CI/CD Framework

"Technology is like a fish. The longer it stays on the shelf, the less desirable it becomes."

Andrew Heller nailed it well. Technology has a tendency, after all, to move towards the oblivion of a shelf, unless this is actively addressed and redressed by both its makers and users. The task becomes more complicated in the modern software age where development and release of an application acquire absolutely-staggering levels of complexity, challenges and process-snags.

This brings to fore the element of Continuous-Integration and Delivery. When work from developers is integrated into a main repository multiple times a day; and when steps to release a code or deploy a build are automated, not only one grasps deficiencies as well as integration snags in time but also irons out a lot of delays and friction.

Automatically deploying a code each time change is made; empowers the process with a new degree of speed and consistency, thereby, spurring developers to weave in the code into the main branch of a shared repository—both early and often. With this, a huge chunk of isolation, which otherwise gets unwieldy, can be sorted out; and issues relegated to end-of-a-development cycle can be taken up head-on and early-on.

The results of making integration simple, consistent, and repeatable– reducing integration costs, timely discovery of problem-areas, high levels of quality assurance and smooth dissolution of conflicts.

Extending the same tenets into the delivery landscape brings in ready automation of software delivery process, keeping the code in a perennial-deployable state; and subsequent code-

deployment into production without any time-hassles. Now software can be released whenever needed without any incidental complexity or hiccups.

With the use of well-tested, functional builds to the production infrastructure; Continuous Delivery ensures accuracy, purity, robustness and deployment-mettle of a code but without compromising on innate flexibility and agility.

This process becomes easier when a flurry of open-source tools and hands make it move away from shelves fast enough.

Top 10 Open Source Platforms

1. ***Jenkins***

A CI Framework that can effortlessly sit at the core of an organisation's CI/CD efforts by allowing the creation of hourly or daily builds automatically, and deployment of builds to QA or production environment. As a standalone build automation system, it is quite a sought-after option and is reinventing the field of CI/CD builds with its strengths and speed. It stands out as a good open-source project for automating projects with clear and consistent time-side advantages. Its usage spans bug detection, code analysis, building projects, running tests and project deployment.

2. Gradle

A relatively-new kid on the block, Gradle accumulated lot of advocates and fans given its smooth and swift performance with build-automation service. It is supposed that it has found usage at giants like Netflix, Google and LinkedIn too.

3. Travis

It is a CI platform for automation of testing and deployment areas and can also readily integrate with GitHub projects. Reported usage from the likes of Facebook, Mozilla, Twitter etc. cement its spot as another major Continuous-Integration tool around.

4. CircleCI

A strong platform for integration and delivery, Circle CI has proved productivity advantages as well as scalability through its traction so far. It covers Build-Automation, Test-automation, and a lot of ground on deployment process.

5. Codeship

It is a hosted version of a Continuous-Integration platform but the flexibility it accords along with pace, simplicity, alignment with both integration and delivery—underlined with its quintessential efficiency- is quite remarkable.

Ability to work with Bitbucket and a compact string of features add to its attractiveness.

6. GitLab

For modern developers, this happens to be catching a lot of steam. It is a code-management platform with a good repository of single-dashboard tools for issue management, code views, continuous integration and deployment, and tracking of project's maturity.

Its impressive installation-speed, a sleek UI and scope for pre-built packages for popular Linux distributions are also quite the eye-catchers. All this is further augmented with its good grip on documentation and project view.

7. Buildbot

What began as a lightweight-tool vis a vis Mozilla Tinderbox, is now evolving strongly as a software-development Continuous-Integration tool. Users like Mozilla, Chromium, and WebKit are giving a lot of credibility and confidence to this Python-based tool.

Its complete functionality for Continuous-Development and Build-Server supporting a variety of integrations is a strong factor. Its friendliness and ease of multiple-channel notifications, distributed builds, Python-runs, POSLX-compliant OS, version-control configurations and custom build steps elevate its capabilities manifold.

8. CruiseControl

This continuous integration server resonates with high-levels of maturity and longevity of existence. It dates back 15+ years when it was conceived as a framework for orchestrating build, testing and other software development lifecycle at ThoughtWorks. But now it has sufficiently and aptly evolved as a standalone Continuous-Integration server. It is amenable to cross-platform usage and implementations in .NET and Ruby.

9. Drone

This brings to the field the excitement of containers-first architecture and a strong equation and readiness for Docker builds. This container-based tool makes for a good coordinating layer between Docker and a repository provider. It can be run as a container and is bolstered with support for multiple database back-ends, repository providers and some noteworthy TLS/SSL certificates.

10. Concourse

New to the playground, this Continuous-Integration platform packs a fresh and radical approach in being as invisible as

possible, thus shrinking the state and abstracting external factors into resources. With the integration-server entirely disposable, the ease and effectiveness of running processes on Concourse server go up impressively. Clear definition of dependencies, smooth construction of pipelines and a lucid-definition of behaviour make it a choice different from others.

Benefits & Future

Bringing in flexibility and automation to software-development and release is now becoming a core area of attention. This landscape is also evolving strongly to trends and implications arising around performance-testing, production-ready deployment, automated-results analysis and the advent of Docker. App-distribution and support have undertaken new contours and forces. The role and attention to user-interface are also assuming new proportions every single day, with every new problem or breakthrough that unfolds in this industry.

Open-Source is paving a new way and making technologies come to the nearest shelf—handy and fast.