

BRIDGING THE GAP

CI/CD

WIND THE GAP

NATHANIEL SCHUTTA

@NTSCHUTTA

NTSCHUTTA.IO

It used to be so simple.

You had a monolith. Maybe two.

You released new
versions semi annually.

Your team all worked on
the same floor.

Or at least within walking distance.

But that isn't the case today is it?

Now you have dozens, hundreds...
maybe thousands of services.

New versions drop daily.

Your team is scattered
around the globe.

Developing applications
was never easy!

Now? Massively distributed apps
with geographically dispersed teams.

Our approach has to evolve.

Most dangerous six words?

“That’s how we’ve always done it”.

HOW'D WE GET
HERE?



Software development was
often very siloed.

Remember waterfall?

One phase flowed into the next.

Requirements



Design



Build



Test



Maintain

That worked. For some
definition of worked.

Teams were separated.

Business
Analysts

Developers

Testers

UI

Operators

Infrastructure



We didn't always...
work together well.

Sometimes there was animosity
between groups.

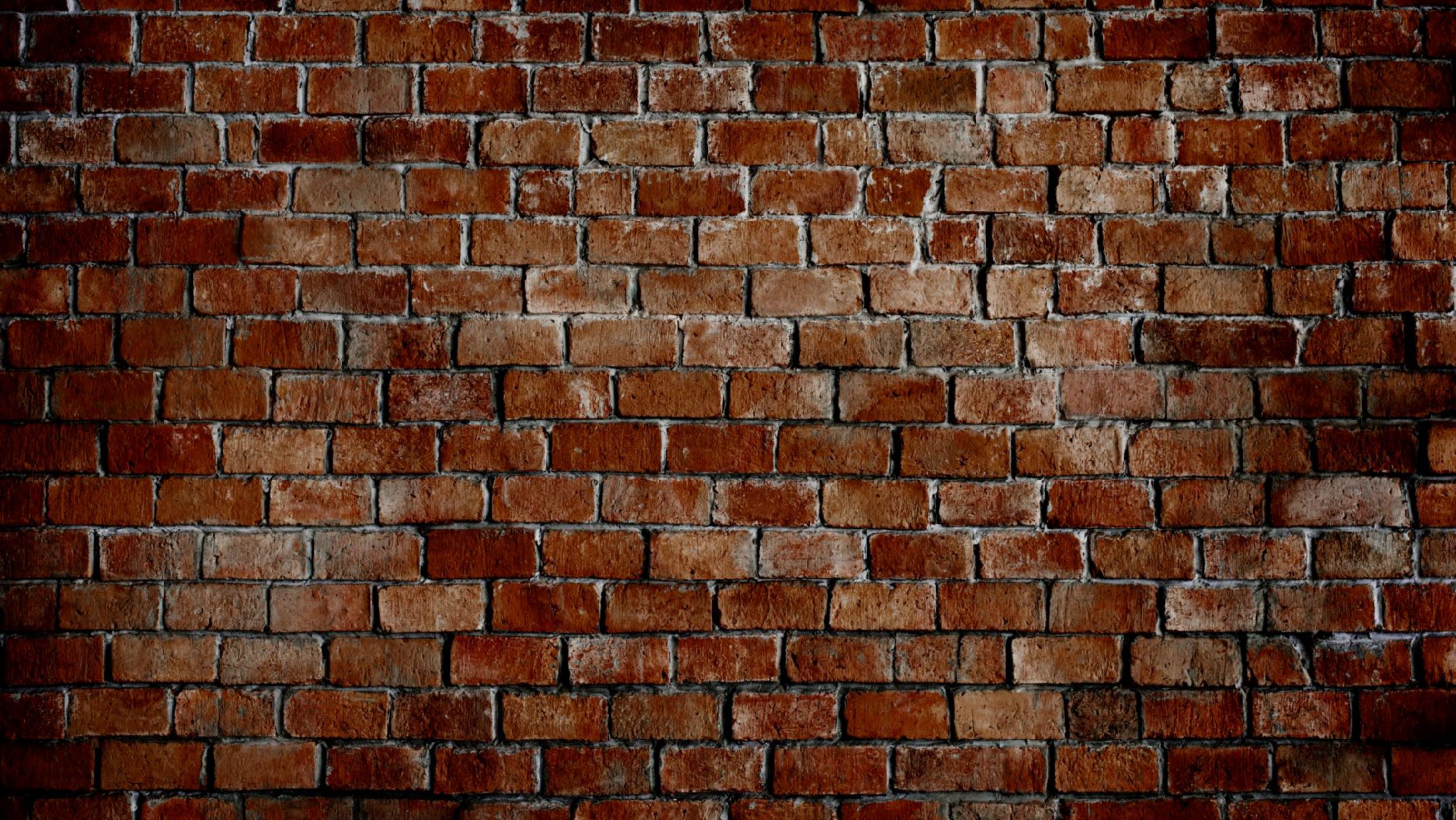




Handoffs were
one way traffic.

Usually scheduled and limited
to a few instances a year.

Or to put it another way...



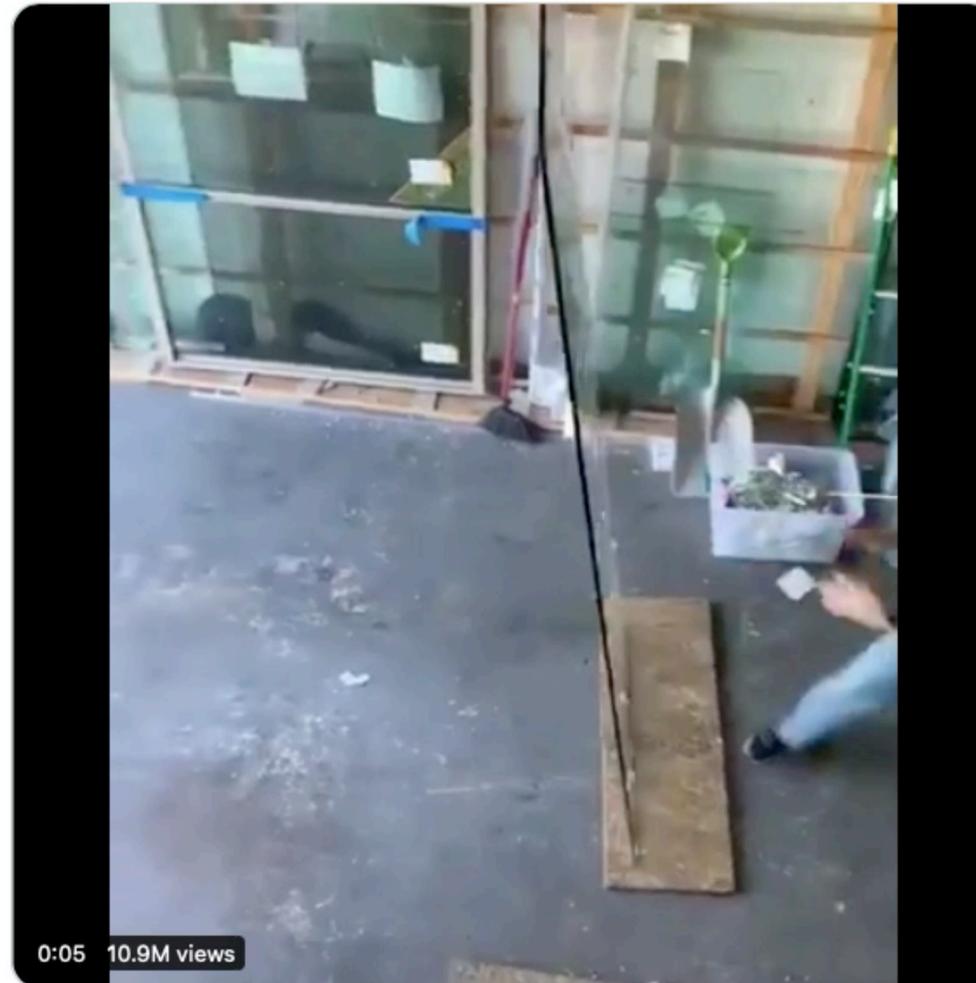
Throw it over the wall to QA.



Angie Jones
@techgirl1908



This is what happens when you throw your feature over the wall to QA instead of collaborating the whole time



From **Best Engineering videos**

7:23 PM · Jan 25, 2021 · Twitter for Android

123 Retweets 21 Quote Tweets 614 Likes

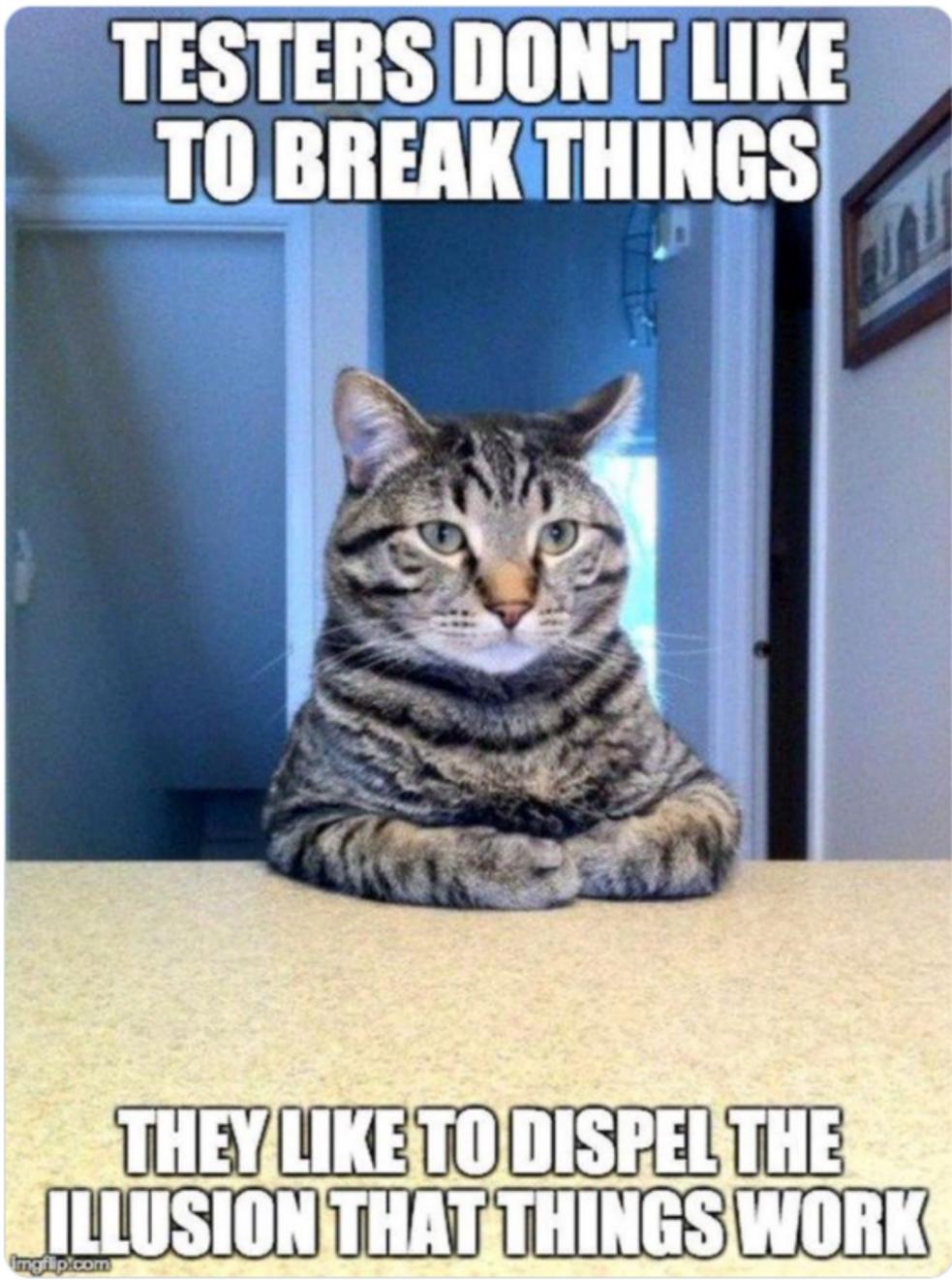


<https://twitter.com/techgirl1908/status/1353876158257926146>



Tristan Lombard 🇺🇸
@TristanLombard2

...



<https://twitter.com/TristanLombard2/status/1502288914874355719>

Fair amount of finger pointing...

“Works as designed.”

“Unable to reproduce.”

“You brought down prod again.”

Needless to say, not an
ideal way to work.

Cycle times were measured in
astronomical units.



The world doesn't stand still.

Business cycles sped up.

We had to adapt.

Always be changing.

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.

What do we do about it?



Ask a software engineer how to
improve something...

And they'll make it look more
like software engineering.



#SNL
More Cowbell - SNL
15,731,005 views • Feb 22, 2019

192K 3.2K SHARE SAVE ...

Saturday Night Live
Get season 46 on YouTube [BUY](#)



Will Ferrell Ruined Christopher Walken's Life with SNL's More...
The Tonight Show Starring Ji...
3.9M views • 1 year ago

All Will Ferrell Jimmy Fallon Comedy

Old Movie Stars Dance to Uptown Funk
Nerd Fest UK
57M views • 5 years ago

Matt Foley: Van Down By The River - SNL
Saturday Night Live
18M views • 7 years ago

Will Ferrell Hilarious Acceptance Speech At The...
cocksandballs123
15M views • 9 years ago

Wayne's World: Aerosmith - SNL
Saturday Night Live
2.6M views • 7 years ago

Funniest Game Show Answers of All Time
theSeanGshow
50M views • 7 years ago

The Barry Gibb Talk Show: Bee

I gotta have more cowbell...

Testing approach isn't working?

Let's make **testing** more like
software engineering!

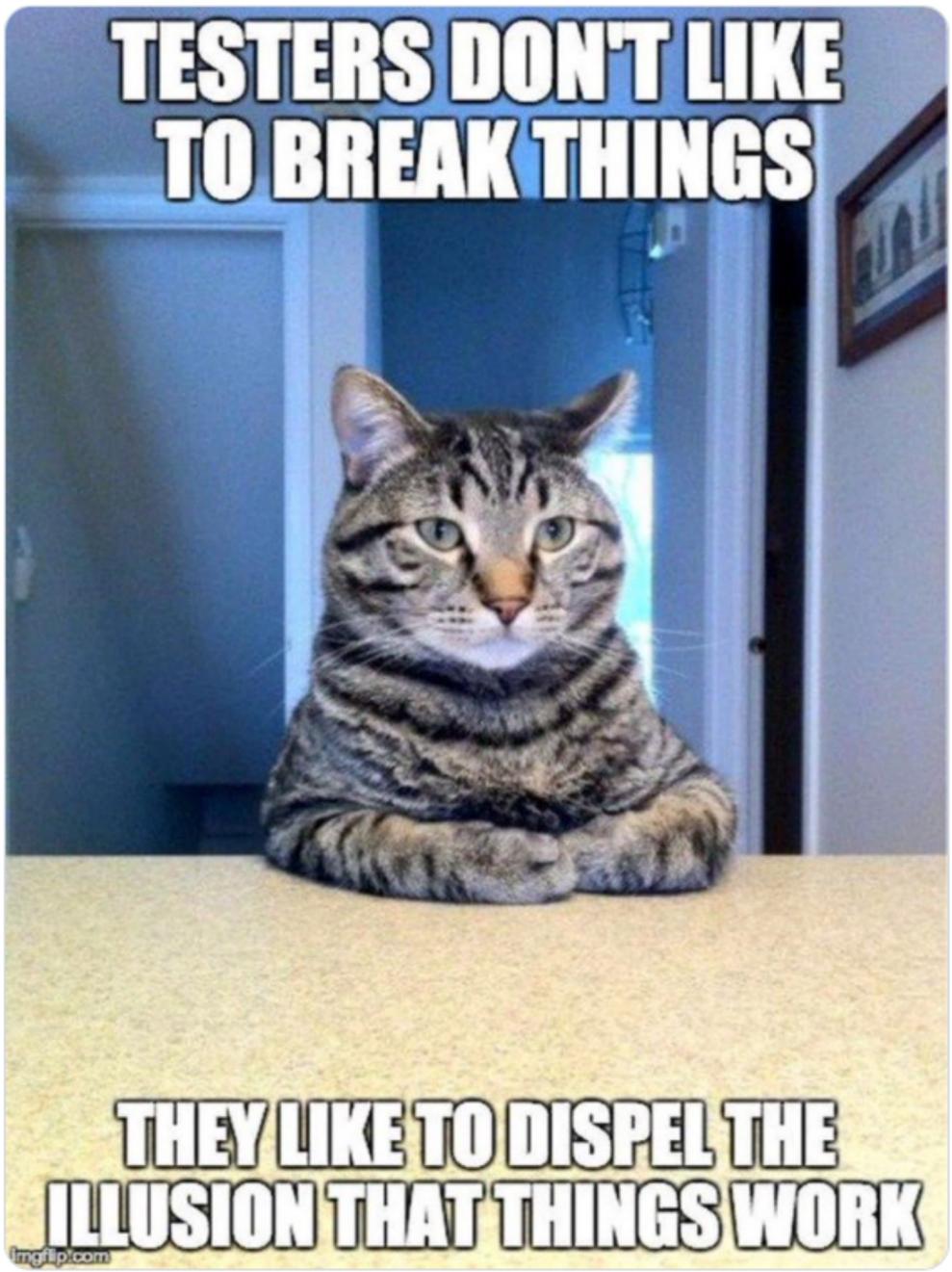
Developers are responsible for
writing automated tests.

Frees the testing professionals
to push the application.



Tristan Lombard 🇺🇸
@TristanLombard2

...



<https://twitter.com/TristanLombard2/status/1502288914874355719>

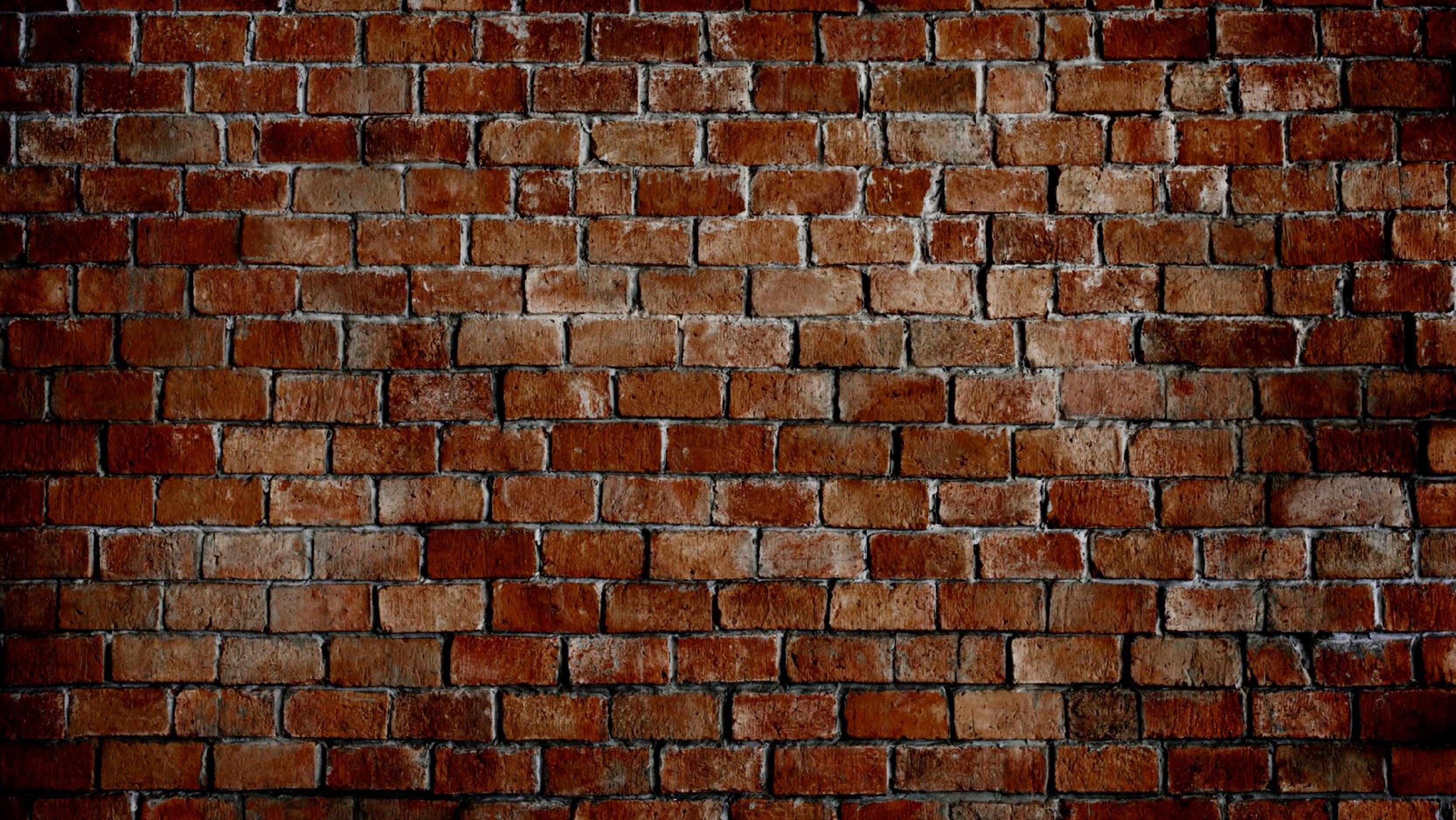
What about production?



Not my problem, besides...

Works on my box.

What did we do?



Think if parent's did this...

The kid is five, they're the
world's problem.

Worked in my house?

Ops wants to keep production
up and running.

And developers just write bugs
and bring prod down.

If we don't release, up
time will be 100%!

Ok, not quite - just make it hard
to release software...

So what did we do?



Let's make ops more like
software engineering!

Developers become the first
line of production support.

Getting paged at 3 am tends to result in more reliable software.

Obviously oversimplified!

But we were able to
release more often.

Nested feedback loops.

Demonstrate progress,
make adjustments.

But that isn't the only feedback loop.

Developers work with operators
to reduce friction.

Operators work with developers to
bake in fault tolerance.

Smaller batch sizes, less risk.

Run experiments. A/B testing.

Respond to business changes.

Increased development velocity!

Production quality increased.

Better software.

More business value.

Happier customers!

Ultimately software has become
a more collaborative game.

Aligned accountability.

Leverage error budget.

Establish your target availability.

Say 99%.

Your error budget is 1.0%.

Spend it however you want!
Just don't go over that limit.

Goal is not zero outages. Goal is to stay within the error budget.

Go ahead with those experiments.

Try different things.

Once we use up the
budget though...

Have to slow our roll.

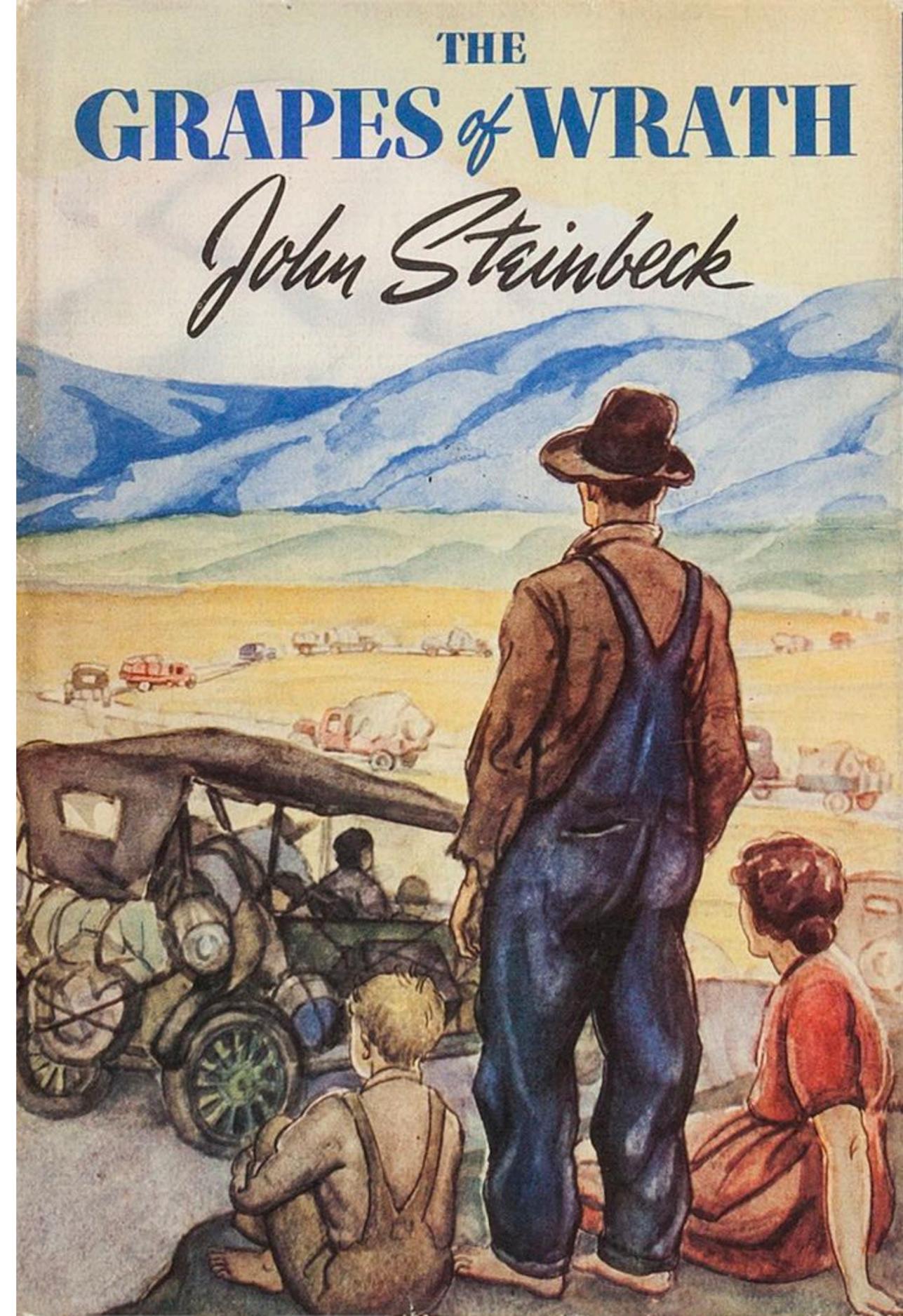
Aligns incentives.

Helps everyone understand
the cost/benefit tradeoff.

Instead of measuring ops on
uptime and devs on velocity...

We are responsible for
delivering business value.

We own this.



Focus on more features to
production more frequently.

How did we do that?

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.



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.



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

We need consistency.

We need CI and CD pipelines.

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](#).

What role does automation play in DevOps?

Share

Twitter

Facebook

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?



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>

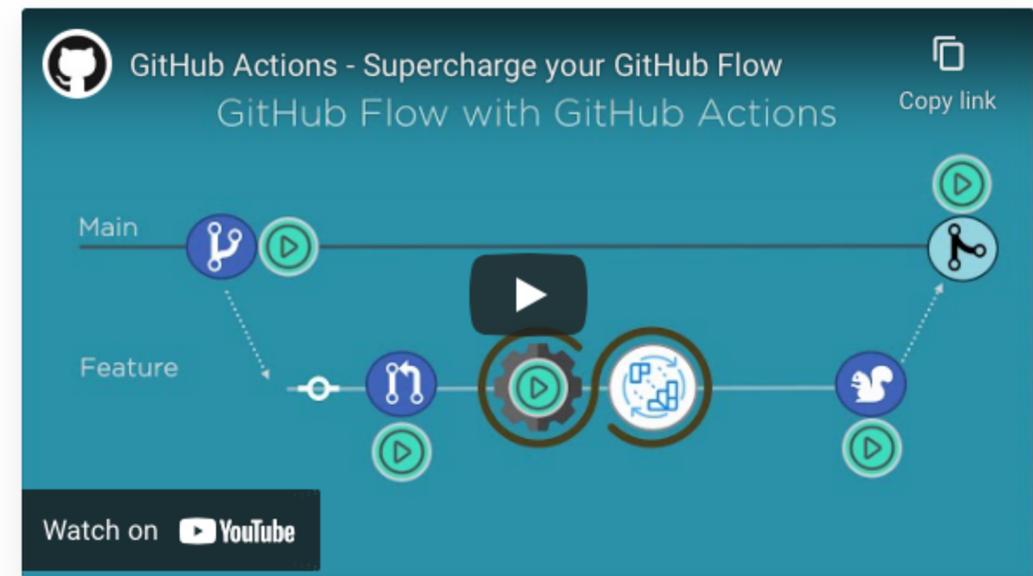
- GitHub Docs
- ← All products
- GitHub Actions
- QUICKSTART FOR GITHUB ACTIONS
- GUIDES
 - About continuous integration
 - Setting up CI using templates
 - Building and testing Node.js
 - Building and testing .NET
 - Building and testing PowerShell
 - Building and testing Python
 - Building and testing 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



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.

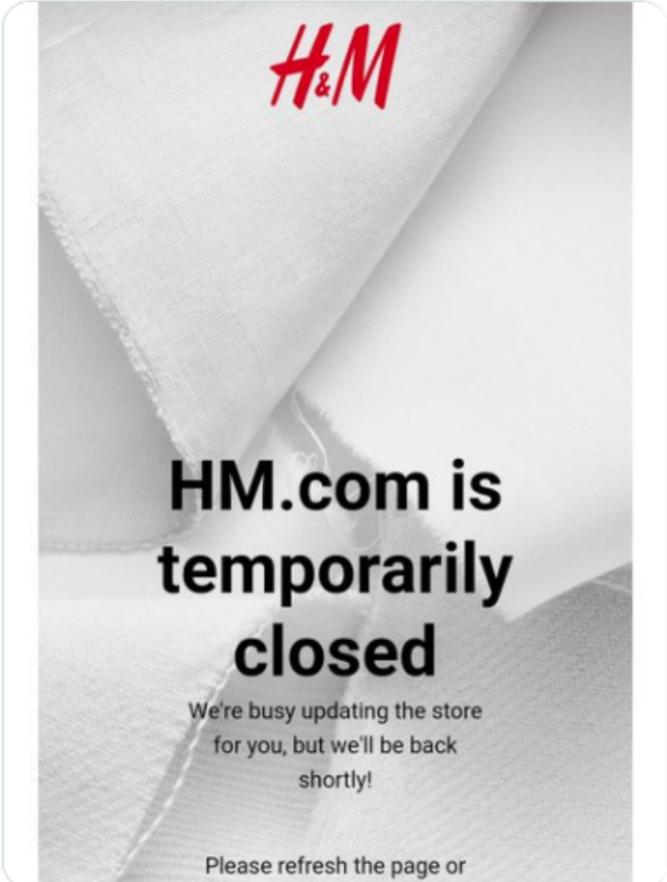
Overlay the current version with
the new version...

And hope for the best!

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

“The application will be down
for maintenance...”

 **Layla #WomenOfDotNet** @LaylaCodesIt · Jul 6, 2021 ...
Taking a store down to update it feels a little archaic these days, @hm !!



 7   24 

 **Nate Schutta**
@ntschutta ...

Replying to @LaylaCodesIt and @hm

Certain deployment strategies aren't evenly distributed in the wild...

5:11 PM · Jul 6, 2021 · Twitter for iPhone

<https://twitter.com/ntschutta/status/1412534801807069184>

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

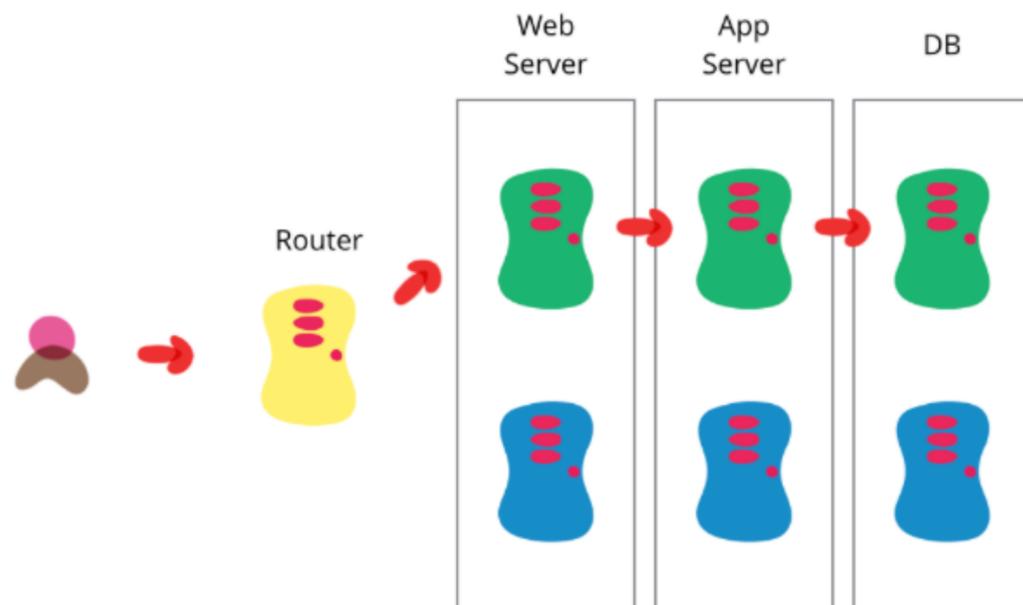
1 March 2010



Martin Fowler

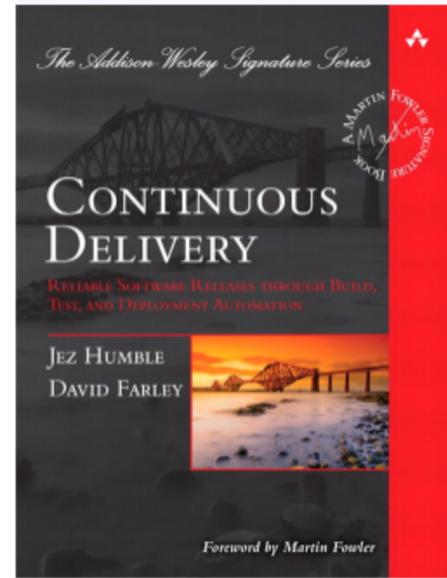
CONTINUOUS DELIVERY

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.





< All Books



Continuous Delivery

Reliable Software Releases through Build, Test, and Deployment Automation

by Jez Humble and David Farley
2010



[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?





<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.





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>

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.

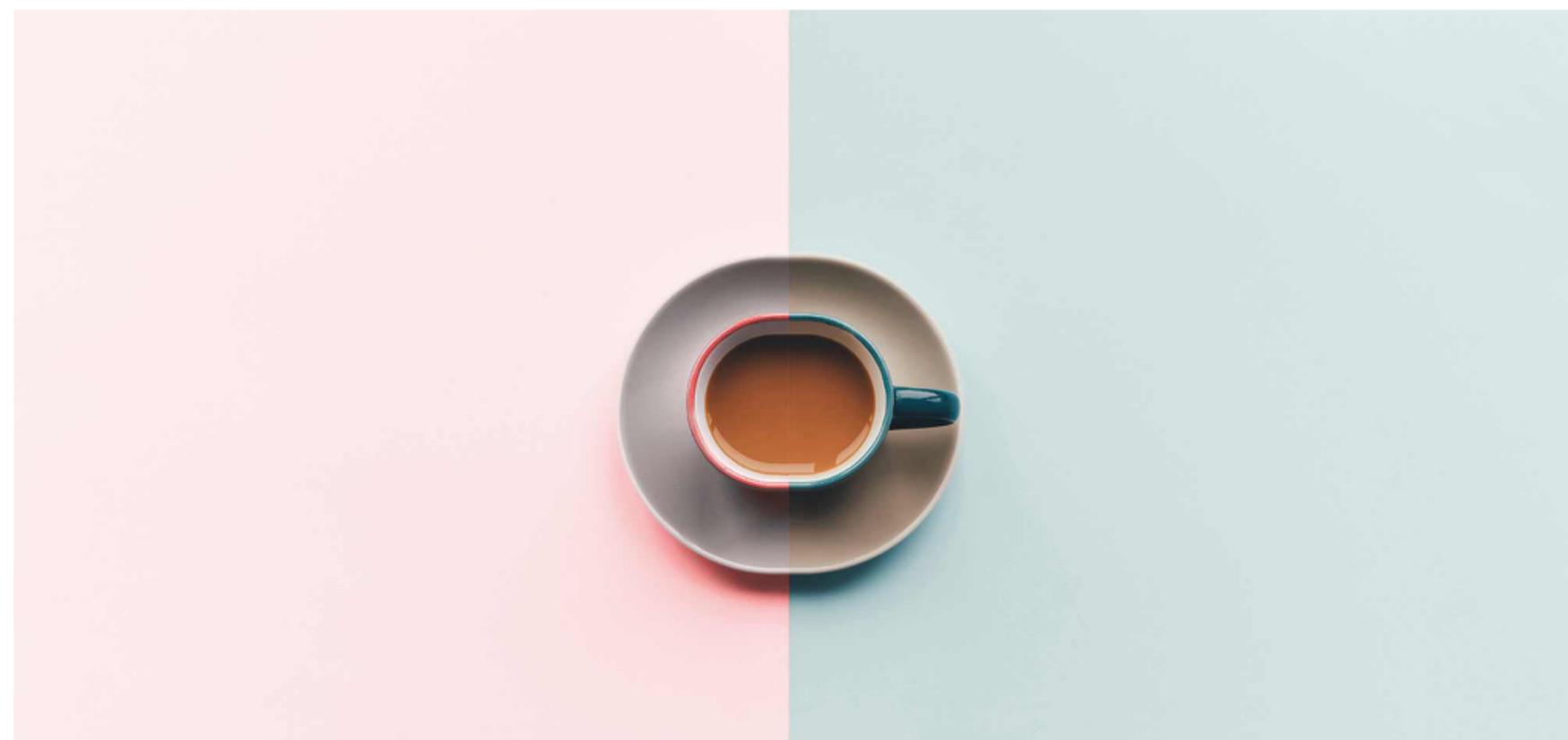
Extremely common.

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>



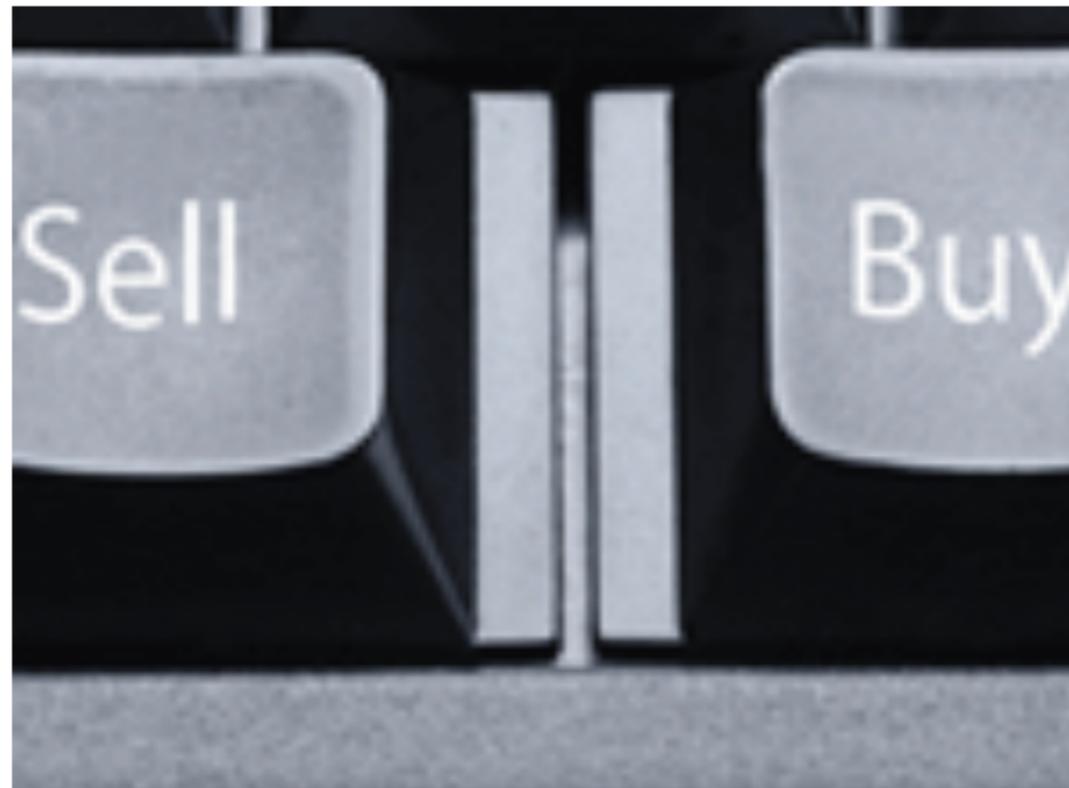
Advertisement

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



- Y
- f
- 🐦
- 👤
- in
- +

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?



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

Just in case.

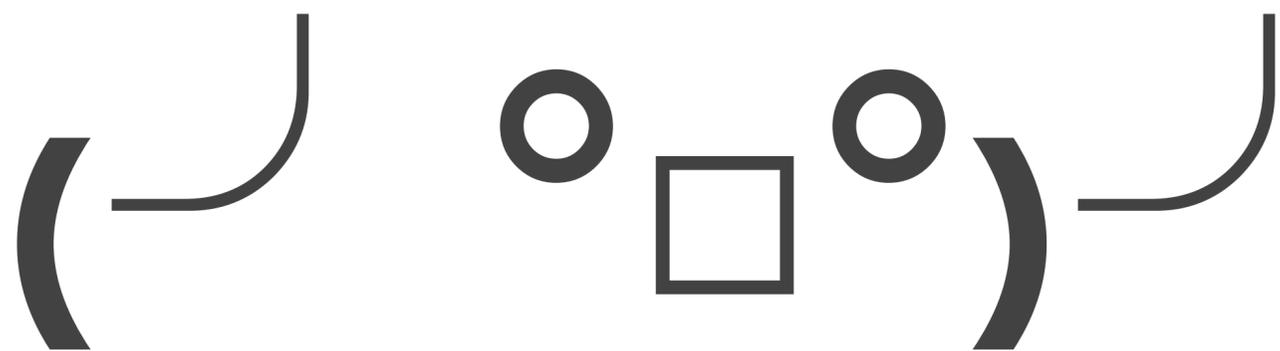
We have a six week (aka month)
lead time on all requests.

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.

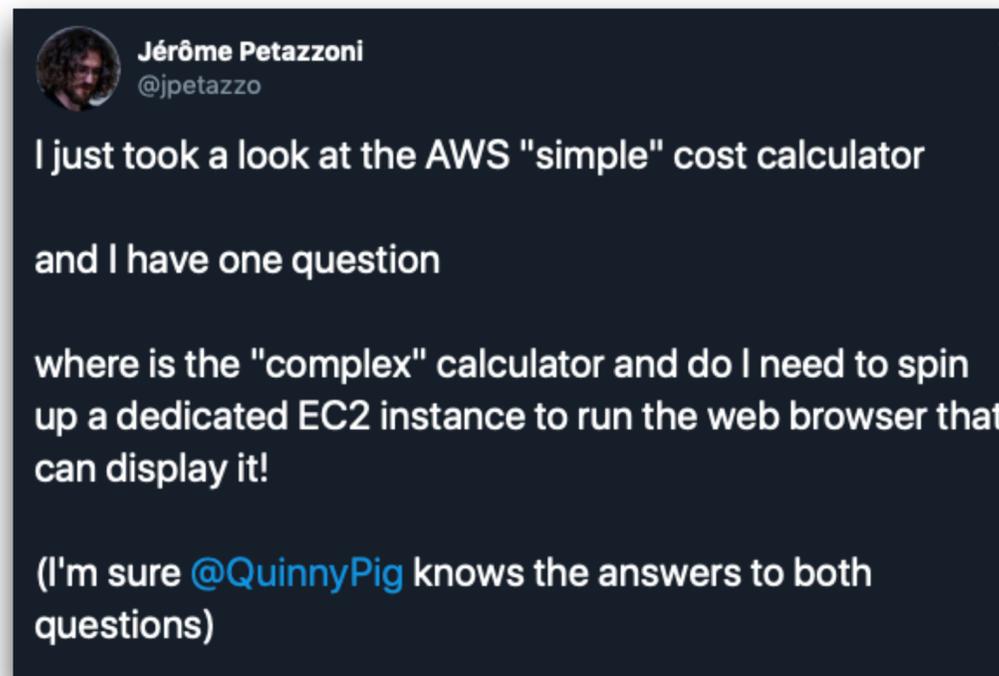
Limitless compute.*

* Additional fees may apply.

Said fees can be...opaque.



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



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

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 EC2 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

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

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:

For Windows and Windows with SQL Web:

Elastic IP:*

Enter values below Calculate

Total time the additional Elastic IPs are attached to running EC2 instances**:

Hours/Month

Total Non-attached time for all the Elastic IPs:

Hours/Month

Number of Elastic IP Remaps: Per Month

Data Transfer:

Inter-Region Data Transfer Out: GB/Month

Data Transfer Out: GB/Month

Data Transfer In: GB/Month

Ultimately a democratization
of infrastructure.

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

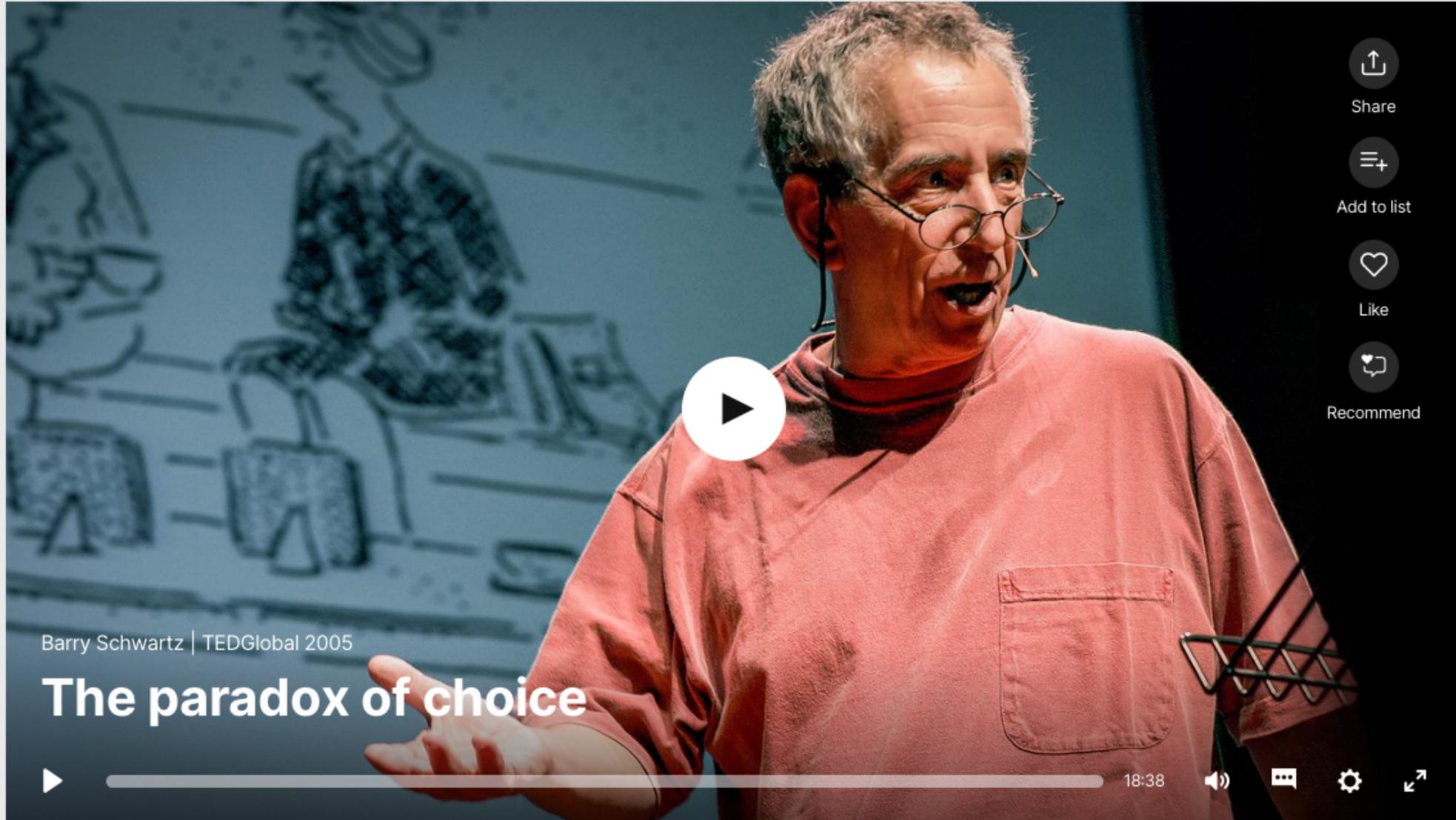


<https://mobile.twitter.com/paulbiggar/status/1228385370439467009>

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

Our operators handled it.

Paradox of choice!



Barry Schwartz | TEDGlobal 2005

The paradox of choice

18:38 [Progress bar and controls]

- Share
- Add to list
- Like
- Recommend

- Details**
- Transcript

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 f

ter Child development Activism Communication **Technology**

Watch next

How to make hard choices
8.6M views



The art of choosing
3.8M views



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>

← Back to index

- 1. Enabling Production-ready Features
- 2. Endpoints
- 3. Monitoring and Management over HTTP
- 4. Monitoring and Management over JMX
- 5. Loggers
- 6. Metrics**
- 6.1. Getting started
- 6.2. Supported monitoring systems
- 6.3. Supported Metrics
- 6.4. Registering custom metrics
- 6.5. Customizing individual metrics
- 6.6. Metrics endpoint
- 7. Auditing
- 8. HTTP Tracing
- 9. Process Monitoring
- 10. Cloud Foundry Support
- 11. What to Read Next

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

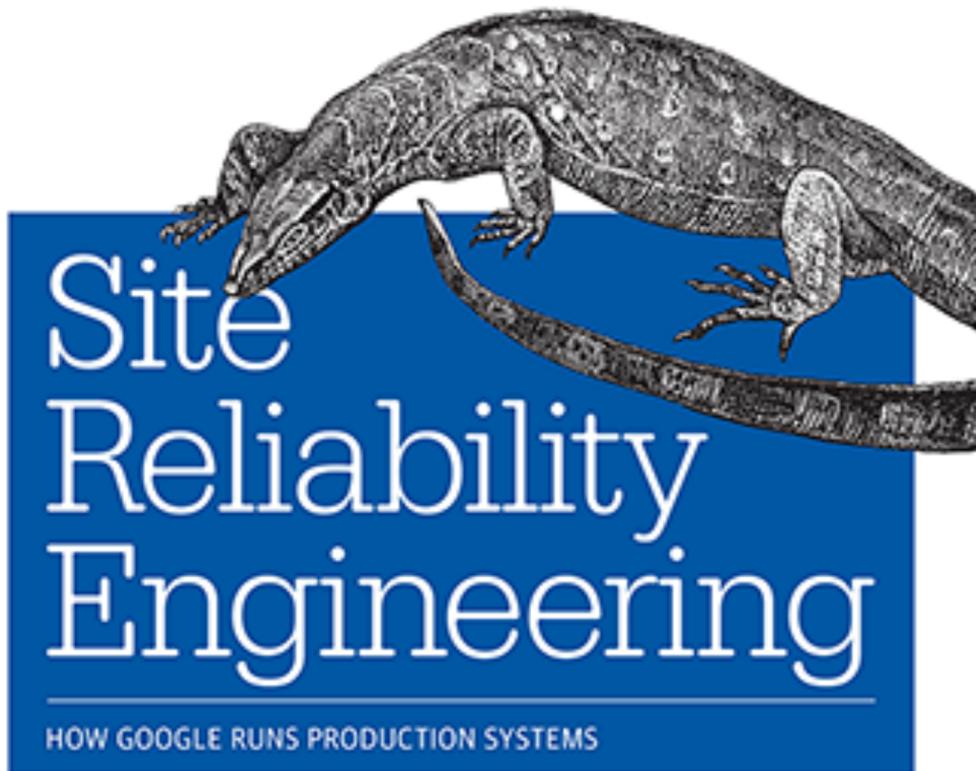
Spring Boot auto-configures a composite `MeterRegistry` and adds a registry to the composite for each of the supported implementations that it finds on the classpath. Having a dependency on `micrometer-registry-{system}` in your runtime classpath is enough for Spring Boot to configure the registry.

Automation is table stakes today.

We can't compete without it.

SRE anyone?

O'REILLY®



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

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

YOU ARE STOOD ON
THE BRIDGE



DEVOPS/SRE

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

Site Reliability Engineers.

Better, clearer title.

What customer doesn't want
reliable applications?

How many want...
developer operators?

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.

That is not, in fact, an
inviolable requirement.

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?

BUSINESS

SEP 14 2017, 3:21 PM ET

Equifax Hackers Exploited Months-Old Flaw

by BEN POPKEN

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.

SHARE

Share

Tweet

Email

Print



▶ **Equifax, Software Company Blame Each Other for Security Breach** 1:52 f t </>

"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





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



REVIEWS

NEWS

VIDEO

HOW TO

SMART HOME

CARS

DEALS

DOWNLOAD



JOIN / SIGN IN

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





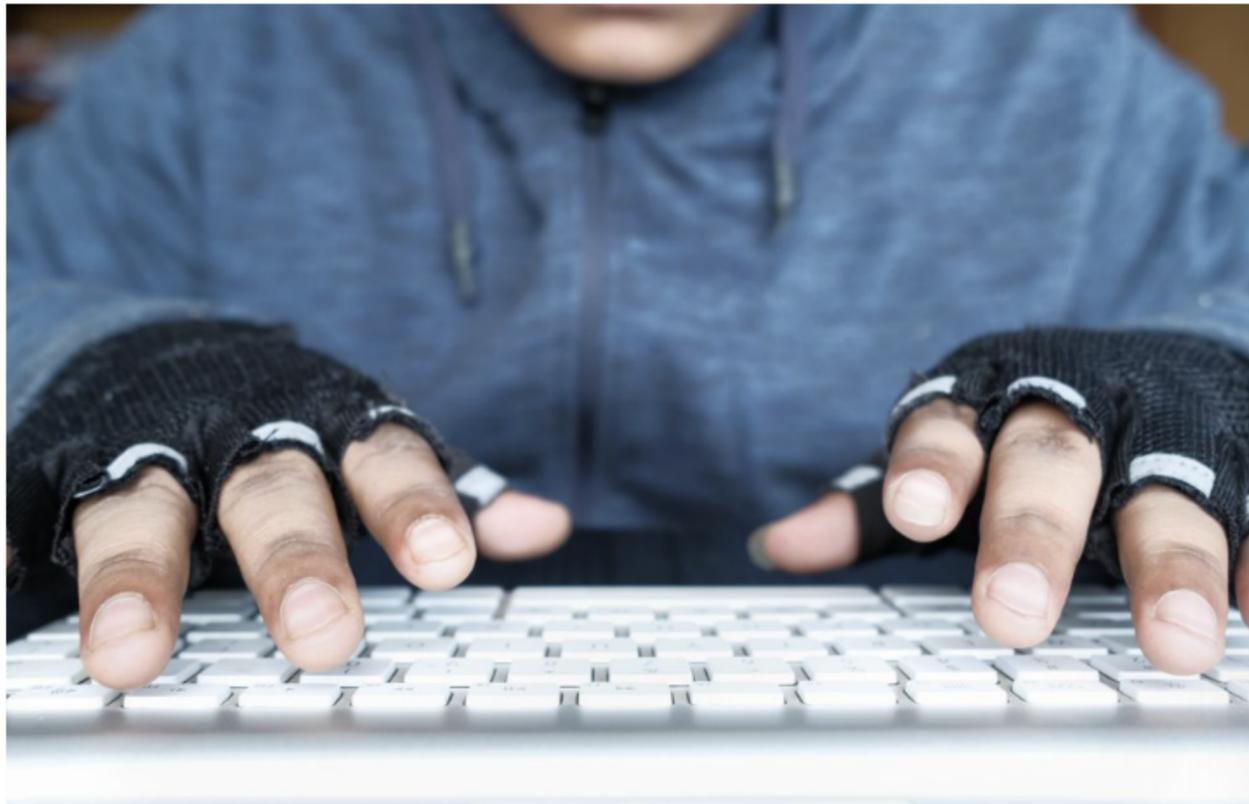
2018 MKC

Introduce yourself to a new Lincoln.

[CURRENT OFFERS](#)

[BUILD & PRICE](#)

Roll over for offer disclaimer




Life's Good

MEGA DEALS ON LG MEGA



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

NEWS

BBC REEL

THE STRANGE DOLLS THAT COME TO LIFE

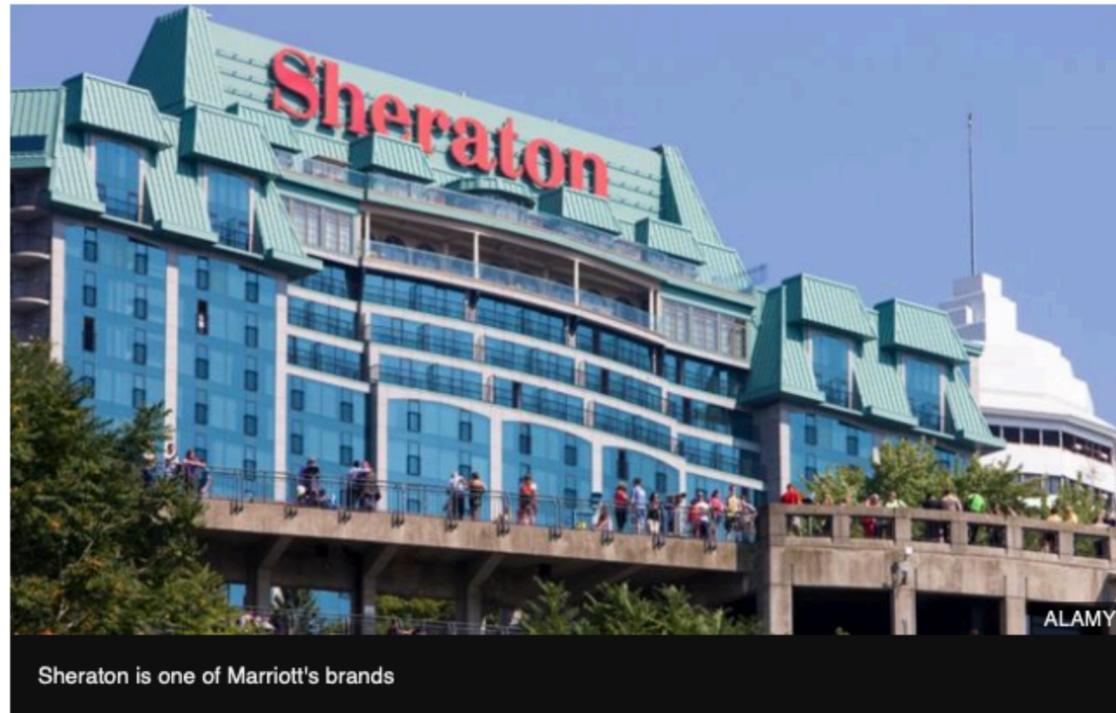


Technology

Marriott hack hits 500 million Starwood guests

30 November 2018

f WhatsApp Twitter Email Share



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

BBC REEL

DID BOND GIRL DIE AFTER BEING PAINTED GOLD?

WATCH NOW



Features

It cannot be a free for all.

You will need some guardrails.

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

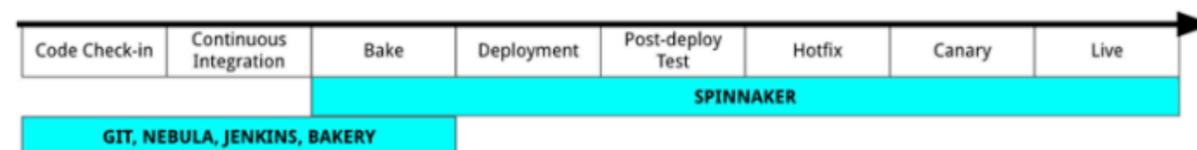
Pick from these 3 flavors. Won't
work for you? Let's talk.

Focus on “paved roads.”



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.

We've come a long way in software.

We still have miles to go.

We have to be willing to
change how we do things.

Evolve. Adapt. Improve.

Good luck!

Thanks!



I'm a Software Architect, Now What?
with Nate Shutta



O'REILLY
O.ΒEΙΓΓΛ.

Presentation Patterns
with Neal Ford & Nate Schutta



O'REILLY
O.ΒEΙΓΓΛ.

Modeling for Software Architects
with Nate Shutta



O'REILLY
O.ΒEΙΓΓΛ.

Nathaniel T. Schutta
@ntschutta
ntschutta.io

Succeeding with a Microservices Architecture
Best practices for making the move to microservices

Topic: **System Administration**



NATHANIEL SCHUTTA



O'REILLY®

Compliments of
Pivotal.

Thinking Architecturally

Lead Technical Change Within
Your Engineering Team



Nathaniel Schutta

[https://tanzu.vmware.com/
content/ebooks/thinking-
architecturally](https://tanzu.vmware.com/content/ebooks/thinking-architecturally)

O'REILLY®



Compliments of
VMware Tanzu

Responsible Microservices

Where Microservices
Deliver Value

Nathaniel Schutta

REPORT

[https://tanzu.vmware.com/
content/ebooks/responsible-
microservices-ebook](https://tanzu.vmware.com/content/ebooks/responsible-microservices-ebook)

Between Chair and Keyboard



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

Nate Schutta
Software Architect
VMware
@ntschutta

Tanzu.TV Shows

LIVE EVERY TUESDAY AT 1PM PT

Tanzu Tuesdays

Live demos of modern application development technologies.

VIEW EPISODES

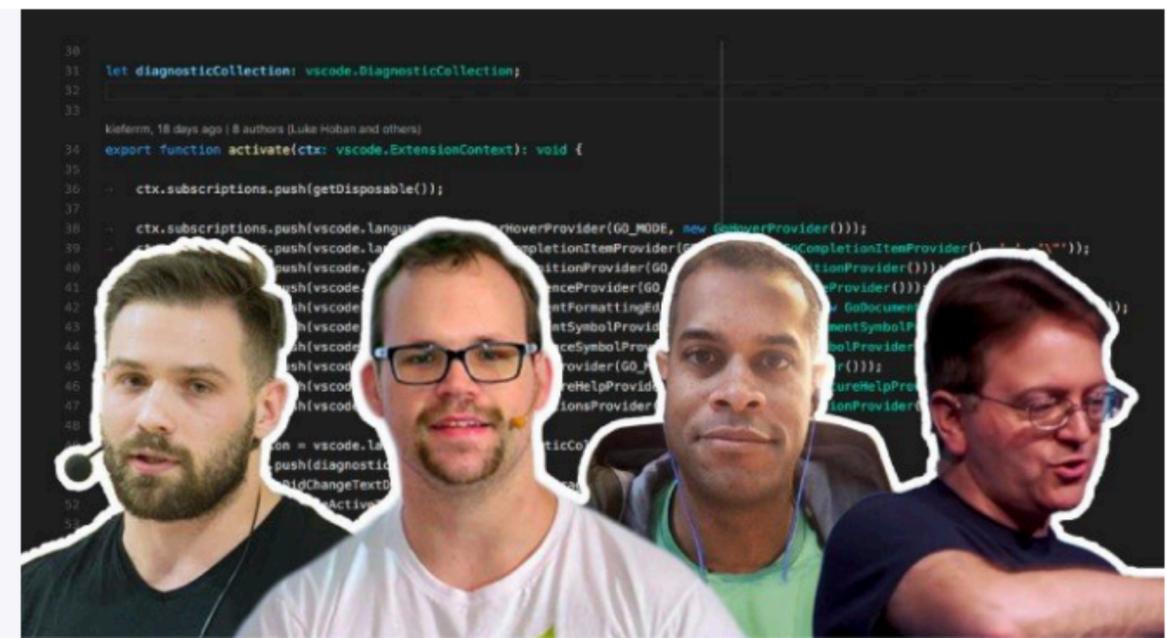


LIVE EVERY WEDNESDAY AT 12PM PT

Code

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

VIEW EPISODES



SpringOne Tour is back, in person, and coming to a city near you!

SpringOne TOUR



It's our vibrant, in-person, two-day event for the Spring developer community, and it's touring across the world. Join us for SpringOne Tour!

Come meet other Spring enthusiasts, experts, and advocates from your local community. Interact with VMware speakers and fellow attendees. Learn about new Spring developments, AppDev best practices, plus tools and techniques that are quickly becoming industry standard.

It's time to take your existing applications to the next level.

Cities

Chicago
APR 26-27

Toronto
JUN 7-8

New York
JUN 28-29