System Reticulation Engineering

Laura Nolan, SRE, Google

Image: Alias 0591 via Flickr

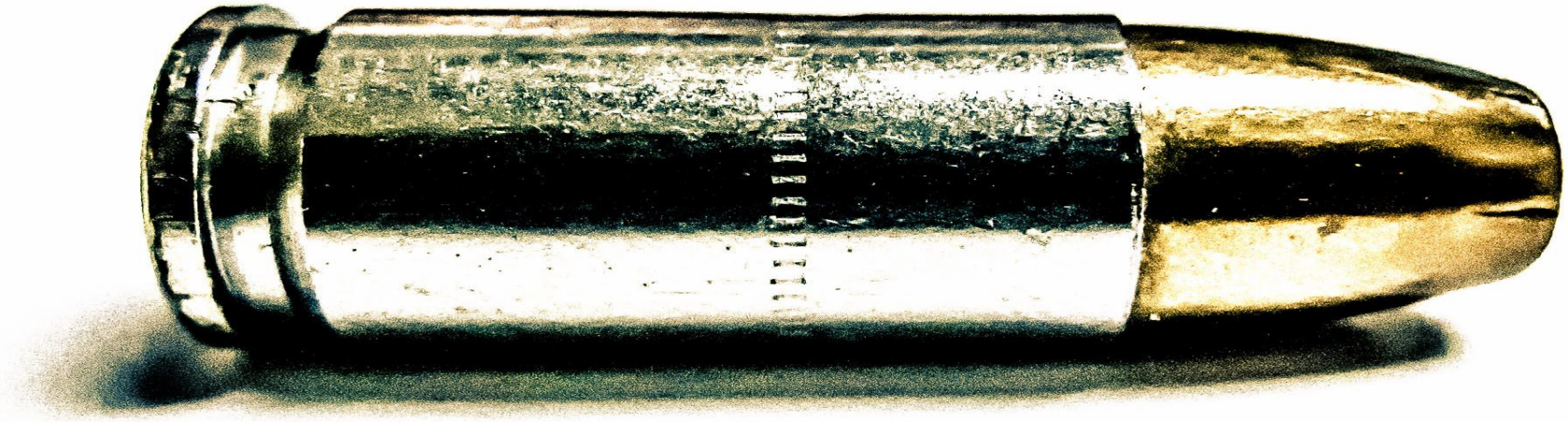
About Laura

- Degree in CS from Trinity College, recent-ish MSc in software engineering from UCD
- Principal software engineer in R&D for an Irish software company (Curam, now part of IBM Smarter Cities)
- Then specialised in software performance, working for an e-commerce company for a year
- Now: 4 years as an SRE at Google
    - Three working on data infrastructure
    - Not quite a year working on the network (so far)
    - Wrote one of the chapters of the O'Reilly SRE book
- Co-chair of the USENIX SRECon EMEA 2017 conference

Site Reliability Engineering

Hope is not a strategy. Engineering solutions to design, build, and run large-scale systems scalably, reliably, and efficiently is a strategy, and a good one.

Misconceptions about SRE



- **SRE is a fancy title for an operations team**
- **SRE is mostly about automation of common tasks**
- **SRE is a silver bullet for your operational issues**

Image: John Spade

The core functions of SRE

- Monitoring and metrics
- Emergency response
- Capacity planning
- Service turnup and turndown
- Change management
- Performance and efficiency

Software defined networking

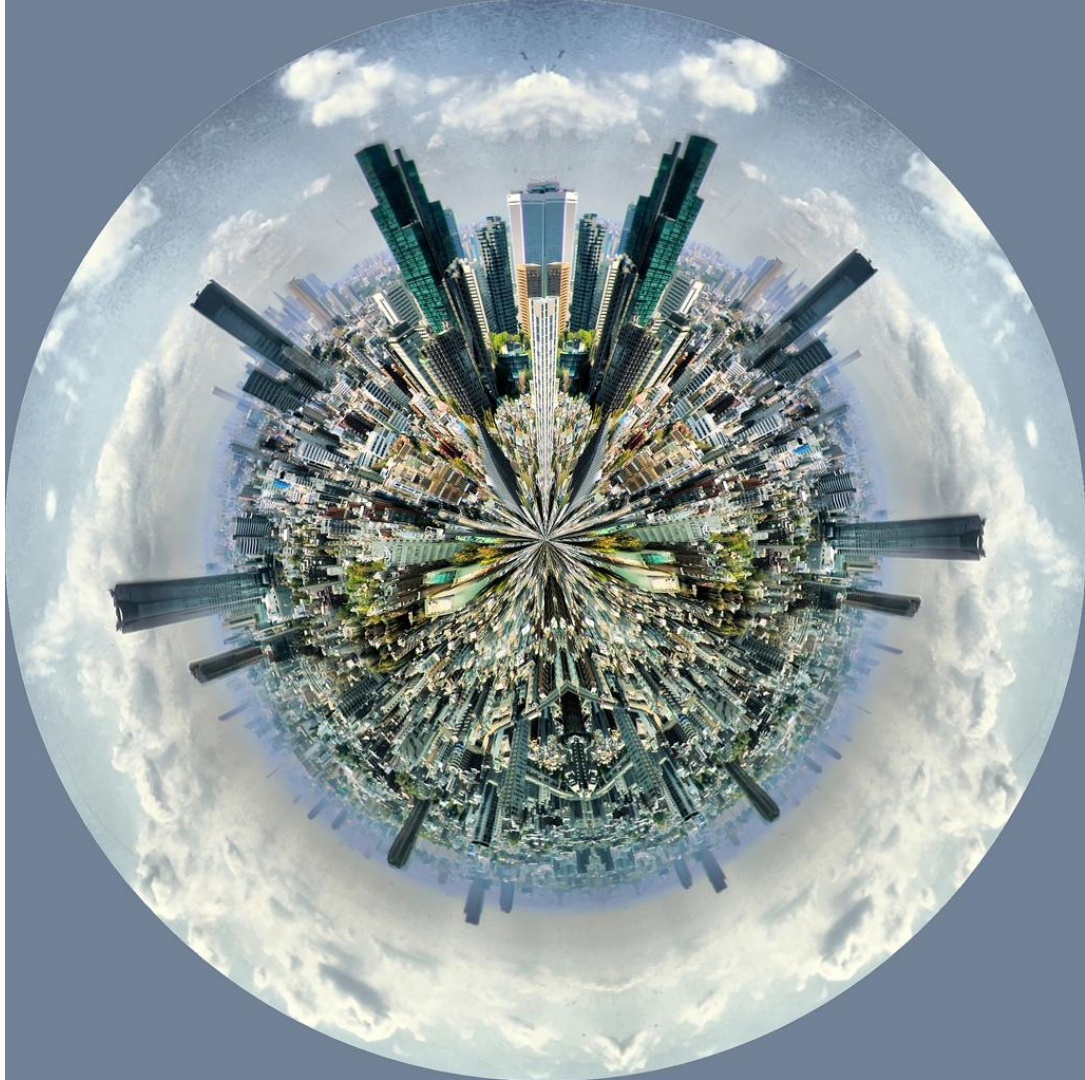


Image: Emran Kassim

Monitoring and Metrics

- SLOs
- Symptom based alerting
- Trust your alerting - don't watch graphs
- Alerts should be actionable
- Do long-term analysis of your metrics
- Use regular production meetings to track your metrics, short and long term

Emergency Response

- Incident management techniques
  - Teamwork and communication. No service is an island, especially the network
  - Being at the bottom of the infrastructure stack makes this vital
- Blame free postmortems
  - Root cause analysis
  - Engineering improvements based on postmortem action items
- Disaster planning and testing
- Wheel of misfortune
- Emergency playbooks
- Troubleshooting and debugging skills

Capacity Planning

- Understanding and modelling demands on your system
- Long term metrics gathered by monitoring tools to support capacity planning
- How does failure in one part of the system affect capacity demand elsewhere?
- Organic growth
- Launches

Service turnup and turndown

- Configuring new devices
- Turning up new links or peerings safely
- Decommissioning older infrastructure
- Software as well as physical infrastructure

Change Management

- Change management is both a huge source of human toil and very risky
- Outages are usually because something changed - risks have to be analysed
  - Software versions
  - Hardware
  - Configuration
- Safer change management involves testing, ideally automated, and canarying
- Large complex systems are constantly in flux
  - Breakage and repairs of physical infrastructure
  - Rollouts - software, configuration
- Most routine change management is best done by automation
- Doing this reliably and safely is one of the most challenging parts of SRE

Performance and efficiency

Design, development and engineering work to improve

- Isolation
- Scalability
- Throughout
- Latency
- Efficiency

Other elements of the SRE mindset

- Career expectations - advancement while working on production
- Time to spend engineering, not doing repetitive 'toil'
- Project focus, rather than ops focus
- Autonomy in prioritizing work
- Input into planned changes, and the authority to say no

Network failure domains

We try to abstract software services, and limit the interaction between parts to reduce complexity. This is harder with networks.

Image: n h via Flickr

Why SRE for the network?

Questions?

Some resources:

- "Site Reliability Engineering: How Google Runs Production Systems", O'Reilly Books, 2016
    - Full content at https://landing.google.com/sre/book/
    - The 'Monitor Lizard Book'
- USENIX SRECon EMEA 2017 will take place in Dublin from August 30th to September 1st - call for participation now up on usenix.org