



AWS Marketplace DevOps Workshop Series

Module 2: CI/CD Pipelines



Steve Pereira

Ambassador, DevOps Institute

@steveelsewhere

devopsto



Chris Chapman

Sr. Specialist Solutions Architect at AWS

chapmanInTheCloud





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



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.



TIME: Start → WAIT → Step → Step → Delivered

MEASUREMENT:

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](#)

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](#)

Software Delivery Value Stream





Value Stream Mapping Key Questions Answered

WATCH

How to Navigate Software Delivery With Confidence

Navigating in 2020 can seem like wandering in the wilderness. How do you know you're heading in the right direction?

WATCH

5 Keys of DevOps + Value Stream Assessment

PLUTORA

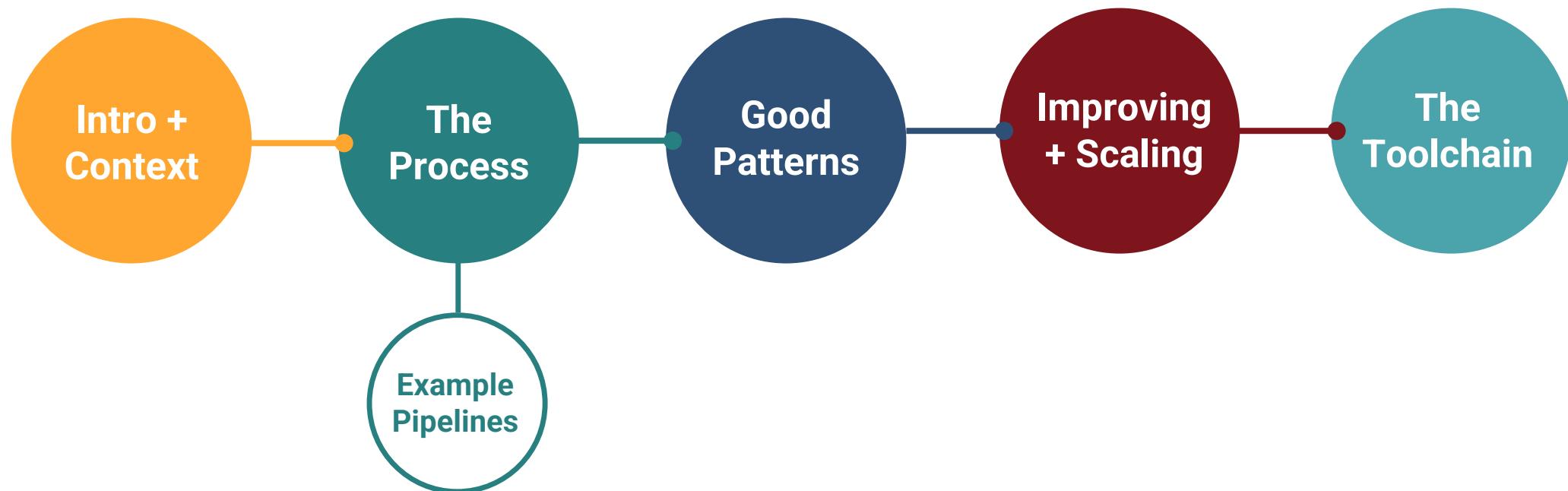
WATCH

PROJECTS



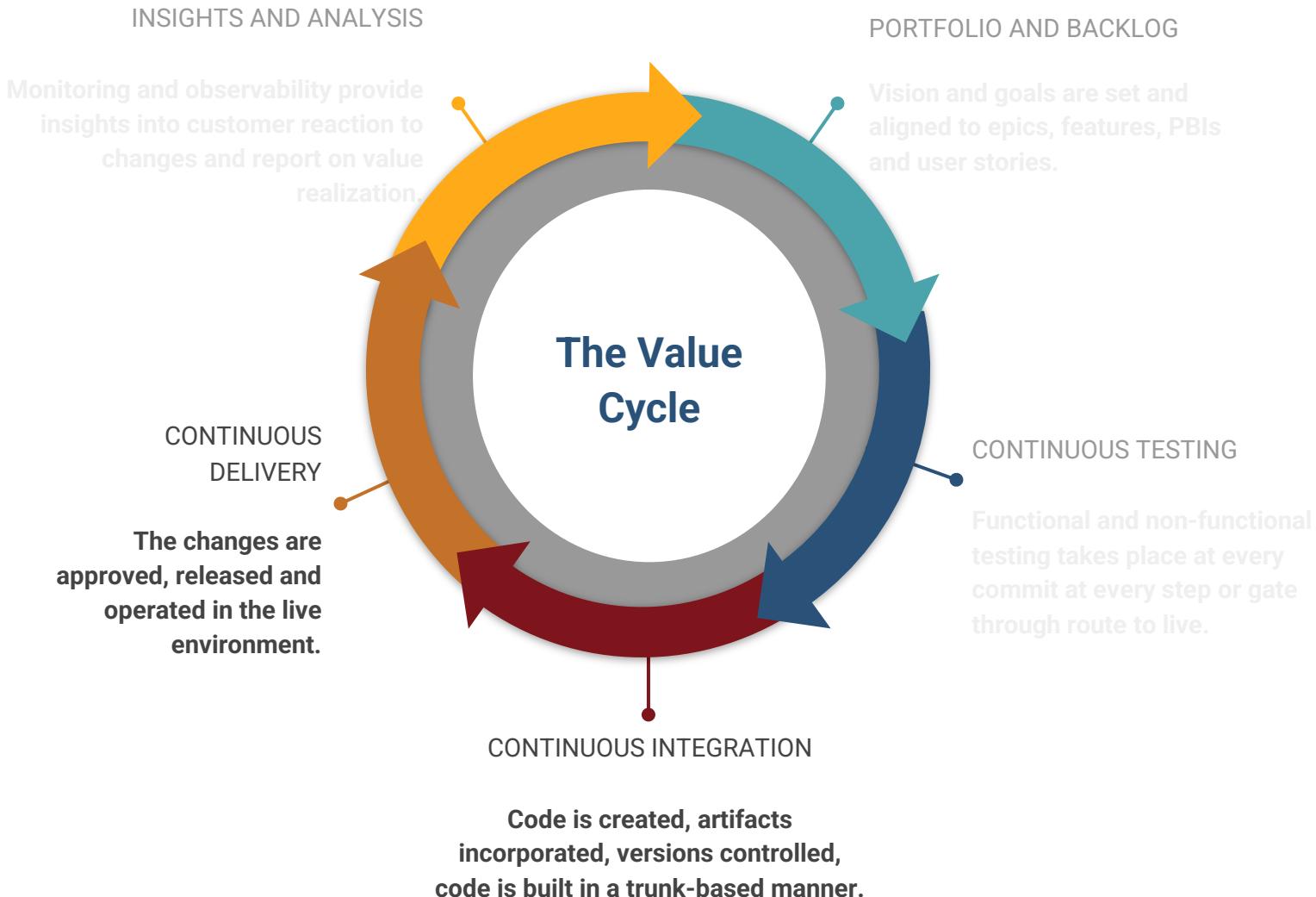


Flow: Talk Map





The Value Cycle

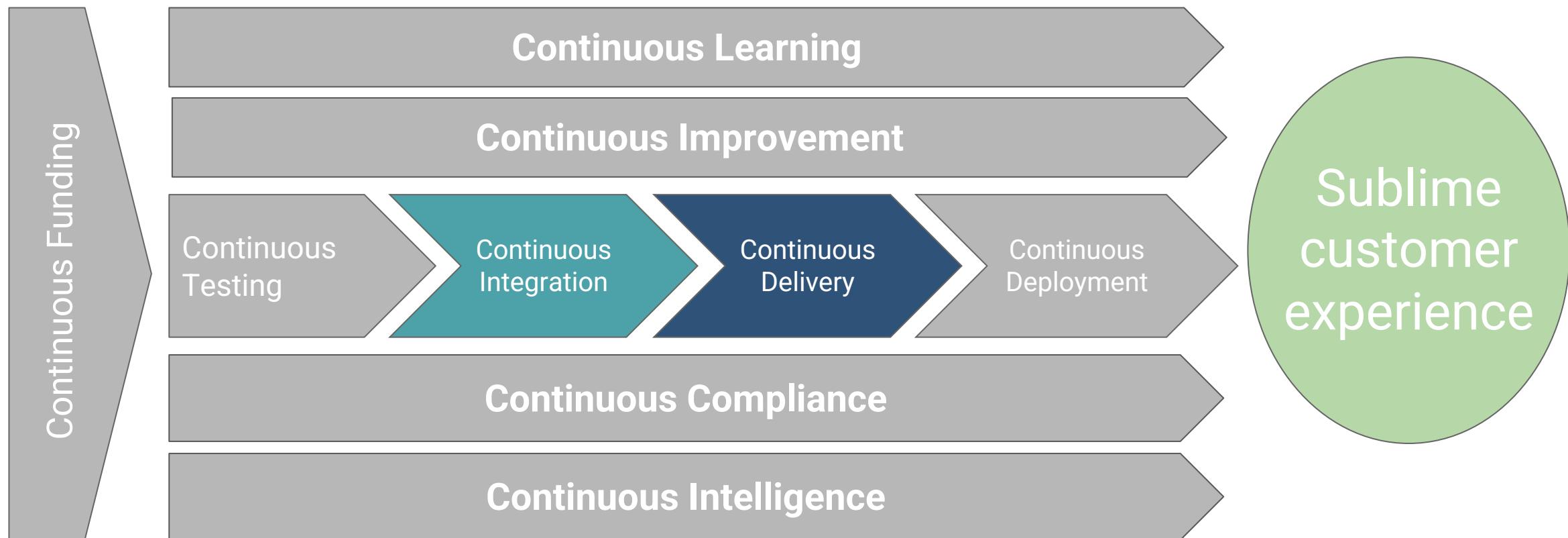




DevOps Practices



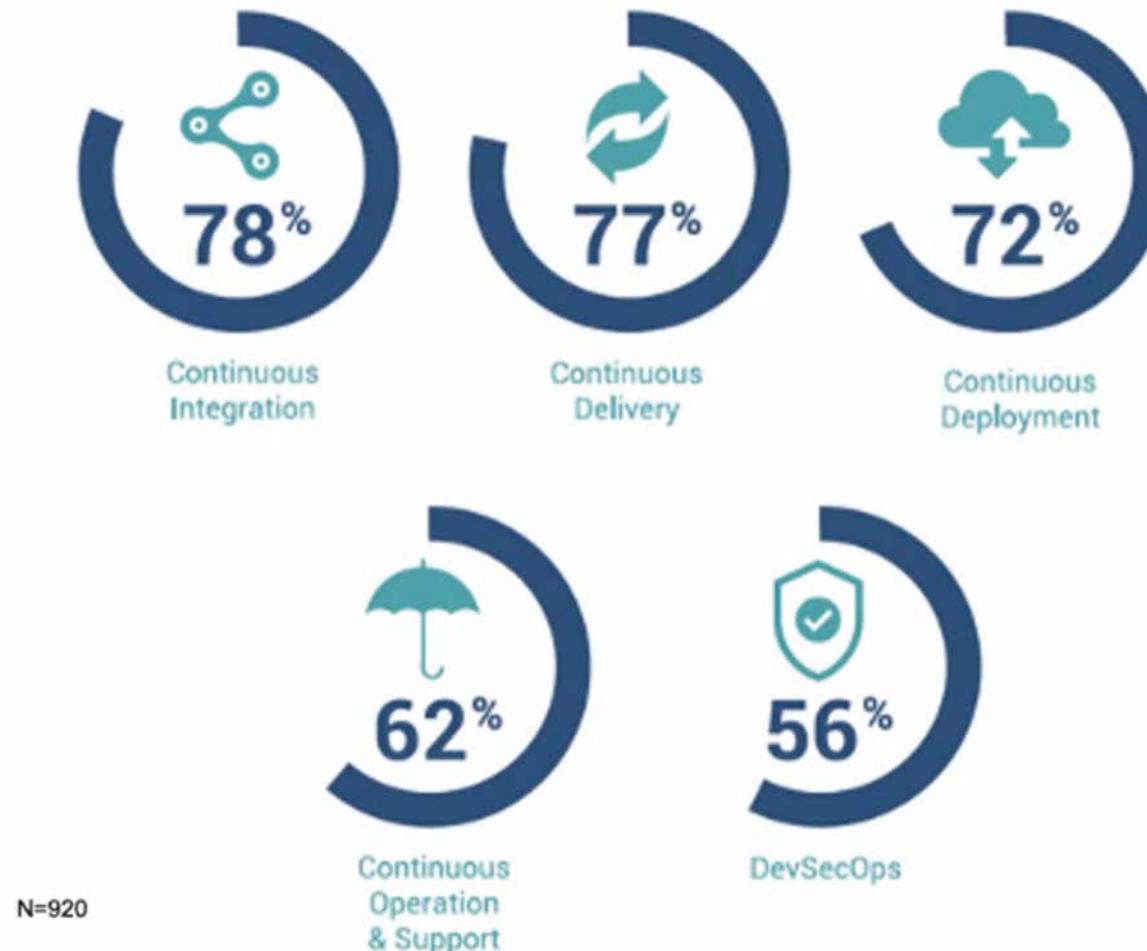
All the continuouses





#1 and 2 Must-Have Automation Skills

CI + CD are leading valuable skills



N=920

2020 survey data N=920



Continuous Integration Defined

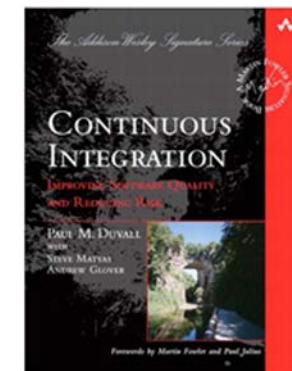
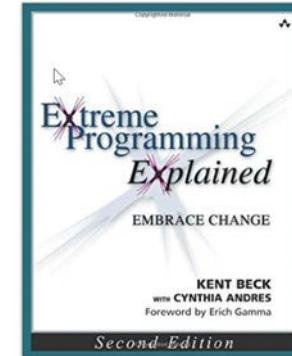
Continuous Integration is a software development practice where members of a team integrate their work frequently.

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'



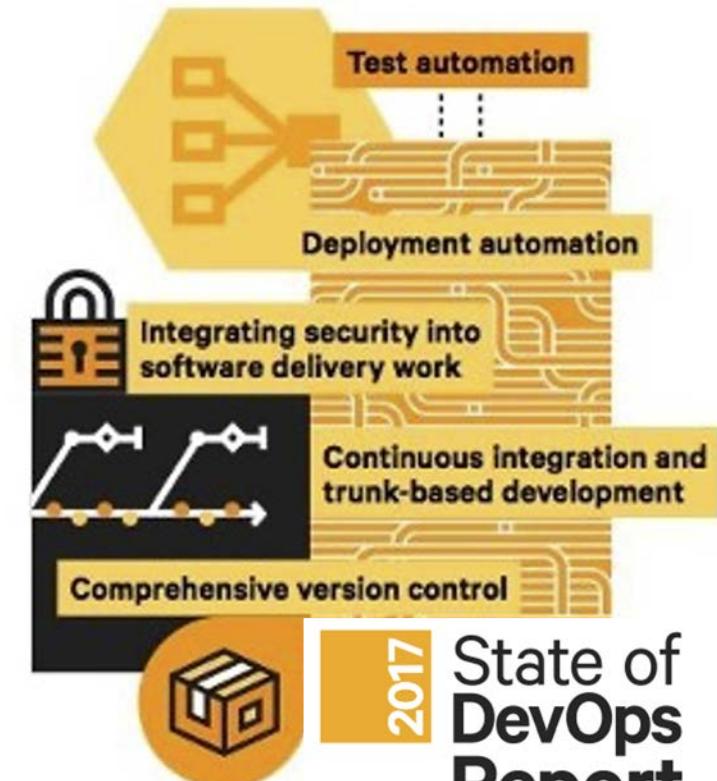
Continuous Delivery

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



Factors that positively contribute to continuous delivery:





Continuous Delivery



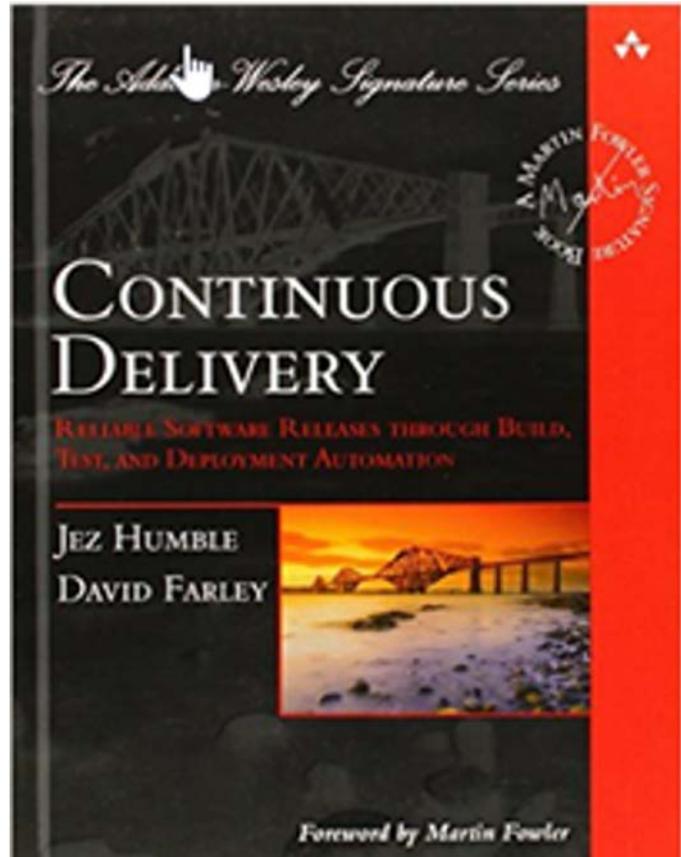
Continuous delivery (CD) is a software engineering approach [associated with DevOps,] in which **teams produce software in short cycles**, ensuring that the software can be **reliably released at any time**. It aims at building, testing, and releasing software **faster and more frequently**.

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...



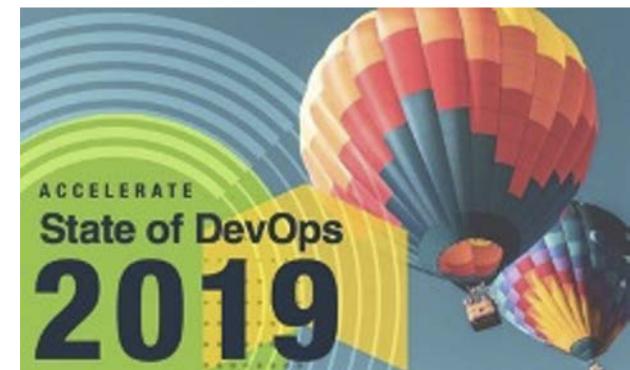
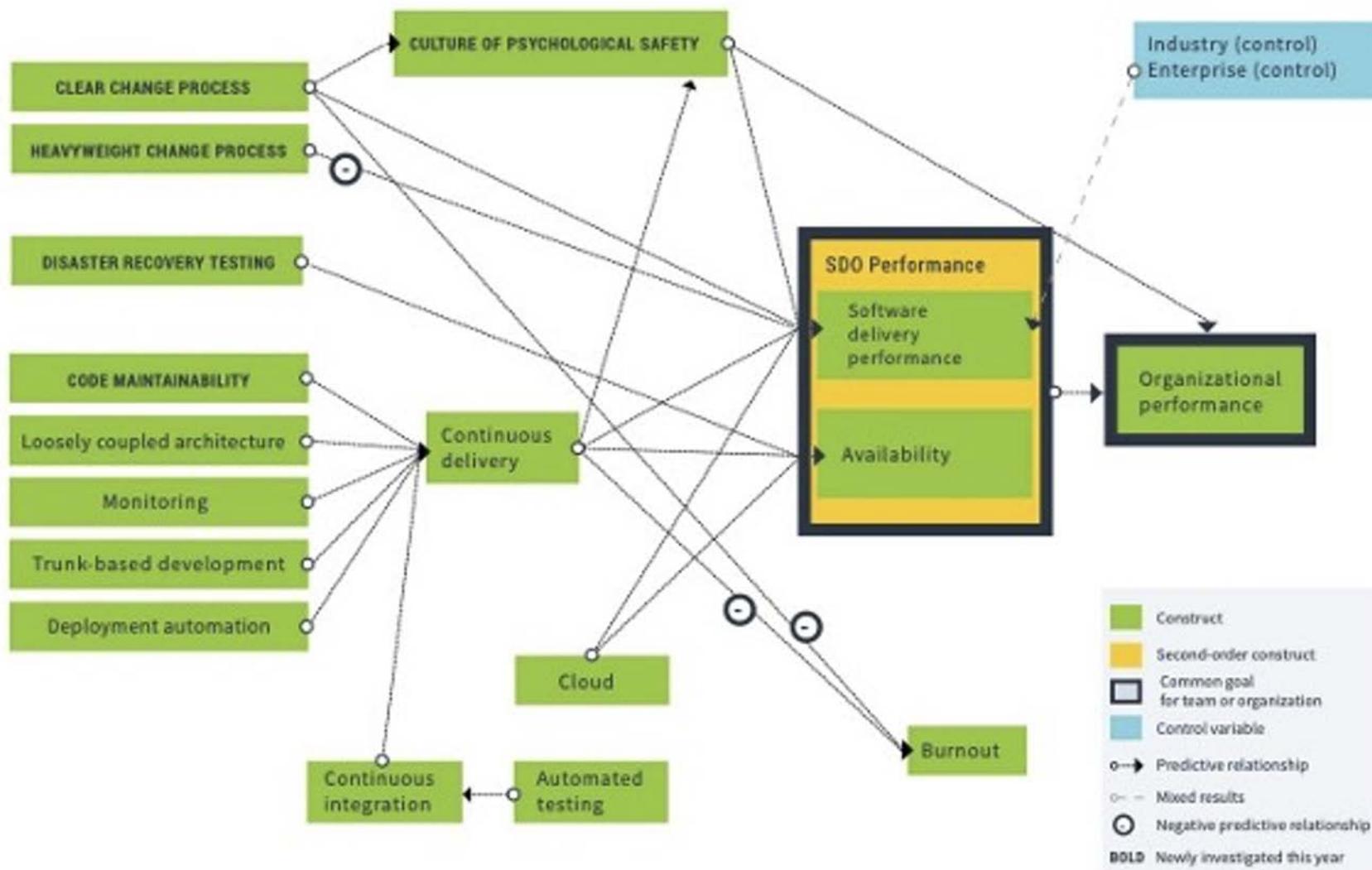
“The ability to get changes - features, configuration changes, bug fixes, experiments - into production or into the hands of users safely and quickly in a sustainable way “

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



Continuous Delivery

Leads to higher organizational performance

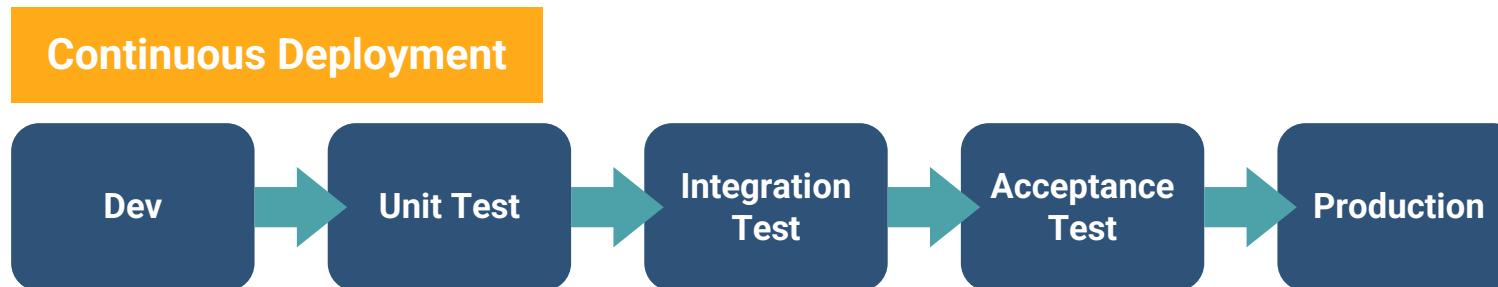
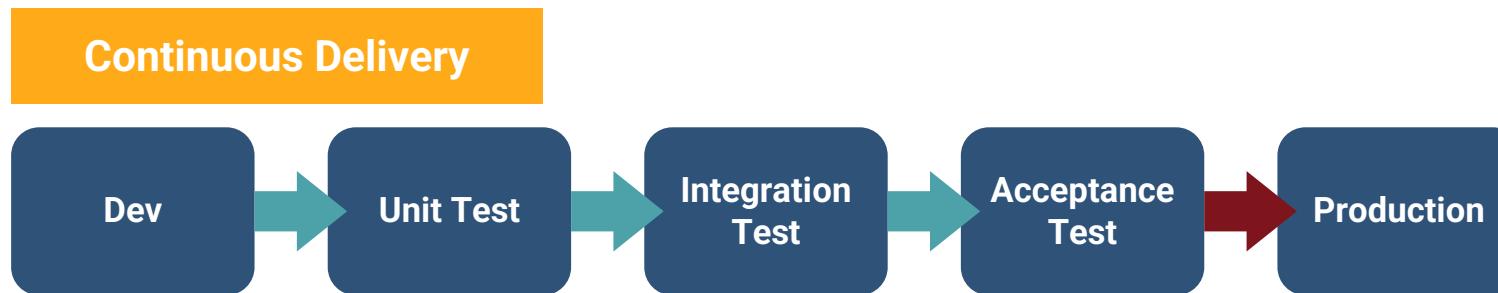




Continuous Delivery & Continuous Deployment



Automatic trigger Manual trigger



DevOps for the Modern Enterprise

Winning Practices to Transform Legacy IT Organizations

Mirco Hering
Foreword by Dr. Bhaskar Ghosh



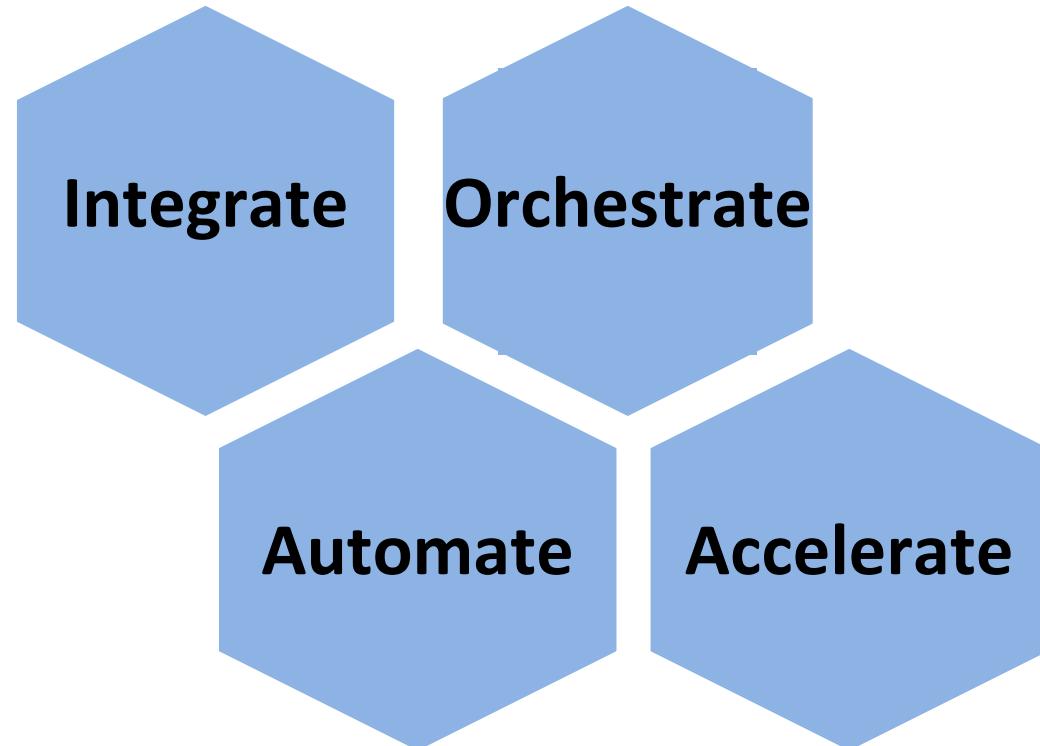
From: Mirco Hering: notafactoryanymore.com, author of 'DevOps for the Modern Enterprise'



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





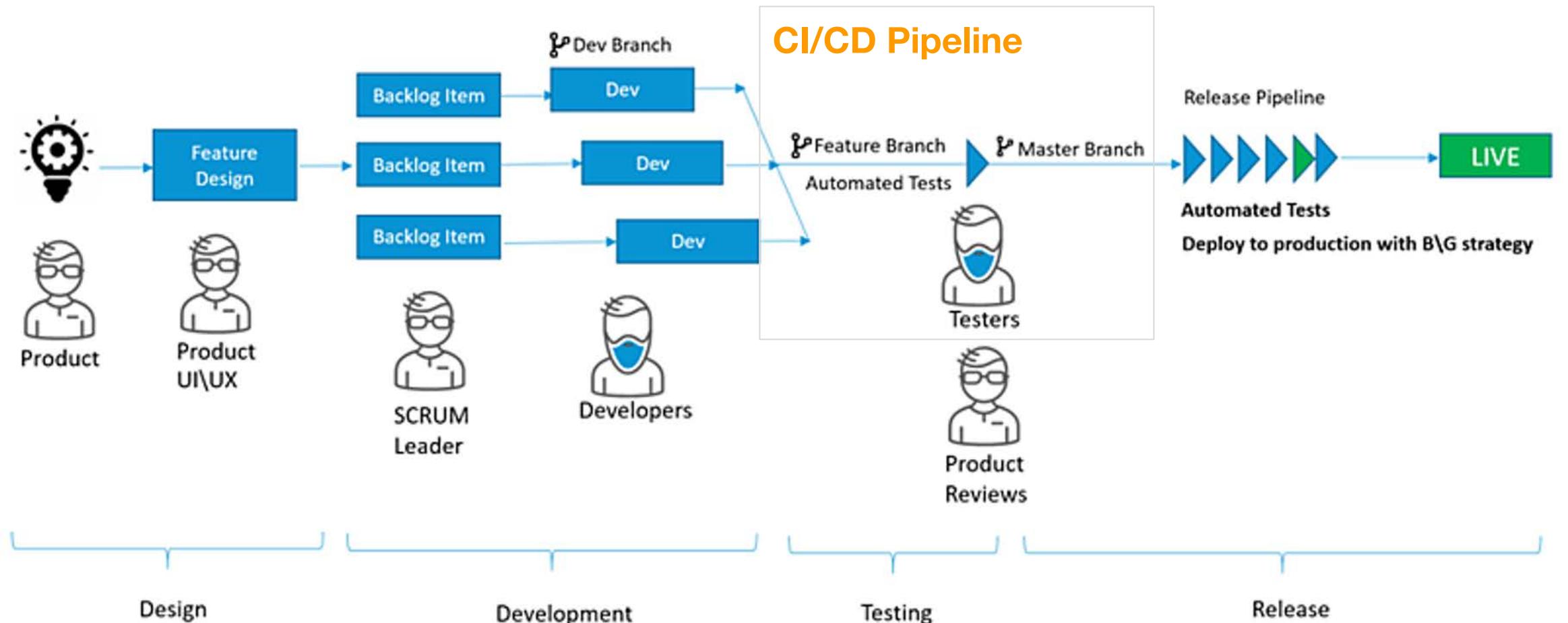
Consequences of NOT Doing Continuous Delivery Properly



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

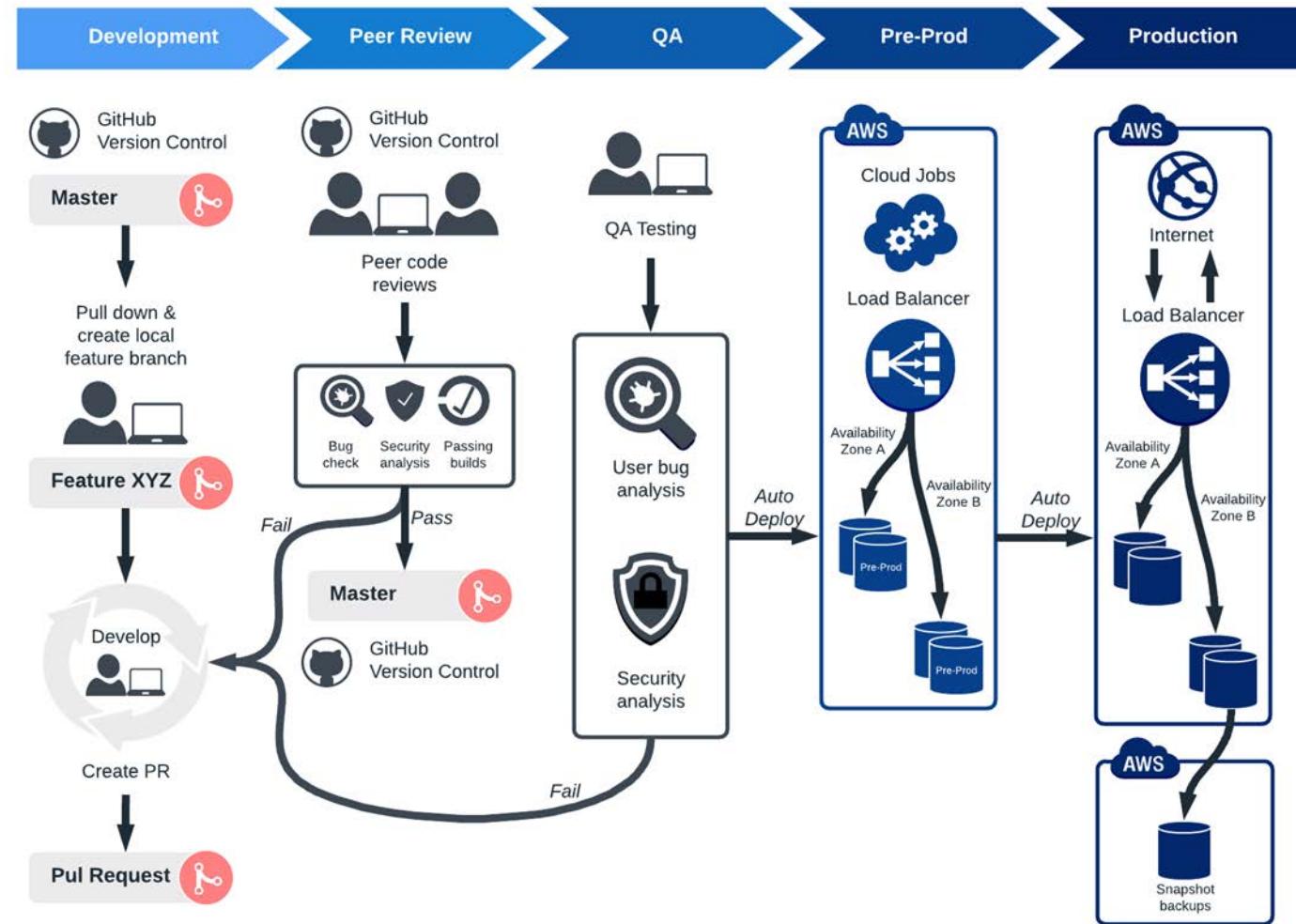


Sample Process



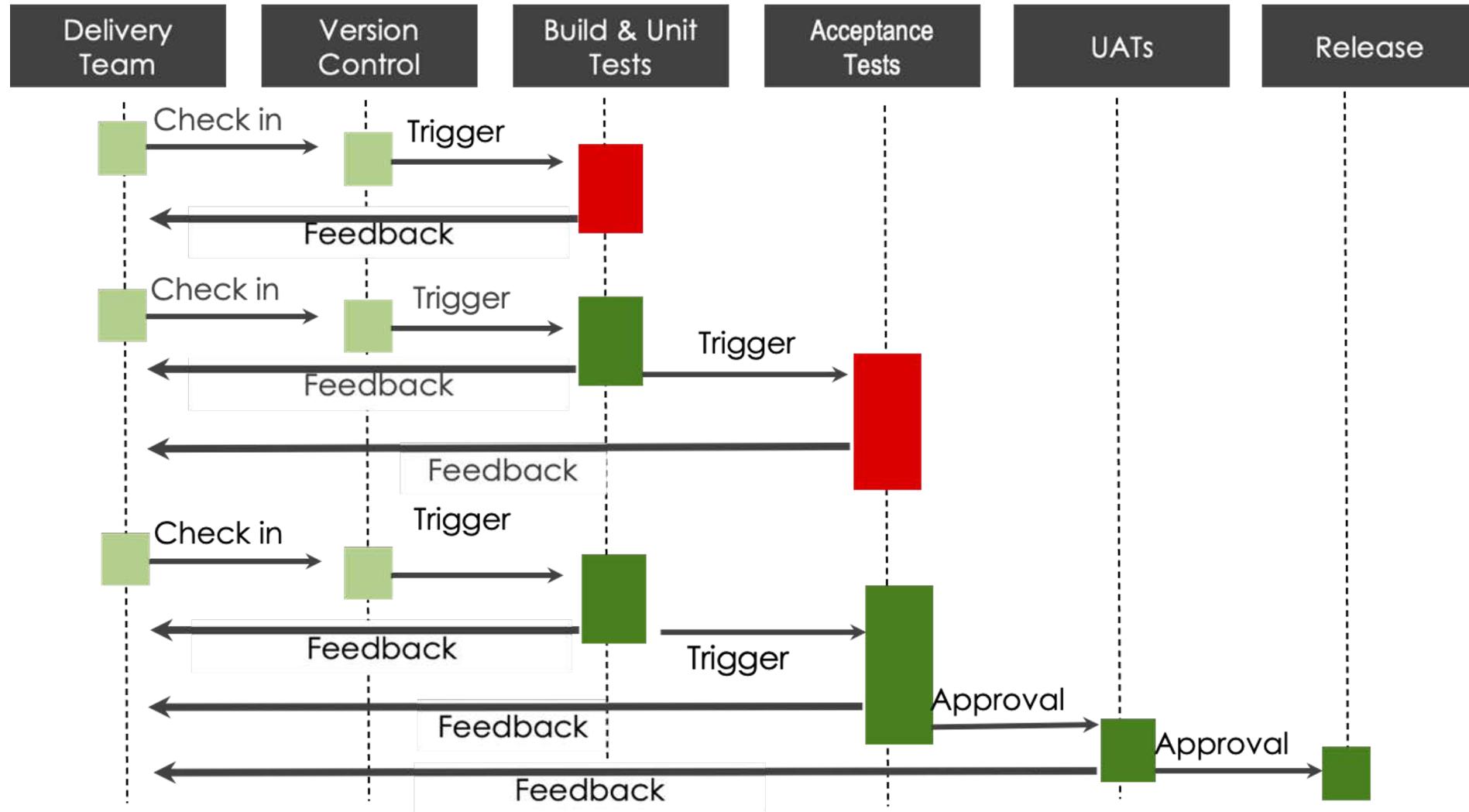


Pipeline Workflow



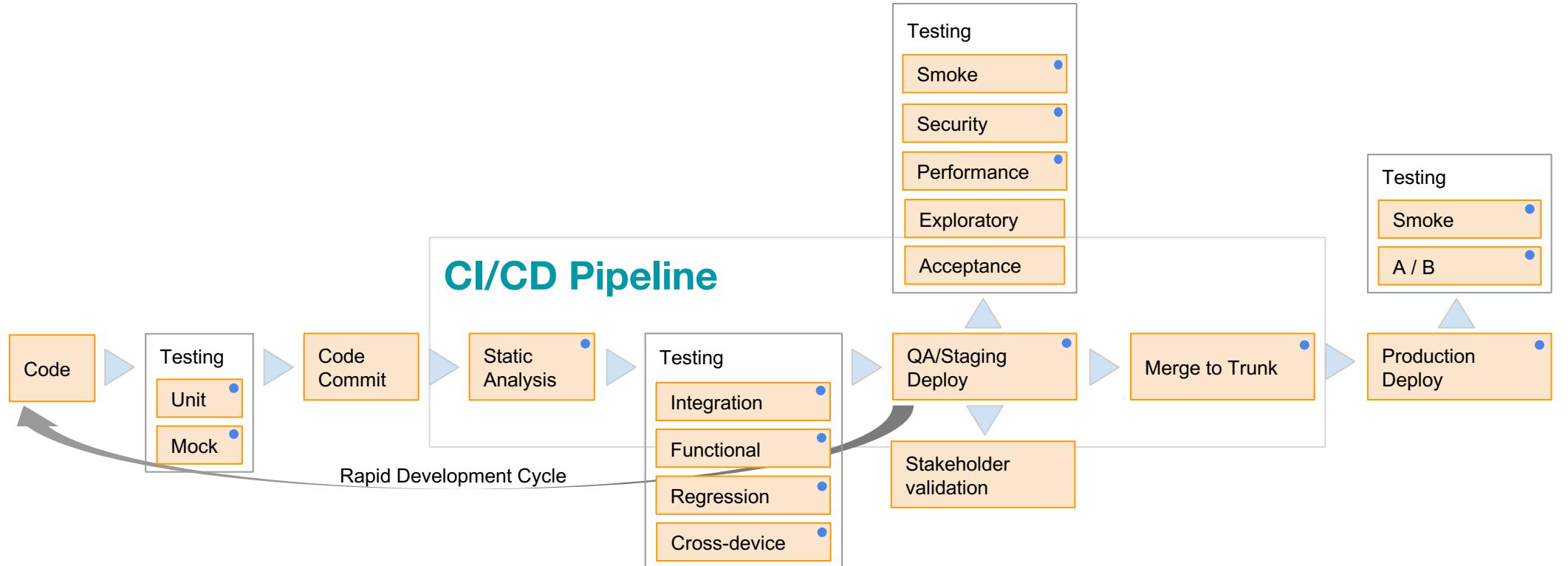


The Delivery Pipeline





Sample Pipeline



- Automation available

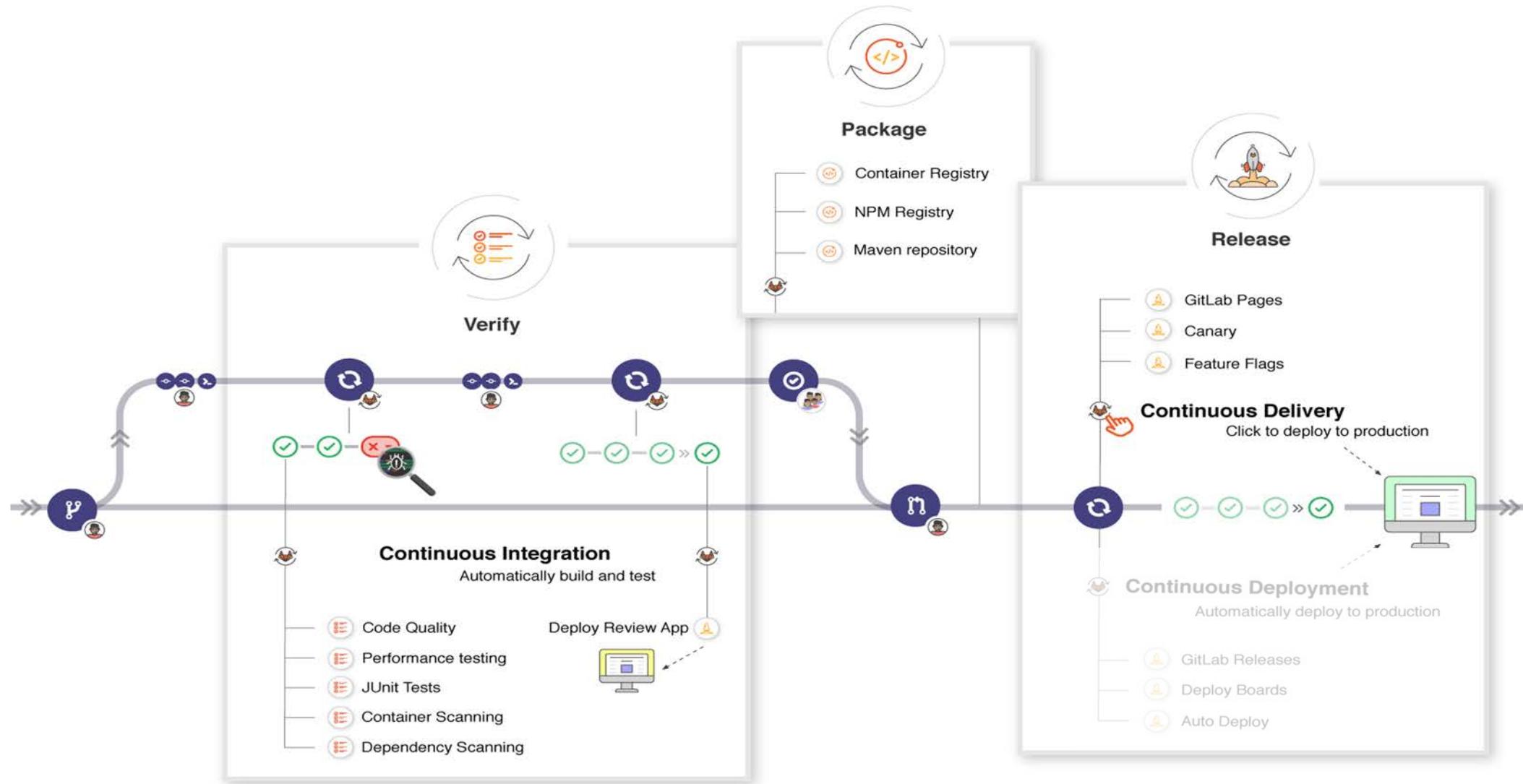
Increased confidence / Decreased risk

Measurement / Data collection / Feedback

Visible

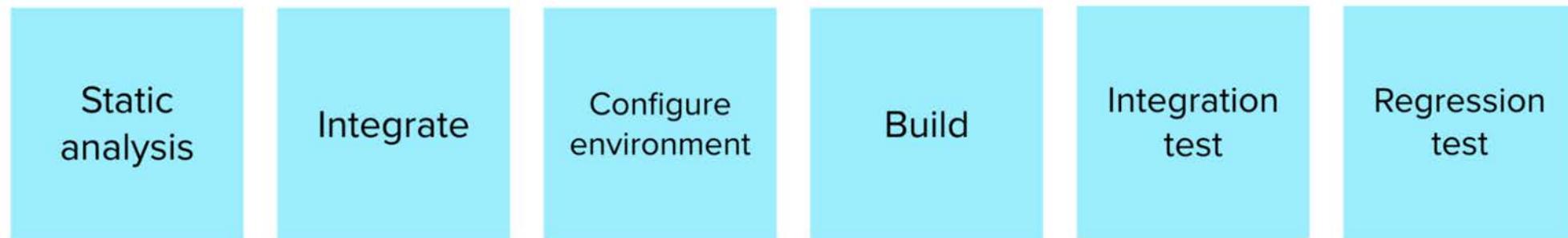


Pipeline Examples





Pipeline Creation and Improvement



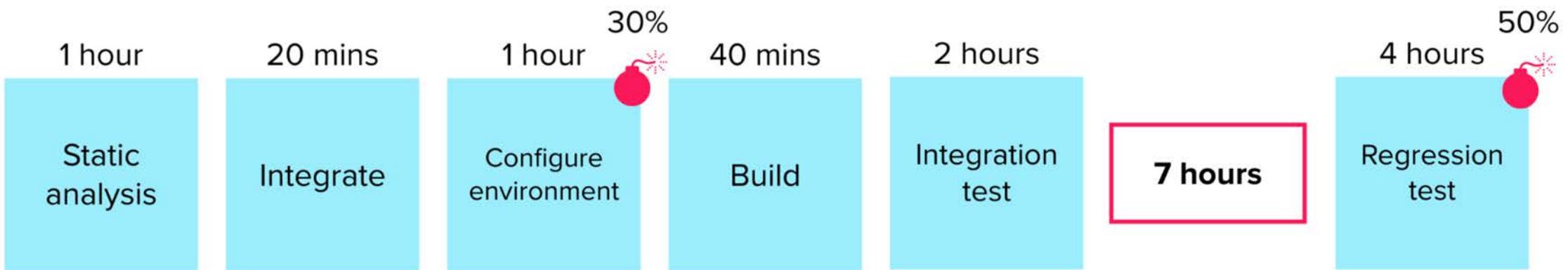


Pipeline Creation and Improvement





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%	0%	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

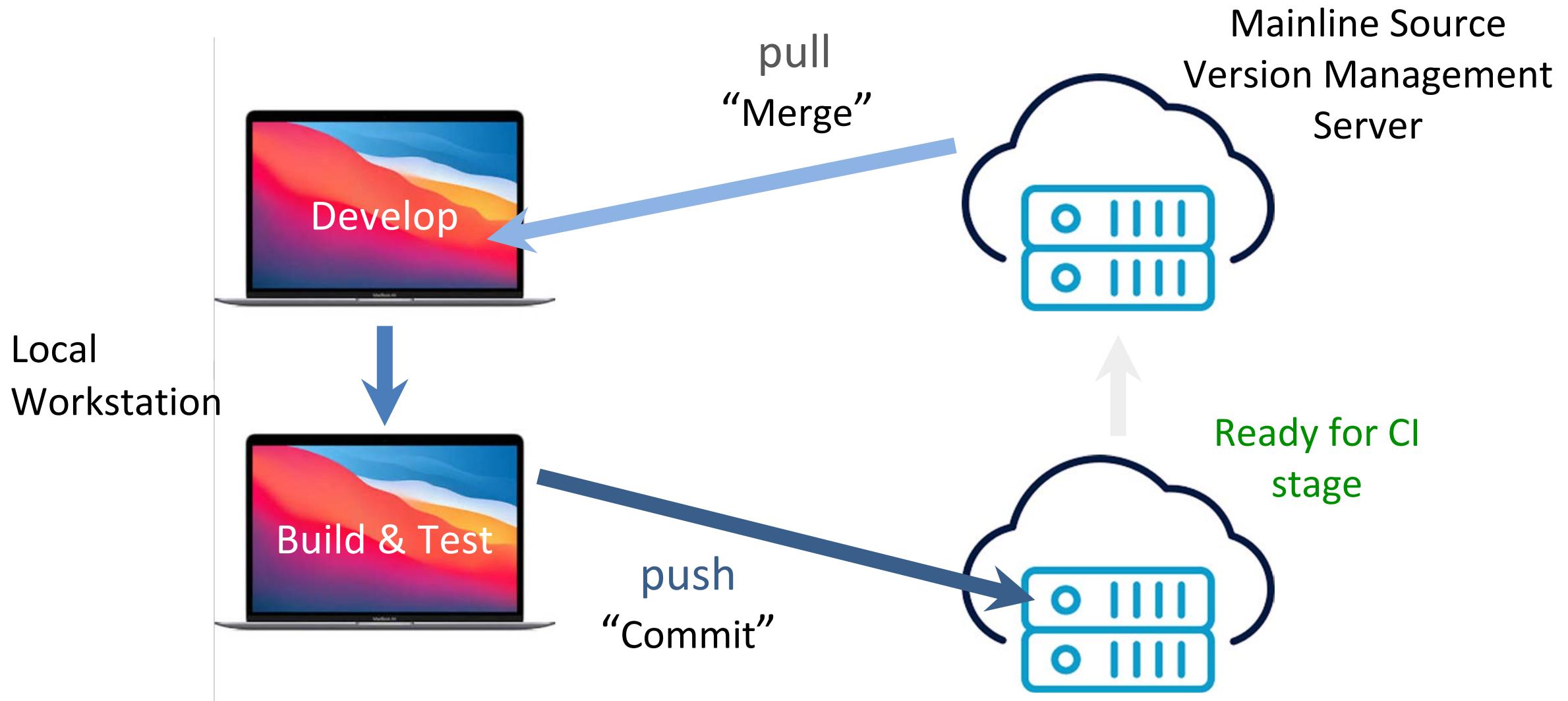
Caution: Distributed development teams must have efficient networked access and self-service capabilities for

- CI environments
- Configurations
- Builds
- Test

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



Everyone Commits to Mainline Every Day



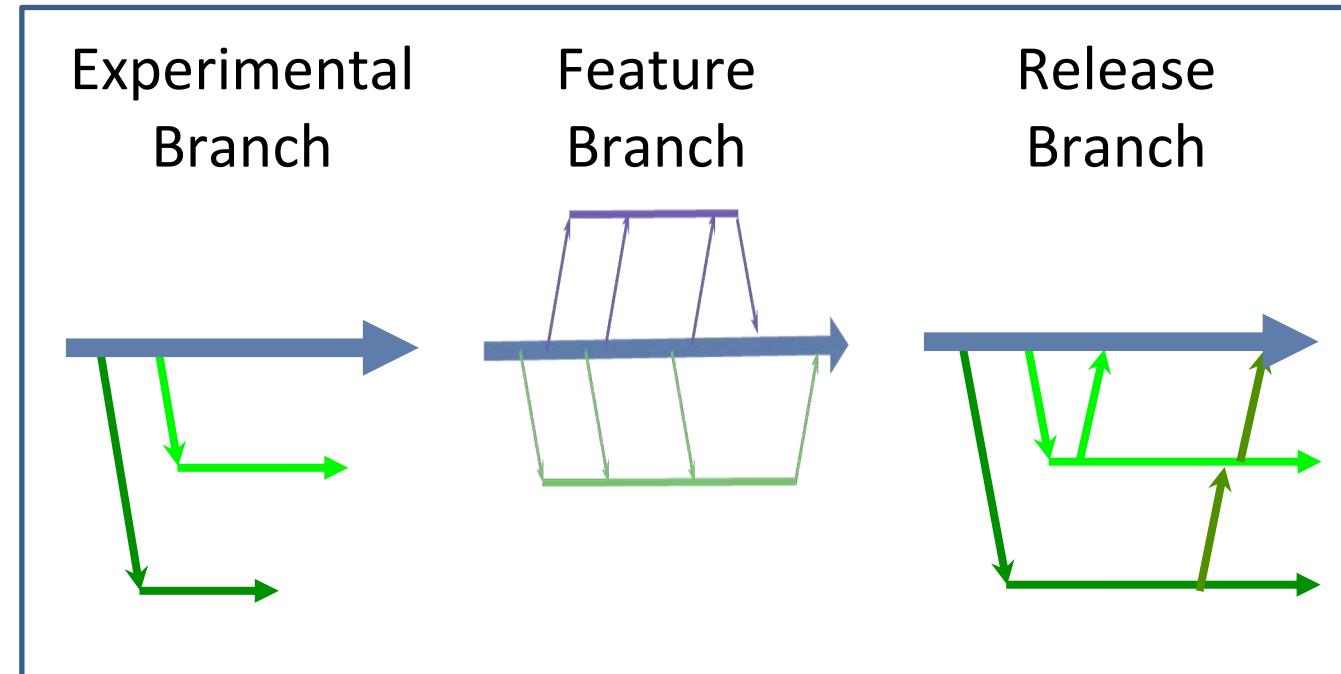


Branching and Merging Best Practices



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

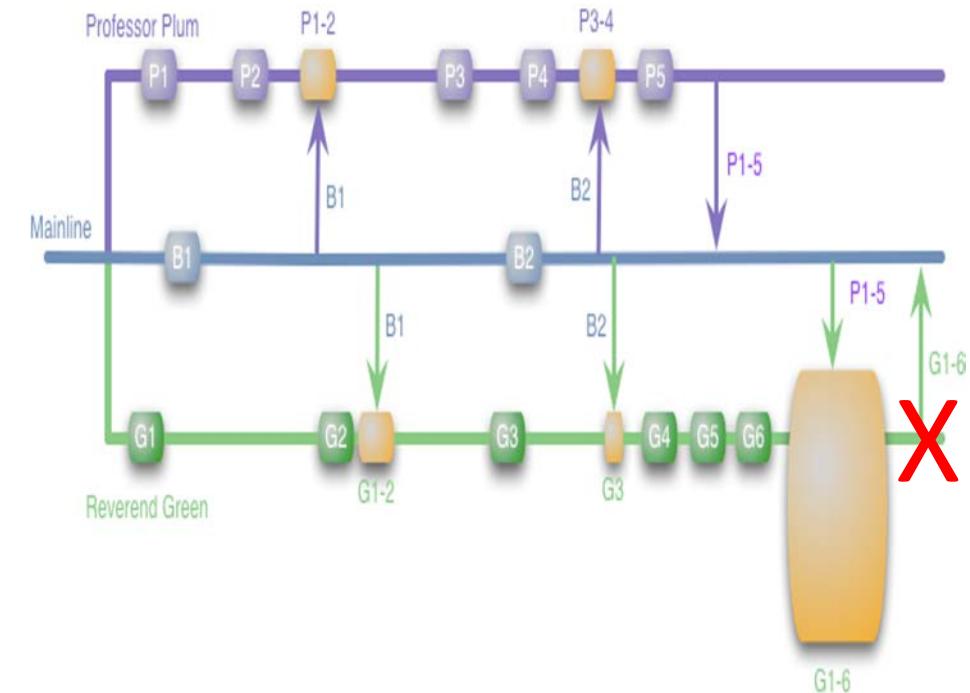


Source code, images and test versions in same branch structure ensures consistency



"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:

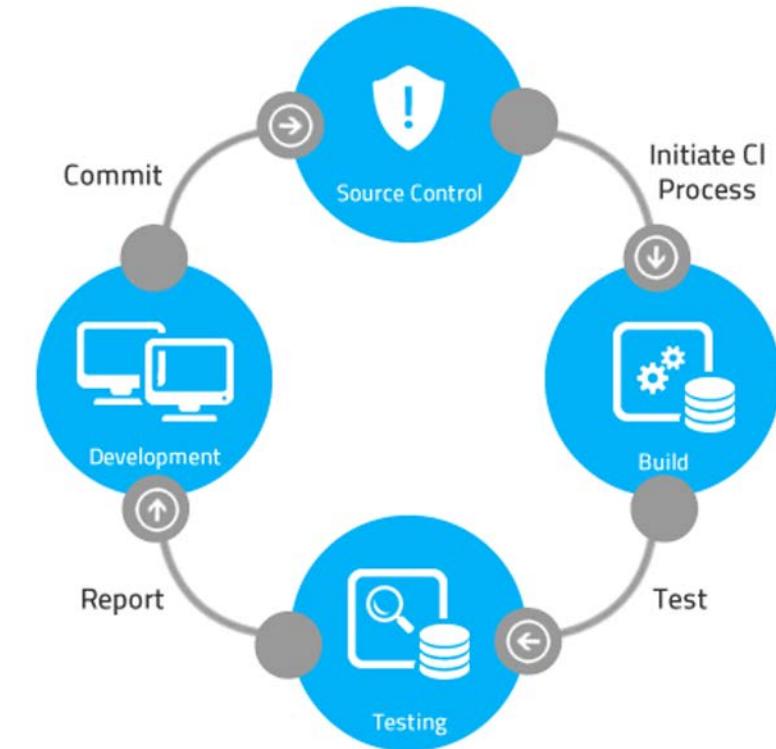
- Run static analysis
- Peer review source code
(E.g., Gitlab, Crucible, Collaborator, etc.)
- Run unit tests
- Run functional tests
- Pre-Flight tests, are performed in a test environment that is equivalent to the production environment of customer deployments.



Value of Continuous Integration

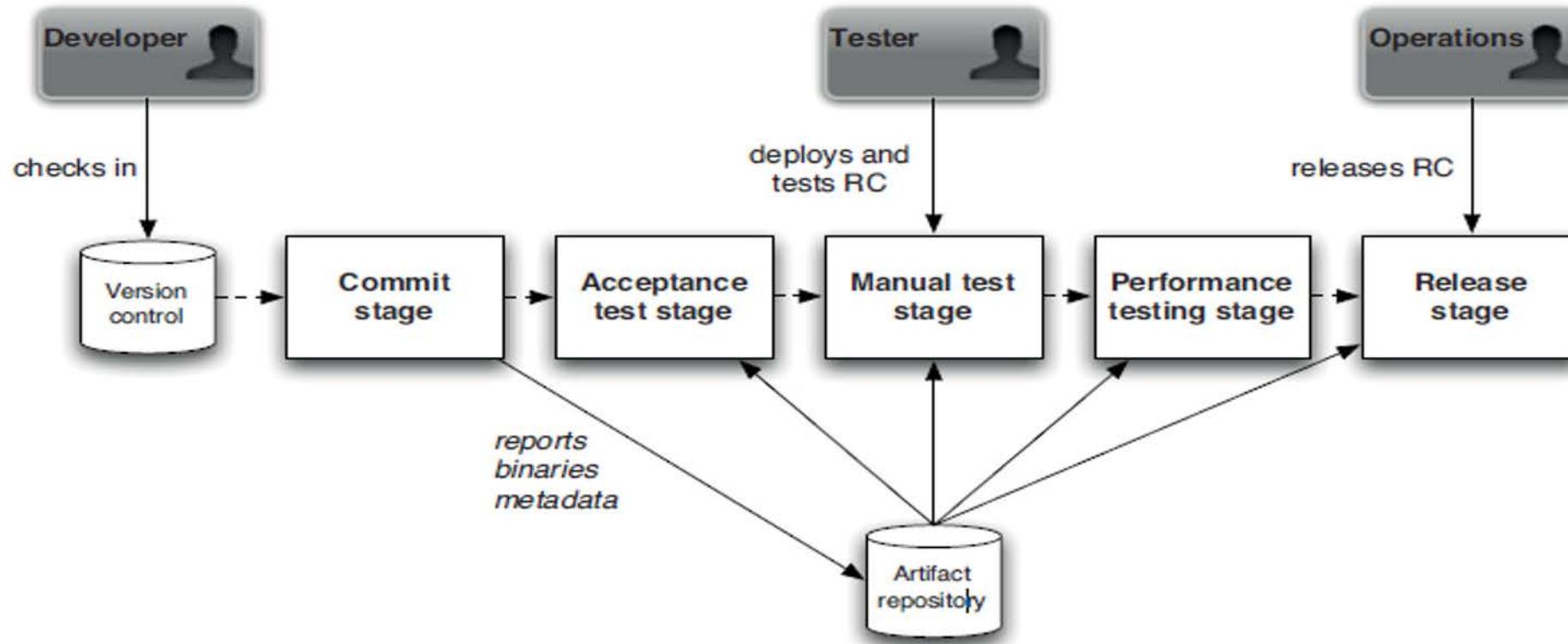
Frequent, small, incremental integrations have many merits

- 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



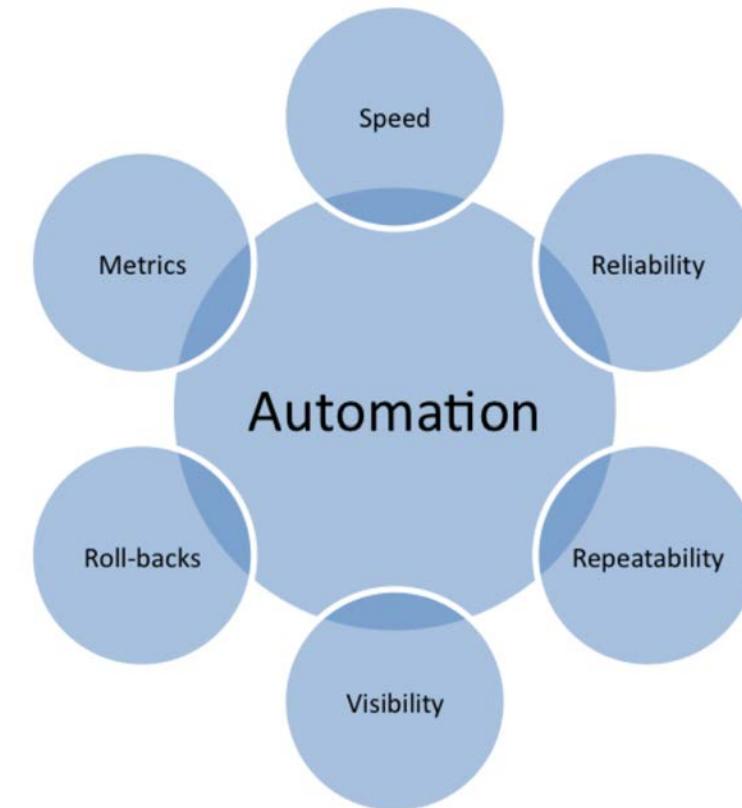
Use an artifact repository to store binaries, reports, and metadata for each of your release candidates (E.g., Archiva, Nexus, JFrog).



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

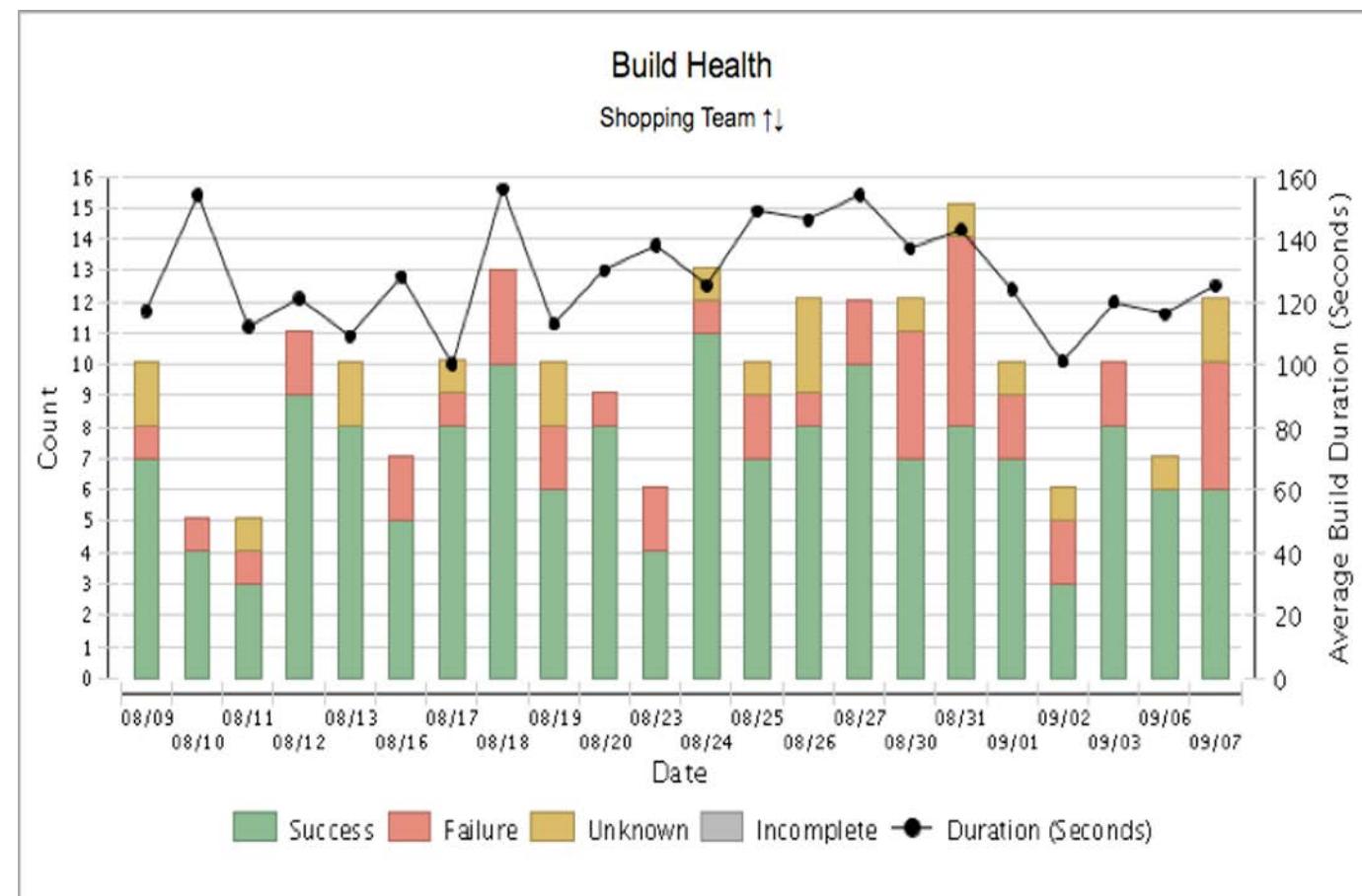




Reasons to Fail a Build

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

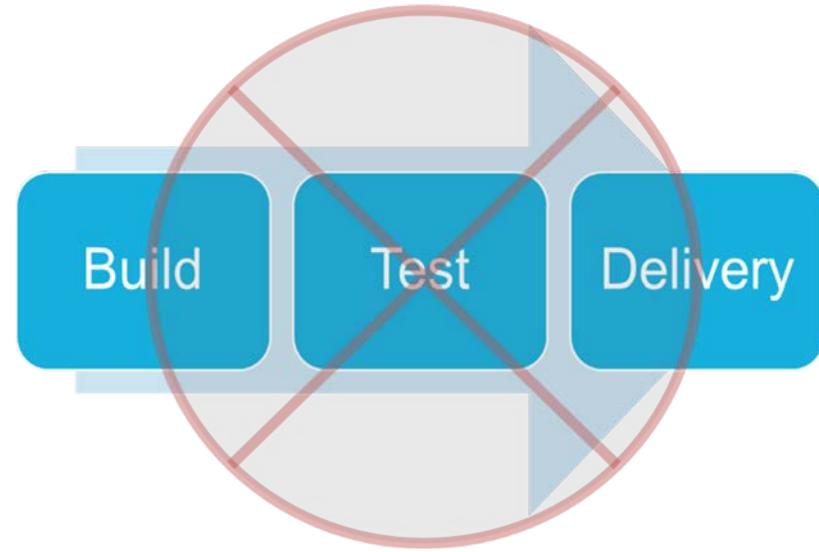


Myth

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.



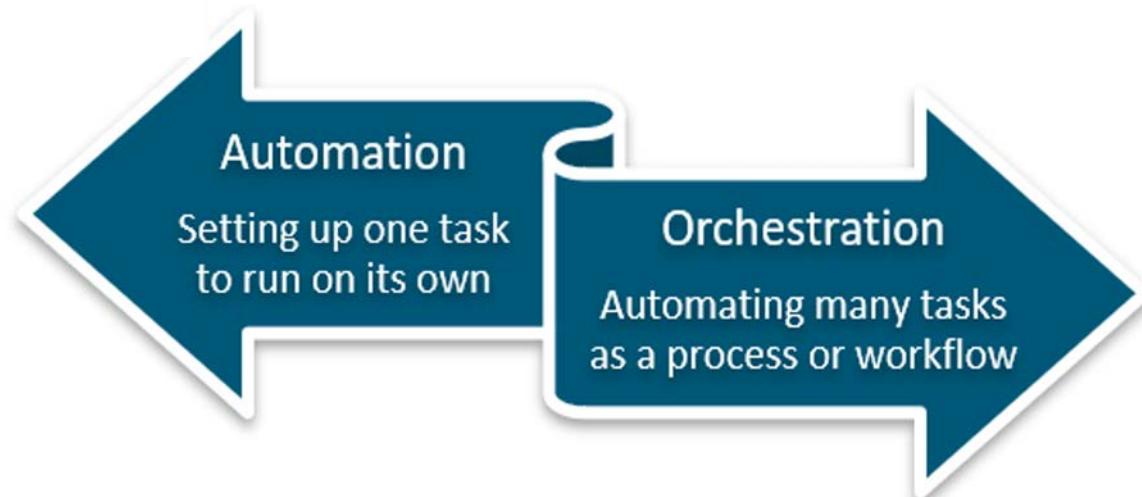
Test Environment Orchestration and Test Automation



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

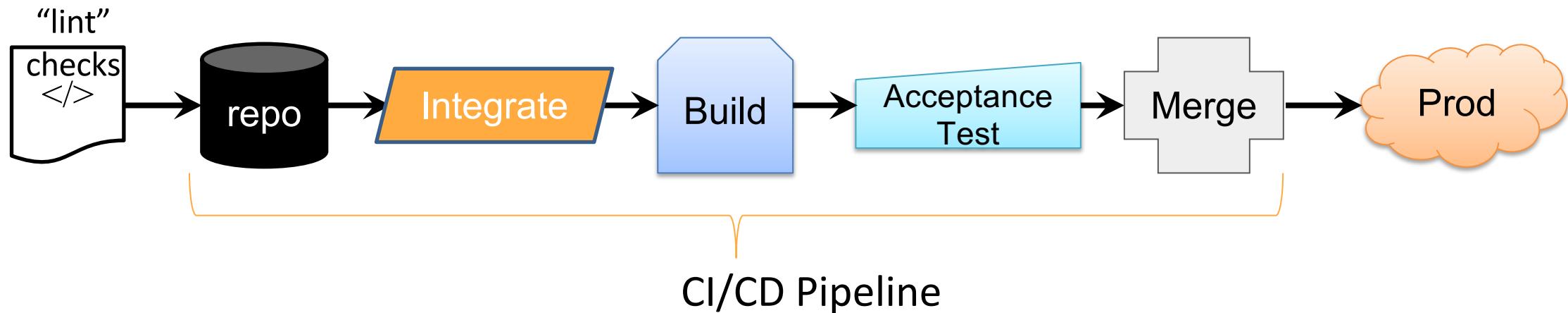
Environment Orchestration: Automatically setup (and **reset**) the test environment and resources (physical and virtual) to match the requirements of a test.

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

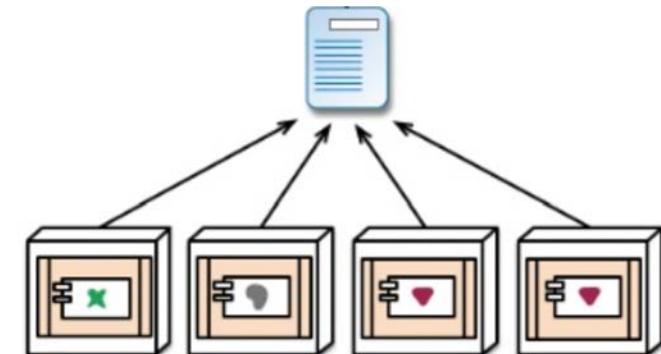




Microservices need to work together and alone

Recommended integration tests

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

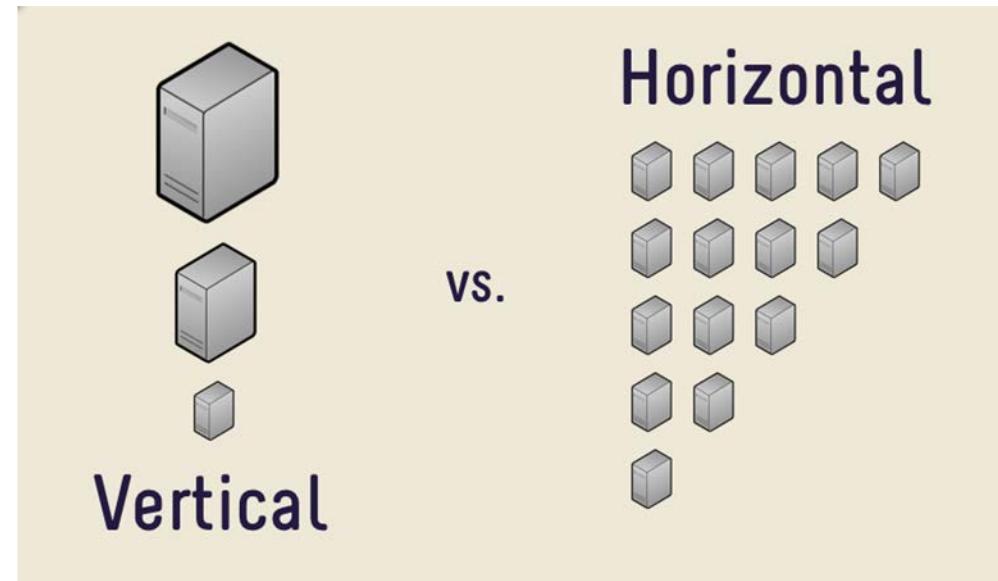




Optimizing CI Workflows – CI Infrastructure



- 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



The Periodic Table of DevOps Tools (V4.2)

		AI/Analytics		Cloud		Collaboration		Configuration Automation		Containers		Continuous Integration		Database Management		Deployment		Enterprise Agile Planning		Issue Tracking/ITSM		Release Management		Security		Serverless/PaaS		Source Control Management		Testing		Value Stream Management																																																																																																																																																																																																																																																																																																																																																																																																																																																											
1	En	Aja	Atlassian Jira Align	2	Os	Gi	Git	3	En	Daa	Digital Agility	4	En	Tp	Targetprocess	5	En	Azp	Azure DevOps Pipelines	6	Os	Ow	OWASP ZAP	7	En	Dap	Digital.ai App Protect	8	En	Dar	Digital.ai Release	9	En	Acp	AWS CodePipeline	10	Fm	Gh	Github	11	En	Pv	Planview	12	En	Br	Broadcom Rally	13	En	Dad	Digital.ai Deploy	14	En	Sni	Sonatype Nexus IQ	15	En	Aq	Aqua Security	16	En	Cfr	CloudBees Flow	17	En	Brl	BMC RLM	18	Os	Gls	GitLab SCM	19	Pd	In	Instans	20	En	Dd	Datadog	21	En	Ja	JFrog Artifactory	22	En	Aws	AWS	23	En	Sl	Slack	24	En	Mt	Microsoft Teams	25	Os	Rha	Red Hat Ansible	26	Os	Ht	HashiCorp Terraform	27	Os	Dk	Docker	28	En	Rho	Red Hat OpenShift	29	En	Lb	Liquibase	30	Os	Dp	Delphix	31	En	Ud	UrbanCode Deploy	32	Os	Ck	CyberArk Conjur	33	Os	Hv	HashiCorp Vault	34	En	Ur	UrbanCode Release	35	En	Al	AWS Lambda	36	Os	Ab	Atlassian Bitbucket	37	En	Sp	Splunk	38	En	Ad	AppDynamics	39	Os	Snx	Sonatype Nexus	40	En	Az	Azure	41	En	Gc	Google Cloud	42	En	Ac	Atlassian Confluence	43	Os	Ch	Chef	44	En	Acf	AWS Cloud Formation	45	Os	Ku	Kubernetes	46	En	Ak	Amazon EKS	47	En	De	Docker Enterprise	48	Os	Id	IDERA	49	En	Ha	Harness	50	En	Vc	Veracode	51	Os	Sr	SonarQube	52	En	Ff	Micro Focus Fortify SCA	53	En	Azf	Azure Functions	54	En	Ci	Compuware ISPW	55	En	Dt	Dynatrace	56	En	Nr	New Relic	57	Fm	Dh	Docker Hub	58	En	Np	npm	59	En	Ic	IBM Cloud	60	En	So	Stack Overflow	61	Fm	Pu	Puppet	62	Os	Hc	HashiCorp Consul	63	En	Ae	Amazon ECS	64	En	Az	Azure AKS	65	Os	Ra	Rancher	66	Fm	Qt	Quest Toad	67	Os	Sk	Spinnaker	68	En	Od	Octopus Deploy	69	En	Sb	Synopsys Black Duck	70	En	Cx	Checkmarx SAST	71	Fm	He	Heroku	72	Os	Sv	Subversion	73	Os	Gr	Grafana	74	Os	EI	Elastic ELK Stack	75	Os	Yn	Yarn	76	Os	Nu	NuGet	77	Os	Os	OpenStack	78	Os	Mm	Mattermost	79	Os	Sa	Salt	80	Os	Hg	HashiCorp Vagrant	81	Os	Hp	HashiCorp Packer	82	En	Gk	Google GKE	83	Os	Hm	helm	84	En	Db	DBMaestro	85	En	Cfd	CloudBees Flow	86	En	Acd	AWS CodeDeploy	87	Os	Sn	Snort	88	Fm	Pbs	PortSwigger Burp Suite	89	En	Gf	Google Firebase	90	Os	Cf	Cloud Foundry	91	Os	Jn	Jenkins	92	En	Azc	Azure DevOps Code	93	Os	Glc	GitLab CI	94	Os	Tr	Travis CI	95	Fm	Cc	CircleCI	96	Os	Mv	Maven	97	Pd	Ab	Atlassian Bamboo	98	Os	Gd	Gradle	99	En	Acb	AWS CodeBuild	100	Os	Aj	Atlassian Jira	101	En	Bi	BMC Helix ITSM	102	Pd	At	Atlassian JFrog	103	En	Sw	ServiceNow	104	Pd	Td	TOPdesk	105	Os	Pd	PaperDuty	106	Fm	Tt	Tricentis Tosca	107	Pd	Nn	Neotys NeoLoad	108	Fm	Se	Selenium	109	Fr	Ju	JUnit	110	Pd	Ct	Compuware Topaz	111	En	Ap	Appium	112	En	Sq	Squash TM	113	Os	Cu	Cucumber	114	Fr	Jm	JMeter	115	Fr	Pa	Parasoft Digital	116	Pd	Dai	Digital	117	En	Tp	Tasktop	118	En	Pr	Plutora	119	En	Gl	GitLab
Os		Open Source		Fr		Free		Fm		Freemium		Pd		Paid		En		Enterprise																																																																																																																																																																																																																																																																																																																																																																																																																																																																									

digital.ai™
CollabNet VersionOne, XebiaLabs, Arxan, Numerity & Expertise are now Digital.ai



Building Your Toolchain



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



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

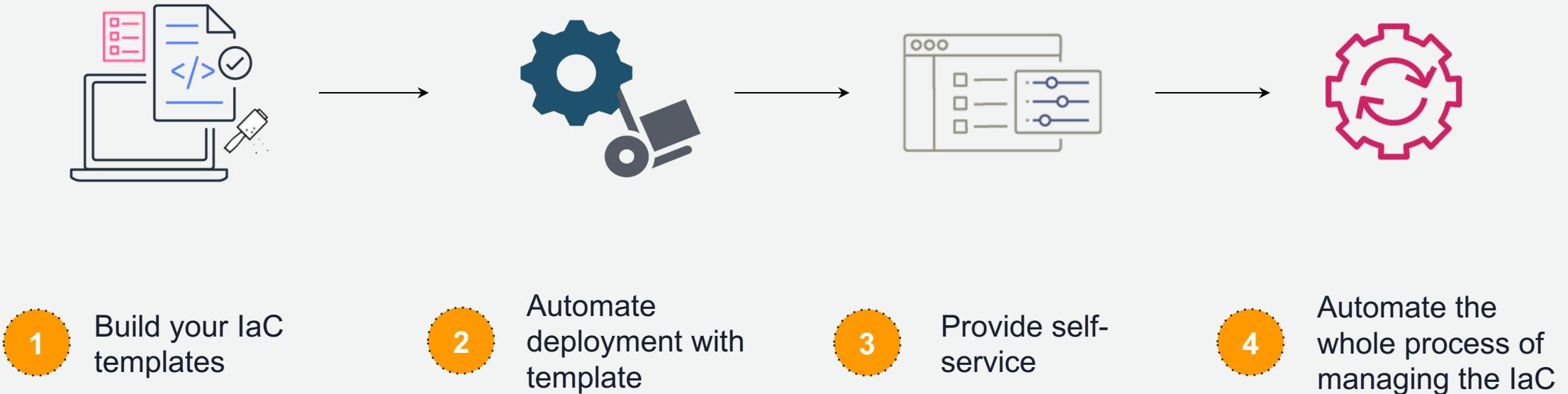


Chris Chapman

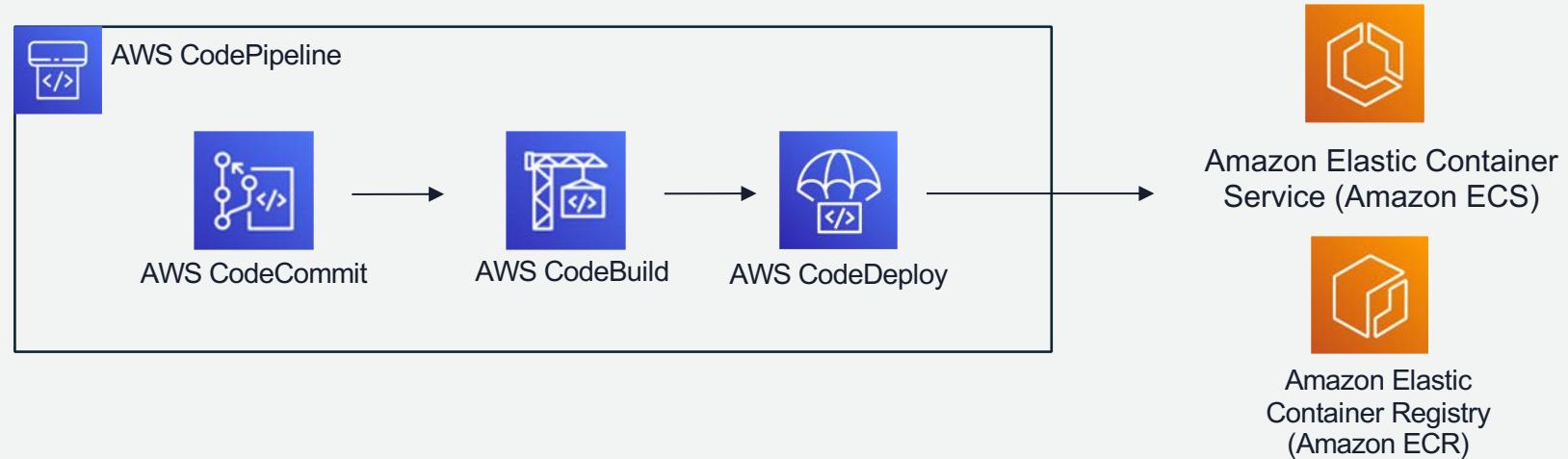
Sr. Specialist Solutions Architect at AWS
chapmanInTheCloud



Steps for automating DevOps pipelines



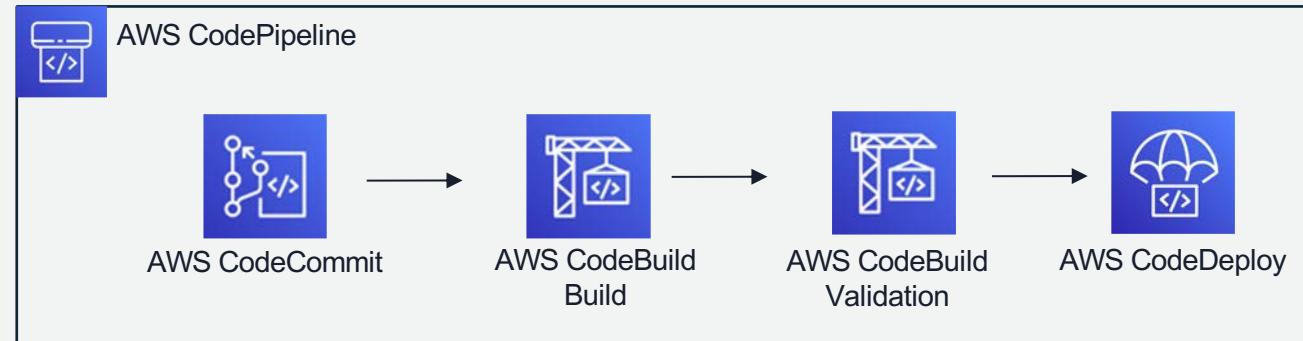
Start Simple



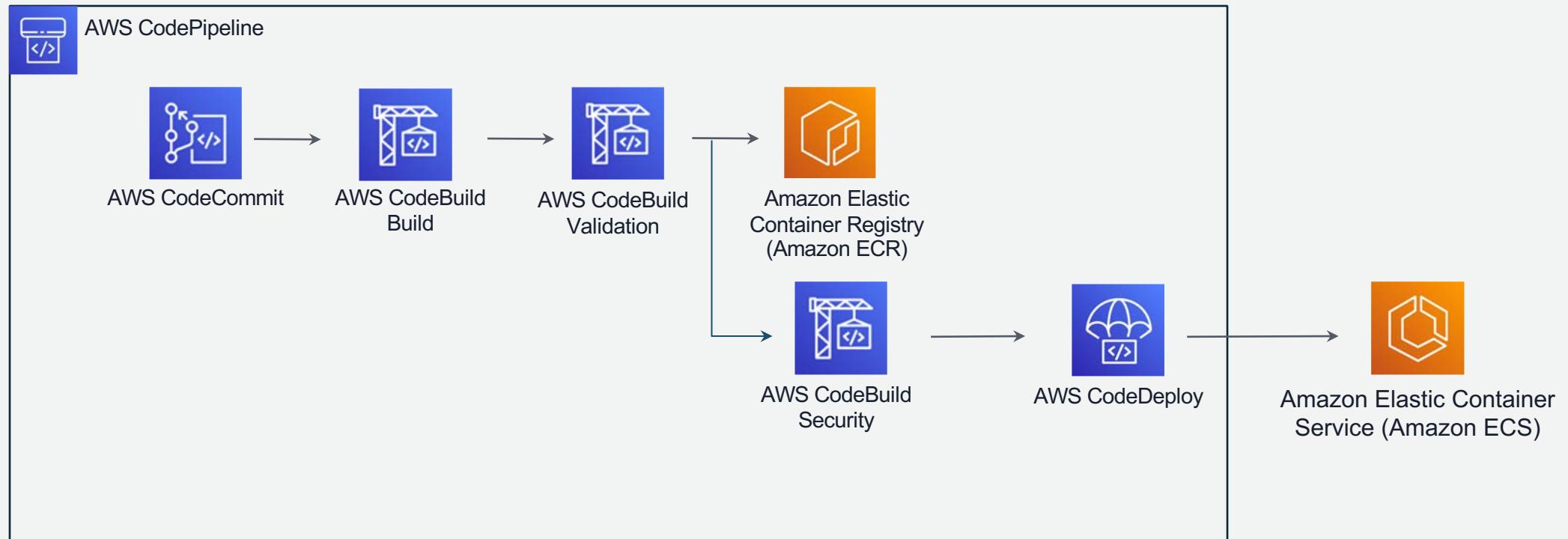
Buildspec files

```
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/$IMAGE_REPO_NAME:$IMAGE_TAG_LATEST
      - 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
```

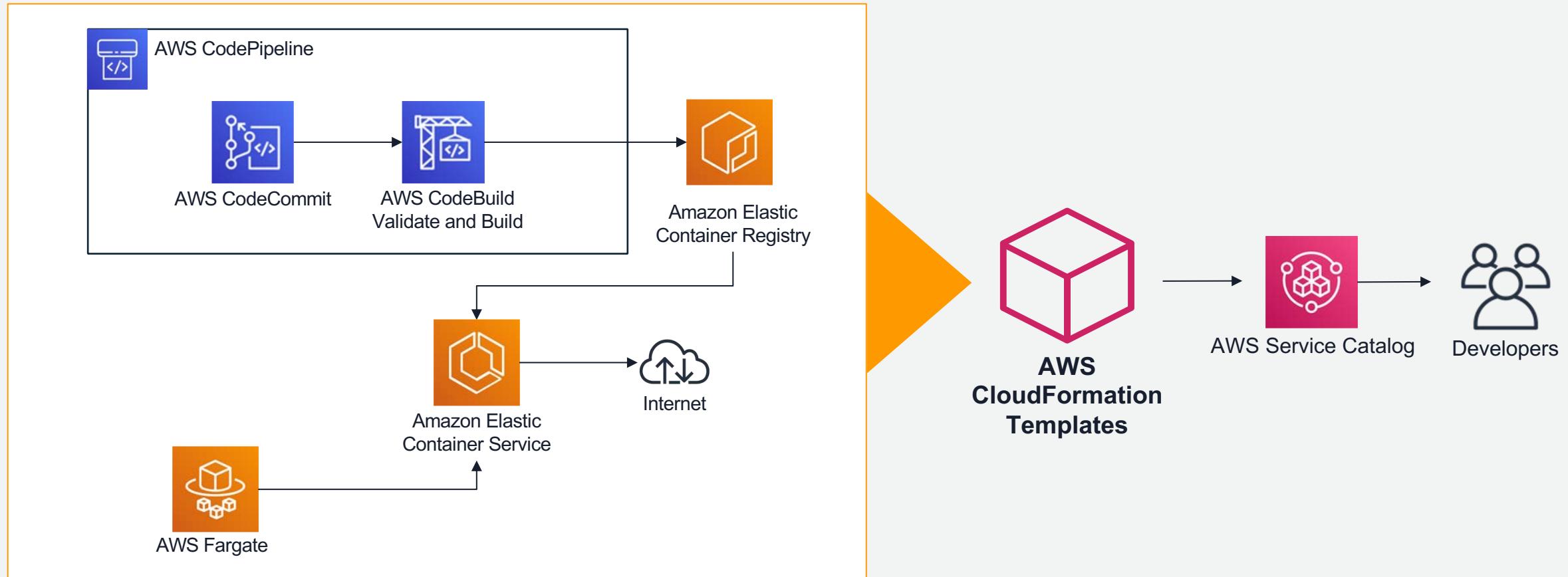
Incremental pipeline build – add validation



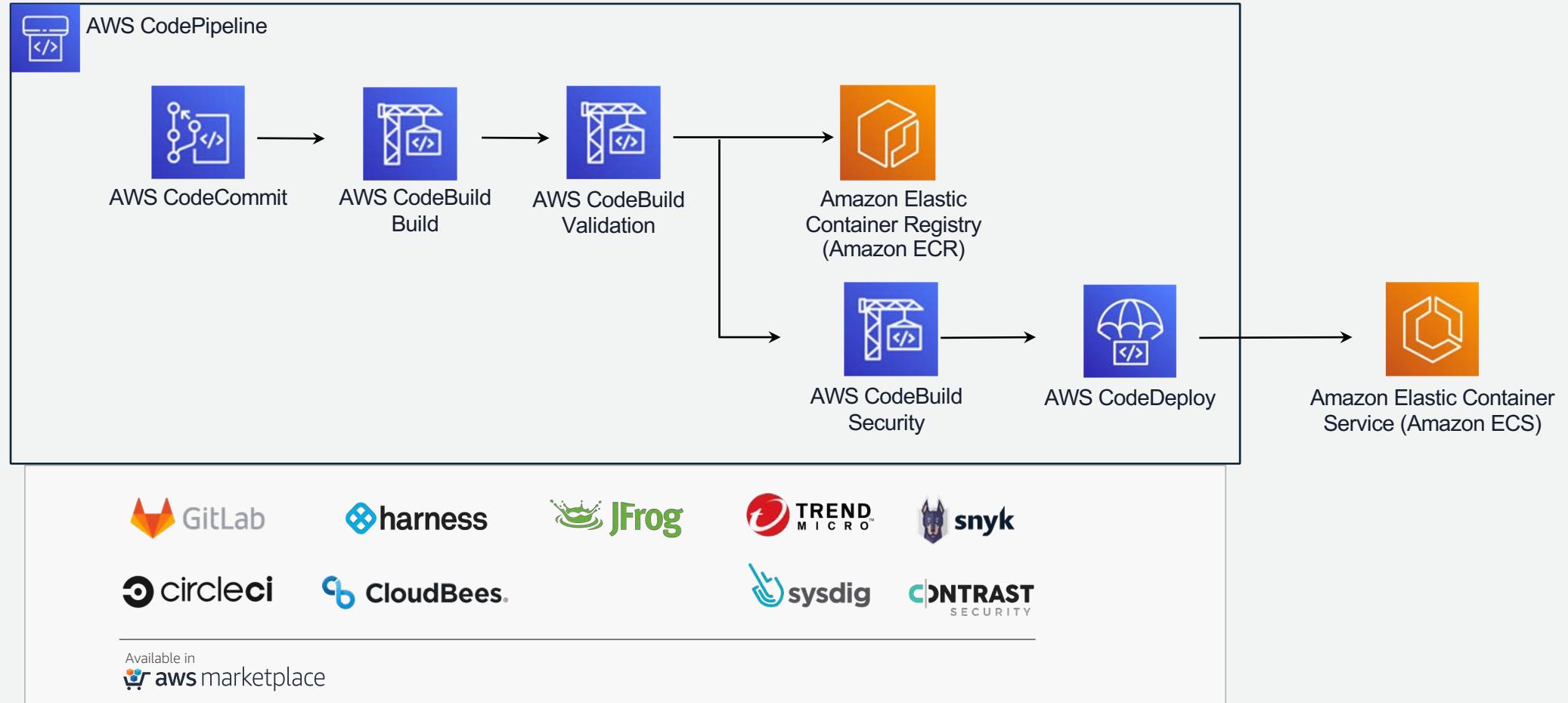
Incremental pipeline build – add security



Building a shared self-service platform



Enhance pipeline capabilities with AWS Partners





8,000+
listings

1,600+
ISVs

24
regions

290,000+
customers

1.5M+
subscriptions

AWS Marketplace DevOps Workshop Series participating partner hands-on labs



*And more
coming soon!*

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

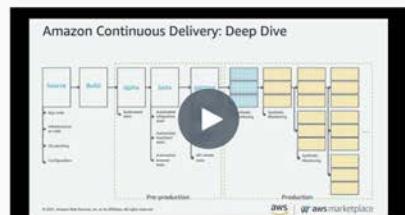
Module 2 Hands-on Labs

The screenshot shows the AWS Marketplace DevOps Workshop Series page for "CI/CD Pipelines Hands-on Labs". It features a dark background with orange 3D cube icons. The title "CI/CD Pipelines Hands-on Labs" is prominently displayed, along with the subtext "Learn from expert presenters and get hands on experience with some of the best tools in DevOps". Below this, there's a section titled "Step 1: Watch the workshop presentation recording" which includes a video thumbnail for "Amazon Continuous Delivery: Deep Dive" and a link to "Workshop Slides: Module 2: CI/CD Pipelines Slides".

Step 1: Watch the workshop presentation recording

The presentation will ground you in DevOps principals and practical approaches you can then apply through doing self-guided labs in step 2.

Module 2 Presentation:
CI/CD Pipelines



[Workshop Slides: Module 2: CI/CD Pipelines Slides](#)



Step 2: Get hands on experience with some of the best CI/CD tools in DevOps

Watch a quick demo to see which tool you'd like to get started with then complete a tutorial at your own pace. You'll receive a sharable proficiency badge for each tool you master!

CircleCI
CI/CD delivery platform that enables teams to orchestrate complex workflows and automate software delivery at scale.
[Watch a demo >](#)
[Start your hands-on lab >](#)

GitLab
Source code management, CI/CD, monitoring, and more all in a single application to enable concurrent DevOps.
[Watch a demo >](#)
[Start your hands-on lab >](#)

JFrog
Scans for security risks and compliance, providing automation and management of binaries and artifacts.
[Watch a demo >](#)
[Start your hands-on lab >](#)

<https://pages.awscloud.com/awsm-p-wsm-dev-workshop-series-module2-cicd-pipelines-ty.html>

The screenshot shows three stacked screenshots of the AWS DevOps Modernization Workshop interface. The middle screenshot is specifically for the JFrog section, showing a navigation menu with options like "Introduction", "Build and Put", "View Results", "Deploy on An", and "Conclusion". The top and bottom screenshots show similar navigation menus for other sections of the workshop.



DEVOPS MODERNIZATION WORKSHOP

Welcome

In this workshop you will learn about the JFrog Platform and how to leverage Artifactory and Xray for managing your Software Development Lifecycle (SDLC) and bring DevOps to the cloud on AWS.

Learning Objectives



Move on to Module 3: Evolving to Continuous Deployment

Choose a module to get started

In each module you will join an instructor-led presentation from AWS and an ambassador of the DevOps Institute. Following the presentation you'll be able to choose a hands-on lab to complete from a selection of the best tools in DevOps. **Pick a module to get started** and **subscribe to email updates** to learn when new content is available.

Module 1



Practicing DevOps

In this first presentation you'll get an overview of the workshop series and receive practical instruction on how to build the right foundation for a successful DevOps practice in AWS.

[Watch now >](#)

▼

Module 2



CI/CD Pipelines

In this module you'll learn how to implement a well-engineered CI/CD pipeline that considers governance and provides traceability from idea to production.

[Register now >](#)

▼

Module 3



Evolving to Continuous Deployment

Deploying code changes live into production is still a terrifying prospect for many organizations. We'll dive deep into how using the right processes and tools can make this safe and advantageous.

[Register now >](#)

Hands-on labs:

 GitLab  CloudBees.
 harness  armory
 LaunchDarkly  JFrog

▲



Presentation: Sept 15, 2021

elements of your applications interact and perform, when and where issues arise, and how to fix and prevent them.

[Get updates >](#)

[s.awscloud.com/awsmpl-wsm-dev-workshop-series-module3-evolving-to-continuous-deployment.html](https://pages.awscloud.com/awsmpl-wsm-dev-workshop-series-module3-evolving-to-continuous-deployment.html)

© 2021, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

 |  aws marketplace