A Complete Guide to DevOps Best Practices

Justin Onyarin Ogala
Department of Computer Science, Faculty of Computing,
University of Delta, Agbor, Delta State, Nigeria
justinoo2001@gmail.com

Abstract— **DevOps is a systematic methodology, tools, and philosophical framework for automating and integrating software development and IT operations teams. Team empowerment, cross-team communication and cooperation, and technological automation are all emphasized. DevOps refers to a collection of integrated activities or procedures used to automate and interconnect software development processes with IT developers with the goal of swiftly and reliably producing, testing, and delivering deliverables. DevOps has resulted in developers or practitioners using endless loops to highlight the link between development lifecycle phases regularly. Even though the numerous activities or processes in a DevOps build a loop and flow sequentially, iteration requires that the flow be collaborative and repeated at all times in order to improve the entire lifecycle. Various software automation trends might be equipped to manage the industry's current software and technology if DevOps investigates them thoroughly.**

Index Terms— **DevOps, CI/CD, Software development practices,**

I. Introduction

DevOps refers to a collection of integrated activities or procedures used to automate and interconnect software development processes with IT, developers, with the goal of swiftly and reliably producing, testing, and delivering deliverables. DevOps is a word that relates to both development and operations, and it reflects a cultural relationship between developers and operators whose functions were previously separated (Whittle, 2014). DevOps has resulted in developers or practitioners using endless loops to highlight the link between development lifecycle phases regularly. Although the many activities or processes in a DevOps make a loop and flow sequentially, the iteration suggests that the flow must be continually collaborative and repeated to enhance the overall lifecycle (Gruver, 2016). Communication was a key factor in the development of DevOps. Other developers and designers saw the process as a collaborative way for developers and engineers to use iterative automation to complete jobs and other activities as quickly as possible.

The integration of the development and operations teams was aided by several reasons. The notion was born out of the organizations' long-standing challenges and concerns, since their tasks are greatly scattered, making their operations more complex (Armstrong, 2016). The capacity to tackle difficulties from an individual position got less as the complexity chain grew more complicated. Providing virtual machines in a large network region, difficult setup of extended network devices and servers, and installing numerous applications are only a few of the essential interventions that led to the merger (Armstrong, 2016). Furthermore, log collecting and aggregation, monitoring services, network performance monitoring, and application performance monitoring had all become complicated issues. Furthermore, the developers, operators, and engineers were unable to comprehend the events that led to these complexities, resulting in false alarms and the need for remediation.

DevOps Best Practices

i. Combination of Strength

When a diverse collection of operators and developers worked, they came up with a reasonable solution to the real-world operating complexity (Gruver, 2016). These complicated processes have been broken down into manageable chunks for simple comprehension. Monitoring, performance management, deployment automation, infrastructure automation, log management, and configuration management are among them.

ii. Infrastructure automation

Unlike in the past, when infrastructural components had to be built from the ground up, now there is no need to build servers from the ground up, acquire electricity, or manually link data centers. In addition, manually connecting a machine to a network is no longer necessary. Most of these procedural and time-consuming processes have become ineffective and inessential when the developers and operations teams were combined. Previously, these processes were laborious and tedious, and they only happened after an operation had run into problems (Gruver, 2016). All operations may be carried out, controlled, and maintained in a single location utilizing a simple code, according to DevOps. The emergence of cloud services allows users to access real-time information via a web application. This eliminates the requirement for data centers or silos to perform normal operations. Most processes, such as IT infrastructure, became automated once DevOps was introduced, eliminating the need to physically visit data centers and silos to receive hardware services or make network changes. Cloud services are extremely advantageous since they lower linear demand costs and allow for automatic provisioning without having to pay for hardware services (Meena, 2014). These services are provided by a variety of suppliers. Azure, Heroku, Ubuntu cloud, Amazon web services, HP cloud, RackSpace cloud, and EngineYard are some of the options.

iii. Configuration management

Previously, when the hardware was installed, developers or network users had to manually implement, execute, and configure numerous packages. DevOps brings the power of intelligent or automated configuration management to bear on the problem. DevOps automates the installation and setup of hardware packages through the use of simple scripted scripts. Because servers are constantly and in real-time deployed in every event, computerized setup solutions are advantageous. Assume a user has to make changes or adjustments to many machines. In that situation, all users have to do is execute them on a single system, and they will automatically duplicate over all other devices (Meena, 2014). Puppet, Ansible, Pallet, Chef, Salt Stack, and Bcfg2 are some of the suppliers providing DevOps tools. The automated tools are also provided by Amazon's OpsWorks and Rightscale providers.

Previously, most operational settings had rigorous standards and limitations on who had access to the production field, who could make changes, and when these changes may be implemented. Physical engagement with hardware, particularly in data centers, was required to bring about these improvements (Meena, 2014). These expectations and unrelated procedures have hampered the capacity of development and operations to work together. However, the application and maintenance of various DevOps methodologies and stand-alone operations are still unclear. Before deciding on the optimal strategy to deploy models for infrastructure development, one must first consider the operating viewpoint, which is critical. Furthermore, when deploying an application in a pre-production region, the Ops teams of developers must comprehend the development perspective. (2014, Google)

iv. Continuous Monitoring

Initially, the developers and operations team had to physically manage or control all of the systems in a network to guarantee that there were no defects or possible threats. These operations were inefficient and ineffective. The designs are automated using DevOps to guarantee that the systems' essential notifications are never missed by the operators, organizations, or commercial organizations. Monitoring, messaging, and ticketing tools are all firmly interwoven into these platforms. Filtered noise is used in these techniques to boost signals (Google, 2014). Unlike previous models, which issued alerts to a single machine or unit, DevOps systems convey signals to numerous channels and provide the required steps to remedy issues, whether potential or ongoing.

Aside from issuing alarms, DevOps-enabled solutions have simplified and streamlined on-call administration. Within a single user interface, the designer may create and modify schedules as well as specify escalation policies. The operation team will know who is on the line and who is responsible in the event of an emergency. The critical alarms are always acknowledged when they emerge in every incident. Furthermore, the monitoring systems provide for comprehensive reporting and analytics (Veritis, 2016). The automated algorithms can identify areas of success as well as places where improvements can be made. All warnings and foreign incident-related occurrences are tracked by the systems. The plans also include strong reporting and analytical patterns for revealing the source of the alert, as well as the team's performance in terms of risk identification and resolution. It also manages how workload is distributed across multiple channels in a network.

v. Log Management

Primary features such as infrastructure monitoring play a key part in the running of production applications, regardless of how quickly businesses adopt new technology advancements. The observability idea is vital for deployment success with minimal disruptions in every stage of operation and a steady delivery pipeline as enterprises and other organizations gain traction (Kuchler, 2016).

Developers and operations teams may monitor and understand the application's behavior before releasing it to production using log and event management, which is part of the delivery pipeline's entire process. Although some may claim that log management adds to the development process, work, and time, it aids in the growth of the organization's important software development activities while avoiding difficulties that may be avoided throughout production. It may also aid in the creation of a fluid, seamless user interaction stage, reducing the need to re-architect production solutions (Kuchler, 2016).

vi. Continuous Testing

The significant requirement for log management inside highly dispersed systems has risen as the technological environment in numerous industries has evolved. There have been significant advancements in the creation of apps and services, as well as the ability to deploy applications across numerous logs as a service vendor and the pervasiveness of containers. In terms of service capacity building with diverse languages, the demand for data gathering, monitoring, and tracking across a linked network has risen dramatically (Bass et al., 2015). The log management procedure used to be a time-consuming and arduous task that entailed executing search commands on all local servers. In instances when the server clusters were restricted to 20 servers and individual operators were required, the system had limited scalability. With DevOps, there has been an increase in the number of virtualized systems with minimal operational expenses inside a normal business firm. With the rising virtualization of systems under DevOps, independent developers now can generate designs in parallel while using the best-suited technology for production. These events take place in a shared environment for construction, production, and staging. Log aggregation eliminates customization difficulties when developers use several technologies with different log formats. (Bass et al., 2015).

vii. Observability

DevOps has developed some rules for data monitoring from many perspectives to forecast a system's correctness and stability. Observability is based on three interconnected ideas that help developers and operations teams operate more efficiently. The first component is external monitoring, which looks at external run-ups to verify that web applications provide consumers with a contemporary experience (Bass et al., 2015).
The metrics and distributed tracings, on the other hand, identify linkages between distributed applications throughout a system. It also aids in the detection of errors and exceptions from an application, as well as the necessary resolution procedures. Finally, the events and logs aid in the provision of context information from invested data in the given situation. When combined with other concepts, it also makes it possible to track out bugs in a program.

II. Literature Review

As technology evolves and assumes new perspectives, DevOps has been characterized and marketed as the next transformation drive in the IT profession. DevOps is a methodology that is based on the facts of a particular company and industrial application (Contributor, 2014). DevOps is in demand across all industries because of its capacity to foster collaboration. DevOps has gained a lot of traction in the manufacturing industry, which is a large field of applicability. DevOps fits into space since the industry is surrounded by interconnected processes and activities. Furthermore, the manufacturing market has been steadily growing due to rapid delivery and innovation, which has fueled integrated development and operations.

DevOps requires enhanced cooperation and simplified communication, which are the cornerstones and foundations of manufacturing (Veritis, 2016). DevOps is booming in a variety of sectors, and the manufacturing industry is benefiting from three of them: innovation, automation, and collaboration. The combination of the three elements resulted in improved communication and output in the sector. Furthermore, the teams work better together toward the same goal, resulting in faster product delivery and less marketing time (Humble and Farley, 2011).

A. Collaboration

According to studies, a lack of teamwork has led to the downfall of numerous businesses, particularly in the automobile sector. Collaboration is an organization's cultural intuition, and every entity looks forward to achieving such levels with ease. However, the process is not as simple as some may believe, which is where DevOps comes into play (Veritis, 2016). DevOps is a term that refers to the integration of a company's development and operations teams. The majority of the automobile manufacturing businesses rely on silos to keep track of the jobs they need to complete. DevOps, on the other hand, helps to bridge the gap between the Ops and Dev teams by making them aware of their roles (Bird, 2016). As they are tied up with testing and preparing the items for manufacturing, this results in a quality improvement at every stage of development. SIEMEN is the greatest example of a manufacturing company that has successfully tapped into good teamwork.

B. Automation

Automation is the use of technology to complete tasks with less human intervention. Automation aids in the speeding up of processes and scaling of environments, as well as the building of CI/CD workflows (continuous integration, continuous delivery, and continuous deployment). The majority of machinery is automated to guarantee that things are produced and delivered quickly to fulfill customer needs (Bird, 2016). The automation process, on the other hand, is not addressed. Automation has aided the industrial industry is gaining momentum on previously provided solutions to time-consuming tasks. Manufacturers, on the other hand, should embrace DevOps to achieve greater robustness in this area. To see success with DevOps, industries must follow the four installation steps, which include reducing inefficiency, testing, deploying, and operating. Enhanced automation is a business that has taken advantage of these automation tools. Inefficiency steering and testing have been used by the industry to shift their operations from a procedural to a product-based setting. As a result of the optimized development, the production has altered (Humble and Farley, 2011). To speed up the development process, they've used automated deployment and operation as a driving factor. Finally, this has facilitated openness, ease of deployment, and rapid market release of deliverables.

C. Innovation

To lessen the pressure that comes with development cycles, industry have assured that quality and output levels are optimal. As a result, an environment receptive to creativity has emerged. Innovativeness has given the automotive industry a competitive advantage in the market. DevOps has aided industries in achieving the maximum degree of innovation (Humble and Farley, 2011). DevOps helps businesses to minimize the amount of time spent on production or activity execution, allowing them to make more informed decisions by transferring processes to teams.

The approach of traditional developers differs in several ways, one of which is based on coding. Initially, industrial developers used a variety of application features and combined them at the end of the coding process. When bugs and flaws appear, the developers must complete the entire work to correct them. As a result, there have been unnecessary delays in manually detecting bugs through integration. Through CI/CD phases, DevOps has brought a significant shift in the sector, where developers release updated code at a comparatively faster pace and with a high frequency. Developers can sample and cross-check code in tiny chunks and do a continuous compilation with such activities (Bob, 2016). For frequent code checks, the testing, security, and UI components are typically automated, and it's time to go to the repository. Naturally, this leads to earlier problem identification and faster bug resolution, increasing developer productivity. The supply of deliverables becomes continuous after the deployments have been handled through pipeline release.

III. The DevOps Life Cycle

The DevOps Lifecycle is a cycle of continuous improvement and self-evaluation that ensures tasks are completed as fast and effectively as possible, both in terms of developing a quality product and maintaining and upgrading it. DevOps brings together the development and operations teams to achieve optimal efficiency. The idea of integrating development and operations personnel helps DevOps achieve its aims through enhancing communication among all stakeholders from planning to delivery, as well as automating the delivery process in a variety of methods. This will benefit organizations:

- ➢ Increase the frequency of deployments
- ➢ Reduce the time it takes to get a product to the market.
- ➢ New releases have a lower failure rate.
- ➢ Reduce the time between fixes.
- ➢ Increase the average time to recovery.

This is performed by following and implementing a simple lifecycle of activities:

- ➢ Continuous Development
- ➢ Continuous Integration
- ➢ Continuous Testing
- ➢ Continuous Monitoring
- ➢ Virtualization and Containerization

DevOps promotes a culture of continuous everything. As a result, there is a considerably shorter iterative cycle of continuous improvement with much shorter update cycles, which reduces time to market while making it much easier to find and rectify flaws. While you repeat the cycle, you'll have a good method to follow when implementing your DevOps endeavor. Each team should be well-equipped with open tools and guidelines before beginning work at any level. The tools can be tailored to the developer's needs and objectives (Bob, 2016). As a result of the technique, high-quality, dependable software can be produced quickly. The cycle is as follows as shown in figure 1:

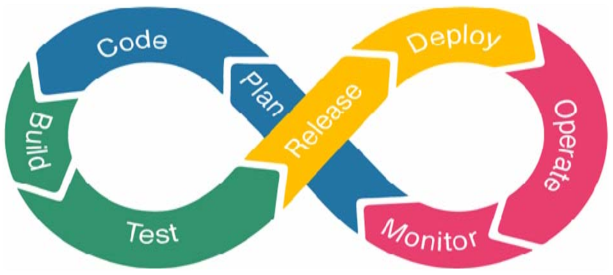

Figure 1: The DevOps Life Cycle

Planning, the building, integrating and deploying, monitoring, operating, and responding through feedback passage are the lifecycles.

a. Planning

A problem statement and a scope description are used to specify the needed resources that will be used during the project. Second, the developers give an overview of the most recent systems and their goals. Finally, they analyze the study's practicability and feasibility to calculate how long it will take to complete. Fourth, the team thinks about the system's possible dangers, hazards, restrictions, security, and integrations. Finally, they create a project-wide feasibility study (Tutorialspoint, 2012).

b. Building

The developer creates system designs and converts the SRS document into a logical component with full requirements for coding execution once the initial stage of project planning is completed and authorized. The construction of contingencies, team training, and maintenance via an operating plan comes next. The papers are sent to the implementation level after the design is complete. Coding with a certain source code and programming language is required for implementation. For error and defect detection, the system elements are coupled in a stressful environment (Tutorialspoint, 2012). After the system has been integrated, a test report is created by allocating resources.

c. Monitoring

DevOps chips in during this phase, primarily throughout the maintenance and support lifecycle, to detect the many problems and bugs in the system. Traditionally, maintenance tasks were carried out under the ongoing observation of users or developers. They made improvements that allowed software to go through a defined time of testing or put in place reasonable criteria once a system was up and running. After the testing step, the method tackles lingering errors and resolves them manually. Today, DevOps steps in to solve the problem by developing simple scripts that can virtually monitor the system, detect external components, and raise alarms. The alerts are triggered and addressed automatically, or they recommend that the developers and operations team take the appropriate activities to resolve them. As a result, the maintenance and support procedure is automated and does not require physical interaction. This allows the engineers to spend their efforts on more productive tasks (Crispin and Gregory, 2015).

d. DevOps Process Flow

The teams have had moments since the introduction of DevOps into the development cycle. Because DevOps is the future of IT operations, it's vital to understand the many procedures that create flow and the best way to apply it. DevOps' various principles serve as a practical and cultural basis for businesses. While developers focus their efforts on the fundamental concepts of automation, collaboration, and iteration, they also strive to enhance the system regularly (Duffy, 2015). The developers and operations team test the system regularly and learn from their previous code issues and failures. They act quickly on system feedback and recommendations to improve their level of performance. This reduces the amount of time, money, and effort required for deployment.

These concepts have been extended to include actions that focus on automation and tool supply for rapid deployment in lean-agile frameworks. By adopting the agile approach, which focuses on software integration, iteration, delivery, and deployment, the automation process has enabled operators and other IT professionals to put their attention into smooth procedures. These are only a few of the actions that add to the cooperation process during the pipeline's development.

e. DevOps Benefits

The industrial platform has benefited greatly from DevOps. Most sectors and corporations, notably in the United States, have discovered the benefits of DevOps via the fast adoption of methodology and technology in their development cycles For example, Enhanced Automation, located in the United States, has eliminated the usual difficulties associated with sluggish IT operations (Opsgenie, 2015). Through agile advantages, the company has solely focused on automation, collaboration, and flexibility. These companies have gained the following advantages: first, they spend less time selling their products or services since they are up-to-date and answering real-time problems. As a result, they can provide their components to the market at a faster rate.

Secondly, this quality of the return on investment has vastly improved with time. Finally, customers have expressed high levels of pleasure as a result of receiving services and products in real-time as a result of speedier delivery (Opsgenie, 2015). Fourth, excellent operational efficiency has been achieved through reduced operating time and little expense and effort. This is made possible via automation. Fifth, improved cooperation has opened a space for developers and IT operators to use simple codes to tackle current challenges. Finally, these codes aid in the rapid discovery and correction of faults and issues. These development teams anticipate changes in processes and culture in real-time, reducing the risk of misinformation and process misalignment. Efficiency and product quality improve with clear and consistent communication. Regular integration, deployment, and testing also aid in the creation of faster processes and the discovery of errors.

f. DevOps Implementation

The process of implementing DevOps in a business appears to be cumbersome and intimidating at first. This is because the operation necessitates both procedural and cultural changes. When a company decides to embrace DevOps, it must examine the steps that will be implemented gradually (Bass et al., 2015). One by one, the phases should be implemented. Depending on an organization's existing position, management may consider using the agile technique. The following are the phases in the sequential implementation process: initially, the company must adopt an agile development model. Second, think about moving their infrastructure, software, and platform to the cloud. This entails moving away from the conventional reliance on cloud-based service delivery (Bass et al., 2015). It also eliminates the need for data centers and silos because the organization's processes are available in real-time. Finally, the organization's activities must be adapted to the CI/CD pipeline approach. Fourth, they should automate their software deployment process to eliminate human contacts and improve market product delivery speed. Finally, they should automate software testing to detect faults and problems in code. Finally, after installation, they must assure continuous deployment.

Organizations must be aware that automated DevOps necessitates changes in tooling and infrastructure. If the organization lacks the necessary tools and infrastructure to enable the DevOps process flow, it risks generating gaps. As a result, a business must use an agile approach to integrate and automate all stages of the DevOps process to create a legitimate DevOps process (Bass et al., 2015). Additionally, a company may consider using graphics to establish DevOps procedures, staff, training, and process deadlines. This aids in the implementation process by ensuring that everyone is on the same page.

IV. DevOps Tools

Various tools must be implemented into the IT field's development and operating procedures in terms of thinking. As a result, while implementing DevOps, the company must consider different devices. Git is a basic solution based on distributed source code that allows developers and operators to follow the progress of their projects. Other components like workflow choices, staging regions, and branching environments can be used to conveniently travel back and forth between distinct laws. Lucidchart is a visual tool that helps developers and IT professionals design quick-to-implement yet easy-to-understand procedures and data. Furthermore, during the beginning stages of DevOps development, the team must be taught and given documentation to track the development progress and keep all members informed (Lucidchart, 2015).

Containerization has often switched to the grouping of objects into different units and clusters for automation and delivery (Kar, 2015). Kubernetes is the most widely used DevOps tool and open-source solution for assisting developers in moving their projects to the next level. These are only a few of the tools used in the DevOps development process, which includes many others. Nonetheless, tool management remains a challenge for IT engineers. Lucidchart resolves any challenges that arise throughout the development process, regardless of the device used in the application.

V. Future

Various software automation trends might be equipped to manage the industry's current software and technology if adequately investigated by DevOps. Industry machines will become autonomous as a result of new technologies and automated software. The key challenge with autonomous equipment is DevOps' capacity to persuade executives, operators, and drivers that these self-driving robots are accurate and safe to ride or run (Abouzaid, 2016). Nonetheless, the whole DevOps process will be modernized in the future through extensive automation, cooperation, and integration to accelerate rapid delivery.

VI. Conclusion

Before understanding the meaning and purpose of DevOps, one must first understand their industry and position. Some systems need a high level of automation and support, while others do not necessitate complicated processes. These methods, on the other hand, necessitate application and requirement alignment, rapid development, and novel innovation testing. DevOps can assist release small batches for market effectiveness and efficiency after the industry has a consistent resource to the market.

Author Details:
Ogala Justin Onyarin

Address all correspondence to:
justinoo2001@gmail.com
Ogala Justin Onyarin
Department of Computer Science, Faculty of Computing, University of Delta, Agbor, Delta State, Nigeria

References

[1] AbouZaid, A. (2016). Continuous delivery and maturity model: DevOps. Retrieved September 18, 2019, from http://tech.aabouzaid.com/2016/01/continuous-delivery-and-maturity-model.html

[2] Armstrong, S. (2016). *DevOps for networking: Boost your organization's growth by incorporating networking in the DevOps culture*. Birmingham, UK: Packt Publishing.

[3] Atlassian. (2015) DevOps: Breaking the Development Operation Barriers. Retrieved February 26, 2021, from https://www.atlassian.com/devops\

[4] Bass, L., Weber, I., & Zhu, L. (2015). *DevOps: A software architect's perspective*. Old Tappan, NJ: Pearson Education.

[5] Bird, J. (2016). *DevOps sec: Delivering secure software through continuous delivery*. Sebastopol, CA: O'Reilly Media.

[6] Bob, R. (2014) The DevOps Life Cycle. Retrieved February 26, 2021, from https://devops.com/the

[7] Contributor. (2014) DevOps with purpose: It's about your application--retrieved February 26, 2021, from https://devops.com/devops-with-a-purpose

[8] Crispin, L., & Gregory, J. (2015). *More agile testing: Learning journeys for the whole team*. Boston, MA: Addison-Wesley.

[9] Duffy, M. (2015). *DevOps automation cookbook: Over 120 recipes covering essential automation techniques*. Birmingham, UK: Packt Publishing.

[10] Google. (2014) 2014 State of DevOps Report. Retrieved February 26, 2021, from https://services.google.com/fh/files/misc/state-of-devops-2014.pdf

[11] Gruver, G. (2016). *Start and scaling DevOps in the enterprise*. Pennsauken, NJ: BookBaby.

[12] Humble, J., & Farley, D. (2011). *Continuous delivery*. Boston, MA: Pearson Education.

[13] Kar, S. (2015) Best 2014 DevOps tools and trends that define DevOps' future in 2015. Retrieved February 26, 2021, from https://siliconangle.com/2015/01/13/best-2014-devops-tools-and-trends-that

[14] Lucidchart. (2015) Understanding the DevOps process flow. Retrieved February 26, 2021, from https://www.lucidchart.com/blog/devops-processJ.

[15] Meena, S. (2014) Configuration Management. Retrieved February 26, 2021, from https://www.slideshare.net/sahilsk/configuration-management-stackexpress-20140610J.

[16] Microsoft. (2013) 32 principles and practices of successful continuous Integration, Continuous Delivery, and DevOps. Retrieved February 26, 2021, from http://davidfrico.com/devops-principles.pdf

[17] Opsgenie. (2015) Modern incident management for operating always-on services. Retrieved February 26, 2021, from https://www.atlassian.com/software/opsgenie/what-is-opsgenie?

[18] Tutorialspoint. (2012) System Development Life Cycle. Retrieved February 26, 2021, from https://www.tutorialspoint.com/system_analysis_and_design/system_analysis_and_design_development_life_cycle.htm

[19] Veritis. (2016) DevOps Implementation in Manufacturing Sector: Meet the Culture-Driven Approach. Retrieved February 26, 2021, from https://www.veritis.com/blog/devops-implementation-in

[20] Whittle, D. (2014) An Introduction to DevOps. Retrieved February 26, 2021, from https://devops.com/introductiontodevops