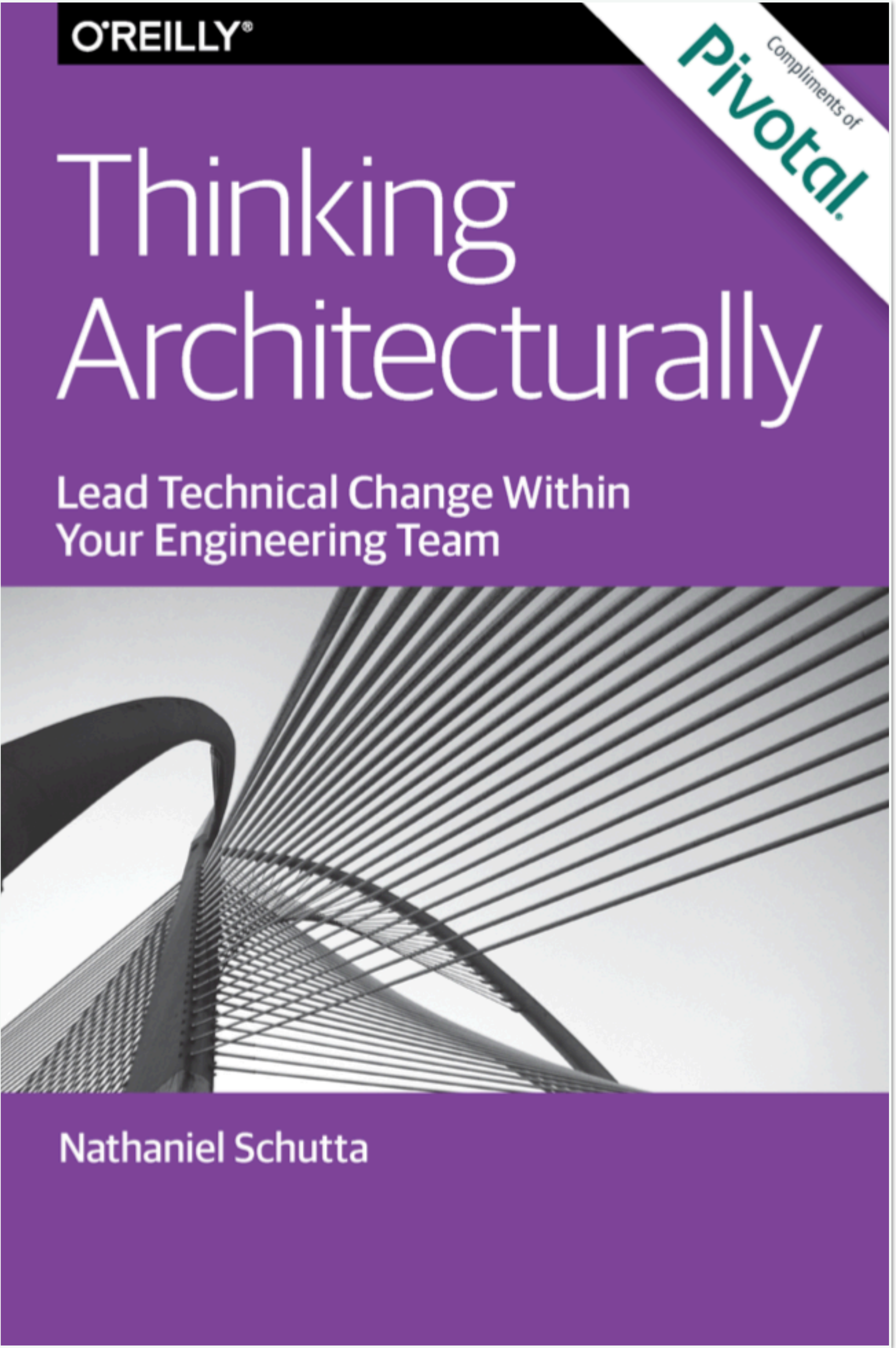
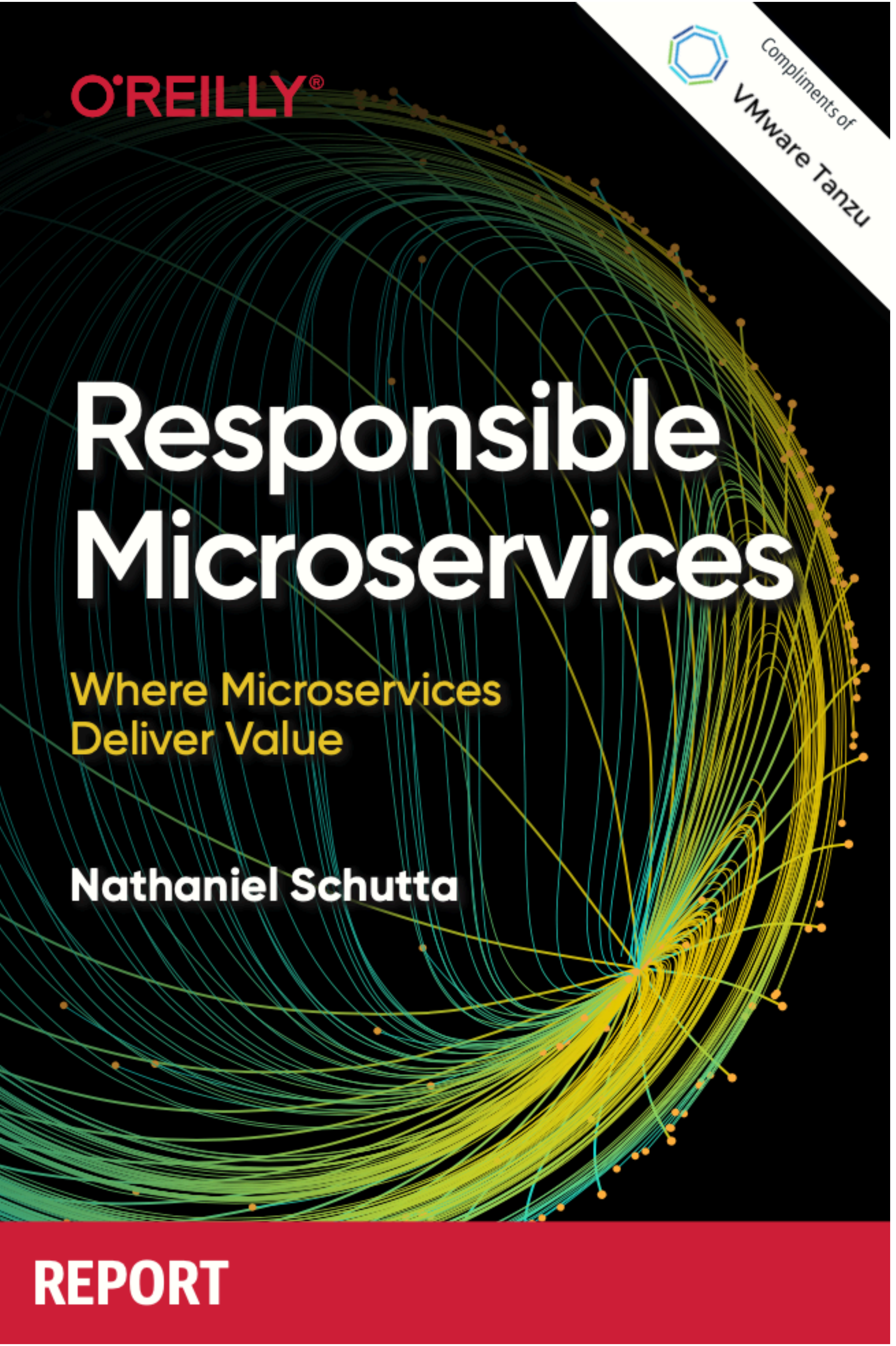
Cloud Native Culture

Nathaniel Schutta
@ntschutta
ntschutta.io



https://tanzu.vmware.com/content/ebooks/thinking-architecturally



https://tanzu.vmware.com/content/ebooks/responsible-microservices-ebook

Ah “the cloud!”

So. Many. Options.

Microservices. Modular monoliths.

Container all the things?

What about serverless?

Functions. As a Service.

Did someone say Polycloud?

https://www.thoughtworks.com/radar/techniques/polycloud

How do we make sense of all this?!?

There are real engineering issues to overcome.

Many believe in magic sparkle ponies…

But technology isn’t the only thing we are changing.

Our culture will have to evolve too.

CULTURE

Culture? What does that have to do with technology?

Ignore culture at your own peril.

Every company has a culture.

What is yours like?

"If they don't change the paint once in a while…"

Culture gets formed *very* early in a company’s existence.

Hire, attract, retain based on culture.

During hiring, often
talk about “culture fit.”

Culture informs everything we do.

In both small and large ways!

The battle over jeans…

Culture is where good ideas go to die.

It is *really* hard to change.

People who have risen to power or excelled in an org…

Have often gamed that culture.

Any change to that culture is a potential threat to position.

“Middle management mafia.”

Most dangerous six words?

“That’s how we’ve always done it”.

"It is difficult to get a man to understand something when his salary depends upon his not understanding it."

–Upton Sinclair

“[T]he innovator has for enemies all those who have done well under the old conditions, and lukewarm defenders in those who may do well under the new.”

–Niccolò Machiavelli

The Curse of Culture.

https://stratechery.com/2016/the-curse-of-culture/

“culture is one of a company’s most powerful assets right until it isn’t…”

–Ben Thompson

Be aware of your culture.

How do we change culture?

"How did you go bankrupt?"
Bill asked.

The Sun Also Rises by Ernest Hemingway

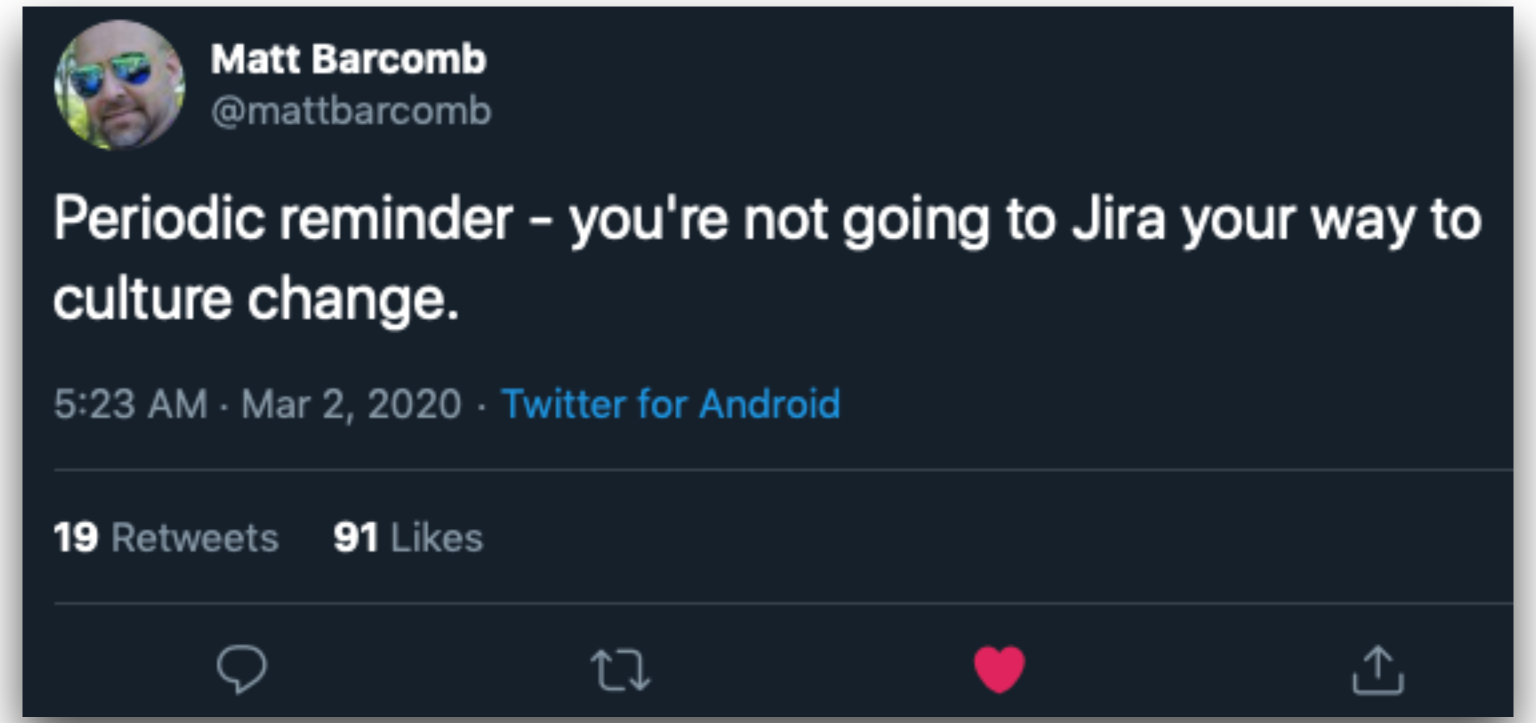
"Two ways," Mike said.
"Gradually and then suddenly."

The Sun Also Rises by Ernest Hemingway

It can be done!

But it isn’t fast.

And it isn’t about buying a tool.



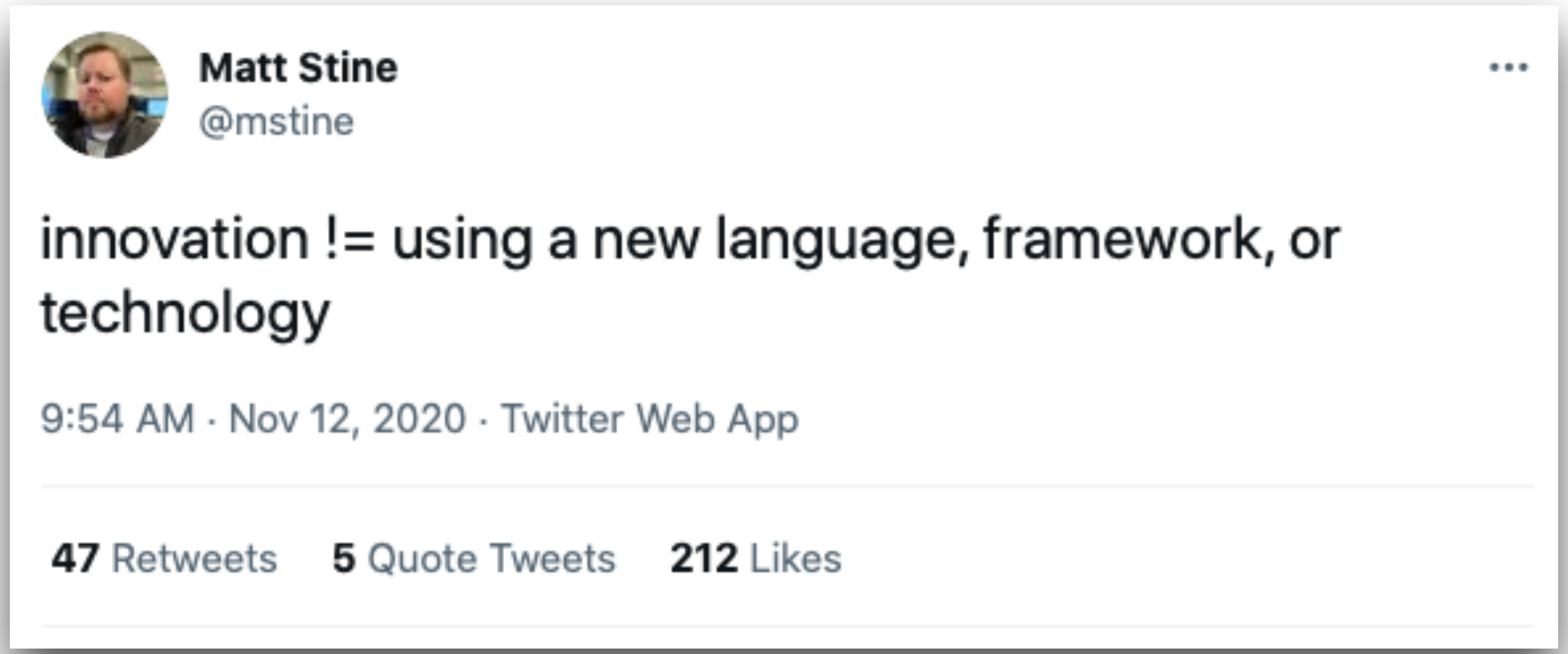
Matt Barcomb
@mattbarcomb

Periodic reminder - you're not going to Jira your way to culture change.

5:23 AM · Mar 2, 2020 · Twitter for Android

19 Retweets 91 Likes

https://mobile.twitter.com/mattbarcomb/status/1234439273077772289


Matt Stine
@mstine

innovation != using a new language, framework, or technology

9:54 AM · Nov 12, 2020 · Twitter Web App

47 Retweets 5 Quote Tweets 212 Likes

https://twitter.com/mstine/status/1326916377740005378

Agile journey…

We need project rooms.

What’s a project room?

We don’t do that, we arrange efficiently placed felt lined boxes.

Pestered them.

They relented. We got the worst conference room imaginable.

Interior, no windows.

No cell coverage.

But it was as a start.

Show success, build credibility.

Serves as a model.

Here’s what we’d like in the next project room…

Grew from there.

Now? IT floors are all project rooms.

Small, medium, large.

Not even a question anymore.

But it took time. And persistence.

Simpler if you are a decision maker!

Bezos mandate.

https://plus.google.com/+RipRowan/posts/eVeouesvaVX

All data will be exposed through a public service interface.

Services are *the* communication method between teams.

No other form of communication is allowed. No direct reads, links etc.

No back doors.

All services must be designed to be public. No exceptions.

Don’t want to do this?
You’re fired.

Unsurprisingly, things began to change.

You probably don't have that kind of clout. Sorry.

There are other approaches!

“we don’t have these Netflix superstar engineers to do the things you’re talking about”, and when I looked around the room at the company names my response was “we hired them from you and got out of their way”

–Adrian Cockcroft

https://medium.com/@adrianco/you-dont-add-innovation-to-a-culture-you-get-out-of-it-s-way-2e6148349aae

"You don't add innovation to a culture, you get out of its way."

– Adrian Cockcroft

https://medium.com/@adrianco/you-dont-add-innovation-to-a-culture-you-get-out-of-it-s-way-2e6148349aae

11 DEC 2019

Driving change: evolving our culture through design

Samantha Rosa
Lead Designer

Sara Michelazzo
Lead Product Designer

Experience Design » Career Hacks » Technology » Careers »

Having a culture that inspires and challenges people is not easy. At ThoughtWorks, we could easily walk down the path of competitiveness and control. Instead, we walk in the opposite direction. We value a safe and autonomous environment where we cultivate each other in order to foster technical excellence.

We are a community of technologists, proud of our diversity. We strive to create an environment where ThoughtWorkers own their growth journey, make their mark, and intentionally and continuously help each other grow. This is what we call cultivation.

Sometimes, it is easier
to start a new culture.

A lab, a special floor,
a different building.

Free of typical constraints.

Allows you to start fresh. With like minded individuals.

Chances are, our org structure isn’t helping matters much.

melconway.com

HOME INVENTOR CONWAY'S LAW

LINKS

Text of Paper

Wikipedia: Fred Brooks

Wikipedia: Conway's Law

In 1967 I submitted a paper called "How Do Committees Invent?" to the *Harvard Business Review*. *HBR* rejected it on the grounds that I had not proved my thesis. I then submitted it to *Datamation*, the major IT magazine at that time, which published it April 1968. The text of the paper is here.

Here is one form of the paper's thesis:

> **Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure.**

Fred Brooks cited the paper and the idea in his elegant classic "The Mythical Man-Month," calling it "Conway's Law." The name stuck.

Following is an extract from an article in Wikipedia. (The concept originated in the software world but is not limited to any specific domain.)

> Conway's law was not intended as a joke or a Zen koan, but as a valid sociological observation. It is a consequence of the fact that two software modules A and B cannot interface correctly with each other unless the designer and implementer of A communicates with the designer and implementer of B. Thus the interface structure of a software system *necessarily* will show a congruence with the social structure of the organization that produced it.

Brooks recognized that the law has important corollaries in management theory. Here is one stated in the paper.

> Because the design that occurs first is almost never the best possible, the prevailing system concept may need to change. Therefore, flexibility of organization is important to effective design.

In retrospect, *HBR*'s basis for rejecting the paper says more about differences in notions of "proof" than it does about the paper.

[Note: I assume no responsibility for information in other Web sites. The reference to Fred Brooks in Wikipedia, for example, was accurate to the best of my knowledge at the time I created the link to it, but it is subject to change beyond my control (as is all information on the Web not in this site).]

Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure.

– Melvin Conway

How do you create a series of small isolated services…

If your organization isn’t a set of small, isolated teams?

Inverse Conway's Law.

Evolve your teams towards the architectural end state you desire.

https://www.thoughtworks.com/radar/techniques/inverse-conway-maneuver

Leads to some interesting outcomes.

Teams are often siloed. But microservices have to work together.

How do we get those teams to work together effectively?

My incentives may not align with yours…how do we solve for that?

How do we build out infrastructure for many disparate teams?

How do we staff up the operations team?

Application to Operations cannot be 1-1!

Though the VP of ops may have some thoughts on the matter…

You build it, you own it.

Are your teams ready for pager duty?

Can’t just pitch it over the wall to Ops anymore.

Changing technology is (comparatively) easy.

Changing culture is crucial to our success.

EVOLVING TO CLOUD NATIVE

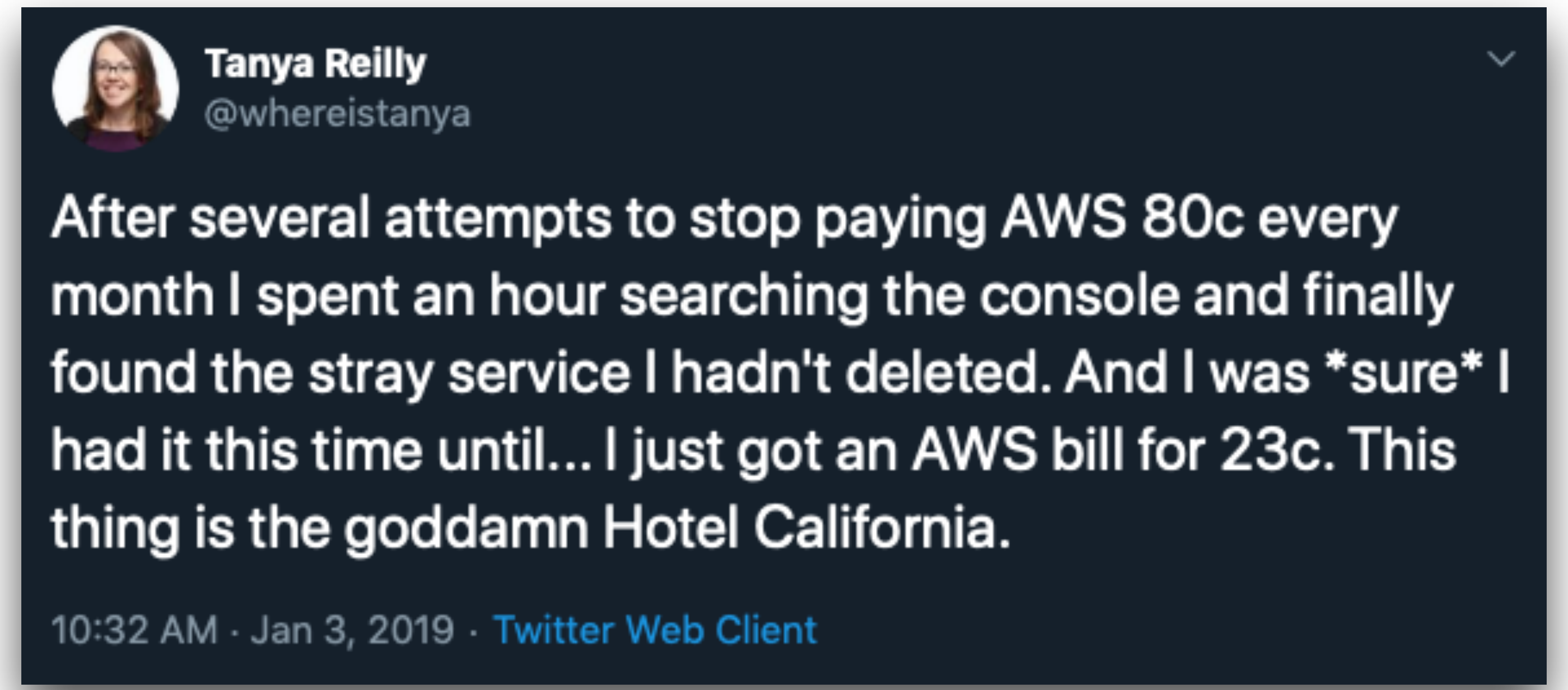
Cloud computing gives us some very interesting abilities.

Scale up. Scale down. On demand.

Limitless compute.*

* Additional fees may apply.

Said fees can be…opaque.

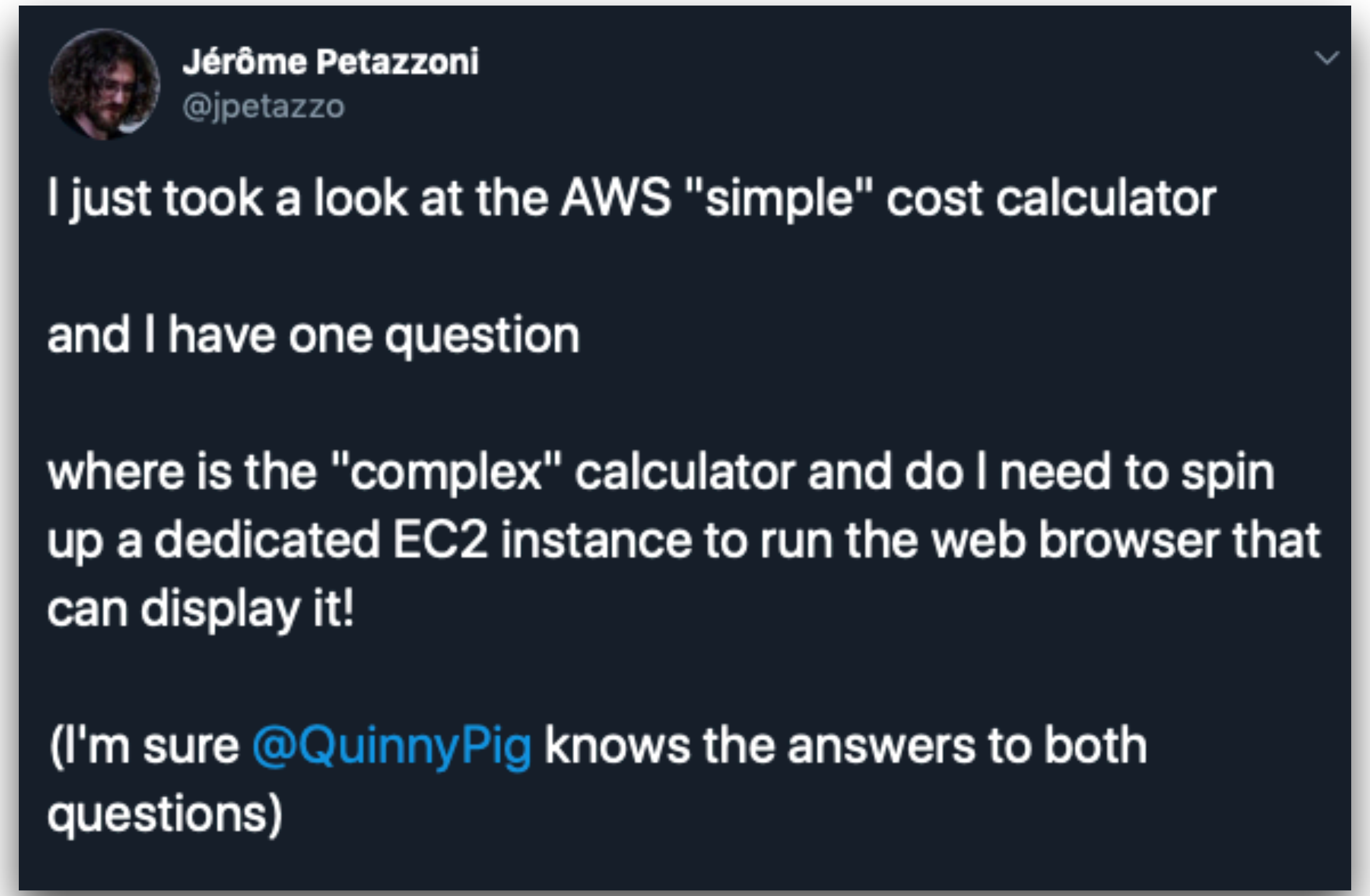


Tanya Reilly
@whereistanya

After several attempts to stop paying AWS 80c every month I spent an hour searching the console and finally found the stray service I hadn't deleted. And I was *sure* I had it this time until... I just got an AWS bill for 23c. This thing is the goddamn Hotel California.

10:32 AM · Jan 3, 2019 · Twitter Web Client

https://mobile.twitter.com/whereistanya/status/1080864493108776961



Jérôme Petazzoni
@jpetazzo

I just took a look at the AWS "simple" cost calculator

and I have one question

where is the "complex" calculator and do I need to spin up a dedicated EC2 instance to run the web browser that can display it!

(I'm sure @QuinnyPig knows the answers to both questions)

https://mobile.twitter.com/jpetazzo/status/1227638126602080256

calculator.s3.amazonaws.com

SIMPLE MONTHLY CALCULATOR

Language: English

Need Help? Watch the Videos or Read How AWS Pricing Works or Contact Sales

Get Started with AWS: Learn more about our Free Tier or Sign Up for an AWS Account »

☑ FREE USAGE TIER: New Customers get free usage tier for first 12 months

Reset All

Services | Estimate of your Monthly Bill ($ 0.00)

Choose region: US East (N. Virginia)

Inbound Data Transfer is Free and Outbound Data Transfer is 1 GB free per region per month

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale computing easier for developers. Amazon Elastic Block Store (EBS) provides persistent storage to Amazon EC2 instances.

Clear Form

Amazon EC2
Amazon S3
Amazon Route 53
Amazon CloudFront
Amazon RDS
Amazon Elastic Load Balancing
Amazon DynamoDB
Amazon ElastiCache
Amazon CloudWatch
Amazon SES
Amazon SNS
Amazon Elastic Transcoder
Amazon WorkSpaces
Amazon WorkDocs
AWS Directory Service
Amazon Redshift
Amazon Glacier
Amazon SQS
Amazon SWF
Amazon Elastic MapReduce
Amazon Kinesis Streams
Amazon CloudSearch
AWS Snowball
AWS Direct Connect
Amazon VPC
Amazon SimpleDB

Compute: Amazon EC2 Instances:

	Description	Instances	Usage	Type	Billing Option	Monthly Cost
⊕	Add New Row					

Compute: Amazon EC2 Dedicated Hosts:

	Description	Number of Hosts	Usage	Type	Billing Option
⊕	Add New Row				

Storage: Amazon EBS Volumes:

	Description	Volumes	Volume Type	Storage	IOPS	Baseline Throughput	Snapshot Storage
⊕	Add New Row						

Compute: Amazon Elastic Graphics:

	Description	Number of Elastic Graphics	Usage	Elastic Graphics Size and Memory
⊕	Add New Row			

Additional T2/T3 Unlimited vCPU Hours per month:

For Linux, RHEL and SLES:	0
For Windows and Windows with SQL Web:	0

Elastic IP:*

[✓ Enter values below] Calculate

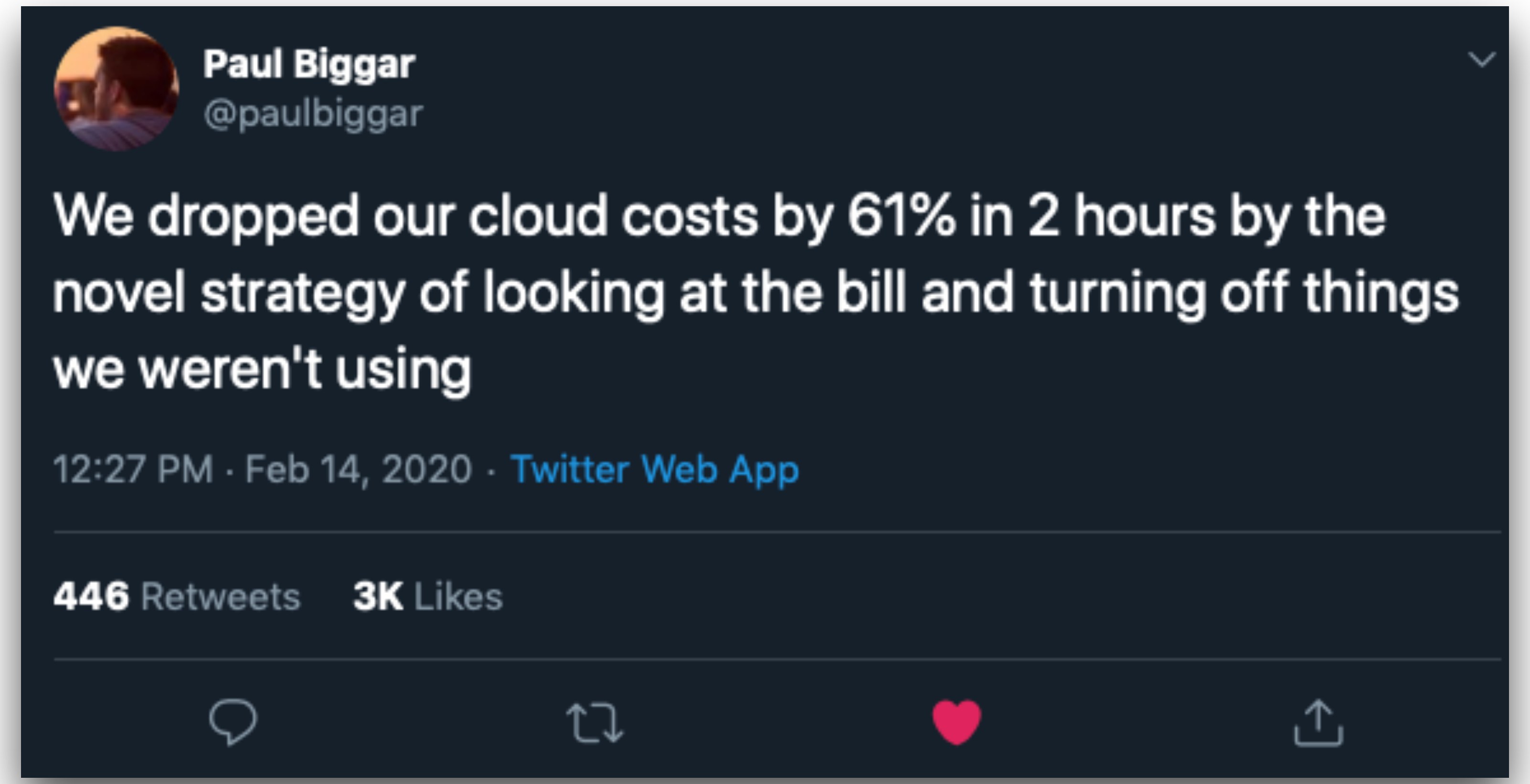
Total time the additional Elastic IPs are attached to running EC2 instances**:	0	Hours/Month
Total Non-attached time for all the Elastic IPs:	0	Hours/Month
Number of Elastic IP Remaps:	0	Per Month

Data Transfer:

Inter-Region Data Transfer Out:	0	GB/Month
Data Transfer Out:	0	GB/Month
Data Transfer In:	0	GB/Month

Common Customer Samples

Free Website on AWS
AWS Elastic Beanstalk Default
Marketing Web Site
Large Web Application (All On-Demand)
Media Application
European Web Application
Disaster Recovery and Backup


Paul Biggar
@paulbiggar

We dropped our cloud costs by 61% in 2 hours by the novel strategy of looking at the bill and turning off things we weren't using

12:27 PM · Feb 14, 2020 · Twitter Web App

446 Retweets 3K Likes

https://mobile.twitter.com/paulbiggar/status/1228385370439467009

Cloud native isn’t just an architectural pattern.

Combination of practices, techniques, technologies.

Agile development.

Continuous delivery.

Automation.

Containers.

Microservices.

Functions.

Changes our culture.

DevOps.

Infrastructure is a different game today isn’t it?

We’ve seen this massive shift.

Servers used to be home grown.

Bespoke. Artisanal.

Spent days hand crafting them.

Treated them like pets…

Did whatever it took to keep them healthy and happy.

Servers were a heavily constrained resource.

They were really expensive!

Had to get our money’s worth…

Thus was born app servers.

Put as many apps as possible on a server.

Maximize the return on investment.

But that has some unintended side effects.

Shared resources.

One application's bug could take down multiple apps.

Coordinating changes hurts.

“Your app can’t get this feature until all other apps are ready.”

Currency === 18 months of freezes, testing, frustration.

Organizations ignored currency issues…pain wasn’t “worth it”.

"Fear is the path to the dark side. Fear leads to anger. Anger leads to hate. Hate leads to suffering."

–Yoda

#YodaOps

Move code from one server to another…

Worked in dev…but not test.

Why?!?

The environments are
the same…right?

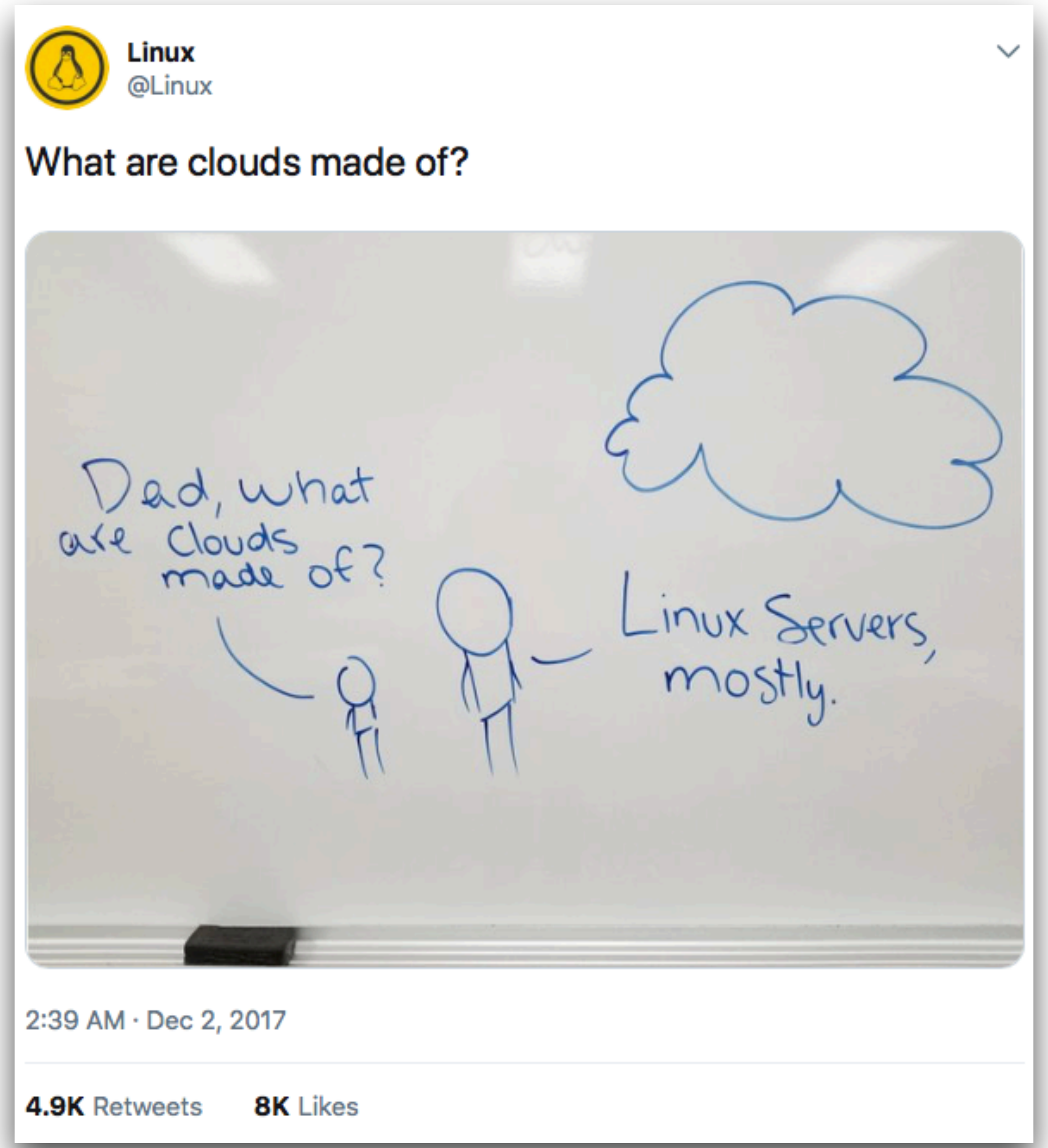
"Patches were applied in a different order…"

Can I change careers?

Things started to change.

Servers became commodities.

Linux and Intel chips replaced custom OS on specialized silicon.



Linux
@Linux

What are clouds made of?

2:39 AM · Dec 2, 2017

4.9K Retweets 8K Likes

https://mobile.twitter.com/linux/status/936877536780283905?lang=en

Prices dropped.

Servers were no longer the constraining factor.

People costs eclipsed hardware costs.

Heroku, AWS, Google App Egine, Cloud Foundry, Azure.

Shared servers became a liability.

Treat them like cattle…when they get sick, get a new one.

15

New abstractions.

Containers and PaaS changed the game.

Package the app up with everything it needs.

Move *that* to a different environment.

Works in dev? You’re testing the exact same thing in test.

So. Much. Win.

Your app needs a spiffy new library? Go ahead!

It doesn’t impact any other app because you are isolated.

Moves the value line.

Less “undifferentiated heavy lifting”.

"Good job configuring servers this year."

–No CEO Ever…

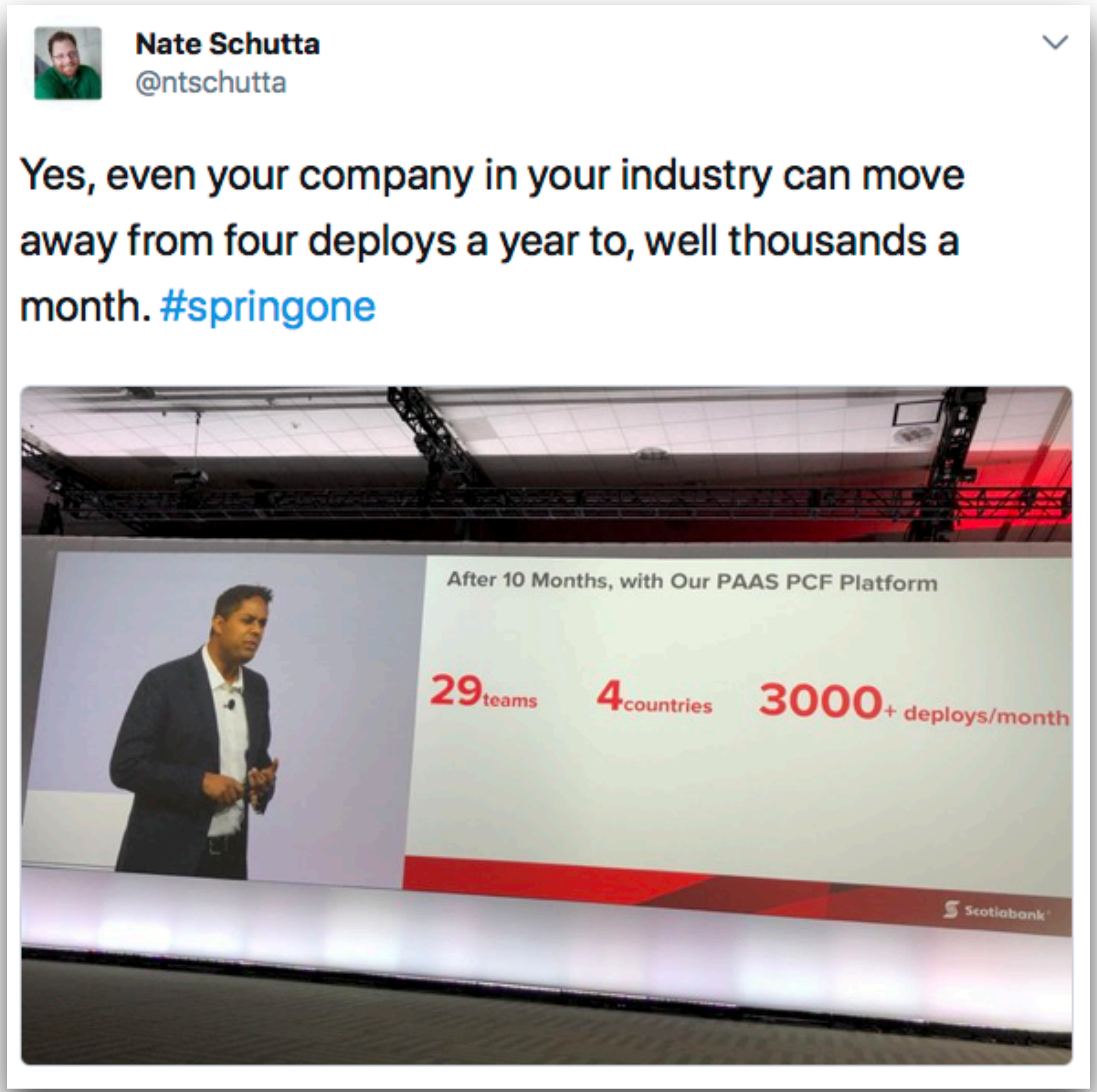
Changes development.

Always be changing.

Run experiments. A/B testing.

Respond to business changes.

Deliver in days not months.


Nate Schutta
@ntschutta

Yes, even your company in your industry can move away from four deploys a year to, well thousands a month. #springone

https://mobile.twitter.com/ntschutta/status/938109379995353088

Speed matters.

Disruption impacts every business.

Your industry is not immune.

Amazon Prime customers can order from Whole Foods.

Some insurance companies view Google as a competitor.

We’re all technology companies today.

AUTOMATION

Back in the day…

Builds were often Rube Goldberg machines.

Lots of manual tasks.

Right click in your IDE…

Hard coded credentials, magic drive locations.

Again, it worked. For some definition of worked.

Besides, we didn’t do it very often.

COFFEE

Artisanal coffee is worth seeking out.

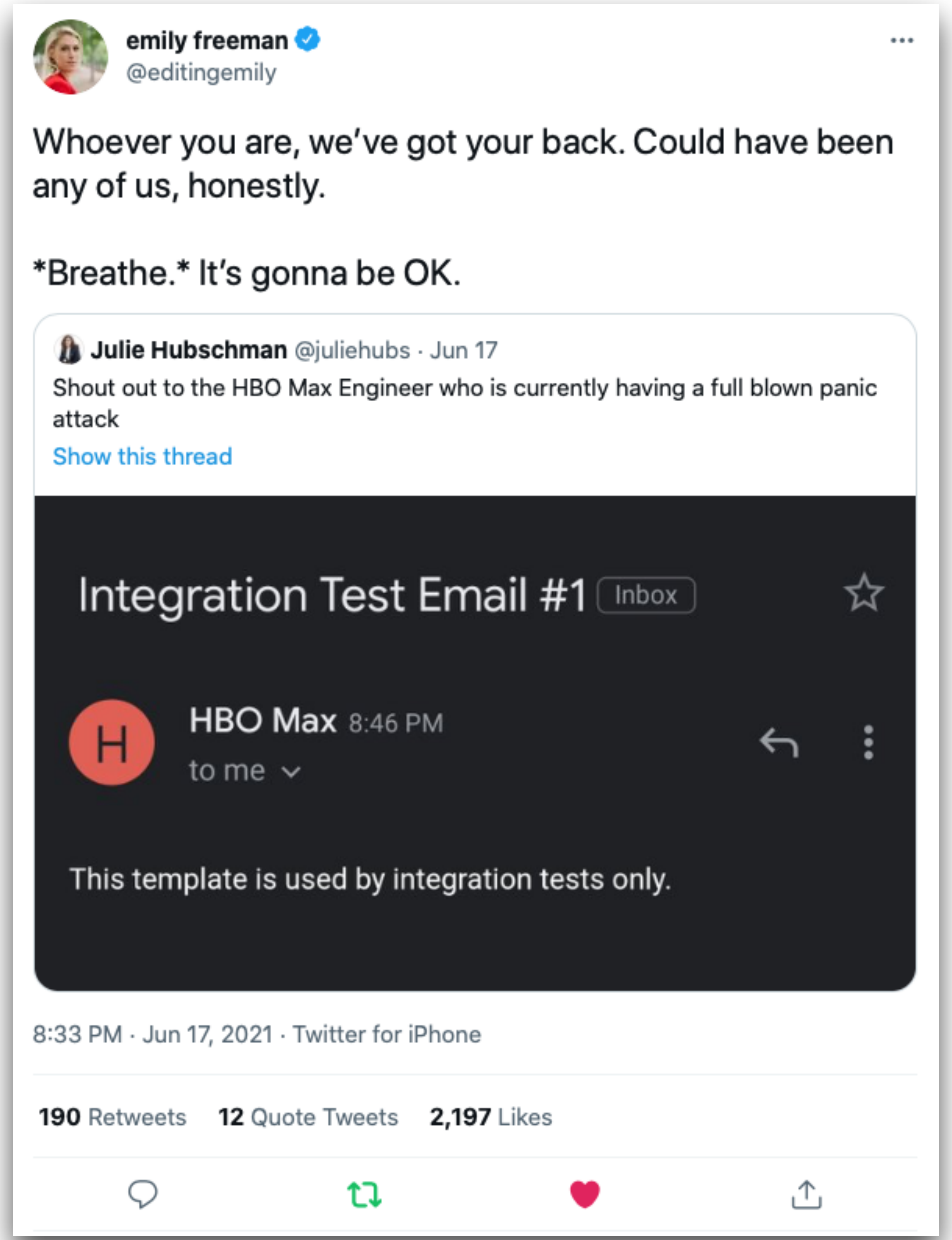
Bespoke builds won’t work today.

People can’t do the same thing twice.

See golf.

People have bad days. They get bored. They skip a step.

They fat finger a command.


emily freeman
@editingemily

Whoever you are, we've got your back. Could have been any of us, honestly.

Breathe. It's gonna be OK.

Julie Hubschman @juliehubs · Jun 17
Shout out to the HBO Max Engineer who is currently having a full blown panic attack
Show this thread

Integration Test Email #1 Inbox

HBO Max 8:46 PM
to me

This template is used by integration tests only.

8:33 PM · Jun 17, 2021 · Twitter for iPhone

190 Retweets 12 Quote Tweets 2,197 Likes

https://twitter.com/editingemily/status/1405700159967678464

Computers…not so much.

In computer science, there are only three numbers.

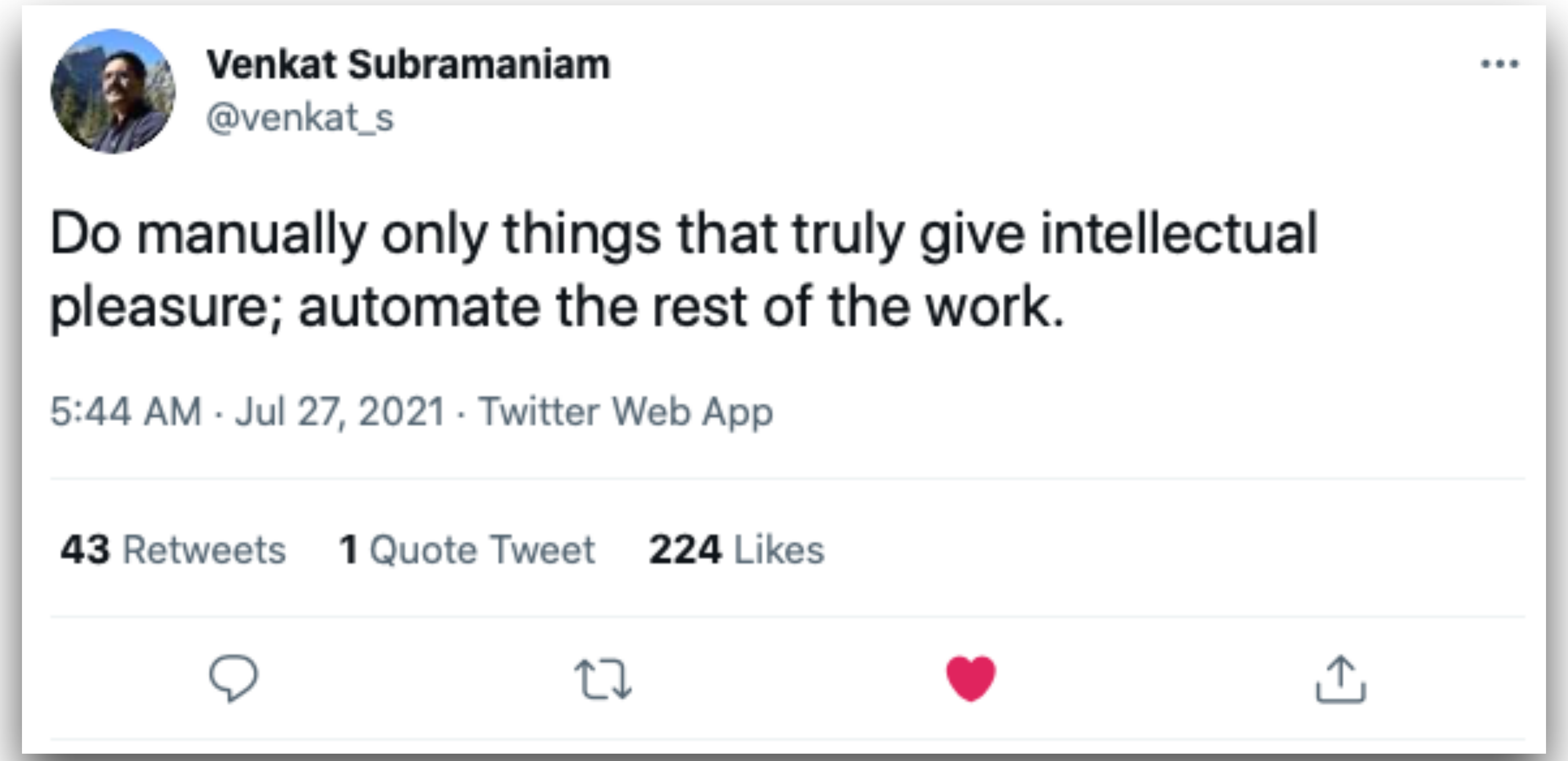
Something we do 0 times,
1 and only 1…

And n. If you do something more than once…

You will do it one billion times.

Anything you do more than once should be automated.

Offload that toil to computers.



Venkat Subramaniam
@venkat_s

Do manually only things that truly give intellectual pleasure; automate the rest of the work.

5:44 AM · Jul 27, 2021 · Twitter Web App

43 Retweets 1 Quote Tweet 224 Likes

https://twitter.com/venkat_s/status/1419971848150806532

We need consistency.

We need CI and CD pipelines.

tanzu.vmware.com

vmware Tanzu Developer Center

Learn Topics Blog Tanzu.TV Community

Guides

Kubernetes

CI/CD

ArgoCD: Getting Started with Kubernetes-native Continuous Delivery

CI/CD: What is it?

Concourse: Getting Started with Concourse CI

Getting Started with Tekton Part 1: Hello World

Getting Started with Tekton Part 2: Building a Container

Containers

Event Streaming

Messaging and Integration

Python

Spring

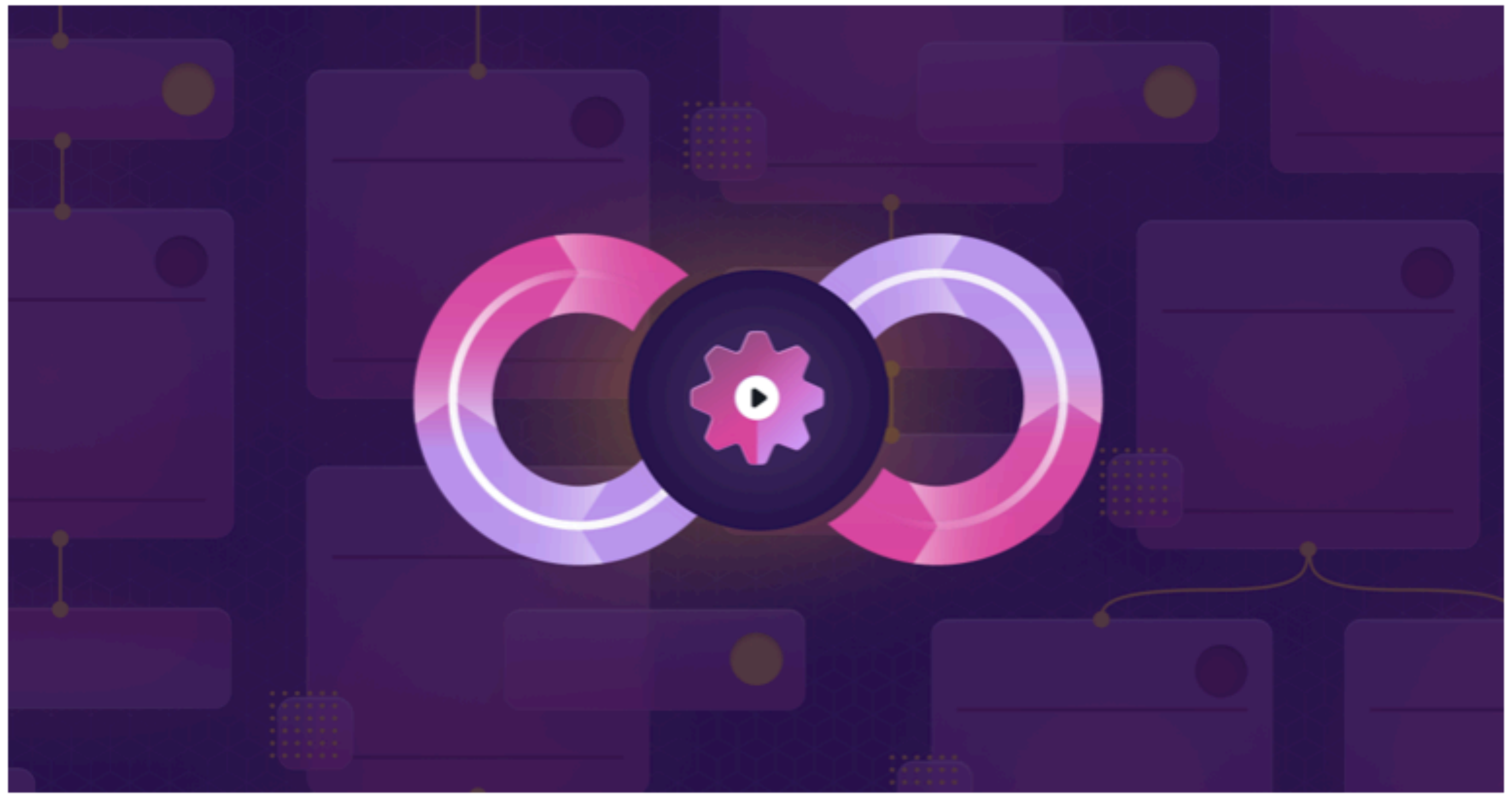
Microservices

Guides / CI/CD

CI/CD

Continuous integration, continuous delivery, and continuous deployment cover different parts of the software development lifecycle, but are similar concepts. The idea is to automate as much of the pipeline as possible so that each change is continuously integrated into your codebase, and able to be continuously delivered and hopefully continuously deployed.

https://tanzu.vmware.com/developer/guides/ci-cd/



October 29, 2020 — Engineering

Getting started with DevOps automation

Jared Murrell

This is the second post in our series on DevOps fundamentals. For a guide to what DevOps is and answers to common DevOps myths check out part one.

Share

Twitter

Facebook

What role does automation play in DevOps?

No, this doesn’t mean commits go to production 30 seconds later.

They can mind you. But
no one starts there.

CI = Continuous Integration.

Code is merged early and often avoiding merge conflicts.

Essential to avoid merge hell.

A commit triggers automated tests, code quality scans, etc.

Ensures new commits don’t break the application.

CD = Continuous Delivery.

Takes the build to the next step - how we release changes to our customers.

Carries automation through to the deployment & release management.

You decide how often you want to release.

Well, your team, your customer…

Goal is to be in a releasable state.

Working in small batches.

Lowers risk!

Quarterly releases contain hundreds, maybe thousands of changes.

The integration of which
almost always leads to breaks.

Which change caused the break?

¯\_(ツ)_/¯

Push one or two changes…
much easier to debug.

Expertise grows with repetition.

Do something once or twice and you won’t improve…

Deploy early, deploy often.

You will get better at it.

Need to develop trust
in the process.

We also need recoverability.

No such thing as zero outages.

Mistakes will be made.

Outages *will* happen.
Bugs will creep into the code.

Mean time to recovery is vital.

How do we get that fix into production quickly?

Automation.

Gives you confidence.

Ever use undo? How would your life change if it didn’t exist?

Imagine developing software without version control…

We need robust pipelines.

Concourse, Circle CI, Travis CI,
Visual Studio Team Services, Jenkins.

Many are cloud based now.

GitHub actions bakes some of this into SCM.

https://github.com/features/actions

docs.github.com

GitHub Docs

GitHub.com English Search topics, products...

← All products

GitHub Actions

QUICKSTART FOR GITHUB ACTIONS

GUIDES

- About continuous integration
- Setting up CI using templates
- Building and testing Node.js
- Building and testing .NET
- Building and testing PowerShell
- Building and testing Python
- Building and testing Ruby
- Building and testing Java with Maven
- Building and testing Java with Gradle
- Building and testing Java with Ant
- Installing an Apple certificate on macOS runners for Xcode development
- About packaging with GitHub Actions
- Publishing Node.js packages
- Publishing Java packages with Maven
- Publishing Java packages with Gradle
- Publishing Docker images
- Storing workflow artifacts
- Caching dependencies
- About service containers
- Redis service containers
- PostgreSQL service containers
- Deploying to Amazon Elastic Container Service
- Deploying to Azure App Service
- Deploying to Google Kubernetes Engine

Product

GitHub Actions

Automate, customize, and execute your software development workflows right in your repository with GitHub Actions. You can discover, create, and share actions to perform any job you'd like, including CI/CD, and combine actions in a completely customized workflow.

Quickstart Reference guides

GitHub Actions - Supercharge your GitHub Flow
Copy link
GitHub Flow with GitHub Actions
Main
Feature
Watch on YouTube

GUIDES View all →

Learn GitHub Actions
Whether you are new to GitHub Actions or interested in learning all...

About continuous integration
You can create custom continuous integration (CI) and continuous deployment (CD) workflows...

About packaging with GitHub Actions
You can set up workflows in GitHub Actions to produce packages and...

POPULAR

- Workflow syntax
- Learn GitHub Actions
- Events that trigger workflows
- Context and expression syntax
- Environment variables
- Encrypted secrets

WHAT'S NEW View all →

Limit workflow run or job concurrency
April 19

GitHub CLI 1.9 enables you to work with GitHub Actions from your terminal
April 15

Setup-java now support Adopt OpenJDK
April 05

Code examples

Deployment Strategies.

In the beginning..

Nuke and pave.

Overlay the current version with the new version…

And hope for the best!

Often resulted in issues, breaks, bugs and sleepless nights.

"The application will be down for maintenance…"

Customers’ expectations
have changed.

Your site is down?
Your competitor’s isn’t.

Evolved to the recreate pattern.

Spin down the current version,
then spin up the new.

Simple! But. Long downtimes.

Not…ideal.

Rolling updates.

Subset of instances (defined by window size) are updated at a time.

No downtime!

Not all users get the new version at the same time…

Harder to rollback.

Best be backwards compatible…

Sticky affinity?

Session management…

Blue-Green Deployments.

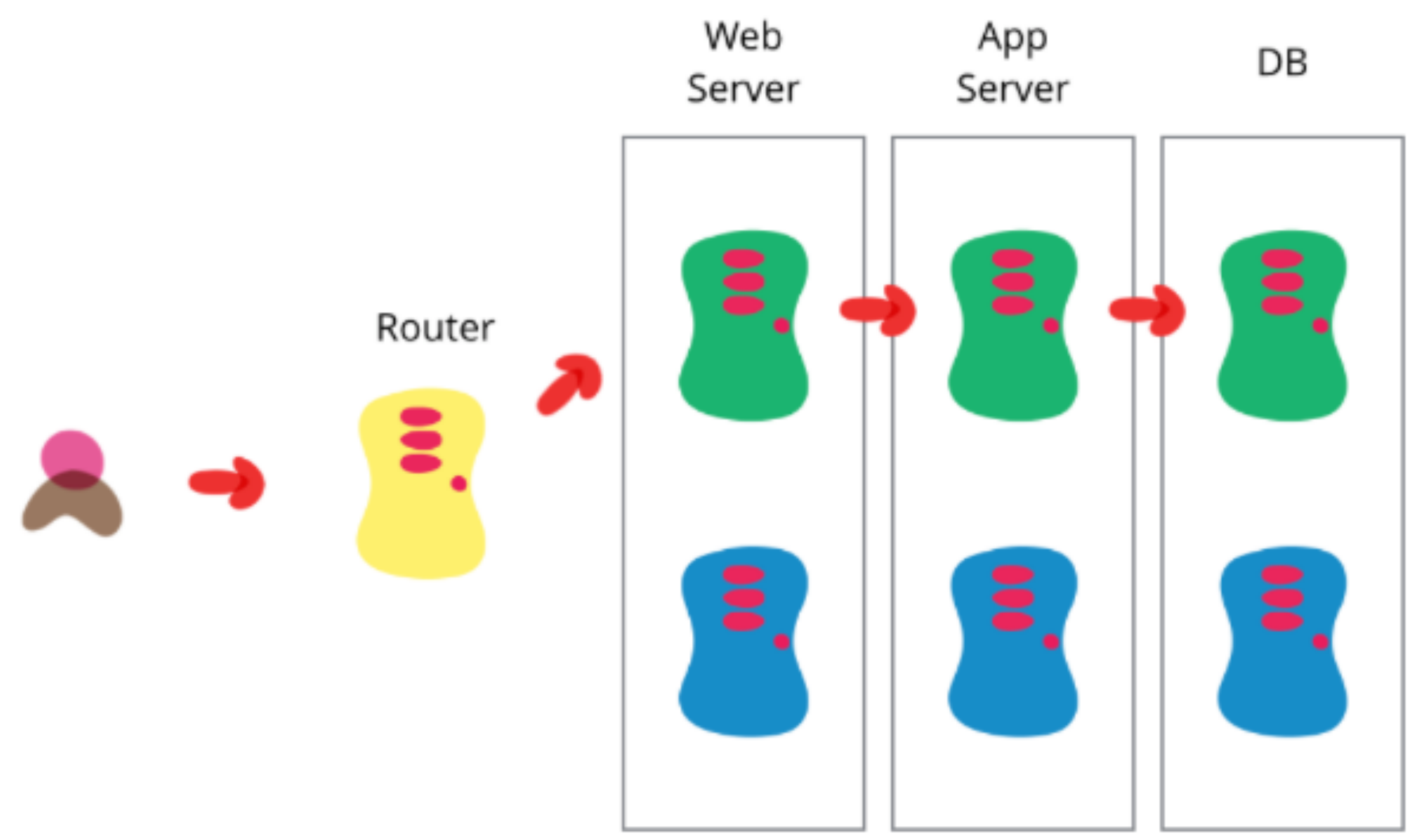
BlueGreenDeployment

1 March 2010

Martin Fowler

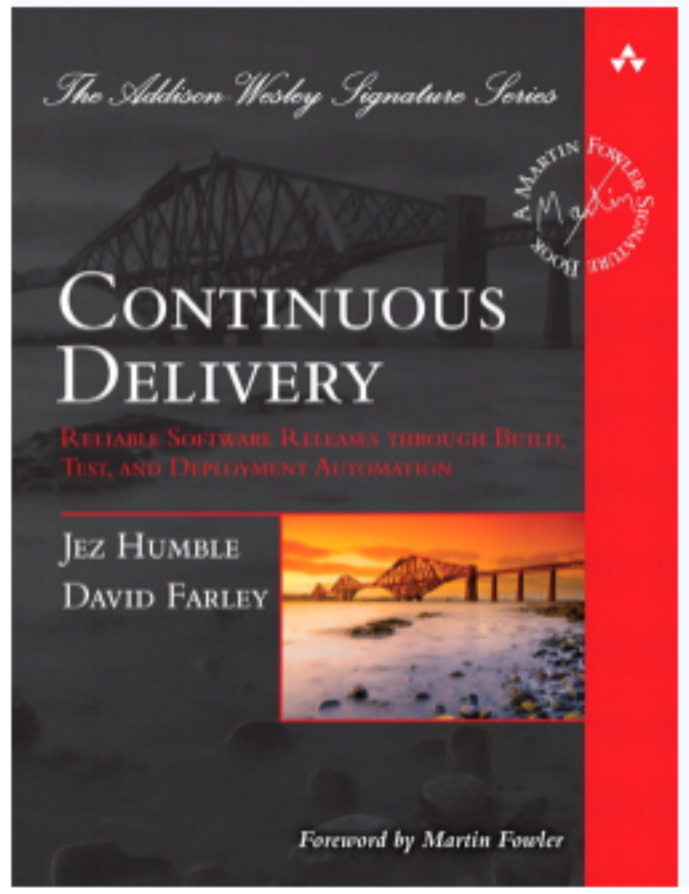
CONTINUOUS DELIVERY

One of the goals that my colleagues and I urge on our clients is that of a completely automated deployment process. Automating your deployment helps reduce the frictions and delays that crop up in between getting the software "done" and getting it to realize its value. Dave Farley and Jez Humble are finishing up a book on this topic - Continuous Delivery. It builds upon many of the ideas that are commonly associated with Continuous Integration, driving more towards this ability to rapidly put software into production and get it doing something. Their section on blue-green deployment caught my eye as one of those techniques that's underused, so I thought I'd give a brief overview of it here.



martinfowler.com

< All Books


The Addison Wesley Signature Series
A Martin Fowler Signature Book
Continuous Delivery
Reliable Software Releases through Build, Test, and Deployment Automation
Jez Humble
David Farley
Foreword by Martin Fowler

Continuous Delivery

Reliable Software Releases through Build, Test, and Deployment Automation

by Jez Humble and David Farley

2010

amazon informIT

Notes for buying my books

In the late 90's I paid a visit to Kent Beck, then working in Switzerland for an insurance company. He showed me around his project and one of the interesting aspects of his highly disciplined team was the fact that they deployed their software into production every night. This regular deployment gave them many advantages: written software wasn't waiting uselessly before it was used, they could respond quickly to problems and opportunities, and the rapid turn-around led to a much deeper relationship between them, their business customer, and their final customers.

In the last decade I've worked at ThoughtWorks and a common theme of our projects has been reducing that cycle time between idea and usable software. I see plenty of project stories and they almost all involve a determined shortening of that cycle. While we don't usually do daily deliveries into production, it's now common to see teams doing bi-weekly releases.

Dave and Jez have been part of that sea-change, actively involved in projects that have built a culture of frequent, reliable deliveries. They and our colleagues have taken organizations that struggled to deploy software once a year, into the world of Continuous Delivery, where releasing becomes routine.

AWARDS

Dr Dobbs: Jolt Excellence (2011)

FURTHER READING

Website for Book

Free chapter on build pipelines

InformIT has made chapter five of the book available as a free download. This provides a good introduction to deployment pipelines.

Software Delivery Guide

Guide to articles on this site that expand on Continuous Delivery

Two (identical) deployment environments.

One is currently serving production traffic - call it Blue.

Actively testing the newest version on Green.

Happy? Switch the routing table to point production traffic at Green.

Blue is now idle.

Oh no, there’s an issue with Green that you missed?

Update the routing table to point back to Blue.

Everything checks out?

Blue now becomes staging.
And you alternate from there.

Essentially testing disaster recovery on every deploy…

Databases can be tricky…

Separate schema changes from application changes.

But. Zero downtime,
simple rollback.

Reduced risk.

Expensive - essentially running two versions of prod.

Backwards compatibility.

Drain down transactions on current before cutting over.

What about Red-Black?

Same thing, different colors.

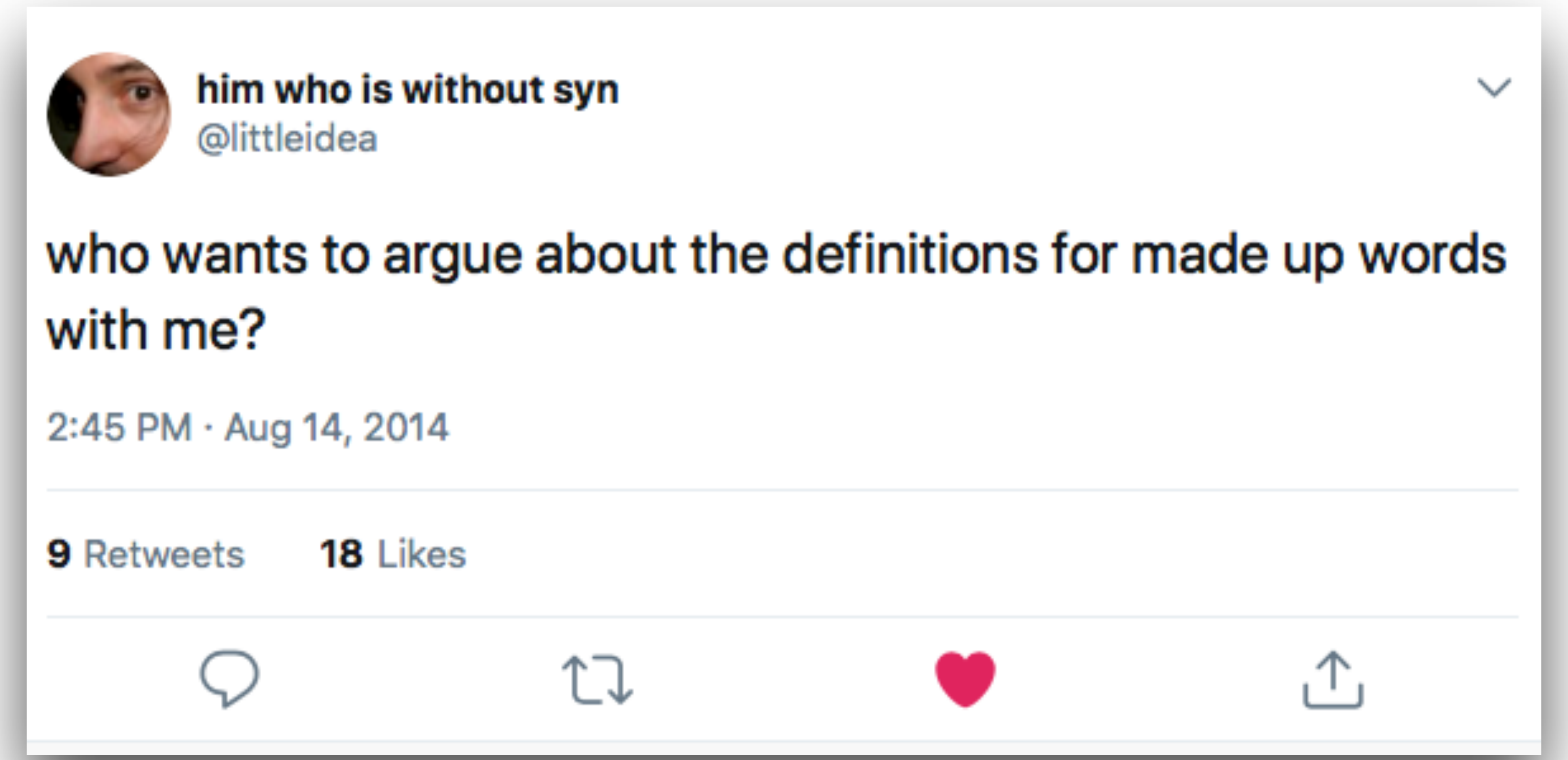
Branding?

Some argue Blue-Green can have both versions serving traffic.

Whereas Red-Black only one version serves traffic.

Is Red-Black a specialization of Blue-Green?

¯\_(ツ)_/¯



him who is without syn
@littleidea

who wants to argue about the definitions for made up words with me?

2:45 PM · Aug 14, 2014

9 Retweets 18 Likes

https://mobile.twitter.com/littleidea/status/500005289241108480

Same concepts, different name.

Canary Releases.

If it checks out in staging, it is going to canary - with real traffic.

Canary - aka the canary in the coal mine.

Find out if we have issues before we do a full production push.

Some percentage of production - 5% or 10%.

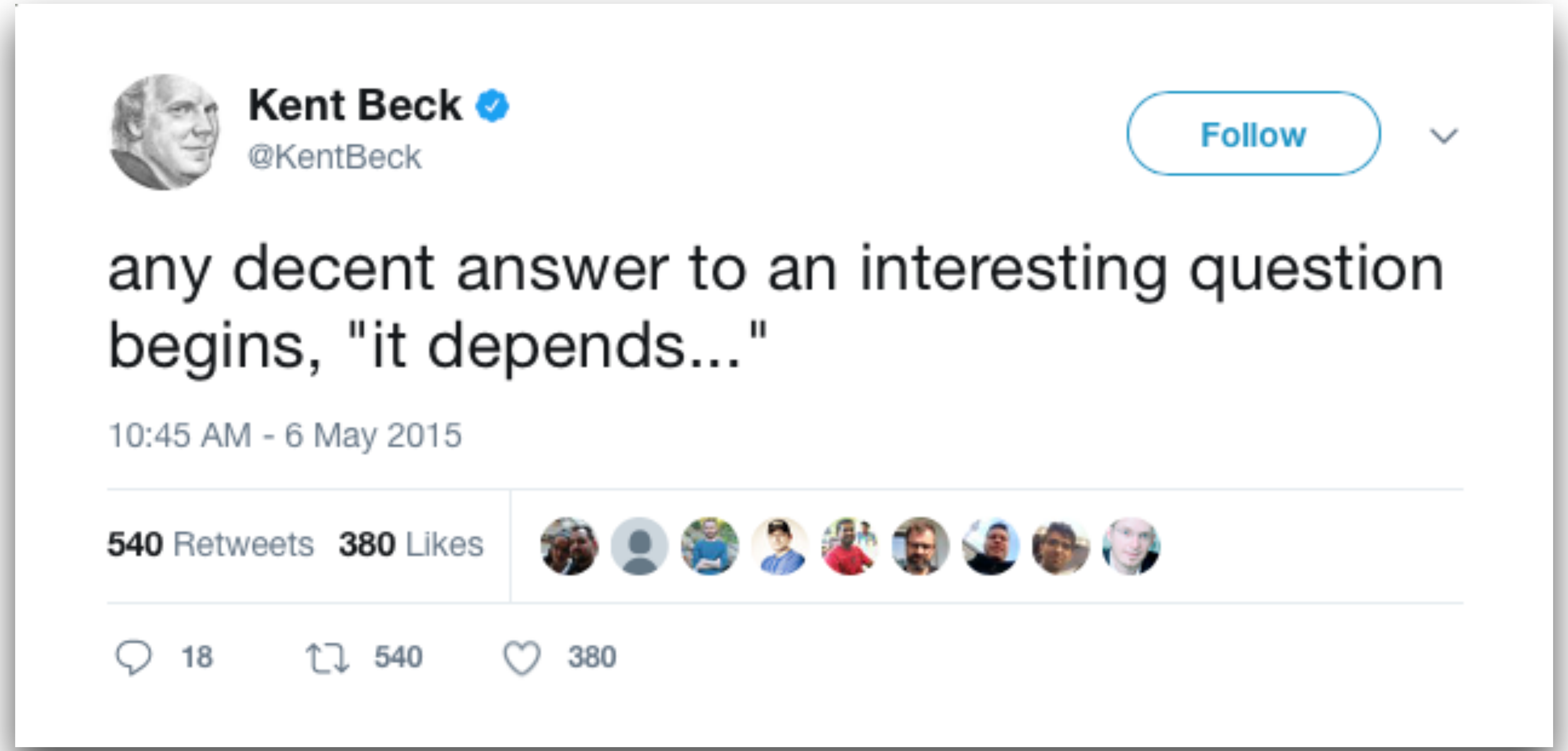
Can be a sliding scale too - start with 5%, move up to 20% etc.

Canaries are serving real production traffic.

Find errors? Automated rollbacks.

How long should our canary stage last? As long as it takes. Hours. Days.

¯_(ツ)_/¯



Kent Beck
@KentBeck

any decent answer to an interesting question begins, "it depends..."

10:45 AM - 6 May 2015

540 Retweets 380 Likes

https://twitter.com/KentBeck/status/596007846887628801

Allows us test in real production scenarios without impact all users.

Zero downtime, simple rollback.

Better have your observability story straight…

Can be time consuming.

Best be backwards compatible…

Sticky affinity?

Session management…

A/B Testing.

Extremely common.

Experimentation

The Surprising Power of Online Experiments

Getting the most out of A/B and other controlled tests by Ron Kohavi and Stefan Thomke

From the Magazine (September–October 2017)

Annie Spratt/Unsplash.com

Summary. In the fast-moving digital world, even experts have a hard time assessing new ideas. Case in point: At Bing a small headline change an employee proposed was deemed a low priority and shelved for months until one engineer decided to do a quick online controlled... **more**

Two (or more) versions of the service are running.

Experiments!

All about testing out ideas.

Some percentage of users get the experiments.

Compare and contrast.

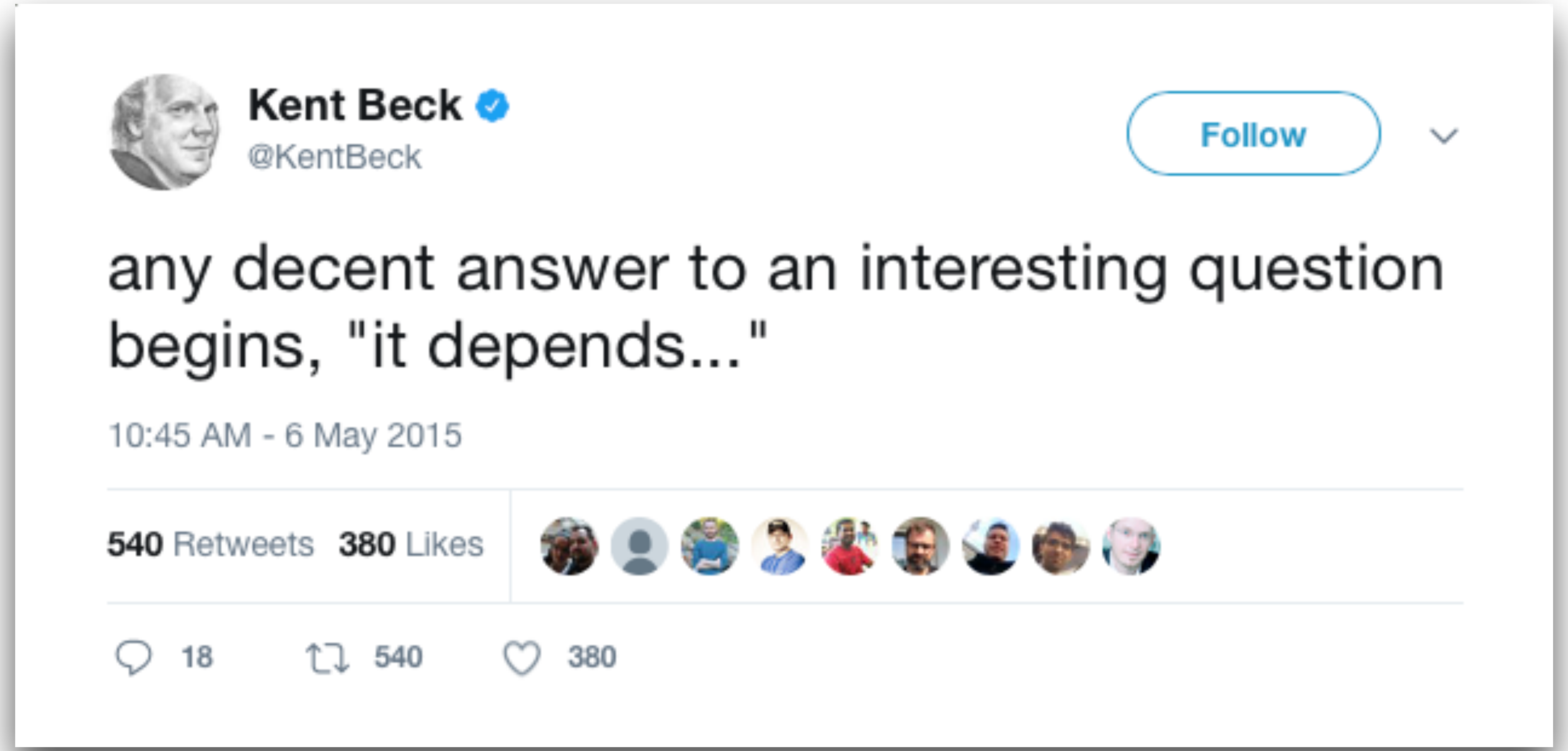
Better have your observability story straight…

Possible to break the application!

These techniques can be combined!

Rolling Blue-Green, Canary Blue-Green…

Which approach is right for you?



Kent Beck
@KentBeck

Follow

any decent answer to an interesting question begins, "it depends..."

10:45 AM - 6 May 2015

540 Retweets 380 Likes

18 540 380

https://twitter.com/KentBeck/status/596007846887628801

Do what’s right for your situation.

Different apps will have different needs.

Whatever the approach, automate it then automate some more.

Having a hard time convincing people deployments matter?

May want to familiarize yourself with the Knight Capital glitch.

https://www.sec.gov/litigation/admin/2013/34-70694.pdf

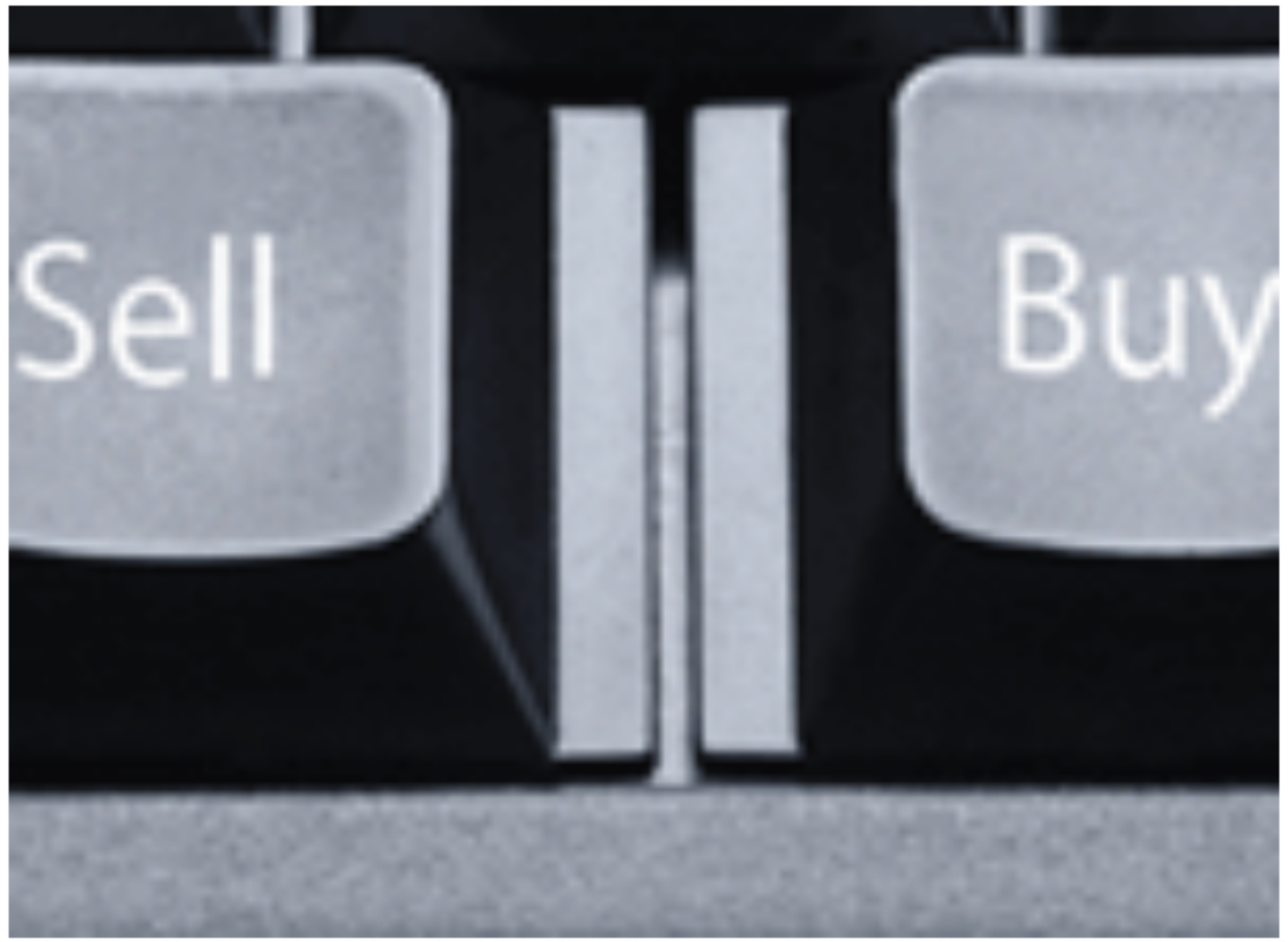
Advertisement

15 Aug 2012 | 19:30 GMT

"Zombie Software" Blamed for Knight Capital Trading Snafu

A previously dormant trading program multiplied, then executed ordinary trades

By **Robert N. Charette**



Repurposed an old flag…

Rolling deployment…
operator missed a server.

Seven servers had the new code, one didn’t.

#ThisIsNotFine.

Knight Capital lost $460 million and 75% of their market value.

A week later they were acquired.

The lesson?

Releases need to be **reliable** and **repeatable**.

Don’t rely on humans to do things perfectly.

Automation is your friend.

Help you get a good night’s sleep!

Another example of “shift left”.

Find issues when they are easiest to fix.

Once the cake is baked, pretty hard to change the recipe.

Not sure how to create a pipeline?

Spring Cloud Pipelines.

https://spring.io/blog/2018/11/13/spring-cloud-pipelines-to-cloud-pipelines-migration

Opinionated build/test/
stage/prod flow.

Gives you a place to start - modify to your hearts content.

Greater automation led to any number of “`X as code`”.

Aka infrastructure as code,
configuration as code. etc.

We built bridges and knocked down walls.

Infrastructure moved to a self service model.

Huge win in terms of responsiveness.

In the past, we had to
make decisions very early.

Often when we knew the least.

For example - how much capacity will you need?

¯_(ツ)_/¯

Take worst case…double it…add some buffer. Then a bit more.

Just in case.

We have a six week (aka month) lead time on all requests.

Lots of tickets.

And meetings.

And email.

And followup.

(╯°□°)╯︵ ┻━┻

It was in our best interest to over allocate resources.

Better to have it and not need it…

Difficult to add more capacity later.

Gave us single digit resource utilization.

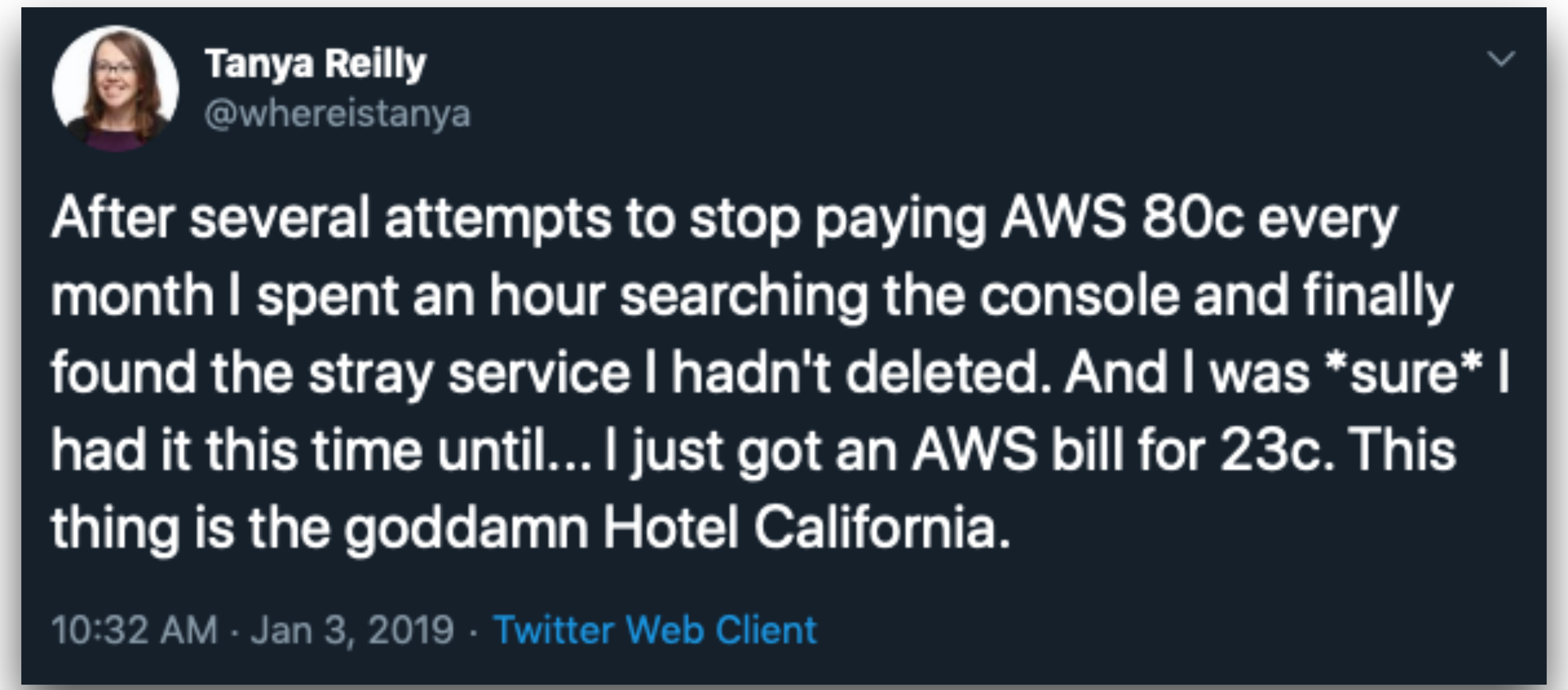
Cloud computing gives us some very interesting abilities.

Scale up. Scale down. On demand.

Limitless compute.*

* Additional fees may apply.

Said fees can be…opaque.

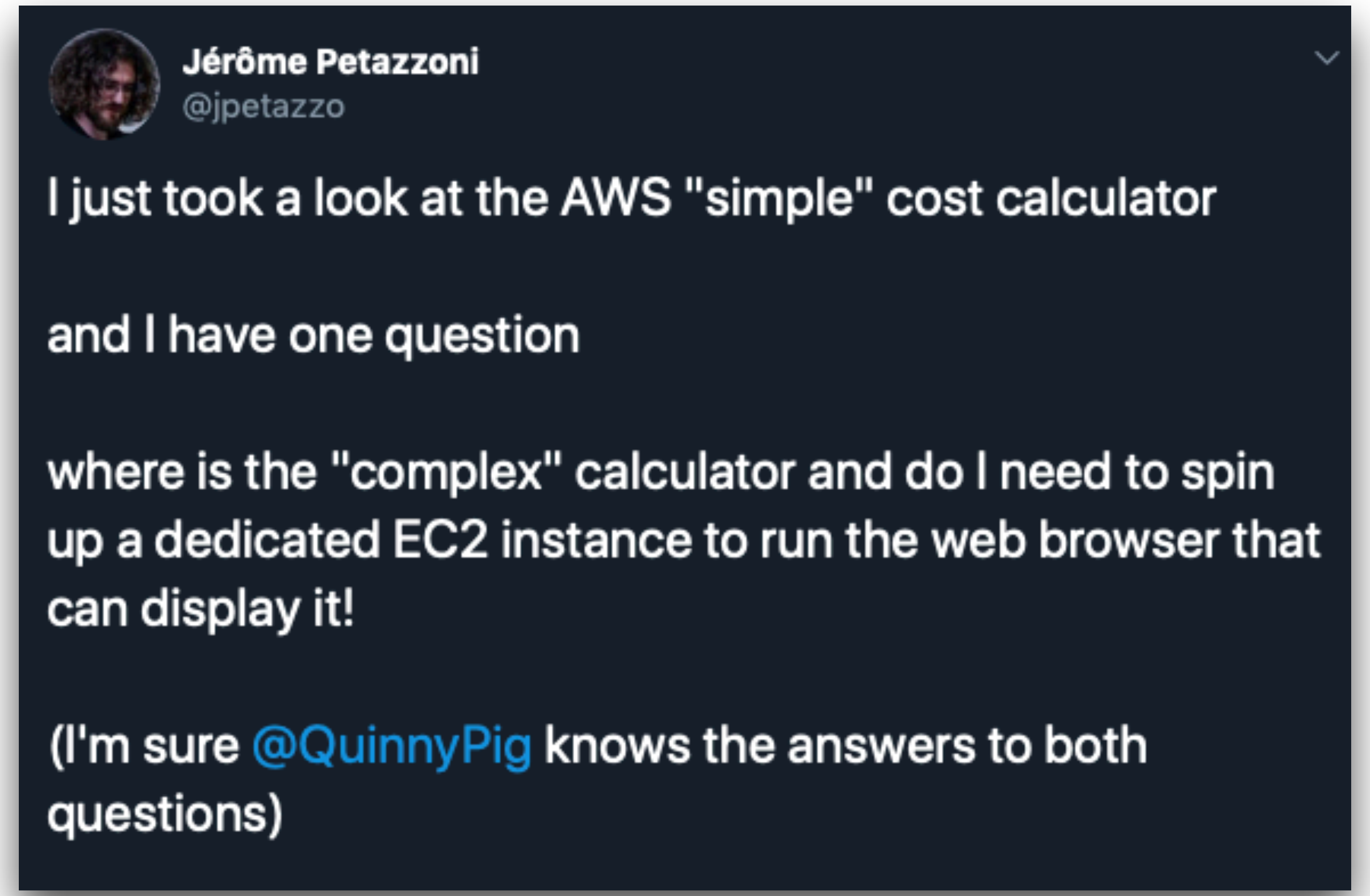


Tanya Reilly
@whereistanya

After several attempts to stop paying AWS 80c every month I spent an hour searching the console and finally found the stray service I hadn't deleted. And I was *sure* I had it this time until... I just got an AWS bill for 23c. This thing is the goddamn Hotel California.

10:32 AM · Jan 3, 2019 · Twitter Web Client

https://mobile.twitter.com/whereistanya/status/1080864493108776961



Jérôme Petazzoni
@jpetazzo

I just took a look at the AWS "simple" cost calculator

and I have one question

where is the "complex" calculator and do I need to spin up a dedicated EC2 instance to run the web browser that can display it!

(I'm sure @QuinnyPig knows the answers to both questions)

https://mobile.twitter.com/jpetazzo/status/1227638126602080256

calculator.s3.amazonaws.com

SIMPLE MONTHLY CALCULATOR

Language: English

Need Help? Watch the Videos or Read How AWS Pricing Works or Contact Sales

Get Started with AWS: Learn more about our Free Tier or Sign Up for an AWS Account »

[✓] FREE USAGE TIER: New Customers get free usage tier for first 12 months

Reset All

Services | Estimate of your Monthly Bill ($ 0.00)

Choose region: US East (N. Virginia)

Inbound Data Transfer is Free and Outbound Data Transfer is 1 GB free per region per month

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale computing easier for developers. Amazon Elastic Block Store (EBS) provides persistent storage to Amazon EC2 instances.

Clear Form

Compute: Amazon EC2 Instances:

	Description	Instances	Usage	Type	Billing Option	Monthly Cost
⊕	Add New Row					

Compute: Amazon EC2 Dedicated Hosts:

	Description	Number of Hosts	Usage	Type	Billing Option
⊕	Add New Row				

Storage: Amazon EBS Volumes:

	Description	Volumes	Volume Type	Storage	IOPS	Baseline Throughput	Snapshot Storage
⊕	Add New Row						

Compute: Amazon Elastic Graphics:

	Description	Number of Elastic Graphics	Usage	Elastic Graphics Size and Memory
⊕	Add New Row			

Additional T2/T3 Unlimited vCPU Hours per month:

For Linux, RHEL and SLES: 0

For Windows and Windows with SQL Web: 0

Elastic IP:*

[✓ Enter values below] Calculate

Total time the additional Elastic IPs are attached to running EC2 instances**: 0 Hours/Month

Total Non-attached time for all the Elastic IPs: 0 Hours/Month

Number of Elastic IP Remaps: 0 Per Month

Data Transfer:

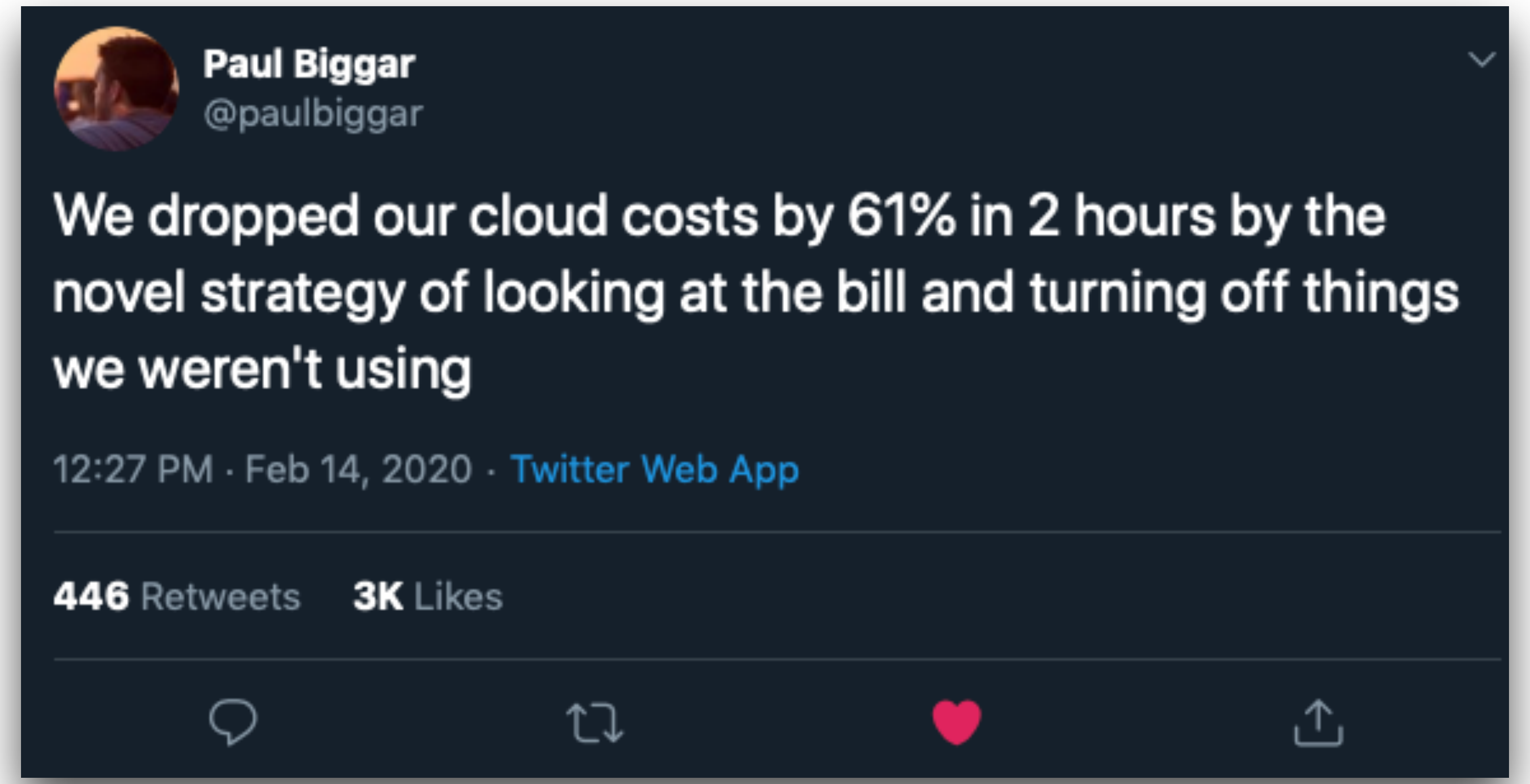
Inter-Region Data Transfer Out: 0 GB/Month

Data Transfer Out: 0 GB/Month

Data Transfer In: 0 GB/Month

Ultimately a democratization of infrastructure.

Very easy to turn something on…and forget about it.


Paul Biggar
@paulbiggar

We dropped our cloud costs by 61% in 2 hours by the novel strategy of looking at the bill and turning off things we weren't using

12:27 PM · Feb 14, 2020 · Twitter Web App

446 Retweets 3K Likes

https://mobile.twitter.com/paulbiggar/status/1228385370439467009

We never had to think about these issues in the past.

Our operators handled it.

Paradox of choice!

ted.com

TED Ideas worth spreading

WATCH DISCOVER ATTEND PARTICIPATE ABOUT MEMBERSHIP SIGN IN

Share

Add to list

Like

Recommend

Barry Schwartz | TEDGlobal 2005

The paradox of choice

18:38

Details
About the talk

Transcript
45 languages

Psychologist Barry Schwartz takes aim at a central tenet of western societies: freedom of choice. In Schwartz's estimation, choice has made us not freer but more paralyzed, not happier but more dissatisfied.

This talk was presented at an official TED conference, and was featured by our editors on the home page.

ABOUT THE SPEAKER

Barry Schwartz · Psychology professor

Barry Schwartz studies the link between economics and psychology, offering startling insights into

15,606,280 views

TEDGlobal 2005 | July 2005

Related tags

Business

Culture

Decision-Making

•••

What inspires you?

Tell us your interests and we'll pick TED Talks just for you.

Get Started

A new perspective · Ideas for self-improvement · Insights about issues tha · Nature · Smart entertainment · Inspiration or motivation · Design · Collaboration · Personal growth · Science · Innovation · A sense of · tter · Child development · Activism · Communication · Technology

Watch next

How to make hard choices
8.6M views
14:41

The art of choosing
3.8M views
24:08

TED Talks are free thanks to our partners and advertisers

Find specialized skills for any scope.

upwork is how. Learn More

New! Activity Feed

Democratization demands more of all of us.

To paraphrase a Founding Father of the United States…

Well informed developers are a prerequisite to successful cloud…

What do you want your developers focussed on?

“With this approach, your developers need to be certified in the application framework, the cloud provider and the container orchestrator.”

–Anonymous Architect

Be prepared. Be aware.

Be careful what you wish for?

We have more control.
And more accountability.

“With great power comes great responsibility.”

–Uncle Ben

Don’t forget about monitoring…

Monitoring is vital to a
thriving cloud architecture.

Monitor Driven Development!

http://benjiweber.co.uk/blog/2015/03/02/monitoring-check-smells/

What would you say your service is doing?

Key components to monitoring:

Logging - what is your service doing?

Dashboards - health of a service.

Alerting - metric is out of band.

Tracing - context and insights into the spinning plates.

We could spend an hour talking about key metrics…

Sampling frequency.

Dash board design.

Pager duty.

Takes time to get monitoring right…tweak, adjust, adapt.

Number of tools from Wavefront to Dynatrace to New Relic.

Spring Boot Actuator!

https://docs.spring.io/spring-boot/docs/current/reference/html/production-ready-metrics.html

6. Metrics

Spring Boot Actuator provides dependency management and auto-configuration for Micrometer, an application metrics facade that supports numerous monitoring systems, including:

- AppOptics
- Atlas
- Datadog
- Dynatrace
- Elastic
- Ganglia
- Graphite
- Humio
- Influx
- JMX
- KairosDB
- New Relic
- Prometheus
- SignalFx
- Simple (in-memory)
- Stackdriver
- StatsD
- Wavefront

To learn more about Micrometer’s capabilities, please refer to its reference documentation, in particular the concepts section.

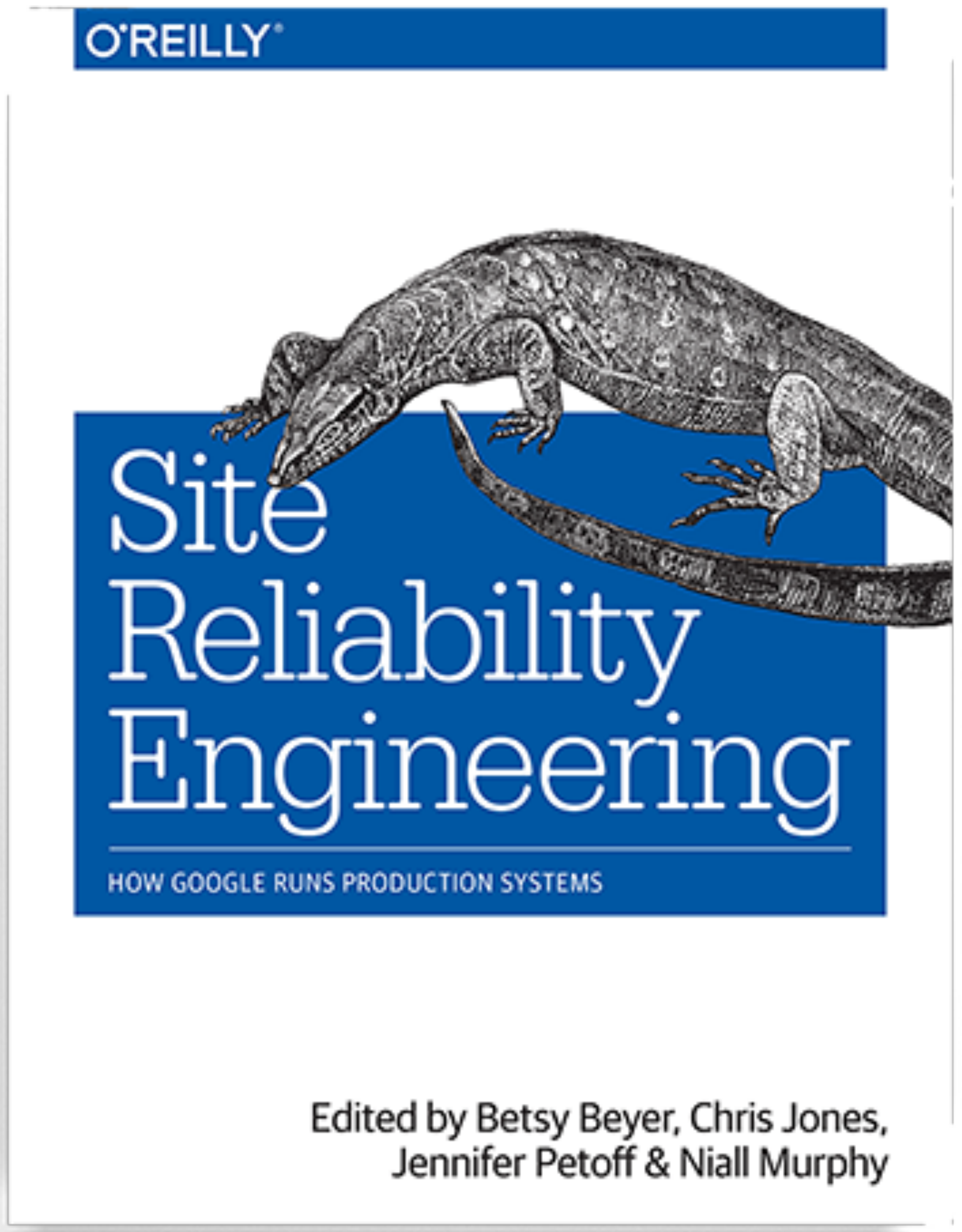
6.1. Getting started

Spring Boot auto-configures a composite `MeterRegistry` and adds a registry to the composite for each of the supported implementations that it finds on the classpath. Having a dependency on `micrometer-registry-{system}` in your runtime classpath is enough for Spring Boot to configure the registry.

Automation is table stakes today.

We can’t compete without it.

SRE anyone?


O'REILLY®

Site Reliability Engineering

HOW GOOGLE RUNS PRODUCTION SYSTEMS

Edited by Betsy Beyer, Chris Jones, Jennifer Petoff & Niall Murphy

https://landing.google.com/sre/book.html

DEVOPS/SRE

Must evolve past “DevOps fills out tickets for developers”.

Site Reliability Engineers.

The traditional sys admin approach doesn’t give us reliable services.

Inherent tension.

Conflicting incentives.

Developers want to release early, release often.

Always Be Changing.

But sys admins want stability.

It works. No one touch anything.

Thus trench warfare.

Doesn’t have to be this way!

We can all get along.

What if we took a different approach to operations?

“what happens when you ask a software engineer to design an operations team.”

https://landing.google.com/sre/book/chapters/introduction.html

Ultimately, this is just software engineering applied to operations.

Replace manual tasks
with automation.

Focus on engineering.

Many SREs are software engineers.

Helps to understand UNIX internals or the networking stack.

Our operational approach has to evolve.

The “Review Board” meeting once a quarter won’t cut it.

How do we move fast safely?

Operations must be able to support a dynamic environment.

That is the core of what we mean by site reliability engineering.

How we create a stable, reliable environment for our services.

It doesn’t happen in spare cycles.

Make sure your SREs have time to do actual engineering work.

On call, tickets, manual tasks - shouldn’t eat up 100% of their day.

SREs need to focus on automating away “toil” aka manual work.

Contain the technical sprawl.

It’s great right? Each team can use just the right tool for the job!

Every developer will have their favorite tools, languages, etc.

Teams will have their pipeline preferences, meaningful metrics…

Leads to an awful lot of ways to do a given thing.

How do we staff up? Go, Haskell, Java, .NET, C++, Ruby, Python?

How many libraries will we need to support all of that?

Can we stay current?

nbcnews.com

BUSINESS
SEP 14 2017, 3:21 PM ET

Equifax Hackers Exploited Months-Old Flaw

by BEN POPKEN

SHARE
Share
Tweet
Email
Print

Equifax announced late Wednesday that the source of the hole in its defenses that enabled hackers to plunder its databases was a massive server bug first revealed in March.

For the rest of the IT world, fixing that flaw was a "hair on fire moment," a security expert said, as companies raced to install patches and secure their servers. But at Equifax, criminals were able to pilfer data from mid-May to July, when the credit bureau says it finally stopped the intrusion.

Equifax, Software Company Blame Each Other for Security Breach 1:52

"We know that criminals exploited a U.S. website application vulnerability," Equifax said in an update on its website Wednesday night. "The vulnerability was Apache Struts CVE-2017-5638." Equifax said it was working with a leading cybersecurity firm, reported to be Mandiant, to investigate the breach. Mandiant declined an NBC News request for comment.

Related: The One Move to Make After Equifax Breach

The Apache Software Foundation, which oversees the Apache Struts project, said in a press release Thursday that a software update to patch the flaw was

advertisement

FROM THE WEB Sponsored Links

Find Out In One Minute If You Pre-Qualify For A Citi Card
Citi

Why These 10 SUVs are the Cream of the Crop
Kelley Blue Book

by Taboola

MORE FROM NBC NEWS

techcrunch.com

Most of the Fortune 100 still use flawed software that led to the Equifax breach

Zack Whittaker

@zackwhittaker / 1 week ago

Almost two years after Equifax's massive hack, the majority of Fortune 100 companies still aren't learning the lessons of using vulnerable software.

In the last six months of 2018, two-thirds of the Fortune 100 companies downloaded a vulnerable version of Apache Struts, the same vulnerable server software that was used by hackers to steal the personal data on close to 150 million consumers, according to data shared by Sonatype, an open-source automation firm.

That's despite almost two years' worth of patched Struts versions being released since the attack.

Sonatype wouldn't name the Fortune 100 firms that had downloaded the

SECURITY / LEER EN ESPAÑOL

Exactis said to have exposed 340 million records, more than Equifax breach

We hadn't heard of the firm either, but it had data on hundreds of millions of Americans and businesses and leaked it, according to Wired.

BY ABRAR AL-HEETI / JUNE 28, 2018 10:14 AM PDT

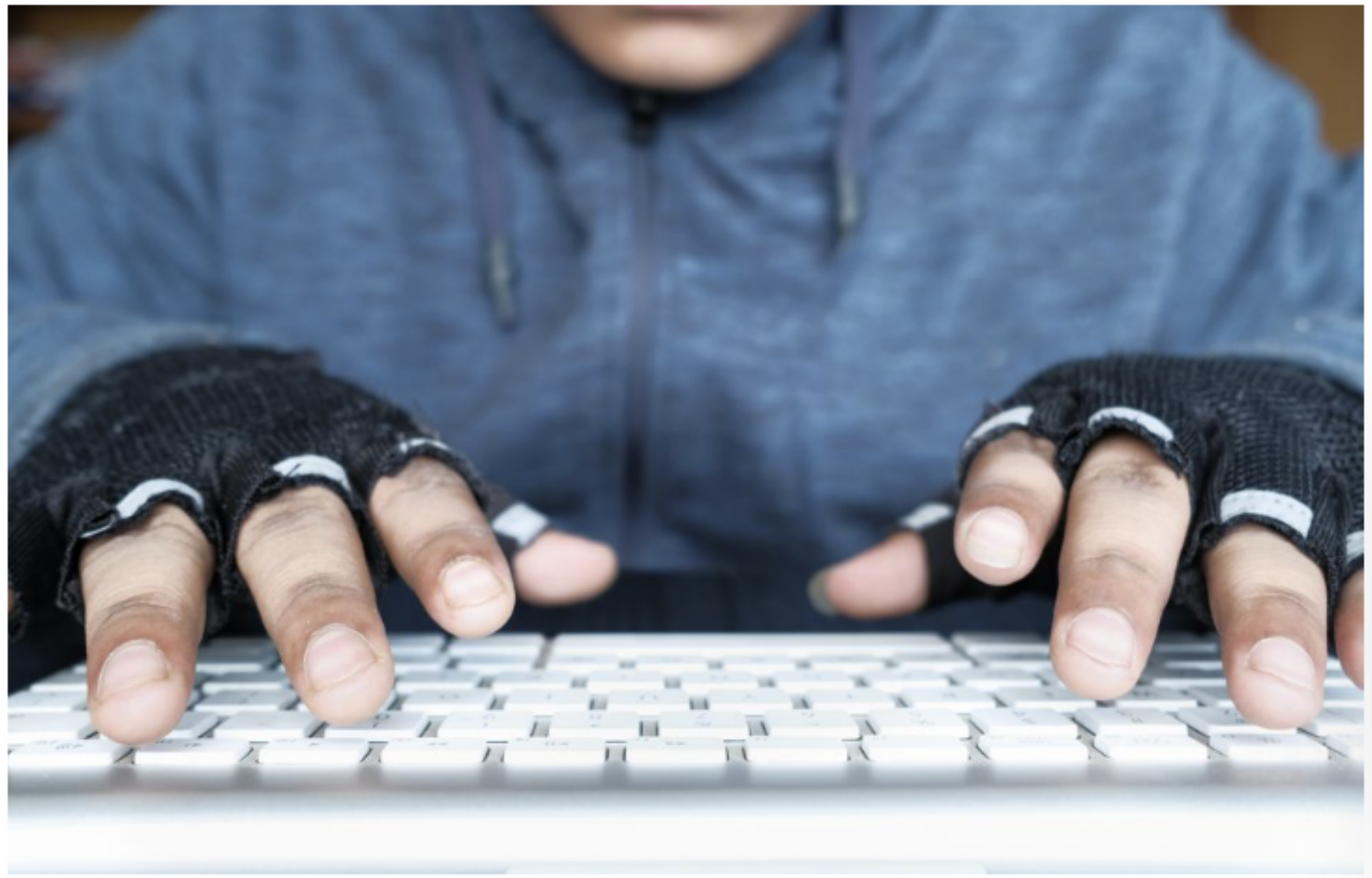
2018 MKC

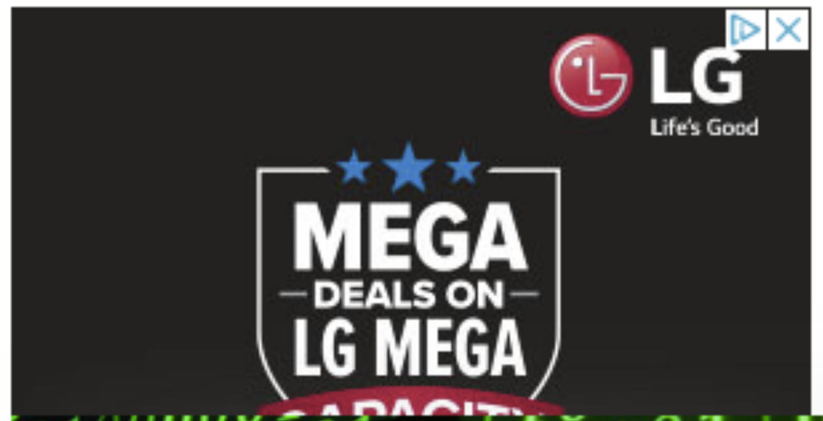
Introduce yourself to a new Lincoln.

CURRENT OFFERS

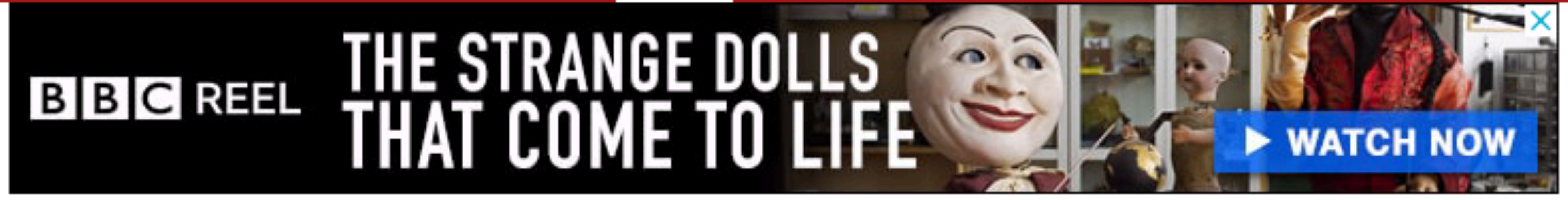
BUILD & PRICE

Roll over for offer disclaimer





Worst hacks of the year 00:00 / 03:24



Technology

Marriott hack hits 500 million Starwood guests

30 November 2018

Share

Sheraton is one of Marriott's brands

The records of 500 million customers of the hotel group Marriott International have been involved in a data breach.

The hotel chain said the guest reservation database of its Starwood division had been compromised by an unauthorised party.

It said an internal investigation found an attacker had been able to access the

Top Stories

Tabloid's owner defends Jeff Bezos report
AMI, owner of a US magazine accused of blackmail by Amazon's founder, says it acted in good faith.
40 minutes ago

What US ruling may mean for Roe v Wade
2 hours ago

Russia probe chief grilled by lawmakers
24 minutes ago

Features

It cannot be a free for all.

You will need some guardrails.

“Use any language as long as it runs on the JVM.”

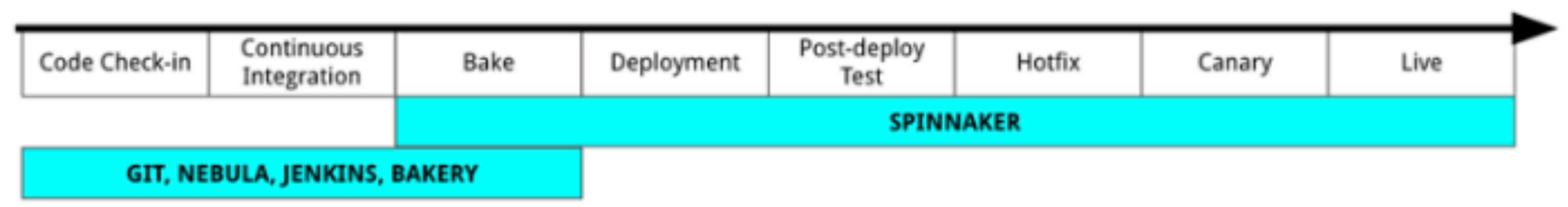
Pick from these 3 flavors. Won’t work for you? Let’s talk.

Focus on “paved roads.”

Netflix Technology Blog Follow
Learn more about how Netflix designs, builds, and operates our systems and engineering organizations
Mar 9, 2016 · 8 min read

How We Build Code at Netflix

How does Netflix build code before it's deployed to the cloud? While pieces of this story have been told in the past, we decided it was time we shared more details. In this post, we describe the tools and techniques used to go from source code to a deployed service serving movies and TV shows to more than 75 million global Netflix members.



The above diagram expands on a previous post announcing Spinnaker, our global continuous delivery platform. There are a number of steps that need to happen before a line of code makes it way into Spinnaker:

- Code is built and tested locally using Nebula
- Changes are committed to a central git repository
- A Jenkins job executes Nebula, which builds, tests, and packages the application for deployment
- Builds are "baked" into Amazon Machine Images
- Spinnaker pipelines are used to deploy and promote the code change

Here is a well worn path, we know it works, we support it.

MINIMUM
MAINTENANCE
ROAD
TRAVEL AT YOUR OWN RISK

You build it, you own it.

Sprawl tends to exacerbate our accumulation of technical debt.

“With great power comes great responsibility.”

–Uncle Ben

You build it, you run it.

Isn’t this just DevOps?

Can argue it is a natural extension of the concept.

Think of SRE as a specific implementation of DevOps.

Establish Principles

We can’t be everywhere…

We can’t be involved with every decision.

We must empower our teams.

Distributed decision making.

We can establish principles.

Guard rails.

Guide posts.

North stars.

Create the environment within which our projects can thrive.

But how do we know if projects are following our principles?

Fitness functions.

We’re all familiar with the second law of thermodynamics…

Otherwise known as a teenagers bedroom.

The universe really
wants to be disordered.

Software is not immune from this!

We go through the thoughtful effort to establish an architecture…

How do we maintain it?

We can’t spend every minute of every day on every project.

How do we ensure teams continue to make good decisions?

We cannot predict the future.

That’s not entirely true.

One constant - change.

Architecture is often defined as the decisions that are hard to change.

Or the decisions we wish we got right.

But we *know* things will change!

Isn’t this approach anti agile?

Contributing factor to “we’re agile, we don’t have architects” theory.

You definitely have people making architectural decisions!

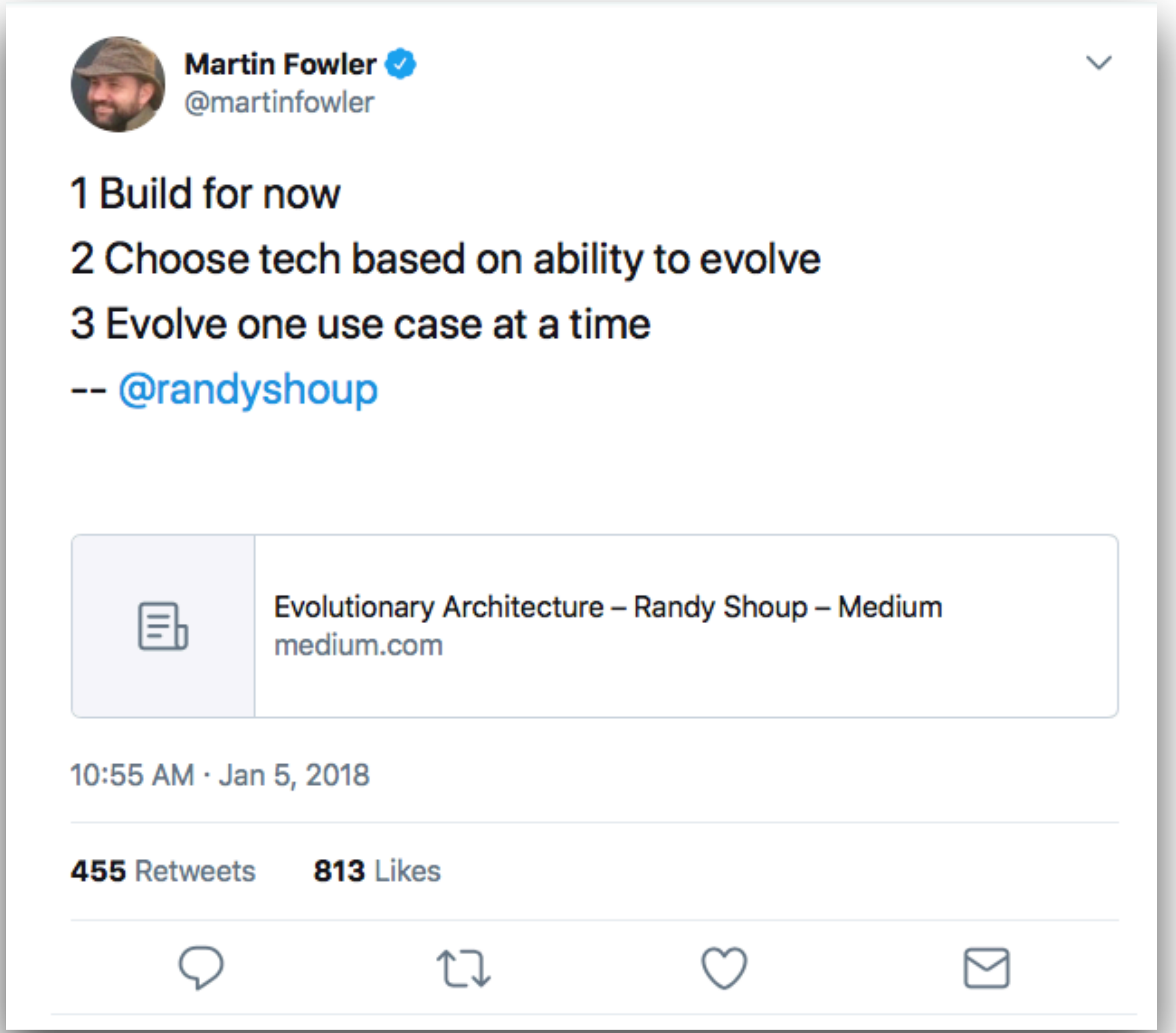
Sure hope they are making good ones…

You’ll know in a year or two.

“Our app has 4 different UI frameworks…”

What do we do about that?

Maybe we should change our assumptions.



Martin Fowler
@martinfowler

1 Build for now
2 Choose tech based on ability to evolve
3 Evolve one use case at a time
-- @randyshoup

Evolutionary Architecture – Randy Shoup – Medium
medium.com

10:55 AM · Jan 5, 2018

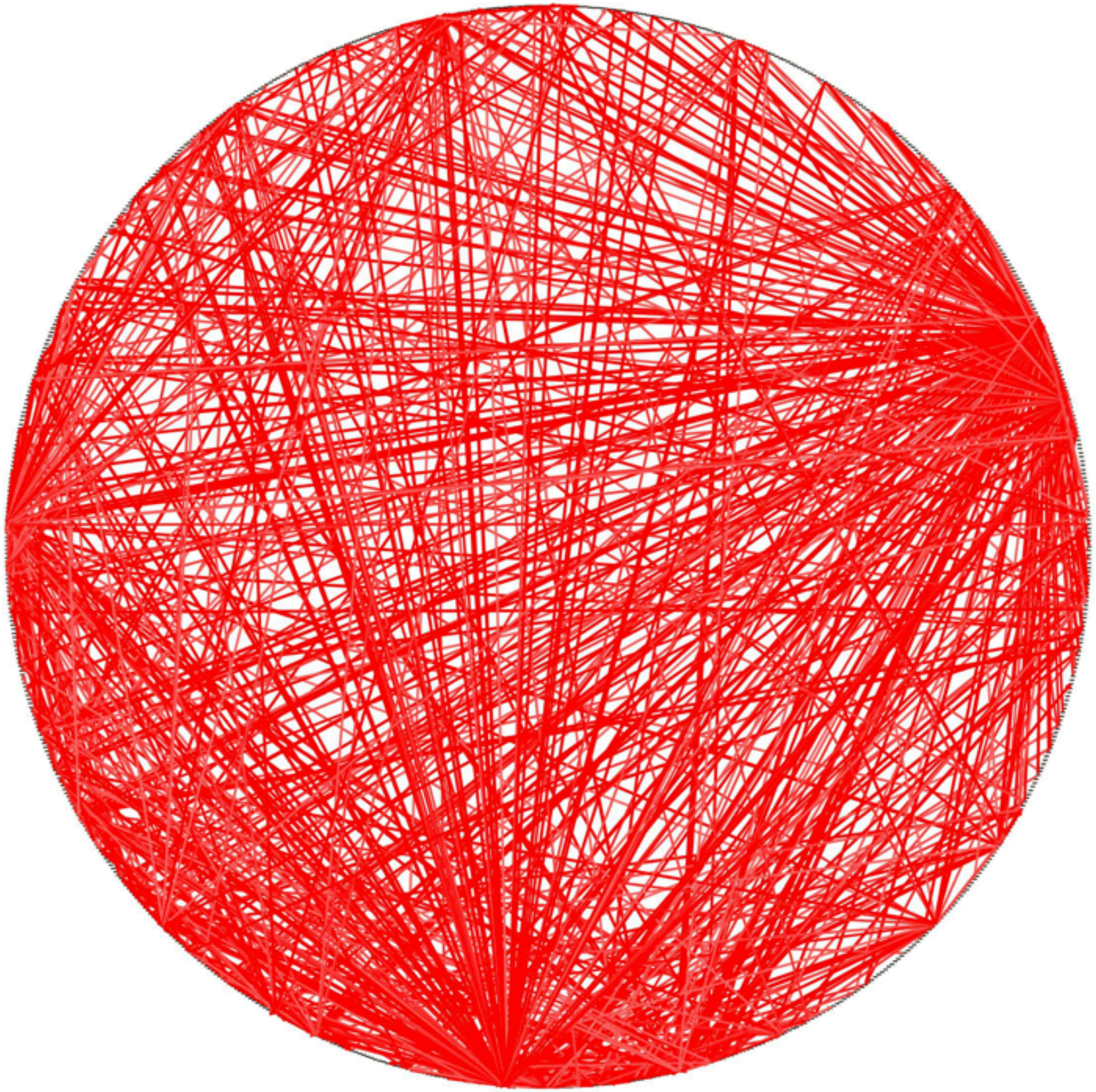
455 Retweets 813 Likes

https://mobile.twitter.com/martinfowler/status/949323421619548161

What if our architectures expected to change?

An evolutionary architecture supports guided, incremental change across multiple dimensions.

Some architectures are more evolvable than others…



http://evolutionaryarchitecture.com

Components are deployed,
features are enabled via toggles.

Allows us to change incrementally.

Also perform hypothesis driven development!

But how do we ensure the architecture still meets our needs?

How do we know if a solution violates part of the architecture?

Fitness functions!

A todo list for developers from architects.

Lightweight, low ceremony, governance.

Concept comes from evolutionary computing.

Is this mutation a success?

Are we closer to or further from our goal?

For architecture, it is all about protecting the ilities.

And balancing the tradeoffs.

We want to capture and preserve the key architectural characteristics.

First, we need to identify those key measures for project success.

Service Level Indicators if you will.

What can we measure?

Sometimes we let what we can measure dictate too much…

Just because we can measure it doesn't mean it matters!

Lines of code anyone?

Once we have our metrics, we can set some goals.

Service Level Objectives.

SLO !== SLA!

Now we can create a fitness function!

Basically, a set of tests we execute to validate our architecture.

How close does this particular design get us to our objectives?

Ideally, all automated. But we may need some manual verifications.

For example…

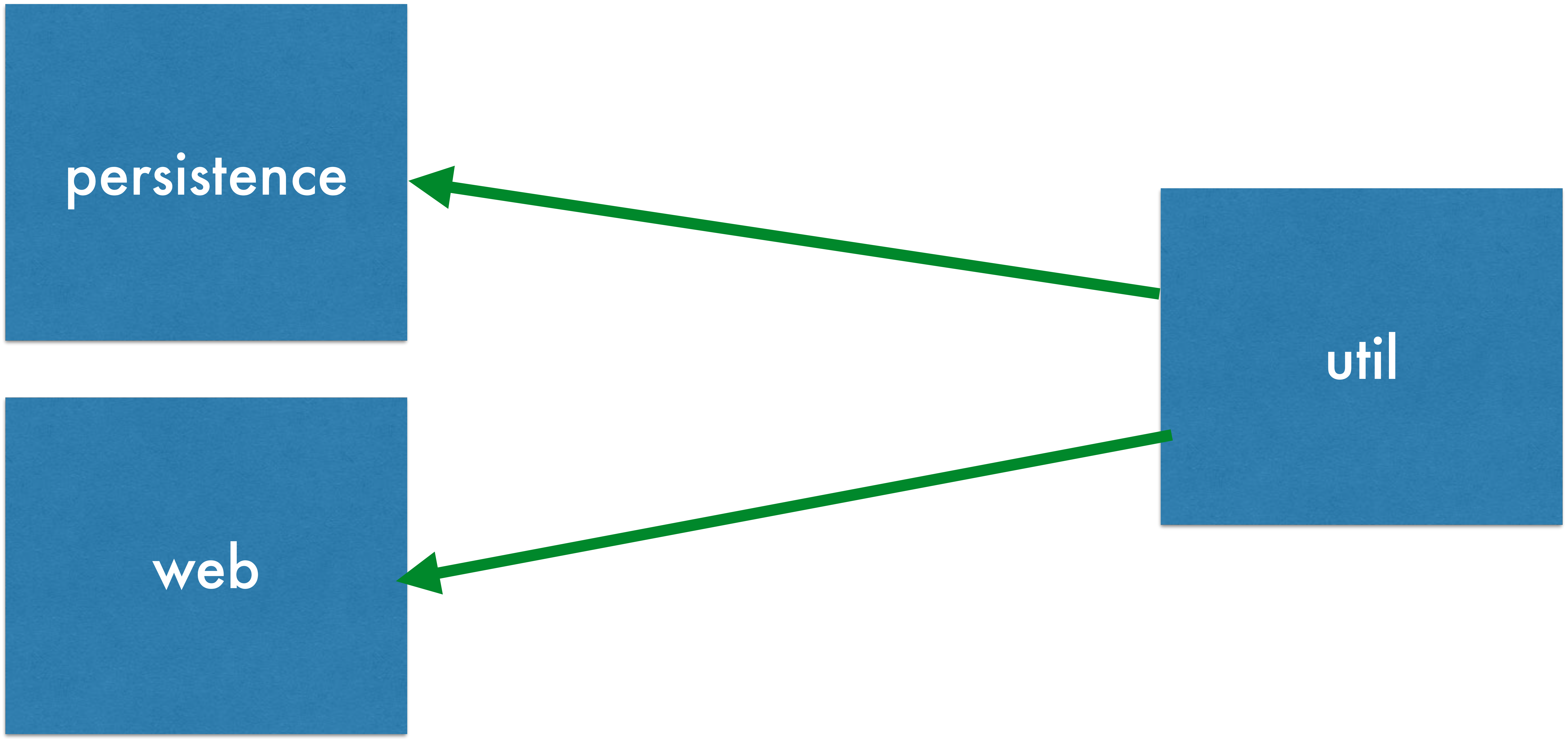
All service calls must respond within 100 ms.

Cyclomatic complexity shall not exceed X.

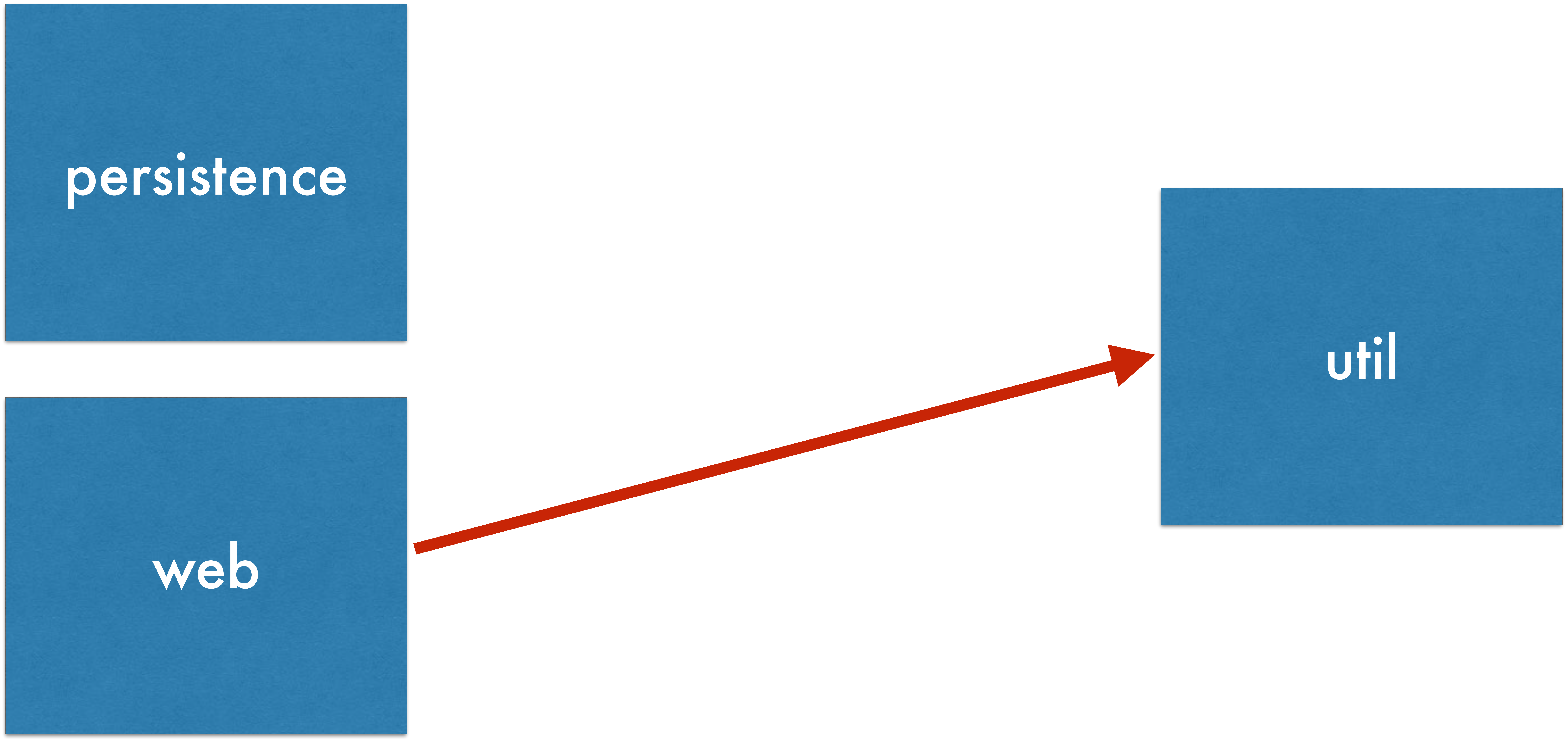
There are no cyclic dependencies.

clarkware.com/software/JDepend.html

Directionality of imports.



packages/namespaces



packages/namespaces

Consumer Driven Contracts.

https://martinfowler.com/articles/consumerDrivenContracts.html

Projects

- Spring Boot
- Spring Framework
- Spring Data
- Spring Cloud
  - Spring Cloud Stream
  - Spring Cloud Azure
  - Spring Cloud for Amazon Web Services
  - Spring Cloud Bus
  - Spring Cloud CLI
  - Spring Cloud for Cloud Foundry
  - Spring Cloud - Cloud Foundry Service Broker
  - Spring Cloud Cluster
  - Spring Cloud Commons
  - Spring Cloud Config
  - Spring Cloud Connectors
  - Spring Cloud Consul
  - **Spring Cloud Contract**
  - Spring Cloud Function
  - Spring Cloud Gateway

Spring Cloud Contract 2.1.1

Overview | Learn | Samples

Spring Cloud Contract is an umbrella project holding solutions that help users in successfully implementing the Consumer Driven Contracts approach. Currently Spring Cloud Contract consists of the Spring Cloud Contract Verifier project.

Spring Cloud Contract Verifier is a tool that enables Consumer Driven Contract (CDC) development of JVM-based applications. It is shipped with Contract Definition Language (DSL) written in Groovy or YAML. Contract definitions are used to produce following resources:

- by default JSON stub definitions to be used by WireMock (HTTP Server Stub) when doing integration testing on the client code (client tests). Test code must still be written by hand, test data is produced by Spring Cloud Contract Verifier.
- Messaging routes if you're using one. We're integrating with Spring Integration, Spring Cloud Stream and Apache Camel. You can however set your own integrations if you want to.
- Acceptance tests (by default in JUnit or Spock) used to verify if server-side implementation of the API is compliant with the contract (server tests). Full test is generated by Spring Cloud Contract Verifier.

Spring Cloud Contract Verifier moves TDD to the level of software architecture.

To see how Spring Cloud Contract supports other languages just check out this blog post.

Features

When trying to test an application that communicates with other services then we could do one of two things:

- deploy all microservices and perform end to end tests
- mock other microservices in unit / integration tests

Both have their advantages but also a lot of disadvantages. Let's focus on the latter.

Deploy all microservices and perform end to end tests

Advantages:

archunit.org
ArchUnit
Getting Started
Motivation
News
User Guide
API
About
Fork me on GitHub

Unit test your Java architecture

Start enforcing your architecture within 30 minutes using the test setup you already have.

Start Now

ArchUnit is a free, simple and extensible library for checking the architecture of your Java code using any plain Java unit test framework. That is, ArchUnit can check dependencies between packages and classes, layers and slices, check for cyclic dependencies and more. It does so by analyzing given Java bytecode, importing all classes into a Java code structure. You can find examples for the current release at ArchUnit Examples and the sources on GitHub.

News

Mar 31, 2019 – New release of ArchUnit (v0.10.2)

Mar 16, 2019 – New release of ArchUnit (v0.10.1)

Mar 16, 2019 – New release of ArchUnit (v0.10.0)

github.com
Why GitHub?
Enterprise
Explore
Marketplace
Pricing
Search
Sign in
Sign up

BenMorris / NetArchTest

Watch 7
Star 52
Fork 10
Code
Issues 0
Pull requests 1
Insights

Join GitHub today

GitHub is home to over 36 million developers working together to host and review code, manage projects, and build software together.

Sign up
Dismiss

A fluent API for .Net that can enforce architectural rules in unit tests.

36 commits
1 branch
4 releases
3 contributors
MIT
Branch: master
New pull request
Find File
Clone or download

BenMorris Merge pull request #9 from BenMorris/prepare-v1.1.4 ... Latest commit 5c6634e 15 days ago

samples	Added console output to samples.	15 days ago
src/NetArchTest.Rules	Added console output to samples.	15 days ago
test	Added more unit tests and forces evaulation of policy results.	15 days ago
.gitattributes	First commit.	6 months ago
.gitignore	Improving test coverage, and added implementations	2 months ago
CONTRIBUTING.md	Added CONTRIBUTING file and updates license.	5 months ago
LICENSE	Added CONTRIBUTING file and updates license.	5 months ago
NetArchTest - all projects.sln	First commit.	6 months ago
README.md	Updated documentation to accommodate new result object.	5 months ago

README.md

github.blog

Back to GitHub.com

The GitHub Blog

Changelog Community Company Engineering Enterprise Product

February 3, 2016 —— Engineering

Scientist: Measure Twice, Cut Over Once

Jesse Toth

Share

Twitter

Facebook

Today we're releasing Scientist 1.0 to help you rewrite critical code with confidence.

As codebases mature and requirements change, it is inevitable that you will need to replace or rewrite a part of your system. At GitHub, we've been lucky to have many systems that have scaled far beyond their original design, but eventually there comes a point when performance or extensibility break down and we have to rewrite or replace a large component of our application.

Problem

A few years ago when we were faced with the task of rewriting one of the most critical systems in our application — the permissions code that controls access and membership to repositories, teams, and organizations — we began looking for a way to make such a large change and have confidence in its correctness.

There is a fairly common architectural pattern for making large-scale changes known as Branch by Abstraction. It works by inserting an abstraction layer around the code you plan to change. The abstraction simply delegates to the existing code to begin with. Once you have the new code in place, you can flip a switch in the abstraction to begin substituting the new code for the old.

Performance - **average** and **maximum** response times.

Average response times across number of users and requests.

Number of timeouts and application faults.

Nearing the next price tier with our cloud provider.

Hard failure of an application will spin up a new instance.

Alert when things start to go out of band!

NETFLIX

https://github.com/Netflix/SimianArmy

Chaos Engineering.

https://medium.com/production-ready/chaos-monkey-for-fun-and-profit-87e2f343db31

Fitness functions remind us what is important in our architecture.

Informs our thinking about tradeoffs.

Different categories of fitness functions.

Atomic vs. Holistic.

Some characteristics must be tested in isolation…others cannot.

Holistic fitness functions test combined features.

We can’t test every possible combination!

Be selective, driven by the value of the architectural characteristic.

Triggered vs. Continual.

Must consider frequency of execution.

Fitness functions can be triggered by something - checkin, QA pass…

Continual tests are just that.

Monitoring Driven Development!

http://benjiweber.co.uk/blog/2015/03/02/monitoring-check-smells/

Static vs. Dynamic.

Static tests have a fixed result - they either pass or they fail.

Nearly any test based on a metric.

Other fitness functions have a shifting definition of success.

Generally defined within a range of acceptable outcomes.

Automated vs. Manual.

Automation is good!

Ideally most of our fitness functions will live in our deployment pipeline.

Not everything is amenable to automation though…

Legal.

Existing projects.

Temporal fitness functions.

Essentially a reminder.

Check for an upgrade of library X.

Break upon upgrade tests.

Clearly we want to identify fitness functions as early as we can.

The discussion about the tradeoffs is invaluable to our understanding.

Help us prioritize features.

May lead us to break a system up to isolate certain features.

We can’t know everything up front.

Fitness functions will emerge as the system changes.

But we should strive to identify as many as we can up front.

We can also classify fitness functions.

Key - critical decisions.

Relevant - considered but unlikely to influence the architecture.

Not Relevant - won’t impact our decisions.

Can still be very useful to identify the non relevant dimensions!

Keep fitness functions visible!

Need to review the fitness functions.

Are they still relevant?

Are there new dimensions we need to track?

Are there better ways of measuring/ testing our current fitness functions?

Aim for at least an annual review.

POSTMORTEMS

We will make mistakes.

Outages will still happen.

Vital we learn from those experiences.

Do not blamestorm.

“Blameless postmortems.”

Goal is to prevent it from happening again.

Document the incident.
What happened?

What was the root cause(s)?

What can we do to prevent this from happening in the future?

Be constructive, not sarcastic.

Consider a basic template.

Title/ID.

Authors.

Status.

Impact.

Root Causes.

Resolution.

Action Items.

Lessons Learned.

Timeline.

Whatever you think will help!

Lessons Learned

What went well

- Monitoring quickly alerted us to high rate (reaching ~100%) of HTTP 500s
- Rapidly distributed updated Shakespeare corpus to all clusters

What went wrong

- We're out of practice in responding to cascading failure
- We exceeded our availability error budget (by several orders of magnitude) due to the exceptional surge of traffic that essentially all resulted in failures

Where we got lucky[166]

- Mailing list of Shakespeare aficionados had a copy of new sonnet available
- Server logs had stack traces pointing to file descriptor exhaustion as cause for crash
- Query-of-death was resolved by pushing new index containing popular search term

Timeline[167]

2015-10-21 **(all times UTC)**

- 14:51 News reports that a new Shakespearean sonnet has been discovered in a Delorean's glove compartment
- 14:53 Traffic to Shakespeare search increases by 88x after post to **/r/shakespeare** points to Shakespeare search engine as place to find new sonnet (except we don't have the sonnet yet)
- 14:54 **OUTAGE BEGINS** — Search backends start melting down under load
- 14:55 docbrown receives pager storm, `ManyHttp500s` from all clusters

Can be difficult to create a postmortem culture.

Consider a postmortem
of the month.

Book club.

Wheel of Misfortune.

Role play a disaster
you faced before.

Ease into it.

Recognize people for their participation.

Senior management needs to encourage the behavior!

Perform retros on your postmortems!

Improve them!

“We cannot learn anything without first not knowing something.”

– Mark Manson

The Subtle Art of Not Giving a F*ck

MOVING FORWARD

This can all seem a bit…
overwhelming.

CHANGE BAD!

Empathy. Compassion.

How do we approach someone new to the idea?

“I’m on one of those agile projects…”

“OK Waterfaller…”

Technology adoption is a journey.

They are where you used to be.

You can help them, you know where the potholes are.

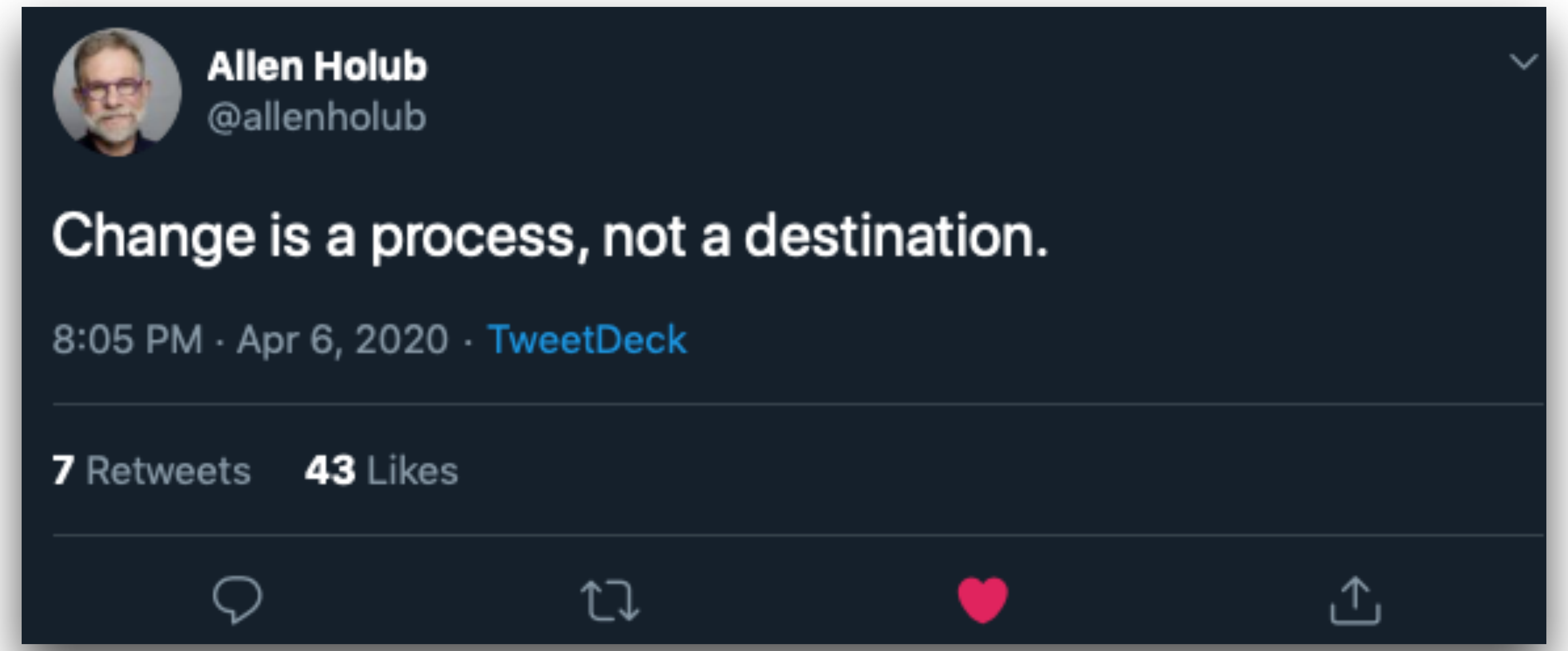
But they have to walk the path.

A day in the life…

Tools will change.

Culture will change.

Be patient.


Allen Holub
@allenholub

Change is a process, not a destination.

8:05 PM · Apr 6, 2020 · TweetDeck

7 Retweets 43 Likes

https://mobile.twitter.com/allenholub/status/1247329663568887808

Positive reinforcement.

You will need some guardrails.

Focus on “paved roads.”

Here is a well worn path, we know it works, we support it.

MINIMUM
MAINTENANCE
ROAD
TRAVEL AT YOUR OWN RISK

You build it, you own it.

You build it, you run it.

Hate to break it to you…your systems will fail.

We cannot prevent it but we can certainly prepare for it.

Resources do not scale to infinity.

There will be competition between teams for hardware, staff, priorities.

Do not underestimate
the battle for headcount.

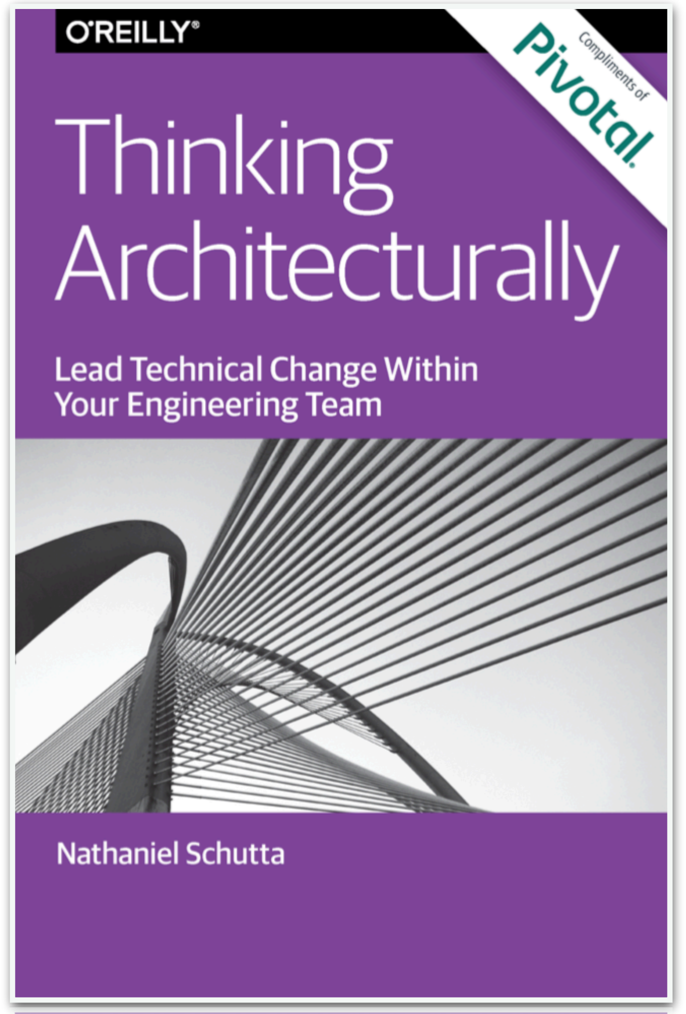
Every manager could use just one more engineer. Or ten.

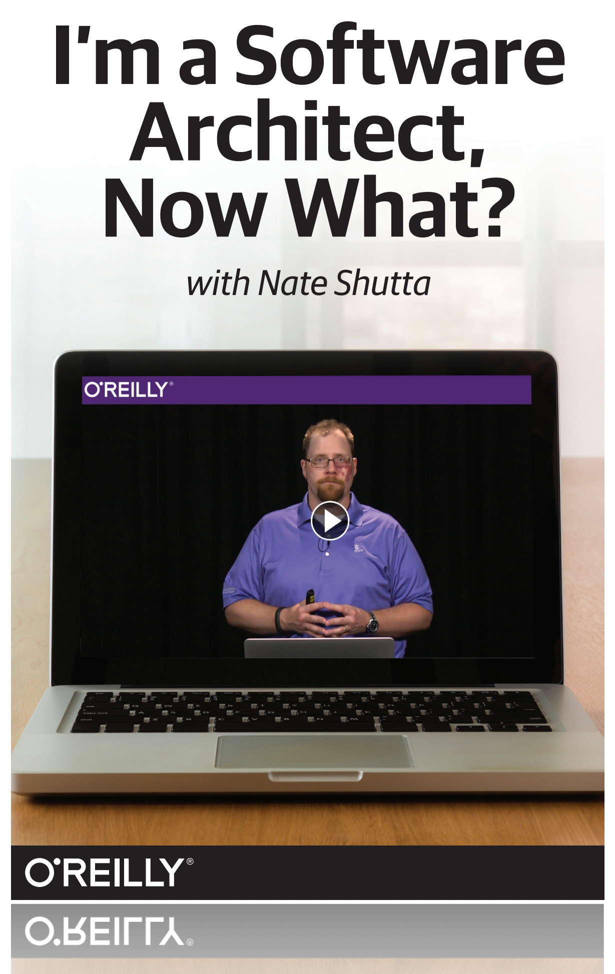
Every VP thinks their portfolio should get the lions share of the budget.

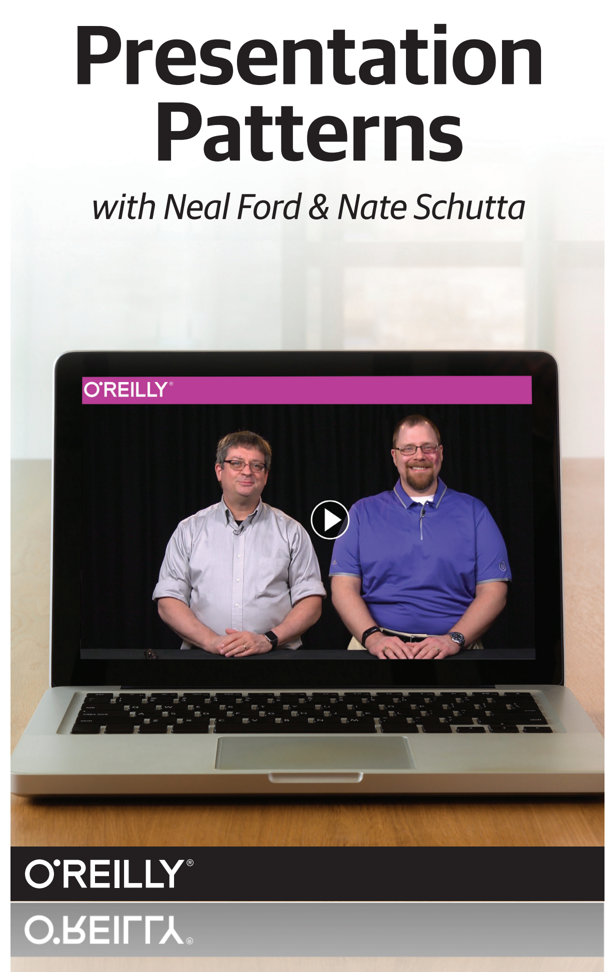
No one could have predicted this.

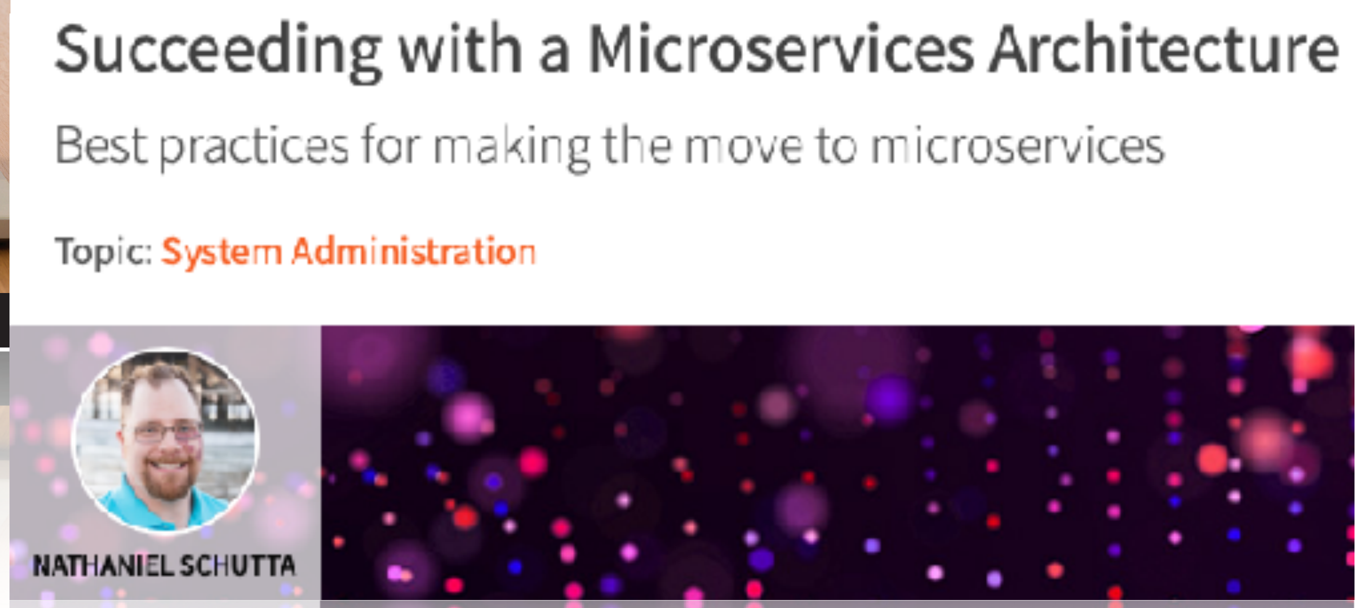
Good luck!

Thanks!








Succeeding with a Microservices Architecture

Best practices for making the move to microservices

Topic: System Administration

Nathaniel T. Schutta
@ntschutta
ntschutta.io

Between Chair and Keyboard

Most Mondays,
around noon Central
https://www.twitch.tv/vmwaretanzu

Nate Schutta
Software Architect
VMware
@ntschutta

tanzu.vmware.com

vmware Tanzu Developer

Learn Topics Blog Tanzu.TV Community

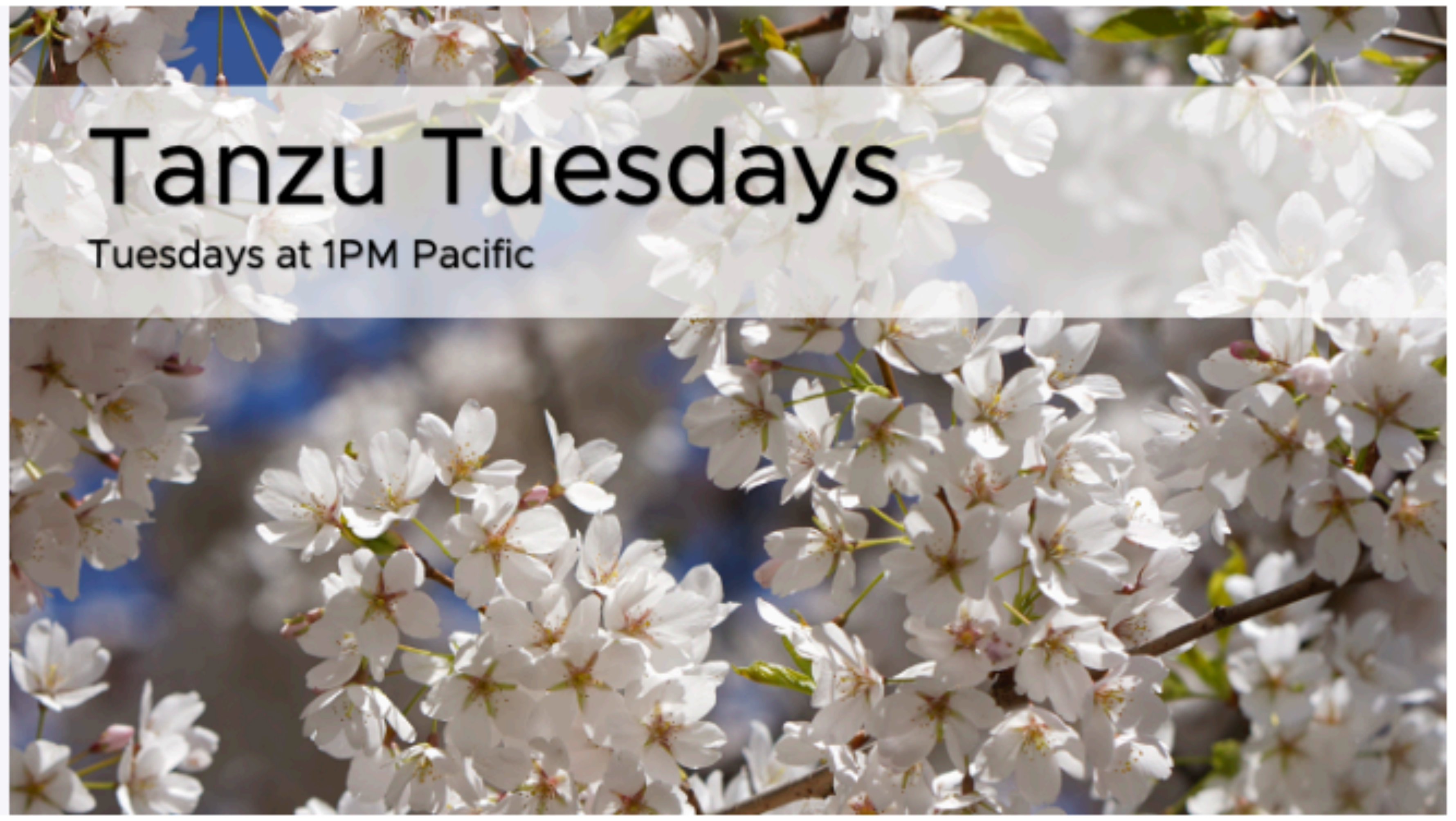
Tanzu.TV Shows

LIVE EVERY TUESDAY AT 1PM PT

Tanzu Tuesdays

Live demos of modern application development technologies.

VIEW EPISODES

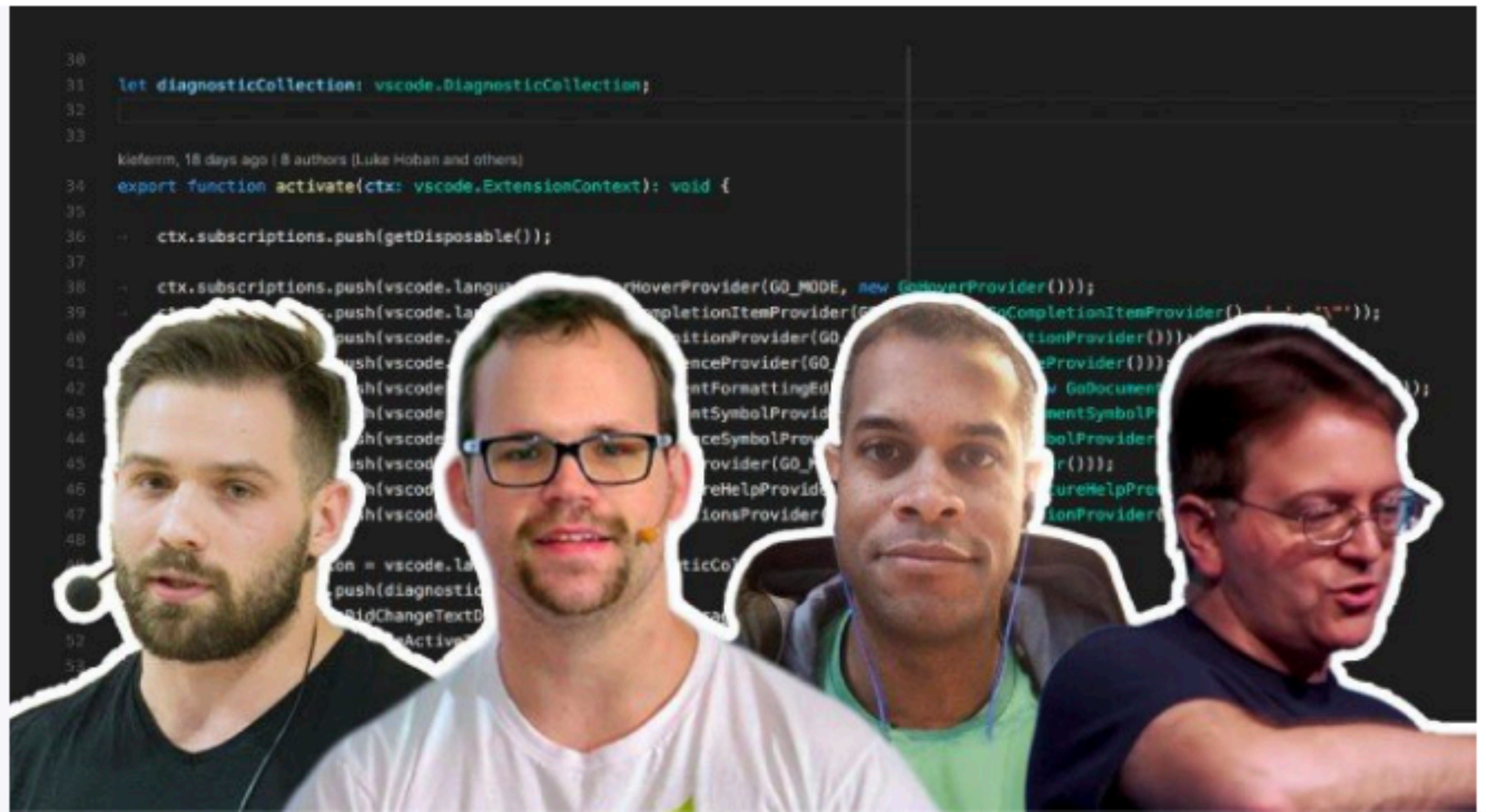


LIVE EVERY WEDNESDAY AT 12PM PT

Code

Every Wednesday at 12pm PT our Developer Advocates code live.

VIEW EPISODES



```
let diagnosticCollection: vscode.DiagnosticCollection;

kieferm, 18 days ago | 8 authors (Luke Hoban and others)
export function activate(ctx: vscode.ExtensionContext): void {

    ctx.subscriptions.push(getDisposable());
```

SpringOne Tour is going virtual

Even in an online format, SpringOne Tour still features the best **cloud native** Java content from **our annual developer conference**. Join us each month for a two-day, live event where your favorites from the cloud native community go in depth on a different topic, featuring a mix of presentations, interactive demos, and panel discussions.

FREE!

All events begin at 9 AM PDT

```
running: unzip -a /Users/jlong/Downloads/service.zip
Archive:  /Users/jlong/Downloads/service.zip
   creating: service/
   creating: service/.mvn/
   creating: service/.mvn/wrapper/
  inflating: service/.mvn/wrapper/maven-wrapper.properties  [binary]
  inflating: service/.mvn/wrapper/MavenWrapperDownloader.java  [binary]
  inflating: service/.mvn/wrapper/maven-wrapper.jar  [binary]
  inflating: service/mvnw            [binary]
  inflating: service/mvnw.cmd        [binary]
  inflating: service/pom.xml         [binary]
   creating: service/src/
   creating: service/src/main/
   creating: service/src/main/java/
   creating: service/src/main/java/com/
   creating: service/src/main/java/com/example/
   creating: service/src/main/java/com/example/service/
  inflating: service/src/main/java/com/example/service/ServiceApplication.java  [binary]
   creating: service/src/main/resources/
  inflating: service/src/main/resources/application.properties  [binary]
   creating: service/src/test/
   creating: service/src/test/java/
   creating: service/src/test/java/com/
   creating: service/src/test/java/com/example/
   creating: service/src/test/java/com/example/service/
  inflating: service/src/test/java/com/example/service/ServiceApplicationTests.java  [binary]
  inflating: service/HELP.md         [binary]
  inflating: service/.gitignore      [binary]
running: idea /Users/jlong/Downloads/service/pom.xml
→  Downloads
```

Topics

May 20–21

Jun 29–30

Jul 22–23