CI/CD and IAC

Continuous Integration :

Continuous Integration (CI) is a software development practice that encourages developers to integrate their code changes into a shared repository frequently, ideally multiple times per day. Each integration is then automatically built and tested to validate the code and identify any potential issues, such as bugs, conflicts, or inconsistencies. By implementing this practice, development teams can collaborate more efficiently, improve code quality, and minimize the time taken to find and fix bugs.

Benefits :

Developer Collaboration

CI helps facilitate communication and collaboration between developers on a team. When multiple developers work on the same project, integrating their code changes becomes a challenge. With CI, developers can merge their code changes regularly, allowing them to identify and resolve conflicts early in the development process. This promotes a more efficient workflow and a seamless exchange of ideas among the team members.

Improved Code Quality

Frequent integrations and automated testing help maintain high code quality throughout the project. By integrating code changes frequently, developers can detect and fix issues before they escalate. As a result, the overall quality of the software improves as potential problems are resolved quickly and efficiently.

Reduced Time to Find and Fix Bugs

When code changes are integrated frequently, bugs and other issues are identified earlier in the development process. This allows developers to address them more rapidly than in a traditional development workflow, where bugs might not be discovered until much later. Early bug detection reduces the time required to fix these issues and ensures that the software remains robust and reliable.

Continuous Delivery

Continuous Delivery is the practice of ensuring that every code change is ready to be deployed to production at any given moment. By automating the various stages of the software delivery pipeline, such as testing, packaging, and configuration, CD allows organizations to release new features and updates more rapidly, providing a faster response to market demands and end-user needs.

It is important to note that while Continuous Delivery ensures that your code is always in a deployable state, it doesn't mean that every change should be immediately deployed to production. The decision to deploy a change is often a manual one, allowing teams to choose when and how they release new features or updates.

Continuous Deployment

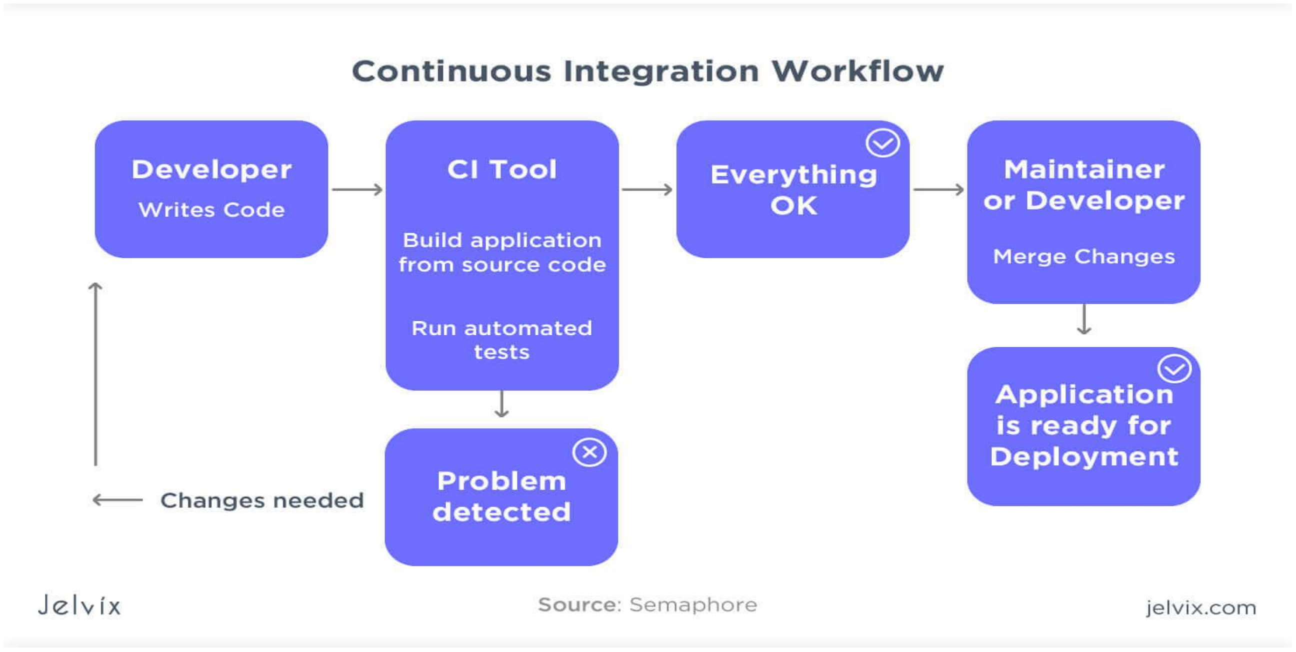
Continuous Deployment takes the Continuous Delivery process one step further by automating the final step of deploying every code change to production, assuming it has passed all previous stages of the pipeline. This means that every new feature, bug fix, or update is automatically released to end-users or production environments without any manual intervention.

This level of automation can bring significant benefits to organizations, enabling them to deliver new functionality and improvements to their users faster than ever before. However, it also requires a high degree of confidence in the testing, monitoring, and rollback mechanisms in place to ensure that any issues arising during deployment can be quickly detected and resolved without causing disruption to end-users.

CI Workflow

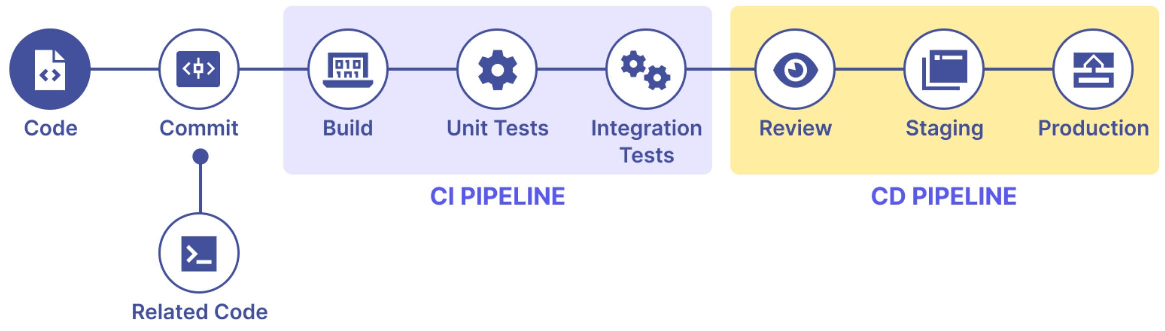
Once the developer commits their code to a version control system like Git, it triggers the CI pipeline which fetches the changes and runs automated build and unit tests. Based on the status of the step, the server then notifies the concerned developer whether the integration of the new code to the existing code base was a success or a failure.

This helps in finding and addressing the bugs much more quickly, makes the team more productive by freeing the developers from manual tasks, and helps teams deliver updates to their customers more frequently. It has been found that integrating the entire development cycle can reduce the developer's time involved by ~25 – 30%.



CI and CD Workflow

The below image describes how Continuous Integration combined with Continuous Delivery helps quicken the software delivery process with lower risks and improved quality.



We have seen how Continuous Integration automates the process of building, testing, and packaging the source code as soon as it is committed to the code repository by the developers. Once the CI step is completed, the code is deployed to the staging environment where it undergoes further automated testing (like Acceptance testing, Regression testing, etc.). Finally, it is deployed to the production environment for the final release of the product.

If the deployment to production is a manual step. In that case, the process is called Continuous Delivery whereas if the deployment to the production environment is automated, it is referred to as Continuous Deployment.

Popular CI/CD Tools :

1. **Jenkins :** An open-source CI/CD tool that automates the building, testing, and deployment of your applications. Jenkins supports a wide range of plugins and integrations, making it easy to scale and customize your CI/CD pipeline.
2. **GitLab CI/CD :** An integrated CI/CD solution within GitLab that simplifies the process of setting up and managing pipelines. GitLab offers an all-in-one solution for source code management, issue tracking, and CI/CD.
3. **GitHub Actions** : CI/CD solution integrated with GitHub repositories. It uses workflows defined in YAML files for automating builds, testing, and deployments.
4. **CircleCI :** A cloud-based CI/CD platform that offers features such as parallel execution, caching, and containerization. CircleCI integrates with various version control systems, including GitHub and Bitbucket.
5. **Azure DevOps (Pipelines) :** Azure Pipelines is a cloud-based CI/CD service by Microsoft that supports building, testing, and deploying applications to any platform or cloud. It integrates with Azure DevOps for end-to-end project management.
6. **AWS CodePipeline :** A fully managed continuous delivery service from Amazon Web Services (AWS) that helps you automate your release pipelines. It integrates seamlessly with other AWS services like CodeBuild, CodeDeploy, and CodeCommit.
7. **Bitbucket Pipelines** : A CI/CD service built into Bitbucket, Atlassian's Git repository management tool. It allows developers to automate build, test, and deployment workflows directly from their Bitbucket repositories**.**
8. **ArgoCD :** A declarative, GitOps-based continuous delivery tool for Kubernetes. ArgoCD synchronizes Kubernetes configurations defined in Git repositories with the actual cluster state.

Infrastructure as Code(IAC)

A critical component of both Continuous Delivery and Continuous Deployment is the concept of Infrastructure as Code (IaC). IaC allows development teams to manage and provision their IT infrastructure using code, rather than relying on manual processes and procedures. By treating infrastructure in the same way as application source code, organizations can leverage version control, automated testing, and other Devops practices to ensure that their infrastructure is consistently and reliably deployed.

Popular IAC Tools:

1. Terraform

An open-source tool by HashiCorp that allows you to define and provision infrastructure across various cloud providers using declarative code.

2. AWS CloudFormation

A native AWS service that enables you to model and provision AWS infrastructure using JSON or YAML templates.

3. Azure Resource Manager (ARM)

Microsoft Azure's native tool for deploying and managing infrastructure using JSON templates for declarative configurations.

4. Google Cloud Deployment Manager

A tool for managing Google Cloud resources using declarative YAML or Python configuration files.

5. Ansible

A configuration management and orchestration tool that automates software provisioning, configuration management, and application deployment using YAML playbooks.

6. Chef

A configuration management tool that uses Ruby-based scripts to automate the provisioning and configuration of infrastructure.

7. Puppet

A configuration management tool that automates the delivery and management of software and infrastructure using a declarative language.