

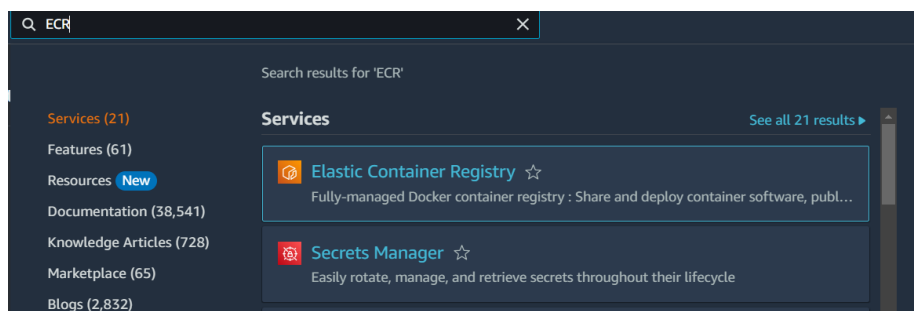
Objective: Create a GitHub Actions workflow that automates the building and deployment of a Node.js application to Amazon ECS using Amazon ECR.

Tasks:

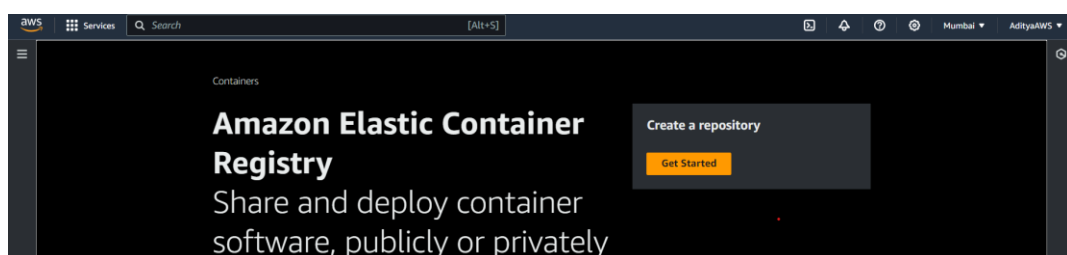
1. Set up GitHub Actions Workflow
2. Define Environment Variables
3. Build and Push Docker Image
4. Deploy to Amazon ECS
5. Complete Workflow

Steps:

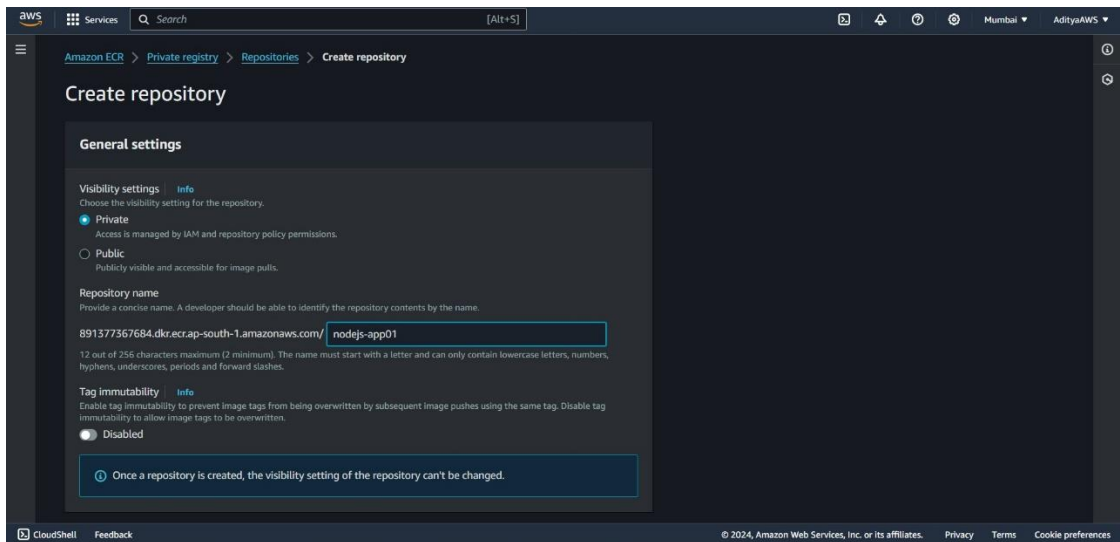
1. Create a ECR (Elastic Container Registry). In AWS console, search ECR.



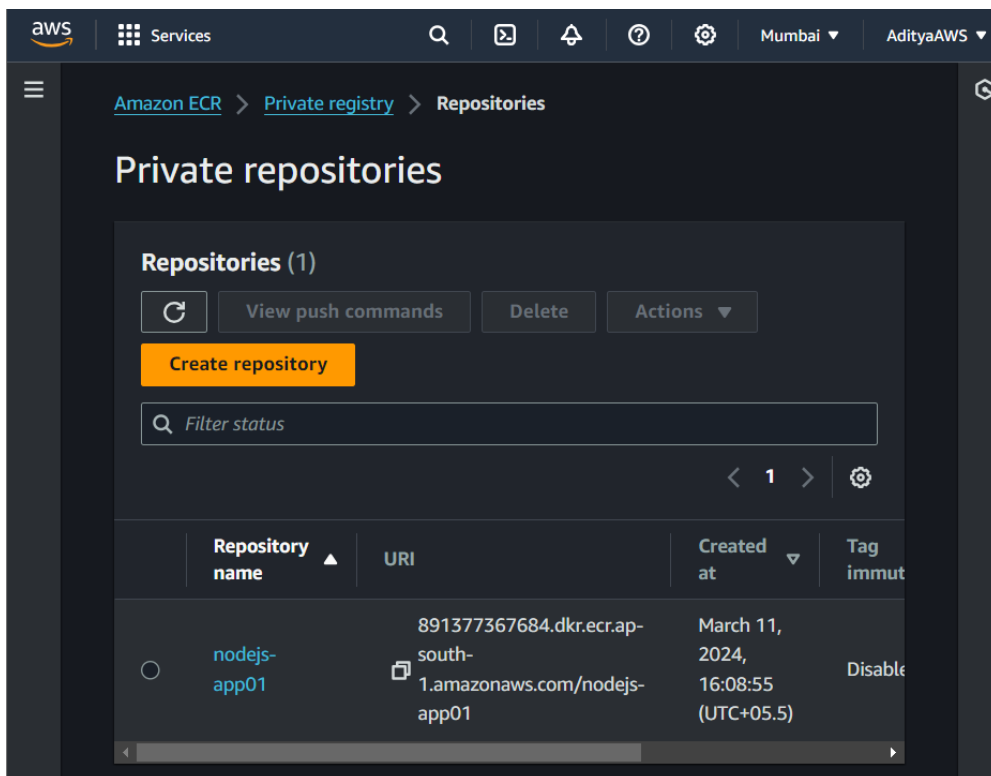
2. Click on “Get Started”



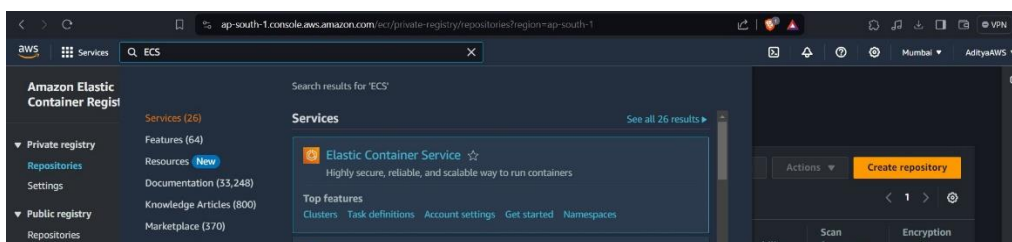
3. Now, create a repository, in visibility settings select private and name the repository. Scroll down and click on “Create repository”.



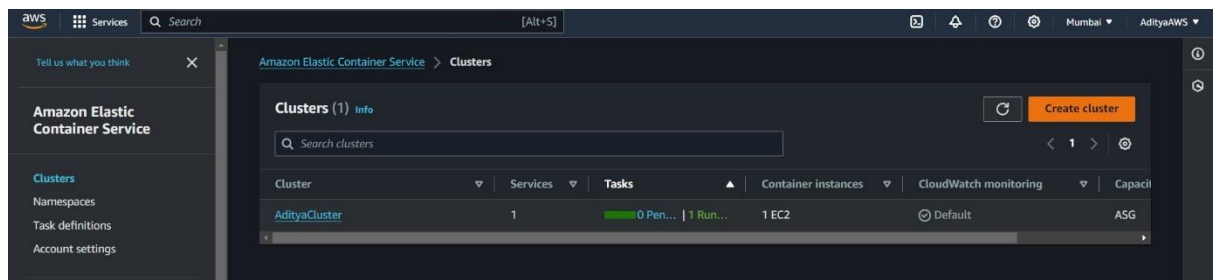
Repository created successfully



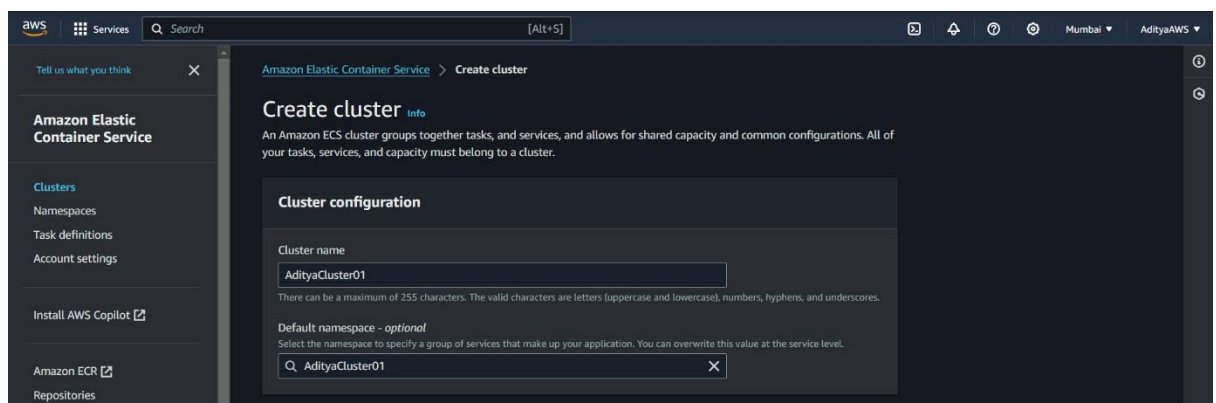
4. Setting up ECS: Search ECS



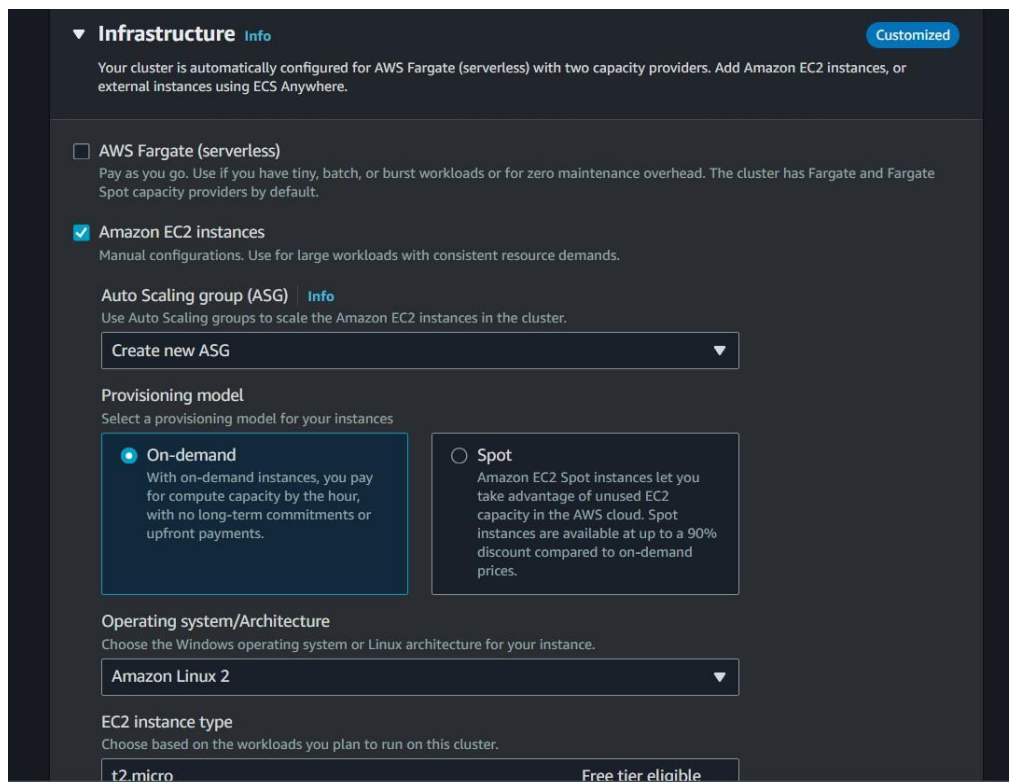
5. Create a cluster by clicking on “Create cluster”



6. Now, name the cluster.



7. In infrastructure select Amazon EC2 instances, Create new ASG with provisioning model “On-demand”, select Amazon Linux 2 in operating system, instance type: t2.micro.



8. Leave these sections as it is

Desired capacity
Specify the number of instances to launch in your cluster.

Minimum Maximum

SSH Key pair
If you do not specify a key pair, you can't connect to the instances via SSH unless you choose an AMI that is configured to allow users another way to log in.

Root EBS volume size
You can increase the size of the root EBS volume to allow for greater image and container storage.

Size (GiB)

☐ External instances using ECS Anywhere
Manual configurations. Use to add data center compute.

▼ Network settings for Amazon EC2 instances [Info](#)
By default Amazon EC2 instances are launched in the default subnets for your default VPC. To use the non-default VPC, specify the VPC and subnets.

VPC
Use a VPC with public and private subnets. By default, VPCs are created for your AWS account. To create a new VPC, go to the [VPC Console](#).

9. Move forward with default subnet and choose the security group SG-nodejs-app also enable the auto assign public IP and then click on create.

Subnets
Select the subnets where your tasks run. We recommend that you use three subnets for production.

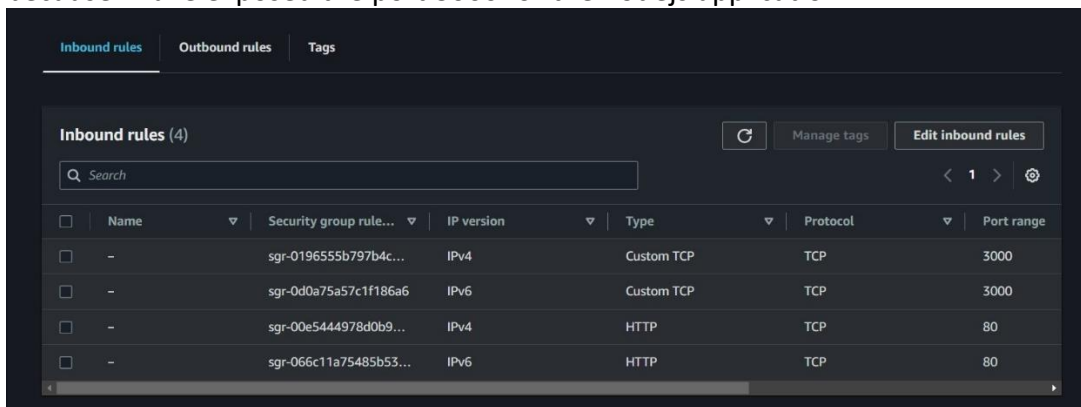
Security group [Info](#)
Choose an existing security group or create a new security group.

☒ Use an existing security group
☐ Create a new security group

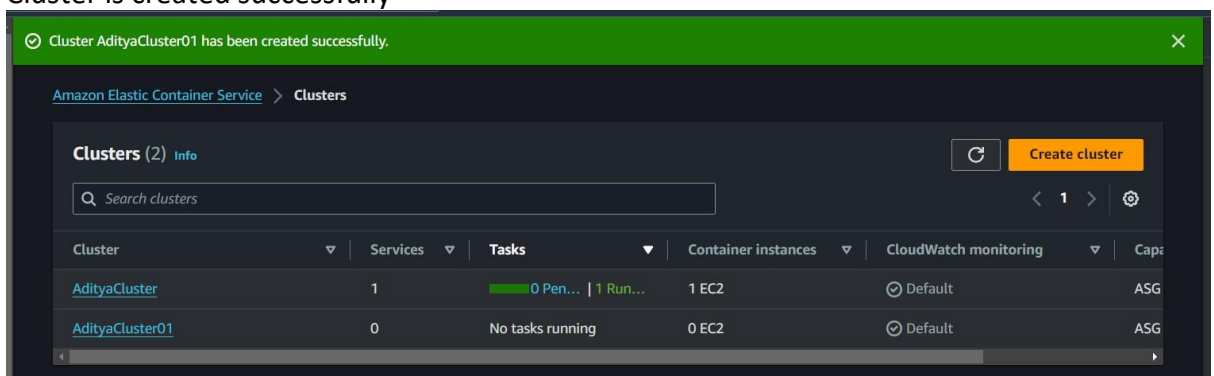
Security group name
Choose an existing security group.

Auto-assign public IP [Info](#)
Choose whether to auto-assign a public IP to the Amazon EC2 instances

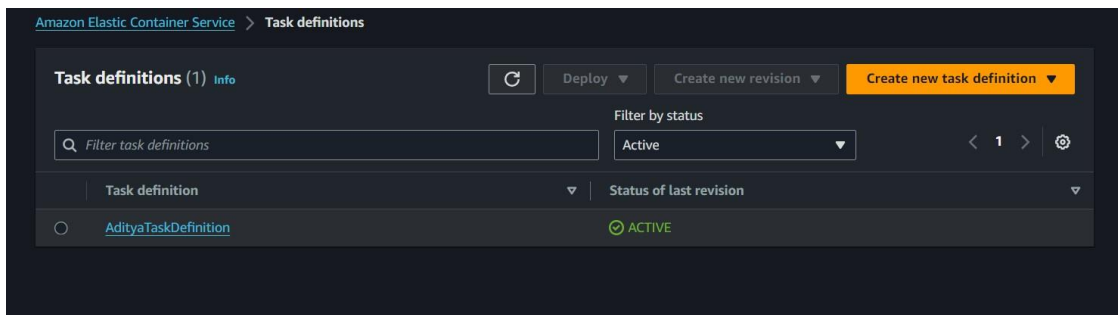
Inbound rules of the security group which I have chosen are: Port range 3000 is here because I have exposed the port 3000 for the nodejs application.



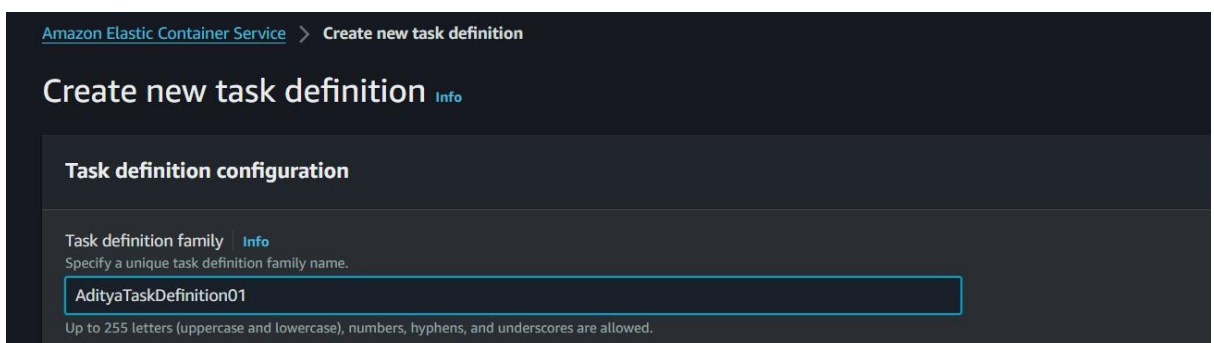
Cluster is created successfully



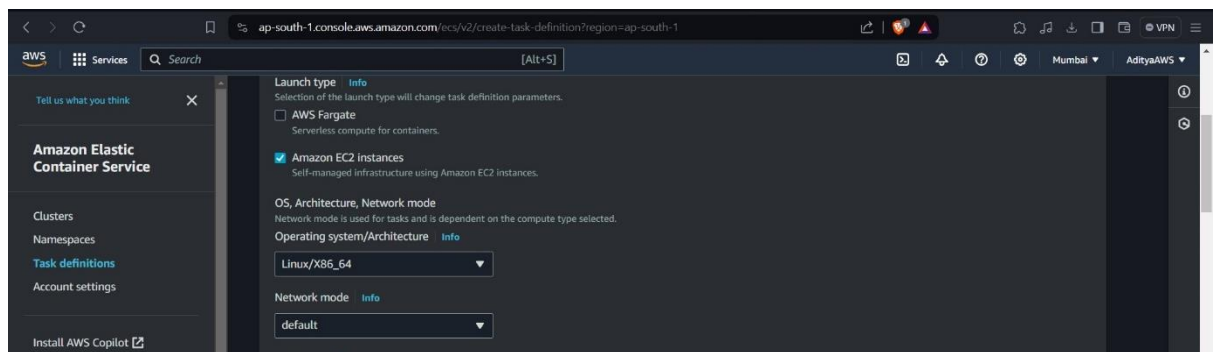
10. Now create "Task definition".



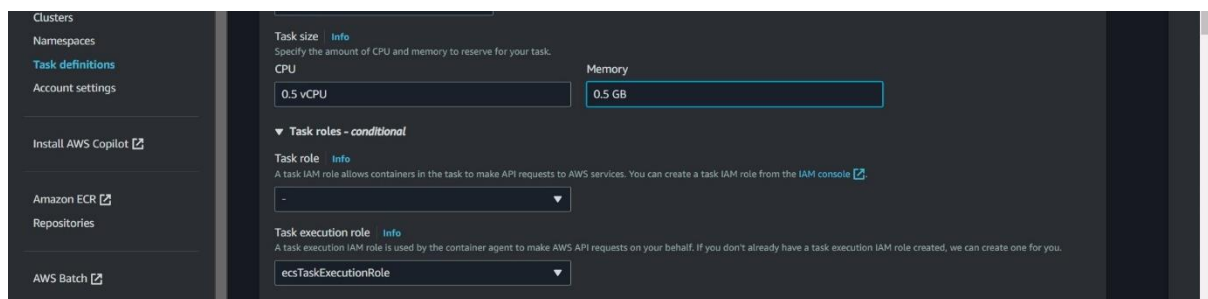
11. Name the task definition family



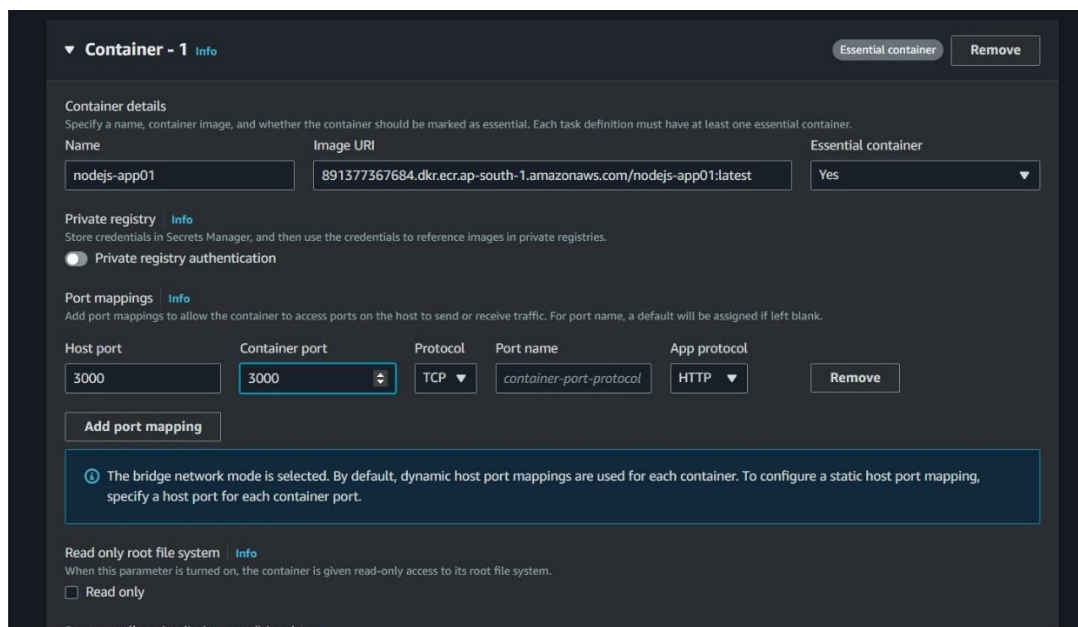
12. Select the EC2 as launch type, in OS “Linux/X86_64”.



13. In task size CPU: 0.5 vCPU, Memory: 0.5 GB and in Task execution role choose “ecsTaskExecutionRole”



14. Now, fill the container details:



and leave everything as it is and click on create and the task definition is created

Amazon Elastic Container Service > Task definitions > AdityaTaskDefinition01 > Revision 1 > Containers

AdityaTaskDefinition01:1

Deploy Actions Create new revision

Overview

ARN arn:aws:ecs:ap-south-1:891377367:684:task-definition/AdityaTaskDefinition01:1	Status ACTIVE	Time created 2024-03-11T11:07:56.221Z	App environment EC2
Task role -	Task execution role ecsTaskExecutionRole	Operating system/Architecture Linux/X86_64	Network mode default

Containers JSON Task placement Volumes (0) Requires attributes Tags

Task size

Task CPU 512 units (0.5 vCPU)	Task memory 512 MiB (0.5 GB)
Task CPU maximum allocation for containers	Task memory maximum allocation for container memory reservation

15. Creating service in ECS cluster:

- Go to “AdityaCluster01” and in services section click create

AdityaCluster01

ASG Update cluster Delete cluster

Cluster overview

ARN arn:aws:ecs:ap-south-1:891377367:684:cluster/AdityaCluster01	Status Active	CloudWatch monitoring Default	Registered container instances -
Services Draining -	Active -	Tasks Pending -	Running -

Services Tasks Infrastructure Metrics Scheduled tasks Tags

Services (0)

Manage tags Update Delete service Create

Filter launch type: Any launch type Filter service type: Any service type

Service name	ARN	Status	Service...	Deployments and tasks	Last dep
No services					

b. Now, leave these as default:

Amazon Elastic Container Service > Clusters > AdityaCluster01 > Create service

Create Info

Environment Amazon EC2

Existing cluster
AdityaCluster01

▼ Compute configuration (advanced)

Compute options Info
To ensure task distribution across your compute types, use appropriate compute options.

☒ **Capacity provider strategy**
Specify a launch strategy to distribute your tasks across one or more capacity providers.

☐ **Launch type**
Launch tasks directly without the use of a capacity provider strategy.

Capacity provider strategy Info
Select either your cluster default capacity provider strategy or select the custom option to configure a different strategy.

☒ **Use cluster default**

☐ Use custom (Advanced)

Capacity provider: Infra-ECS-Cluster-AdityaClust... Base: 0 Weight: 1

c. Now, in task definition choose “AdityaTaskDefinition01”, Revision: 1(LATEST), Service name: “AdityaClusterService” and leave other sections as it is.

Deploying to Amazon Elastic Container Service > Aditya deployment test - aditya... > v3.1.0 Deploy NodeJS Application on... > Create service | Elastic Container... > Elastic Container Registry

ap-south-1.console.aws.amazon.com/ec2/v2/clusters/AdityaCluster01/create-service?region=ap-south-1

Task definition
Select an existing task definition. To create a new task definition, go to [Task definitions](#).
☐ Specify the revision manually
Manually input the revision instead of choosing from the 100 most recent revisions for the selected task definition family.

Family: AdityaTaskDefinition01 Revision: 1 (LATEST)

Service name
Assign a unique name for this service.
AdityaClusterService

Service type Info
Specify this service type that the service scheduler will follow.

☒ **Replica**
Place and maintain a desired number of tasks across your cluster.

☐ **Daemon**
Place and maintain one copy of your task on each container instance.

Desired tasks
Specify the number of tasks to launch.
1

► Deployment options

► Deployment failure detection Info

d. Click on create

Service discovery - optional
Service discovery uses Amazon Route 53 to create a namespace for your service, which allows it to be discoverable via DNS.

► Load balancing - optional
Configure load balancing using Amazon Elastic Load Balancing to distribute traffic evenly across the healthy tasks in your service.

► Service auto scaling - optional
Automatically adjust your service's desired count up and down within a specified range in response to CloudWatch alarms. You can modify your service auto scaling configuration at any time to meet the needs of your application.

► Task placement Info
Let's you customize how tasks are placed on instances within your cluster. Different placement strategies are available to optimize for availability and efficiency.

► Volume
Configure a data volume to provide additional storage for the containers in the task.

► Tags - optional Info
Tags help you to identify and organize your resources.

Cancel Create

16. Now, download the AdityaTaskDefinition01-revision1.json for this go to task definitions and open the “AdityaTaskDefinition01:1”

The screenshot shows the Amazon ECS console page for the task definition **AdityaTaskDefinition01:1**. The breadcrumb navigation at the top is: Amazon Elastic Container Service > Task definitions > AdityaTaskDefinition01 > Revision 1 > Containers. The page has three buttons: **Deploy**, **Actions**, and **Create new revision**. The **Overview** tab is selected, showing a table with the following details:

ARN arn:aws:ecs:ap-south-1:891377367684:task-definition/AdityaTaskDefinition01:1	Status ACTIVE	Time created 2024-03-11T11:07:56.221Z	App environment EC2
Task role -	Task execution role ecsTaskExecutionRole	Operating system/Architecture Linux/X86_64	Network mode default

Below the table are tabs for **Containers**, **JSON**, **Task placement**, **Volumes (0)**, **Requires attributes**, and **Tags**. The **Task size** section shows:

Task CPU 512 units (0.5 vCPU)	Task memory 512 MiB (0.5 GB)
Task CPU maximum allocation for containers	Task memory maximum allocation for container memory reservation

17. Click on JSON and then click “Download JSON”

The screenshot shows the **JSON** tab of the task definition. At the top, there are three buttons: **Download JSON**, **Download AWS CLI input**, and **Copy to clipboard**. The JSON content is as follows:

```
1 {
2   "taskDefinitionArn": "arn:aws:ecs:ap-south-1:891377367684:task-definition/AdityaTaskDefinition01:1",
3   "containerDefinitions": [
4     {
5       "name": "nodejs-app01",
6       "image": "891377367684.dkr.ecr.ap-south-1.amazonaws.com/nodejs-app01:latest",
7       "cpu": 0,
8       "portMappings": [
9         {
10          "name": "nodejs-app01-3000-tcp",
11          "containerPort": 3000,
12          "hostPort": 3000,
13          "protocol": "tcp",
14          "appProtocol": "http"
15        }
16      ],
17       "essential": true,
18       "environment": [],
19       "environmentFiles": [],
20       "mountPoints": [],
21       "volumesFrom": [],
22       "ulimits": [],
23       "logConfiguration": {
24         "logDriver": "awslogs",
25         "options": {
26           "awslogs-create-group": "true"
27         }
28       }
29     }
30   ]
31 }
```

18. After downloading, move it in the project root folder

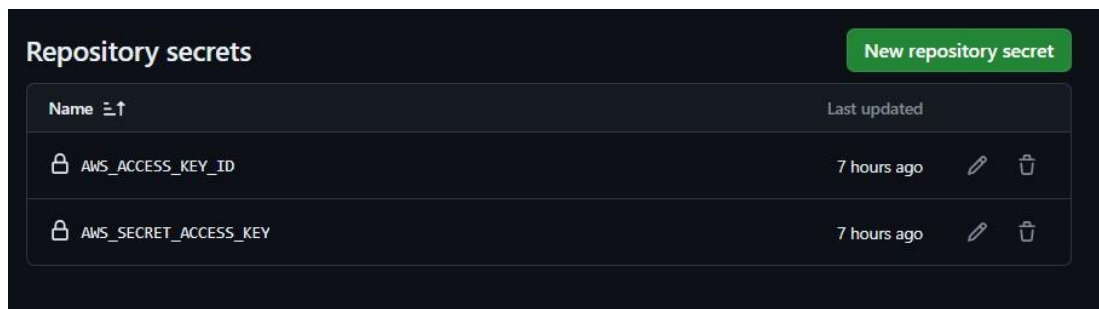
The screenshot shows a file explorer view of a project root folder. The files and folders are:

- .github\workflows
- aditya-deploy.yaml
- Documentation
- node_modules
- public
- .gitignore
- AdityaTaskDefinition01-revisio...
- app.js
- Dockerfile
- package-lock.json
- package.json

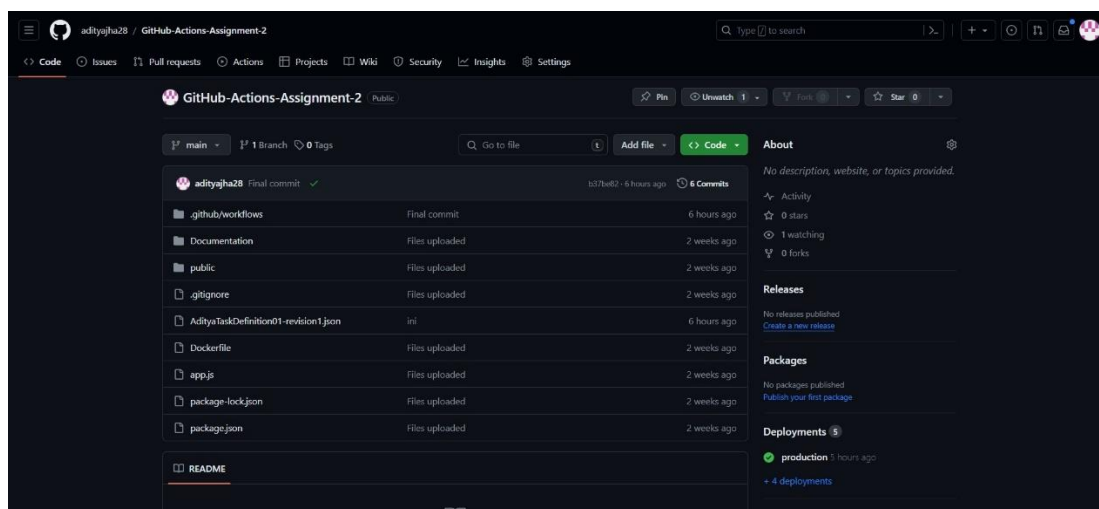
19. Create a folder .github, inside it create workflows and in workflows create a file named "aditya-deploy.yaml"

```
aditya-deploy.yaml X
.github > workflows > aditya-deploy.yaml
1
2 name: Deploy to Amazon ECS
3
4 on:
5   push:
6     branches:
7       - main
8
9 env:
10  AWS_REGION: ap-south-1           # set this to your preferred AWS region, e.g. ap-south-1
11  ECR_REPOSITORY: nodejs-app01      # set this to your Amazon ECR repository name
12  ECS_SERVICE: AdityaClusterService # set this to your Amazon ECS service name
13  ECS_CLUSTER: AdityaCluster01      # set this to your Amazon ECS cluster name
14  ECS_TASK_DEFINITION: AdityaTaskDefinition01-revision1.json # set this to the path to your Amazon ECS task definition
15                                     # file, e.g. .aws/task-definition.json
16  CONTAINER_NAME: nodejs-app01      # set this to the name of the container in the
17                                     # containerDefinitions section of your task definition
18
19 jobs:
20   deploy:
21     name: Deploy
22     runs-on: ubuntu-latest
23     environment: production
24
25     steps:
26       - name: Checkout
27         uses: actions/checkout@v4
28
29       - name: Configure AWS credentials
30         uses: aws-actions/configure-aws-credentials@0e613a0980cbf65ed5b322eb7a1e075d28913a83
31         with:
32           aws-access-key-id: ${ secrets.AWS_ACCESS_KEY_ID }
33           aws-secret-access-key: ${ secrets.AWS_SECRET_ACCESS_KEY }
34           aws-region: ${ env.AWS_REGION }
35
36       - name: Login to Amazon ECR
37         id: login-ecr
38         uses: aws-actions/amazon-ecr-login@62f4f872db3836360b72999f4b87f1ff13310f3a
39
40       - name: Build, tag, and push image to Amazon ECR
41         id: build-image
42         env:
43           ECR_REGISTRY: ${ steps.login-ecr.outputs.registry }
44           IMAGE_TAG: ${ github.sha }
45         run: |
46           # Build a docker container and
47           # push it to ECR so that it can
48           # be deployed to ECS.
49           docker build -t $ECR_REGISTRY/$ECR_REPOSITORY:$IMAGE_TAG .
50           docker push $ECR_REGISTRY/$ECR_REPOSITORY:$IMAGE_TAG
51           echo "image=$ECR_REGISTRY/$ECR_REPOSITORY:$IMAGE_TAG" >> $GITHUB_OUTPUT
52
53       - name: Fill in the new image ID in the Amazon ECS task definition
54         id: task-def
55         uses: aws-actions/amazon-ecs-render-task-definition@c804dfbdd57f713b6c079302a4c01db7017a36fc
56         with:
57           task-definition: ${ env.ECS_TASK_DEFINITION }
58           container-name: ${ env.CONTAINER_NAME }
59           image: ${ steps.build-image.outputs.image }
60
61       - name: Deploy Amazon ECS task definition
62         uses: aws-actions/amazon-ecs-deploy-task-definition@df9643053eda01f169e64a0e60233aacca83799a
63         with:
64           task-definition: ${ steps.task-def.outputs.task-definition }
65           service: ${ env.ECS_SERVICE }
66           cluster: ${ env.ECS_CLUSTER }
67           wait-for-service-stability: true
68
```

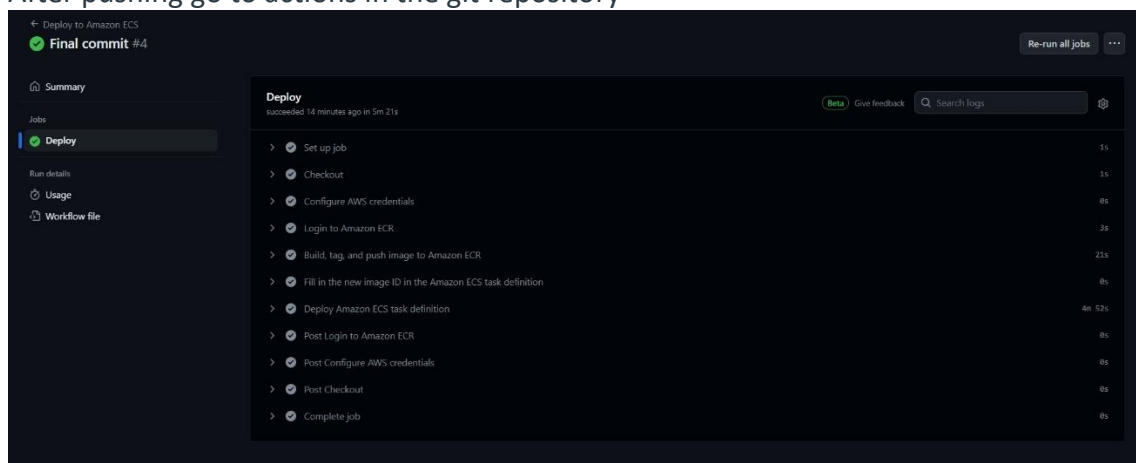
20. Create a repository and add secrets in the repository



21. Push the files in the repository



After pushing go to actions in the git repository



When the deployment gets completed.

22. Checkout the deployment:

Amazon Elastic Container Service > Clusters > AdityaCluster01 > Tasks > 98a7d33f7b8a4875b8e67f9977158b4e > Configuration

98a7d33f7b8a4875b8e67f9977158b4e

Configuration | Logs | Networking | Volumes (0) | Tags

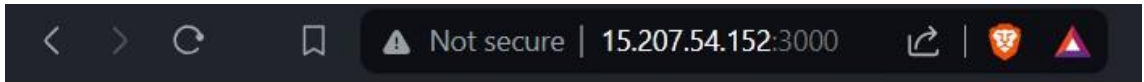
Task overview

ARN arn:aws:ecs:ap-south-1:891377367684:task/AdityaCluster01/98a7d33f7b8a4875b8e67f9977158b4e	Last status Running	Desired status Running	Started/Created at 2024-03-11T11:59:57.089Z 2024-03-11T11:57:25.970Z
--	------------------------	---------------------------	--

Configuration

Operating system/Architecture Linux/X86_64	Capacity provider Infra-ECS-Cluster-AdityaCluster01-3d12cb05-EC2CapacityProvider-puw1MSVOrNsE	ENI ID -	Public IP 15.207.54.152 open address
CPU Memory .5 vCPU .5 GB	Network mode -	Private IP 172.31.8.181	

By accessing this public IP “15.207.54.152:3000”




Hello, Docker!

Also on <http://15.207.54.152:3000/form>

Not secure | 15.207.54.152:3000/form

Docker Form

Enter Your Name:



Hello, Vivek! Now, you can Build, Ship, and Run.