

# AWS ECR

## What is ECR?

Amazon Elastic Container Registry (ECR) is a fully managed container image registry service by AWS.

- Similar to Docker Hub, but integrated with AWS.
- Used to store, manage, share, and deploy Docker container images.
- Works seamlessly with ECS, EKS, and EC2.

## Key Features of ECR

### 1. Fully Managed

- No need to manage storage or registry servers.
- AWS handles scaling, availability, and security.

### 2. Private & Public Repositories

- Private Repos → Store images securely for your organization.
- Public Repos → Share images with anyone (like Docker Hub).

### 3. IAM Integration

- Access controlled via IAM roles and policies.
- Example: EC2 can pull images automatically using IAM role (no passwords needed).

### 4. Security

- Supports image scanning (detect vulnerabilities in images).
- Integrated with AWS KMS encryption (images stored encrypted at rest).

## 5. Performance

- Optimized for AWS network → low latency image pulls for ECS/EKS/EC2.

## 6. Lifecycle Policies

- Automatically delete old/unneeded images (e.g., keep only last 10).
- Helps save S3 storage costs.

## How ECR Works (Flow)

- **Developer builds Docker image** locally (e.g., React app, Spring Boot app).
- **Push image to ECR:**
  - Authenticate with `aws ecr get-login-password`.
  - Use `docker push` to upload image.
- **Store & Secure:**
  - Image stored in **ECR repository**.
  - Managed with IAM permissions and lifecycle policies.
- **Pull & Deploy:**
  - ECS/EKS/EC2 pulls the image from ECR using IAM role.
  - The container runs in your cluster or instance.

## ECR vs Docker Hub

Feature	AWS ECR	Docker Hub
Security (IAM/KMS)	✓ Strong (IAM, KMS encryption)	✗ Limited (username/password)
Integrated with ECS/EKS	✓ Yes	✗ Manual
Image Scanning	✓ Native (Amazon Inspector, Trivy)	✓ Available (with Docker Hub Pro)
Private Repos	✓ Unlimited	✗ Limited (only paid)
Performance on AWS	✓ Fast (AWS internal network)	✗ Slower for AWS workloads

## Step 1: Open ECR Console

- Go to **AWS Console** → **Services** → **Elastic Container Registry (ECR)**.



## Step 2: Create a Repository

1. Click **Repositories** → **Create repository**.
2. Choose **Private** (default) or **Public** (if you want open access).
3. Enter **Repository name** (e.g., `my-app`).
4. (Optional) Enable:
  - **Scan on push** → scans images for vulnerabilities automatically.
  - **KMS encryption** → default AWS-managed key is fine.
5. Click **Create repository**.

Amazon ECR > Private registry > Repositories > Create a private repository

## Create a private repository

### General settings

**Repository name**  
Enter a concise name. Repositories support namespaces, which you can use to group similar repositories.

491766119730.dkr.ecr.us-east-1.amazonaws.com/

10 out of 256 characters maximum (2 minimum). The name must start with a letter and can only contain lowercase letters, numbers and special characters \_-./.

### Image tag settings [Info](#)

**Image tag mutability**  
Choose the tag mutability setting.

☐ Mutable  
Image tags can be overwritten

☒ Immutable  
Image tags can't be overwritten.

**Immutable tag exclusions**

### ▼ Image scanning settings - *deprecated*

**⚠ Deprecation warning**  
This setting has been moved to a registry level configuration with repository filtering. [Find out more](#)

**Scan on push**  
Enable scan on push to have each image automatically scanned after being pushed to a repository. If disabled, each image scan must be manually started to get scan results.

☒

Amazon ECR > Private registry > Repositories

Amazon Elastic Container Registry

- ▼ Private registry
  - Repositories
  - Features & Settings
- ▼ Public registry
  - Repositories
  - Settings

## Private repositories (1)

[View push commands](#) [Delete](#) [Actions](#) [Create repository](#)

Search by repository substring

	Repository name	URI	Created at	Tag immutability	Encryption type
<input type="radio"/>	test-vinai	491766119730.dkr.ecr.us-east-1.amazonaws.com/test-vinai	09 September 2025, 17:05:25 (UTC+05.5)	Immutable	AES-256

## Step 3: Get Push Commands

- After repository creation, click **View push commands**.
- AWS shows the **4 exact steps** (login, build, tag, push). Since you're doing it via console, copy those commands and run them in your **local terminal** where Docker is installed.

Images (0)

Delete
Details
Scan
View push commands

< 1 >

<input type="checkbox"/>	Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest	Last recorded pull time
No images							
No images to display							

## Push commands for test-vinai



macOS / Linux

Windows

Make sure that you have the latest version of the AWS CLI and Docker installed. For more information, see [Getting started with Amazon ECR](#).

Use the following steps to authenticate and push an image to your repository. For additional registry authentication methods, including the Amazon ECR credential helper, see [Registry authentication](#).

1. Retrieve an authentication token and authenticate your Docker client to your registry. Use the AWS CLI:

```
aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 491766119730.dkr.ecr.us-east-1.amazonaws.com
```

Note: if you receive an error using the AWS CLI, make sure that you have the latest version of the AWS CLI and Docker installed.

2. Build your Docker image using the following command. For information on building a Docker file from scratch, see the instructions [here](#). You can skip this step if your image has already been built:

```
docker build -t test-vinai .
```

3. After the build is completed, tag your image so you can push the image to this repository:

```
docker tag test-vinai:latest 491766119730.dkr.ecr.us-east-1.amazonaws.com/test-vinai:latest
```

4. Run the following command to push this image to your newly created AWS repository:

```
docker push 491766119730.dkr.ecr.us-east-1.amazonaws.com/test-vinai:latest
```

## Step 4: Authenticate Docker

- The **first push command** lets Docker authenticate with your ECR repo.
- Copy & run it on your laptop/server terminal.
- This uses your IAM credentials (make sure you are logged in with AWS CLI or AWS SSO).

```
[root@ip-172-31-34-254 Frontend]# aws configure
AWS Access Key ID [*****2BDR]:
AWS Secret Access Key [*****gPFN]:
Default region name [us-east-1]:
Default output format [json]:
[root@ip-172-31-34-254 Frontend]#
```

```
[root@ip-172-31-34-254 Frontend]# aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 491766119730.dkr.ecr.us-east-1.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
[root@ip-172-31-34-254 Frontend]#
```

## Step 5: Build and Push Image

- Locally (on your machine):
  - Build your Docker image (**docker build**).
  - Tag it with the ECR repo URI (you'll see the exact tag format in the console).
  - Push it to ECR (**docker push**).

```
[root@ip-172-31-34-254 Frontend]# ll
total 264
-rw-r--r--. 1 root root 265 Sep  9 11:51 Dockerfile
-rw-r--r--. 1 root root 856 Aug 11 08:56 README.md
drwxr-xr-x. 3 root root  85 Aug 27 04:41 dist
-rw-r--r--. 1 root root 844 Aug 11 08:56 eslint.config.js
-rw-r--r--. 1 root root 208 Aug 11 08:56 index.css
-rw-r--r--. 1 root root 496 Aug 11 08:56 index.html
drwxr-xr-x. 375 root root 16384 Aug 27 04:40 node_modules
-rw-r--r--. 1 root root 225213 Aug 27 04:40 package-lock.json
-rw-r--r--. 1 root root 864 Aug 11 08:56 package.json
drwxr-xr-x. 2 root root  53 Aug 11 08:56 public
drwxr-xr-x. 7 root root 111 Aug 27 04:39 src
-rw-r--r--. 1 root root 161 Aug 11 08:56 vite.config.js
[root@ip-172-31-34-254 Frontend]#
```

```
[root@ip-172-31-34-254 Frontend]# cat Dockerfile
# Step 1: Use Apache HTTPD as base image
FROM httpd:2.4

# Step 2: Copy built React files (dist folder) into Apache's web root
COPY dist/ /usr/local/apache2/htdocs/

# Step 3: Expose port 80
EXPOSE 80

# Step 4: Start Apache in foreground
CMD ["httpd-foreground"]

[root@ip-172-31-34-254 Frontend]#
```

```
[root@ip-172-31-34-254 Frontend]# docker build -t test-vinai .
[+] Building 0.2s (7/7) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 357B
=> [internal] load metadata for docker.io/library/httpd:2.4
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load build context
=> => transferring context: 1.10kB
=> [1/2] FROM docker.io/library/httpd:2.4@sha256:c4edd43414d68d7abd223ef20bef385c95bbbbeaea53b01c6ffef01d62bf6309
=> CACHED [2/2] COPY dist/ /usr/local/apache2/htdocs/
=> exporting image
=> => exporting layers
=> => writing image sha256:425dd631a8dad5b30a4c1d1d0bd788babe3b6195c698eed9a76ca75c9c6b1581
=> => naming to docker.io/library/test-vinai
[root@ip-172-31-34-254 Frontend]#
```

```
[root@ip-172-31-34-254 Frontend]# docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
test-vinai latest 425dd631a8da 4 minutes ago 122MB
[root@ip-172-31-34-254 Frontend]#
```

```
[root@ip-172-31-34-254 Frontend]# docker tag test-vinai:latest 491766119730.dkr.ecr.us-east-1.amazonaws.com/test-vinai:latest
[root@ip-172-31-34-254 Frontend]# docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
491766119730.dkr.ecr.us-east-1.amazonaws.com/test-vinai latest 425dd631a8da 5 minutes ago 122MB
test-vinai latest 425dd631a8da 5 minutes ago 122MB
[root@ip-172-31-34-254 Frontend]#
```

```
[root@ip-172-31-34-254 Frontend]# docker push 491766119730.dkr.ecr.us-east-1.amazonaws.com/test-vinai:latest
The push refers to repository [491766119730.dkr.ecr.us-east-1.amazonaws.com/test-vinai]
87a449e85422: Pushed
4ed3261aa08c: Pushed
664c74752319: Pushed
09c56534a346: Pushed
5f70bf18a086: Pushed
d694d07f5d65: Pushed
daf557c4f08e: Pushed
latest: digest: sha256:60d919edff57ea02c636c82dcd08970b90bf6e23f57ad9c9181daed97e445d6c size: 1783
[root@ip-172-31-34-254 Frontend]#
```

Once pushed, the image appears in your **ECR** → **Repositories** → **Images** list.

Amazon Elastic Container Registry

Private registry

Repositories

Summary

Images

Permissions

Lifecycle Policy

Repository tags

Features & Settings

Public registry

Repositories

Settings

ECR public gallery

Images (1)

Search artefacts

Delete

Details

Scan

View push commands

< 1 >

<input type="checkbox"/>	Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest	Last recorded pull time
<input type="checkbox"/>	latest	Image	09 September 2025, 17:27:34 (UTC+05.5)	50.04	Copy URI	sha256:60d919edff57ea0...	-

Image

Details

Image tags  
latest

URI  
 491766119730.dkr.ecr.us-east-1.amazonaws.com/test-vinai:latest

Digest  
 sha256:60d919edff57ea02c636c82dcd08970b90bf6e23f57ad9c9181daed97e445d6c

General information

Artifact type Image	Repository test-vinai	Pushed at 09 September 2025, 17:27:34 (UTC+05.5)
Last recorded pull time -	Size (MB) 50.04	



## Step 6: Manage Images in Console

- Click into your repo → **Images** → you can see tags, digests, vulnerabilities.
- You can **delete** or **scan** images directly here.

### Scanning and vulnerabilities

**Status**  
🟢 Complete  
The scan was completed successfully.

**Scan completed at**  
09 September 2025, 17:27:42 (UTC+05.5)

**Vulnerability source updated at**  
09 September 2025, 17:27:42 (UTC+05.5)

Scan

Critical  
0

High  
0

Medium  
1

Low  
0

Info  
0

**Vulnerabilities (1)**

Q

< 1 >

Name	Vulnerable package	Severity	Description
<a href="#">CVE-2025-26434</a>	libxml2::2.12.7+Bdfsg%2Breally2.9.14-2.1%2Bdeb13u1	MEDIUM	In libxml2, there is a possible out of bounds read due to a buffer overflow. This could lead to local information disclosure with no additional execution privileges needed. User interaction is not needed for exploitation.

## Step 7: Setup Lifecycle Policy (Optional but Recommended)

1. Inside your repository → **Lifecycle Policy** → Create policy.
2. Example: Keep last 10 images, delete older ones.
3. Save.

This helps control storage & cost.

Private registry > Repositories > test-vinai > Lifecycle policy > Create rule

### Create lifecycle rule

**Lifecycle rule configuration**

⚠ If you apply this rule as your lifecycle policy, it may be evaluated immediately. It is recommended that you dry run your rules first.

**Rule priority**  
Specify a rule priority, which must be unique. Values do not need to be sequential across rules in a policy.

1

**Rule description**  
Specify a description for your lifecycle policy.  
*example: remove alpha images*

**Image status**  
Indicates whether the image is tagged or not.

☒ Tagged (wildcard matching)  
☐ Tagged (prefix matching)  
☐ Untagged

**Specify tags for wildcard matching**  
Specify a list of image tags with wildcard (\*) to match image tags to apply lifecycle rule towards using a comma-separated list.  
*example: prod\*,test\*linux*

**Match criteria**  
Specify the count type to apply to the images.

Image count more than 10

**Rule action**  
The only supported value is expire. The image will be deleted from your repository. This action cannot be undone.

expire

Cancel Save

## Step 8: (Optional) Enable Enhanced Scanning

- Go to ECR → Repositories → Settings.
- Turn on **Amazon Inspector scanning** for deeper vulnerability checks.

Amazon ECR > Private registry > Scanning

### Scanning configuration

Basic scanning is provided by default for your private registry. Enhanced scanning can be enabled for your registry to provide automated, continuous scanning to find vulnerabilities in your container images.

**Scan type**  
Select the scanning type that will be used for this registry. Enhanced scanning is a paid service. [Find out more](#)

☐ Basic scanning  
Basic scanning allows manual scans and scanning of push of images in this registry. This is a free service.

☒ Enhanced scanning  
Enhanced scanning with Amazon Inspector provides automated continuous scanning. Inspector identifies vulnerabilities in both the operating system and programming language (such as Python, Java, Ruby etc.) packages in real time.

**Continuous scanning filters**  
Select which repositories will continuously have images scanned for vulnerabilities. Filters with no wildcard will match all repository names that contain the filter. Filters with wildcards (\*) will match a repository name where the wildcard replaces zero or more characters in the repository name.

☒ Continuously scan all repositories

Add filter

**Scan on push filters**  
Select which repositories to scan for vulnerabilities on image push. Filters with no wildcard will match all repository names that contain the filter. Filters with wildcards (\*) will match a repository name where the wildcard replaces zero or more characters in the repository name.

☐ Scan on push all repositories

Add filter

Now your image is stored in **ECR** and ready to be pulled by ECS, EKS, or any EC2 with proper IAM permissions.

### Step 9: Pull image from ECR

- Go to **ECR** → **Repositories** → **Image** → **Copy Image URI**
- Goto console where you want to pull
- Do aws configure with your IAM Credentials access key & secret key which have permissions to pull from ECR
- Authenticate with docker

1. Retrieve an authentication token and authenticate your Docker client to your registry. Use the AWS CLI:

```
aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 491766119730.dkr.ecr.us-east-1.amazonaws.com
```

Note: if you receive an error using the AWS CLI, make sure that you have the latest version of the AWS CLI and Docker installed.

```
[root@ip-172-31-34-254 Frontend]# docker pull 491766119730.dkr.ecr.us-east-1.amazonaws.com/test-vinai:latest
latest: Pulling from test-vinai
Digest: sha256:60d919edff57ea02c636c82dcd08970b90bf6e23f57ad9c9181daed97e445d6c
Status: Image is up to date for 491766119730.dkr.ecr.us-east-1.amazonaws.com/test-vinai:latest
491766119730.dkr.ecr.us-east-1.amazonaws.com/test-vinai:latest
[root@ip-172-31-34-254 Frontend]#
```