Defining DevOps

Cadegan-Syms, Joshua
B00528943

I. ABSTRACT

DevOps is an umbrella term used to describe a variety of methods employed by companies and organizations seeking to adopt agile and lean software development techniques as a means to increase the pace by which software is developed and deployed, as well as to improve the overall quality of products [21][22]. DevOps is a loosely defined set of practices wherein development and operations teams collaborate to build, test, and release software faster and more reliably. However, while DevOps continues to increase in popularity and practice, researchers and practitioners have yet to produce a definitive definition of the practices, methods, and tools that define and encapsulate DevOps. As such, for the purposes of this paper we will be using a conceptual framework as means to define DevOps and will focus on how collaboration, automation, measurement, and monitoring can be used in conjunction with continuous implementation, development, and delivery as a means to effectively deploy DevOps[21][22].

Key words- Continuous integration, continuous deployment, continuous development, configuration as code, infrastructure as a code.

II. INTRODUCTION

Changes in agile methods have presented practitioners and researchers with the desire to produce a more holistic approach to software delivery cycles [21]. While each agile method is unique in its approach, fundamentally, each seeks to incorporate iterative and continuous processes as a means to successfully refine and deliver better products [7]. Traditionally, these continuous applications are used in agile methods throughout both the development and operations phases and are used to plan, test, integrate, and deploy software products[5][6][8]. As a result, when compared to the traditional waterfall-style process, agile methods tend to result in products that are more lightweight, and adaptable.

DevOps is an extension of agile methods and was first coined by Patrick Debois and Andrew Shafer in 2008 as as means to streamline the software development process, empower full ownership of software applications, improve communication between/amongst teams, and promote team cohesion throughout the deployment and delivery stages [3][5][6][8][22].

Collaboration is an integral component of DevOps and is meant, among other things, to promote greater cohesion between development and operations teams by way of continuous integration, delivery and deployment. This continuous pipeline is supported by tools, methods, and practices meant to facilitate in addition to collaboration: automation, measurement, and monitoring [20]. As a result, DevOps teams that are able to build, test, and release software faster and more reliably [7]. According to the 2015 State of DevOps Report, DevOps can result in high-performing teams able to deploy 30 times more frequently, while having 60 times less failures and the ability to recover 168 times faster than teams that do not deploy DevOps methods[9].

However, despite the claims made by the 2015 State of DevOps Report and it's increasing ubiquity among practitioners, the exact practices, methods, tools and even benefits associated with DevOps continues to be undefined [8][13]. This lack of definition is both reflected in the literature, as well as throughout the industry as no conclusive unifying definition of DevOps has emerged both among industry insiders as well as researchers [8][13].

Therefore, for the purposes of our paper, we have chosen to focus on DevOps as a conceptual framework and will explore how collaboration, automation, measurement and monitoring can be used as a means to support a developmental and operational pipeline in order to provide continuous integration, delivery, and deployment of software systems and products. We will also categorize and explore some of the tools used by industry professionals as a means of deploying this continuous pipeline. Lastly, our paper will offer a literature review as well as a critique of some of the research into DevOps.

III. CONTINUOUS PIPELINE

DevOps first emerged as an extension of Lean and Agile methods. While DevOps continues to be loosely defined, continuous integration, development and deployment are commonly associated practices used to reduce average release cycle times, improve software quality, and improve team cohesion [4] [12].

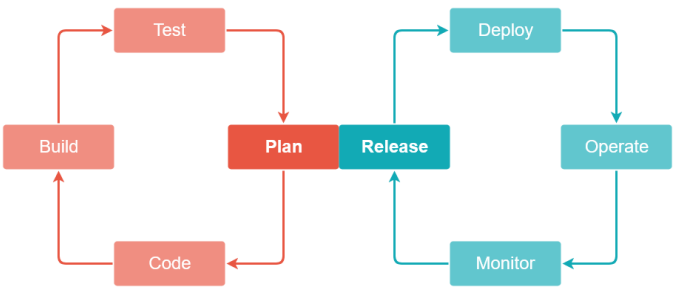

Fig. 1. Different aspects of Dev and Ops. Red shows work done by Dev while blue shows the work done by Ops.

A. CONTINUOUS INTEGRATION

In practice, teams deploying continuous integration regularly push changes to software to a main repository. These changes are then validated through the initialization of a build cycle and then tested. Continuous integration is highly dependant on automated testing in order to guarantee that changes to the main repository are integrated effectively, efficiently, and without disrupting the overall system [18][20].

As a result, continuous integration claims to be able to identify and fix bugs and errors earlier in the development process, resulting in build releases that are smoother, while also reducing the cost and time spent testing [18][19]. In order to achieve this, it is necessary for testing to be automated as a means to enable developers to capture issues early and thus minimize disruptions in the work flow and the development process. These tests include, but are not limited to:

Unit Testing - Used to ensure behaviour for methods or functions. Integration Testing - Ensures that multiple components interact correctly. Acceptance Testing - Focuses on testing business cases. UI Testing- Tests the function of applications a user perspective [18][19].

B. CONTINUOUS DELIVERY

Commonly viewed as an extension of continuous integration, continuous delivery requires releasing changes or updates systems to customers both quickly and seamlessly. Just as with continuous integration, this requires the automation of testing, as well as an automated release process. Continuous delivery offers companies the ability to release changes at a pace that best reflects organizational requirements and is best deployed as early and often as possible. This requires developers to release small batches, enabling them to troubleshoot issues as they arise [18] [20].

When deployed effectively, continuous delivery can lead to a reduction in the complexity of software deployment. As a result, because teams are not required to spend time preparing, releases are able to occur more frequently, thus can result in more feedback from end users which can then be used to improve the system[18].

As software makes its way to production, continuous delivery is supported by automating build, test and deployment. These pipelines are meant to automatically determine whether a commit, or a batch of commits, ultimately makes its way to production, ensuring the delivery of quality software frequently and predictably [18].

C. CONTINUOUS DEPLOYMENT

Just as continuous delivery extends continuous integration, continuous deployment is seen as an extension of continuous delivery; however, unlike with continuous delivery, continuous deployment releases all software that successfully passes every stage of production, without human intervention [18]. Only software that fails testing is prevented from being released [20].

The benefits of adopting continuous deployment include development that is faster as a result of deployments being triggered automatically, as well as a reduction in risk as of small batches or fixes are continuously deployed allowing end users see a continuous pace for improvements as quality continuously improves [18].

As such continuous deployment is a full end-to- end automated software pipeline offering a high level productivity allowing DevOps teams to rapidly deploy and validate new software ideas and features. As such, continuous delivery is the precursor to deployment and is the final approval step prior to releasing to end users [18].

D. CONTINUOUS DELIVERY VS. CONTINUOUS DEPLOYMENT

While continuous delivery and deployment may seem similar, it is important that we take the time to underline their differences as a means to understand the important and distinct roles each plays in DevOps. In order to achieve this, we will rely on a metaphor of receiving a package from an online store provided to us by industry insider and researcher Sten Pittet as a means to clearly differentiate the difference between continuous delivery and deployment

Imagine waiting for a package to arrive and coordinating with a delivery service. Once the package has been delivered, it is then opened and its contents inspected in order to determine whether they matched expectations. If the contents fail to match expectations, it is then returned; however, if the contents are correct, we can then use or "deploy" our new purchase [18].

During the delivery phase, teams review and merge components that are then packaged and moved through a production environment where it awaits approval for deployment, during which they are reviewed by means of automated checks. Deployment is the final step of the overall continuous pipeline that consists of integration, delivery, and deployment [18].

IV. IMPLEMENTATION

A. COLLABORATION

While academics and professionals differ in their identification of the main defining characteristics associated with DevOps, there is consensus that collaboration is a fundamental and defining characteristic [1][12][17] . Collaboration is meant to encourage a culture of openness and communication both within a team as well as between both software development and operational teams [15]. This enables organizations to break down barriers between development and operation, resulting in a more streamlined process.

Simply put, collaboration means that DevOps teams work and create together. Software development requires regular and effective communication between teams throughout the development and operations phases. By effectively collaborating, teams can reduce operational overhead and delays by providing feedback and communication thus allowing teams to make necessary changes quickly whilst maintaining a stable and robust environment [2][13][16].

However, simply adopting collaboration tools is insufficient enough to produce the necessary communication needed;

instead, organizations need to adopt a complete cultural change in order to facilitate a level of collaboration that will lead to a reduction in operational overhead, delays, and in better software overall.

Firstly teams need to identify a common set of objectives for all members. Often members of teams struggle when priorities are unclear. A common set of priorities lays the foundation for developing an approach and working together to achieve the goals. Next, cohesion is required in order to build trust and mutual respect amongst teams [16][17][20]. This begins with Team leaders, who must actively cultivate a culture of trust, honesty and respect as a means to ensure members feel part of a team. Lastly, by providing a road map illustrate how teams will work together in order to achieve objectives. An effective road map will help outline each member's roles, responsibilities and duties, as well as how show them how their work fits into the bigger picture [16][17][20].

B. AUTOMATION

Automation is utilized through both the development and operational cycles and is a key component of DevOps effective deployment of a continuous pipeline. To achieve this, many of the manual processes need to be reduced in order for teams to be nimble, iterative and fast [16]. To achieve this, "infrastructure as code" (IAC) and "configuration as code" (CAC), are both often adopted by DevOps practitioners.

IAC is a concept that emphasizes the development of automation in order for deployment to be used to configure, as well as upgrade software and infrastructure, particularly in a cloud environment. Within this framework, automation allows the infrastructure to be provisioned and the functionality to be executed iteratively and quickly[10][11][15].

With CAC, developers seek to build more modular applications that are reliable and maintainable and whereby migration of code between environments is supported by a form of version control[10] [11].

C. MEASUREMENT

The success of DevOps is largely dependant on efficiency. As such, in order determine whether efforts at continuous improvement benefit the software infrastructure, data is required. DevOps requires the ability to measure development through the incorporation of various metrics as a means to effectively measure efficiency. Forsgren and Kersten [10] suggest using a combination of both system-based and survey based metrics to better understand the DevOps release cycle at a given company. However, those specific metrics change from project-to-project and from company-to-company, and have yet to be definitively defined within the literature or by industry experts [11] [15] [16].

D. MONITORING

Monitoring provides the necessary information needed in order to effectively measure an application's performance. This feedback, or "validated learning", enables teams to measure the impact of changes within the software [10] [11]. As a means of providing adequate monitoring, operations teams utilize a library of monitoring tools and logs in order to produce the required information regarding systems [11].

By encouraging collaboration between developers and operations, DevOps seeks to addresses some of the challenges often associated with monitoring, thus monitoring and logs have become essential components used to determine the overall health of a system[10] [11]. Thus, DevOps requires monitored data to be effectively used as a means to produce analytics that can then be used to integrate data with end user behaviours as a means to provide information to developers and product management for improving software [11] [15] [16].

V. TOOLS

Tools can be used in a variety of ways and are essential components of DevOps. Tools are used to foster collaboration and automation, as well in measuring and monitoring, each of which supports a continuous pipeline for both development and operations teams. One of the main goals of DevOps is to deliver quality software quickly [1]. To achieve this, a high degree of automation is required and is supported by a large library of tools. These tools are used both by development and operations teams in order to test, build, deploy, version control, monitor software, and to facilitate communication and collaboration [1].

New tools are being developed everyday and within the literature over 50 major tools can currently be identified. This vast library allows practitioners to monitor, implement, deploy, automate and analyze software and can be classified into 7 different categories, shown below in table 1.

Tool Category	Description
Continuous Integration	Allows development teams to integrate code within a shared repository, allowing for a verification process by an automated build to allow teams to identify bugs quickly. [1]
Continuous Management	Tools helps DevOps teams to maintain a consistent level of product performance, attributes, as well as a functional state given its requirements, design, and operational information.[1]
Deployment	Allows code to be managed and uploaded to the host servers, managing the process through the entire pipeline and put it into automatic production allowing for continuous deployments.[1][13] packages, and compile it, transforming into executable code[1][13].
Monitoring	Used to make critical decisions regarding performance and service efficiency with each tool focusing on infrastructure, performance, logs…etc [1][13].
Testing	Used to automate debugging in order to identify defects and/or security flows within source code [1][13].
Code analyzer	Tools used to ensure code quality and discover violations of standards[1][13]
Repository	Are used to manage repositories in order to support development teams to manage code states, as well as add/delete features, versioning, and manage access controls.[1][13]

TABLE I: Categories and descriptions for tools used throughout the DevOps communities.

Each category has a multitude of different tools available and it would be impossible to list them all. These range from free to use tools to tools that are made specifically for the team using them. Some examples of tools being: Github (Repository), Git or SVN (Deployment), JUnit (Testing), Jenkins (Continuous Integration) and countless others.

VI. LITERATURE REVIEW

According to several academic literature reviews, research regarding DevOps continues to be impeded by several factors, chief among a general lack of definition. For the purposes of this paper, we have chosen to define and focus our understanding of DevOps as a conceptual framework, or mind-set [22]. However, within the literature there continues to be no clear definition of what exactly DevOps encapsulates. As a result, a robust literature review requires us to take a multidirectional view in order to investigate DevOps from a multitude of perspectives, in order to allow our research to best resolve various conflicting definitions [8] [13]. However, due to the limited scope of this paper, we have chosen to narrow our investigation of DevOps to what we have identified as some of the key components and fundamental practices commonly associated with the practice. By no means is this paper meant to exhaustive, or attempt to provide a definitive definition of the practices, methods and tools associated with DevOps.

While previous research has mostly treated development and operations of software as two different fields of study, as DevOps continues to be embraced by the industry, trends in research show a shift in treating development and operations as a single topic of study: DevOps. While some researchers believe that academic research should not be affected by industry trends or "hype", others, such as Erich and Amrit, believe academic researchers should reflect these changes as a means to produce research that best reflects changes within the industry. This will ultimately result in better, more relevant research which either supports or rejects the perceived benefits of industry practices such as DevOps [8] [13].

As both a term and subject of research, DevOps is a relatively new. As such a cohesive industry wide understanding of what encapsulates DevOps has yet to be satisfactorily defined [19]. As a result, there continues to be an absence of a concrete shared understanding of what DevOps is both in practice as well as any benefits associated with the practice. This, entail, is reflected within the literature and as such, determining the specific principles, practices, or benefits most commonly associated with DevOps continues to be elusive. This absence of a clear definition, combined with both unsatisfactory quantitative and qualitative research, the benefits resulting in DevOps has not effectively communicated or interrogated, and thus its potential industry wide impact not fully understood [13] [15].

Regardless, of the research available, it has been suggested that effective and successful transitions to DevOps methods has been observed in teams where collaboration between development and operations teams is already high, as well as in companies whose cultures already promote active communication [12][14].

As such for those interested in adopting DevOps methods, we feel it necessary to underscore that while there is no set definition of the practice, we can definitively say that DevOps means greater collaboration and communication between development and operations; however, how this is implemented and what that means in practice, changes form company-to-company. Whereas one company may interpret DevOps by adopting specific styles, policies, methods and even tools, others may choose to interpret and practice DevOps differently in such a way that works best for them [7].

In conclusion, in its current state, it is our opinion that the

literature regarding DevOps leaves much to be desired. While we identified practices that we feel are integral to the practice, we cannot provide at this time a definition definition of what DevOp in practice and as such, we cannot convincingly state whether or not DevOps practices has any positive or negative effect on the success of a project. In order to answer these important questions, we believe that future research into DevOps should focus on providing a rigorous and exhaustive exploration of the methods, practices and tools used in DevOps as a means to provide an exact definition of the practice; only then can we begin to explore the potential benefits or drawbacks of the practice.

VII. CONCLUSION

While literature and industry insiders have yet to produce a definitive definition, we have identified continuous implementation, development, and delivery, as essential attributes of DevOps and are implemented by practices and tools that facilitate collaboration, automation, measurement and monitoring. Additionally, of the practices and methods we have identified, collaboration is the most essential to effectively implementing DevOps. However, there continues to be an absence within the literature of a concrete shared understanding of what DevOps means in practice and how DevOps methods can best be most effectively implemented. As such, DevOps is best defined less as a set of defined practices, and more as a conceptual framework used to foster better collaboration between development and operations teams.

VIII. REFERENCES

[1] Aljundi, M. (2018) Tools and Practices to Enhance DevOps Core Values. Lappeenranta University of Technology - School of Business and Management - Degree Program in Computer Science.

[2] Bobrov, E., Bucchiarone, A., Capozucca, A., Guelfi, N., Mazzara, M., & Masyagin, S. (2019). Teaching DevOps in academia and industry: Reflections and vision.

[3] Brunnert, A., Van Hoorn, A., Willnecker, F., Danciu, A., Hasselbring, W., Heger, C., . . . Wert, A. (2015). Performance-oriented DevOps: A Research Agenda.

[4] Davis, J., Daniels, K. (2016). *Effective devOps: building a culture of collaboration, affinity, and tooling at scale*. O'Reilly Media.

[5] Dornenburg, E. (2018). The Path to DevOps. *IEEE Software, 35*(5), 71-75.

[6] Ebert, C., Gallardo, G., Hernantes, J., & Serrano, N. (2016). DevOps. *IEEE Software, 33*(3), 94– 100. doi: https://doi.org/10.1109/MS.2016.68

[7] Erich, F. M. A., Amrit, C., & Daneva, M. (2017). A qualitative study of DevOps usage in practice. *Journal of software: Evolution and Process, 29*(6). doi: https://doi.org/10.1002/smr.1885

[8] Erich, F., Amrit, C., & Daneva, M. (2014). Report: DevOps Literature Review.

[9] Forsgren, N., Humble, J., & Kim, G. (2018). *Accelerate: State of DevOps 2018: Strategies for a New Economy*. Dora: DevOps Research and Assessment. Retrieved from https://services.google.com/fh/files/misc/state-of-devops-2018.pdf

[10] Forsgren, N., & Kersten, M. (2018). DevOps metrics. *Communications of the ACM, 61*(4), 44-48.

[11] Guckenheimer, S. (2018). What is DevOps?. Retrieved November 7, 2019 from https://docs.microsoft.com/en-us/azure/devops/learn/what-is-devops

[12] Hemon, A., Lyonnet, B., Rowe, F., & Fitzgerald, B. (2019). From Agile to DevOps: Smart Skills and Collaborations. *Information Systems Frontiers*, 1-19.

[13] Leite, L., Rocha, C., Kon, F., Milojicic, D., & Meirelles, P. (2019). A Survey of DevOps Concepts and Challenges.

[14] Luz, W., Pinto, G., & Bonifácio, R. (2019). Adopting DevOps in the real world: A theory, a model, and a case study. *The Journal of Systems & Software, 157*. doi: https://doi.org/10.1016/j.jss.2019.07.083

[15] Lwakatare L.E., Kuvaja P., Oivo M. (2015) Dimensions of DevOps. Agile Processes in Software Engineering and Extreme Programming. *Lecture Notes in Business Information Processing, 212*, 212 - 217.

[16] Lwakatare, L., Kilamo, T., Karvonen, T., Sauvola, T., Heikkilä, V., Itkonen, J., . . . Lassenius, C. (2019). DevOps in practice: A multiple case study of five companies. *Information and Software Technology, 114*, 217-230.

[17] Mueller, E. (2010). What Is DevOps? The Agile Admin. Retrieved from https://theagileadmin.com/what-is-devops/

[18] Pittet, S. (n.d.). Continuous integration vs. continuous delivery vs. continuous deployment. Retrieved November 7, 2019 from https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment.

[19] Smeds J., Nybom K., Porres I. (2015) DevOps: A Definition and Perceived Adoption Impediments. *Agile Processes in Software Engineering and Extreme Programming. Lecture Notes in Business Information Processing, 212*, 166 - 177.

[20] Virmani, M. (2015). Understanding DevOps & bridging the gap from continuous integration to continuous delivery. *Fifth International Conference on the Innovative Computing Technology (INTECH 2015)*, 78 - 82. doi: https://doi.org/10.1109/INTECH.2015.7173368

[21] Watts, S., & Kidd, C. (2017). DevOps vs Agile: What's the Difference and How Are They Related? Retrieved from https://www.bmc.com/blogs/devops-vs-agile-whats-the-difference-and-how-are-theyrelated/

[22] Woodburn, D. (2016). DevOps: More than just a passing fad? *Computer Reseller News*, 15 - 17.