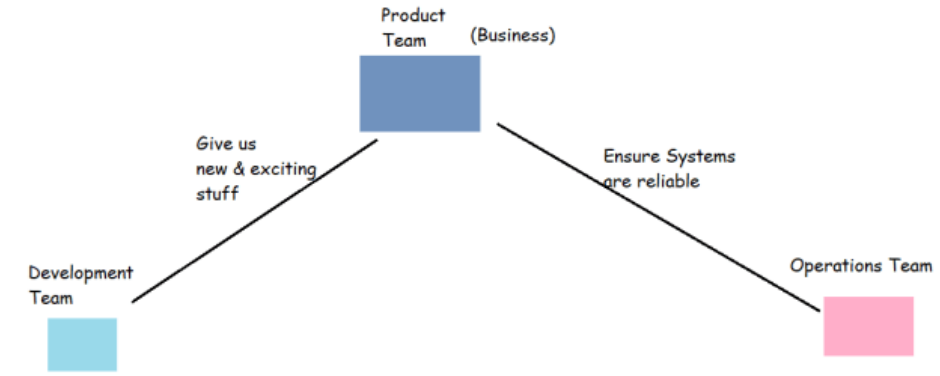
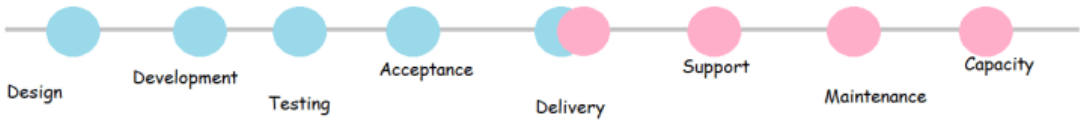
Site Reliability Engineering (SRE)

- SRE is principles based on how Google runs production systems
    - Engineering approach to operations
- Basic problem statement





- Functions of Site Reliability Engineering
    - Reducing Toil
    - Managing Risk
    - Handling Failures
- For any application there are four golden signals
    - Latency: This is the time taken to send a request and recieve a response
    - Traffic: This is measured in number of requests flowing across the n/w
    - Errors: Errors can tell us about misconfigurations in infrastructure, bugs in application code or broken dependencies
    - Saturation: This defines th load on your network and server resources
- Service Level Indicator
    - Success Rate: for every 5000 requests send to the server 4800 requests are be successful
        - SLI : 96% of requests successful
    - Latency: For the last 5000 requests 4000 requests have latency less than 0.5 seconds, 600 with in 2 seconds and 300 within 5seconds
        - SLI : Latency of 80% request with in 0.5%, 92% with in 2s, 99.5 within 5 seconds
- Service Level Objective:
    - application will be up and available for 99.5% in a year

Site reliability engineering (SRE) is crucial for scaling software systems. The SRE process is now an integral aspect of IT since most companies need to ensure reliability across large projects. Using code to support operations, SRE functions as an efficient implementation of DevOps.

SRE fosters operational excellence within any software project or system, and it can offer several benefits to your organization. Let's explore the core principles and practices of SRE, so you can better understand why and how SRE can improve your business efficiency and software lifecycles.

What is SRE?

SRE is a series of practices designed to improve system operations so that developers can focus on achieving velocity and reliability at scale.

The SRE approach gradually expanded as the IT industry started to shift into the DevOps cultural mindset. While a sysadmin traditionally took on the operations role of production, the nature of the job title contributed to a further split between development and operations.

In response to the shift away from siloed teams and the introduction of new DevOps engineer skills, site reliability engineers took on both the development and system operations roles. In a word, engineers designed operations teams. After immense success, the role expanded, with SRE teams using code to help automate operations tasks and manage production systems.

These improvements made it far easier to reliably scale a system. SRE teams took over manual operation tasks and replaced them with far more efficient practices, achieving a balance between project speed and system strength.

Today, site reliability engineers focus on how code is deployed and monitored, taking responsibility for operations improvements and change management.

DevOps vs. SRE process

SRE has considerable overlap with the goals of DevOps. Both combine development and operations for a more integrated lifecycle. Both improve the product release timeline with enhanced collaboration. Proponents of SRE and DevOps embrace a mindset that focuses on dismantling business silos. As a result, many consider SRE an offset of DevOps culture.

But while DevOps puts more focus on pipeline problems and issues related to continuous delivery, the SRE process emphasizes streamlining current operations for better execution. DevOps is interested in the "what", while SRE is interested in the "how." They embody different routes to the same goal. And when integrated into a workplace culture, they can introduce efficient workflows.

7 fundamental SRE principles

SRE facilitates system and service reliability. It accomplishes that with seven core principles.

Embracing risk

No system is perfect. SRE accepts that things will go wrong. But while errors are a problem for system reliability (especially for systems used by consumers), site reliability engineers are expected to lean into the potential failures.

By embracing risk, an engineer can discover problems before deployment or release. Failure is the simplest way to locate problems. By extension, it is also the simplest way to fix them; thereby improving reliability.

Of course, experimental risk acceptance can impinge upon the rapid delivery of services. There are time and labor costs associated with reliability through risk, and that requires the engineer to find a balance. A faulty service can decrease customer satisfaction, but so can a delayed release date. Determining the costs associated with both actions is crucial to the SRE process, and engineers must make choices about which type of risk to accept.

Service level objectives

Service level objectives (SLO) are a set of predetermined performance targets outlined within a service level agreement (SLA). The objectives are measured against service level indicators (SLI), the raw metrics of the current system.

Service level objectives are important because they outline the pain points and desires of the customer. To earn customer satisfaction, the system must achieve a set performance level, preferably

beyond the metrics written in the service level agreement. Engineers must identify customer needs, map out objectives to meet those needs, and achieve system performance and reliability. They must also map those goals according to budget and timeline.

When followed, this SRE principle helps achieve system reliability, efficient project delivery, and a satisfied customer.

Eliminating toil

Toil refers to the tedious work or repetitive tasks an SRE team must do. To streamline operations and improve efficiency, SRE attempts to automate as many tasks as possible. The core principle of eliminating toil improves pipeline velocity and is crucial to scaling larger systems.

When possible, any site reliability engineer should limit their total amount of toil, which frees them up to engage in priority tasks. In addition, they should use supporting processes and tools that contribute to further operational efficiency. Even documentation can eliminate toil, as it can reduce cognitive demand.

SRE principles and practices also put a premium on operational innovation as a method of fighting toil. If a new feature can improve system reliability with less effort, it deserves attention and resources.

Monitoring

Monitoring is crucial for system reliability, as it ensures that all services are running as intended. With the help of monitoring tools, any errors or issues can be rectified with minimal delay. Tracking uptime and availability act as fail-safes — aspects of secure services.

Monitoring also serves as a way to discover improvements, since actionable data can lead to informed decision-making. For example, by collecting information on how a customer uses a service, you can adjust objectives and engage in optimization.

SRE outlines four primary metrics for all monitoring:

- **Latency:** The time delay for a service to respond to a request
- **Traffic:** The amount of demand or load on a system
- **Errors:** The number of services failing to respond to a request
- **Saturation:** The amount of degradation to a service and its usable resources

Automation

To limit as much manual labor as possible, SRE posits automation as a crucial component of system scalability. Finding a way to eliminate human effort will lead to increased velocity as more team members can focus on the tasks that demand human intervention.

Automation also contributes to reliability, since it helps facilitate the functioning of large systems that can grow unwieldy. When numerous operations tasks are pre-determined, consistency can be maintained. In particular, site reliability engineers focus on automating testing, allocating load, responding to incidents, and facilitating communication between individuals and teams.

SRE automation further supports job roles that have become integrated with the shift into DevOps, such as functional testing.

Release engineering

As one of the SRE principles, release engineering refers to delivering software in a manner that is consistent and repeatable. Creating a series of one-time services that cannot repeat is a bad use of automation and introduces unnecessary toil. Instead, engineers who discover improved operational practices should implement those enhancements repeatedly to enhance deployment consistency.

For example, site reliability engineers should create singular release configurations. Errors will occur, but it is far easier to locate and adjust problem areas when an entire team understands the root configuration. Implementing automated and continuous testing provides similar benefits, allowing for rapid releases in manageable increments. Any tool or practice that improves release reliability fits within the SRE fundamentals.

Simplicity

Reliable systems are simple. The more complexity introduced, the more risk and the higher likelihood of failure. A simple system is easy to manipulate, adjust, test, and monitor, all with less toil. A goal of SRE is a boring, uneventful, and mundane project timeline.

Site reliability engineering is meant to operate with consistency at scale, which means a level of complexity is necessary as new services and features are introduced. To balance the need for more comprehensive operations that can meet larger project needs, SRE invests in a holistic coherence. Different engineers, stakeholders, and the client — everyone benefits from clarity and ease of use.

Expect a site reliability engineer to remove unnecessary nodes or inefficient development practices.

Ready to take advantage of SRE?

Anywhere Business offers website development services that help you build functional software. Leverage our expertise in DevOps and SRE, bringing your entire project pipeline up to speed with the latest industry practices.

contact us today

Why SRE principles and practices matter

As digital transformation takes hold, site reliability takes on greater importance. Whether you run a restaurant, a small business, or a large corporation, reliable access to digital and IT services is crucial to the continued operation of critical business resources. If the central tech of an organization fails, downtimes can cause significant financial damage in addition to immense customer friction and dissatisfaction.

SRE functions to eliminate potential disasters, no matter how large the system, traffic, or load grows.

Achieving this kind of reliability is not possible as an afterthought. It must be intentional and extend through the entire product lifecycle, helping facilitate rapid, consistent releases. If development is left unchecked, customer satisfaction or business priorities may fall to the wayside. If operations continue to delegate tasks across a cultural divide between admins and developers, communication can break

down as individuals nestle back into siloed responsibilities. SRE principles can help align development goals from the start.

The SRE process also incorporates lasting best practices beyond the standard development lifecycle. If a policy or procedure helps increase reliability, the entire organization's operations should evolve according to the new precept. Automation, principle adherence, and the introduction of new tools can transform the entire workplace environment, further contributing to system reliability.

How to apply SRE practices to your project

SRE is not a set of methods or prescribed solutions. Instead, it offers overarching principles that can guide production management. The right solutions will be different for each unique business. Still, there are common approaches to applying SRE principles and best practices:

- Determine acceptable levels of reliability
- Empower management to take on predetermined levels of risk
- Build robust service level objectives and service level agreements
- Create a budget with room for error
- Eliminate areas of high toil
- Create case-dependent standards of efficiency
- Monitor services and act on possible areas of improvement
- Invest in automation tools and automate wherever possible
- Document release standards and educate all stakeholders
- Investigate complex systems and invest in tools that improve system simplicity

Combining SRE with DevOps

In many ways, SRE is the operational application of DevOps. If you want to integrate DevOps services into the fabric of your organizational culture, consider the following steps standard to the SRE process:

- **Tear down business silos:** Individuals or segregated teams place nuanced focus on particular components, but the rewards pale in comparison to the troubles caused by lack of communication. SRE stresses collaboration. Integrate teams at every possible juncture.
- **Accept failure:** DevOps places stress on mitigating failures or bugs before they arise, but SRE accepts failure as a necessary path to reliability. Allow leadership to encourage experimentation and the rapid correction of inevitable problems.
- **Automate:** Both DevOps and SRE value automation whenever possible. Whether it is development practices or operational procedures, improve consistency and system efficiency through automation.
- **Monitor:** DevOps and SRE can both achieve rapid deployments, but SRE focuses on quantifying uptimes, availability, and incident response. Invest in monitoring tools for better data capture and actionable system improvement insights.

Get started with SRE

As a development philosophy, site reliability engineering offers many benefits to organizations. The core principles of SRE will not only support the cultural mindset of DevOps that is typical of an integrated development project, they will also lead to system efficiencies that result in massive system improvements. More importantly, it delivers the reliability needed to achieve customer satisfaction. Any organization can benefit from SRE, that’s why SRE is undoubtedly a well-recognized discipline and methodology.