[ICRTCET-2018]

ISSN 2348 – 8034
Impact Factor- 5.070

GLOBAL JOURNAL OF ENGINEERING SCIENCE AND RESEARCHES

A STUDY ON DEVOPS WORKFLOW AUTOMATION

Sivakumar D[*1], Sujay M[2], Syed Nabeeq[3] & Varshitha R[4]
[*1,2,3&4]Department Of Computer Science, Rajarajeswari College Of Engineering, Bangalore, India

ABSTRACT

Companies have benefited by adopting agile development methodologies and as part of such methodology have seen benefits by automating and adopting continuous integration principles in their software lifecycle. Such practices have improved the efficiency of development teams. With the efflux of time, companies have realized that the Continuous integration alone will not help to make the delivery lifecycle efficient and therefore will not drive organization efficiency. Software development life cycle involves several steps namely - the development cycle (requirements to design to coding and unit testing), testing cycle (deployment, system integration testing and user testing) and use cycle (deployment, monitoring and application availability). Continuous Integration did not apply automation testing cycle and use cycle. DevOps automates development cycle, testing cycle and use cycle. DevOps is about automating the entire software delivery life cycle. DevOps is a software engineering philosophy and practice that aims at merging software development (Dev) and software operation (Ops). Automation is brought about by implementing various tools at different stages of software development process. Companies that rehearse DevOps have reported important profits like lesser time to market, enhanced customer satisfaction, amplified product quality and increased ability to build the correct product by rapid experimentation. In this paper, we aim to understand all aspects of DevOps automation steps of Software Development Lifecycle. Further we will review various tool and technology combinations in aspect of DevOps.

Keywords: *Agile, Software development lifecycle, Continuous Integration, Automation.*

I. INTRODUCTION

Many software companies are involved in production of application software. When the companies are responsible for large distributed and complex applications, the operational complexity grows at a quick rate. While developing an application many questions arise regarding how we deploy the applications, how we monitor services, how we monitor the performance of the application, how alerts and remedies are given when there is a problem [1].

DevOps concentrates on combining the power of developers and operations. The focus on this collaboration enables a new way to manage the complexity of these applications. [2] DevOps is the practice of operations and development engineers participating together in the entire service lifecycle of an application, starting from design to development to production support.

DevOps has become a trending technology in the modernized software industry. Before DevOps, Agile methodology and principles had created a similar hype in bringing a change to the methods of software development. This had led the software industries to spend huge money in giving a new form to the organization's structure, implementing and deploying new solutions, and hiring employees who self-proclaimed to be experts.[4] Few of the main objectives of agile principles like delivering small and iterative working solutions seemed to be forgotten. The methodologies were implemented only during the phases of development cycle which led to the creation of backlogs in the operation team, because these teams failed to release the products fast enough by being unable to cope up with the pace of development teams.

Limitations with agile methodologies emerged when development teams relied on false assumptions such as coherent system design with iteration, paying technical debts at later stages, assuming constant refactoring to be the best practice, systems are flexible and adaptable to incorporate changes made by agile principles. [5]Such assumptions failed to improve the productivity and ended up the development team to practice microcosm planning, development and testing activities as short phases or sequences. The ops team was left behind with fast piling up of deployments to be released and the customers were left unsatisfied and did not receive what they demanded.

This led to the evolution of DevOps, a new and improvised version of agile methodology including the Dev and Ops teams to enable to service with quality and speed. Automation is brought about by implementing various tools at different stages of software development process. Companies that rehearse DevOps have reported important profits like lesser time to market, enhanced customer satisfaction, amplified product quality and increased ability to build the correct product by rapid experimentation.

II. LITERATURE REVIEW

The paper portrays the communication practices from a distributed agile team which comprises of the two teams that is the development and operational team based on communication challenges which may include geographical, socio-cultural, and temporal distance and many other strategies. A large multinational organization is taken as an example and a study is conducted on how two way communication takes place with respect to distributed DevOps. Since DevOps is all about joint effort to better serve the client, it is vital that the two teams are very well aligned and adjusted on their communication practices. [6]

The paper [7] portrays a net based system (*DevOpsEnvy*) which is acquainted to support the administration and observing of student teams involved in DevOps. By coordinating a few popular open source tools, this framework gives students with facilities such as group management, project status observing and student performance data enquiry and so on.

The paper examines the fit of DevOps for regulated medical device software development. The connection of the DevOps approach and regulated software development of medical devices is studied. The procedure, strategy and the environment of software creation and programming must be recorded and documented. This is possible by the DevOps tools, for example, GIT and Jenkins [8]

The target of the exploration is to examine whether quality of the software or the product gets enhanced when practicing DevOps in programming and software companies [9]. This examination has shown the elements that are to be considered to improve the quality of the software and how organisations can rehearse DevOps. Information has been accumulated through conducting web surveys and meets with DevOps specialists in software development industry.

The paper acquaints the DevOps approach with security affirmation in multi-cloud applications which empowers early feedback of the application security status to the development phase keeping in mind the end goal to take restorative activities as quickly as time permits at whatever point they are required. The assessment of DevOps and dynamic security assurance demonstrated that the proposed strategies and tools lessen the security defects in the application usage and guarantee the multi-cloud application consistence to information protection necessities (including information integrity and privacy).

III. DEVOPS WORKFLOW

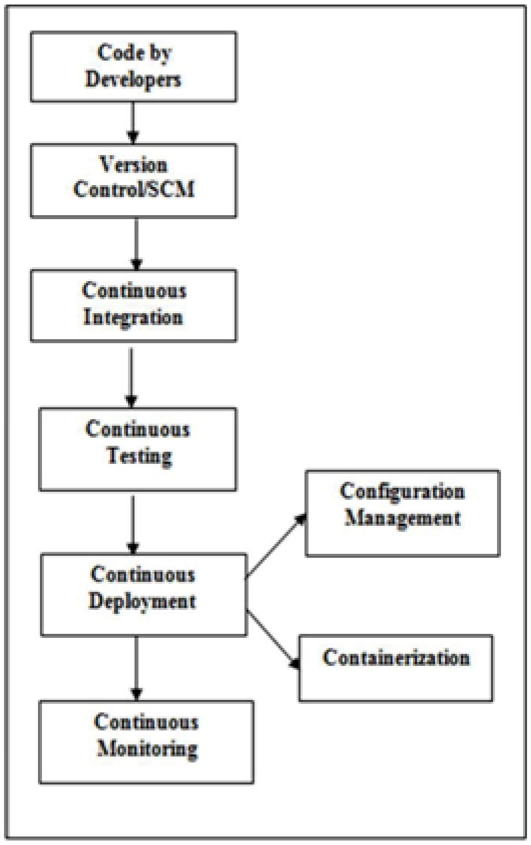


Figure 1: Block diagram of DevOps workflow

Phases of DevOps:
Figure 1 shows the different phases of DevOps. The various phases includes :

- Continuous Development
- Continuous Integration
- Continuous Testing
- Continuous Deployment
- Continuous Monitoring

Continuous Development
This phase involves planning and coding of application. The vision of the project is decided into the planning phase, and when they start writing the code, it's referred to as coding phase. Developers write the code in any language and once they commit the code, it goes to source code repository where we can manage the codes written by developers. This is nothing but a version control, this process is automated using tools such as GIT, SVN and JIRA.

Versions are maintained in a central repository so that all the developers can collaborate on the 'latest committed' code, and even operations can have access to that same code when they plan to make a release.

Whenever a mishap happens during a release, or even if there are lots of bugs in the code (faulty feature), there is nothing to worry. Ops can quickly rollback the deployed code and thus revert back to the previous stable state.

Continuous Integration

Continuous Integration (CI) plays a major role during the first release. It helps massively to integrate the CI tools with configuration management tools for deployment. The most popular CI tool in the market is Jenkins, other popular tools are bamboo and Hudson.

This tool (Jenkins) helps in the automation of tools falling under other DevOps lifecycle phases. It supports large number of plugins which can be used to integrate with other Continuous Development tools or Continuous Testing tools or Continuous Deployment tools or even Continuous Monitoring tools.

When integrated with Git/ SVN, Jenkins schedule jobs by pulling the code from shared repositories automatically and make it ready for builds and testing (Continuous Development). Jenkins can build jobs either at scheduled times of day or whenever a commit is pushed to the central repository. Maven is used for building the code in Jenkins.

POM is a representation for Project Object Model. It is central unit of work in Maven. It is an XML file that exists in the base directory of the project in the name pom.xml. The POM comprises of data related to the project and various formation detail used by Maven to form the project.

POM also encompasses the objectives and plugins. While accomplishing a task or goal, Maven searches for the POM in the present directory. It goes through the POM, obtainss the required configuration information and data, and then implements the goal.

Continuous Testing

When integrated with testing tools like Selenium, we can achieve Continuous Testing. The developed code can be built using tools like Maven/ Ant/ Gradle. When the code is built, then Selenium can automate the execution of that code by creating a suite of test cases and executing the test cases one after the other.

The role of Jenkins/ Hudson/ Bamboo here would be to schedule/ automate. Selenium to automate test case execution.

Continuous Deployment

This phase has two parts- Configuration Management and Containerization

(a) Configuration management

It is the act of releasing deployments, scheduling updates keeping the configurations consistent across all the servers. Configuration management can be achieved with the help of Infrastructure as code (IAC).

With this approach, infrastructure can be shared, tested and version controlled. In IAC we write the code for infrastructure in one central location and it can be shared with any number of nodes. If we want to update software then we write the code in one location and same will be replicated to 100’s of nodes. Advantage of this is, since everything is well documented in central location rolling back to previous version is not time consuming, configuring large infrastructure has been easy.

Most Popular Configuration Management tools are:
Puppet, Chef, Ansible, and SaltStack.

(b) Containerization

It comprises of the whole run time environment, an application, all its reliance, libraries and other binaries, and configuration files needed to run it, bundled into a single package. If we need to run an application for shorter duration in production environment (prod), we can create a container in prod and run the application in that container.

Most popular containerization tool is Docker.

Continuous Monitoring

It ensures that application is performing as desired and the environment is stable. It quickly determines when service is unavailable and understands the underlying causes. Tools of this phase continuously record the system's health and alert administrators when they find out problems so that the administrators can take necessary corrective actions.

Nagios is a well-known open source tool for monitoring IT infrastructures such as end-user stations, IT services, and active network components. Other popular monitoring tools are: Splunk, ELK Stack, Sensu and NewRelic.

IV. RESULTS AND DISCUSSION

We will be depicting the results of Continuous Integration which is brought about by using the platform of the DevOps tool called Jenkins. It is collaborated with other tools like Git and Maven. Git contains the source code of the application. Maven is used to build the code of the application.

We have considered a web application called 'Game of Life' to show the Continuous Integration phase of DevOps.

An account needs to be created in Jenkins to start using it. Later using the account details we can log in and start the kind of project we wish to build as shown in figure 2.

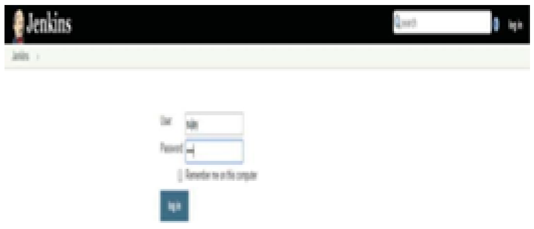
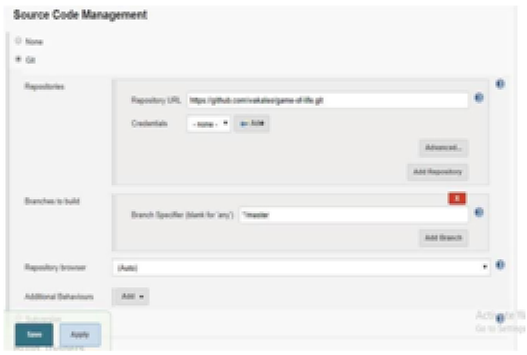


Figure 2: Logging in to Jenkins

The specification of the path of source code in GitHub while configuring the build of the desired project is shown in figure 3.


Source Code Management

None

Git

Repositories

Repository URL https://github.com/wakaleo/game-of-life.git

Credentials - none -

Branches to build

Branch Specifier (blank for 'any') */master

Repository browser (Auto)

Additional Behaviours

Figure 3: Specification of source code path

We have to specify the build schedule that is, for what duration the source code has to be checked for any new modifications. The path of the application's code must be specified. Whenever there is modification in the source code in GitHub, the code is automatically checked out to Jenkins and build process will begin.

The status of the build can be determined at the end of the build. If the project is built successfully it shows "Build Success" with a color depiction of blue color else shows "Build Failure" with a color depiction of red color. A snapshot of the console output in which a build of the code of a web application is initiated is shown in figure 4.

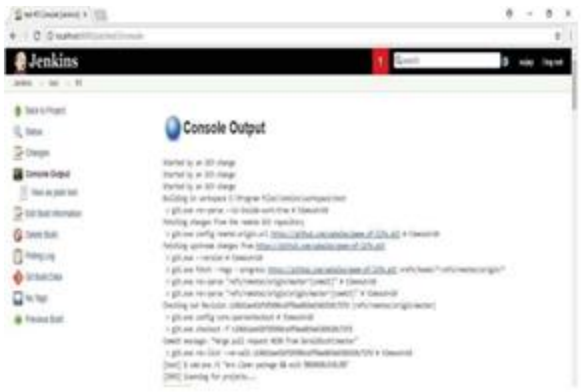
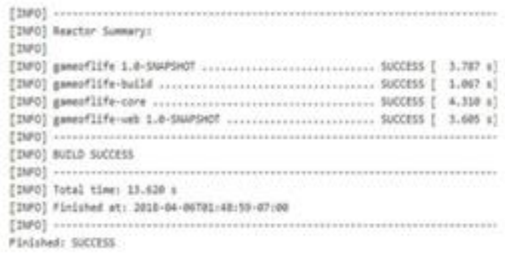


Figure 4: An attempt to build the Application

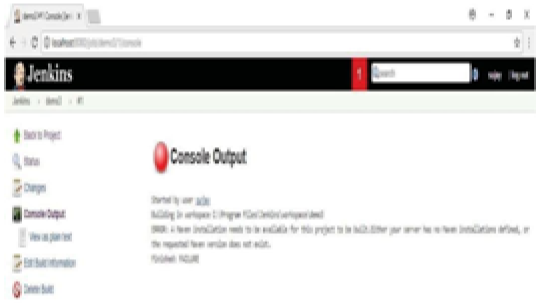
If the code of the application satisfies all the necessary criteria and contains all the requested files to build then it is shown with the color blue and a message in the console output as 'Build Success'. This is shown in the figure 5.



```
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary:
[INFO]
[INFO] gameoflife 1.0-SNAPSHOT ............................ SUCCESS [  3.787 s]
[INFO] gameoflife-build ................................... SUCCESS [  1.067 s]
[INFO] gameoflife-core .................................... SUCCESS [  4.310 s]
[INFO] gameoflife-web 1.0-SNAPSHOT ........................ SUCCESS [  3.605 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 13.620 s
[INFO] Finished at: 2018-04-06T01:48:59-07:00
[INFO] ------------------------------------------------------------------------
Finished: SUCCESS
```

Figure 5: Build of the application is successful

Incase if the application criteria does not satisfy the given requirements, for example if requested version of Maven does not exist, then it is indicated with red color and a message in the console output as 'Failure'. This is shown in the figure6.



```
Started by user [illegible]
Building in workspace [illegible]
ERROR: A Maven installation needs to be available for this project to be built.Either your server has no Maven installations defined, or
the requested Maven version does not exist.
Finished: FAILURE
```

Figure 6: Build of the application is unsuccessful

V. CONCLUSION

DevOps is neither a job nor a thing. It is a philosophical way to solve a problem. In a software industry where functional boundaries are marked and developed rapidly, the way of working with DevOps unites cross functional teams that is the development team and the operation team, which are reliant on each other for mutual success. With the implementation of DevOps workflow, there is a steep learning curve involved in getting internal teams productive.

The various limitations include, good domain knowledge is required for automation projects involving serious integration work. It is hard to hire good DevOps engineers. Most of them are system admins and operators who are unable to code or who may not understand infrastructure and systems, depending on their background.

REFERENCES

1. *Hui Kang, " Container and Microservice Driven Design for Cloud Infrastructure DevOps", in 2016 IEEE International Conference on Cloud Engineering. 978-1-5090-1961-8/16 $31.00 © 2016 IEEE, pp. 202-211.*
2. *Manuel Peuster, "Profile Your Chains, Not Functions: Automated Network Service Profiling in DevOps Environments", in 2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN). 978-1-5386-3285-7/17/$31.00 ©2017 IEEE, pp. 1-6.*
3. *Elisabetta Di Nitto, "DevOps: Introducing Infrastructure-as-Code", in 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C). 978-1-5386-1589-8/17 $31.00 © 2017 IEEE, pp. 497-498.*
4. *Alberto Belalcázar,"Incorporation of Good Practices in the Development and Deployment of Applications through Alignment of ITIL and Devops", in 2017 International Conference on Information Systems and Computer Science (INCISCOS). 978-1-5386-2644-3/17 $31.00 © 2017 IEEE, pp. 224-230.*
5. *Cornel Barna, "Delivering Elastic Containerized Cloud Applications to Enable DevOps", in 2017 IEEE/ACM 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS) 978-1-5386-1550-8/17 $31.00 © 2017 IEEE, pp. 65-75.*
6. *Elisa Diel, "Communication Challenges and Strategies in Distributed DevOps", in proceedings of the 11th International Conference on Global Software Engineering. 2329-6313/16 $31.00 © 2016 IEEE, pp. 24-28.*
7. *Guoping Rong, "DevOpsEnvy: An Education Support System for DevOps", in proceedings of the 30th IEEE Conference on Software Engineering Education and Training. 2377-570X/17 $31.00 © 2017 IEEE, pp. 37-46.*
8. *Teemu Laukkarinen, "DevOps in Regulated Software Development: Case Medical Devices", in IEEE/ACM 39th International Conference on Software Engineering: New Ideas and Emerging Technologies Result. 978-1-5386-2675-7/17 $31.00 © 2017 IEEE, pp. 15-18.*
9. *Pulasthi Perera, "Improve Software Quality through Practicing DevOps", in International Conference on Advances in ICT for Emerging Regions (ICTer): 013 – 018. 978-1-5386-2444-9/17/$31.00 © 2017 IEEE, pp. 1-6*
10. *Erkuden Rios, " Dynamic Security Assurance in Multi-Cloud DevOps", in 3rd IEEE Workshop on Security and Privacy in the Cloud (SPC 2017). 978-1-5386-0683-4/17/$31.00 ©2017 IEEE, pp. 467-475.*
11. *Robert Heinrich, "Architectural Runtime Modelling and Visualization for Quality-Aware DevOps in Cloud Applications", in 2017 IEEE International Conference on Software Architecture Workshops (ICSAW). 978-1-5090-4793-2/17 $31.00 © 2017 IEEE, pp. 199-201.*
12. *Guoping Rong, "CMMI Guided Process Improvement for DevOps Projects: An Exploratory Case Study", in 2016 IEEE/ACM International Conference on Software and System Processes(ICSSP). c 2016 ACM. ISBN 978-1-4503- 4188-2/16/05. . . $15.00, pp. 76-85.*
13. *Tsuneo Nakanishi, "Dynamic SPL and Derivative Development with Uncertainty Management for DevOps", in 2016 5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI). 978-1-4673-8985-3/16$31.00 © 2016 IEEE, pp. 244-249.*