What is a CI/CD Pipeline?

A CI/CD pipeline is used to automate software or infrastructure-as-code delivery, from source code to production. You can think of it as a set of orchestrated steps that guide your code from development to deployment.

Continuous Integration (CI): Continuous integration (CI) helps to ensure that software components are working together. Integration should be completed frequently; on an hourly or daily basis, if possible.

In CI practice, developers build, run, and test code on their own workstations before they commit code to the version control repository (GitHub, GitLab etc). After changes are made to the repository, a chain of events is set in motion. A typical first step in this chain is the creation of the latest version of the source code. If the build is successful, the unit tests will be carried out. If unit testing is successful, the build is deployed in test environments where system testing is performed (usually by automated testing). The team is notified of the status of this process and is provided with a report to provide details, such as build number, defects, and number of tests. Continuous integration (CI) helps ensure that the software is fully integrated.

Typically, the CI pipeline involves the following tasks:

- Detect changes to the source code repository (new commits appear)
- Analysis of the quality of source code
- Project build
- Perform all unit tests
- Run all integration tests
- Generate deployable artifacts
- Status of report

If one of the steps above fails:

- Integration may stop or continue depending on the severity and configuration of the defect.
- Results are reported to the team via email or chat system.
- Team fixes the defect and commits it again
- Tasks are carried out again

Continuous Delivery / Deployment (CD): takes place after testing. Continuous delivery refers to automatically releasing code to a repository, while continuous deployment involves automatically deploying thoroughly tested artifacts. Continuous delivery (CD) picks up where continuous integration is over. While CI is the process to build and test automatically, CD deploys all code changes to the testing or staging environment in the build.

CD enables builds to be released to the production environment when needed. Allowing the team to deploy on its own, the CD effectively reduces time on the market.

Before deploying software for production, the CD process includes automated system testing, unit testing, and integration testing. The steps from CI to CD are usually completed automatically, including automated unit testing, integration, and system level. As testing can fail at any level and environment, CI / CD must include a feedback channel to quickly report failures to developers. Dependent on policies and processes defined by teams, developers may do the following with CI/CD:

A CI / CD pipeline is a method for delivering a change unit that begins from development to delivery, usually consisting of the following main phases:

Phase 1: Commit

When the developers make a change, they commit the change to the repository.

Phase 2: Build

The source code in the repository is integrated into the build.

Phase 3: Automate tests

Automated tests are being run against the build. Test automation is a key component of any CI / CD pipeline.

Phase 4: Deploy

The built version will be delivered to production.

Conclusion

CI / CD are two of the best practices of DevOps in addressing misalignment between developers and the operational team. With the presence of automation, developers can release changes and new features more frequently, while operation teams have better overall stability. I hope after reading this article you may have fair understanding of CI / CD.