

# Deploying an Application on AWS ECS with ECR and Docker

Creating an Amazon ECR repository, building and pushing a Docker image, and deploying the application on Amazon ECS using Fargate.

## Steps

### 1. Create an ECR Repository

- i. Navigate to the Amazon ECR console.
- ii. Click on **Create repository**.
- iii. Enter 'unnati' as the repository name.

#### Create repository

**General settings**

Visibility settings [Info](#)

Choose the visibility setting for the repository.

☒ Private  
Access is managed by IAM and repository policy permissions.

☐ Public  
Publicly visible and accessible for image pulls.

Repository name

Provide a concise name. A developer should be able to identify the repository contents by the name.

767398120915.dkr.ecr.us-east-2.amazonaws.com/unnati

6 out of 256 characters maximum (2 minimum). The name must start with a letter and can only contain lowercase letters, numbers, hyphens, underscores, periods and forward slashes.

- iv. Click on **Create repository**.

Cancel

Create repository

## 2. Create an IAM Policy

- i. Navigate to the IAM console.
- ii. Select **policies** from the sidebar.
- iii. Click on **Create policy**.



- iv. Add the following JSON content to the policy document.

```
Policy editor
1 ▼ {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": [
7         "ecr:GetAuthorizationToken",
8         "ecr:BatchCheckLayerAvailability",
9         "ecr:GetDownloadUrlForLayer",
10        "ecr:GetRepositoryPolicy",
11        "ecr:DescribeRepositories",
12        "ecr:ListImages",
13        "ecr:DescribeImages",
14        "ecr:BatchGetImage",
15        "ecr:GetLifecyclePolicy",
16        "ecr:GetLifecyclePolicyPreview",
17        "ecr:ListTagsForResource",
18        "ecr:PutImage",
19        "ecr:UploadLayerPart",
20        "ecr:InitiateLayerUpload",
21        "ecr:DescribeImageScanFindings",
22        "ecr:CompleteLayerUpload"
23      ],
24      "Resource": "*"
25    }
26  ]
27 }
28 |
```

v. Give name for the policy.

**Policy details**

Policy name  
Enter a meaningful name to identify this policy.

My-ECR-Policy

Maximum 128 characters. Use alphanumeric and '+=, @-\_' characters.

Description - *optional*  
Add a short explanation for this policy.

Maximum 1,000 characters. Use alphanumeric and '+=, @-\_' characters.

vi. Click on **Create policy**.

### 3. Create an IAM User and Attach Policy

- i. Navigate to the IAM console.
- ii. Select **Users** from the sidebar.
- iii. Click on **Add user**.
- iv. Enter a username.

Specify user details


**User details**

User name

my-ecr-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're providing console access to a person, it's a [best practice](#) to manage their access in IAM Identity Center.

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

Cancel **Next**

v. Click on **Next: Permissions**.

vi. Select **Attach Policy** directly.

## Set permissions

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. [Learn more](#)

### Permissions options

☒ Add user to group

Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

☐ Copy permissions

Copy all group memberships, attached managed policies, and inline policies from an existing user.

☐ Attach policies directly

Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

- vii. Search and select the policy that we created earlier.

**Permissions policies (1/1299)** [Create policy](#)

Choose one or more policies to attach to your new user.

Search: My-ECR Filter by Type: All types 1 match

<input checked="" type="checkbox"/>	Policy name	Type	Attached entities
<input checked="" type="checkbox"/>	<a href="#">My-ECR-Policy</a>	Customer managed	0

► Set permissions boundary - optional

[Cancel](#) [Previous](#) [Next](#)

- viii. Click on **Next: Review**.

- ix. Click on **Create user**.

- x. Click on the created user and go to the **Security credentials** tab.

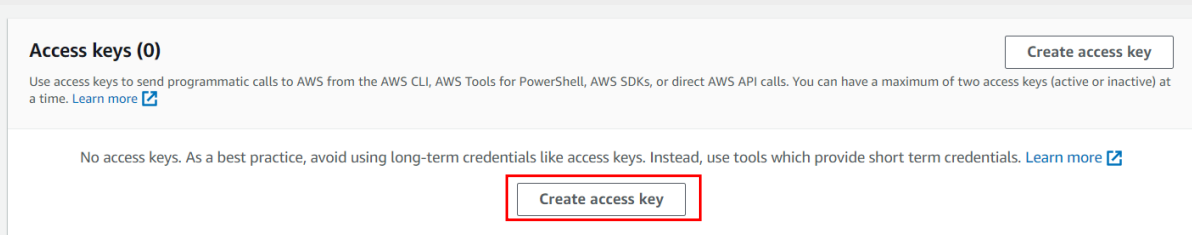
**my-ecr-user** [Info](#)

### Summary

<b>ARN</b> arn:aws:iam::767398120915:user/my-ecr-user	<b>Console access</b> Enabled without MFA	<b>Access key 1</b> <a href="#">Create access key</a>
<b>Created</b> July 19, 2024, 10:41 (UTC+05:30)	<b>Last console sign-in</b> Never	

[Permissions](#) [Groups](#) [Tags](#) [Security credentials](#) [Access Advisor](#)

xi. Click on **Create access key**.



xii. Save the access key and secret key for later use.

#### 4. Configure AWS CLI

- i. Open a terminal (I'm using killercoda environment).
- ii. Run the **aws configure** command:
- iii. Enter the access key and secret key when prompted.

```
ubuntu $ aws configure
AWS Access Key ID [None]: 
AWS Secret Access Key [None]: 
Default region name [None]: us-east-2
Default output format [None]: json
ubuntu $
```

#### 5. Create a Dockerfile and index.html file

- i. Create a '**Dockerfile**' with the following content:
- ii. Create an '**index.html**' file with a simple message:

```
ubuntu $ echo "Hello from Divya!" > index.html
ubuntu $ cat Dockerfile
FROM docker.io/ubuntu
RUN apt update -y
RUN apt install apache2 -y
COPY index.html /var/www/html/
CMD ["apachectl","-D","FOREGROUND"]
ubuntu $
```

## 6. Build and Push Docker Image to ECR

- i. Run the following command to authenticate Docker to your Amazon ECR registry:

```
ubuntu $ aws ecr get-login-password --region us-east-2 | docker login --username AWS --password-stdin 767398120915.dkr.ecr.us-east-2.amazonaws.com
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
ubuntu $
```

- ii. Build the Docker image:

```
ubuntu $ docker build -t unnati .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

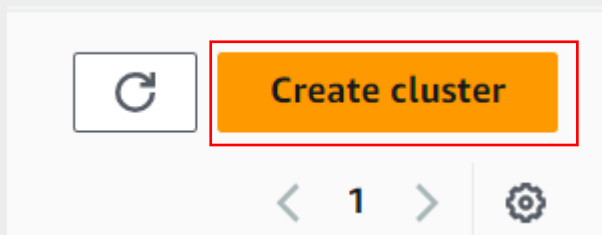
Sending build context to Docker daemon 33.28kB
Step 1/5 : FROM docker.io/ubuntu
--> 35a88802559d
Step 2/5 : RUN apt update -y
--> Using cache
--> a6ef5a7835a3
Step 3/5 : RUN apt install apache2 -y
--> Using cache
--> cf6131a6d86a
Step 4/5 : COPY index.html /var/www/html/
--> Using cache
--> c78dd00cff5d
Step 5/5 : CMD ["apachectl","-D","FOREGROUND"]
--> Using cache
--> 802d531edd75
Successfully built 802d531edd75
Successfully tagged unnati:latest
ubuntu $
```

### iii. Tag and Push the docker image to ECR:

```
ubuntu $ docker tag unnati:latest 767398120915.dkr.ecr.us-east-2.amazonaws.com/unnati:latest
ubuntu $ docker push 767398120915.dkr.ecr.us-east-2.amazonaws.com/unnati:latest
The push refers to repository [767398120915.dkr.ecr.us-east-2.amazonaws.com/unnati]
3047dad3da1a: Pushed
85705c77aae6: Pushed
d965f09cfbce: Pushed
a30a5965a4f7: Layer already exists
latest: digest: sha256:8ef0b7b1bd53085601bd3d6640385a37f67dd341e2d2b2c7890122ff9488a1f5 size: 1160
ubuntu $
```

## 7. Create an ECS Cluster

- i. Navigate to the Amazon ECS console.
- ii. Click on **Clusters** in the sidebar.
- iii. Click on **Create Cluster**.



### iv. Enter a name for the cluster.

**Cluster configuration**

Cluster name

ecs-cluster

Cluster name must be 1 to 255 characters. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (\_).

Default namespace - optional

Select the namespace to specify a group of services that make up your application. You can overwrite this value at the service level.

Q ecs-cluster X

▼ **Infrastructure** Info Serverless

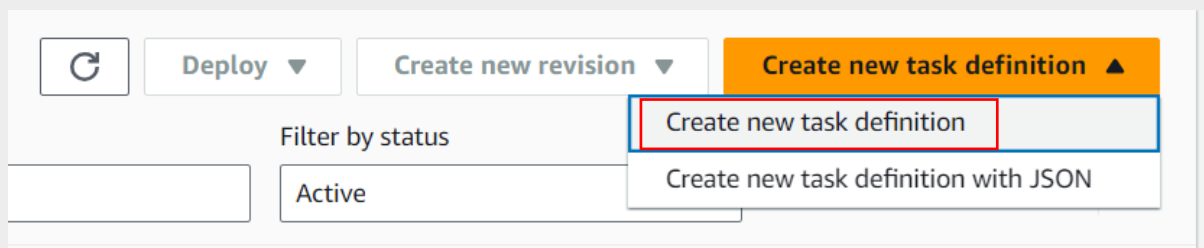
Your cluster is automatically configured for AWS Fargate (serverless) with two capacity providers. Add Amazon EC2 instances.

☒ **AWS Fargate (serverless)**  
Pay as you go. Use if you have tiny, batch, or burst workloads or for zero maintenance overhead. The cluster has Fargate and Fargate Spot capacity providers by default.

- v. Click on **Create**.

## 8. Create a Task Definition

- i. Navigate to the Amazon ECS console.
- ii. Click on **Task Definition** in the sidebar.
- iii. Click on **Create new Task Definition**.



- iv. Enter a name for task definition.

A screenshot of the 'Task definition configuration' form in the Amazon ECS console. The form is divided into sections. The first section is 'Task definition family', which includes a link to 'Info' and a text input field containing 'my-ecs-task'. Below the input field, a note states: 'Up to 255 letters (uppercase and lowercase), numbers, hyphens, and underscores are allowed.' The second section is 'Infrastructure requirements', which includes a link to 'Info' and a text input field. Below this, there are two options for the launch type: 'AWS Fargate' (selected with a checked checkbox and highlighted with a red box) and 'Amazon EC2 instances' (unchecked). The 'AWS Fargate' option is described as 'Serverless compute for containers.' and the 'Amazon EC2 instances' option is described as 'Self-managed infrastructure using Amazon EC2 instances.'

- v. In the **Container** section, click on Add container.



- vi. Enter a name for the container.
- vii. In the **Image** field, enter the **URI** of the **ECR repository**.

▼ Container - 1 [Info](#) Essential container Remove

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 Yes ▼

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

☐ Private registry authentication

- viii. Click on **Create**.

## 9. Create a Service

- i. Navigate to ECS console.
- ii. Click on **Clusters** in the sidebar.
- iii. Select the cluster created earlier.
- iv. Click on **Create** under **Services** tab.

Services (0) [Info](#) Refresh Manage tags Update Delete service Create

Filter launch type Any launch type ▼ Filter service type Any service type ▼ < 1 > Settings

Service name	ARN	Status	Service...	Deployments and tasks	Last dep
No services No services to display. <span>Create</span>					

- v. Under **Environment**, select **Capacity provider strategy** and choose **FARGATE**.

## Create [Info](#)

### Environment

Existing cluster

ecs-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.

Capacity provider strategy [Info](#)

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

##### ☐ Use cluster default

No default capacity provider strategy configured for this cluster.

##### ☒ Use custom (Advanced)

Capacity provider

FARGATE

Base [Info](#)

0

Weight [Info](#)

1

Add capacity provider

- vi. Under **Deployment configuration**, select **Task**.
- vii. **Task definition**: Select the created task definition.

### Deployment configuration

Application type [Info](#)

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

Manually input the revision instead of choosing from the 100 most recent revisions for the selected task definition family.

Family

my-ecs-task

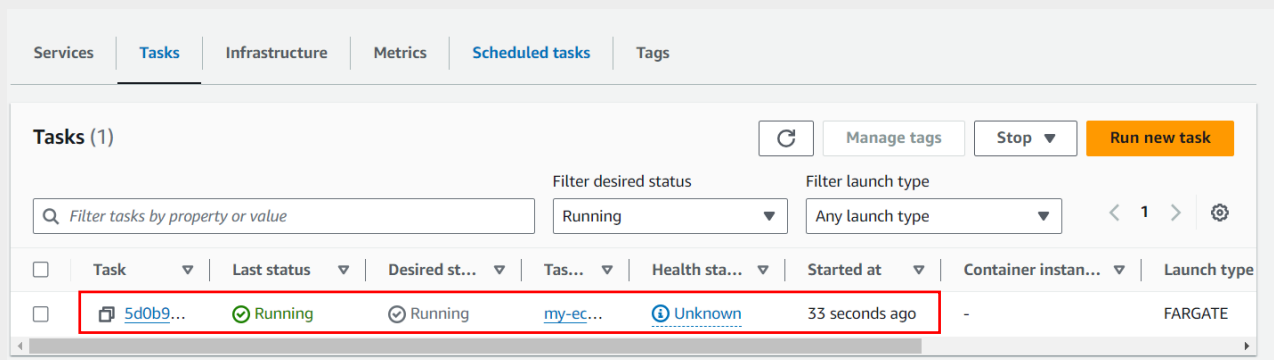
Revision

1 (LATEST)

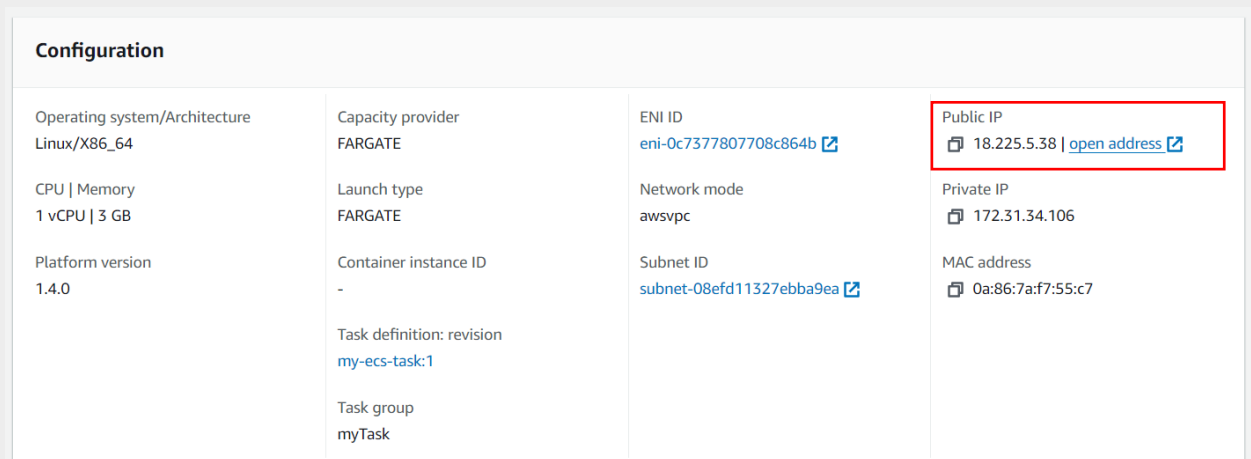
viii. Click **Next** and **Create service**.

## 10. Access the Deployed Application

- i. Navigate to the Amazon ECS console.
- ii. Select the cluster and click on the service created earlier.
- iii. Click on **Task** tab and select the running task.



iv. Under Configuration, click on Open address.



- v. Open the address in a web browser to and navigate to the public IP address to view the 'index.html' page created earlier.



**By following these steps you've successfully deployed an application on AWS ECS using ECR and Docker.**