Google Cloud
SUMMIT

SRE Overview

April, 2018

What worries us about reliability?

Topics

What is SRE?

Key principles of SRE

Practices of SRE

How to get started

Ways to get help

Google Cloud

What is SRE?

1

Google Cloud
SUMMIT

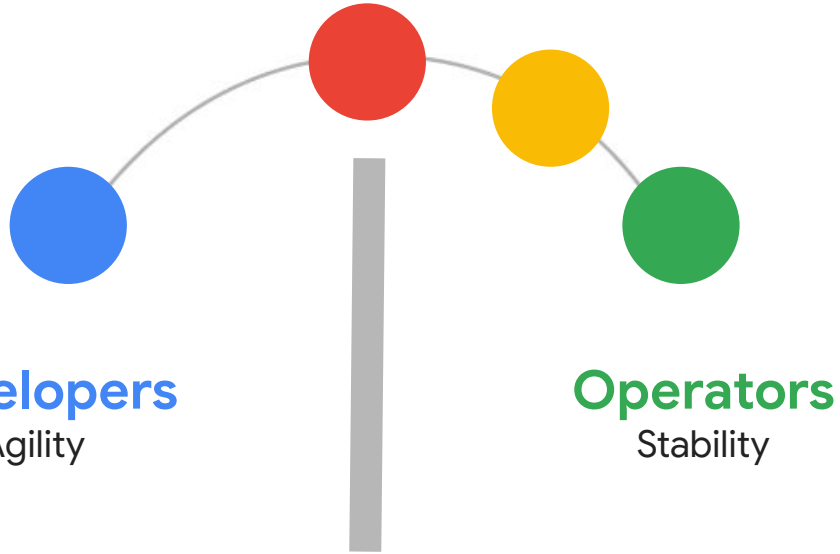
Software's long-term cost

Software engineering as a discipline focuses on designing and building rather than operating and maintaining, despite estimates that 40%[1] to 90%[2] of the total costs are incurred after launch.

[1] Glass, R. (2002). Facts and Fallacies of Software Engineering, Addison-Wesley Professional; p. 115.
[2] Dehaghani, S. M. H., & Hajrahimi, N. (2013). Which Factors Affect Software Projects Maintenance Cost More? Acta Informatica Medica, 21(1), 63–66. http://doi.org/10.5455/AIM.2012.21.63-66

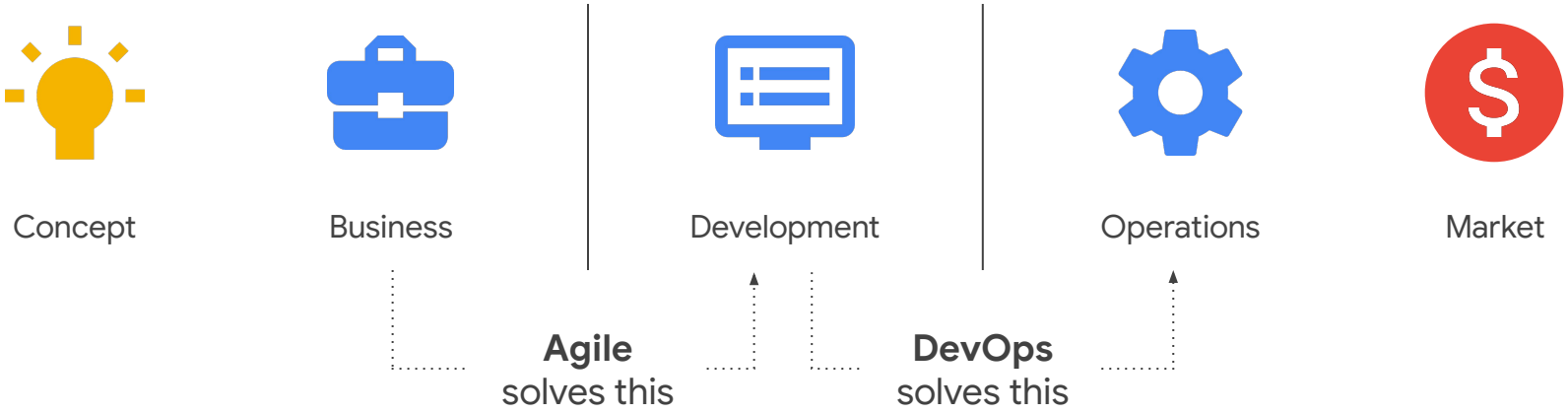
Google Cloud

Incentives aren't aligned.



Google Cloud

Reducing product lifecycle friction



Google Cloud

interface DevOps

DevOps

is a set of practices, guidelines and culture designed to break down silos in IT development, operations, architecture, networking and security.

5 key areas

1. Reduce organizational silos
2. Accept failure as normal
3. Implement gradual changes
4. Leverage tooling and automation
5. Measure everything

Google Cloud

The SRE approach to operations

Use data to guide decision-making.

Treat operations like a software engineering problem:

- Hire people motivated and capable to write automation.
- Use software to accomplish tasks normally done by sysadmins.
- Design more reliable and operable service architectures from the start.

Google Cloud

What do SRE teams do?

- Site Reliability Engineers develop solutions to design, build, and run large-scale systems **scalably, reliably, and efficiently**.
- We **guide system architecture** by operating at the intersection of software development and systems engineering.

- SRE is a job function, a mindset, and a set of **engineering approaches** to running better production systems.
- We approach our work with a spirit of constructive pessimism: we **hope for the best, but plan for the worst**.

class SRE implements DevOps

DevOps

is a set of practices, guidelines and culture designed to break down silos in IT development, operations, architecture, networking and security.

Site Reliability Engineering

is a set of practices we've found to work, some beliefs that animate those practices, and a job role.

5 key areas

1. Reduce organizational silos
2. Accept failure as normal
3. Implement gradual changes
4. Leverage tooling and automation
5. Measure everything

Error Budgets

The key principle of SRE

2

How to measure reliability

- Naive approach:

$$\text{Availability} = \frac{\text{good time}}{\text{total time}}$$

= which fraction of time
the service is available and working

- Intuitive for humans

- Relatively easy to measure for a continuous binary metric e.g. machine uptime

- Much harder for distributed request/response services
  - Is a server that currently does not get requests up or down?
  - If 1 of 3 servers are down, is the service up or down?

How to measure reliability

- More sophisticated approach:

$$\text{Availability} = \frac{\text{good interactions}}{\text{total interactions}}$$

= which fraction of real users for whom the service is available and working

- Handles distributed request/response services well

- Enables these cases:
  - Is a server that currently does not get requests up or down?
  - If 1 of 3 servers are down, is the service up or down?

Reliability level	Allowed unreliability window		
	per year	per quarter	per 30 days
90%	36.5 days	9 days	3 days
95%	18.25 days	4.5 days	1.5 days
99%	3.65 days	21.6 hours	7.2 hours
99.5%	1.83 days	10.8 hours	3.6 hours
99.9%	8.76 hours	2.16 hours	43.2 minutes
99.95%	4.38 hours	1.08 hours	21.6 minutes
99.99%	52.6 minutes	12.96 minutes	4.32 minutes
99.999%	5.26 minutes	1.30 minutes	25.9 seconds

Reliability level	Allowed unreliability window		
	per year	per quarter	per 30 days
90%	36.5 days	9 days	3 days
95%	18.25 days	4.5 days	1.5 days
99%	3.65 days	21.6 hours	7.2 hours
99.5%	1.83 days	10.8 hours	3.6 hours
99.9%	8.76 hours	2.16 hours	43.2 minutes
99.95%	4.38 hours	1.08 hours	21.6 minutes
99.99%	52.6 minutes	12.96 minutes	4.32 minutes
99.999%	5.26 minutes	1.30 minutes	25.9 seconds

Error Rate	Allowed duration
100%	21.6 minutes
10%	3.6 hours
1%	36 hours
0.1%	15 days
<0.05%	all month

Google Cloud

“100% is the wrong reliability target for basically everything.”

Benjamin Treynor Sloss, Vice President of 24x7 Engineering, Google

Google Cloud

Error budgets

- Product management & SRE define an **availability target**.
- 100% - availability target is a “budget of unreliability” (or the **error budget**).
- Monitoring measures **actual uptime**.
- Control loop for utilizing budget!

Google Cloud

Benefits of error budgets

- **Common incentive for devs and SREs**
  Find the right balance between innovation and reliability
- **Dev team can manage the risk themselves**
  They decide how to spend their error budget
- **Unrealistic reliability goals become unattractive**
  Such goals dampen the velocity of innovation
- **Dev team becomes self-policing**
  The error budget is a valuable resource for them
- **Shared responsibility for system uptime**
  Infrastructure failures eat into the devs' error budget

Glossary of terms

SLI

service level **indicator**: a well-defined measure of 'successful enough'

- *used to specify SLO/SLA*
- *Func(metric) < threshold*

SLO

service level **objective**: a top-line target for fraction of successful interactions

- *specifies goals (SLI + goal)*

SLA

service level **agreement**: consequences

- *SLA = (SLO + margin) + consequences = SLI + goal + consequences*

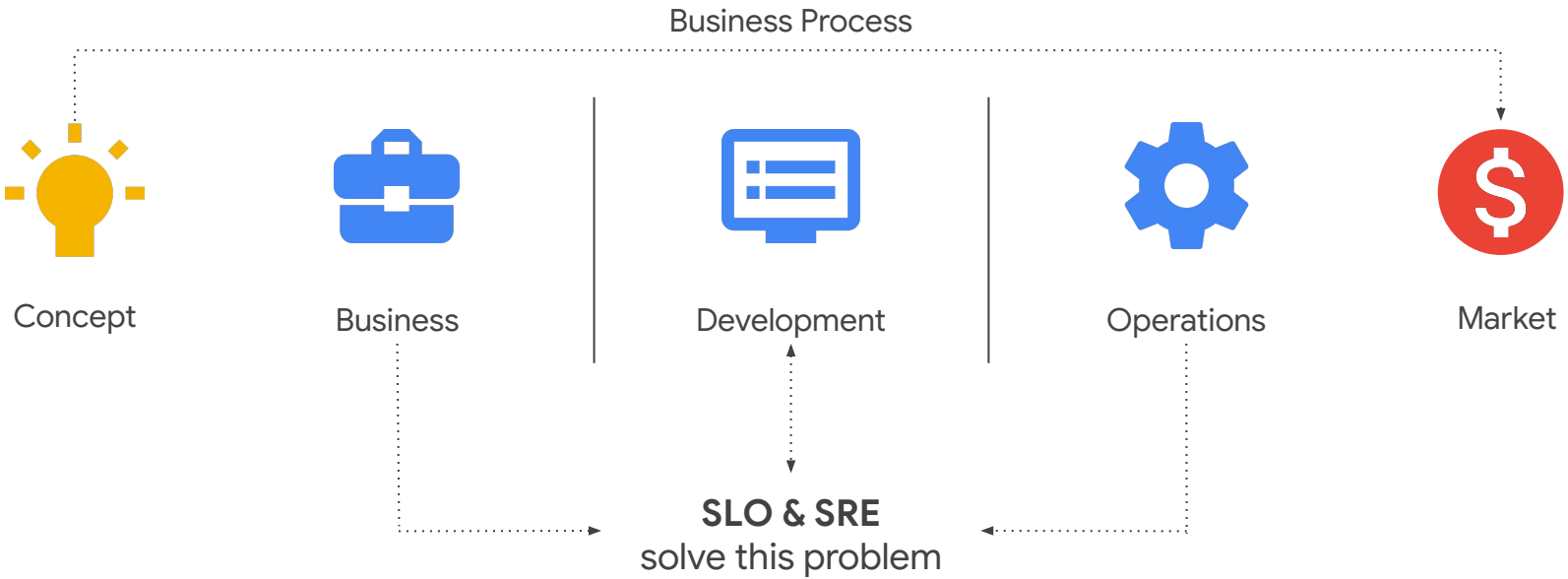
SLO definition and measurement

- Service-level objective (SLO): a target for SLIs **aggregated over time**
  - Measured using an **SLI (service-level indicator)**
  - Typically, **sum(SLI met) / window >= target percentage**
- Try to exceed SLO target, *but not by much*

- Choosing an appropriate SLO is complex. Try to keep it simple, avoid absolutes, perfection can wait.

  Why?
  - Sets priorities and constraints for SRE and dev work
  - Sets user expectations about level of service

Product lifecycle



Google Cloud

class SRE implements DevOps

DevOps

is a set of practices, guidelines and culture designed to break down silos in IT development, operations, architecture, networking and security.

Site Reliability Engineering

is a set of practices we've found to work, some beliefs that animate those practices, and a job role.

5 key areas

1. Reduce organizational silos: Share ownership
2. Accept failure as normal: Error budgets
3. Implement gradual changes
4. Leverage tooling and automation
5. Measure everything: Measure reliability

The practices of SRE

3

Google Cloud
SUMMIT

Areas of practice

Metrics & Monitoring

Capacity Planning

Change Management

Emergency Response

Culture

Google Cloud

Monitoring & Alerting

Monitoring: automate recording system metrics

- Primary means of determining and maintaining reliability

Alerting: triggers notification when conditions are detected

- Page: Immediate human response is required
- Ticket: A human needs to take action, but not immediately

Only involve humans when SLO is threatened

- Humans should never watch dashboards, read log files, and so on just to determine whether the system is okay

Demand forecasting and capacity planning

Plan for organic growth
Increased product adoption and usage by customers.

Determine inorganic growth
Sudden jumps in demand due to feature launches, marketing campaigns, etc.

Correlate raw resources to service capacity
Make sure that you have enough spare capacity to meet your reliability goals.

Google Cloud

Efficiency and performance

Capacity can be expensive —> optimize utilization

- Resource use is a function of demand (load), capacity, and software efficiency
- SRE demands prediction and provisioning, and can modify the software

SRE monitors utilization and performance

- Regressions can be detected and acted upon
- Immature team: by adjusting the resources or by improving the software efficiency
- Mature team: rollback

Google Cloud

Change management

- Roughly 70%[1] of outages are due to changes in a live system

[1] Analysis of Google internal data, 2011-2018

Mitigations:

- Implement progressive rollouts
- Quickly and accurately detect problems
- Roll back changes safely when problems arise

- Remove humans from the loop with automation to:
  - Reduce errors
  - Reduce fatigue
  - Improve velocity

Pursuing maximum change velocity

100% is the wrong reliability target for basically everything

- Determine the desired reliability for your product
- Don't try to provide better quality than desired

Spend error budget to increase development velocity

- The goal is not zero outages, but maximum velocity within the error budget
- Use error budget for releases, experiments etc.

Google Cloud

Provisioning

A combination of change management and capacity planning

- Increase the size of an existing service instance/location
- Spin up additional instances/locations

Needs to be done quickly

- Unused capacity can be expensive

Needs to be done correctly

- Added capacity needs to be tested
- Often a significant configuration change —> risky

Google Cloud

Emergency response

“Things break, that’s life”

Few people naturally react well to emergencies, so you need a process:

- **First of all, don’t panic!**
  You aren’t alone and the sky isn’t falling.
- Mitigate, troubleshoot, and fix.
- If you feel overwhelmed, pull in more people.

Google Cloud

Incident & postmortem thresholds

- User-visible downtime or degradation beyond a certain threshold
- Data loss of any kind
- On-call engineer significant intervention (release rollback, rerouting of traffic, etc.)
- A resolution time above some threshold

It is important to define incident & postmortem criteria before an incident occurs.

Google Cloud

Postmortem philosophy

The primary goals of writing a postmortem are to ensure that:

- The incident is documented
- All contributing root causes are well understood
- Effective preventive actions are put in place to reduce the likelihood and/or impact of recurrence

Postmortems are expected after any significant undesirable event

- Writing a postmortem is not a punishment

Blamelessness

- Postmortems must focus on identifying the contributing causes without indicating any individual or team
- A blamelessly written postmortem assumes that everyone involved in an incident had good intentions
- "Human" errors are **systems problems**. You can't "fix" people, but you can fix systems and processes to better support people in making the right choices.
- If a culture of finger pointing prevails, people will not bring issues to light for fear of punishment

Google Cloud

Toil management/operational work

Why?

Because:

- Exposure to real failures guides how you design systems
- You can't automate everything
- If you do enough Ops work, you know what to automate

What?

Work directly tied to running a service that is:

- Manual (manually running a script)
- Repetitive (done every day or for every new customer)
- Automatable (no human judgement is needed)
- Tactical (interrupt-driven and reactive)
- Without enduring value (no long-term system improvements)
- O(n) with service growth (grows with user count or service size)

Team skills

Hire good **software engineers (SWE)** *and* good **systems engineers (SE)**.

Not necessarily all in one person.

Try to get a 50:50 mix of SWE and SE skillsets on team

Everyone should be able to code.

SE != "ops work"

For more detail, see "Hiring Site Reliability Engineers," by Chris Jones, Todd Underwood, and Shylaja Nukala, ;login:, June 2015

Google Cloud

Empowering SREs

- SREs must be empowered to enforce the error budget and toil budget.
- SREs are valuable and scarce. Use their time wisely.
- Avoid forcing SREs to take on too much operational burden; load-shed to keep the team healthy.

Google Cloud

class SRE implements DevOps

DevOps

is a set of practices, guidelines and culture designed to break down silos in IT development, operations, architecture, networking and security.

Site Reliability Engineering

is a set of practices we've found to work, some beliefs that animate those practices, and a job role.

5 key areas

1. Reduce organizational silos: Share ownership
2. Accept failure as normal: Error budgets & blameless postmortems
3. Implement gradual changes: Reduce cost of failure
4. Leverage tooling and automation: Automate common cases
5. Measure everything: Measure toil and reliability

How to get started

4

Google Cloud
SUMMIT

Do these four things.

1. Start with Service Level Objectives.
   SRE teams work to a SLO and/or error budget. They defend the SLO.
2. Hire people who write software.
   They'll quickly become bored by performing tasks by hand and replace manual work.
3. Ensure parity of respect with rest of the development/engineering organization.
4. Provide a feedback loop for self-regulation.
   SRE teams choose their work.
   SREs must be able to shed work or reduce SLOs when overloaded.

You can do this.

- Pick **one** service to run according to SRE model
- Empower the team with strong executive sponsorship and support
- Culture and psychological safety is critical.
- Measure Service Level Objectives & team health.
- Incremental progress frees time for more progress.

Spread the love.

- Spread the techniques and knowledge once you have a solid case study within your company
- If you have well-defined SLOs, Google can work with you to reduce friction via shared monitoring and other collaboration.

SRE solves cloud reliability.

- Effortless scale shouldn't meet escalating operational demands.
- Automation and engineering for operability enable scaling systems without scaling organizations.
- Tension between product development and operations doesn't need to exist.
- Error budgets provide measurement and flexibility to deliver both reliability *and* product velocity.

Ways to get help

5

Google Cloud
SUMMIT

Customer Reliability Engineering

Google SREs who focus on *customer* production systems instead of specific Google systems.

- Conduct reliability reviews on customer applications to identify problems
- Build shared monitoring and alerting.
- Advise customers to prepare them for critical periods
- Do joint postmortems and assign joint bug fixes.
- Conduct regular design reviews, DiRT, and wheel-of-misfortune.

Google Cloud

How to get a CRE

Requirements:

1. Executive support for creating and adhering to error budgets.
2. Willingness to invest engineering time and effort on reliability.

Contact your Account Manager and express interest.

CRE is in high demand, so there is a selection process and queue.

Google Cloud

PSO Consulting: SRE Fundamentals

Consulting offering aiming to improve reliability of your GCP Applications with a focus on SRE Culture, technical design optimization, SLO definition and management.

Ideal for customers with active workloads running on GCP and requiring support in defining SLOs and error budgets.

Contact your Account Manager

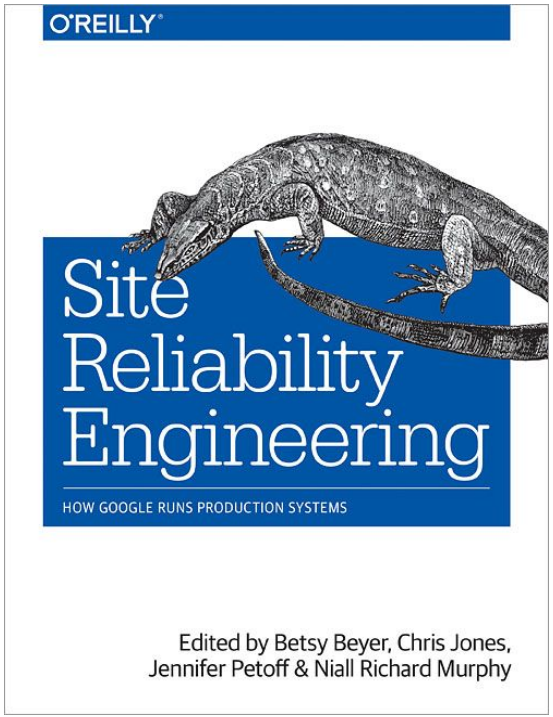
Google Cloud

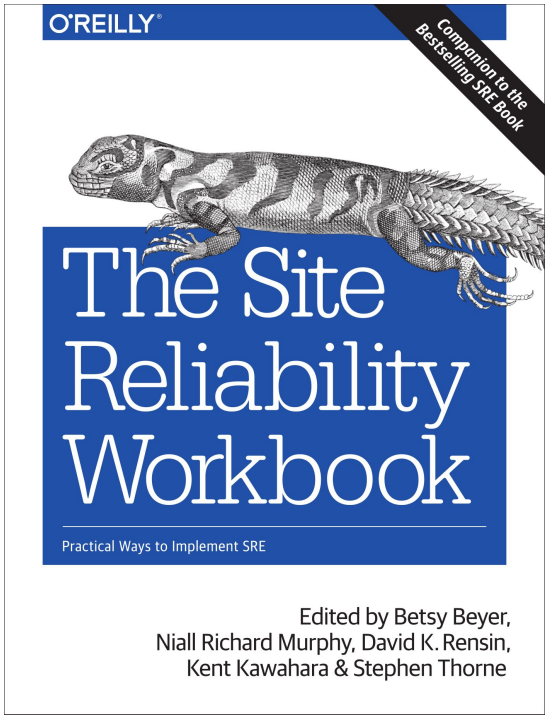
Objectives

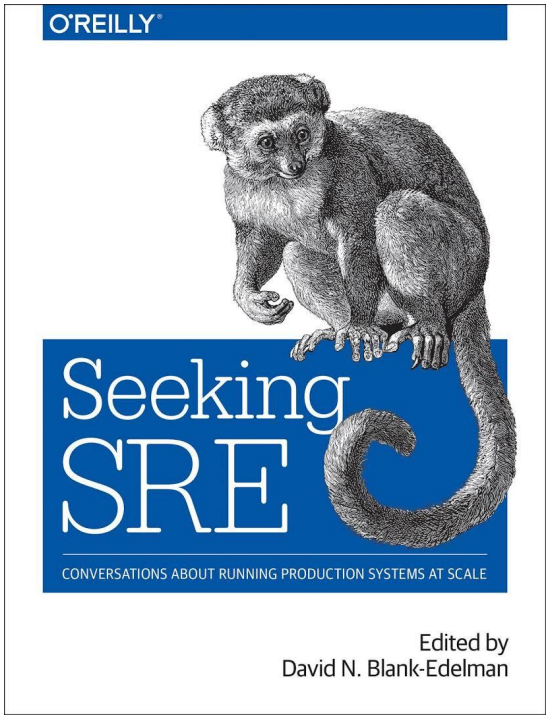
- Introduction to SRE principles & practices
- Establishing effective Error Budgets and practical measured SLOs
- Review architecture design of applications in regards to reliability on GCP

Activities and Deliverables

1. Application maturity analysis
2. SLO Workshop
3. Design and Operations Review
4. Recommendations for improving architecture and reliability







Google Cloud

Cover images used with permission. These books can be found on shop.oreilly.com.

Questions?

Google Cloud
SUMMIT

Google Cloud
SUMMIT

Thank you

Google Cloud
SUMMIT

Customer Stories

Since Evernote introduced SLOs...

[Evernote] were able to better **quantify** customer impact and reduce our release windows from five to two to minimize customer pain

Relationship between **operations** and **development** teams has subtly but markedly **improved**

Removing the human interpretation of quality of service (QoS) has allowed both teams to maintain the **same view** and standards

Google Cloud

"SLOs provided a **common ground** when we had to facilitate multiple releases in a **compressed timeline** in 2017. While [Evernote] chased down a complex bug, product development requested that [Evernote] apportion normal weekly release over multiple separate windows, each of which would potentially impact customers"

Evernote