SRE VS DEVOPS: WHAT'S THE DIFFERENCE?

Google was among the pioneering adopters of the DevOps SDLC methodology. The company scaled its business rapidly, setting an example for organizations pursuing similar growth trajectory by leveraging the DevOps framework.

In recent years, however, Google has expanded its approach to service management, calling it site reliability engineering (SRE). Let's look at the differences between DevOps and SRE.

Understanding DevOps

DevOps is an ITSM framework that defines the mindset, culture, and philosophy of working on IT projects as a collaboration among developers, operations, and QA teams. The collaboration spans the SDLC pipeline with no proverbial walls or silos that produce hard separation between the three roles. Instead of using tools or processes as descriptors, DevOps is best described in terms of its characteristics, both interpersonal and cultural, necessary to achieve:

- Continuous application delivery
- Shorter release cycles
- Reduced waste processes
- Lower expenses
- Reduced friction among the workforce

However, DevOps doesn't outline specific practical guidelines to apply those concepts in real-world environments, partly because DevOps principles are designed to be followed with organizational context and use cases. Organizations have the freedom to optimize tradeoffs in following the DevOps principles. For example, DevOps requires IT teams to accept failure as a normal. According

to DevOps, there's no specific definition of 'failure' and 'normal'. This leaves potential gaps in translating DevOps principles into concrete and practical steps that an IT organization can adopt and replicate Google's DevOps strategies.

What is SRE?

Site reliability engineering (SRE) fuses the software engineering and operations disciplines. SRE professionals spend about half their time in development tasks and half on operations tasks. The SRE role enables collaboration and information sharing between Dev and Ops departments, similar to the DevOps principles but for additional specific objectives.

SRE satisfies DevOps principles

If you're asking which is better, you're asking the wrong question. Instead, consider Google's analogy of DevOps as a programming language interface and SRE as a programming class used to implement DevOps. The philosophy of DevOps may define the overall behavior of a service management framework, with the specific implementation strategy left up to the author. SRE, then, is a prescriptive approach to implement, measure, and achieve DevOps objectives.

There are five key ways SRE complies with DevOps:

1. Reducing organizational silos

SRE treats Ops as a software engineering problem. Engineers are required to spend their efforts in solving issues that were previously thrown across the proverbial wall between Dev and Ops. The shared sense of responsibility leads to practical initiatives such as reducing bugs early during the development sprints and using similar tools between Dev and Ops. These issues may be more focused on the engineering design and applications, instead of business logic problems. For example, the SRE may have competencies to improve application performance and latency instead of managing infrastructure resources. Like DevOps, there is less focus on specific tools used in the Dev or Ops environment, but adequate expertise are available to use similar technologies in either department.

2. Accepting failure as normal

Similar to DevOps, SREs don't pass the blame for failures and production incidents between the IT teams. SRE guidelines encourage radical changes (within limits) that potentially lead to failure. A measured risk budget is allowed for SREs to test these limits and potentially innovate faster. SREs also assume that 100% availability and performance targets are not viable to facilitate growth. Strong collaboration between business and IT is required from an SRE perspective to evaluate optimal targets for service level objectives (SLOs) and service level indicators (SLIs). Any violation funnels a feedback look to the IT teams, targets are re-evaluated and optimized for the changing business and IT circumstances. Blameless postmortem of incidents and failures is mandated as part of the SRE framework.

3. Implementing gradual change

Like DevOps, SRE also encourages continuous improvement through change. SRE requires the

changes to be small and frequent. As a result, any negative repercussions are less impactful and low-risk improvements can be readily tested and implemented. Automated testing of such changes is readily used in the change management strategies, including CI/CD as adopted in the DevOps framework. An objective measurement of change needs to be defined such that the cost of failure reduces at the same time. For example, reducing the mean time to repair (MTTR) for production incidents allows more time for devs to invest in feature improvements. As a result, the product quality improves on a gradual but continuous basis.

4. Leveraging tooling and automation

While DevOps encourages automation and technology adoption, SRE is focused on embracing consistent technologies and information access across the IT teams. Google pioneered this concept by unifying its codebase, albeit not specifically for its SRE framework. SRE requires all teams working on the same service to adopt the same technology solutions. Incompatibility and integration issues between technologies from different vendors, era and use cases can create unnecessary silos even in the DevOps environment. In essence, the adopted tooling also determines the skillset necessary for each particular team and service area. However, SRE is not entirely about using a specific and well-defined set of technologies to fulfil certain IT tasks, but focused on the API orientation of the tooling and how the associated ITSM activities are served.

5. Measuring everything

Both DevOps and SRE encourage measurement. DevOps is primarily focused on the process performance and results achieved with the feedback loop to realize continuous improvement. SRE requires measurement of SLOs as the dominant metrics, since the framework observes Ops problems as software engineering problems. SRE also requires budgeting for toil activities and risk. It is assumed that the definition of 'normal' will evolve on a continuous basis. In the spirit of gradual change and improvements, IT teams must observe some slack that allows them to perform repeatable and automatable tasks manually. Once the ITSM model of the organization matures, these tasks can be automated, redefining the new 'normal' including a lower budget for risk through the improved ITSM capability. Both DevOps and SREs follow a data-driven approach. Evaluating appropriate targets however, remains to be a contextual challenge is less prescriptive in nature for either ITSM practices.

Additional resources

From Apollo 13 to Google SRE from **Sanjeev Sharma**