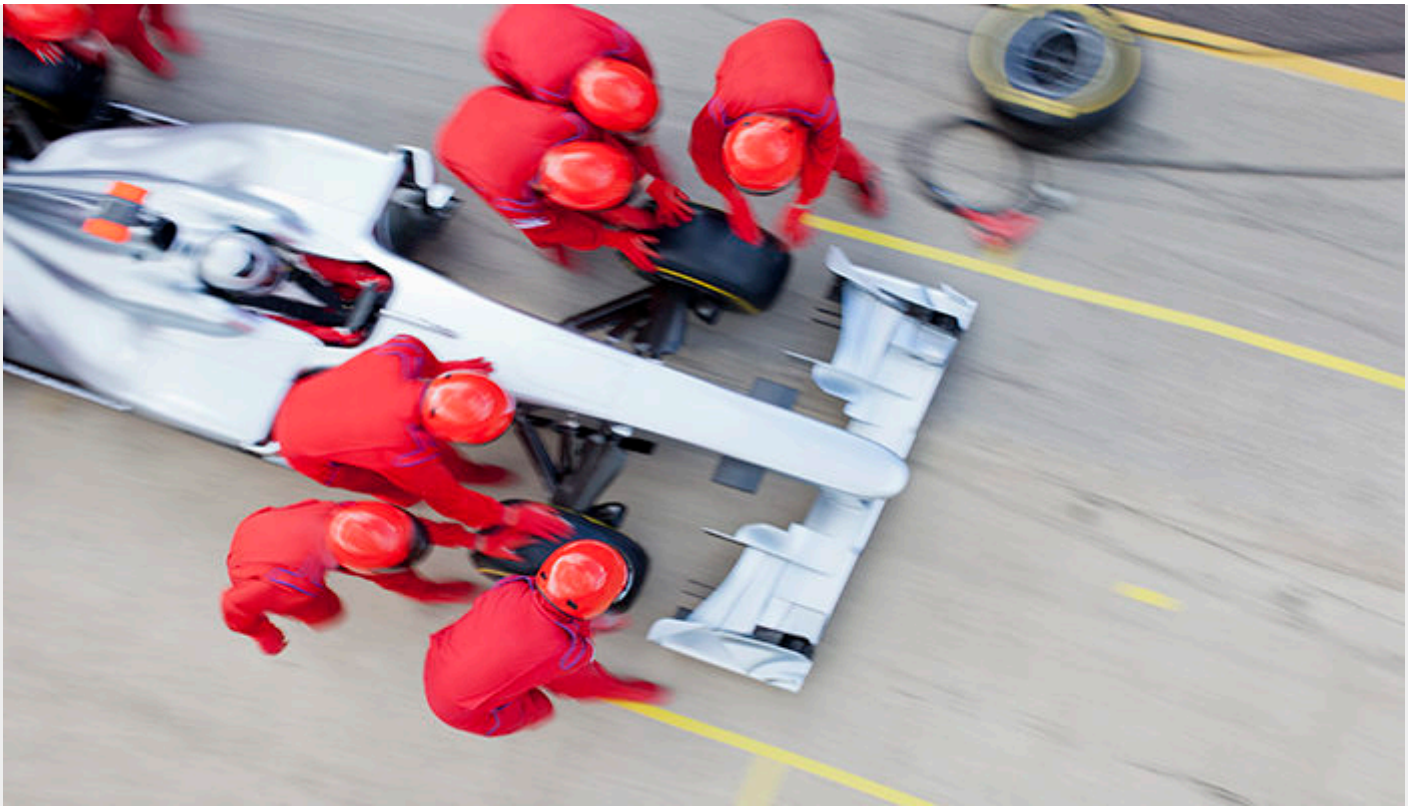
SRE BASICS: SITE RELIABILITY ENGINEERING EXPLAINED



The cloud is well and truly becoming the default mode for running software in the digital age. And when it comes to managing application performance and stability while responding to changes in business need, modern approaches such as SRE are fast taking root.

Introduced by Google, SRE has rapidly attracted interest globally as DevOps practices and automation become essential capabilities for effective management of applications hosted in hybrid cloud environments. In fact, LinkedIn's 2020 Emerging Jobs report ranked SRE in 5th place, while the DevOps Institute's 2021 Upskilling Enterprise DevOps Skills Report indicated a jump from 15% to 21% in the adoption of SRE as an operating model.

Let's look at what SRE is all about in this article.

What is site reliability engineering?

Short for Site Reliability Engineering, SRE is a discipline that applies aspects of software engineering to IT operations, with the goal of creating ultra-scalable and highly reliable software systems.

SRE originated from Google as its approach to service management. Ben Treynor, the senior VP overseeing technical operations at Google, was the originator of the term Site Reliability Engineering and the famous quote:

> *"SRE is what happens when you ask a software engineer to design an operations team."*

In 2003, with a background in software engineering, Treynor set up a "production team" that has become the blueprint for SRE, based on two main characteristics:

- Using engineers with software development backgrounds to perform work done by an operations team.
- Banking on these engineers to automate work that was previously done manually.

While continuing to carry out some manual operational tasks, like ticketing and handling escalations, the SRE team is expected to primarily focus on engineering new products and services by automating and scaling applications and production environment systems. This is because as service usage increases, the effort to manage the operational load increases. Automation reduces the operational overhead, resulting in extra bandwidth that can be applied in supporting new features and improvements to existing applications.

Another benefit of having software engineers as SREs is reducing the friction of handovers from product development teams since they understand their perspective, even though the SRE's focus is on operational stability.

(Compare SRE & ITOps in more detail.)

How does SRE work?

The term reliability is at the heart of SRE. The ITIL® 4 Foundation Publication defines reliability as:

> *"The ability of a product, service, or other configuration item to perform its intended function for a specified period of time or number of cycles."*

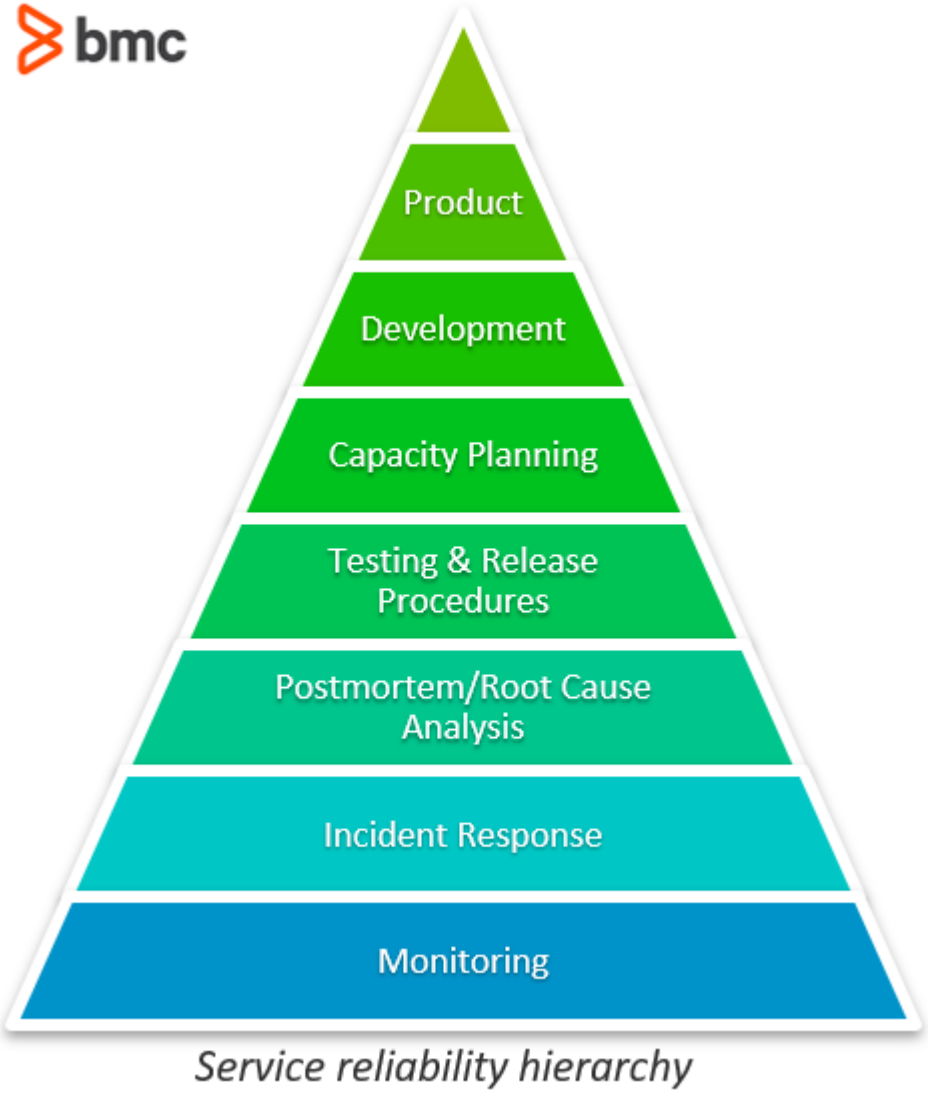
For this to be achieved, an SRE team has to be responsible for these components of each of their services:

- Availability
- Latency
- Performance
- Efficiency
- Change management
- Monitoring
- Emergency response
- Capacity planning

Even though they maintain a strong focus on supporting product engineering, the SRE priority is on making sure that existing services work as they should.

In simple terms, SREs run services and are ultimately responsible for the health of these services. The SRE's day-to-day activity involves building and operating large distributed computing systems. As the hierarchy below demonstrates, the foundational practices in making a service reliable are:

1. Supporting existing services by monitoring and addressing any issues arising.
2. Then building up to scaling.
3. Finally, supporting development of new product features.



Service reliability hierarchy

At Google, a 50% cap is placed on SRE activities that make the service stable and operable. This is contrary to other service management approaches which would put such focus to near 80-90% of a systems administrator's time. The SRE is expected to spend the other 50% of the time on coding, in an attempt to design and deploy automation tools and other efforts to reduce the workload and introduce efficiency and effectiveness in operations.

By working with users and product teams, the SRE is able to innovatively develop automated tools that result in better production environments characterized by:

- Scalability
- Graceful degradation during failure
- Integration with other services and systems
- Additional benefits

SRE principles

To adopt SRE in your organization, these are principles to strive for:

Embracing risk

SREs do not believe in 100% reliable services. Instead, they target services to be reliable enough based on risk the business is willing to bear based on a cost-benefit analysis. A target of 99.99% availability is set, with SRE seeking to balance the risk of unavailability with the goals of rapid innovation and efficient service operations, so that users' overall happiness is optimized.

Using Service Level Objectives

To understand and therefore deliver services that meet the needs of users, service level objectives (SLOs) are used. SLOs are a target value or range of values for a service level that is measured by a Service Level Indicator (SLI) such as:

- Availability
- Error rate
- Request latency
- Throughput
- Other indicators

The structure for SLOs is either:

- SLI ≤ target
- Lower bound ≤ SLI ≤ upper bound

Choosing and publishing SLOs to users sets expectations about how a service will perform, which helps reduce unfounded complaints to service owners about service performance (e.g., slowness).

(Explore error budgets, a tool that works in SRE & non-SRE environments.)

Eliminating toil

Toil is operational activity tied to running a production service that tends to be manual, repetitive, automatable, tactical, devoid of enduring value, and that scales linearly as a service grows. Sources of toil include:

- Interrupts (non-urgent service-related messages and emails)
- On-call (urgent) response
- Releases
- Pushes

The work of reducing toil and scaling up services is the "engineering" part of SRE. While not all toil is bad, too much of it is dangerous, hence the need to devise automated solutions through software and systems engineering to eliminate it.

Monitoring distributed systems

Monitoring is a key attribute of SRE with the goal being to find out what is broken (or about to break) and why. If a system cannot self-heal, then an SRE should investigate the alert, identify and mitigate the issues, and determine the root cause. The priority metrics that SRE monitoring covers are:

- Latency
- Traffic
- Errors
- Saturation

Automate

Automation is the raison d'être when it comes to SRE, especially if manual activities can be replaced by automated ones, or the need for automation is eliminated by designing systems that are autonomous. The benefits of automation include:

- Consistency
- Time saving
- Faster repairs and actions
- Providing a platform that can be extended to other services

The SRE writes code to automate management of the lifecycle of systems infrastructure, not their data.

Release engineering

The SRE supports developers and release engineers to ensure new services and features are working well and can be supported in the future.

The SRE will use tools developed by release engineers to ensure projects are released using consistent and repeatable methodologies. They will work together to develop strategies for:

- Canarying changes
- Pushing out new releases without interrupting services
- Rolling back features that demonstrate problems

Simplicity

The SRE approach to managing systems is keeping agility and stability in balance in the system. SREs work to create procedures, practices, and tools that render software more reliable, with little impact on developer agility.

SRE teams push back when accidental complexity is introduced into the systems for which they are responsible, and constantly strive to eliminate complexity in systems they onboard and for which they assume operational responsibility.

SRE supports agile operations & development

With systems growing increasingly complex, some kind of failure is inevitable. Because SRE doesn't demand perfection, it can instead look at how to support service reliability—while being prepared when failure does occur.

"Gain insight to the capabilities necessary to attract top SRE talent and make them successful in your organization with artificial intelligence for operations (AIOps) and artificial intelligence for service management (AISM) capabilities."

Related reading

- BMC DevOps Blog
- SRE vs DevOps: What's The Difference?
- The State of SRE Today
- How & Why To Become a Software Factory
- Top IT Operations Trends