WHAT IS CI/CD? CONTINUOUS INTEGRATION & CONTINUOUS DELIVERY EXPLAINED

Flexibility, speed, and quality are the core pillars of modern software development. Increased customer demand and the evolving technological landscape have made software development more complex than ever, making traditional software development lifecycle (SDLC) methods unable to cope with the rapidly changing nature of developments.

Practices like Agile and DevOps have gained popularity in facilitating these changing requirements by bringing flexibility and speed to the development process without sacrificing the overall quality of the end product.

Together, Continuous Integration (CD) and Continuous Delivery (CD) is a key aspect that helps in this regard. It allows users to build integrated development pipelines that spread from development to production deployments across the software development process. So, what exactly are Continuous Integration and Continuous Delivery? Let's take a look.

(This article is part of our DevOps Guide. Use the right-hand menu to navigate.)

What is CI/CD?

CI/CD refers to Continuous Integration and Continuous Delivery. In its simplest form, CI/CD introduces automation and monitoring to the complete SDLC.

- Continuous Integration can be considered the first part of a software delivery pipeline where application code is integrated, built, and tested.
- Continuous Delivery is the second stage of a delivery pipeline where the application is

deployed to its production environment to be utilized by the end-users.

Let's deep dive into CI and CD in the following sections.

What is Continuous Integration?

Modern software development is a team effort with multiple developers working on different areas, features, or bug fixes of a product. All these code changes need to be combined to release a single end product. However, manually integrating all these changes can be a near-impossible task, and there will inevitably be conflicting code changes with developers working on multiple changes.

Continuous Integrations offer the ideal solution for this issue by allowing developers to continuously push their code to the version control system (VCS). These changes are validated, and new builds are created from the new code that will undergo automated testing.

This testing will typically include unit and integration tests to ensure that the changes do not cause any issues in the application. It also ensures that all code changes are properly validated, tested, and immediate feedback is provided to the developer from the pipeline in the event of an issue enabling them to fix that issue quickly.

This not only increases the quality of the code but also provides a platform to quickly identify code errors with a shorter automated feedback cycle. Another benefit of Continuous Integrations is that it ensures all developers have the latest codebase to work on as code changes are quickly merged, further mitigating merge conflicts.

The end goal of the continuous integration process is to create a deployable artifact.

What is Continuous Delivery?

Once a deployable artifact is created, the next stage of the software development process is to deploy this artifact to the production environment. Continuous delivery comes into play to address this need by automating the entire delivery process.

Continuous Delivery is responsible for the application deployment as well as infrastructure and configuration changes, monitoring and maintaining the application. CD can extend its functionally to include operational responsibilities such as infrastructure management using automation tools such as:

- Terraform
- Ansible
- Chef
- Puppet

Continuous Delivery also supports multi-stage deployments where artifacts are moved through different stages like staging, pre-production, and finally to production with additional testing and verifications at each stage. These additional testing and verification further increase the reliability and robustness of the application.

Why we need CI/CD

CI/CD is the backbone of all modern software developments allowing organizations to develop and

deploy software quickly and efficiently. It offers a unified platform to integrate all aspects of the SDLC, including separate tools and platforms from source control, testing tools to infrastructure modification, and monitoring tools.

A properly configured CI/CD pipeline allows organizations to adapt to changing consumer needs and technological innovations easily. In a traditional development strategy, fulfilling changes requested by clients or adapting new technology will be a long-winded process. Moreover, the consumer need may also have shifted when the organization tries to adapt to the change. Approaches like DevOps with CI/CD solve this issue as CI/CD pipelines are much more flexible.

For example: suppose there is a consumer requirement that is not currently addressed with a DevOps approach. In that case, it can be quickly identified, analyzed, developed, and deployed to the software product in a relatively short amount of time without disrupting the normal development flow of the application.

Another aspect is that CI/CD enables quick deployment of even small changes to the end product, quickly addressing user needs. It not only resolves user needs but also provides visibility of the development process to the end-user. End-users can see that the product grows with frequent deployments related to bug fixes or new features.

This is in stark contrast with traditional approaches like the waterfall model, where the end-users only see the final product after the complete development is done.

CI/CD today

CI/CD has come a long way since its inception, where it began only as a platform to support application delivery. Now it has evolved to support other aspects, such as:

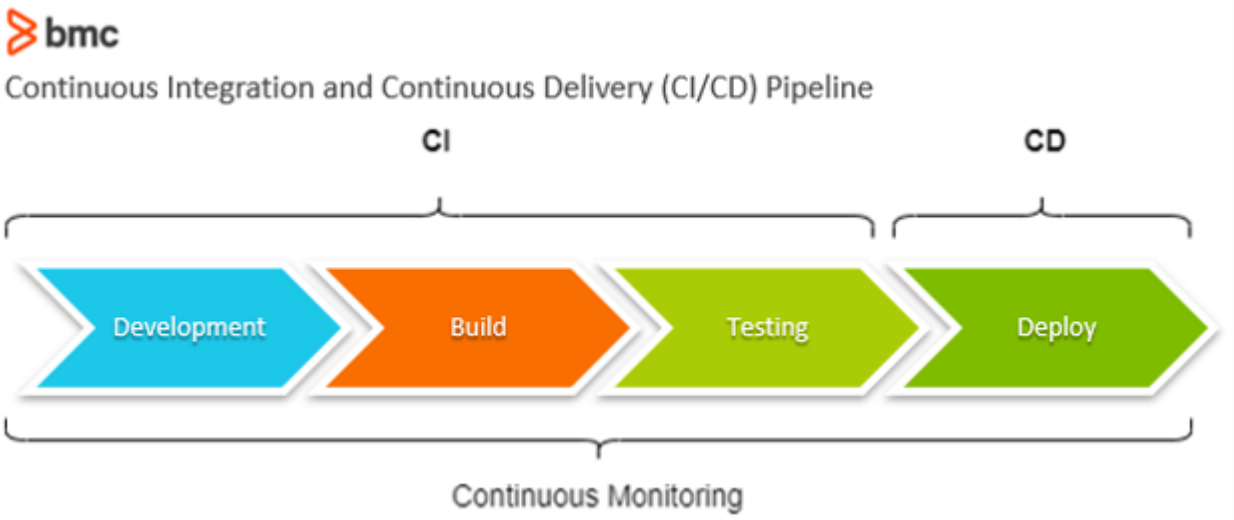
- Database DevOps, where database changes are continuously delivered.
- GitOps, where infrastructure is defined in a declarative version-controlled manner to be managed via CI/CD pipelines.

Thus, users can integrate almost all aspects of the software delivery into Continuous Integration and Continuous Delivery. Furthermore, CI/CD can also extend itself to DevSecOps, where security testing such as vulnerability scans, configuration policy enforcements, network monitoring, etc., can be directly integrated into CI/CD pipelines.

CI/CD pipeline & workflows

CI/CD pipeline is a software delivery process created through Continuous Integration and Continuous Delivery platforms. The complexity and the stages of the CI/CD pipeline vary depending on the development requirements.

Properly setting up CI/CD pipeline is the key to benefitting from all the advantages offered by CI/CD. One pipeline might have a multi-stage deployment strategy that delivers software as containers to a multi-cloud Kubernetes cluster, and another may be a simple pipeline that builds, tests, and deploys the application as a serverless function.



A typical CI/CD pipeline can be broken down into the following stages:

1. **Development.** This stage is where the development happens, and the code is merged to a version control repository and validated.
2. **Build.** The application is built using the validated code, and this artifact is used for testing.
3. **Testing.** Usually, the built artifact is deployed to a test environment, and extensive tests are carried out to ensure the functionality of the application.
4. **Deploy.** This is the final stage of the pipeline, where the tested application is deployed to the production environment.

All the above stages are continuously monitored for any errors and quickly notified to the relevant parties.

Advantages of Continuous Integration & Delivery

CI/CD undoubtedly increases the speed and the efficiency of the software development process while providing a top-down view of all the tasks involved in the delivery process. On top of that, CI/CD will have the following benefits reaching all aspects of the organization..

- **Improve developer and QA productivity** by introducing automated validations, builds, and testing
- **Save time and resources** by automating mundane and repeatable tasks
- **Improve overall code quality**
- **Increase the feedback cycles** with each stage and the process in the pipeline being continuously monitored
- **Reduce the bugs or defects** in the system
- **Provide the ability to support other areas of application delivery**, such as database and infrastructure changes directly through the pipeline
- **Support varying architectures and platforms** from traditional server-based deployment to container and serverless architectures
- **Ensure the application's reliability**, thanks to the ability to monitor the application in the production environment with continuous monitoring

CI/CD tools & platforms

When it comes to CI/CD tools and platforms, there are many choices ranging from simple CI/CD platforms to specialized tools that support a specific architecture. There are even tools and services directly available through source control systems. Let's look at some of the popular CI/CD tools and platforms.

Continuous Integration tools & platforms

- Jenkins
- TeamCity
- Travis CI
- Bamboo
- CircleCI

Continuous Delivery tools & platforms

- ArgoCD
- JenkinsX
- FluxCD
- GoCD
- Spinnaker
- Octopus Deploy

Cloud-Based CI/CD

- Azure DevOps
- Google Cloud Build
- AWS CodeBuild/CodeCommit/CodeDeploy
- GitHub Actions
- GitLab Pipelines
- Bitbucket Pipelines

Summing up CI/CD

Continuous Integration and Continuous Delivery have become an integral part of most software development lifecycles. With continuous development, testing, and deployment, CI/CD has enabled faster, more flexible development without increasing the workload of development, quality assurance, or the operations teams.

Today, CI/CD has evolved to support all aspects of the delivery pipelines, thus also facilitating new paradigms such as GitOps, Database DevOps, DevSecOps, etc.—and we can expect more to come.

BMC supports Enterprise DevOps

From legacy systems to cloud software, BMC supports DevOps across the enter enterprise. Learn more about Enterprise DevOps.

Related reading

- BMC DevOps Blog
- DevOps Branching Strategies Explained
- What's CD4ML? Continuous Delivery with Machine Learning Explained
- DevOps Job Titles, Roles, & Responsibilities
- The Complete DevOps Certifications Guide 2021-2022
- Control-M Python Client & Integrations