levi nine

WHITE PAPER

Continuous Delivery

Content

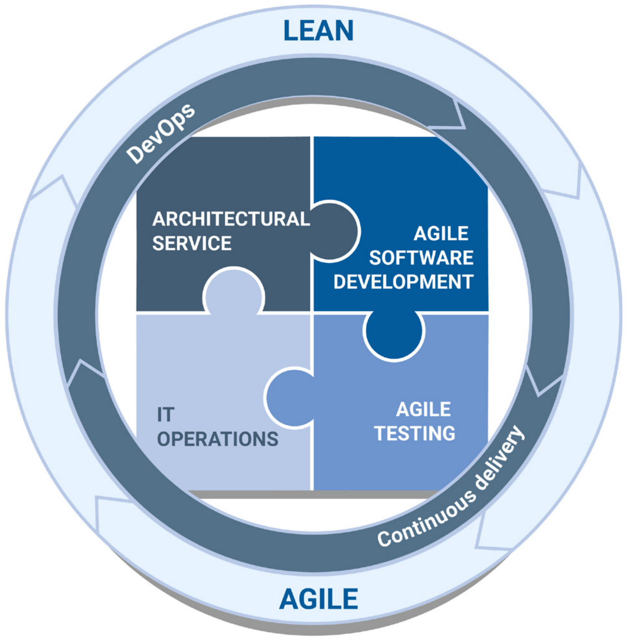
Introduction

Releasing software is often a long, difficult and risky process. Defects and integration issues pop-up at the very last moment and cause dissatisfaction to the development teams and even worse to end users. Furthermore, lack of collaboration between the software development and IT operations teams generally results in integration and deployment errors and finger-pointing.

Continuous Delivery can solve all of that and add even more.

Continuous Delivery can help IT organizations to become lean and agile. It can help them to continuously adapt software according to the user feedback and to stay ahead of changing market conditions, while improving the quality and speed of delivery. Continuous Delivery makes the delivery of the software, from the hands of the developers to production environment a repeatable, reliable, visible and largely automated process with well understood and quantifiable risks.

Continuous Delivery creates an organized evolution of the software development process.



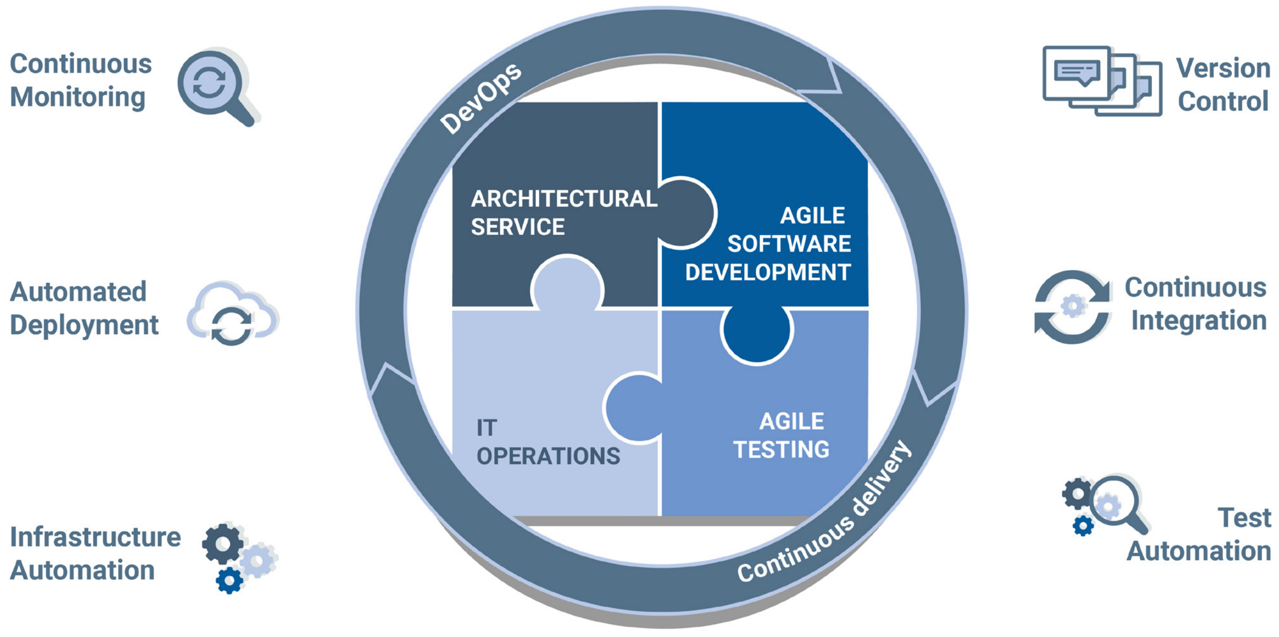
"Our highest priority is to satisfy the Customer through early and Continuous Delivery of valuable software."

Principle #1, Agile Manifesto

How to Achieve This

Saying that it is all about practices, tools and people would not be saying anything new. Practices such as Version Control and Continuous Integration have been in existence for quite some time.

Practices such as Test Automation, Infrastructure Automation, Automated Deployment and Continuous Monitoring have developed significantly over the last few years. The complete set of these practices, fully integrated, we call Continuous Delivery.



Source code and configuration changes are checked-in to a Version Control System and each check-in (potentially) triggers a Continuous Integration build. Each integration is verified by an automated build and by unit and integration tests to detect component and integration errors as quickly as possible. This approach leads to a reduced number of integration problems, and allows the teams to develop software more rapidly. If the build is broken, all ongoing activities stop until the problem is solved and the build becomes releasable again.

Test Automation as a practice enables automated execution of test scripts and comparison of actual with expected results. Test Automation encompasses unit, integration, functional, performance and security tests and reduces the need for manual tests to a minimum. This requires skills from both testers as well as developers and their continuous teamwork, but also that quality is embedded in the architecture of the software itself.

Infrastructure Automation enables automated provisioning and configuration of hosting environments and forms the bridge between Continuous Integration and Test Automation. With modern cloud and container possibilities, this enables us to spin-up quickly testing environment from the code, run our tests and destroy the environment when automated tests are finished. Even more, we are able to run as many testing environments as we need and ease dependencies between the teams and between different software modules.

Finally, the so called “last mile” of the deployment pipeline is covered by Automated Deployment. This part of the process can be fully or semi-automated. Full automation of the deployments is usually used in the case of testing environments, while in the case of acceptance or production environment we potentially want to keep additional level of control which includes manual step where an operator selects the application version and the deployment environment. Continuous System and Application Monitoring enables a quick response to outages and decreases time to recover.

Most importantly, to make Continuous Delivery a success, the Development, Test and IT Operations teams need to work together as one delivery team from the very start.

Adopting Continuous Delivery

We have established a **Continuous Delivery Maturity Model** allowing us to help our customers to adopt Continuous Delivery practices in a pragmatic and gradual manner. We have been using this approach many times to help large customers from E-Retail, Digital Marketing and FinTech sector to increase quality and improve time to market.

The initial phase is an assessment of the current status. This step is quantifiable, to show to what extent you have already adopted the various Continuous Delivery practices.

The second phase is to decide what you would like to aim for, both in the short and long term, and which benefits the introduction of some of the Continuous Delivery practices should achieve for you. This must be done realistically and pragmatically, higher is not always better or cost-effective. We aim for quick wins first and work with development teams on enabling "more difficult" aspects later in the process.

Based on the assessment and the pragmatic ambition, a clear plan is created where improvements are prioritized and evaluations are planned for re-assessing the value achieved by the improvements as compared with the set goals. Our approach is to introduce elements of Continuous Delivery gradually and to measure the benefits of each step.

The approach we take not only affects the Software Development process (in its widest sense) but may also result in recommendations for improvements in software architecture or changes in the organizational aspect of the software development process.

With this approach we aim is to reduce time to market and take the out the risk from the release process.

First goal is to have all source code and application configuration versioned in the source code repository. In that step we would establish clear practices about versioning the code, we would define and implement together a branching and tagging strategy which would give more transparency over the work of the development teams.

Second goal would be to establish Continuous Integration. This would be done by use of integrated and automated tooling such as continuous integration server, artifact repository and source code quality server. This would enable us to continuously monitor the technical quality of the software and work on the areas identified as technical debt. Furthermore, this step would enable us to take care of dependencies much better and potentially use software techniques like feature toggles that would enable added-value to the business (for instance A/B Testing of features in production).

As the third step, we would establish Automated Deployment from the artifact repository server to the various application environments.

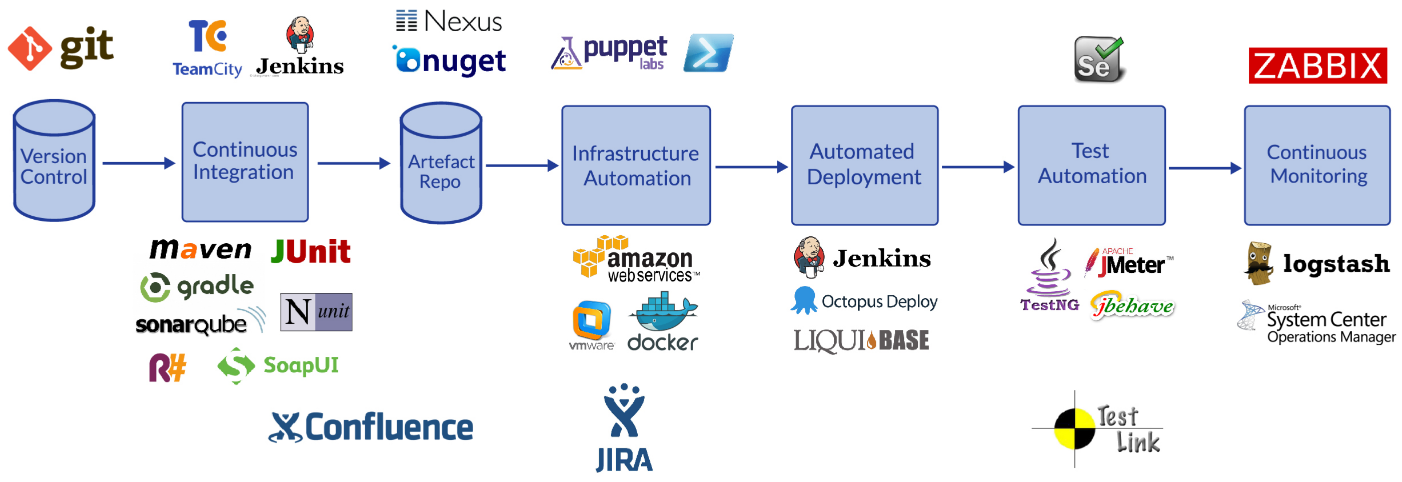
Fourth element to tackle would be Infrastructure Provisioning and Configuration Management. This step would be used to script the environments and applications, describe configurations in the code which would enable us to automatically provision fully configured environments anytime we need them. We would start from testing environment first and gradually promote to acceptance and production.

In the final step we would establish Continuous Monitoring and setup Log Management. This step would be enabled by previous actions in the software itself (establishing monitoring hooks in the application) and in the configuration (collecting monitoring resources). The same principles apply to log management, which would enable secure access of the development teams to application and system logs.

Continuous Delivery Maturity Level	**INITIAL**	**NOVICE**	**INTERMEDIATE**	**ADVANCED**
Version Control	Source code	Unit tests Application configuration code	Database schema Functional tests and test data Clear branching and merging strategy Embedding traceability	Infrastructure configuration code Main branch always deployable Automated release notes
Integration	Manual or scripted builds	Scheduled automated builds (nightly) Dependency management	Continuous integration of source code and unit tests (commit based) Artifact repository	CI build cluster CI orchestrated infrastructure provisioning Continuous integration of acceptance tests
Quality	Manual testing	Test management Automated unit and integration tests	Automated functional tests Static code analysis Facilitated peer reviews	Automated acceptance tests Automated non-functional tests (performance, security) Test cluster
Deployment	Manual Manual database schema changes	Scripted Same process across different environments	Automated Deployment pipeline Gated automated promotions	Orchestrated deployments One-click deployments Continuous deployments to production
Infrastructure	Physical Manual provisioning and configuration	Virtual Highly available, redundant with fail-over	Cloud Automated provisioning Infrastructure as a Code	Self-service for experimentation Container based infrastructure Network automation
Monitoring	No monitoring	System level monitoring	Application level monitoring Log management Real-time centralized logs monitoring	Application performance monitoring Anomaly detection Dynamic scaling
Architecture	Tightly coupled	De-coupled Covered with unit tests	Service oriented Fit for purpose frameworks Testable end-to-end Enables monitoring	Component based Scalable Micro-services
Organization	Waterfall approach Silo structure	Iterative approach Direct collaboration Requirements management	Agile Lean	Multi-functional teams DevOps Technical debt addressed regularly

Recommended Tooling

Based on our extensive experience, we have compiled a toolset that enables us to establish Continuous Delivery end-to-end. This toolset is supported by real-life examples, knowledge and experience with installation, configuration and integration into coherent deployment pipeline.



Category	JAVA, PHP, ANDROID	.NET
Version Control System	Git	Git, TFS
Continuous Integration	Jenkins, Bamboo, Nexus, Maven, Gradle	TeamCity, TFS, NuGet
Test Automation and Code Quality	Selenium, TestNG, Cucumber, REST Assured, soapUI, JMeter SonarQube, Phabricator, Crucible	Selenium, TestNG, ReSharper, FxCop, Review Assistant
Automated Deployment	Jenkins, Liquibase	Octopus Deploy, RedGate
Infrastructure Automation and Configuration Management	Puppet, VMware, Amazon, Docker	VMware, Amazon, OpenStack, Azure, Puppet, PowerShell DSC
Continuous Monitoring	Zabbix, AppDynamics, Graylog, Logstash, ElasticSearch, Kibana	Zabbix, SCOM
Team Collaboration	Jira, Confluence	Jira, Confluence, TFS

Conclusion

In its very essence, Continuous Delivery is the implementation of lean principles. It defines "Done" as released into production and reaping business benefits. It enables organizations to become truly agile. It provides fast feedback and requires a prompt response from the whole team. This increases quality since the defects are found early in the process, while they are still easy to fix.

Continuous Delivery automates, accelerates and integrates all of the processes needed to develop software. This embeds quality, shortens release cycle times, reduces the defect rate and increases the frequency of delivery. This results in better customer focus, faster time to market and improved reliability of the delivered solution.

At Levi Nine, we are very passionate about Continuous Delivery and we have been successful in helping our customers to implement Continuous Delivery in their organizations and be more successful on the competitive market.

"Continuous Delivery automates, accelerates and integrates all of the processes needed to develop software."

Levi Nine is a Dutch IT Services company with currently over 850 highly educated and skilled IT professionals in Eastern and Central Europe.

Our teams work remotely on software products and revenue generating systems for Dutch and International clients. We are the technology partner for clients that are looking for re-inforcement and a flexible layer of their own IT department, or are looking to co-source or outsource IT development and maintenance.

Levi Nine is strong in the area of common software R&D, software product development and client implementation projects. We strongly believe in DevOps principles, Infrastructure and Test Automation and Continuous Delivery.

We work for industry leaders such as TomTom, bol.com, Drukwerkdeal, WoodWing, Sanoma, Xerox and Swarco.