

Deploy, Release, CI/CD, Oh My!

DevOps for the Rest of us

Coming Soon(ish)



Made with by [Creative Tim](#). Free download [here](#).

 [@IAmJerdog](#)

Jeremy Meiss

Director, DevRel & Community



timeline.jerdog.me





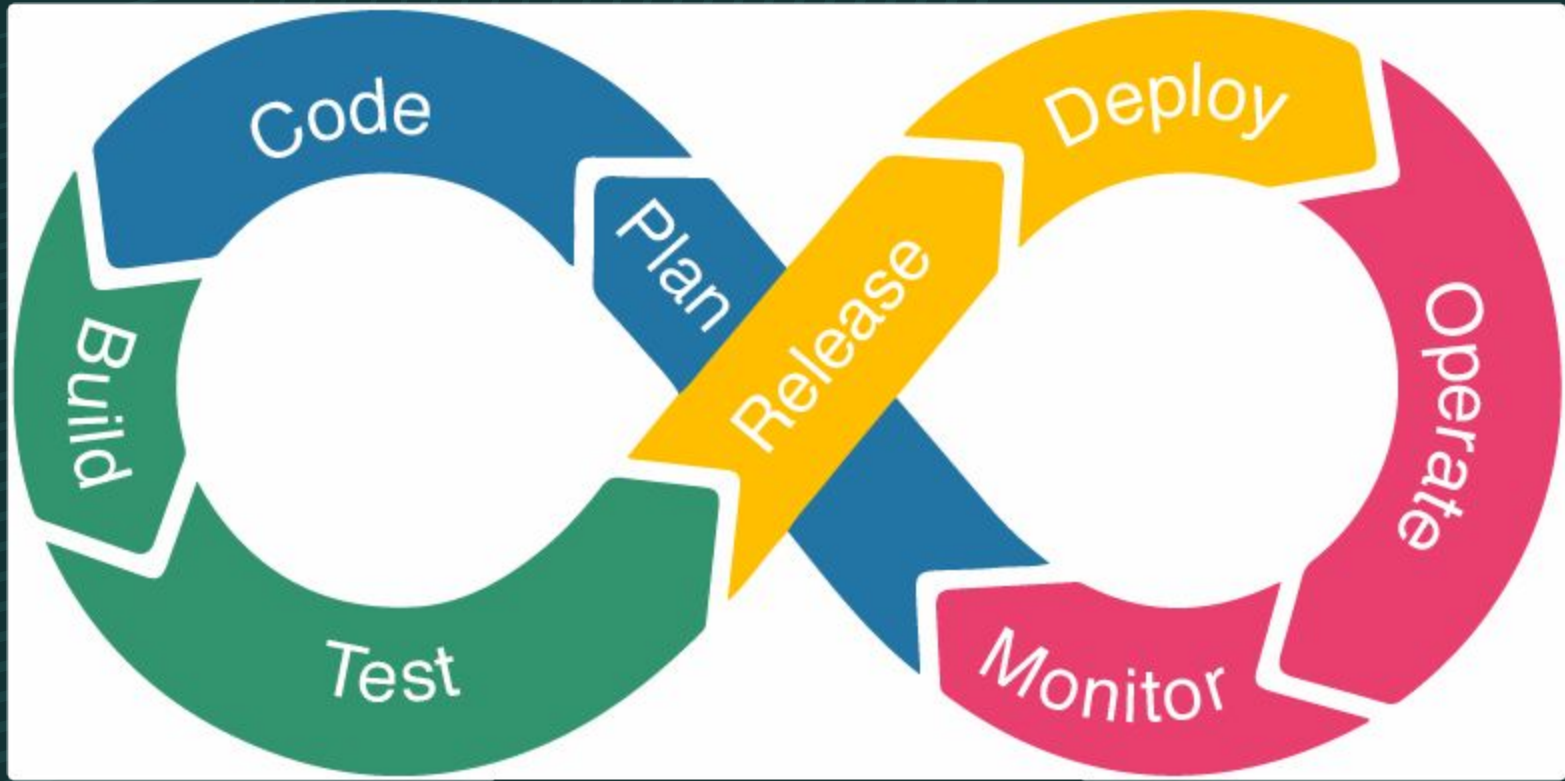


Image credit: [ROELBOB](#) at [DevOps.com](#)

 [@IAmJerdog](#)



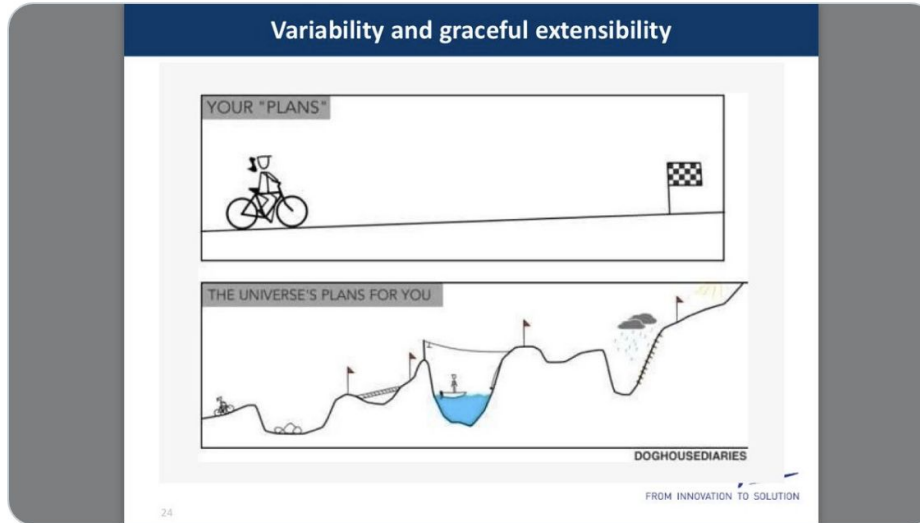
John Allspaw

@allspaw

Normal 0%



Work-as-imagined versus work-as-done



3:00 AM · Apr 28, 2016 · Twitter for iPhone

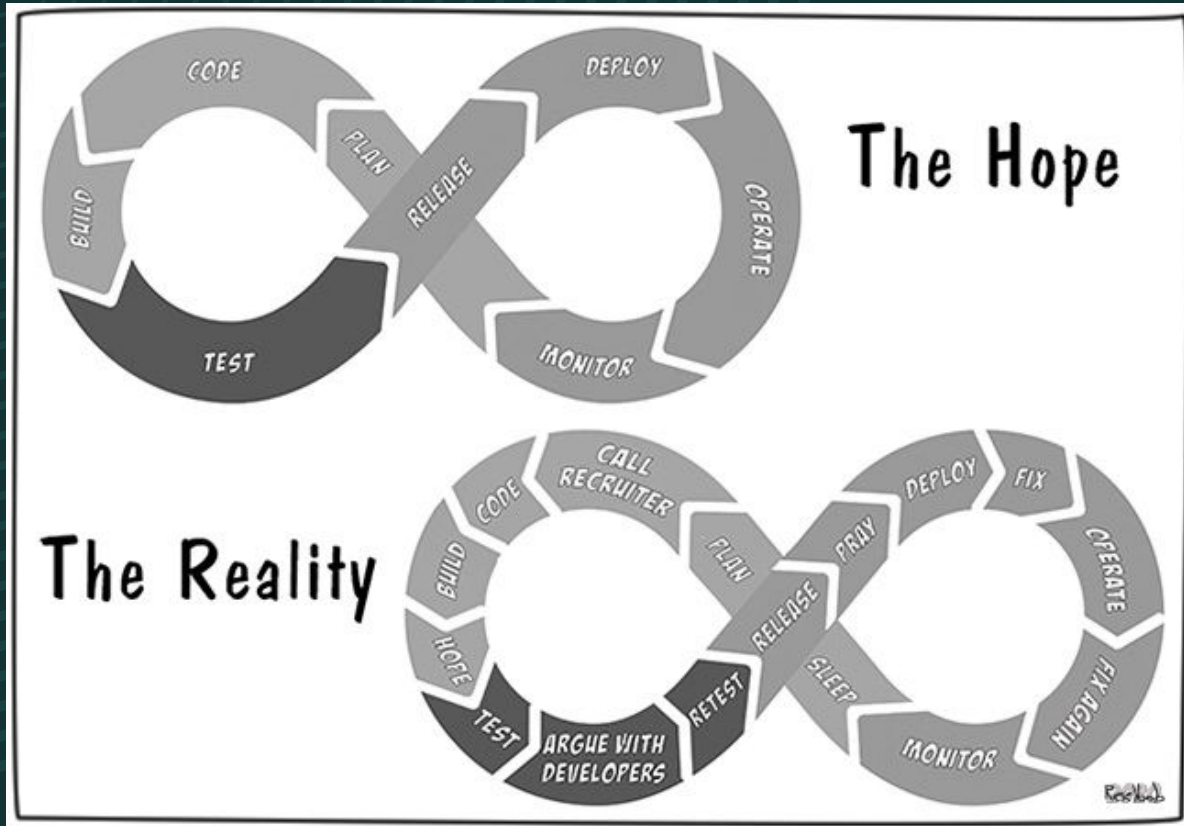


Image credit: [ROELBOB](#) at [DevOps.com](#)

AIOps

“AIOps combines big data and machine learning to automate IT operations processes, including event correlation, anomaly detection, and causality determination.”

— Gartner

“an industry category for machine learning analytics technology that enhances IT operations analytics”
— Wikipedia entry for AIOps

Machine Learning

Big Data

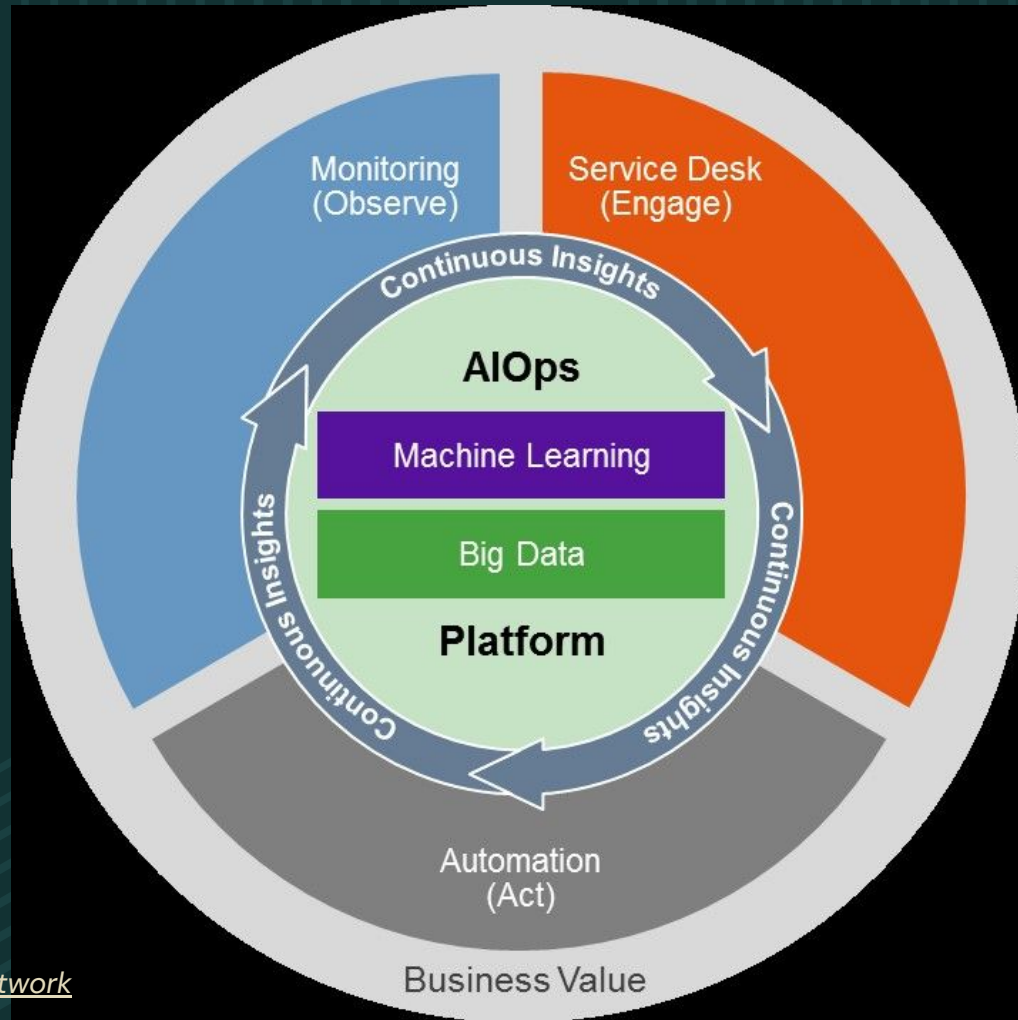
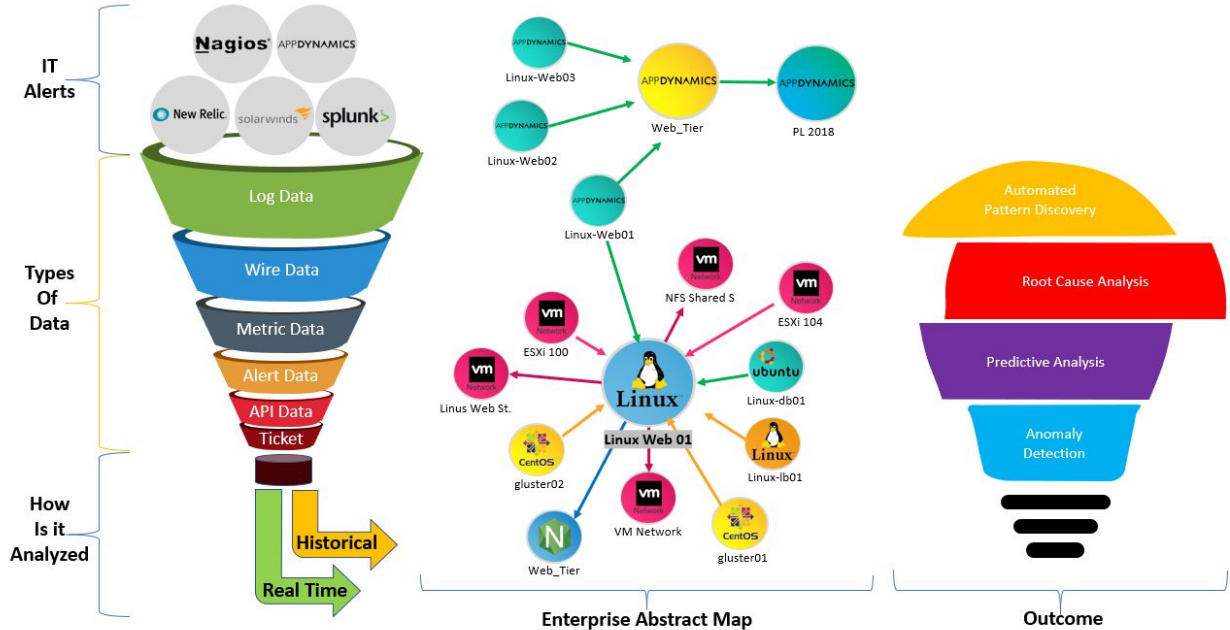


Image credit: [Gartner Blog Network](#)

AIOPS in a nutshell



GitOps



WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Current events
Random article
About Wikipedia
Contact us
Donate

Contribute

Help
Learn to edit
Community portal
Recent changes
Upload file

Tools
Special pages
Printable version

Languages

Not logged in Talk Contributions Create account Log in

Special page

Search Wikipedia

Search results

Help

gitops **Search**

Results 1 – 2 of 2

Advanced search:

Search in:

Did you mean: [git ops](#)

The page "[Gitops](#)" does not exist. You can ask for it to be created, but consider checking the search results below to see whether the topic is already covered.

DevOps (redirect from **GitOps**)

16 August 2021. "Getting Started with **GitOps**". TheNewStack.io. 13 December 2021.

Retrieved 5 April 2022. "**GitOps** Workflows and Principles for Kubernetes"

20 KB (2,224 words) - 13:27, 11 May 2022

Kubeflow

"Kubeflow 1.1 improves ML Workflow Productivity, Isolation & Security, and **GitOps**".

Kubeflow. July 31, 2020. "Release v1.1.0 · kubeflow/kubeflow". GitHub

7 KB (674 words) - 15:47, 8 May 2022

[Privacy policy](#) [About Wikipedia](#) [Disclaimers](#) [Contact Wikipedia](#) [Mobile view](#) [Developers](#) [Statistics](#) [Cookie statement](#)

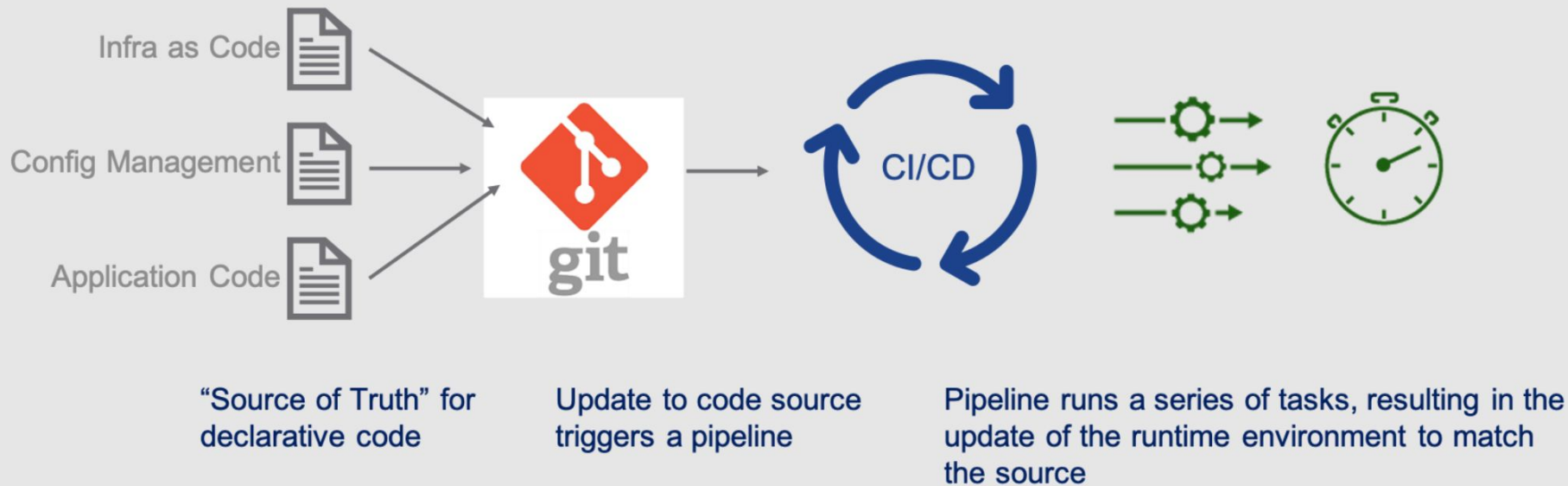






“GitOps is a way to do Kubernetes cluster management and application delivery... by using Git as a single source of truth for declarative infrastructure and applications.”

— Weaveworks



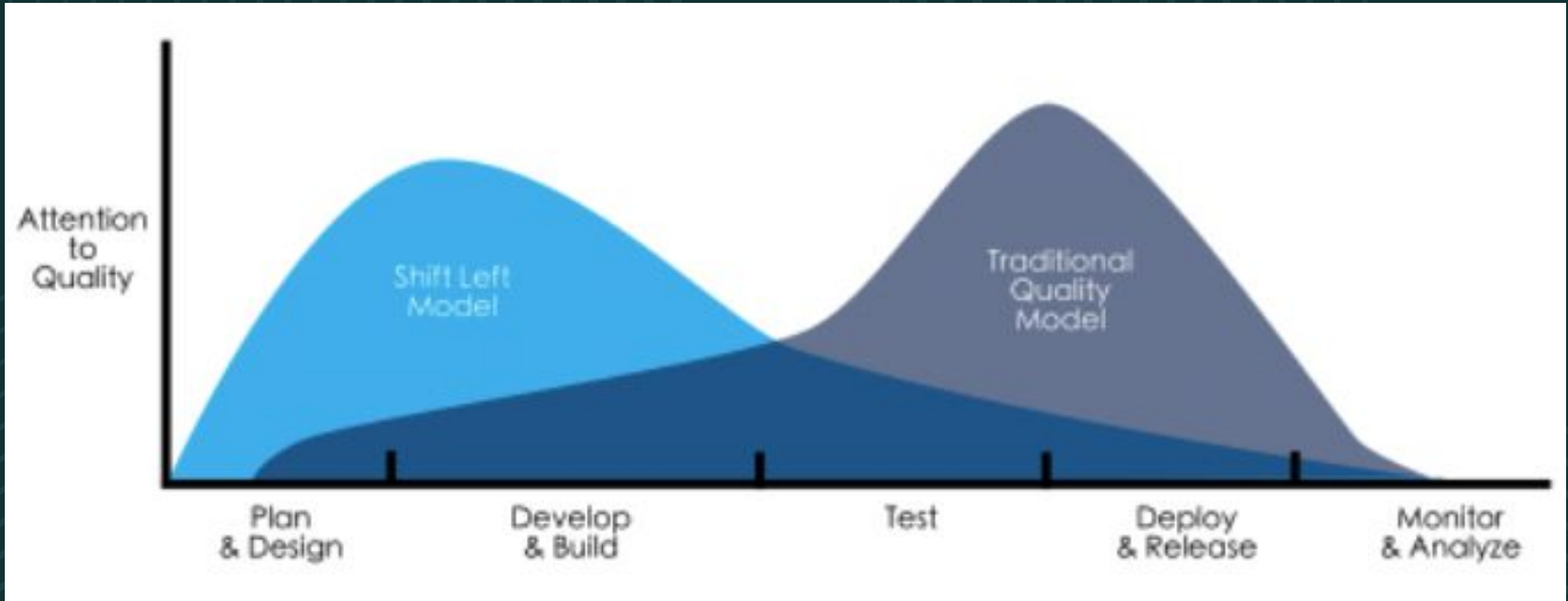
DevSecOps

“DevSecOps puts security at the forefront of the development process as a whole, ensuring that good cyber-hygiene remains top-of-mind for developers and operators from start to finish.”
— Phil Richards

“an augmentation of DevOps to allow for security practices to be integrated into the DevOps approach... Each delivery team is empowered to factor in the correct security controls into their software delivery. Security practices and testing are performed earlier in the development lifecycle...”

— Wikipedia entry for DevSecOps

“Shifting security left”



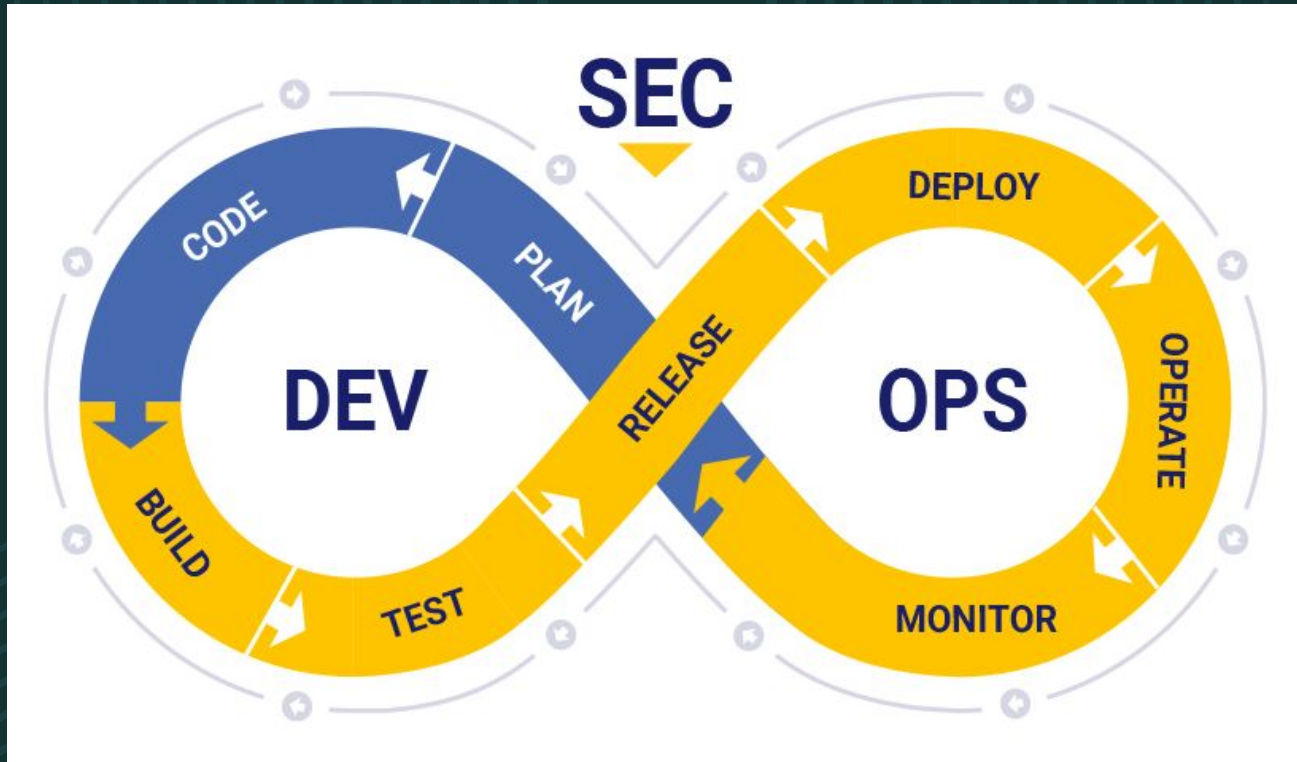


Image credit: Threatpost

Progressive Delivery

“Progressive delivery is continuous delivery with fine-grained control over the blast radius.”
— James Governor, RedMonk

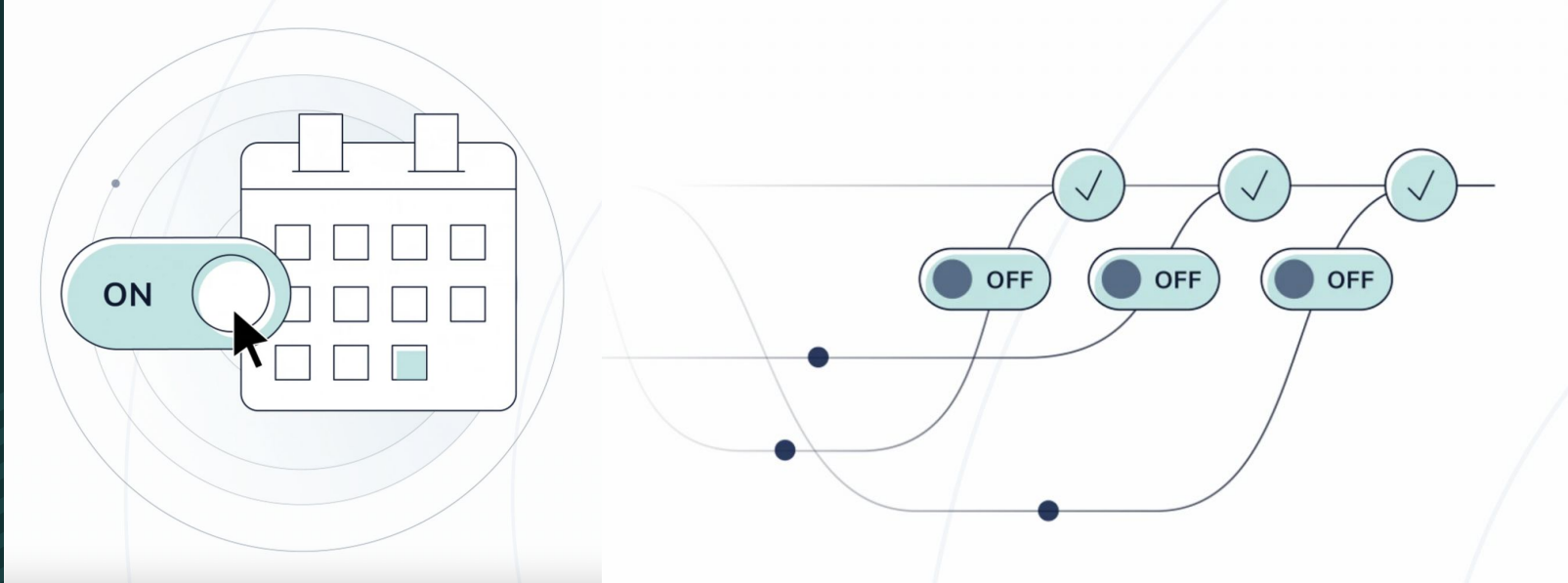


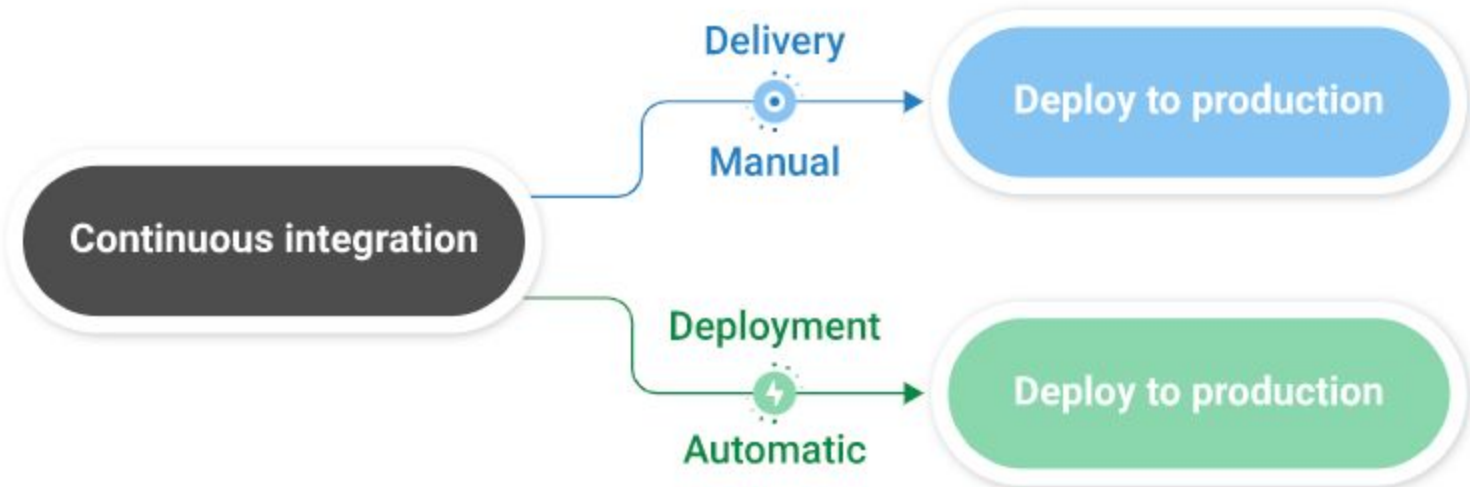


Image credit: [LaunchDarkly](#)

Is it a “release”, or a
“deploy(ment)?”

Deploy





Release



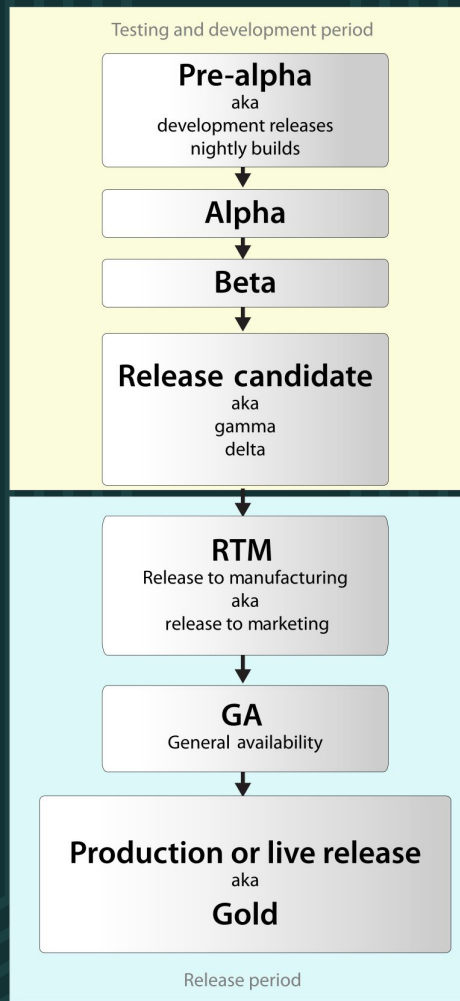


Image credit: [Wikipedia](#)

Continuous {Integration|Deployment}

Agile software development principles [\[edit \]](#)

The *Manifesto for Agile Software Development* is based on twelve principles:^[23]

1. Customer satisfaction by early and continuous delivery of valuable software.
2. Welcome changing requirements, even in late development.
3. Deliver working software frequently (weeks rather than months)
4. Close, daily cooperation between business people and developers
5. Projects are built around motivated individuals, who should be trusted
6. Face-to-face conversation is the best form of communication (co-location)
7. Working software is the primary measure of progress
8. Sustainable development, able to maintain a constant pace
9. Continuous attention to technical excellence and good design
10. Simplicity—the art of maximizing the amount of work not done—is essential
11. Best [architectures](#), requirements, and designs emerge from self-organizing teams
12. Regularly, the team reflects on how to become more effective, and adjusts accordingly

Framework	Main contributor(s)
Adaptive software development (ASD)	Jim Highsmith, Sam Bayer
Agile modeling	Scott Ambler, Robert Cecil Martin
Agile unified process (AUP)	Scott Ambler
Disciplined agile delivery	Scott Ambler
Dynamic systems development method (DSDM)	Jennifer Stapleton
Extreme programming (XP)	Kent Beck, Robert Cecil Martin
Feature-driven development (FDD)	Jeff De Luca
Lean software development	Mary Poppendieck, Tom Poppendieck
Lean startup	Eric Ries
Kanban	Taiichi Ohno
Rapid application development (RAD)	James Martin
Scrum	Ken Schwaber, Jeff Sutherland
Scrumban	
Scaled agile framework - SAFe	Scaled Agile, Inc.



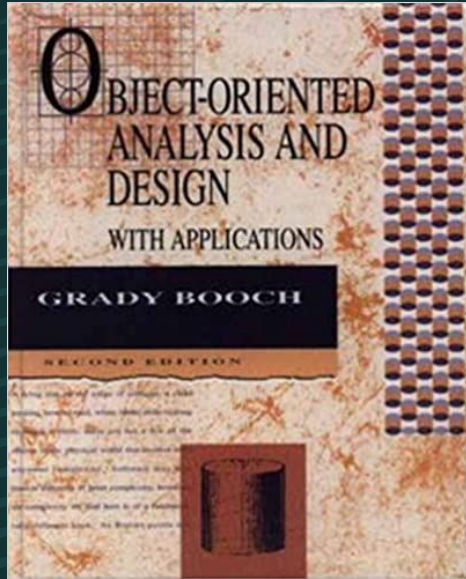
Scrumban!

Practice	Main contributor(s)
Acceptance test-driven development (ATDD)	
Agile modeling	
Agile testing	
Backlogs (Product and Sprint)	Ken Schwaber
Behavior-driven development (BDD)	Dan North, Liz Keogh
Continuous integration (CI)	Grady Booch
Cross-functional team	
Daily stand-up / Daily Scrum	James O Coplien
Domain-driven design (DDD)	Eric Evans
Iterative and incremental development (IID)	
Pair programming	Kent Beck
Planning poker	James Grenning, Mike Cohn
Refactoring	Martin Fowler
Retrospective	
Scrum events (sprint planning, sprint review and retrospective)	
Specification by example	
Story-driven modeling	Albert Zündorf
Test-driven development (TDD)	Kent Beck
Timeboxing	
User story	Alistair Cockburn
Velocity tracking	

Continuous integration (CI)

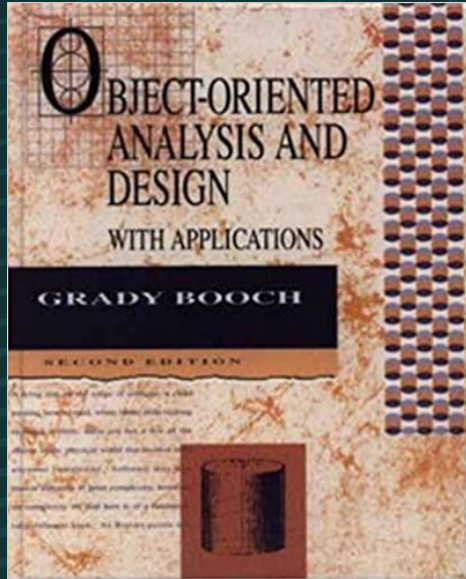
Grady Booch

Quick History of CI/CD

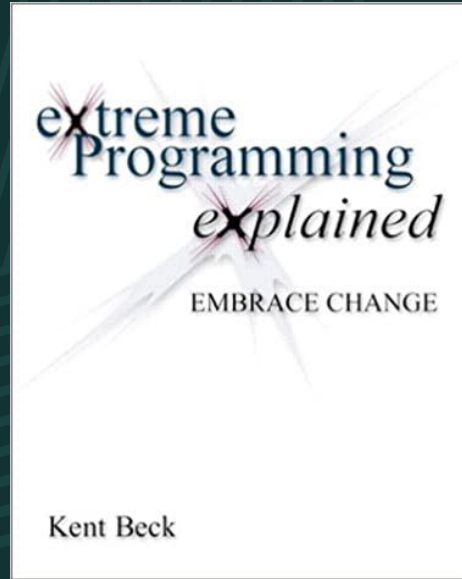


1991

Quick History of CI/CD

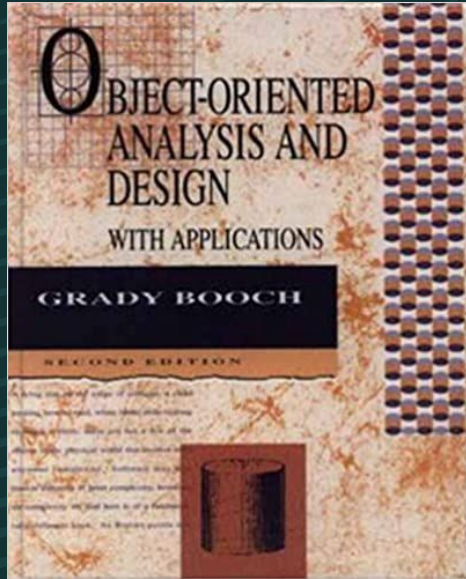


1991

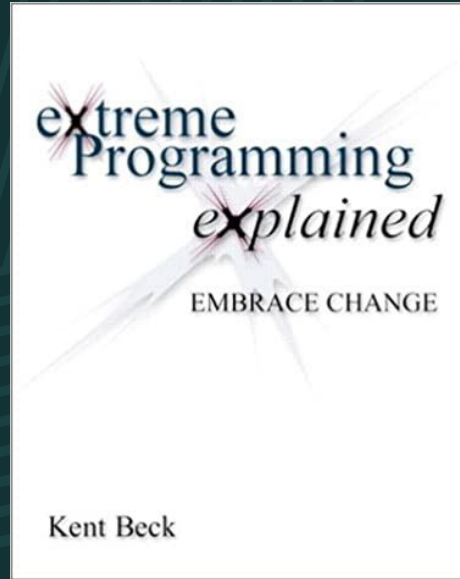


1997

Quick History of CI/CD



1991



1997



2001

Quick History of CI/CD



Quick History of CI/CD



Quick History of CI/CD

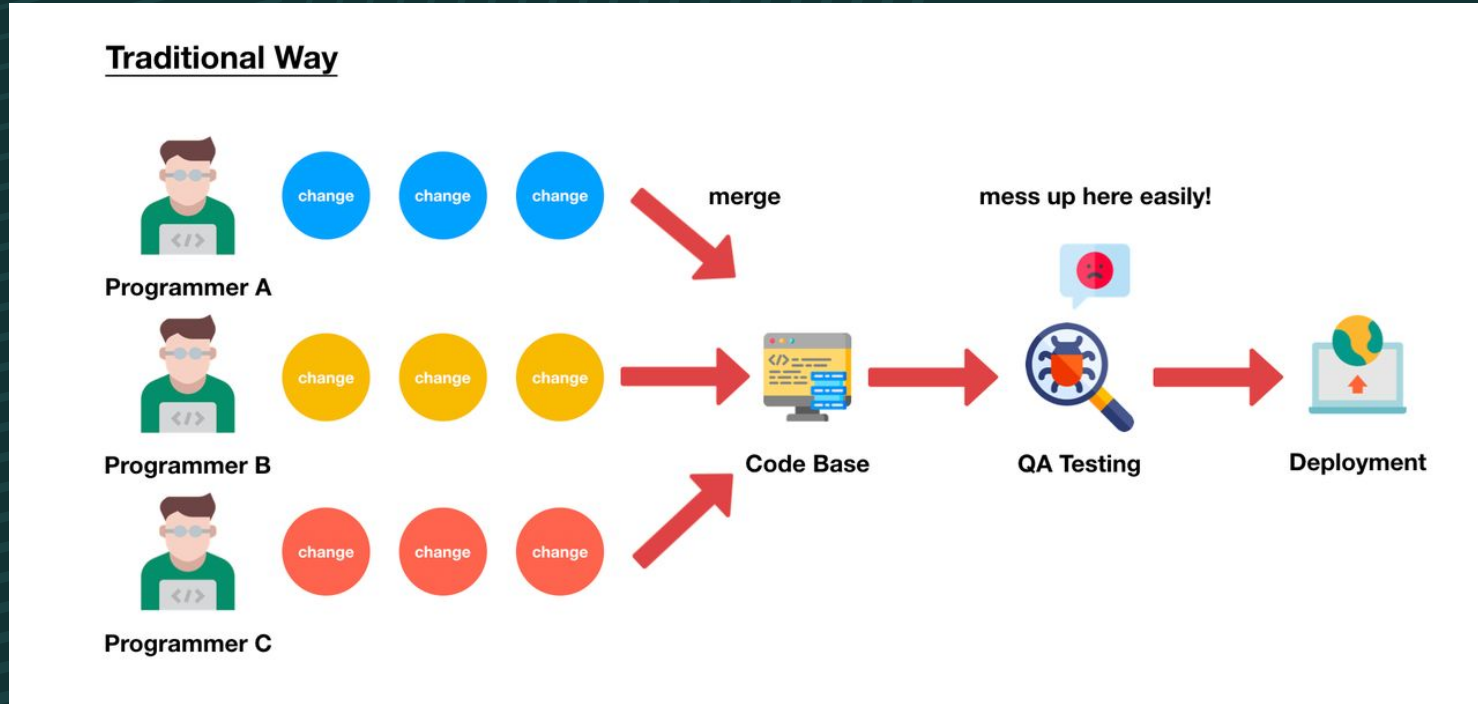


circleci

Software Development without CI/CD



Software Development without CI/CD



CI == Continuous Integration

the **practice** of merging all developers' working copies to a **shared** code repository



Code



Shared Code
Repo



CI/CD Runtime



Test



Scan for
Vulnerabilities



Build Website

CI == Continuous Integration

- Merge code changes often



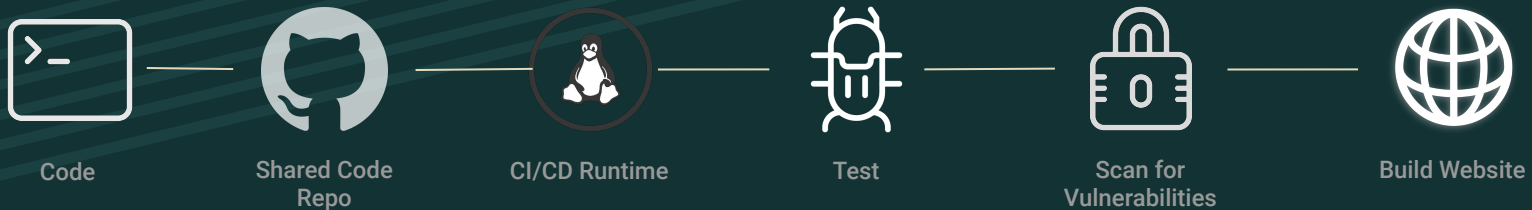
CI == Continuous Integration

- Merge code changes often
- Run automated tests to validate builds



CI == Continuous Integration

- Merge code changes often
- Run automated tests to validate builds
- Only integrate tested code into code base



CI == Continuous Integration

- Merge code changes often
- Run automated tests to validate builds
- Only integrate tested code into code base
- Changes frequently merged into release branches



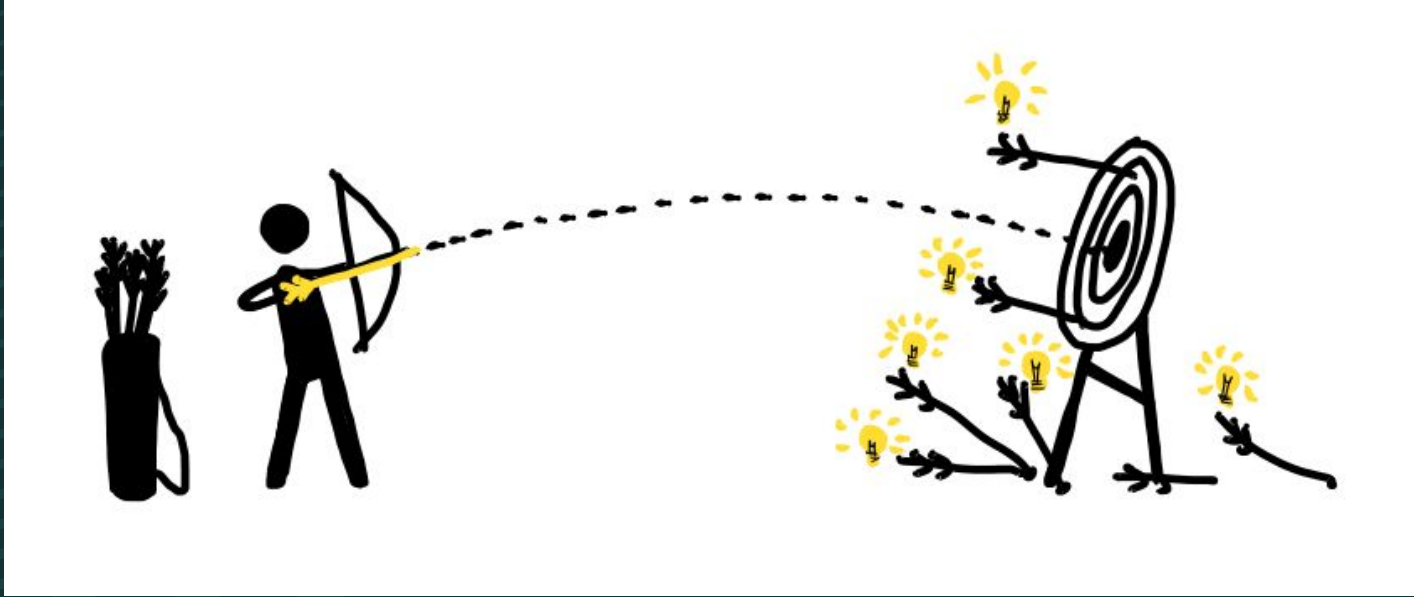


Image credit: [Product Coalition](#)

CD == Continuous Deployment

the **practice** of automatically deploying new software releases to **target environments**



Build Website



Server
deployment



Active Users

CD == Continuous Deployment

- Faster release cycles



Build Website



Server
Deployment



Active Users

CD == Continuous Deployment

- Faster release cycles
- Low-risk releases



Build Website



Server
Deployment



Active Users

CD == Continuous Deployment

- Faster release cycles
- Low-risk releases
- Higher quality



Build Website



Server
Deployment



Active Users

CD == Continuous Deployment

- Faster release cycles
- Low-risk releases
- Higher quality
- Lower costs



Build Website



Server
Deployment



Active Users

“Is CI/CD a TOOL?”

“It depends...”
-every DevOps Advocate



Are there benefits to CI/CD?

CI/CD Benefits



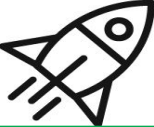



- Improve team productivity & efficiency
- Accelerate speed to market
- Identify product/market fit & ROI
- Release higher quality, more stable products
- Increase customer satisfaction
- Keep devs happy and shipping code

I feel the need... the
need for SPEED!





CI/CD Benchmarks for high performing teams

		Suggested Benchmarks
	Throughput The average number of workflow runs per day	Merge on any pull request
	Duration The average length of time for a workflow to run	10 minutes
	Mean time to recovery The average time between failures & their next success	Under 1 hour
	Success rate The number of successful runs / the total number of runs over a period of time	90% or better on default branch

Full Report



<https://circle.ci/ssd2022>

Thank you.

For feedback and swag: circle.ci/jeremy

glue₂₀₂₂
conference



Timeline.jerdog.me



[IAmJerdog](https://twitter.com/IAmJerdog)



[jerdog](https://dev.to/jerdog)



[/in/jeremyeiss](https://in.linkedin.com/in/jeremyeiss)