International Journal of Research in Engineering, Science and Management
Volume 4, Issue 6, June 2021
https://www.ijresm.com | ISSN (Online): 2581-5792

Path to Success with CICD Pipeline Delivery

Rakhi Parashar*
Tech Arch Delivery Team Lead, LKM Intelligent Platforms Delivery, Accenture, Mumbai, India

***Abstract*: Continuous Integration and Continuous Deployment is two very important practices in DevOps. Continuous Integration (CI) is a key player during Agile Development and DevOps processes. Here, Dedicated User stories is assigned, and respective builds developed by different developers (under a regular development environment) and tasks of other teams are delivered commonly to a team build server. All the individual work is then integrated in a common build area to form an integration single build. The process then moves across higher environment build server and then applied system-wide or application-wide i.e. in Production Environment. This way the continuous integrations are done, and builds happen, making the Continuous Integration pipeline implemented. Continuous Integration (CI) is considered as a key practice in SDLC (Software Development Life Cycle's), where software changes (CRs/user stories/tasks) are implemented in isolation, tested immediately and reported when they are added to a larger code base. The objective of CI is to ensure timely detection of errors/bugs/defects reported during the product lifecycle, address them and provide feedback to correct the same. Thus, DevOps lifts deployment more frequently and provides more opportunities to re-assess the delivery process, via automation, effective testing and monitoring strategy. DevOps CICD practices provide valuable data for CI (Continuous Improvement) around monitoring and metrics. Ideally CI must be a part of every DevOps process, irrespective of organizational scale or size, and should certainly be driven by a solid Quality Assurance (QA) strategy. Moving on to Continuous Delivery (CD), DevOps relation with the CD pipeline spins around the new features that developers work with and those released to customers, in a timely manner. All the builds that pass-through QA need not go into production. Only those with functional stability can be moved to production environment and will then become 'production-ready' before staging. This Best practice of regular delivery of applications (under development environment) to QA and Operations for validation, and potential release to clients is known as a Continuous Delivery (CD).**

***Keywords*: Continuous Integration and Continuous Deployment (CICD), DevOps, SDLC, Agile.**

1. Introduction

We are in Digital World and Speed always wins in Digital Revolution, each and every enterprise needs to move faster and more flexibly in application development, testing and QA to keep up with the Client's demand and expectation and be competitive in the market. Hence this requires integrating the tasks done by their development and QA teams and reducing manual processes wherever possible and automate most of them. This new way of working can effectively improve Employee's productivity, bring new revenue networks and customer satisfaction. Continuous Integration (CI) and Continuous Delivery (CD) are discipline and implemented as a practice by agile organizations to speed their development and test processes to meet the new business demands. The benefits of this approach include reduced time, risk, and expense of software delivery. This paper looks at the benefits of CI and CD together and the requirements for effectively deploying these practices in IT organization.

2. Literature Review

A. Continuous Integration and Delivery (CI/CD)

Continuous Integration (CI) is a development practice that requires developers to integrate i.e. merge their code changes into a shared version controlling repository several times a day. Each check-in done by developer is then verified by an automated build, allowing teams to detect problems early.CI/CD is a set of software practices and techniques that enable the frequent releases of small sets of feature changes, the process is visible to everyone and also traceable. It involves the making of a largely automated pipeline that orchestrates the build, test and deployment of software across staged environments, ultimately leading to deployment in production.

Traditionally, developers used to build the application features in silos and submitted them separately. This new CI way has completely changed how developers work and share their code changes with the master branch. With CI, the developers frequently integrate the code changes into a central version controlling repository several times a day. As a result, merging the different code changes becomes easier and is also less time-consuming. Integration bugs can be detected and resolved early.

Continuous delivery (CD) is about incremental delivery of updates/software changes to production. It serves as an extension to CI; CD enables automation our entire software release process. It widens our horizon and allows you to look beyond just the unit tests and perform other tests such as integration tests, functional tests, performance tests, security tests, UI tests, etc... This upshot the developers to perform a more complete validation on updates to ensure bug-free deployment. With CD implemented we can have frequent releases of new features, which boosts the customer feedback, and also will have healthier customer involvement. Hence, CI/CD serve as cornerstones to any DevOps pipeline.

*Corresponding author: rakhi.parashar@accenture.com

Table 1
DevOps Tools RoadMap

Requirement Gathering	**Plan/ Analyze/ Design**	**Development/Coding/ Implementation**	**Build**	**Artifacts Management Repos**	**Testing**	**Create Environment/ Deployment**	**Support/ Maintenance/ Monitoring**
JIRA	Microsoft	SCM Tools: Git, SVN, CVS, Mercurial, Bazaar, Bitkeeper, etc.	Maven	Nexus	Junit	Shell Scripts	AWS Services like Cloud Watch, SNS
TFS	Draw.io		Gradle	JFrog Artifactory	Jmeter	Mutable Type of Infra: Chef, Ansible, Puppet, Saltstack, opscode, Terraform, etc..	ELK, EFK
HP ALM		Cloud/Server/Host based Repository: Github, Bitbucket, Gerrit, Gitlab, Code Commit service of AWS, etc..	Ant	Gemfury	TestNG	Immutable Type of Infra: Docker Containers, OpenShift, Kubernetes,etc..	Grafana, Prometheus and cAdvisor
Rally			MSBuild	archiva	Selenium		AppDynamics
Pivotal Tracker			Make		Load Runner		Splunk
ColabNet					Cucumber		
Continuous Integration Tool/Orchestration: Jenkins, Bamboo, Teamcity, Gitlab, Bamboo, Continuum, etc..							

Table 2
CICD Roles and Required Skillsets

CI/CD Roles	Required Skillsets
✓ No automated builds or automated tests in place, we have work with Engineering and QA to define them. ✓ Help IT setup, CICD pipeline setup and plan, design and execute application and server maintenance/monitoring.	➢ Writing pipeline as code i.e. jenkinsfile. Groovy e jenkinsfile, Groovy scripting. VM Configurations. Understanding of software execution workflow configurations and dependency managements. ➢ Jenkins associated plug-ins and their usages. Different types of testing automations. ➢ Learn how to write .YML files which help in write configurations Gitlab i.e. .gitlab-ci.yml.

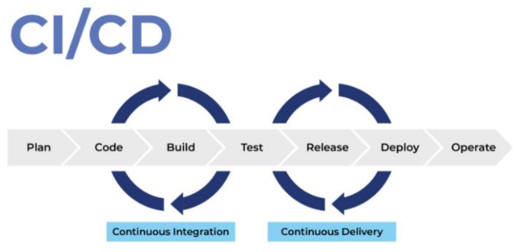

Fig. 1. CICD Pipeline

3. Benefits of CI/CD

CI/CD offers many significant benefits across SDCL in the IT organizations as below:

- Rapid identification and resolution of defects.
- Reduced overhead costs.
- Improved quality assurance.
- Assumptions can be reduced.
- Speed to market.
- Software health can be tracked.
- Better project process visibility.
- Core to Business Agility.

A. *CI/CD Tool Landscape*

There are excess of tools that can add value to your CI/CD pipeline but selecting the right tool and getting them integrated and working together requires time, good planning, and specialized expertise.

4. Continuous Delivery Impacts Entire SDLC – Leads to New Role Requirements and Ownerships

CI/CD impacts across the entire software development lifecycle i.e. to all the stages of SDLC. Following this approach in IT organization has created new Job Role (CICD Engineer, DevOps Engineer, DevOps Architect/DevOps Lead Engineer, etc...) compared to traditional roles we had earlier e.g. developer/tester/QA. As a result, it is very important to define roles and responsibilities clearly to avoid misunderstanding and accountability gaps.

5. CI/CD Design Principles

Assurance to success is CI/CD Best design practices and tools. Selection of right tool weather it for Build phase, testing, deployment/infrastructure/Monitoring phase of SDLC, it is key step to accelerate the CICD process. Below are the design principles for CICD:

CI/CD Design Principles:

- Self Service - Speed to adoption
- Best-practice - One Standard
- Configurable Technology Integration - Separation of concern

- Flexible Process - Separation of concern, Continuity
- Extensible - Best-of-breed, Inclusion, Innovation
- Cloud Ready/Elastic
- HA/DR – Fidelity

6. Conclusion

In this research path of CICD we are highlighting popular CI tools like Jenkins, Bamboo, Teamcity, Gitlab. Jenkins and Gitlab are open source tools whereas we need to buy license for Bamboo and Teamcity. To implement CICD pipelines in Jenkins, you can configure Jobs manually first for Build, Test and Deploy and then create a pipeline from those Jobs. To eradicate this manual effort, we can write Pipeline as code using Jenkinsfile which can contain Pipeline Flow script for all the stages of you SDLC. This will help in managing and running jobs or pipeline for multiples source repositories and multi-branch also. Jenkins offer 3 ways to automate this process: 1. Declarative Pipline 2. Scripted Pipline 3. Pipeline via Jenkinsfile script. Groovy language knowledge helps in writing these Pipeline as Code. Alternate to Jenkins we have another tool as Gitlab. Gitlab offers many features like: 1. Synchronize collaborations with Plan i.e. Jira like features for tracing all Issues/CRs/User Stories. 2. Build better code and branch powerfully with Create i.e. It has Web IDE to write the code and file synching to web terminal which helps test code changes in a preconfigured terminal environment and Design Management.3. Build and share packages in Package Managers. 4. Continous delivery in simpler with Release i.e. CICD Pipeline via writing the code in .gitlab-ci.yml, Auto DevOps. 5. Support for multiple Kubernetes clusters i.e. Easily deploy your applications to different environments like Dev,Test,Staging,QA and Production to different Kubernetes clusters.6. Network policies for container network security. Gitlab is a one stop solution for all tools as mentioned above in DevOps Tools Roadmap. Gitlab is limitless and is a treat for all users as a single application for entire DevOps lifecycle.

References

[1] https://www.infostretch.com/resources/white-papers/continuous-integration-and-delivery/
[2] https://www.veritis.com/wp-content/uploads/2016/09/devops-a-success-ful-path-to-continuous-integration-and-continuous-delivery-white-paper.pdf
[3] https://www.plutora.com/devops-at-scale/pipeline
[4] https://www.linkedin.com/pulse/15-ultimate-devops-quotes-digital-transformation-pavan-belagatti/
[5] https://www3.dbmaestro.com/blog/18-great-devops-quotes.
[6] https://about.gitlab.com/blog/2020/03/30/new-features-to-core/
[7] https://medium.com/faun/most-popular-ci-cd-pipelines-and-tools-ccfdce429867