International Journal of Advances in Software Engineering & Research Methodology– IJSERM
Volume 2: Issue 2 [ISSN : 2374-1619]
Publication Date : 30 October, 2015

Continuous delivery as a service

An approach to implement continuous delivery as a service in large organizations

[Ionut Gheorghe Hrinca, Mircea Raducu Trifu]

***Abstract*—In agile software development small and incremental pieces of software are created and delivered to the stakeholders in order to validate their value. From the moment of finishing coding until the software is delivered to the client there is a repetitive quality assurance process that is the perfect candidate for automation. The mission of continuous delivery is to create a pipeline containing continuous integration, automated deployment and automated testing offering feedback at each stage in order to fix issues early in the process.**

Building and maintaining a continuous delivery pipeline requires a lot of effort. In case of large organizations with many development teams spread across the globe having replicated continuous delivery pipelines generate high costs compared to a central delivery pipeline offered as a service. Reusability and collaboration between development teams are some of the key topics that continuous delivery as a service is targeting. For building a central pipeline for the whole organization a lot of things need to be considered in order to be reliable, secure, efficient and easy to use and maintain.

***Keywords*—continuous delivery, continuous integration, code repository, build stage, autmated testing, automated deployment**

I. Introduction

Change is inevitable when using an information system in a large organization. To keep the business running and to preserve the competitive advantage, the organizations need to come with new ideas of improvement that most of the time turns into process automation or new functionality in the information system. Any practitioner can confirm that there is a long way from the idea to deploy in production of the development, moment when it starts creating value. One of the aspects that make this journey long is the fact that there are different teams for development and software operations with different skill sets. This makes communication to be hard when it comes to deploying the development on different environments. Also environment differences can make this journey longer and even can lead to production deployment issues.

In order to improve this process an organization needs to think to automate as much as possible and the testing and deployment are one of the best candidates for this, reducing the delivery time but also mitigating all the risks of human errors in the manual process. Building a continuous delivery process will reduce the delivery time.

The benefits of implementing a continuous delivery process are[3]:

- Enables collaboration among teams creating clarity at each step from the build, testing and deploy processes
- Offers immediate feedback and issues are identifies and solved early in the process
- Automated deployment on any environment.

Either in small companies or big international organization that develop software for internal use or for third parties, implementation of a CD pipeline became a must in order to have a successful agile development process.

If a company decides to use a CD it has two options. One option would be to implement its own CD pipeline and spend resources in managing the underlying infrastructure instead of concentrating on application development. The second option is to use is as a service. Second option looks to be suitable for small companies that don't have necessary resources to manage the CD pipeline infrastructure. Large, globally spread corporations might also go for a CDaaS because it reduces costs (one infrastructure) and enables cooperation and code reuse.

In this article we will describe an approach for implementation of a CDaaS in large organizations. The main focus will be on defining the key components of a CDaaS pipeline, a governance model and security aspects.

II. Why continuous delivery as a service?

A. *Factors that lead large organizations towards a CDaaS*

In a large organization spread across the globe one can often notice that in every branch, even if they use a quite large common set of applications, the implementations are quite different and that mainly because each branch develop in isolation. This has a significant impact on the reusability degree that makes the overall organization development cost to be high. The enterprise features that a CDaaS needs to deliver for hundreds of agile development teams are:

- Shared global, searchable version control system to foster re-use
- Backlog management and governance tooling across multiple team

Ionut Gheorghe Hrinca *(Author)*
PhD Economic Informatics - Bucharest University of Economic Studies
Bucharest, Romania

Mircea Raducu Trifu *(Author)*
PhD Economic Informatics - Bucharest University of Economic Studies
Romania

- Reporting and metrics across multiple teams to know where they are
- Knowledge sharing across teams
- Self service onboarding on the continuous delivery pipeline (tutorials, web casts, etc.) to avoid onboarding cost and to be able to scale up quickly

Building a continuous delivery pipeline is hard and expensive. Operating it involves also some costs too. There is a need for an operations team to manage the infrastructure and all the tools used by the pipeline. Having the continuous delivery pipeline replicated throughout the enterprise lead to increased costs and low collaboration between development teams.

B. *Impact on the organization*

Implementing a continuous delivery process is not an easy task to do for an organization. It takes a lot of effort and time to reach the end state and make it stable. This is the main reason any organization might ask from the first place why it wants to implement continuous delivery, what are the business reasons and what are the expected outcomes. The organization needs to be prepared to invest 2 to 3 years to reach the state of having continuous delivery[3]. Considering the amount of time required to build the process and the pipeline, the organization should avoid a classical approach of analyzing and planning in advance for a long period and only after that to start implementing the process. An incremental approach is preferable for this because it is hard (if not impossible) to state from the beginning how the end state will really look like.

Building a continuous delivery process generates also changes in almost all organizational levels. In order to avoid a failure it needs to be implemented in such a way that small incremental business value results can be presented to increase awareness of the benefits that continuous delivery brings. Getting feedback from all organizational levels is very important for adjusting the direction of the continuous delivery implementation.

C. *How to start implementing continuous delivery as a service*

First step for building a continuous delivery is to implement continuous integration. This step is the hardest to implement because of the resistance to change of the organization. If you think with which you should start the implementation of continuous integration, think about starting with the development teams and don't even consider to start with the operations teams because they are already overwhelmed with daily activities.

For implementing a successful continuous delivery process the organizational structure and the system architecture need to be considered. Another aspect with high importance is building the organizational culture around continuous delivery process and also around continuous improvement process.

In designing and using continuous delivery as a service the following things need to be considered [1]:

- Suitable applications for continuous delivery are the ones that follow the principle of "convention over configuration";
- Applications need to be designed and developed having in mind the continuous delivery
- Infrastructure needs to be treated in similar way as the code and must be automated;
- Choosing a convention based on service means that decisions related to continuous delivery infrastructure were already taken and the focus should be on how the applications should be designed to use the CDaaS.

III. Governance

Centralizing the continuous delivery process into a CDaaS for the whole organization involves also putting in place governance around it. There are many ways of setting up the governance model but we will focus on describing two options: central governance model and meritocracy model.

A. *Central governance model*

In central governance model a named team is responsible for the governance. There are advantages and disadvantages related to this model but one of the main reasons this model is not the most desirable one is that it can lead to creation of an "ivory tower". The risk of loosing the reality about the process is quite high and the governance team might take wrong decisions based on what they suppose it is happening in the continuous delivery process.

B. *Meritocratic governance model*

Meritocratic governance model is a commonly founded model in which the influence is gained through the recognitions of the contributors. One of the most famous examples of this model is the Apache Software Foundation. Executing this governance model is mainly based on collaboration between groups of individuals sharing common interests but also sharing "good to know" type of knowledge and solutions to different problems in regards to the continuous delivery pipeline offered as a service.

The more users are involved in this model and contribute sending suggestions, fixes, or just answering questions, the bigger becomes the community and its contribution.

Using this model, an individual or a team gain its right to be part of the development team of the CDaaS by playing an active role in the contribution process. In this way the access rights increases and, in a natural manner, this access rights increase is due to the evolved ability of the contributors facing solving different issues raised during operating the CDaaS pipeline.

The roles in the meritocracy model are:

- User - someone that uses the software
- Contributor - user that contributes in form of code or documentation
- Committer - developer that has the write access to the code repository

- PMC (project management committee) member - a contributor or a committer that was elected due to the merit of the evolution of the project.
- PMC chair - appointed by the board formed by the PMC members.

IV. Reference model for CDaaS

The current literature describe the following high level components of a continuous delivery pipeline [3]:

- Commit stage
- Automated testing stage
- Manual testing stage
- Delivery stage

In order to create an efficient continuous delivery pipeline as a service that will actually create value inside the organization, some key and enterprise specific components need to be present inside it:

- Exploring portal
- Metrics, analytics and monitoring portals
- Documentation and training portal.

The following diagram depicts at high-level model of a continuous delivery as a service implementation including some examples of solutions available for each component.

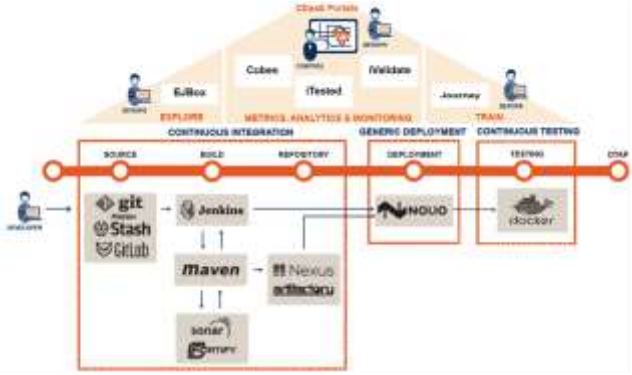

Figure 1. Continuous delivery as a service

A. *Continuous integration*

In order to have an enterprise level continuous integration inside the CDaaS that can be used by development teams spread around the globe, it must be designed on the following three pillars:

- Global version control system available to all development teams. Collaboration between the teams must be possible and encouraged
- Stateless build server. Anytime needed, a new instance of it should be possible to be created. In order to make this possible it should follow the following three principles: build server configuration must be kept in version control; new instances of the build server should be able to be updated easily; destroying and creating instances of the build server should happen without any loss.
- Global artifact repository. This is required in the build stage and in the test stage. One of the important artifact repository capability requirements is the ability to import libraries and artifacts from the Internet in a secured way. This capability is required because not always the software is in-house developed and artifacts and libraries could be developed and shipped by third parties.

Distributed nature of the development teams in organization makes the versioning control inside de CDaaS to require having specific features. One of the options for a distributed version control system is Git. The main advantage of Git is that it allows replication of the main line on the developers' working station allowing him to make local commits (even in offline mode). The developer can sync later on with the main version control server.

In order to have all the elements required by the first pillar that we mentioned for the continuous integration, Git alone or any other distributed version control system is not enough. There is a need for an additional tool (i.e. Stash) that fulfill the rest of the requirements:

- Creation and management of the code repositories
- Traceability in the development process
- Fine grained security model
- Collaboration tools at code level
- Scalability in order to be part of a organization wide CDaaS

The build server in a CDaaS needs to be stateless in order to assure high availability and scalability, but also to allow updates without any impact on the build process. The following diagram represents a more detailed view of the build stage.

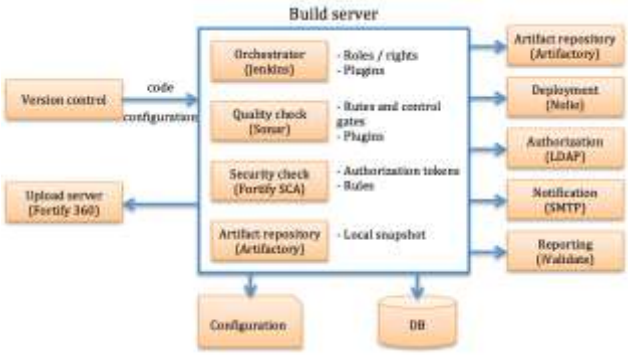

Figure 2. Architecture of the build stage

For choosing a solution for the artifact repository the following requirements need to be met:

- High availability deployment
- Secure access to the artifact repository
- Existence of an API for integration with other tools in the delivery pipeline (i.e. Jenkins, Nolio) for the following operations: artifact search, repository creation, users and groups management, etc.
- Sharing capabilities in order to define access to different repositories
- Tagging capabilities to facilitate artifacts finding
- Logging and monitoring capabilities

B. *Deployment*

In the continuous delivery pipeline the deployment must be automated. The deployment processes have to be configured in the tool (i.e. Nolio) in an environment agnostic way.

In this stage the artifacts created during the build stage are picked up from the artifact repository and, using the deployment processes, the development deployed on different environments. As a good practice, the environments have to be created to be as similar as possible to the production environment. This way the risk of something to go wrong during delivery to production is mitigated because the deployment process was also tested so many times on a similar environment.

In a CDaaS the deployment process needs to be as generic as possible. In order to accomplish that a manifest file describing the artifacts to be deployed, the order and constraints needs to be created and placed in the repository in order to be picked up at deployment time.

C. *Testing*

In a continuous delivery as a service the automated testing requires a powerful infrastructure. Every automated test must be run isolated in order avoid interference with other automated tests that run in parallel. Here, the interference refers to both functional (capabilities) and nonfunctional (performance, resilience, etc.) topics.

An option to isolate the testing is running them into so-called containers. A good candidate for doing that is Docker. A container uses a minimum set of libraries and resources required by the software to be tested. The hardware resources are used in an optimum way in contrast with the virtualization alternative. In this way the automated test can be scaled up and the infrastructure can support the load of having the automated test inside the continuous delivery pipeline offered as a service.

D. *Monitoring and analytics portal*

In order to deliver a continuous delivery pipeline service, monitoring and analytics are very important. Because of that, existence of a portal is crucial. Even the acceptance criteria for the new developments should be possible to be monitored through these portals (acceptance criteria can be managed by tools like iValidate). The continuous delivery process can check if all acceptance criteria are met in an automated way and the build could deploy directly to production. If there are manual steps defined, the process is waiting for the manual acknowledgment of the manual test completions.

Different metrics measurements need to be displayed on dashboards. These metrics can be grouped by DevOps teams or by application or system. Some metrics examples are:

- Average time of a development cycle
- Number of exceeded deadlines
- Median for issues per day
- Cost of infrastructure

V. Security

Security in a continuous delivery pipeline is a very important. At the organizational level there should be one security policy. The security around CDaaS needs to ensure the access based on the needs that a role must have. In order to have a central security for the whole organization around CDaaS there is a need for the following pre-conditions:

- Unique users database
- Single point for definition and search of user groups
- Existence of a way to map the IT assets on users

It is useful to use for the CDaaS the IAM security model. As Gartner describes [2] IAM is security discipline that grants to the right person the right to access the right resources at the right time and for the right purpose.

IAM system is a combination of business processes, security policies and allows the organization to grant the access to confidential information in a secured way [4]. In order to have an IAM integrated system the following four elements:

- Means for the user to access the applications, systems and the documents from the organization for which his role is entitled to access.
- Ability to authenticate the user using the PLOP (Principle Of Least Privilege) principle through which the user access is set to a minimum that can permit him to do the daily job.
- SSO (Single Sign On) to easily facilitate the access to the resources the user is entitled to access.
- A way to audit the IAM system in order to validate if it is aligned to security requirements.

The following picture depicts the IAM model applied to the CDaaS.

Figure 3. Security model for a CDaaS

Acknowledgment

We thank ING continuous delivery team that helped with the infrastructure to test the proposed model of continuous delivery pipeline as a service.

References

[1] Dave Ohara - Moving toward continuous delivery as a service - 2013 http://research.gigaom.com/report/moving-toward-continuous-delivery-as-a-service/

[2] Gartner - Identity and Access Management - http://www.gartner.com/it-glossary/identity-and-access-management-iam/

[3] Jez Humble, David Farley - Continuous Delivery: Reliable software releases through build, test and deploy automation, Addison-Wesley, New York 2011

[4] Margaret Rouse - Identity Access Management - http://searchsecurity.techtarget.com/definition/identity-access-management-IAM-system