2018 International Conference on Advances in Computing and Communication Engineering (ICACCE-2018)
Paris, France 22-23 June 2018

Continuous and Integrated Software Development using DevOps

Aayush Agarwal[1], Subhash Gupta[2], Tanupriya Choudhury[3]
Amity University Uttar Pradesh,Noida,India[1,2],University of Petroleum and Energy Studies,Dehradun,India[3]
amityaayush@gmail.com[1],scgupta@amity.edu[2],tanupriya1986@gmail.com[3]

Abstract: Devops is all about culture, automation, measurement and sharing (CAMS).It is gaining popularity because of its continuous approach – continuous integration(CI), continuous deployment (CD) and software delivery. CI requires the developer to commit the code several times in a day followed by automatic build and test and immediate feedback to the developer whenever any bug is encountered. If no bug is encountered, the committed code is pushed to the production. According to the 2017 State Of Devops Report, high performance companies like Amazon and Netflix deploy thousands of times per day. Configuration management allows the rollback of previous code for the developer. However several factors prevent the organizations from adopting these approaches like lack of fully automated acceptance test, poor rollback methodology, manually driven quality check etc. This research paper provides various methodologies and tools that can constitute an effective CD/CI pipeline.

Keywords: Devops, deployment, integration, performance, Testing

1. INTRODUCTION

Continuous Delivery (CDE), Continuous Deployment (CD) is a key practice for making software development process reliable and faster. The feedbacks from the Production and Operations team are made available to the developer at frequent stages facilitating improvement and automation. CDE is not a fully automated process. It simply means that the application is potentially capable of being deployed. There is an obvious waste of time in decision making process. Whereas,CD implies that the application's source code can be immediately pushed to production without any manual decision making. This saves a lot of time in software development and allows early and frequent feedback. Also, various automatic configuration tools like Jenkins can make an application compatible in various environments. For example – An application works in developer's laptop but fails when it is transferred to production or testing. This can be irritating. But the research will provide how various Devops tools can work together effectively to solve such problems. The research paper deals with the following research questions:

RQ1: What tools and methodologies can be adopted at each stage of CD/CI pipeline for effective software development?
RQ2: What are various challenges faced by organizations while adopting CD and CDE practices?
RQ3: What are possible solutions to challenges in automatic deployment and integration practices?

2. LITERATURE SURVEY

Stephen Macdonell [4] illustrated that DevOps was seen not just a change in processes but a major change in culture of software development. Also it presented how gaps surrounding continuous deployment to integrated software be reduced to minimum.
Liamping Chen [2] worked towards adopting continuous deployment which focusses on how software companies can benefit from applying several automated tools for automatic deployment and almost every aspect of SDLC.
Mojtaba Shahin [1] performed the empirical investigation on deployment challenges. It provided the survey results conducted on small and big organizations adopting DevOps regarding the several challenges faced in automatic deployment activity. It was found that lack of better tools and manually driven non- technical activities like bureaucracy also badly impact such practices.
John Ferguson Smart [6] gave the main idea about automated acceptance testing with Screenplay pattern. The idea presented here was to make continuous deployment, configuration of software components possible by making automated acceptance test possible through tests written in view of the actor who interacts rather than interactions with the system.

Thus based majorly on above reports an algorithmic approach was developed whose core lies on the actor of the system which can make it easier for organizations to achieve automated acceptance tests and also minimize other challenges.

Present Work

A systematic study and research was involved where around 25 papers which included IEEE research, Google Scholar, ACM library etc. were studied in detail and a flow of processes and tools was mapped. Also, the research paper involved studying the latest trends in Devops with the help of State of Devops Report published in 2016 and 2017. [8]
The review method involved applying the keywords relating to the paper to major data sources like IEEE Xplore, ACM Digital Library and Science Direct. The major labels that were

978-1-5386-4485-0/18/$31.00 ©2018 IEEE

searched for include continuous deployment, DevOps tools, and automatic acceptance testing. [7]

Current state of automation in CD/CI Pipeline

It is quite evident that Automatic CD/CI pipeline allows building, testing and configuring immediately as soon as the code gets committed on a source repository like git and thus accelerating the feedback process. According to a survey regarding scale of automation over 70% (69/98) respondents practiced semi-automated CD/CI pipeline.[2]

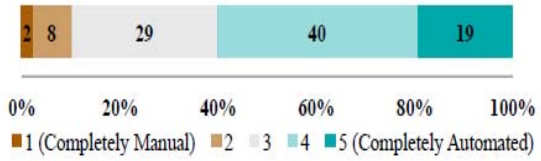

*Figure*1: Scale of automation in CD/CI pipeline

Regarding a survey on availability of tools required for automation, Following results were found:

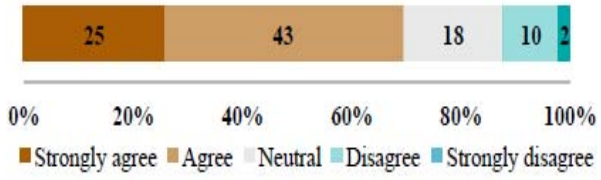

*Figure*2: Availability of tools for Automation

3. TOOLS AND APPROACHES

A. Automatic Commit and deploy with Jenkins

The problems faced before Automatic and continuous integration includes:

- Developer has to wait too long for the test results, and thus wasting a lot of time that could have been well spent in innovation and developing new product.
- Developer need to go through the entire source code to fix any bug encountered.
- No continuous feedback from the production or test team at every stage.

Consider a Case study of Nokia that adopted *Nightly build* approach, where the source code was pulled from the repository only at night. After CI, code is pulled over every commit and thus immediate build, test followed by appropriate feedback.

Now the developer will be required to go through the source code of last commit and not the entire program.

B. Automated Testing using Selenium

Before the code can be deployed, it requires testing which can be efficiently done using automated tools like Selenium, Test NG etc. This provides several benefits like we can trigger them to perform at a particular timing; they manage report generation, and reduce mistakes.

But several of the current tools pose challenges like they do not have support for testing images, QR codes, and also we can use Selenium just for web application. The thing is discussed in detail later along with the solution approach.

C. Version Control with Git

Git allows each programmer to maintain and local repository of its own and push changes to the centralized repository or pull requests from the centralized node. The older version is available of the code whenever needed.

D. Containerization with Docker

Imagine the plight of a developer whose code works on his machine but it gives error on production side or the tester side. This is irritating. It can be a case of different environment. For example – The development unit might be using Tomcat 9 version, On the other hand Test teams are still using Tomcat 7.What if we use something known as a Docker container in which several versions of the server are installed with the application?

The micro services before Docker were to be implemented on virtual machines that resided over the host OS. This had the following drawbacks:

- Wastage of resources like RAM
- Instability in performance.

Whereas, The Docker containers are light weight alternatives to Virtual machine (VM) that will run over simply a single VM above the host.[4] They help in providing a consistent environment throughout the SDLC and also eliminate the need of allocation of RAM to each container.

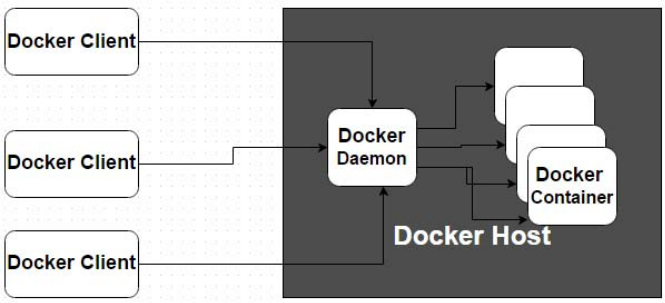

*Figure*3: Working of Docker containers

E. Configuration Management using Puppet and Chef-

Take a case when someone wants to deploy a code or OS on hundreds of systems. Carrying it out manually is time consuming and might not prove to be something reliable in the upcoming days. Also In case of any error, rollback to the previous version is quite difficult.

But automated tools like Chef and Puppet require only managing of the infrastructure of a centralized server and all the other nodes can get automatically configured using push/pull mechanism. [4]

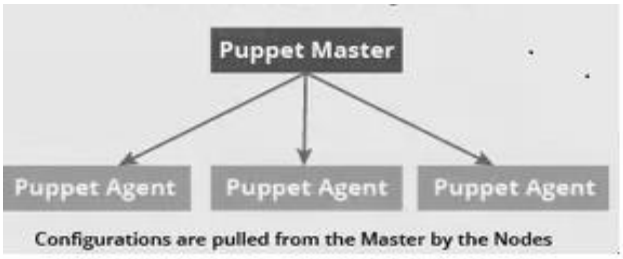

*Figure*4: Push and pull configuration using Puppet

What happened in New York Stock exchange (NYSE) describes how Configuration management can be crucial.Here, the installation of some new software disrupted the trading for 90 minutes, but their configuration management allowed NYSE to recover back the old software in less amount of time. Another example is Zynga social game developers which use thousands of servers. Manual configuration can lead to inconsistency, importability and reduced speed of recovery from failure. Therefore, an automatic configuration management tool like Puppet is needed to maintain a consistent state among all the servers.

4. CHALLENGES

1. Long lived Branching- Stay away from long branching. Since Devops incorporates various automation tools at all stages of SDLC, it is preferable to adapt Trunk based development.
2. Lack of experience and skill – The use of various automated tools in CD/CI pipeline requires necessary technical and soft skills.
3. Lack of fully automated acceptance test – According to some surveys, many organizations resist undertaking automated testing because of various factors like unavailability of tools, low gain with much overhead etc.[1]
4. Communication gap between Dev and Ops – Developer team always looks for new technology and new changes. On the other hand, Change is an enemy for operations team. It relies on reliability and stability.
5. Manual code review – Even though many organizations claim to practice automatic deployment, yet there are quality checks within the team because of some sort of caution around.
6. Deployment as business decision – Deployment is a bureaucratic process in many organizations which requires approach of managers.
7. No effective Rollback methods – According to 2017 State of Devops Report, around 24.5% of organizations surveyed resisted automatic deployment due to manually driven rollback mechanism.[5]

5. PROPOSED SOLUTION

As mentioned above, one of the major challenges in adoption of Continuous deployment and integration as a part of Devops is the lack of acceptance tests that are fully automated, and goal oriented. In the current scenario, many tools like Selenium focus on interactions and clicks a user performs, rather than requirements.

The solution comes from writing the test codes not in the form of interactions but focusing directly on the user's goal. The approach is somewhat similar to SDLC models we adopt during software development. The problem faced in current scenario by most of the organizations is lack of better tools and often, the other units find it difficult to understand what the tests do and hidden data.[6]

Consider the case of Todo application with the following components:
Actor
Goals (like Record things to do, find things marked as important)
Tasks (They include entering the things to be done, filtering the list. marking things as important)
Interaction (the user interaction with the application through clicks like Enter pressed, typing something)

Look at this sample case:
Scenario To add a To-do thing
Given An empty Todo list
When actor tries to add "Make a tea"
Then Todo list must contain "Make a tea"

A. Selenium approach

The focus here is largely on interactions that an operator does with the application. The test therefore consists of expressing what needs to be done in the form of clicks, identifiers, links, text, css, tag names etc. The approach followed here will look like: [6]
1. Instantiate a driver like Firefox Driver.
2. Move to the add field in Todo using driver instance.
inputElement = driver.find_element_by_name ("Add item");
3. Now you type in the add field "Make a tea" by submitting form. This needs to be done using sending appropriate key.
inputElement.sendKeys ("Make a tea");

Obviously, finding elements using their identifier adds to overhead. This approach had several drawbacks. It encompasses unreliable and difficult to understand tests. Thus they make it difficult to produce comprehensive documentation and reports.

B. Proposed approach

The approach provides simpler to read, simpler to extend techniques for acceptance testing. This test is built upon the actor, and not the interactions with the system.

1. An actor is added to scenario of test.
Actor aayush=Actor.*named*("aayush");

2. Instantiate Web Driver instance.
3. Then, You can give the actor 'abilities'. For instance, it can be browsing the web. Since it is a precondition, we can use

simple syntax methods like *givenThat()* or *@Before* method in JUnit.
A better way is to use a builder pattern which can be coded in simple English language.
4. Now the actor can perform tasks to achieve goals. This can be implemented using annotations supported in some open source libraries.

AddTodoItemClass.*called(*"Make a tea");
The class implements the Task interface.
5) Finally we can code a Test which is in a readable form when we use API methods of open source which can support readable codes like Screenplay pattern.

```
@Test
 public void shouldBeAbleToAddthings

  //Given
 aayush.isOnEmptyToDo ();

  //When
aayush.attemptsToAdd("Make a tea");
//Then
ayush.seeItems("Make a tea");
```

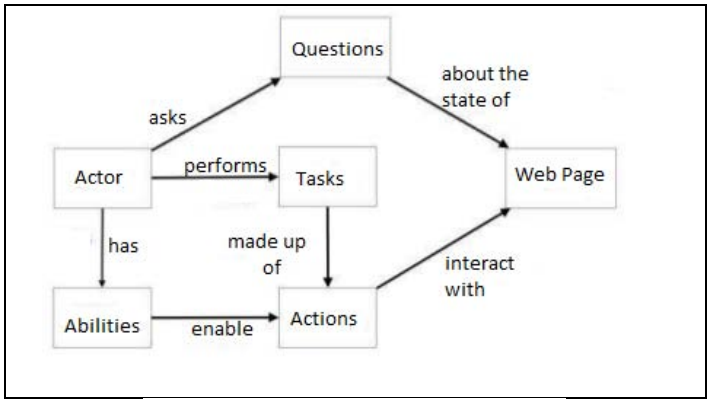

Figure5: Activity flow for Todo application

Table: Comparative analysis of Selenium approach and proposed approach

Parameter	Selenium Approach	Proposed Approach
Basis	Interaction based , hence not deterministic	Requirement based , hence deterministic
Time Complexity	Exponential in some cases	Majorly polynomial
Executed test result	Basic information	Includes test scenario, passes, pending, failed etc.
Success rate	Around 80.6 %	90-95%
Applicability	Web application test	Also includes QR codes, various formats etc.

6. CONCLUSION

This research paper provides an empirical investigation on several tools and challenges encountered during the adoption of continuous practices in Software develops using DevOps. The paper provides several challenges and how several computer science fields can help in dealing with those challenges. Also it was found that relatively few research papers and scholarly articles have been published till date about Devops practices, and the area demands greater extent of research.

REFERENCES

[1] J. Mojtaba Shahin, Muhammad Ali Babar, Mansooreh Zahedi, Liming Zhu on "Beyond Continuous Delivery: An Empirical Investigation of Continuous Deployment Challenges," In Proceedings of 11th international Symposium on Empirical Software Engineering and Measurement (ESEM), Toronto, Canada.

[2] Lianping Chen on "Continuous Delivery: Overcoming adoption challenges" , The Journal of Systems and Software 128 (2017) 72–86, Science Direct

[3] Sagar Narendrasing Deshmukh on "A System for Application Deployment Automation", International Conference on Innovations in Power and Advanced Computing Technologies [i-PACT2017]

[4] Waqar Hussain, Stephen MacDonell on "Emerging Trend for Global Devops", 2017 IEEE 12th International Conference on Global Software Engineering (ICGSE)

[5] Morgan B. Kamuto, Josef L. Langerman on "Factors inhibiting adoption of Devops in large organizations", 2017 2nd IEEE International Conference On Recent Trends In Electronics Information & Communication Technology, May 19-20, 2017, India

[6] John Ferguson Smart, Antony Marcano, Andy Palmer, Jan Molak on "Automated acceptance testing with Screenplay pattern" IEEE 12th International Conference on Global Software Engineering (ICGSE)

[7] "Five Habits of Highly Successful Continuous Delivery Practitioners, at http://info.perforce.com/continuous-delivery-five-habits-successfulpractitioners.html?sc=blog , [Last accessed: 5 March 2018]."

[8] J. Humble. "Principle 2: Decouple Deployment and Release"at:http://www.informit.com/articles/article.aspx?p=1833567&seqNum=2 [Last accessed: 15 March 2018].

[9] B. A. Kitchenham,and S. L. Pfleeger, "Personal Opinion Surveys, "Guide to Advanced Empirical Software Engineering, F. Shull, J. Singerand D. I. K. Sjøberg, eds., pp. 63-92, London: Springer London, 2008.

[10] 2017 State of Devops Report available at https://puppet.com/resources/whitepaper/state-of-devops-report [Last accessed: 5 March 2018]