Assignment 2

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

Define Environment Variables

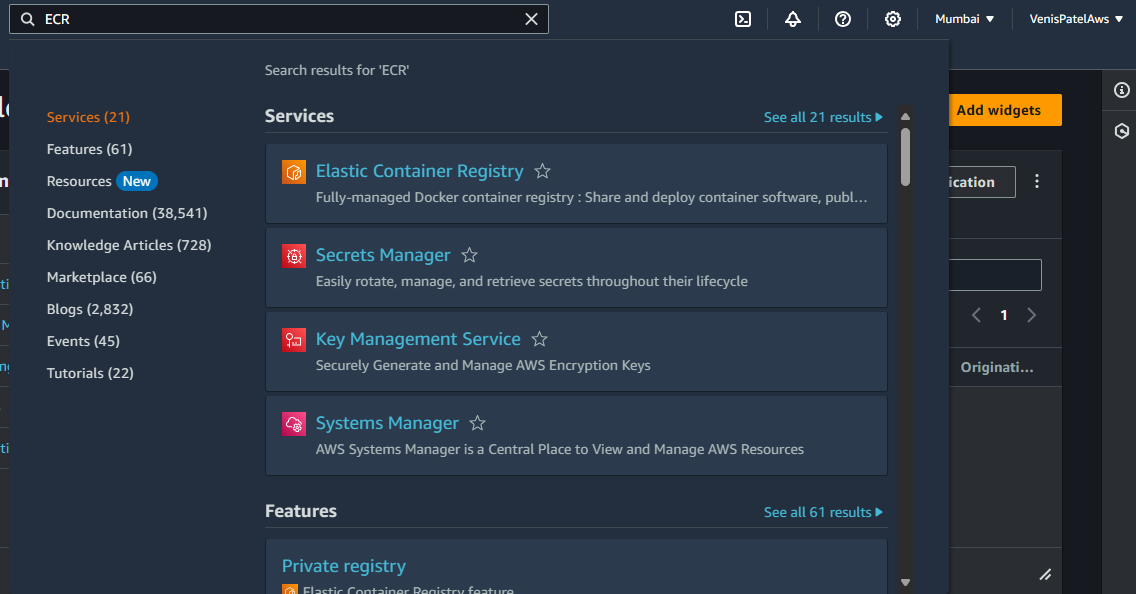
Build and Push Docker Image

Deploy to Amazon ECS

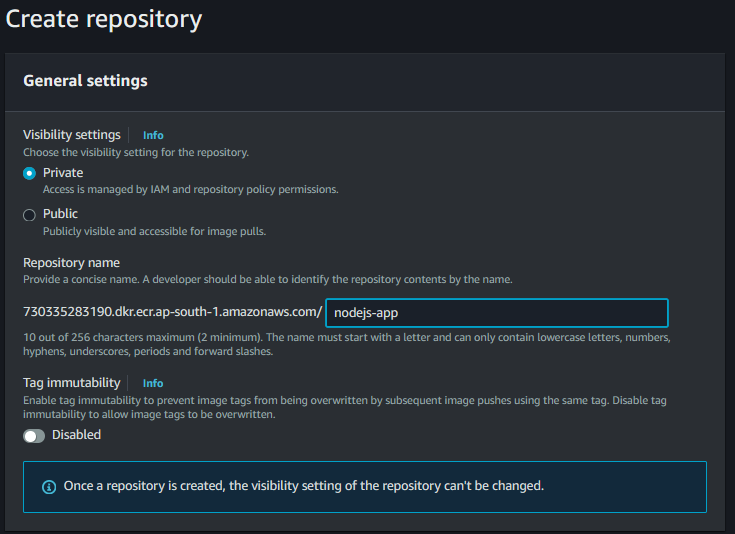
Complete Workflow

**Steps:**

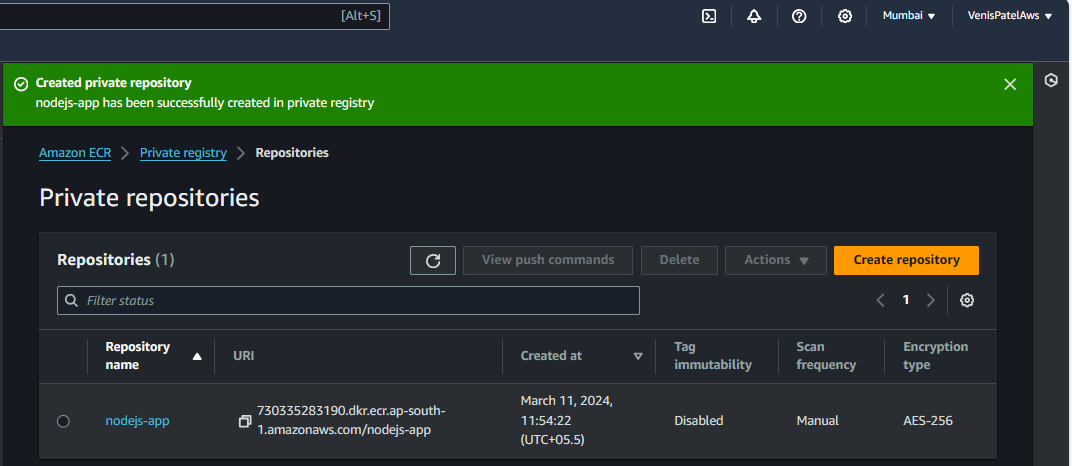
1. Go to Aws console and search “ECR” and click on “Elastic Container Registry” and then click on “Create repository”.



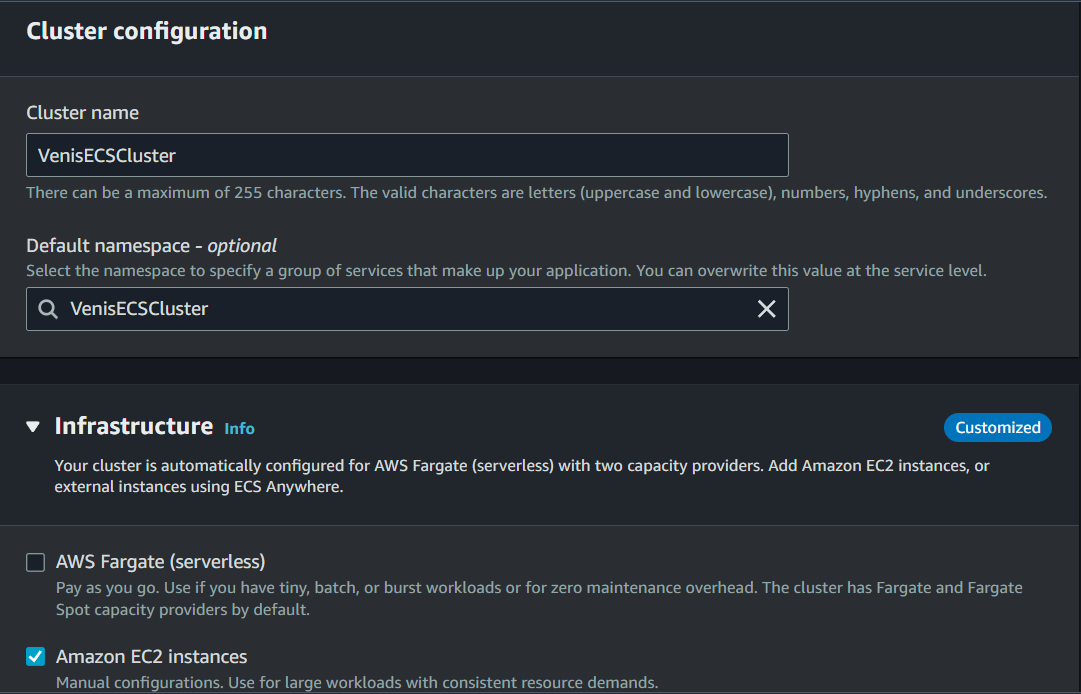
1. Now make the repository as private and give repository name and keep everything as it is.



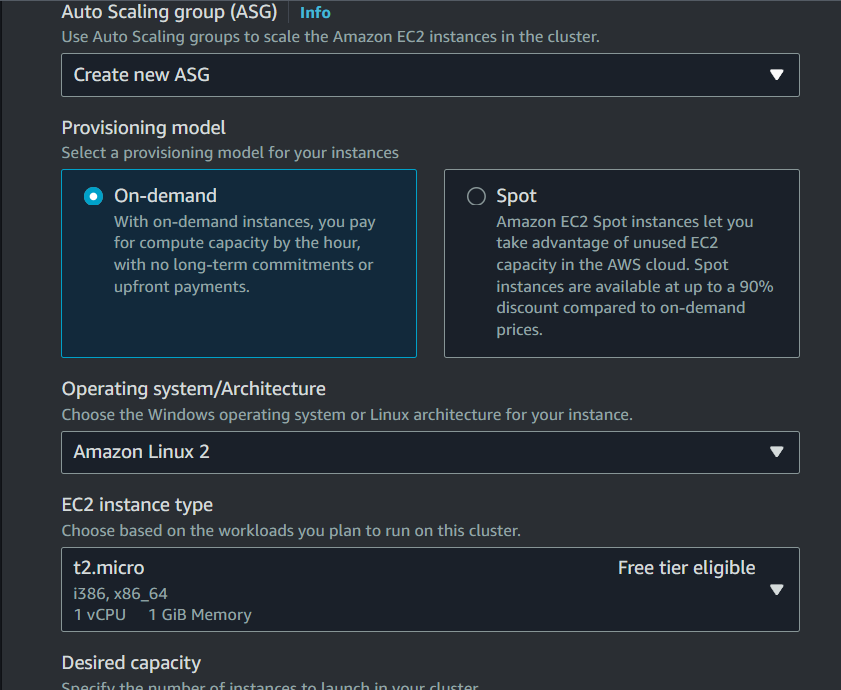
1. Here we can see that our repository has been created.



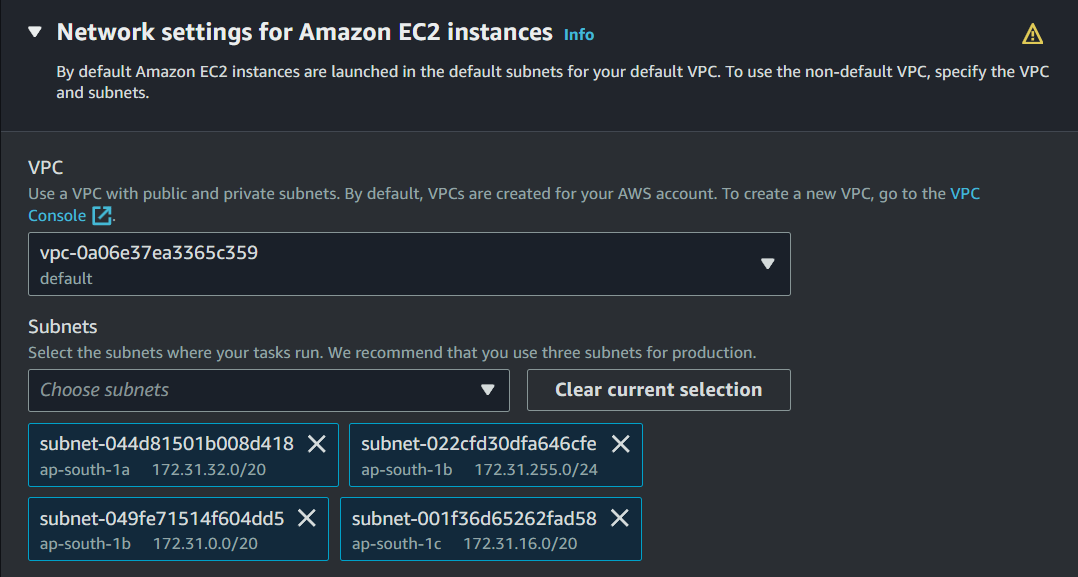
1. Now again go to Aws console and search for “ECS” and then go to cluster and click on “Create Cluster” and give Cluster name as “VenisECSCluster” and select infrastructure as “Amazon EC2 instances”.



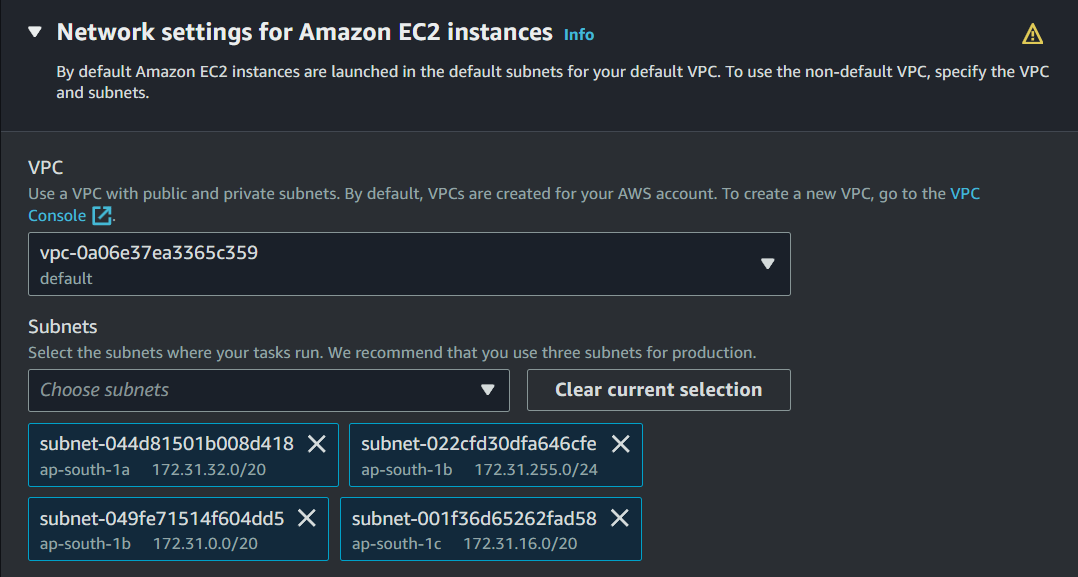
1. Now keep Provisioning model as On-demand and select Amazon Linux 2 as OS and Ec2 instance type as t2.micro.

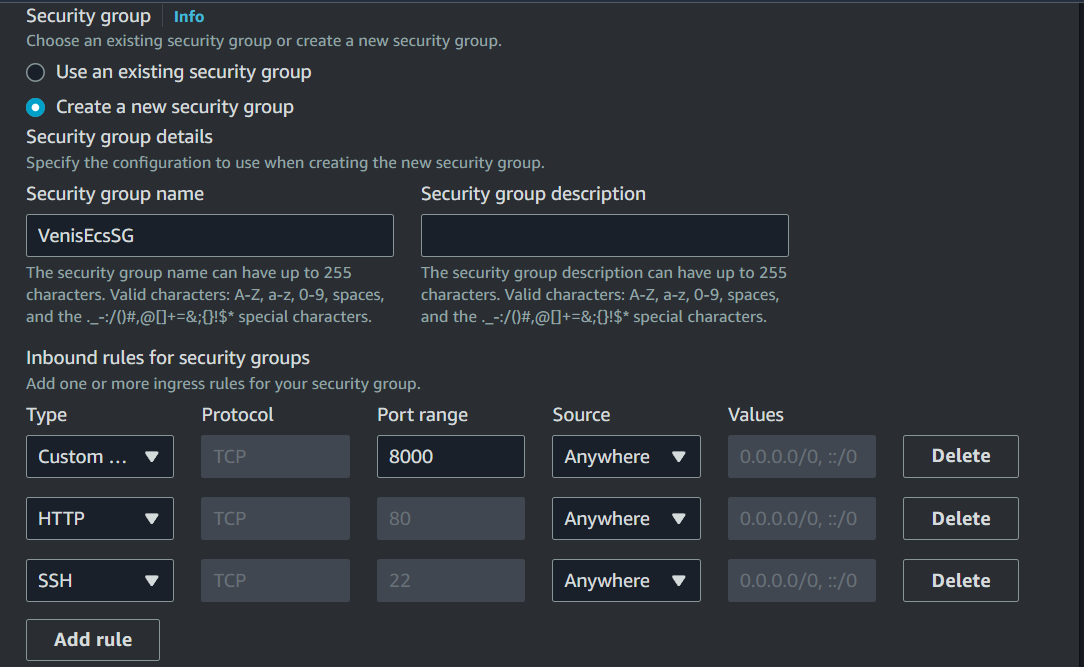


1. Set Min Desired capacity as 0 and Max as 2, select ssh key pair.

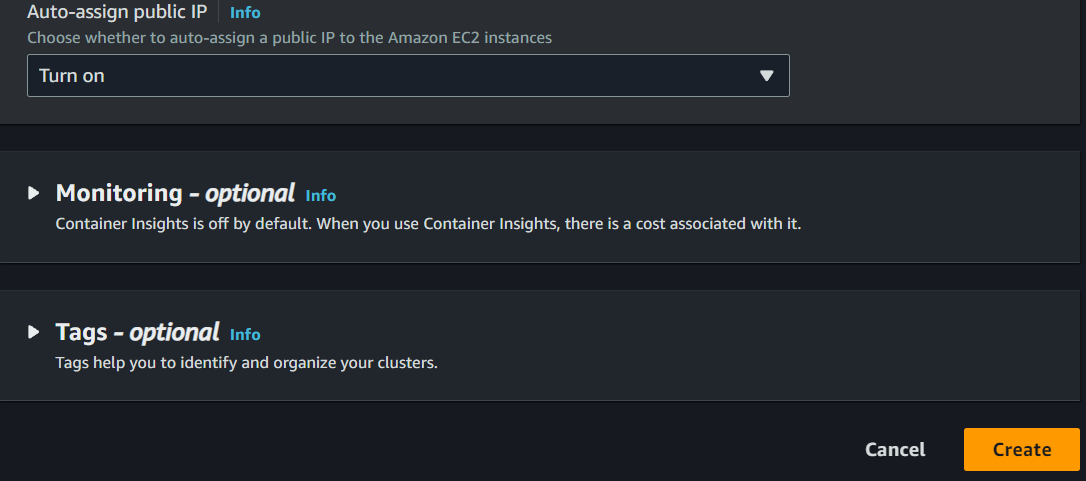


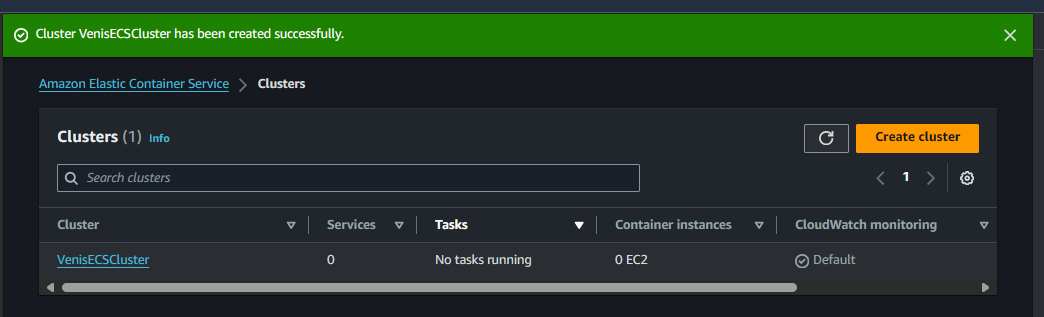
1. Now select any default vpc and all its subnets and then create a new Security group, Give name and in inbound rules allow port 8000 , HTTP and SSH .



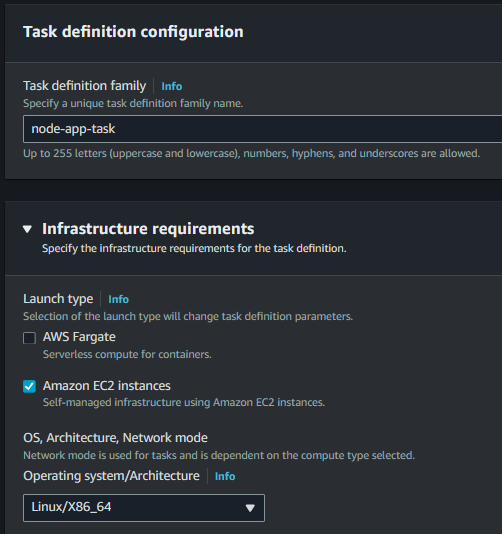


1. Now Turn on Auto-assign public ip and click on “Create”. Here we can see that our cluster has been created.

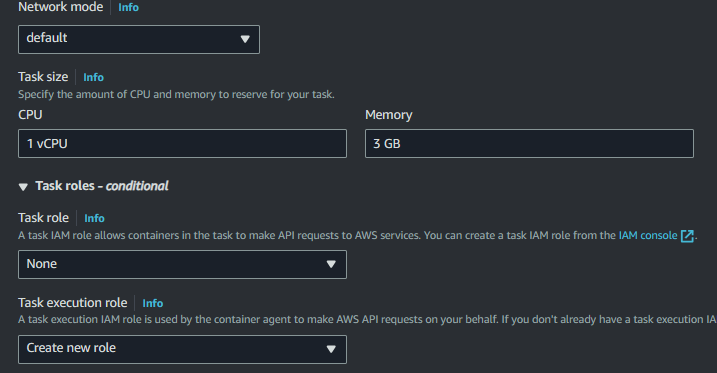


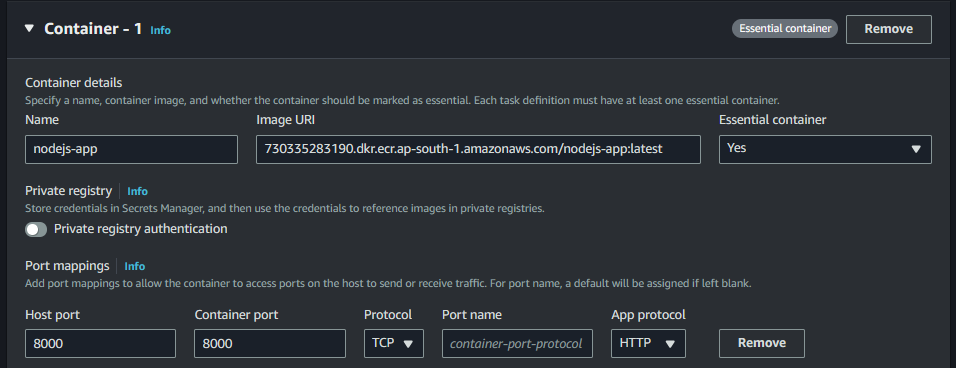


1. Now go to Task definitions and click on “Create” then select Task-definition family and Infrastructure requirements as “Amazon EC2 instances”.

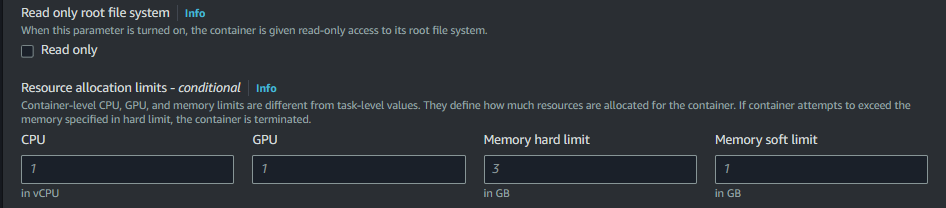


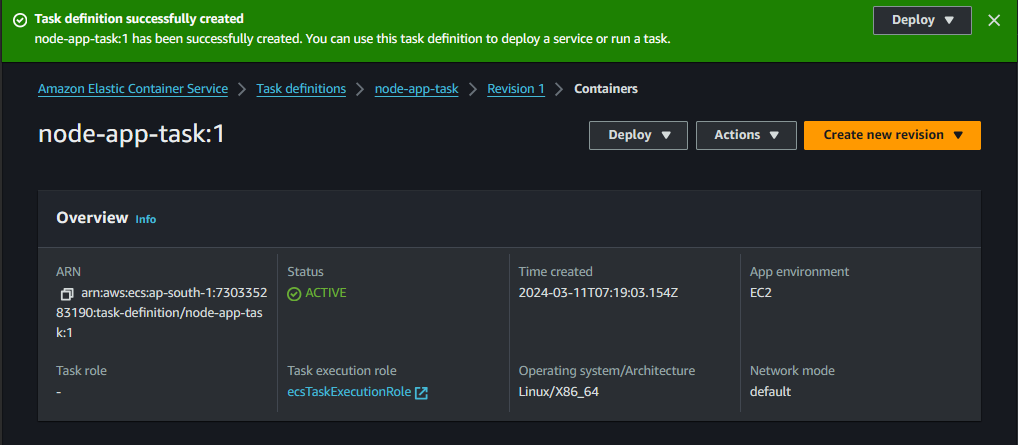
1. Keep Network mode as “default” and Give container name as “nodejs-app” and Give Image URI as Elastic Computing Repository link and in port mapping select 8000 as Host Port and Container Port.



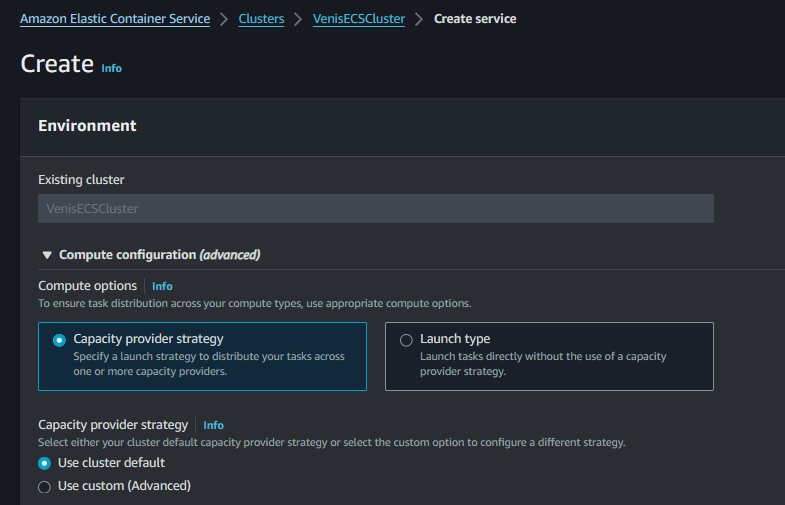


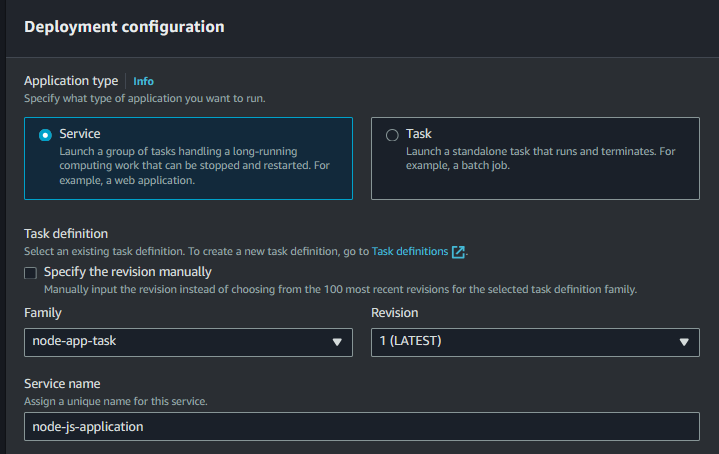
1. Now everything else as it is and click on “Create Task”.

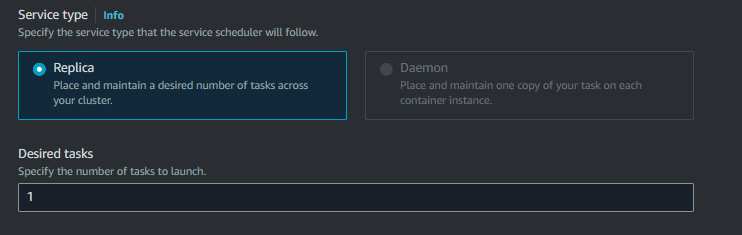




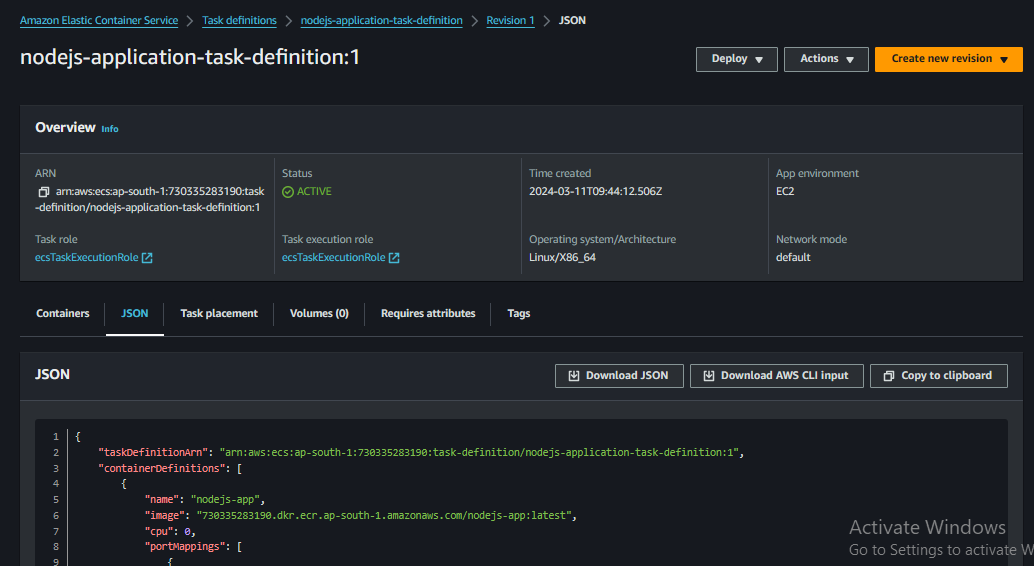
1. Now go to services and select on Create service then Select Application as Application as Service and give service name as “node-js-application”, set Desired tasks as 1 and select on Create service.

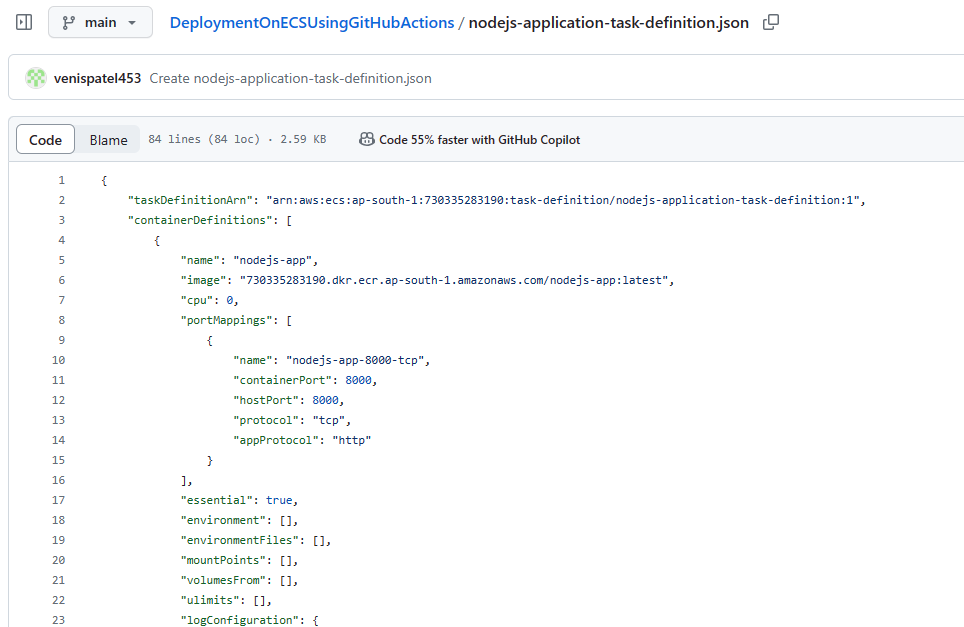






1. Now go to Task definitions and select task and now copy the json file of it, Now go to the GitHub Repository and make a new file named “nodejs-application-task-definition.json”.





1. Now create CI-CD.yml file in .github/workflows folder.

name: Deploy to Amazon ECS

on:

push:

branches:

- main

env:

AWS\_REGION: ap-south-1 # set this to your preferred AWS region, e.g. us-west-1

ECR\_REPOSITORY: nodejs-app # set this to your Amazon ECR repository name

ECS\_SERVICE: VenisECSService # set this to your Amazon ECS service name

ECS\_CLUSTER: VenisECSCluster

CONTAINER\_NAME: nodejs-app

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@v4

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@v2

with:

mask-password: 'true'

- name: Build, tag, and push image to Amazon ECR

id: build-image

env:

ECR\_REGISTRY: ${{ steps.login-ecr.outputs.registry }}

IMAGE\_TAG: latest

REPOSITORY: nodejs-app

run: |

docker build -t $ECR\_REGISTRY/$REPOSITORY:$IMAGE\_TAG .

docker push $ECR\_REGISTRY/$REPOSITORY:$IMAGE\_TAG

echo "image=$ECR\_REGISTRY/$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@v1

with:

task-definition: nodejs-application-task-definition.json

container-name: nodejs-app

image: ${{ steps.build-image.outputs.image }}

- name: Deploy Amazon ECS Task Definition

uses: aws-actions/amazon-ecs-deploy-task-definition@v1

with:

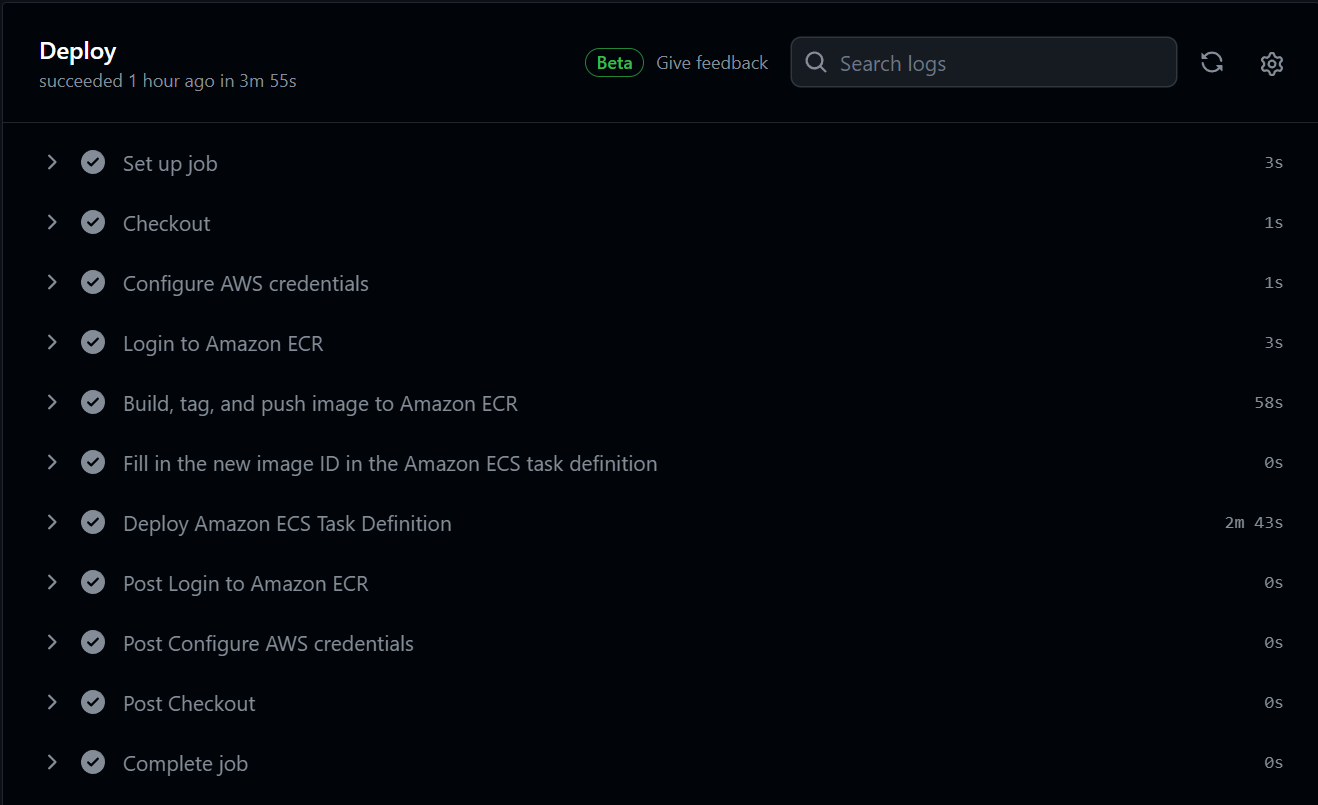
task-definition: ${{ steps.task-def.outputs.task-definition }}

service: ${{ env.ECS\_SERVICE }}

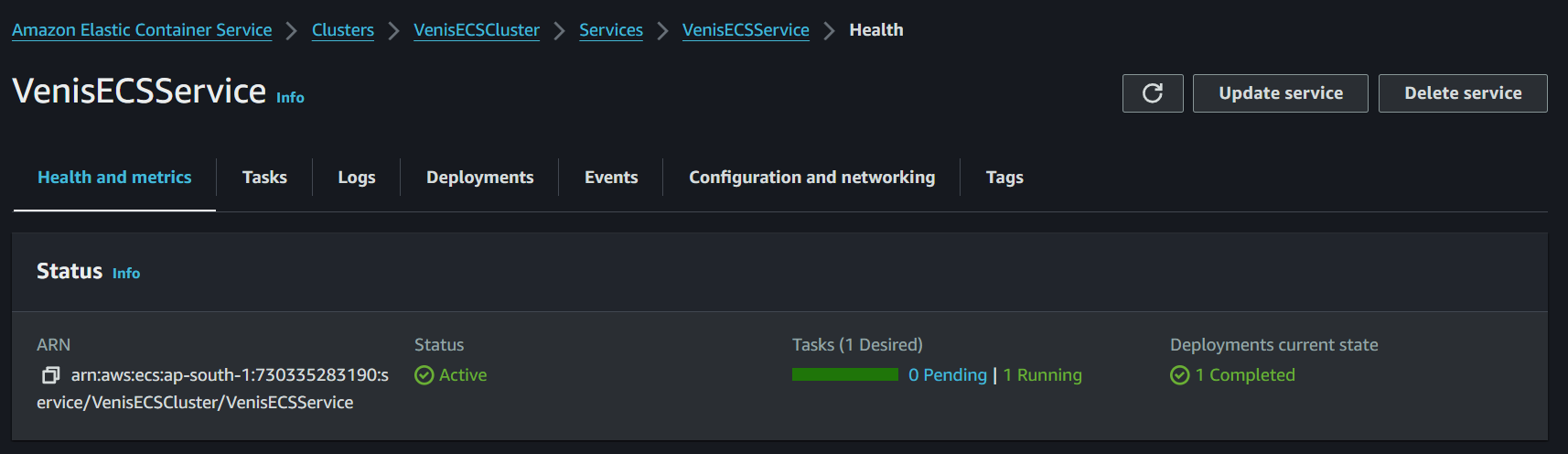
cluster: ${{ env.ECS\_CLUSTER }}

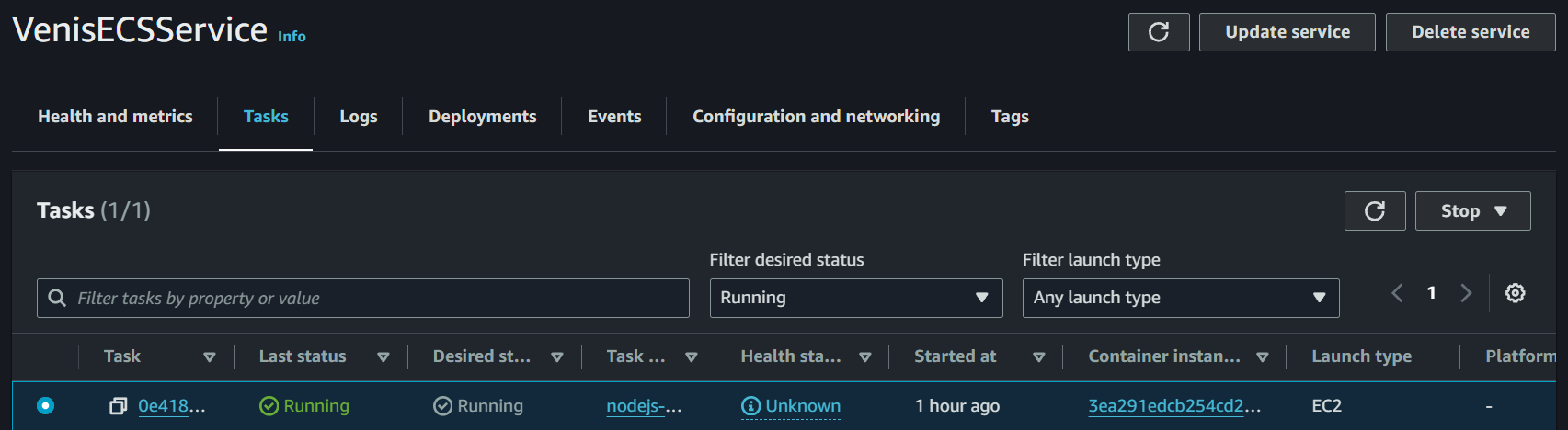
wait-for-service-stability: true

1. Now go to GitHub Actions and here we can see that our workflows has been successfully executed.



1. Now go to running service and here we can see that it is running a task.





1. Now go to Task Configuration and Now open the url on browser as publicIP:8000. Here we can see that Our Nodejs application is successfully running on ECS.

