International Journal of Computer Applications (0975 – 8887)
Volume 185 – No. 30, August 2023

Application Development Feasibility: DevOps or SRE?

Olumide Bashiru Abiola
Beechnet Solutions Limited
2967 Dundas Street West, #724D,
Toronto, Ontario M6P 1Z2,
Canada

Olusola Gbenga Olufemi
Hood College
401 Rosemont Ave
Frederick, MD, U.S.

ABSTRACT

DevOps Engineers are operation-focused professionals who solve development pipeline problems. In contrast, SREs are development-focused engineers that resolve operational, scalability, and reliability problems. That is, DevOps primarily focuses on core development activities while SRE is primarily concerned with implementing and maintaining the core infrastructure. Can this be explained better? Of course, it can. When it comes to the distinction between DevOps and SRE, it is important to note that these are distinct functions that have evolved through various phases over the years. Each has its own responsibilities, DevOps refers to Development Operations, while SRE stands for Site Reliability Engineering [3]. It is therefore of utmost importance to know how each contributes to software development since organizations will want to utilize their potential in deriving the maximum value during application development. A lot has been discussed in the public domain about what DevOps and SRE are and what they are not, which calls for thorough exploration and research into what they truly are. The insight from this work should streamline knowledge of how software development feasibility can be enhanced by DevOps or SRE practices or both, in line with the frequently changing application development tooling.

General Terms

Agile, automation, application, software development, project, features, operations.

Keywords

DevOps, SRE, feasibility, reliability, pipeline.

1. INTRODUCTION

IT teams in several organizations have imbibed DevOps and SRE as primary strategies for building modern applications. Hence, it is important to distinguish the difference between these two functions, more importantly, their applications to software development projects [3]. DevOps teams focus mainly on core development, working on products or applications that provide solutions to a specific problem. DevOps teams use agile software development methodology to build, test, deploy, and monitor applications with speed, quality, and control. SREs on the other hand work on the implementation of the core, they consistently give feedback to the core development group. SRE makes use of operations data and software engineering to automate IT operation tasks and accelerate software delivery while bringing associated IT risks to the barest minimum. As observed in the current technology landscape, software development has progressively become broader, and more complex, hence requiring high-quality approaches like DevOps and SRE [15].

2. DEVOPS & SRE COMPARISON

As DevOps focus on streamlining changes, SREs ensure that these changes do not add to or increase the total failure rates. Hence, they are different. DevOps aids in automating speed, while SRE helps with automating reliability [2]. DevOps processes from left to right along the development life cycle, by applying automation to speed up new capabilities rollout which usually gets measured by deployment frequency and lead time for a visualized change. Whereas, for SRE, there is a move from right to left of the development life cycle which involves production-level requirements in development with a focus on limiting failure rates and lessening the required time to restore services [2]. DevOps and SRE have something in common regarding - Service Level Objectives *(*SLOs), both align with SLOs to support business goals. The role of DevOps is to expand the user base and improve the end-user experience while the objective of SRE personnel is to contribute to business success by offering the right features at the right time to facilitate adaptation to change. SLOs are means to unite DevOps and SREs [2]. The overlap between SRE and DevOps can clearly be seen here since both goals tend to measure success or failure, by gaining continuous reliability on all application development [1].

3. DEVOPS – THE BELIEFS

DevOps as a culture delivers software swiftly with reduced errors, DevOps improves quality and delivery time by removing human error. DevOps is simply practicing building software iteratively through linking operations with development [3]. Therefore, DevOps assists operations teams in delivering automated infrastructure with less engineering, enabling developers to create reliable and predictable environments with minimal bottlenecks [1]. Implementing DevOps methodology is worth the transformation effort because it drives: (1) better business values (2) faster delivery time (3) silos removal (4) improved customer experience (5) early problem detection, and (6) unleashing innovations [11]. The study of DevOps is so important that it is being made mandatory in software engineering research [13]. However, software developers are impacted in one way or the other in their turnover by DevOps practices throughout the software development life cycle of any application [14].

3.1 DevOps Benefits

DevOps application mindset and skills for software reliability can reduce silos between development and operations teams because there is shared responsibility in detecting reliability and performance issues early in the development life cycle [2]. Other benefits include [1]:

3.1.1 Better products are delivered faster.
3.1.2 Issue resolution is fast and made less complex.
3.1.3 Scalability and availability are greater.
3.1.4 Operating environments are made more stable.
3.1.5 Resource utilization is made better.
3.1.6 Automation is made more efficient.
3.1.7 Visibility into system outcomes is made greater.

3.1.8 Innovation is made greater.

3.2 DevOps Metrics

Benchmarking DevOps efforts and outcomes is a significant undertaking. However, the DevOps team can use some performance indicators (metrics) to measure and gauge the effectiveness of its approach. Below are some important DevOps metrics:

3.2.1 Application availability.
3.2.2 Traffic and application usage.
3.2.3 Number of tickets.
3.2.4 Commit count.
3.2.5 Number of tests conducted.
3.2.6 Deployment Rate.

Metrics such as Deployment Speed, Deployment rollbacks/fail frequency, Version lead time, and Rate of response to tickets (MTTR) should also be considered. However, having a clear objective for each of the DevOps metrics mentioned above will help set the team in the right direction towards achieving an optimized DevOps architecture.

4. SRE – THE PERCEPTION

The SRE philosophy has been evolving since 2003, establishing its presence prior to the emergence of DevOps. Ben Treynor made the term popular when he created Google's Site Reliability Team [1]. SRE focuses on improving software system reliability across availability, performance, latency, efficiency, capacity, and incident response [2]. SRE embodies the ideology of optimizing the reliability and overall quality of software developed [2]. Site reliability problems cannot always be completely solved. The continuous demand for new services and applications combined with evolving enterprise requirements means that there will always be work for SREs, and as such, there will always be room for improvement [2].

4.1 SRE Benefits

SRE strives to reduce duplication or redundancy of effort as much as possible. SREs focus is on automating manual tasks, like provisioning access and infrastructure, accounts setting up, and self-service tool building. With this, the development teams can focus on delivering features, while the operations teams can focus on managing infrastructure [2]. The following are some benefits of SRE:

4.1.1 Resiliency-based engineering is what SREs drive. SREs provide guidance and make sure that they give resiliency the highest priority for developers and operations, therefore narrowing the gap between Dev and Ops [2].

4.1.2 SREs imbibe reliability principles from Dev to Ops, inputting reliability and resiliency into every process, application, and code change to enhance software quality that ends up in production [2]. Taking early, proactive steps to ensure quality and reliability are built-in from the beginning is an SRE objective. SRE can streamline processes and promote managing testing across the enterprise in support of CI/CD practices [2]

4.1.3 SRE promotes higher change rates while maintaining resiliency and aims to get 99.999% uptime. In multi-cloud environments, resiliency measurement is done across multiple key metrics such as performance, user experience, responsiveness, and conversion rates. SREs build and implement services that improve operations and accelerate the release process across all areas to achieve their objectives. These include adjusting monitoring and alerting for making code changes in production. Building custom tooling from scratch to meet specific needs in the software delivery or incident management workflow is often done by SREs [2].

4.1.4 SRE ensures that a lot of the changes made do not break the systems [2].

4.1.5 Caring about every process from source code to deployment is what SRE does, earning the accolade of being a true bridge from development to operations [1].

4.2 SRE Metrics

SREs focus principally on the following five metrics **[7]:**

4.2.1 **V**: Volume - Current number of requests, drops, spikes.

4.2.2 **A**: Availability - Are all services up and running?

4.2.3 **L**: Latency - Are service response times within the expected range?

4.2.4 **E**: Errors - What errors are occurring, and why?

4.2.5 **T**: Tickets - What are the complaints of the users?

5. DEVOPS & SRE ROLES

5.1 SREs Usually:

5.1.1 Spend more time programming as compared to DevOps engineers.

5.1.2 Ensure that binaries and configurations are applicable for integration and deployment in different environments.

5.1.3 Write code and manage configurations for automation.

5.1.4 Monitor software infrastructure, track and solve tickets to resolve problems.

5.1.5 Must plan software deployments with immutable infrastructure using CI/CD.

5.2 DevOps Engineers Focus On:

5.2.1 Ensuring software development and deployment are as easy as possible for the development team.

5.2.2 Spending time with tools like Jenkins, Kubernetes, and Docker to automate software builds, tests, and deployments aligned with CI/CD priorities.

5.2.3 Configuring, supporting, and documenting infrastructural components.

5.2.4 Developing workflows to enable CI/CD for projects.

5.2.5 Setting up and maintaining various virtual environments (VMs, Containers).

5.2.6 Implementing and maintaining cluster environments.

Table 1. Comparison - DevOps & SRE [3]

Parameters	DevOps	SRE
Works with	Product development team	Operations Team
Focus	The development side of product management	The operations side of product management
Approach	Streamlining development and deployment processes, reducing risk, and increasing speed	Treats production environment as a highly available service
Use case	Applied in agile software development projects	Used with lean infrastructure practices
Goals	Aims to improve communication in the	Aims to create systems that a

	entire lifecycle from ideation through deployment.	small number of skilled engineers can easily maintain
Tools Used	Use automation tools like Puppet or Chef to ensure consistency across environments.	Use scripting languages like Python or Bash

5.3 DevOps & SRE Commonalities

Table 1 gives more insight into commonalities. Both DevOps Engineers and SREs share a common belief that embracing change is necessary for driving continuous improvement in application development. It is worth noting that no software remains the same always, likewise, no system can be idle or remain unchanged forever. DevOps and SRE both have a strong focus on working together as a team with shared responsibilities and knowledge, none can work in a silo. Both promote making software changes as small as possible since small units usually merge more smoothly and are easier to roll back when problems arise. DevOps and SRE both advocate a strong preference for automation wherever possible in development [10].

Tools utilized by DevOps and SREs are generally similar and often nearly identical, except for team-specific ones that are created to address their unique needs and responsibilities. Another similarity is the requirement for good measurement and observability. Data, especially good data, is vital to both DevOps and SRE.

6. TITLE & AUTHORS

Olumide Bashiru Abiola - CISSP, CISA, CISM, CRISC | Olumide currently serves as a Project Manager with one of the largest banks in North America and is an independent Cybersecurity Consultant. Olumide draws upon his experience in the financial and technology industry after holding a myriad of other positions in System, Network, and Infrastructure Administration. He is responsible for clients' end-to-end cybersecurity programs, coordinating cybersecurity efforts within the enterprise.

Olusola Gbenga Olufemi – Olusola is extensively skilled in Distributed Systems Implementations, Information System Security and Management. He is vast in Cloud Application & Infrastructure Provisioning. Olusola is an active member of ISACA, PMI, and ACM. He is recognized by elite associates as a lifelong learner and researcher.

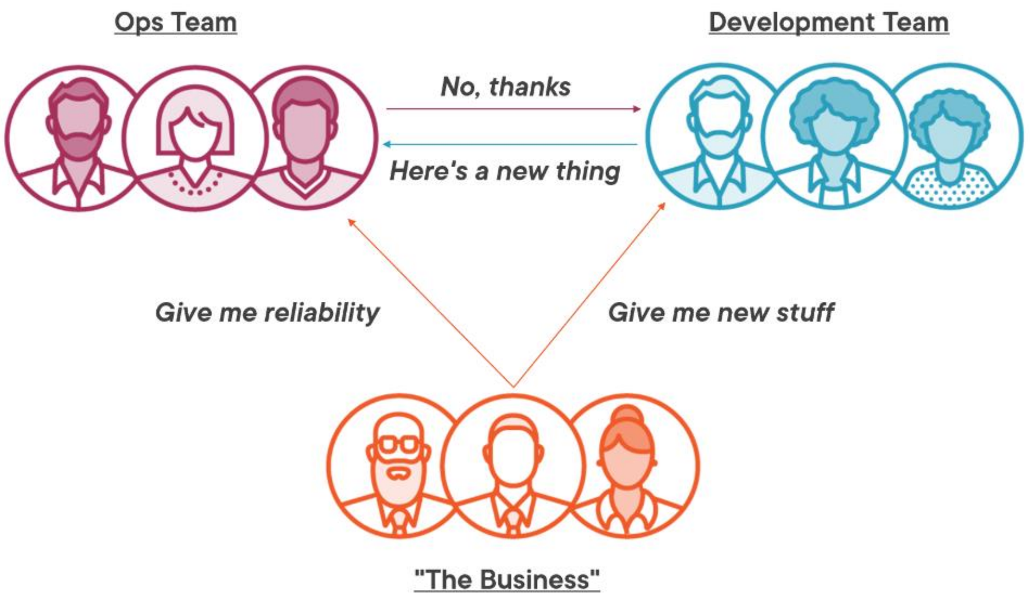

Fig 1: DevOps Transformations [12]

7. DECIDING ON DEVOPS OR SRE

Fig 1 above shows how the business application development sustainability depends on both DevOps and SRE. When an organization prioritizes downtime over uptime, then DevOps will suffice. There is undoubtedly a need for improvement on past software development, deployment, and operations methods [10]. As there are growing complexities associated with containerized microservices running on cloud services, orchestrating and keeping everything working, even with components or services failure, is a major undertaking. Planning for site reliability is vital [10]. If your application or services are expected to be reliable to the level of two or more nines of uptime and availability, then the focus of SRE with its error budgets and SLOs will help remove the politics and guesses from the process. This enables stakeholders to see clearly how to impact the availability and reliability of the system most directly and effectively. DevOps and SRE both aim to enhance the release cycle by helping development and operations transparently see each other's side of the process throughout the application lifecycle. They also promote automation and monitoring, reducing the time from when a developer commits a change to when it is deployed to production. SRE and DevOps aim for this outcome without compromising the quality of the code or the product itself [1]. SRE and DevOps both can narrow the gap between development and operations teams in improving the application development life cycle without jeopardizing quality [4]. DevOps and SRE can coexist since the two practices need each other. Although the work culture is different in both, they still share some foundational values needed in application development [10]. SRE and DevOps ask two different but equally valuable questions:

- DevOps asks **what** needs to be done.
- SRE asks **how** that can be done [1].

7.1 Observability & Incident Response

DevOps and SRE teams can prevent many incidents in the application development cycle, however, no team can prevent all development related production incidents in any organization. Companies that use SRE and think about how long it has been since they had incidents that impacted customers can be observed. The nature of the incident and how quickly it was resolved can also be investigated. Statistics have shown that SRE is the solution when uptime and minimization of incident-related downtime and costs are necessary [10]. For SRE, incident response is concerned with restoring deployment from an unfavorable state back to its expected condition. Nevertheless, the incident response lifecycle involves prevention, discovery, and resolution, aiming at automating as much as possible in each stage [8].

Prevention comes as the first and last step of incident response. In an ideal situation, incidents are prevented first with test-driven development in the CI/CD pipeline. Sadly, deployment sometimes does not execute as planned in production [8]. Discovery means knowing when an incident might have occurred, then offering the right alert to the right channels to respond. When discovery is automated, it gives maximum incident response coverage, also minimizes the mean time to discover (MTTD) and protects the SLOs [8]. Resolution assists with bringing deployment back to its desired state. However, some incidents have automated solutions, for instance, in autoscaling services, more capacity is automatically provisioned when the computing resource is drained or overwhelmed. At the same time, some incidents will need human attention, particularly when the symptoms are unrecognized, or the cause of this incident is unknown. To be precise, observability helps drive the incident response lifecycle with monitoring, alerting, and search [8]. Hence, the collective solution minimizes the mean time to resolution (MTTR) in protecting service reliability and customer loyalty and the end [8].

7.1.1 Observability with Data

An issue cannot be resolved if it can't be observed. Hence, an incident response will require having visibility into the full stack of the affected deployment over time in any development environment.

7.1.1.1 Monitoring, Alerting & Taking Action

Observability assists in automating the incident response lifecycle through monitoring, discovering, and alerting on the Service Level Indicators (SLI) and SLOs that are most important.

When SLIs and SLOs are known, they can be defined as alerts and actions which serve as the right data to share with the right people whenever there is a breach in SLO. With an elastic alert, queries are scheduled that trigger actions when the results meet some set conditions. The conditions define metrics (SLIs) and thresholds (SLOs). Actions are what are delivered as messages to one or more channels, to signal the beginning of an incident response process.

7.1.1.2 Investigation & Doing Research

It should be known that incident response is a research problem. Research assists in delivering timely and appropriate answers to questions asked. Research serves as the key to fast incident response. Not just because the approach might be fast with technological involvement, but because the experience might be insightful.

8. CONCLUSION

In conclusion, the collaboration between developers, operations, and product owners enables SREs to define and meet uptime and availability targets [2]. However, SRE and DevOps can coexist. While the two share some foundational values, the focus of their work is different. Even though they share similar tooling and development practices, a big differentiator is that SRE have a strong and deliberate focus on keeping a site up and running [10]. Organizations at times create wider DevOps teams in such a way that the SREs collaborate with them or act as a subset of the team. [10]. Teams must also comprehend that these two software engineering practices often work together to create a reliable and secure software product. Hence, finding the better of both methodologies is impossible, as the two constantly overlap each other [9]. Though there are differences in their processes and objectives, SRE and DevOps nevertheless shares core principles that enable the teams to proactively build reliable services, which further leads to greater operational efficiency, business value, and overall happiness for all stakeholders [9]. However, there is a need for further research in DevOps and SRE practices to help software development professionals in the use of these practices to better deliver robust applications.

9. ACKNOWLEDGMENTS

Praise be to God for the blessing and guidance in this journey, and for making the writers impactful in this present information technology sphere. Many thanks also to friends and family members who have been inspirational and supportive all these years in this information technology endeavor.

10. REFERENCES

[1] Netapp, What is site reliability engineering (SRE)? https://www.netapp.com/devops-solutions/what-is-site-reliability-engineering/

[2] Dynatrace, What is SRE (site reliability engineering)? And what do site reliability engineers do? https://www.dynatrace.com/news/blog/what-is-site-reliability-engineering/Fröhlich, B. and Plate, J. 2000. The cubic mouse: a new device for three-dimensional input. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems

[3] Knowledgehut, DevOps vs SRE: Major Differences - https://www.knowledgehut.com/blog/devops/devops-vs-sre.

[4] Veritis, SRE vs DevOps: Which Productivity Approach is Better? https://www.veritis.com/blog/sre-vs-devops-which-productivity-approach-is-better/.

[5] Opsera, SRE vs DevOps: Responsibilities, Differences and Salaries - https://www.opsera.io/learn/sre-vs-devops-responsibilities-differences-salaries

[6] Platora, The 10 Essential DevOps Metrics That Really Matter - https://www.plutora.com/blog/10-essential-devops-metrics-that-really-matter

[7] Performetriks, 5 Key Metrics of Successful Site Reliability Engineers - https://www.performetriks.com/post/5-key-metrics-of-successful-site-reliability-engineers

[8] Elastic, Elastic Observability in SRE and Incident Response - https://www.elastic.co/blog/elastic-observability-sre-incident-response

[9] Professional DevOps, SRE VS DevOps -

https://www.professional-devops.com/sre-vs-devops.html

[10] Gremlin, SRE vs DevOps: CAN THEY COEXIST OR DO THEY COMPETE? https://www.gremlin.com/site-reliability-engineering/sre-vs-devops/

[11] Harrisonclarke, What Emerging Tech Companies Are Looking for in DevOps/SRE Roles - https://www.harrisonclarke.com/blog-2023/what-emerging-tech-companies-are-looking-for-in-devops-sre-roles

[12] All Hands On Tech, SITE RELIABILITY ENGINEERING: COMPARING SRE AND DEVOPS - https://www.allhandsontech.com/it-ops/devops/comparing-sre-and-devops/

[13] Leonardo Leite, Carla Rocha, Fabio Kon, Dejan Milojicic, and Paulo Meirelles. 2019. A Survey of DevOps Concepts and Challenges. ACM Comput. Surv. 52, 6, Article 127 (November 2019), 35 pages. https://doi.org/10.1145/3359981

[14] Liming Zhu, Len Bass, George Champlin-Scharff, DevOps and Its Practices, https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7458765

[15] Alok Mishra, Ziadoon Otaiwi, DevOps and software quality: A systematic mapping, https://www.sciencedirect.com/science/article/pii/S1574013720304081, Volume 38, November 2020, 100308