Deploying an application **first to Amazon ECR (Elastic Container Registry)** and then running it on **EC2 (Elastic Compute Cloud)** is a **common DevOps workflow** when working with Docker containers. Here's **why this two-step process is needed**:

### **🔹 What is ECR?**

Amazon **ECR** is a **managed Docker container registry** where you can **store, version, and manage Docker images**.

### **🔹 Why Deploy to ECR First?**

1. **Central Image Repository**:

* ECR acts as a centralized and secure place to **store container images**.
* Once uploaded, the image can be accessed by **multiple EC2 instances**, ECS tasks, or even other AWS services.

1. **Scalability**:

* By uploading the image once to ECR, it can be pulled by many instances or containers, instead of copying files manually.

1. **CI/CD Compatibility**:

* Tools like GitHub Actions, Jenkins, CodePipeline, etc., push Docker images to ECR as part of automated pipelines.

### **🔹 Then Why Run on EC2?**

1. **EC2 is the Compute Layer**:

* EC2 runs your **actual application** (e.g., the Docker container).
* It pulls the image from ECR and **executes it** using Docker.

1. **Decoupling Build and Runtime**:

* You **build** and store your application (image) in ECR.
* You **run** your application on EC2 (or ECS, EKS, etc.).

1. **Flexibility**:

* EC2 lets you customize the server environment (RAM, CPU, OS).
* You can use it for containerized workloads that require custom setup or non-standard configurations.

### **🔁 Workflow Summary:**

1. Build Docker image

2. Push image to Amazon ECR

3. SSH into EC2 instance (or use a startup script)

4. Pull the image from ECR using:

docker pull <ECR\_URI>

5. Run the container on EC2:

docker run <image\_name>

### **✅ Benefits of This Flow:**

* Secure and versioned image management
* Reuse the same image across environments
* Easier rollback (by using previous image versions)
* Better integration with CI/CD and automation tools

Let me know if you'd like the same setup using ECS instead of EC2 — that's even more automated.