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).

KOLLURI VINAI

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.

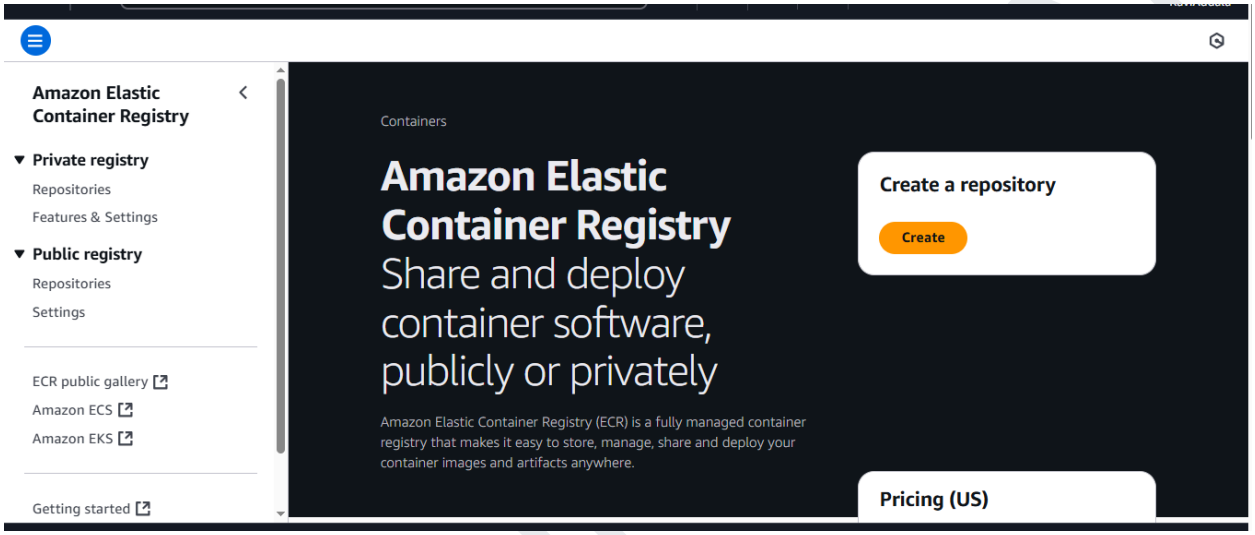
ECR vs Docker Hub

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

KOLLURI VINAI

Step 1: Open ECR Console

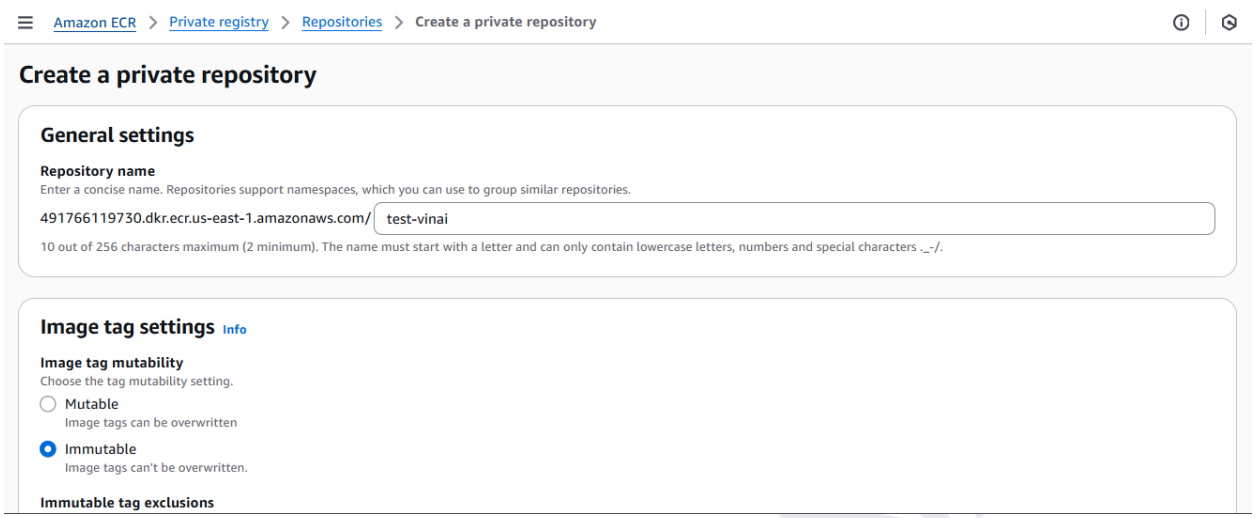
- Go to **AWS Console → Services → Elastic Container Registry (ECR)**.

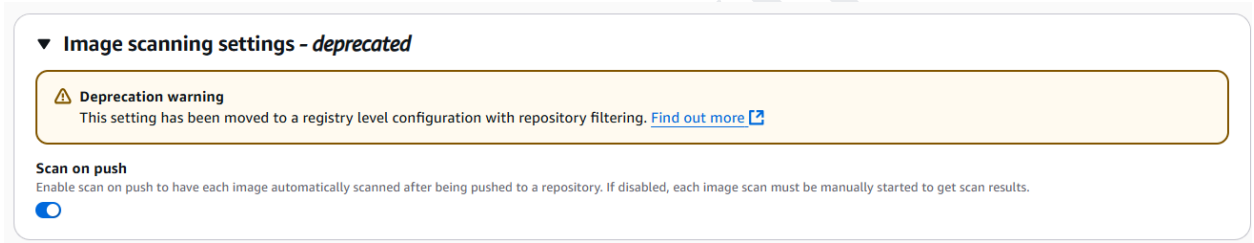


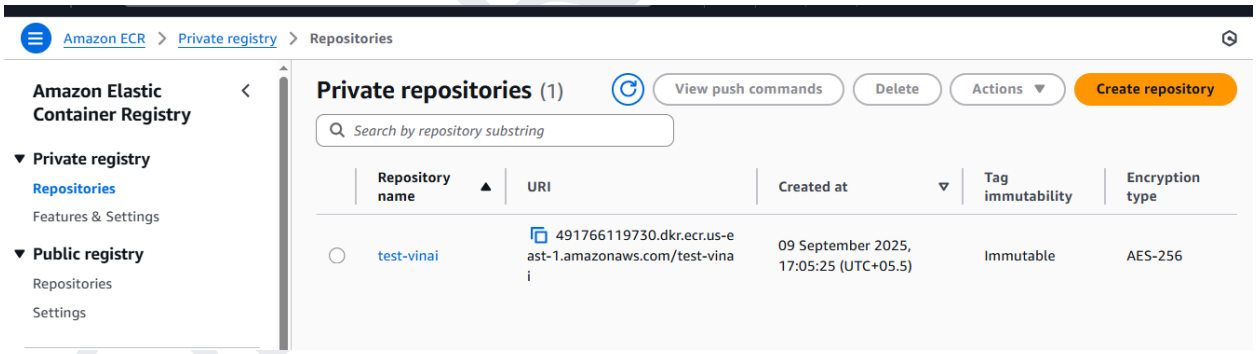
Step 2: Create a Repository

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**.

KOLLURI

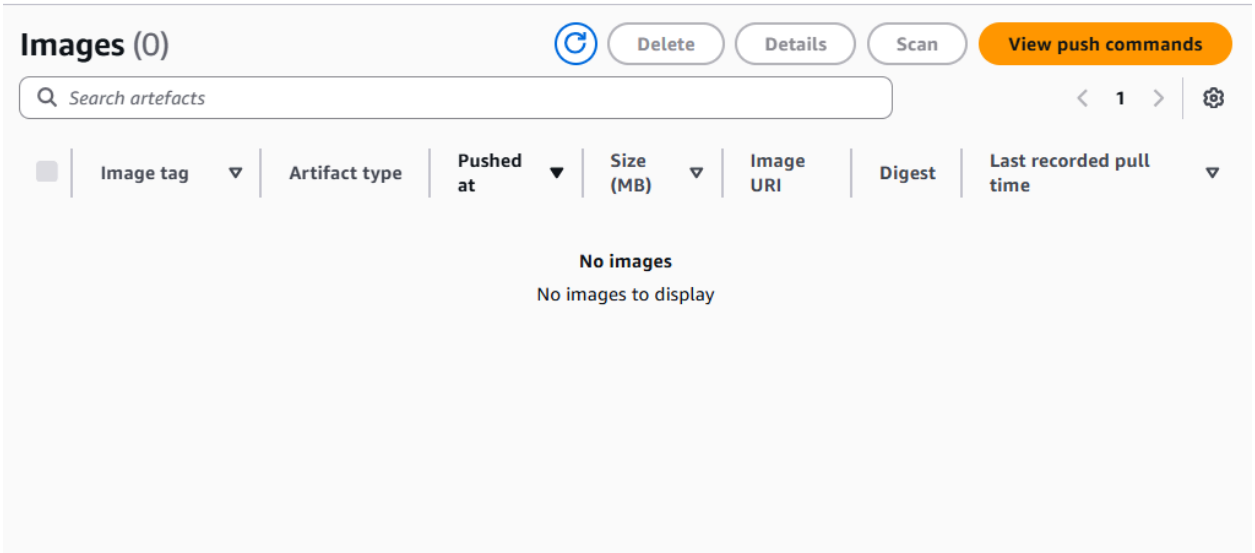






Step 3: Get Push Commands

- 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.



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 [****************gFPN]:
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]#
```

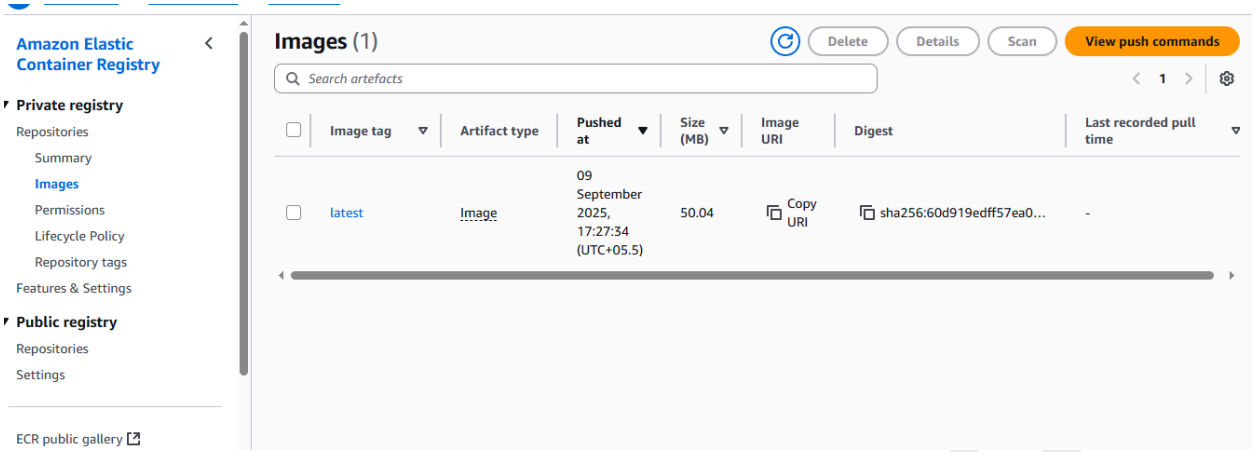
```
[root@ip-172-31-34-254 Frontend]# docker build -t test-vinai .
[+] Building 0.2s (7/7) FINISHED                                                                                    docker:default
 => [internal] load build definition from Dockerfile                                                                        0.0s
 => => transferring dockerfile: 357B                                                                                        0.0s
 => [internal] load metadata for docker.io/library/httpd:2.4                                                                0.1s
 => [internal] load .dockerignore                                                                                           0.0s
 => => transferring context: 2B                                                                                             0.0s
 => [internal] load build context                                                                                           0.0s
 => => transferring context: 1.10kB                                                                                         0.0s
 => [1/2] FROM docker.io/library/httpd:2.4@sha256:c4edd43414d68d7abd223ef20bef385c95bbbbeaea53b01c6ffef01d62bf6309           0.0s
 => CACHED [2/2] COPY dist/ /usr/local/apache2/htdocs/                                                                      0.0s
 => exporting to image                                                                                                      0.0s
 => => exporting layers                                                                                                     0.0s
 => => writing image sha256:425dd631a8dad5b30a4c1d1d0bd788babe3b6195c698eed9a76ca75c9c6b1581                                0.0s
 => => naming to docker.io/library/test-vinai                                                                               0.0s
[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.



Image

Details

Image tags
latest

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

Digest
sha256:60d919edff57ea02c636c82dcd08970b90bf6e23f57ad9c9181daed97e445d6c

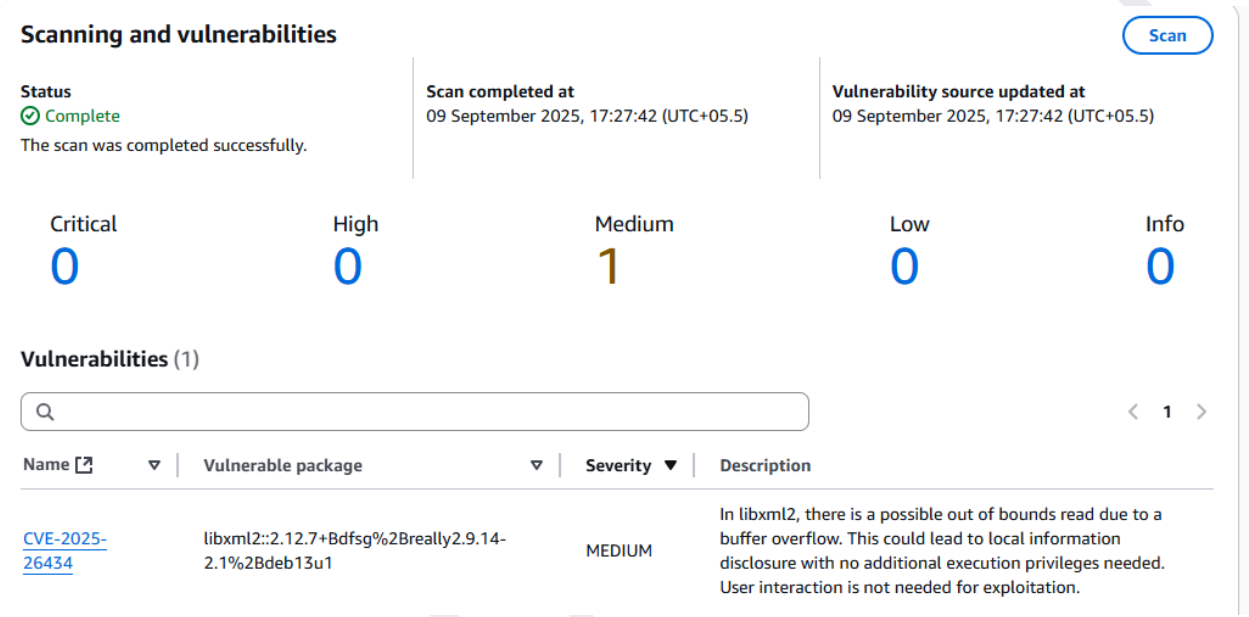
General information

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	

KOLL

Step 6: Manage Images in Console

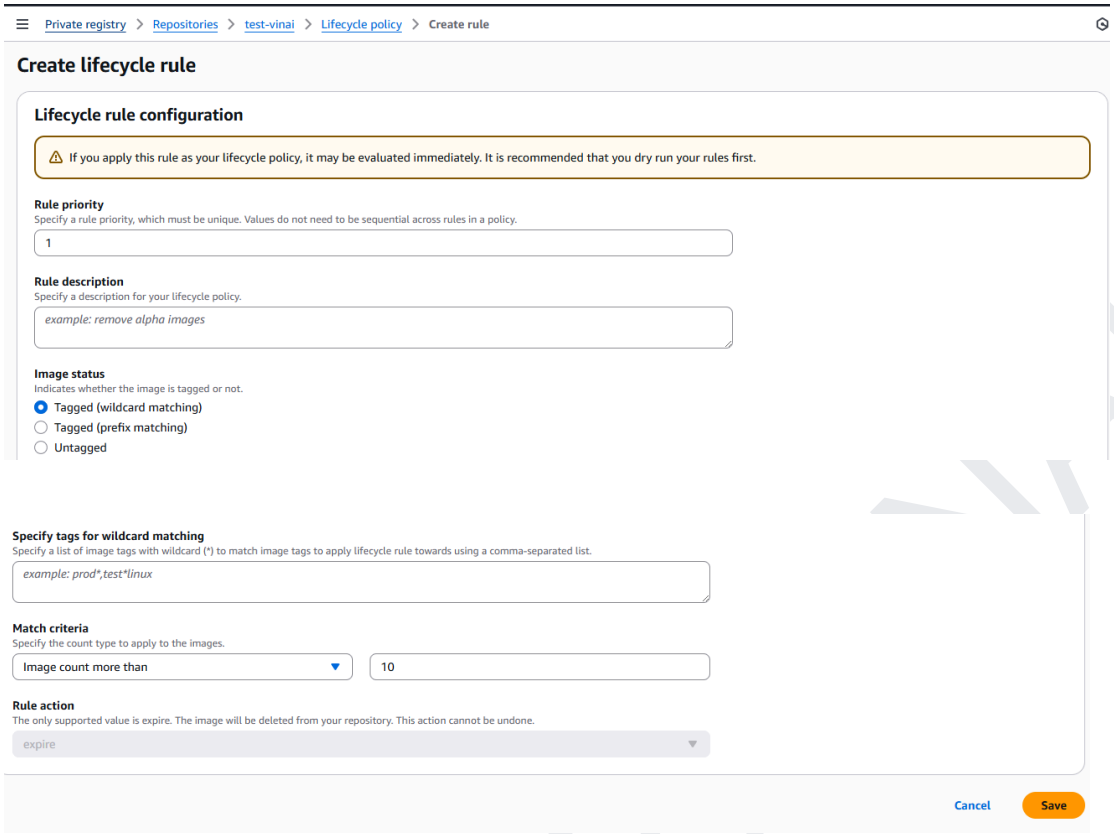
- Click into your repo → **Images** → you can see tags, digests, vulnerabilities.
- You can **delete** or **scan** images directly here.



Step 7: Setup Lifecycle Policy (Optional but Recommended)

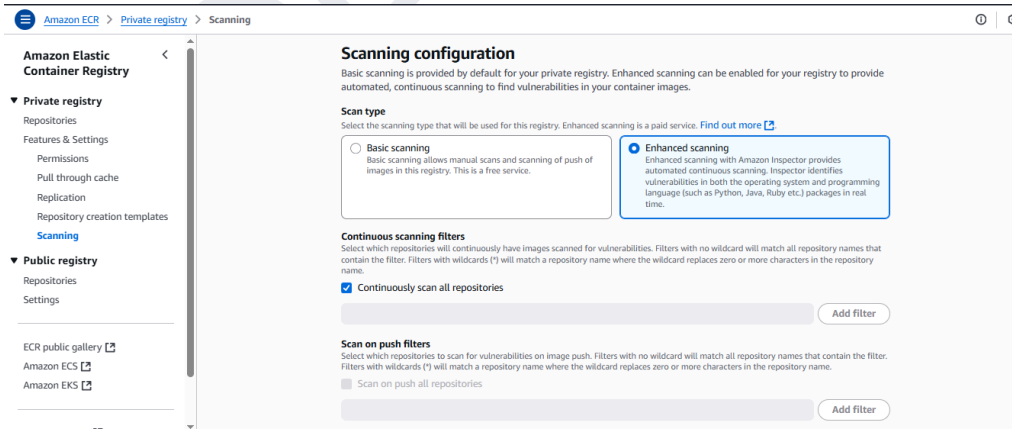
1. Inside your repository → **Lifecycle Policy** → Create policy.
2. Example: Keep last 10 images, delete older ones.
3. Save.

   This helps control storage & cost.



Step 8: (Optional) Enable Enhanced Scanning

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



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]#
```