE. I am using VS Code.



24) After successfully configuring, run these command one by one in the terminal in your project directory.

25) While running the first command from the commands given, if it gives error like: Error saving credentials: error storing credentials - err: exit status 1, out: `The stub received bad data.'



<u>Follow the following steps:</u>

<u>Go to:</u> c:\Users\admin\.docker\config.json

And remove the "credsStore": "wincred" or "credsStore": "desktop" key from the json file

26) Run the command again and it will show Login succeeded. If it does not work try to repeat the steps.

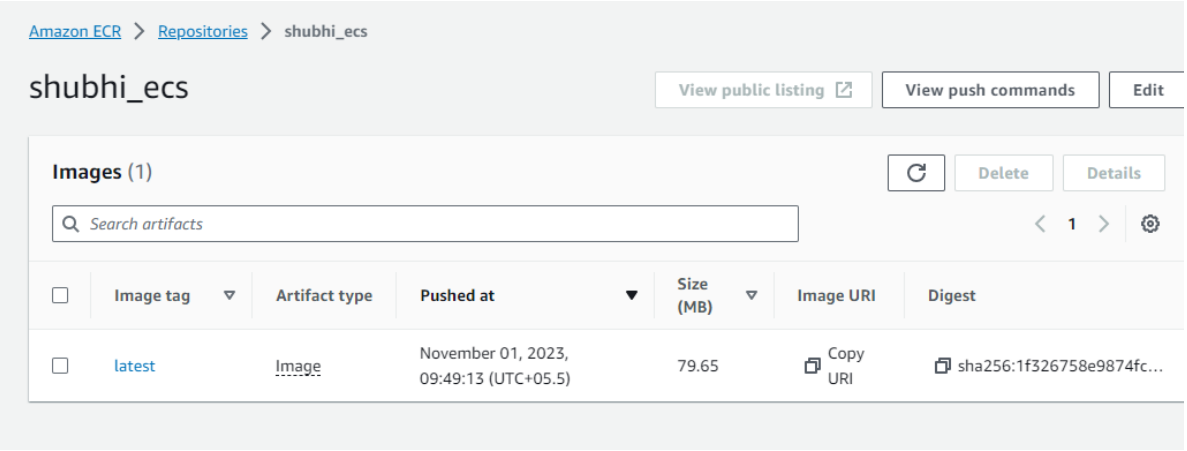
```
PS C:\Users\admin\ecs> aws ecr-public get-login-password --region us-east-1 | docker login --username AWS --password-stdin public.ecr.aw
s/y3e6o5a2
WARNING! Your password will be stored unencrypted in C:\Users\admin\.docker\config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
PS C:\Users\admin\ecs> docker build -t shubhi_ecs .
[+] Building 3.9s (8/8) FINISHED                                                                                    docker:default
```

27) Then run the rest of the commands one by one

```
PS C:\Users\admin\ecs> docker build -t shubhi_ecs .
[+] Building 3.9s (8/8) FINISHED
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 193B
 => [internal] load metadata for docker.io/library/httpd:2.4
 => [1/3] FROM docker.io/library/httpd:2.4@sha256:7b8098fbbdaa8d12c3ae2208b249e7adbe12376fe8d86ef2aae27ce1f13b7167
 => [internal] load build context
 => => transferring context: 182B
 => CACHED [2/3] RUN apt-get update
 => [3/3] COPY ./index.html /usr/local/apache2/htdocs
 => exporting to image
 => => exporting layers
 => => writing image sha256:8e7f3958526058e5b7fe7842d262252cf17a90639bcdbe525458b8f6d5c484b9
```

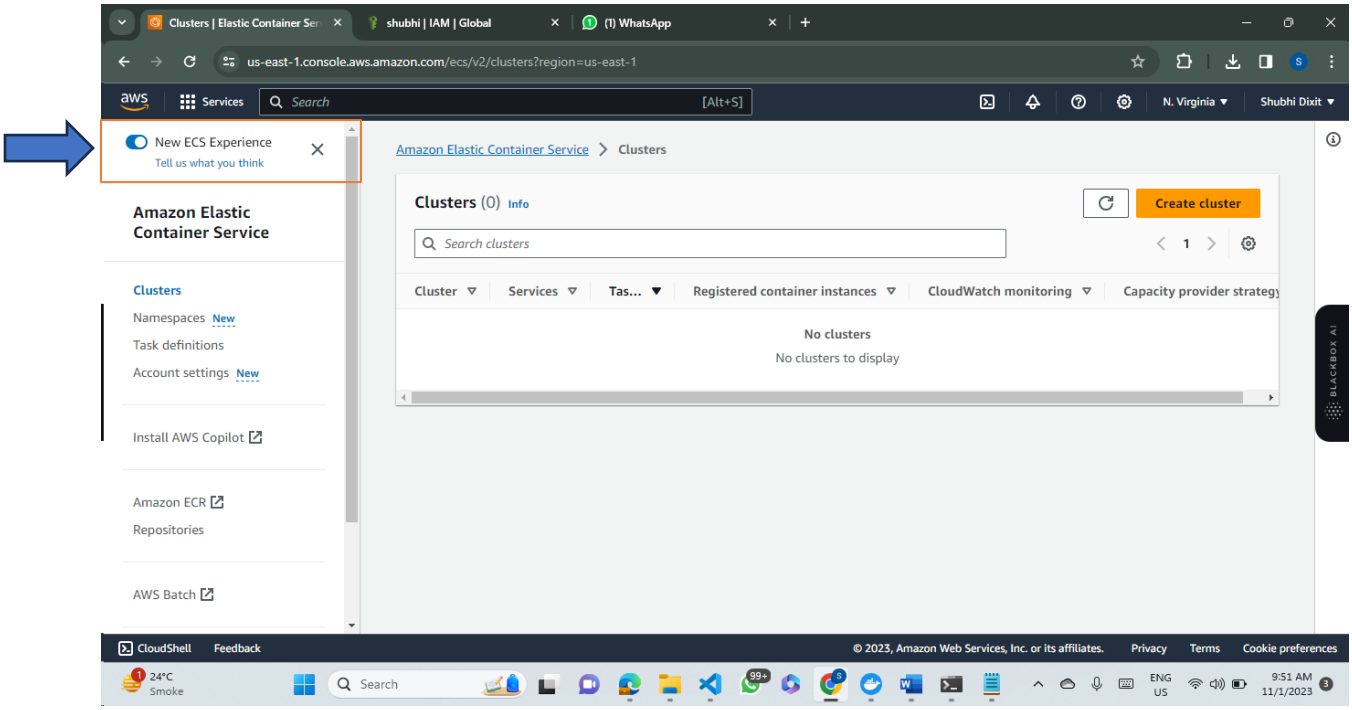
```
PS C:\Users\admin\ecs> docker tag shubhi_ecs:latest public.ecr.aws/y3e6o5a2/shubhi_ecs:latest
PS C:\Users\admin\ecs> docker push public.ecr.aws/y3e6o5a2/shubhi_ecs:latest
The push refers to repository [public.ecr.aws/y3e6o5a2/shubhi_ecs]
7c8bd21646f4: Pushed
6b7ad43c9a9e: Pushed
cdae29f197b2: Pushed
5ca84525a215: Pushed
e0ba343b5cab: Pushed
2905795cb5b8: Pushed
ec983b166360: Pushed
latest: digest: sha256:1f326758e9874fcffae6061777d6ee52a807972fe7fe41a64c2944c617c153d7 size: 1785
```

28) After running all the commands, check whether the image is pushed successfully or not:

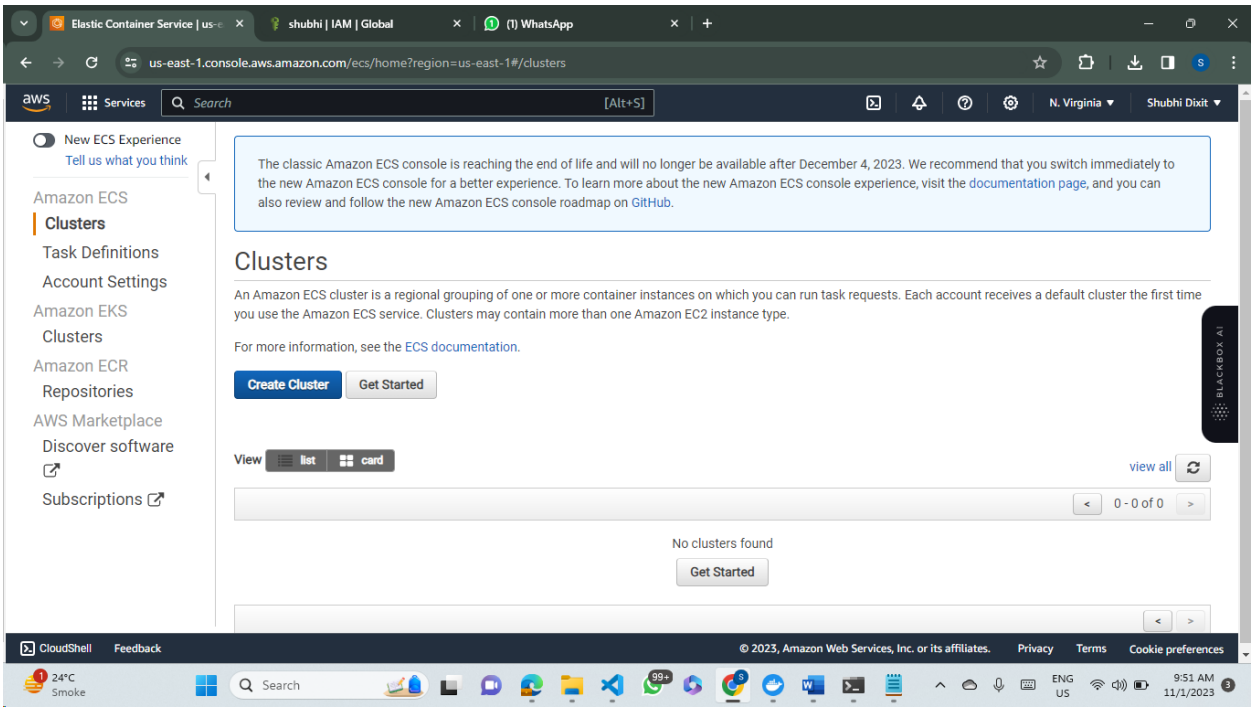


Creating Cluster

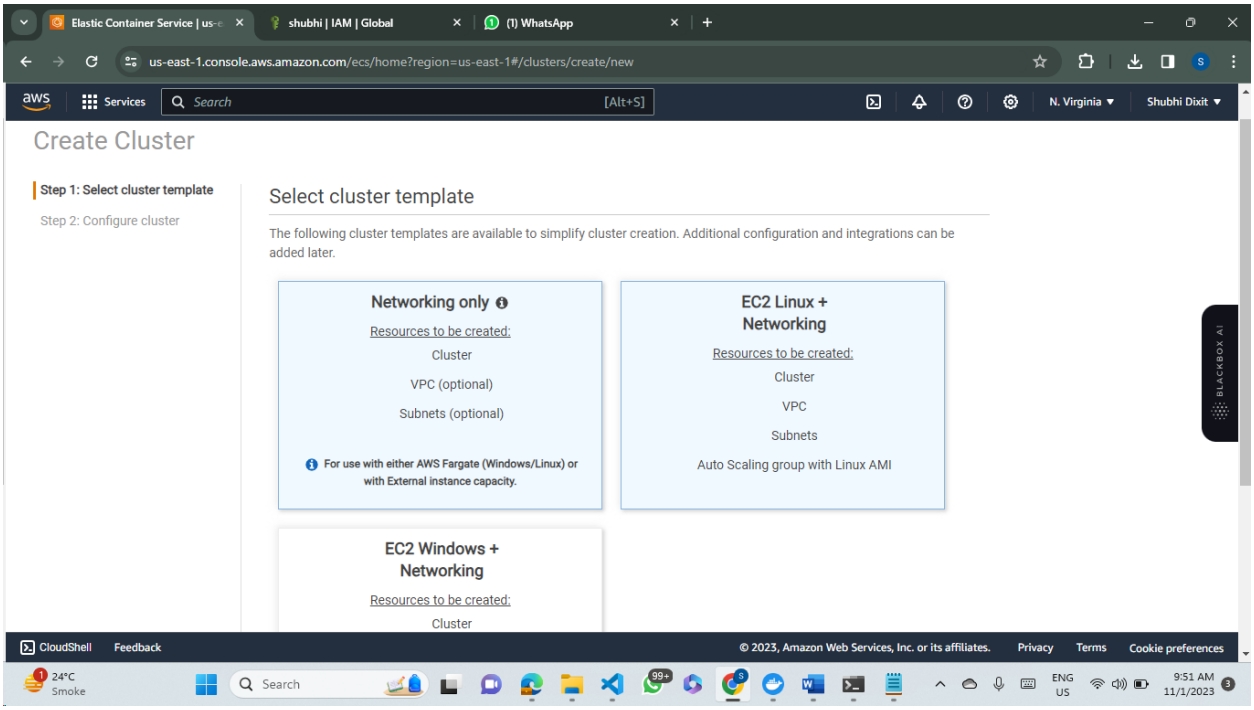
1) Switch to the “old experience for ECS” from the top left corner.



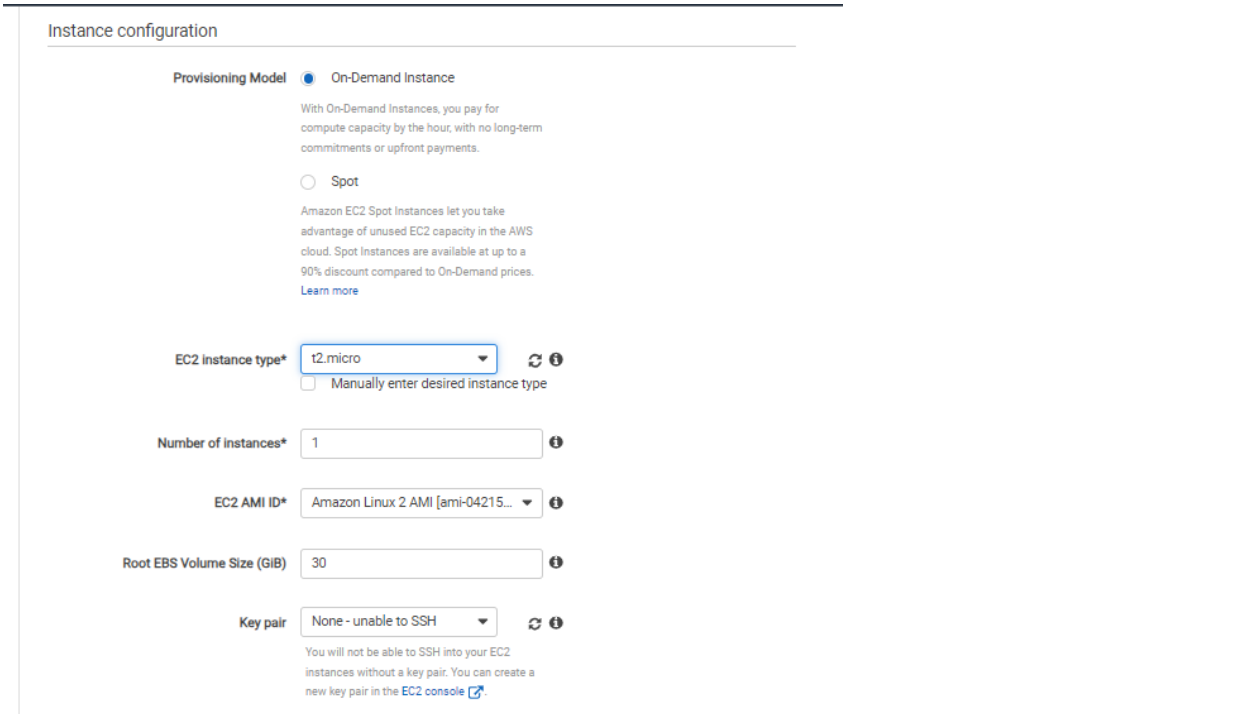
2) Next step is to create the cluster. Go to ECS and click on “Create cluster”



3) Choose the “EC2 Linux + Networking option”.



4) Set the settings in the respective fields:

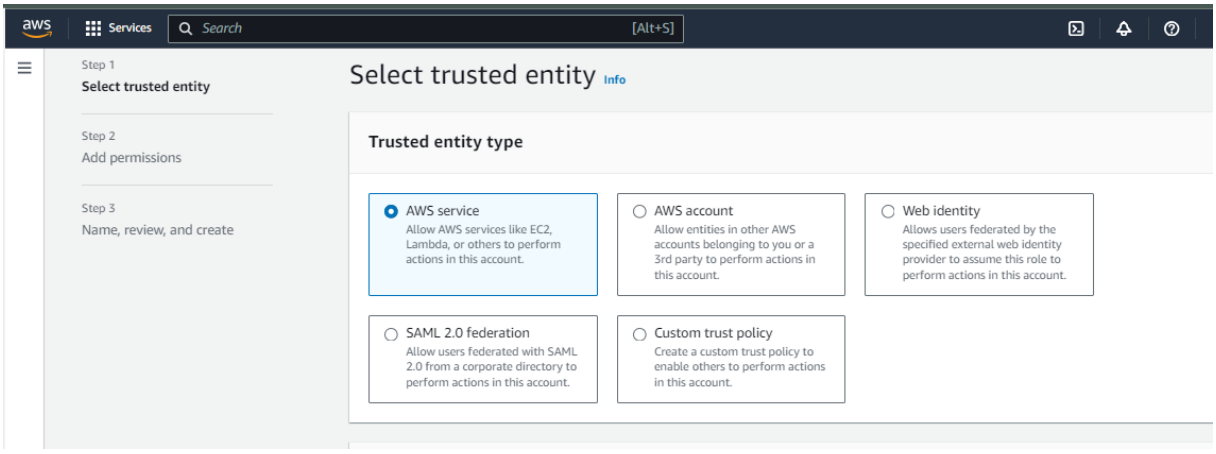


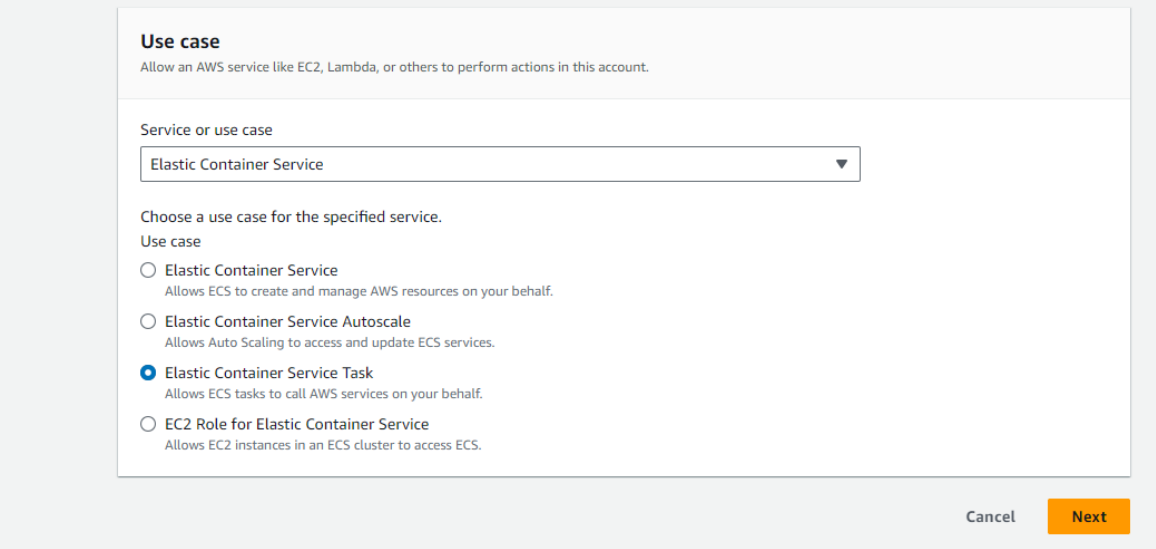
5) Select the default VPC and the subnets that you have. Leave the other settings default and click on create.

Click on “Create Cluster”

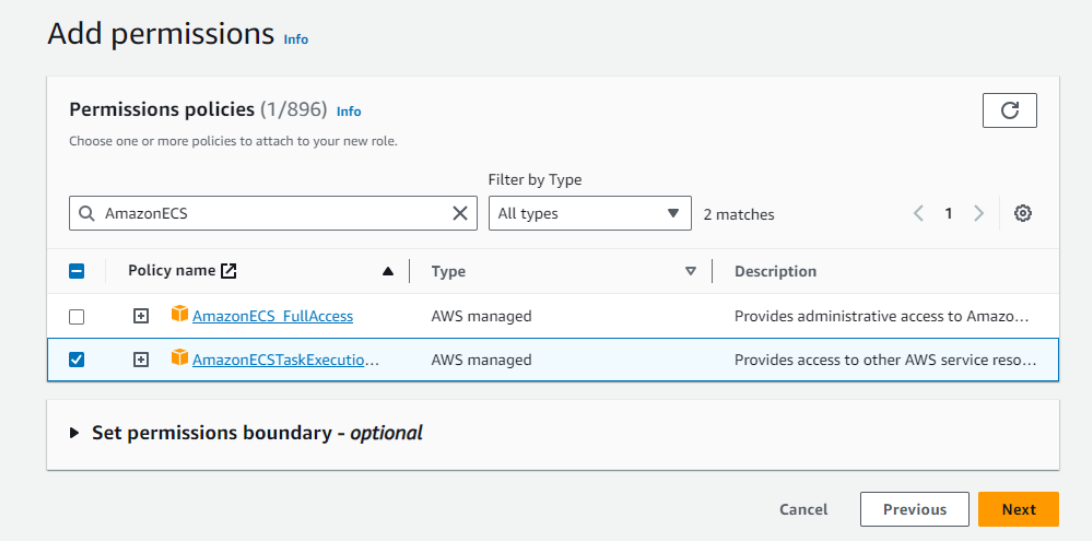
Role Creation

1) Go to IAM and create a new role. It will be used further while creating the task definitions
2) Choose the following options:

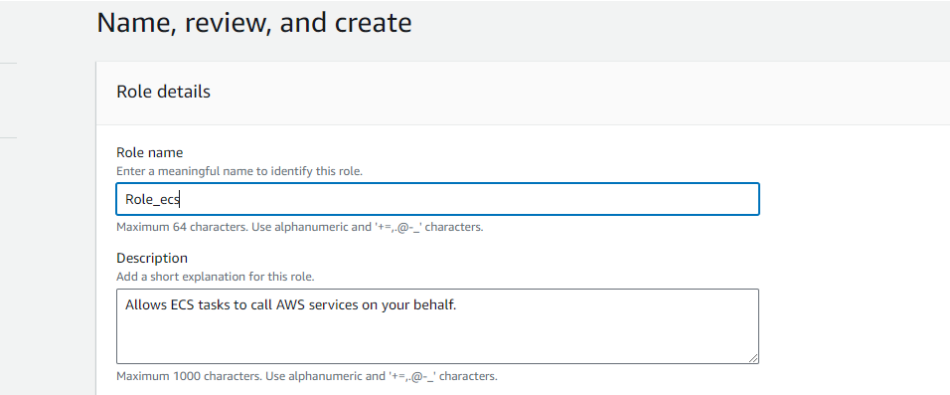




3) Add permissions and choose “AmazonECSTaskExecutionRolePolicy”



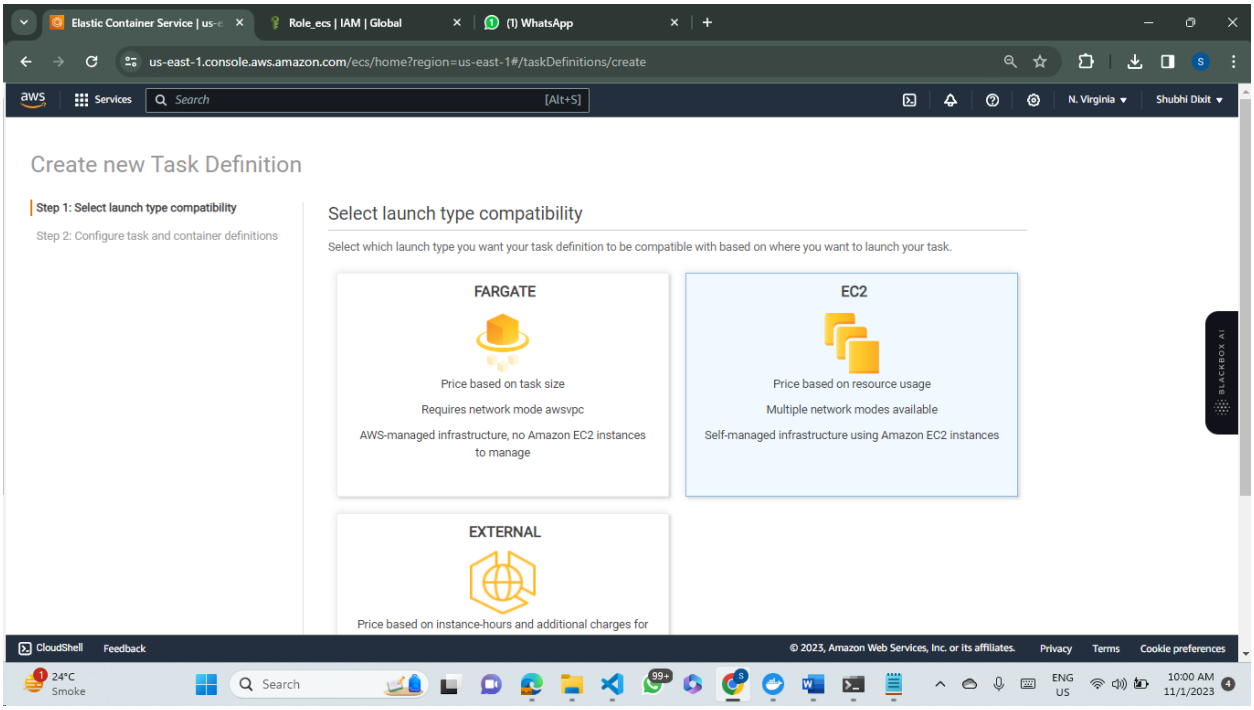
4) Click on next and fill the role name and proceed with “Create role”



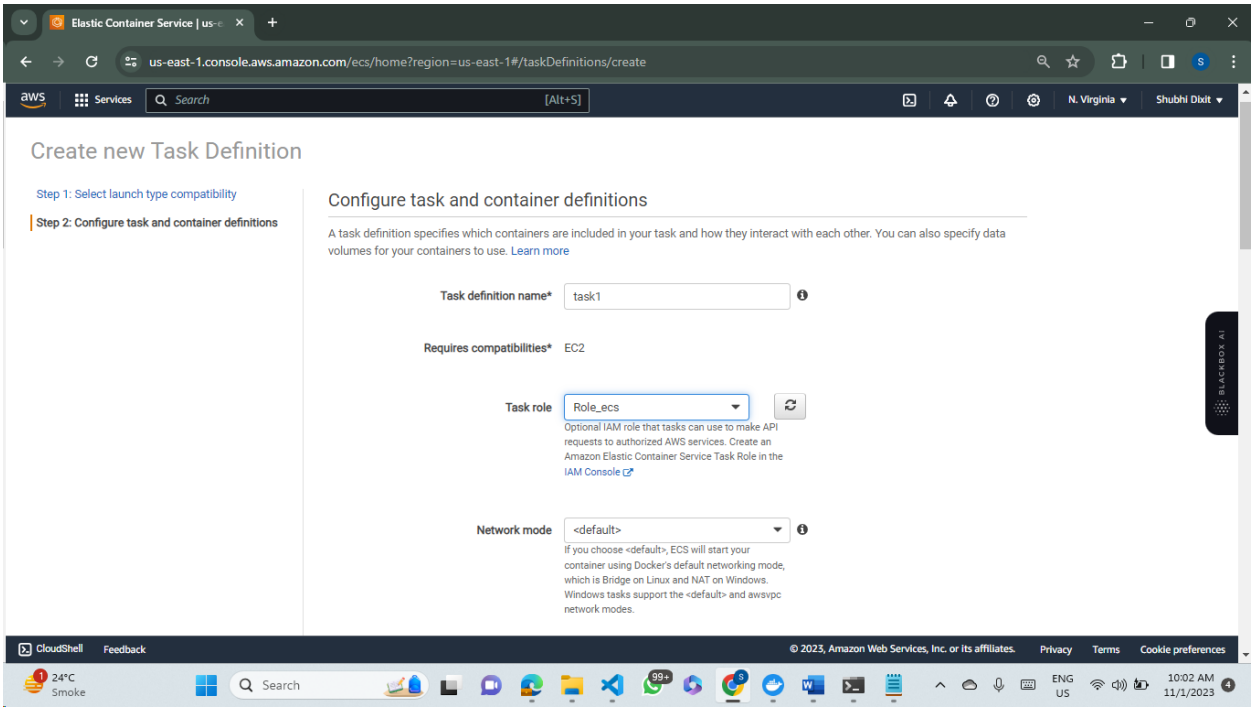
Task Definition

1) Go to ECS again, and go to “Task definitions”.

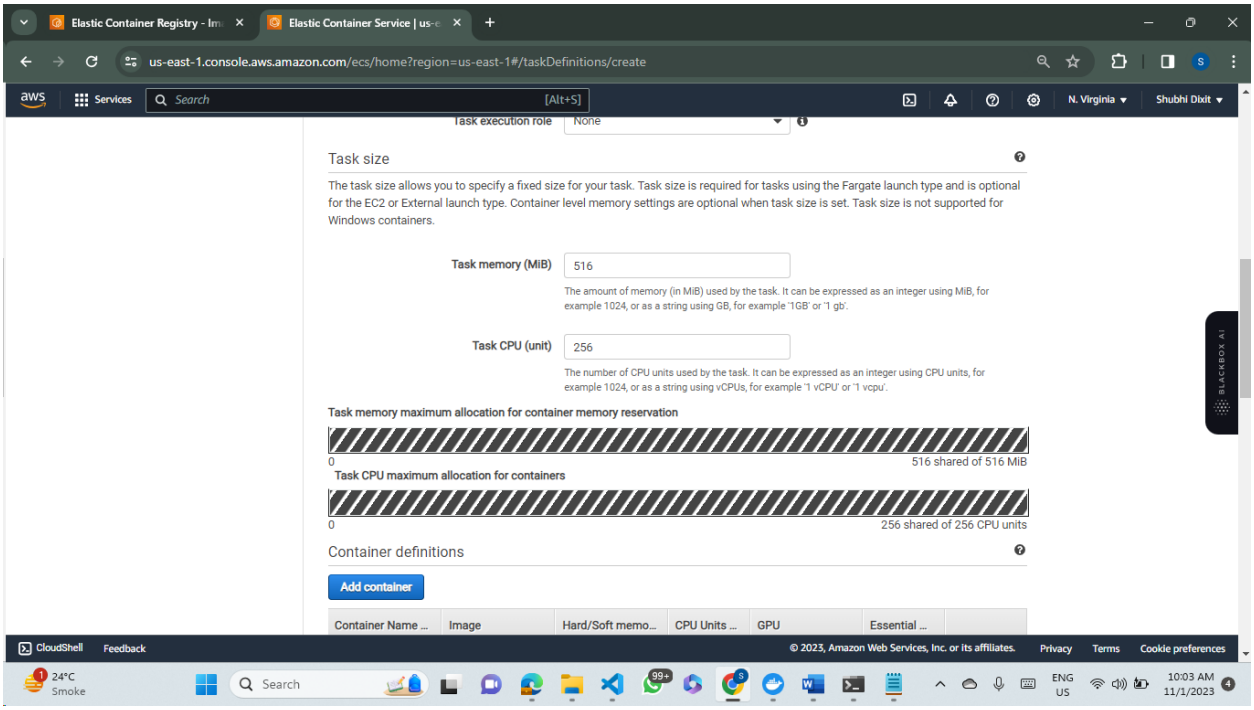
2) Click on “Create new task definition”.



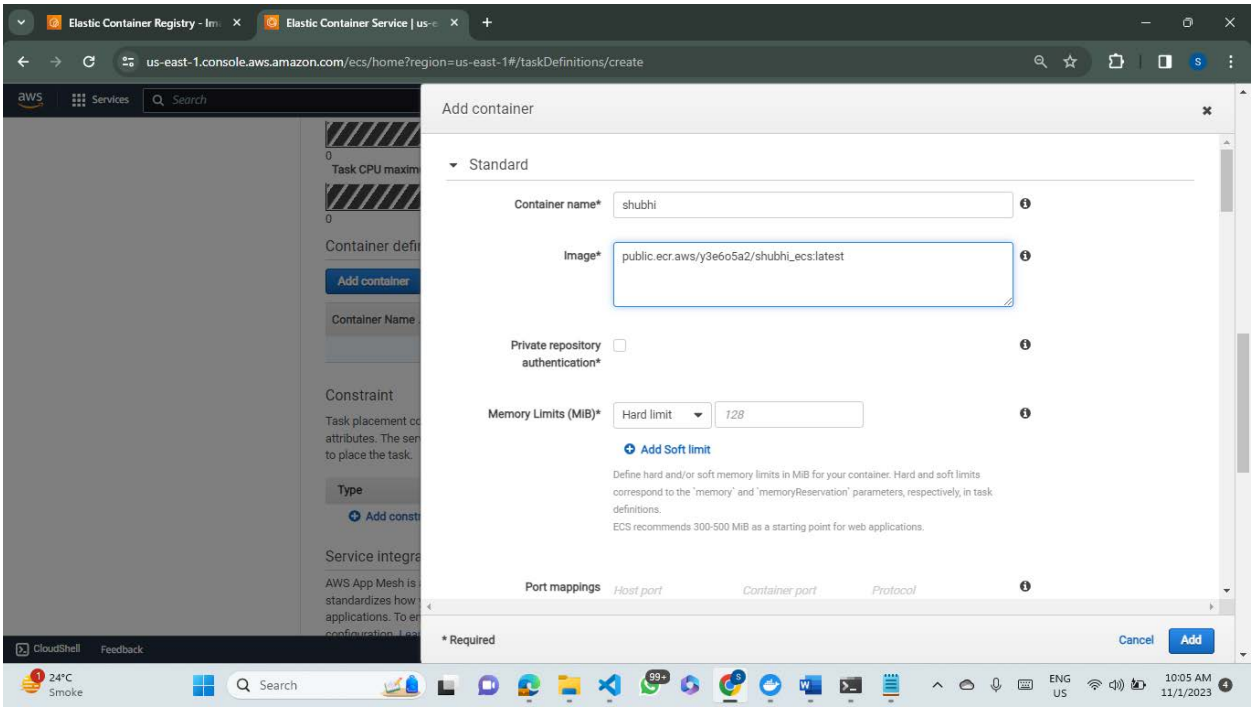
3) Now configure task and container definitions. Fill the task definition name, task role(attach the role you created earlier)

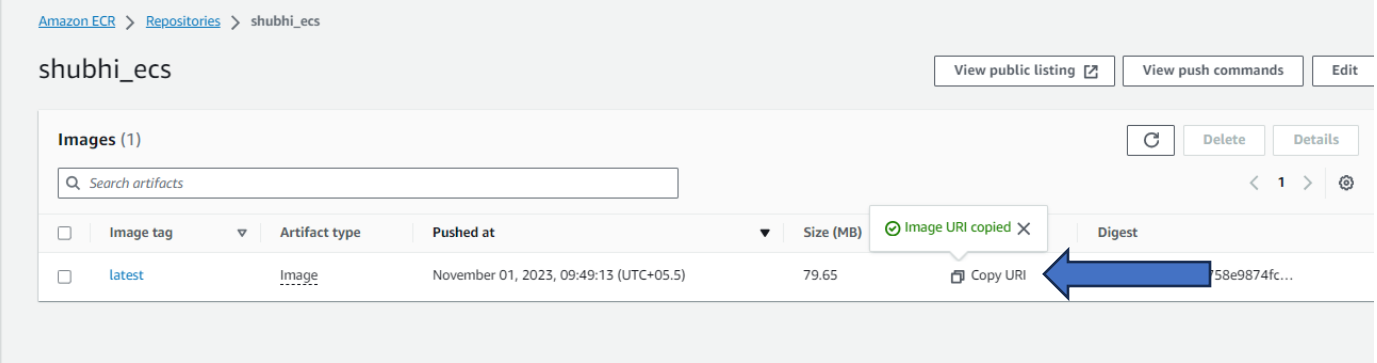


4) Fill the Task memory and Task CPU(unit).

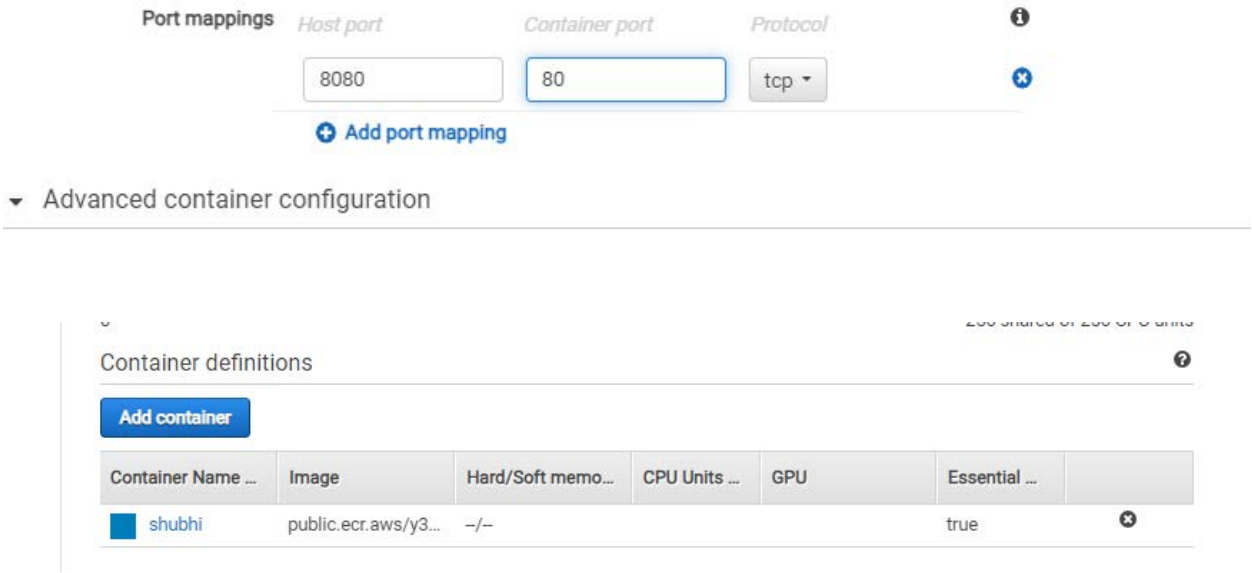


5) Add the container and copy the image URL(the image that you pushed to ECR) and all the necessary details. Assign the Host post as 8080 and Container port as 80.





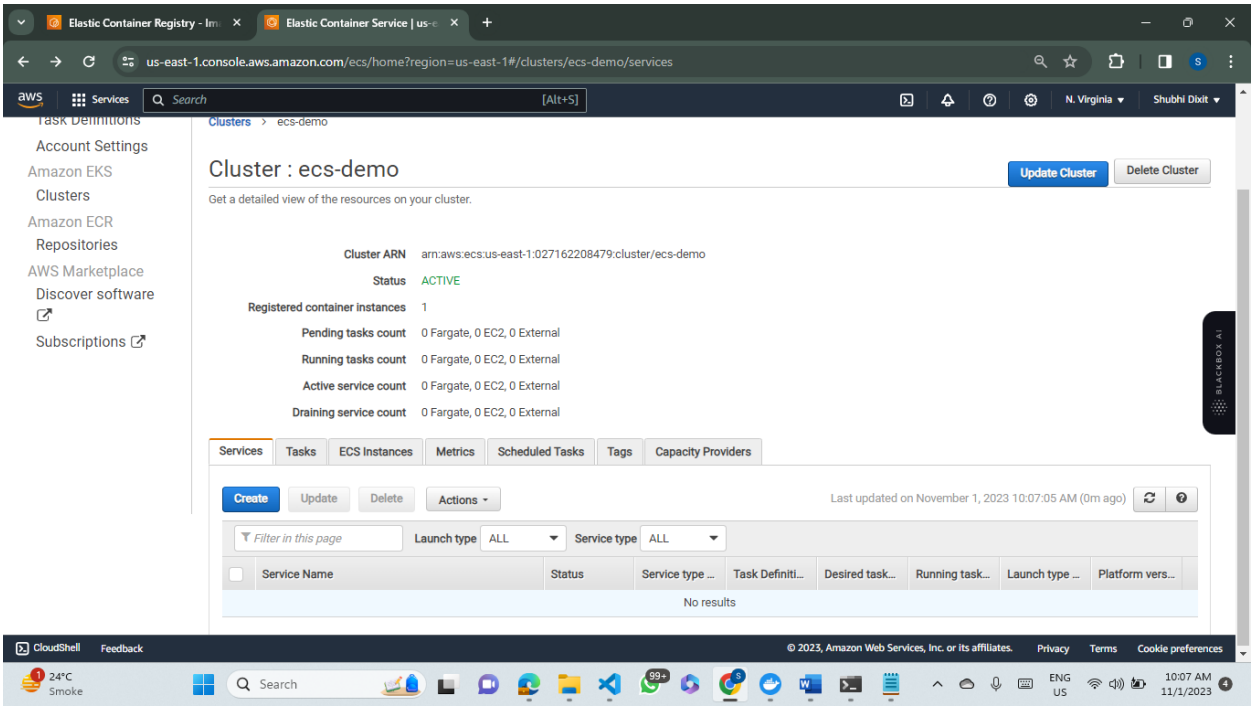
Copy Image URL from here.



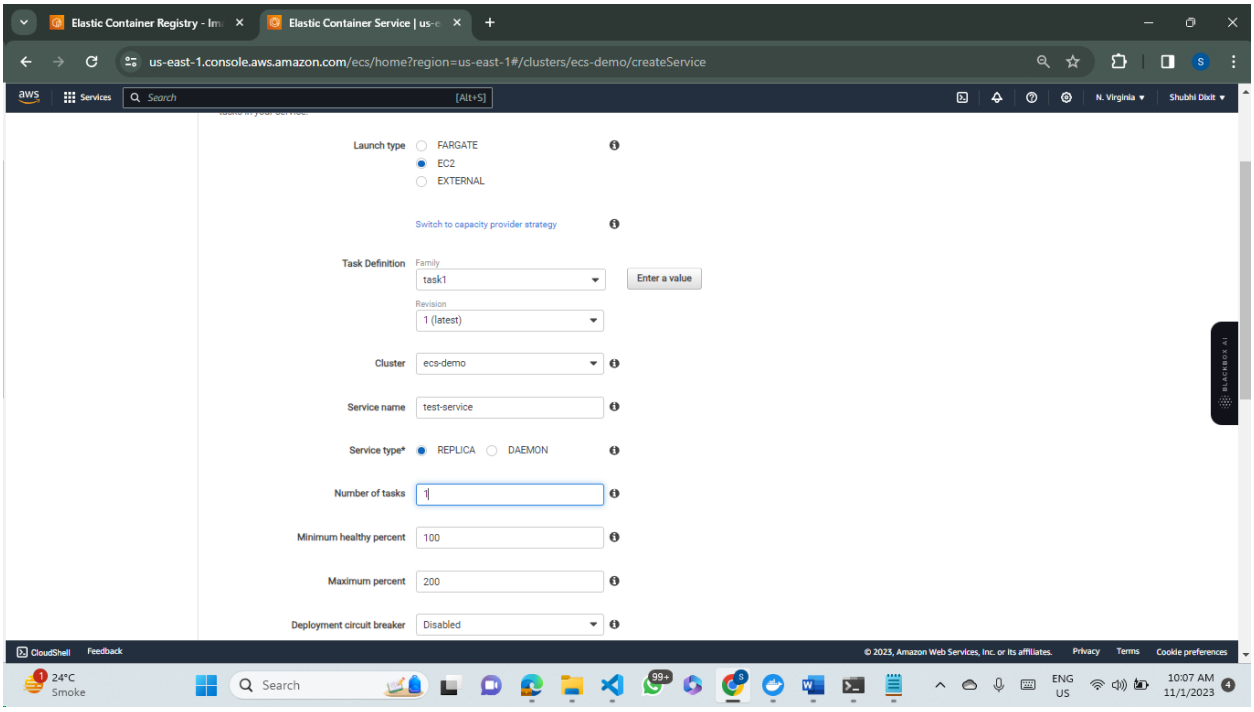
6) Task definition is created successfully.

Service Deployment

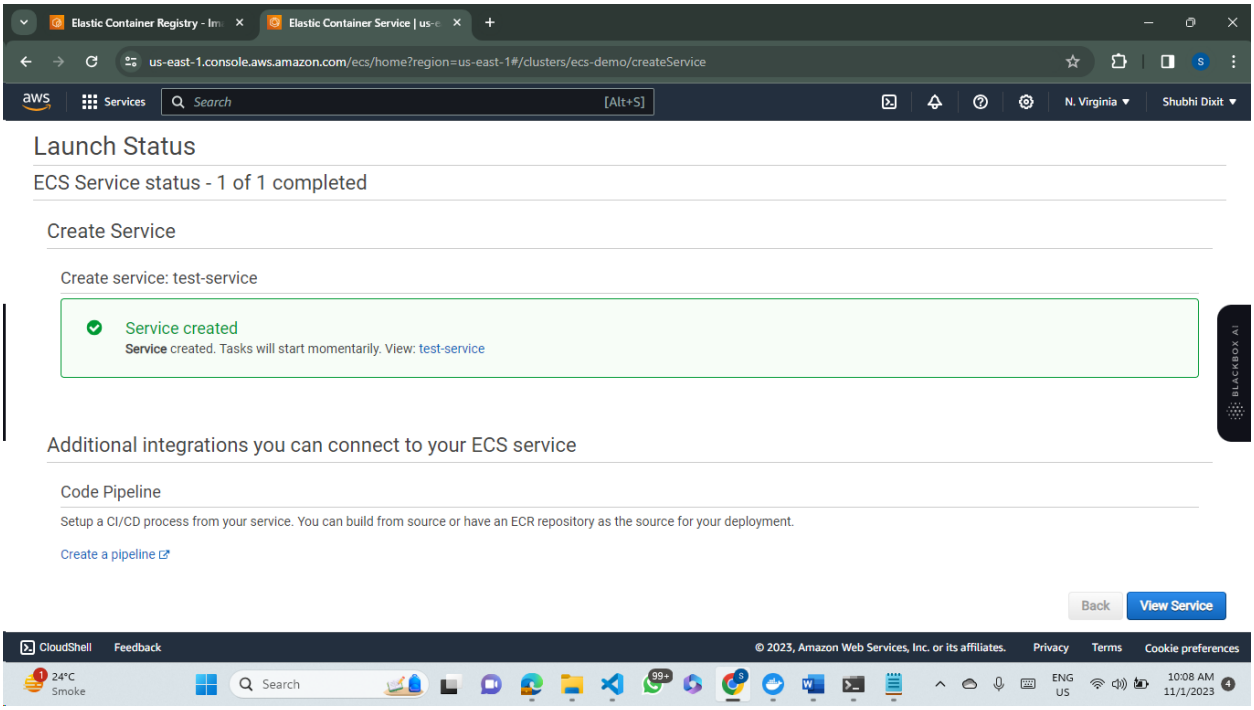
1) Now go to the cluster and choose the “Services” tab and click on “create”



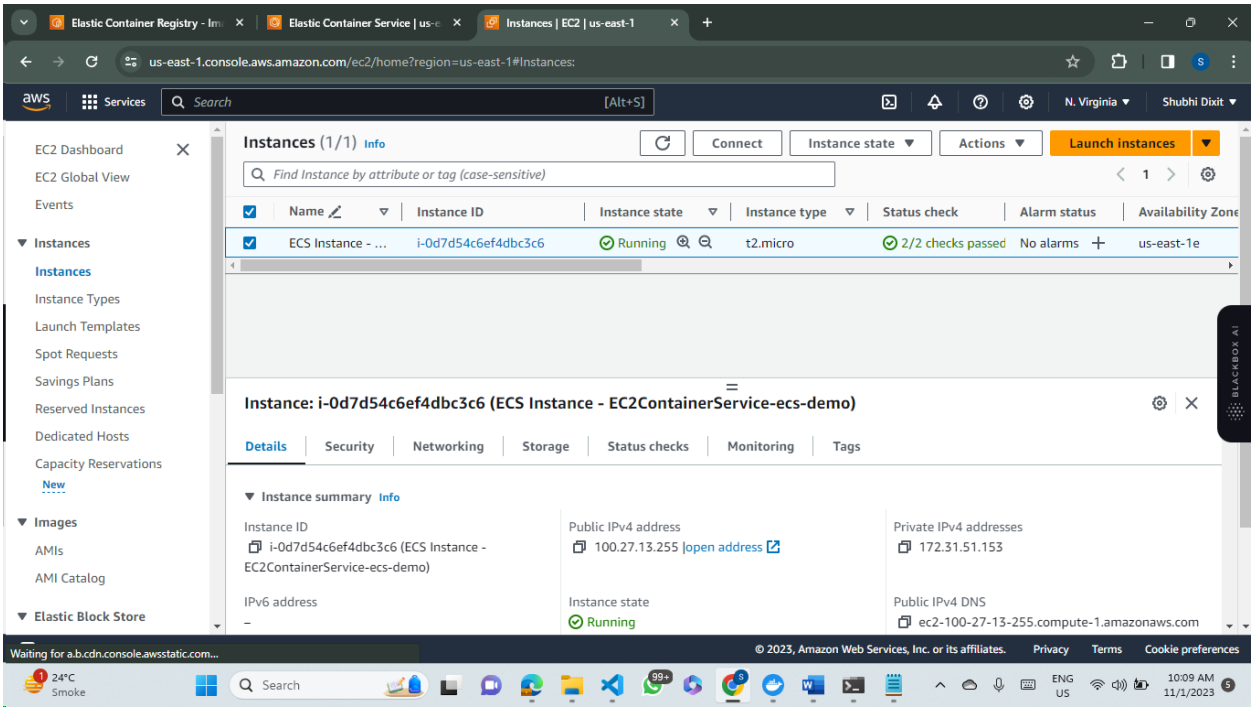
2) Choose the EC2 option and select the task definition that you just created.



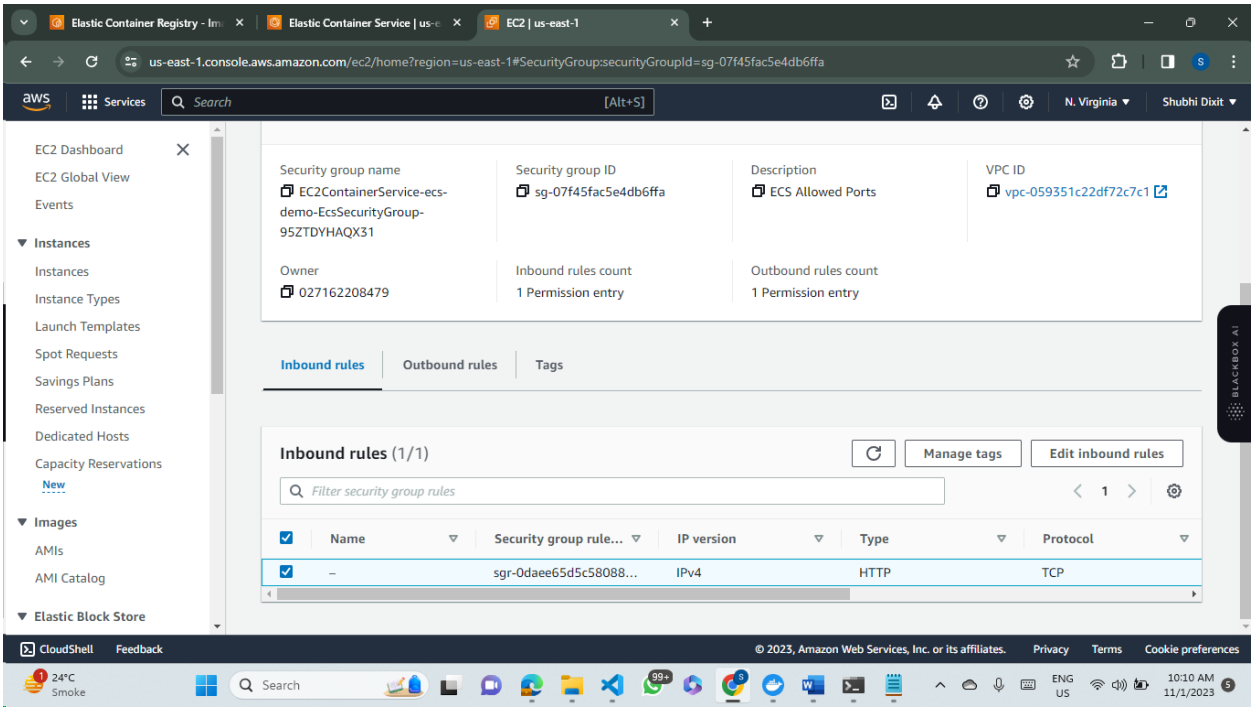
3) Service is successfully created.



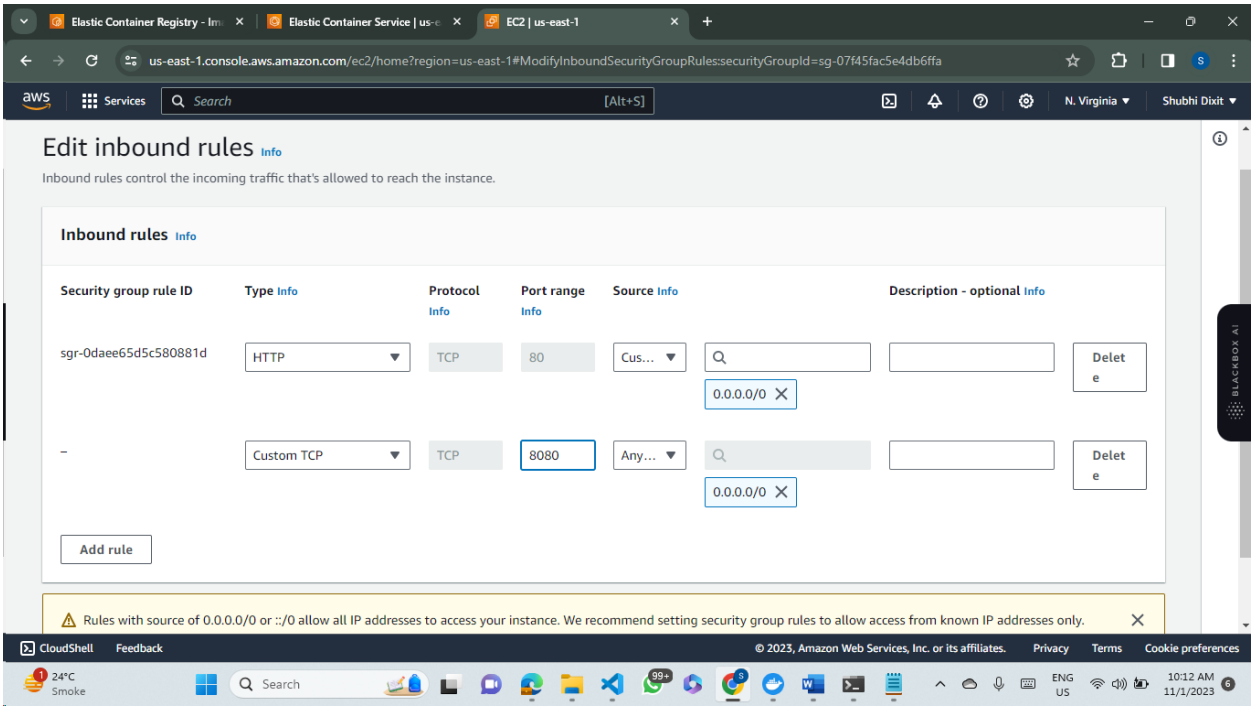
3) Now go to the EC2 instances and select the container instance.



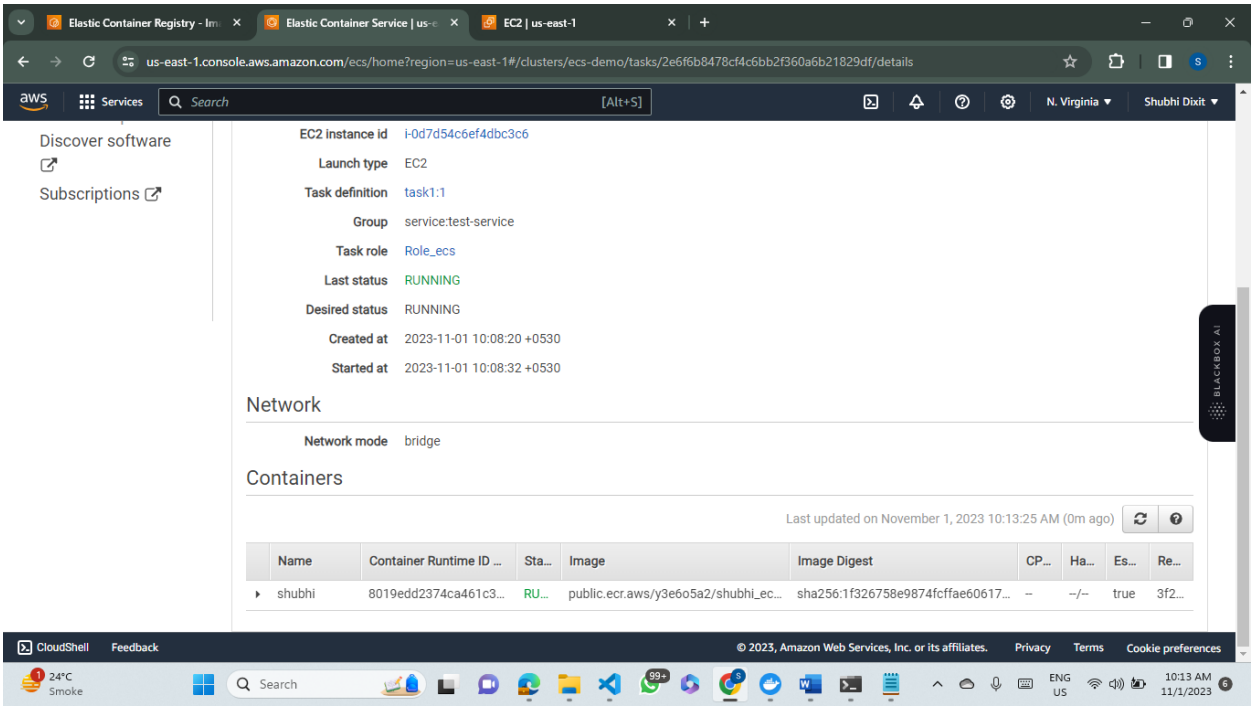
5) Select it and open the security groups. Click on “Edit inbound rule”



6) Add rule of type “Custom tcp”, port range “8080” and source “Anywhere”

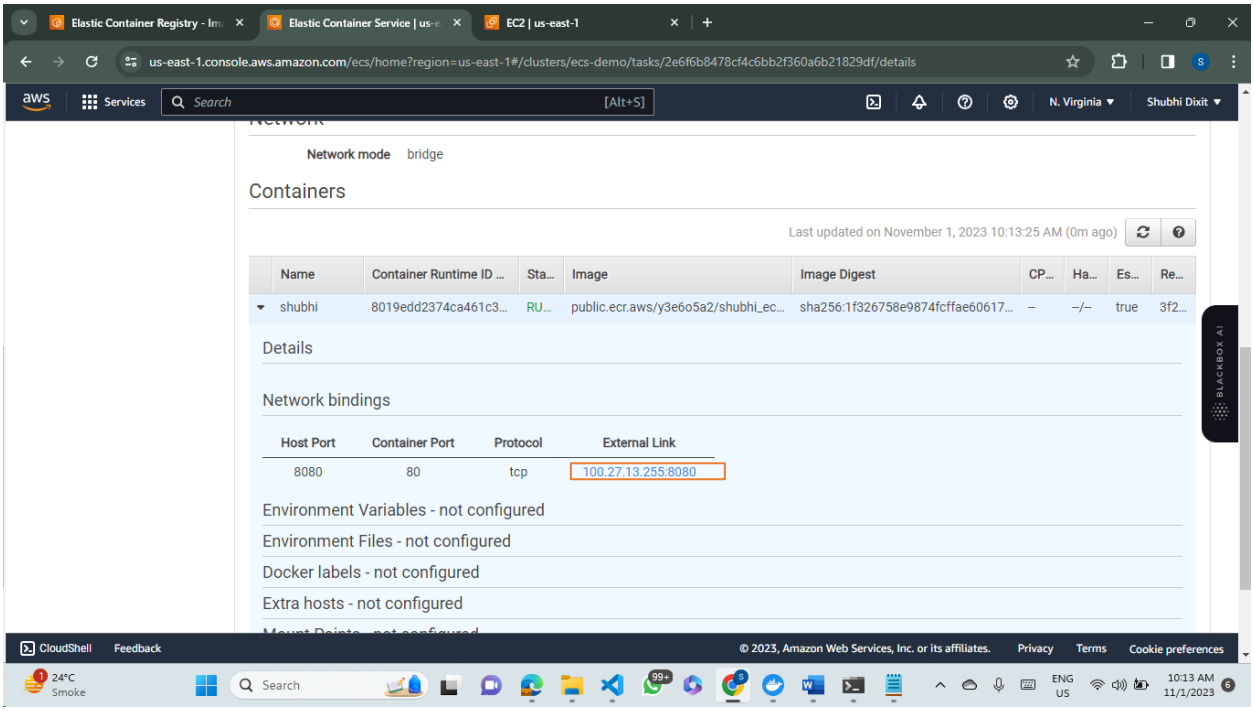


7) Now go to the ECS -> Cluster -> Tasks. Open the task that is shown

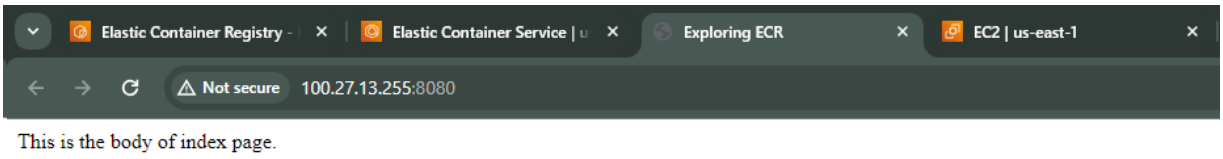


8) Open the container details shown below

9) Copy the external link and access the html page on browser.

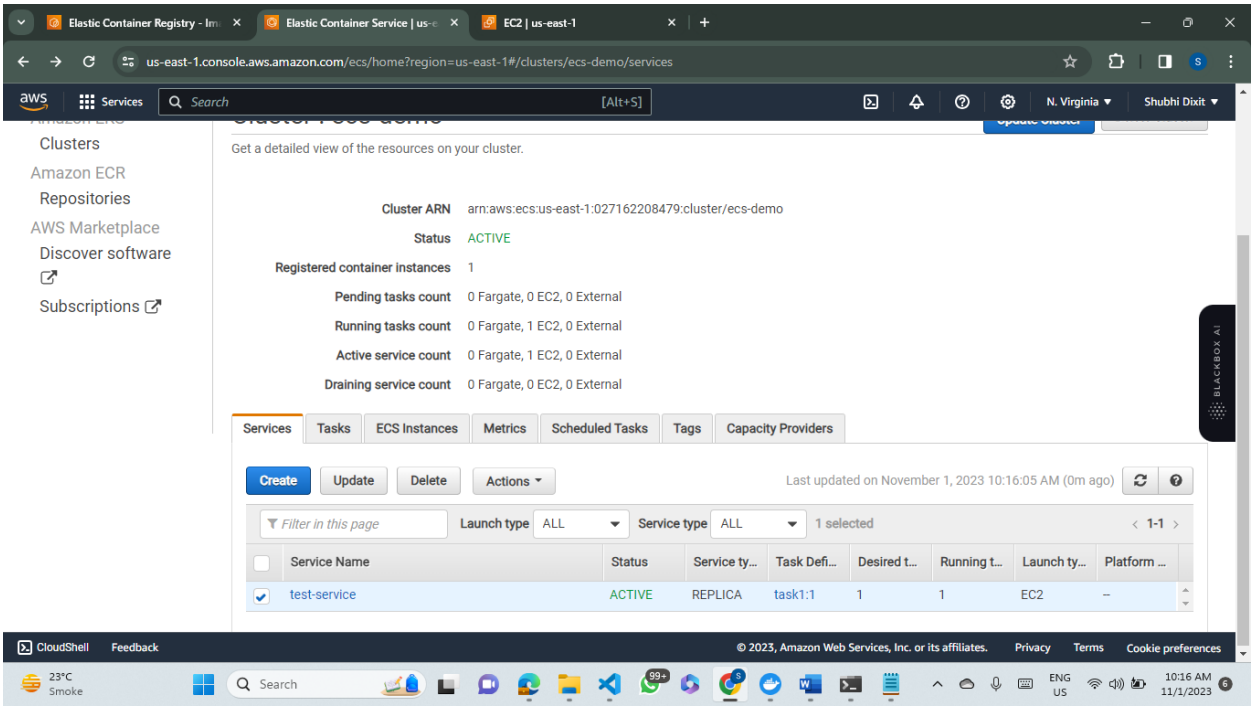


Paste in local browser.

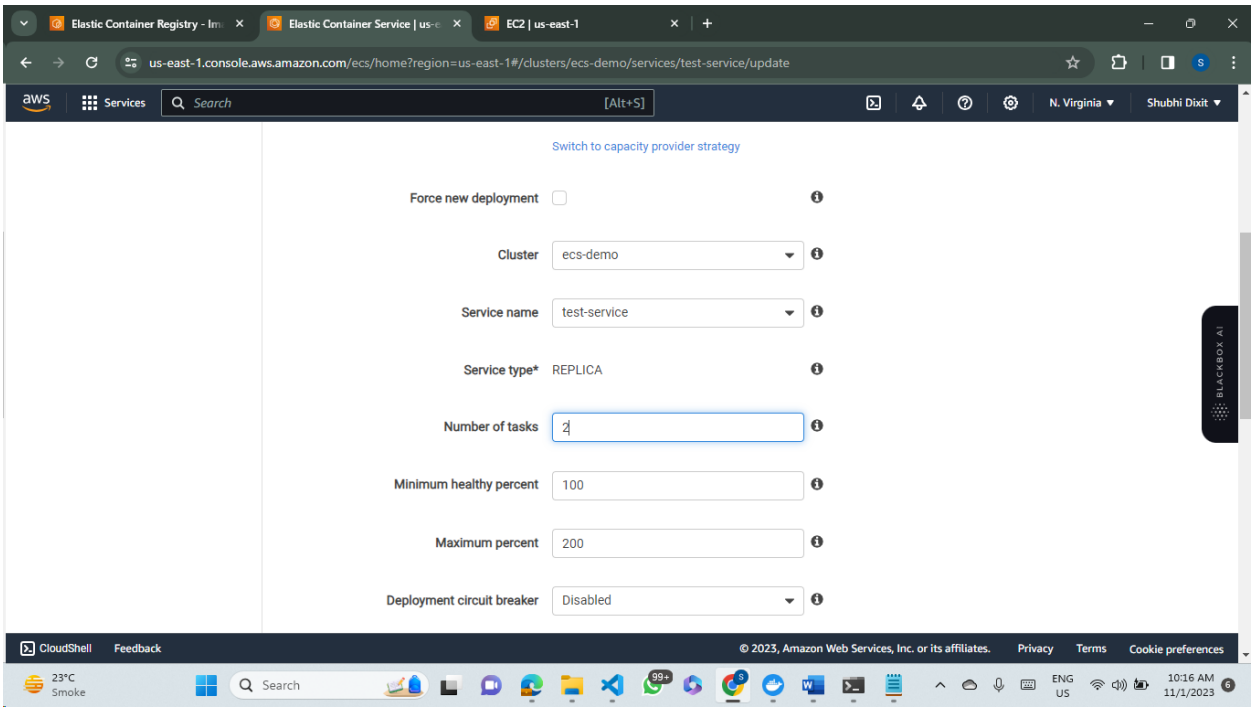


Multiple tasks on EC2 instances

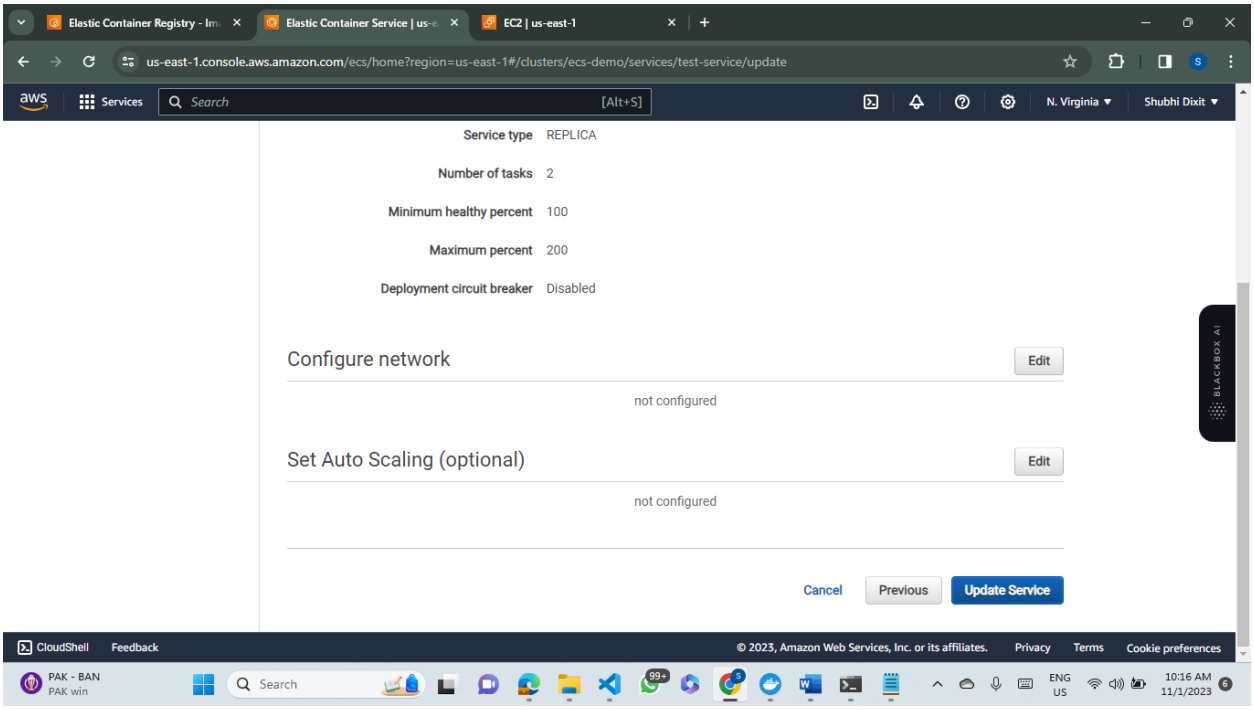
1) Go to clusters and go to services tab. Select the service that we created and click on update.

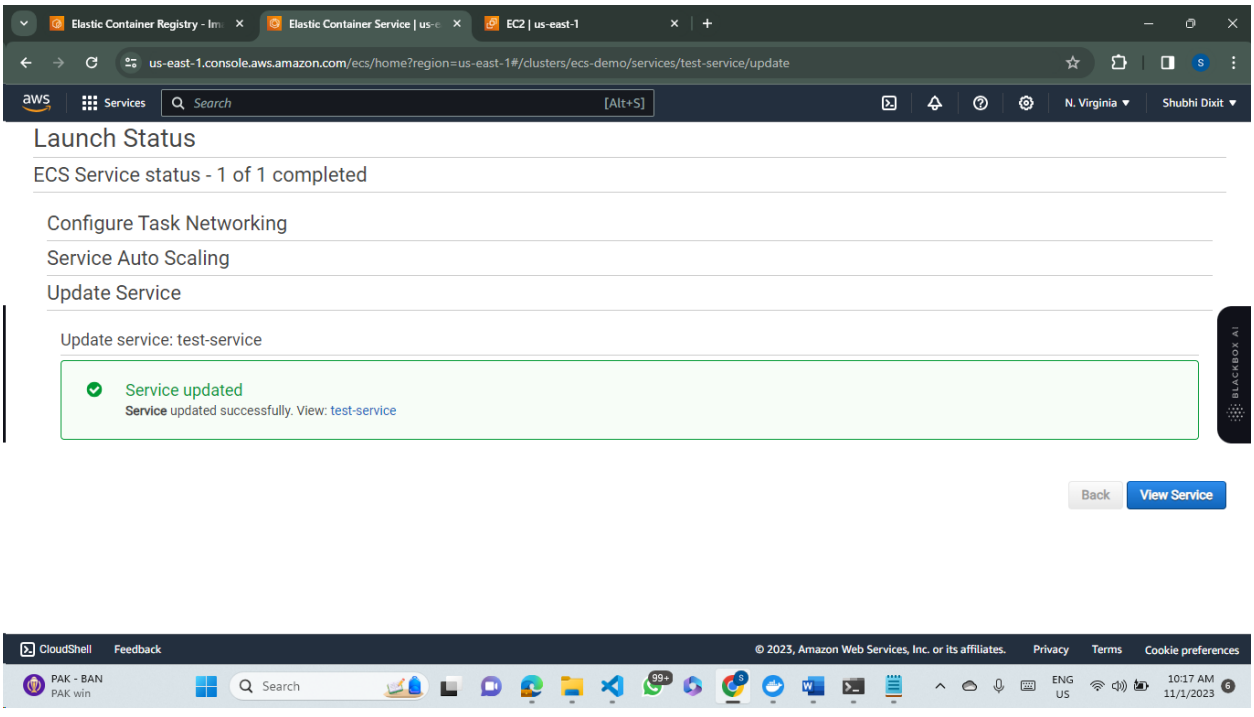


2) Now change the “Number of tasks” to 2.

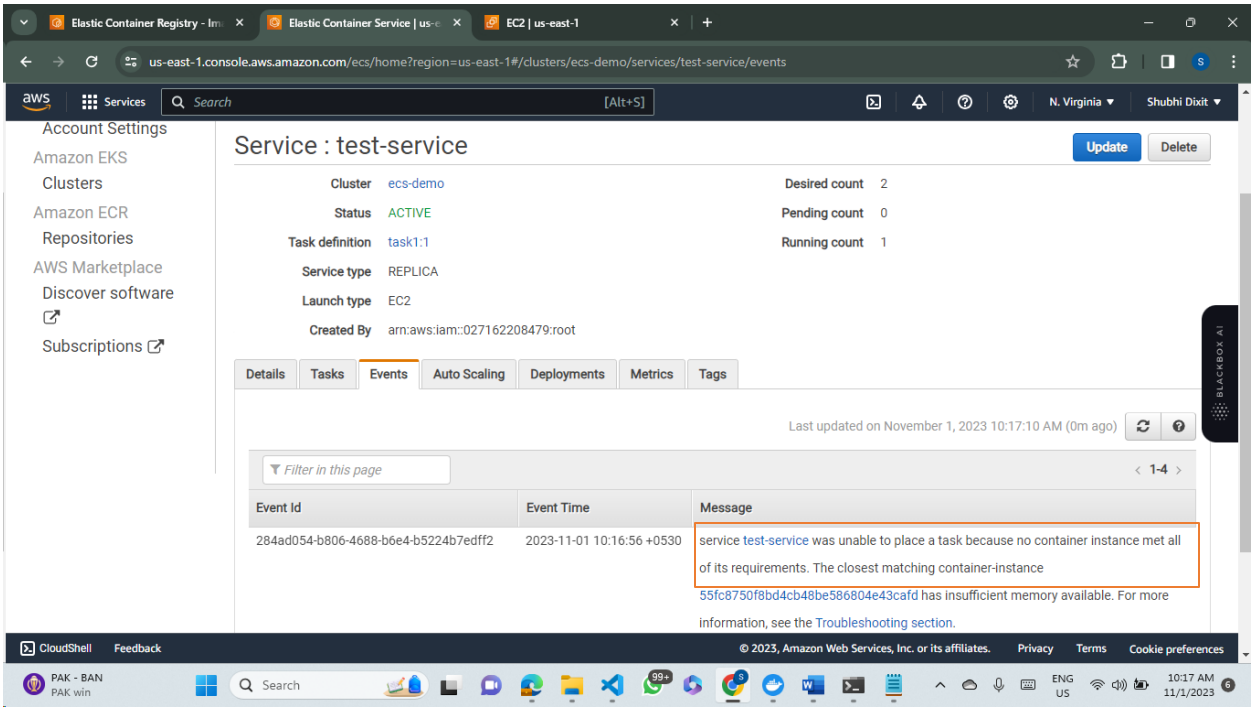


3) Leave the rest of the settings as default and update the service.





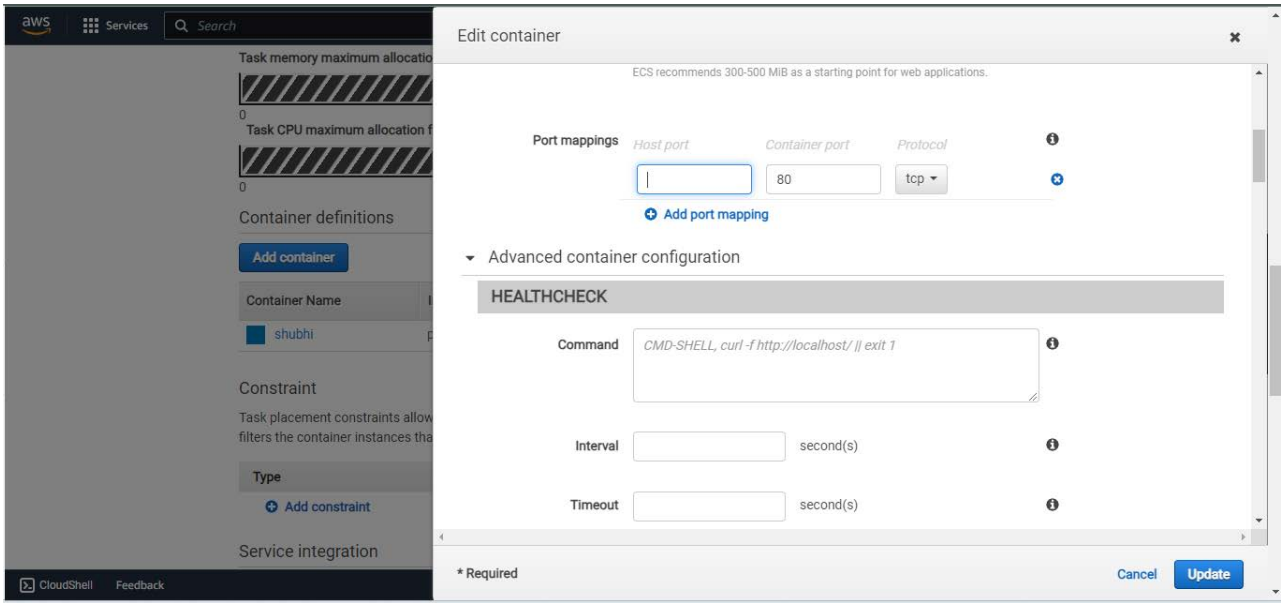
4) Now select the service and go to the “events” tab. We can see an error message.



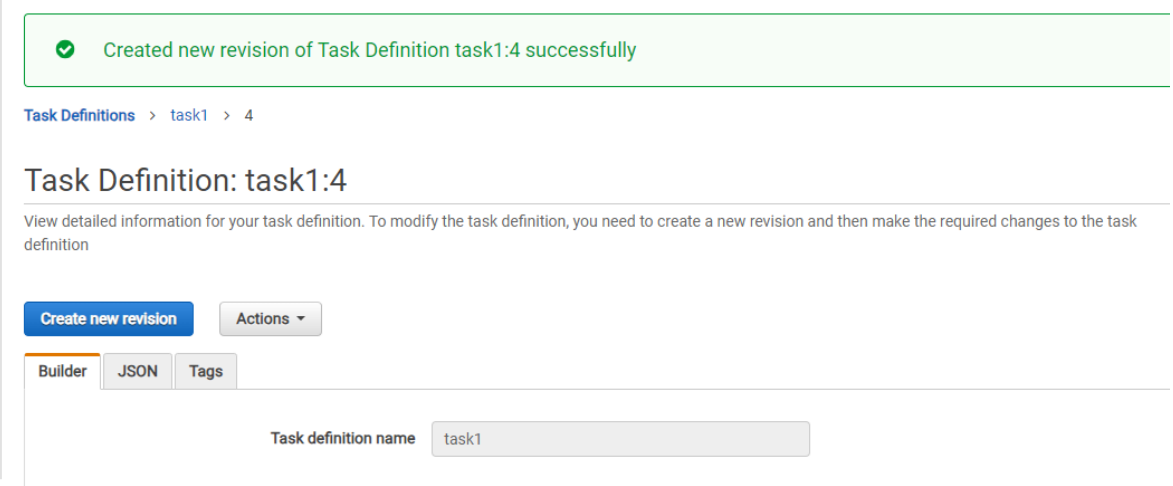
This is because the port that we assigned while creating the task definition is busy and can be assigned to only one instance so we have to change it.

5)Go to task definitions and select the task and update it.

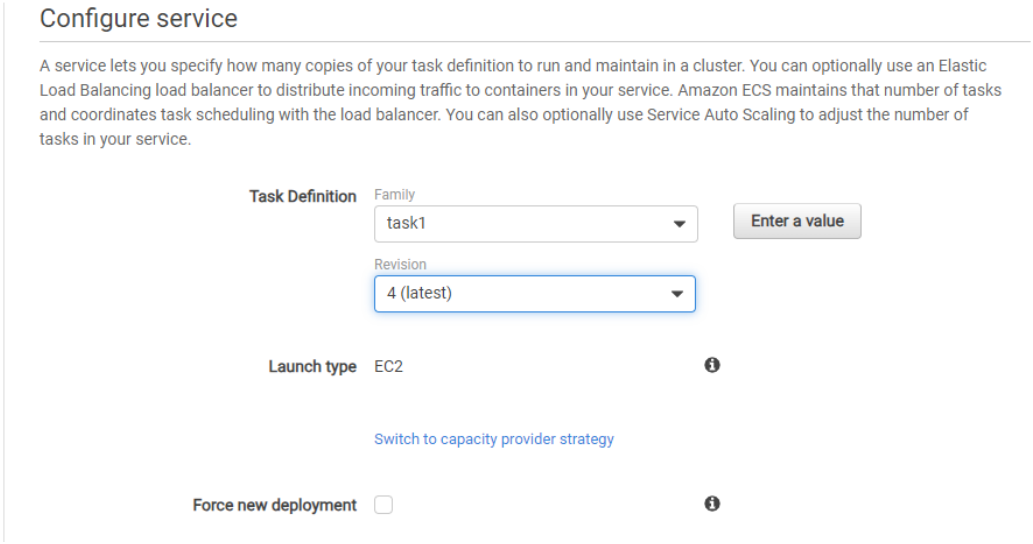
6) Remove the host port and leave it blank.



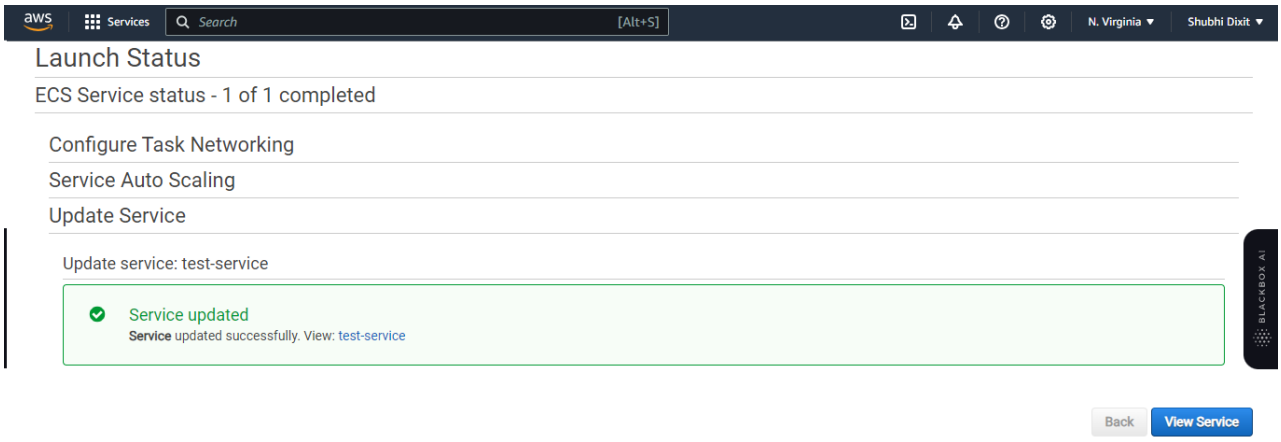
7) Now click on update and the revised task definition will be created.



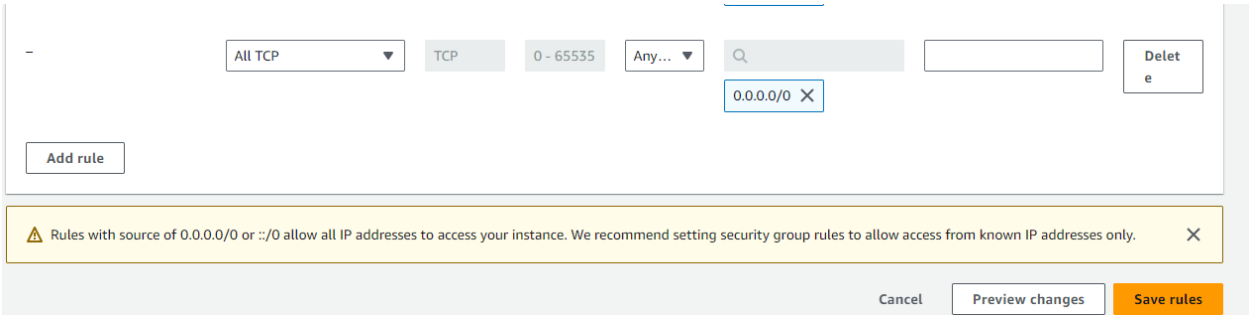
8) Now go to service again and select the revised task definition that we just created i.e. 4(latest) here.



9) Leave rest of the fields unchanged and update the service.



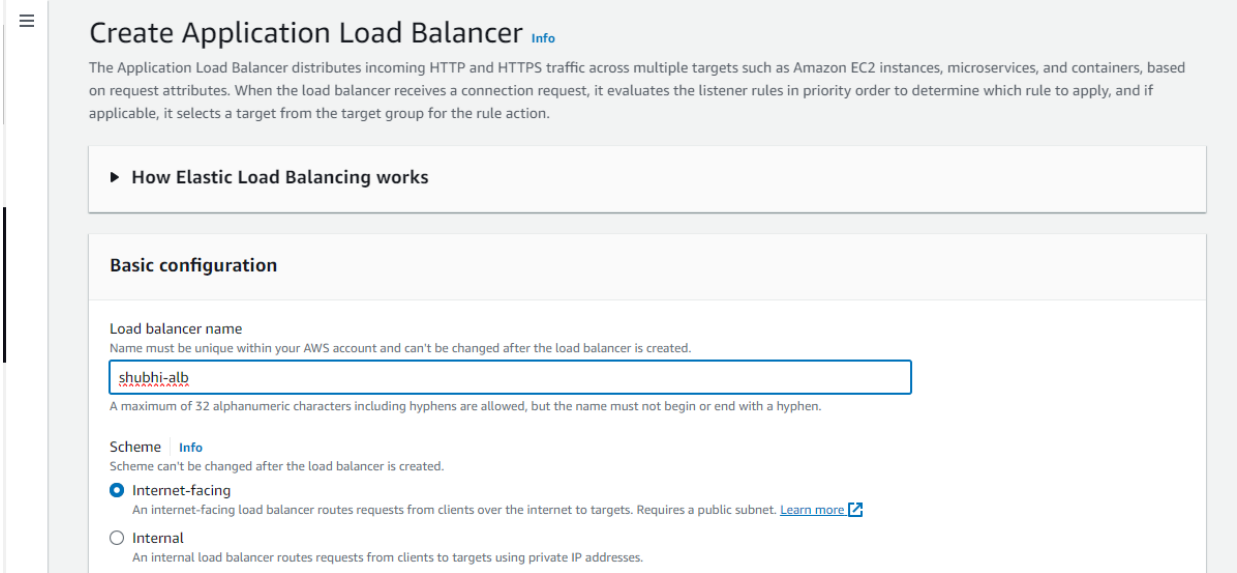
10) Now multiple tasks are running and in order to access them we need to add the security groups to each of them



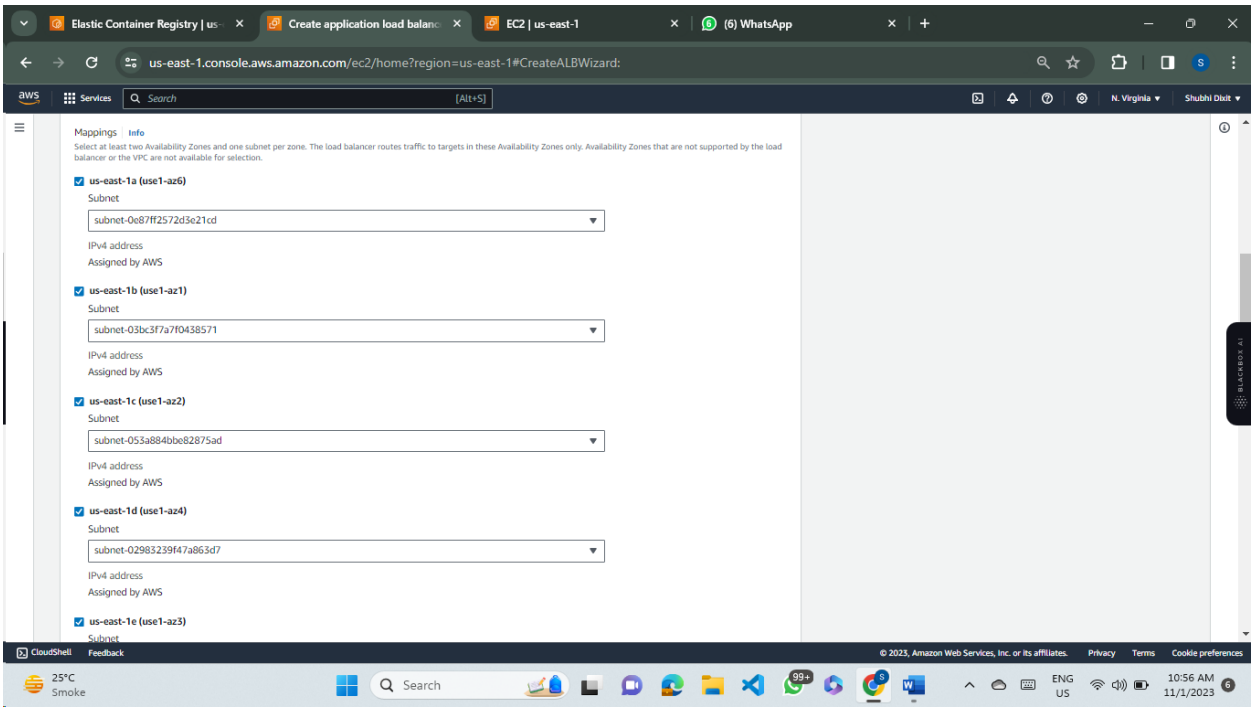
11) After adding the rules in the security group, the web page will be accessible from the two tasks

Attaching Load Balancer

1) Firstly, Go to “Load balancer” and create an application load balancer. Name it first.



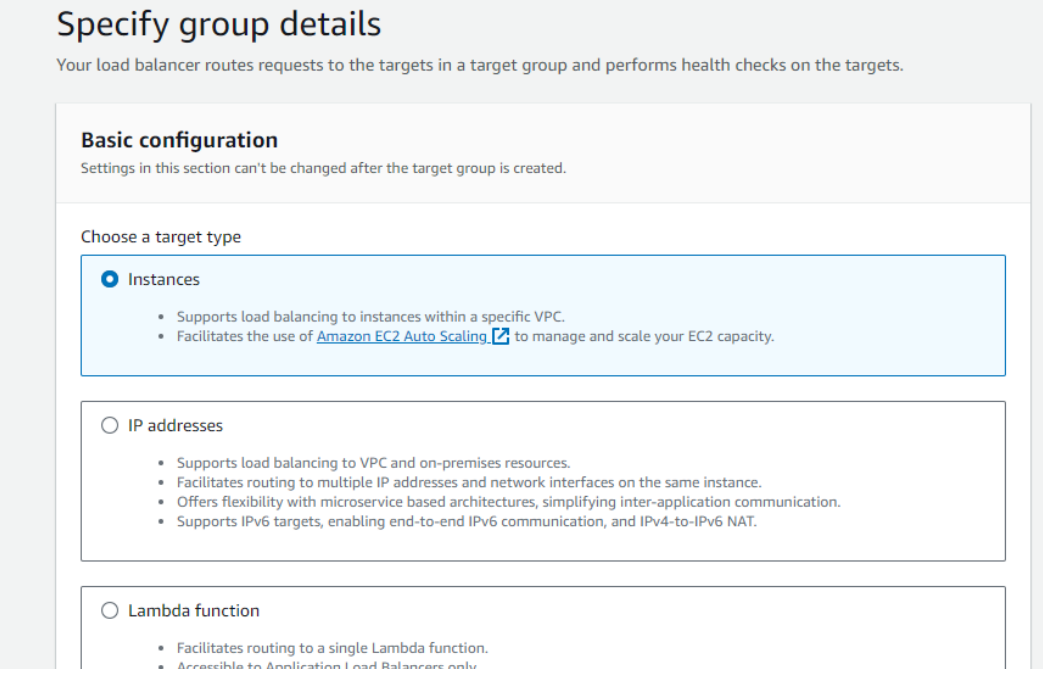
2) Select all the AZs in the mapping field.



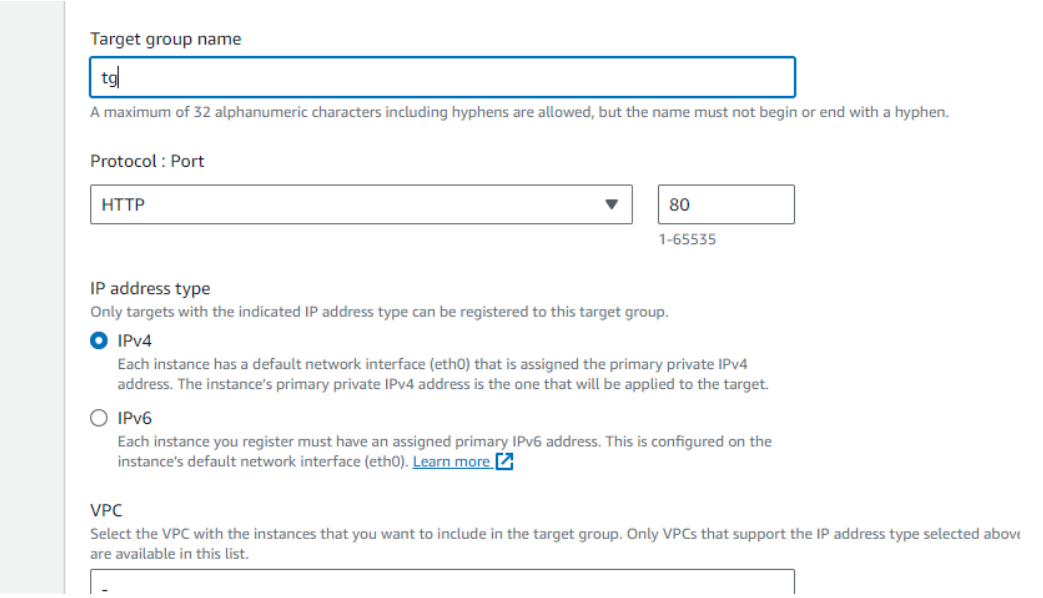
4) Select a target group or create one if you haven’t already.

Target Group Creation

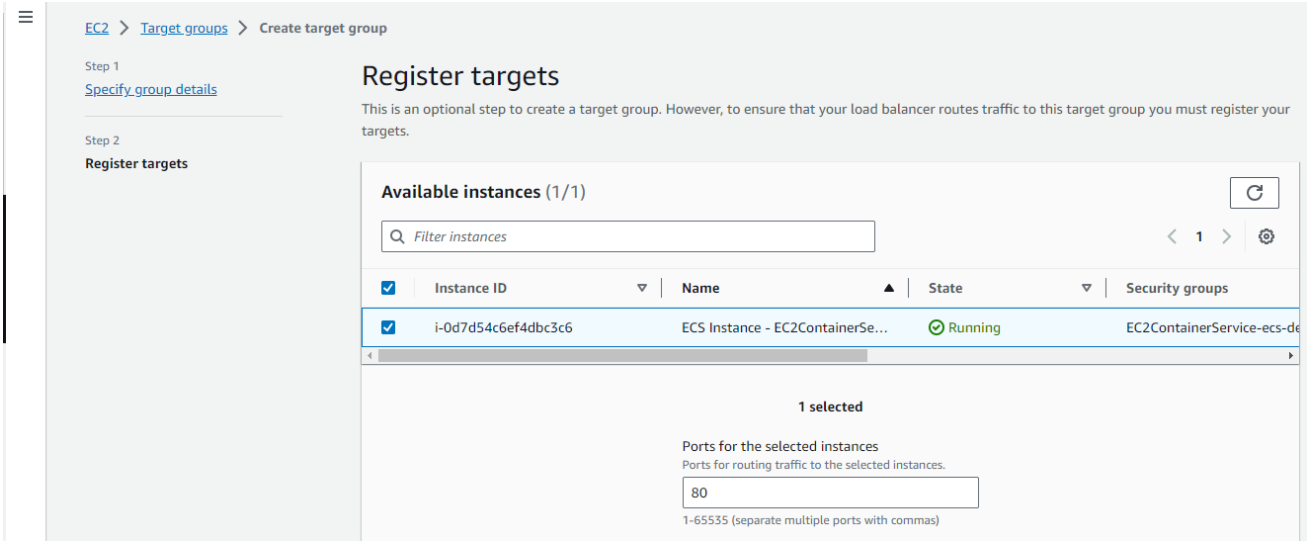
1) Click on instances.



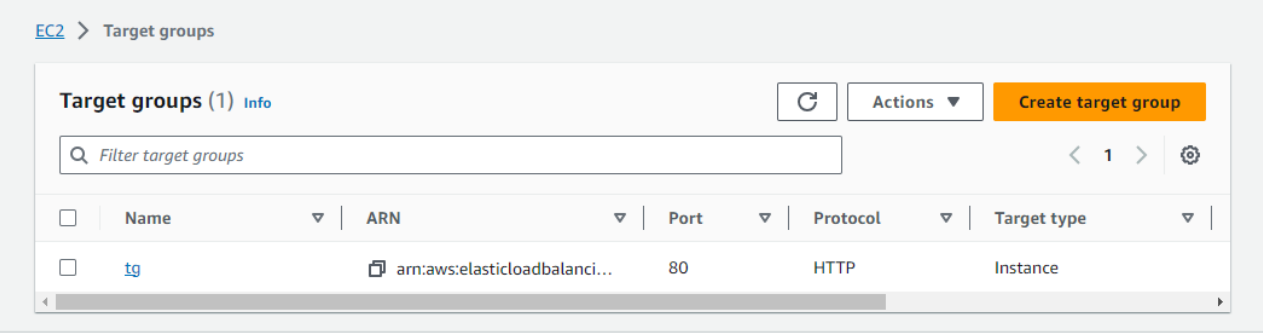
2) Give it a name



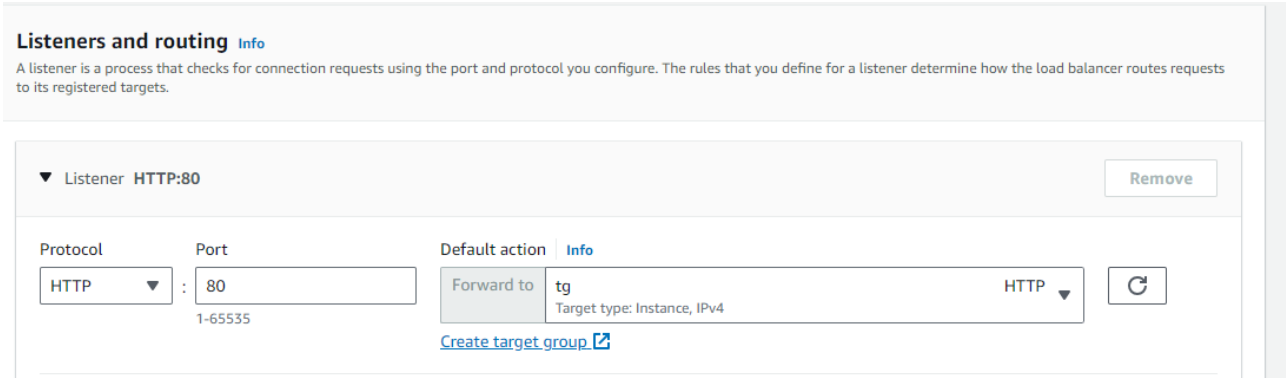
3) Add the container instance which we created earlier.



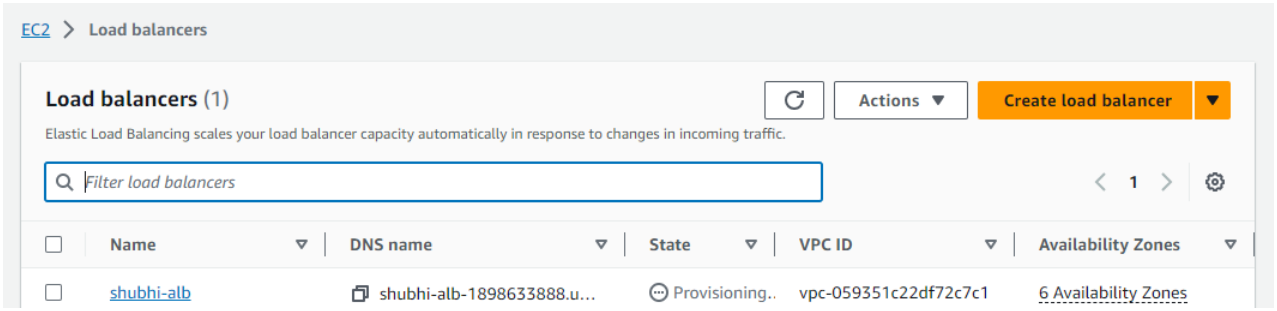
4) Review the changes and click “create target group”

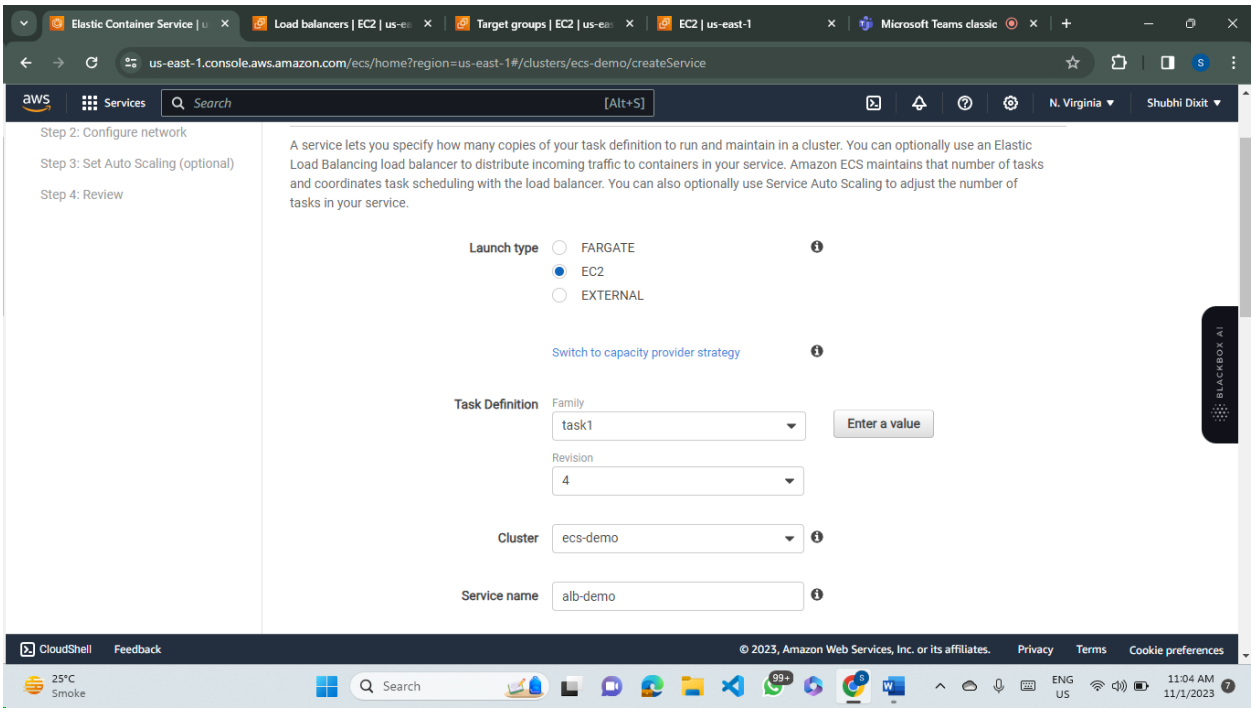


Add this target group to your load balancer.

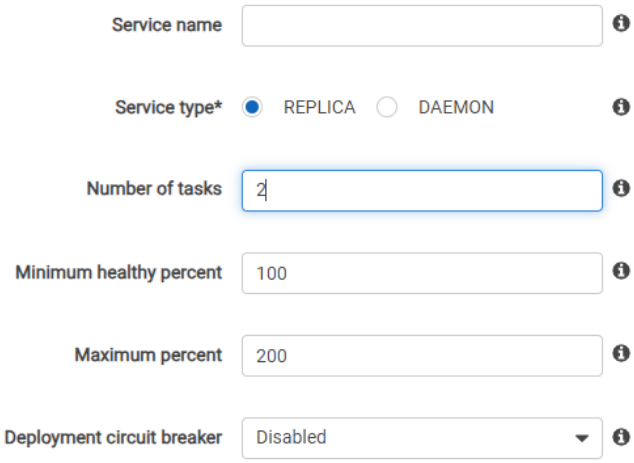


4) Leave other settings default and click on create load balancer

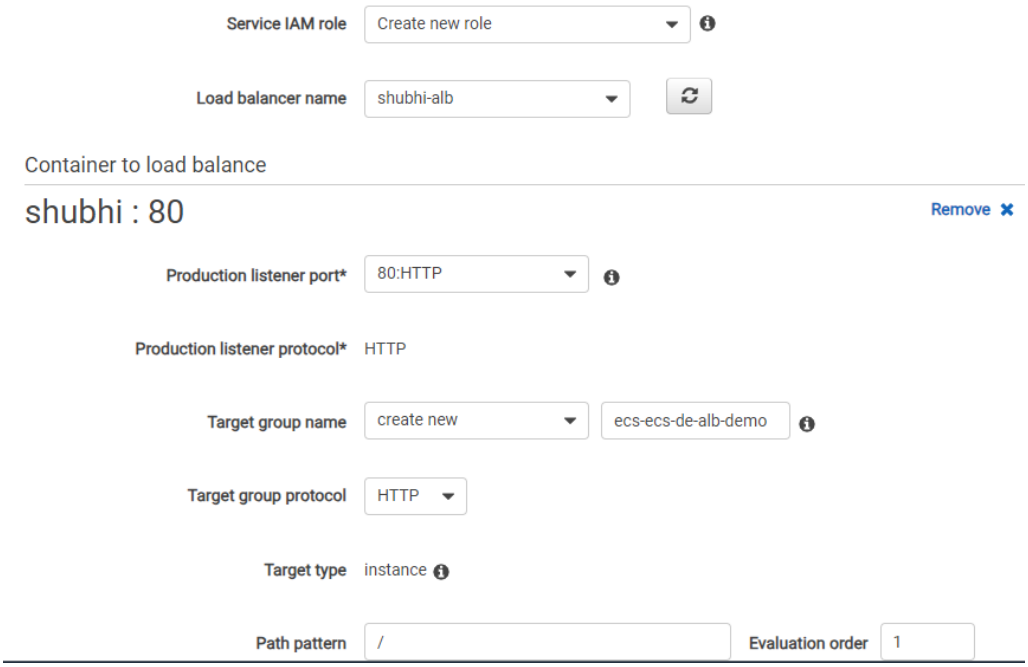




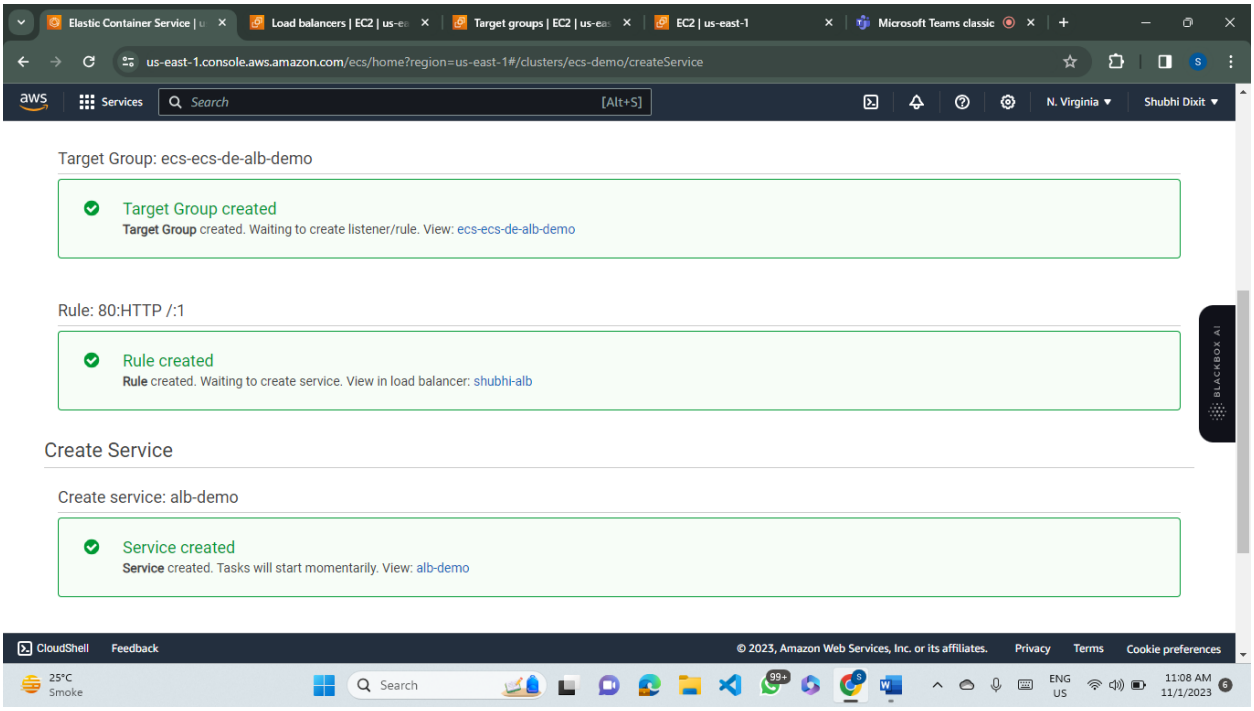
5) Follow the steps for service creation and create a new service.



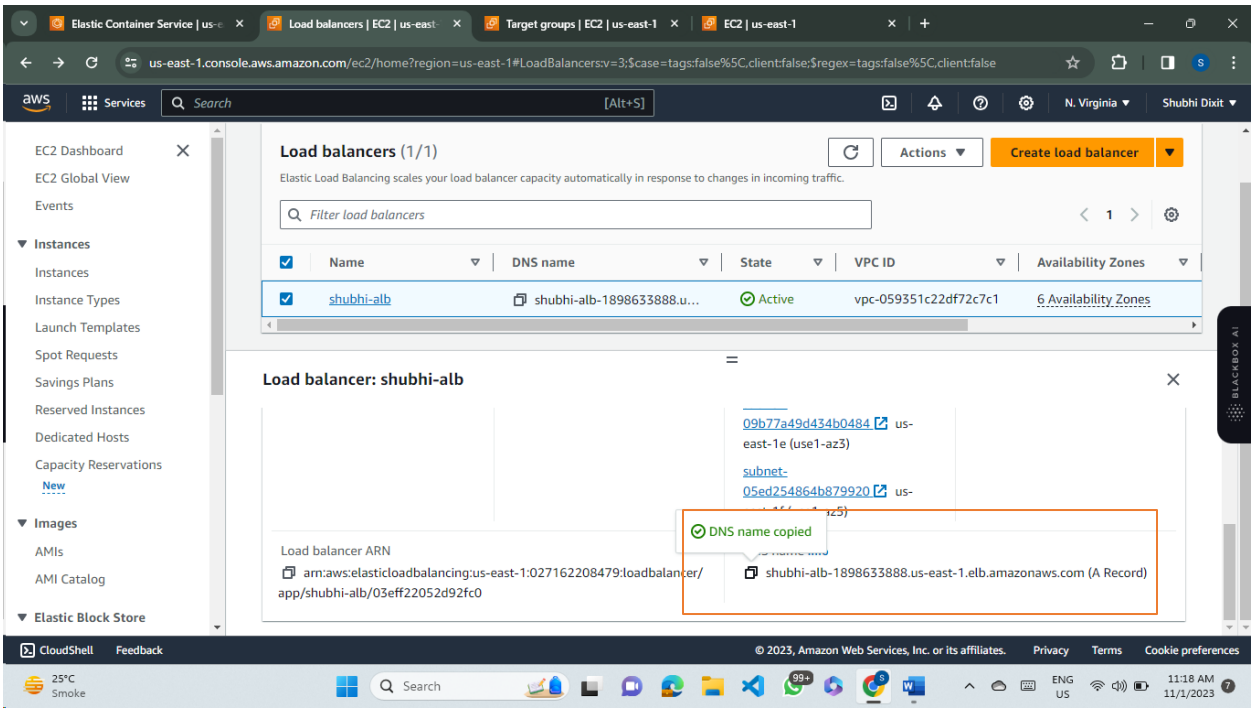
6) During creating the service, add the target group and ALB to the service



7) Set other settings as default and click on create service

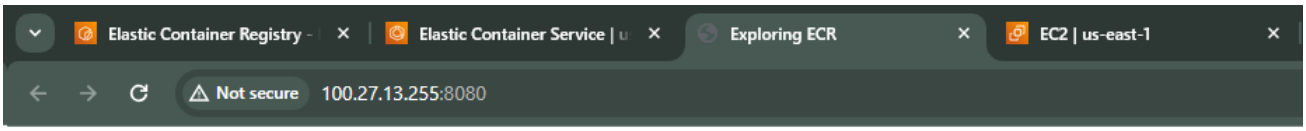


8) Now access the web page through load balancer's **DNS**



It is accessible.

Also, you can access through external link



This is the body of index page.