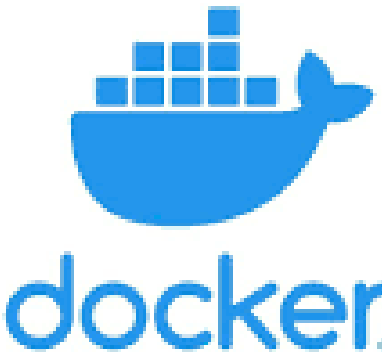
Notes

AWS Project Documentation

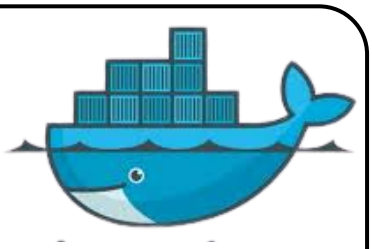
Deploy an App Across AWS Accounts



@sAnandha

Notes

DEPLOY AN APP WITH DOCKER

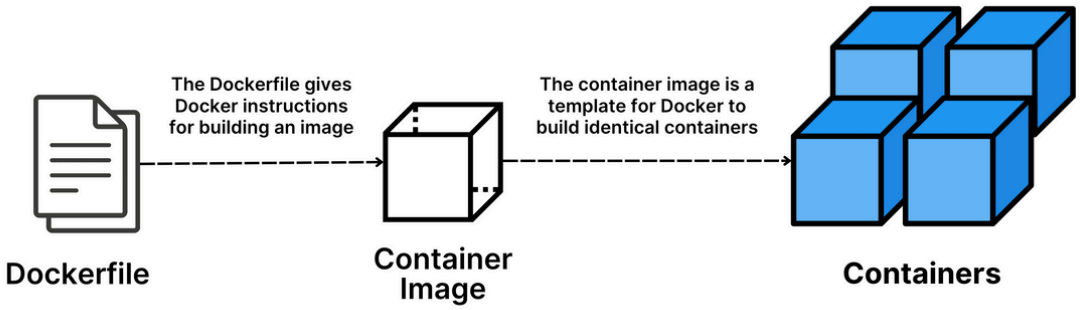


What is Docker?

Docker is a platform that enables developers to build, package, and distribute applications as containers. A container is a lightweight, standalone, executable package that includes everything needed to run an application.

Components in Docker Workflow

1. Dockerfile: The Dockerfile defines the steps needed to create a Docker image, from setting up the base image to copying the application code into the image and exposing the ports.
2. Docker Image: The image is the packaged version of your application, including the environment and configuration necessary to run it.
3. Docker Container: A running instance of the Docker image. Once the image is pulled, it runs in a containerized environment, executing the application.
4. Docker Compose: A tool for defining and running multi-container Docker applications. It can be useful for orchestrating the deployment of multiple services.

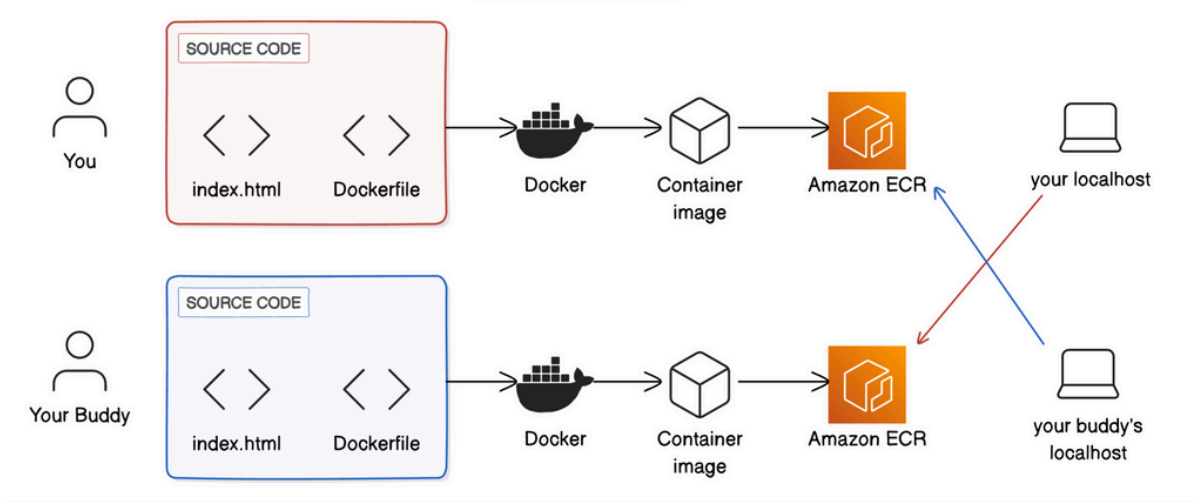


What is Amazon ECR?

- Amazon ECR is a fully managed container registry that allows you to store, manage, and deploy Docker container images. It eliminates the need to manage your own registry infrastructure.
- ECR integrates with Amazon ECS, Amazon EKS, and other AWS services, providing a seamless way to deploy Docker containers at scale.

- **ECR Usage**
  - Store Docker images securely.
  - Version control for images.
  - Secure authentication and authorization for accessing images.
  - Support for automated image scanning for vulnerabilities.

ARCHITECTURE DIAGRAM :



Architecture Breakdown:

I. Development and Packaging:

- **Source Code:** Each developer (You and Your Buddy) has their own source code, including application files (e.g., index.html) and a Dockerfile.
- **Dockerfile:** This file contains instructions to build a Docker image of the application, defining the environment, dependencies, and execution commands.
- **Docker:** The Docker engine is used to build the container image based on the Dockerfile. This encapsulates the application and its dependencies into a single, portable unit.
- **Container Image:** The output of the Docker build process is a container image, a lightweight and executable package of software that includes everything needed to run the application.

II. Centralized Image Storage and Distribution:

- **Amazon Elastic Container Registry (ECR):** This is a fully managed Docker container registry service provided by AWS.
- **Pushing Images:** Both you and your buddy push their respective container images to the same ECR repository. This creates a centralized location for storing and sharing the images.

III. Deployment and Execution:

- **Pulling Images:** You and your buddy can pull (download) each other's container images from ECR to your local machines.
- **Localhost:** Docker runs the pulled container images on your respective local machines, accessible via localhost. This allows for local development, testing, and execution of the shared application.

Key Components and their Roles:

- **Source Code**: The application code and configuration files.
- **Dockerfile:** Instructions for building the Docker image.
- **Docker Engine**: The tool used to build and run container images.
- **Container Image**: A packaged application with all its dependencies.
- **Amazon ECR**: A centralized repository for storing and sharing container images.

Steps to Deploy the App Across AWS Accounts

1. **Prepare the Application Files**
   - You need to have a basic application, such as an index.html file, that you want to serve via the containerized application.
2. **Create the Dockerfile**
   - Purpose: The Dockerfile defines the steps needed to build your Docker image.
   - It typically starts with a base image, copies the application files, installs necessary dependencies, and sets up the container environment.
   - **Steps in Dockerfile Creation:**
     - Step 1: Choose a Base Image
       - Start with an appropriate base image for your app, like nginx for serving static content.
     - Step 2: Copy the Application Files
       - Copy your index.html into the image.
     - Step 3: Expose Necessary Ports
       - If your app uses HTTP, expose port 80.

3. **Build the Docker Image**
   - Purpose: After creating the Dockerfile, the next step is to build the Docker image that will be used to deploy the application.
   - Steps:
     - Run docker build to create the image.
     - Tag the image with a name to push it to the ECR registry.
4. **Create an Amazon ECR Repository**
   - Purpose: ECR is where the Docker images are stored and managed.
   - Steps:
     - Log into the AWS Management Console.
     - Go to the ECR service and create a repository.
     - Set permissions, like using IAM roles, to control access to the ECR repository.
     - Note the repository URI (used when pushing/pulling images).
5. **Authenticate Docker to Amazon ECR**
   - Purpose: Before pushing an image to ECR, you need to authenticate Docker with AWS.
   - Steps:
     - Use AWS CLI to get authentication credentials for Docker (aws ecr get-login-password).
     - Configure Docker to use these credentials for the ECR repository.

6. Push the Docker Image to Amazon ECR
   - Purpose: After building the Docker image, push it to ECR so that it can be pulled by other accounts.
   - Steps:
     - Tag the image with the ECR repository URI.
     - Push the image to ECR using docker push.

7. Pull the Docker Image from ECR in Another AWS Account
   - Purpose: Once the Docker image is in ECR, it can be accessed from other AWS accounts.
   - Steps:
     - Ensure the AWS account has the necessary permissions to pull from the ECR repository.
     - Use the docker pull command to retrieve the image from ECR.

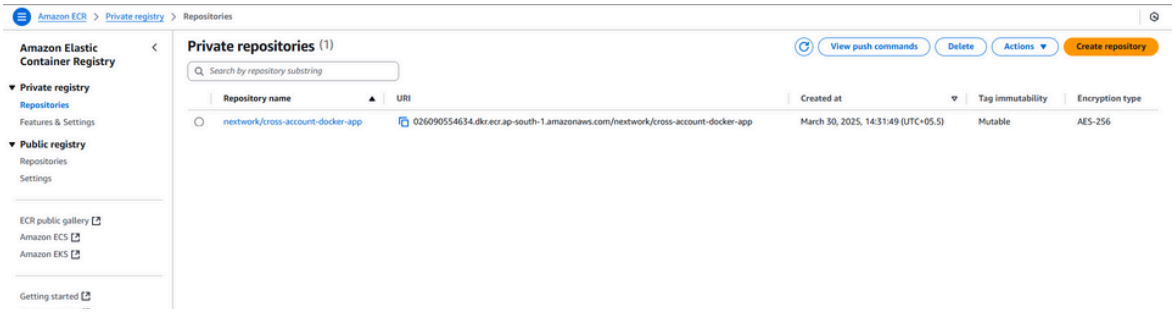
8. Run the Docker Container in Another AWS Account
   - Purpose: Pull the image and deploy it in an AWS environment (ECS, EC2).
   - Steps:
     - Create an ECS Task Definition that specifies the image location in ECR.
     - Launch ECS services to pull the image and run the container.

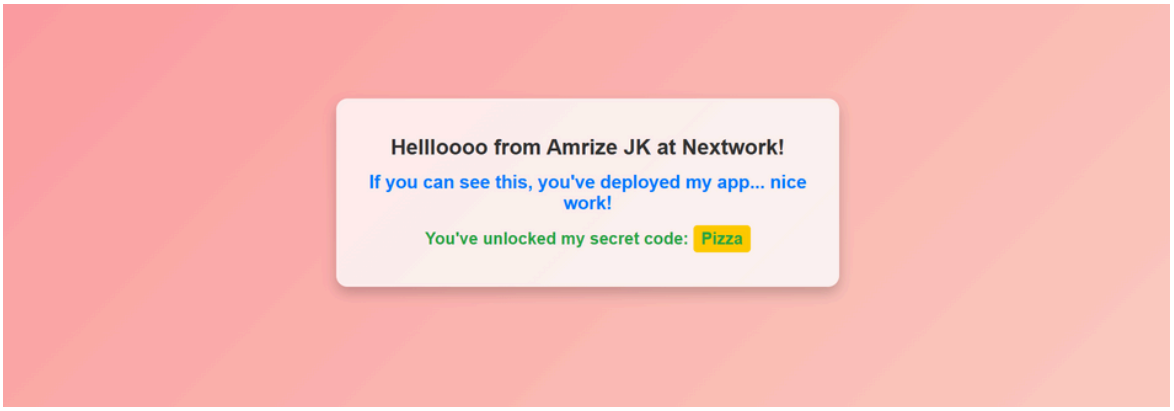
Notes

SAMPLE OUTPUT :

```
Start a build

C:\Users\Anandha Nivas\OneDrive\Desktop\aws\DockerECR>docker build -t cross-account-docker-app .
[+] Building 4.3s (7/7) FINISHED                                                    docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                0.1s
 => => transferring dockerfile: 96B                                                                 0.0s
 => [internal] load metadata for docker.io/library/nginx:latest                                     3.6s
 => [internal] load .dockerignore                                                                   0.1s
 => => transferring context: 2B                                                                     0.0s
 => [internal] load build context                                                                   0.1s
 => => transferring context: 1.05kB                                                                 0.0s
 => CACHED [1/2] FROM docker.io/library/nginx:latest@sha256:124b44bfc9ccd1f3cedf4b592d4d1e8bddb78b51ec2ed5056c52d  0.0s
 => [2/2] COPY index.html /usr/share/nginx/html/                                                    0.1s
 => exporting to image                                                                              0.1s
 => => exporting layers                                                                             0.1s
 => => writing image sha256:e701b9c2cd6c48587d175bc61427ecae6dba95ffdf1fb7944f8fbcf293534a8e        0.0s
 => => naming to docker.io/library/cross-account-docker-app                                         0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/ijo319zv640ih589ljg0mg23m

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview

C:\Users\Anandha Nivas\OneDrive\Desktop\aws\DockerECR>
```

```
docker/welcome-to-docker                                  docker push 026090554634.dkr.ecr.ap-south-1.amazonaws.com/nextwork/cross-account-docker-app:latest
PS C:\Users\Anandha Nivas\OneDrive\Desktop\aws\DockerECR>
The push refers to repository [026090554634.dkr.ecr.ap-south-1.amazonaws.com/nextwork/cross-account-docker-app]
9dd16419bbf2: Pushed
03d9365bc5dc: Pushed
d26dc06ef910: Pushed
aa82c57cd9fe: Pushed
d98dcc720ae0: Pushed
ad2f08e39a9d: Pushed
135f786ad046: Pushed
1287fbecdfcc: Pushed
latest: digest: sha256:5dd17891057e7317e5cd34000b459be88a894377158d4553568f80cd8d1ede77 size: 1985
PS C:\Users\Anandha Nivas\OneDrive\Desktop\aws\DockerECR>
```





For References :
https://learn.nextwork.org/projects/aws-compute-ecr?track=low