CHICAGO
INTERNATIONAL
SOFTWARE DEVELOPMENT
CONFERENCE 2016

goto; conference

The Idiot's Guide to Quashing MicroServices

Hani Suleiman

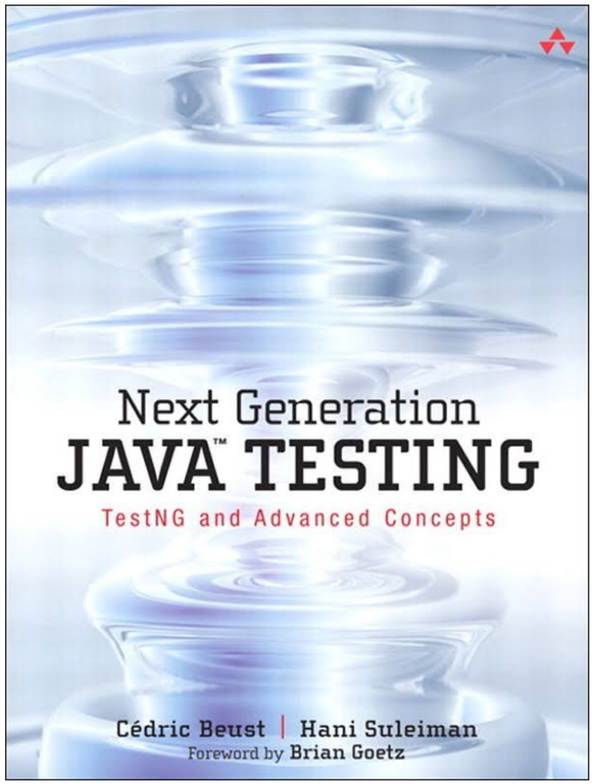
follow us @gotochgo

Conference: May 24th-25th / Workshops: 23th-26th

- The Promised Land
- Welcome to Reality
  - Logging
  - HA/DR
  - Monitoring
  - Provisioning
  - Security
  - Debugging
  - Enterprise frameworks
- Don’t Panic

WHOAMI

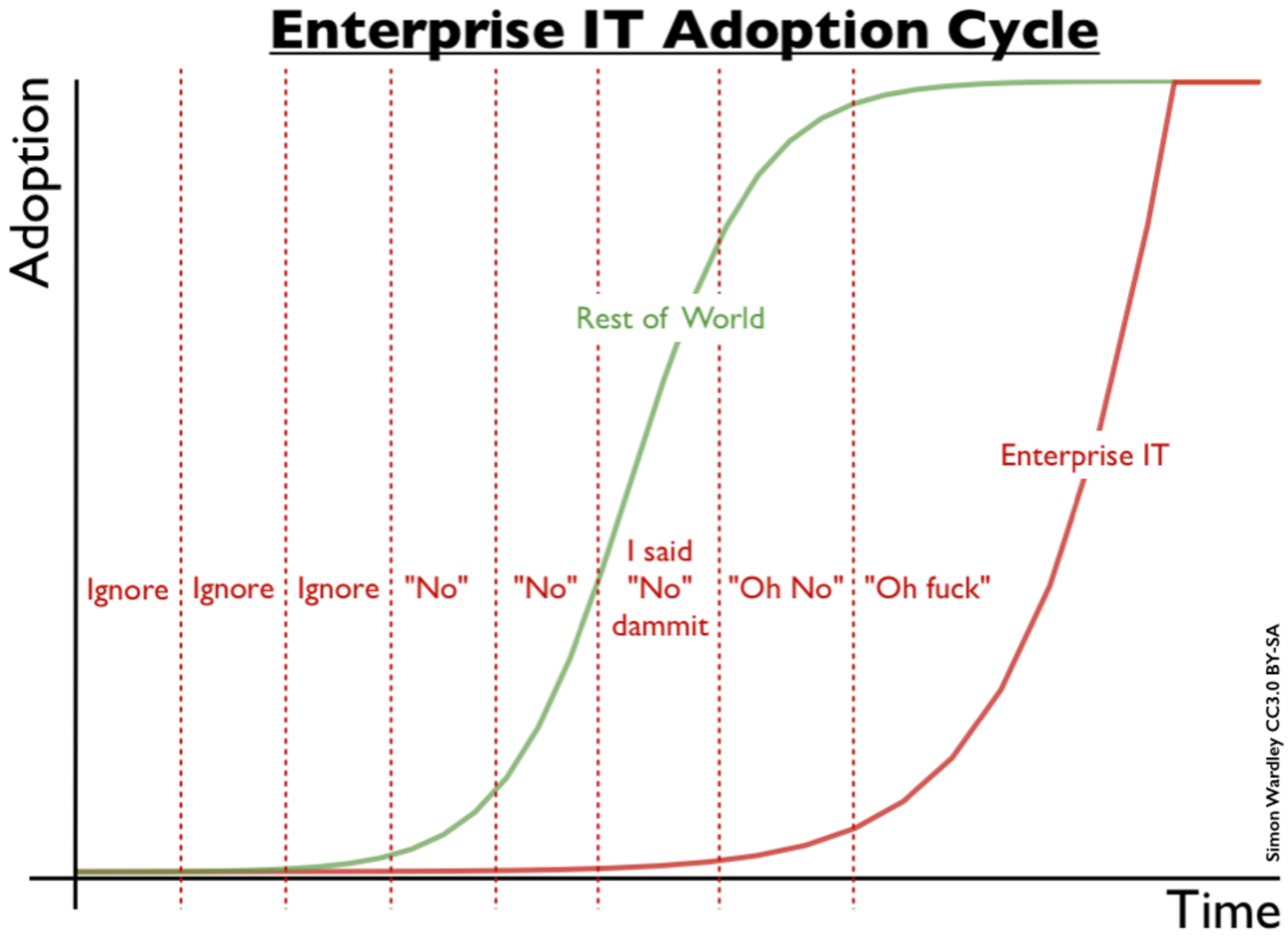
- I wrote a book about testing



- I had a blog
- CTO
- Twitter @bileblog

WHOAMI

- What makes me an expert?
  - I’m not
  - I’ve screwed this up many times so you don't have to
  - I work with big banks



AN AWKWARD CONVERSATION

Yes, this really happened. The following movie is basely on real life events at a very large investment bank. I was one of the people getting shouted at.

https://youtu.be/Q65Q1Mi5Jtw


We broke our application
into itty bitty microservices

TL:DR

- If you hate microservices, use these points to ensure you never have to use them
- If you love microservices, make sure you have sensible answers to these points before suggesting adoption

THE PROMISED LAND

- Infinite linear scalability
- Well defined contracts
- Unix philosophy (do one thing well and play nice with others)
- Faster releases
- Team scalability
- ‘Everything fits in your head’

WELCOME TO REALITY

- Something went wrong, where do I look?
- How do I reproduce an error?
- The unix analogy is crap
- The audit guys found out we're using unauthenticated http between services. They are first sad, then angry
- I'm only allowed to use our enterprise DB for storage
- I need a new server and I'm told if all goes well it'll be here in 2 months, then another 2 to get it production ready

LOGGING

- Many services logging to different files
- How do we correlate logs?
- Where do I start looking?

LOGGING

- You should never need to log in a server to view logs
- You must aggregate
- Logs must be semi-structured
- Consider ELK stack if you're poor, Splunk if you have budget


HIGH AVAILABILITY / DISASTER
RECOVERY

HIGH AVAILABILITY/DISASTER RECOVERY

- Differing SLAs across services
- Different loads
- Trade off between scalability and performance

HIGH AVAILABILITY/DISASTER RECOVERY

- You probably need a service locator. Sorry
  - Consul
  - Zookeeper
  - etcd
- State is the root of all evil
- Use a modern distributed messaging system like Kafka
  - Consumers pick their own rates
  - Recovery from a point in time
  - Easy replay


MONITORING

MONITORING

- Many more processes to keep track of now
- Knock-on impact of related services
- Multiple SLAs
- Proactive vs reactive

MONITORING

- Need both aggregate and drill down views
- Once a minute is not enough
- Correlation with logs
- Historic, not just point in time


PROVISIONING

PROVISIONING

- Too many services to rely on manual deployment unless you have a meat cloud.
- Complex interactions/orchestration
- Different deployment profiles
  - Deploy just this service
  - Deploy these related services
  - Deploy the world

PROVISIONING

- Agile hardware moves from nice to have to must have
- Decouple configuration from application
- Use a configuration management solution
  - Puppet
  - Chef
  - Ansible
- Do not roll your own


SECURITY

SECURITY

- I have to worry about security IN my application now?!
- Multiple data stores
- I've gone polyglot, and now I need to learn about security across so very many frameworks and libraries
- That's ok, I can just google for best practices

SECURITY

- Principal propagation
- Service authorization/authentication
- Secure communications
- Logging and monitoring, again

DEBUGGING

FAILURE MODES

- What happens if a service is down?
- What happens if a service is slow?
- What happens if I have ‘at-least-once’ semantics?
- What happens if a service is wrong?

DEBUGGING

- Debugging a distributed system is stupidly difficult
- Two services both think they're right, together they're wrong
- Fatal failures are awesome
- Non-fatal failures need tooling to reproduce state
  - Journal of state events needed to get us to the bad state
  - Dynamic instrumentation to interrogate a naughty service

ENTERPRISE FRAMEWORKS

ENTERPRISE FRAMEWORKS

- Enterprises love frameworks but individuals love libraries, guess who wins?
- Frameworks impose constraints on your service
  - You must use JMS and MQSeries
  - You must use Oracle
  - Domain objects must be modeled by our modeling czar
  - API contracts must be approved by the Subcommittee for API Contract Approvals
  - ‘Insert business logic here’
- Too easy to fall into shared state

DON'T PANIC

- You don't need all this stuff to get started
- You do need to keep it in mind and be able to make sensible noises when asked
- Most of these concerns are orthogonal, different teams can work on them in parallel
- There are frameworks that simplify some of these things (Kubernetes and friends)

CHICAGO
INTERNATIONAL
SOFTWARE DEVELOPMENT
CONFERENCE 2016

goto; conference

Please

Remember to rate this session

Thank you!