IJCSE International Journal of Computer Sciences and Engineering Open Access

Survey Paper Vol.8, Issue.6, June 2020 E-ISSN: 2347-2693

DevOps: Concept, Technology and Tools

Pallavi Deshwal[1*], Poonam Ghuli[2]

[1]Dept. of Computer Science, RV College of Engineering, Bangalore, India
[2]Dept. of Computer Science and Engineering, R V College of Engineering, Bengaluru, India

[*]*Corresponding Author: pallavideshwal7311@gmail.com, Tel.: +91-7060865539*

DOI: https://doi.org/10.26438/ijcse/v8i6.7378 | Available online at: www.ijcseonline.org

Received: 22/May/2020, Accepted: 20/June/2020, Published: 30/June/2020

***Abstract*—** DevOps is a new concept that consolidates development and operations team to intently incorporate individuals, procedures and innovation for automated end to end delivery and deployment of software. DevOps Engineers have start to finish obligation of the Application (Software) directly from gathering the prerequisite to improvement, to development, to testing, to application deployment lastly checking and assembling input from the end clients, again implement the changes as per end client requirements. This paper presents DevOps concept, how it has been evolved from traditional methods, technologies involved such as CI/CD pipeline, project gating and DevOps tools that automates the software development cycle.

***Keywords*—** DevOps, CI/CD, Jenkins, Git, Docker, ZUUL, Artifactory, CI/CD Visualization Dashboard, JIRA, Valgrind

I. INTRODUCTION

The Time to Market of an item basically influences its achievement particularly when discussing innovations, which must be conveyed while they are still new. In such condition, at that point, what truly separates the item on the market isn't just the item itself, or its quality, yet in addition the speed at which it can develop: for an organization, taking the item to showcase quick intends to prevail upon the contenders and being constantly lined up with new inclinations [15].

As increasing popularity of agile methodology, DevOps comes as an extension to agile where development and deployment process is iterative. To reduce cost and time to deploy a software, DevOps leads to iterative and repetitive deployment. In this concept developers can deploy the code they develop at faster pace and in shorter cycles. As this software deployment process evolved from the traditional methods to agile and then DevOps, use of these methods have significantly increased over time. This research work explains deeply CI/CD pipeline for continuous integration and delivery, Git as source code management system, Jenkins as an extensible automation server or CI server that can lead to continuous delivery hub for any development work, Zuul as project gating system, Docker as virtualization technology, Artifactory for Artifacts repository. CI/CD pipeline gives a wider picture of why in DevOps everything is continuous: integration, testing, deployment and delivery. CI/CD pipeline design is displayed in Figure 1.

Rest of the paper is organized as follows, Section I contains the introduction of DevOps concept deeply with Section II contain background work, Section III contains how evolution of software development procedure from waterfall model to agile then later to DevOps happened, Section IV contains technologies related to DevOps culture, Section V contains tools to automate the process of development in DevOps and Section VI concludes the research work with future scope.

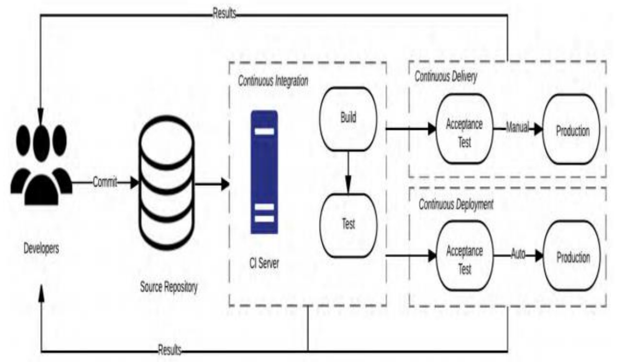

Figure 1. CI/CD Pipeline in DevOps [24]

II. BACKGROUND WORK

DevOps practices enables sharing of goals, incentive, processes and tools [1]. DevOps integrates developers and operations team to combine and collaborate individuals, processes and technology for an automated delivery of software which follows agile methodology, scalable and cost-effective [2]. DevOps practices includes Continuous Delivery and Deployment (CD), testing that is automated, and code for infrastructure [3; 4]. DevOps reduces time when developer commits a change and deploys that change to without compromising the quality of software [4]. There is high correlation among Continuous delivery and deployment practices, and tight binding in which the correct information for defining these practices is not available [5; 6; 7]. Sometimes there are issues to make difference between these practices and depends on how any given organization integrates them [8; 9]. Continuous deployment enables continuous delivery to gradually deploy applications to production environment if checks and automated tests are passed. In continuous delivery practice, the management team takes decision about when to deliver changes to customers, all steps or decisions in continuous deployment are automated no manual steps are used; once developers commits a change in code, the change is deployed to production using a continuous deployment pipeline [11]. How to apply these two practices depends on source. Sometimes every type of systems and organizations can practice continuous delivery, continuous deployment but it may not suite to every types of organizations [12; 13].

III. EVOLUTION

From Waterfall Methodology to Agile Methodology and then to DevOps is a complete evolution towards software development lifecycle and also to project management. Waterfall model is oldest model and defines various steps to develop a complete software. However conditions for waterfall model are to complete one step of development cycle before moving to next step. Figure 2 is waterfall model architechture.

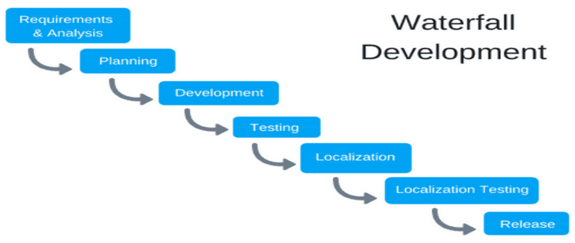

Figure 2. Waterfall Model [25]

Whilst due to speed and related issues with waterfall models, Agile and DevOps were introduced to provide flexibility. Agile methods enable product releases to be delivered faster. Agile methodology is based on continuous iterative sprints in Figure 3. Also, agile helps in faster delivery of features with immediate feedback from customer.

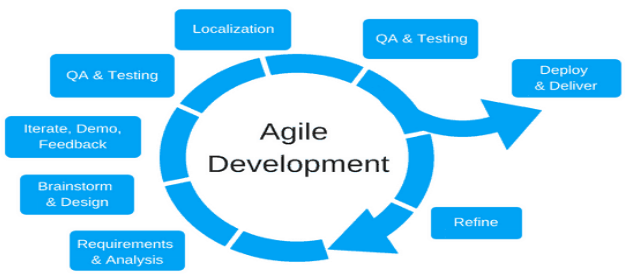

Figure 3. Agile Methodology [25]

DevOps optimizes the development process by integrating development and deployment process together. DevOps introduces agile methodology also, synchronizes with fast iterative agile process of development with Ops process testing and deploying to prevent from backlogs to happen Figure 4.

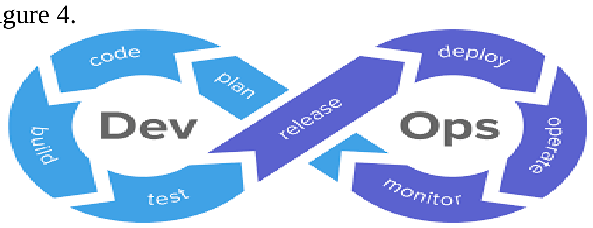

Figure 4. DevOps Development Model [25]

DevOps is a methodology wherein conventional software development steps are consolidated and upgraded to improve the frequent process of production release and keep up quality of software.

IV. TECHNOLOGIES INVOLVED

There are various technologies involved in DevOps practices. Major technology is Continuous Integration/ Continuous Delivery (CI/CD) pipeline that integrates different development stages. And project gating that involves testing pipeline that ensures correct code changes to be merged.

A. CI/CD Pipeline: CI/CD Pipeline enables the automation of software delivery process such as initializing code builds, automated testing, deployment to staging or production environment. Automation pipeline helps in removing errors that happens with manual process, standardized feedback loops and enables fast production iterations. CI stands for continuous integration and is a software development process where all developers merges their code changes in a central repository such as gerrit, many times in a day. CI enables each code change to trigger automated build and testing for the given software. It provides feedback to developer who made the code changes. CD stands for continuos delivery and adds automation of complete software release process with continuous integration.This process is completely automated with logs which is visible to entire team members.

Elements of CI/CD Pipeline:
1. **Source Stage**: In this stage developer pushes code changes to source code repository that triggers a run in CI/CD pipeline. Some common triggers can be scheduled automatically or can be initiated by user as well as by results of previous pipelines.
2. **Build Stage**: Build stage includes code compiling based on languages that needs to be compiled. Some coding languages do not require this step. Other cloud native software deployment happens with Docker where this step CI/CD pipeline (triggers) builds Docker containers. If this build step fails that means, there is problem with configuration of the project.

3. **Testing Stage**: In testing phase the code should pass certain tests to validate if the code and behaviour of project is correct. Developers writes these tests to provide checks such as unit tests, integration tests, sanity checks also entire system tests. Failure to pass this stage reflects that developer should check for problems which were not addressed while writing the code.
4. **Deployment Stage**: As soon as the instance of code is runnable, and it has passed all required tests the deployment is ready. There are certain deployment environments such as staging and production. Deployment in "staging" is done for production team for internal use and, deployment in "production" environment is for customer.

B. Project Gating: The elements of CI/CD Pipeline are pillars of DevOps practices. If any mistake in these steps that can lead to problems in development process. Project Gating creates some testing approaches that product should pass before it goes for deployment. These approaches involves unit testing, integration testing, functional testing, acceptance testing. Project Gating provides quality assurance and readiness of software by placing these tests to be passed by product to move ahead.

V. TOOLS

1. **Jenkins:** Jenkins is a tool that automates the software delivery process and accelerate it. Jenkins is called automation server for continuous integration and continuous delivery environment. In Jenkins any language for source code can be used and offers simple way for setting up source code repositories as well as other development tasks. Jenkins started from being a platform for continuous integration then became a continuous delivery platform. Jenkins automates the process of building and testing. Jenkins as a CI tool automates complete development process and reduces workload and time for developer. Jenkins works as orchestrator to perform certain development steps and to automate the process. Architecture of Jenkins is distributed. Jenkins is a web dashboard that where projects or jobs can be configured and builds happens on nodes. Nodes are virtual machines configured to run Jenkins jobs. Jenkins follows master/slave architecture shown in Figure 5. Wherein Jenkins server is called master. Nodes are called slaves. Jenkins server also called master schedules the builds for jobs and dispatches builds to slaves where actual execution of builds happens. Master node also monitors of slave nodes, records and presents build results.

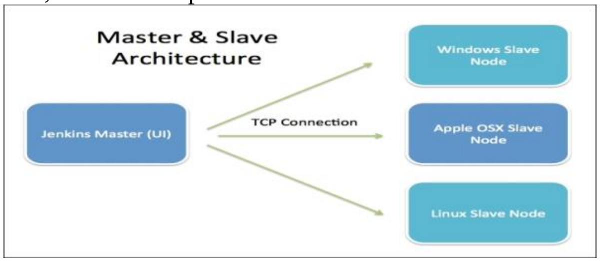

Figure 5. Master/Slave Architecture of Jenkins [26]

Jenkins also supports tools for configuration management such as Ansible, chef. Jenkins provides a variety of plugins that supports various features for project configuration. These plugins enable various tasks such as scripts deployment, virtual machine launching and running Docker containers virtual environment.

2. **Git:** A version control system provides a single repository to store all files at one place. If there is a code change it is needed that all codes are checked out that are stored in repository. Whenever these changes in code happen, changes are added to new version. Every time new versions of files are stored when edits in code are done. This enables to have a central repository of code from where people can use this code. This also creates challenges when centralized repository is shared with all team members and code changes happen simultaneously. Whilst distributed version control system provides facility to share code across the team of developers. All the developers maintain a local copy of code and all the time keeps that code updated. Distributed version control system facilitates that all the developers and clients share the updated version of software. If there is any change in code it is shared with all development team. Here comes Git as a distributed version control system. Git is open source and provides distributed environment for version controlling. Git tracks source code changes. Git tool allows developers to work simultaneously. GitHub as a remote server facilitates source code management and development team connect clients remotely to this server. GitHub have some remarkable features such as commit history, code review with pull requests, notifies via email displayed in Figure 6. Git and GitHub can be integrated to provide access to repository via internet. Git is a software tool that can be installed on local systems, GitHub is a web service. Git manages different versions of code. GitHub facilitates shared repository on local systems. Git provides better performance, more secure and flexible way of version controlling.

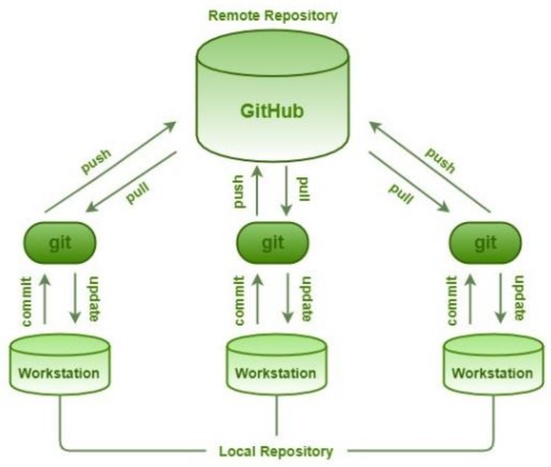

Figure 6. Git Working Process [27]

3. **Docker:** Today application development takes much more time than writing code for application. Every

stage of lifecycle requires multiple languages, architectures, framework, deployment environment and tools interfaces. Docker creates Linux containers. Docker is a technology for virtualization and uses Docker engine. Docker speeds up the workflow and facilitates developers with their own tools, deployment environments and application stack. Docker as a tool is designed for creation, deployment and running applications using containers. Containers are isolated virtual environment that provides application as a complete packages, libraries and dependencies to deploy. Docker benefits both development and operations team as a part of DevOps toolchain. Containers enables portability for applications. It is done by containers by isolating code in one container, this also makes updating code easy. Containers share Operating System. Unlike Virtual Machines, containers virtualize operating systems not hardware as in Virtual Machines. Using containers code of application and dependencies can be packaged together to provide deployment environment for applications. On single shared OS kernel multiple containers can be running in user space as isolated processes. Containers enable more application to run than in VMs and uses less space and need fewer Virtual Machines and Operating Systems. Docker containers uses less space, provides high speed to start-up and easier integration.

4. **CI/CD Pipeline Visualization Dashboard:** In IT industry, development and testing teams normally have a lot of unique information dispersed and lying in various frameworks. Because of an absence of visibility into the CI/CD work process, it prompts poor discharge quality, deferred cut-off times because of a more drawn out time overlay from improvement to creation. What's more, there's an absence of value checking of construct or organization. Visualizing dashboard controlled Continuous Integration/Continuous Deployment (CI/CD) enhances the input circle with configurable dashboards that help measure the exhibition, empowering speedy choices. CI/CD totals information and utilizations representation strategies to help battle the absence of a folded-up progress of a project. With instinctive and configurable dashboards CI/CD helps project workflow and the presentation measurements or performance metrics at each stage. CI/CD Dashboards gives a consolidated view of various components of a project. It provides a simple way to view all builds, tests, releases on a single page. This dashboard can be used by both development and operations team to have a view of project builds and related data at one place. It also helps developers to have a view if code has broken. Some dashboards provide facility to have performance metrices based on all reports generated during development and testing phases. These dashboards present measurements on how your DevOps team functions and can help distinguish present or potential issues in group coordinated effort, application conveyance, and stage health status. They additionally empower DevOps teams to upgrade their capacities in zones, for example, quick application conveyance, made sure about runtimes, and automated CI/CD.

5. **Zuul:** Project gating process prevents changes from being merged if the changes introduce regressions. With this, working mainline can be open for all developers and after the confirmation of a change that it will work without any disruption, that change can be merged. Some informal methods of gating are being practice for projects wherein developers make sure to run a test suite when commit happens in working mainline before this commit merges. At a larger scale developer are more, changes are more, and test cases are more these informal gating practices does not scale well and becomes complex and time consuming to handle. Zuul is a framework for project gating that automates this process. Zuul can help to test correctly the larger number of commits. It is a CI testing tool that was developed in 2012. Essentially it was designed to solve problems of developers of slower testing when testing is serialized. Zuul came into picture as testing tool then later it became automatic building, merging and testing tool. Zuul provides automation process for merging, building and testing any new change done to a project. When code reviewing event happens it facilitates automated tasks and running tests in response. Zuul also introduced concept of co-gating, creating testing environment for multiple shared repositories to make sure test fails if any change breaks its project as well as test should fail even if that change breaks any other projects in that shared repositories. Zuul provides facility for multiple users to make changes at same time enabling parallelism features. Zuul knows where the tests should run and in what order. Zuul ensures changes done parallelly should be tested in correct order. Zuul have a dependent pipeline manager to handle testing of changes done in parallel. If any of the change done in parallel fails testing that should be re- tested without failed change. If all succeeds all can be merged once. Best case is when as many possible changes are tested, passed and merged at once. Worst case is when one change is tested at once. Zuul follows concept of pipeline. For example, check, gate, post are some pipelines of Zuul. And pipelines have triggers causing projects to be enqueued in any pipeline. Zuul as a CI project gating tool have various features and works as a tool in DevOps toolchain.

6. **Artifactory:** A product by JFrog is a binary repository manager. A binary repository is where the Artifacts, which are created by build results, are stored. It is a good practice to use a package manager for binary repository. Artifactory is a manger for binary repository. Different application components are stored, and these components can be assembled together to create full software product. This facilitates that any build can be broken into small chunks, build times can be reduced, and using resources efficiently. Artifactory can host both public and private repository. Artifactory as a public repository manager uses proxy. With this proxy the time to retrieve Artifacts can be reduced and bandwidth can be conserved.

DevOps team uses Artifactory as a tool to manage binaries, Artifacts, environment and distributed sites for project workflow.

7. **Valgrind:** For QA (Quality Assurance) perspective in DevOps Valgrind is used as a tool for memory leak testing. Memory leak is when available memory is lost by program that fails to return temporarily used memory. Valgrind is a memory debugging tool and works as a dynamic code analysis tool for DevOps. Valgrind have different modules for analysis, main module is Memcheck. Memcheck is used for memory leak detection. It can detect different type of memory related errors.

8. **JIRA:** For project management and issue tracking DevOps toolchain have various tools one of them is called JIRA. JIRA is a tool developed for tracking bugs and issues. JIRA is a tool for project management. For software and applications JIRA tracks all related bugs and issues. Dashboard for JIRA have different features that enables it to handle issues easily. JIRA tool for coding, team collaboration and stages for release, works as a central hub. JIRA across teams provides visibility and traceability for information related to software. JIRA have various admin related features that can be provides to users. Audit logs, all information related to issues such as creating issues and updating, is stored under audit logs. Mailing system in JIRA can provide mail issues to be sent via external mail service. JIRA have workflow for issues that provides issue all stages and transitions issues go through such as open, resolved, InProgress, Reopened and close. For DevOps practices, all teams must use same issue tracking tool. It provides more visible and responsive workflow. JIRA tool also known as task management tool, as JIRA issues are also called tasks. User stories are also added in JIRA which are description about features developed. JIRA benefits both software development and project management teams.

VI. CONCLUSION AND FUTURE SCOPE

This paper explores all aspects of DevOps practices, how DevOps integrates the development and operations team. In this paper, we discussed how DevOps automates complete software delivery process with the help of its tool chain and that DevOps tool chain consists set of tools which are used during complete DevOps cycle such as continuous integration tool, monitoring tool, binary repository manager, source code version control system, dynamic analysis tool, issue tracking and project management tool. This paper also presents that DevOps facilitates quality assurance and team collaboration to provide faster and cost-effective solution for software development lifecycle. The major concern with DevOps is to introduce and structure organization to adopt DevOps. To embrace DevOps in organization with proper structuring of organization and handling of DevOps role are some areas for future research.

VII. ACKNOWLEDGMENT

We would like to express our deep gratitude towards our Guide Dr. Poonam for her guidance, enthusiastic encouragement, valuable and constructive suggestions during planning and development of this paper.

REFERENCES

[1] M. Httermann, “DevOps for developers”, Apress Publisher, 2012.
[2] Jay Shah, Dushyant Dubaria and Prof. John Widhalm, “A Survey of DevOps tools for Networking”, IEEE, 2018.
[3] “2015 State of DevOps Report”, Available at: https://puppetlabs.com/2015- devops-report.
[4] BASS, L., WEBER, I., and ZHU, L., “DevOps: A Software Architect's Perspective”, Addison-Wesley Professional Publisher, 2015.
[5] FITZGERALD, B. and STOL, K.-J., “Continuous Software Engineering: A Roadmap and Agenda”, the Journal of Systems and Software, 2017.
[6] Chellamalla Mamatha, S C V S L S Ravi Kiran, “Implementation of DevOps Architecture in the project development and deployment with help of tools”, ISROSET, Vol.6, Issue.2, pp.87-95, 2018.
[7] HUMBLE, J., “Continuous Delivery vs Continuous Deployment”, Available at: https://continuousdelivery.com/2010/08/continuous-delivery-vs-continuousdeployment/ [Last accessed: 1 March 2016].
[8] LUKE, E. and PRINCE, S., 2016. No One Agrees How to Define CI or CD. Available at: https://blog.snap-ci.com/blog/2016/07/26/continuous-deliveryintegration-devops-research/ [Last accessed: 1 August 2016]
[9] THIELE, A., 2014. Continuous Delivery: An Easy Must-Have for Agile Development, Available at: https://blog.inf.ed.ac.uk/sapm/2014/02/04/continuous-delivery-an-easy-musthave-for-agile-development/ [Last accessed: 10 July 2016].
[10] WEBER, I., NEPAL, S., and ZHU, L., “Developing Dependable and Secure Cloud Applications”, IEEE Internet Computing 20, 3, 74-79, 2016.
[11] DINGSØYR, T. and LASSENIUS, C., “Emerging themes in agile software development: Introduction to the special section on continuous value delivery”. Information and Software Technology 77, 56-60,2016.
[12] MOONEY, M., “Continuous Deployment For Practical People”, https://www.airpair.com/continuous-deployment/posts/continuousdeployment-for-practical-people.
[13] REED, J.P., “The business case for continuous delivery”, Available at https://www.atlassian.com/continuous-delivery/business-case-for-continuousdelivery, [Last accessed: 12 July 2016].
[14] FORD, N., “Continuous Delivery for Architects”, Available at: http://nealford.com/downloads/Continuous_Delivery_for_Architects_Neal_Fo rd.pdf [Last accessed: 20 October 2016].
[15] Valentina Armenise, “Continuous Delivery with Jenkins”, IEEE/ACM 3rd International Workshop on Release Engineering, 2015.
[16] Nikita Seth,Rishi Khare, “ACI (Automated Continuous Integration) using Jenkins: Key for Successful Embedded Software Development“, Proceeding of the RAECS UIET Panjab University ,Chandigarh, 21-22nd December 2015.
[17] Mojtaba Shahin, M. Ali Babar, Liming Zhu, “Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices”, IEEE, Received February 16, 2017, accepted March 16, 2017, date of

publication March 22, 2017, date of current version April 24, 2017.

[18] Pulasthi Perera, Roshali Silva, Indika Perera, “Improve Software Quality through Practicing DevOps”, International Conference on Advances in ICT for Emerging Regions (ICTer): 013 – 018, 2017.

[19] Jay Shah, Dushyant Dubaria, Prof. John Widhalm, “A Survey of DevOps tools for Networking”, IEEE, 2018.

[20] Hessa Alfraihi and Kevin Lano,“The Integration of Agile Development and Model Driven Development - A Systematic Literature Review”, In Proceedings of the 5th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2017). SCITEPRESS, 451–458, 2017.

[21] Gerald Schermann, Jürgen Cito, Philipp Leitner, Uwe Zduny, Harald C. Gall, “An Empirical Study on Principles and Practices of Continuous Delivery and Deployment”, PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.1889v1 | CC-BY 4.0 Open Access | rec: 22 Mar 2016, publ: 22 Mar 2016.

[22] Tony Savor, Mitchell Douglas,Michael Gentili, “Continuous Deployment at Facebook and OANDA”, IEEE/ACM 38th IEEE International Conference on Software Engineering Companion, 2016.

[24] Y. Sundman., “Continuous Delivery vs Continuous Deployment”, 2013 [Online] Available: http://blog. crisp.se/2013/02/05/yassalsundman/continuous-delivery-vs-continuousdeployment

[25] Alexander Eck, Falk Uebernickel, and Walter Brenner, “Fit For Continuous Integration: How Organizations Assimilate An Agile Practice,” 2014.

[26] Amit Deshpande and Dirk Riehle, “Continuous Integration in Open Source Software Development,” 2008.

[27] Daniel Ståhl and Jan Bosch, “Experienced Benefits of Continuous Integration in Industry Software Product Development: A Case Study,” 2015.

Authors Profile

Ms. Pallavi Deshwal pursed Bachelor of Technology from Uttar Pradesh Technical University, Lucknow, India. She is currently pursuing Master of Technology in Computesr Science and Engineering from RV College of Engineering, Bangalore, India since 2018. Her area of interest includes DevOps, Data Science, Machine Learning and Big Data Analytics.

Dr. Poonam Ghuli has been working as Associate Professor in the Department of Computer Science and Engineering over the past fifteen years at R V College of Engineering, Bengaluru, India. She has a couple of papers published in reputed Journals and conferences. She is working in the area of Data Analytics. She is actively involved in many researches and consultancy work supported by renowned companies such Citrix, Cisco, Samsung etc.