SRE at a Startup: Lessons From LinkedIn

Craig Sebenik
SREcon16
8 April 2016

Does SRE work at a startup?

Jumping ahead; Yes.
And it isn’t rocket science or brain surgery…

LinkedIn: The Early Years

- Joined in (late) 2010
- Central NOC, only 24/5
    - “Pager” passed around SREs
- Single AppOps (SRE) team for everything
- 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
  - “Auto-merge” to future branches
- “Release team” for changes
- Centralized service with configuration details

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
- Levels of tests: pre-commit, PCS, PCL
- Devs were able to deploy to production
  - First to a canary, then entire cluster
- “A/B tests” for new features
- Devs involved in oncall (varied by team)

LinkedIn: Dev Self Service

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

Matterport

Aka “the startup”


EVERYTHING IS BROKEN
FIX ALL THE THINGS!
memegenerator.net

“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
- Alpha/beta environments
  - “Lagging” edits 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

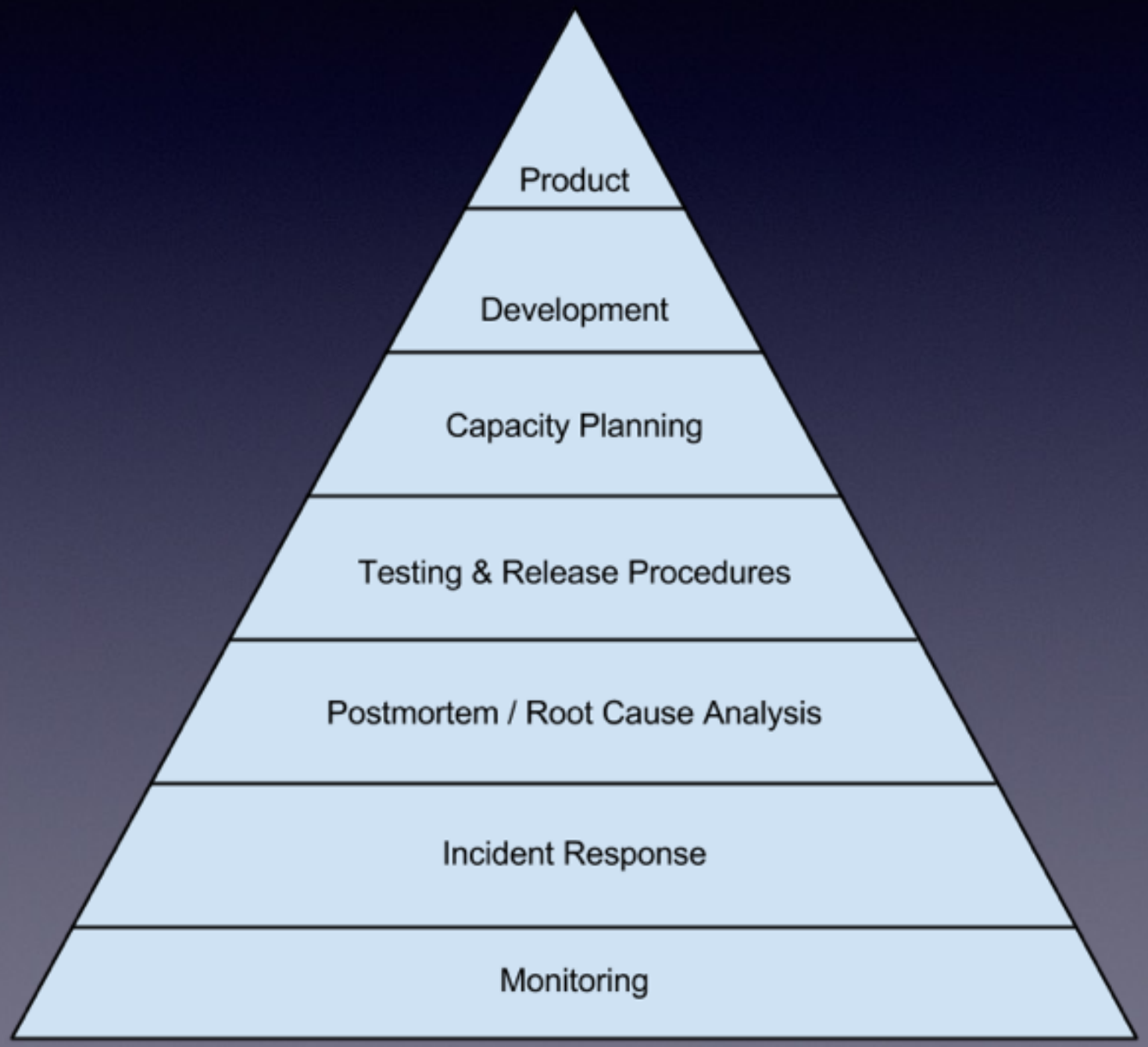
Not Atypical

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

First Things First

- Prioritize
- 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



My Hierarchy of Needs

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

Metrics and Monitoring

- Datadog
  - Show lead dev statsd and its docs
  - Create a few sample dashboards
  - *Everyone* has access (login)
- Expand access to monitoring system
- Remove unused systems

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
- Circle CI

SCM For Infra

- All salt changes are committed to git
- Simple 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-prod 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

More Release Improvements

- No more planning meeting
  - Everything in Jira and the wiki
- Slack channel
- Feature flags

Communication

- Slack everywhere
  - Release channel
  - “Outage” channel
  - 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 3 more people
- Each assigned to 2-4 products
- Standups
    - They also attend product standups
- Weekly meetings
    - Ticket triage: all new, all “blockers”
- Kanban… But, “reported” as sprints

Lots Left To Do

- Devs still cannot deploy to prod
- Do not have AWS access
    - Not even read-only
- But, some are submitting PRs to salt code
- The rest would fill a dozen slides…

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
- Names matter (don’t call it ops, nor devops)

More Lessons

- Shared experience with failure better than “preaching”
- Hiring is hard at all sizes
- Make decisions with data
- Demonstrate effectiveness to management
- Support of senior dev(s) necessary
- SRE and DevOps are cooperative

Questions?

- One more thing…

We’re Hiring!

Questions? (For real)

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