

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

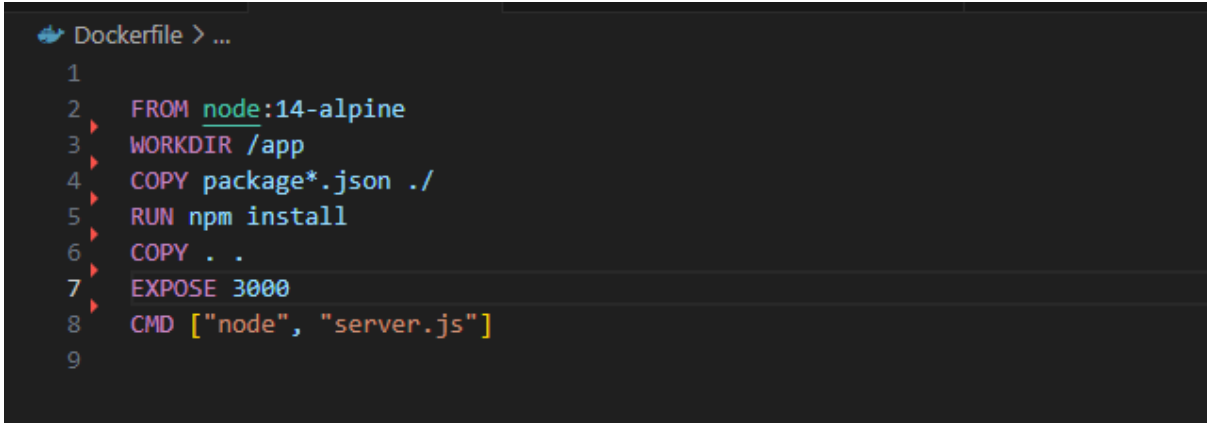
- Create a git repository and give name to repository.
- Repo name: **GitHub_Action_Assignmet2**

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.

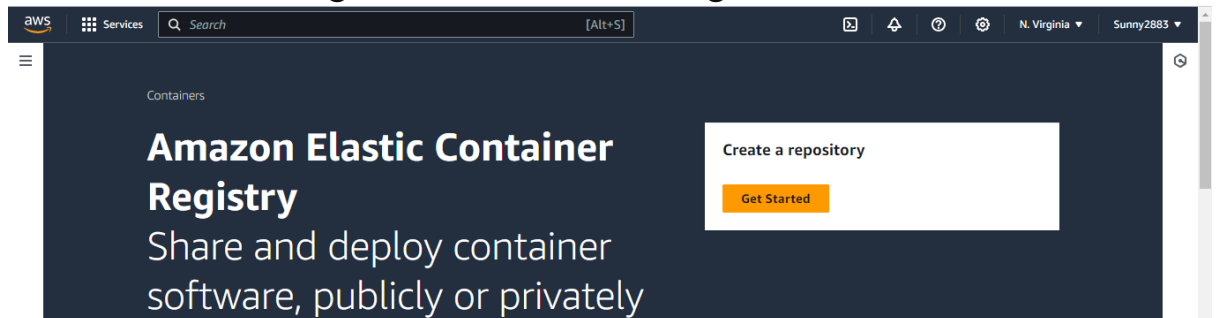
A screenshot of a code editor showing a Dockerfile. The editor has a dark background with light-colored text. The title bar of the editor shows 'Dockerfile > ...'. The code is as follows:

```
1
2 FROM node:14-alpine
3 WORKDIR /app
4 COPY package*.json ./
5 RUN npm install
6 COPY . .
7 EXPOSE 3000
8 CMD ["node", "server.js"]
9
```

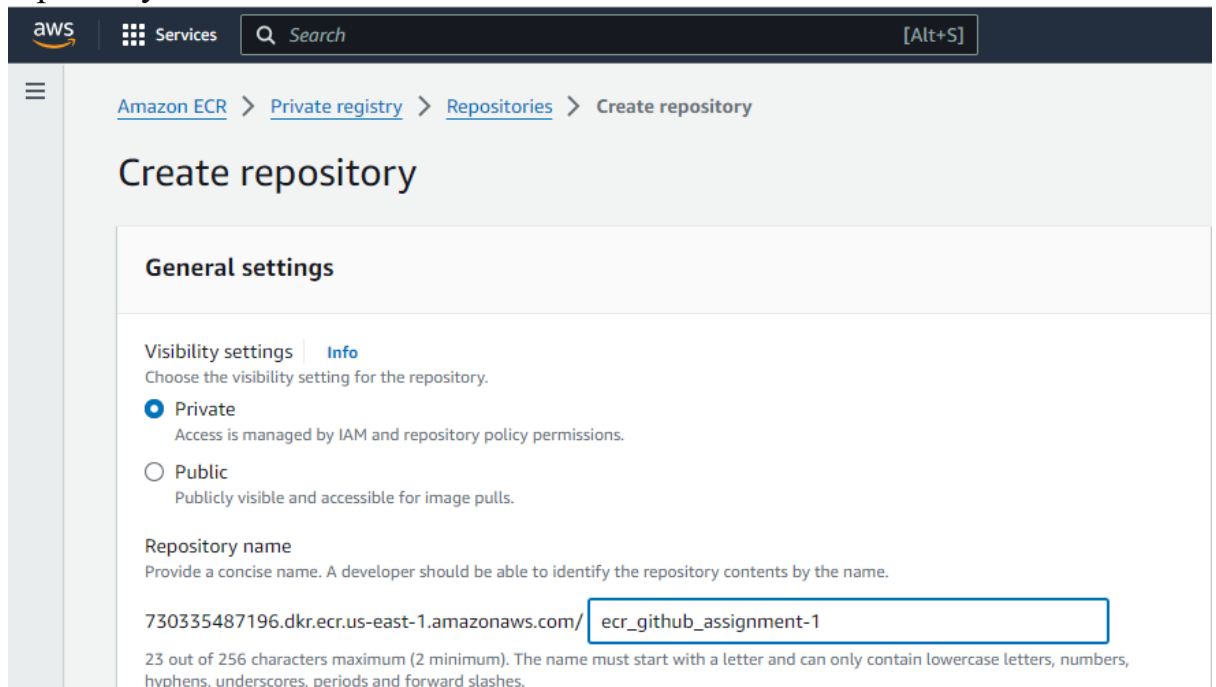
AWS Setup:

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.

The image shows two parts of the AWS ECR console. The top part is the 'Encryption settings' dialog, which has a section for 'KMS encryption' with a note: 'You can use AWS Key Management Service (KMS) to encrypt images stored in this repository, instead of using the default encryption settings.' The 'KMS encryption' toggle is currently 'Disabled'. A blue information box states: 'The KMS encryption settings cannot be changed or disabled after the repository is created.' At the bottom right of the dialog are 'Cancel' and 'Create repository' buttons.

The bottom part is the 'Private repositories' page. It shows a breadcrumb trail: 'Amazon ECR > Private registry > Repositories'. The page title is 'Private repositories'. Below the title is a 'Repositories (1)' section with buttons for 'View push commands', 'Delete', 'Actions', and 'Create repository'. There is a search bar labeled 'Filter status'. Below this is a table with the following columns: 'Repository name', 'URI', 'Created at', 'Tag immutability', 'Scan frequency', and 'Encryption type'. The table contains one entry:

Repository name	URI	Created at	Tag immutability	Scan frequency	Encryption type
ecr_github_assignment-1	730335487196.dkr.ecr.us-east-1.amazonaws.com/ecr_github_assignment-1	March 11, 2024, 13:19:52 (UTC+05.5)	Disabled	Manual	AE

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.

The image shows the AWS Management Console interface. The top navigation bar includes the AWS logo, 'Services', a search bar, and the current region 'N. Virginia' and account 'Sunny2883'. The left sidebar shows the 'Amazon Elastic Container Service' section with links to 'Clusters', 'Namespaces', 'Task definitions', and 'Account settings'. The main content area is titled 'Create new task definition' and includes a sub-header 'Task definition configuration'. Under 'Task definition family', there is a text input field containing 'GitHubAction_Sunny' and a note: 'Specify a unique task definition family name. Up to 255 letters (uppercase and lowercase), numbers, hyphens, and underscores are allowed.'

- 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

The screenshot shows the AWS console interface for configuring infrastructure requirements. The left sidebar contains the 'Amazon Elastic Container Service' menu with options like Clusters, Namespaces, Task definitions, Account settings, Install AWS Copilot, and Amazon ECR. The main content area is titled 'Infrastructure requirements' and includes a sub-header 'Specify the infrastructure requirements for the task definition.' Under 'Launch type', 'AWS Fargate' is unchecked and 'Amazon EC2 instances' is checked. The 'Operating system/Architecture' dropdown is set to 'Linux/X86_64' and the 'Network mode' dropdown is set to 'default'.

- Task Execution Role: ecsTaskExecutionRole.

The screenshot shows the 'Task execution role' configuration page. It includes a dropdown for 'Task role' (currently empty) and a dropdown for 'Task execution role' set to 'ecsTaskExecutionRole'. Below this, the 'Task placement - optional' section is visible, containing a 'Constraint' dropdown and an 'Add placement constraint' button.

- **Container Details:**
- Enter a name for your container , 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**

Container details
Specify a name, container image, and whether the container should be marked as essential. Each task definition must have at least one essential container.

Name: Image URI: Essential container:

Private registry [Info](#)
Store credentials in Secrets Manager, and then use the credentials to reference images in private registries.
☐ Private registry authentication

Port mappings [Info](#)
Add port mappings to allow the container to access ports on the host to send or receive traffic. For port name, a default will be assigned if left blank.

Container port	Protocol	Port name	App protocol	
<input type="text" value="3000"/>	<input type="text" value="TCP"/>	<input type="text" value="container-port-protocol"/>	<input type="text" value="HTTP"/>	<input type="button" value="Remove"/>

Read only root file system [Info](#)
When this parameter is turned on, the container is given read-only access to its root file system.
☐ Read only

- 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**

Read only root file system [Info](#)
When this parameter is turned on, the container is given read-only access to its root file system.
☐ Read only

Resource allocation limits - conditional [Info](#)
Container-level CPU, GPU, and memory limits are different from task-level values. They define how much resources are allocated for the container. If container attempts to exceed the memory specified in hard limit, the container is terminated.

CPU	GPU	Memory hard limit	Memory soft limit
<input type="text" value="1"/> in vCPU	<input type="text" value="1"/>	<input type="text" value="3"/> in GB	<input type="text" value="1"/> in GB

▼ **Environment variables - optional**

Step 3: Review and Create:

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

Volumes [Info](#)
Add one or more data volumes for your task to provide additional storage for the containers in the task. For each data volume, you must add a mount point to specify where to mount the data volume in the container.

Volumes from [Info](#)
Mount data volumes from another container.

► **Monitoring - optional**
Configure your application trace and metric collection settings using the AWS Distro for OpenTelemetry integration.

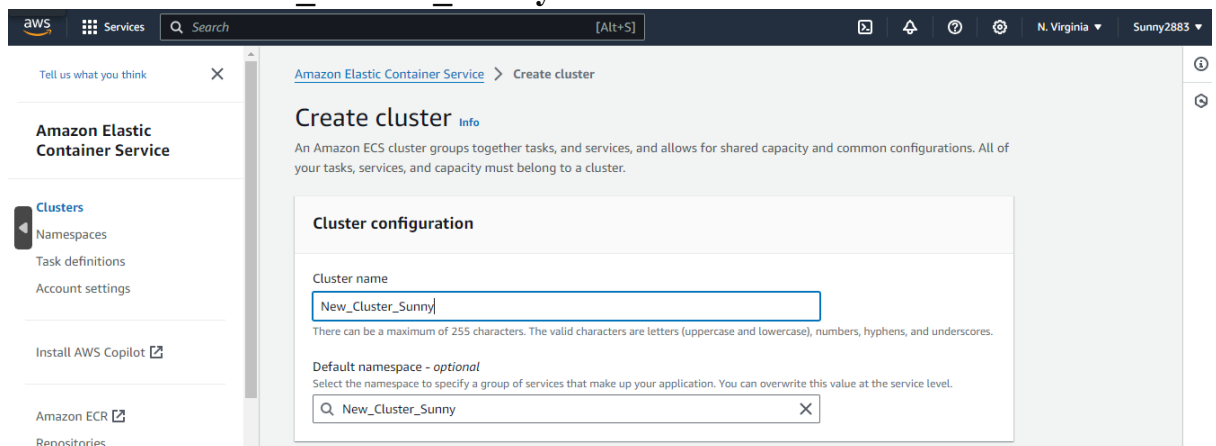
► **Tags - optional** [Info](#)
Tags help you to identify and organize your task definitions.

Step 4: Create Cluster

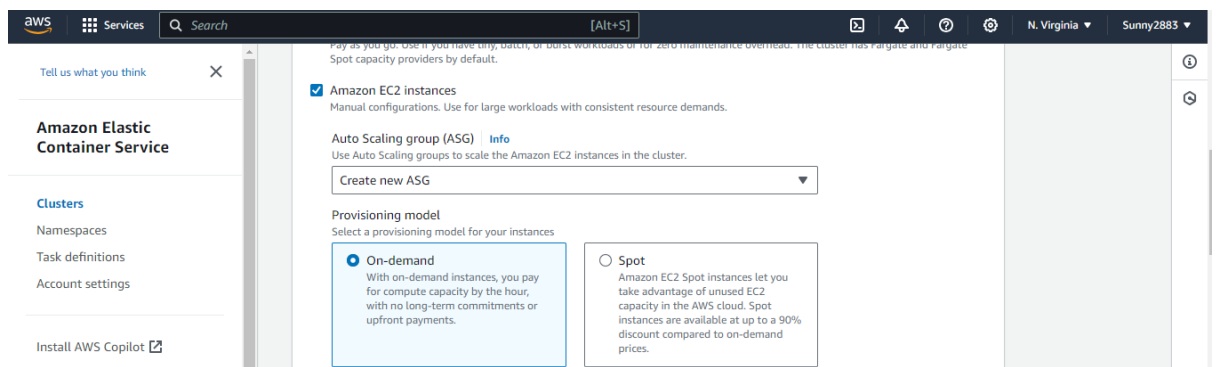
- In the AWS Management Console, search for "ECS" or find it under the "Compute" section and click on "Clusters" in the left sidebar.



- 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**

Specify the number of instances to launch in your cluster.

Minimum: 0 Maximum: 2

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.

testkey [Create a new key pair](#)

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

30
A min of 30 GiB and a max of 16,384 GiB is allowed.

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

- configure other settings such as Security group VPC.

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](#).

vpc-0dd5df57e8fa97582
default

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

Choose subnets [Clear current selection](#)

subnet-091d86a8baf01bc7d us-east-1d 172.31.0.0/20
subnet-0672a36621df94bfd us-east-1b 172.31.16.0/20
subnet-0cf2dc4ed8702e152 us-east-1c 172.31.32.0/20
subnet-0835f960f28d11c64 us-east-1a 172.31.80.0/20
subnet-0a984d5d43bae14da us-east-1e 172.31.48.0/20
subnet-052d65c7608b2a2c6 us-east-1f 172.31.64.0/20

Select inbound rules for security group:

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

Choose an existing security group or create a new security group.

☐ Use an existing security group

☒ Create a new security group

Security group details
Specify the configuration to use when creating the new security group.

Security group name:

Security group description:

The security group name can have up to 255 characters. Valid characters: A-Z, a-z, 0-9, spaces, and the `._-:/!@#%&*~'` special characters.

The security group description can have up to 255 characters. Valid characters: A-Z, a-z, 0-9, spaces, and the `._-:/!@#%&*~'` special characters.

Inbound rules for security groups
Add one or more ingress rules for your security group.

Type	Protocol	Port range	Source	Values	
SSH	TCP	22	Anywhere	0.0.0.0/0, ::/0	Delete
HTTP	TCP	80	Anywhere	0.0.0.0/0, ::/0	Delete
Custom ...	TCP	3000	Anywhere	0.0.0.0/0, ::/0	Delete

[Add rule](#)

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

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

Monitoring - optional [info](#)
Container Insights is turned off by default. To change the default behavior, use the CloudWatch Container Insights account setting. When you use Container Insights, there is a cost associated with it.

Tags - optional [info](#)
Tags help you to identify and organize your clusters.

[Cancel](#) [Create](#)

Step 5: Create Service:

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

Amazon Elastic Container Service > Clusters > New_Cluster_Sunny > Create service

Create [info](#)

Environment Amazon EC2

Existing cluster

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.

- Give family and service name for service:

Amazon Elastic Container Service

Clusters

- Namespaces
- Task definitions
- Account settings
- Install AWS Copilot
- Amazon ECR
- Repositories
- AWS Batch

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
GitHubAction_Sunny

Revision
2 (LATEST)

Service name
Assign a unique name for this service.
NewService_Sunny

Service type [Info](#)
Specify the 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.

Step 6: Create Workflow File:

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

Sunny2883 / GitHub_Action_Assignmet2

Q Type to search

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

⚠ GitHub users are [now required](#) to enable two-factor authentication as an additional security measure. Your activity on GitHub includes you in this requirement. You will need to enable two-factor authentication on your account before April 23, 2024, or be restricted from account actions. [Enable 2FA](#)




















GitHub_Action_Assignmet2 / .github / workflows / action.yaml in master

Cancel changes [Commit changes...](#)

Edit Preview Code 55% faster with GitHub Copilot Spaces 2 No wrap Marketplace Documentation

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  

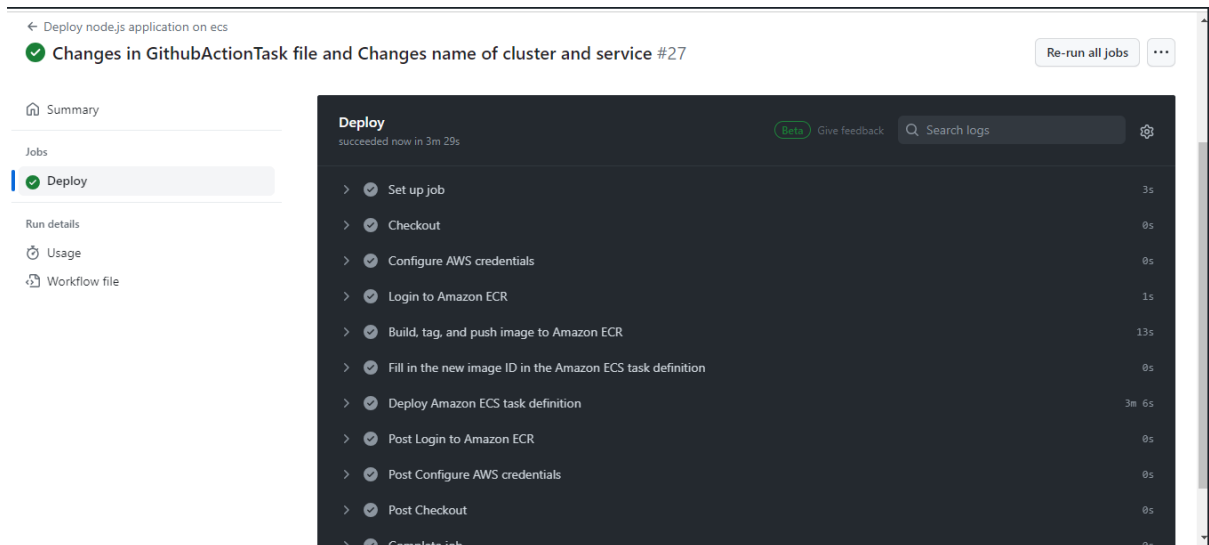
Define Workflow:

```
.github > workflows > action.yml
1
2 name: Deploy node.js application on ecs
3
4 on:
5   push:
6     branches:
7       - "master"
8   pull_request:
9     branches: ["master"]
10  workflow_dispatch:
11
12 env:
13   AWS_REGION: us-east-1
14   ECR_REPOSITORY: ecr_github_assignment-1
15   ECS_SERVICE: NewService_Sunny
16   ECS_CLUSTER: New_Cluster_Sunny
17   ECS_TASK_DEFINITION: GitHubActionTask-revision1.json
18
19   CONTAINER_NAME: ecr_github_assignment-1
20
21
22 jobs:
23   deploy:
24     name: Deploy
25     runs-on: ubuntu-latest
26     environment: production
27
28     steps:
29       - name: Checkout
30         uses: actions/checkout@v4
31
32       - name: Configure AWS credentials
```

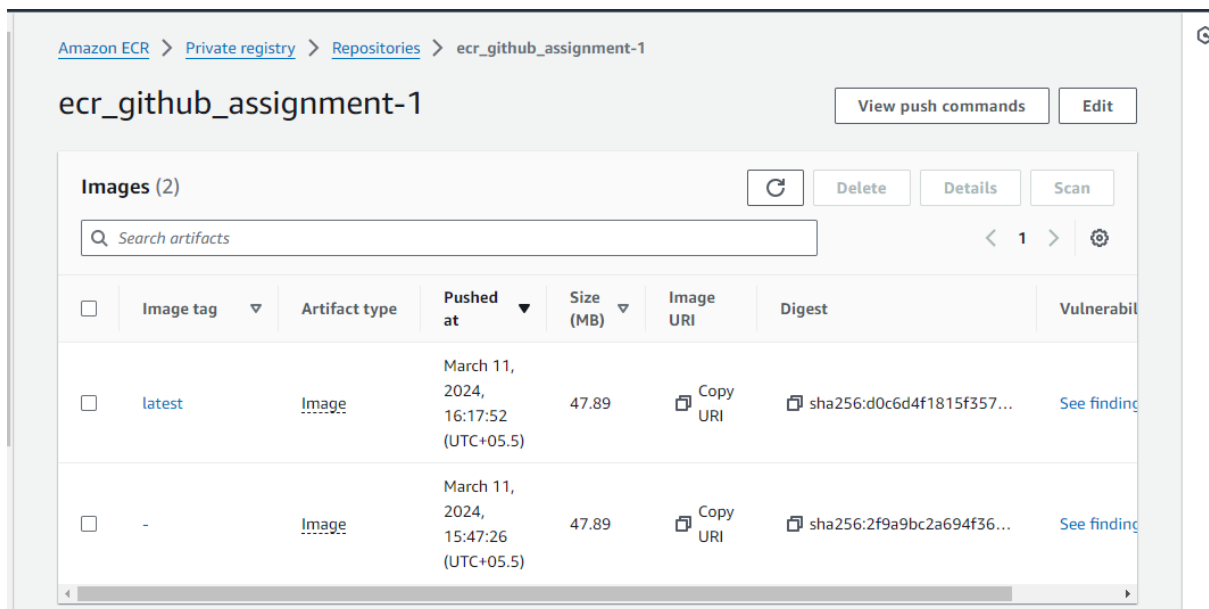
```
action.yml 3 x Dockerfile M {} GitHubActionTask-revision1.json
github > workflows > action.yml
22 jobs:
23   deploy:
24     steps:
32     - name: Configure AWS credentials
33       with:
34         aws-access-key-id: ${ secrets.AWS_ACCESS_KEY_ID }
35         aws-secret-access-key: ${ secrets.AWS_SECRET_ACCESS_KEY }
36         aws-region: ${ secrets.AWS_REGION }
37
38
39     - name: Login to Amazon ECR
40       id: login-ecr
41       uses: aws-actions/amazon-ecr-login@62f4f872db3836360b72999f4b87f1ff13310f3a
42
43     - name: Build, tag, and push image to Amazon ECR
44       id: build-image
45       env:
46         ECR_REGISTRY: ${ steps.login-ecr.outputs.registry }
47         IMAGE_TAG: latest
48       run: |
49
50         docker build -t $ECR_REGISTRY/$ECR_REPOSITORY:$IMAGE_TAG .
51         docker push $ECR_REGISTRY/$ECR_REPOSITORY:$IMAGE_TAG
52         echo "image=$ECR_REGISTRY/$ECR_REPOSITORY:$IMAGE_TAG" >> $GITHUB_OUTPUT
53
54     - name: Fill in the new image ID in the Amazon ECS task definition
55       id: task-def
56       uses: aws-actions/amazon-ecs-render-task-definition@804dfbdd57f713b6c079302a4c01db7017a36fc
57       with:
58         task-definition: ${ env.ECS_TASK_DEFINITION }
59         container-name: ${ env.CONTAINER_NAME }
60         image: ${ steps.build-image.outputs.image }
61
62
63     - name: Deploy Amazon ECS task definition
64       uses: aws-actions/amazon-ecs-deploy-task-definition@df9643053eda01f169e64a0e6023aacca83799a
65       with:
66         task-definition: ${ steps.task-def.outputs.task-definition }
67         service: ${ env.ECS_SERVICE }
68         cluster: ${ env.ECS_CLUSTER }
69         wait-for-service-stability: true
```

Commit Changes:

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

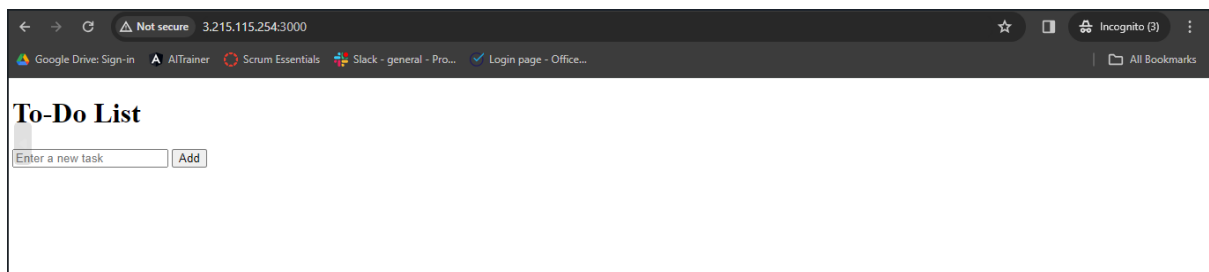


Review the changes.



Address to access the application:

<http://3.215.115.254:3000/>



Thank You