IJCSE International Journal of Computer Sciences and Engineering Open Access

Research Paper Vol.-7, Special Issue-15, May 2019 E-ISSN: 2347-2693

Devops: A Culture, Not A Technology Continuous Integration for Digital Enterprises

Syeda Misba[1*], Shilpa Biradar[2]

[1,2]Department of ISE, Dr. AIT, India

DOI: https://doi.org/10.26438/ijcse/v7si15.232234 | Available online at: www.ijcseonline.org

Abstract: Devops is a collaboration of development and operation devised to stress on communication and integration between them. It is supported by a culture collaboration it helps an organization to grow with these help organization can produce software products and services. Organization associated themselves with devops to startup new methodology. Continuous development and innovation are required in an organization so devops training has been started in the orientation. Companies are focusing on the automation of the process this way timely deliver and quality results are achieved We also found that DevOps is supported by a culture of collaboration, automation, measurement, information sharing and web service usage. DevOps benefits IS development and operations performance. It also has positive effects on web service development and quality assurance performance. Finally, our mapping study suggests that more research is needed to quantify these effects.

Keywords- Technology, Digital Enterprises

I. Introduction

DevOps is best defined as a culture and not a technology. Itisnotaskillset, noraspecifictool ora marketplace to innovate. Many of the underlying concepts and languageof the Dev Opsphilosophyare based on a combination of agile software development plus Kaizen, Lean manufacturing and Six Sigma methodologies. The relevancy of this methodology will be apparent even though you are technically sound

Adopting a Devops philosophy means fostering a highly productive culture of collaboration between the development (Dev) and operations (Ops) team.

Continuous Integration is not all about collecting few scripts and running them on a platform. In the scenario of developer writing an automation scripts, should add the same to version control repository to add value to the end product and make a working part of the build process It is best to implement CI early in the project. Although possible, it is more difficult to implement CI late in a project, as people will be under pressure and more likely to resist change. If you do implement CI later in a project, it is especially important to start small and add more as time permits. CI is not just a technical implementation; it is also an organizational and cultural implementation. People often resist change, and the best approach for an organization may be to add these automated mechanisms to the process piece by piece.

II. Related Work

In this section, we research some new technology as to overcome problem which facing in a company to do the project.where the developer develop the code and send for operations but the operation team takes several days to test code,build infrastructure and deploy So to overcome this devops was the solution

III. Methodology

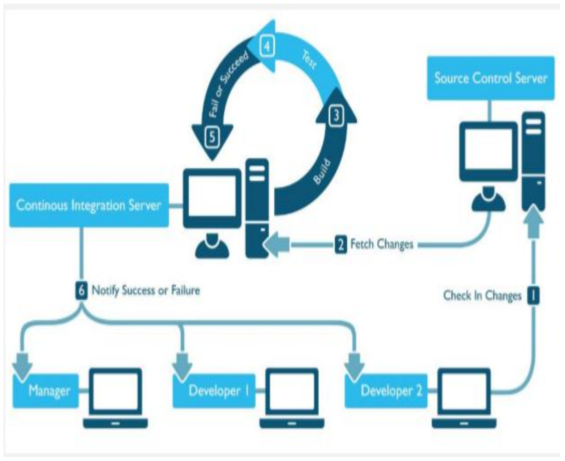
Practices of Continuous Integration

- Single Source Repository
  1. Separate requirements and dependencies
  2. Perform database versioning
  3. Commit as much as possible
  4. Associate a bug trackerto your SCM
- Automate the Build
  1. Command lineto generate a standalone binary
  2. Stable, flexible and portable across platforms
  3. Buildtools: Make files, Maven,Jenkins,Ant, shell script, Etc
- CI Server Setup
  1. Use adedicated server
  2. Define a job for every product and every environment (branches/clones)
  3. Keep deliverables only for production jobs.
  4. Only send alerts for failures
- Automate Deployment

1. Always deployto a dedicated server
2. All deliverablefrom CIservershould be deployed
3. Clones, snapshots andremote executionare essentials
4. Define snapshots

- Testing

1. Unit testing
2. User interface testing
3. API testing
4. Regression testing
5. Load testing
6. Security testing



CONTINUOUS TESTING & AUTOMATION

It is important to integrate quality into our software, making sure that we get a faster feedback on the impactofchanges.Ideallycontinuoustestingthisisinpracticetoeradicatetheerrorscausedbymanualinspectionofcodechangesandmanualtesting.However,thisstrategyhasseveraldrawbacks:

- Manual regression testing takes a long time and is relatively expensive to perform, creating a bottleneck that prevent susre leasing software more frequently
- Manual tests and inspections are not very reliable, since people are notoriously poor at performing repetitive tasks such as regression testing manually

In order to build quality in to software, we need to adopt a different approach. Our goal is to run many different types of tests—both manual and automated—continually throughout the delivery process.

CONTINUOUS DELIVERY AND DEPLOYMENT

Continuous delivery (CD) is a software development practice where code changes are automatically built, tested, and prepared for production release. It expands upon continuous integration by deploying all code changes to a testing environment, a production environment, or both after the build stage has been completed. When continuous delivery is properly implemented, developers always have a deployment-ready build artifact that has passed through a standardized test process.

With continuous deployment, revisions are deployed to a production environment automatically without explicit approval from a developer, making the entire software release process automated. This, in turn, allows for the product to be in front of its customers early on, and for feedback to start coming back to the development teams

IV. CHALLENGES

We have recorded some of the challenges we have faced in our CI journey with the appropriate solutions for successful delivery.

- GIT Repository Integration

Issue while creating a trusted link between GIT and Jenkins Server.

Solution Approach

Trust was established between GIT and Jenkins using the public and private keys of Jenkins server.

- Application Complexity

Most of the application in-herited complex architecture, denying the DevOps culture.

Solution Approach

Considered application architecture changes based on on-premises, cloud, and containers early in the process.

- RegularBuildFailures

Failures of jobsdue low running memory of the implemented node servers.

Solution Approach

Cronjobs were created to clean up the workspace on a regular interval of time and to release memory

- Improved customer Experience and satisfaction

Ultimately,the primarygoal ofDevOps is to deliverhigherqualitysoftwareto endusers at afaster pace, driving topline benefits around improved customer experience and increased revenue opportunity. The underlying goal is to become more agile and efficient in general, and this spans everything from driving greater productivity out of the IT workforce to subsequent benefits in operating expense, but at the end of the day it all goes back to deepening engagement with customers by creating increasingly useful applications in a more responsive manner.

- Breaking Down Silos
  TherearenumeroustacticalbenefitsthatDevOpsaffordsthebusiness,but taking the long view, I believe the most important strategic advantage is proving that self-organizational approaches can successfully break down the silos that result from hierarchical organizational models. The Agile Manifest out self-organization for small lteams, but extending it across organizational lines – and actually getting it to work – will be themo stimp or tantbottom-lineadvantage to DevOps overtime.
- DigitalTransformation
  Everyenterprise, ineveryindustryishavingtodigitallytransformthewa

ytheyoperate.Thismeansusing innovations in technology (e.g. mobile, IoT, connected cars etc.) to deliver new digital services that enhance customer experience and improve employee productivity. At the centre of these digital services is software. DevOps is essential to being able to deliver digital services at speed and with quality,and so the bottom-line advantage of DevOps is that it's a foundational element of successful digital transformation.

TOOLS ADOPTION

Continuous integration is all about performing operations with right set of tools. The selection of tools is generally driven by the various parameters like Vision, IT policies, existing technology, IT landscape, vendor partnership and other considerations. It is therefore advised that every organization is must do proper due diligence before adopting the tools for running this operation.

Some of the major tools are represented in the diagram, showing the tools andits capabilities. These are successfully implemented across all the organization, who are reaping benefits from this. Celstream majorly uses Jenkins for serving most of the customers with Bamboo, Chef and puppet being other popular in the company.

To achieve continuous integration and testing, below are the tools we can use.

Jenkins

Jenkins, originally called Hudson, is an open source continuous integration tool written in Java. Jenkins is used by teams of all sizes, for projects in a wide variety of languages and technologies, including .NET, Ruby, Groovy, Grails, PHP and more, as well as Java. Jenkins is easy to use. The user interface is simple, intuitive, and visually appealing, andJenkins's whole has a very low learning curve.

PMD

PMD is open source and a static Java source code analyser. It finds un used variables, empty catch blocks, and unnecessary object creation. This plug in issued to generate the automated code reviews.

Check style

Check style is a static code analysis plug n used for checking if Java source code complies with coding standard.

Javadoc

Javadocisatoolthatparsesthedeclarationsanddocumentationcommentsinasetofsourcefilesand produces a set of HTML pages describing the classes, interfaces, constructors, methods and fields. This plug in issued to generate the automated API documentation.

V. RESULT

The result of doing this is that there is a stable piece of software that works properly and contains few bugs. Everybody develops off that shared stable base and never gets so far away from that base that it takes very long to integrate back with it. Less time is spent trying to find bugs because they show up quickly

VI. CONCLUSION

Continuous integration has always been the fulcrum of successful DevOps implementation. This would requirearightstrategyandneedtobeabletoseebenefitsuponthedeployment.Sincemostoftheaspects revolves around the tools and the integration, making appropriate choices is very much vital. Organization should start investing more the resource training of tools and start smoothly shifting the focus on adoptingtheDevOpsculture,thusavoidingthepitfallsandotherbarriers.

As with most of the technological breakthrough today, implementation of DevOps would also see few unknown challenges and this would slow down the process of delivery. The team should be well equipped tosolvesuchissuesandseethistoitsgoal.

REFRENCES

[1]. https://d1.awsstatic.com/whitepapers/DevOps/practicing-continuous-integration-continuous-delivery-on-AWS.pdf

[2]. http://possible.mindtree.com/rs/574-LHH-431/images/devops-are-we-there-yet.pdf

[3]. https://d1.awsstatic.com/whitepapers/jenkins-on-aws.pdf

[4]. https://hexaware.com/casestudies/at-ad-wp-01.pdf