Adding Security to DevOps

Mixing security with modern ways of working

Mats Persson

Security Consultant at Omegapoint

Secure Development

Modern Ways of Working

Cloud Security

DevOps vs “good luck with the release”

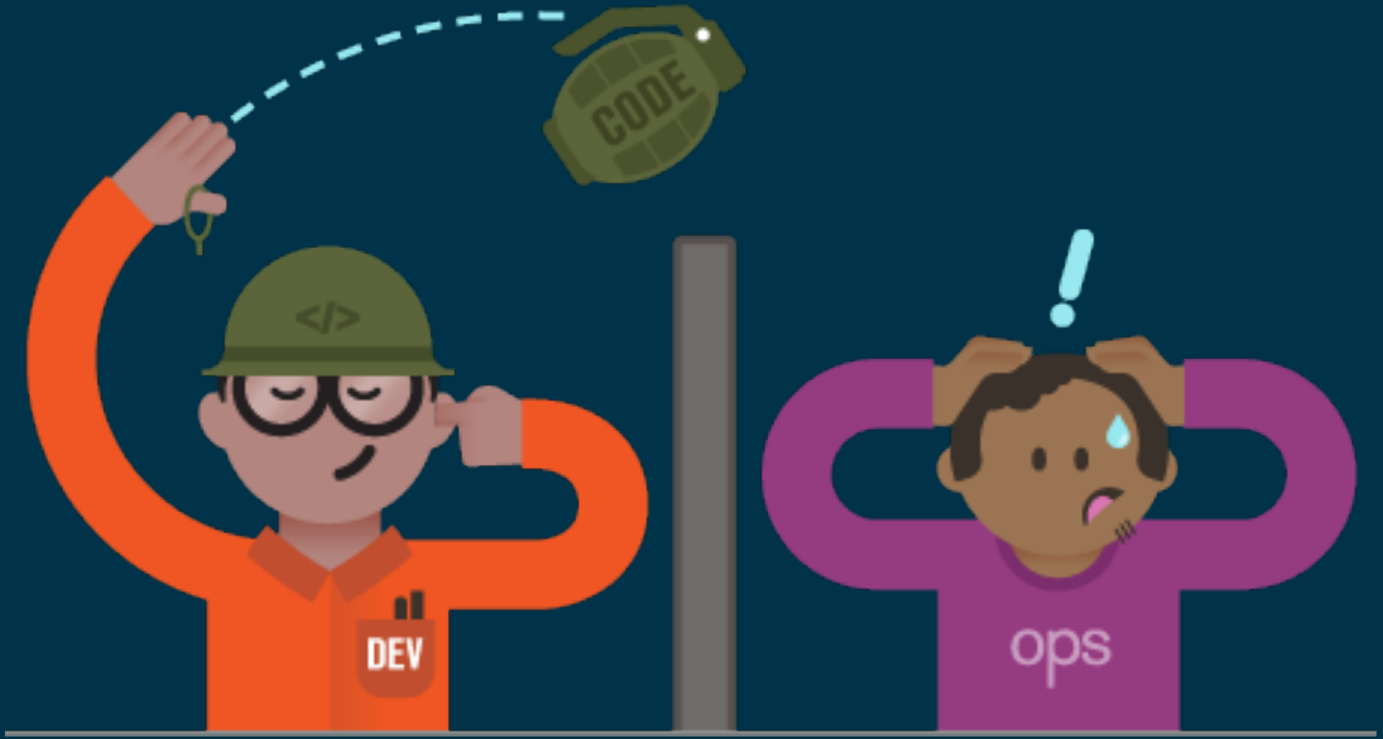


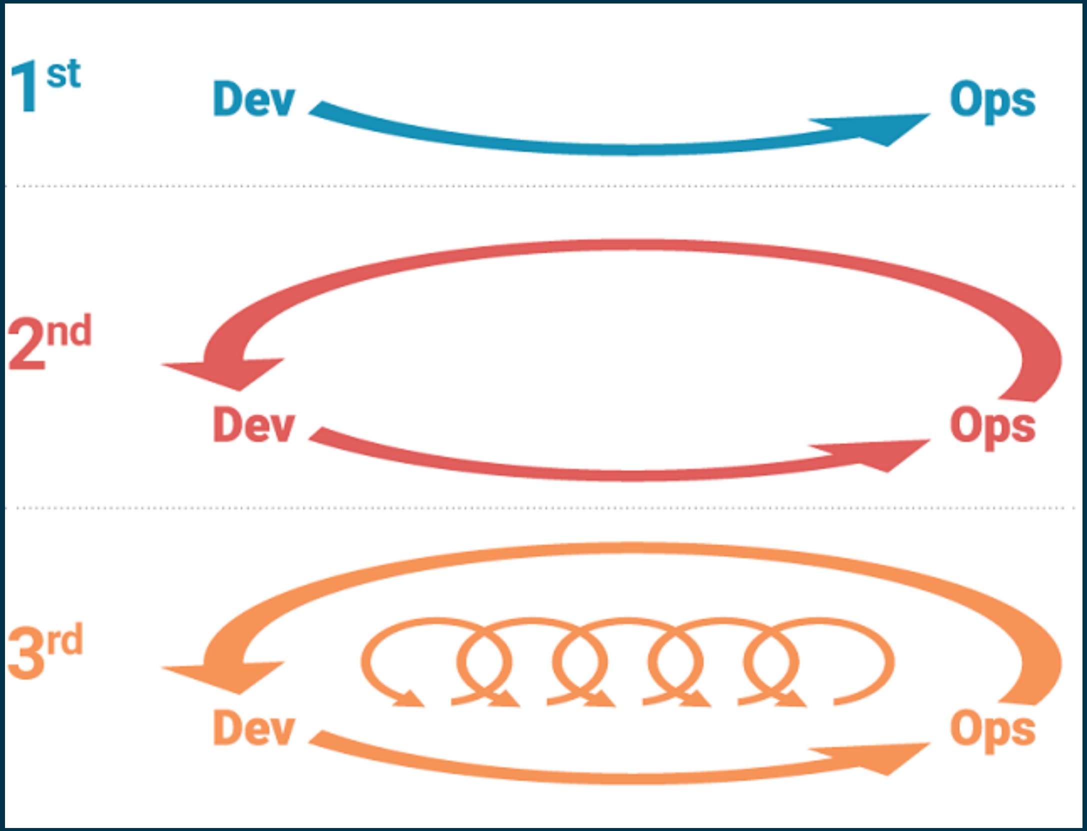
Image from “The Stack”

DevOps three ways – from “The Phoenix Project”

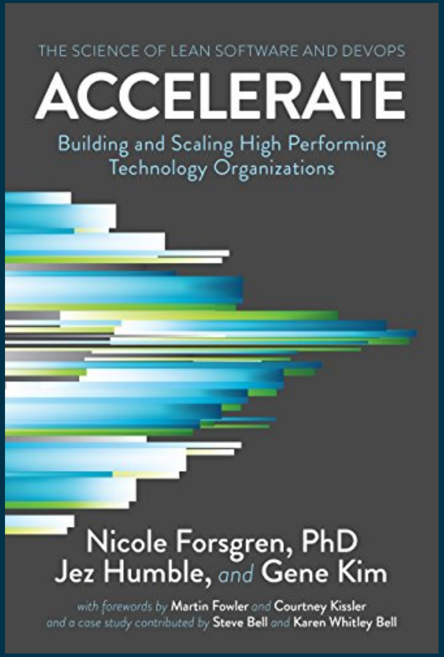
Flow

Feedback

Continuous Learning



The DevOps research


THE SCIENCE OF LEAN SOFTWARE AND DEVOPS
ACCELERATE
Building and Scaling High Performing Technology Organizations
Nicole Forsgren, PhD
Jez Humble, and Gene Kim
with forewords by Martin Fowler and Courtney Kissler
and a case study contributed by Steve Bell and Karen Whitley Bell

Elite performers vs low performers (2021):

- 6570 times faster lead time from commit to deploy (1h vs 6-12 months)
- 973 times more frequent code deployments (4/day vs 1-2/year)
- 6570 times faster mean time to recover from downtime (1h vs 6-12 months)
- 3 times lower change failure rates (1/3 as likely for a change to fail)

- The DORA Four Key Metrics

The DevOps paradox

“High performers deliver more, faster, and with higher stability”

Speed: Faster code to production and more frequent releases

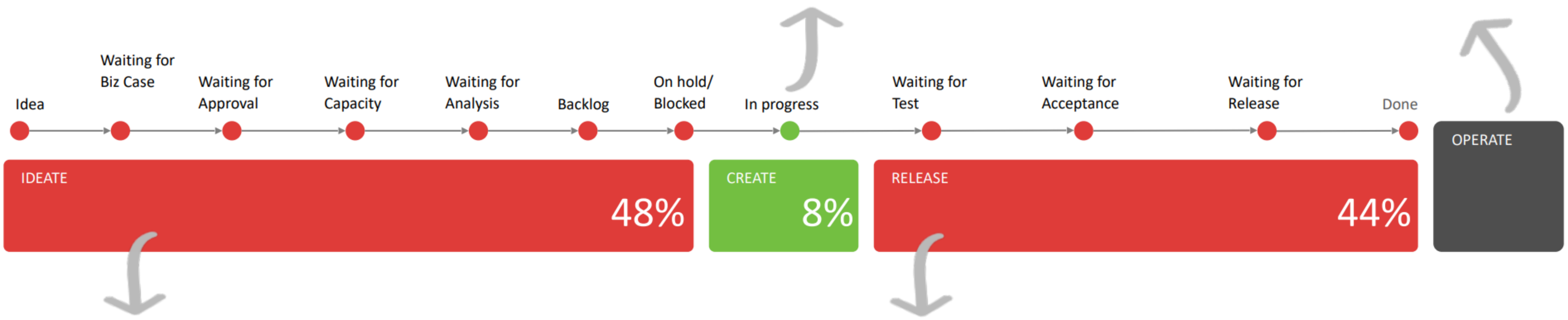
Stability: Lower change failure rate and mean time to recover

They move in tandem!

Slower process -> less stable

(not practicing to release, more changes between releases)

Most organizations are not ready to scale delivery



"Agile" but not nimble
Relative acceleration during development

Poor quality creates unplanned work
Incidents and war rooms throw off plans

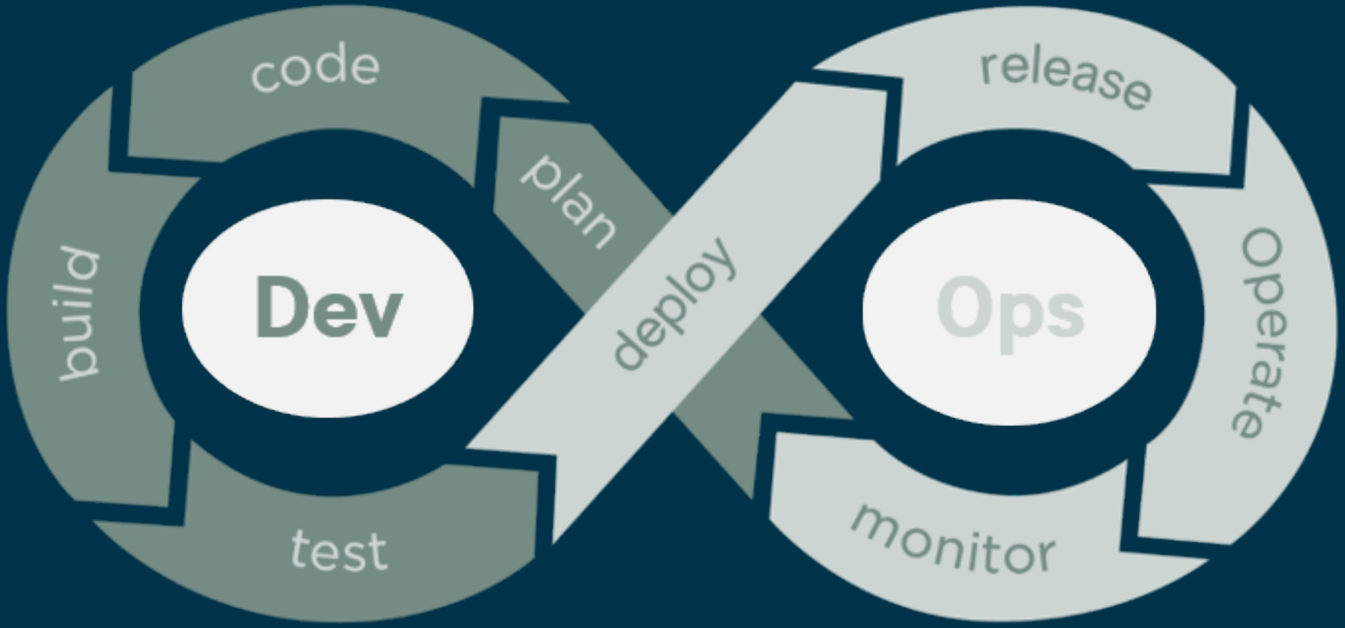
Heavy planning and funding processes
Starting and stopping from frequent cancellation, replanning and reprioritization

DevOps but not at scale
Delivery slowed down by dependencies,
tech debt and manual processes

Mik Kersten - author of "Project to Product"

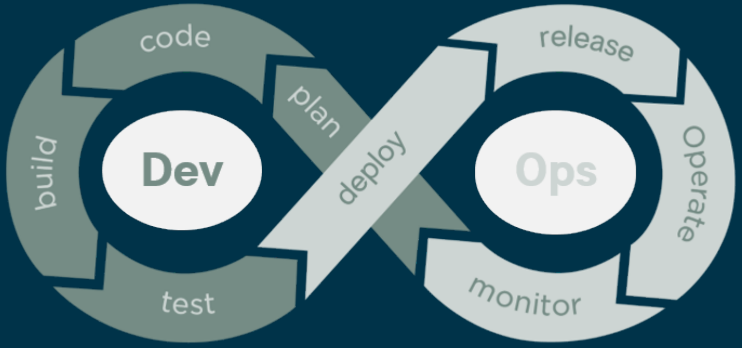
planview

A secure DevOps inspired development lifecycle

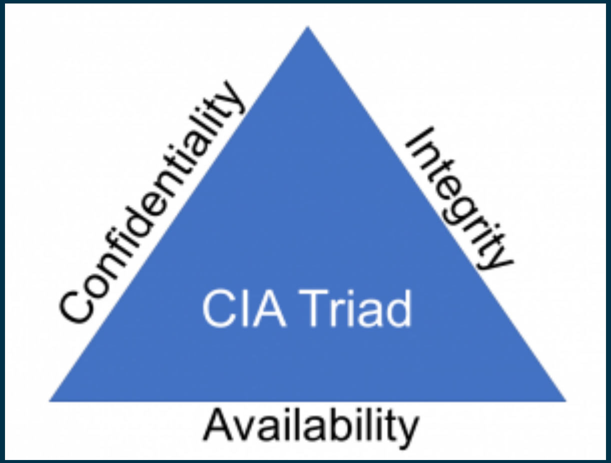


The DevOps loop and some good practices

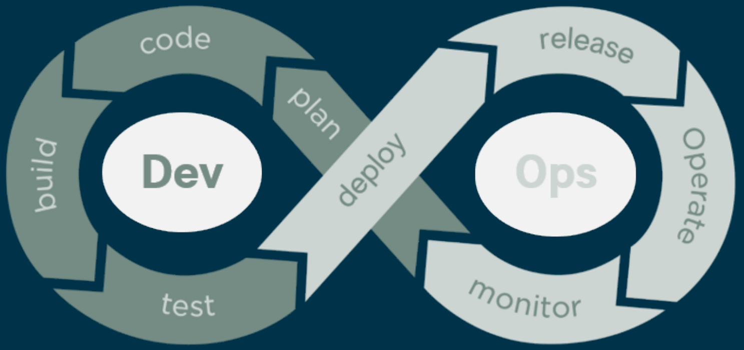
Plan



- Consider confidentiality, integrity, and availability (C-I-A) requirements for your system or application:
    - Confidentiality - prevent unauthorized disclosure of information
    - Integrity – prevent unauthorized modification of information
    - Availability – ensure information is available when needed
- Discuss what is important for you
- Don’t forget the Privacy aspects

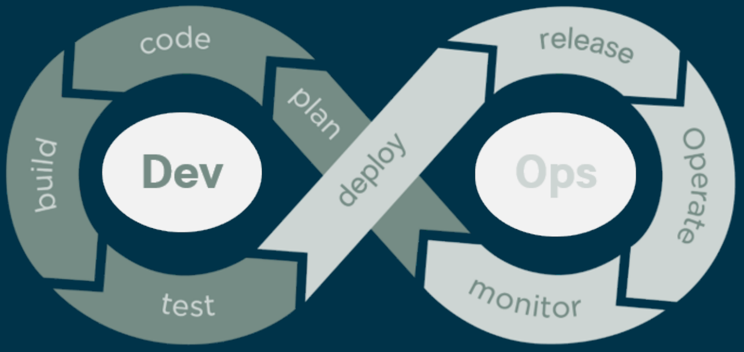


Plan



- Plan for all 4 work item types
    - Features
    - Defects
    - Risk (regulatory, security, compliance)
    - Technical Debt (old software, architecture, test/build automation)
- Spend at least 20% of the team's time on “technical excellence” (Risk and Technical Debt above)

Plan



Features

Risks/Debts

Defects

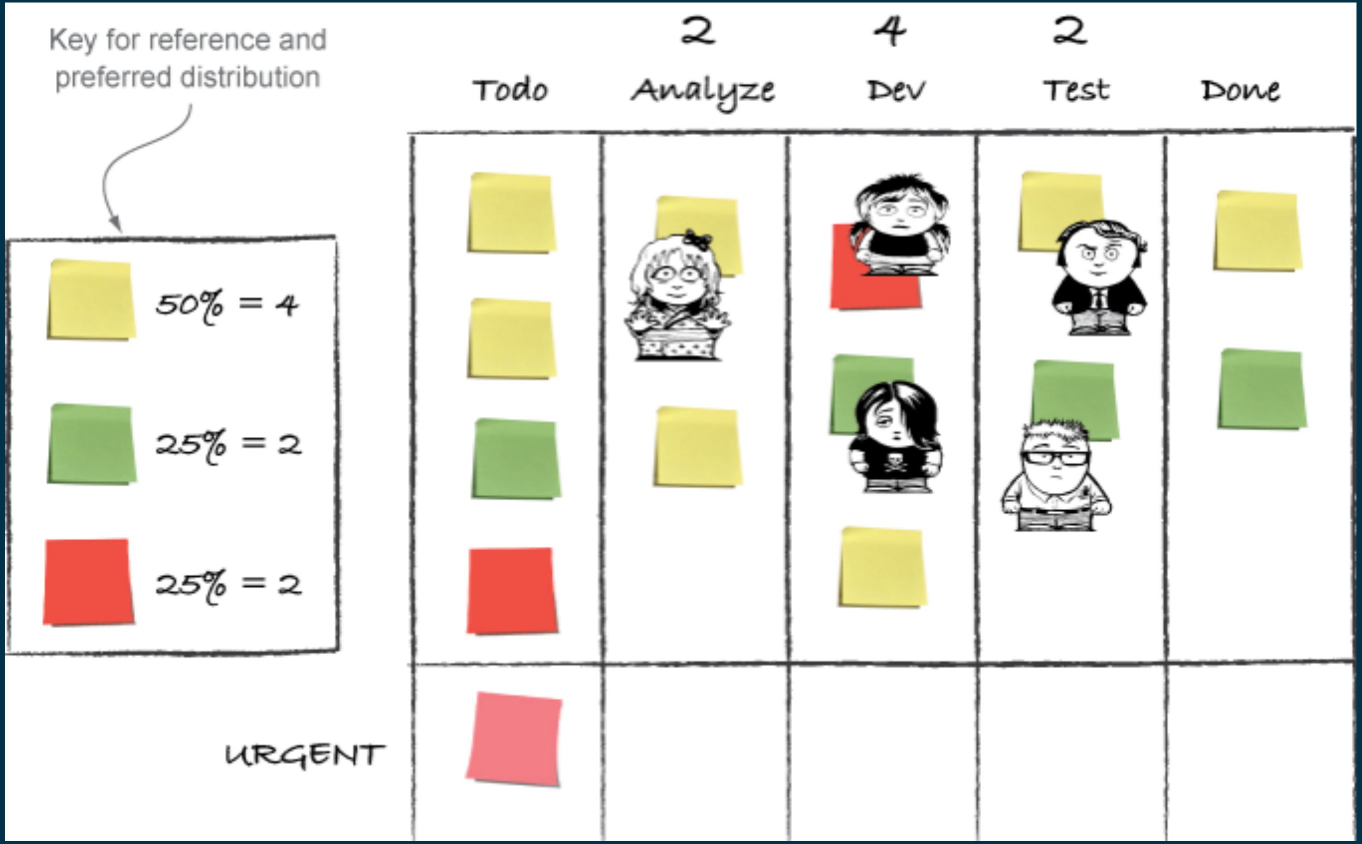
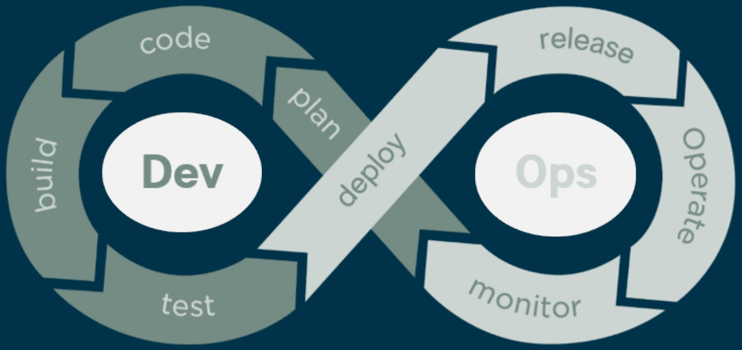


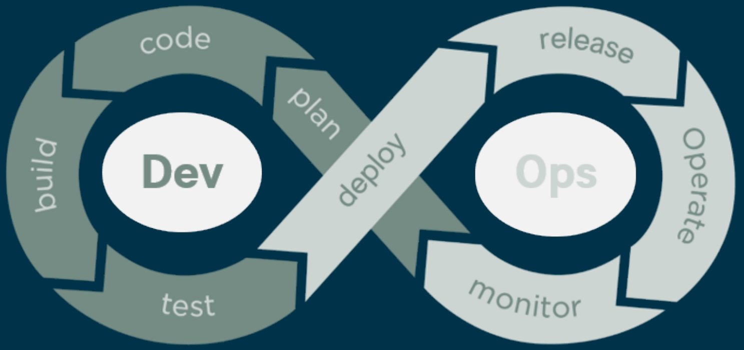
Image from the book “Kanban in Action”

Plan



- Threat Modeling (https://threatmodelingmanifesto.org)
  1. What are we working on?
  2. What can go wrong?
  3. What are we going to do about it?
  4. Did we do a good enough job?

Code

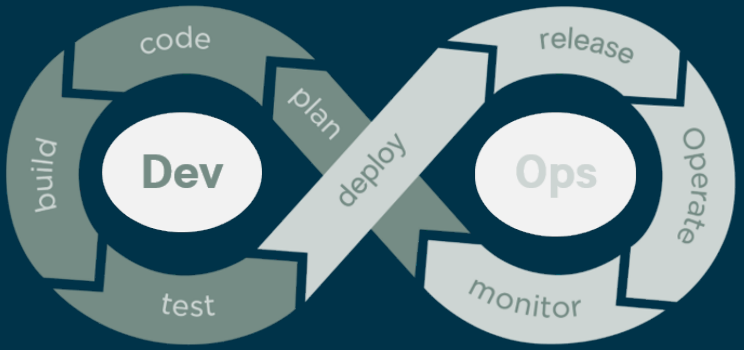


- Enable branch protection and the use of pull-requests (perform peer code review, “two pair of eyes”)

- Always 1 other reviewer!
  - From the team
  - The team decide how much friction it adds
  - Smaller updates
  - Prioritize each others PR’s
  - Pair-program (one PR creator and the other reviewer)

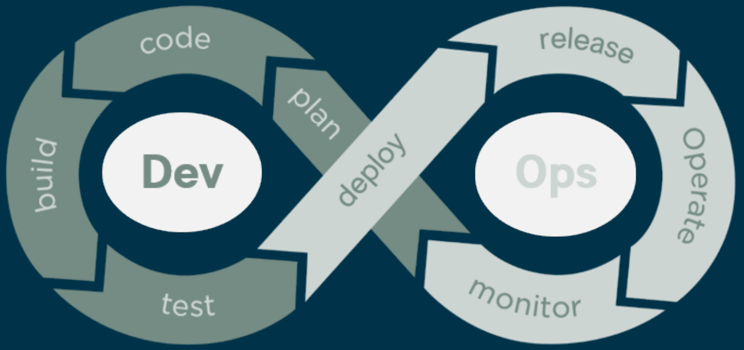
CICD-SEC-1

Code



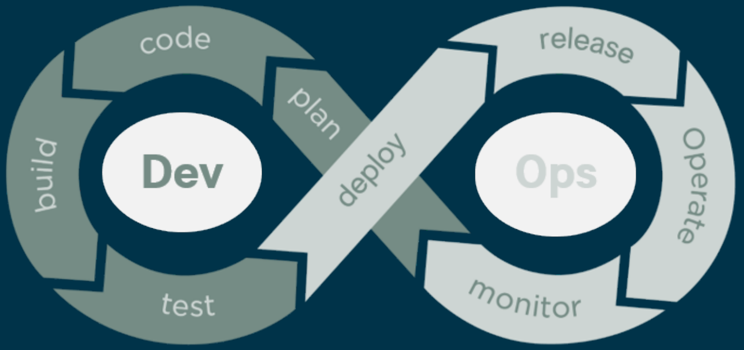
- Secrets Management, API-keys, creds for different environments (no secrets in code or config files, use vault if possible)
- Scan for secrets, enable “push protection”
- Do input validation
- Remember logging and traceability (for your own sake)

Build



- Continuous Integration (daily), trunk-based development
- Scan the code you write with a static code analysis tool (SAST) *(Semgrep, GitHub CodeQL, Coverity)*
- Scan your infrastructure as code (IaC) as well

Build

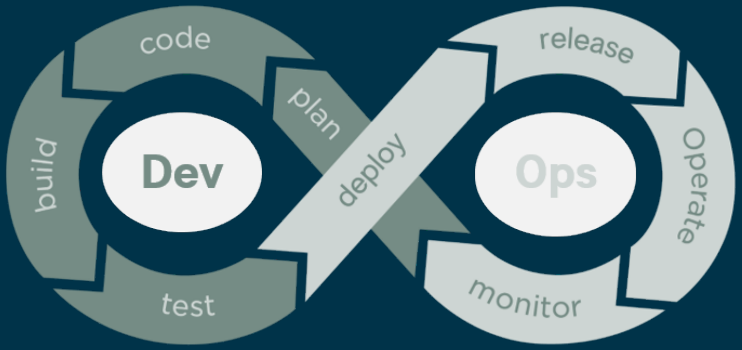


- Scan external components (Software Composition Analysis) *(OWASP Dependency Check, Semgrep, Mend, Snyk)*

- There are reports* that 78% of the code in a typical application is open-source external components

* Synopsys Open Source Security and Risk Analysis Report 2022

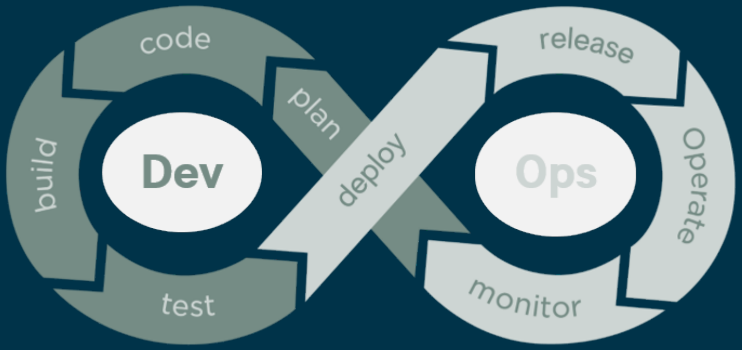
Build



Open Source strategy: “Pick components that are actively updated, highly rated with many downloads”

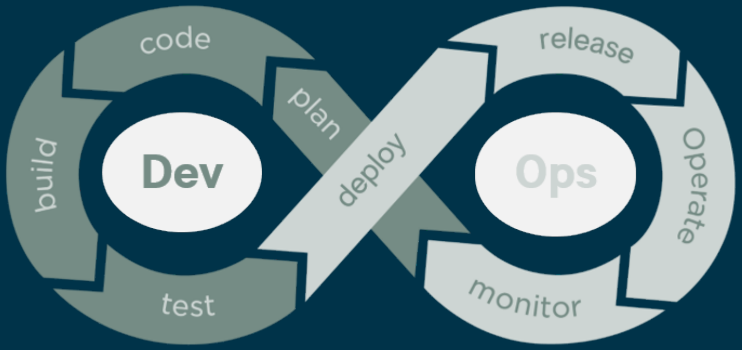
1. Use SCA tooling (daily) and patch critical and high
2. Do NOT update otherwise, wait 2-3 weeks to prevent hidden supply chain attacks (avoid auto-updates)
3. Update dependencies every month (to not fall behind)

Test



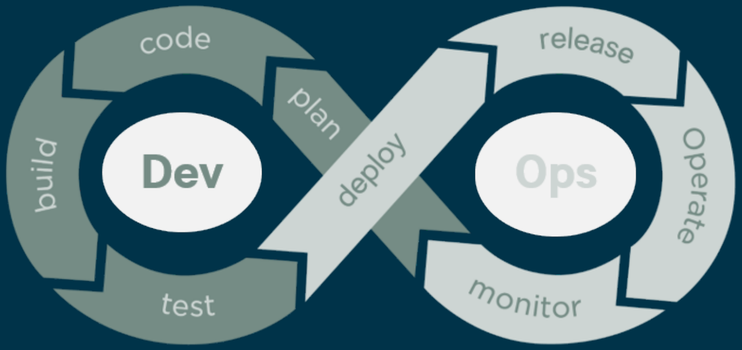
- Automate test cases (to build confidence in your releases)
- Negative tests (send unauthenticated request, expect 401 error)
- Derive test cases from Threat Modeling and penetration tests

Deploy



- Continuous Delivery – make every build releasable and maybe even deployed to the production environment.
- Use feature toggles to enable/disable new features or not yet complete features from being released to customers.
- Make sure all environments are production like (infrastructure as code, easier done in the cloud)

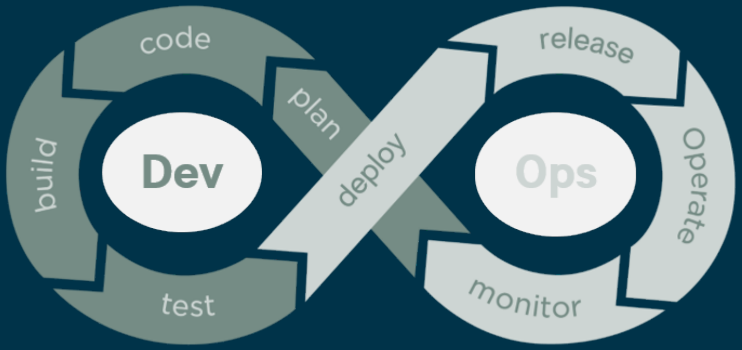
Release



- Release on demand
  Depending on business requirements (or maturity) releasing to customers might be manual or automatic.

- Release to customers without downtime and during daytime when everybody involved is available at work.

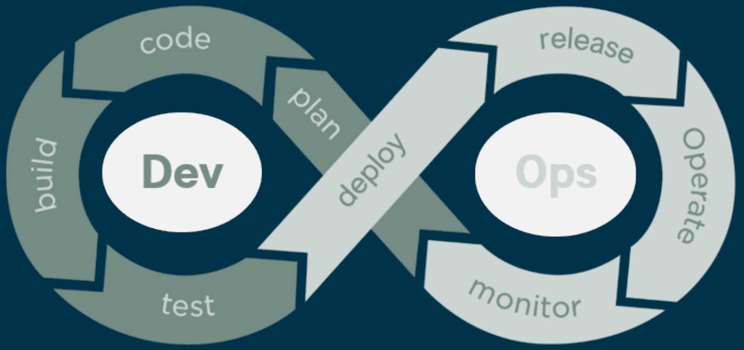
Release



- Use Blue/Green or Canary releases and/or feature toggles to release new features to customers.

- Use A/B testing to verify if new features deliver the intended value to customers.

Release

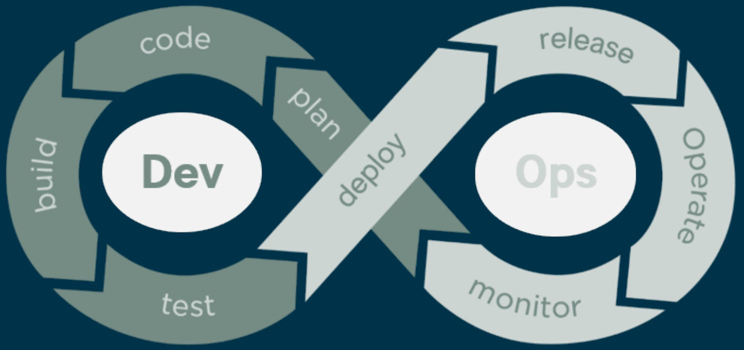


- "If it hurts, do it more often"

  Releasing often helps to reduce risk.

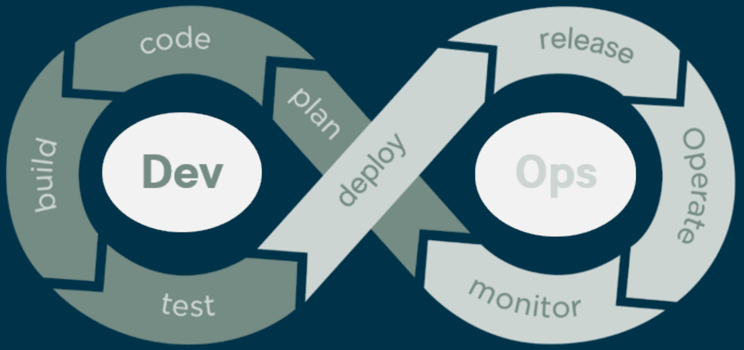
  You practice the release process which will make releasing less painful and the difference between releases gets smaller.

Operate



- Consider making servers immutable, no manual modifications (everything as code)
- No humans in production ☺
- Dare running scans in production or somebody else will!

Monitor

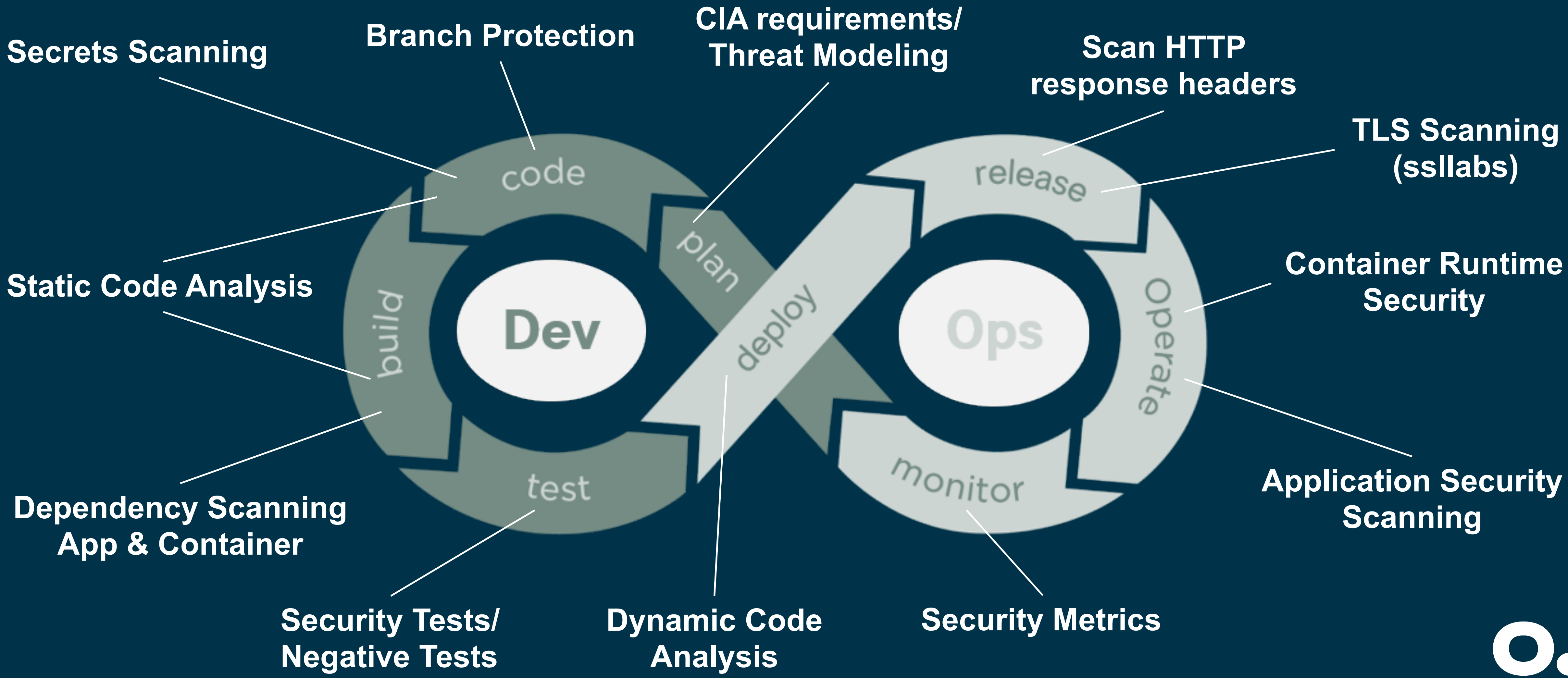


- Create metrics
  - Business value
  - Operational
  - To build confidence that the new release works as expected
  - Security related (successful/failed logon attempts etc)

- Create relevant alerts (that only fires when really needed)

- If developers also receive alerts, the number of bugs tend to decrease ☺

Security activities in DevOps


Secrets Scanning
Branch Protection
CIA requirements/ Threat Modeling
Scan HTTP response headers
TLS Scanning (ssllabs)
Static Code Analysis
Container Runtime Security
Dependency Scanning App & Container
Application Security Scanning
Security Tests/ Negative Tests
Dynamic Code Analysis
Security Metrics
code
plan
build
Dev
test
deploy
release
Ops
Operate
monitor

Securing the CI/CD environment!

- Threat Model the CI/CD pipeline environment
- Use branch protection/pull-requests CICD-SEC-1
- Monitor user permissions (code/artifacts/production)
- Use managed runners istof self-hosted runners
- Limit use of service connection to cloud account (one pipeline, on main, using PR)
- Pipeline permissions to secrets

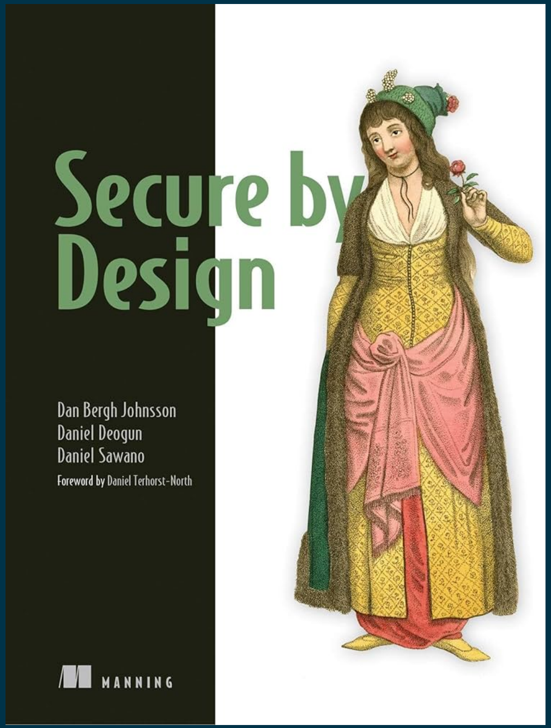
Don’t start doing everything at once, it’s a journey!

Book Tip

Secure by Design

Secure by Design teaches developers how to use design to drive security in software development.

Dan Bergh Johnsson, Daniel Deogun, Daniel Sawano
(2019)

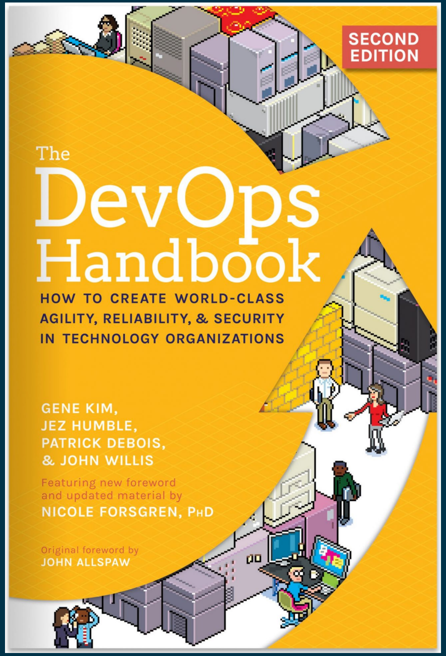


Thank You!

Questions?

mats.persson@omegapoint.se

Book Tip


SECOND EDITION
The DevOps Handbook
HOW TO CREATE WORLD-CLASS AGILITY, RELIABILITY, & SECURITY IN TECHNOLOGY ORGANIZATIONS
GENE KIM, JEZ HUMBLE, PATRICK DEBOIS, & JOHN WILLIS
Featuring new foreword and updated material by NICOLE FORSGREN, PHD
Original foreword by JOHN ALLSPAW

The DevOps Handbook

How to Create World-Class Agility, Reliability & Security in Technology Organizations

Gene Kim, Jez Humble, Patrick Debois, John Willis
(2016/2021)