EE295 TECHNICAL WRITING

Charles W. Davidson College of Engineering San Jose State University

RESEARCH REPORT

ON

DevOps (Development & Operations)

BY:

SWAPNASHEEL DATTU SONKAMBLE
SJSU ID: 011452610

ABSTRACT

DevOps presents a significant broader perspective in delivering software solutions. The perception of DevOps being a mere tool has been formulated into a change in culture, paradigms in automation, clarity in measuring risks and facilitating ease of sharing. DevOps is responsible for companies to move from a monolithic pace of software releases to an agile mindset. This report discovers the significance of DevOps and tries to explore the process of transformation from a legacy method onto an impact of change on the overall architecture, software scale out, and growth of the company. This transformation from a legacy mindset to agile mindset requires a formidable change to the organizational setup. The journey of adoption of such a culture requires numerous degrees of effort, techniques, and tools. In this report, we cover the concepts, adoption strategies and the impact of these adoption strategies leading towards the growth of the organization.

Table of Contents

LIST OF FIGURES AND TABLES

SUMMARY

Development & Operations (DevOps) integrates the two worlds of development and operations, using automated developments, deployments, and infrastructure monitoring. It's an organizational shift in which, instead of distributed solitary groups performing functions separately, cross-functional teams work on continuous operational feature deliveries. This approach helps deliver value faster and continuously, reducing problems due to miscommunication between team members and accelerating problem resolution.

In this report we focus on the methods used in DevOps to obtain a shift towards a collaboration between development, quality assurance, and operations. We also study DevOps focus on automation where development can automate code testing, workflows, and infrastructures that will help the developers to write small chunks of software to be added in hours instead of large chunks in weeks or months. This adds new dimensions of benefits to the build and delivery of the product like [1] –

- Time saving – Faster time to market
- Automation
- Low Hardware dedication requirement
- Reliable and Scalable infrastructure provisioning
- Periodic and continuous deployment and development

INTRODUCTION

Over the years organizations have adopted many process optimizations in their software development (agile transformation) practices. However, in this entire evolution – the focus has been mainly towards software development, leaving the operations side of the software delivery lagging behind in the optimization race. As a result, software development teams are able to deliver at a much faster rate than the pace at which the operations can absorb. Therefore, whatever optimizations one is doing in the software delivery cycle and if one unit is not able to keep up the pace, entire delivery of the product will be delayed.

In a highly competitive environment, for any software to survive in the market, it is evident that there should be flexible and better way of approaching customer needs and software delivery. To retain dominance of the product among the competitors, being agile in nature will result in a marketing edge which is only possible with a channel of communication and understanding between the development and operations teams. In a case of an enterprise, the scenario becomes complicated and to ease this, we can define new paradigm with a DevOps mindset [5].

DevOps is set of practices that try to bridge developer-operations gap at the core of things but not limited to this development and operations handoff instead covers all the aspects which help in optimizing, speeding and high quality software delivery [3]. DevOps is also set of principles towards software delivery where the key focus is on continuous testing in the production like environment, speed of delivery – be it in shippable state at any day, continuous feedback, and the ability to react to the change more quickly as a team accomplishing a common goal [4]. The culture of DevOps has been acting as an enabler to deliver

more features continually while maintaining security and stability of the system. DevOps extends agile principles to entire software delivery pipeline making sure operations team can run along with the development teams.

METHODS

DevOps is applied to various phases of the software delivery process. Following are the methods used [1] _

1. Continuous Planning
    - Businesses plans have to be agile to an extent, i.e. able to adjust quickly to the changing market conditions. Always have interim checkpoints in the plan to reassess the situation and modify/adjust the plans as needed based on the market feedback.
    - It is difficult for development or testing teams to adapt to the quick changes required in the business environments. Thus DevOps allows you to adapt by always having a prioritized product backlog all the time, a continuous channel of feedback with the customers, and directly taking the business angle in consideration. This is a continuous process to plan small portion – execute – get feedback – react to feedback and adjust the plan if needed and the cycle continuous.

2. Continuous Integration
    - This refers to integrate early, do not keep changes localized to your workspace/ development area for long, instead share tour changes with the team and validate how the code behaves continuously.
    - Further of optimization, achieve automation such that as soon as developer delivers the changes, the build system detects the changes (may even be before the scheduled event) and trigger a build carries out tests and post the build into the repository so that other team

members can validate and test the change to a further extent. This has to be a repeatable and a continuous process within the development cycle of the product or software.

3. Continuous Deployment
    - This method is the heart of the DevOps and forms the critical piece of the overall software delivery optimization.
    - Setting up a hardware for the operations team and test the changes can take time varying from days to weeks. Thus it is important to have same development and operation environment. It often happens that manuals are not consistent and software's that ran on the production/ development doesn't necessarily run on the operations – resulting in a delay.
    - DevOps approach proposes that the entire infrastructure provisioning should be maintained as code in source code repository – concept being called Infrastructure as a code (IAAC) [6].

4. Continuous Testing
    - Prerequisite to continuous testing is – *Automate every test case!!* . Any process that has to be repeated over time – should get automated, there are technologies available to meet that goal.
    - The manual testing process must be evaluated for the possibility of automation such that software delivery process should be able to execute the test suite without the need of any user intervention; thus resulting in releases of quality products.

- The whole principle of continuous testing is to move the testing process to early in the production cycle.

5. Continuous monitoring
    - This enables the capability to test early on a production like a system and observe various quality parameters throughout resulting in quality products and an ability to react to any surprises in a timely manner.

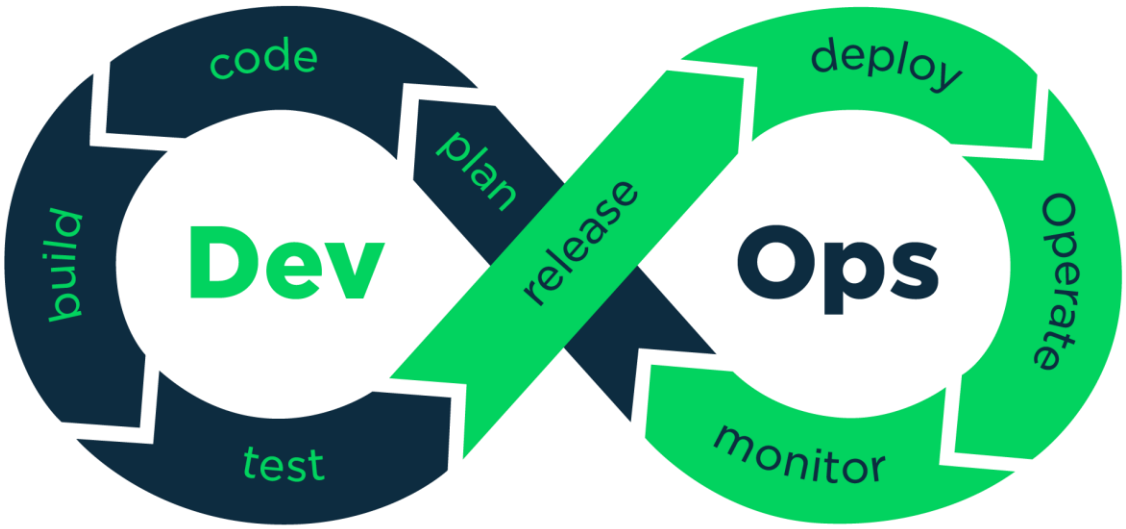

Figure 1 DevOps Methodologies [8]

Procedures and DevOps Tools:

Methods of DevOps can be gained with the help DevOps tools. Tools are mandatory in automating DevOps as quality delivery in a short cycle need a high degree of automation. Thus choosing the right tools for the production is important when we move to DevOps. Table below gives a brief description of the different build tools used in different stages.

1. Build tools
    - With agile and fluid development, build tools have become tools for managing the software development and service life cycle, which involve compiling code, managing dependencies, generating documentations, running tests, or deploying an application to different environments.
    - Apache Ant has become the standard tool for building applications, especially in open source when you need to build from the sources. Ant uses XML, which is verbose for simple tasks and isn't a procedural programming language.
    - Maven aims to solve some Ant's problems. It also uses XML, but the statements define the nature of the project instead of the tasks that build it.

2. Continuous-integration tools
    - CI is a key to the agile concept. CI tools integrate developers work as early and as often as possible. In this way, the system is continuously tested.
    - Jenkins is an open source, Java-based system and one of the most implemented tools, thus having many plugin options. Because of this, finding support is fairly easy from its large user base.

3. Deployment

- During this phase, the most important shift is to treat infrastructure as a code. This makes it easy to be shared, tested, and version controlled (using GIT version control toolkit).
- DevOps pushes automation from application to infrastructure. When compared this with manual infrastructure provisioning, configuration management tools can reduce provisioning and increase complications while recreating production systems on the development machines.
- Puppet-based infrastructure generally consists of a master server with agents on each client giving an approach to provision the system's desired end state [7].
- Ansible is the easiest to implement because it doesn't require agents to be installed on the client machines, as it uses SSH (secure shell) to push configurations. It's an open source tool based on Python but configured in YAML (YAML Ain't Markup Language) file coding, reducing the learning curve.

Tool	DevOps Phase	Tool type	Configuration format	Language	License
Ant	Build	Build	XML	Java	Apache
Maven	Build	Build	XML	Java	Apache
Jenkins	Build	Continuous Integration	UI	Java	MIT
Bamboo	Build	Continuous Integration	UI	Java	Commercial
Puppet	Deployment	Continuous Integration	DSL similar to JSON	Ruby	Apache
Chef	Deployment	Continuous Integration	Ruby-based DSL	Ruby	Apache
Ansible	Deployment	Continuous Integration	YAML	Python	Apache

Figure 2 Build Tools for DevOps [1]

REFERENCES

[1] Christof Ebert, Gorka Gallardo, Josune Hernantes, Nicolas Serrano *"DevOps"*. IEEE Software, Year 2016, Volume: 33, Issue: 3, pages: 94-100, URL: http://ieeexplore.ieee.org/document/7458761/

[2] Edwards, Damon. "*What is DevOps*" (2010); URL: http://dev2ops.org/2010/02/what-is-devops/

[3] "*Understanding DevOps – Infrastructure as a Code"*, URL: https://sdarchitect.wordpress.com/2012/12/13/infrastructure-as-code/

[4] *"DevOps for Dummies – by Sanjeev Sharma"*, URL: https://www14.software.ibm.com/webapp/iwm/web/signup.do?source=swg-rtl-sdwp&S_PKG=ov18162

[5] Prabal Mahanta, Vittalraya Shenoy Adige, Anil Kumar Pole, Rajkumar M; *"DevOps Culture and its ipact on Cloud Delivery and Software Development"*; 2016 International Conference on Advances in Computing, Communication, & Automation (ICACCA)
URL: http://ieeexplore.ieee.org/document/7578902/

[6] "*How to do a full stack release with IBM UrbanCode Deploy and IBM SmartCloud Orchestrator*", URL:https://www.ibm.com/developerworks/community/blogs/invisiblethread/entry/how_to_do_a_full_stack_release_with_ibm_urbancode_deploy_and_ibm_smartcloud_orchestrator1?lang=en

[7] The Puppet enterprise architecture
URL: https://docs.puppet.com/pe/latest/pe_architecture_overview.html

[8] https://www.suse.com/promo/devops/