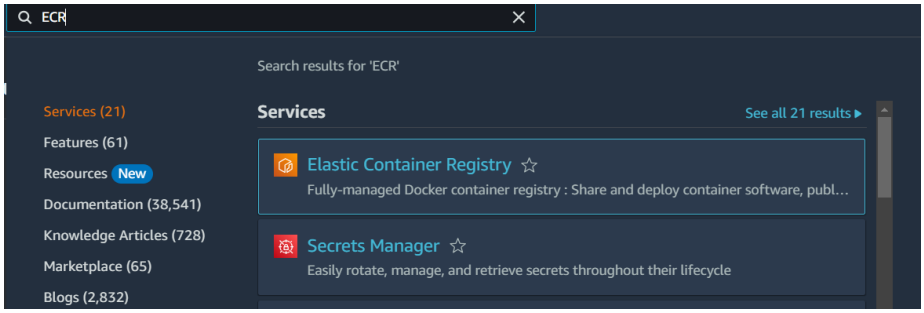
Objective: Create a GitHub Actions workflow that automates the building and deployment of a Node.js application to Amazon ECS using Amazon ECR.

Tasks:

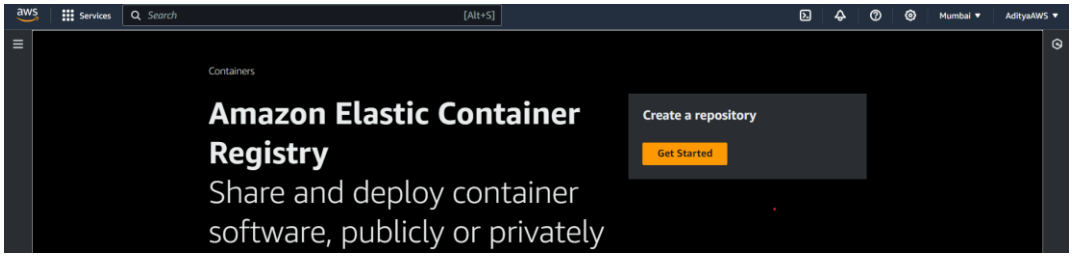
1. **Set up GitHub Actions Workflow**
2. **Define Environment Variables**
3. **Build and Push Docker Image**
4. **Deploy to Amazon ECS**
5. **Complete Workflow**

Steps:

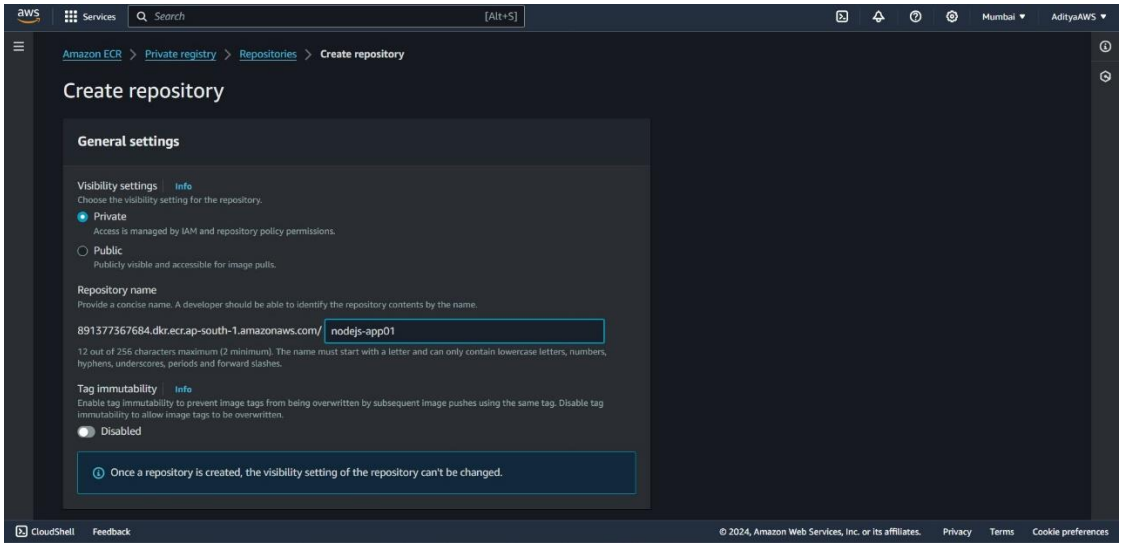
1. Create a ECR (Elastic Container Registry). In AWS console, search ECR.



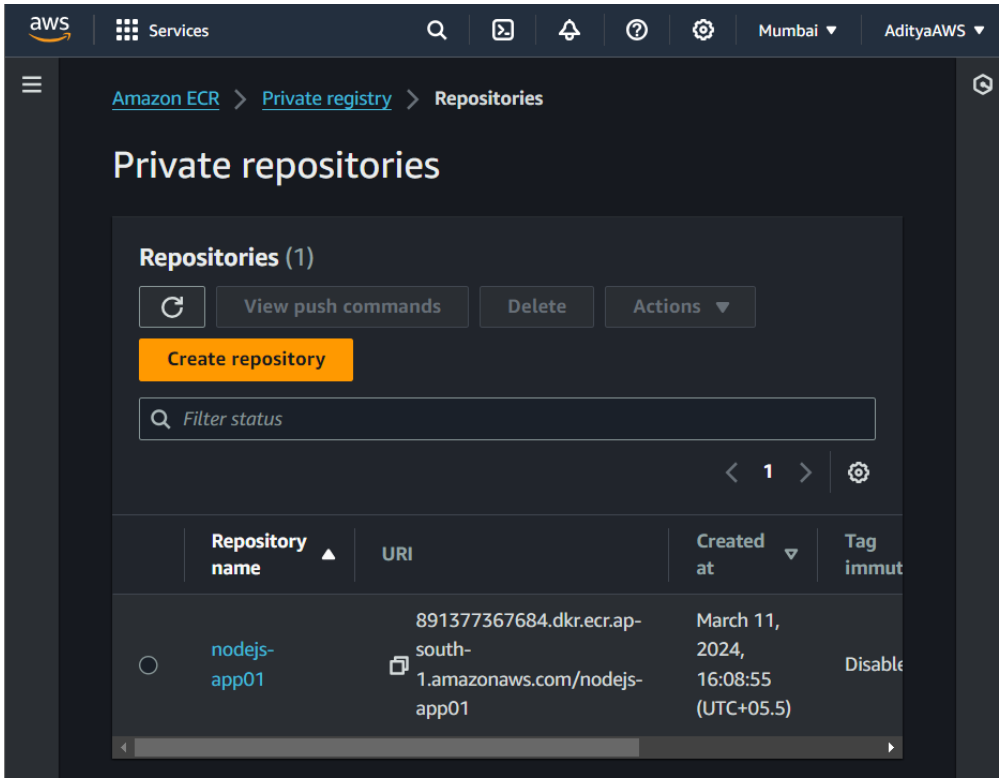
2. Click on “Get Started”



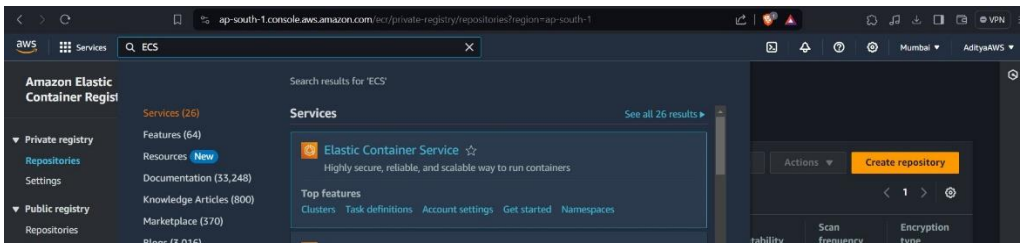
3. Now, create a repository, in visibility settings select private and name the repository. Scroll down and click on “Create repository”.



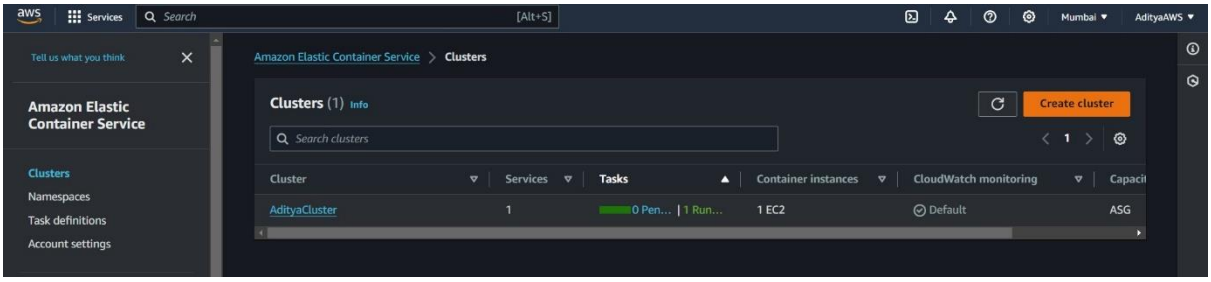
Repository created successfully



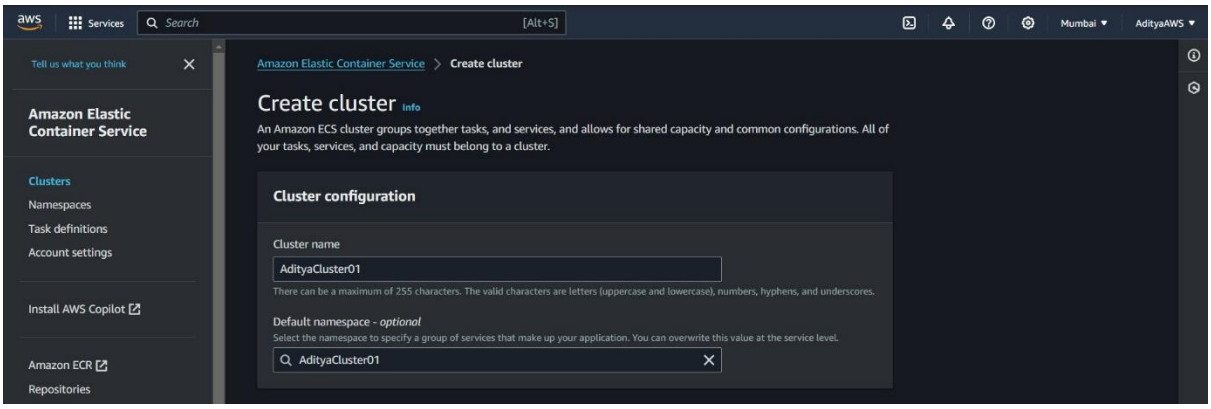
4. Setting up ECS: Search ECS



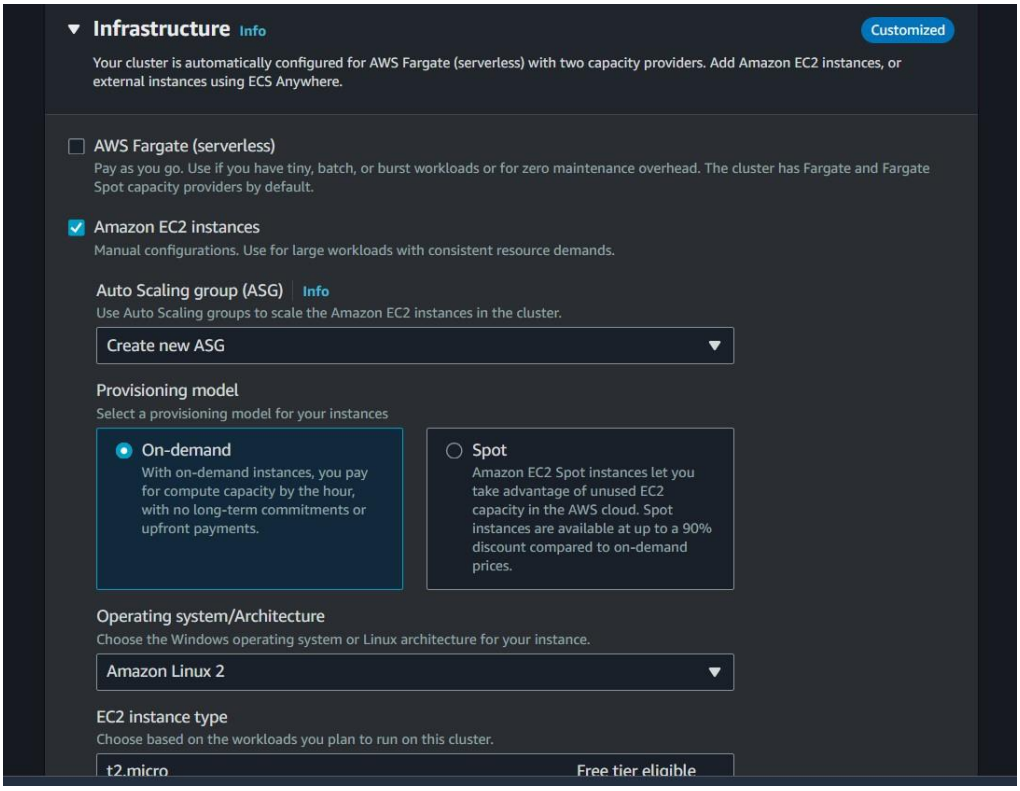
5. Create a cluster by clicking on “Create cluster”



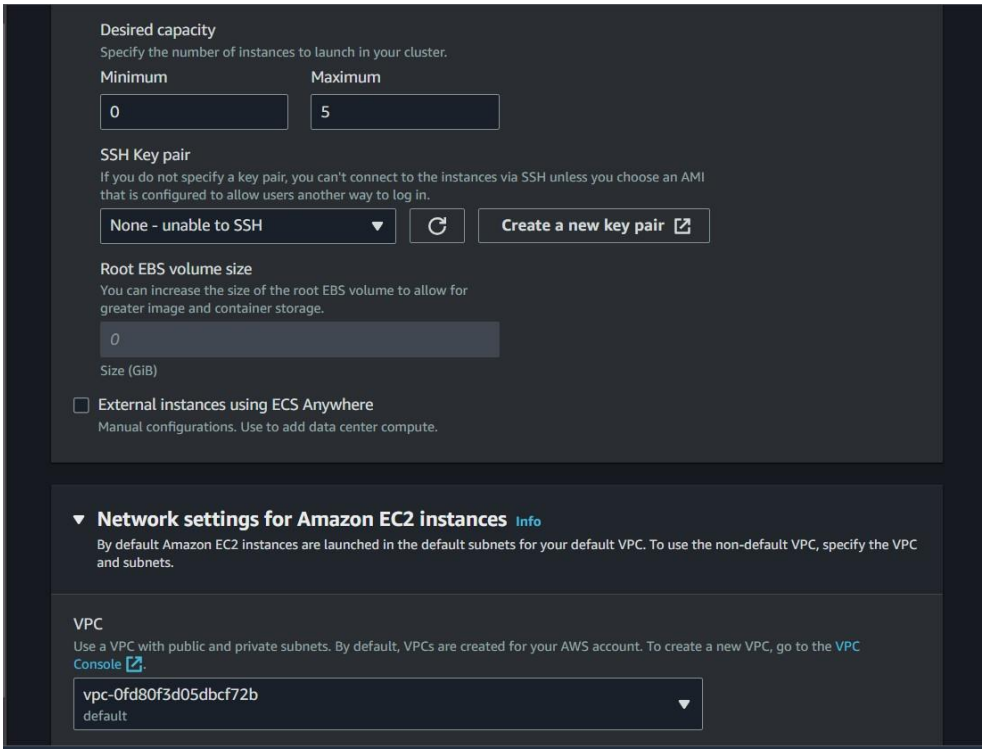
6. Now, name the cluster.



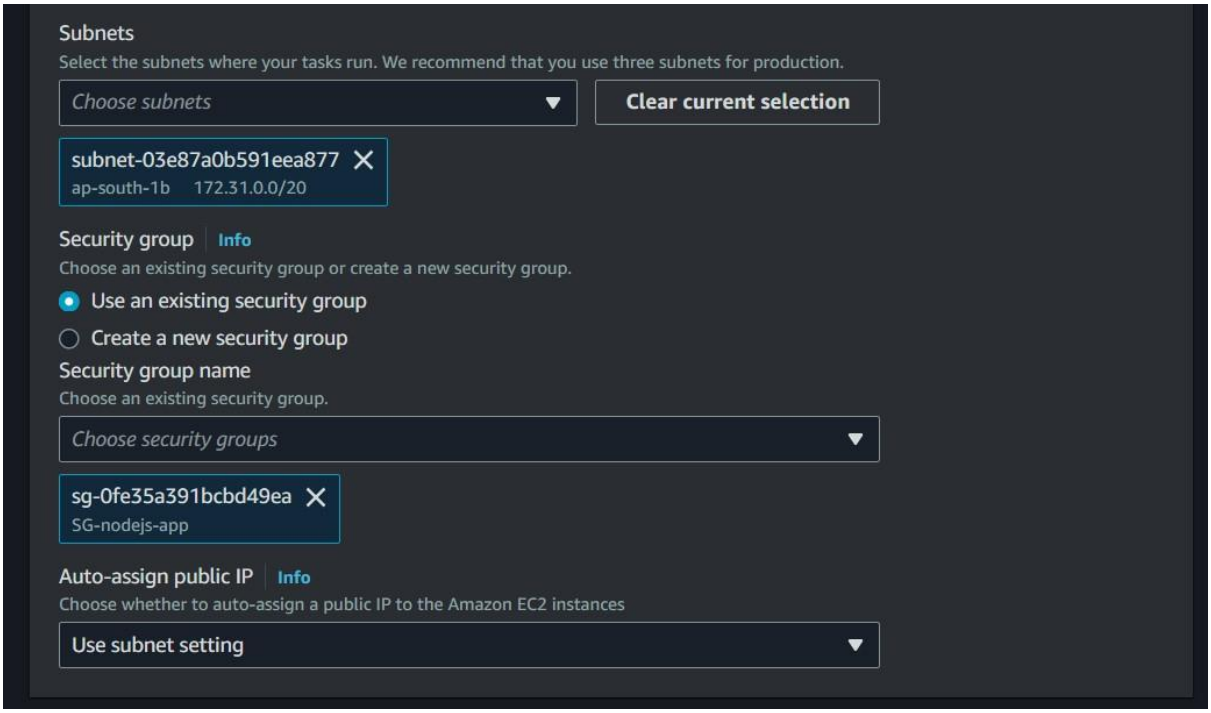
7. In infrastructure select Amazon EC2 instances, Create new ASG with provisioning model “On-demand”, select Amazon Linux 2 in operating system, instance type: t2.micro.



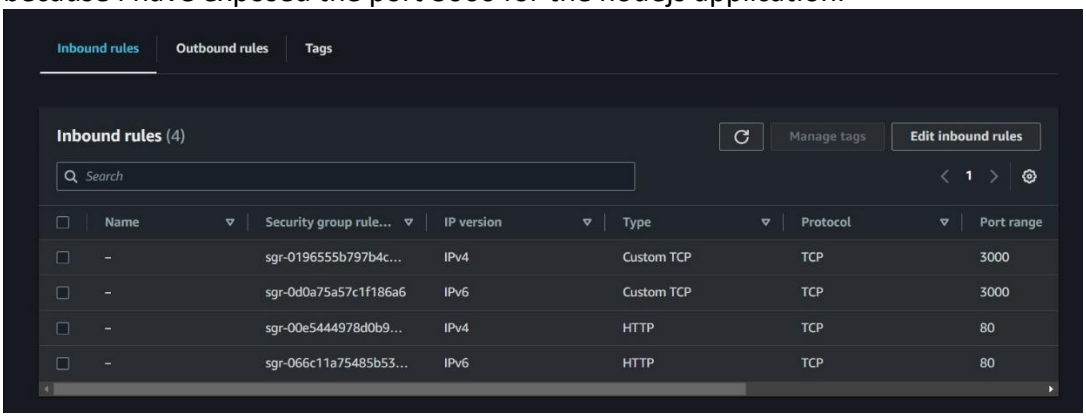
8. Leave these sections as it is



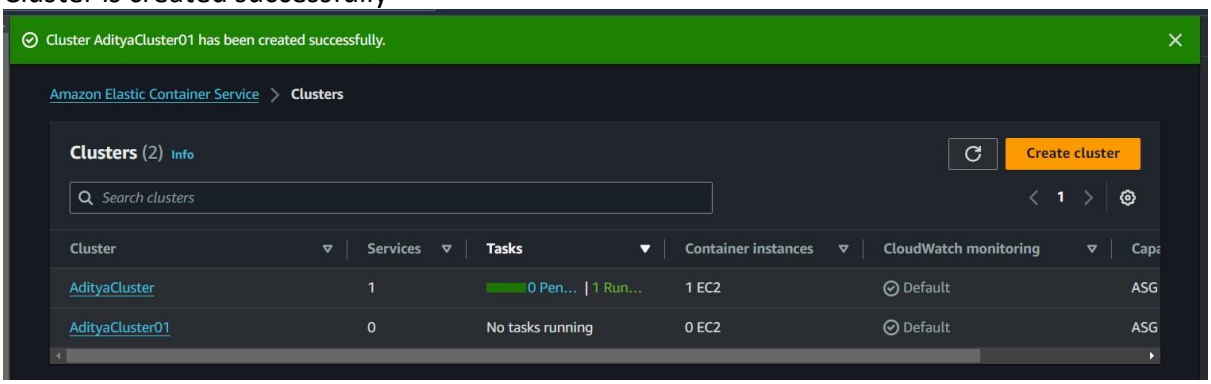
9. Move forward with default subnet and choose the security group SG-nodejs-app also enable the auto assign public IP and then click on create.



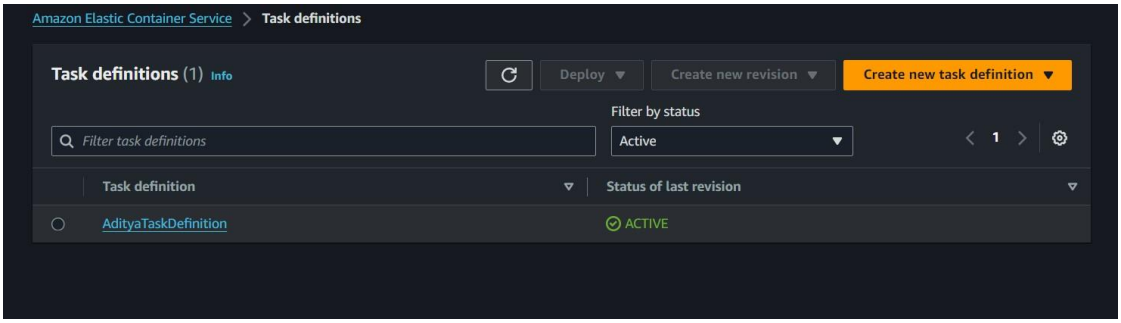
Inbound rules of the security group which I have chosen are: Port range 3000 ishere because I have exposed the port 3000 for the nodejs application.



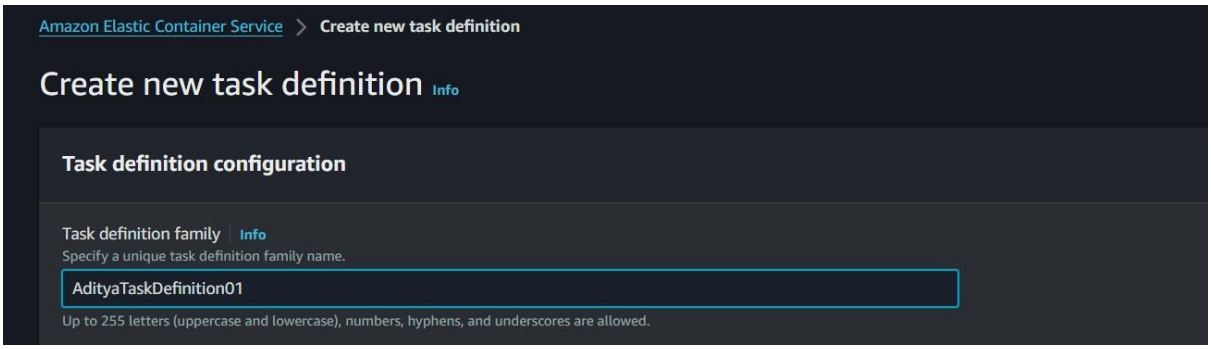
Cluster is created successfully



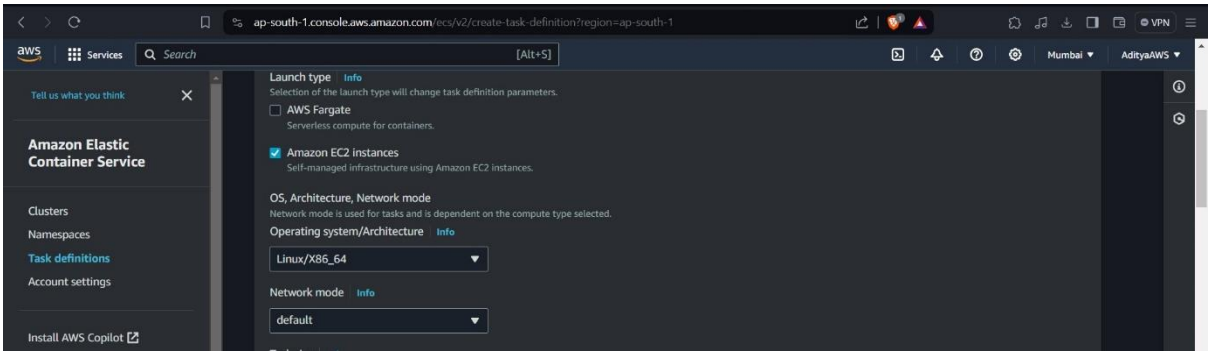
10. Now create “Task definition”.



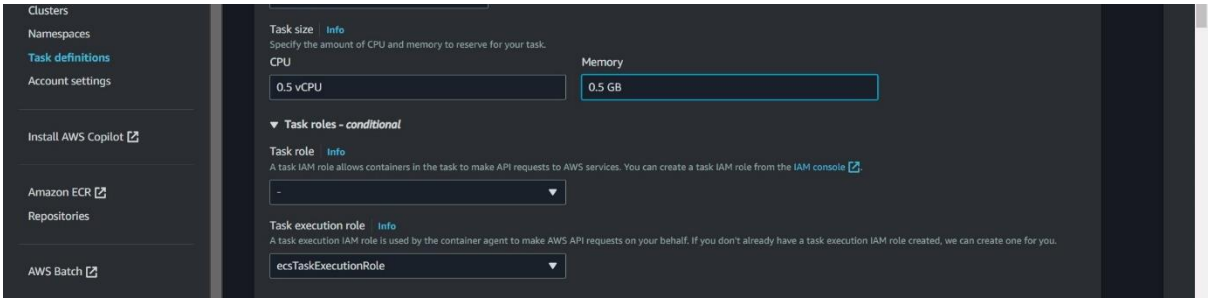
11. Name the task definition family



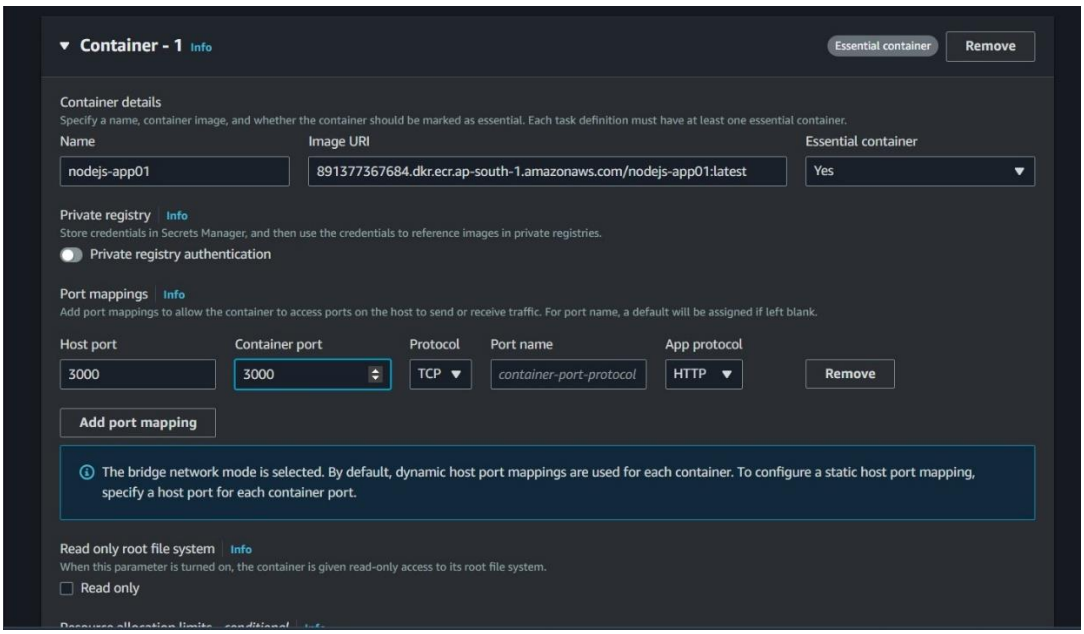
12. Select the EC2 as launch type, in OS “Linux/X86_64.



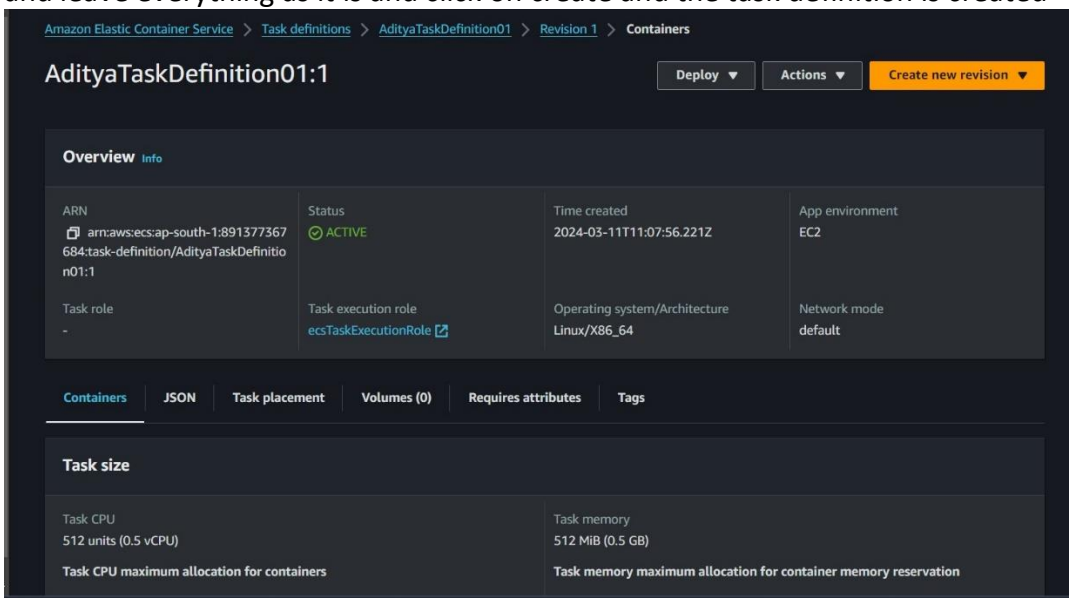
13. In task size CPU: 0.5 vCPU, Memory: 0.5 GB and in Task execution role choose “ecsTaskExecutionRole”



14. Now, fill the container details:

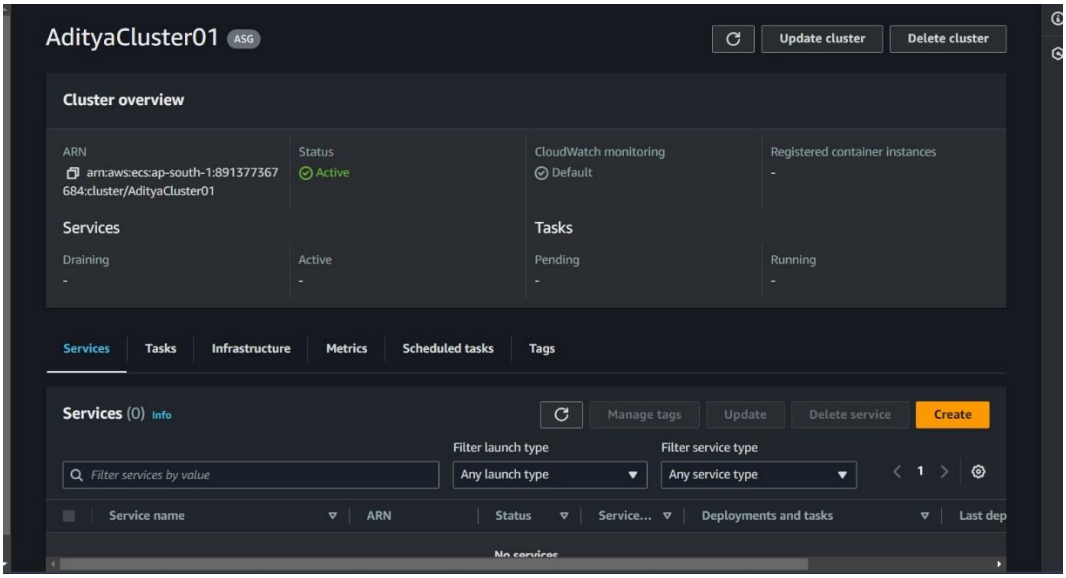


and leave everything as it is and click on create and the task definition is created

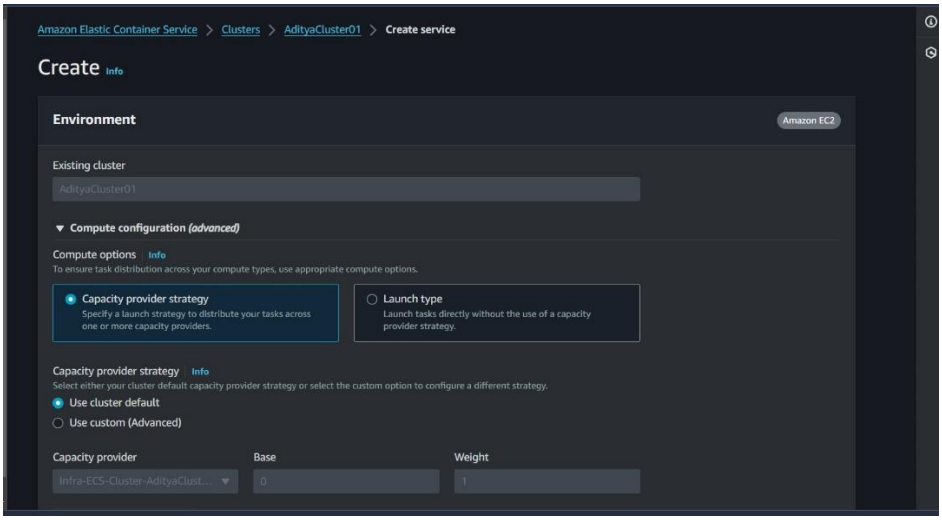


15. Creating service in ECS cluster:

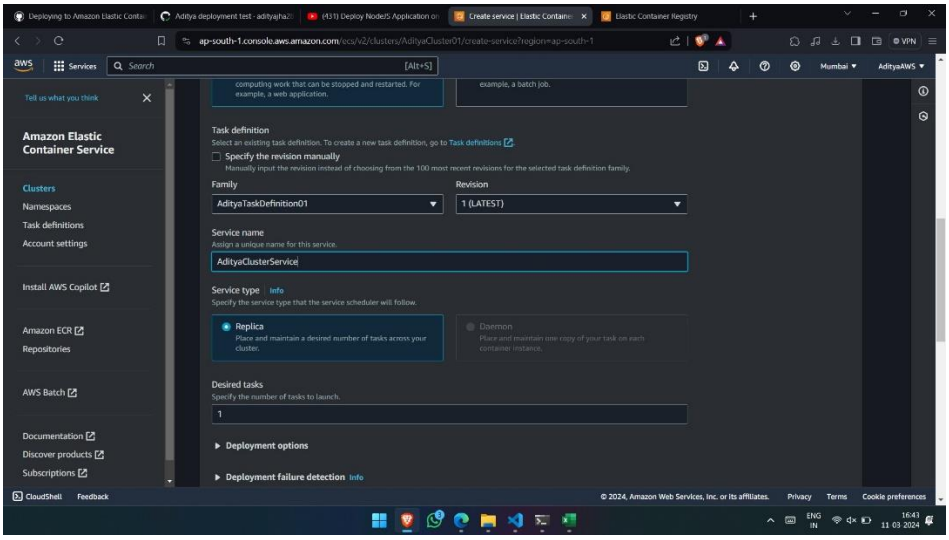
    a. Go to “AdityaCluster01” and in services section click create



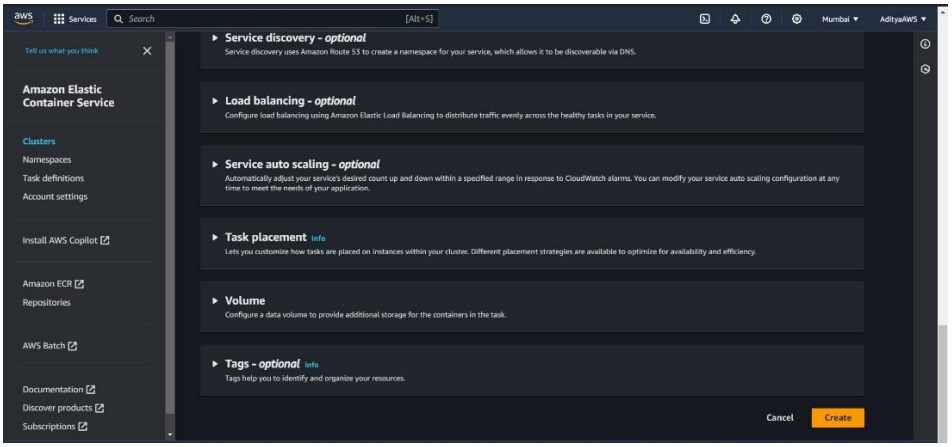
b. Now, leave these as default:



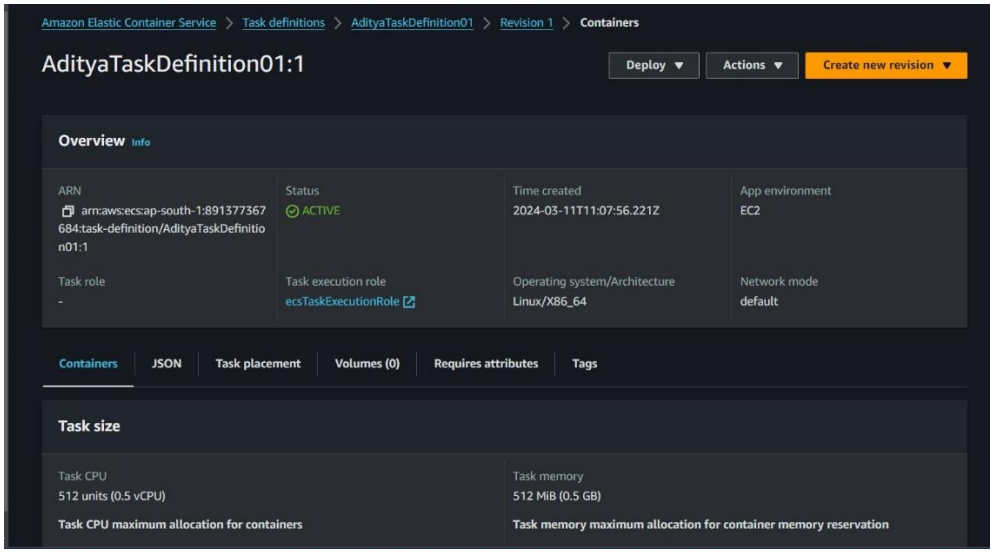
c. Now, in task definition choose “AdityaTaskDefinition01”, Revision: 1(LATEST), Service name: “AdityaClusterService” and leave other sections as it is.



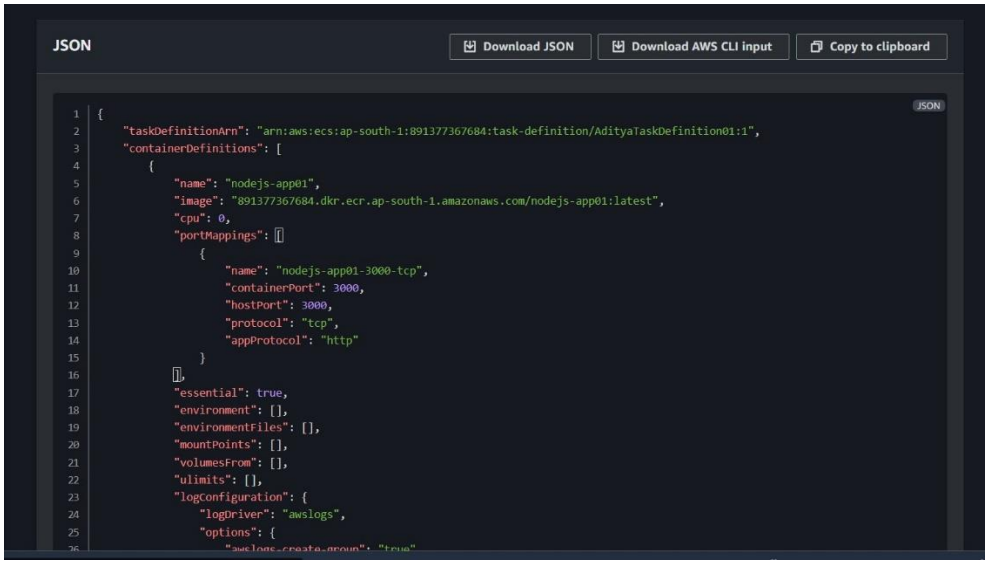
d. Click on create



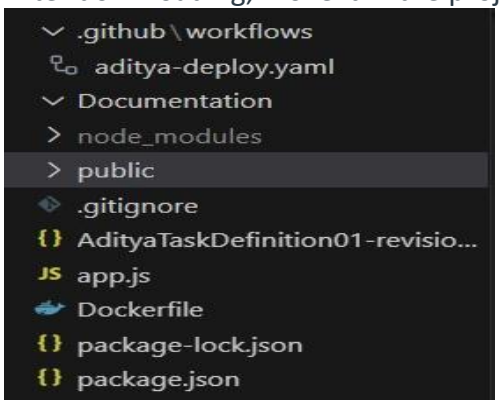
16. Now, download the AdityaTaskDefinition01-revision1.json for this go to task definitions and open the “AdityaTaskDefinition01:1”



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



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



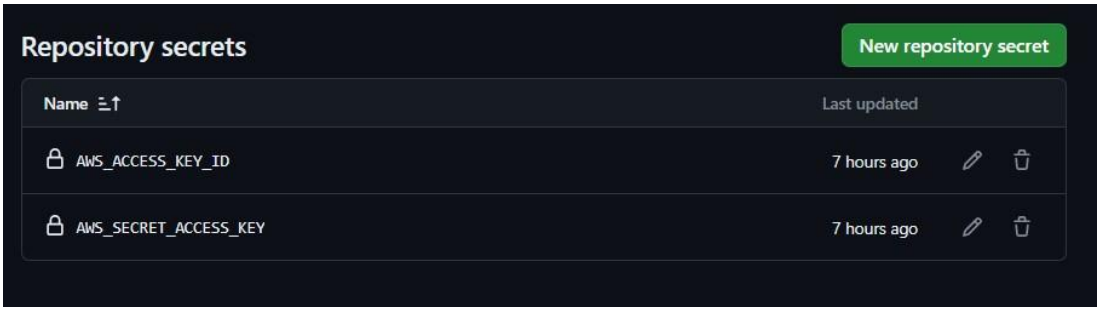
19. Create a folder .github, inside it create workflows and in workflows create a file named “aditya-deploy.yaml

aditya-deploy.yaml

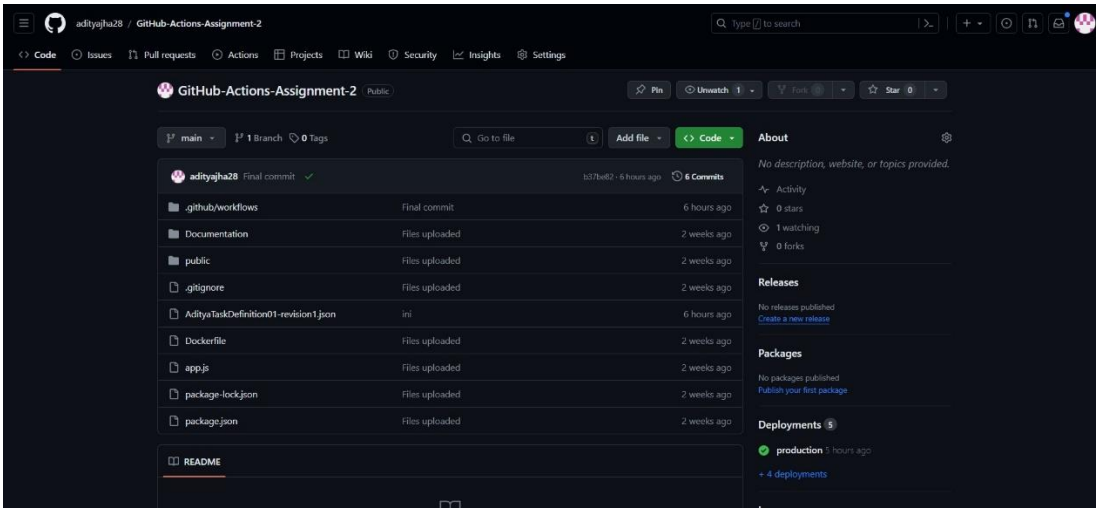
.github > workflows > aditya-deploy.yaml

```yaml

name: Deploy to Amazon ECS

on:
  push:
    branches:
      - main

env:
  AWS_REGION: ap-south-1                                  # set this to your preferred AWS region, e.g. ap-south-1
  ECR_REPOSITORY: nodejs-app01                            # set this to your Amazon ECR repository name
  ECS_SERVICE: AdityaClusterService                       # set this to your Amazon ECS service name
  ECS_CLUSTER: AdityaCluster01                            # set this to your Amazon ECS cluster name
  ECS_TASK_DEFINITION: AdityaTaskDefinition01-revision1.json # set this to the path to your Amazon ECS task definition
                                                          # file, e.g. .aws/task-definition.json
  CONTAINER_NAME: nodejs-app01                            # set this to the name of the container in the
                                                          # containerDefinitions section of your task definition

jobs:
  deploy:
    name: Deploy
    runs-on: ubuntu-latest
    environment: production

    steps:
      - name: Checkout
        uses: actions/checkout@v4

      - name: Configure AWS credentials
        uses: aws-actions/configure-aws-credentials@0e613a0980cbf65ed5b322eb7a1e075d28913a83
        with:
          aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
          aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
          aws-region: ${{ env.AWS_REGION }}

      - name: Login to Amazon ECR
        id: login-ecr
        uses: aws-actions/amazon-ecr-login@62f4f872db3836360b72999f4b87f1ff13310f3a

      - name: Build, tag, and push image to Amazon ECR
        id: build-image
        env:
          ECR_REGISTRY: ${{ steps.login-ecr.outputs.registry }}
          IMAGE_TAG: ${{ github.sha }}
        run: |
          # Build a docker container and
          # push it to ECR so that it can
          # be deployed to ECS.
          docker build -t $ECR_REGISTRY/$ECR_REPOSITORY:$IMAGE_TAG .
          docker push $ECR_REGISTRY/$ECR_REPOSITORY:$IMAGE_TAG
          echo "image=$ECR_REGISTRY/$ECR_REPOSITORY:$IMAGE_TAG" >> $GITHUB_OUTPUT

      - name: Fill in the new image ID in the Amazon ECS task definition
        id: task-def
        uses: aws-actions/amazon-ecs-render-task-definition@c804dfbdd57f713b6c079302a4c01db7017a36fc
        with:
          task-definition: ${{ env.ECS_TASK_DEFINITION }}
          container-name: ${{ env.CONTAINER_NAME }}
          image: ${{ steps.build-image.outputs.image }}

      - name: Deploy Amazon ECS task definition
        uses: aws-actions/amazon-ecs-deploy-task-definition@df9643053eda01f169e64a0e60233aacca83799a
        with:
          task-definition: ${{ steps.task-def.outputs.task-definition }}
          service: ${{ env.ECS_SERVICE }}
          cluster: ${{ env.ECS_CLUSTER }}
          wait-for-service-stability: true

```

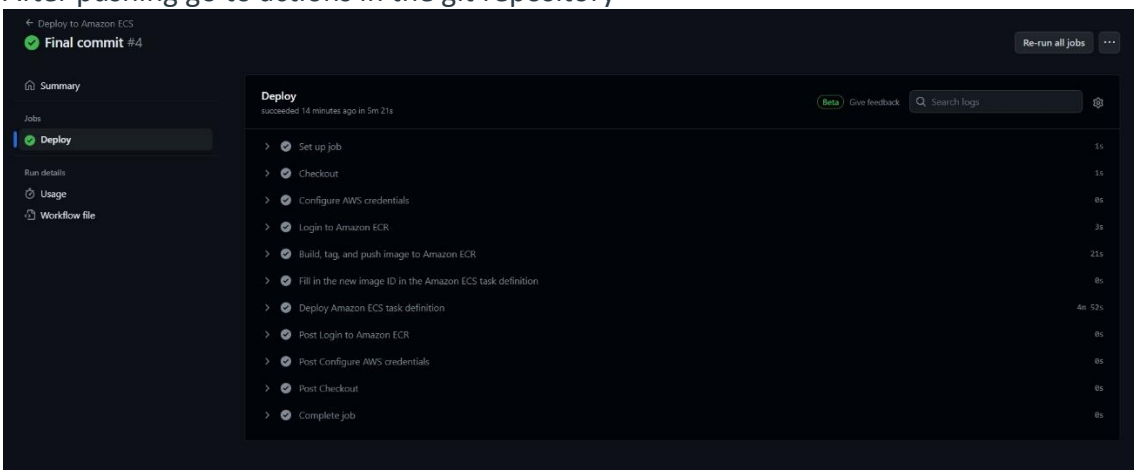
20. Create a repository and add secrets in the repository



21. Push the files in the repository

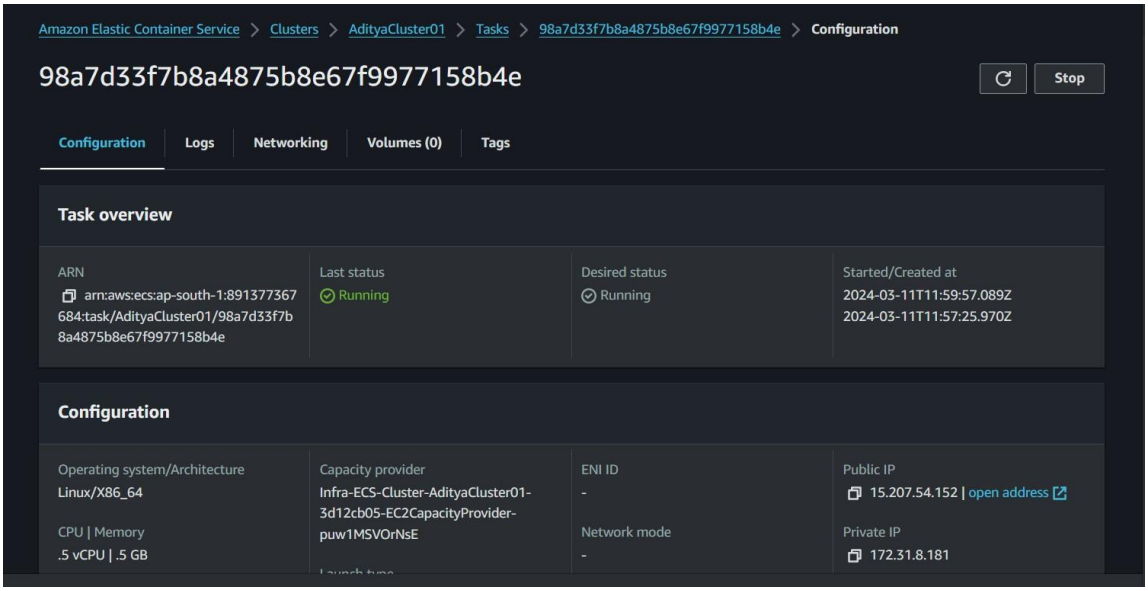


After pushing go to actions in the git repository



When the deployment gets completed.

22. Checkout the deployment:



By accessing this public IP “15.207.54.152:3000”

Also on http://15.207.54.152:3000/form