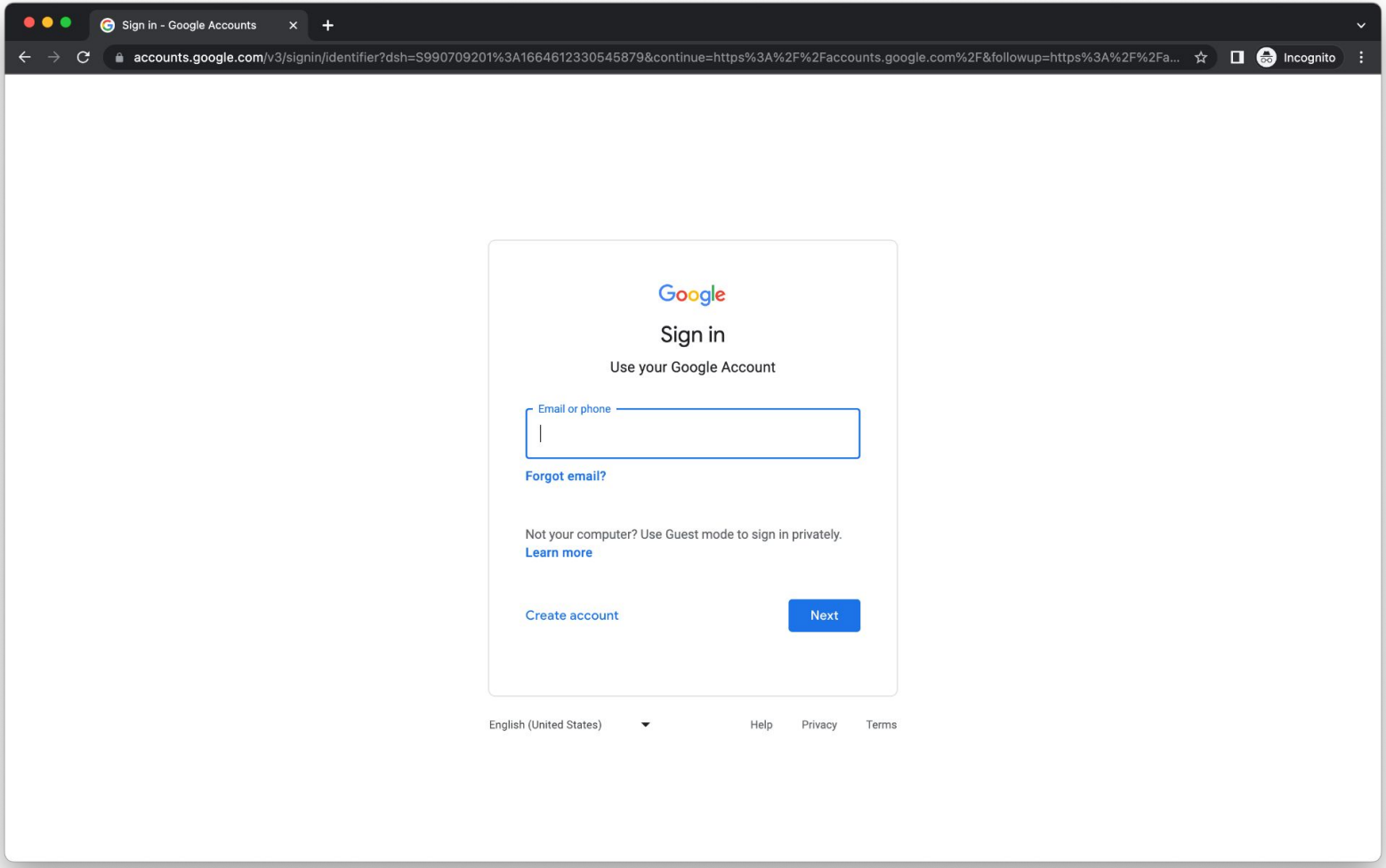
Google

SRE & Python

Ramón Medrano Llamas, Sr. Staff SysEng
@rmedranollamas

Site Reliability Engineering

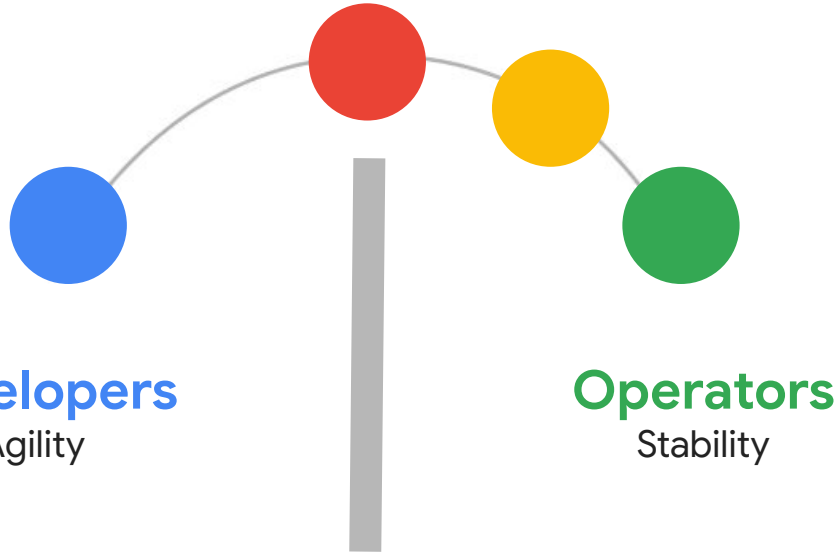

Sign in - Google Accounts
accounts.google.com/v3/signin/identifier?dsh=S990709201%3A1664612330545879&continue=https%3A%2F%2Faccounts.google.com%2F&followup=https%3A%2F%2Fa...
Incognito
Google
Sign in
Use your Google Account
Email or phone
Forgot email?
Not your computer? Use Guest mode to sign in privately.
Learn more
Create account
Next
English (United States)
Help
Privacy
Terms

What is SRE?

1

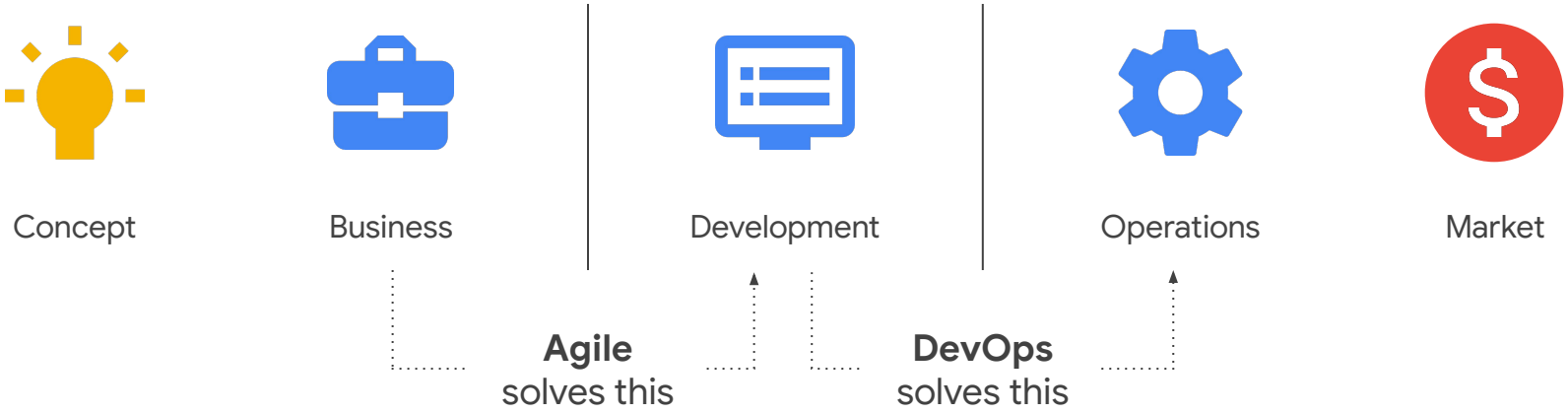
Google

Incentives aren't aligned.



Google

Reducing product lifecycle friction



Google

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**.

Google

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

Google

The practices of SRE

2

Google

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

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

Source: Pixabay (no attribution required)

Google

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

Google

Provisioning

A combination of change management and capacity planning

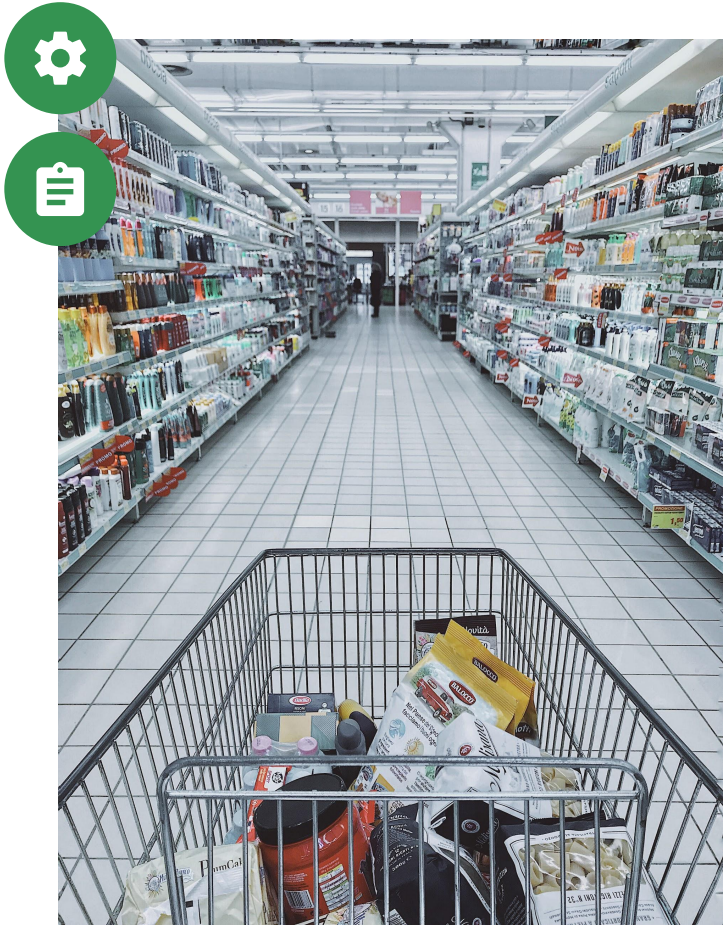
- Increase the size of an existing service instance/location
- Spin up additional instances/locations

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

Software engineering within SRE

3

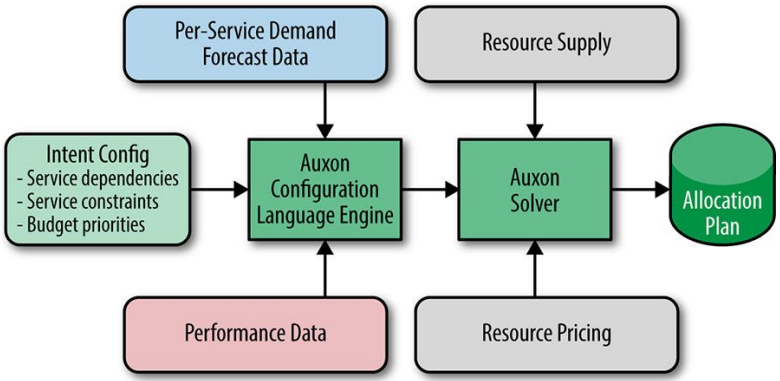
Google

SRE is unique within Google

1. Breadth and depth of production knowledge. Scalability, graceful degradation during failure, and the ability to easily interface with other infrastructure or tools.
2. SREs are embedded in the subject matter. They easily understand the needs and requirements of the tool being developed.
3. Direct relationship with the intended user. This results in frank and high-signal user feedback. Releasing a tool to an internal audience with high familiarity with the problem space means that a development team can launch and iterate more quickly.

Google

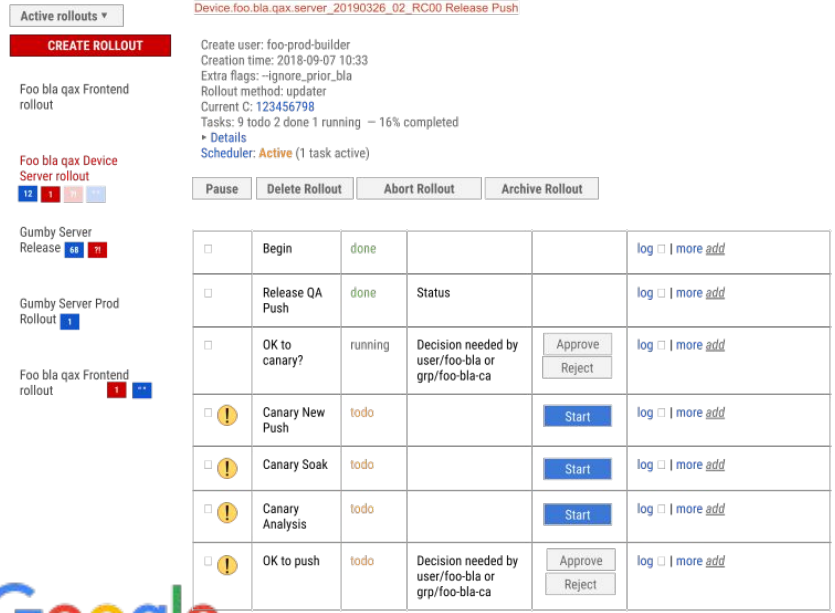
Case study: Auxon



- Intent-based capacity management.
- State what you need, let the Solver find out how.
- All the configuration language is Python.
- Well integrated with tens of data sources (demand, performance data).
- The Cluster and Network Topology are Python rules, too. Checked into source control.
- The solver is a C++ kernel.

Google

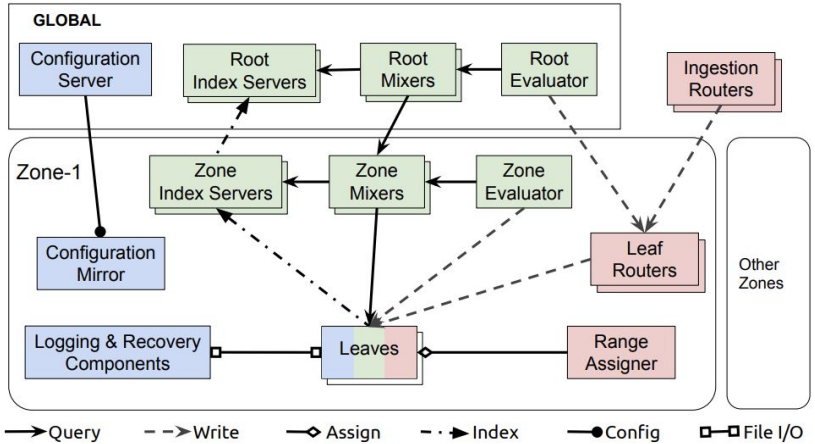
Case study: Sisyphus



- Framework for automation of rollouts.
- Iterate quickly! Plugins! Flexibility!
- Sisyphus got a lot of adoption: it came at the right time, with the right flexibility.
- Managing this Python codebase was a very large challenge. Its strength was its weakness.
- Used typing and static analysis to improve code quality.

Google

Case study: Monarch



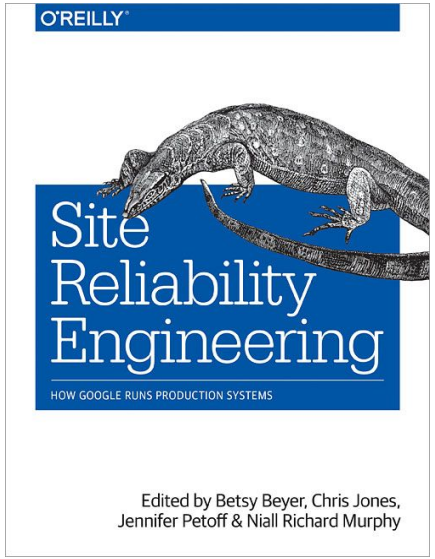
- Planet scale monitoring system.
- Huge in-memory time series database. Hierarchical, very high throughput.
- Base for mostly all alerting and SLO measurement.
- Query language, "mash", it is a Python DSL.
- Most of the dashboards build in a Python framework, Gmon/Viceroy.

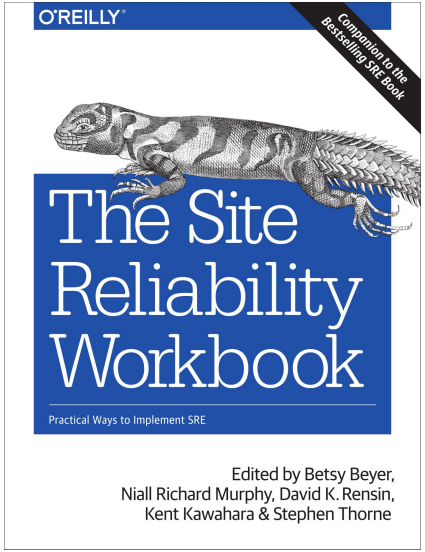
Google

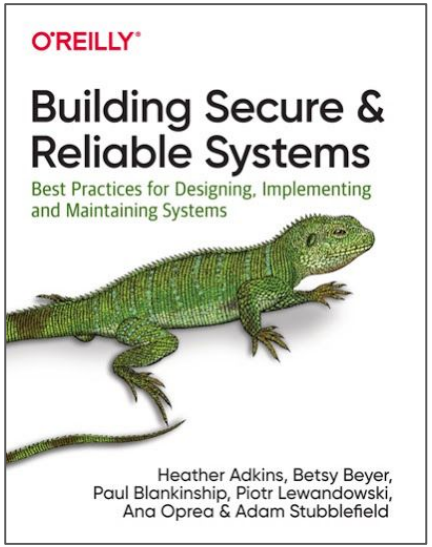
Questions?

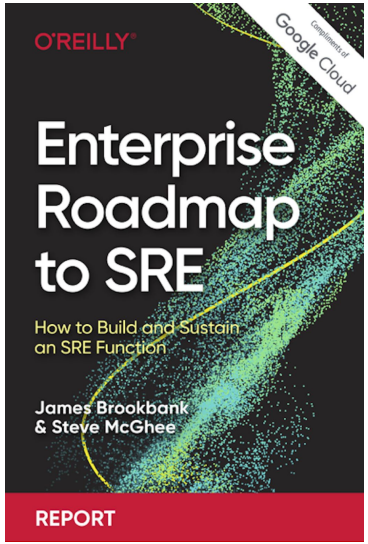
Google

Find Google SRE publications—including the SRE Books, articles, trainings, and more—for free at sre.google/resources.









Thank you

Ramón Medrano Llamas, Sr. Staff SysEng
@rmedranollamas