aws marketplace

AWS Marketplace DevOps Workshop Series

Module 2: CI/CD Pipelines

DevOps Institute

aws marketplace

Steve Pereira

Ambassador, DevOps Institute

@steveelsewhere

devopsto

Chris Chapman

Sr. Specialist Solutions Architect at AWS

chapmanInTheCloud

aws marketplace

Steve Pereira

Ambassador, DevOps Institute

@steveelsewhere

Devopsto

About DevOps Institute

DevOps Institute's mission is to advance the human elements of DevOps by creating a safe and interactive environment where our members can network, gain knowledge, grow their careers, support enterprise transformation and celebrate professional achievements.

We connect and enable the global DevOps community to drive change in the digital age.

Become a professional member at www.devopsinstitute.com

Steve Pereira

Marie Kondo of Automation

@steveelsewhere

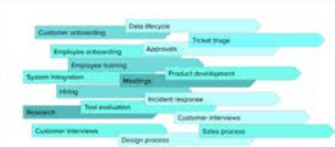
MISSION: Clarity, Value, Flow.

Helping Teams Define and Optimize their Value Streams

Steve is obsessed with making tech human and leveraging it to deliver continuous value.

For the past 20 years, his focus has been on using mapping techniques to guide ambitious and struggling teams towards their true north.

He's a former startup CTO, agency consultant, systems and release engineer, finance IT manager, tech support phone jockey, and pizza maker. All focused on the flow of value, all the time.



WHAT IS A VALUE STREAM?

It seems everyone in the Agile, DevOps and Tech scene are talking about Value Streams in 2020, and for good reasons. Business and manufacturing folks have focused on value streams for decades...

READ MORE

WHAT IS A VALUE STREAM MAP?

I've always been a visual person, and used flow charts, swim lane diagrams and every other technique to analyze and share information. It's easy to look at a typical diagram and dismiss it as overly simplistic, idealistic or inaccurate...

READ MORE

HOW IS A VALUE STREAM MAP DIFFERENT?

As DevOps, systems thinking and continuous improvement become more common knowledge, more and more people are looking beyond those foundations to how they can be best applied...

READ MORE

VALUE STREAM MAPPING 101

This talk is the Value Stream Mapping 101 you never knew you needed for Software Development. I'll share the who, what, where, why, when and especially how of VSM.

WATCH

NAVIGATE WITH CONFIDENCE

Working in software can seem like wandering in the wilderness.
How do you know you're heading in the right direction?

WATCH

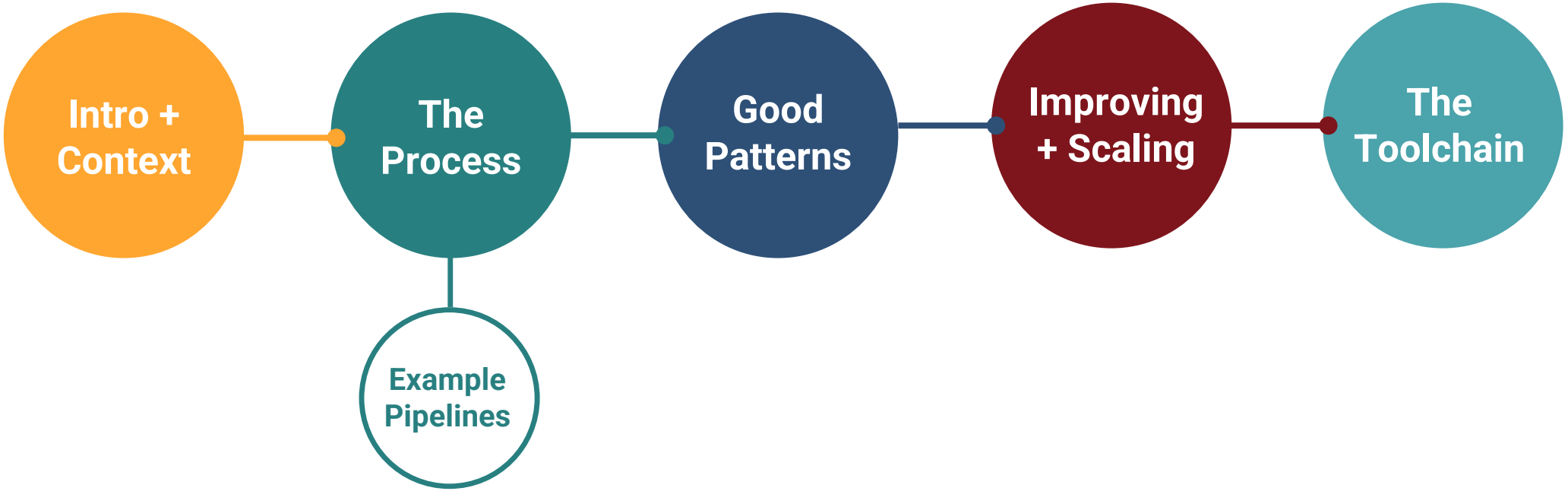
5 KEYS OF ASSESSMENT

With 70% of transformation efforts failing it pays to understand your surroundings, strengths, and gaps before you invest in what's next.

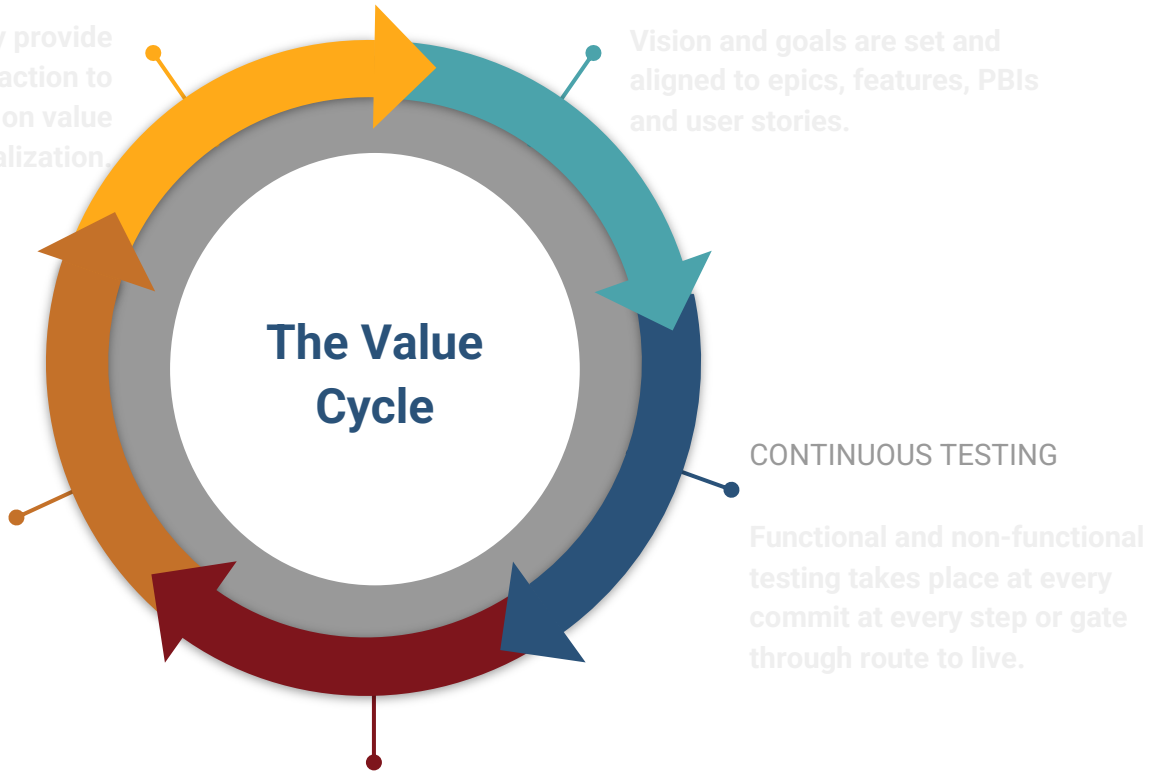
WATCH

PROJECTS

Flow: Talk Map

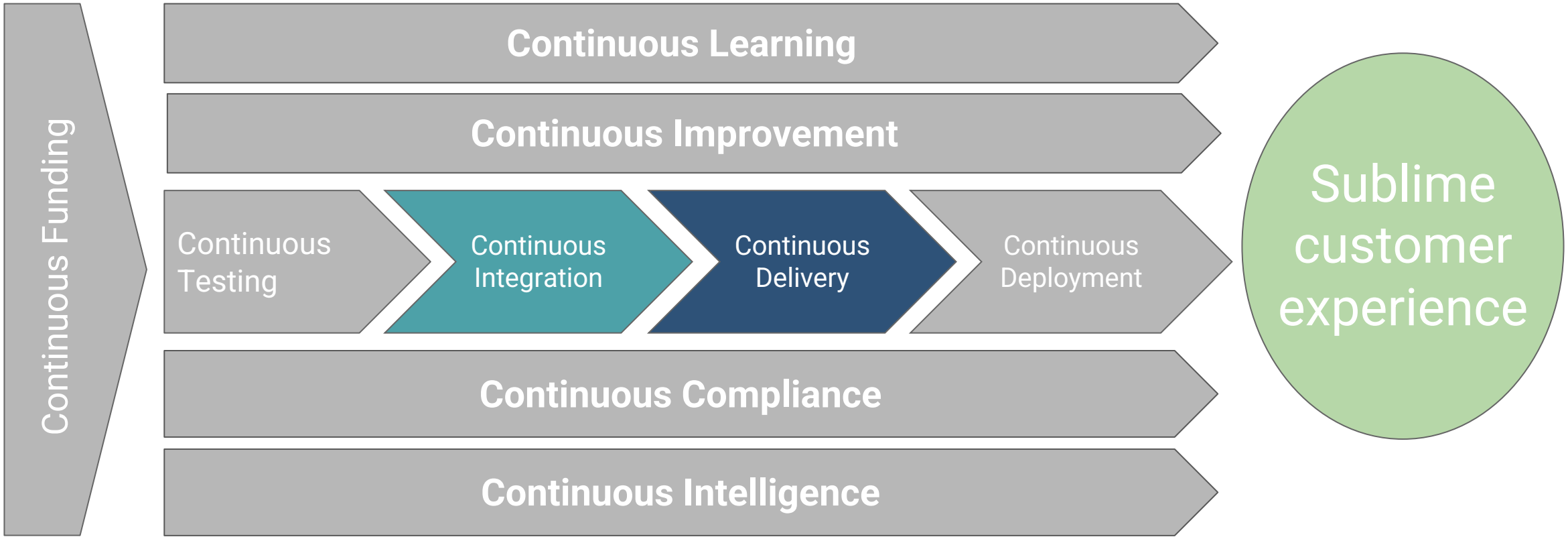


The Value Cycle



DevOps Practices

All the continuouses



#1 and 2 Must-Have Automation Skills

CI + CD are leading valuable skills

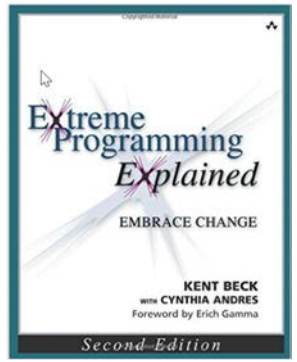
2020 survey data N=920

Continuous Integration Defined

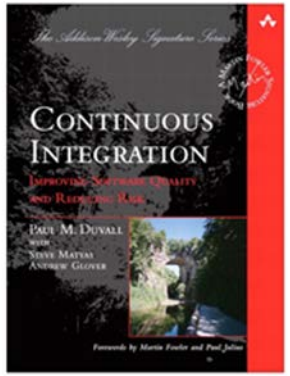
Continuous Integration is a software development practice where members of a team integrate their work frequently.

In most instances, each person of the team integrates their code at least daily - leading to multiple integrations per day.

Each integration is verified by an automated build and test in order to detect integration errors as quickly as possible.



Kent Beck
1999



Paul Duvall

Continuous Integration

You can do this in waterfall too… if you want to

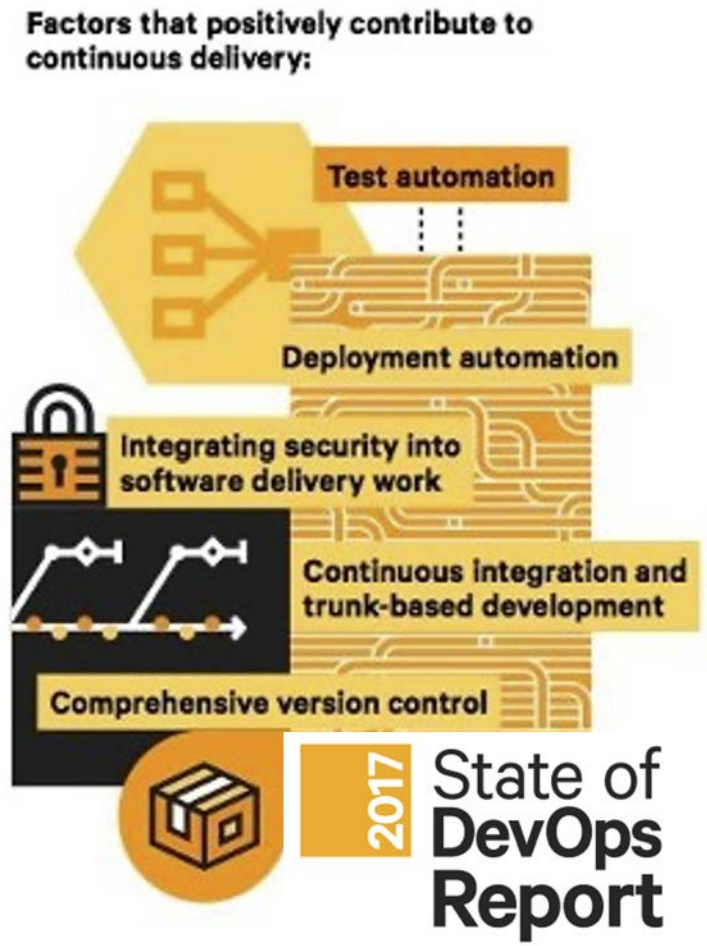
- All developers check code in at least daily to trunk
  - Trunk based development
- Each check-in is validated by
  - An automated build
  - Automated unit, integration and acceptance tests
- Is dependent on consistent coding standards
- Requires version control repositories and CI servers to collect, build and test committed code together
- Runs on production-like environments
- Allows for early detection and quick remediation of errors from code changes before moving to production

Avoid ‘merge hell’

Continuous Delivery

Software is always in a releasable state - ready to go, at the push of a button

- Takes continuous integration to the next level
- Provides fast, automated feedback on a system's production-readiness
- Prioritizes keeping software releasable/deployable over working on new features
- Relies on a deployment pipeline that enables push-button deployments on demand
- Reduces the cost, time, and risk of delivering incremental changes



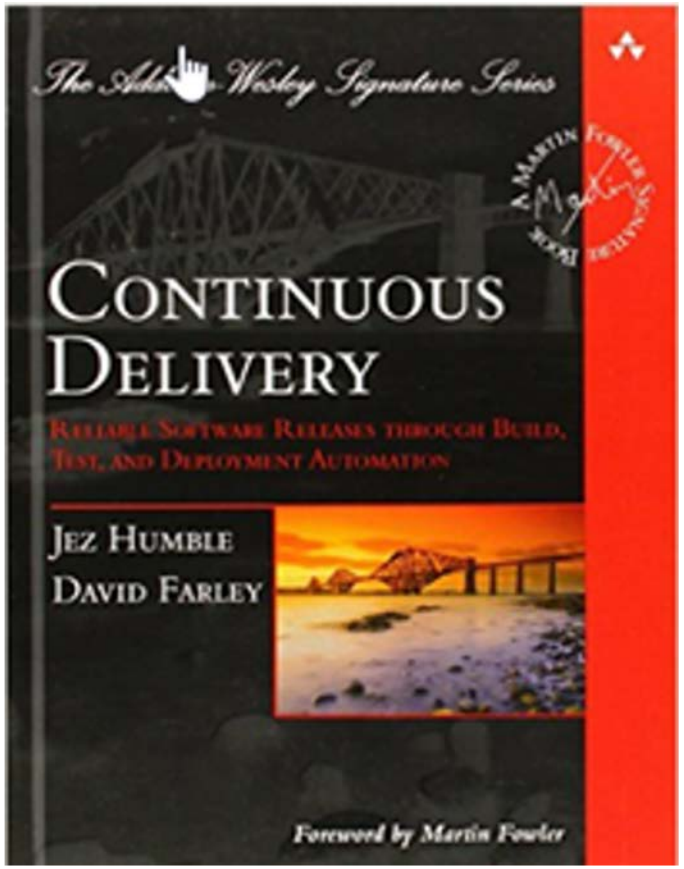
Continuous Delivery

Continuous delivery (**CD**) is a software engineering approach [associated with DevOps,] in which **teams produce software in short cycles**, ensuring that the software can be **reliably released at any time**. It aims at building, testing, and releasing software **faster and more frequently**.

The approach helps **reduce the cost, time, and risk of delivering changes** by allowing for more **incremental updates to applications in production**. A **straightforward and repeatable deployment process** is important for continuous delivery.

https://en.wikipedia.org/wiki/Continuous_delivery

Or in Other Words…

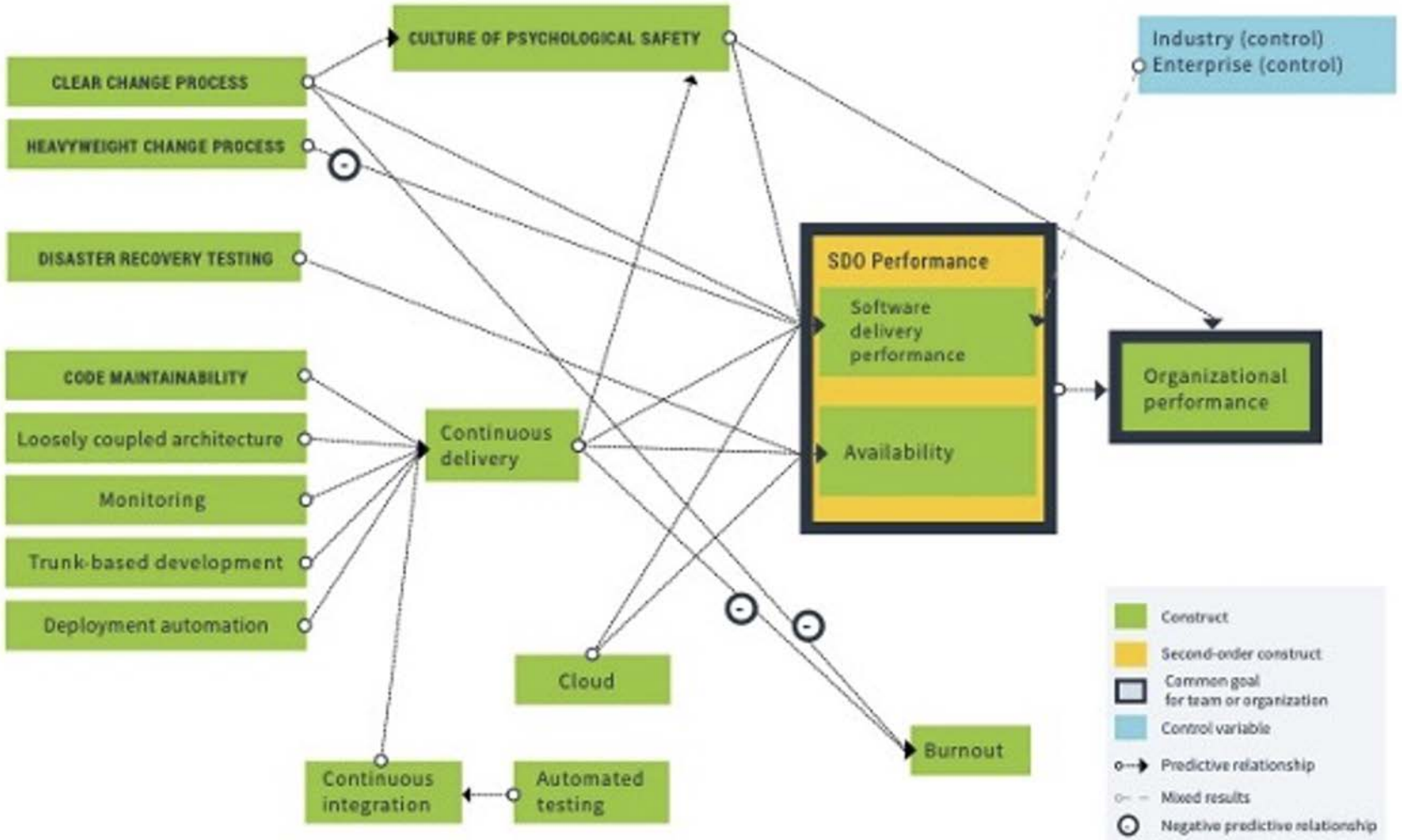


"The ability to get changes - features, configuration changes, bug fixes, experiments - into production or into the hands of users safely and quickly in a sustainable way "

Jez Humble
Author of "Continuous Delivery"
co-author of The DevOps handbook"

Continuous Delivery

Leads to higher organizational performance



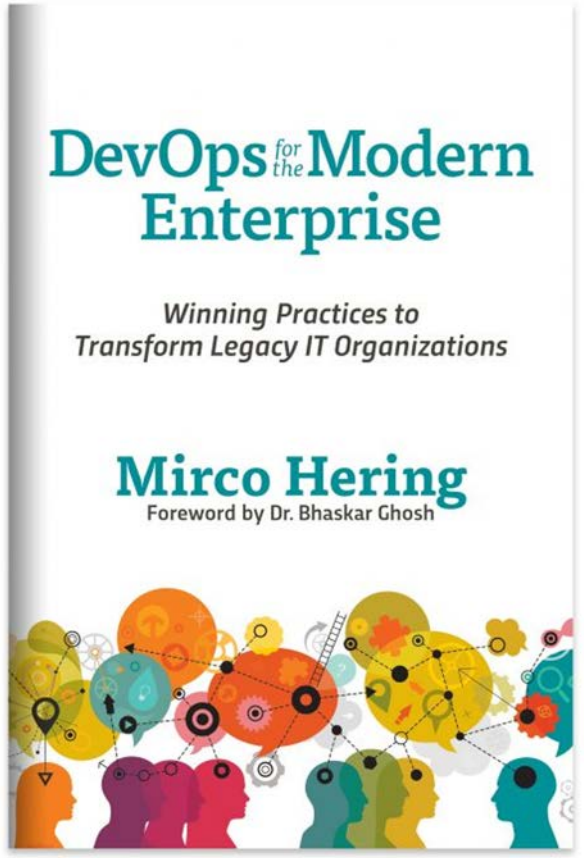
Continuous Delivery & Continuous Deployment

Automatic trigger

Manual trigger

Continuous Delivery

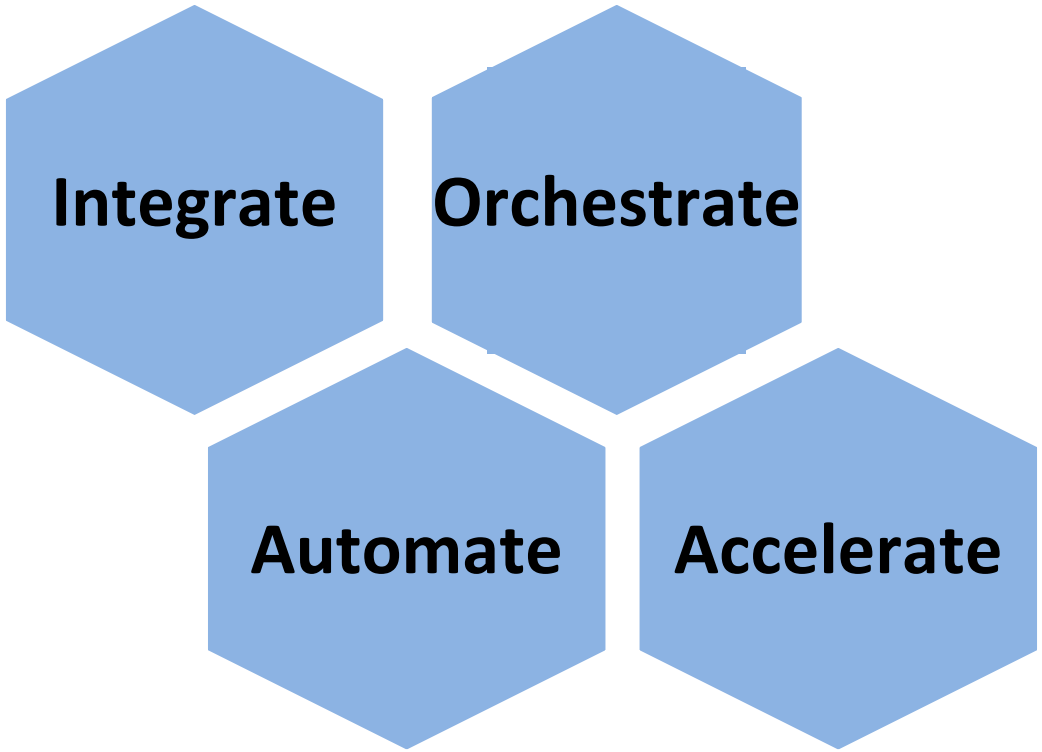
Continuous Deployment



From: Mirco Hering: notafactoryanymore.com, author of 'DevOps for the Modern Enterprise'

Key Ingredients that Differentiate Continuous Delivery

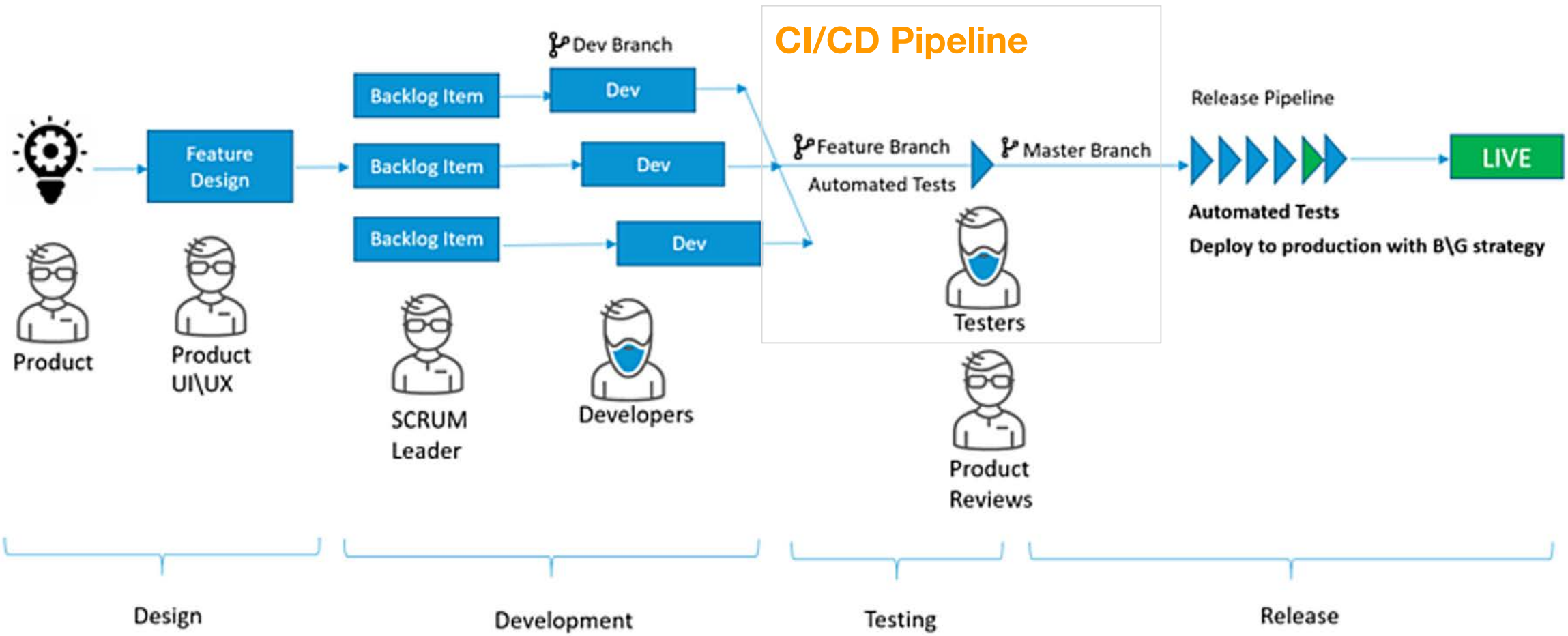
- CD uses an integrated infrastructure
- CD emphasizes orchestration of the environment
- CD tasks are automated as much as possible
- CD goal is to accelerate activities as early in the pipeline as possible



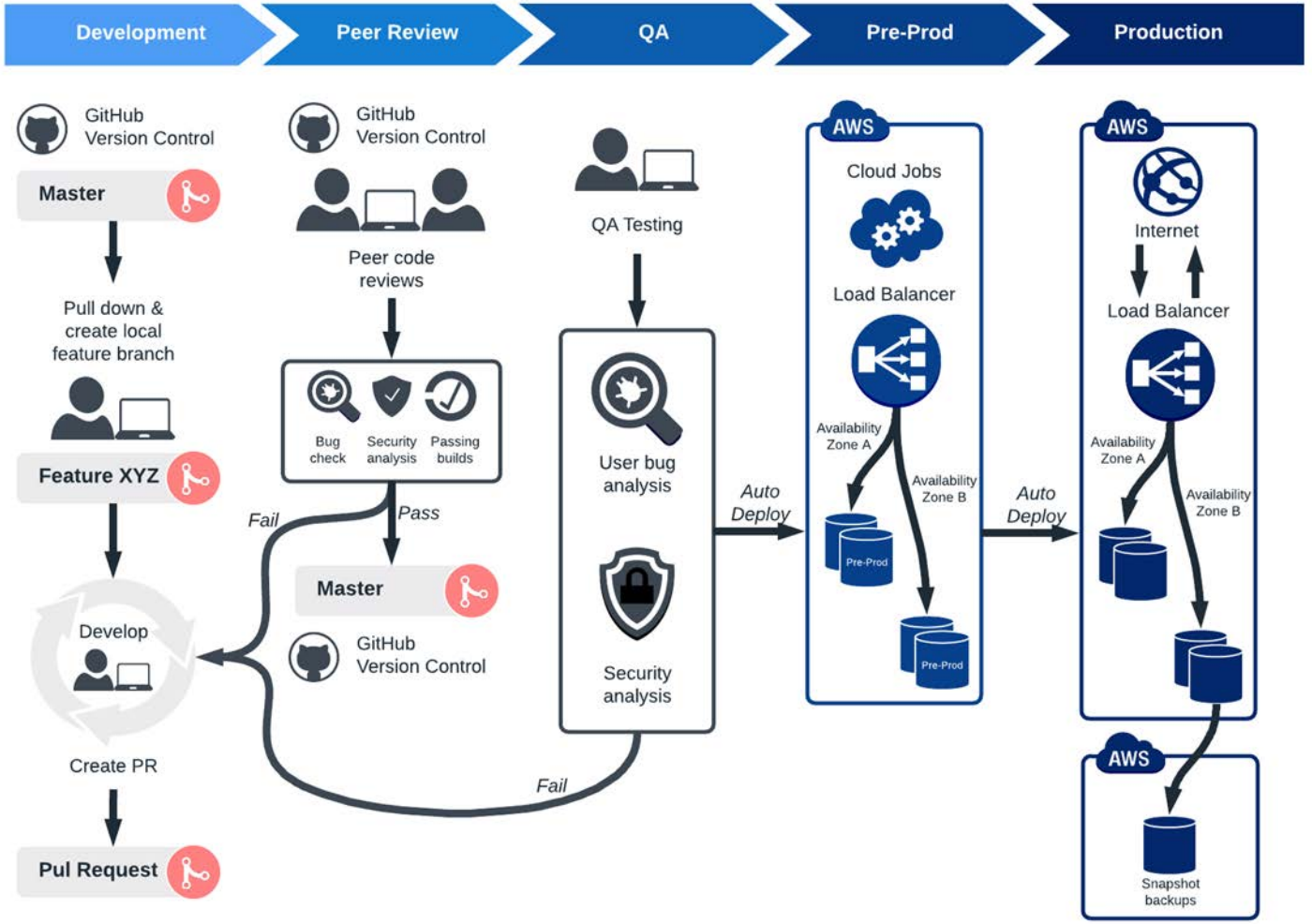
Consequences of NOT Doing Continuous Delivery Properly

- ☒ Application quality issues
- ☒ Complex merge issues
- ☒ Security events
- ☒ Pipeline failures
- ☒ Interruptive reverts
- ☒ Process delays
- ☒ Schedule delays
- ☒ Cost overruns
- ☒ Poor morale / unhappiness
- ☒ Audit failures

Sample Process

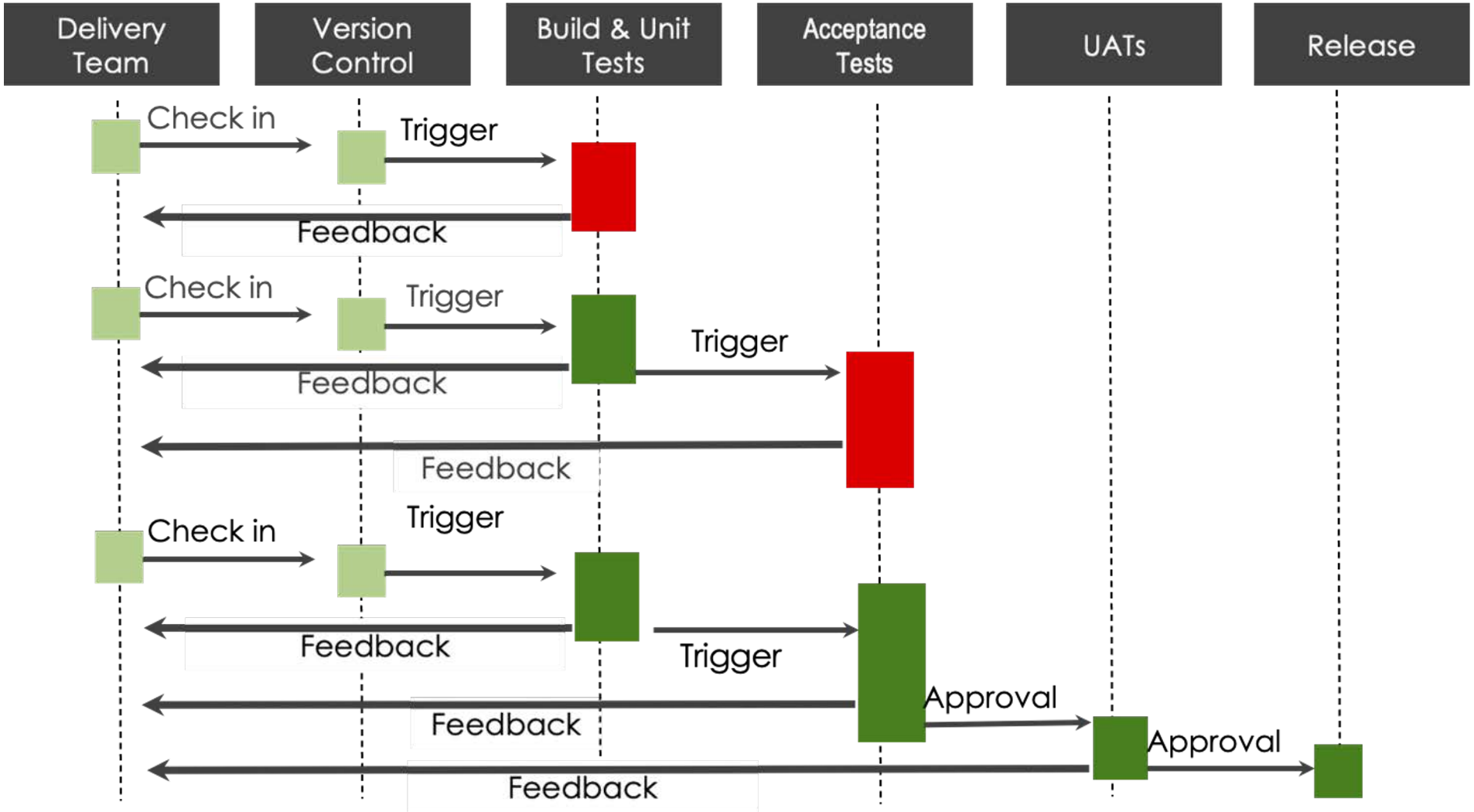


Pipeline Workflow

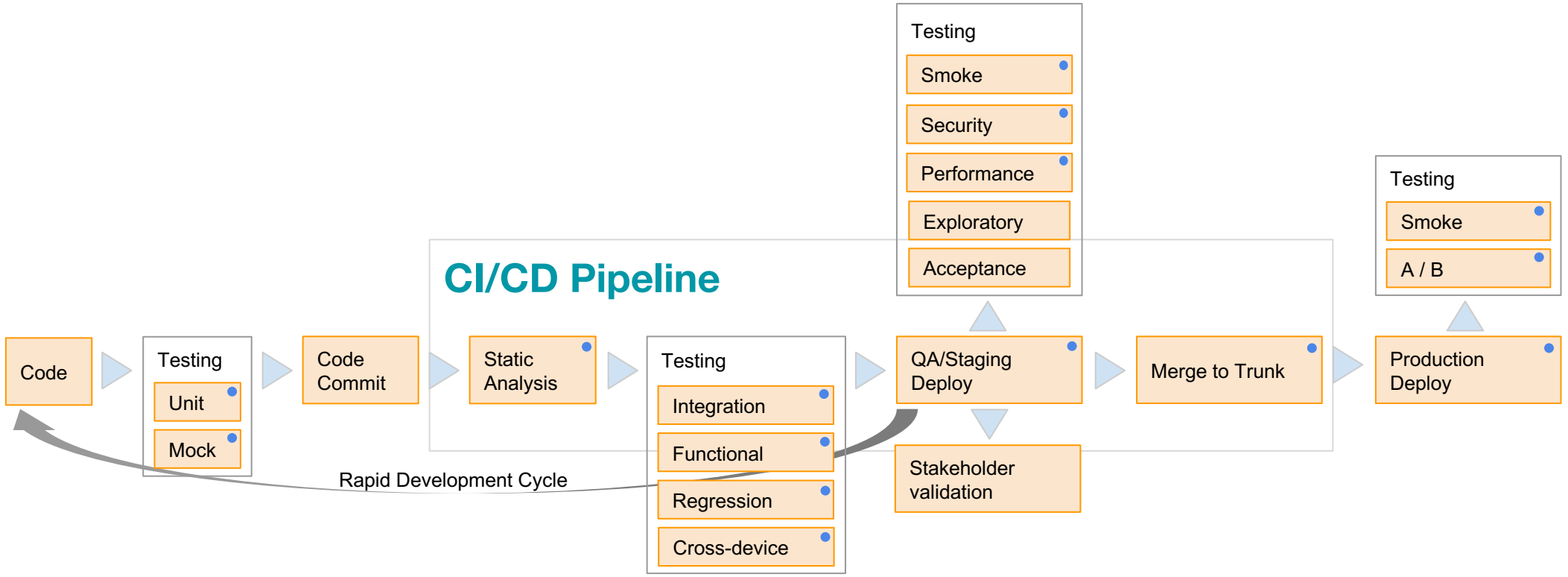


Lucidchart

The Delivery Pipeline



Sample Pipeline



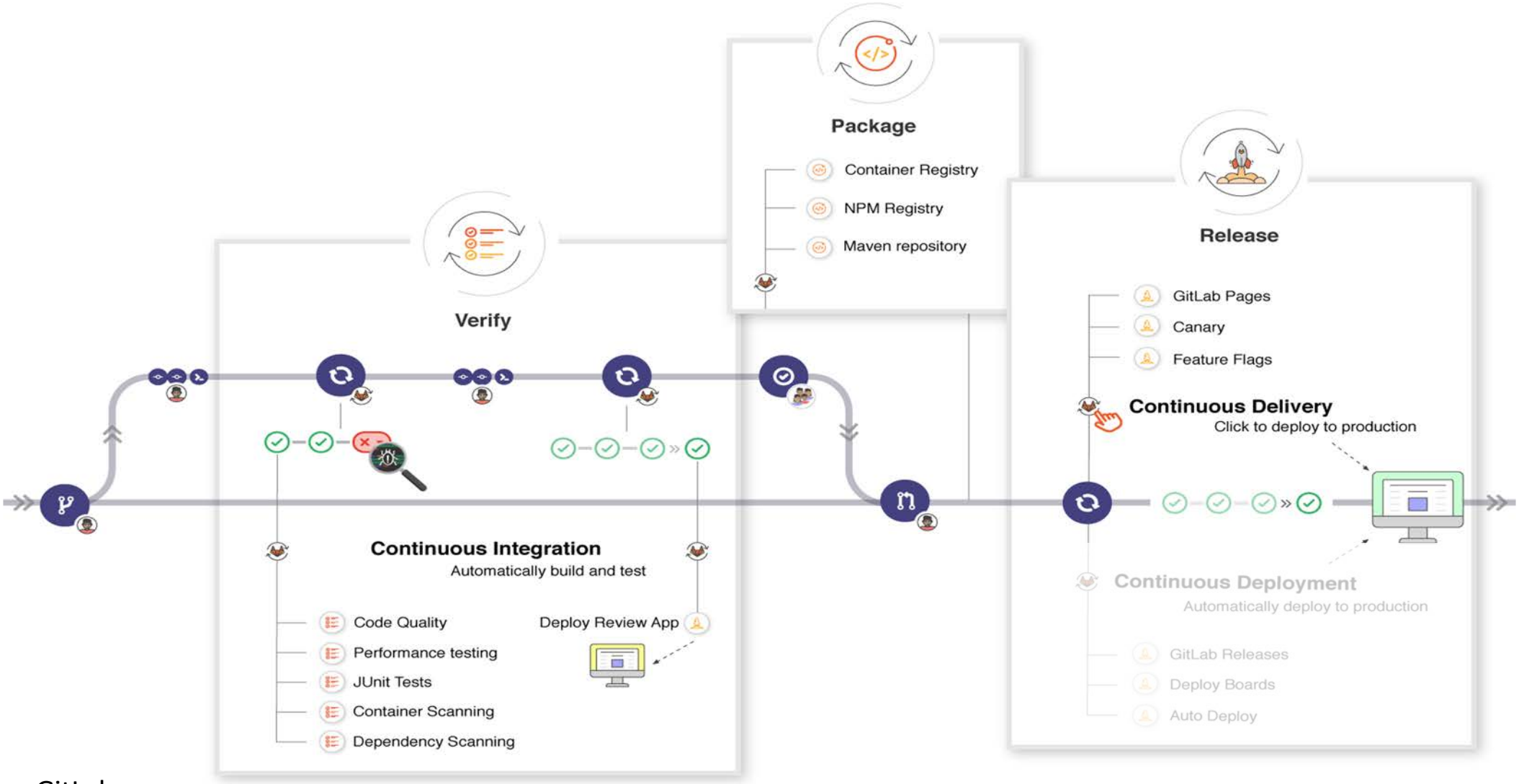
● Automation available

Increased confidence / Decreased risk

Measurement / Data collection / Feedback

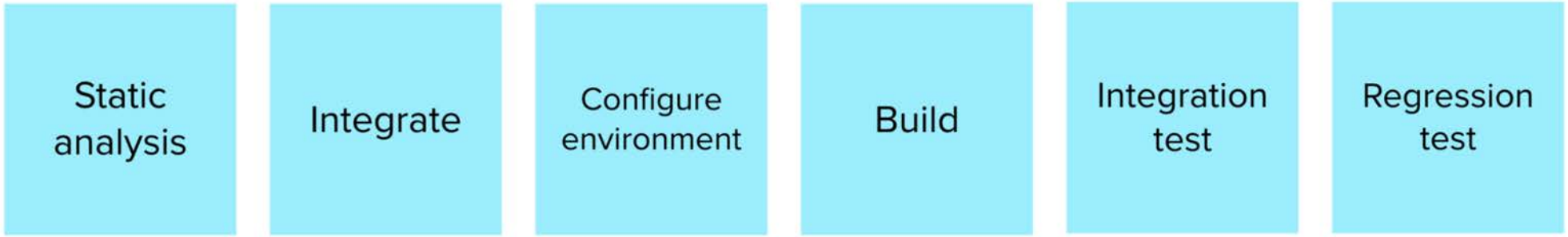
Visible

Pipeline Examples

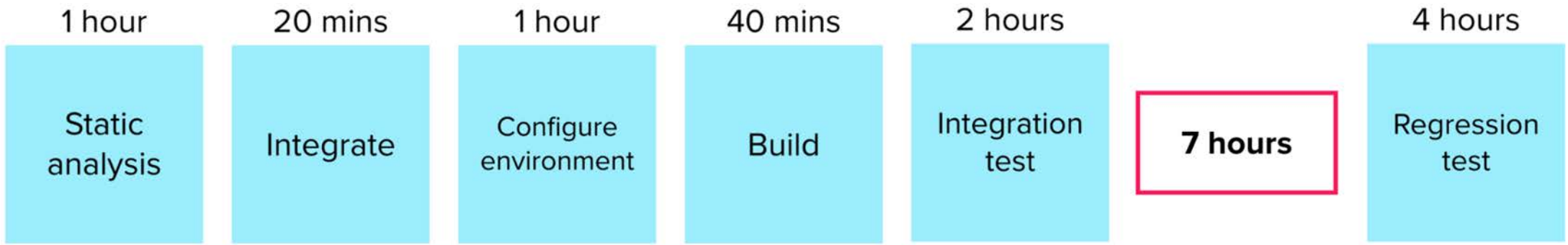


GitLab

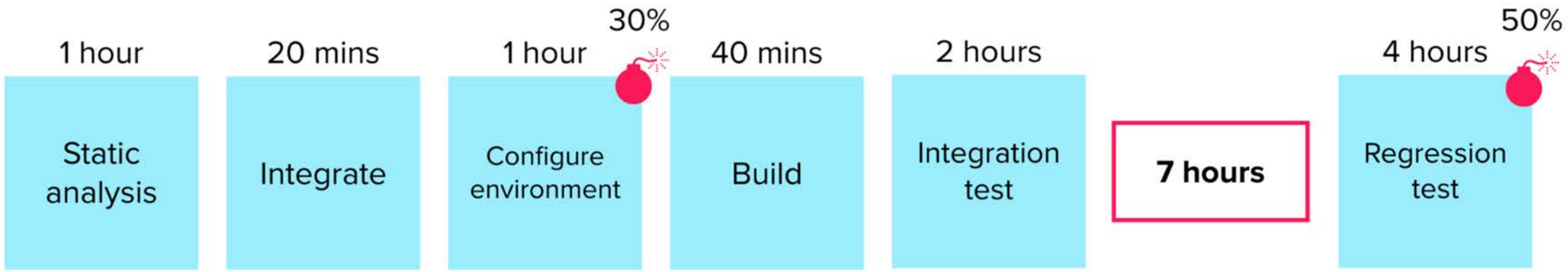
Pipeline Creation and Improvement



Pipeline Creation and Improvement



Pipeline Creation and Improvement



Pipeline Creation and Improvement

	Measure 1	Measure 2	Measure 3	Measure 4	Measure 5	Measure 6	Measure 7	Measure 8	Measure 9	Measure 10	Measure 11	Measure 12	Measure 13	
Product 1	100%	100%	0%	0%	0%	0%	0%	100%	100%	100%	0%	0%	0%	38%
Product 2	0%	0%	100%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	15%
Product 3	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
Product 4	100%	100%	100%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	31%
Product 5	100%	100%	100%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	31%
Product 6	100%	100%	100%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	31%
Product 7	100%	100%	0%	100%	0%	100%	100%	100%	100%	100%	0%	0%	100%	69%
Product 8	100%	100%	0%	0%	0%	100%	100%	0%	100%	0%	0%	0%	0%	38%
Product 9	0%	0%	0%	100%	0%	0%	0%	100%	0%	0%	0%	0%	100%	23%
Product 10	100%	100%	100%	100%	100%	100%	0%	100%	100%	0%	0%	0%	100%	69%
Product 11	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	8%
Product 12	100%	100%	0%	100%	0%	100%	100%	100%	100%	0%	0%	100%	100%	69%
Product 13	73%	100%	18%	73%	18%	64%	0%	18%	18%	18%	0%	0%	0%	31%
Product 14	82%	86%	0%	79%	61%	79%	0%	0%	18%	18%	0%	0%	0%	32%
Product 15	83%	83%	78%	78%	70%	76%	48%	97%	46%	0%	0%	56%	15%	56%
Product 16	100%	100%	0%	100%	100%	100%	100%	100%	100%	0%	0%	0%	0%	62%
Product 17	100%	100%	100%	100%	100%	100%	100%	100%	100%	0%	0%	100%	0%	77%
Product 18	100%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	15%
Product 19	100%	100%	100%	100%	100%	100%	50%	0%	100%	100%	0%	0%	0%	65%
Product 20	100%	100%	0%	100%	0%	100%	0%	0%	0%	0%	0%	0%	0%	31%
Product 21	100%	100%	0%	100%	100%	91%	84%	100%	94%	94%	0%	100%	100%	82%
Product 22	38%	38%	100%	25%	0%	0%	0%	38%	25%	25%	0%	0%	0%	22%
Product 23	62%	59%	100%	56%	56%	67%	33%	100%	59%	59%	0%	59%	59%	59%

Centralized Continuous Integration

Centralized CI is efficient for both developers and administration:

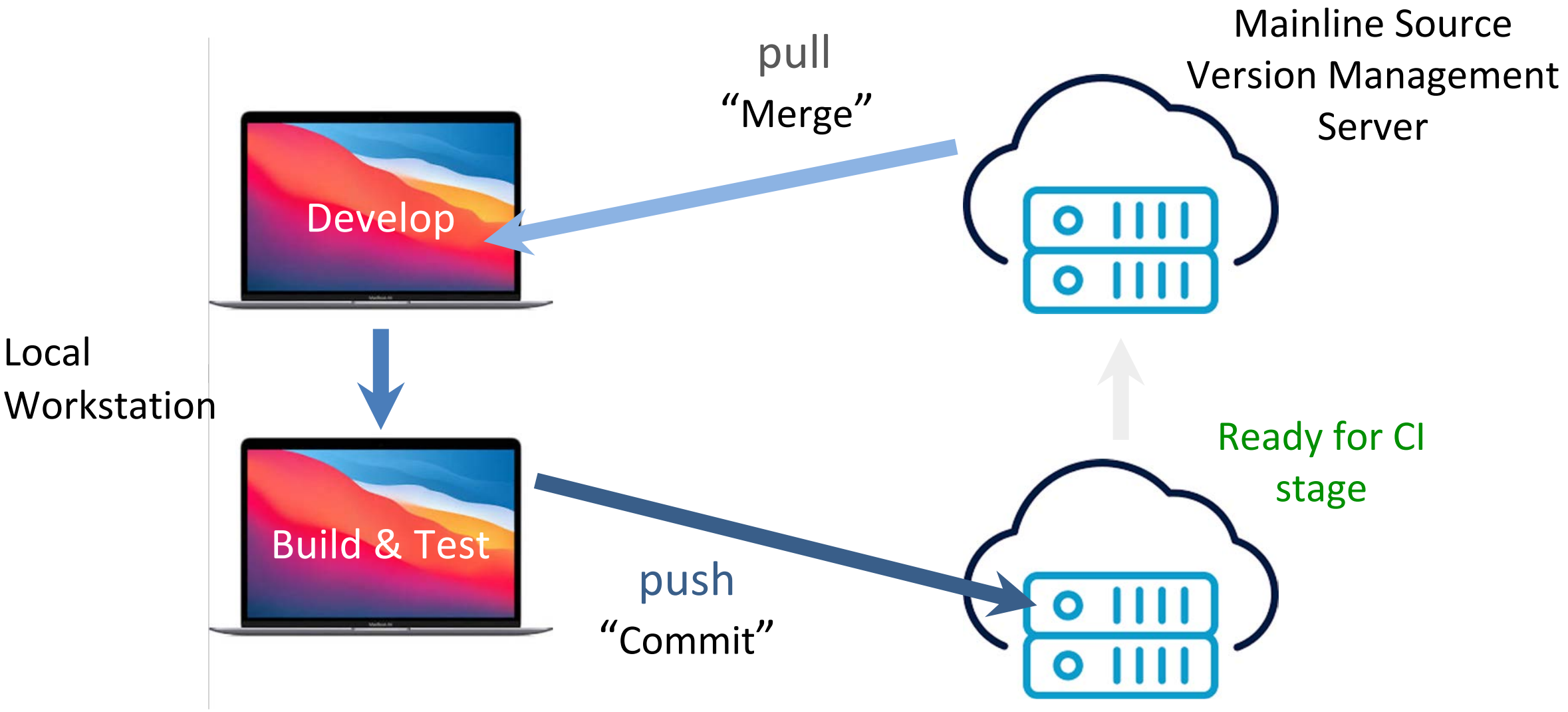
Efficiencies: Common system admin, security, backup, cloud management

Caution: Distributed development teams must have efficient networked access and self-service capabilities for

- CI environments
- Configurations
- Builds
- Test

In short: It should be a highway, not a bottleneck

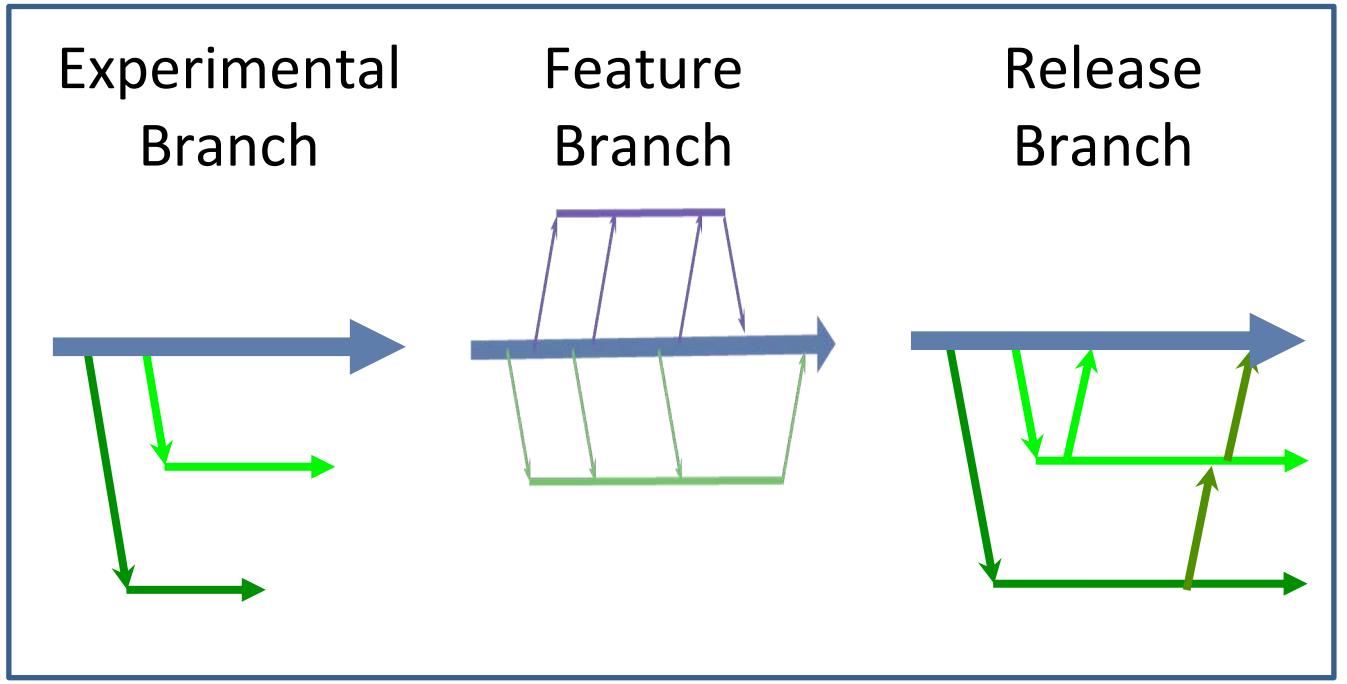
Everyone Commits to Mainline Every Day



Branching and Merging Best Practices

One mainline “trunk”

- Code is continuously integrated
- Tests run on integrated codebase
- Developers see others changes immediately
- Avoids “merge hell” at the end
- Instant feedback about application status

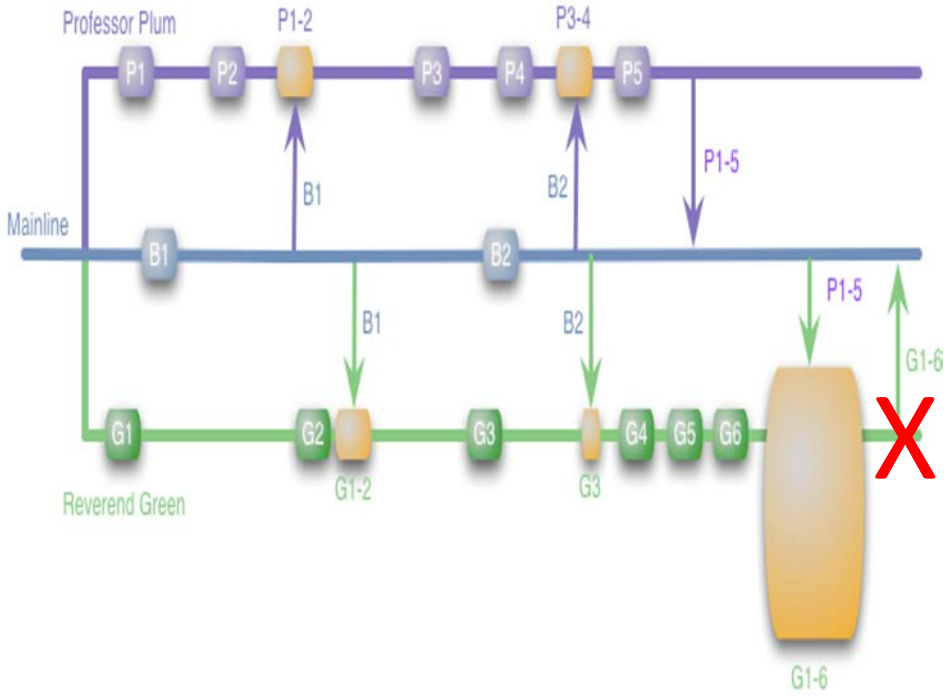


Source code, images and test versions in same branch structure ensures consistency

http://eugenedvorkin.com/continuous-integration-strategies-for-branching-and-merging/

"No Feature Branches" Best Practice

- No separate feature branches
- Merge when code is pulled from mainline
- Branch in-code (within mainline) using feature toggles (i.e., config flags)
- Eliminates merging issues and keeps everyone accountable
- Simple and easy to understand



“Pre-Flight Testing" Best Practice

Before committing changes to mainline:

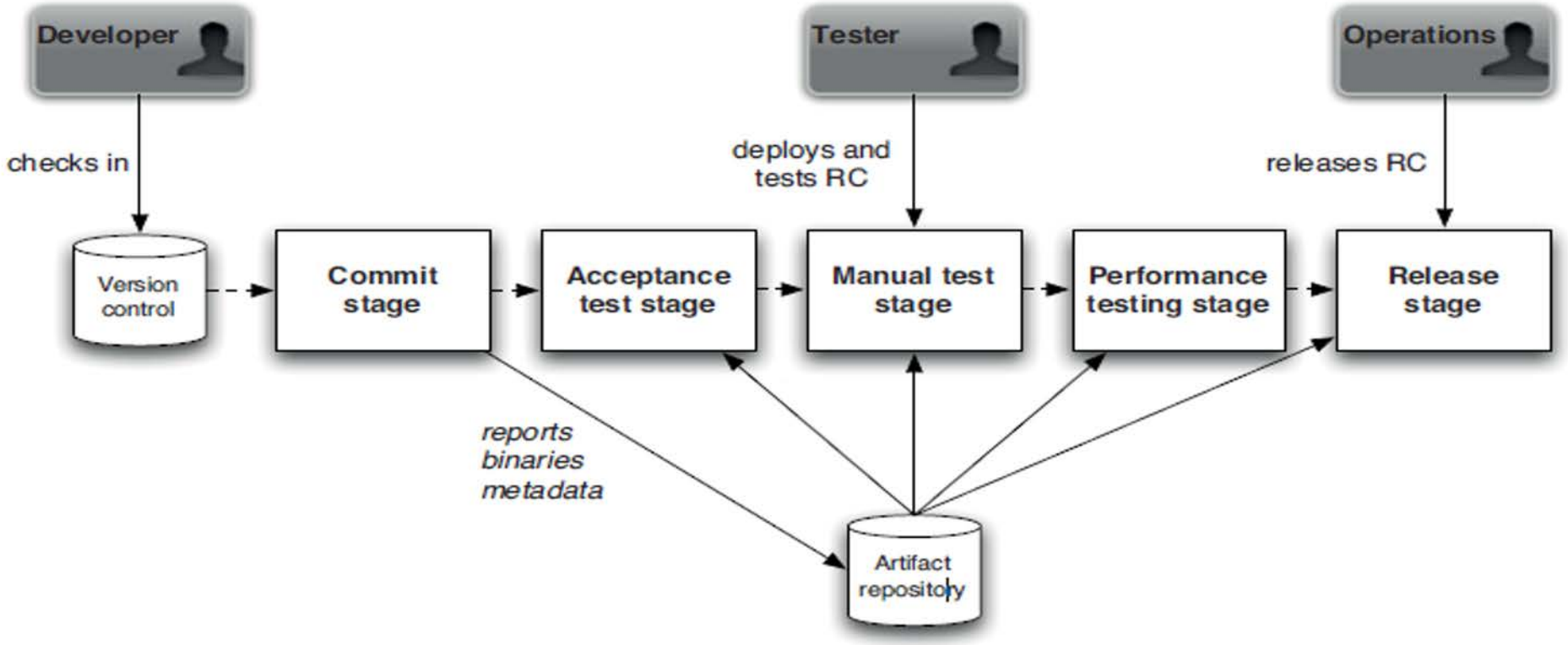
- Run static analysis
- Peer review source code
  (E.g., Gitlab, Crucible, Collaborator, etc.)
- Run unit tests
- Run functional tests
- Pre-Flight tests, are performed in a test environment that is equivalent to the production environment of customer deployments.

Value of Continuous Integration

Frequent, small, incremental integrations have many merits

- Quick feedback on frequent integration problems enables a release consisting of many individual changes to be built incrementally with confidence
- The root cause of integration problems can be isolated much faster when the changes are integrated incrementally

The Role of the Artifact Repository

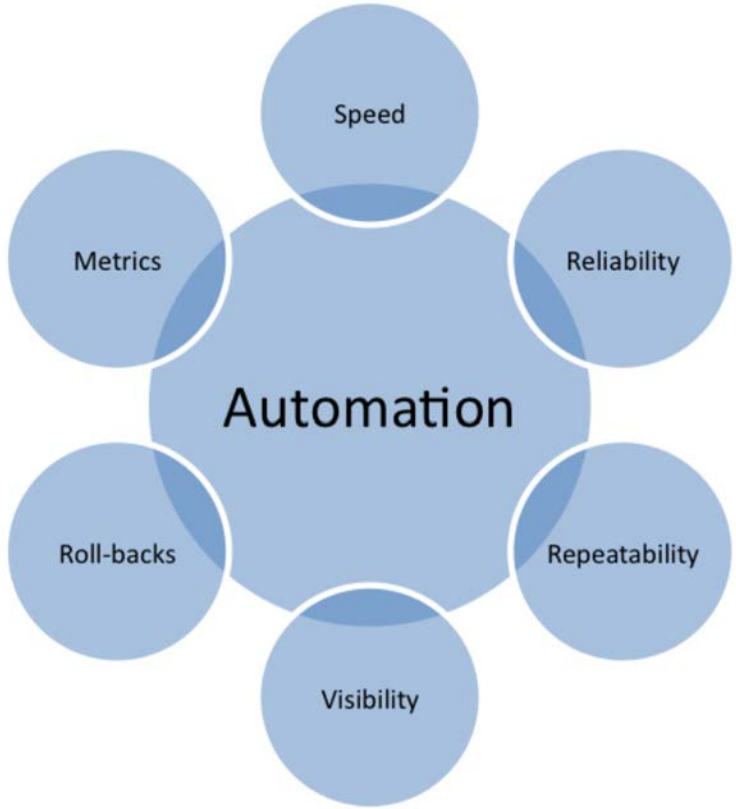


Use an artifact repository to store binaries, reports, and metadata for each of your release candidates (E.g., Archiva, Nexus, JFrog).

CI Prerequisite - Automated Build

A person or computer can run the build, test, and deployment processes in an automated fashion via:

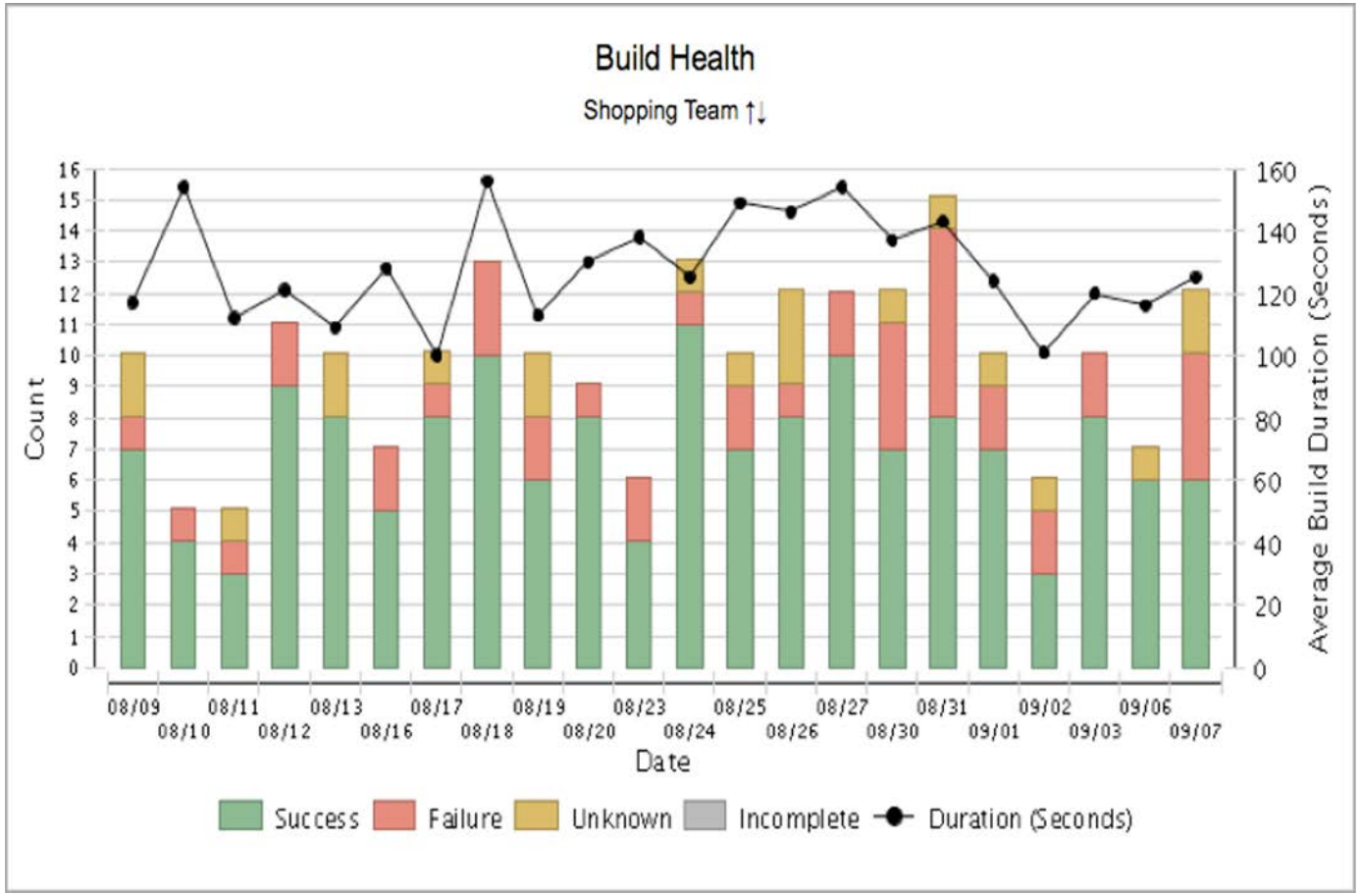
- Command-line (CLI) program starts the build and then runs tests
- May be a collection of multistage build scripts that call one another



Reasons to Fail a Build

The CI build process should have explicit checks for the following:

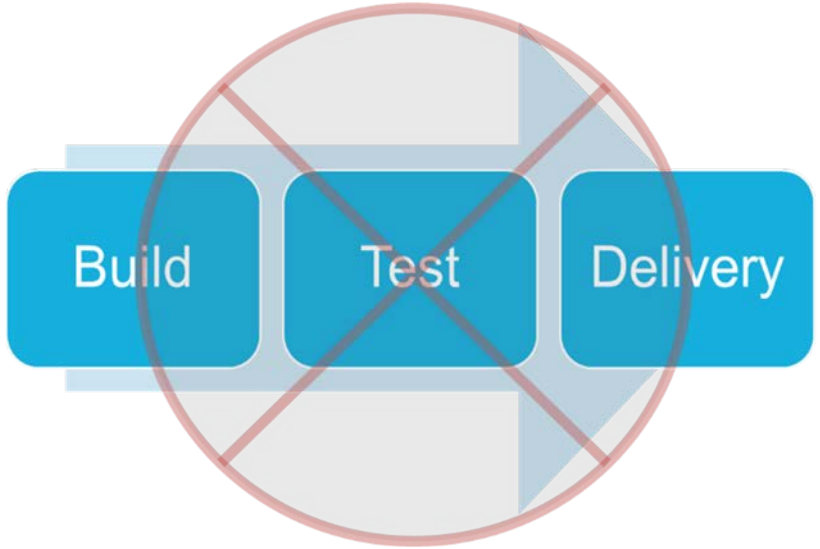
- Compilation failures
- Excessive warnings and code style breaches
- Unit test failures
- Functional test failures
- Architectural breaches
- Slow tests



The Testing Myth

Myth
Testing is a phase between integration and delivery.

Truth
Testing is not a phase!



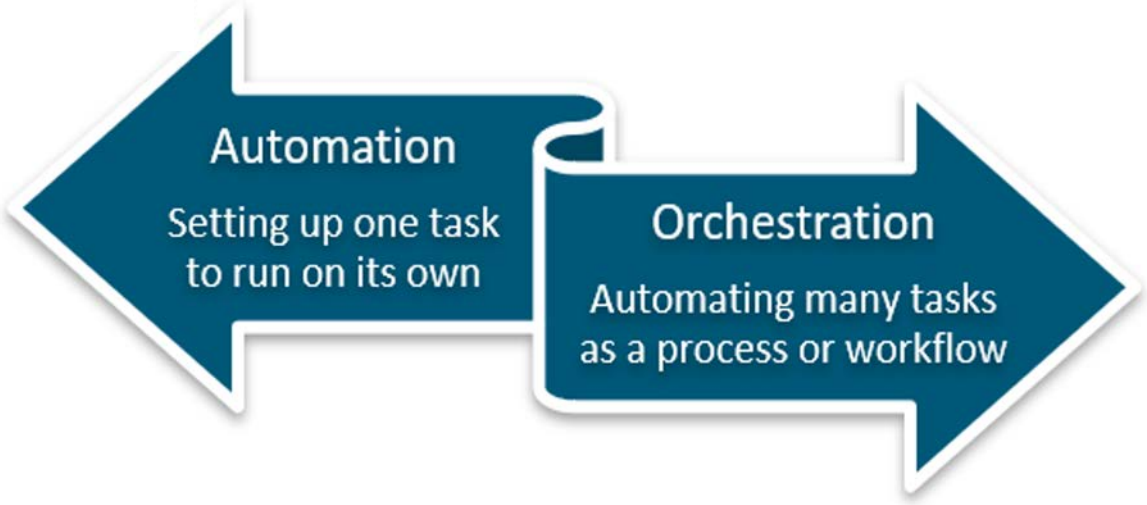
Testing is implemented as part of all pipeline stages, end-to-end, in accordance with a “continuous” testing strategy.

Test Environment Orchestration and Test Automation

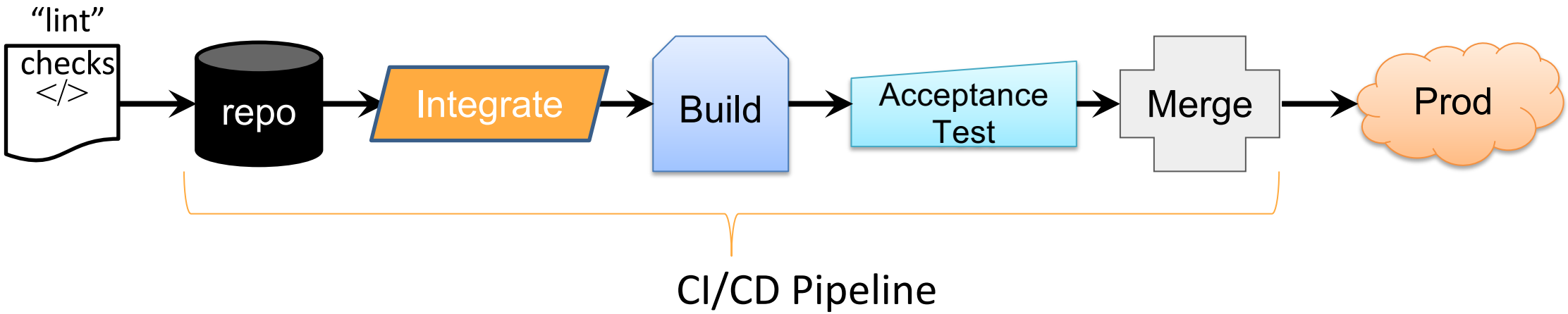
The test environment has to be ready, coordinated and capable.

Environment Orchestration: Automatically setup (and **reset**) the test environment and resources (physical and virtual) to match the requirements of a test.

Test automation: Execute test tasks without manual work required.



Testing at every stage



CI/CD pipeline provides an opportunity to integrate security tools.

Optimizing CI Workflows – Modular Product Design

- Modular code designs, using 12-factor-app principles
- Microservice architectures enable each portion of a product to be integrated as ready
- Containers optimize infrastructure orchestration
- Remove dependencies wherever possible
- Work off a common trunk, toggle functionality and features

Optimize CI Workflow – Accelerate CI Processes

- Pretest and pre-check integration deliverables
- Pipeline build and test processes reduces setup delays
- Fail Fast prioritizes important tests early
- Risk-based specific test selection speed up CI testing
- In-process analytics detect threshold exceptions
- Remediate problems with automated roll-backs
- Automatically revert changes that break CI processes

Continuous Integration and Containers

Containers are consistent, scalable, reusable

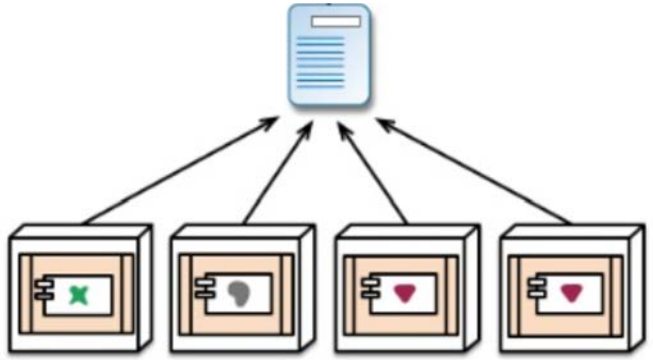
- Containers are the fastest means to launch short-lived, purpose-built testing sandboxes as part of a continuous integration process
- Containers are the end result of a build pipeline (artifact-to-workload)
- Test tools and test artifacts should be kept in separate test containers

Continuous Integration and Microservices

Microservices need to work together and alone

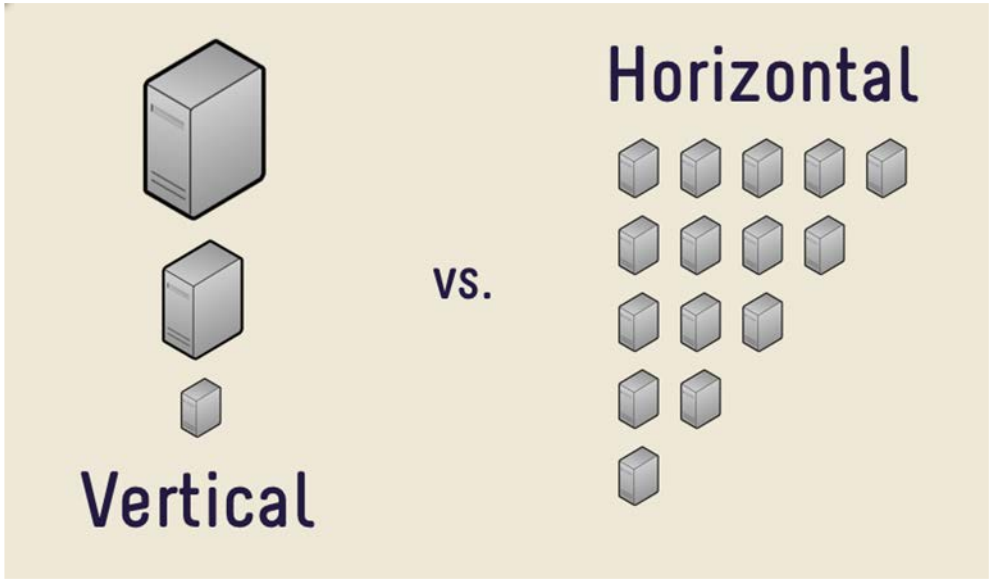
Recommended integration tests

1. Connection failures
2. Interactions between services
3. Dependencies between services
4. API contract
5. Aggregate performance



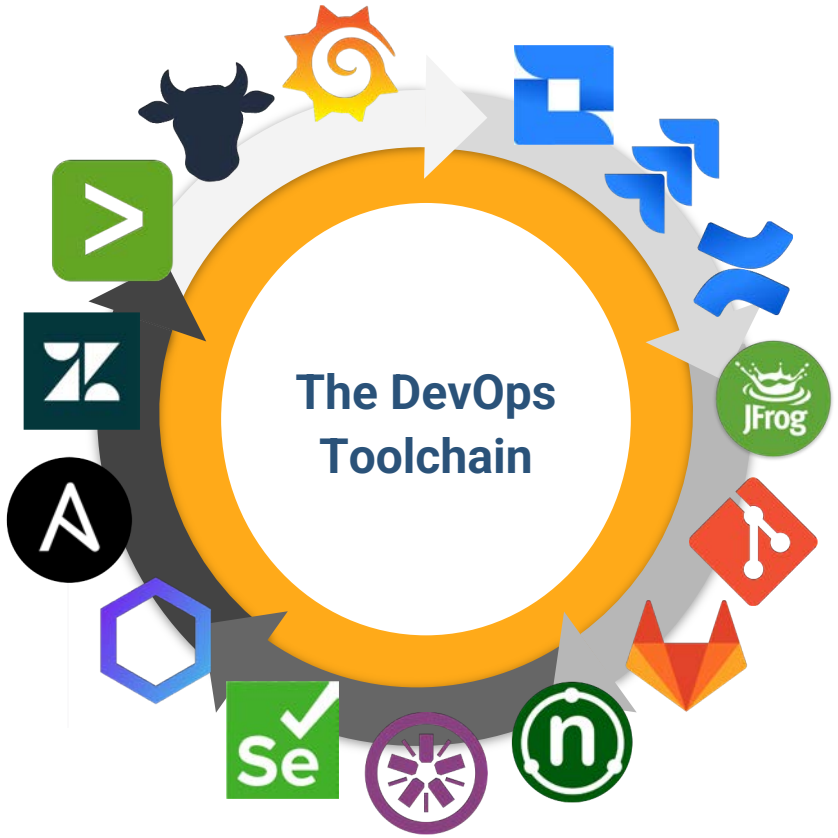
Optimizing CI Workflows – CI Infrastructure

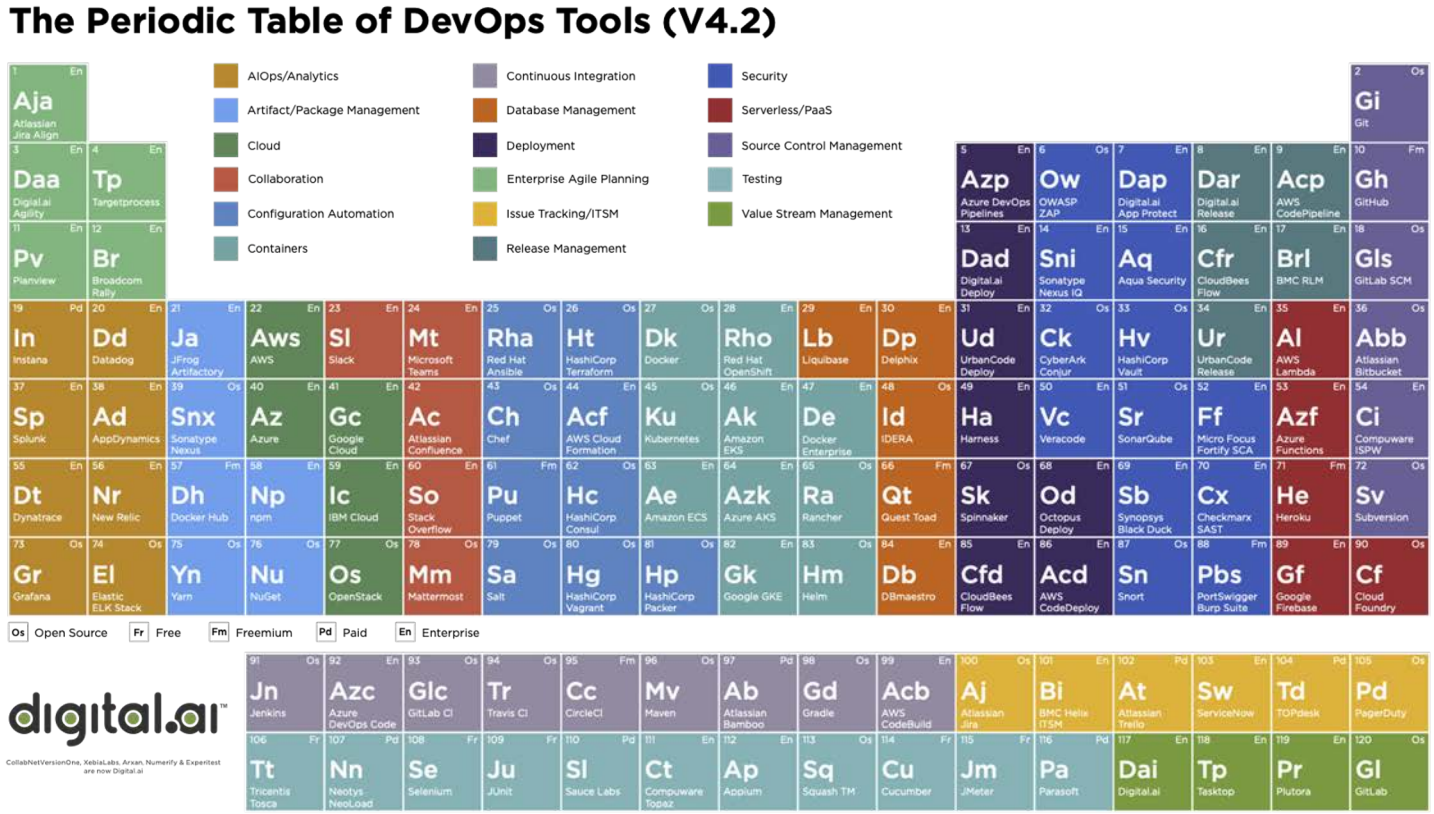
- Vertical and horizontal scaling of both the CI build resources and test resources creates an infrastructure optimized for each CI stage run.
- Predictive orchestration of CI infrastructures enables CI and test resources and processes to be set up in advance.



https://devops.com/continuous-can-integration/

The DevOps Ecosystem Has Many Tool Options





Building Your Toolchain

DevOps INSTITUTE

Categories of tools

1. **Code** —development, review, versions, merging
2. **Build** — continuous integration tools, build status
3. **Test** — test and results determine performance
4. **Package** — artifact repositories, pre-deployment
5. **Validate** — change management, release approvals

Key Takeaways

DevOps INSTITUTE

CI/CD Pipelines are the Factories of your team's value

Build for Speed

- CI/CD is for getting fast feedback, not just shipping fast
- Driving down cycle time for feedback means less context switching and context loss
- Small changes mean quick fixes

Test and Fix Continuously

- It's never too early to test
- Build failures need an effective triage process
- Automation helps you level up and focus on harder problems
- Unified tools can simplify operations and reduce overhead

Borrow & Build

- You can't shoehorn your team into someone else's workflow
- Start small and incrementally automate what will have the most positive impact
- Automation should support your humans

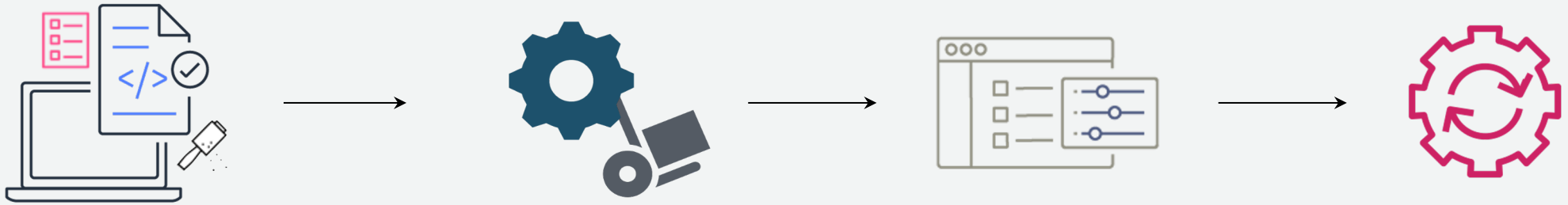
aws marketplace

Chris Chapman

Sr. Specialist Solutions Architect at AWS

chapmanInTheCloud

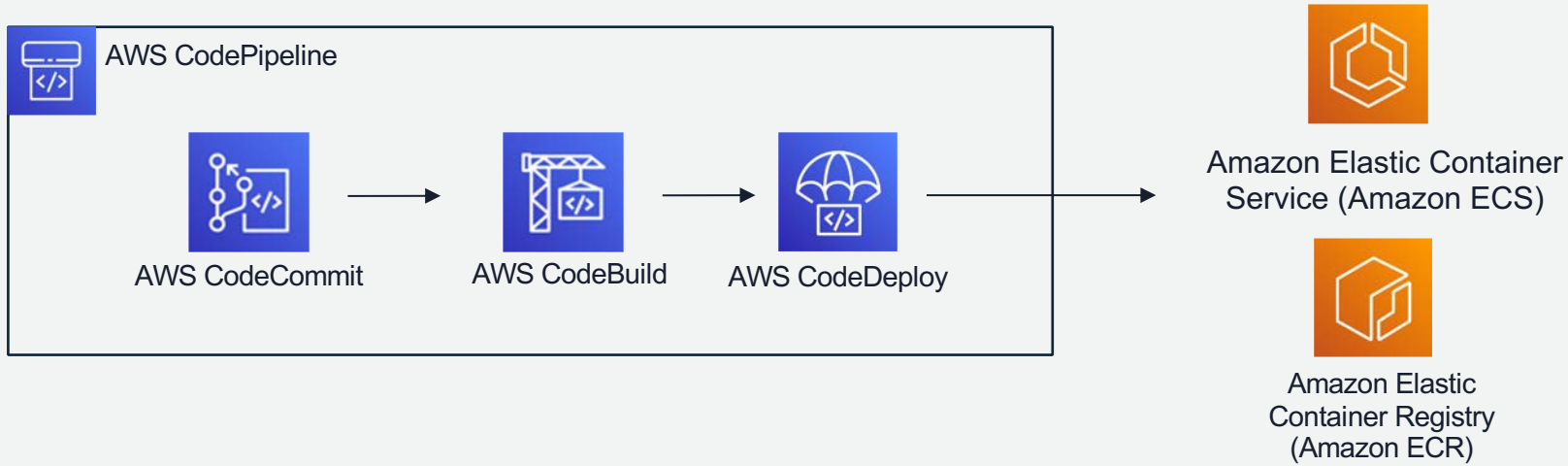
Steps for automating DevOps pipelines



1. Build your IaC templates
2. Automate deployment with template
3. Provide self-service
4. Automate the whole process of managing the IaC

aws | aws marketplace

Start Simple

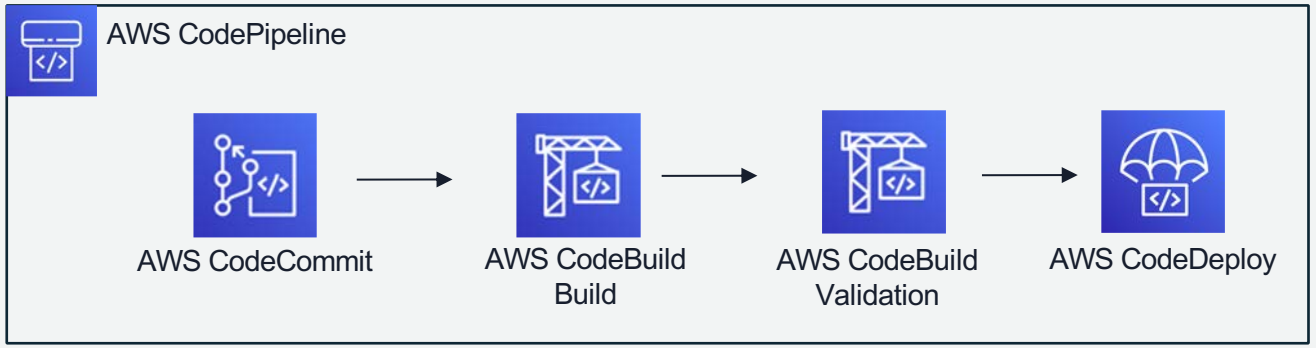


aws | aws marketplace

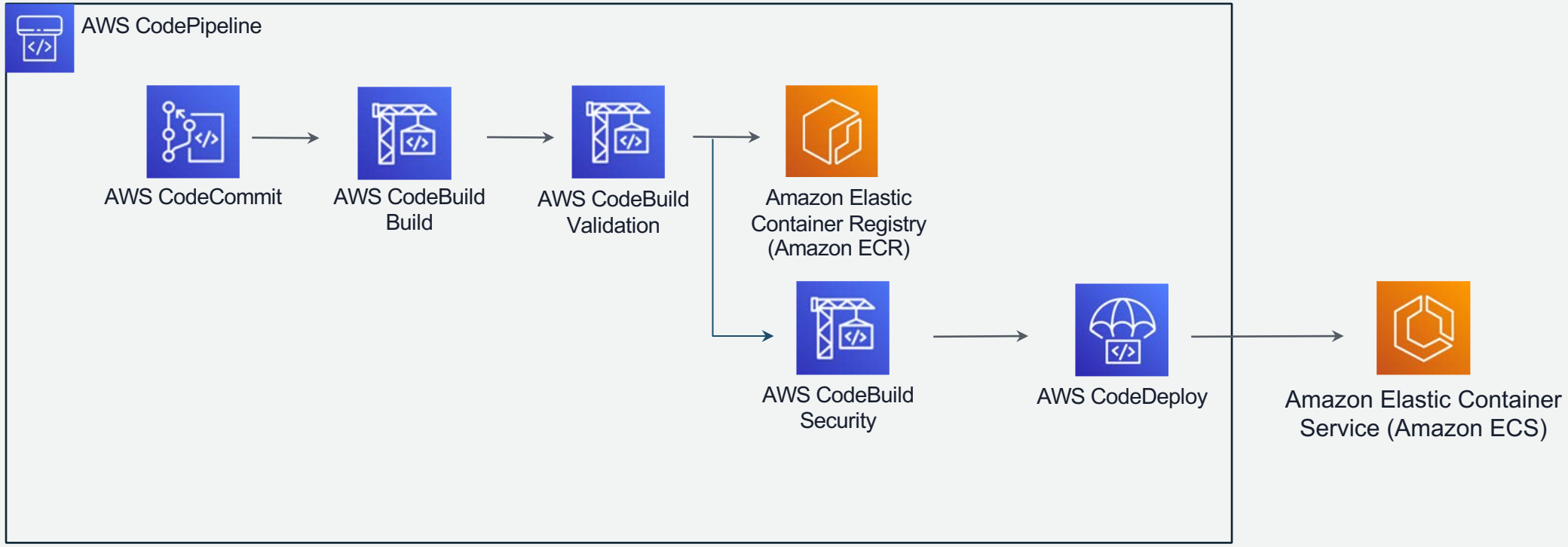
Buildspec files

```
version: 0.2
phases:
  install:
    runtime-versions:
      docker: 18
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      - $(aws ecr get-login --no-include-email --region $AWS_DEFAULT_REGION)
  build:
    commands:
      - echo Build started on `date`
      - echo building the C binary
      - make all
      - ./pytest.py
      - mkdir -p flaskapp\
      - cp flask/requirements.txt ./flaskapp
      - cp flask/application.py ./flaskapp
      - cp -R ./pycalc/ ./flaskapp
      - python3 -m venv ./flaskapp
      - cp ./bin/* ./flaskapp/bin/
      - echo Building the Docker image...
      - docker build -t $IMAGE_REPO_NAME:$IMAGE_TAG_LATEST .
      - docker tag $IMAGE_REPO_NAME:$IMAGE_TAG_LATEST $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/$
      - echo Pushing the Docker image...
      - docker push $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:$IMAGE_TAG_LATEST
  post_build:
    commands:
      - echo Build completed on `date`
      - ls
```

Incremental pipeline build – add validation

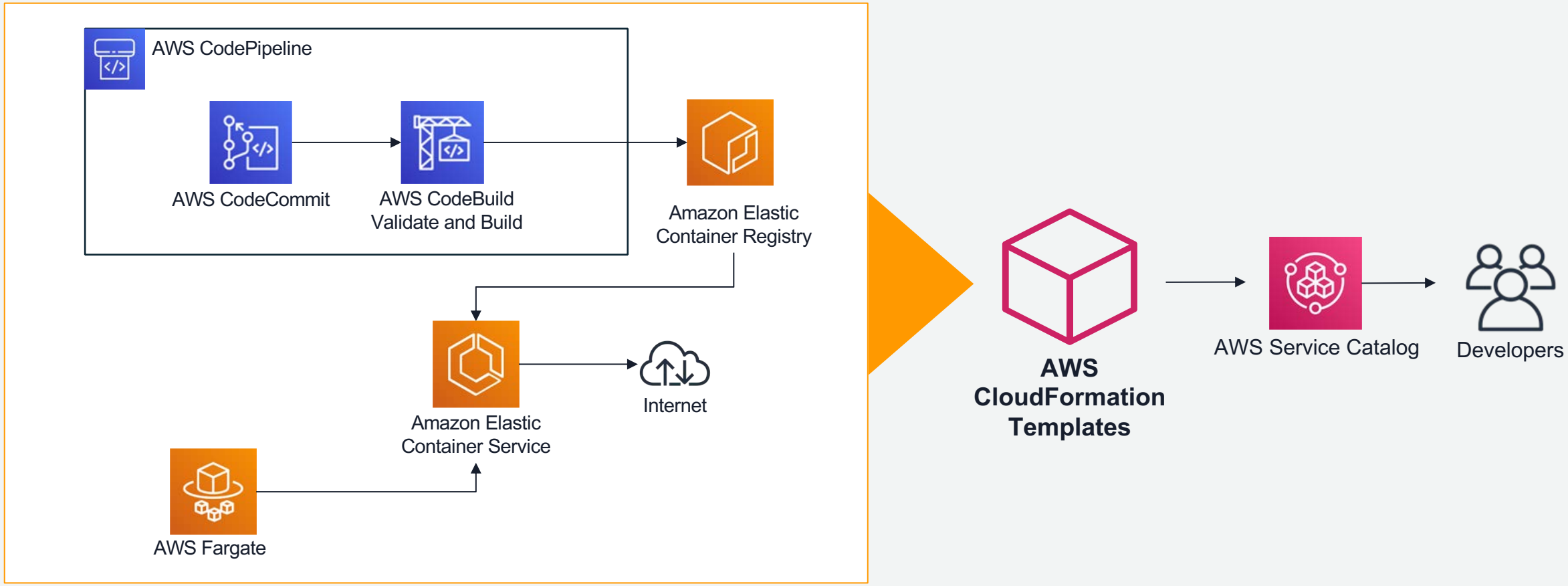


Incremental pipeline build – add security



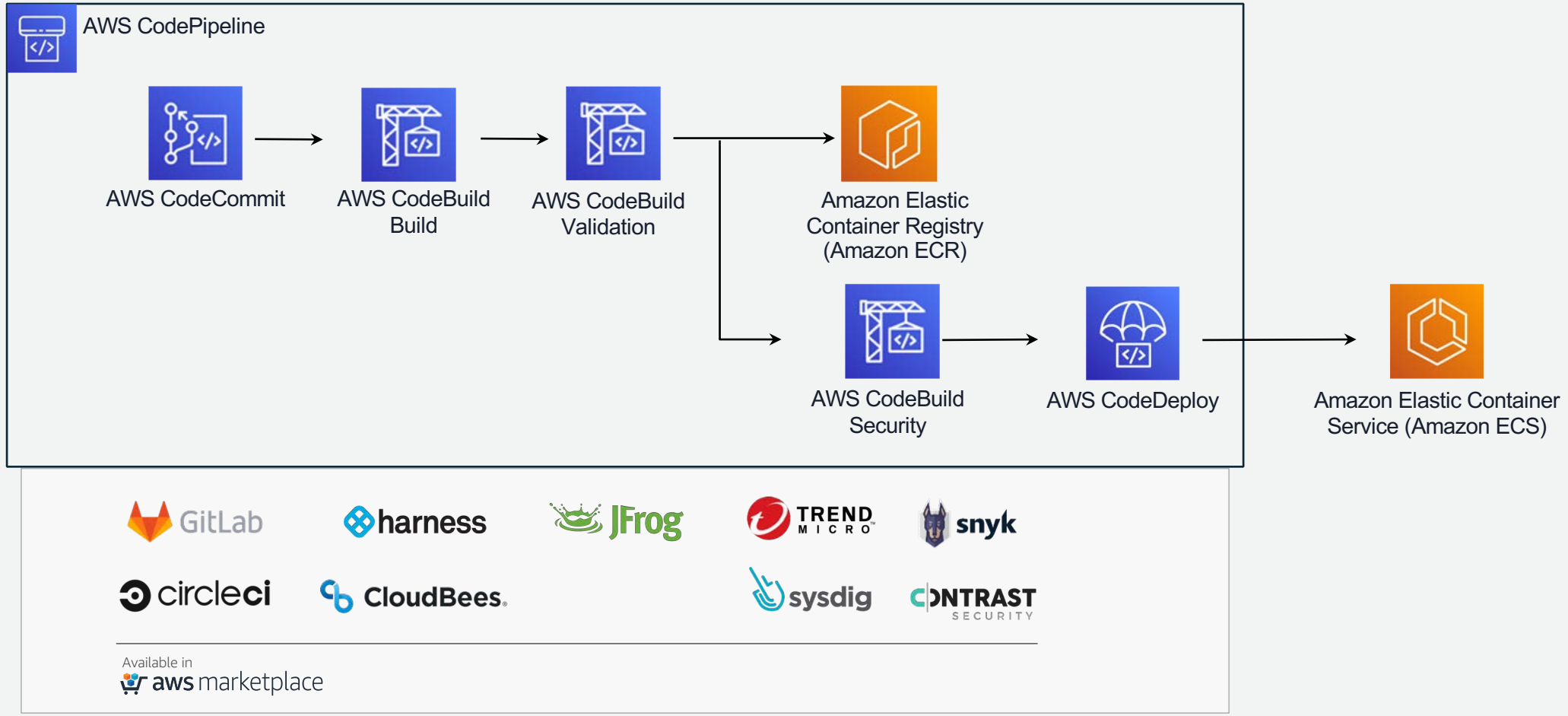
aws | aws marketplace

Building a shared self-service platform



aws | aws marketplace

Enhance pipeline capabilities with AWS Partners



aws | aws marketplace

aws marketplace

8,000+ listings • **1,600+** ISVs • **24** regions • **290,000+** customers • **1.5M+** subscriptions

AWS Marketplace DevOps Workshop Series participating partner hands-on labs

GitLab · JFrog · circleci · harness · armory · sumo logic

puppet · sysdig · TRICENTIS · MICRO FOCUS · CloudBees. · GENYMOTION

LaunchDarkly · TREND MICRO · PagerDuty · Gremlin · New Relic. · Quali

dynatrace · DATADOG · CONTRAST SECURITY · CORESTACK · Gatling · *And more coming soon!*

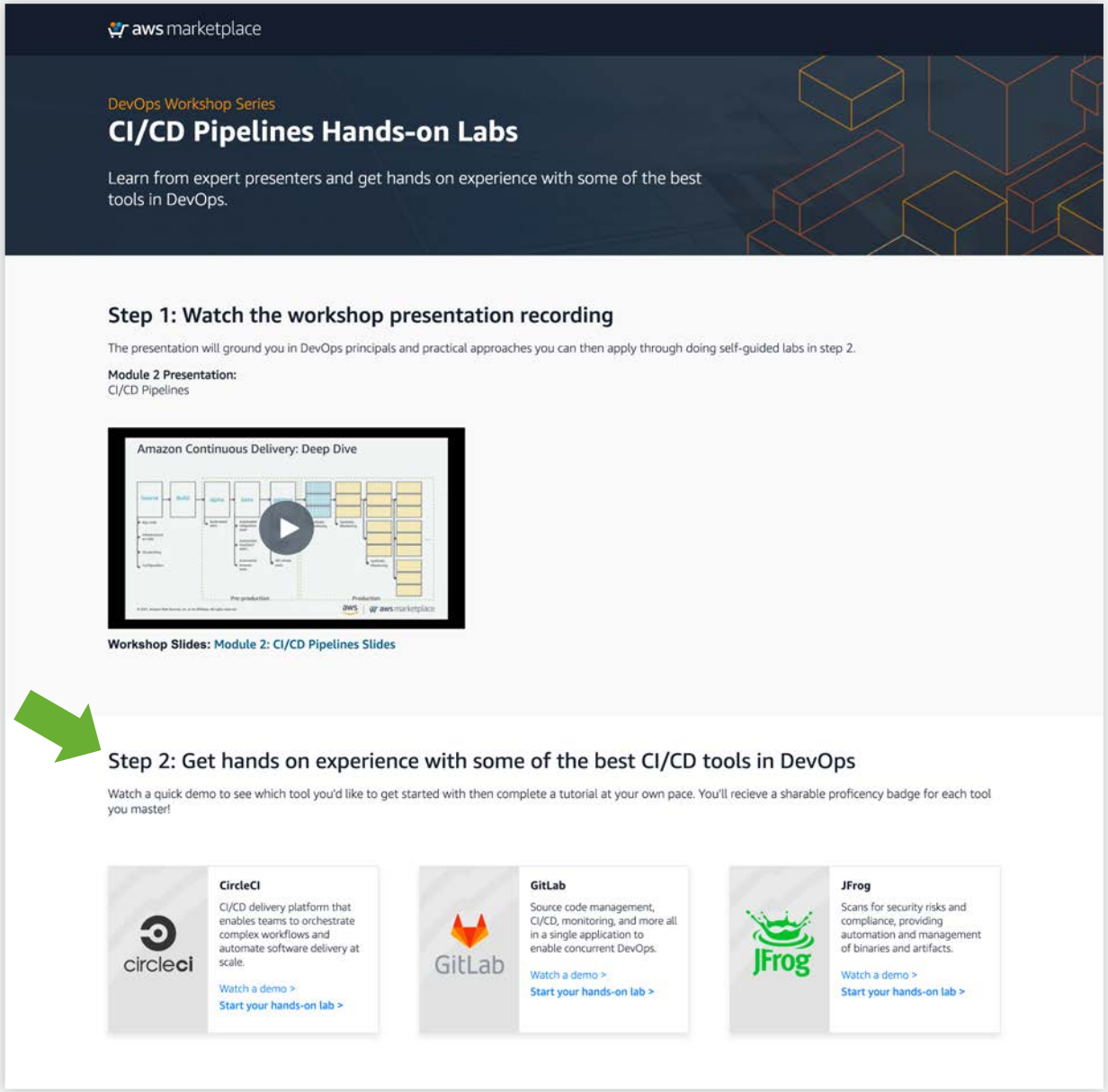
aws | aws marketplace

Important links and next steps

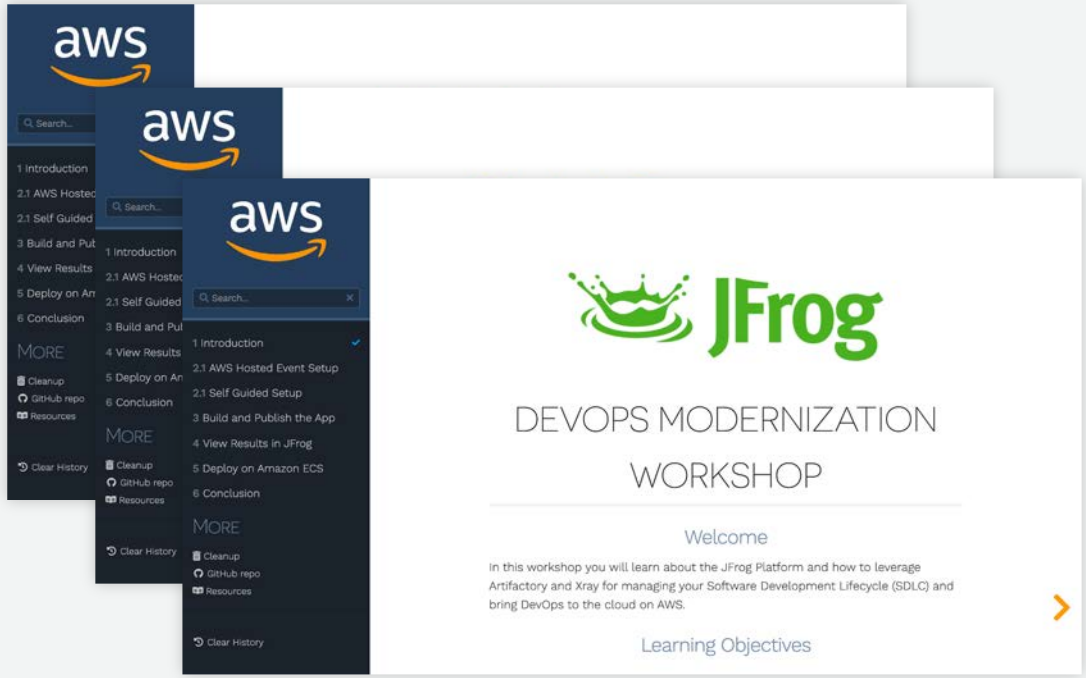
- Bookmark the Workshop Series landing page to navigate to all series content and subscribe for email updates
- Start putting knowledge into practice with the Module 2 hands-on labs
- Move on to Module 3: Evolving to Continuous Deployment
- Visit the AWS Marketplace website to learn more and experiment with DevOps tooling

aws | aws marketplace

Module 2 Hands-on Labs

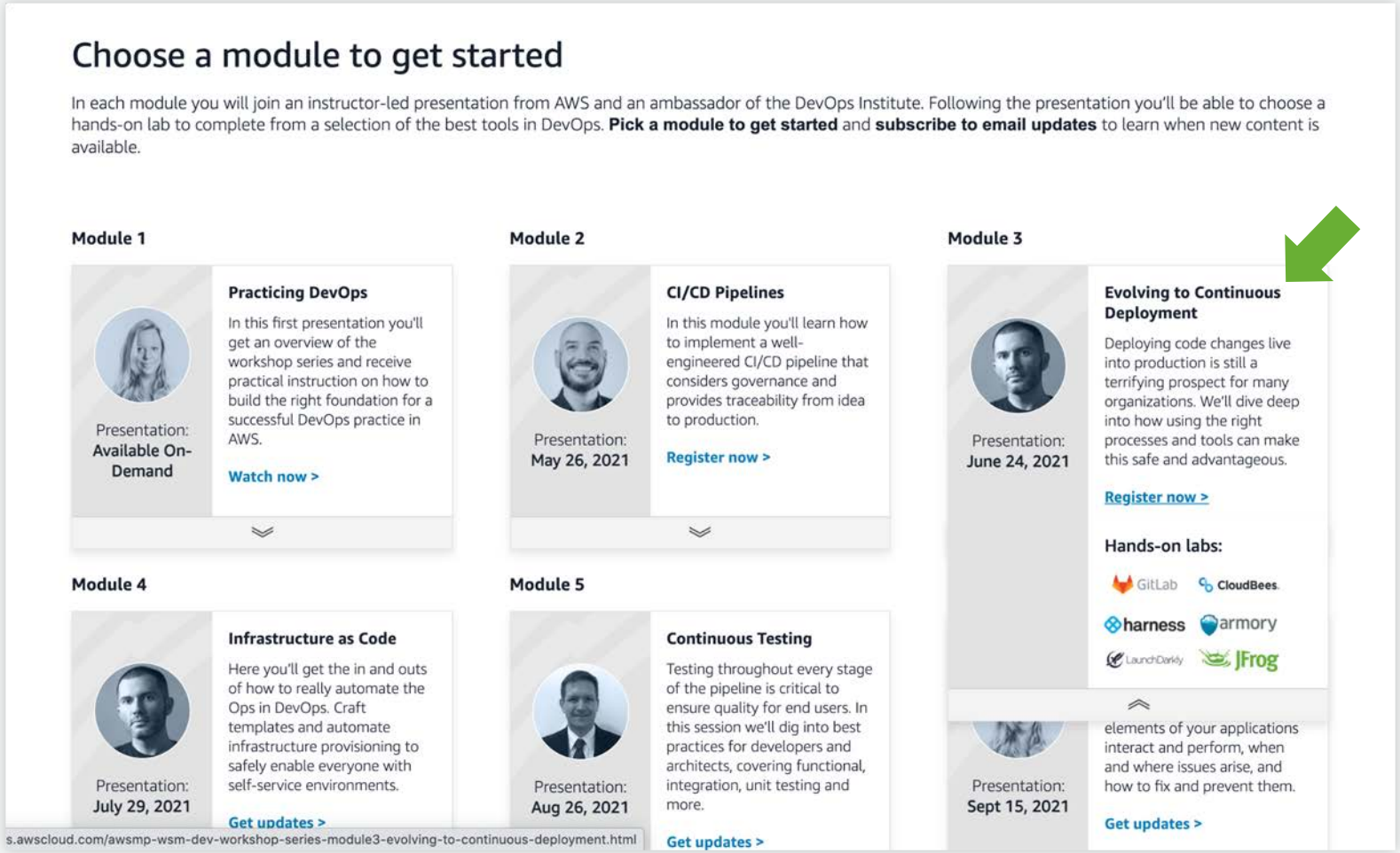


https://pages.awscloud.com/awsmp-wsm-dev-workshop-series-module2-cicd-pipelines-ty.html



aws | aws marketplace

Move on to Module 3: Evolving to Continuous Deployment



https://pages.awscloud.com/awsmp-wsm-dev-workshop-series-module3-evolving-to-continuous-deployment.html

aws | aws marketplace