# 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

Evaluation Criteria:

1. Successful setup of GitHub Actions workflow. - 10 points
2. Secure handling of sensitive information using GitHub Secrets. - 10 points
3. Successful building and pushing of the Docker image to Amazon ECR. - 10 points
4. Successful deployment of the Docker image to Amazon ECS. - 10 points
5. Proper error handling and documentation in the workflow. - 20 points

Step 1: Create a Git Repository

- Create a git repository and give name to repository.
- Repo name: **GitHub_Action_Assignmet2**

Step 2: Move Existing Application:

- Move your existing Node.js todo list application into the root directory of your Git repository.

Step 3: Create Dockerfile:

- Inside your repository directory, create a file named Dockerfile.
- Define the Docker image configuration.

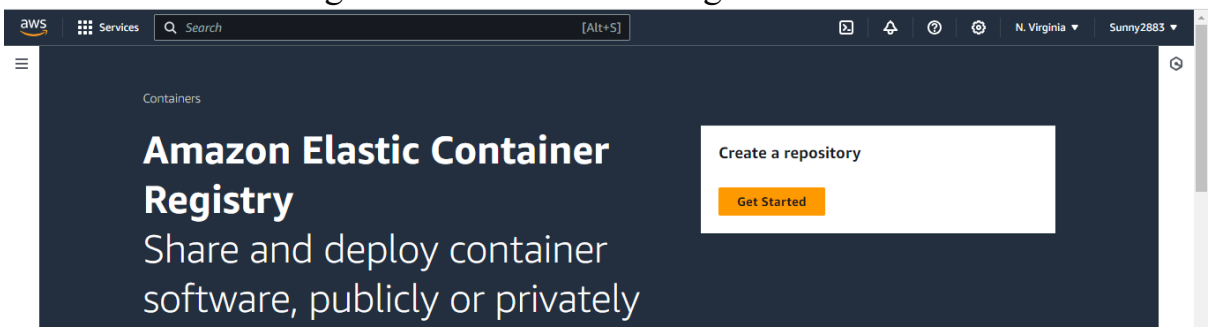
Dockerfile > ...

```
FROM node:14-alpine
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 3000
CMD ["node", "server.js"]
```

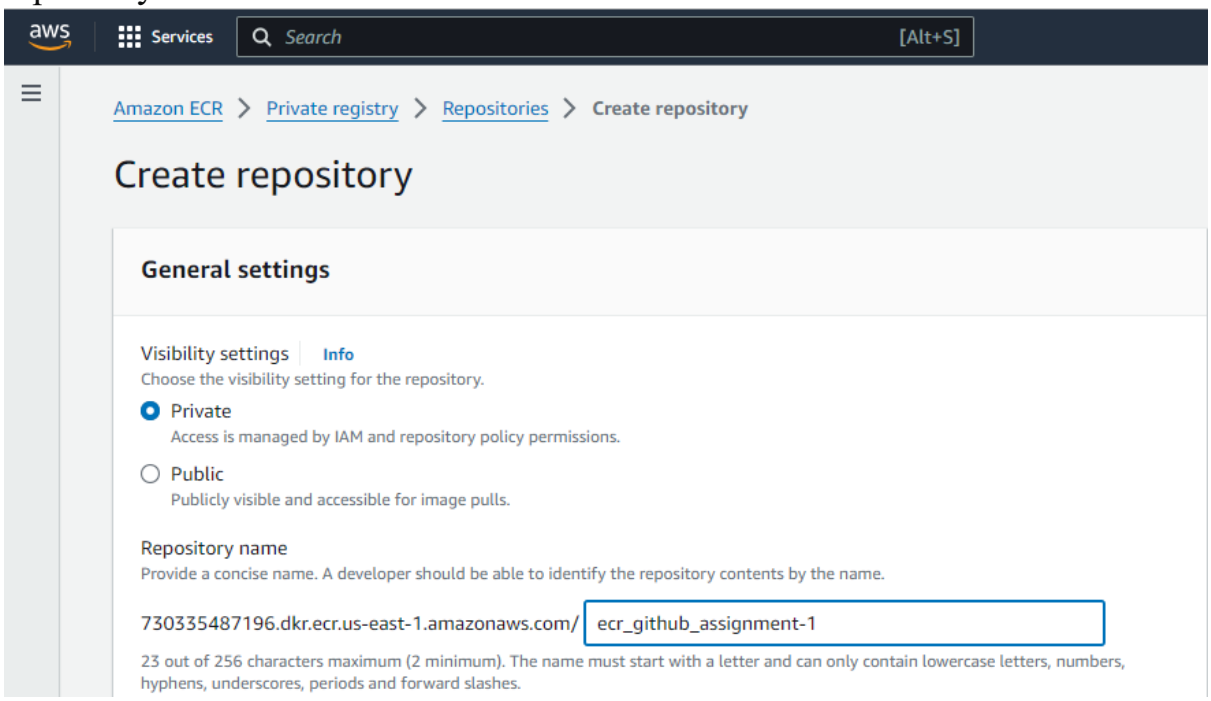
AWS Setup:

Step 1: Create Amazon ECR Repository:

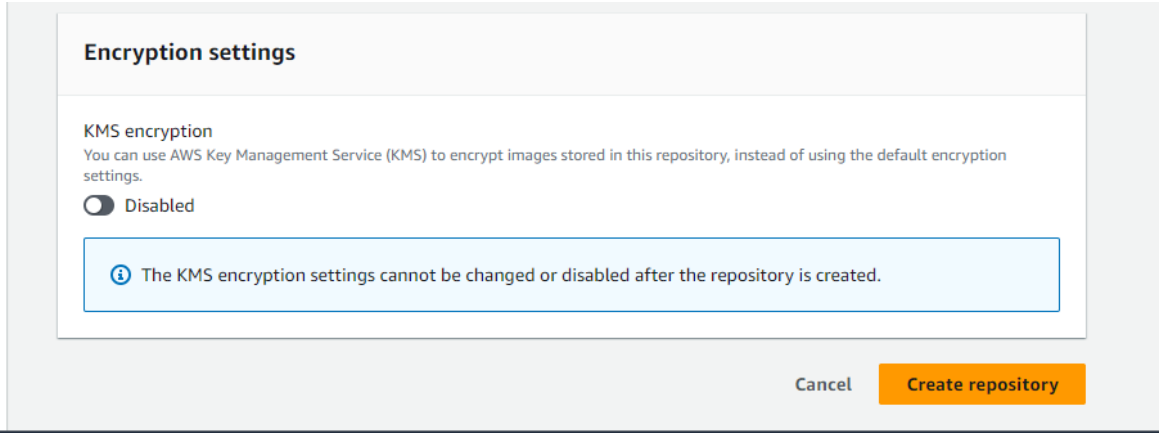
- Go to the AWS Management Console and navigate to Amazon ECR.

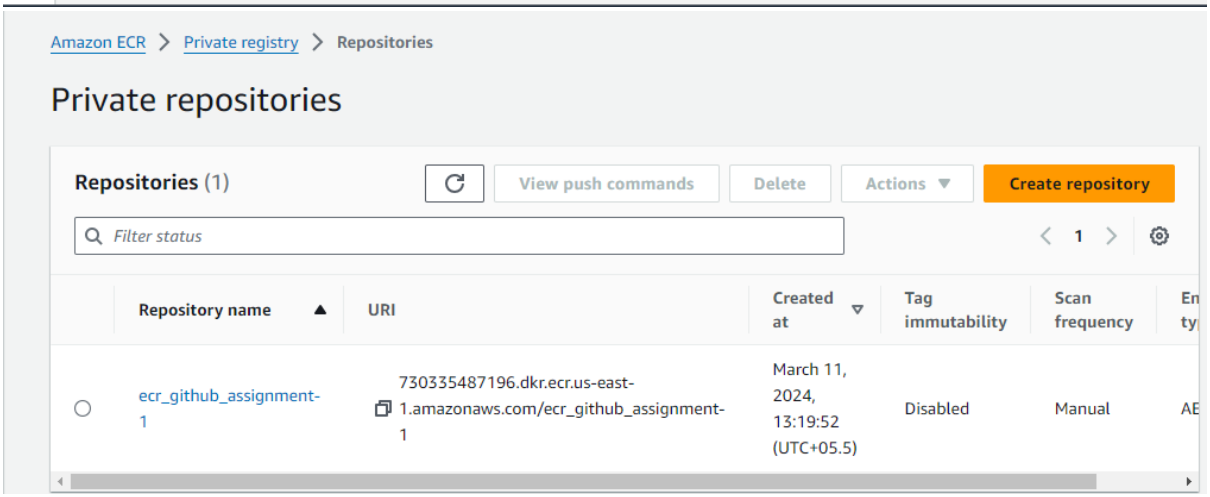



- Click on the "Create repository" button and Enter a name for your repository.



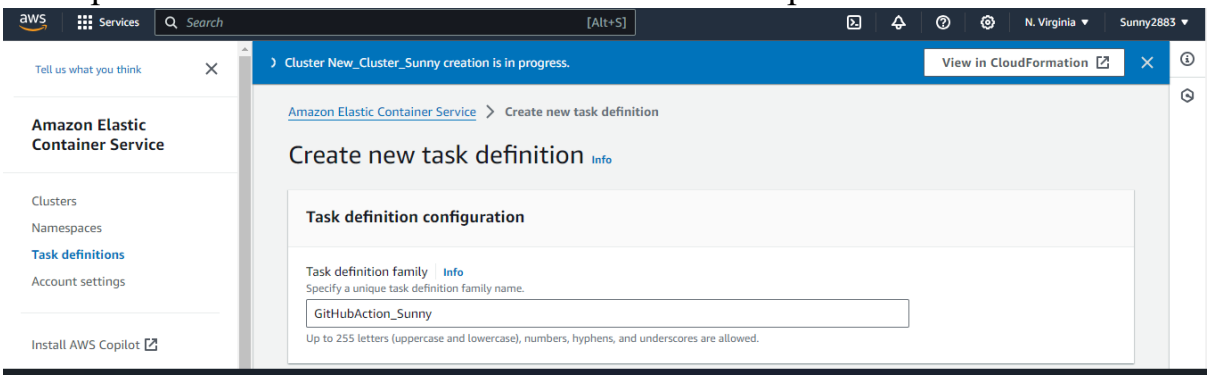
- Optionally, add tags or configure permissions and click on the "Create repository" button to create the repository.



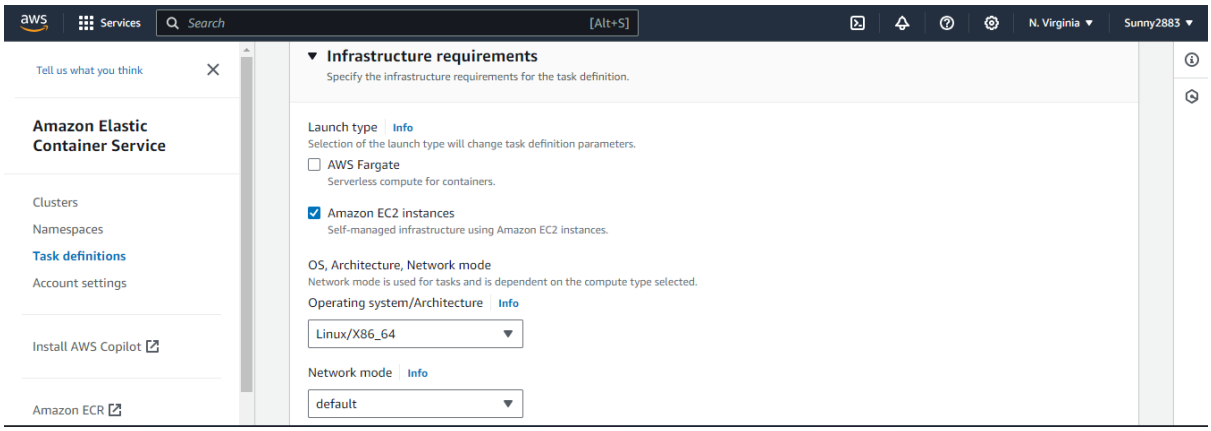

Step 2: Define ECS Task Definition:

- In the AWS Management Console, search for "ECS" or find it under the "Compute" section and click on "Amazon ECS" to open the service.

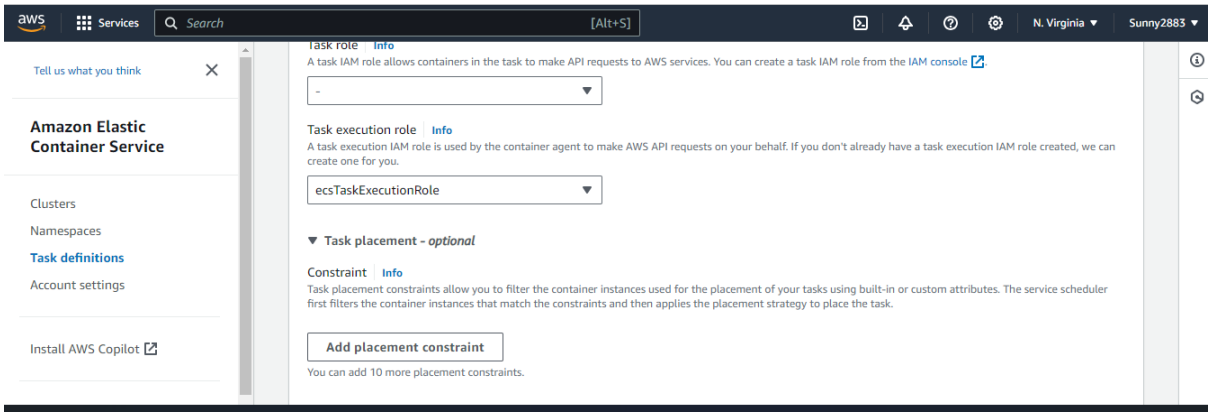



- Configure the task definition with container details, including image, CPU, memory, ports, environment variables, etc.

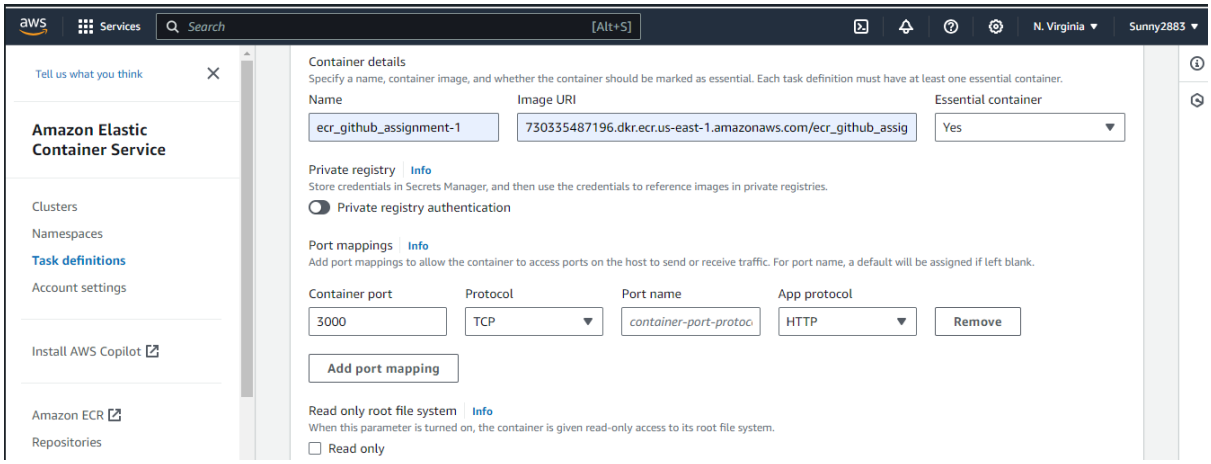
- Select Launch Type: Amazon EC2 Instances.
- OS: Linux
- Network : default



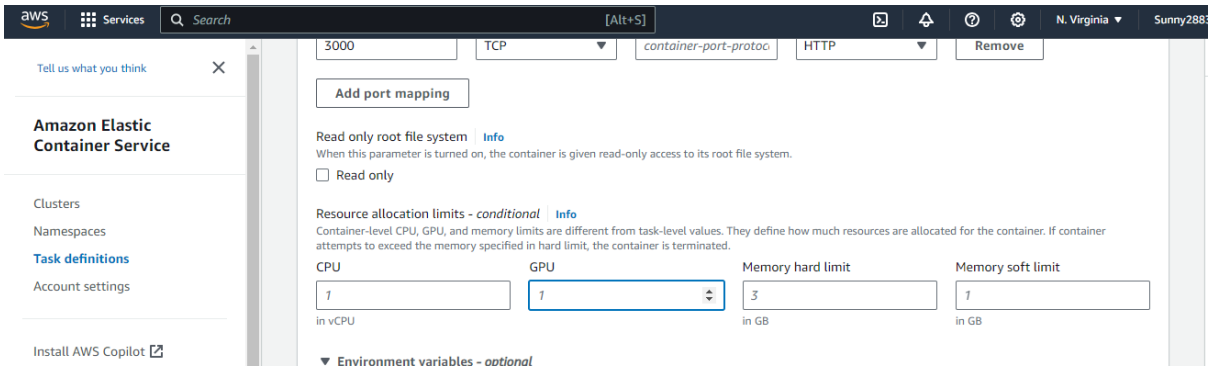

- Task Execution Role: ecsTaskExecutionRole.




- **Container Details:**
- Enter a name for your containe , Specify the Docker image to use for the container .
- **Name: ecr_github_assignment-1**
- **Image URI: 730335487196.dkr.ecr.us-east-1.amazonaws.com/ecr_github_assignment-1:latest**
- **Port mapping:**
- **Container port:3000**
- **Protocol: 3000**
- **App protocol: Http**

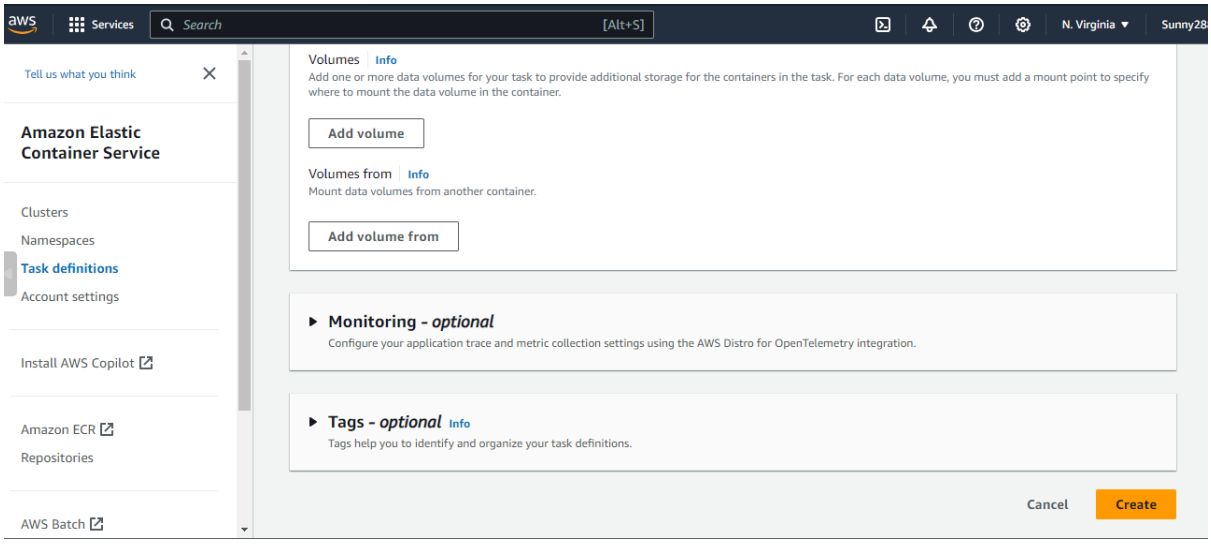



- Specify the maximum amount of memory the container can use in MiB etc.
- **CPU: 1**
- **GPU: 1**
- **Memory hard limit: 3**
- **Memory soft limit: 1**



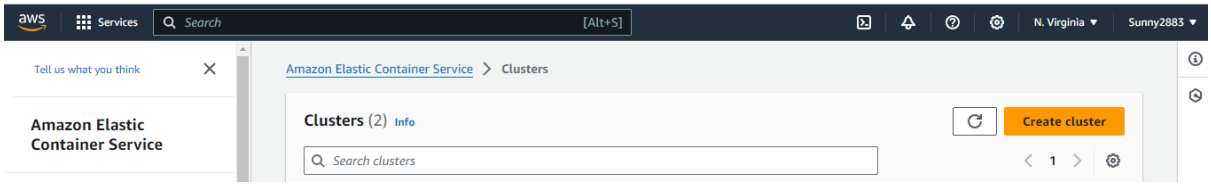

Step 3: Review and Create:

- Review the task definition configuration and click on the "Create" button to create the task definition.

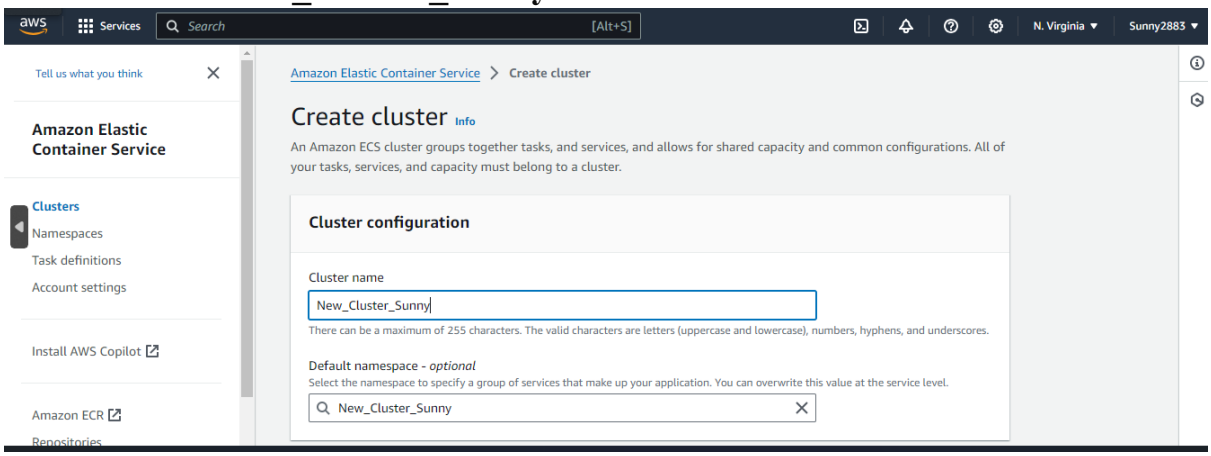


Step 4: Create Cluster

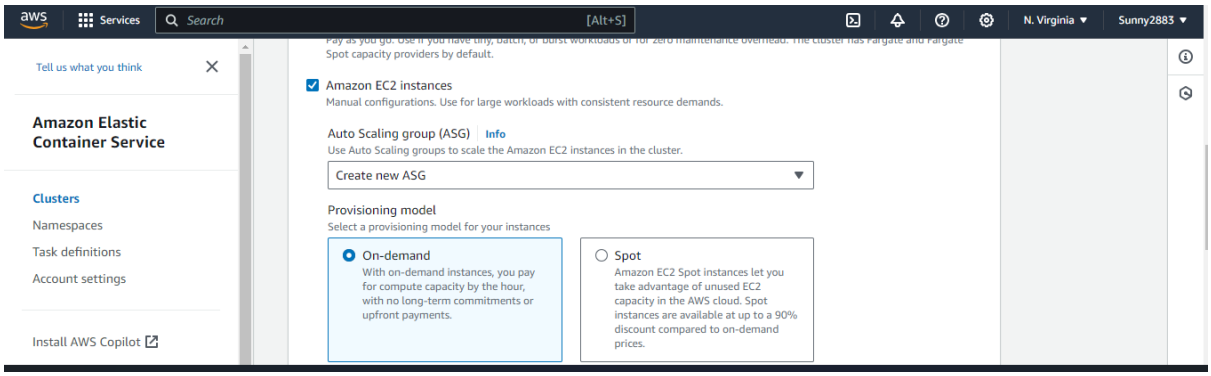
- In the AWS Management Console, search for "ECS" or find it under the "Compute" section and click on "Clusters" in the left sidebar.



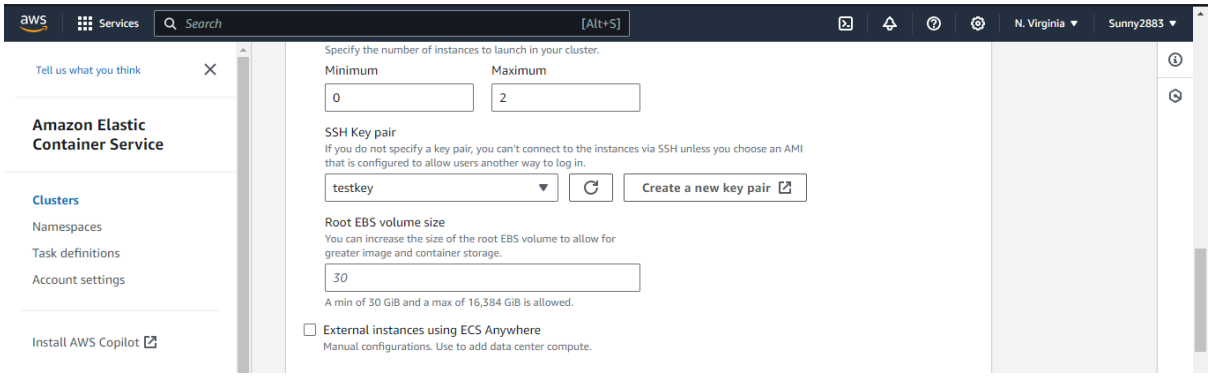

- Provide a name for your cluster in the "Cluster name" field.
- Cluster name: **New_Cluster_Sunny.**



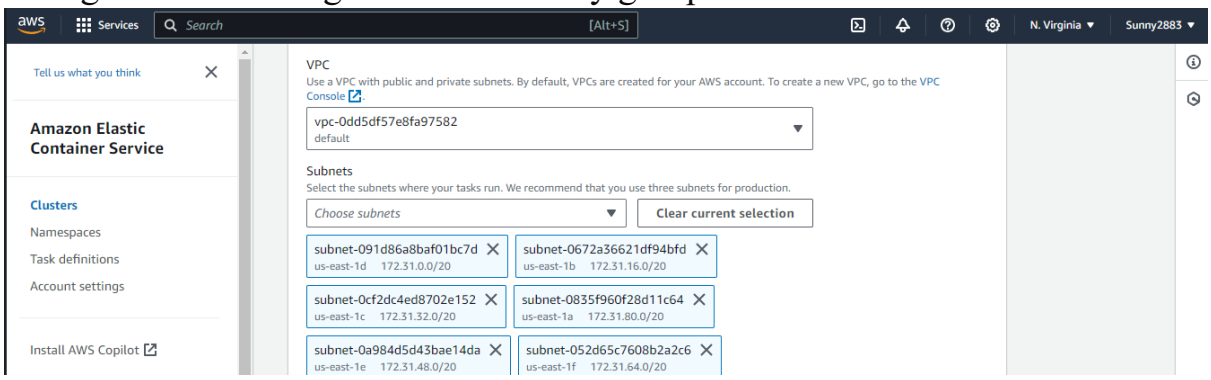

- Select Cluster Template:
- Amazon EC2 Instances:




- configure other settings such as cluster capacity, instance type, key pair.
- Capacity **: minimum-0, maximum-2**
- Key-pair**; testkey**

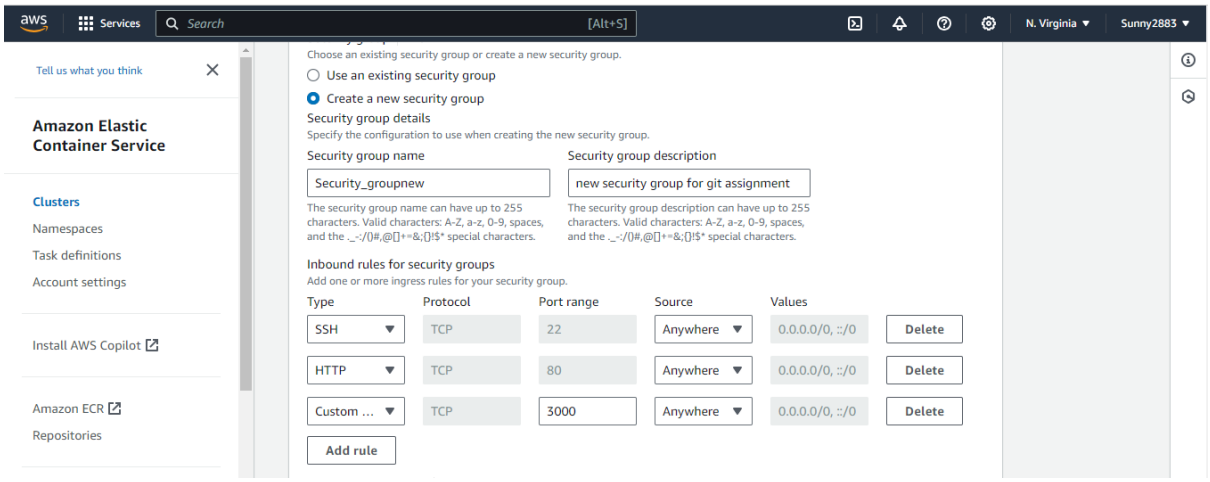



- configure other settings such as Security group VPC.

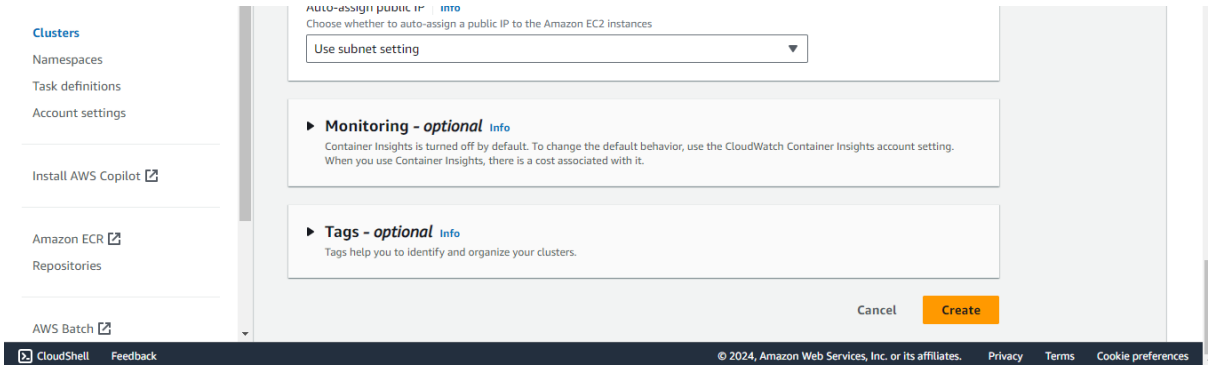



Select inbound rules for security group:

| Type | Protocol | Port Range | Source Values |
|---|---|---|---|
| SSH | TCP | 22 | Anywhere |
| HTTP | TCP | 80 | Anywhere |
| Custom | TCP | 3000 | Anywhere |

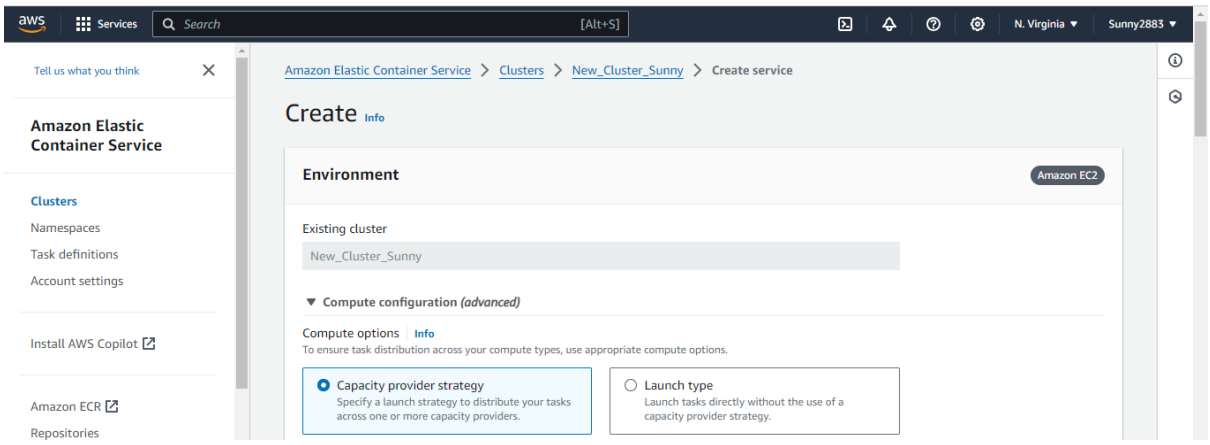



Review the configuration to ensure it meets your requirements and click on the "Create" button to create the cluster.

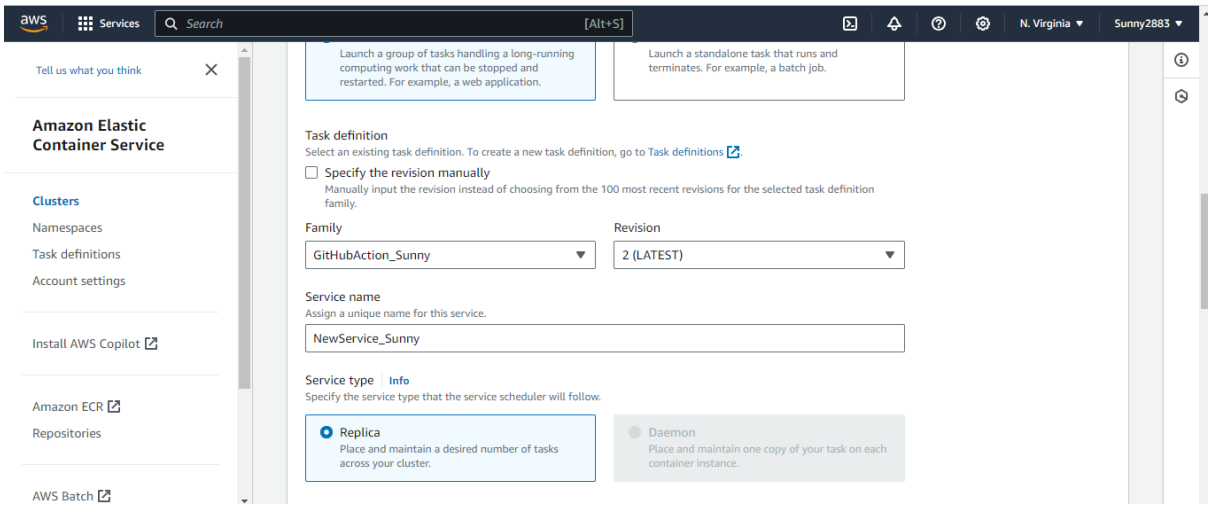



Step 5: Create Service:

- Select the cluster where you want to deploy your task definition:

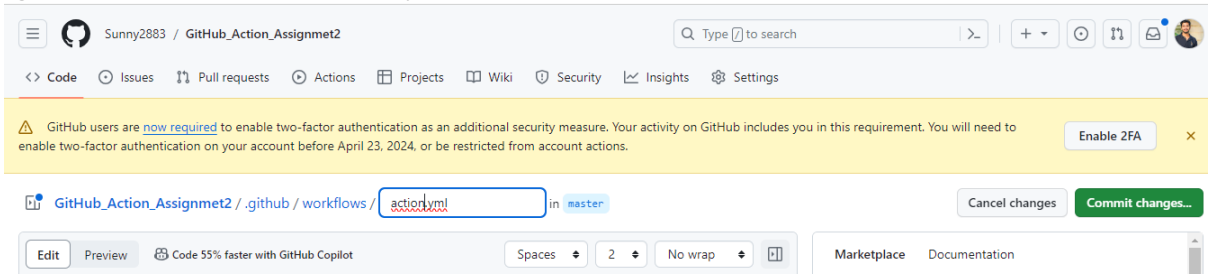



- Give family and service name for service:




Step 6: Create Workflow File:

- Inside your repository, navigate to the .github/workflows directory and Create a new file with a .yml extension (action.yml) in the .github/workflows directory.



Set GitHub Secrets:

Add GitHub Secrets for the identified sensitive information, ensuring secure handling.

| Name | Last updated |
|---|---|
| AWS_ACCESS_KEY_ID | yesterday |
| AWS_REGION | yesterday |
| AWS_SECRET_ACCESS_KEY | yesterday |
| DOCKER_PASSWORD | yesterday |
| DOCKER_USERNAME | yesterday |
| REPOSITORY_NAME | yesterday |

Define Workflow:

action.yml 3 × | Dockerfile M | GitHubActionTask-revision1.json

.github > workflows > action.yml

```yaml

name: Deploy node.js application on ecs

on:
  push:
    branches:
      - "master"
  pull_request:
    branches: ["master"]
  workflow_dispatch:

env:
  AWS_REGION: us-east-1
  ECR_REPOSITORY: ecr_github_assignment-1
  ECS_SERVICE: NewService_Sunny
  ECS_CLUSTER: New_Cluster_Sunny
  ECS_TASK_DEFINITION: GitHubActionTask-revision1.json

  CONTAINER_NAME: ecr_github_assignment-1


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

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

      - name: Configure AWS credentials
```

action.yml 3 × Dockerfile M GitHubActionTask-revision1.json

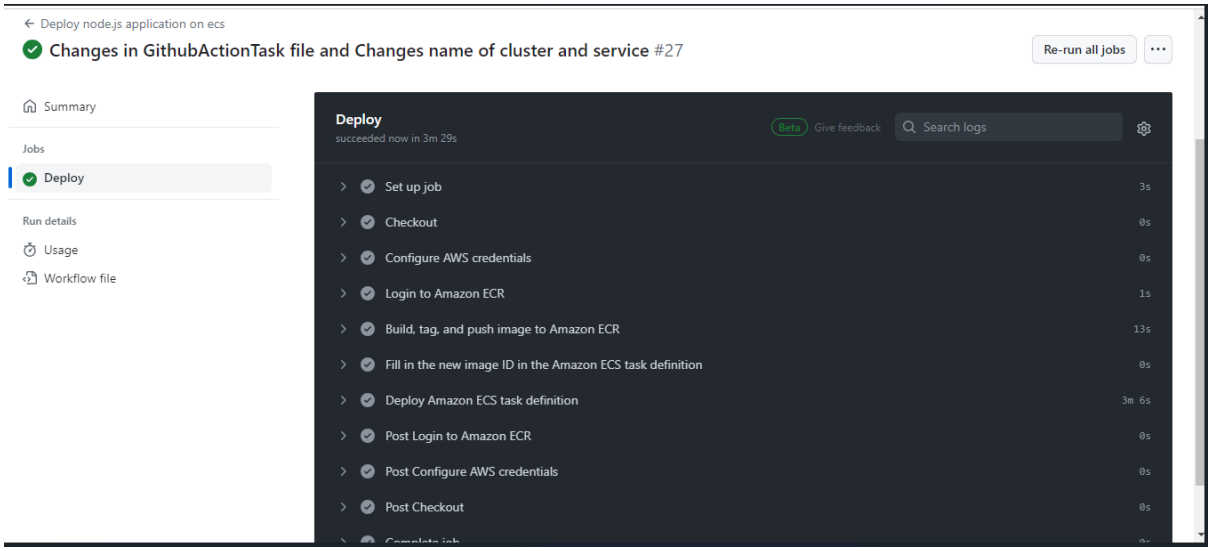
github > workflows > action.yml

```yaml
jobs:
  deploy:
    steps:
      - name: Configure AWS credentials
        with:
          aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
          aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
          aws-region: ${{ secrets.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: latest
        run: |

          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 }}
```

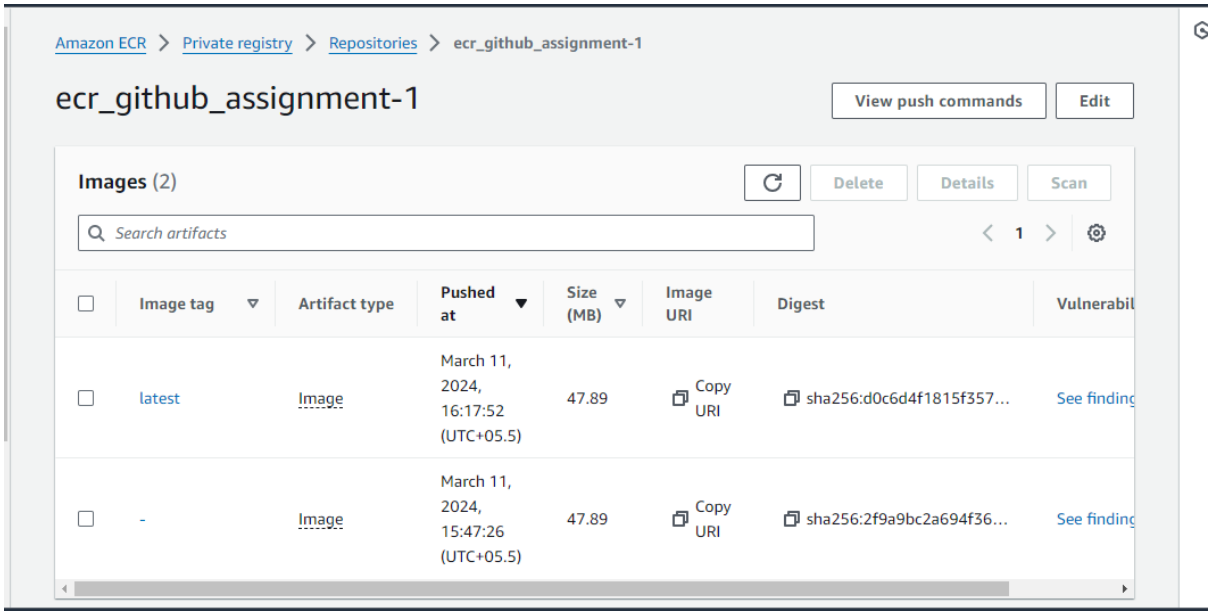
```yaml
      - name: Deploy Amazon ECS task definition
        uses: aws-actions/amazon-ecs-deploy-task-definition@df9643053eda01f169e64a0e60233aacca83799a
        with:
          task-definition: ${{ steps.task-def.outputs.task-definition }}
          service: ${{ env.ECS_SERVICE }}
          cluster: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

Commit Changes:

- Save the changes to the YAML file.
- Commit and push the changes to your GitHub repository.

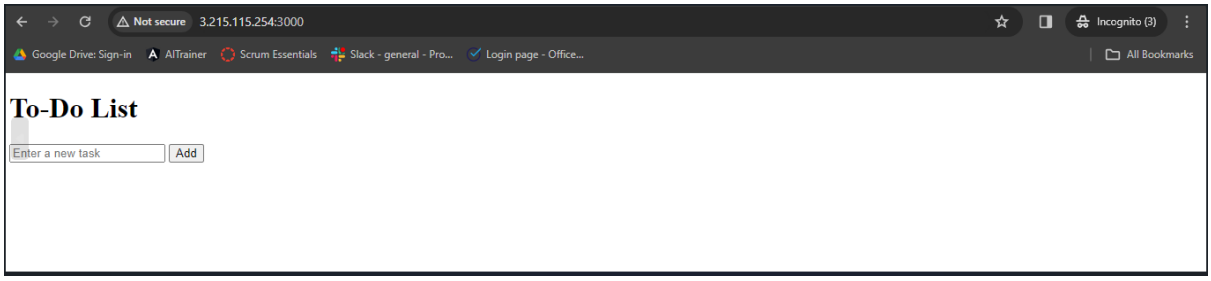



Review the changes.




Address to access the application:

http://3.215.115.254:3000/




# Thank You