Google

The 1k SRE Project

Sublinear Scaling in Practice

Nikolaus Rath <nikolausr@google.com>
Presented at SREcon19, March 27th, 2019

Outline

1.	Who we are	Get to know my team
2.	Sublinear scaling	Maintaining more services with fewer people
3.	How we are doing	Doing well (so far) by automating everything
4.	Where we want to go	Maintaining *even more* services with *even less* people
5.	How we are getting there	Going beyond automation

About the team

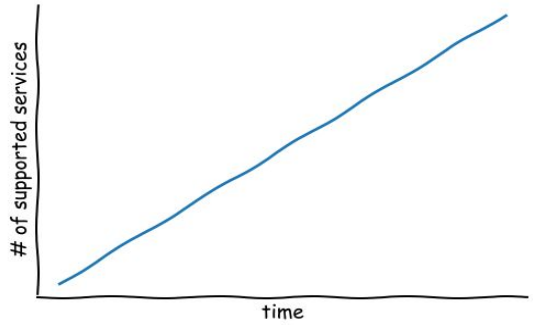
- 40 people in 2 locations
- Supporting ~400 services related to Google's advertising business
- Focused on operations work at large scale
- Typical workload:
  - 30% interrupt work
  - 20% service maintenance
  - 50% project work
- 2 oncall rotations, several service ownership groups and project groups

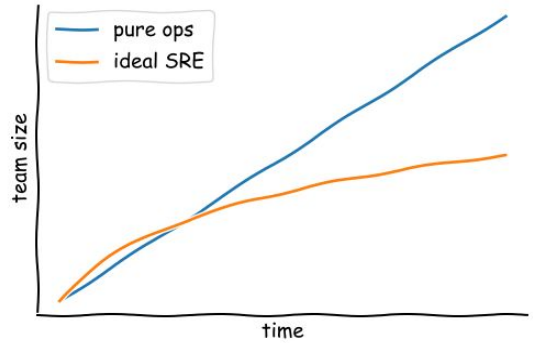
The [illegible] SRE Logo

Google

Sub-linear scaling

- Generally, *<#people> = n ✕ <#services>*
- For an SRE team, *n* should decrease over time
  → number of required SREs grows slower than number of supported services
- This is "sub-linear scaling"
- Not about how much work is done, but what kind of work is being done
- We decreased *n* by a factor of 2 over the last 3 years by increasing automation





Google

Current State of Automation:

Imperative vs Declarative Automation

Example: move service between datacenters, imperative

Recipe with pre-defined steps:

1. get quota in new datacenter (DC)
2. start process in new DC
3. reconfigure load-balancer to send traffic to new DC
4. stop process in old DC
5. release quota in old DC

Drawbacks:

- Needs clean starting point
- Difficult to handle failure
- Can't change course while automation is running
- Needs separate recipe for each task - duplication of steps

Current State of Automation:

Imperative vs Declarative Automation

Example: move service between datacenters, declarative

Intent: Service should run in datacenter "foo"

Continuous actuation:

According to intent:

- add/release quota
- start/stop processes
- reconfigure LB

permission to proceed

Sequencer with global ruleset:

- Running process requires quota
- Sending traffic requires running process

\- More complex

\+ No need for clean initial state or explicit recipes, handling failure is easy

Google

Current State of Automation:

What's automated?

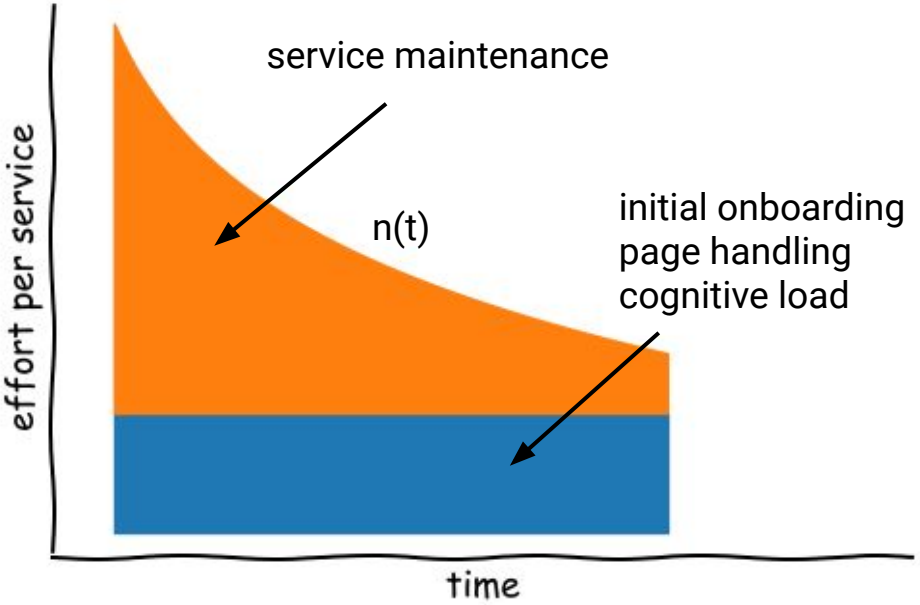
Intent defines:

- Redundancy
- Proximities
- Dependencies
- SLO
- Rollout policy
- Service specific configuration ("flags")

Automation:

- Starts, stops and moves service
- Selects DC, replica count, resource requirements (predictive)
- Sets up monitoring and alerting
- Runs load-tests
- Releases new versions
- Enforces policy

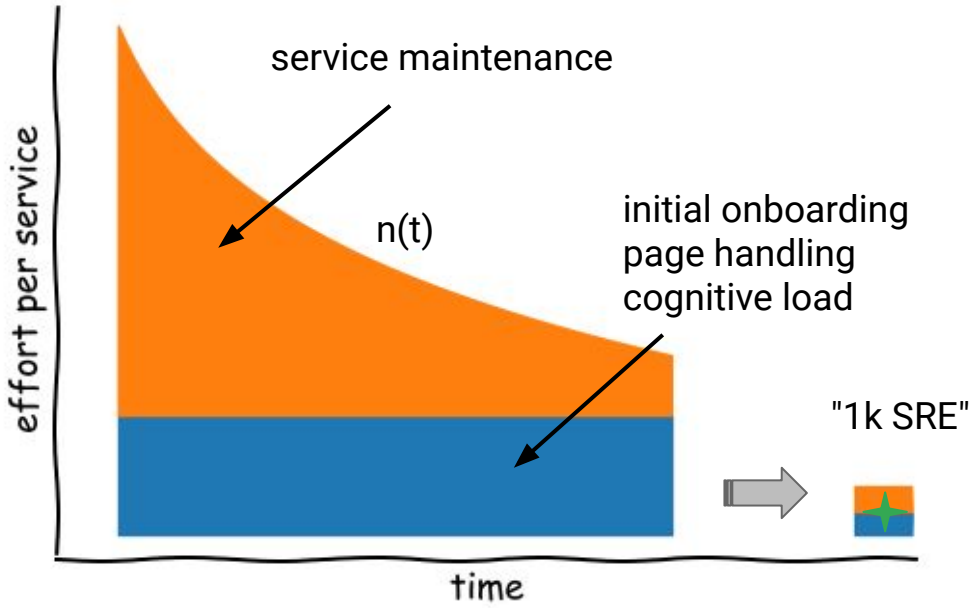
Automation only gets you so far...



- Gains from automation start to level off
- Progress is increasingly incremental
- Other factors start to dominate

Google

The 1k SRE Project: where do we want to go?



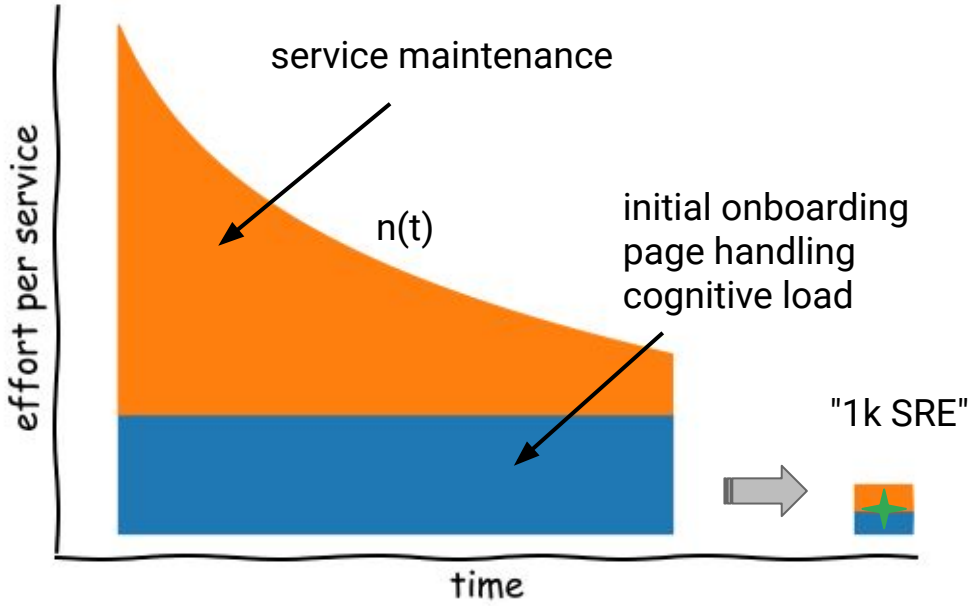
To go further, we set a moonshot goal:

Support 1000 services with the same team of 40 people.

- a different way to think about the problem space.

Google

The 1k SRE Project: what would it take?



- Automation as it is now
- Many pages are handled automatically
- Onboardings are done by developers
- Cognitive load is small
- SRE value is purely in production expertise

Google

Ongoing Projects:

Automatic, continuous Production Readiness Reviews

- Instead of checklist, have programmatic validators
- Validation failure reported as bug
- To onboard new service, run once and fix all the bugs
- Also, validators run continuously afterwards - regressions get noted!
- Already used by SRE
- Currently being polished for developer use (better documentation of policies)

Ongoing Projects:

Automated Incident Handling

- Annotate pages with additional information
  (e.g. "correlates with recent config push") - just became available
- Automatically redirect traffic if problem is localized
  - being tested
- Automatically get emergency quota - planning stage
- Automatically escalate pages - planning stages

Google

Summary

- We doubled the number of supported services by increasing automation
- We then got limited by pager load, cognitive load, and onboarding cost
- We are in the middle of tackling these by:
  - Automatically handling (some) pages
  - Dropping production readiness reviews in favor of continuous, automated policy validation
  - Shedding service specific knowledge, and becoming service-agnostic production experts

Questions?