Part II - Principles

Part II. Principles

This section examines the *principles* underlying how SRE teams typically work—the patterns, behaviors, and areas of concern that influence the general domain of SRE operations.

The first chapter in this section, and the most important piece to read if you want to attain the widest-angle picture of what exactly SRE does, and how we reason about it, is Embracing Risk. It looks at SRE through the lens of risk—its assessment, management, and the use of error budgets to provide usefully neutral approaches to service management.

Service level objectives are another foundational conceptual unit for SRE. The industry commonly lumps disparate concepts under the general banner of service level agreements, a tendency that makes it harder to think about these concepts clearly. Service Level Objectives attempts to disentangle indicators from objectives from agreements, examines how SRE uses each of these terms, and provides some recommendations on how to find useful metrics for your own applications.

Eliminating toil is one of SRE's most important tasks, and is the subject of Eliminating Toil. We define *toil* as mundane, repetitive operational work providing no enduring value, which scales linearly with service growth.

Whether it is at Google or elsewhere, monitoring is an absolutely essential component of doing the right thing in production. If you can't monitor a service, you don't know what's happening, and if you're blind to what's happening, you can't be reliable. Read Monitoring Distributed Systems, for some recommendations for what and how to monitor, and some implementation-agnostic best practices.

In The Evolution of Automation at Google, we examine SRE's approach to automation, and walk through some case studies of how SRE has implemented automation, both successfully and unsuccessfully.

Most companies treat release engineering as an afterthought. However, as you'll learn in Release Engineering, release engineering is not just critical to overall system stability—as most outages result from pushing a change of some kind. It is also the best way to ensure that releases are consistent.

A key principle of any effective software engineering, not only reliability-oriented engineering, simplicity is a quality that, once lost, can be extraordinarily difficult to recapture. Nevertheless, as the old adage goes, a complex system that works necessarily evolved from a simple system that works. Simplicity, goes into this topic in detail.

Further Reading from Google SRE

Increasing product velocity safely is a core principle for any organization. In "Making Push On Green a Reality" [Kle14], published in October 2014, we show that taking humans out of the release process can paradoxically reduce SREs' toil while *increasing* system reliability.

previous

Chapter 2 - The Production Environment at Google, from the Viewpoint of an SRE

next

Chapter 3 - Embracing Risk