CI/CD Pipeline

# **CI/CD Pipeline and Environments**

## **Introduction**

The software development process has evolved significantly over the years. In 2007, teams usually worked with a few members and integrated code changes manually under the supervision of a tech lead. The integration process was scheduled from Monday to Friday, where developers checked in their code and merged it with the main branch. However, as applications grew complex, automation became essential. This is where CI/CD pipelines came into practice, bringing automation, efficiency, and reliability into software delivery.

## **CI/CD Pipelines**

A CI/CD pipeline is a sequence of automated steps that ensure smooth software development, testing, and deployment. It stands for Continuous Integration and Continuous Deployment/Delivery.

### **Continuous Integration (CI)**

Continuous Integration is the process where developers frequently check in their code. Each change triggers an automated build, test, and packaging process. The main steps include:  
• Check-in the code  
• Code is merged  
• Build process is executed  
• Package is created  
**Continuous Deployment / Delivery (CD)**

After CI completes successfully, Continuous Deployment or Delivery comes into play. The package created in CI is deployed into a server environment. Continuous Delivery requires manual approval before production deployment, while Continuous Deployment automates the entire process.

## **Environments in CI/CD**

### **1. Development Environment (Dev Env)**

The development environment is where developers write and test their code locally. It serves as the starting point of the CI/CD pipeline.

### **2. Quality Assurance Environment (QA Env)**

In the QA environment, the testing team validates the application. They test for bugs, provide feedback, and once issues are fixed and retested successfully, the application is marked GREEN (stable).

### **3. Staging / Integration Environment**

The staging environment is also called the integration environment. Here, end-to-end features like order processing, payment workflows, and integration with external services are validated by testers. This environment closely resembles production.

### **4. User Acceptance Testing (UAT)**

UAT is performed to validate the system from the end-user perspective. Stakeholders and business users test whether the system meets business requirements. This ensures the product is ready for production.

### **5. Pre-Production Environment (PPE)**

The Pre-Production Environment mirrors 99% of the production environment. It is used for final validation before going live to ensure performance, scalability, and reliability.

### **6. Production Environment (Prod Env)**

The production environment is the LIVE system used by end-users. After passing all the previous environments, the application is finally deployed here.

## **Benefits of CI/CD**

• Faster and more reliable releases  
• Early bug detection  
• Reduced manual intervention  
• Consistency across environments  
• Higher developer productivity  
• Better collaboration between development and operations teams  
**Conclusion**

CI/CD pipelines have become the backbone of modern software development. They provide a streamlined approach for code integration, testing, and deployment across multiple environments. From development to production, CI/CD ensures that software is delivered efficiently, reliably, and with minimal errors.

**Continuous Integration (CI)**

Continuous Integration is the process where developers frequently check in their code. Each change triggers an automated build, test, and packaging process. The main steps include:

* Check-in the code
* Code is merged
* Build process is executed
* Package is created

**Why Multiple Developer Teams?**

In real-world applications, one team cannot handle all the features of a large system. For example, consider the Amazon portal — it has multiple features such as product search, payment, order tracking, delivery, return, and refund processes. A single team cannot manage all of these efficiently.

Therefore, there are **multiple developer teams**, each working on a specific feature or module. One team may work on the payment process, another on delivery, another on returns, and so on. These teams collaborate, share details, and integrate their work.

The code developed by each team initially works only in their **local system (localhost)**. Through **Continuous Integration (CI)**, this code is merged into a shared repository, built, and tested, ensuring that all features work together seamlessly.

**Continuous Deployment / Delivery (CD)**

After integration and testing, the next step is to make the new features available to end-users. This is the role of Continuous Deployment/Delivery. The package built by CI is deployed into the server so that it becomes live for real users.

**IN SIMPLE WORDS:**

**CI** – Developers build specific features in their local systems, which initially work only on localhost. Since one team cannot handle all features, multiple teams work on different parts (e.g., payment, delivery, returns, refunds in an Amazon portal). Teams coordinate by sharing details with each other to complete the process. After testing, the next step is **CD**, where the feature is deployed and made live for users as part of the CI/CD pipeline.

