Continuous Integration & Continuous Delivery

DevOps has come the new methods of Continuous Integration, Continuous Delivery, (CI/CD) and Continuous Deployment. Understand How CI/CD helps in automating on the Cloud.

With the rise of DevOps has come the new methods of Continuous Integration, Continuous Delivery, (CI/CD) and Continuous Deployment.

As discussed in the previous topic conventional software development and delivery methods are rapidly becoming obsolete. Historically, in the agile age, most companies would deploy/ship software in monthly, quarterly, bi-annual, or even annual releases. Now however, in the DevOps era, weekly, daily, and even multiple times a day is the norm.

With the DevOps paradigm shift most teams have automated processes to check in code and deploy to new environments. This has been coupled with a focus on automating the process along the way.

Continuous Integration (CI)

With continuous integration, developers frequently integrate their code into a main branch of a common repository. Rather than building features in isolation and submitting each of them at the end of the cycle, a developer will strive to contribute software work products to the repository several times on any given day.

The big idea here is to reduce integration costs by having developers do it sooner and more frequently. In practice, a developer will often discover boundary conflicts between new and existing code at the time of integration. If it's done early and often, the expectation is that such conflict resolutions will be easier and less costly to perform.

Of course, there are trade-offs. This process change does not provide any additional quality assurances. Indeed, many organizations find that such integration becomes more costly since they rely on manual procedures to ensure that new code doesn't introduce new bugs, and doesn't break existing

code. To reduce friction during integration tasks, continuous integration relies on test suites and an automated test execution. It's important, however, to realize that automated testing is quite different from continuous testing.

The goal of CI is to refine integration into a simple, easily-repeatable everyday development task that will serve to reduce overall build costs and reveal defects early in the cycle. Success in CI will depend on changes to the culture of the development team so that there is incentive for readiness, frequent and iterative builds, and eagerness to deal with bugs when they are found much earlier.

Continuous Delivery (CD)

Continuous delivery is actually an extension of CI, in which the software delivery process is automated further to enable easily and confident deployments into production at any time. A mature continuous delivery process exhibits a codebase that is always deployable on the spot. With CD, software release becomes a routine event without emotion or urgency. Teams proceed with daily development tasks in the confidence that they can build a production-grade release any old time they please without elaborate orchestration .

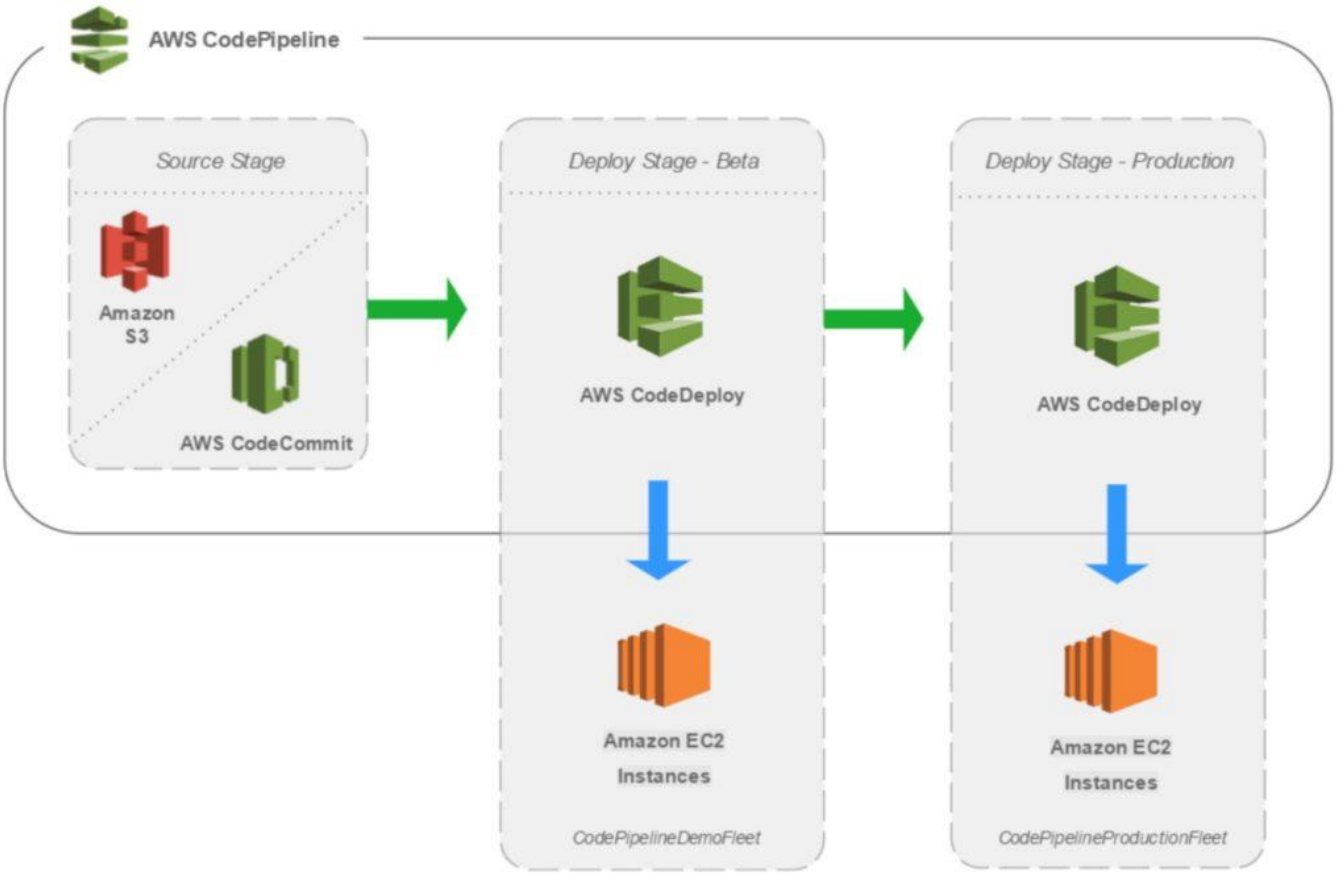
CD depends centrally on a deployment pipeline by which the team automates the testing and deployment processes. This pipeline is an automated system that executes a progressive set of test suites against the build. CD is highly automatable and in some cloud-computing environments easily configurable.

In each segment in the pipeline, the build may fail a critical test and alert the team. Otherwise, it continues on to the next test suite, and successive test passes will result in automatic promotion to the next segment in the pipeline. The last segment in the pipeline will deploy the build to a production-equivalent environment. This is a comprehensive activity, since the build, the deployment, and the environment are all exercised and tested together. The result is a build that is deployable and verifiable in an actual production environment.

A solid exhibit of a modern CI/CD pipeline is available on on the cloud with AWS. Amazon is one of the cloud-computing providers that offers an impressive CI/CD pipeline environment, and provides a walk-through

procedure in which you can choose from among its many development

resources and link them together in a pipeline that is readily configurable and easily monitored.



Continuous deployment extends continuous delivery so that the software build will automatically deploy if it passes all tests. In such a process, there is no need for a person to decide when and what goes into production.

The last step in a CI/CD system will automatically deploy whatever build components/packages successfully exit the delivery pipeline. Such automatic deployments can be configured to quickly distribute components, features, and fixes to customers, and provide clarity on precisely what has is presently in production.

Organizations that employ continuous deployment will likely benefit from very quick user feedback on new deployments. Features are quickly delivered to users, and any defects that become evident can be handled promptly. Quick user response on unhelpful or misunderstood features will help the team refocus and avoid devoting more effort into to functional area that is unlikely to produce a good return return on that investment.