SRE at a Startup: Lessons From LinkedIn

Craig Sebenik
Lead Infrastructure Engineer
Matterport
Who Am I?

- Scientist (computational chemistry)
- Sysadmin
- Developer (Perl, Java, C)
- Systems Engineer/SRE/Infrastructure Engineer/…
- Big Companies: Kodak, Kraft General Foods, NetApp, LinkedIn
- Small: 4info, Skyfire, Matterport
- Chef (real chef, not the software)
Does SRE work at a startup?
Outline

- What is SRE?
- LinkedIn
  - Early Times
  - SRE
- Matterport
  - Early Times
  - Changes
  - Lessons
What is “SRE”?

• Site Reliability Engineer (Engineering)
  • Started at Google ca. 2003.

• “SRE is what happens when you ask a software engineer to design an operations team.” [sre]

• Member of team focused on build, deployment, monitoring, etc.
  • But, entire team is still responsible.

• Each dev team is self-reliant.

• Still need someone to support the “core infrastructure”.
DevOps

Caution: These are my opinions!

• NO DEVOPS TEAM!

• DevOps is a paradigm not a job title.
  • “Everyone is a devops engineer.”

• “If you build it, you run it.” [in production]

• Werner Vogels, Amazon CTO [vogels]

• SRE is an implementation of the DevOps paradigm.
Outline

- What is SRE?
- LinkedIn
  - Early Times
- SRE
- Matterport
  - Early Times
  - Changes
  - Lessons
LinkedIn: The Early Times

- Joined in (late) 2010
- Single AppOps team for everything
- Central NOC, only 24/5
  - “Pager” passed around AppOps (SRE)
- Other ops teams: sysadmin, network, DBAs
LinkedIn: Early Releases

- Large releases, every other week
- Complex dependencies
- Done after work hours and took many hours to complete
- Complicated “release” branches
- “Release team” for changes
- Centralized (REST) service with configuration details
Outline

- What is SRE?
- LinkedIn
  - Early Times
  - SRE
- Matterport
  - Early Times
  - Changes
  - Lessons
LinkedIn: SRE Changes

- Changed name; aka “rebranded” (AppOps -> SRE)
- SRE broken up into dev specific teams
  - Eventually, moved to sit with dev team
  - Worked much closer with subset of devs
- Implemented Salt
- More coding
- Eventually, SRE teams for internal products
LinkedIn: DevOps

- Simplified “trunk” model*
  - No more feature branches
- Devs had access to configuration data
- Richer testing platform
- Devs were able to deploy to production
  - First to a canary, then entire cluster
- “A/B tests” for new features (aka “feature flags”)
- Devs involved in oncall (varied by team)
LinkedIn: Dev Self Service

- Automated code metrics
  - Dev would annotate code to produce metrics
  - No limits on number of metrics sent*

- inGraphs
  - Dashboards in YAML

- Self service alerts
Why is this important?

• Implementing a new paradigm is hard.
  • Need management support.

• LinkedIn changed a LOT of things allowing it grow.

• Self service is important.

• If LinkedIn can do it, so can your startup.
Outline

- What is SRE?
- LinkedIn
  - Early Times
  - SRE
- Matterport
  - Early Times
  - Changes
  - Lessons
aka “The Startup”
Who/What Is Matterport?

- 3D visualization of spaces.
  - Current focus: residential real estate
- Over 150 employees
- Based in Silicon Valley. (Offices in Chicago and UK.)
Matterport Technology

- Camera w/firmware
- Client (javascript, Unity)
- C++
- Python (DJango)
- Salt
- AWS
- Tons of third parties (“startup micro-economy”)
Outline

- What is SRE?
- LinkedIn
  - Early Times
  - SRE
- Matterport
  - Early Times
  - Changes
  - Lessons
“Ops” Work

- Single dev doing “operations” work
- He was the only one that knew entire stack
- Wrote tooling as needed
- Not all tools checked-in
- Several snowflake servers
- No Metrics
- Minimal monitoring
Releases

• All deploys by one person
• “Blue - Green” deployments (2 environments: active and dormant)
  • Lagging writes manually copied to new DB
• Little or no communication
• Hour+ downtime
  • Scheduled for late at night
• Hand-edits made to code in prod for hotfix
EVERYTHING IS BROKEN

FIX ALL THE THINGS!
OVERWHELMED?

When you’re waist-deep in tribbles, it’s a bit difficult to remember that your original objective was to guard the quadrotriticale.
Does this sound familiar?
SNAFU - Situation Normal …

- Startups start with “dev”
  - Engineers want to code, not deploy
- Start getting “real customers”…
- “Ops” work happens organically
- Just the challenge I was looking for!
Outline

- What is SRE?
- LinkedIn
  - Early Times
  - SRE
- Matterport
  - Early Times
  - Changes
  - Lessons
First Things First

• Prioritize fixing things
• Simplify
• File lots of Jira tickets
• Communication is important
• Get management buy-in
• All hands meeting: “We all own the site.”
Dickerson’s Hierarchy of Reliability

- Monitoring
- Incident Response
- Postmortem / Root Cause Analysis
- Testing & Release Procedures
- Capacity Planning
- Development
- Product
My Hierarchy of Needs

• Metrics and Monitoring
• Reproducible Builds (required SCM commit)
• Stable/Predictable Release Schedule
• More Communication (and documentation)
• Dev Ownership (includes testing)
• Build a Team
Metrics and Monitoring

I DON'T ALWAYS GRAPH MY DATA...

OH WAIT, YES I DO!
Metrics and Monitoring

- Datadog (statsd under the hood)
- Grassroots effort.
  - Show lead dev statsd and its docs
  - Create a few sample dashboards
  - *Everyone* has access (login)
- Expand access to monitoring system
- Simplify by removing unused systems (third parties)
SCM and Builds

- Simplify: everyone uses git
  - Github Enterprise
- Buildbot
  - Reproducible builds of C++ code
- Python code deployed directly from git
  - Not ideal… bigger fish to fry
- Automated tests with CircleCI
  - Eventually; automated builds its CircleCI
- Self-support!
SCM For Infra

• All salt changes are committed to git
• Simple unit tests run on salt changes
• Testing hosts for every member of infra
• Simplify salt code
  • Remove conditionals where possible
• Implement data structures
Release Improvements

• All non-production environments are free-game
  • Devs do “trip over each other” once in a while
  • They figured it out and adapted
• Prod releases are during business hours
  • If something goes wrong, dev needs to be available
More Release

- Backwards compatible
  - Devs updated the process for schema changes
- Release tickets in Jira
- Release plans in wiki
- Release planning meetings
Release Got Better

- Moved to every week
- Has become routine
  - “Smooth” release is the norm
  - Senior management has complete confidence in the process
Then Even More Betterer

• No more planning meeting
  • Everything in Jira and the wiki
• Slack channel and bots
• Feature flags
• Dev deploy directly to production
  • On their own schedule
Communication

- Slack everywhere
- Release channel
- “Outage” channel
- Bots integrated with automation
- Even the recruiters and marketing are using slack
- Docs on wiki
- Lots of Jira tickets
Dev Ownership

- Self service: Github and Circle
- Root access to all dev hosts
- Most have root to staging hosts
- Many have root on production
- Access to datadog, loggly, sentry, etc
Team

- Hired 2 more people
- Daily Standups
  - They also attend (some) product standups
- Weekly meetings
  - Ticket triage: all new, all “blockers”
- Kanban
Lots Left To Do

• Only some products have devs that can deploy to prod.
• Some developers submitting PRs to salt code
  • But, not enough.
• More *automation*, but only 1 *autonomous* system
• The rest would fill a dozen slides…
Outline

• What is SRE?
• LinkedIn
  • Early Times
  • SRE
• Matterport
  • Early Times
  • Changes
• Lessons
Lessons

• Patience: culture shifts take a long time
  • Devs *have* to be involved
• Don't be afraid of failure
• Respect your ancestors
• Start small and iterate
More Lessons

• Shared experience with failure is better than “preaching”
• Hiring is hard at all sizes
• Make decisions with data
• Demonstrate effectiveness to management
• Support of senior dev(s) necessary
• SRE is an implementation of DevOps
• Constant teaching/learning
Questions?

One more thing…
We’re hiring!

https://matterport.com/careers/
Related Talk

Closing Plenary: “SRE in the Small and in the Large”
Niall Murphy and Todd Underwood, Google
Constitution Ballroom
Q & A
(For real this time…)

• craig -AT- matterport -DOT- com
• Twitter: @craigs55
• Github: craig5
• https://www.linkedin.com/in/craigsebenik
• https://matterport.com/careers/
Notes

• [vogels] “A Conversation with Werner Vogels”
  http://queue.acm.org/detail.cfm?id=1142065

• [sre] “Site Reliability Engineering”
  Betsy Beyer, Chris Jones, Jennifer Petoff, Niall Richard Murphy
  http://shop.oreilly.com/product/0636920041528.do
Additional Resources

• Infrastructure as Code (Kief Morris)
  • http://shop.oreilly.com/product/0636920039297.do

• The Phoenix Project (Gene Kim, Kevin Behr, George Spafford)
  • https://www.amazon.com/Phoenix-Project-DevOps-Helping-Business/dp/0988262509

• The DevOps Handbook (Gene Kim, Patrick Debois, John Willis, Jez Humble)

• Continuous Delivery (Jez Humble, David Farley)
  • https://www.amazon.com/Continuous-Delivery-Deployment-Automation-Addison-Wesley/dp/0321601912