

## CONTAINER ORCHESTRATION AND INFRASTRUCTURE AUTOMATION

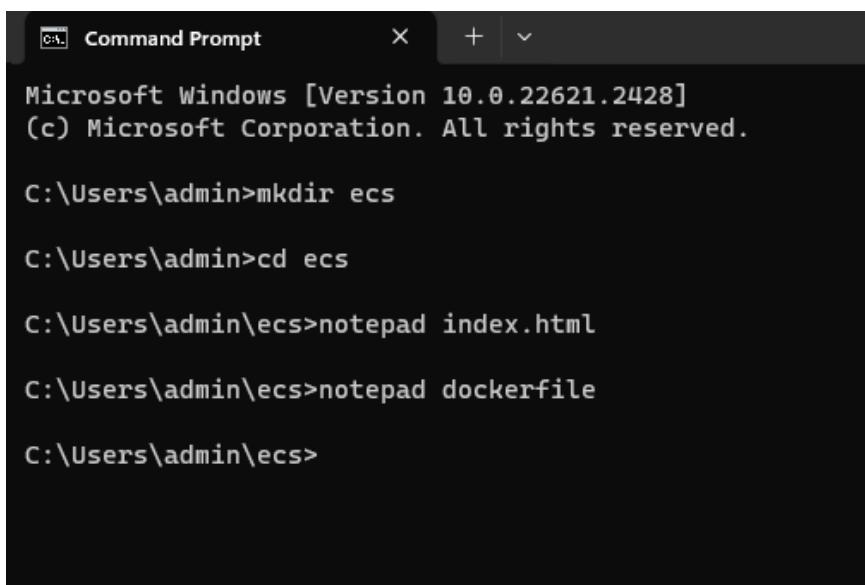
NAME- Shubhi Dixit

BATCH- 05

SAP ID- 500094571

### ECS Experiment

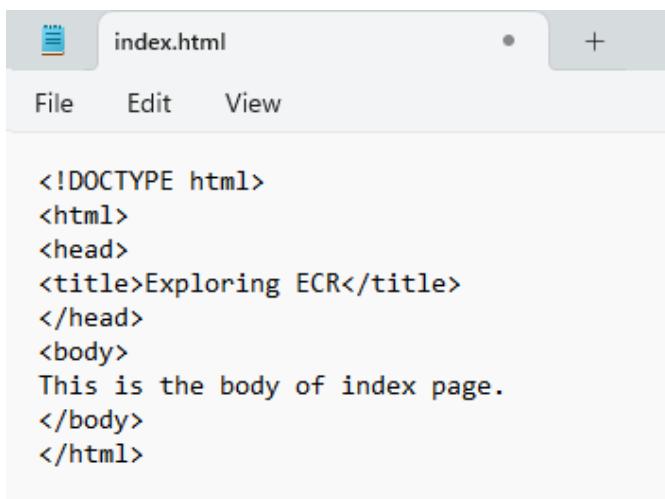
Step: 1- Firstly using the dockerfile, build the image of the setup.



```
Command Prompt      X + ▾
Microsoft Windows [Version 10.0.22621.2428]
(c) Microsoft Corporation. All rights reserved.

C:\Users\admin>mkdir ecs
C:\Users\admin>cd ecs
C:\Users\admin\ecs>notepad index.html
C:\Users\admin\ecs>notepad dockerfile
C:\Users\admin\ecs>
```

### index.html



```
index.html      +
File Edit View

<!DOCTYPE html>
<html>
<head>
<title>Exploring ECR</title>
</head>
<body>
This is the body of index page.
</body>
</html>
```

## dockerfile

```
FROM httpd:2.4
RUN apt-get update
COPY ./index.html /usr/local/apache2/htdocs
EXPOSE 80
ENTRYPOINT ["/usr/local/apache2/bin/httpd", "-D", "FOREGROUND"]
```

```
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@sha256:7b8098fbbdaa8d12c3ae2208b249e7adbe12376fe8d86ef2aae27cef13b7167
=> => resolve docker.io/library/httpd:2.4@sha256:7b8098fbbdaa8d12c3ae2208b249e7adbe12376fe8d86ef2aae27cef13b7167
=> => sha256:7b8098fbbdaa8d12c3ae2208b249e7adbe12376fe8d86ef2aae27cef13b7167 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
```

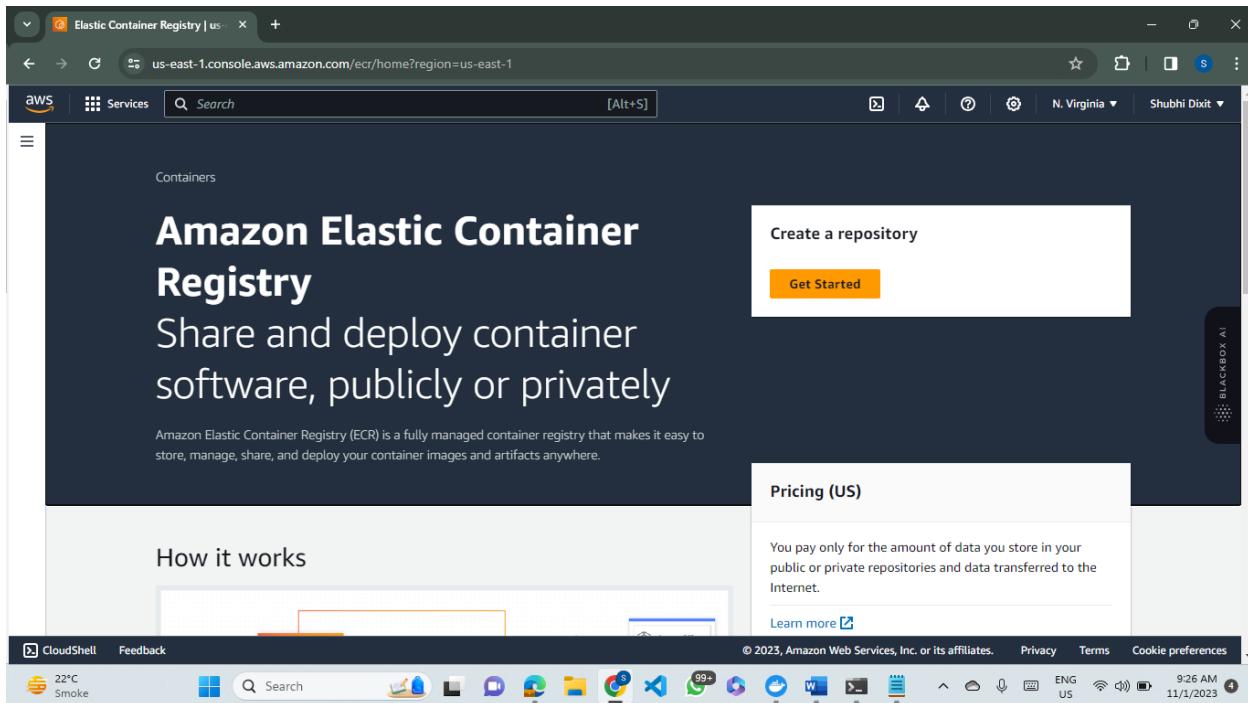
Step-2: After successfully building the image from the container, access the application at localhost: 3000

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

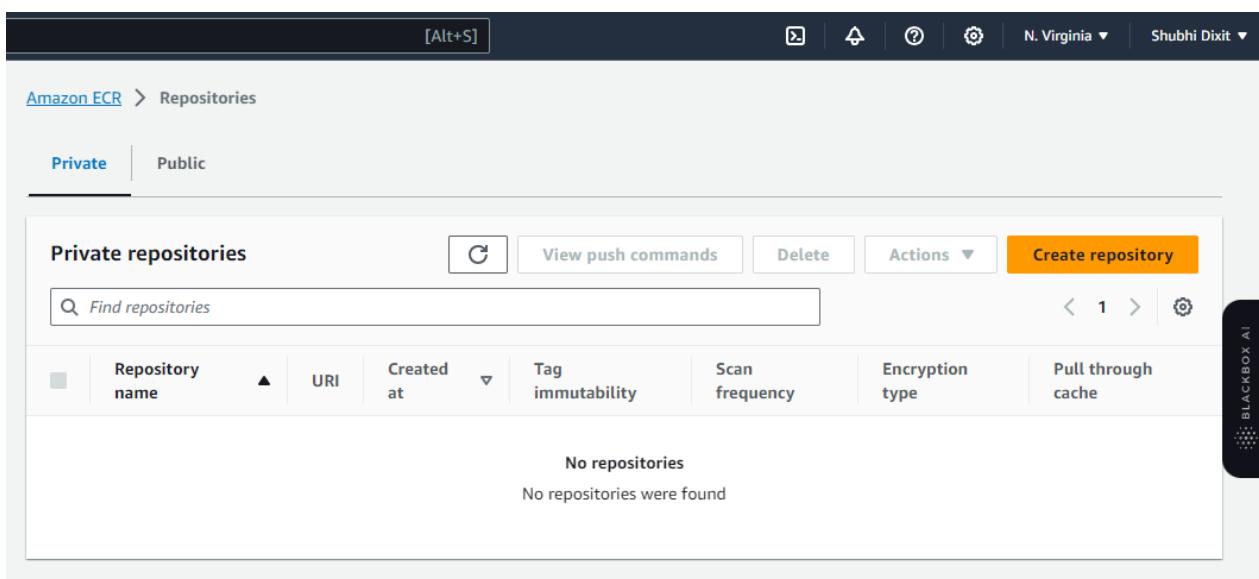
If this is working fine, then follow the next steps.

## **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

The screenshot shows the AWS ECR 'Create repository' interface. In the 'General settings' section, the visibility is set to 'Public'. A note states that visibility cannot be changed once the repository is created. In the 'Detail' section, the repository name is 'shubhi\_ecs', which is also listed as a default alias. A note explains that a default alias is associated with the public registry once a repository is created.

**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.

**Detail**

Repository name | [Info](#)  
A namespace can be included with your repository name (e.g. namespace/repo-name).

public.ecr.aws/registry-alias/ shubhi\_ecs

10 out of 205 characters maximum (2 minimum). The name must start with a letter and can only contain lowercase letters, numbers, hyphens, underscores, periods and forward slashes.

**Note:** A default alias is associated with your public registry once your first public repository is created. The registry alias is displayed as a prefix to the repository name in the repository URI. A custom alias can be requested on the Registry settings page.

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

The screenshot shows the AWS ECR 'Repositories' page. It lists a single public repository named 'shubhi\_ecs' with a creation date of November 01, 2023, at 09:28:56 UTC+05:55. The 'Public' tab is selected.

Amazon ECR > Repositories

Private | **Public**

**Public repositories (1)**

**Repository name** ▲ **URI** | **Created at**

Repository name	URI	Created at
shubhi_ecs	public.ecr.aws/y3e6o5a2/shubhi_ecs	November 01, 2023, 09:28:56 (UTC+05:55)

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.

The screenshot shows the AWS ECR console with the URL [https://us-east-1.console.aws.amazon.com/ecr/repositories/public/027162208479/shubhi\\_ecs?region=us-east-1](https://us-east-1.console.aws.amazon.com/ecr/repositories/public/027162208479/shubhi_ecs?region=us-east-1). The left sidebar has a 'Images' section selected. The main area shows a table titled 'Images (0)' with a single row indicating 'No images'. At the top right of this table are buttons for 'View public listing', 'View push commands', and 'Edit'. The status bar at the bottom shows the date and time as 11/1/2023 9:29 AM.

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.

The screenshot shows a modal window titled 'Push commands for shubhi\_ecs'. It contains instructions for pushing a Docker image to the repository. Step 1: 'Retrieve an authentication token and authenticate your Docker client to your registry.' It includes a command: `aws ecr-public get-login-password --region us-east-1 | docker login --username AWS --password-stdin public.ecr.aws/y3e6o5a2`. Step 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 is already built.' It includes a command: `docker build -t shubhi_ecs .`. Step 3: 'After the build completes, tag your image so you can push the image to this repository.' It includes a command: `docker tag shubhi_ecs:latest public.ecr.aws/y3e6o5a2/shubhi_ecs:latest`. Step 4: 'Run the following command to push this image to your newly created AWS repository.' It includes a command: `docker push public.ecr.aws/y3e6o5a2/shubhi_ecs:latest`. A 'Close' button is at the bottom right.

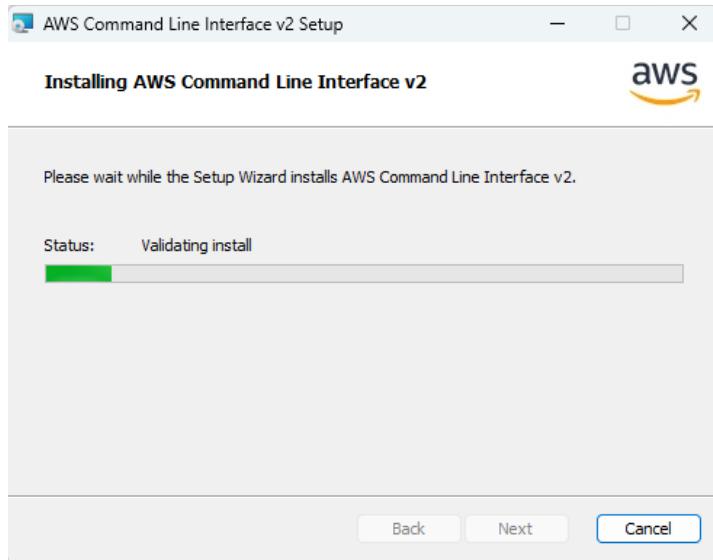
10) Before running these commands, you need to install AWS CLI and configure it if you haven't already: <https://aws.amazon.com/cli/>

The screenshot shows the AWS CLI landing page in a web browser. The URL in the address bar is [aws.amazon.com/cli/](https://aws.amazon.com/cli/). The page features the AWS logo and navigation links for Products, Solutions, Pricing, Documentation, Learn, Partner Network, AWS Marketplace, Customer Enablement, Events, Explore More, Contact Us, Support, English, My Account, and Sign In to the Console.

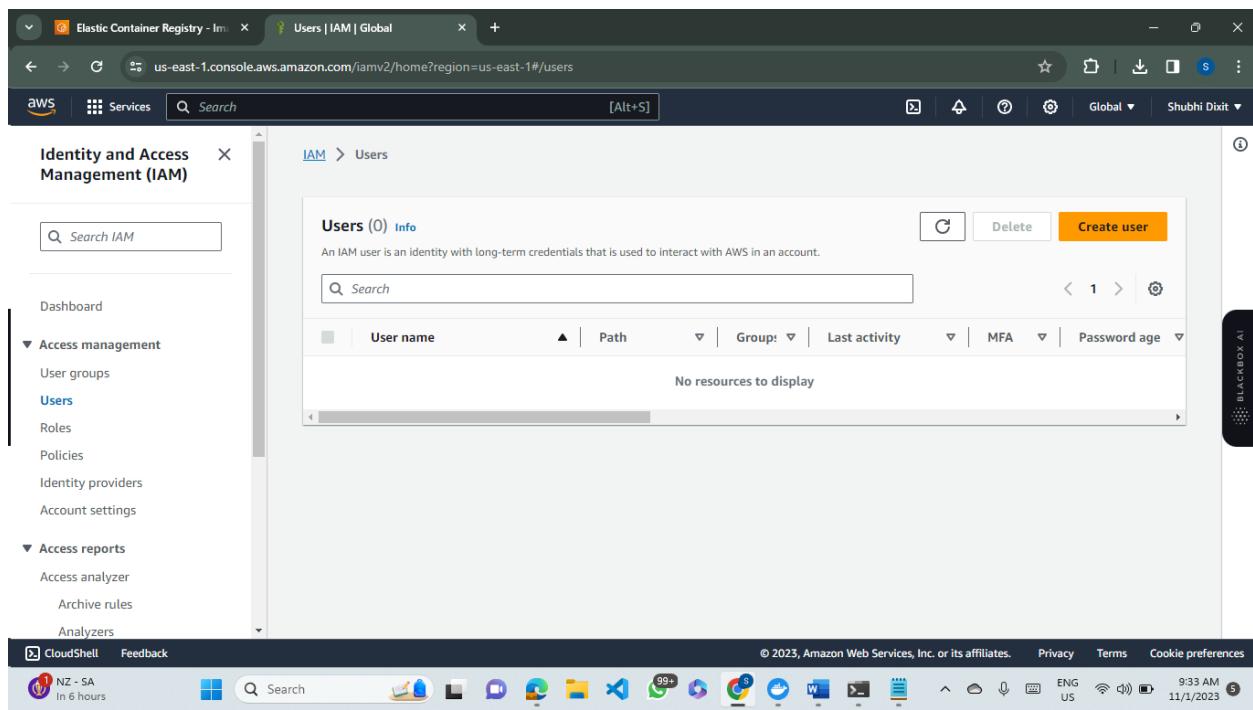
The main content area is titled "AWS Command Line Interface". It describes the AWS CLI as a unified tool to manage AWS services from the command line. It highlights several new features in version v2, such as improved installers, new configuration options like AWS IAM Identity Center, and various interactive features. Below this text are four icons: a large number 1, a document, a GitHub repository, and a cube.

Below the icons are links to "Getting Started", "AWS CLI Reference", "GitHub Project", and "Community Forum". To the right, there are sections for Windows, macOS, Linux, and Amazon Linux, each with download links. A "Release Notes" section provides a link to more information. At the bottom of the page is a Windows taskbar showing various open applications and system status.

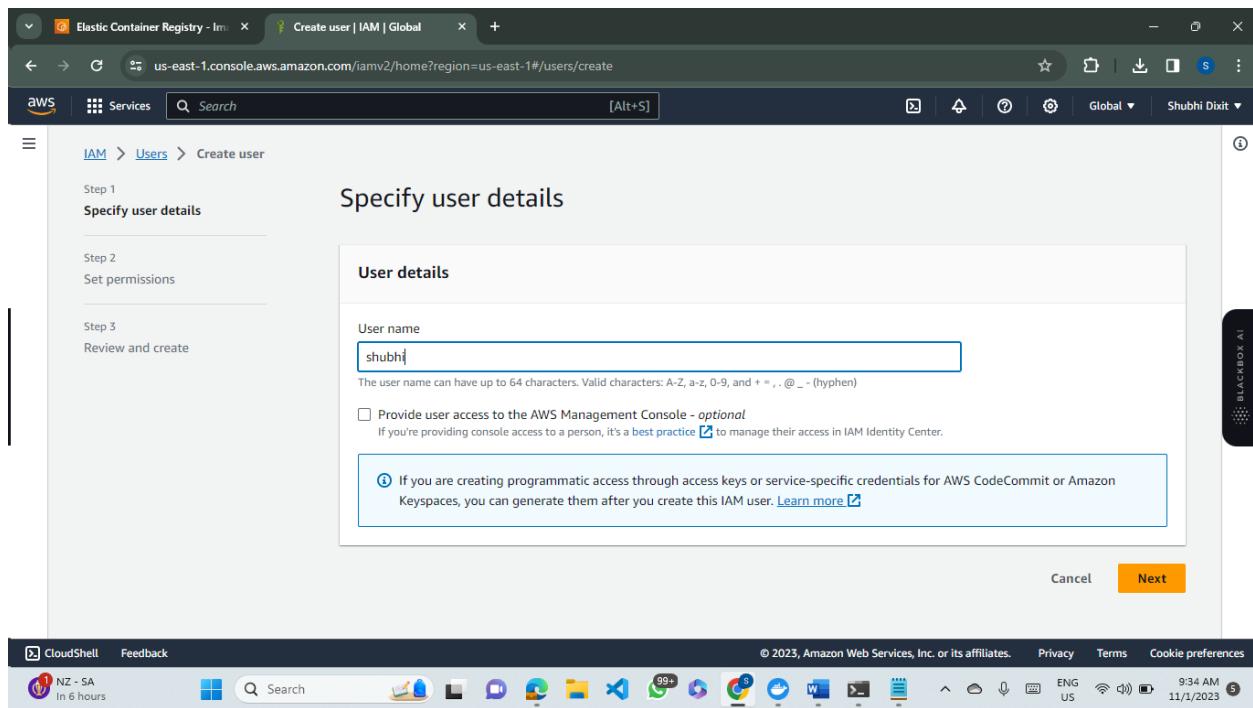
The screenshot shows the "AWS Command Line Interface v2 Setup" window. The title bar reads "AWS Command Line Interface v2 Setup". The main content area features the AWS logo and the heading "Welcome to the AWS Command Line Interface v2 Setup Wizard". Below this, a message states: "The Setup Wizard will install AWS Command Line Interface v2 on your computer. Click Next to continue or Cancel to exit the Setup Wizard." At the bottom of the window are three buttons: "Back", "Next", and "Cancel".



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



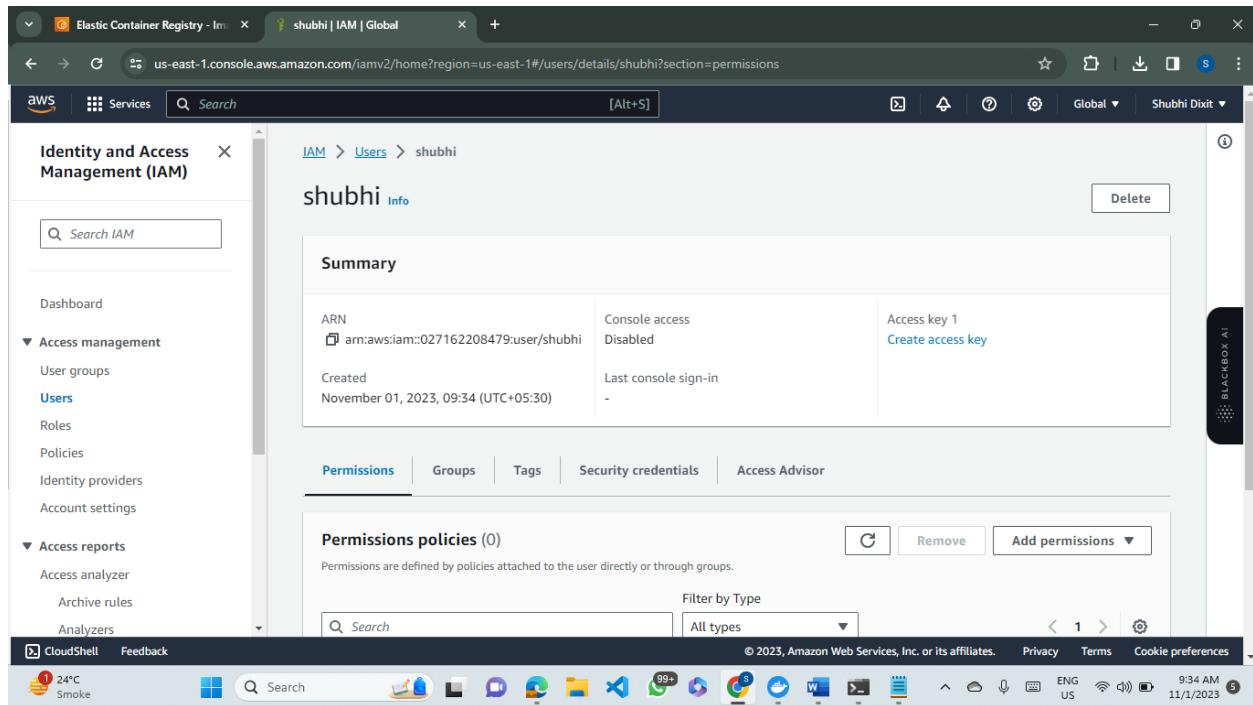
12) Specify the user details, add a user name and proceed with the following check boxes given in the image



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

14) Select the user.

15) Click on “Create access key”



## 16) Select “Other” and proceed

The screenshot shows the AWS IAM 'Create access key' wizard. In Step 2, under 'Use case', the 'Other' option is selected. A note below says: 'Your use case is not listed here.' A message at the bottom states: 'It's okay to use an access key for this use case, but follow the best practices:'.

## 17) Note these Access key and Secret access key.

The screenshot shows the 'Retrieve access keys' page. It displays the 'Access key' and 'Secret access key' fields. The 'Secret access key' field is masked with asterisks. A 'Show' link is available to view the full key. Below this, there is a section titled 'Access key best practices' with the following bullet points:

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

For more details about managing access keys, see the [best practices for managing AWS access keys](#).

Summary		
ARN arn:aws:iam::027162208479:user/shubhi	Console access Disabled	Access key 1  Never used. Created today.
Created November 01, 2023, 09:34 (UTC+05:30)	Last console sign-in -	Access key 2 <a href="#">Create access key</a>

Access key added

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

Policy name	Type	Attached via

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

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket"
      ]
    }
  ]
}
```

```

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

```

**Specify permissions** Info

Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.

Policy editor
Visual
JSON
Actions ▾

```

1 Version: "2012-10-17",
2   Statement: [
3     {
4       Sid: "VisualEditor0",
5       Effect: "Allow",
6       Action: [
7         "ecr-public:*",
8         "sts:GetServiceBearerToken"
9       ],
10      Resource: "*"
11    }
12  ]
13 }
14 }
```

**Edit statement**

Select a statement

Select an existing statement in the policy or add a new statement.

**+ Add new statement**

20) Name the Policy and click on create.

IAM > Users > shubhi > Create policy

Step 1  
[Specify permissions](#)

Step 2  
[Review and create](#)

**Review and create** Info

Review the permissions, specify details, and tags.

**Policy details**

**Policy name**  
Enter a meaningful name to identify this policy.

Maximum 128 characters. Use alphanumeric and '+,-,.,@,\_-' characters.

**Permissions defined in this policy** Info

Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it.

Search

Allow (2 of 383 services)

Show remaining 381 services

Screenshot of the AWS IAM Permissions Policies page. The navigation bar includes 'Permissions', 'Groups', 'Tags', 'Security credentials', and 'Access Advisor'. The main section shows 'Permissions policies (1)'. A table lists one policy: 'policy1' (Customer inline, Attached via [link]).

Policy created successfully.

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

Screenshot of the AWS IAM Add permissions page. The 'Attach policies directly' option is selected. Below, the 'Permissions policies' table shows several AWS managed policies, with 'AmazonEC2ContainerRegistryReadOnly' selected.

22) Make sure the permissions are attached.

The screenshot shows the 'Permissions' tab of the AWS IAM console. It lists two policies: 'AmazonEC2ContainerRegistryReadOnly' (AWS managed, Directly) and 'policy1' (Customer inline). There are buttons for 'Add permissions' and 'Remove'.

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

A terminal window showing the AWS configuration command. It asks for AWS Access Key ID and AWS Secret Access Key, both of which are redacted. Default region name is set to Global and default output format is json. The command is 'aws configure'.

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.'

A terminal window showing the command 'aws ecr-public get-login-password --region us-east-1 | docker login --username AWS --password-stdin public.ecr.aws/y3e6o5a2'. An error message follows: 'Error saving credentials: error storing credentials - err: exit status 1, out: `error storing credentials - err: exit status 1, out: `The stub received bad data.'`

### **Follow the following steps:**

Go to: 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.aws/y3e605a2
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@sha256:7b8098fbbdaa8d1c3ae2208b249e7adbe12376fe8d86ef2aae27ce1f13b7167
=> [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/y3e605a2/shubhi_ecs:latest
● PS C:\Users\admin\ecs> docker push public.ecr.aws/y3e605a2/shubhi_ecs:latest
The push refers to repository [public.ecr.aws/y3e605a2/shubhi_ecs]
7c8bd21646f4: Pushed
6b7ad43c9a9e: Pushed
cdae29f197b2: Pushed
5ca84525a215: Pushed
e0ba343b5cab: Pushed
2905795cb5b8: Pushed
ec983b166360: Pushed
latest: digest: sha256:1f326758e9874fcffae6061777d6ee52a807972fe7fe41a64c2944c617c153d7 size: 1785

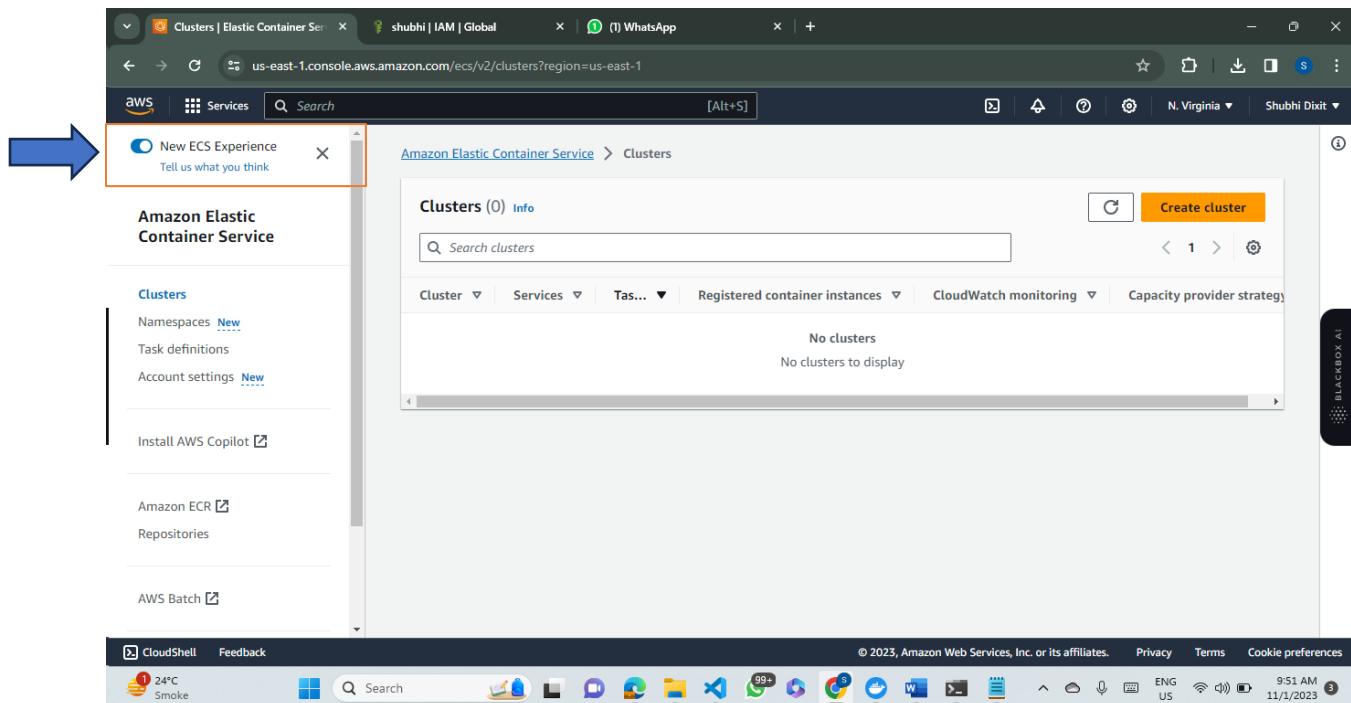
```

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

	Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest
<input type="checkbox"/>	latest	Image	November 01, 2023, 09:49:13 (UTC+05.5)	79.65	<input type="button" value="Copy URI"/>	<input type="text" value="sha256:1f326758e9874fc..."/>

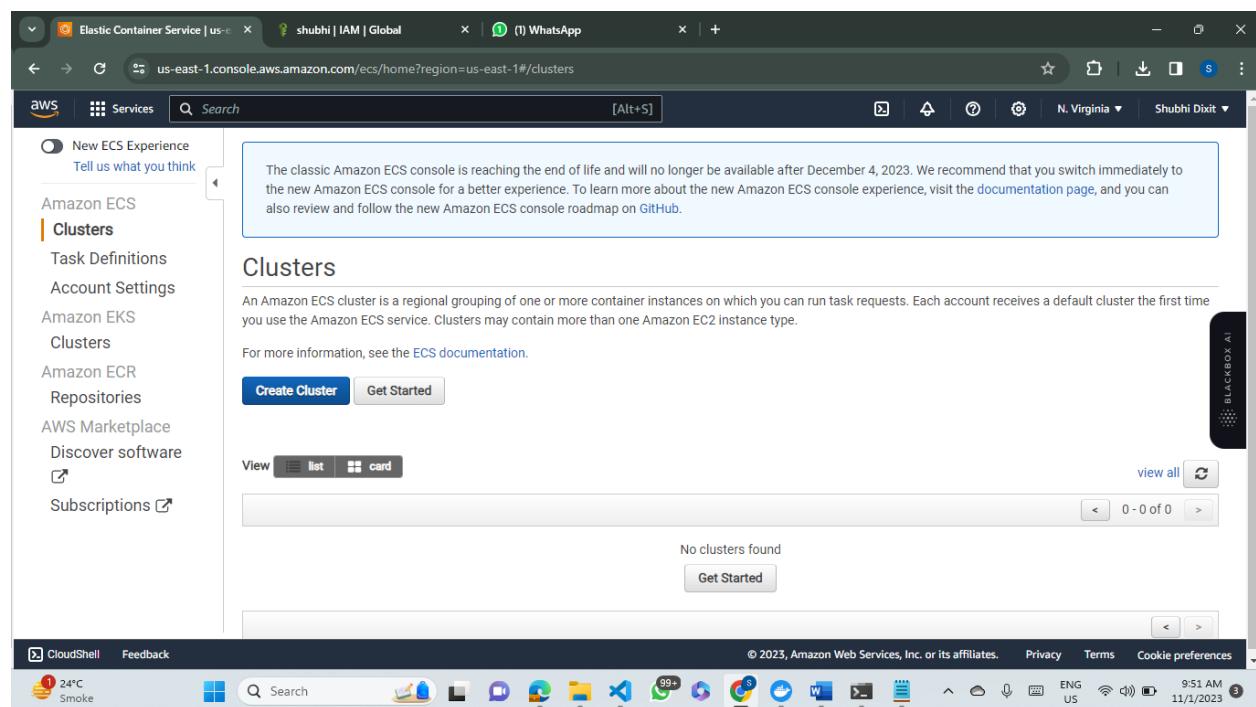
## Creating Cluster

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



The screenshot shows the AWS CloudShell interface. On the left, there is a sidebar with various AWS services listed: Amazon Elastic Container Service (selected), Clusters, Namespaces, Task definitions, Account settings, Install AWS Copilot, Amazon ECR, Repositories, and AWS Batch. A blue arrow points to the 'New ECS Experience' button at the top of the sidebar. The main content area shows the 'Clusters' page for the Amazon Elastic Container Service, which currently displays 'No clusters' and 'No clusters to display'. At the top right of the main content area, there is a 'Create cluster' button. The bottom of the screen shows the standard Windows taskbar with various pinned icons.

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



The screenshot shows the AWS CloudShell interface again. The sidebar now shows 'Amazon ECS' as the selected service, with options like Clusters, Task Definitions, Account Settings, Amazon EKS, Amazon ECR, AWS Marketplace, Discover software, and Subscriptions. A message box in the center of the screen informs users that the classic Amazon ECS console is reaching the end of life and will no longer be available after December 4, 2023. It encourages switching to the new Amazon ECS console for a better experience, providing links to documentation and GitHub. Below this message, the 'Clusters' section is visible, featuring a 'Create Cluster' button and a 'Get Started' button. The main content area also includes a 'View' dropdown with 'list' and 'card' options, a 'view all' link, and a 'Get Started' button. The bottom of the screen shows the standard Windows taskbar.

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

The screenshot shows the 'Create Cluster' wizard on the AWS Elastic Container Service console. The current step is 'Step 1: Select cluster template'. There are three options available:

- Networking only**:  
Resources to be created:
  - Cluster
  - VPC (optional)
  - Subnets (optional)

For use with either AWS Fargate (Windows/Linux) or with External instance capacity.
- EC2 Linux + Networking**:  
Resources to be created:
  - Cluster
  - VPC
  - Subnets
  - Auto Scaling group with Linux AMI
- EC2 Windows + Networking**:  
Resources to be created:
  - Cluster

4) Set the settings in the respective fields:

Instance configuration

Provisioning Model  On-Demand Instance  
With On-Demand Instances, you pay for compute capacity by the hour, with no long-term commitments or upfront payments.

Spot  
Amazon EC2 Spot Instances let you take advantage of unused EC2 capacity in the AWS cloud. Spot Instances are available at up to a 90% discount compared to On-Demand prices.  
[Learn more](#)

EC2 instance type\*     
 Manually enter desired instance type

Number of instances\*

EC2 AMI ID\*

Root EBS Volume Size (GB)

Key pair     
You will not be able to SSH into your EC2 instances without a key pair. You can create a new key pair in the [EC2 console](#).

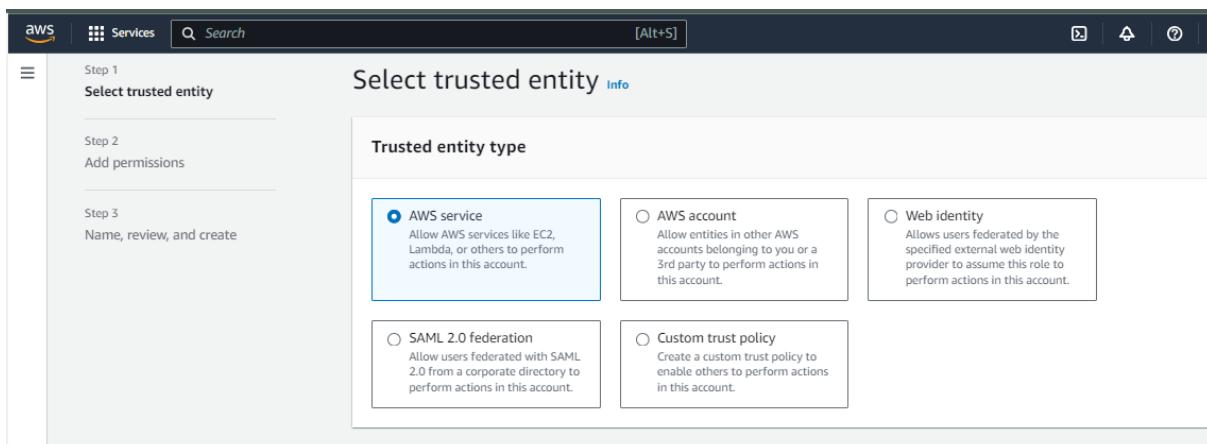
5) Select the default VPC and the subnets that you have. Leave the other settings default and click on create.



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:



**Use case**

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case

Elastic Container Service

Choose a use case for the specified service.

Use case

- Elastic Container Service
 

Allows ECS to create and manage AWS resources on your behalf.
- Elastic Container Service Autoscale
 

Allows Auto Scaling to access and update ECS services.
- Elastic Container Service Task
 

Allows ECS tasks to call AWS services on your behalf.
- EC2 Role for Elastic Container Service
 

Allows EC2 instances in an ECS cluster to access ECS.

**Cancel** **Next**

3) Add permissions and choose “AmazonECSTaskExecutionRolePolicy”

**Add permissions** Info

Permissions policies (1/896) Info

Choose one or more policies to attach to your new role.

Filter by Type

Policy name	Type	Description
<input type="checkbox"/> <a href="#">AmazonECS</a>	AWS managed	Provides administrative access to Amazo...
<input checked="" type="checkbox"/> <a href="#">AmazonECSTaskExecutio...</a>	AWS managed	Provides access to other AWS service reso...

**Set permissions boundary - optional**

**Cancel** **Previous** **Next**

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

Name, review, and create

Role details

Role name  
Enter a meaningful name to identify this role.

Maximum 64 characters. Use alphanumeric and '+=\_,@-' characters.

Description  
Add a short explanation for this role.

Maximum 1000 characters. Use alphanumeric and '+=\_,@-' characters.

## Task Definition

- 1) Go to ECS again, and go to “Task definitions”.
- 2) Click on “Create new task definition”.

The screenshot shows the 'Create new Task Definition' wizard on the AWS ECS console. The current step is 'Step 1: Select launch type compatibility'. It displays three options:

- FARGATE**: Price based on task size. Requires network mode awsvpc. AWS-managed infrastructure, no Amazon EC2 instances to manage.
- EC2**: Price based on resource usage. Multiple network modes available. Self-managed infrastructure using Amazon EC2 instances.
- EXTERNAL**: Price based on instance-hours and additional charges for.

The FARGATE option is highlighted with a blue border. The browser's address bar shows the URL: us-east-1.console.aws.amazon.com/ecs/home?region=us-east-1#/taskDefinitions/create. The status bar at the bottom right indicates it's 10:00 AM on November 1, 2023, in N. Virginia.

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

The screenshot shows the 'Create new Task Definition' wizard on the AWS ECS console, now at Step 2: Configure task and container definitions. The task definition name is set to 'task1'. The task role is set to 'Role\_ecs'. The browser's address bar shows the URL: us-east-1.console.aws.amazon.com/ecs/home?region=us-east-1#/taskDefinitions/create. The status bar at the bottom right indicates it's 10:02 AM on November 1, 2023, in N. Virginia.

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

The screenshot shows the 'Task size' configuration section of the AWS ECS task definition creation interface. It includes fields for 'Task memory (MiB)' (set to 516) and 'Task CPU (unit)' (set to 256). Below these are two progress bars: 'Task CPU maximum allocation for containers' (516 shared of 516 MiB) and 'Task memory maximum allocation for container memory reservation' (256 shared of 256 CPU units). A 'Container definitions' section with a 'Add container' button is also visible.

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.

The screenshot shows the 'Add container' dialog box overlaid on the main task definition creation page. In the 'Standard' configuration, a container named 'shubhi' is defined with the image 'public.ecr.aws/y3e6o5a/shubhi\_ecs:latest'. Other settings include a hard memory limit of 128 MiB. The 'Port mappings' section at the bottom is partially visible, showing a mapping from Host port 8080 to Container port 80.

shubhi\_ecs

[View public listing](#)[View push commands](#)[Edit](#)**Images (1)** Search artifacts

<input type="checkbox"/>	Image tag	Artifact type	Pushed at	Size (MB)	Digest
<input type="checkbox"/>	latest	Image	November 01, 2023, 09:49:13 (UTC+05:5)	79.65	58e9874fc...

Image URI copied

[Copy URI](#)

Copy Image URL from here.

Port mappings	Host port	Container port	Protocol	
	<input type="text"/> 8080	<input type="text"/> 80	<input type="button" value="tcp"/>	

[+ Add port mapping](#)

▼ Advanced container configuration

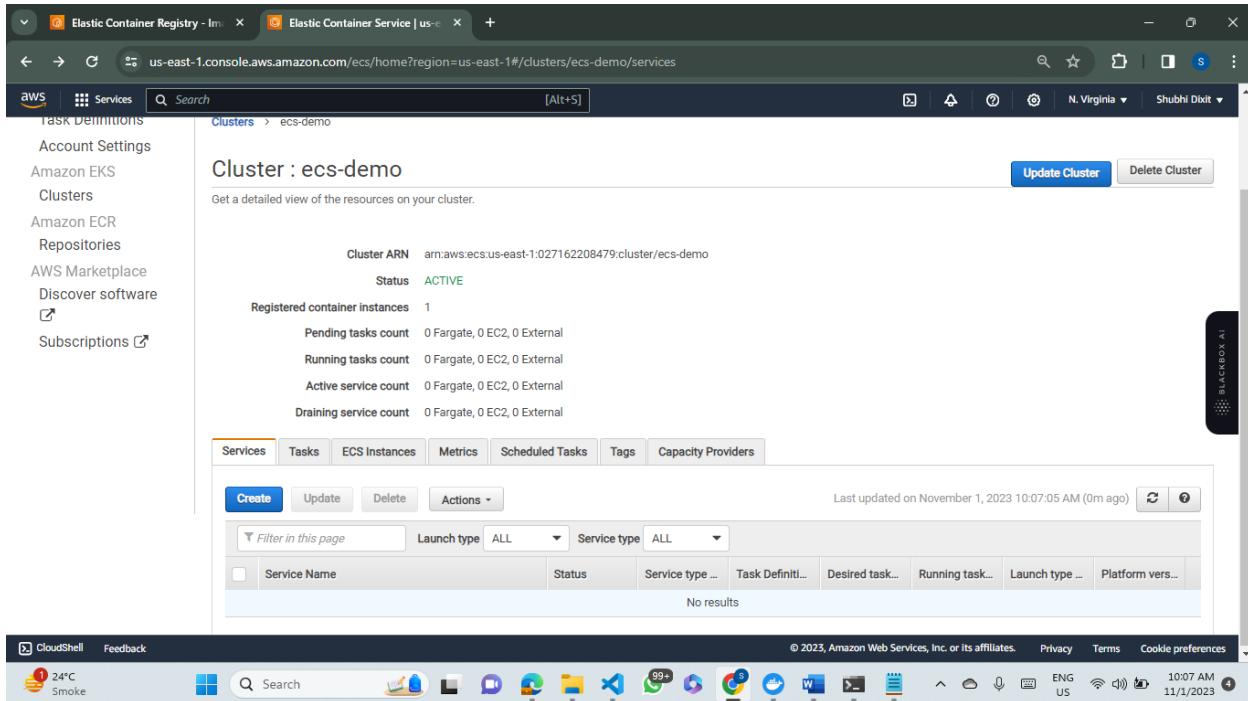
**Container definitions**[Add container](#)

Container Name ...	Image	Hard/Soft memo...	CPU Units ...	GPU	Essential ...	
shubhi	public.ecr.aws/y3...	-/-			true	

6) Task definition is created successfully.

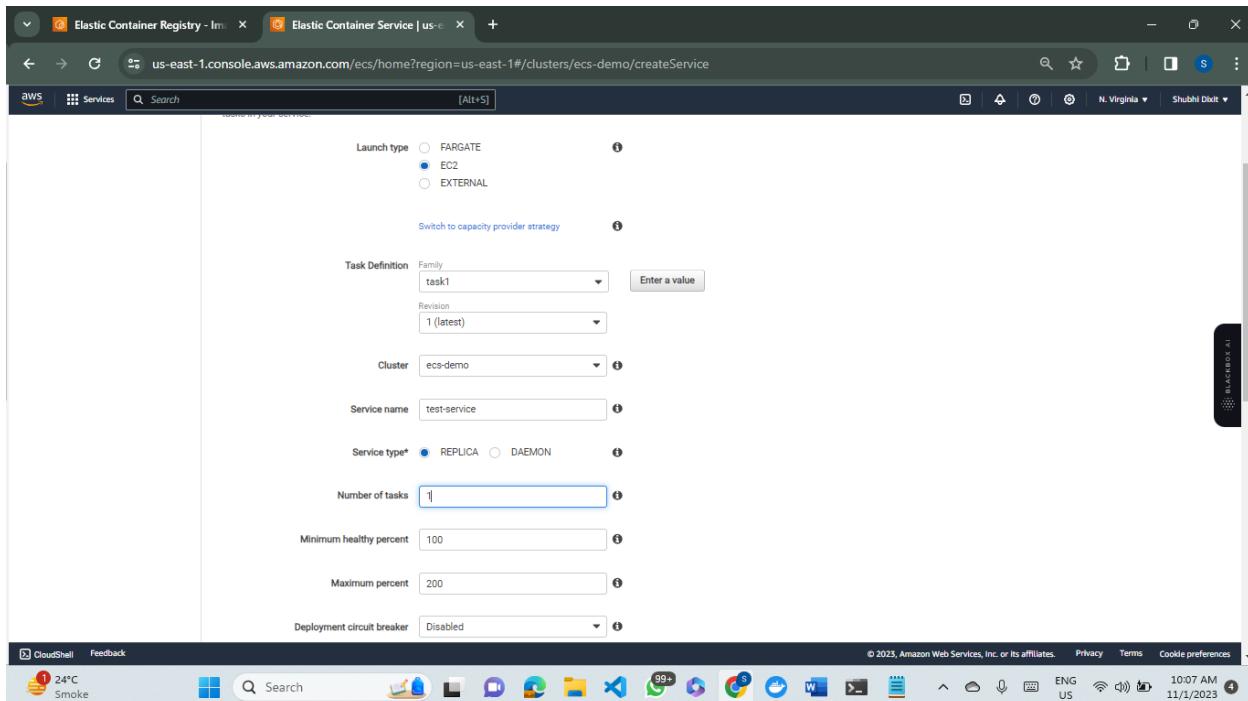
## Service Deployment

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



The screenshot shows the AWS ECS console interface. On the left, there's a sidebar with links like Task Definitions, Account Settings, Amazon EKS, Clusters, Amazon ECR, Repositories, AWS Marketplace, Discover software, and Subscriptions. The main content area is titled 'Cluster : ecs-demo' and displays cluster statistics: Cluster ARN (arn:aws:ecs:us-east-1:027162208479:cluster/ecs-demo), Status (ACTIVE), Registered container instances (1), Pending tasks count (0 Fargate, 0 EC2, 0 External), Running tasks count (0 Fargate, 0 EC2, 0 External), Active service count (0 Fargate, 0 EC2, 0 External), and Draining service count (0 Fargate, 0 EC2, 0 External). Below this is a table with columns for Service Name, Status, Service type, Task Definition, Desired task count, Running task count, Launch type, and Platform version. A message 'No results' is shown. At the bottom of the main content area, there are buttons for Create, Update, Delete, and Actions.

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



The screenshot shows the 'createService' wizard in the AWS ECS console. The 'Launch type' is set to 'EC2'. The 'Task Definition' dropdown shows 'task1'. The 'Service name' is 'test-service'. The 'Service type\*' is 'REPLICA'. The 'Number of tasks' is 1. The 'Minimum healthy percent' is 100, and the 'Maximum percent' is 200. The 'Deployment circuit breaker' is set to 'Disabled'. At the bottom, there are 'Next Step' and 'Cancel' buttons.

### 3) Service is successfully created.

The screenshot shows the AWS ECS console in the N. Virginia region. The main heading is "Launch Status" with the sub-section "ECS Service status - 1 of 1 completed". Below this, under "Create Service", it says "Create service: test-service". A green box indicates "Service created" with the message "Service created. Tasks will start momentarily. View: test-service". Further down, there's a section titled "Additional integrations you can connect to your ECS service" with a "Code Pipeline" link. At the bottom right, there are "Back" and "View Service" buttons.

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

The screenshot shows the AWS EC2 Instances page in the N. Virginia region. The left sidebar lists various EC2 services like CloudShell, Feedback, Events, Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, and Images. The main area displays "Instances (1/1)" with a table showing one instance: "ECS Instance - ... i-0d7d54c6ef4dbc3c6" (Running, t2.micro, 2/2 checks passed, us-east-1). Below the table, a detailed view for "Instance: i-0d7d54c6ef4dbc3c6 (ECS Instance - EC2ContainerService-ecs-demo)" is shown with tabs for Details, Security, Networking, Storage, Status checks, Monitoring, and Tags. The "Details" tab shows fields like Instance ID, Public IPv4 address (100.27.13.255), Private IPv4 addresses (172.31.51.153), and Public IPv4 DNS (ec2-100-27-13-255.compute-1.amazonaws.com). At the bottom, there are "Back" and "View Service" buttons.

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

The screenshot shows the AWS EC2 Security Groups page. A single security group is listed:

Security group name	Security group ID	Description	VPC ID
EC2ContainerService-ecs-demo-EcsSecurityGroup-95ZTDYHAQX31	sg-07f45fac5e4db6ffa	ECS Allowed Ports	vpc-059351c22df72c7c1

Under the 'Inbound rules' tab, there is one rule:

Name	Security group rule ID	IP version	Type	Protocol
-	sgr-0daee65d5c580881d	IPv4	HTTP	TCP

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

The screenshot shows the 'Edit inbound rules' page. There are two rules listed:

Security group rule ID	Type	Protocol	Port range	Source	Description
sgr-0daee65d5c580881d	HTTP	TCP	80	Cus... 0.0.0.0/0	Optional
-	Custom TCP	TCP	8080	Any... 0.0.0.0/0	Optional

A warning message at the bottom states: "⚠️ Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only."

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

The screenshot shows the AWS CloudShell interface with the following details:

**Task Details:**

- EC2 Instance ID: i-0d7d54c6ef4dbc3c6
- Launch type: EC2
- Task definition: task1:1
- Group: service:test-service
- Task role: Role\_ecs
- Last status: RUNNING
- Desired status: RUNNING
- Created at: 2023-11-01 10:08:20 +0530
- Started at: 2023-11-01 10:08:32 +0530

**Network:**

- Network mode: bridge

**Containers:**

Name	Container Runtime ID	Status	Image	Image Digest	CP...	Ha...	Es...	Re...
shubhi	8019edd2374ca461c3...	RU...	public.ecr.aws/y3e605a2/shubhi_ec...	sha256:1f326758e9874fcffae60617...	-	-/-	true	3f2...

8) Open the container details shown below

The screenshot shows the AWS CloudShell interface with the following details:

**Containers:**

Name	Container Runtime ID	Status	Image	Image Digest	CP...	Ha...	Es...	Re...
shubhi	8019edd2374ca461c3...	RU...	public.ecr.aws/y3e605a2/shubhi_ec...	sha256:1f326758e9874fcffae60617...	-	-/-	true	3f2...

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

The screenshot shows the AWS CloudWatch Metrics Insights interface. A search query is displayed:

```
MetricsFilterContainerName:shubhi AND MetricsFilterStatus:RUNNING
```

The results show a single metric data source named "shubhi". The data table includes columns for Metric Name, Value, and Unit. One row is shown:

Metric Name	Value	Unit
task_id	2e6f6b8478cf4c6bb2f360a6b21829df	

Below the table, there is a section titled "Container Details" which includes a table for "Network bindings". The table has columns for Host Port, Container Port, Protocol, and External Link. One entry is present:

Host Port	Container Port	Protocol	External Link
8080	80	tcp	100.27.13.255:8080

Other sections visible include "Environment Variables - not configured", "Environment Files - not configured", "Docker labels - not configured", "Extra hosts - not configured", and "Mount Points - not configured".

Paste in local browser.

The screenshot shows a local browser window with the address bar displaying "Not secure 100.27.13.255:8080". The page content is as follows:

This is the body of index page.

## Multiple tasks on EC2 instances

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

The screenshot shows the AWS CloudShell interface with the following details:

- Clusters**: A sidebar menu item.
- Amazon ECR**: A sidebar menu item.
- Repositories**: A sidebar menu item.
- AWS Marketplace**: A sidebar menu item.
- Discover software**: A sidebar menu item.
- Subscriptions**: A sidebar menu item.
- ECS Services**: The main content area, showing a detailed view of the resources in the cluster.
- Cluster ARN**: arn:aws:ecs:us-east-1:027162208479:cluster/ecs-demo
- Status**: ACTIVE
- Registered container instances**: 1
- Pending tasks count**: 0 Fargate, 0 EC2, 0 External
- Running tasks count**: 0 Fargate, 1 EC2, 0 External
- Active service count**: 0 Fargate, 1 EC2, 0 External
- Draining service count**: 0 Fargate, 0 EC2, 0 External
- Services Tab**: The active tab in the navigation bar.
- Create**, **Update**, **Delete**, **Actions**: Buttons in the top right of the table.
- Last updated on November 1, 2023 10:16:05 AM (0m ago)**: Last update timestamp.
- Filter in this page**: A search/filter bar.
- Launch type**: ALL.
- Service type**: ALL.
- 1 selected**: Number of selected items.
- test-service**: A table row showing the service details.
- Status**: ACTIVE.
- Service type**: REPLICA.
- Task Definition**: task1:1.
- Desired tasks**: 1.
- Running tasks**: 1.
- Launch type**: EC2.
- Platform**: -.

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

The screenshot shows the AWS CloudShell interface with the following details:

- CloudShell**: A sidebar menu item.
- Feedback**: A sidebar menu item.
- Elastic Container Registry - Im**: A tab in the browser.
- Elastic Container Service | us-e**: A tab in the browser.
- EC2 | us-east-1**: A tab in the browser.
- us-east-1.console.aws.amazon.com/ecs/home?region=us-east-1#clusters/ecs-demo/services/test-service/update**: The URL in the address bar.
- Force new deployment**: A checkbox.
- Cluster**: ecs-demo.
- Service name**: test-service.
- Service type\***: REPLICA.
- Number of tasks**: 2.
- Minimum healthy percent**: 100.
- Maximum percent**: 200.
- Deployment circuit breaker**: Disabled.
- CloudShell**: A sidebar menu item.
- Feedback**: A sidebar menu item.
- Search**: A search bar.
- Icons**: A row of small application icons.
- © 2023, Amazon Web Services, Inc. or its affiliates.**, **Privacy**, **Terms**, **Cookie preferences**: Footer links.
- ENG US**: Language and region settings.
- 10:16 AM 11/1/2023**: Date and time.

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

The screenshot shows the AWS ECS console interface. The top navigation bar includes tabs for 'Elastic Container Registry - Images', 'Elastic Container Service | us-east-1', and 'EC2 | us-east-1'. The main content area is titled 'Service type' with 'REPLICA' selected. It shows 'Number of tasks' set to 2, with 'Minimum healthy percent' at 100 and 'Maximum percent' at 200. The 'Deployment circuit breaker' is set to 'Disabled'. Below this, there are sections for 'Configure network' and 'Set Auto Scaling (optional)', both of which are currently 'not configured'. At the bottom right, there are 'Cancel', 'Previous', and a large blue 'Update Service' button. The status bar at the bottom indicates the user is in the N. Virginia region and has 10:16 AM on 11/1/2023.

Service type REPLICA

Number of tasks 2

Minimum healthy percent 100

Maximum percent 200

Deployment circuit breaker Disabled

Configure network Edit

not configured

Set Auto Scaling (optional) Edit

not configured

Cancel Previous Update Service

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

PAK - BAN PAK win 10:16 AM 11/1/2023

ECS Service status - 1 of 1 completed

Launch Status

Configure Task Networking

Service Auto Scaling

Update Service

Update service: test-service

Service updated  
Service updated successfully. View: test-service

Back View Service

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

Service : test-service

Event Id	Event Time	Message
284ad054-b086-4688-b6e4-b5224b7edff2	2023-11-01 10:16:56 +0530	service test-service was unable to place a task because no container instance met all of its requirements. The closest matching container-instance 55fc8750f8bd4cb48be586804e43caf9 has insufficient memory available. For more information, see the Troubleshooting section.

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.

Edit container

Port mappings

Host port	Container port	Protocol
<input type="text"/>	80	tcp

+ Add port mapping

Advanced container configuration

HEALTHCHECK

Command: CMD-SHELL, curl -f http://localhost/ || exit 1

Interval: [ ] second(s)

Timeout: [ ] second(s)

\* Required

Cancel Update

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

The screenshot shows the AWS Task Definitions console. A green success message box at the top says "Created new revision of Task Definition task1:4 successfully". Below it, the URL "Task Definitions > task1 > 4" is shown. The main title is "Task Definition: task1:4". A sub-header says "View detailed information for your task definition. To modify the task definition, you need to create a new revision and then make the required changes to the task definition". There are three tabs at the top: "Create new revision" (blue), "Actions" (grey), and "Builder" (selected). Below are tabs for "JSON" and "Tags". A search bar shows "Task definition name task1".

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

The screenshot shows the "Configure service" page. It has a "Task Definition" section with a dropdown set to "task1" and a "Revision" dropdown set to "4 (latest)". Below is a "Launch type" field set to "EC2". There are two buttons: "Switch to capacity provider strategy" and "Force new deployment" with an unchecked checkbox. Each button has an info icon.

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

The screenshot shows the AWS ECS Service status page. At the top, there are navigation links for 'Services' and a search bar. On the right, there are icons for notifications, help, and user information ('N. Virginia' and 'Shubhi Dixit'). Below the header, the 'Launch Status' section indicates 'ECS Service status - 1 of 1 completed'. Underneath, there are links for 'Configure Task Networking', 'Service Auto Scaling', and 'Update Service'. A sub-section titled 'Update service: test-service' shows a green success message: 'Service updated' and 'Service updated successfully. View: test-service'. At the bottom right, there are 'Back' and 'View Service' buttons.

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

The screenshot shows the AWS Security Group rule configuration page. It features a search bar at the top with filters for 'All TCP', 'TCP', '0 - 65535', 'Any...', and a search icon. Below the search bar is a row of buttons: 'Delete' (with an 'X'), 'Add rule' (highlighted in blue), and a warning message. The warning message states: '⚠ Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.' At the bottom, there are 'Cancel', 'Preview changes', and 'Save rules' buttons.

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.

The screenshot shows the 'Basic configuration' step of the 'Create Application Load Balancer' wizard. It includes fields for 'Load balancer name' (shubhi-alb), 'Scheme' (set to 'Internet-facing'), and 'Mappings' (which lists five Availability Zones: us-east-1a, us-east-1b, us-east-1c, us-east-1d, and us-east-1e, each associated with a specific subnet). A note at the bottom states: 'Select at least two Availability Zones and one subnet per zone. The load balancer routes traffic to targets in these Availability Zones only. Availability Zones that are not supported by the load balancer or the VPC are not available for selection.'

2) Select all the AZs in the mapping field.

The screenshot shows the 'Mappings' section of the 'Create application load balance' wizard in the AWS CloudShell. It lists five Availability Zones (us-east-1a, us-east-1b, us-east-1c, us-east-1d, us-east-1e) each with a selected checkbox and a dropdown menu showing a subnet. The dropdown for us-east-1a contains 'subnet-0e87ff2572d3e21cd'. The dropdown for us-east-1b contains 'subnet-03bc3f7a7f0438571'. The dropdown for us-east-1c contains 'subnet-053a884bbe82875ad'. The dropdown for us-east-1d contains 'subnet-02983239f47a863d7'. The dropdown for us-east-1e contains 'subnet-02983239f47a863d7'. Below the dropdowns, there are 'IPv4 address' and 'Assigned by AWS' sections.

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

## **Target Group Creation**

- 1) Click on instances.

**Specify group details**

Your load balancer routes requests to the targets in a target group and performs health checks on the targets.

**Basic configuration**  
Settings in this section can't be changed after the target group is created.

**Choose a target type**

Instances

- Supports load balancing to instances within a specific VPC.
- Facilitates the use of [Amazon EC2 Auto Scaling](#) to manage and scale your EC2 capacity.

IP addresses

- Supports load balancing to VPC and on-premises resources.
- Facilitates routing to multiple IP addresses and network interfaces on the same instance.
- Offers flexibility with microservice based architectures, simplifying inter-application communication.
- Supports IPv6 targets, enabling end-to-end IPv6 communication, and IPv4-to-IPv6 NAT.

Lambda function

- Facilitates routing to a single Lambda function.
- Accessible to Application Load Balancers only.

- 2) Give it a name

**Target group name**

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

**Protocol : Port**

1-65535

**IP address type**  
Only targets with the indicated IP address type can be registered to this target group.

IPv4  
Each instance has a default network interface (eth0) that is assigned the primary private IPv4 address. The instance's primary private IPv4 address is the one that will be applied to the target.

IPv6  
Each instance you register must have an assigned primary IPv6 address. This is configured on the instance's default network interface (eth0). [Learn more](#)

**VPC**  
Select the VPC with the instances that you want to include in the target group. Only VPCs that support the IP address type selected above are available in this list.

- 3) Add the container instance which we created earlier.

The screenshot shows the 'Register targets' step of creating a target group. On the left, there's a navigation bar with 'EC2 > Target groups > Create target group'. Below it, 'Step 1 Specify group details' is completed. In 'Step 2 Register targets', the user has selected an instance from the 'Available instances' list. The instance 'i-0d7d54c6ef4dbc3c6' is listed with its name, state (Running), and security group (EC2ContainerService-ecs-de). The port '80' is selected for routing traffic. A note at the bottom says 'Ports for the selected instances Ports for routing traffic to the selected instances.'

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

The screenshot shows the 'Target groups (1)' list page. It displays one target group named 'tg' with ARN 'arn:aws:elasticloadbalancing:...', port '80', protocol 'HTTP', and target type 'Instance'. The 'Actions' dropdown and 'Create target group' button are visible at the top right.

Add this target group to your load balancer.

## Listeners and routing [Info](#)

A listener is a process that checks for connection requests using the port and protocol you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets.

▼ Listener HTTP:80

Protocol: HTTP Port: 80 Default action: [Info](#)

Forward to: tg Target type: Instance, IPv4

HTTP [Remove](#) [Edit](#)

[Create target group](#)

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

EC2 > Load balancers

Load balancers (1)

Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

Filter load balancers Actions [Create load balancer](#)

Name	DNS name	State	VPC ID	Availability Zones
shubhi-alb	shubhi-alb-1898633888.u...	Provisioning..	vpc-059351c22df72c7c1	6 Availability Zones

us-east-1.console.aws.amazon.com/ecs/home?region=us-east-1#clusters/ecs-demo/createService

Step 2: Configure network

Step 3: Set Auto Scaling (optional)

Step 4: Review

A service lets you specify how many copies of your task definition to run and maintain in a cluster. You can optionally use an Elastic Load Balancing load balancer to distribute incoming traffic to containers in your service. Amazon ECS maintains that number of tasks and coordinates task scheduling with the load balancer. You can also optionally use Service Auto Scaling to adjust the number of tasks in your service.

Launch type:  EC2  FARGATE  EXTERNAL

Switch to capacity provider strategy

Task Definition: Family: task1 Revision: 4

Cluster: ecs-demo

Service name: alb-demo

CloudShell Feedback © 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 25°C Smoke 11:04 AM 11/1/2023 ENG US

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

Service name  ⓘ

Service type\*  REPLICA  DAEMON ⓘ

Number of tasks  ⓘ

Minimum healthy percent  ⓘ

Maximum percent  ⓘ

Deployment circuit breaker  ⓘ

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

Service IAM role  ⓘ

Load balancer name  ⓘ

Container to load balance

shubhi : 80 Remove

Production listener port\*  ⓘ

Production listener protocol\* HTTP

Target group name  ⓘ  ⓘ

Target group protocol  ⓘ

Target type instance ⓘ

Path pattern  Evaluation order

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

The screenshot shows the AWS CloudWatch Logs interface with three log entries:

- Target Group created**: Target Group created. Waiting to create listener/rule. View: [ecs-ecs-de-alb-demo](#)
- Rule created**: Rule created. Waiting to create service. View in load balancer: [shubhi-alb](#)
- Service created**: Service created. Tasks will start momentarily. View: [alb-demo](#)

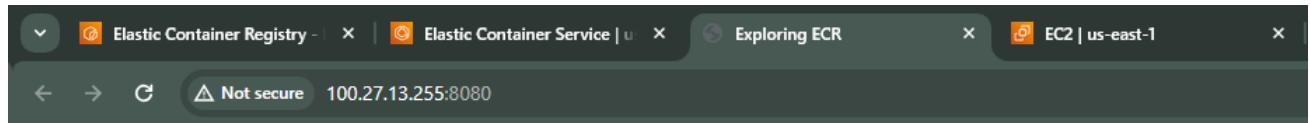
## 8) Now access the web page through load balancer's DNS

The screenshot shows the AWS Load Balancers console with the following details:

- Load balancers (1/1)**: shubhi-alb (Active, vpc-059351c22df7c7c1, 6 Availability Zones)
- Load balancer: shubhi-alb**:
  - Load balancer ARN: arn:aws:elasticloadbalancing:us-east-1:027162208479:loadbalancer/app/shubhi-alb/03eff22052d92fc0
  - DNS name copied: shubhi-alb-1898633888.us-east-1.elb.amazonaws.com (A Record)

**It is accessible.**

Also, you can access through [external link](#)



This is the body of index page.