

## 1. What is Amazon ECR and how does it differ from Docker Hub?

#### **Answer:**

Amazon Elastic Container Registry (ECR) is a **fully managed container image** registry provided by AWS. It allows developers to store, manage, and deploy Docker container images securely and at scale.

### **Key Differences from Docker Hub:**

- Integration: ECR integrates deeply with AWS services like ECS, EKS, and IAM, while Docker Hub is a third-party registry.
- **Security:** ECR supports IAM-based access control and encryption at rest by default.
- **Private by default:** ECR repositories are private unless you configure otherwise; Docker Hub has both public and private tiers.
- **Networking:** ECR works well with VPC endpoints for private access, which isn't natively available in Docker Hub.

### 2. What are the key features of ECR?

#### Answer:

Key features of Amazon ECR include:

- Fully Managed: No need to manage infrastructure or registry servers.
- IAM-based Access Control: Fine-grained permissions using AWS IAM.
- **Image Scanning:** Detect vulnerabilities in your images using Amazon Inspector or native ECR scanning.
- High Availability and Scalability: Designed to support high-scale CI/CD pipelines.
- Lifecycle Policies: Automatically expire older images to reduce storage costs.
- VPC Endpoints: Secure access within your private VPC.
- Encryption at Rest & In-Transit: Data is encrypted using AWS KMS and HTTPS.

#### 3. How do you authenticate Docker to ECR?

#### STAR

**S:** In one of our microservices deployments using EKS, we needed our CI/CD pipeline to authenticate and push images securely to ECR.

**T:** The challenge was to authenticate Docker with ECR in a secure and automated way during the build process.

#### A:

We used the AWS CLI to get a temporary Docker login token: bash

\_\_\_\_

aws ecr get-login-password --region us-east-1 | docker login --username AWS -- password-stdin <account-id>.dkr.ecr.us-east-1.amazonaws.com

In Jenkins/GitHub Actions, we added this to the build script to automate authentication.

**R:** The authentication worked seamlessly in the pipeline, and we were able to securely push/pull images as part of the CI/CD flow.

Tools: AWS CLI, Docker, GitHub Actions, Jenkins.

## 4. What are the different types of ECR repositories?

#### **Answer:**

There are two types of ECR repositories:

- 1. Private Repositories (default):
  - Used for internal image storage.
  - o Access controlled via IAM policies.
- 2. Public Repositories (Amazon ECR Public):
  - Used to share container images with the broader community.
  - o Offers features similar to Docker Hub's public repositories.
  - Useful when you want global distribution of images without requiring AWS credentials.

## 5. How do you push and pull images to/from ECR?

## **Answer (START Format):**

**S:** During the deployment of a Node.js app on ECS Fargate, we needed to push a Docker image to ECR and later pull it from the ECS task definition.

**T:** The objective was to automate push/pull of Docker images during the CI/CD workflow.

A:

1. Tag the image:

bash

----

docker tag my-app:latest <account-id>.dkr.ecr.us-east-1.amazonaws.com/my-app:latest

2. Push the image:

bash

----

docker push <account-id>.dkr.ecr.us-east-1.amazonaws.com/my-app:latest

3. Pull the image (used by ECS/EKS automatically after proper IAM permissions are set):

bash

----

docker pull <account-id>.dkr.ecr.us-east-1.amazonaws.com/my-app:latest

**R:** This allowed smooth image versioning and deployment across environments like dev, staging, and production.

Tools: Docker, AWS CLI, GitHub Actions, ECS.

# Step 2: Build Docker Image

bash

docker build -t my-ecr-app.

## ✓ Step 3: Create ECR Repository

bash

----

aws ecr create-repository --repository-name my-ecr-app Note the repository URI from the output: php-template

----

{

<aws\_account\_id>.dkr.ecr.<region>.amazonaws.com/my-ecr-app

```
"repository": {
    "repositoryArn": "arn:aws:ecr:us-east-2:906253564515:repository/my-ecr-app",
    "registryId": "906253564515",
    "repositoryName": "my-ecr-app",
    "repositoryUri": "906253564515.dkr.ecr.us-east-2.amazonaws.com/my-ecr-app",
    "createdAt": 1752575915.343,
    "imageTagMutability": "MUTABLE",
    "imageScanningConfiguration": {
        "scanOnPush": false
    },
    "encryptionConfiguration": {
        "encryptionType": "AES256"
    }
}
```

## Step 4: Authenticate Docker to ECR

bash

----

aws ecr get-login-password --region <your-region> | docker login --username AWS --password-stdin <aws\_account\_id>.dkr.ecr.<region>.amazonaws.com
Replace <your-region> and <aws\_account\_id> accordingly.

# Step 5: Tag the Image

bash

----

docker tag my-ecr-app:latest <aws\_account\_id>.dkr.ecr.<region>.amazonaws.com/my-ecr-app:latest



bash

docker push <aws\_account\_id>.dkr.ecr.<region>.amazonaws.com/my-ecr-app:latest You should now see the image in the ECR console under the "my-ecr-app" repository.



## Step 7 (Optional): Pull Image from ECR

bash

docker pull <aws\_account\_id>.dkr.ecr.<region>.amazonaws.com/my-ecr-app:latest

### Which of the following best describes how Amazon ECR differs from Docker Hub?

- A. ECR is hosted by Docker and Docker Hub is hosted by AWS
- B. ECR is public by default, Docker Hub is private
- C. ECR integrates deeply with AWS services like IAM, ECS, and EKS
- D. Docker Hub supports IAM roles



✓ Correct Answer: C

## Which of the following is a security feature of ECR that Docker Hub lacks natively?

- A. Built-in CI/CD pipelines
- B. VPC endpoint support for private access
- C. Unlimited public repositories
- D. Automatic application scaling



✓ Correct Answer: B

## What is the purpose of tagging an image before pushing to ECR?

- A. To scan the image
- B. To build the Dockerfile
- C. To identify the image with ECR-compatible URI
- D. To encrypt the image



Correct Answer: C