DevOps: Culture, Agile Development, & Cloud Native Technologies

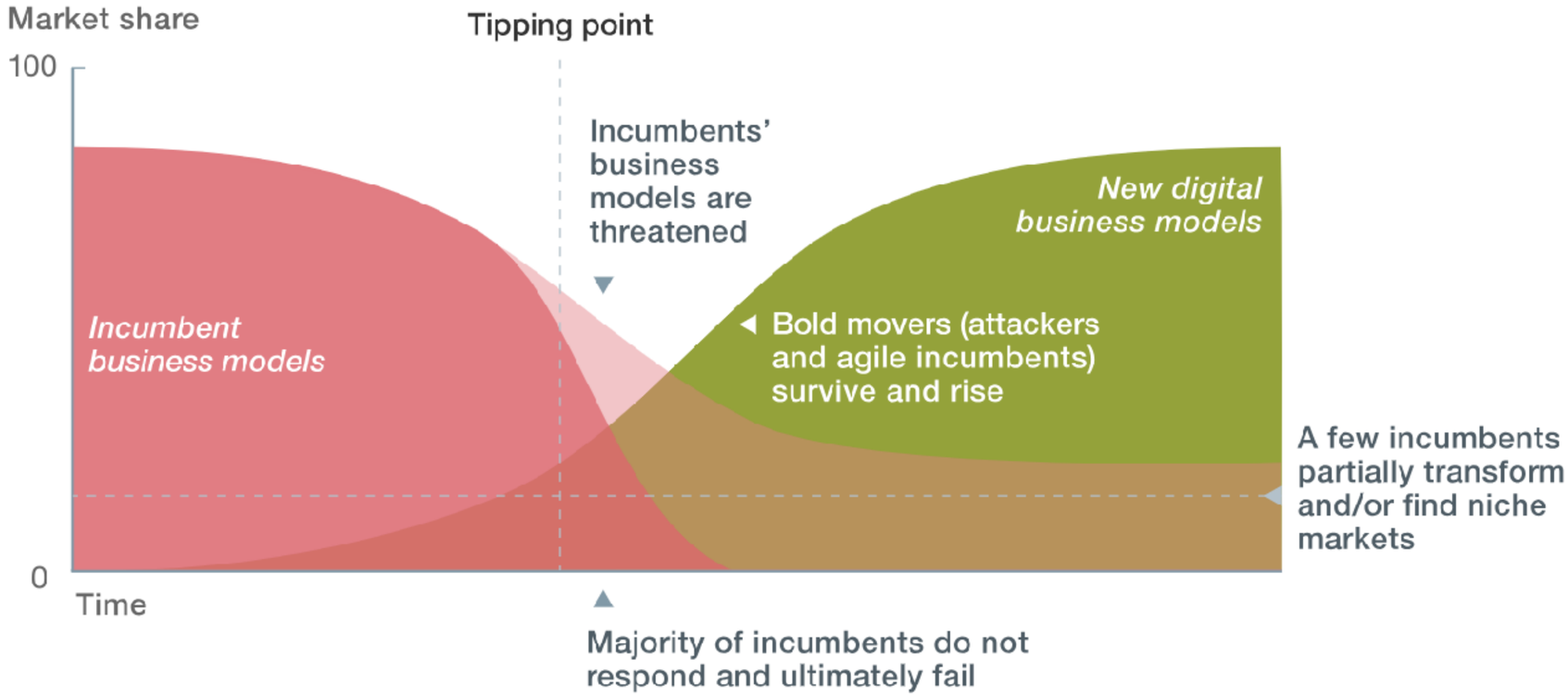
Instructor:

John J Rofrano

Senior Technical Staff Member, DevOps Champion
IBM T.J. Watson Research Center
rofrano@us.ibm.com (@JohnRofrano)

What are our clients facing?

What are our clients facing? – Disruptive Business Models



Example:
Call Taxi Service → Uber

8%

of companies believe their business model will remain economically viable through digitization

Source: McKinsey Digital Global Survey, 2016 and 2017; McKinsey analysis

Unlearn what you have learned

Often easier said than done

Consider this...

- **What if you could fail fast and roll back quickly?**
  - Then the impact of failure would be a minimal "blast radius"

Consider this...

- **What if you could fail fast and roll back quickly?**
  - Then the impact of failure would be a minimal "blast radius"
- **What if you could test in-market instead of analyzing?**
  - Then you could experiment with customers instead of second-guessing them

Consider this...

- **What if you could fail fast and roll back quickly?**
  - Then the impact of failure would be a minimal "blast radius"
- **What if you could test in-market instead of analyzing?**
  - Then you could experiment with customers instead of second-guessing them
- **What if your application design allowed individual components to be replaced?**
  - Then you wouldn't have "big bang" release weekends

Allspaw @ Velocity 2009

https://www.youtube.com/watch?v=LdOe18KhtT4

- Most people will cite John Allspaw (@allspaw) and Paul Hammond's Velocity 2009 presentation titled "**10+ Deploys Per Day: Dev and Ops Cooperation at Flickr**" as a pivotal moment in the Devops movement.

- "In the last week there were **67** deploys of **496** changes by **18** people" – Flickr Dev Blog, **December 17th 2008**.

https://conferences.oreilly.com/velocity/velocity2009/public/schedule/detail/7641

2011 Etsy Deploys to Production every ~25 min*

- In January 2011 (a month in which we did over **a billion** page views)
- Code committed by **76** unique individuals
- Was deployed to production by **63** different folks
- A total of **517** times

* Based on 22 days 10 hrs/day (517 / 22 / 10 = 2.35/hr)

“Our deployment environment requires a lot of trust, transparency, communication, coordination, and discipline across the team.”

–Chad Dickerson, CTO Etsy

...but these are "mythical" companies, right?

...big enterprisers can't possibly work like this, right?

2016 DevOps Enterprise Summit

- **Ticketmaster** - 98% reduction in MTTR
- **Nordstrom** - 20% shorter Lead Time
- **Target** - Full Stack Deploy 3 months to minutes
- **USAA** - Release from 28 days to 7 days
- **ING** - 500 application teams doing DevOps
- **CSG** - From 200 incidents per release to 18

ticketmaster

NORDSTROM

USAA®

TARGET

ING

How are they doing this?

They have embraced the DevOps culture

“The term (development and operations) is an extension of agile development environments that aims to enhance the process of software delivery as a whole.”

–Patrick Debois, 2009

What is DevOps?

DevOps is a recognition that Development and Operations needs to stop working alone in their "siloed" towers and start working together

- To do this we need:
  - A **culture** of collaboration valuing openness, trust, and transparency
  - An **application design** that does not require entire systems to be redeployed just to add a single function
  - **Automation** that accelerates and improves the consistency of application delivery so that we can develop and deliver software with speed and stability
  - A dynamic software-defined, **programmable platform** to continuously deploy onto

Things DevOps is Not

- DevOps is **not** simply combining Development & Operations teams
- DevOps is **not** a separate team
- DevOps is **not** a tool
- DevOps is **not** a one-size-fits-all strategy
- DevOps is **not** just automation

What's so hard about getting Dev and Ops to work together?

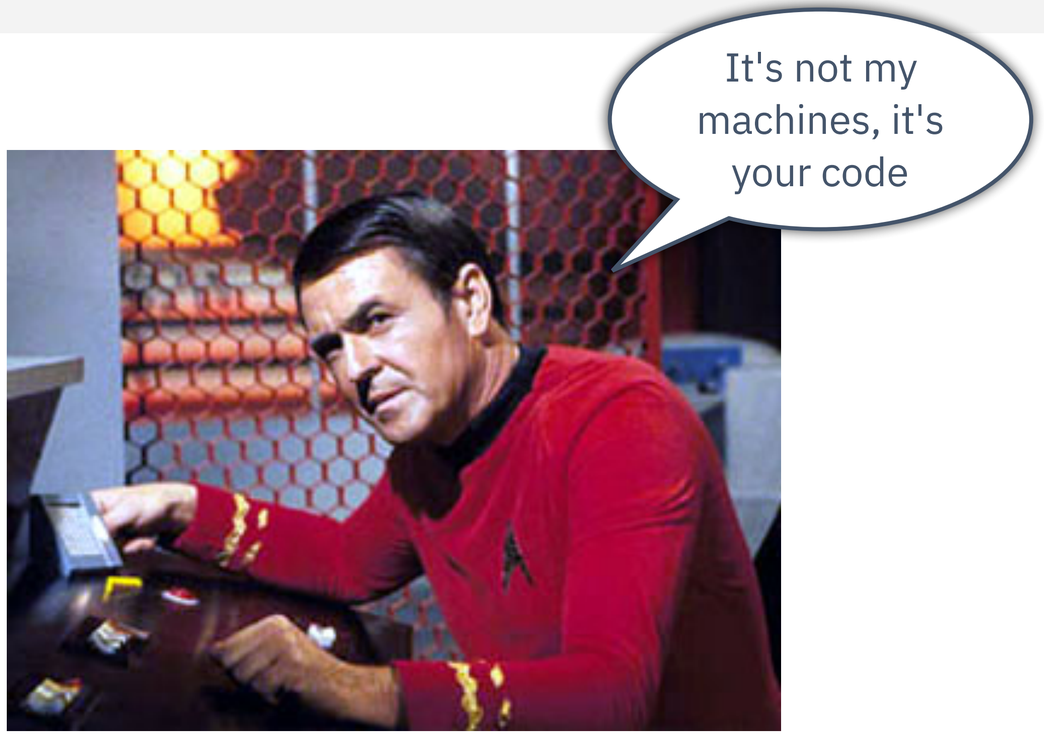
Dev verses Ops

Little bit weird

Sits closer to the boss

Thinks too hard

vs



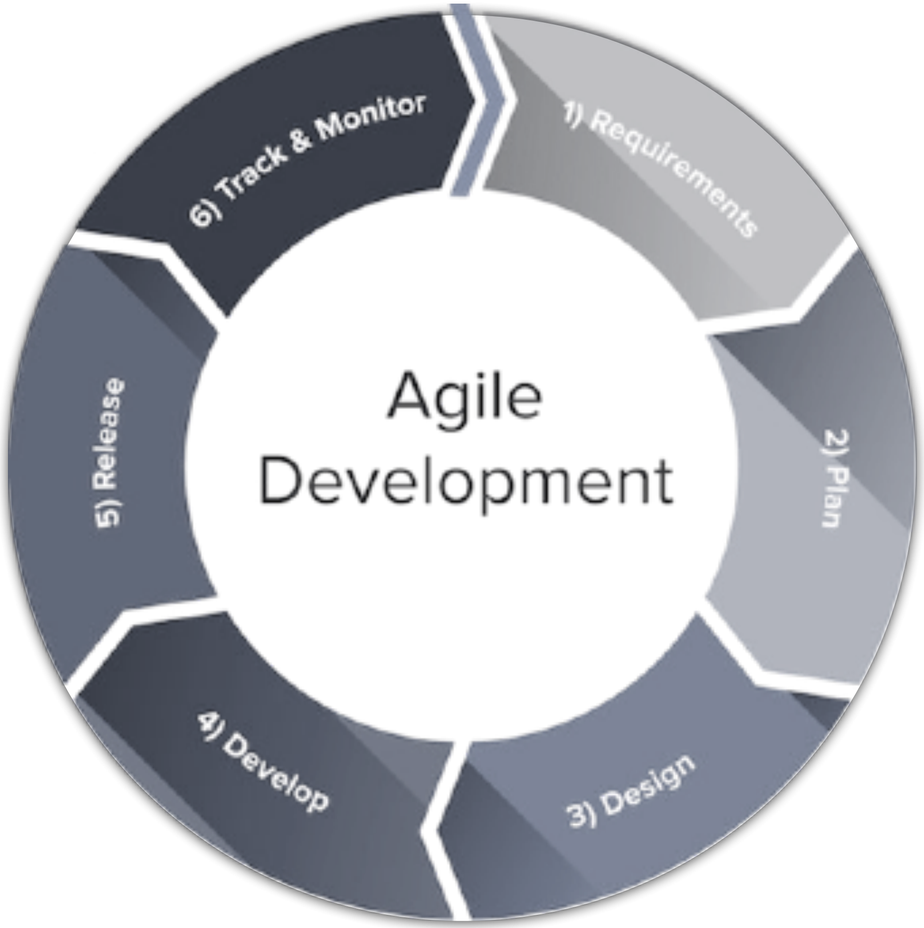
Pulls levers & turns knobs

Easily excited

Yells a lot in emergencies

...but we're already Agile, right?

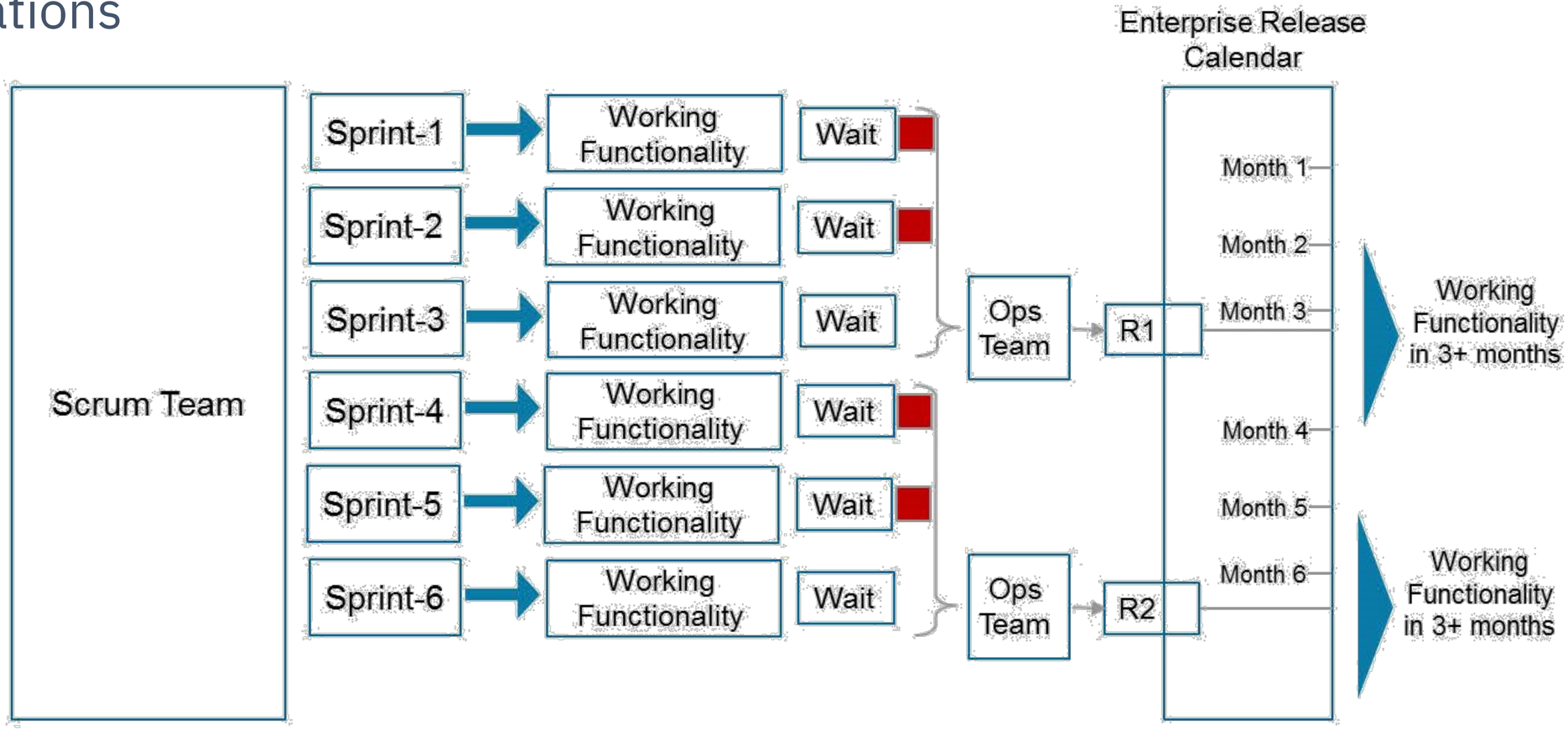
Agile Development



- Requirements and solutions evolve through the collaborative effort of **self-organizing** and **cross-functional** teams and their customers
- It advocates **adaptive planning**, evolutionary development, early delivery, and **continual improvement**
- It encourages rapid and flexible **response to change**

Why isn’t Agile alone good enough?

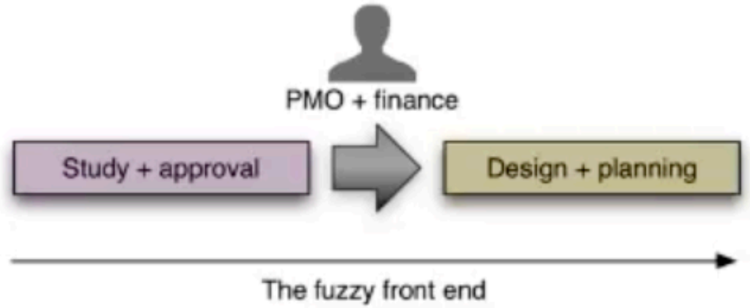
- While Agile improved the speed and accuracy of software for developers, it did nothing for operations



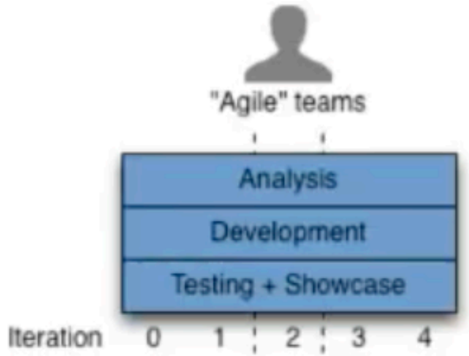
https://www.scrumalliance.org/community/articles/2014/april/devops-and-agile

Water-Scrum-Fall

This is an organization that wants to change without actually changing anything!



water

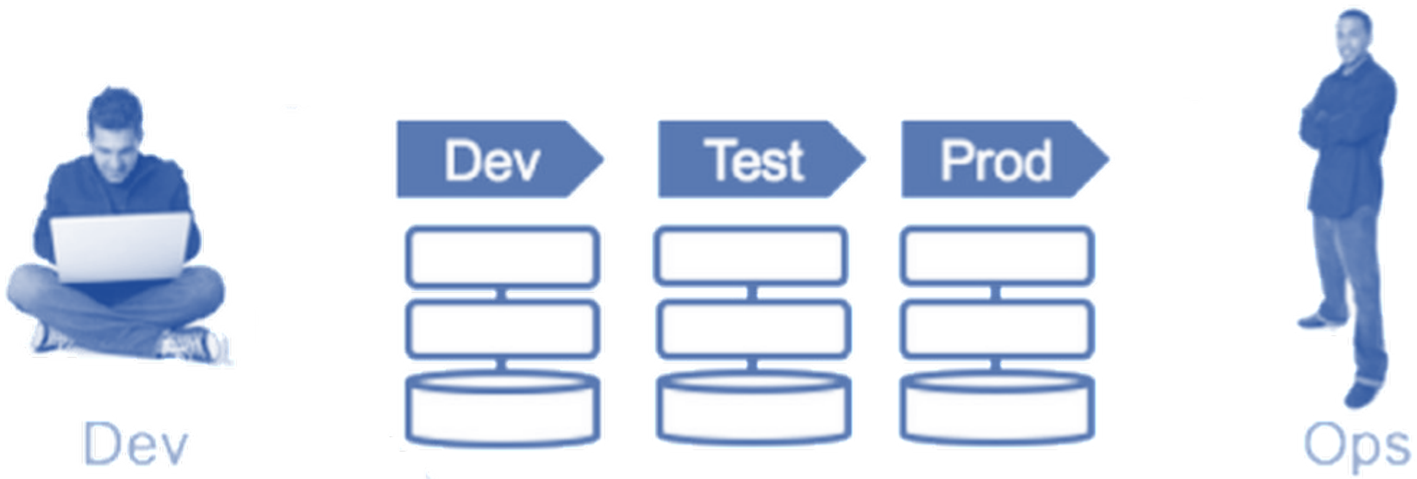


scrum

fall

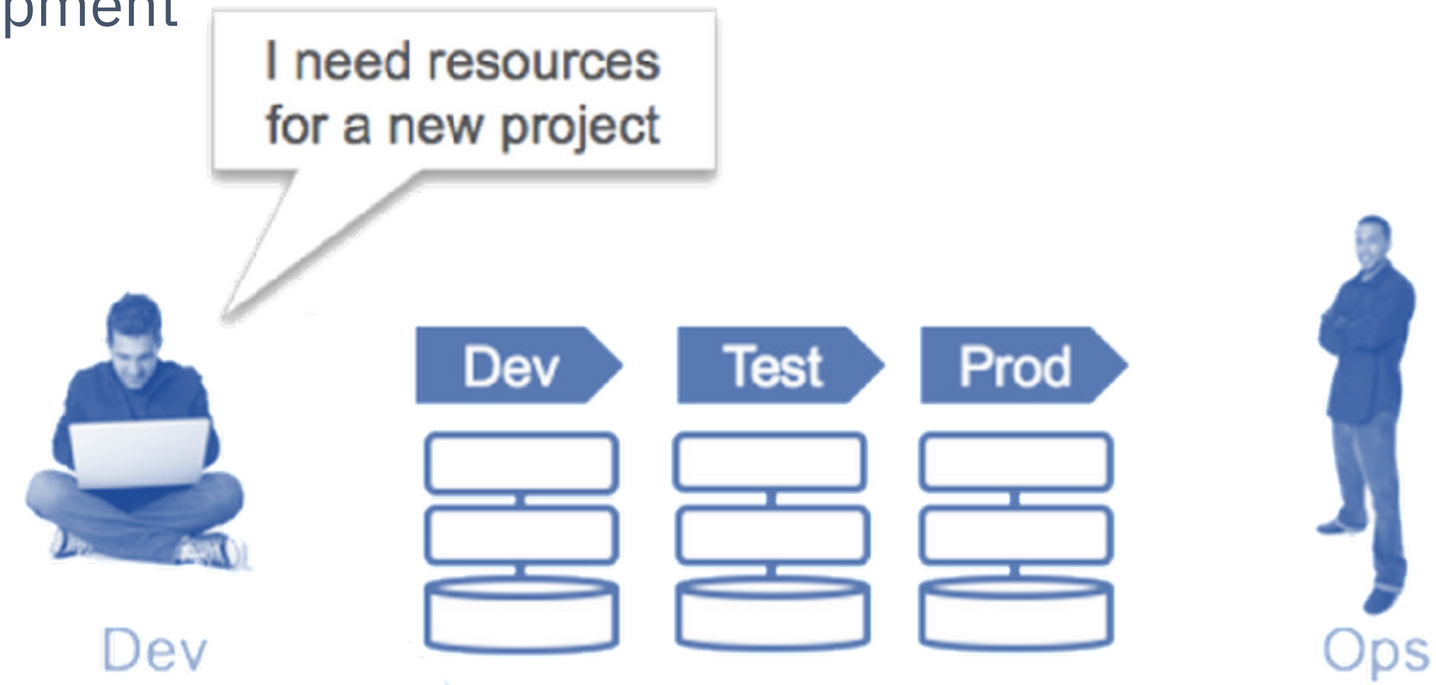
This is how "Shadow IT" is born

- Many development teams just got frustrated by ops not being able to deliver at the speed of development



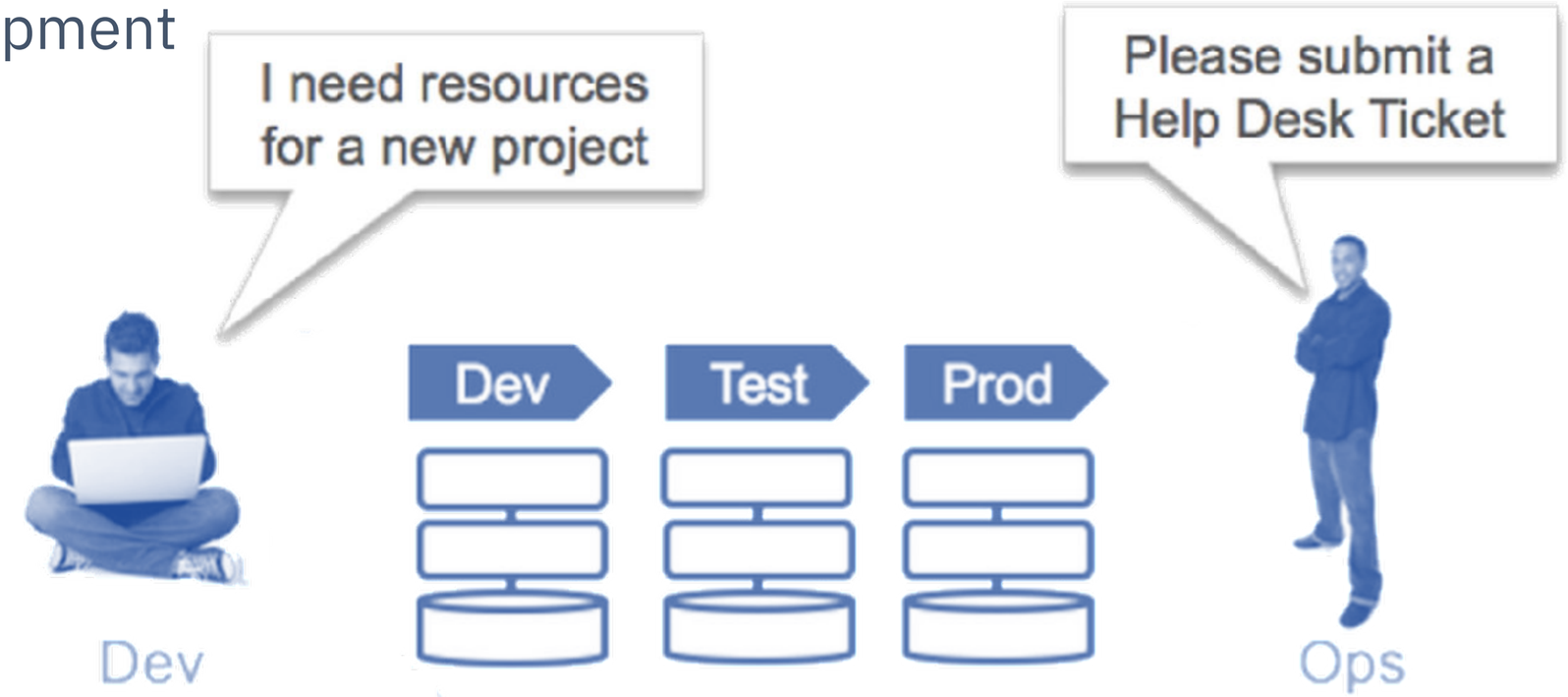
This is how "Shadow IT" is born

- Many development teams just got frustrated by ops not being able to deliver at the speed of development



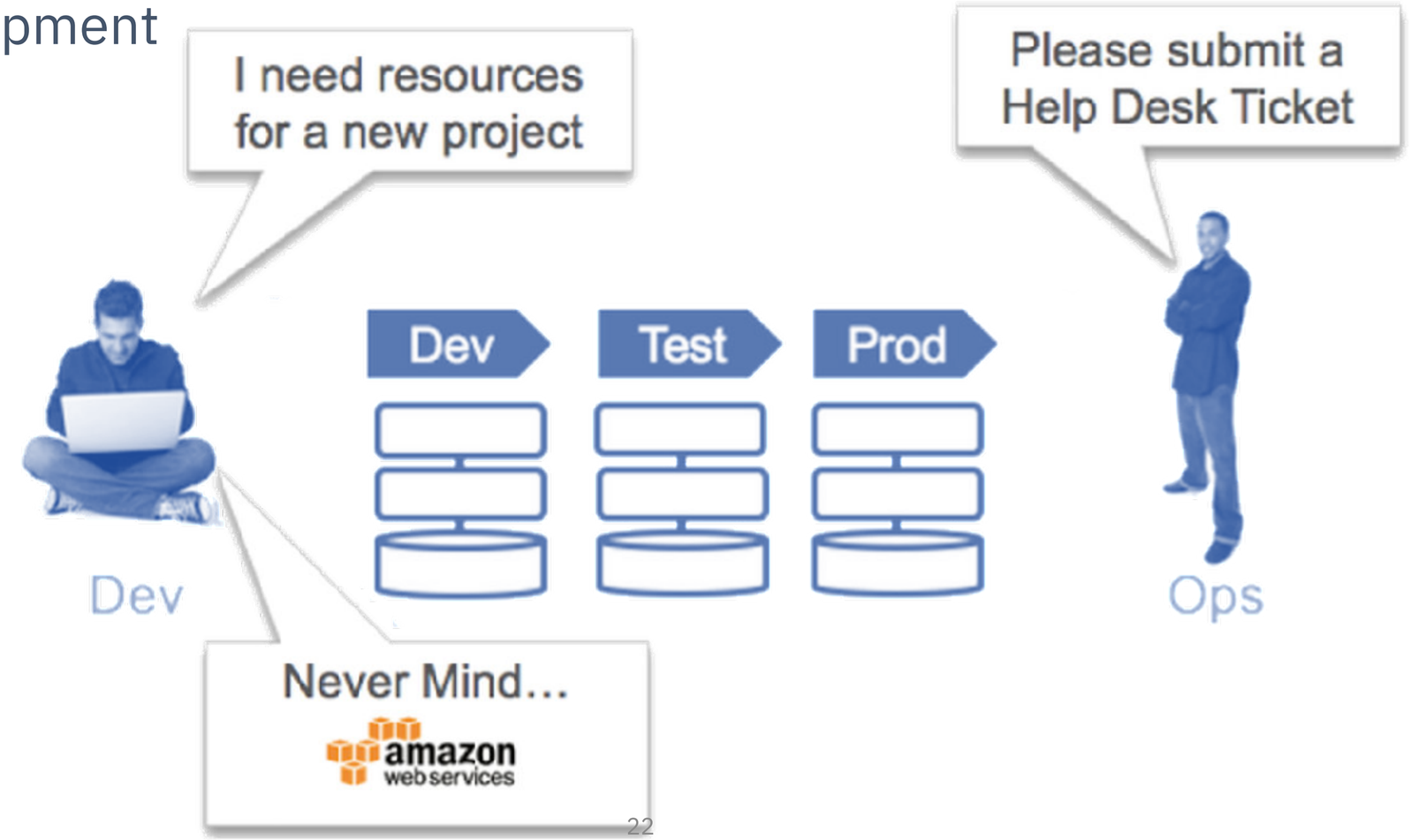
This is how "Shadow IT" is born

- Many development teams just got frustrated by ops not being able to deliver at the speed of development



This is how "Shadow IT" is born

- Many development teams just got frustrated by ops not being able to deliver at the speed of development



“DevOps starts with learning how to work differently. It embraces cross-functional teams with openness, transparency, and respect as pillars.”

–Tony Stafford, Shadow Soft

DevOps is a Grass Roots Movement

It is important that DevOps is...

- from the practitioners, by practitioners
- not a product, specification, job title
- an experience-based movement
- decentralized and open to all

Damon Edwards (https://www.youtube.com/watch?v=o7-IuYS0iSE)

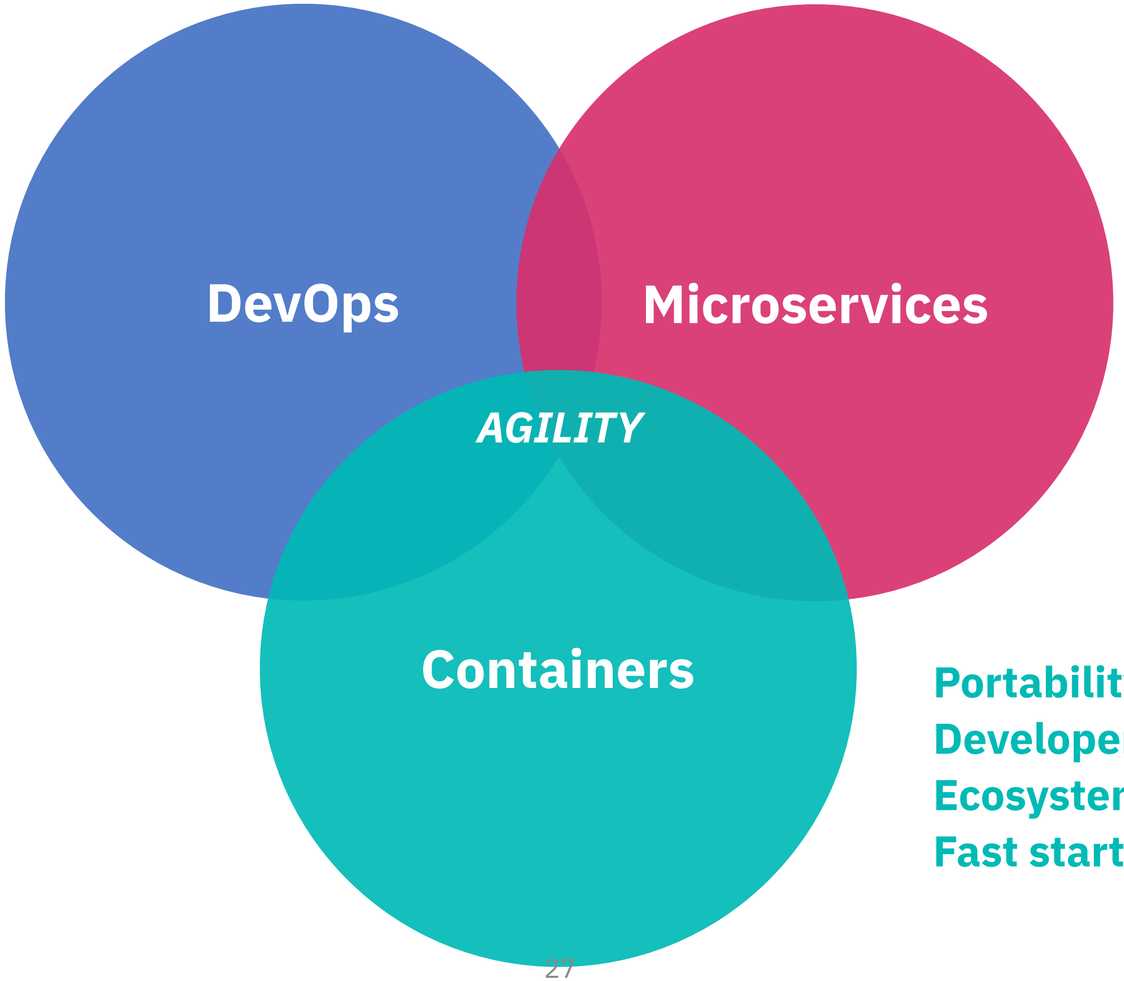
What is the Goal?

Agility is the goal

- Smart experimentation
- Moving in–market with maximum velocity and minimum risk
- Gaining quick valuable insight to continuously change the value proposition and quality

Agility: The Three Pillars

Cultural Change
Automated Pipeline
Everything as Code
Immutable Infrastructure



Loose Coupling/Binding
RESTful APIs
Designed to resist failures
Test by break / fail fast

Portability
Developer Centric
Ecosystem enabler
Fast startup

The Perfect Storm

- DevOps for speed and agility
- Microservices for small deployments
- Containers for ephemeral runtimes

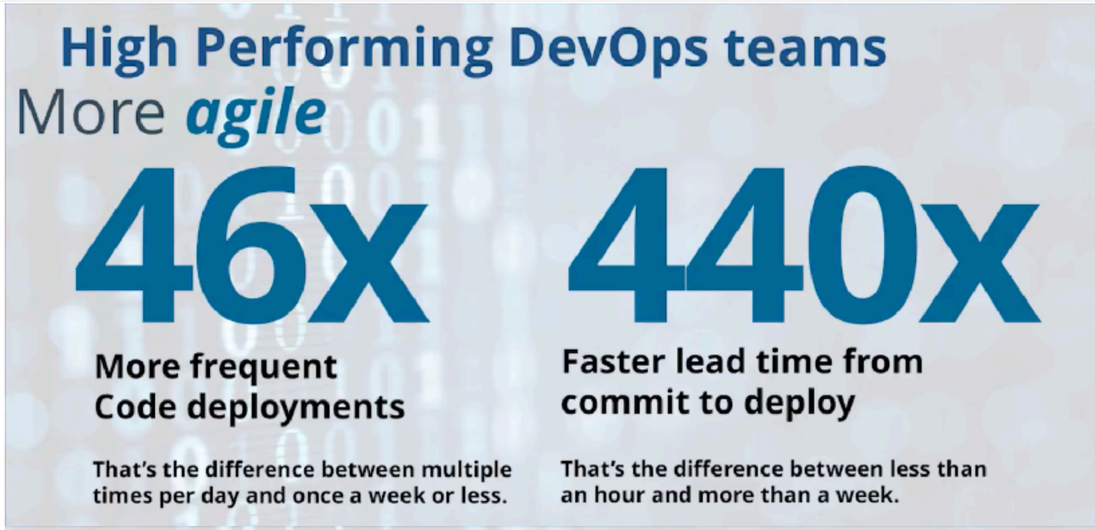
State of DevOps Report

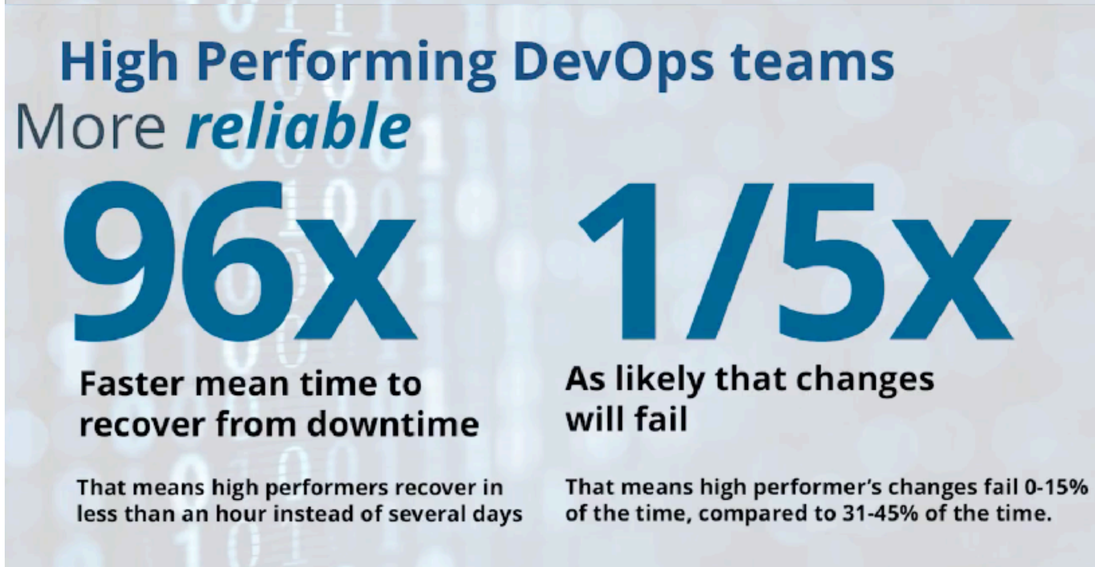
- Puppet Labs and DORA began measuring DevOps success in 2015 lead by Dr. Nicole Forsgren
- We have seen steady increase in companies adopting DevOps since then

Current DevOps Impact

State of DevOps Report by Puppet Labs 2017

- Taking an experimental approach to product development can improve your IT and organizational performance
- High-performing organizations are decisively outperforming their lower-performing peers in terms of throughput
- Undertaking a technology transformation initiative can produce sizeable cost savings for any organization

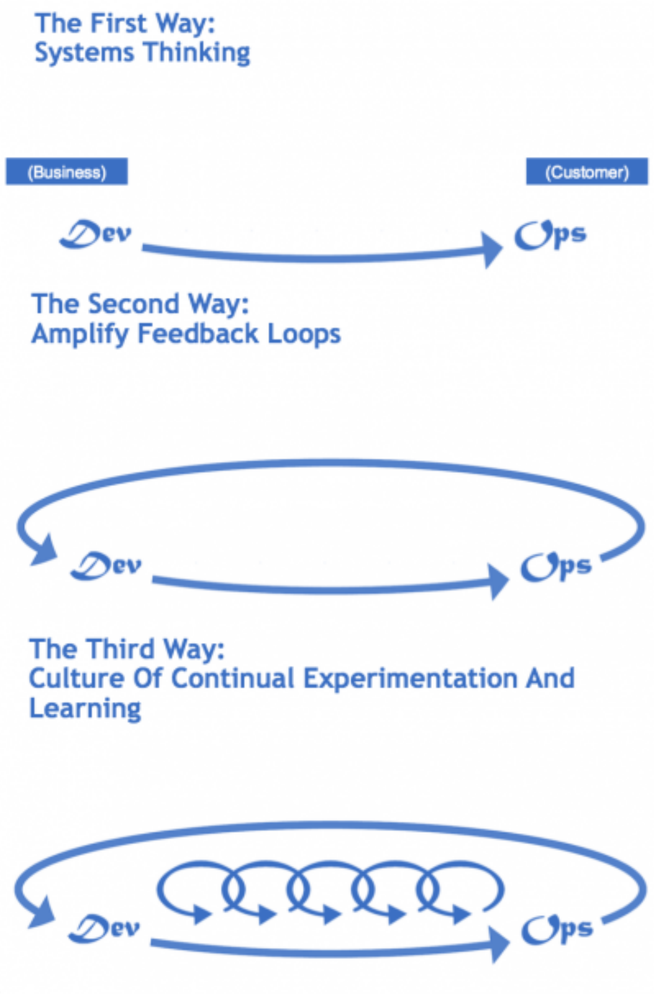




https://puppet.com/resources/whitepaper/state-of-devops-report

The Three Ways

The Phoenix Project - Gene Kim



The First Way helps us understand how to create fast flow of work as it moves from Development into IT Operations, because that's what's between the business and the customer.

The Second Way shows us how to shorten and amplify feedback loops, so we can fix quality at the source and avoid rework.

And **the Third Way** shows us how to create a culture that simultaneously fosters experimentation, learning from failure, and understanding that repetition and practice are the prerequisites to mastery.

“Culture is the #1 success factor in DevOps. Building a culture of shared responsibility, transparency and faster feedback is the foundation of every high performing DevOps team.”

–Atlassian

DevOps has Three Dimensions

1. Culture
2. Methods
3. Tools

“While tools and methods are important
… it’s the culture that has the biggest impact”

How do you change a culture?

How do you change a culture?

- You must change the way people **think**
- You must change the way people **work**
- You must change the way people are **organized**
- You must change the way people are **measured**

You must change the way people think

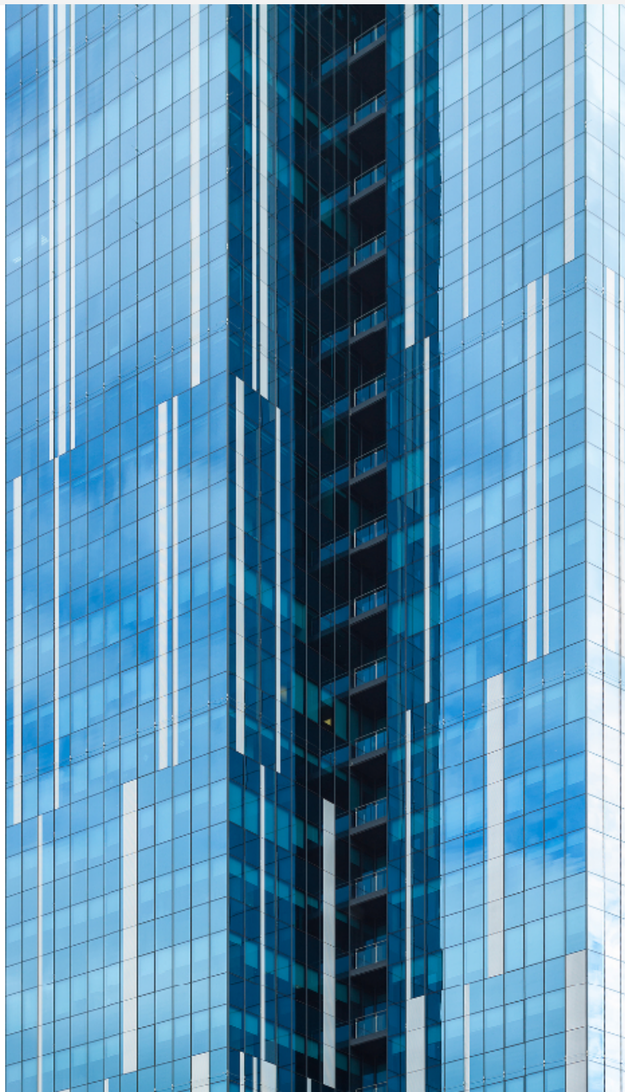
DevOps Thinking

- Social Coding
- Behavior and Test Driven Development
- Working in small batches*
- Build Minimum Viable Products for gaining insights*
- Failure leads to understanding

* from Lean Manufacturing

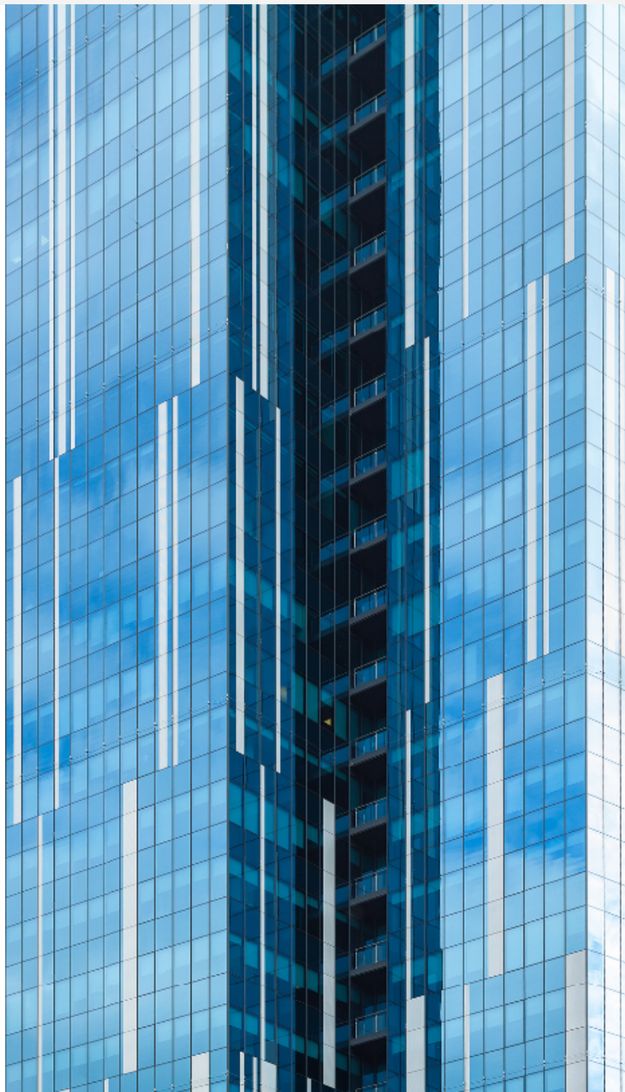
Traditional Thinking

- In the past developers worked on private repositories and you had to be a member of the team to contribute
- You see a project that is 80% of what you need but there are some missing features but...
- You feel that if you make a feature request of the project owner, your request will be rejected or go to the bottom of their priorities
- So you rebuild 100% of what you need so as not to have a dependency on another project



Think Social Coding

- Repositories are **public** and everyone is encouraged to Fork the code and contribute to the project
- **Discuss** the new feature with the repo owner and agree to develop it
- Open an **Issue** and assign it to yourself so that everyone knows what you are working on
- **Fork** the code, create a **branch**, and make your changes
- Issue a **Pull Request** when you are ready to review and merge your work back into the main project



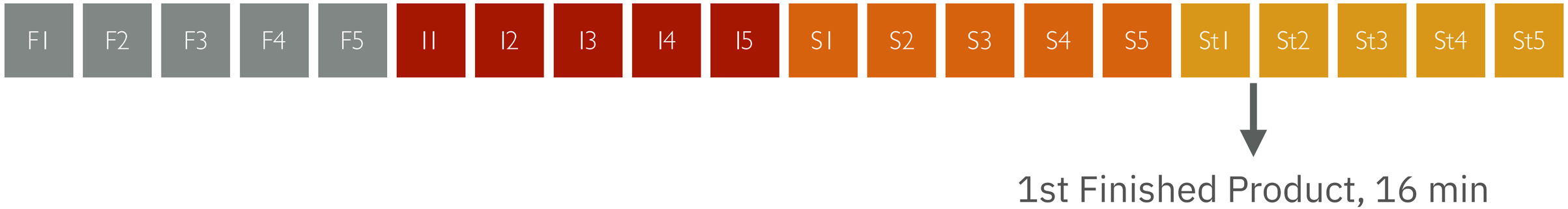
Think Working in Small Batches

- Imagine you need to mail 10 brochures to customers, by going through the following stages:
  - Step 1: Fold 10 brochures
  - Step 2: Insert them into envelopes
  - Step 3: Seal the envelopes
  - Step 4: Stamp the envelopes

Delivering 5 Brochures

Assume each step takes 1 minute to complete

Batch of 5 Brochures



Delivering 5 Brochures

Assume each step takes 1 minute to complete

Batch of 5 Brochures

1st Finished Product, 16 min

Single Piece Flow

1st Finished Product, 4 min

Delivering 5 Brochures

Assume each step takes 1 minute to complete

Batch of 5 Brochures

1st Finished Product, 16 min

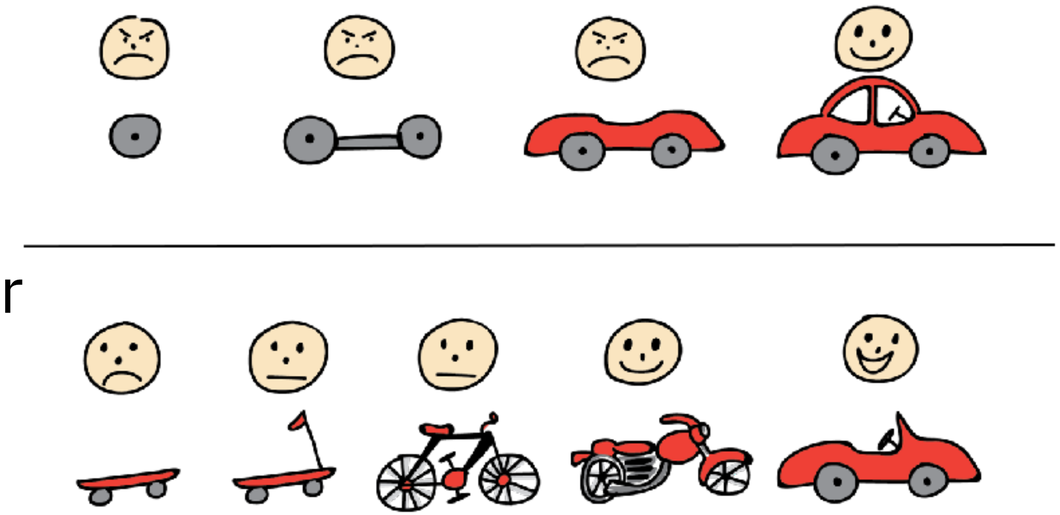
Single Piece Flow

1st Finished Product, 4 min

- What if the envelopes had no glue?
- What of there were a typo in the brochure?

Think Minimum Viable Product

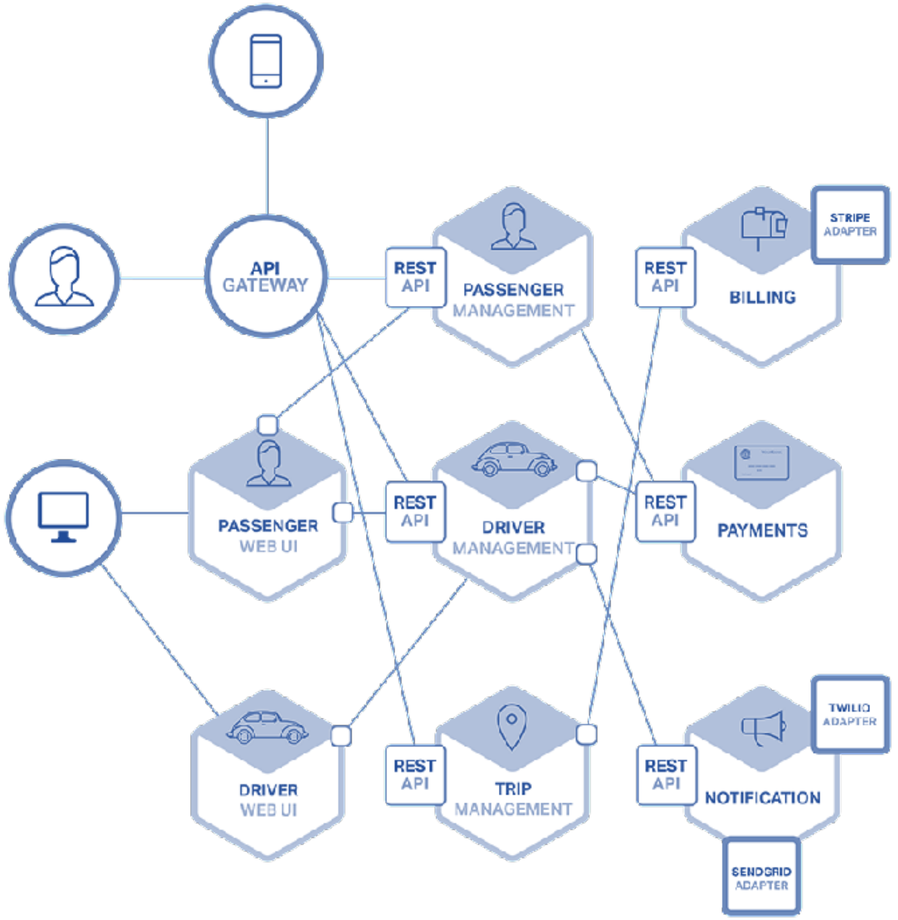
- MVP is NOT the result of "Phase 1" of a project
- It IS the cheapest/easiest thing you can build to start testing your **value hypothesis** and **learning**
- The former focuses on *delivery*, while the latter focuses on *learning*
- At the end of each MVP you decide whether to pivot or persevere



https://blog.crisp.se/2016/01/25/henrikkniberg/making-sense-of-mvp

Think Cloud Native

- **The Twelve-Factor App** describes patterns for cloud-native architectures which leverage microservices
- Applications are design as a collection of stateless microservices
- State is maintained in separate databases and persistent object stores
- Resilience and horizontal scaling is achieved through deploying multiple instances
- Failing instances are killed and re-spawned, not debugged and patched (cattle not pets)
- DevOps pipelines help manage continuous delivery of services



https://www.nginx.com/blog/introduction-to-microservices/

Think Microservices

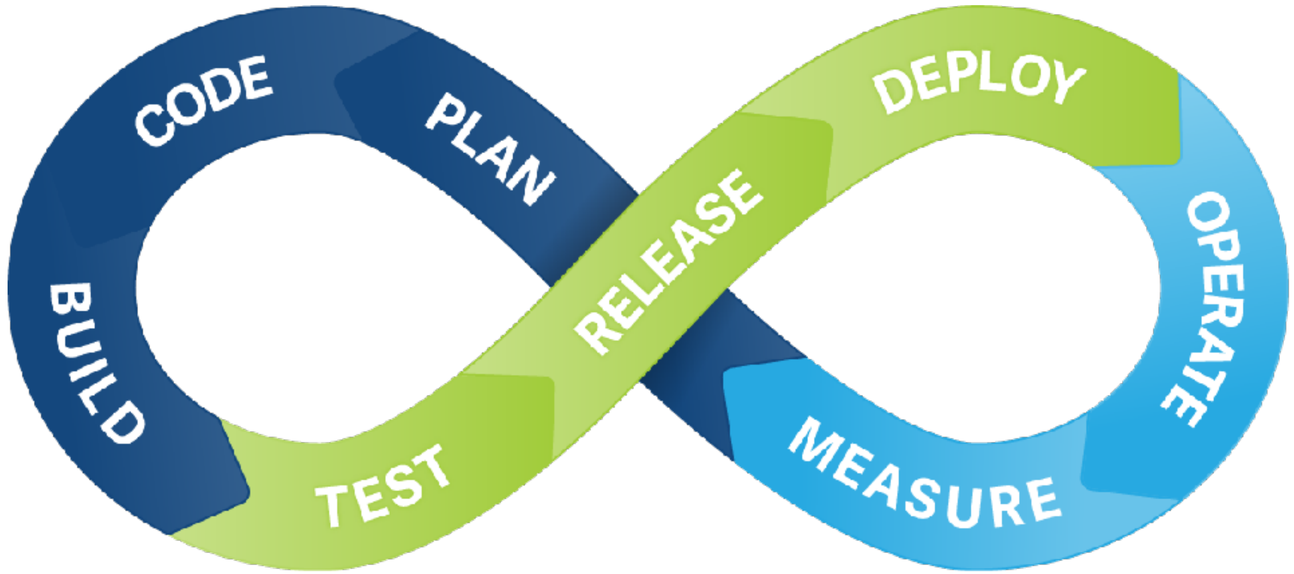
"...the microservice architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and independently deployable by fully automated deployment machinery."

- James Lewis and Martin Fowler

“Since failure is inevitable, architect your software to be resilient and to scale horizontally.”

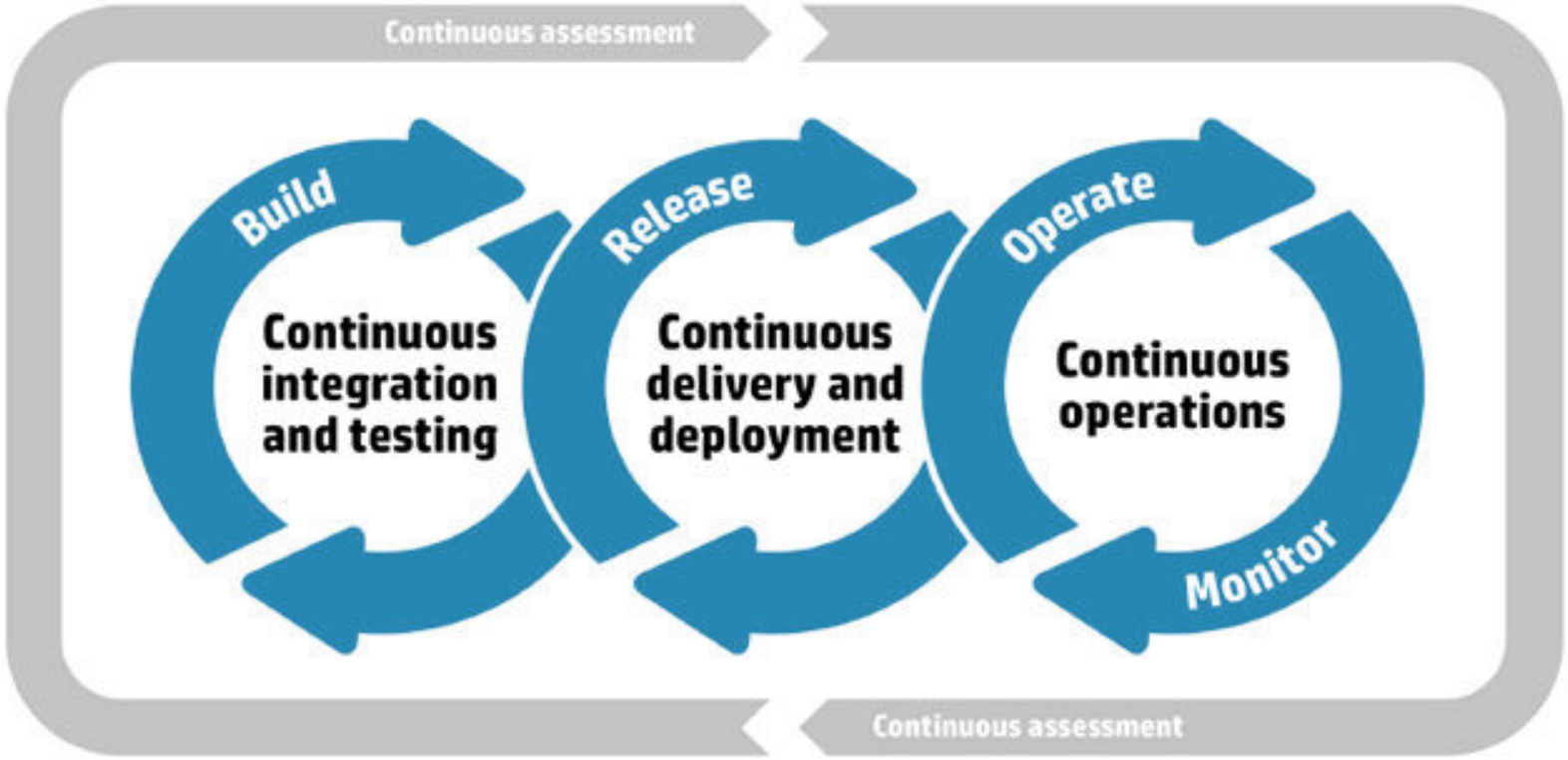
Think Continuous Automation

- Continuous Integration (CI)
- Continuous Delivery (CD)
- Build Automation
- Canary Rollouts / Blue-Green
- Failing Forward
- Application Release Automation



Automation is not just about speed... it's about repeatability.

Think Continuous Improvement



Actually it's a continuous pipeline that exists in a giant feedback loop

“Being able to recover quickly from failure is more important than having failures less often.”

–John Allspaw, CTO at Etsy

http://www.kitchensoap.com/2010/11/07/mttr-mtbf-for-most-types-of-f/

How do you change a culture?

You must change the way people work

Working DevOps

- Facilitate a culture of **teaming and collaboration**
- Establish **agile development** as a shared discipline
- **Automate relentlessly** to enable rapid DevOps response
- Push **smaller releases faster**, measure and remediate impact

DevOps

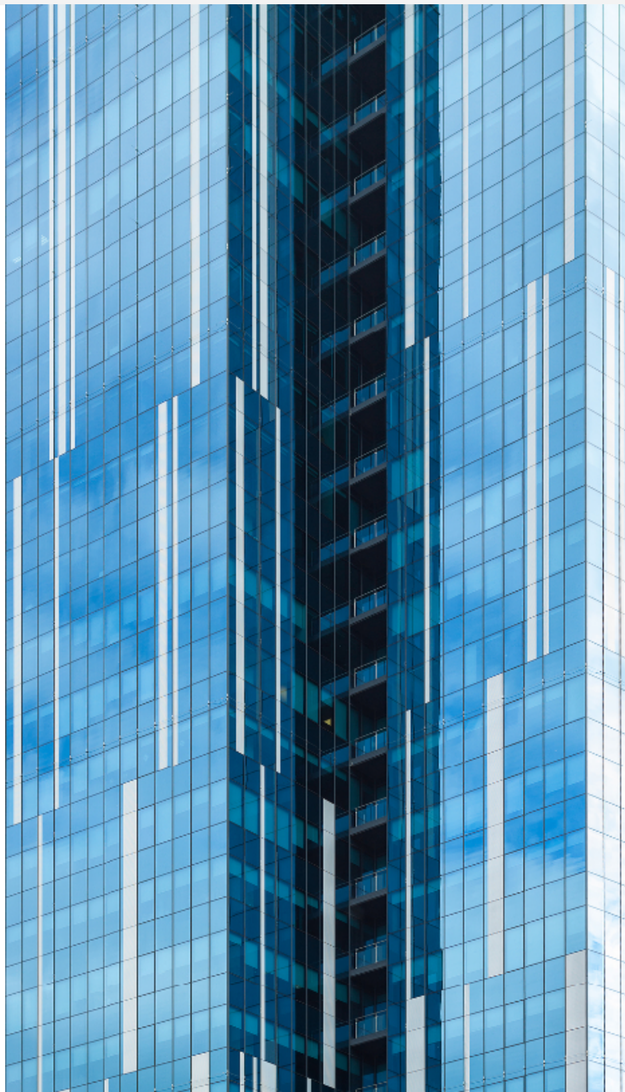
DevOps

We have worked the same way since the industrial revolution

Taylorism

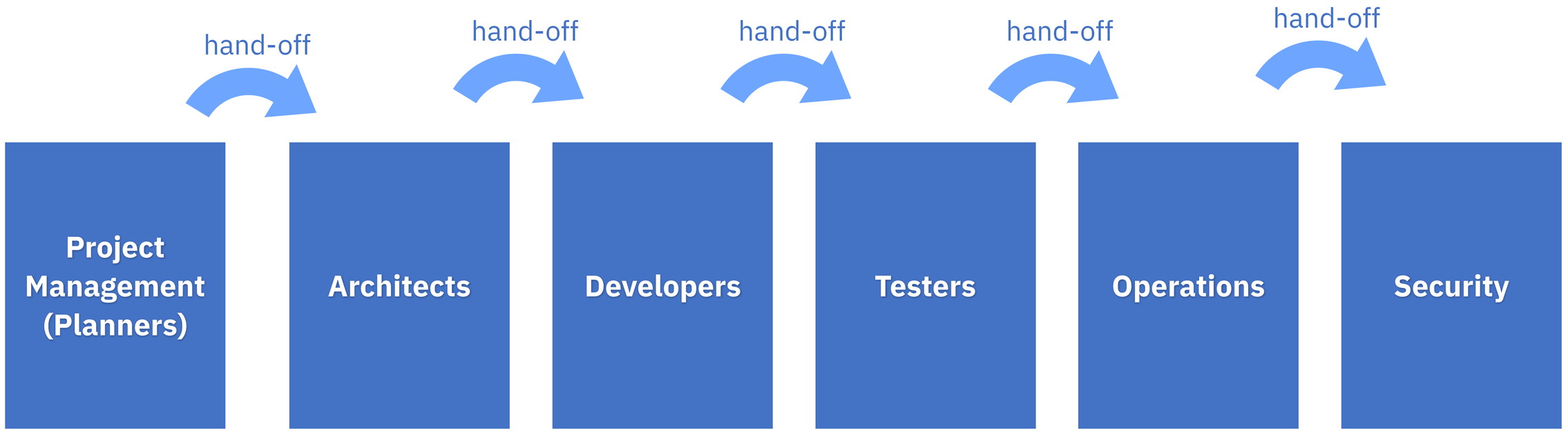
Named after the US industrial engineer **Frederick Winslow Taylor** (1856-1915) who in his 1911 book ***'Principles Of Scientific Management'*** laid down the fundamental principles of large-scale manufacturing through **assembly-line** factories.

- Adoption of **Command and Control Management**
  - The dominant method of management in the Western world
- Organizations divided into (ostensibly) independent **functional silos**
  - Workers are separated into task specific roles for greater efficiency
- **Decision-making** is separated from work
  - Managers do the planning and decide what workers should do
  - Workers mindlessly do the tasks they are ask to accomplish



Impact on Information Technology

Optimized roles may work great making cars (assembly line), not so good for software development (yields waterfall / silos)



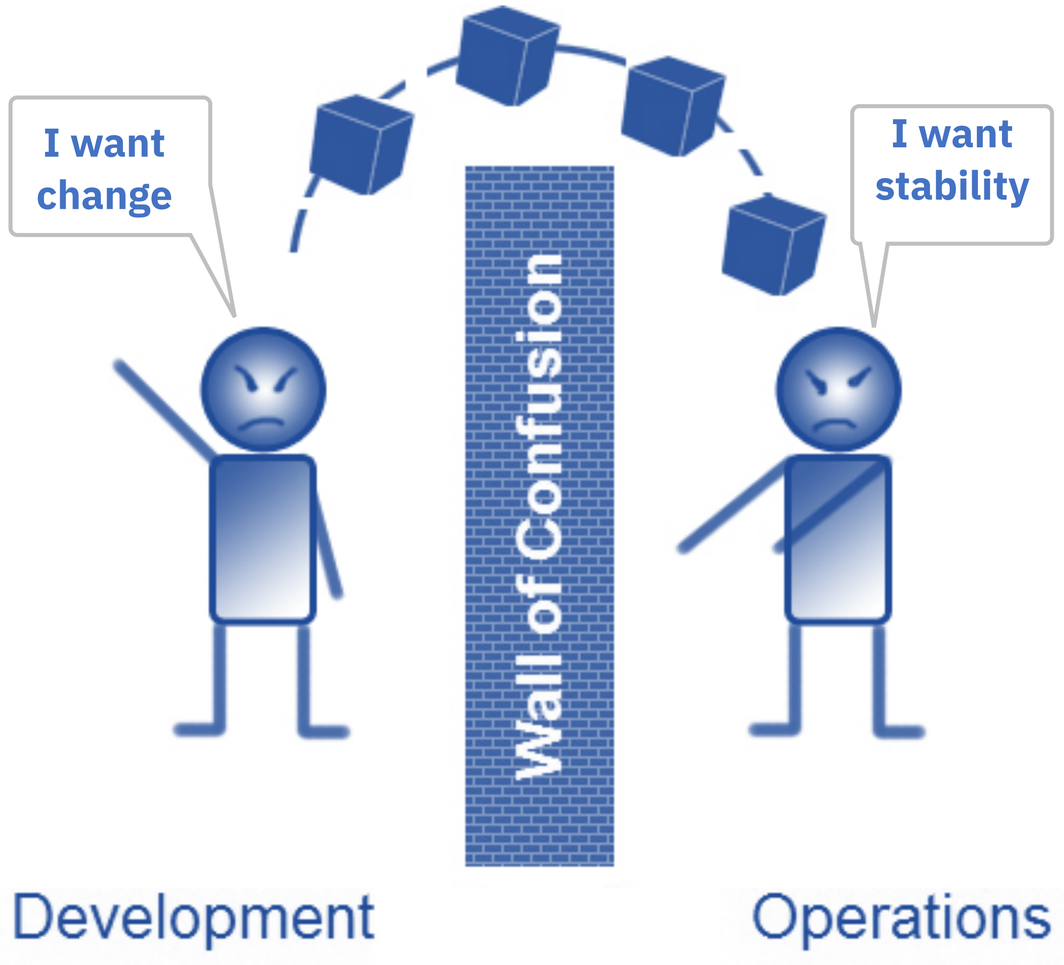
DevOps represents the reincarnation of "craft work"

...because software development is always bespoke

The Kobayashi Maru

...the no-win scenario

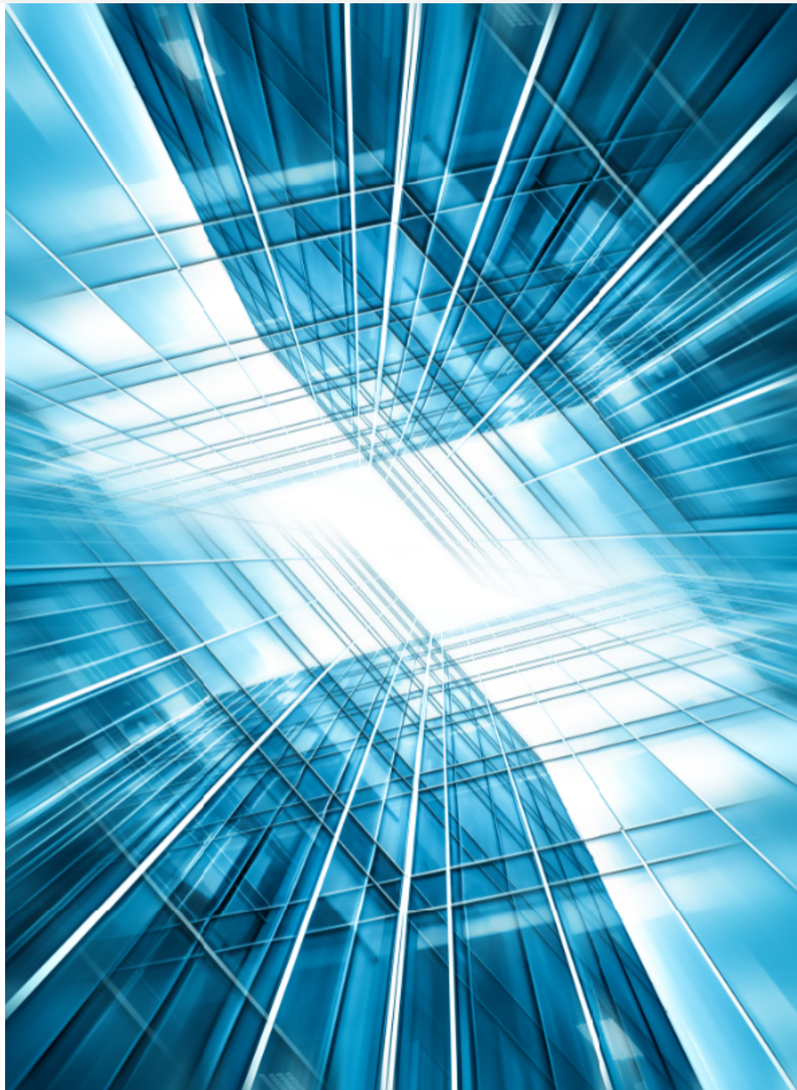
- **Development wants Innovation**
  - Keep up users' changing needs by developing and deploying new and enhanced capabilities.
- **Operations want Stability**
  - Ensure that users can use those services and applications and that their data and information is kept safe.



Source: Andrew Clay Shafer, Senior Director of Technology at Pivotal

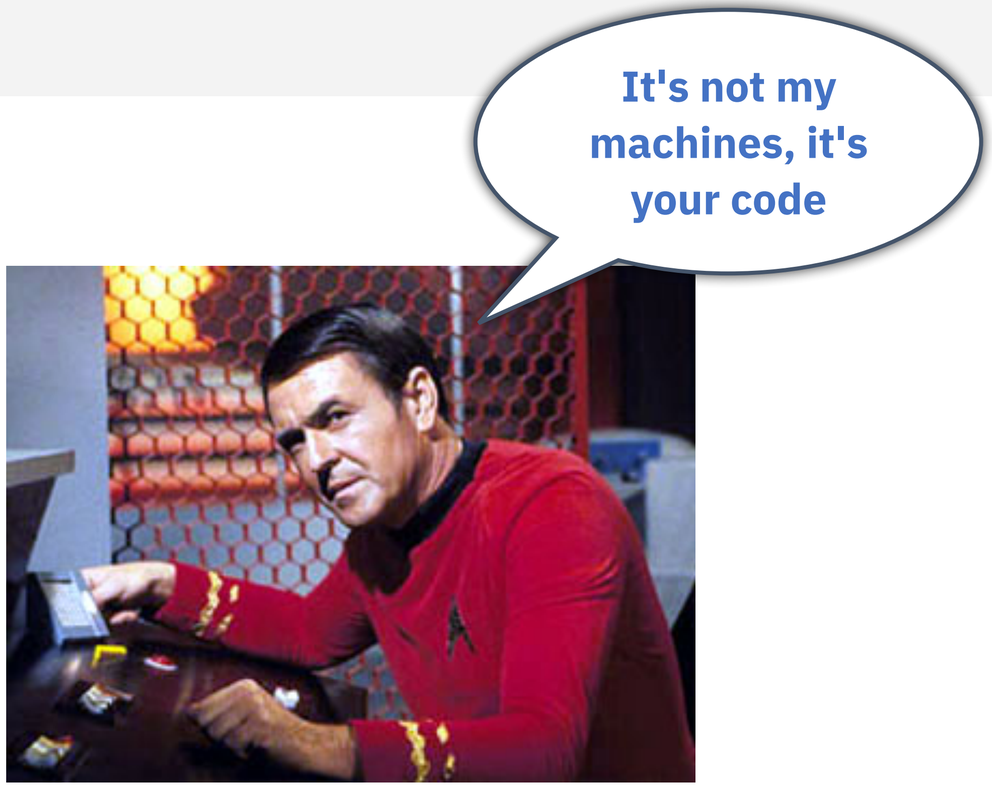
Diametrically Opposed Views

- **Enterprises** are built around end-to-end processes that see “new” as complex, high risk, expensive and time consuming
  - To be delivered as a one-time project
- **DevOps** seeks to turn this on its head
  - DevOps delivers a continual series of small changes
  - These cannot survive traditional overheads



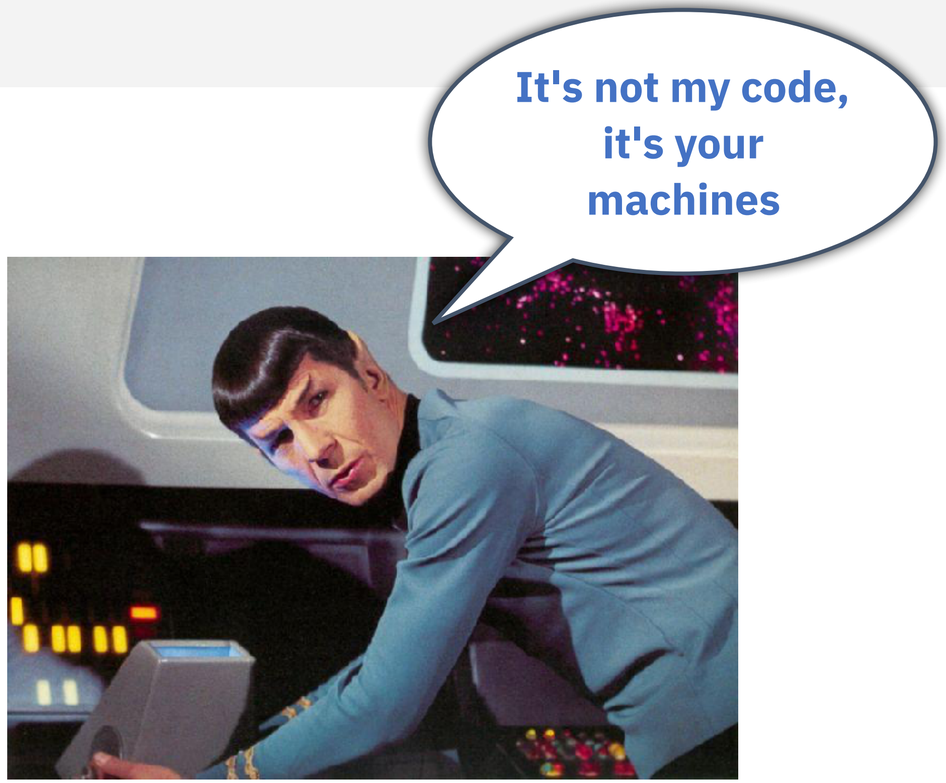
Operations View of Development

- Development teams throw dead cats over the wall
- Manually implemented changes
- Lack of back-out plans
- Lack of testing
- Environments that don’t look like Production



Development View of Operations

- All-or-nothing changes
- Change windows in the dead of night
- Implemented by people furthest away from the application
- Using cut-and-paste from Word documents called “run-books”



IT Silos Breed Apathy

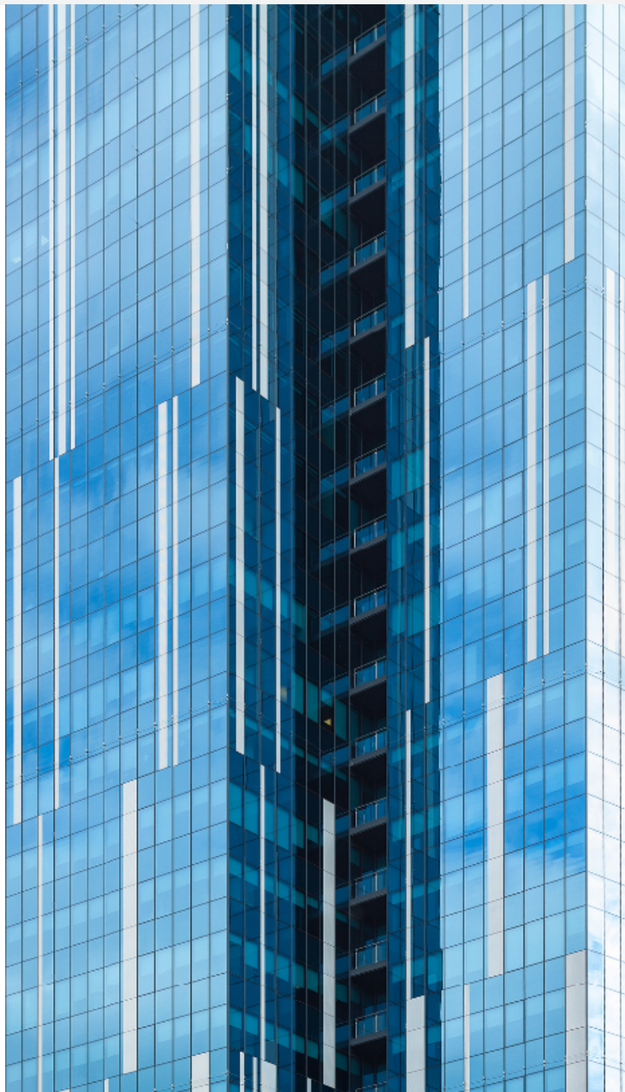
“If the web site works, the developers get the praise
If the web site is down, operations gets the blame!”

...Did I mention the Kobayashi Maru?

Required DevOps behaviors

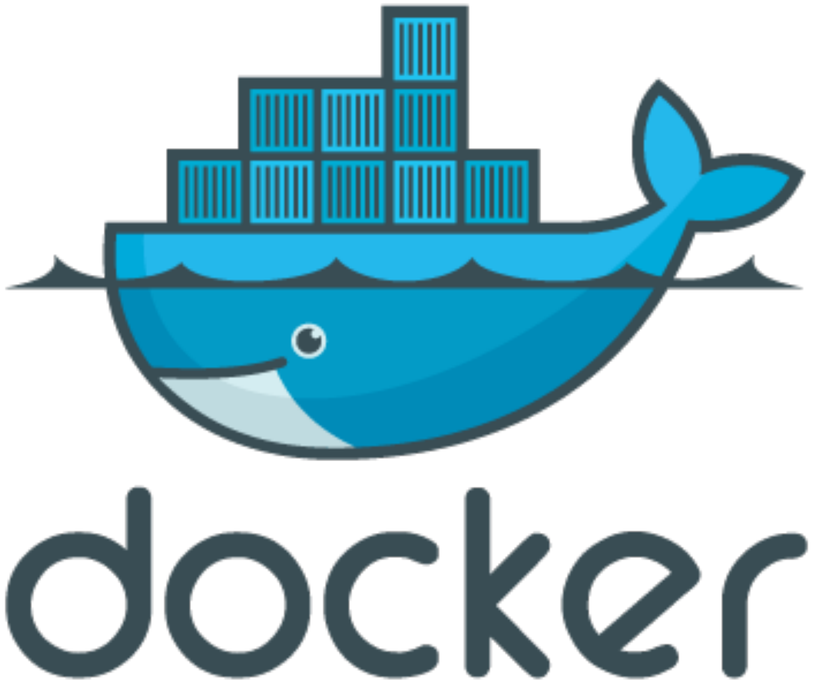
DevOps turns a number of traditional IT practices on their head

- **Shared ownership and high collaboration**
  - vs. organizational silos and hand-offs
- **Risk management by embracing change**
  - vs. fear of change
- **Ephemeral infrastructure**
  - vs. build once, hand crafted “snow flakes”
- **Automate Everything**
  - vs. manual fulfilment
- **Feedback loops and data driven responses**
  - vs. alarms, call-backs, and escalations

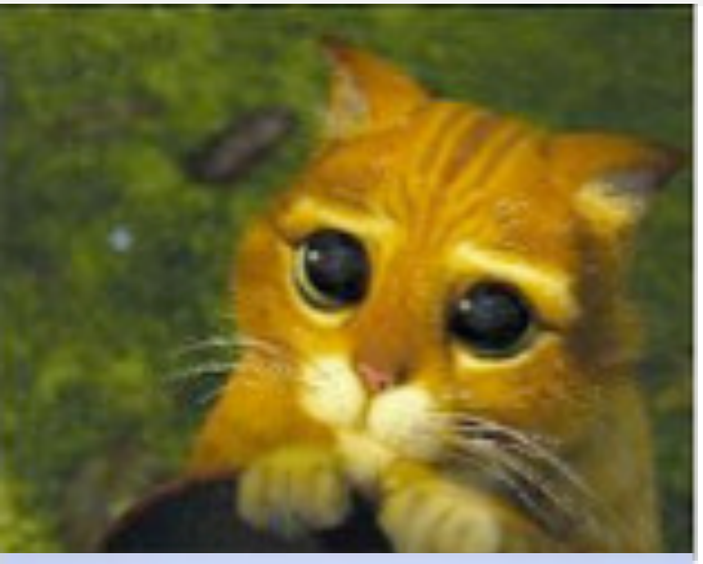


Immutable Delivery

- Applications are packaged in containers
- Same container that developer runs on their laptop runs in production
- Rolling updates with immediate roll-back
- No variance limits side-effects
- Dependencies are contained



Servers are Cattle not Pets



- Pets are given names like pussinboots.cern.sh
- They are unique, lovingly hand raised and cared for
- When they get ill, you nurse them back to health

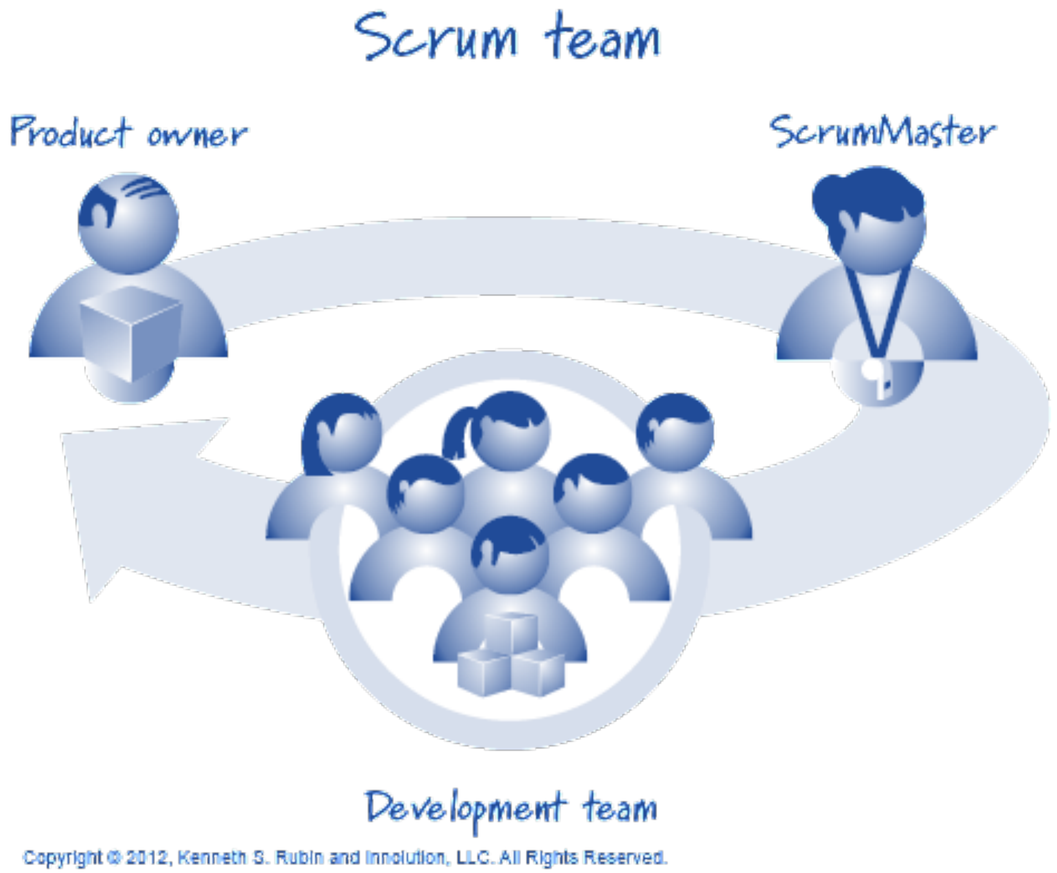
- Cattle are given numbers like vm0042.cern.ch
- They are almost identical to other cattle
- When they get ill, you get another one

Zero-Downtime Deployments

- Blue-green deployment is a zero-downtime deployment technique that consists of two nearly identical production environments, called Blue and Green.
- They differ by the artifacts that the developer has intentionally changed, typically by the version of the application. At any given time, at least one of the environments is active.
- Using the blue-green deployment technique, you can realize the following benefits:
  - Take software quickly from the final stage of testing to live production.
  - Deploy a new version of an application without disrupting traffic to the application.
  - Rollback rapidly. If there is something wrong with one of your environments, you can quickly switch to the other environment.

“DevOps is about breaking down the silos and working as a Single Agile Team.”

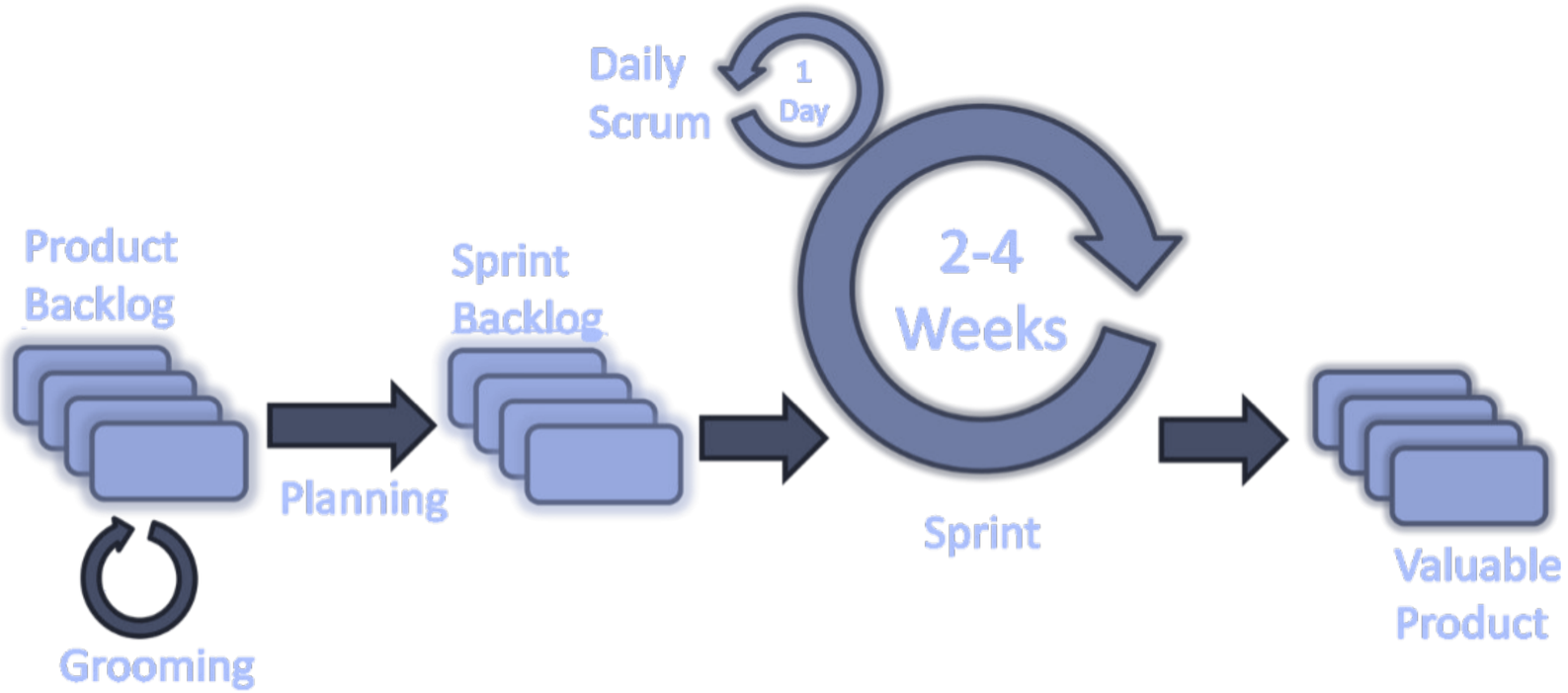
Organizing as a Scrum Team



- Small team (5 +/- 2)
- Dedicated
- Co-located
- Cross-functional
- Self managing

Working as an Agile Team

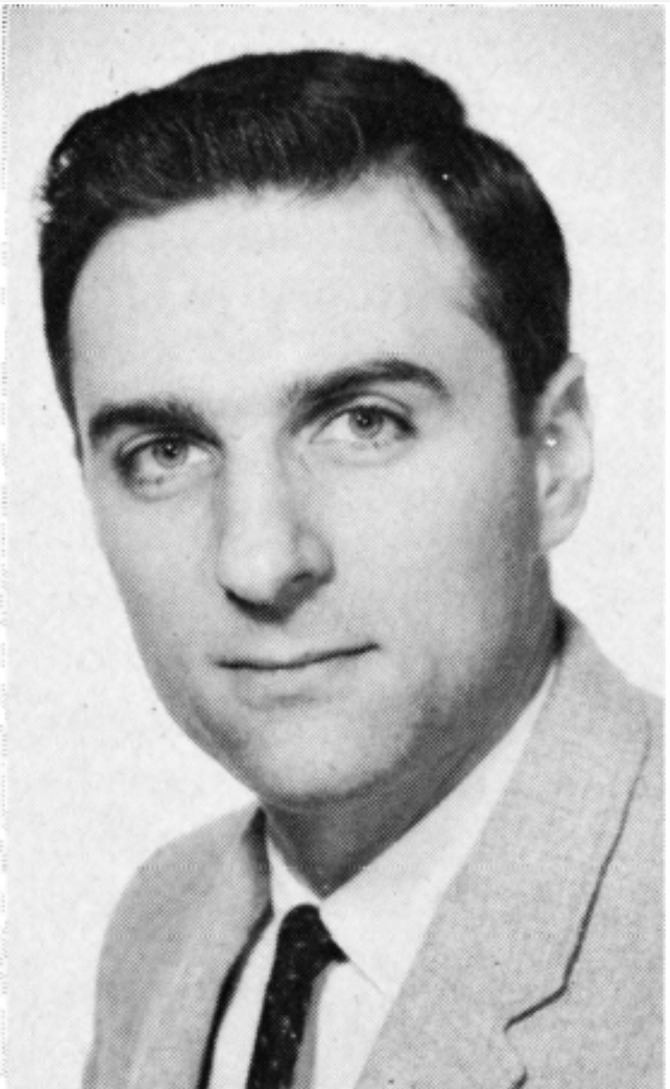
- Iterative Sprints
- Groomed Backlogs
- Customer Stories
- 2 Week Deliverables



How do you change a culture?

You must change the way people are organized

Conway’s Law



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, Datamation, 1968

e.g., if you ask an organization with 4 teams to write a compiler... you will get a 4-pass compiler!

http://www.melconway.com/Home/Conways_Law.html

Traditional Organization around Technology (Bad)

Organization Structure

User Interface Team

Application Team

Database (DBA) Team

Application Structure

Traditional Organization around Technology (Bad)

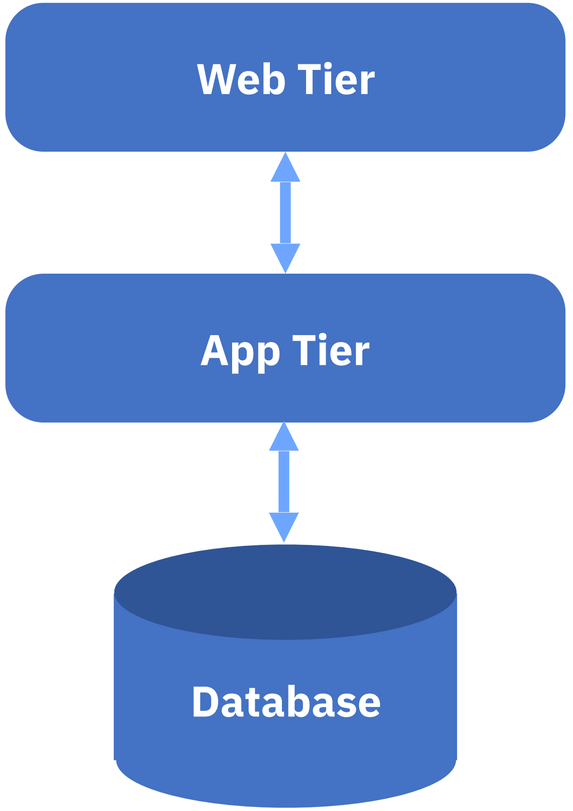
Organization Structure

User Interface Team

Application Team

Database (DBA) Team

Application Structure



Organization around Business Domains (Good)

Login
Registration
Users

Account Team

Personalization

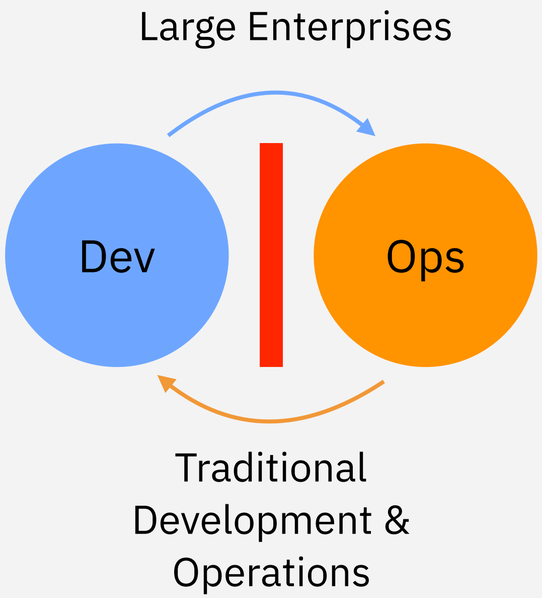
Personalization Team

Inventory
Shipping
Receiving

Warehouse Team

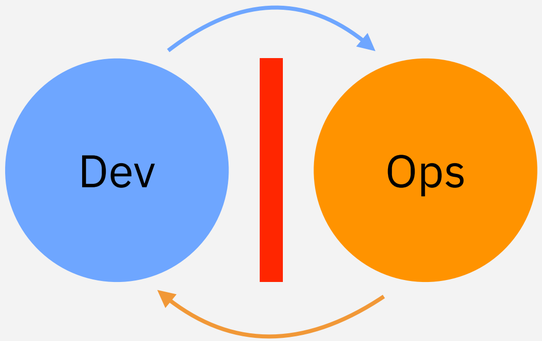
“If you are not going to organize around business domains, you are not going to get the full benefit of DevOps”

Different Perspectives of DevOps

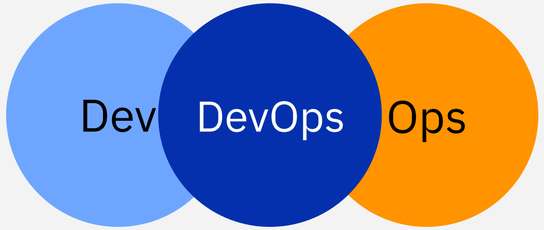


Different Perspectives of DevOps

Large Enterprises



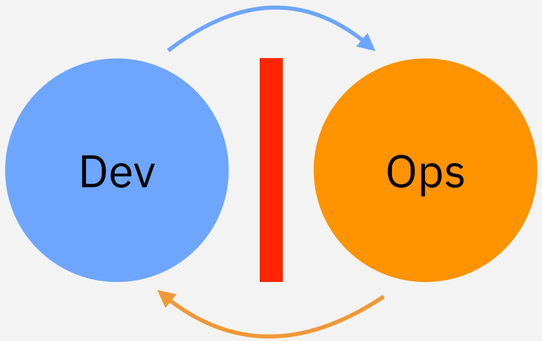
Traditional
Development &
Operations



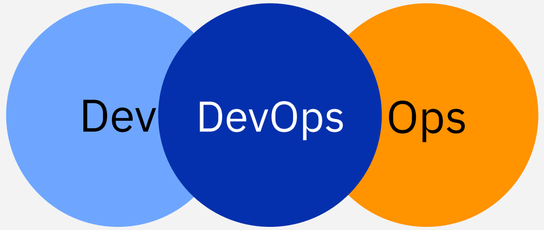
DevOps is a separate team

Different Perspectives of DevOps

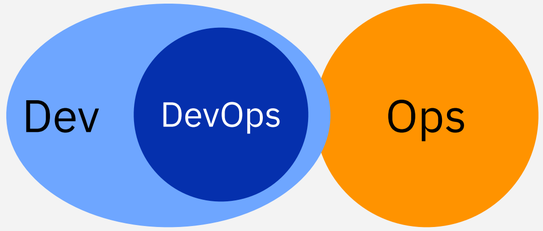
Large Enterprises



Traditional
Development &
Operations



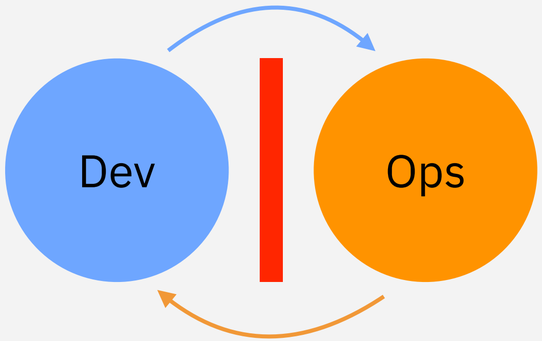
DevOps is a separate team



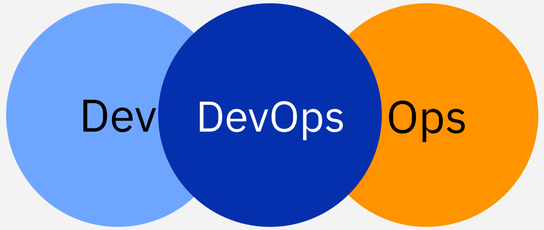
DevOps is something Devs do

Different Perspectives of DevOps

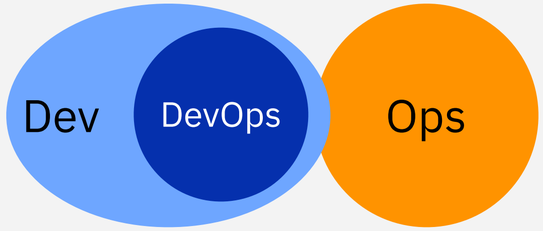
Large Enterprises



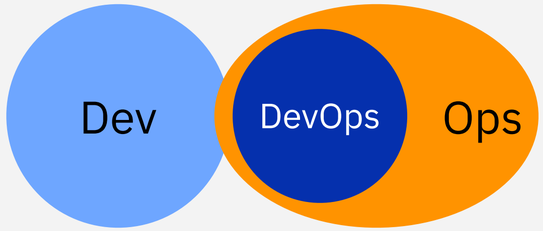
Traditional
Development &
Operations



DevOps is a separate team



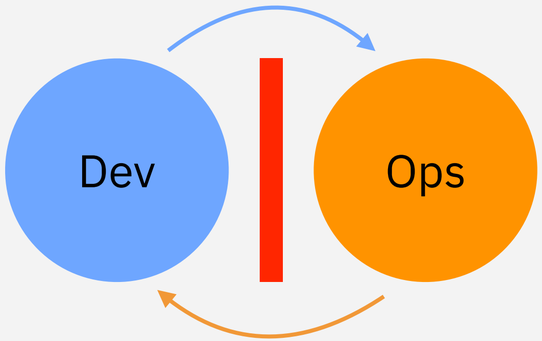
DevOps is something Devs do



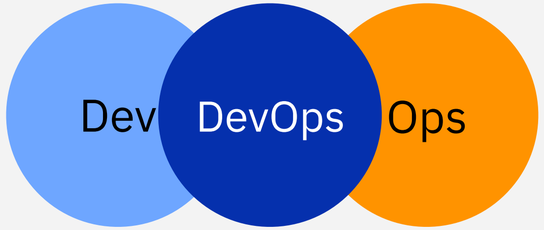
DevOps is something Ops does

Different Perspectives of DevOps

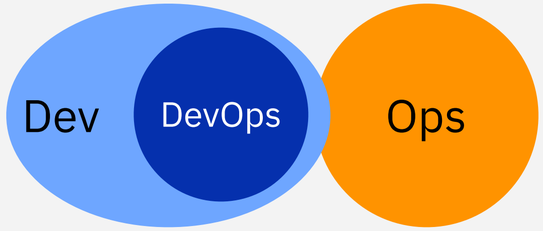
Large Enterprises



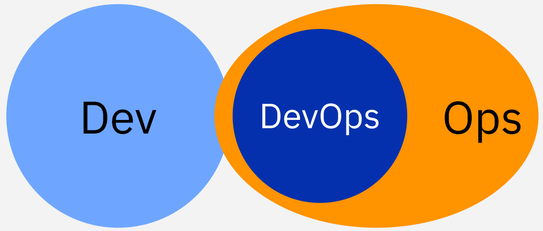
Traditional Development & Operations



DevOps is a separate team



DevOps is something Devs do



DevOps is something Ops does

Cloud Native Startups

DevOps is the company culture

There's No Such Thing as a "DevOps Team"

That’s an anti-pattern

“The DevOps movement addresses the dysfunction that results from organizations composed of functional silos. Thus, creating another functional silo that sits between dev and ops is clearly a poor (and ironic) way to try and solve these problems.”

– Jez Humble, The DevOps Handbook

https://continuousdelivery.com/2012/10/theres-no-such-thing-as-a-devops-team/

DevOps Organizational Objective

Shared Consciousness

...with

Distributed (local) Control

“Bad behavior arises when you abstract people away from the consequences of their actions.”

– Jez Humble

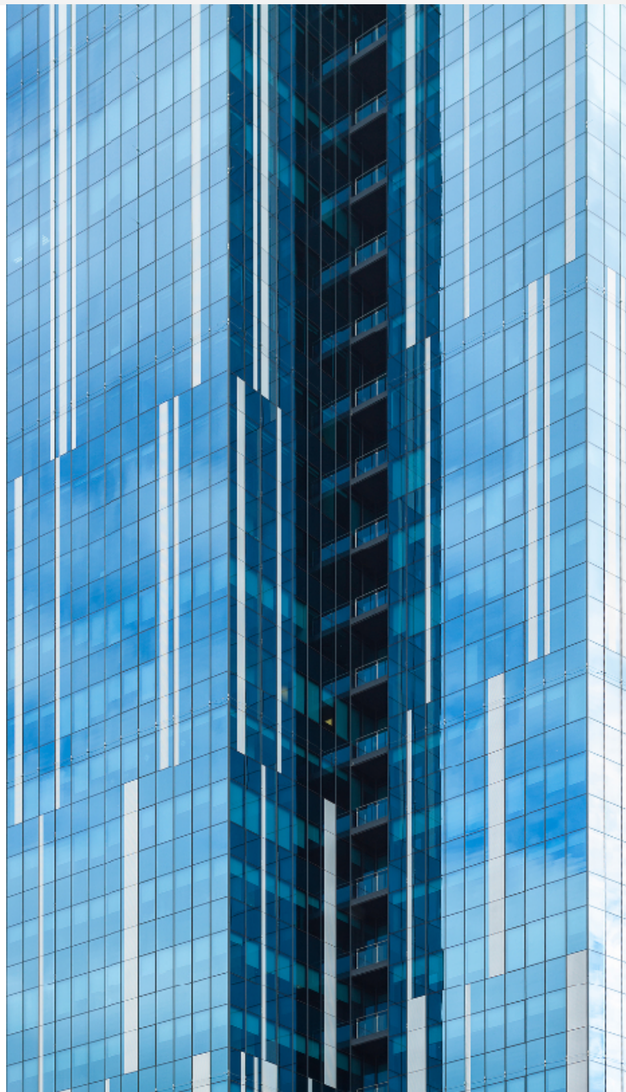
https://continuousdelivery.com/2012/10/theres-no-such-thing-as-a-devops-team/

Functional Silos Breed Bad Behavior

- Bad behavior arises when you abstract people away from the consequences of their actions.
- Functional silos abstract people away from the consequences of their actions.
- For example: By adding a QA Team, developers are abstracted away from the consequences of writing buggy code.

Actions have Consequences

- **Make people aware of the consequences of their actions**
  - Create cross-functional teams - or -
  - Have developers rotate through operations teams
  - Have operations people attend developer standups and showcases
- **Make people responsible for the consequences of their actions**
  - Having developers on Pager Duty, or own the SLA for the products and services they build



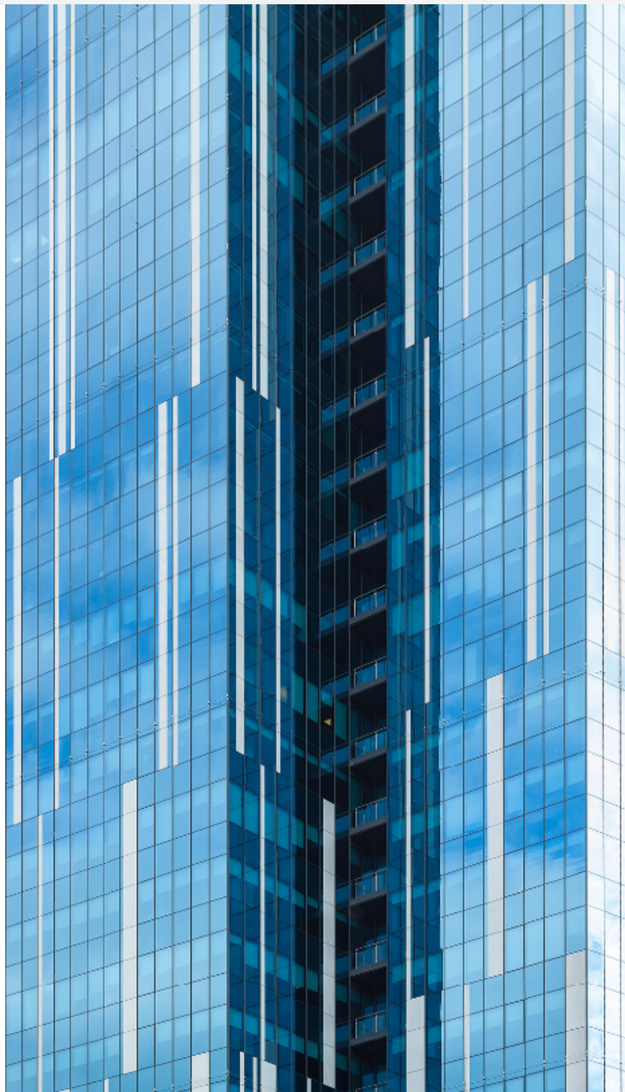
https://continuousdelivery.com/2012/10/theres-no-such-thing-as-a-devops-team/

“You build it, you run it!”

–Werner Vogels, CTO of Amazon

The project model is fundamentally flawed

- **Software development efforts are usually run as if they were civil engineering projects**
  - When a project is complete, the system gets tossed over the wall to operations to run and maintain as part of a "business as usual" effort
  - All the people in the project team get reallocated to new work
- **The project model is fundamentally flawed as a way of doing software development**
  - Software development should be treated as product development instead



https://continuousdelivery.com/2012/10/theres-no-such-thing-as-a-devops-team/

How do you change a culture?

You must change the way people are measured

On the folly of rewarding for A, while hoping for B

"Whether dealings with monkeys, rats, or human beings, it is hardly controversial to state that most organisms seek information concerning what activities are rewarded, and then seek to do (or at least pretend to do) those things, often to the exclusion of activities not rewarded. The extent to which this occurs of course will depend on the perceived attractiveness of the rewards offered, but neither operant nor expectancy theorists would quarrel with the essence of this notion."

–Steven Kerr, Ohio State University

Academy of Management Journal, Dec 1975

You get what you measure

Measure what Matters

...because you get what you Measure

- If you measure widget production you will get **lots of widgets**
- If you measure lines of code you will get **many lines of code**
- If you measure people against each other by ranking them, you will get **anti-social behavior**
- But if you measure developers by their social interaction and sharing... you will get **Social Coders!**

Vanity Metrics

...good for feeling awesome, bad for action

- Consider the total number of daily “hits” to your website is 10,000
- Now what? (what does a "hit" represent?)
  - Do you really know what actions you took in the past that drove those visitors to you?
  - Do you really know which actions to take next?
  - In most cases, I don’t think it’s very helpful

Actionable Metrics

- Imagine you add a new feature to your website, and you do it using an A/B split-test in which 50% of customers see the new feature and the other 50% don't.
- A few days later, you take a look at the revenue you've earned from each set of customers, noticing that group B has 20% higher revenue per-customer.
- **Think of all the decisions you can make:**
  - Obviously, roll out the feature to 100% of your customers
  - Continue to experiment with more features like this one and …
  - Realize that you've probably learned something that's particular valuable to your customers.

http://tim.blog/2009/05/19/vanity-metrics-vs-actionable-metrics/

Top 4 Actionable Metric

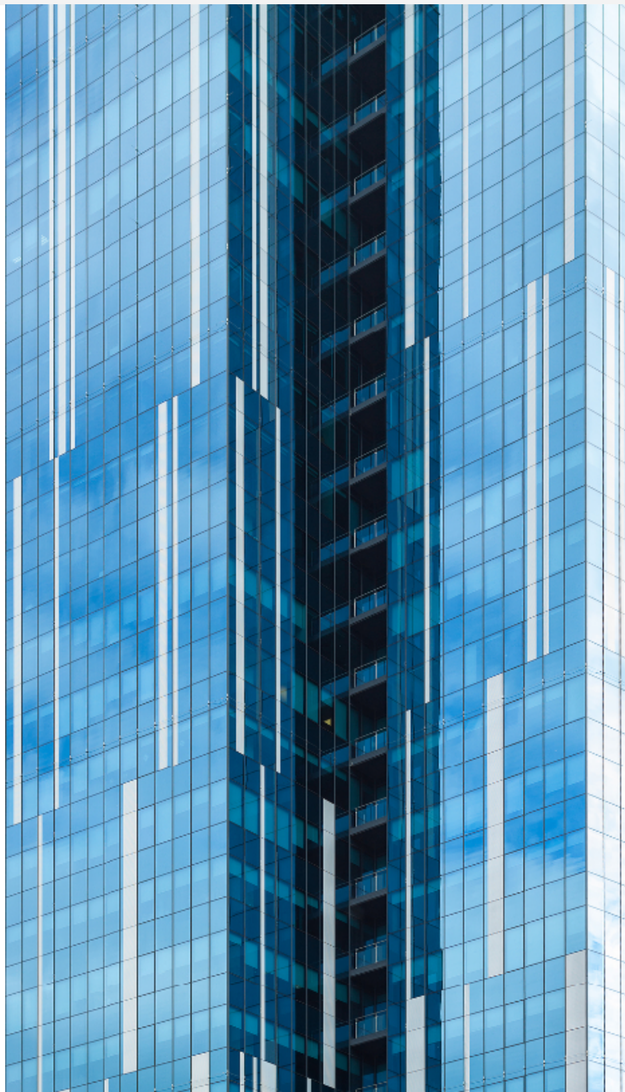
1. **Mean Lead Time**
   - How long does it take from idea to production?
2. **Release Frequency**
   - How often can you deliver changes?
3. **Change Failure Rate**
   - How often to changes fail?
4. **Mean Time to Recovery (MTTR)**
   - How quickly can you recover from failure?

Nicole Forsgren - AWS re:Invent 2017: Tools Won’t Fix Your Broken DevOps
https://www.youtube.com/watch?v=gsjCWrCUjNg

“Measurements should encourage innovation and collaboration, and not punish failure”

Busted DevOps Myths

- You cannot buy DevOps In-A-Box
- You cannot order 20 units of DevOps for this quarter
- You cannot sprinkle DevOps on something to make it better
- You cannot become DevOps without changing your culture
  - You can’t change your companies culture just by adopting new tools ...but they can help reinforce it
- Using Containers won't fix your broken culture
- You cannot maintain your current organizational structure and become DevOps



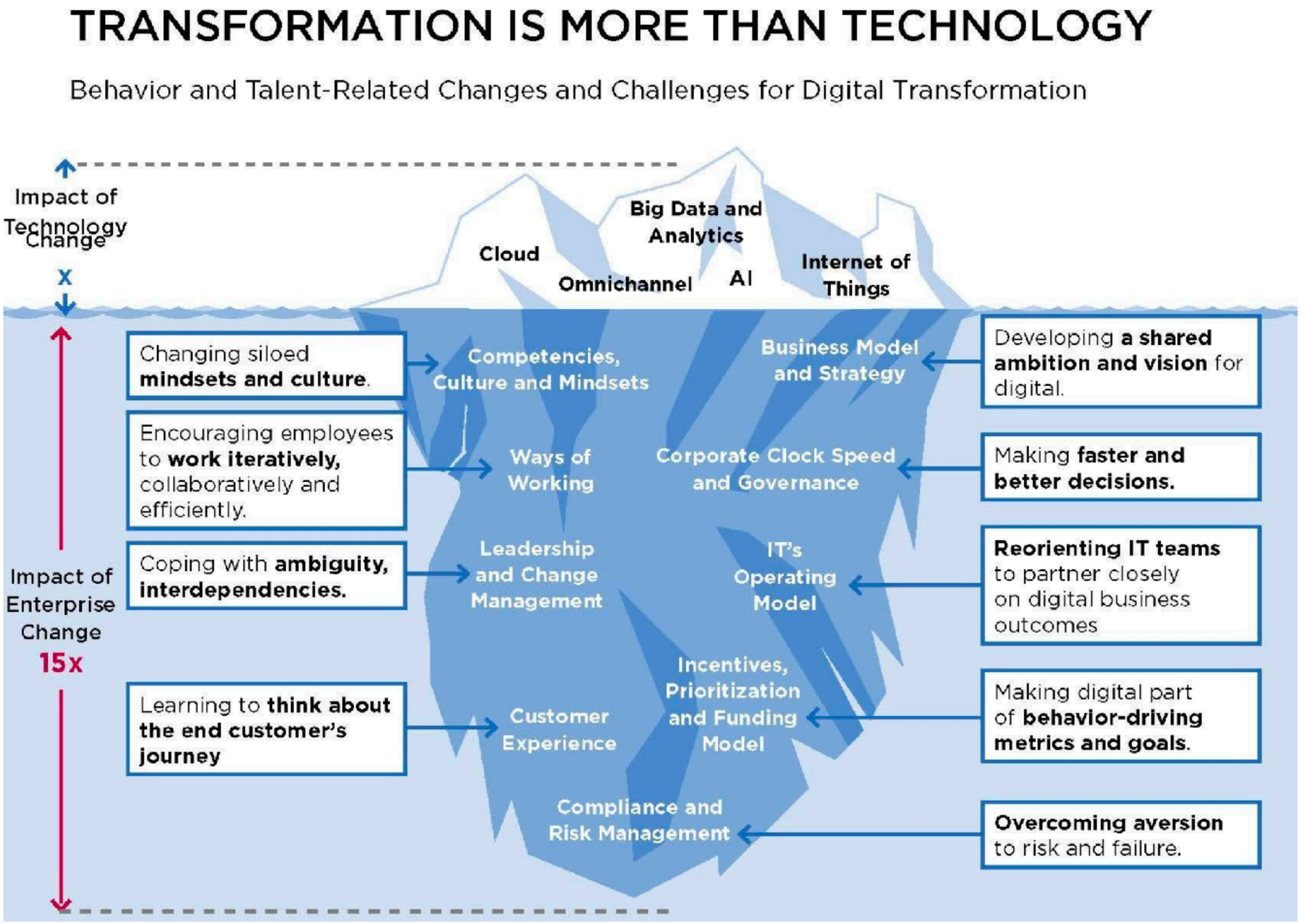
“Don't fear failure. Fear being in the exact same place next year as you are today.”

Technology is the easy part

Changing delivery teams to be aligned to the business services, evolving service management from reactive to proactive, and implementing Site Reliability Engineering practices often are a longer journey than the technology transformation.

See: IBM CSMO

https://www.ibm.com/cloud/garage/architectures/serviceManagementArchitecture/



DevOps Maturity Matrix*

* Cloud Computing: The Cloud and DevOps by Dave Linthicum

Maturity Level	People	Process	Technology
Level 1 Ad Hoc	• Silo based • Blame and finger-pointing • Dependent on experts • Lack of accountability	• Manual processes • Tribal knowledge the norm • Unpredictable and reactive	• Manual builds and deployments • Manual testing • Environmental inconsistencies
Level 2 Repeatable	• Managed communications • Limited knowledge sharing	• Processes established within silos • No standards • Can repeat what is known, but can't react to unknowns	• Automated builds • Automated tests written as part of story development • Painful but repeatable releases
Level 3 Defined	• Collaboration exists • Shared decision making • Shared accountability	• Process automated across the software life cycle • Standards across organization	• Automated build and test cycle for every commit • Push button deployments • Automated user and acceptance testing
Level 4 Measured	• Collaboration based on shared metrics with a focus on removing bottlenecks	• Proactive monitoring • Metrics collected and analyzed against business goals • Visibility and predictability	• Build metrics visible and acted on • Orchestrated deployments with automatic rollbacks • Nonfunctional requirements defined and measured
Level 5 Optimized	• A culture of continuous improvement permeates through the organization	• Self-service automation • Risk and cost optimization • High degree of experimentation	• Zero downtime deployments • Immutable infrastructure • Actively enforce resiliency by forcing failures

Key Takeaways

- DevOps is about breaking down the silos and working as a Single Agile Team
- Culture is the #1 success factor in DevOps. Building a culture of shared responsibility, transparency and faster feedback is the foundation of every high performing DevOps team
- DevOps starts with learning how to work differently. It embraces cross-functional teams with openness, transparency, and respect as pillars
- Being able to recover quickly from failure is more important than having failures less often
- Measurements should encourage innovation and collaboration, and not punish failure (blameless culture)

“If you want to build a ship, don’t drum up the men to gather wood, divide the work, and give orders.
Instead, teach them to yearn for the vast and endless sea.”

– Antoine de Saint-Exupery*

* Author of The Little Prince

We don’t "DO" DevOps

We become DevOps