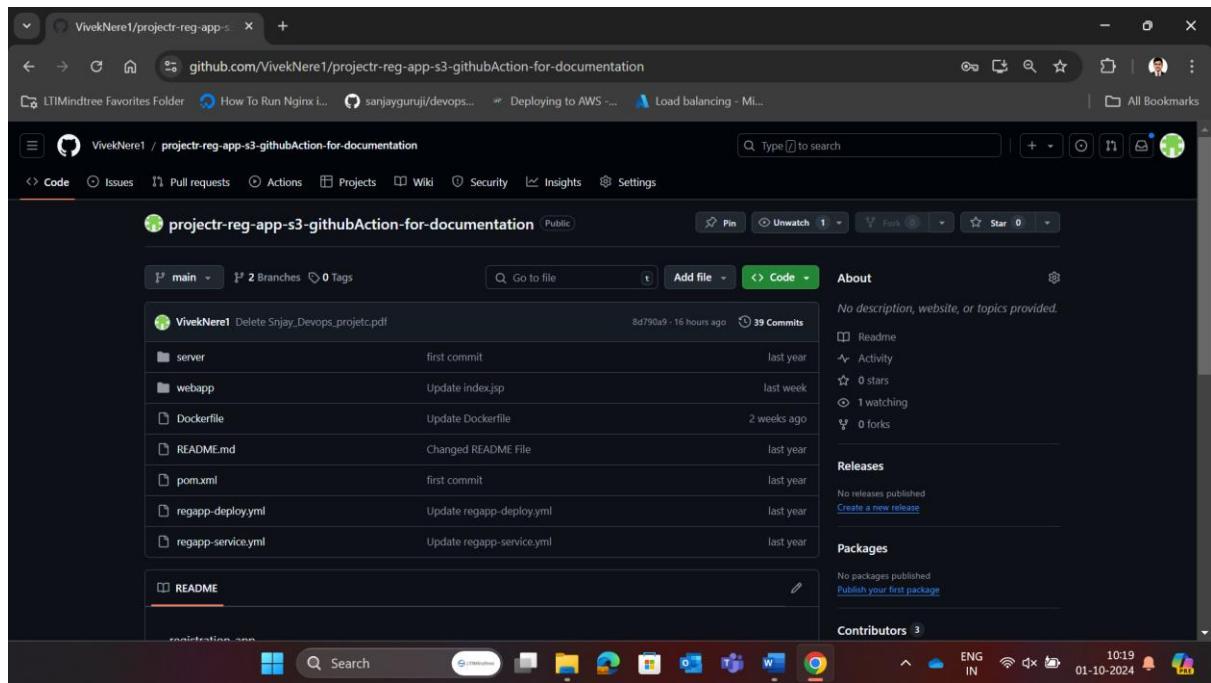
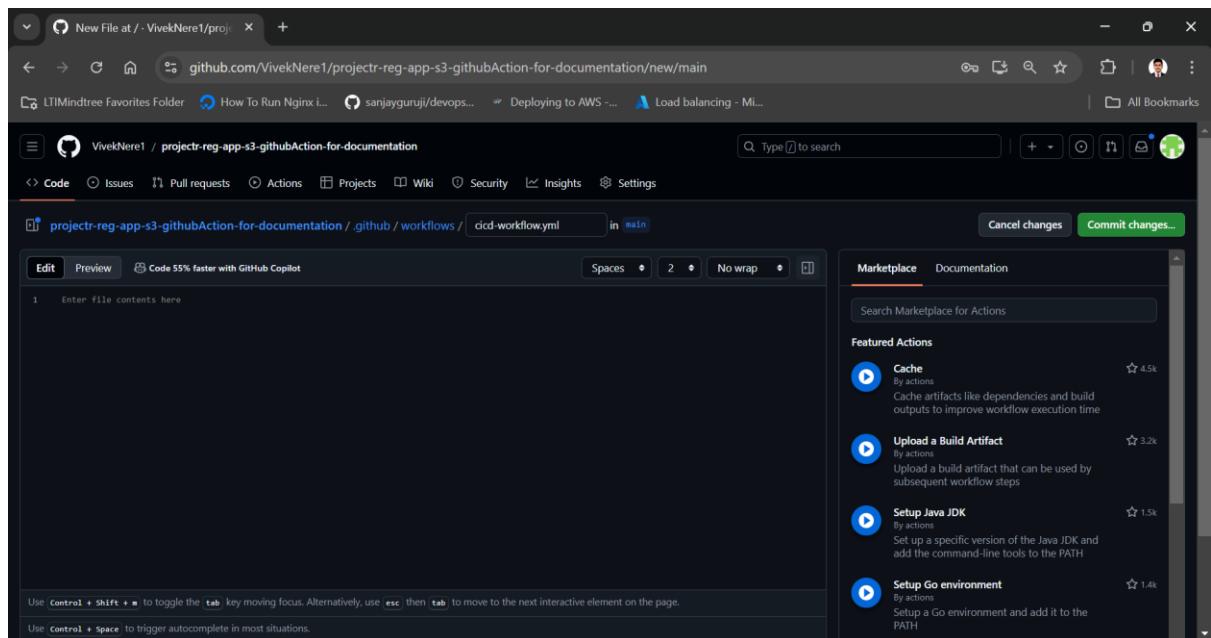


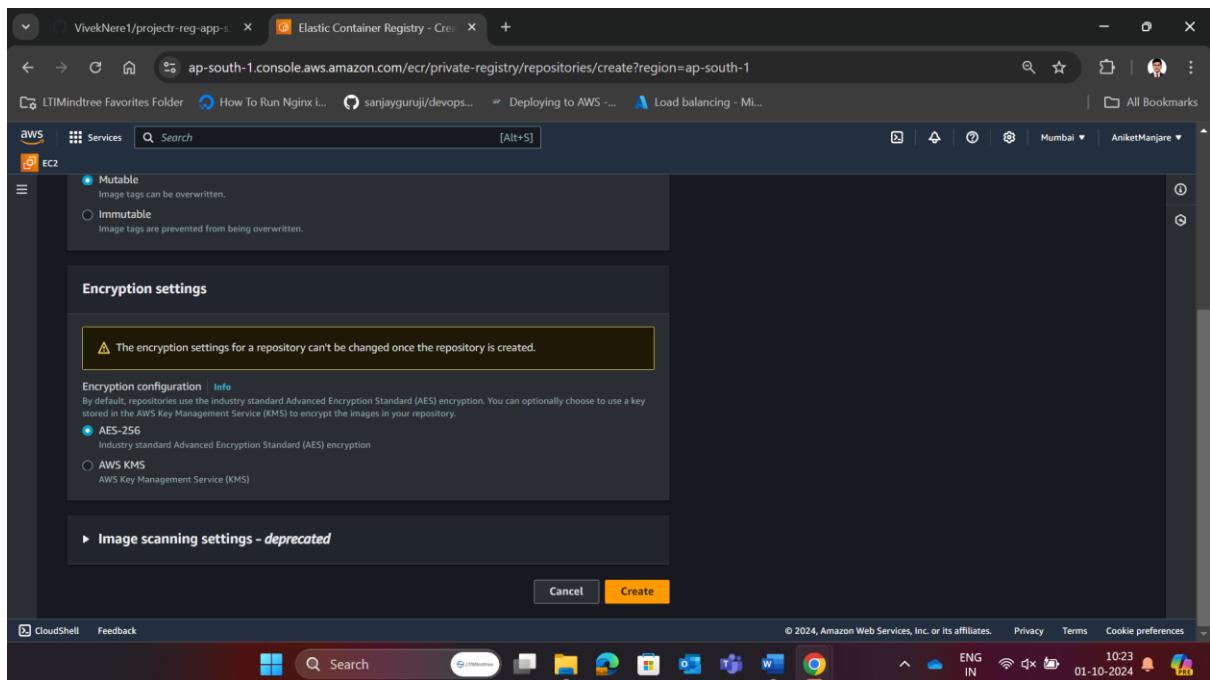
## Create one repository which consists of java project



## Create one workflow file



## Create one ECR :



The screenshot shows the AWS Elastic Container Registry (ECR) interface. The left sidebar navigation includes 'Private registry' (Repositories, Summary, Images, Permissions, Lifecycle Policy, Repository tags, Features & Settings), 'Public registry' (Repositories, Settings), 'ECR public gallery' (Amazon ECS, Amazon EKS), and 'CloudShell'.

The main content area displays the 'reg-app-repository' under 'Amazon ECR > Private registry > Repositories > reg-app-repository'. The 'Images (0)' section shows a search bar and a table header with columns: Image tag, Artifact type, Pushed at, Size (MB), Image URI, and Digest. A message 'No images' and 'No images to display' is present.

Below this, another browser tab is open with the URL [ap-south-1.console.aws.amazon.com/ecr/private-registry/repositories/create?region=ap-south-1](https://ap-south-1.console.aws.amazon.com/ecr/private-registry/repositories/create?region=ap-south-1), showing the 'Create private repository' form. The 'General settings' section contains fields for 'Repository name' (set to '084375547362.dkr.ecr.ap-south-1.amazonaws.com/reg-app-repository') and 'Image tag mutability' (set to 'Mutable'). The 'Encryption settings' section is partially visible.

Create on ecs

The screenshot shows the 'Create cluster' wizard in the AWS Elastic Container Service (ECS) console. The 'Cluster configuration' step is displayed, where the user has specified the cluster name as 'DevCluster-reg-app-1-oct' and the default namespace as 'DevCluster-reg-app-1-oct'. The 'Infrastructure' section is set to 'Serverless' mode, using AWS Fargate as the capacity provider. The 'Monitoring - optional' section indicates Container Insights is turned off by default. The 'Encryption - optional' section allows choosing KMS keys for storage encryption. The 'Tags - optional' section provides a field for adding tags to the cluster. A 'Create' button is visible at the bottom right.

The screenshot shows the AWS Elastic Container Service (ECS) Cluster overview page. The cluster 'DevCluster-reg-app-1-oct' is selected. Key details displayed include:

- ARN:** arn:aws:ecs:ap-south-1:08437554736:cluster/DevCluster-reg-app-1-oct
- Status:** Provisioning
- CloudWatch monitoring:** -
- Registered container instances:** -
- Services:** Draining (Active)
- Tasks:** Pending (Running)
- Encryption:** Managed storage (Fargate ephemeral storage)

The sidebar on the left includes links for Clusters, Namespaces, Task definitions, Account settings, and various AWS services like ECR and Batch.

Now we have to create one task definition:

The screenshot shows the 'Create new task definition' page in the AWS Elastic Container Service (ECS). The task definition family is set to 'reg-app-task-definition-1-oct'. The infrastructure requirements section includes:

- Launch type:** AWS Fargate (selected)
- AWS Fargate: Serverless compute-for containers.
- Amazon EC2 Instances: Self-managed infrastructure using Amazon EC2 instances.

The sidebar on the left shows the 'Task definitions' option is selected.

Create one role for ecs task execution with `AmazonEC2ContainerRegistryFullAccess` and `AmazonECSTaskExecutionRolePolicy`

**Select trusted entity**

**Trusted entity type**

- AWS service**  
Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- AWS account**  
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- Web identity**  
Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
- SAML 2.0 federation**  
Allows users federated with SAML 2.0 from a corporate directory to perform actions in this account.

**Service or use case**

Elastic Container Service

Choose a use case for the specified service.

Use case

- Elastic Container Service**  
Allows ECS to create and manage AWS resources on your behalf.
- Elastic Container Service Autoscale**  
Allows Auto Scaling to access and update ECS services.
- Elastic Container Service Task**  
Allows ECS tasks to call AWS services on your behalf.
- EC2 Role for Elastic Container Service**  
Allows EC2 instances in an ECS cluster to access ECS.

**Cancel** **Next**

The screenshot shows the AWS IAM 'Create role' wizard in progress, specifically Step 2: Add permissions. The user is selecting permissions for a new role.

**Screenshot 1 (Top):** The user has selected the 'AmazonEC2ContainerRegistryFullAccess' policy. The policy details are visible:

Policy name	Type	Description
AmazonEC2ContainerRegis...	AWS managed	Provides administrative access to Ama...

**Screenshot 2 (Bottom):** The user has selected the 'AmazonECSTaskExecutionRolePolicy' policy. The policy details are visible:

Policy name	Type	Description
AmazonECSTaskEx...	AWS managed	Provides access to other AWS service r...

In both screenshots, the 'Set permissions boundary - optional' section is visible at the bottom of the permissions list.

The screenshot shows the 'Create role' wizard in the AWS IAM console. The title bar says 'Name, review, and create'. The left sidebar lists 'Step 1 Select trusted entity', 'Step 2 Add permissions', and 'Step 3 Name, review, and create'. The main area is titled 'Role details' with fields for 'Role name' (containing 'ecsTaskExecutionRole1OCT') and 'Description' (containing 'Allows ECS tasks to call AWS services on your behalf').

The screenshot shows the 'Create role' wizard in the AWS IAM console, specifically Step 2: Add permissions. It displays a trust policy JSON code:

```
1 [ { 2     "Version": "2012-10-17", 3     "Statement": [ 4         { 5             "Sid": "", 6             "Effect": "Allow", 7             "Principal": { 8                 "Service": [ 9                     "ecs-tasks.amazonaws.com" 10                ] 11            }, 12            "Action": "sts:AssumeRole" 13        } 14    ] 15 } ]
```

The screenshot shows the 'Add tags - optional' step of the 'Create role' wizard. It displays two existing tags: 'AmazonEC2ContainerRegistryFullAccess' and 'AmazonECSTaskExecutionRolePolicy', both labeled as 'AWS managed' and 'Permissions policy'. Below this, there is a section titled 'Step 3: Add tags' with a sub-section 'Add tags - optional'. It says 'No tags associated with the resource.' and has a button 'Add new tag'. A note states 'You can add up to 50 more tags.' At the bottom right are 'Cancel', 'Previous', and 'Create role' buttons.

The screenshot shows the 'Roles (21)' page in the AWS IAM console. A search bar at the top shows 'ecsTaskExecutionRole1OCT'. The results list shows one item: 'ecsTaskExecutionRole1OCT' under 'Role name', with 'AWS Service: ecs-tasks' listed under 'Trusted entities'. Below this, there are sections for 'Roles Anywhere' and 'Temporary credentials'. At the bottom right are 'Manage' and 'Delete' buttons.

Now attach that role in task definition

VivekNere1/project-reg-app-s Create task definition | Elastic Roles | IAM | Global Elastic Container Registry - Im... - +

ap-south-1.console.aws.amazon.com/ecs/v2/create-task-definition?region=ap-south-1

LTMindtree Favorites Folder How To Run Nginx ... sanjayguruji/devops... Deploying to AWS ... Load balancing - Mi... All Bookmarks

aws Services Search [Alt+S]

EC2

Amazon Elastic Container Service

Clusters Namespaces Task definitions Account settings

Install AWS Copilot

Amazon ECR Repositories

AWS Batch

Documentation Discover products Subscriptions

CloudShell Feedback

Self-managed infrastructure using Amazon EC2 Instances.

OS, Architecture, Network mode

Network mode is used for tasks and is dependent on the compute type selected.

Operating system/Architecture info Network mode info

Linux/X86\_64 AWS Fargate

Task size info

Specify the amount of CPU and memory to reserve for your task.

CPU Memory

1 vCPU 3 GB

Task roles - conditional

Task role info

A task IAM role allows containers in the task to make API requests to AWS services. You can create a task IAM role from the IAM console.

ecsTaskExecutionRole1OCT

Task execution role info

A task execution IAM role is used by the container agent to make AWS API requests on your behalf. If you don't already have a task execution IAM role created, we can create one for you.

ecsTaskExecutionRole1OCT

Task placement - optional

Task placement constraints are not supported for AWS Fargate launch type.

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

ENG IN 10:30 01-10-2024

VivekNere1/project-reg-app-s Create task definition | Elastic Roles | IAM | Global Elastic Container Registry - Priv... - +

ap-south-1.console.aws.amazon.com/ecs/v2/create-task-definition?region=ap-south-1

LTMindtree Favorites Folder How To Run Nginx ... sanjayguruji/devops... Deploying to AWS ... Load balancing - Mi... All Bookmarks

aws Services Search [Alt+S]

EC2

Amazon Elastic Container Service

Clusters Namespaces Task definitions Account settings

Install AWS Copilot

Amazon ECR Repositories

AWS Batch

Documentation Discover products Subscriptions

CloudShell Feedback

Container - 1 info

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

reg-app-1-oct 084375547362.dkr.ecr.ap-south-1.amazonaws.com/reg-app-repository Yes

Up to 255 letters (uppercase and lowercase), numbers, hyphens, underscores, colons, periods, forward slashes, and number signs are allowed.

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
80	TCP	container-port-protocol	HTTP

Add port mapping

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

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

ENG IN 10:31 01-10-2024

The screenshot shows the AWS Cloud Console interface for creating a task definition in the Amazon Elastic Container Service (ECS). The left sidebar lists various AWS services, with 'Task definitions' selected under the ECS category. The main content area is titled 'Create task definition' and includes the following sections:

- Resource allocation limits - conditional**: CPU (1 vCPU), GPU (1), Memory hard limit (3 GB), Memory soft limit (1 GB).
- Environment variables - optional**: Environment variables (Info), Add individually, Add from file.
- Logging - optional**: Log driver (awslogs), Log group (arn:aws:logs:ap-south-1:123456789012:task-definition/test-task), Log stream (arn:aws:logs:ap-south-1:123456789012:log-group:/aws/lambda/test-task:task-12345678901234567890123456789012).
- CPU and memory allocation for a sidecar**: CPU (1 vCPU), GPU (1), Memory hard limit (3 GB), Memory soft limit (1 GB).

At the bottom right, there are 'Cancel' and 'Create' buttons. The status bar at the bottom indicates the date (01-10-2024) and time (10:32). The browser address bar shows the URL: ap-south-1.console.aws.amazon.com/ecs/v2/create-task-definition?region=ap-south-1.

The screenshot shows the AWS Cloud Console interface for the Amazon Elastic Container Service (ECS). The left sidebar is titled "Amazon Elastic Container Service" and includes options like Clusters, Namespaces, Task definitions, and Account settings. The main content area displays a success message: "Task definition successfully created" followed by "reg-app-task-definition-1-oct:1 has been successfully created. You can use this task definition to deploy a service or run a task." Below this, the navigation path is "Amazon Elastic Container Service > Task definitions > reg-app-task-definition-1-oct > Revision 1 > Containers". The task definition name is "reg-app-task-definition-1-oct:1". The "Overview" tab is selected, showing details such as ARN (arn:aws:ecs:ap-south-1:08437554:task-definition/reg-app-task-definition-1-oct:1), Status (ACTIVE), Time created (October 01, 2024 at 10:32 (UTC+5:30)), App environment (Fargate), Task role (ecsTaskExecutionRole1OCT), Task execution role (ecsTaskExecutionRole1OCT), Operating system/Architecture (Linux/X86\_64), and Network mode (awsvpc). Other tabs include JSON, Task placement, Volumes (0), Requires attributes, and Tags. At the bottom, there are buttons for Deploy, Actions, and Create new revision.

Created task definition successfully!

Now create one user for github action:

**Specify user details**

User name: **githubaction-user**

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = . @ \_ - (hyphen)

Provide user access to the AWS Management Console - *optional*

If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keypairs, you can generate them after you create this IAM user. [Learn more](#)

**Review and create**

Review your choices. After you create the user, you can view and download the autogenerated password, if enabled.

**User details**

User name githubaction-user	Console password type None	Require password reset No
--------------------------------	-------------------------------	------------------------------

**Permissions summary**

Name	Type	Used as
<a href="#">AdministratorAccess</a>	AWS managed - job function	Permissions policy
<a href="#">AmazonS3FullAccess</a>	AWS managed	Permissions policy

Now create access key

The screenshot shows the 'Create access key' wizard in the AWS IAM console. The user is on Step 2 - optional, specifically setting a description tag. A modal window titled 'Set description tag - optional' is open, showing a text input field with 'githubuser-access-key' entered. The input field has a placeholder 'Description tag value' and instructions: 'Describe the purpose of this access key and where it will be used. A good description will help you rotate this access key confidently later.' Below the input field, a note states: 'Maximum 256 characters. Allowed characters are letters, numbers, spaces representable in UTF-8, and: \_ . : / = + - @'. At the bottom of the modal are 'Cancel', 'Previous', and 'Create access key' buttons.

**Set description tag - optional**

The description for this access key will be attached to this user as a tag and shown alongside the access key.

Description tag value  
Describe the purpose of this access key and where it will be used. A good description will help you rotate this access key confidently later.

githubuser-access-key

Maximum 256 characters. Allowed characters are letters, numbers, spaces representable in UTF-8, and: \_ . : / = + - @

Cancel Previous Create access key

The screenshot shows the 'Create access key' wizard in the AWS IAM console. The user has completed Step 2 and is now on Step 3: Retrieve access keys. A green success message at the top states: 'Access key created. This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.' The main content area shows the newly created access key details:

Access key	Secret access key
AKIARHJJMVXREM2AXCHZ	I5LPtbh0gXZu0ujrRGcQfstvxnHe61HyPP2ID9HO <a href="#">Hide</a>

Access key best practices

Now we have to add that access key and secret key in GitHub secrets and variables

The screenshot shows a Microsoft Edge browser window with the GitHub Actions secrets creation page. The URL in the address bar is <https://github.com/VivekNere1/projectr-reg-app-s3-githubAction-for-documentation/settings/secrets/actions/new>. The page title is "Actions secrets / New secret". On the left, there's a sidebar with sections like General, Access, Collaborators, Moderation options, Code and automation (Branches, Tags, Rules, Actions, Webhooks, Environments, Codespaces, Pages), Security (Code security, Deploy keys), and Settings. The main form has a "Name" field set to "AWS\_ACCESS\_KEY\_ID" and a "Secret" field containing the value "AKIARHJIMVXREM2AXCHZ". A green "Add secret" button is located at the bottom right of the secret input field. The browser's taskbar at the bottom shows various pinned icons.

The screenshot shows two consecutive views of the GitHub Actions settings interface.

**Top View: Actions secrets / New secret**

- Left Sidebar:** Shows navigation links for General, Access, Collaborators, Moderation options, Code and automation (Branches, Tags, Rules, Actions, Webhooks, Environments, Codespaces, Pages), Security (Code security, Deploy keys), and Environment secrets.
- Right Content Area:** Title: Actions secrets / New secret. A form with:
  - Name: AWS\_SECRET\_ACCESS\_KEY
  - Secret: ISLPtbh0gXZu0ujRGCQlsvnHe61HyPP2ID9Hc

**Add secret** button.

**Bottom View: Repository secrets**

- Left Sidebar:** Shows navigation links for Rules, Actions, Webhooks, Environments, Codespaces, Pages, Security (Code security, Deploy keys), and Secrets and variables (Actions, Codespaces, Dependabot).
- Right Content Area:** Title: Repository secrets. A table showing two secrets:

Name	Last updated
AWS_ACCESS_KEY_ID	now
AWS_SECRET_ACCESS_KEY	now

**New repository secret** button.

Now we have to create task definition json file in our github repository

The screenshot shows the AWS CloudWatch Metrics console. A single metric named "AWS/CloudWatch Metrics" is listed. The value is 1, and the last value is also 1. The metric has a unit of "Count".

Metric Name	Value	Last Value
AWS/CloudWatch Metrics	1	1

The screenshot shows the AWS CloudWatch Metrics console. A single metric named "AWS/CloudWatch Metrics" is listed. The value is 1, and the last value is also 1. The metric has a unit of "Count".

Metric Name	Value	Last Value
AWS/CloudWatch Metrics	1	1

The screenshot shows a GitHub repository named 'VivekNere1/projectr-reg-app-s3-githubAction-for-documentation'. The 'main' branch is selected. A file named 'reg-app-task-definition.json' is open in the code editor. The code content is as follows:

```
1  {
2      "taskDefinitionArn": "arn:aws:ecs:ap-south-1:084375547362:task-definition/reg-app-task-defination-1-oct:1",
3      "containerDefinitions": [
4          {
5              "name": "reg-app-1-oct",
6              "image": "084375547362.dkr.ecr.ap-south-1.amazonaws.com/reg-app-repository",
7              "cpu": 0,
8              "portMappings": [
9                  {
10                      "name": "reg-app-1-oct-80-tcp",
11                      "containerPort": 80,
12                      "hostPort": 80,
13                      "protocol": "tcp",
14                      "appProtocol": "http"
15                  }
16              ],
17              "essential": true,
18              "environment": []
19          }
20      ]
21  }
```

The GitHub interface shows standard navigation and search tools at the top, and a toolbar with 'Edit' and 'Preview' buttons below the code editor. The status bar at the bottom indicates the date as '01-10-2024' and the time as '10:42'.

now create one service to create the cluster name

The screenshot shows the AWS Elastic Container Service (ECS) Create service wizard. The current step is "Compute configuration (advanced)".

**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:**

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

Use cluster default

**Capacity provider:** FARGATE

**Base:** 0

**Weight:** 1

**Platform version:** LATEST

**Deployment configuration:**

**Application type:**

Specify what type of application you want to run.

- Service: Launch a group of tasks handling a long-running computing work that can be stopped and restarted. For example, a web application.
- Task: Launch a standalone task that runs and terminates. For example, a batch job.

**Task definition:**

Select an existing task definition. To create a new task definition, go to [Task definitions](#).

Specify the revision manually

The screenshot shows the 'Create service' wizard for the Amazon Elastic Container Service (ECS). The 'Service' type is selected, indicating a long-running task definition. The 'Task definition' dropdown is set to 'Specify the revision manually'. The 'Family' dropdown contains 'reg-app-task-definition-1-oct' and the 'Revision' dropdown shows '1 (LATEST)'. The 'Service name' field is filled with 'reg-app-service-1oct'. The 'Service type' dropdown shows 'Replica' is selected. The 'Desired tasks' section is collapsed.

**Service**  
Launch a group of tasks handling a long-running computing work that can be stopped and restarted. For example, a web application.

**Task**  
Launch a standalone task that runs and terminates. For example, a batch job.

**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** reg-app-task-definition-1-oct    **Revision** 1 (LATEST)

**Service name** reg-app-service-1oct

**Service type** Replica

**Desired tasks**

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

**Networking**

**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.

**Volume - optional** Info  
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

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

The screenshot shows the AWS Elastic Container Registry (ECR) console. On the left, there's a sidebar with options like Clusters, Namespaces, Task definitions, and Account settings. The main area is titled 'Services' and shows a single service named 'reg-app-service-1oct'. This service is listed under the 'Active' status and has 0/1 tasks running. The interface includes tabs for Services, Tasks, Infrastructure, Metrics, Scheduled tasks, and Tags. At the bottom, there are buttons for Create, Manage tags, Update, Delete service, and a search bar.

Service get created!

Because now we don't have any image so it will fail but as soon as we run workflow file it will run.

Now we have to create workflow yml file

The screenshot shows the GitHub Actions workflow editor for a repository named 'projectr-reg-app-s3-githubAction-for-documentation'. The workflow file is named 'cicd-workflow.yml' and is located in the '.github/workflows' directory. The code defines a CI job that runs on Ubuntu-latest, performing steps like checkout source, configuring AWS credentials, setting up Java 14, and building the project with Maven.

```
1 name: CI
2 on:
3   push:
4     branches: [ main ]
5   jobs:
6     build-and-deploy:
7       runs-on: [ ubuntu-latest ]
8       steps:
9         - name: Checkout source
10           uses: actions/checkout@v3
11         - name: Configure AWS credentials
12           uses: aws-actions/configure-aws-credentials@v2
13           with:
14             aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
15             aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
16             aws-region: 'ap-south-1'
17         - name: Set up JDK 14
18           uses: actions/setup-java@v1
19           with:
20             java-version: 14
21
```

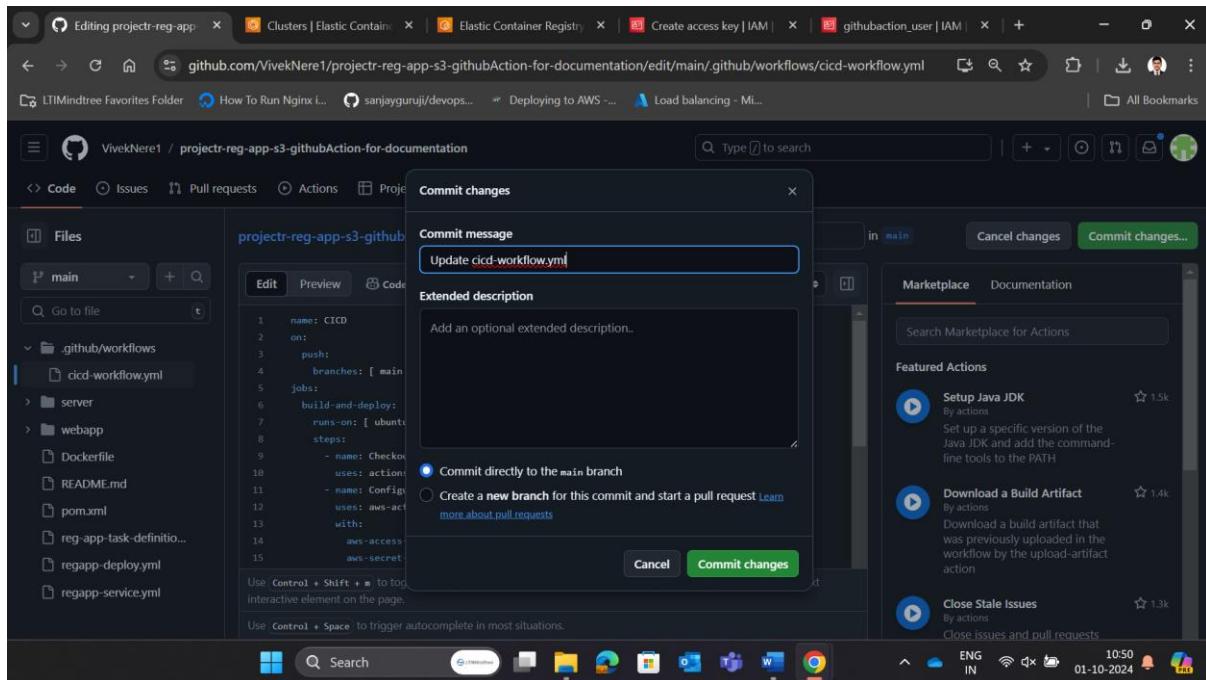
This screenshot is identical to the one above, showing the same GitHub Actions workflow editor interface and code for a CI/CD pipeline using Maven.

```
1 name: Build project with Maven
2 run: mvn -B package --file pom.xml
3
4 - name: Login to Amazon ECR
5   id: login-ecr
6   uses: aws-actions/amazon-ecr-login@v1
7   with:
8     mask-password: 'true'
9
10 - name: Build, tag, and push image to Amazon ECR
11   id: build-image
12   env:
13     ECR_REGISTRY: ${{ steps.login-ecr.outputs.registry }}
14     IMAGE_TAG: $(git rev-parse --short HEAD)
15     REPOSITORY: reg-app-repository
16   run:
17     # Build a docker container and
18     # push it to ECR so that it can
19     # be deployed to ECS.
20     docker build -t $ECR_REGISTRY/$REPOSITORY:$IMAGE_TAG .
21     docker push $ECR_REGISTRY/$REPOSITORY:$IMAGE_TAG
22
```

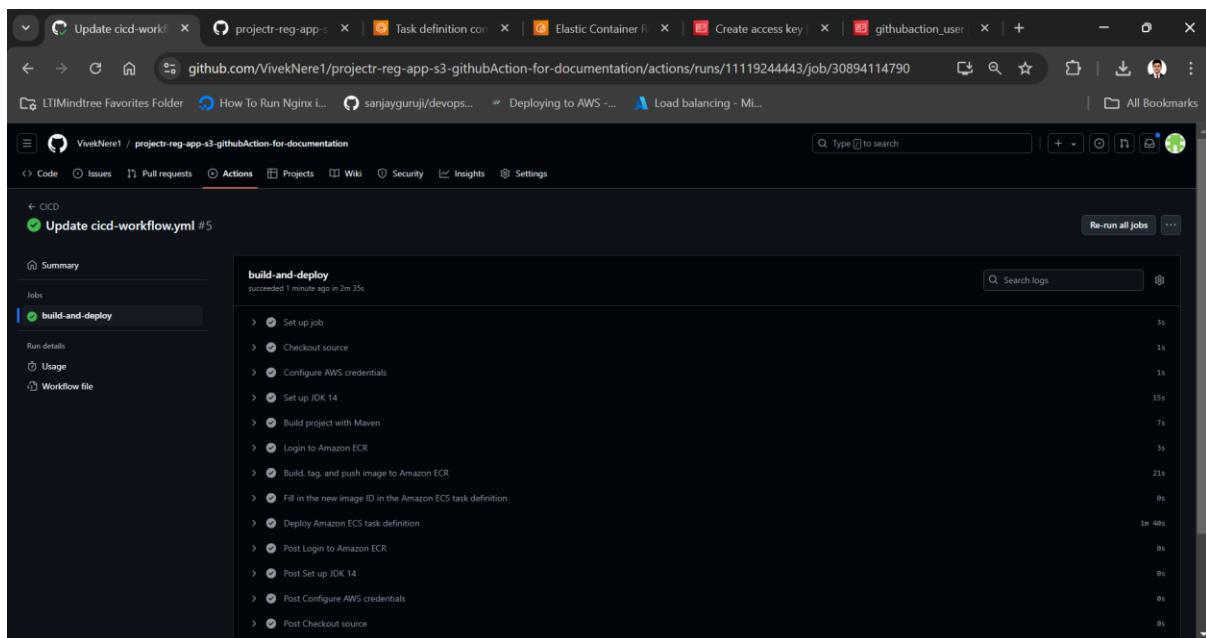
The screenshot shows two nearly identical views of the GitHub Actions workflow editor. Both views display the same code for a workflow named 'cicd-workflow.yml' located in the 'main' branch of a repository named 'projectr-reg-app-s3-githubAction-for-documentation'. The code defines a workflow with a single job named 'build-image'. This job uses an ECR registry, pushes an image to it, and then deploys it to an Amazon ECS task definition. It also includes steps to render and deploy the task definition.

```
id: build-image
env:
  ECR_REGISTRY: ${{ steps.login-ecr.outputs.registry }}
  IMAGE_TAG: ${{ github.sha }}
  REPOSITORY: reg-app-repository
run:
  - 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: reg-app-task-definition.json
      container-name: reg-app-1-ocf
      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: reg-app-service-ocf
      cluster: DevCluster-reg-app-1-ocf
      wait-for-service-stability: true
```

Commit the changes:



Now go to actions:



Our project successfully get deployed on ecs!

Now we have to give port 8080 access to our service.

The screenshot shows the AWS Cloud Console interface for the Amazon Elastic Container Service (ECS). The URL in the browser is <https://ap-south-1.console.aws.amazon.com/ecs/v2/clusters/DevCluster-reg-app-1-oct/services/reg-app-service-1oct/configuration?region=ap-south-1>.

The left sidebar navigation includes:

- Clusters
- Namespaces
- Task definitions
- Account settings
- Install AWS Copilot
- Amazon ECR
- Repositories
- AWS Batch
- Documentation
- Discover products
- Subscriptions

The main content area displays the "Service configuration" tab for the service `reg-app-service-1oct`. The configuration details are as follows:

Service ARN	Task definition: revision	Service type	Created by
<code>arn:aws:ecs:ap-south-1:084375547562:service/DevCluster-reg-app-1-oct/reg-a</code>	<code>reg-app-task-definition-1-oct:2</code>	REPLICA	arn:aws:iam::084375547562:root
Capacity provider	Capacity provider weight	Capacity provider base	CloudFormation stack
FARGATE	1	0	ECS-Console-V2-Service-reg-app-service-1oct-DevCluster-reg-app-1-oct-eafc4d2b

Below the table, there is a section for "Fargate ephemeral storage".

At the bottom of the page, the URL is <https://ap-south-1.console.aws.amazon.com/ecs/home?region=ap-south-1#>.

The screenshot shows the AWS VPC Security Groups 'Edit inbound rules' page. The table lists two rules:

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-0b774c82e9e71f3c3	All traffic	All	All	Custom	sg-0b49c105b4a2a82e5
sgr-0344ff112019483729	Custom TCP	TCP	8080	Custom	0.0.0.0/0

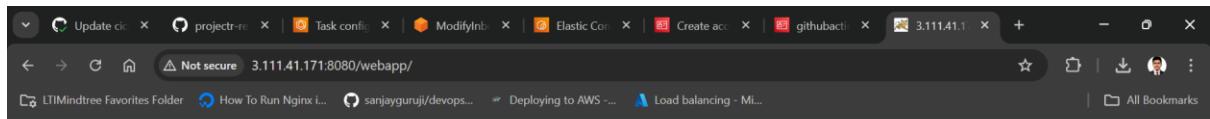
Buttons at the bottom include 'Add rule', 'Cancel', 'Preview changes', and 'Save rules'.

Now we have to check our hosted website

Ecs>cluster>services>task>copy the public ip

The screenshot shows the AWS CloudWatch Metrics interface. At the top, there are tabs for 'Metrics' and 'Logs'. Below the tabs, a search bar and a filter section are visible. The main area displays a table of metrics for the 'HelloWorld' function. The columns include Metric Name, Unit, and Value. One row shows the metric 'Function execution duration' with a unit of 'ms' and a value of 10. A detailed view of this metric entry is shown on the right, including a graph of the metric over time and a table of data points.

Now we have to check this on chrome



## New user Register for DevOps Learning made by Vivek Nere for Milestone-2 project

Please fill in this form to create an account.

Enter Name	Enter Full Name
Enter mobile	Enter mobile number
Enter Email	Enter Email
Password	Enter Password
Repeat Password	Repeat Password

By creating an account you agree to our [Terms & Privacy](#).

[Register](#)

Already have an account? [Sign in](#).

**Thank You, Happy Learning for devops!!**

**See You Again**

---

We can see our project get deployed successfully on ecs!