

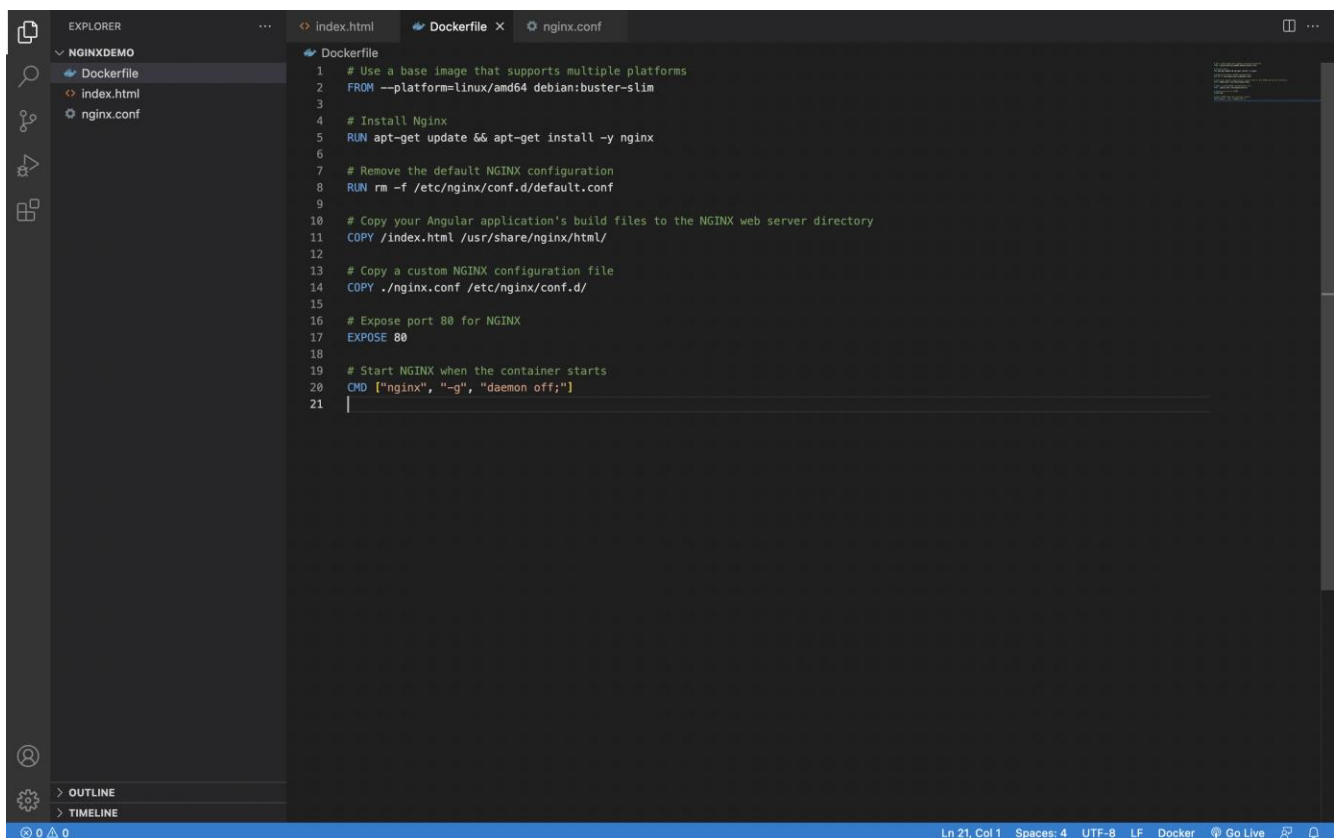
Pushing Docker Image to AWS ECR

Pre-requisites:

1. Docker Desktop
2. AWS CLI
3. Visual Studio Code

Steps:

1. Create folder and open in visual studio code.
2. Create Docker file and build Docker Image in local machine.



The screenshot shows the Visual Studio Code interface with a project named 'NGINXDEMO'. The Explorer sidebar on the left shows the file structure: 'Dockerfile', 'index.html', and 'nginx.conf'. The main editor area has two tabs open: 'Dockerfile' and 'nginx.conf'. The 'Dockerfile' tab is active, displaying the following content:

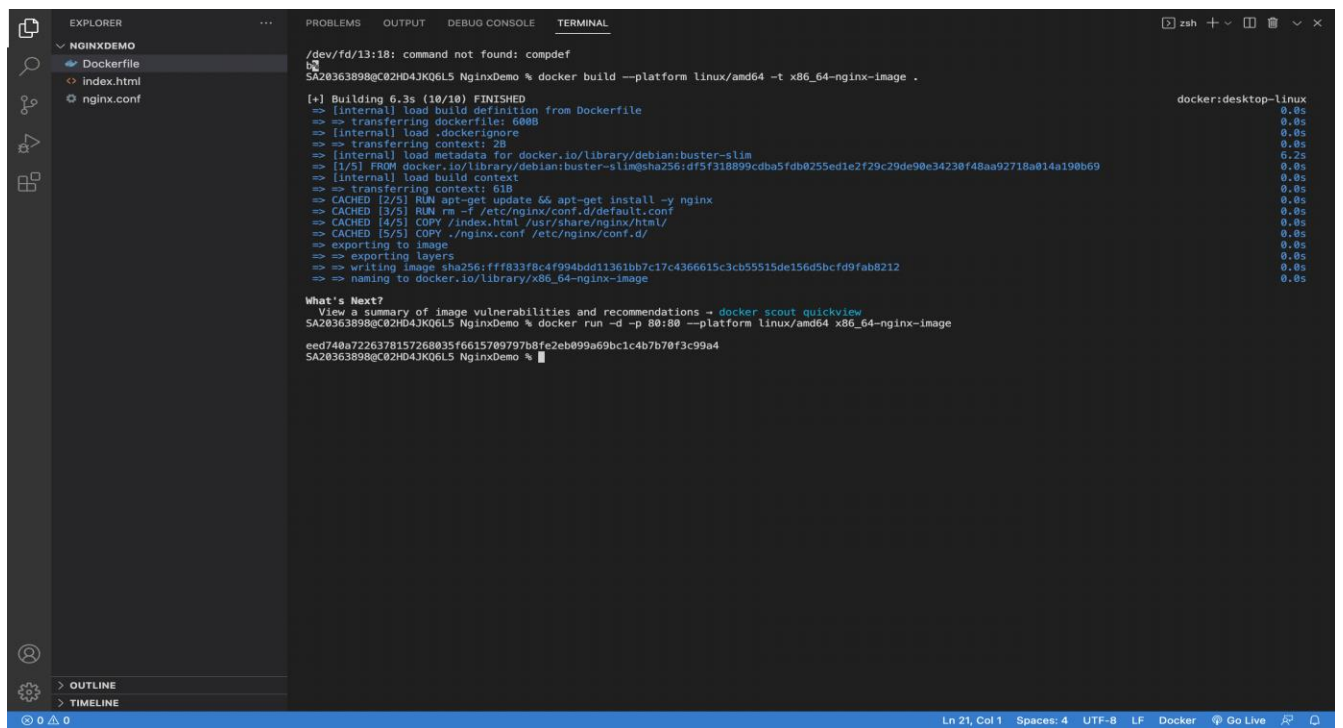
```
1 # Use a base image that supports multiple platforms
2 FROM --platform=linux/amd64 debian:buster-slim
3
4 # Install Nginx
5 RUN apt-get update && apt-get install -y nginx
6
7 # Remove the default NGINX configuration
8 RUN rm -f /etc/nginx/conf.d/default.conf
9
10 # Copy your Angular application's build files to the NGINX web server directory
11 COPY /index.html /usr/share/nginx/html/
12
13 # Copy a custom NGINX configuration file
14 COPY ./nginx.conf /etc/nginx/conf.d/
15
16 # Expose port 80 for NGINX
17 EXPOSE 80
18
19 # Start NGINX when the container starts
20 CMD ["nginx", "-g", "daemon off;"]
21
```

The 'nginx.conf' tab is also open but its content is not visible. The status bar at the bottom indicates 'Ln 21, Col 1', 'Spaces: 4', 'UTF-8', 'LF', 'Docker', and 'Go Live'.

Open terminal and run following commands

For Building Docker Image command is : `docker build -t imagename .`

For running Container command is : `docker -p hostport:containerport imagename`



The screenshot shows a VS Code interface with a terminal window open. The Explorer pane on the left shows a project named 'NGINXDEMO' with files 'Dockerfile', 'index.html', and 'nginx.conf'. The terminal window displays the following commands and output:

```
/dev/fd/13:18: command not found: compdef
SA20363898@c02HD4JKQ6L5 NginxDemo % docker build --platform linux/amd64 -t x86_64-nginx-image .

[+] Building 6.3s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 600B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/debian:buster-slim
=> [1/5] FROM docker.io/library/debian:buster-slim@sha256:df5f318899cdba5fdb0255ed1e2f29c29de90e34230f48aa92718a014a198b69
=> [internal] load build context
=> => transferring context: 61B
=> CACHED [2/5] RUN apt-get update && apt-get install -y nginx
=> CACHED [3/5] RUN rm -f /etc/nginx/conf.d/default.conf
=> CACHED [4/5] COPY /index.html /usr/share/nginx/html/
=> CACHED [5/5] COPY ./nginx.conf /etc/nginx/conf.d/
=> exporting image
=> => exporting layers
=> => writing image sha256:fff833f8c4f904bdd11361bb7c17c4366615c3cb55515de156d5b6d9fab0212
=> => naming to docker.io/library/x86_64-nginx-image

What's Next?
View a summary of image vulnerabilities and recommendations -> docker scout quickview
SA20363898@c02HD4JKQ6L5 NginxDemo % docker run -d -p 80:80 --platform linux/amd64 x86_64-nginx-image
eed748a7226378157268035f6615709797b0fe2eb099a69bc1c4b7b70f3c99a4
SA20363898@c02HD4JKQ6L5 NginxDemo %
```

Now login to Amazon Web Service Console

1. Open AWS Elastic Container Registry (ECR)
2. In menu open Repositories and create Repository

App Store | Email | OneNote | AWS Docs | Dongari | (9) how | Elastic C | Elastic C | EC2 | us | Check H | New Tab | +

us-east-1.console.aws.amazon.com/ecr/repositories?region=us-east-1

aws Services Search [Option+S] N. Virginia Veerendra @ 3350-0516-4513

Amazon Elastic Container Registry

- Private registry
- Public registry
- Repositories
- ECR public gallery
- Amazon ECS
- Amazon EKS
- Getting started
- Documentation

Successfully deleted repository x86_64-nginx-image

Amazon ECR > Repositories

Private Public

Public repositories

View push commands Delete Actions Create repository

Find repositories

Repository name	URI	Created at
No repositories No repositories were found		

CloudShell Feedback © 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

- After creating Repository select that Repository and view push commands

The screenshot shows the AWS Elastic Container Registry console. A modal window titled "Push commands for x86_64-nginx-image" is displayed, providing instructions and commands for pushing an image to the repository. The modal includes a success message at the top: "Successfully created repository x86_64-nginx-image". The instructions are as follows:

- Make sure that you have the latest version of the AWS CLI and Docker installed. For more information, see [Getting started with Amazon ECR](#).**
- Use the following steps to authenticate and push an image to your repository. For additional registry authentication methods, including the Amazon ECR credential helper, see [Registry authentication](#).
- 1. Retrieve an authentication token and authenticate your Docker client to your registry.
Use the AWS CLI:

```
aws ecr-public get-login-password --region us-east-1 | docker login --username AWS --password-stdin public.ecr.aws/h4w9i4p3
```

Note: If you receive an error using the AWS CLI, make sure that you have the latest version of the AWS CLI and Docker installed.
- 2. Build your Docker image using the following command. For information on building a Docker file from scratch, see the instructions [here](#). You can skip this step if your image has already been built:

```
docker build -t x86_64-nginx-image .
```
- 3. After the build is completed, tag your image so you can push the image to this repository:

```
docker tag x86_64-nginx-image:latest public.ecr.aws/h4w9i4p3/x86_64-nginx-image:latest
```
- 4. Run the following command to push this image to your newly created AWS repository:

```
docker push public.ecr.aws/h4w9i4p3/x86_64-nginx-image:latest
```

The modal also includes a "Close" button at the bottom right.

4. Run the above commands in visual studio code.
5. After running the above commands, the docker image is successfully pushed to AWS ECR.

The screenshot shows the Amazon Elastic Container Registry (ECR) console. The left sidebar contains navigation links for Private registry, Public registry, Repositories, Images, Gallery detail, Permissions, Repository tags, ECR public gallery, Amazon ECS, Amazon EKS, Getting started, and Documentation. The main content area displays the 'x86_64-nginx-image' repository. At the top, there are buttons for 'View public listing', 'View push commands', and 'Edit'. Below this, a section titled 'Images (1)' includes a search bar and a table of images. The table has columns for Image tag, Artifact type, Pushed at, Size (MB), Image URI, and Digest. One image is listed with the tag 'latest', pushed on 02 November 2023, 00:05:18 (UTC+05.5), with a size of 67.06 MB. The footer of the console shows 'CloudShell', 'Feedback', and copyright information for Amazon Web Services.

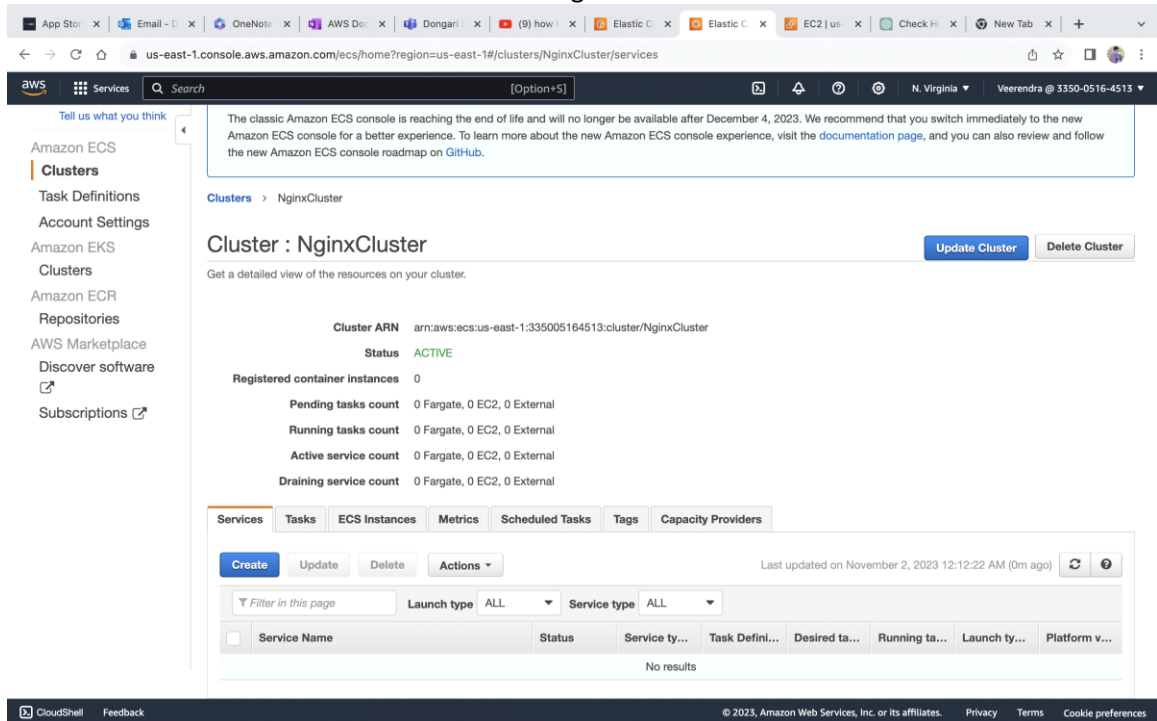
Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest
latest	Image	02 November 2023, 00:05:18 (UTC+05.5)	67.06	Copy URI	sha256:40c202e9305b30...

6. Now open AWS Elastic Container Service (ECS)

7. Create Cluster in ECS

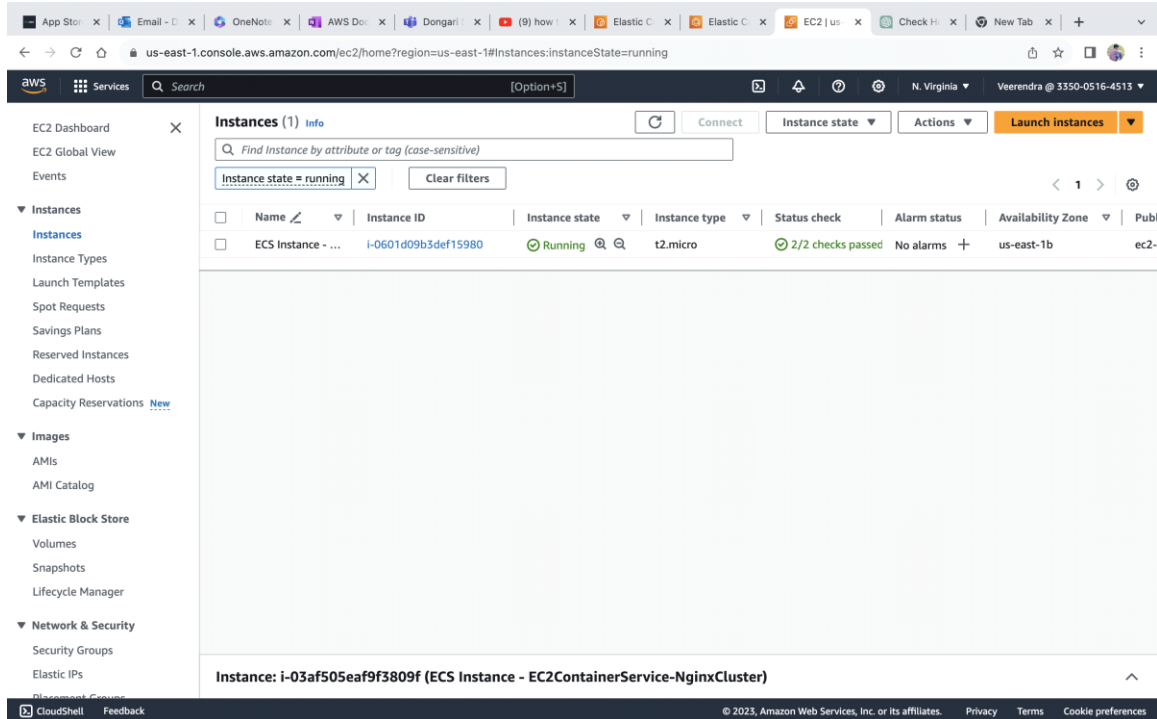
The screenshot shows the Amazon Elastic Container Service (ECS) console. The left sidebar contains navigation links for Amazon ECS, Clusters, Task Definitions, Account Settings, Amazon EKS, Clusters, Amazon ECR, Repositories, AWS Marketplace, Discover software, and Subscriptions. The main content area displays the 'Clusters' page. At the top, there is a notification about the classic Amazon ECS console reaching the end of its life. Below this, a section titled 'Clusters' provides a description of an Amazon ECS cluster and a link to the ECS documentation. There are buttons for 'Create Cluster' and 'Get Started'. Below the buttons, there is a 'View' section with 'list' and 'card' tabs, and a '0 loaded of 0 clusters' status. A table with columns for Cluster name, CloudWatch monitoring, Services, Running tasks, Pending tasks, and Container instances is shown, with a 'loading' status at the bottom. The footer of the console shows 'CloudShell', 'Feedback', and copyright information for Amazon Web Services.

8. Click the create cluster button and make configurations based on the need.

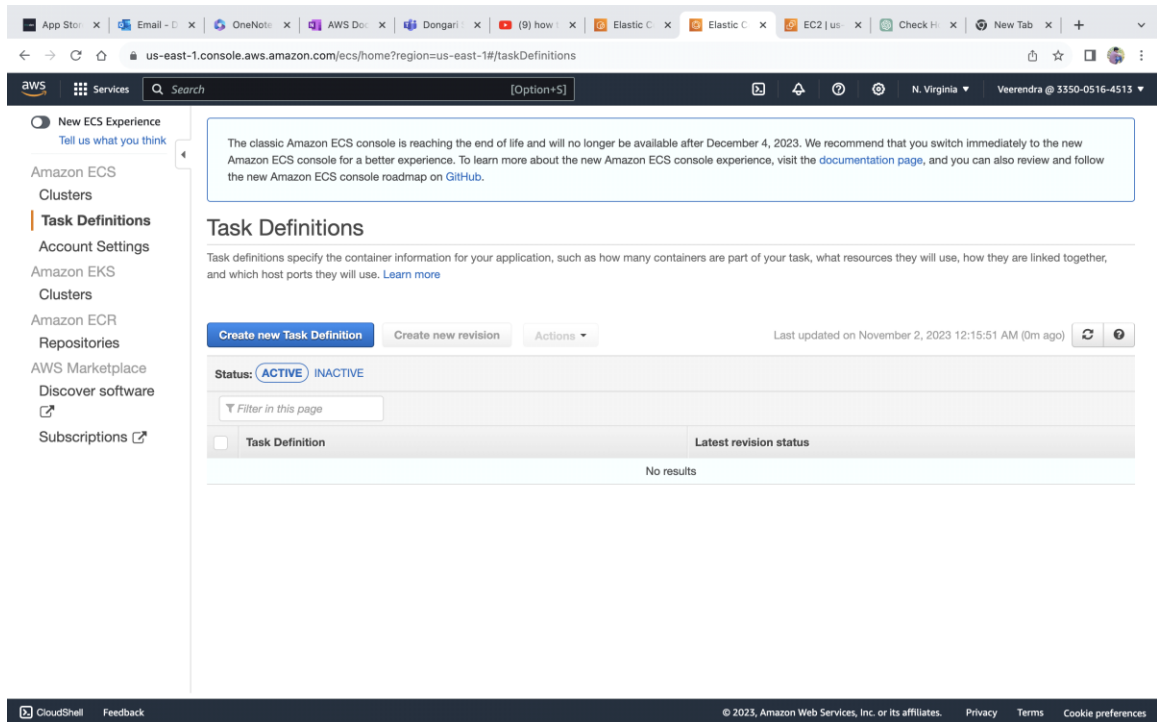


9. After creating the cluster, it will look like in the above screenshot.

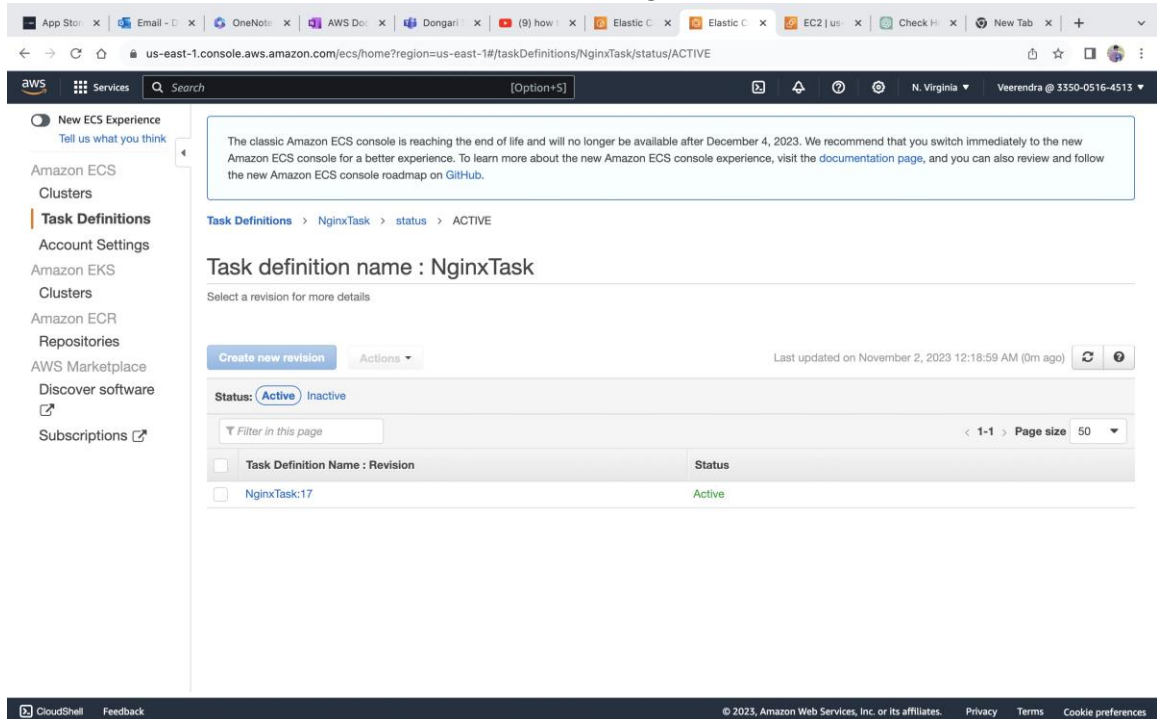
10. After creating the cluster, it also creates one EC2 instance.



11. Now create task definition to run tasks.



12. Click the create task definition button and make configurations based on the need.



13. After creating the Task Definition, it will look it in the above screenshot.

14. Now go to cluster that we created and go to tasks and Run new Task.

Task Ids : ["NginxCluster/eacc74f369b43a395af35f484e6fd0d"]

Cluster : NginxCluster

Get a detailed view of the resources on your cluster.

Cluster ARN: `arn:aws:ecs:us-east-1:335005164513:cluster/NginxCluster`

Status: **ACTIVE**

Registered container instances: 1

Pending tasks count: 0 Fargate, 1 EC2, 0 External

Running tasks count: 0 Fargate, 0 EC2, 0 External

Active service count: 0 Fargate, 0 EC2, 0 External

Draining service count: 0 Fargate, 0 EC2, 0 External

Services | **Tasks** | ECS Instances | Metrics | Scheduled Tasks | Tags | Capacity Providers

Run new Task | Stop | Stop All | Actions

Last updated on November 2, 2023 12:24:12 AM (0m ago)

Desired task status: **Running** Stopped

Filter in this page Launch type: ALL

Task	Task definit...	Container i...	Last status...	Desired sta...	Started at ...	Started By ...	Group	Launch typ...	Platform ve...
<input type="checkbox"/>	eacc74f369...	NginxTask:17	bb3b65789...	RUNNING	RUNNING	2023-11-02 ...	family:Nginx...	EC2	--

15. After running a task it will create one task like in the above screenshot.

16. Now to go to EC2 instance and select that EC2 instance go to security tab and select security groups.

Inbound security group rules successfully modified on security group (sg-0f8c03da2032f4266 | default)

sg-0f8c03da2032f4266 - default

Details

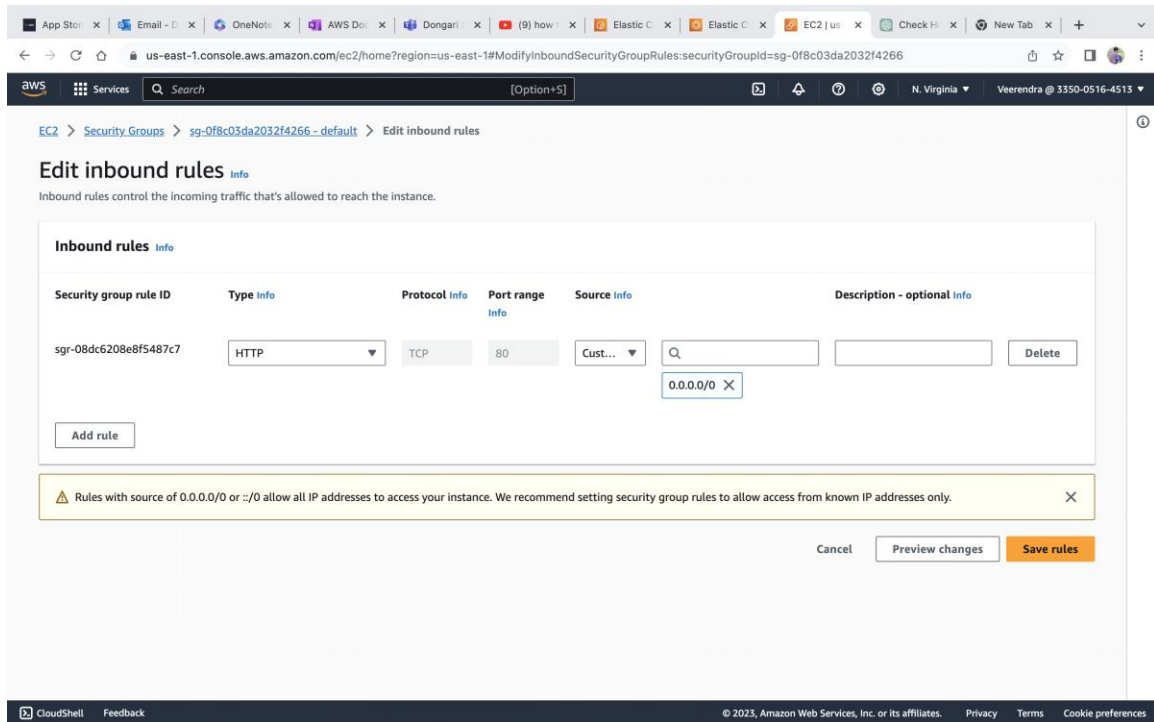
Security group name default	Security group ID sg-0f8c03da2032f4266	Description default VPC security group	VPC ID vpc-0325236a4f8a8d9f
Owner 335005164513	Inbound rules count 1 Permission entry	Outbound rules count 1 Permission entry	

Inbound rules | Outbound rules | Tags

Inbound rules (1/1)

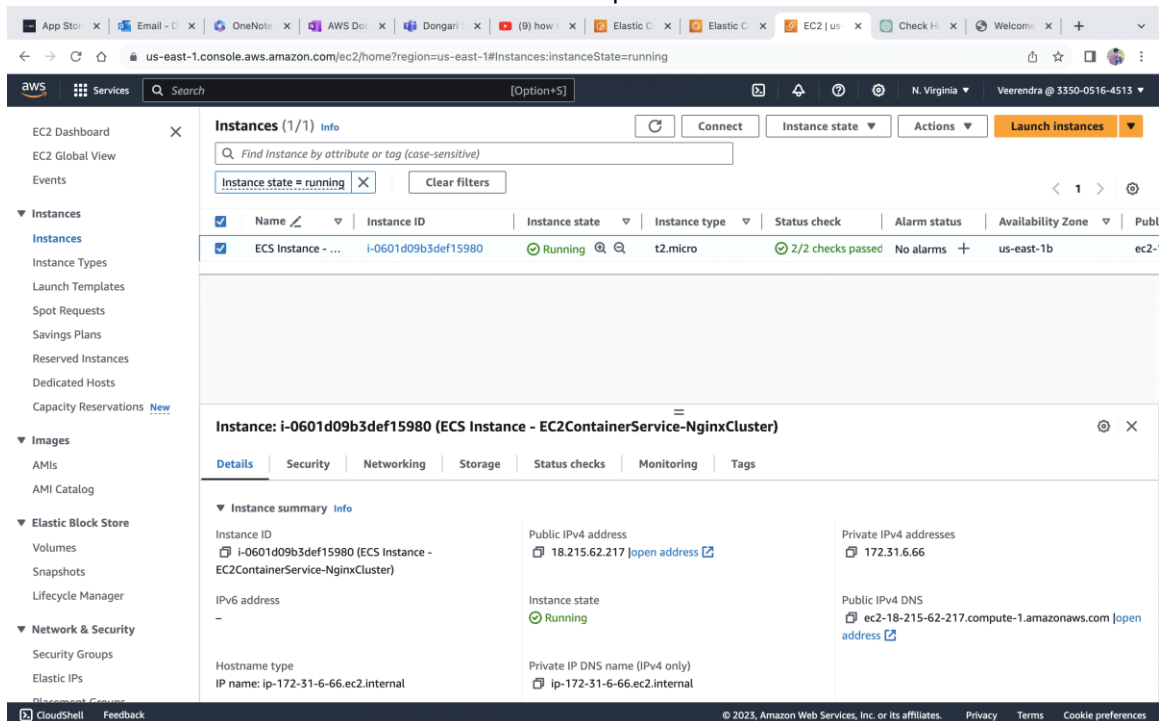
Name	Security group rule...	IP version	Type	Protocol	Port rang
<input checked="" type="checkbox"/>	sg-r-04cc1f3c4979672c8	IPv4	HTTP	TCP	80

17. Click Edit inbound rules and add configuration.



18. Click on Add rule and add port number and select AnyWhere IPv4 in Source dropdown and click on save rule.

19. After that go to EC2 instance and go to Details tab and copy public IPv4 address and then paste in browser and click enter then it will show the output.



20. This is the process for pushing the local docker image to AWS ECR and running container in AWS

