DevOps Foundations: Site Reliability Engineering

with Ernest Mueller

Introduction

Your fearless instructors are James Wickett (@wickett) and Ernest Mueller (@ernestmueller)
https://theagileadmin.com

Conclusion

Why wait? Dessert first, I say.

SRE Books (free online) – https://sre.google/books/

Release It! by Michael Nygard – https://pragprog.com/book/mnee2/release-it-second-edition

SREcon – https://www.usenix.org/conferences/byname/925

DevOpsDays – https://www.devopsdays.org/

Awesome Site Reliability Engineering –
https://github.com/dastergon/awesome-sre/blob/master/README.md

Always remember:

- Eliminate toil
- Product teams own their service in production
- Design reliability in
- Practice makes perfect

For information on monitoring, see *DevOps Foundations: Monitoring and Observability* in the LinkedIn Learning library.

Chapter 1: SRE Basics

Your job as a DevOp

The three pillars of DevOps

1. Continuous delivery
2. Infrastructure automation
3. Site reliability engineering

SRE is "what happens when you ask a software engineer to design an operations function."
—Ben Traynor

Keys to SRE by Ben Traynor at SREcon14 –
https://www.usenix.org/conference/srecon14/technical-sessions/ presentation/keys-sre

You aren’t Google or Netflix

Operations Maturity Model handout included – Operations-Maturity-Model.pdf

Lean IT – https://www.amazon.com/Lean-Enabling-Sustaining-Your-Transformation/dp/1439817561

Chapter 2: SRE Practice Areas

Release engineering

Chapter 8, “Release Engineering” by Dinah McNutt –
https://landing.google.com/sre/book/chapters/release- engineering.html

“The 10 Commandments of Release Engineering” presentation by Dinah McNutt at the RELENG conference in 2014:

Slides – http://releng.polymtl.ca/RELENG2014/html/presentations/The%2010%20Commandments%20 of%20Release%20Engineering%20-%20RELENG%202014%20-%20Google%20Slides.pdf

Video – https://www.youtube.com/watch?v=RNMjYV_UsQ8

1. Thou shalt use a source control system
2. Thou shalt use the right tool(s) for the job
3. Thou shalt write portable and low-maintenance build files
4. Thou shalt use a release process that is reproducible
5. Thou shalt use a package manager
6. Thou shalt design an upgrade process before releasing version 1.0
7. Thou shalt provide a detailed log of what thou hath done
8. Thou shalt canary
9. Thou shalt keep the big picture in mind
10. Thou shalt apply these commands to thyself

Feature flag options

- https://github.com/wix/petri
- https://www.togglz.org
- https://github.com/pda/flip
- https://launchdarkly.com
- https://www.split.io

Continuous Delivery by Jez Humble and David Farley – https://continuousdelivery.com

Change management

The Visible Ops Handbook – https://www.amazon.com/Visible-Ops-Handbook-Implementing-Practical/ dp/0975568612

1. Phase 1 – stabilize the patient, modify first response
2. Phase 2 – catch and release, find fragile artifacts
3. Phase 3 – establish repeatable-build library
4. Phase 4 – enable continuous improvement

Self-service automation

DevOps Audit Defense Toolkit – https://itrevolution.com/devops-audit-defense-toolkit

SLAs and SLOs

Tips for crafting SLAs and SLOs

- Involve the team
- Use simple metrics
- Skip absolute language
- Avoid marketing
- Use error budgets

Chapter 3, "Embracing Risk," SRE book – https://landing.google.com/sre/book/chapters/embracing-risk.html

Incident management

Incident Command System – https://en.wikipedia.org/wiki/Incident_Command_System

Incident Command for IT: What We Can Learn from the Fire Department by Brent Chapman – https://cdn.oreillystatic.com/en/assets/1/event/7/Incident%20Command%20for%20IT_%20 What%20We%20Can%20Learn%20from%20the%20Fire%20Department%20Presentation.pdf

Services

- PagerDuty – https://www.pagerduty.com
- Splunk On-Call – https://www.splunk.com
- OpsGenie – https://www.opsgenie.com

Open source

- Cabot – https://github.com/arachnys/cabot
- Openduty – https://github.com/ustream/openduty

Metrics

- MTBF – mean time between failures
- MTTA – mean time to acknowledge
- MTTR – mean time to resolve

Introducing postmortems

How Complex Systems Fail – https://www.adaptivecapacitylabs.com/HowComplexSystemsFail.pdf

The postmortem process

Postmortem template handout included – Postmortem Template.pdf

Troubleshooting

The scientific method – https://en.wikipedia.org/wiki/Scientific_method

Chapter 12, "Effective Troubleshooting," SRE book – https://landing.google.com/sre/book/chapters/effective- troubleshooting.html

DevOps Troubleshooting by Kyle Rankin – https://www.amazon.com/DevOps-Troubleshooting-Linux- Server-Practices/dp/0321832043

Performance engineering

High Performance Web Sites by Steve Souders – http://shop.oreilly.com/product/9780596529307.do

Open source tracking tools

- Zipkin – https://zipkin.io
- OpenTracing – http://opentracing.io
- JavaMelody – https://github.com/javamelody/javamelody/wiki

Performance engineering in a nutshell

- See the whole
- Instrument for success
- Write performant apps
- Continuous practice

Capacity and scalability

Scalability we define as short-term expansion.

Capacity planning we define as long-term expansion.

Diagonal scaling problem –

http://highscalability.com/strategy-diagonal-scaling-dont-forget-scale-out-and

Release It! by Michael Nygard – https://pragprog.com/book/mnee2/release-it-second-edition

Release It! capacity patterns

- Pool connections
- Use caching carefully
- Precompute content
- Tune the garbage collector

Distributed design

Reliability math: Overall reliability is the chance of failure of any one tier, multiplied by the chance of failure of the other tiers (assuming the tiers operate independently, which may not be the case). So three tiers that are 99% available are .99 * .99 * .99 = .97.

If you have redundancy in a tier, its chance of failure is instead the chance of all its component members failing. That's 1 (100%) minus the unreliability of each tier (1%) multiplied by each other. So the chance that a load-balanced three-server tier with each server being 99% available jumps to 1 – (1 – .99) (1 – .99) (1 – .99) = 0.999999.

Retries and so on can also be used to make the math come out in your favor.

The Twelve-Factor App – https://12factor.net

I. Codebase
One codebase tracked in revision control, many deploys

II. Dependencies
Explicitly declare and isolate dependencies

III. Config
Store config in the environment

IV. Backing services
Treat backing services as attached resources

V. Build, release, run
Strictly separate build and run stages

VI. Processes
Execute the app as one or more stateless processes

VII. Port binding
Export services via port binding

VIII. Concurrency
Scale out via the process model

IX. Disposability
Maximize robustness with fast startup and graceful shutdown

X. Dev/prod parity
Keep development, staging, and production as similar as possible

XI. Logs
Treat logs as event streams

XII. Admin processes
Run admin/management tasks as one-off processes

Release It! by Michael Nygard – https://pragprog.com/book/mnee2/release-it-second-edition

Deliberate adversity

Chaos Monkey –
https://medium.com/netflix-techblog/netflix-chaos-monkey-upgraded-1d679429be5d

Netflix Simian Army – https://github.com/Netflix/SimianArmy

Gauntlt – http://gauntlt.org

Gauntlt repo – https://github.com/gauntlt/gauntlt

Security Testing – https://www.linkedin.com/learning/devsecops-automated-security-testing?trk=lynda_redirect_learning

Chapter 3: SRE Organization

Organizing SREs

Conway's law – https://en.wikipedia.org/wiki/Conway%27s_law

The softer side of SRE

Time Management for System Administrators by Thomas Limoncelli –
http://shop.oreilly.com/product/9780596007836.do

Pragmatic Thinking and Learning by Andy Hunt –
https://pragprog.com/book/ahptl/pragmatic-thinking-and-learning