How could small teams get ready for SRE

Zehua Liu

Zendesk Singapore

SRECon Asia / Australia 2017

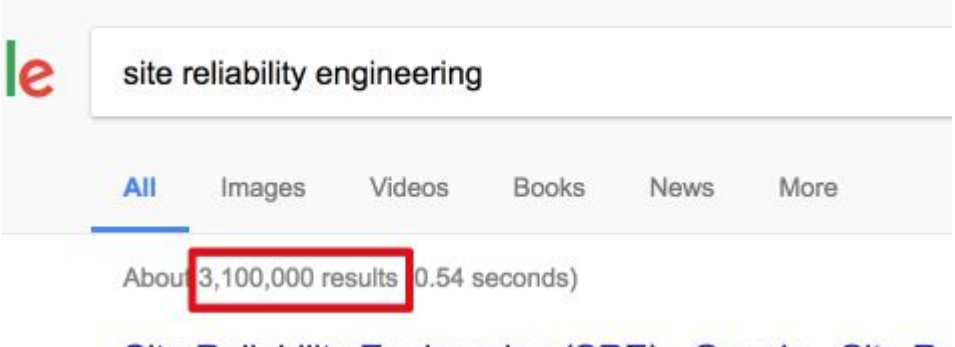
22 May 2017

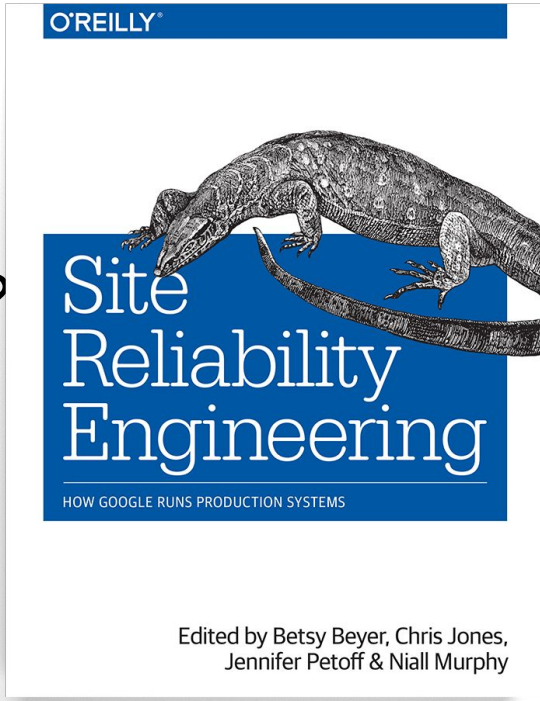
About Me

- Zehua Liu
  - With Zendesk Singapore since 2015
  - Worked at startups at various stages (Atlassian, mig33, Circos Brand Karma)
  - Leads the tooling team at Zendesk SG

The Problem

- Small teams?
  - Small companies
  - Small but independent teams within big companies
- How do small teams get started with SRE?
- SRE book: 34 chapters, 500+ pages
- Abundant amount of info online





The Problem

- A case study from Zendesk Singapore
  - A small team within a bigger company
  - Develops the Chat product
  - 10 engineers in SG vs hundreds globally
- How did we get into SRE

History - Zendesk Singapore

zopim

Local startup founded in 2008



Acquired by Zendesk in 2014

chat

Rebranded as Zendesk Chat in 2016

The Problem - Team Growth



Acquired by Zendesk
in 2014

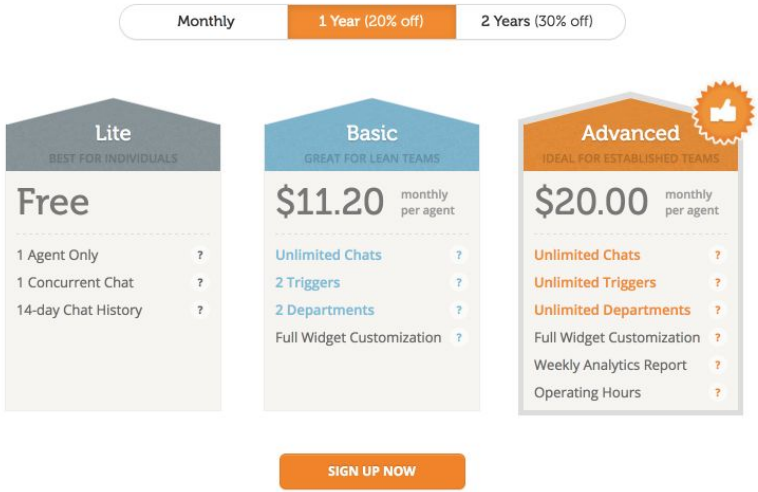
10 engineers
2 teams

chat

2017

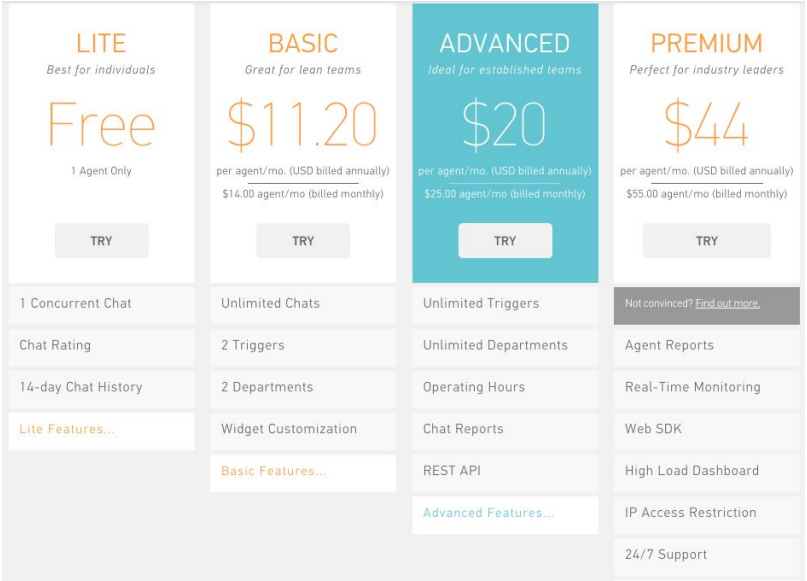
~40 engineers
8 teams

The Problem - Product Changes



2014

Mostly SME facing features
Standalone product
Private company, no compliance at all



2017

More Enterprise features
Integration with other Zendesk products
IPO-ed, compliance
… ...

chat

The Problem

- Growth in team size and changes in product focus
- ⇒ Issues with productivity and site reliability
  - Onboarding new hires
  - Harder to do deployment
  - More incidents
  - Failure to meet stricter SLA from enterprise customers

Solution?

- Started a series of engineering initiatives
  - Engineering dedication
  - Simple deployment
  - Maintainable staging environment
  - Easy-to-use development environment
  - Automated integration tests
  - …

Solution?

- Started a series of engineering initiatives
  - Engineering dedication
  - Simple deployment
  - Maintainable staging environment
  - Easy-to-use development environment
  - Automated integration tests
  - …

Engineering dedication - Past

- No dedicated engineering resources
  - Two technical co-founders leading feature development
  - Engineering issues fixed when encountered, ad-hoc

Engineering dedication - Now

- Hired a (real) Director of Engineering
- Someone who could spend time on *engineering* issues instead of *product* issues





chat

Engineering dedication - Now

- Hired more engineers
- Tried various ways to clear technical debts
  - A full engineering quarter in 2015
  - Engineering weeks
- Spawned engineering teams out of feature teams
  - Dedicated to the backend stacks
  - Supporting feature work
  - Spending time on scalability and reliability

Engineering dedication - Now

- Built a team dedicated to *tooling* and *reliability*
  - No feature work development for the team
  - Works closely with DevOps
  - Developing tools to:
    - Support developers
    - Improve production reliability





chat

Solution?

- Started a series of engineering initiatives
  - Engineering dedication
  - Simple deployment
  - Maintainable staging environment
  - Easy-to-use development environment
  - Automated integration tests
  - …

Simple deployment - Past

- Manual deployment
  - ssh and copy and paste scripts
  - Error prone
- Only few engineers could do it
- Paired deployment
  - Paired staring at the same terminal screen

Simple deployment - Past

- Gradually became a bottleneck as:
  - More engineers coming onboard
  - More frequent deployments are required

Simple deployment - Now

- Adopted Samson, a deployment tool built by Zendesk
  - https://github.com/zendesk/samson
- Consolidated the deploy scripts into Samson
- Any engineer could deploy
- Compliance (SOC2)
- Took one quarter to migrate the deployment of almost all apps

Simple deployment - Now

- Adoption of Samson deployment was initially slow
- Eventually picked up
- Once engineers get used to it, they stick to it
  - Easy of use
  - Less error prone

Simple deployment - Now

- Engineers took over ownership
  - Maintaining the deployment scripts
  - Adding deployment for new apps
  - Tooling team stepped back into consultation mode



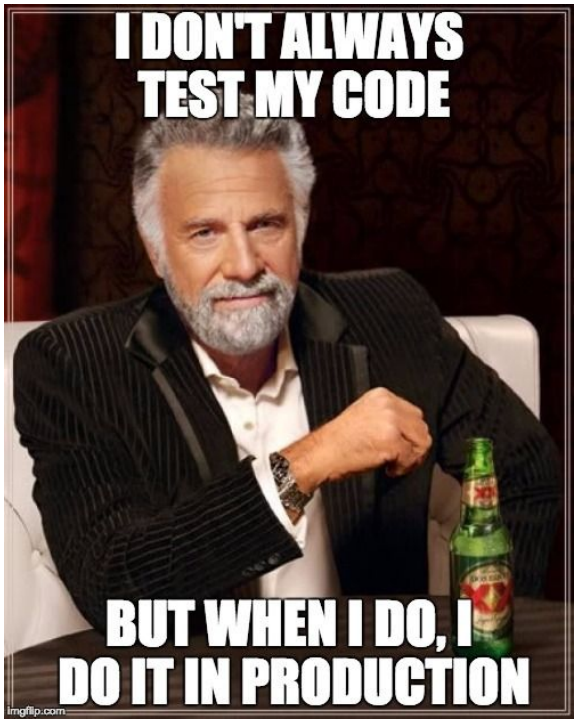
Solution?

- Started a series of engineering initiatives
  - Engineering dedication
  - Simple deployment
  - Maintainable staging environment
  - Easy-to-use development environment
  - Automated integration tests
  - …

Maintainable Staging Environment - Past

- No usable and sharable staging environment
- One previous attempt to setup a staging
  - Not mimicking production
  - Not maintainable
- Zendesk has a staging environment for the rest of the apps, but not for Chat
- Where to test changes after local development?!
- How did we test the Zendesk / Chat integration??!!

Maintainable Staging Environment - Past





- How did we test the Zendesk / Chat integration??!!

chat

Maintainable Staging Environment - Now

- Spent two quarters working with DevOps to setup a staging environment
  - Reused (almost) the same configuration management scripts as the production ones
  - Updated the codes to removing hardcoding of references to production domain names / URLs
- A challenging task for the tooling team:
  - Full stack knowledge of the apps
  - Complete knowledge about Zendesk / Chat integration

Maintainable Staging Environment - Now

- Integration with Samson, the deployment tool
  - Deploy to staging, test, then deploy to production
- For the first time, we don’t have to test on production
  - Not even for the Zendesk / Chat integration

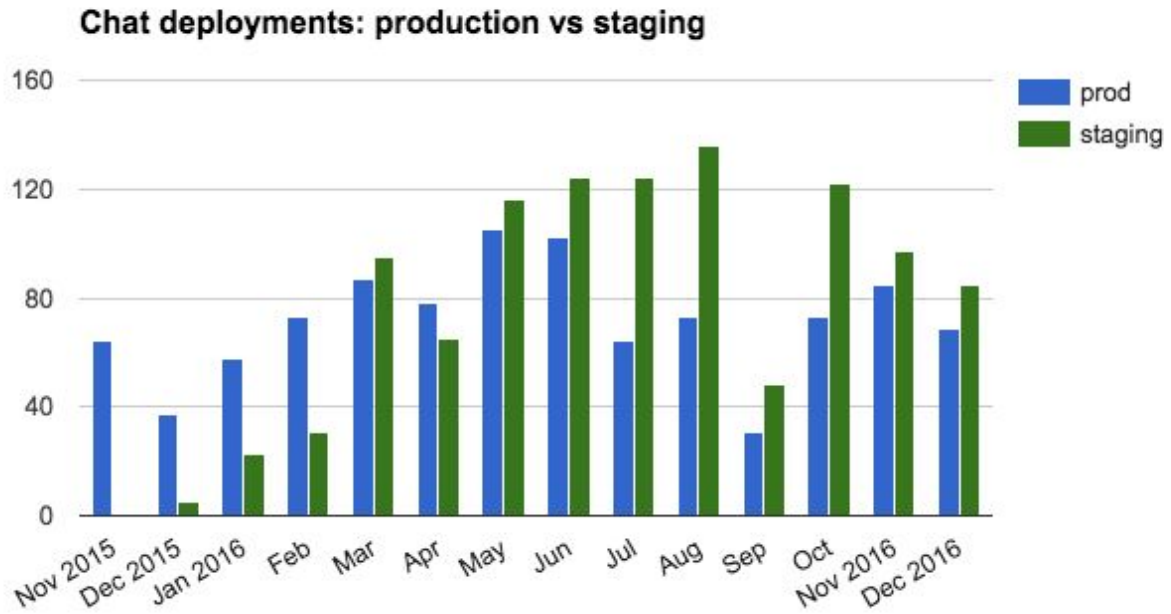


Maintainable Staging Environment - Now

- Initially, staging was not used a lot
  - Only by QA
  - No buy-in from engineers, seen as process / trouble
  - Engineers continued to deploy straight to production
- A gradual education process:
  - Slowly, we showed engineers that bugs on production could have been caught by staging
  - Sometimes, staging was the only place (other than production) to test some changes

Maintainable Staging Environment - Now

- Buy-in from engineers
  - Some numbers from Nov 2015 to Dec 2016



Maintainable Staging Environment - Now

- Engineers took over ownership
  - Update their codes to work on staging
  - Work with DevOps on new configuration changes
  - Tooling team stepped back into consultation mode



Solution?

- Started a series of engineering initiatives
  - Engineering dedication
  - Simple deployment
  - Maintainable staging environment
  - Easy-to-use development environment
  - Automated integration tests
  - …

Easy-to-use development environment - Past

- Every old guy has his/her own setup
- Hard for new engineers
- Backend team started dockerizing the full stack
- No integration with the dev env of the rest of Zendesk
  - Hard to test integration in dev

Easy-to-use development environment - Now

- Spent one quarter polishing the docker dev env
- Automated the VM setup with vagrant
- Made the Chat docker env work with the docker env of the rest of Zendesk
  - Possible to test integration in docker

Easy-to-use development environment - Now

- Again, engineers proactively owned it
  - Kept existing images up to date
  - Created new docker images for new apps
  - Tooling team stepped back into consultation mode



Easy-to-use development environment - Now

- The basic development / deployment pipeline:
  - Develop locally ⇒ test in docker ⇒ deploy to staging ⇒ test on staging ⇒ deploy to production



Solution?

- Started a series of engineering initiatives
  - Engineering dedication
  - Simple deployment
  - Maintainable staging environment
  - Easy-to-use development environment
  - Automated integration tests
  - …

Automated Integration Tests - Past

- There were unit tests and integration tests
- Some browser tests were written by QA
  - Tests are flakey
  - Not easily reusable
  - Not used in the deployment pipeline
- Limited QA resources
  - Only 1 QA engineer at that time

Automated Integration Tests - Now

- Spent one quarter improving the browser tests and integrating them into the deployment pipeline
- (Tried to) make the test results reliable
- Improved Samson / Jenkins integration

Automated Integration Tests - Now

- Completed the development / deployment pipeline:

develop locally ⇒ test in docker ⇒ ***PR*** ⇒ auto test in Travis ⇒ ***deploy to staging*** ⇒ auto test on staging ⇒ result notification ⇒ ***deploy to production*** ⇒ auto test on production ⇒ result notification



chat

Solution?

- Started a series of engineering initiatives
  - Engineering dedication
  - Simple deployment
  - Maintainable staging environment
  - Easy-to-use development environment
  - Automated integration tests
  - …

Misc initiatives

- Change velocity, weekly release vs continuous deployment
  - Beta / canary first, dogfooding
  - Tools to support progressive rollout
- Incident management and postmortem

Mapping back to SRE

Ensuring a Durable Focus on Engineering

As already discussed, Google caps operational work for SREs at 50% of their time. Their remaining time should be spent using their coding skills on project work. In

- Dedicated engineering team for tooling & reliability

CHAPTER 5

Eliminating Toil

- Simple deployment
- Automated integration tests

CHAPTER 8

Release Engineering

- Simple deployment
- Maintainable staging environment
- Release process
- Easy-to-use development environment
- Automated integration tests

CHAPTER 17

Testing for Reliability

- Automated integration tests
- Maintainable staging environment
- Easy-to-use development environment

Lessons Learnt

- Don’t have to do SRE for the sake of doing SRE
  - You don’t even need an SRE team!
- Address your needs / issues as you grow and SRE will come to you
- Preventive vs reactive measures
  - Easier to accomplish but less measurable
  - SRE seems to be more reactive?

Lessons Learnt

- It takes time to change engineers' mindset
  - Bottom up instead of top down
  - Let the tools speak for themselves
    - Once the product team / engineers see the benefits, they will use it
    - Once they use it, they will also own it
  - From *move fast and break things* to reliability conscious

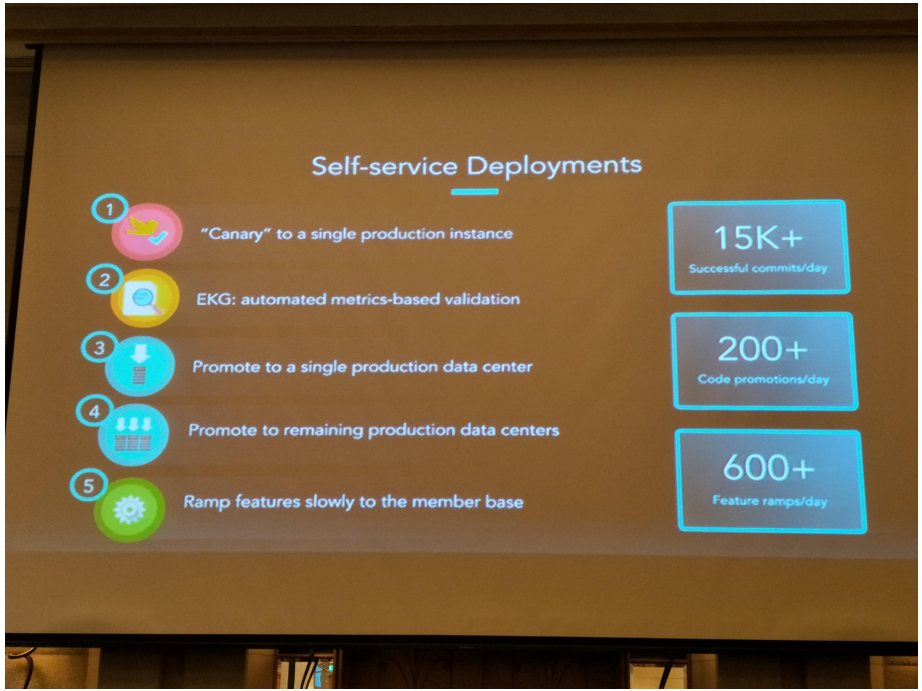
Lessons Learnt

- Dedicated tooling team really helped
  - 4 person team in one year
- Tooling team needs to know almost the full stack
- Once the tools are working, let the engineers own them
  - Engineers help maintain the tools
  - Tooling team steps back and serves as consultants



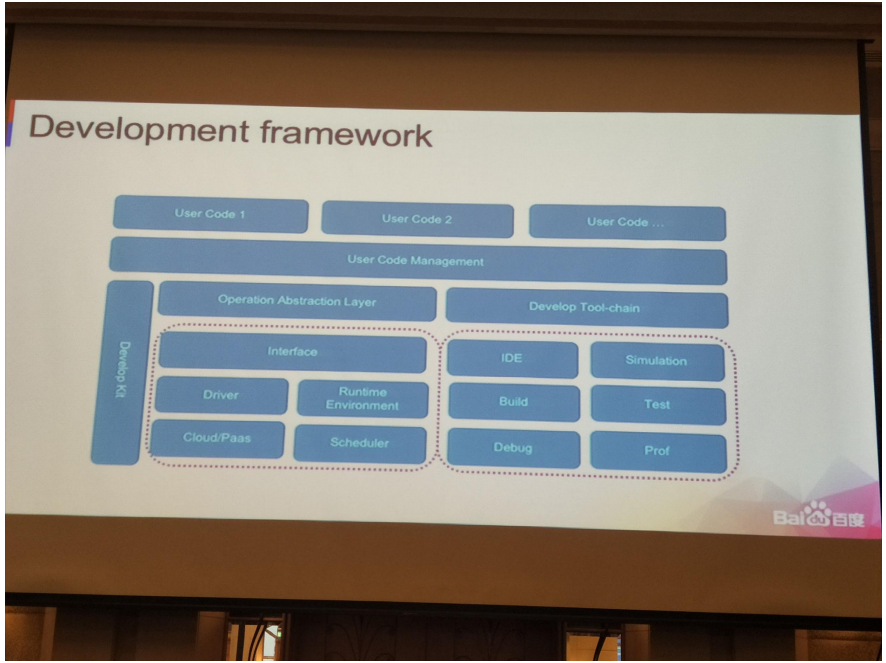
Small teams vs Big guys

develop locally ⇒ test in docker ⇒
PR ⇒ auto test in Travis ⇒
deploy to staging ⇒ auto test on staging ⇒ result notification ⇒
deploy to production ⇒ auto test on production ⇒ result notification



Small teams vs Big guys

develop locally ⇒ test in docker ⇒ ***PR*** ⇒ auto test in Travis ⇒ ***deploy to staging*** ⇒ auto test on staging ⇒ result notification ⇒ ***deploy to production*** ⇒ auto test on production ⇒ result notification



chat

Small teams vs Big guys

- Haven’t touched:
  - Data centre operations
  - Hardware
  - SLO
  - Monitoring
  - Incident management
- We don’t even have a proper SRE team
- But we are getting ready for those!

Small teams,
Identify reliability issues,
Address them,
and get ready for SRE!

Questions?