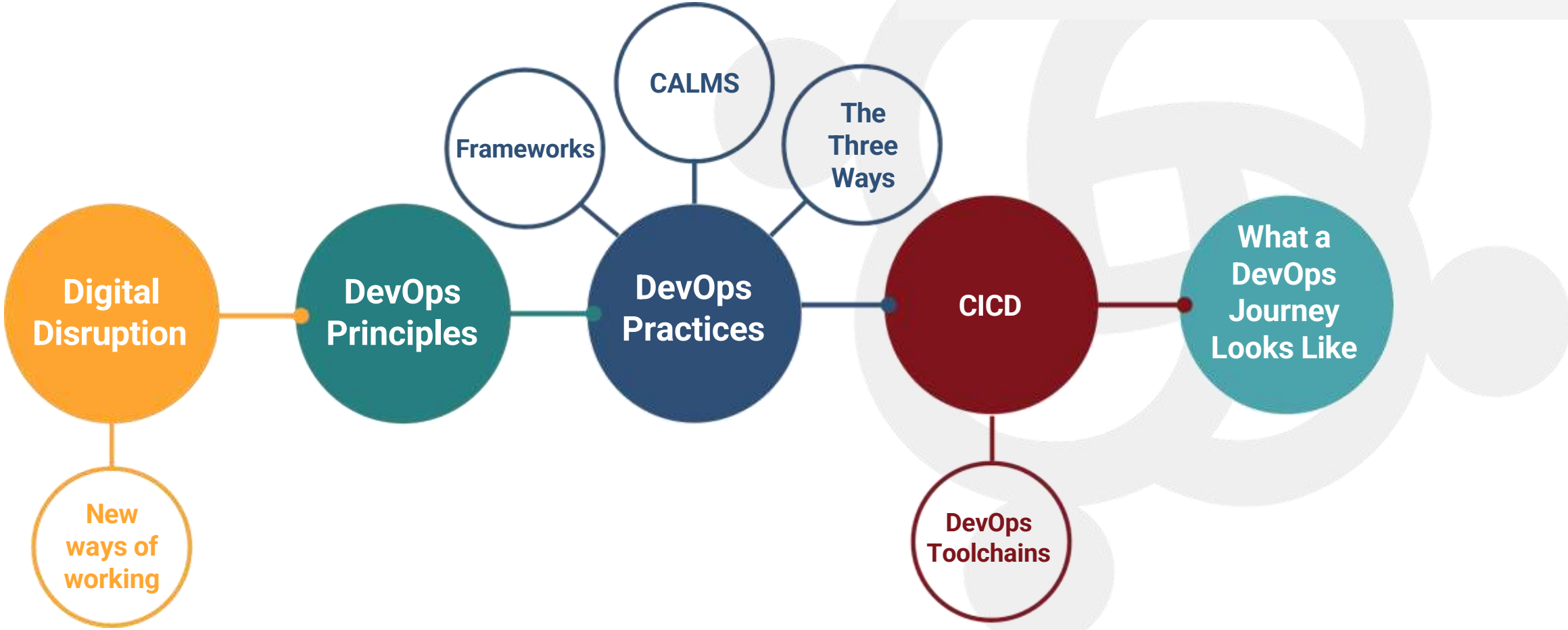
DevOps Workshop Series

Practicing DevOps

Flow: Talk Map

You will learn:

- How DevOps influences organizational, team and system design in cloud
- Why value stream centric thinking is essential to achieve continuous compliance
- How CICD and DevOps toolchains accelerate value outcomes and support safety engineering



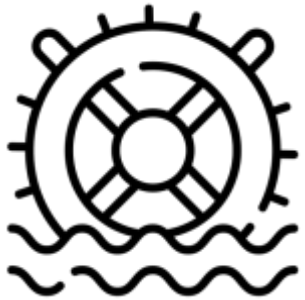
Digital Disruption

The 5th Technology Revolution

- Enterprises have young, nimble start-up competitors
- Agile software development and cloud infrastructure is increasing
- IT can no longer operate in a silo culture
- More organizations are migrating to the cloud
- Consumers have “app” mentalities and expectations
- There is more data available to the business
- Time to value must accelerate

To meet these changing conditions, IT must adapt its culture, practices and automation to be more ‘continuous’.

1



The industrial revolution

1771

2

The age of steam and railways

1829

3

Age of steel, electricity and heavy engineering

1875

4

Age of oil, automobiles and mass production

1908

5

Age of information and telecomms

1971

1

Steam engines

18th -19th century

2

Steel, oil, electricity, combustion engines

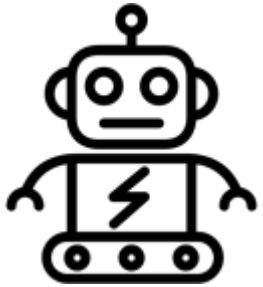
1870-1914

3

Digital revolution

1980’s

4



AI, big data, robotics, IoT, blockchain and crypto

21st Century

5

Connection of frontier tech to purpose and inclusivity

21st Century

Icons made by Freepik, Goodware and Eucalyp from www.flaticon.com

New Ways of Working

Better, sooner, faster, safer, happier

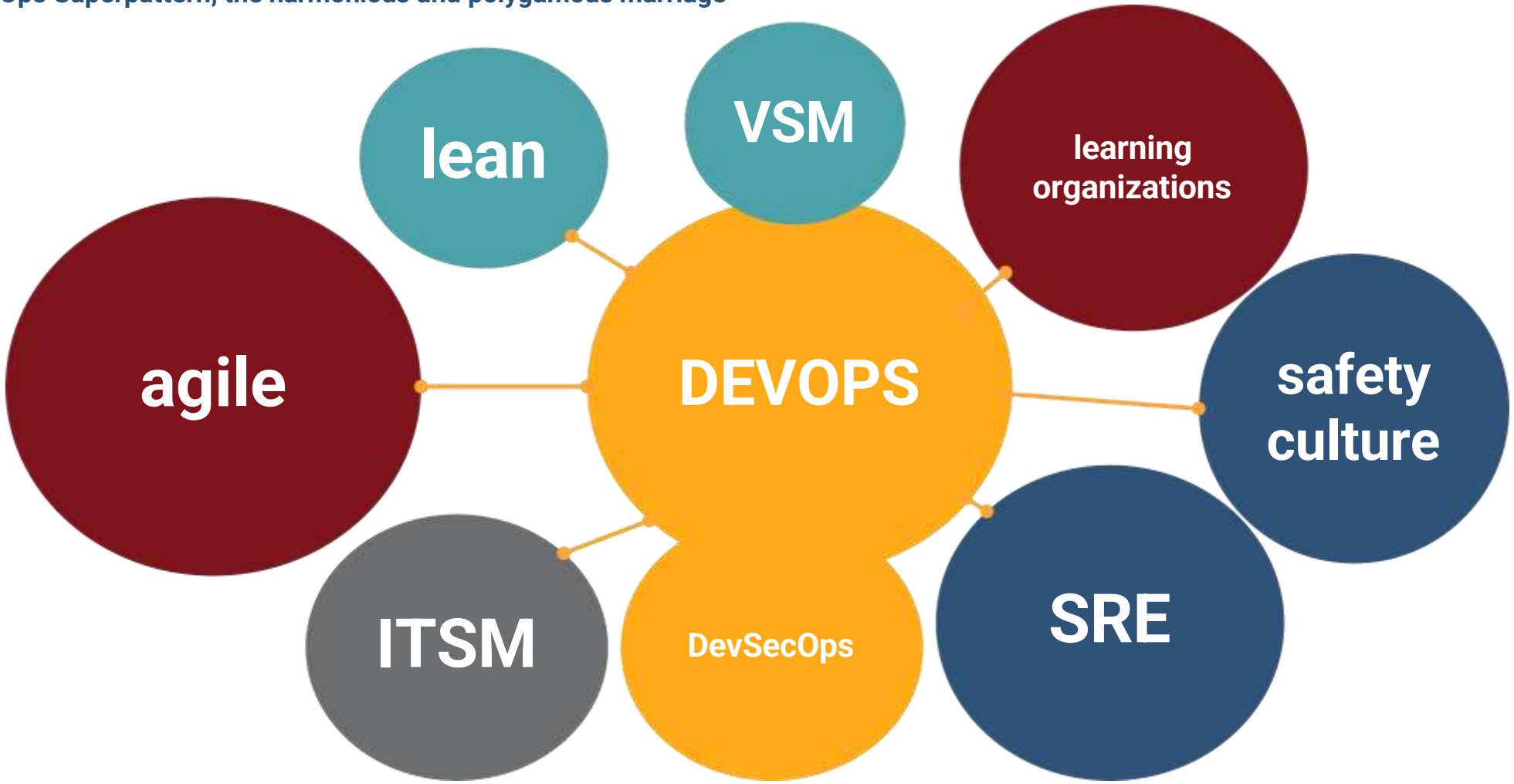
Dimension	Traditional IT	DevOps
Batch size	Large & Monolithic	Micro & Loosely Coupled
Organization	Skill Centric Silos	Autonomous squads
Scheduling	Centralized	Decentralized & Continuous
Release	High Risk Event	“Like Breathing”
Information	Disseminated	Actionable
Culture	Do Not Fail	Fail Early
Metric	Cost & Capacity	Flow
‘Definition of Done’	“I did my job.”	“The customer has received value”

Adapted from an original article by Mustafa Kapadia

How does cloud help us with these transitions?

Frameworks

The DevOps Superpattern, the harmonious and polygamous marriage



CALMS

An elevator acronym to describe DevOps

- **C** CULTURE
- **A** AUTOMATION
- **L** LEAN
- **M** MEASUREMENT
- **S** SHARING

John Willis

Damon Edwards

Jez Humble

10 YEARS DEV OPS DAYS GHENT 2019

Culture	Automation	Lean	Measurement	Sharing
Organizational purpose has clarity	Goal is to be high performing IT and organization	Focus is on the customer	High level goals linked to PBIs	Transparency and clarity throughout the organization
Authority is distributed, teams have autonomy	Loosely coupled systems	Value stream centric thinking	Teams measure themselves	Teams reward each other for collaboration
Failure is a learning opportunity	'Shift left', fast feedback	Focus is on removing waste	Data driven decision making	Stories are shared -good AND bad
Leaders are transformational	Observability leads to discovery leads to improvement	Work is visible	Measurements used to drive experiments to inspect and adapt	Leaders do not punish failure but globalize local learnings

Transformational Leadership

Distributing authority, breaking down silos: “We build it, we own it”

The characteristics of transformational leadership are highly correlated with IT performance and employee Net Promoter Score (eNPS).

From The State of DevOps Report 2017

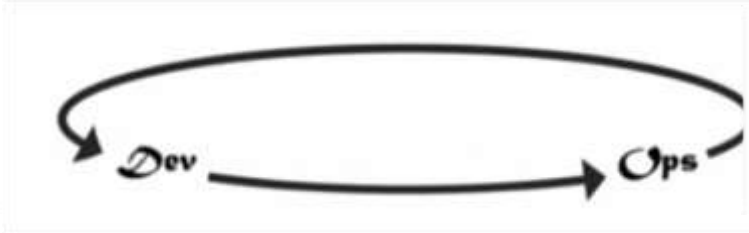
"The goal of leadership is not to command, control, berate, intimidate, and evaluate workers through some set of contrived metrics. Instead, the job of leaders is to help organizations become better at self-diagnosis, self-improvement, and to make sure that local discoveries can be translated and converted to global improvements.”

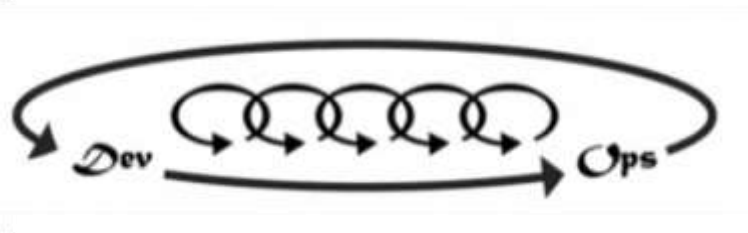
Dr Stephen Spear cited by Gene Kim in Beyond the Phoenix Project

“You build it, you run it” originated with Werner Vogels. Let’s talk about it!

The Three Ways

Key principles of DevOps as featured in The Phoenix Project



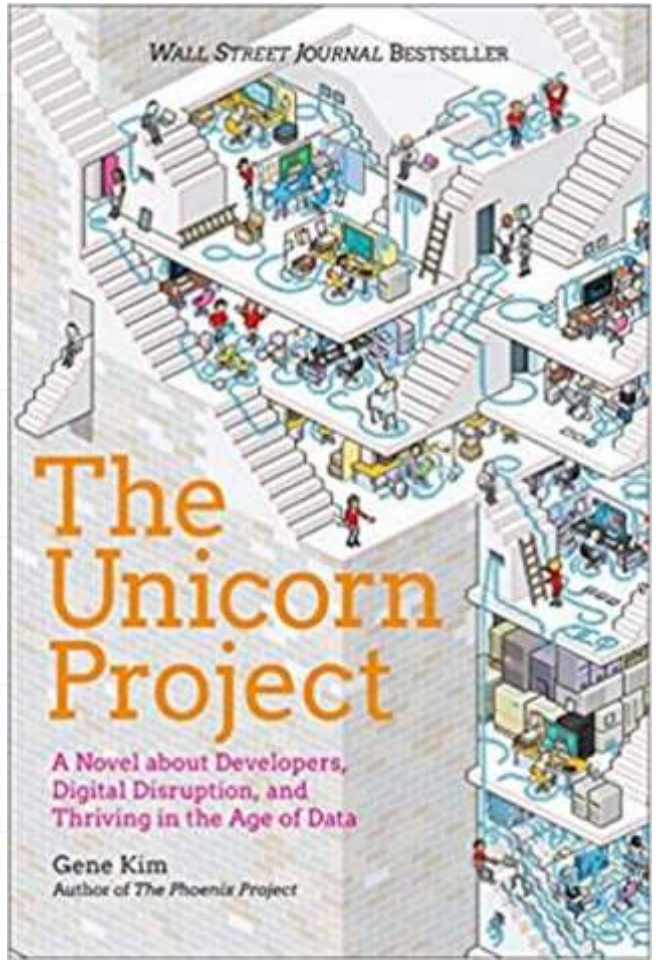


The First Way	The Second Way	The Third Way
Flow	Feedback	Continuous Experimentation & Learning
Understand and increase the flow of work (left to right)	Create short feedback loops that enable continuous improvement (right to left)	Create a culture that fosters: • Experimentation, taking risks and learning from failure • Understanding that repetition and practice is the prerequisite to mastery

The Five Ideals

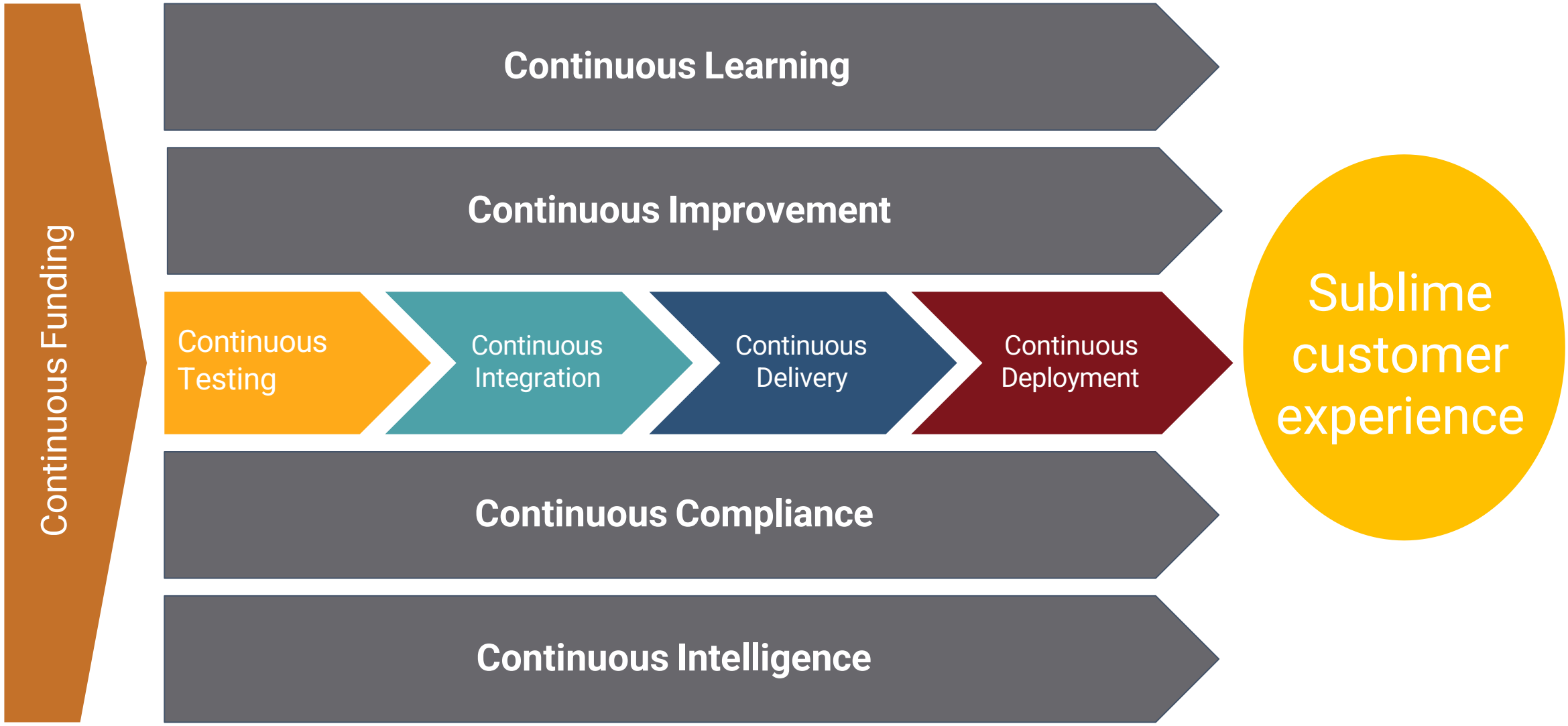
As featured in The Unicorn Project

The First Ideal	Locality and Simplicity
The Second Ideal	Focus, Flow, and Joy
The Third Ideal	Improvement of Daily Work
The Fourth Ideal	Psychological Safety
The Fifth Ideal	Customer Focus



DevOps Practices

All the continuouses



CI/CD

"In short, CI/CD toolchains help with velocity and quality of code, allow for better collaboration among the teams and automates many steps, tasks and processes which reduced the risk and cost of software

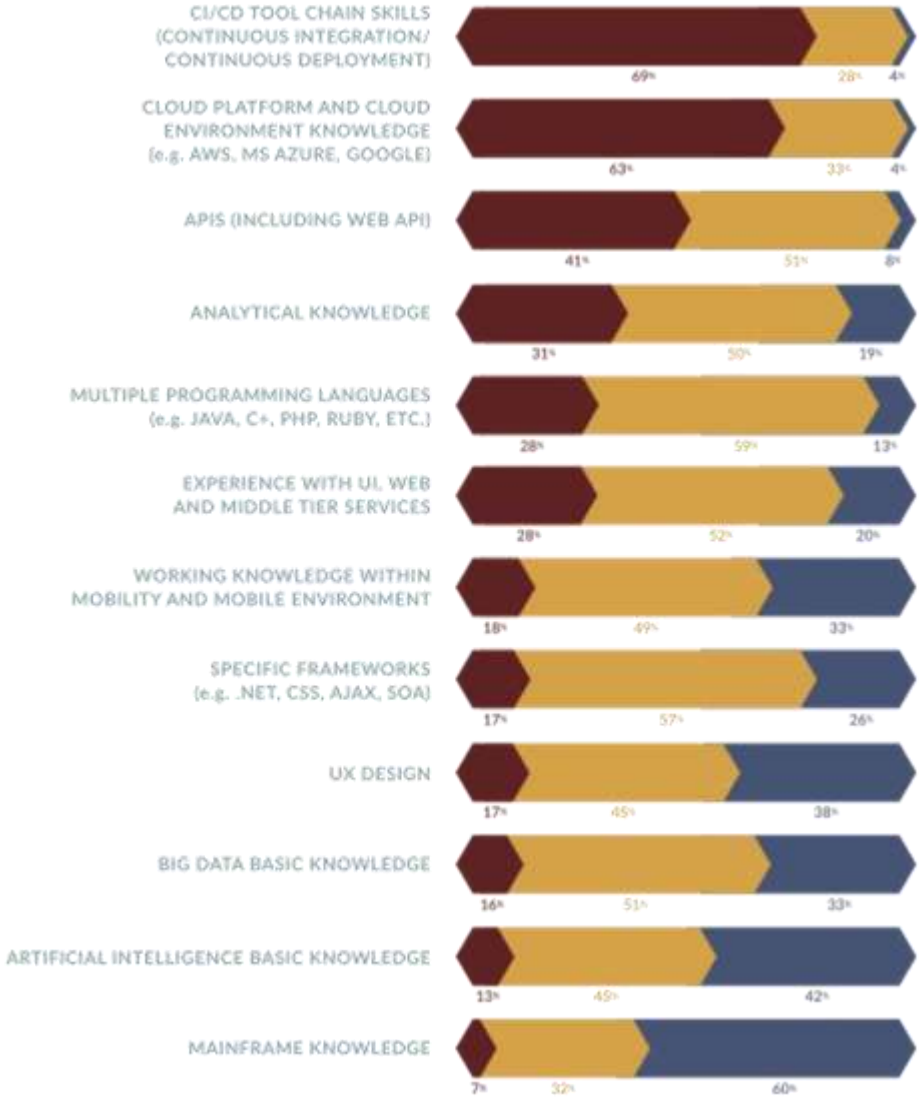


Figure 13: Technical Skills For The DevOps Human CI/CD Toolchain, Cloud Platform And Understanding APIs Are The Top 3 Must-have Technical Skills

Q How would you rate the importance of the following technical skills for your DevOps team members?

- Very Important (Must-Have Skills)
- Important (Nice-to-Have Skills)
- Not Important (Optional Skills)

N 447

Delta from 2019: New category CI/CD rose to the top.

Delta from 2019: Analytical knowledge, cloud platform, specific frameworks (.NET, etc.), multiple programming languages gained must-have votes since 2019.

Delta from 2019: Mainframe skills are still must-have but declined from 11% to 7%.

UPSKILLING 2020 | Enterprise DevOps Skills Report

DevOps Institute

Continuous Integration

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

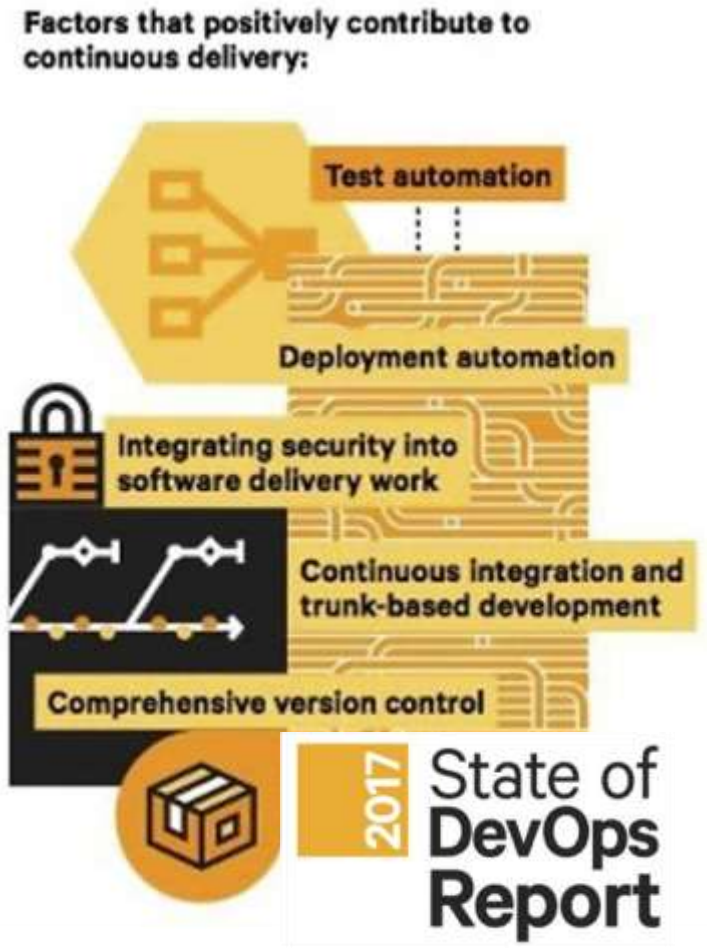
- All developers check code in at least daily to trunk
  - Trunk based development
- Each check-in is validated by
  - An automated build
  - Automated unit, integration and acceptance tests
- Is dependent on consistent coding standards
- Requires version control repositories and CI servers to collect, 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

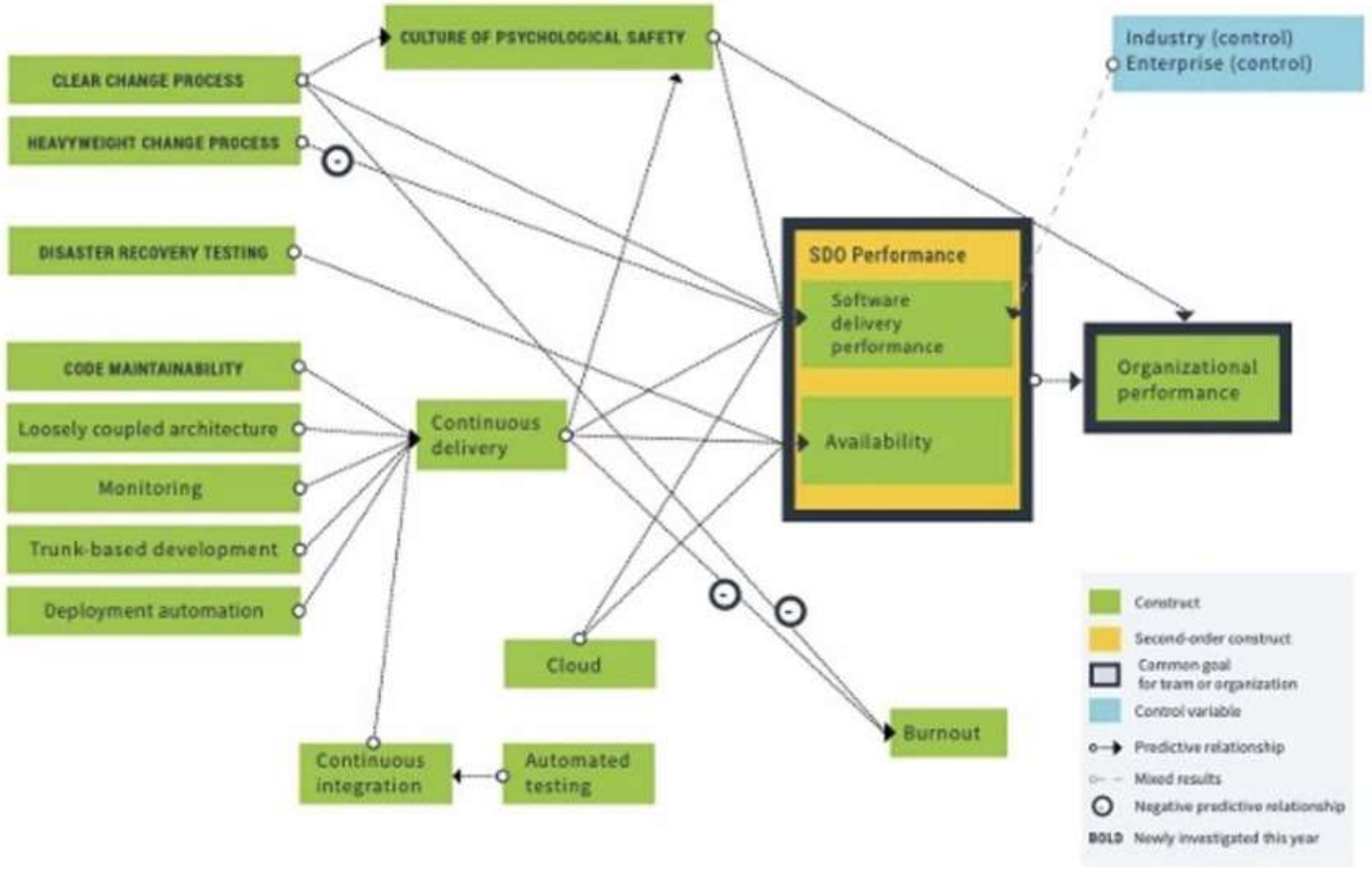
Software is always in a releasable state - ready to go, at the push of a button

- 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

Leads to higher organizational performance



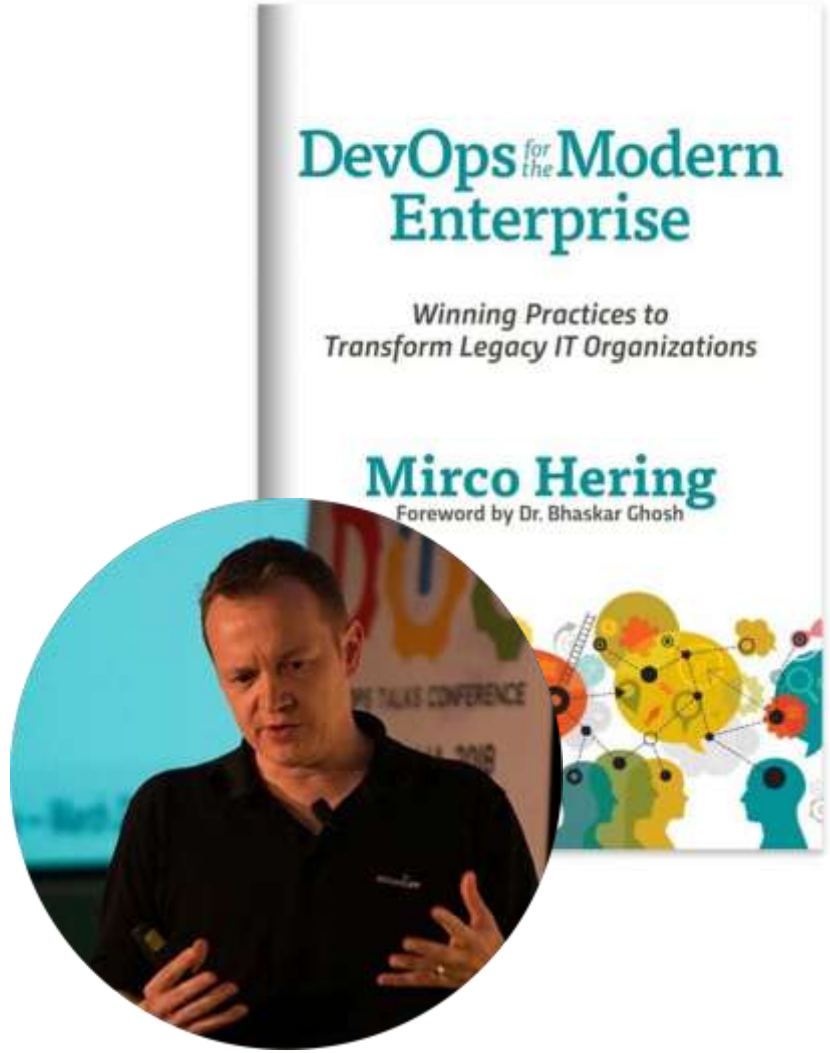
Continuous Deployment

Continuous Delivery

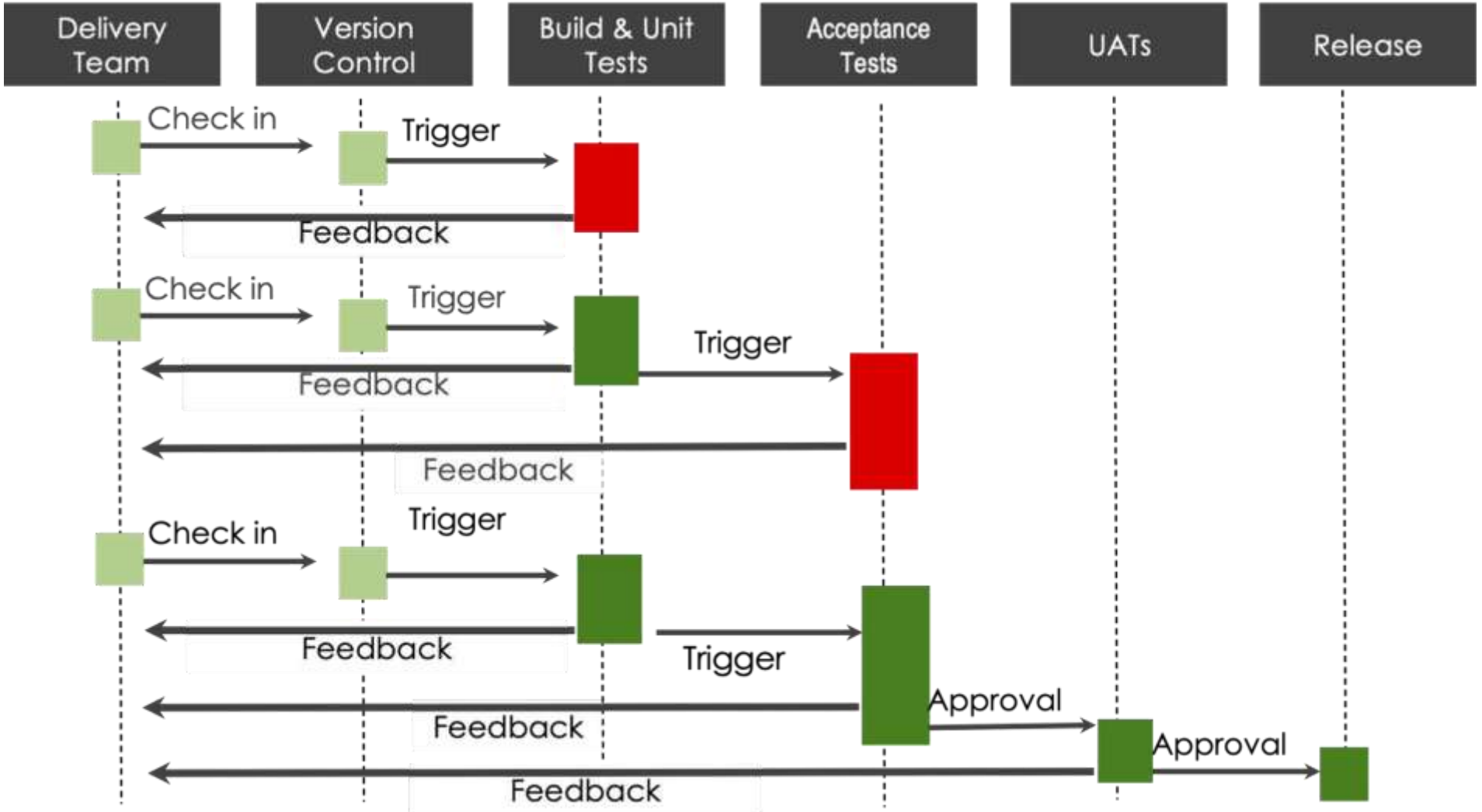
Manual trigger

Automatic trigger

Continuous Deployment

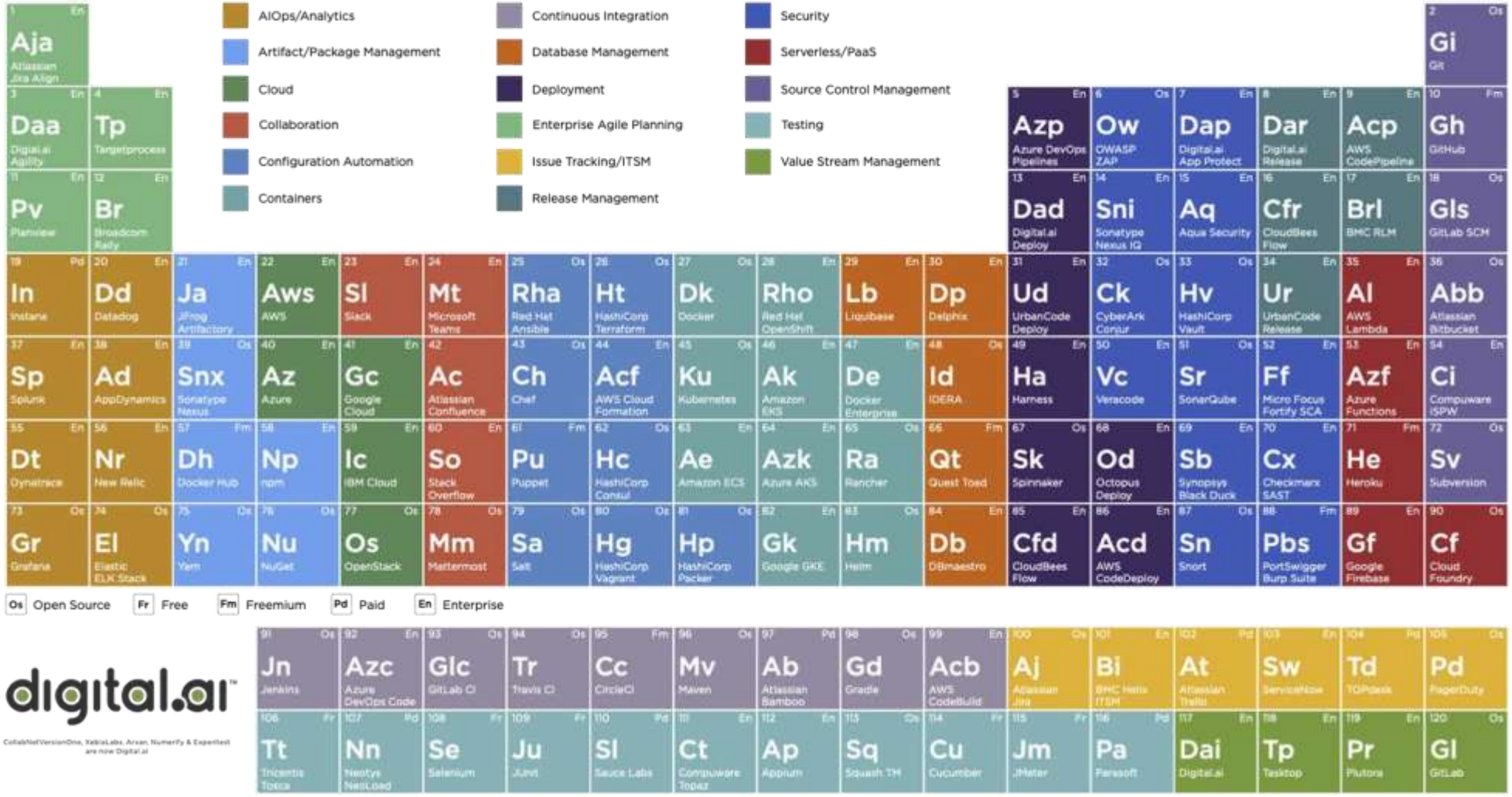


The Deployment Pipeline

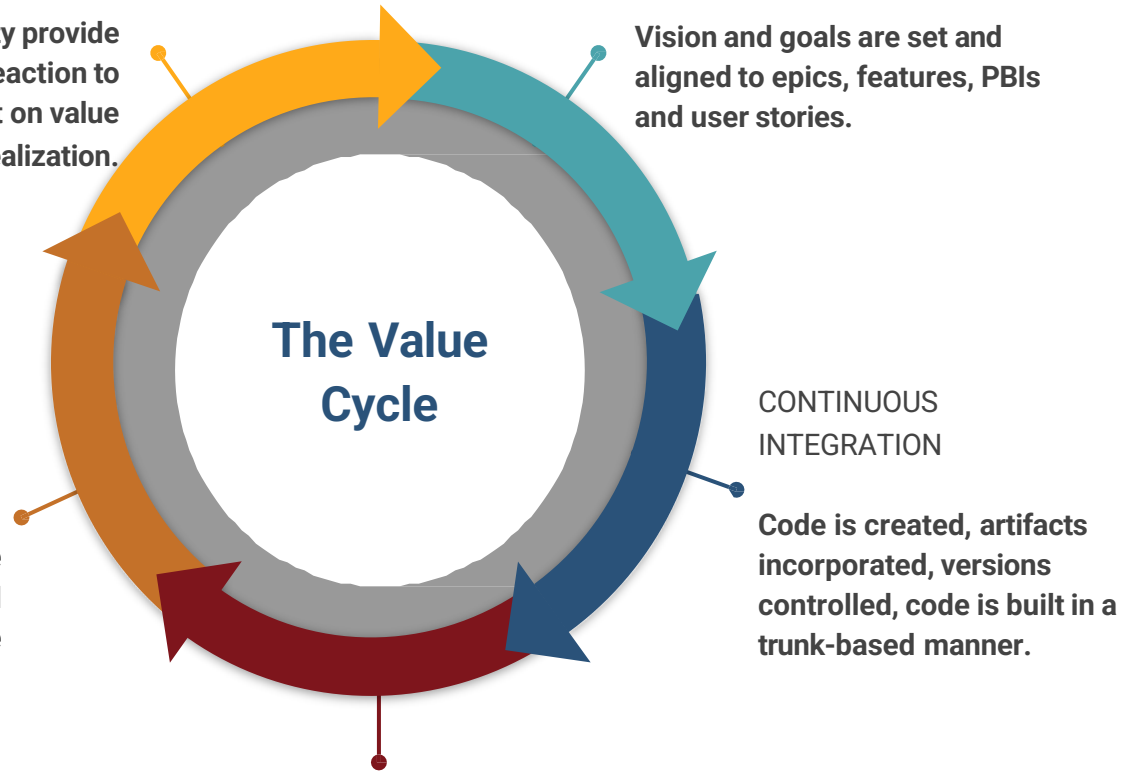


DevOps Toolchains

The Periodic Table of DevOps Tools (V4.2)



The Value Cycle



INSIGHTS AND ANALYSIS

Monitoring and observability provide insights into customer reaction to changes and report on value realization.

PORTFOLIO AND BACKLOG

Vision and goals are set and aligned to epics, features, PBIs and user stories.

CONTINUOUS INTEGRATION

Code is created, artifacts incorporated, versions controlled, code is built in a trunk-based manner.

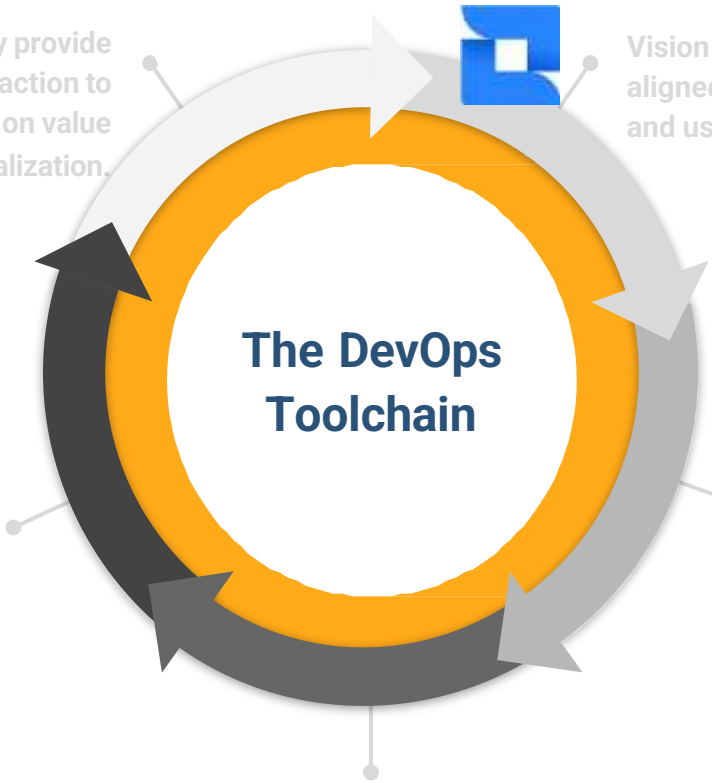
CONTINUOUS TESTING

Functional and non-functional testing takes place at every commit at every step or gate through route to live.

CONTINUOUS DELIVERY

The changes are approved, released and operated in the live environment.

Portfolio Management



INSIGHTS AND ANALYSIS

Monitoring and observability provide insights into customer reaction to changes and report on value realization.

PORTFOLIO AND BACKLOG

Vision and goals are set and aligned to epics, features, PBIs and user stories.

CONTINUOUS INTEGRATION

Code is created, artifacts incorporated, versions controlled, code is built in a trunk-based manner.

CONTINUOUS TESTING

Functional and non-functional testing takes place at every commit at every step or gate through route to live.

CONTINUOUS DELIVERY

The changes are approved, released and operated in the live environment.

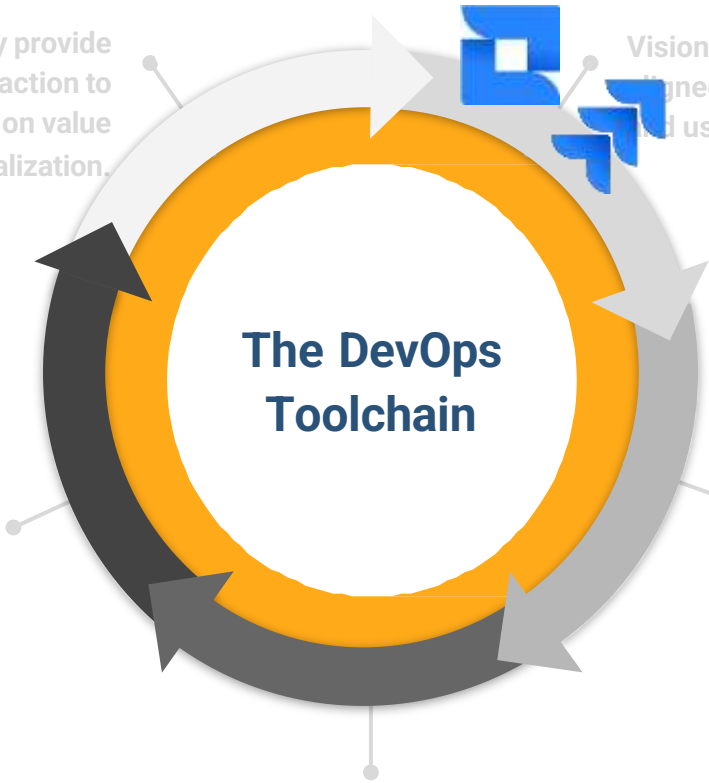
Product Backlog

INSIGHTS AND ANALYSIS

Monitoring and observability provide insights into customer reaction to changes and report on value realization.

PORTFOLIO AND BACKLOG

Vision and goals are set and aligned to epics, features, PBIs and user stories.



CONTINUOUS INTEGRATION

Code is created, artifacts incorporated, versions controlled, code is built in a trunk-based manner.

CONTINUOUS DELIVERY

The changes are approved, released and operated in the live environment.

CONTINUOUS TESTING

Functional and non-functional testing takes place at every commit at every step or gate through route to live.

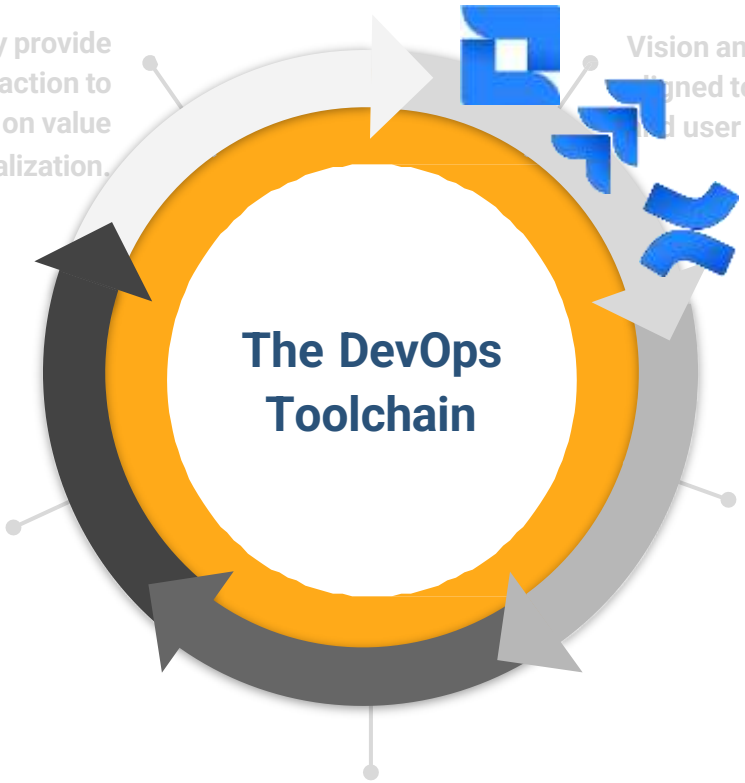
Planning

INSIGHTS AND ANALYSIS

Monitoring and observability provide insights into customer reaction to changes and report on value realization.

PORTFOLIO AND BACKLOG

Vision and goals are set and aligned to epics, features, PBIs and user stories.



CONTINUOUS INTEGRATION

Code is created, artifacts incorporated, versions controlled, code is built in a trunk-based manner.

CONTINUOUS DELIVERY

The changes are approved, released and operated in the live environment.

CONTINUOUS TESTING

Functional and non-functional testing takes place at every commit at every step or gate through route to live.

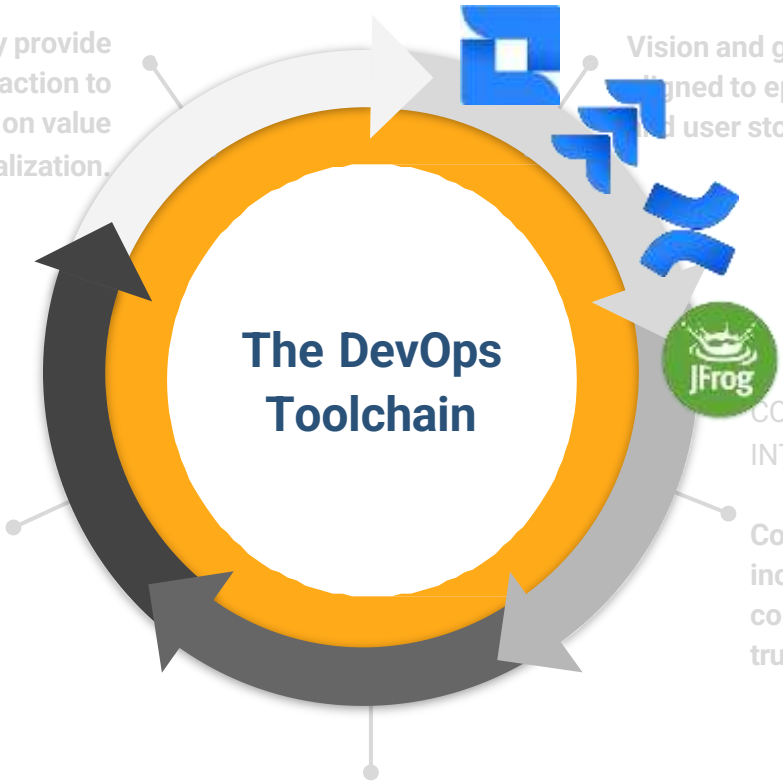
Artifact Repository

INSIGHTS AND ANALYSIS

Monitoring and observability provide insights into customer reaction to changes and report on value realization.

PORTFOLIO AND BACKLOG

Vision and goals are set and aligned to epics, features, PBIs and user stories.



CONTINUOUS INTEGRATION

Code is created, artifacts incorporated, versions controlled, code is built in a trunk-based manner.

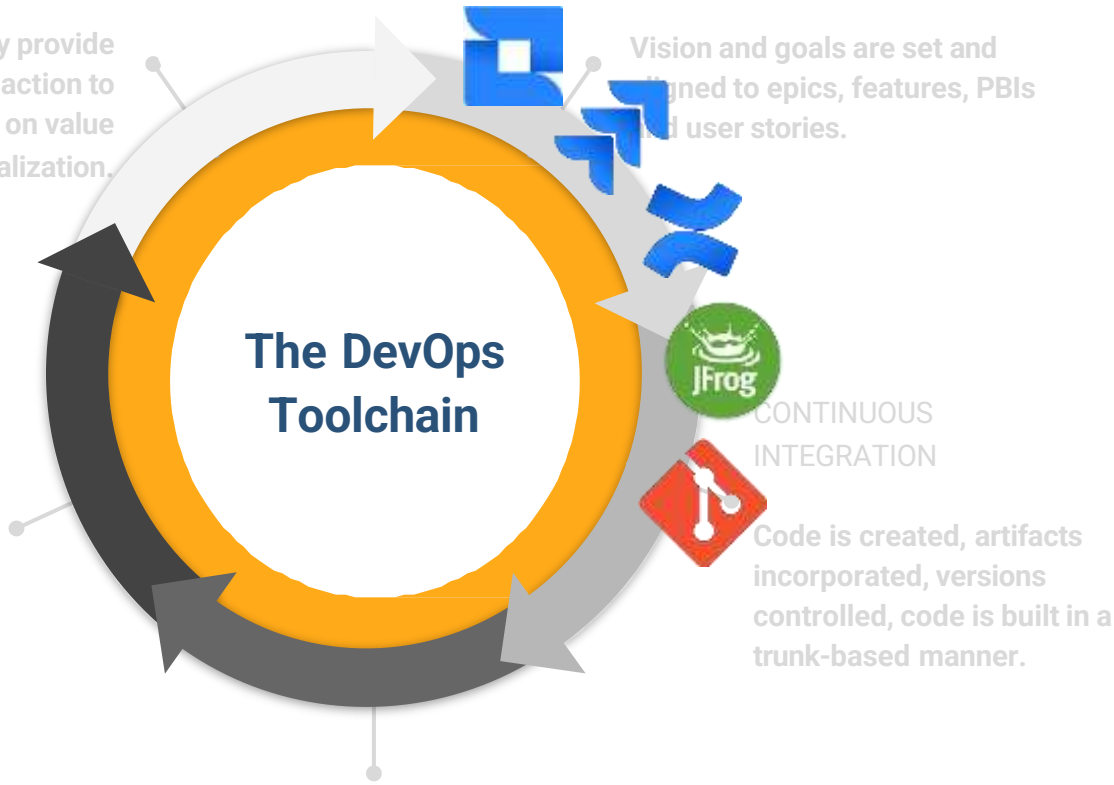
CONTINUOUS DELIVERY

The changes are approved, released and operated in the live environment.

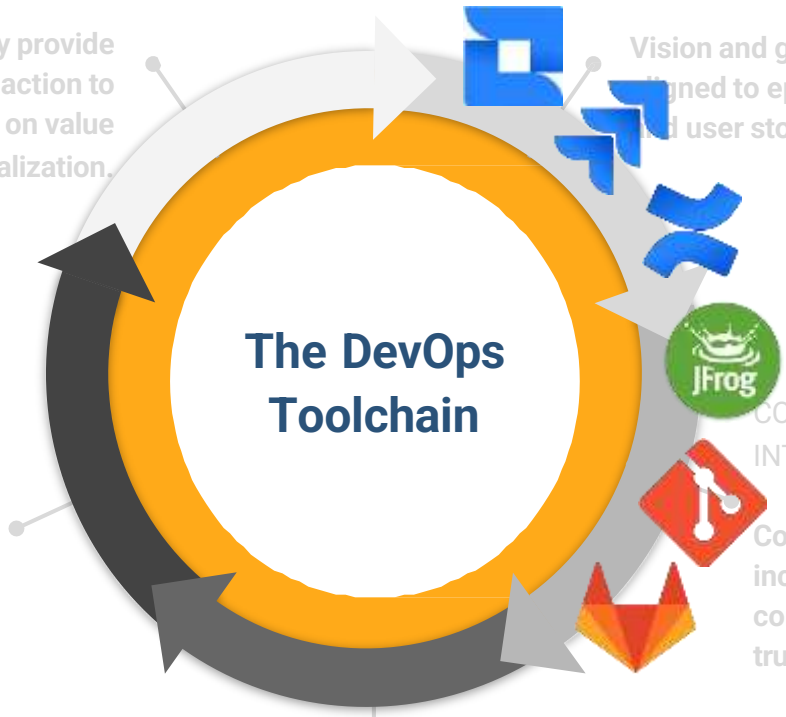
CONTINUOUS TESTING

Functional and non-functional testing takes place at every commit at every step or gate through route to live.

Version/Source Control



CI Server



INSIGHTS AND ANALYSIS

Monitoring and observability provide insights into customer reaction to changes and report on value realization.

PORTFOLIO AND BACKLOG

Vision and goals are set and
igned to epics, features, PBIs
d user stories.

CONTINUOUS INTEGRATION

Code is created, artifacts incorporated, versions controlled, code is built in a trunk-based manner.

CONTINUOUS DELIVERY

The changes are approved, released and operated in the live environment.

CONTINUOUS TESTING

Functional and non-functional testing takes place at every commit at every step or gate through route to live.

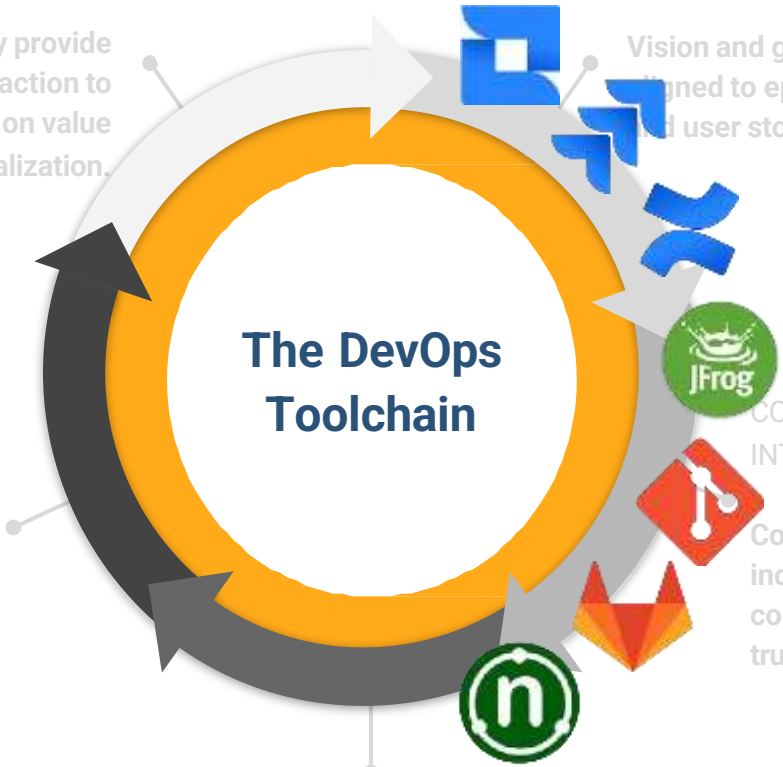
Unit Testing

INSIGHTS AND ANALYSIS

Monitoring and observability provide insights into customer reaction to changes and report on value realization.

PORTFOLIO AND BACKLOG

Vision and goals are set and
igned to epics, features, PBIs
d user stories.



CONTINUOUS INTEGRATION

Code is created, artifacts incorporated, versions controlled, code is built in a trunk-based manner.

CONTINUOUS DELIVERY

The changes are approved, released and operated in the live environment.

CONTINUOUS TESTING

Functional and non-functional testing takes place at every commit at every step or gate through route to live.

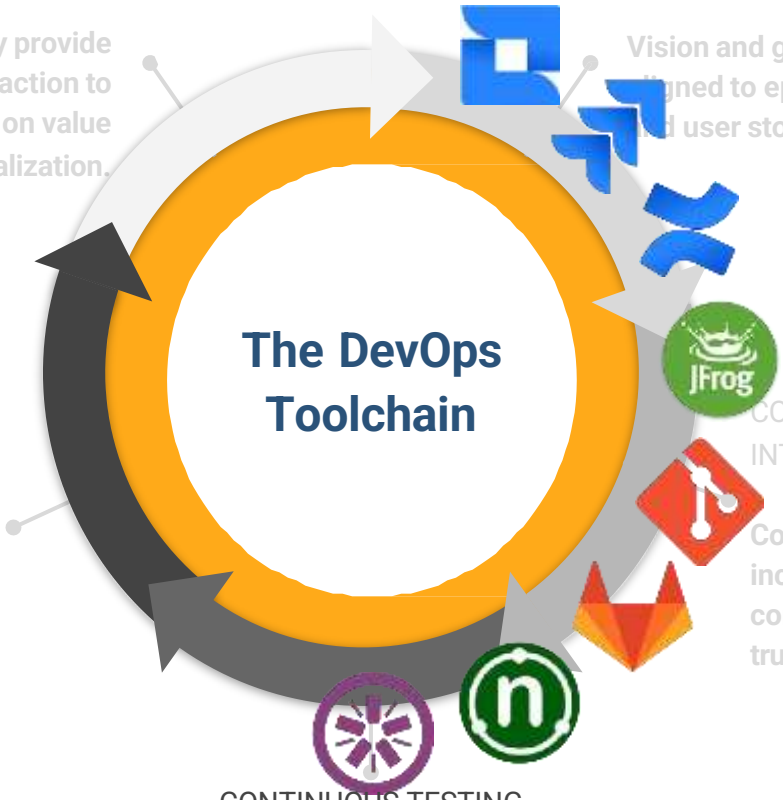
Integration Testing

INSIGHTS AND ANALYSIS

Monitoring and observability provide insights into customer reaction to changes and report on value realization.

PORTFOLIO AND BACKLOG

Vision and goals are set and aligned to epics, features, PBIs and user stories.



CONTINUOUS INTEGRATION

Code is created, artifacts incorporated, versions controlled, code is built in a trunk-based manner.

CONTINUOUS DELIVERY

The changes are approved, released and operated in the live environment.

CONTINUOUS TESTING

Functional and non-functional testing takes place at every commit at every step or gate through route to live.

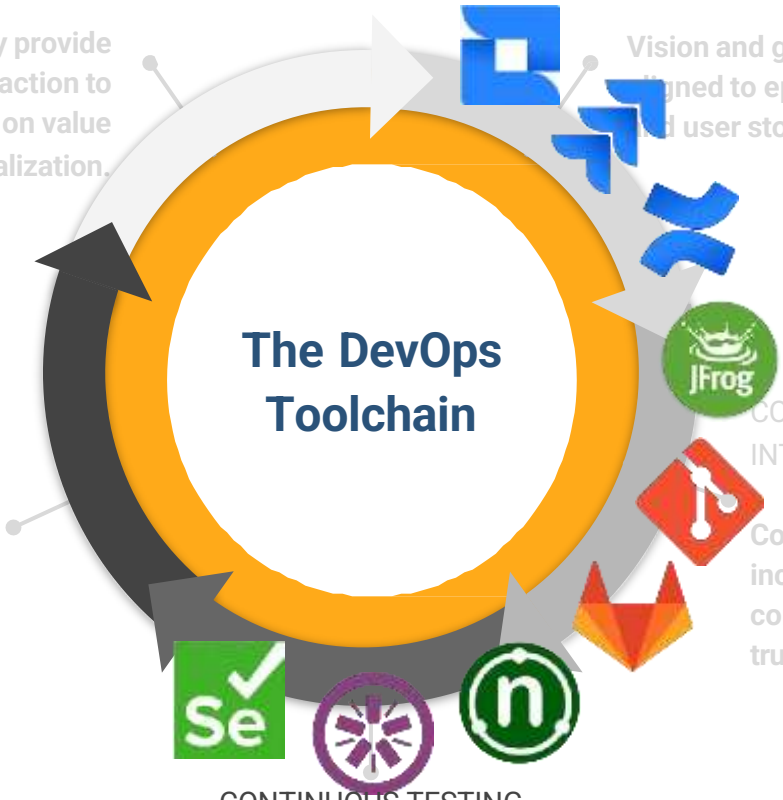
User Acceptance Testing

INSIGHTS AND ANALYSIS

Monitoring and observability provide insights into customer reaction to changes and report on value realization.

PORTFOLIO AND BACKLOG

Vision and goals are set and
igned to epics, features, PBIs
d user stories.



CONTINUOUS INTEGRATION

Code is created, artifacts incorporated, versions controlled, code is built in a trunk-based manner.

CONTINUOUS DELIVERY

The changes are approved, released and operated in the live environment.

CONTINUOUS TESTING

Functional and non-functional testing takes place at every commit at every step or gate through route to live.

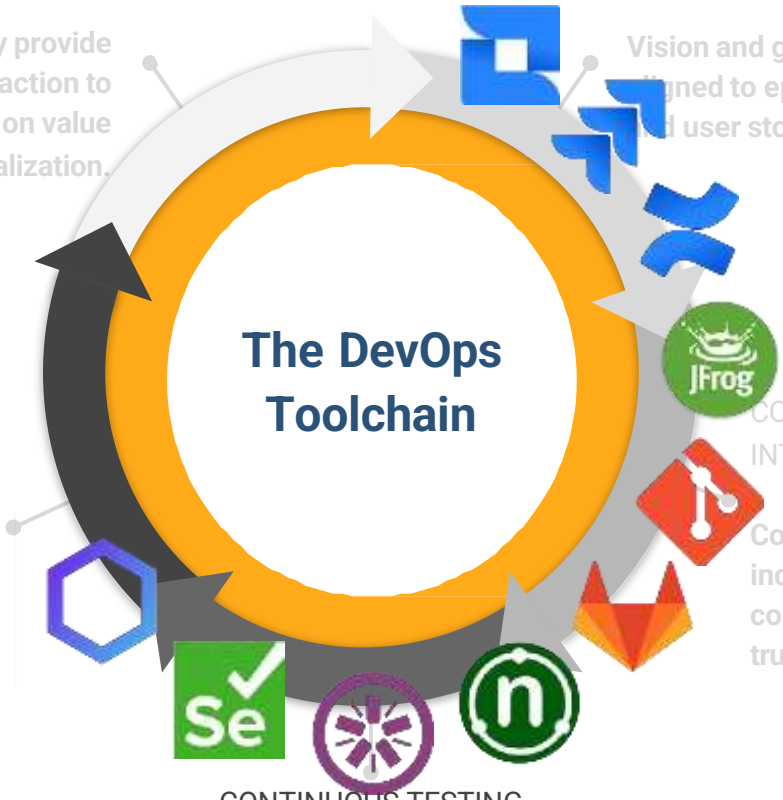
Security Testing

INSIGHTS AND ANALYSIS

Monitoring and observability provide insights into customer reaction to changes and report on value realization.

PORTFOLIO AND BACKLOG

Vision and goals are set and
igned to epics, features, PBIs
d user stories.



CONTINUOUS INTEGRATION

Code is created, artifacts incorporated, versions controlled, code is built in a trunk-based manner.

CONTINUOUS DELIVERY

The changes are approved, released and operated in the live environment.

CONTINUOUS TESTING

Functional and non-functional testing takes place at every commit at every step or gate through route to live.

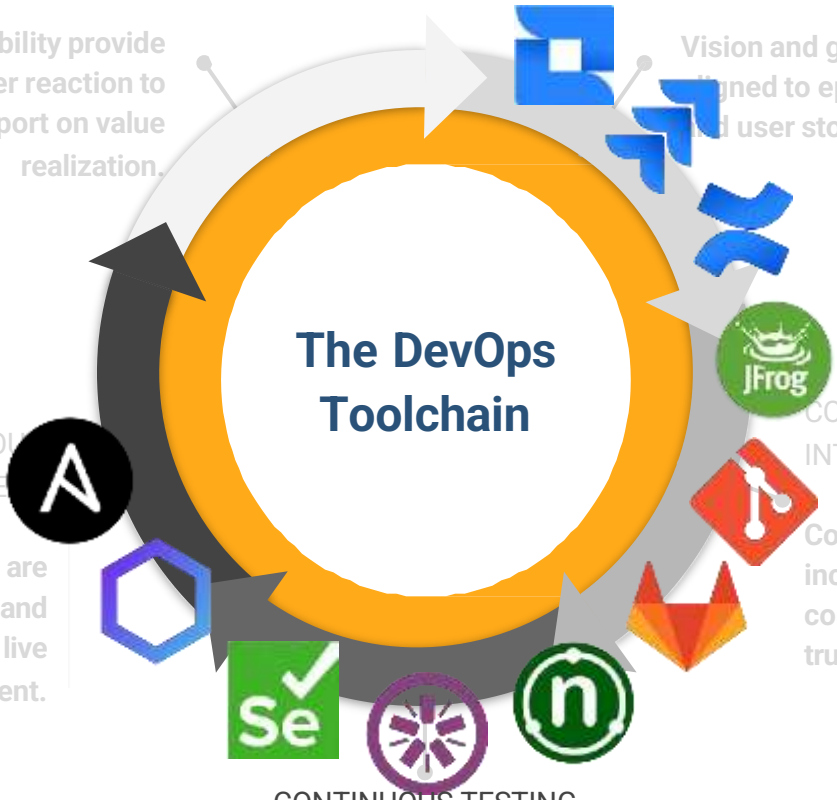
Environment Orchestration

INSIGHTS AND ANALYSIS

Monitoring and observability provide insights into customer reaction to changes and report on value realization.

PORTFOLIO AND BACKLOG

Vision and goals are set and
igned to epics, features, PBIs
d user stories.



CONTINUOUS INTEGRATION

Code is created, artifacts incorporated, versions controlled, code is built in a trunk-based manner.

CONTINUOU
DELIVE

The changes are approved, released and operated in the live environment.

CONTINUOUS TESTING

Functional and non-functional testing takes place at every commit at every step or gate through route to live.

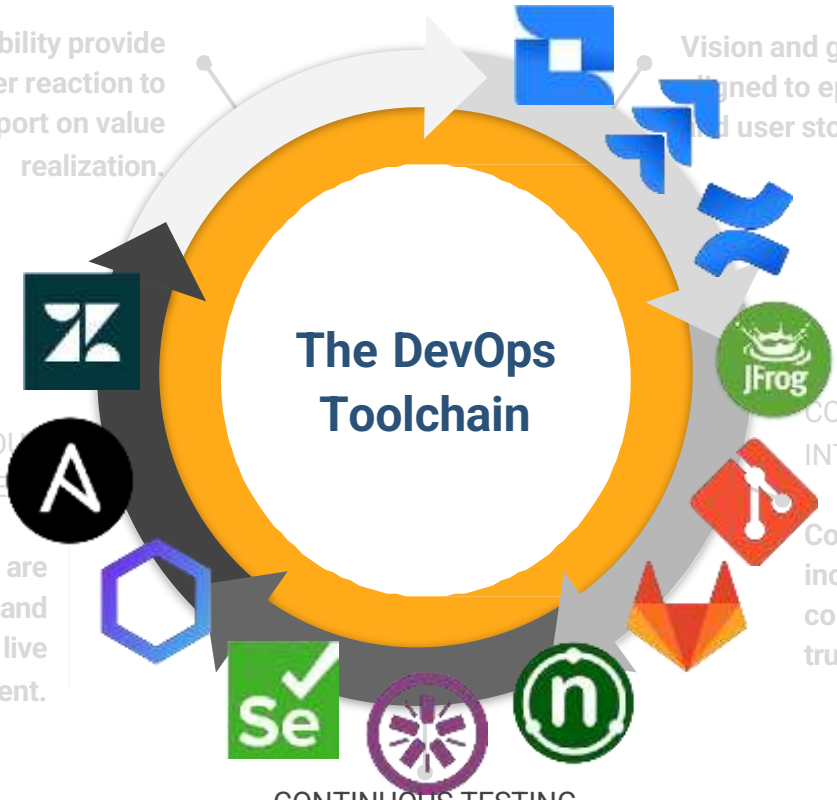
Service Desk

INSIGHTS AND ANALYSIS

Monitoring and observability provide insights into customer reaction to changes and report on value realization.

PORTFOLIO AND BACKLOG

Vision and goals are set and
igned to epics, features, PBIs
d user stories.



CONTINUOUS INTEGRATION

Code is created, artifacts incorporated, versions controlled, code is built in a trunk-based manner.

CONTINUOU
DELIVE

The changes are approved, released and operated in the live environment.

CONTINUOUS TESTING

Functional and non-functional testing takes place at every commit at every step or gate through route to live.

Logging and Monitoring

INSIGHTS AND ANALYSIS

Monitoring and observability provide insights into customer reaction to changes and report on value realization.

PORTFOLIO AND BACKLOG

Vision and goals are set and aligned to epics, features, PBIs and user stories.

CONTINUOUS INTEGRATION

Code is created, artifacts incorporated, versions controlled, code is built in a trunk-based manner.

CONTINUOUS DELIVERY

The changes are approved, released and operated in the live environment.

CONTINUOUS TESTING

Functional and non-functional testing takes place at every commit at every step or gate through route to live.

AIOps

INSIGHTS AND ANALYSIS

Monitoring and observability provide insights into customer reaction to changes and report on the reliability

PORTFOLIO AND BACKLOG

Vision and goals are set and aligned to epics, features, PBIs and user stories.

CONTINUOUS INTEGRATION

Code is created, artifacts incorporated, versions controlled, code is built in a trunk-based manner.

CONTINUOUS DELIVERY

The changes are approved, released and operated in the live environment.

CONTINUOUS TESTING

Functional and non-functional testing takes place at every commit at every step or gate through route to live.

Observability

INSIGHTS AND ANALYSIS

Monitoring and observability provide insights into customer reaction to changes and report anything re

PORTFOLIO AND BACKLOG

Vision and goals are set and aligned to epics, features, PBIs and user stories.

CONTINUOUS DELIVE

The changes are approved, released and operated in the live environment.

CONTINUOUS INTEGRATION

Code is created, artifacts incorporated, versions controlled, code is built in a trunk-based manner.

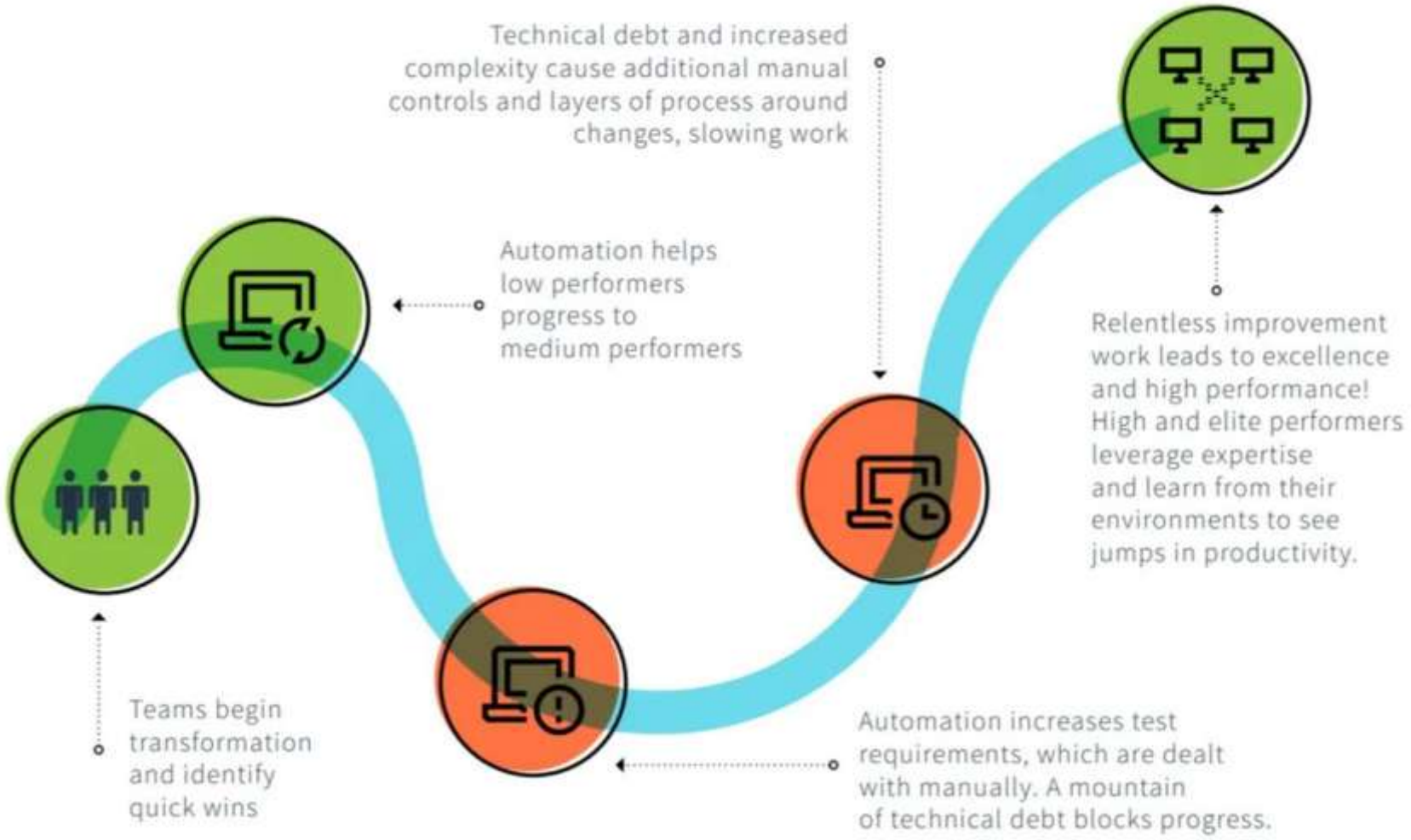
CONTINUOUS TESTING

Functional and non-functional testing takes place at every commit at every step or gate through route to live.

Value Stream Management Platform

What a DevOps Journey Looks Like

J-Curve of Transformation - 2018 State of DevOps Report



Key Takeaways

DevOps = Better, faster, safer, sooner, happier

Continuousness

- Continuous testing
- Continuous integration
- Continuous delivery
- Continuous deployment
- Continuous improvement
- Continuous compliance
- Continuous intelligence
- Continuous funding...

CALMS

- Culture
- Automation
- Lean
- Measurement
- Sharing

DevOps + Cloud

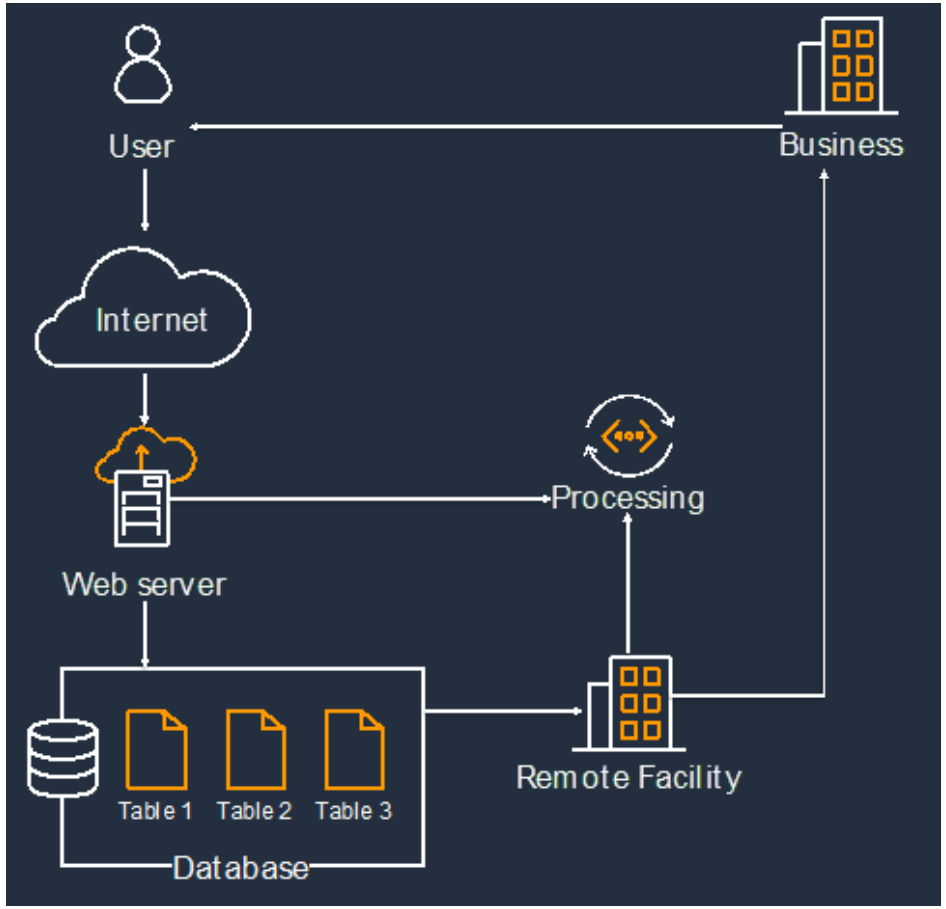
- Cloud tech correlates to DevOps and organizational performance
- Cloud solves common DevOps problems:
  - Production-like test environments
  - Loosely coupled services
  - Integrated toolchains

Amazon’s journey

Just starting out

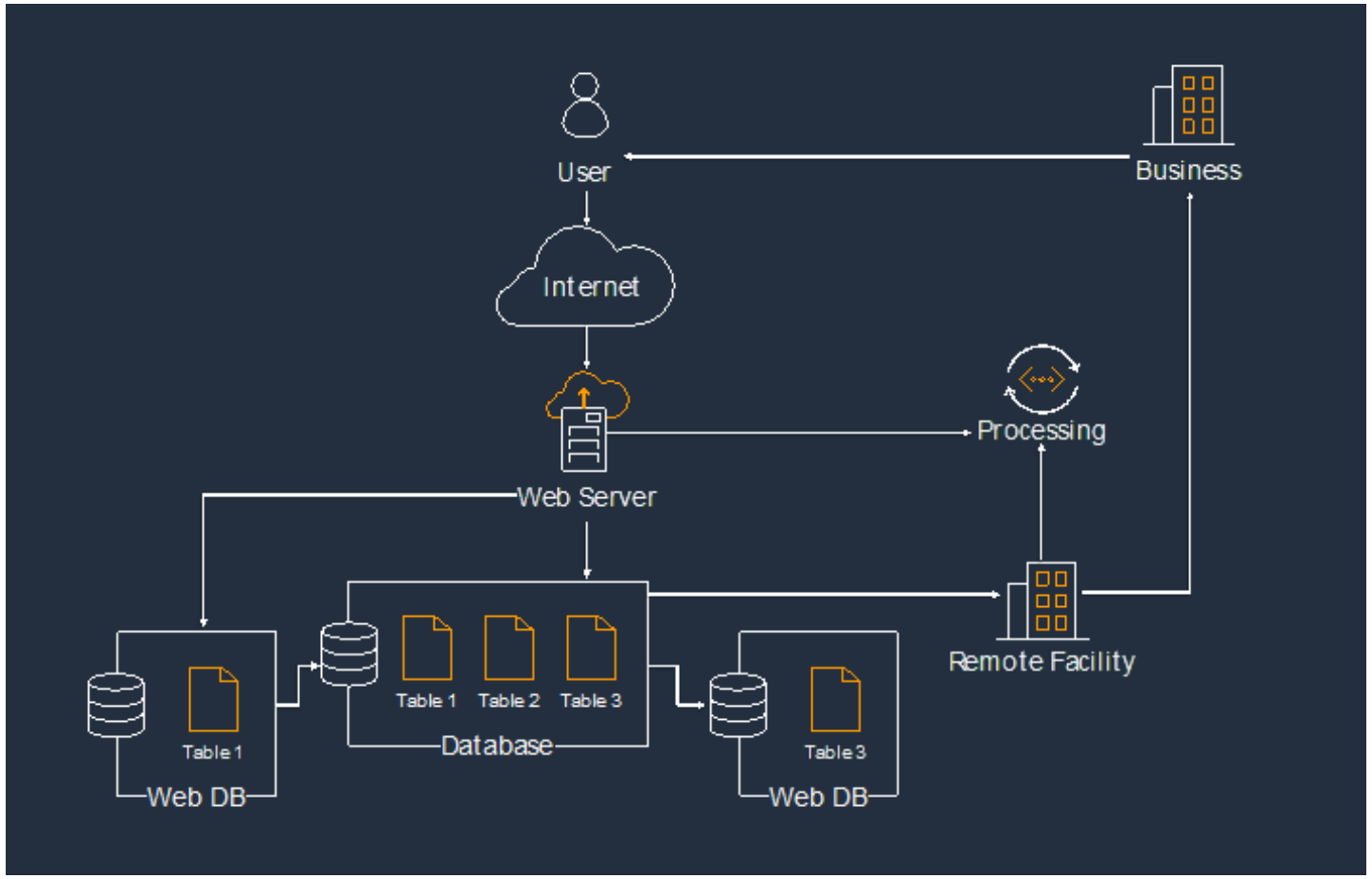
This is how many web architectures started out, and it is how Amazon started too…

There any many bottlenecks, and scaling of the web server was an immediate factor



Scaling v1

This was a bit better, still not very scalable



Challenges

- Dependencies on other teams
- Communication
- Speed of innovation
- Deployment risk

Our mission

Our task was to improve:

- Innovation
- Speed
- Agility
- Safety
- Team Dynamics

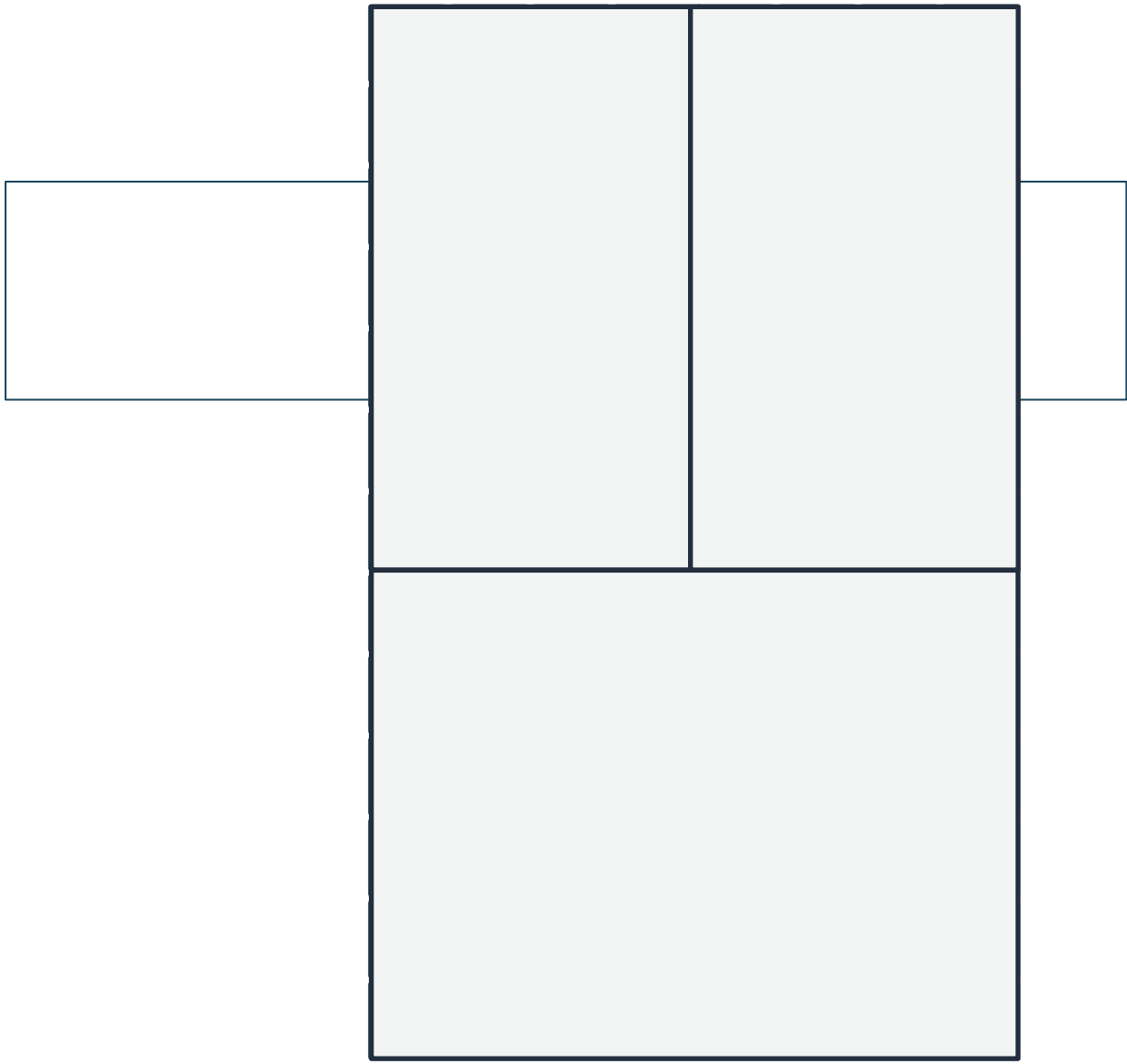
What we did:

- Decomposed for agility
- Cultural and operational shift
- Created tools for software delivery

Going further

Principles

- Make units as small as possible (Primitives)
- De-couple based on scaling factors,
  not functions
- Each service operates independently
  “Communication is terrible!” — Jeff Bezos
- APIs (contracts) between services

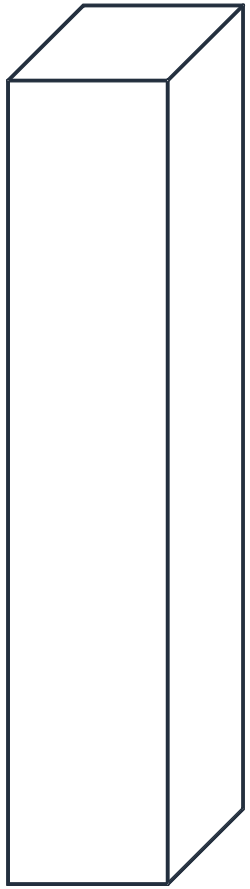


Impact to our development

Monolith development lifecycle

Developers

Services



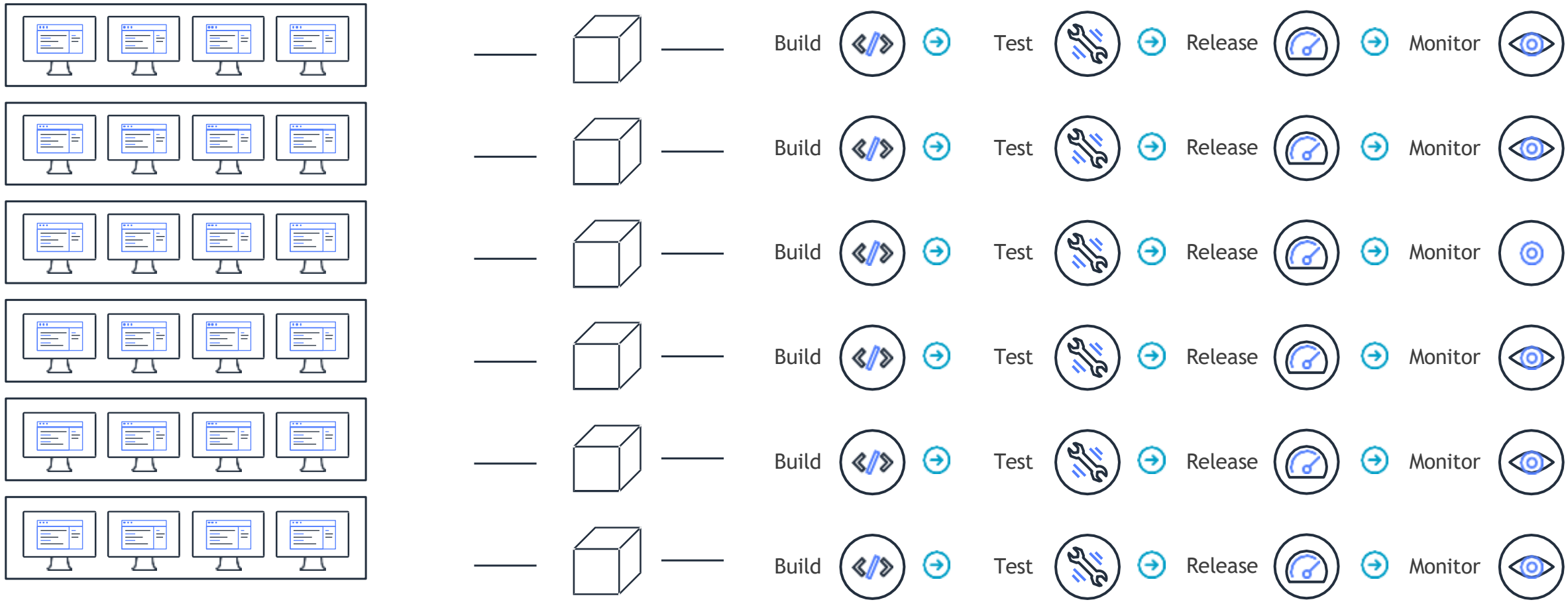
Delivery pipelines

Monolith development lifecycle

This led to changes in organization

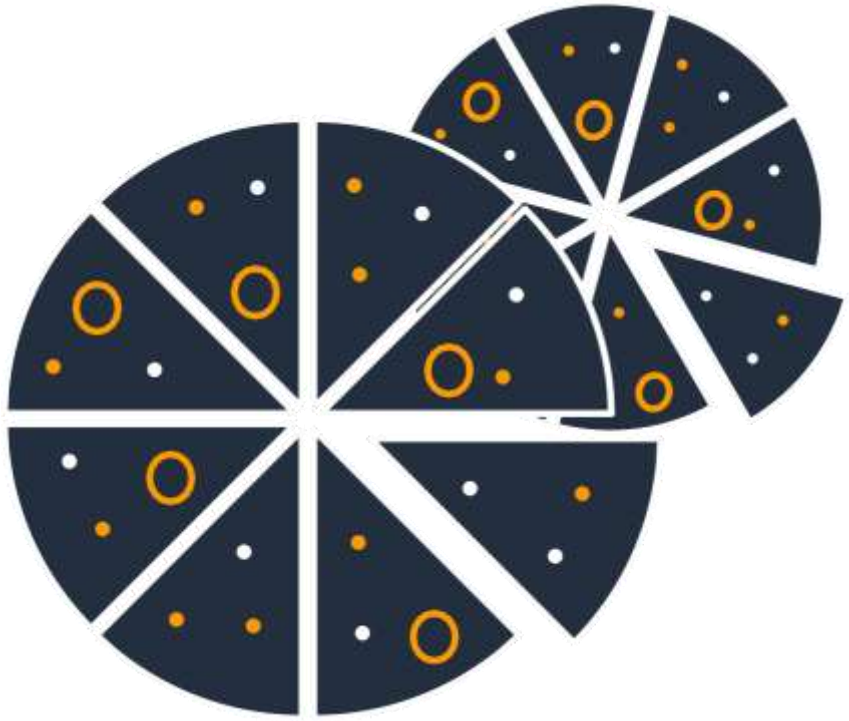
Developers

Services



Impact to our organization

Getting (re)organized

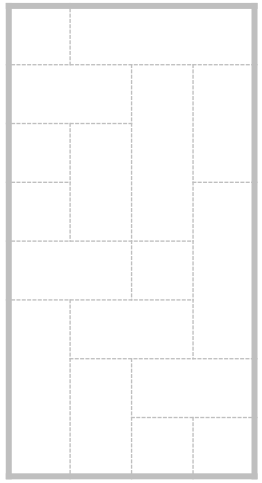


“Two-pizza” teams

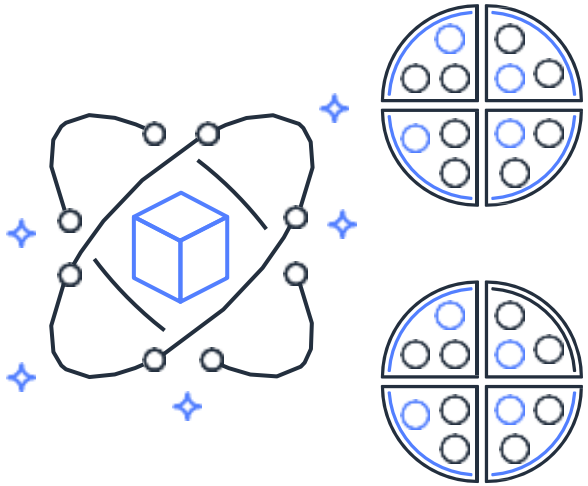
- Own a service
- Minimizes social constraints (Conway’s law)
- Autonomy to make decisions

Transformation timeline

2001 ⟶ **2002**



Monolithic
application + teams



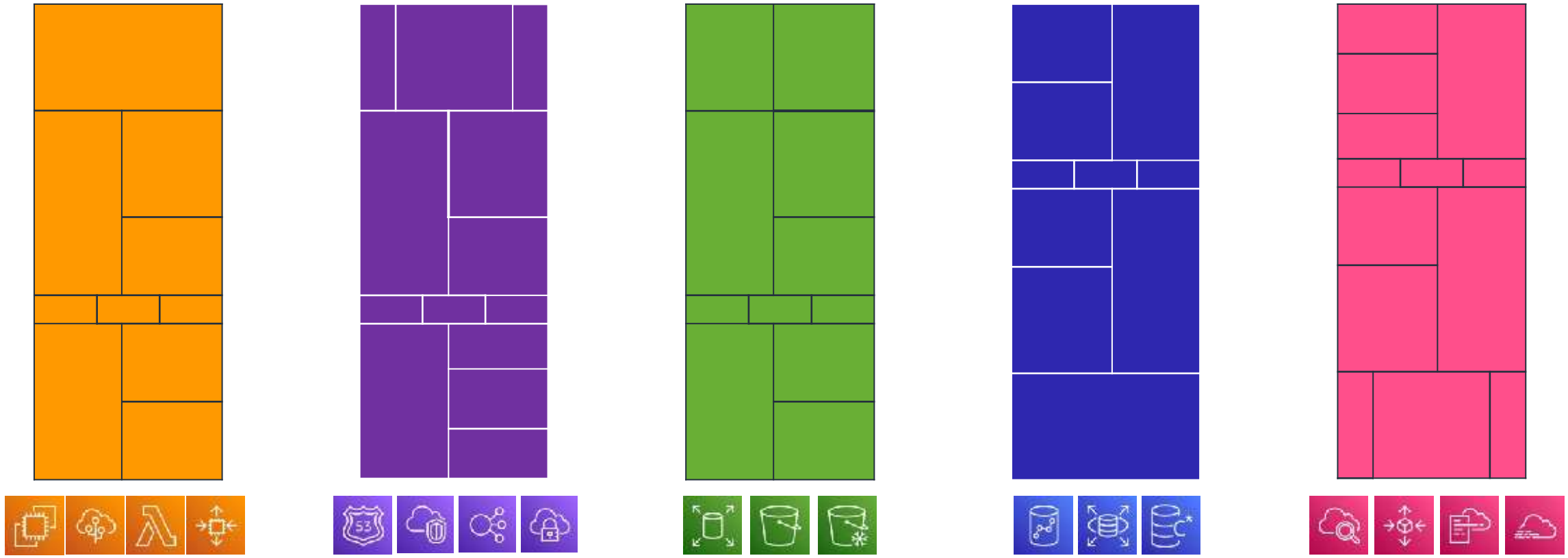
Microservices
+ 2-pizza teams

Teams own everything

- Planning
- Security
- Performance
- Scalability
- Deployment

- Operation
- Bugs
- Documentation
- Testing…

1. Building Blocks



2. Guardrails

What are guardrails?

Guardrails are mechanisms, such as processes or practices, that reduce both the occurrence & blast radius of undesirable application behavior

What are some real-world guardrails?

Monitoring

- CPU Utilization
- Database throughput
- Business processes

Provisioning

- Access permissions
- Resource availability
- Configuration

Deployment

- Time window
- Toolsets available
- Size or timing of test releases

Cost management

- Resource costs
- Resource utilization
- Spend run rates

Security & compliance

- Account set up/access
- Standards compliance
- Certificate maintenance

3. Fully Automated Deployments

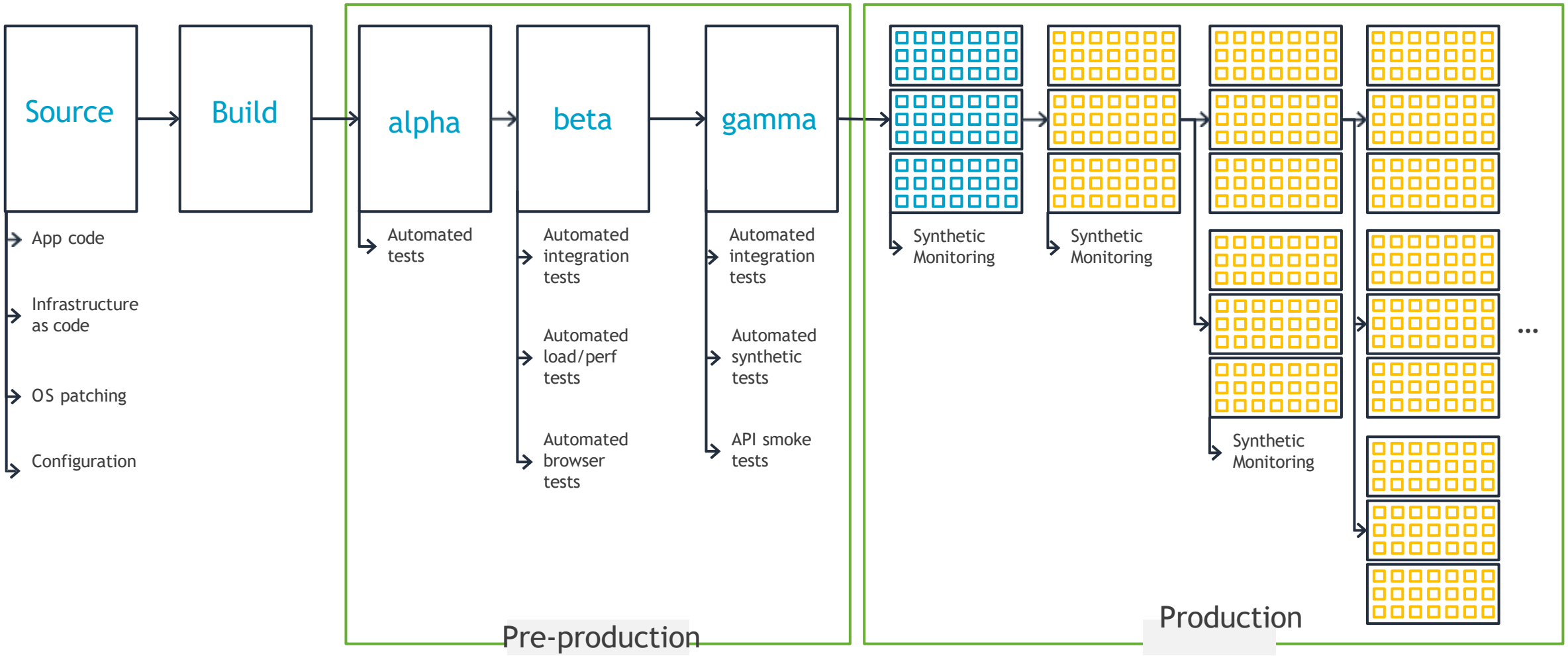
- Check-in source code such as .java files and Dockerfile
- Peer review new code

- Compile code
- Unit tests
- Style checkers
- Create container images

- Integration tests with other systems
- Load testing
- UI tests
- Security testing

- Deploy to production environments
- Monitor code in production in order to quickly detect errors

Amazon Continuous Delivery: Deep Dive

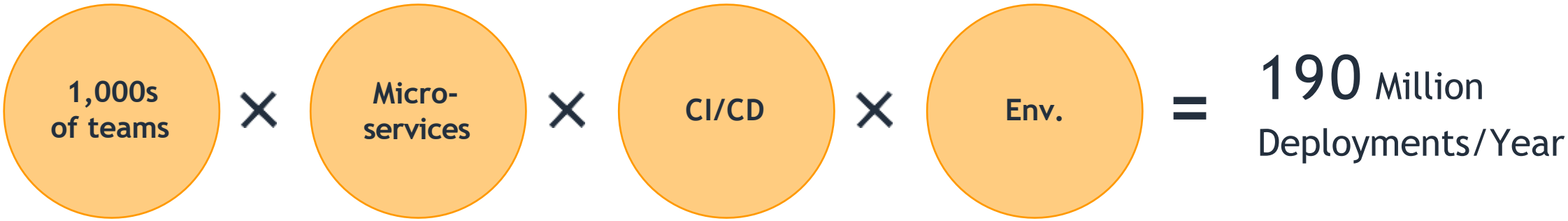


Modern applications

Today we have modern applications

- Use independently scalable microservices (serverless, containers...)
- Connect through APIs
- Deliver updates continuously
- Adapt quickly to change
- Scale globally
- Are fault tolerant
- Carefully mange state and persistence
- Have security built-in

Deployment at scale



Just the beginning

Along the way we have learned a lot about writing software

- That’s performant, safe, and scalable

We have had to solve some really hard problems

- At massive scale

We know our way is not the only way, and many of our solutions are not fancy

- But we know they work

We are long obsessed with building things to help our customers

- We want to share the benefits of what we learned along the way

McDonald’s brings home delivery to market in four months

“This was a four month-duration for us—from idea, to development to massive scale. That's the new norm that we see everyday.”

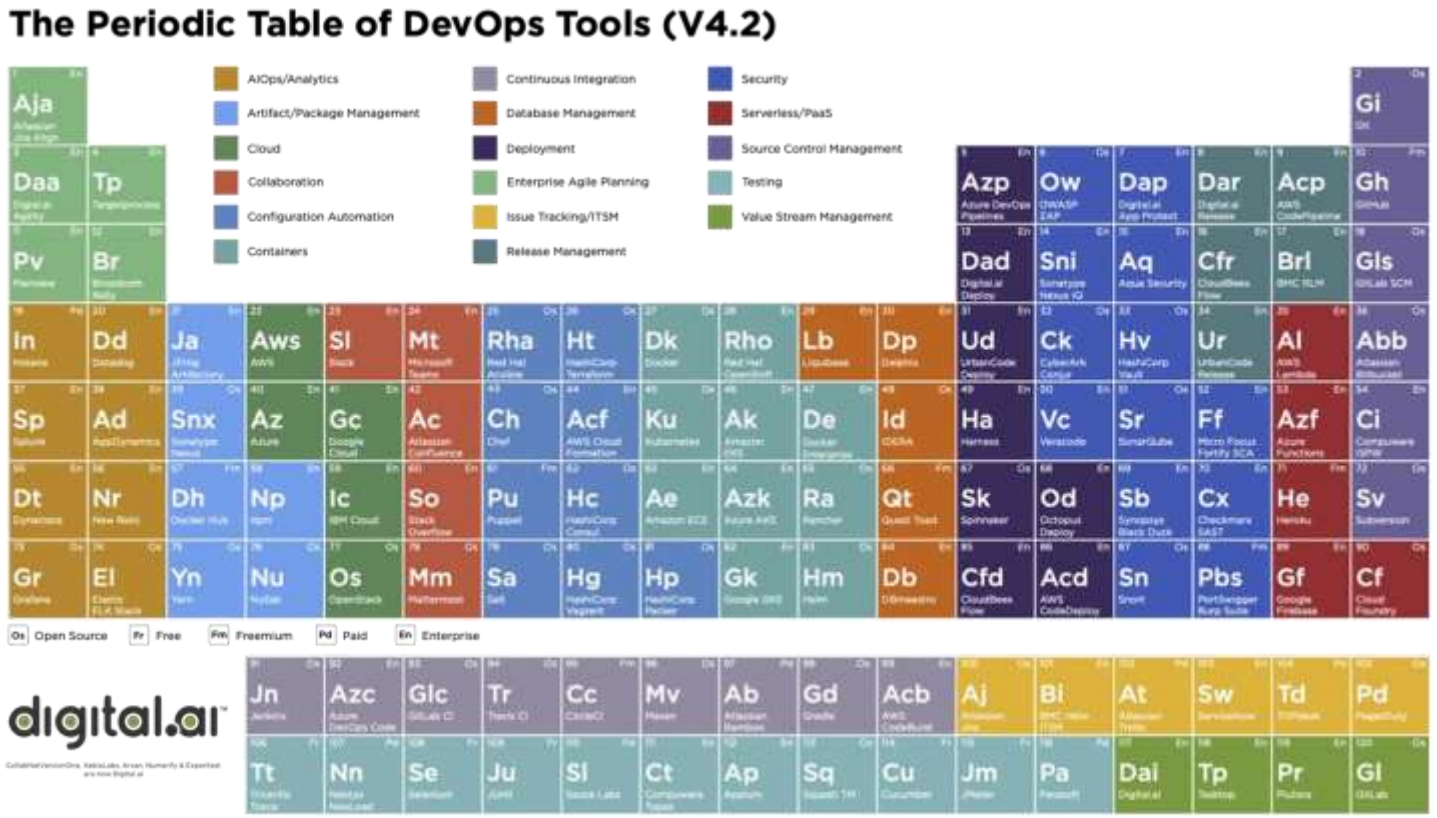
- Thilina Gunasinghe, Chief Technology Architect, McDonald’s

Scalability and reliability to deliver over 1 million orders per hour

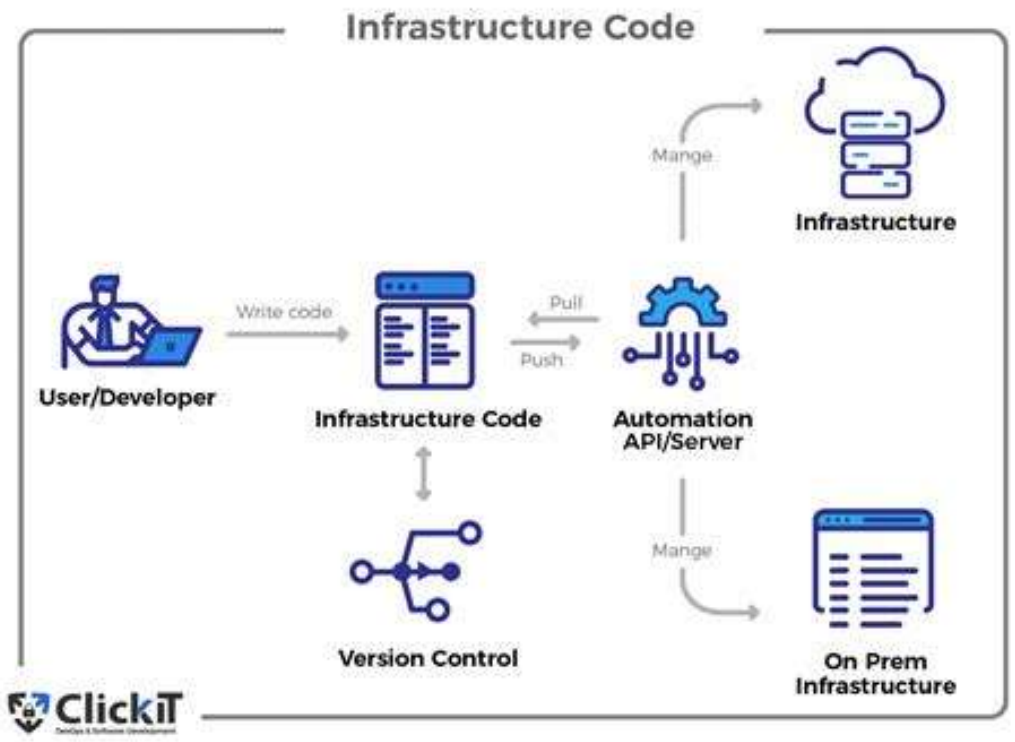
Multi-country support, each with multiple delivery partners

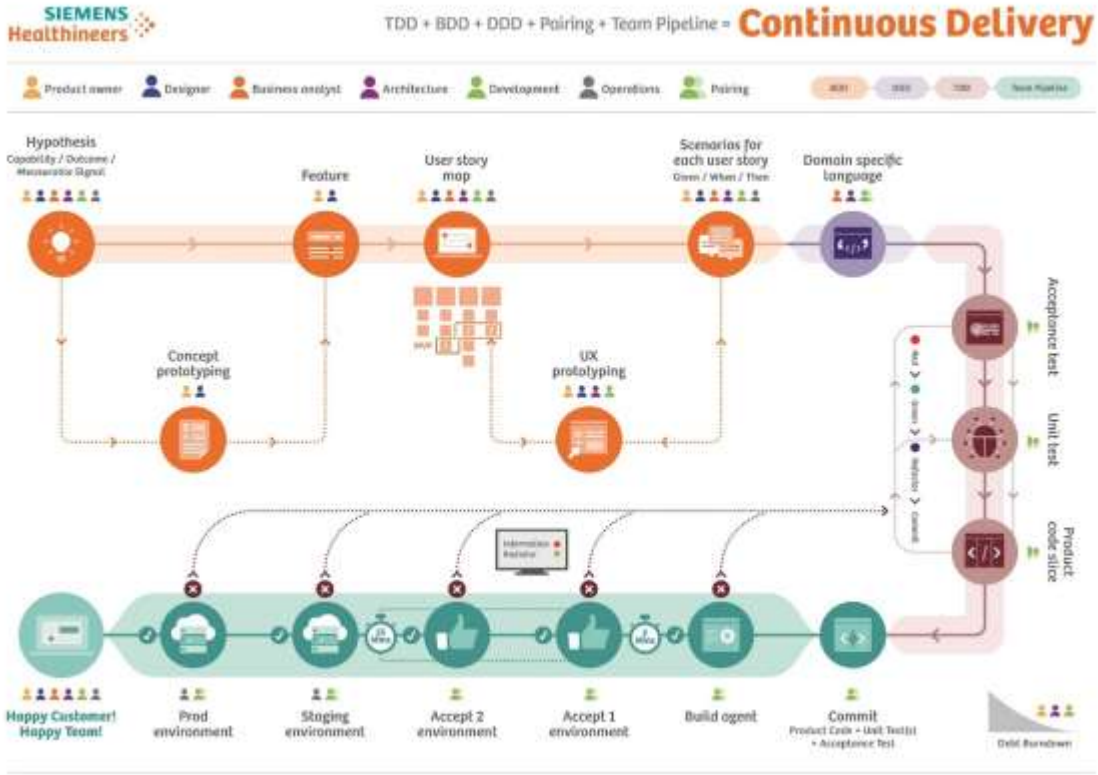
Cost sensitive – selling hamburgers!

DevOps tooling is critically important for successful practices



Devops Practices

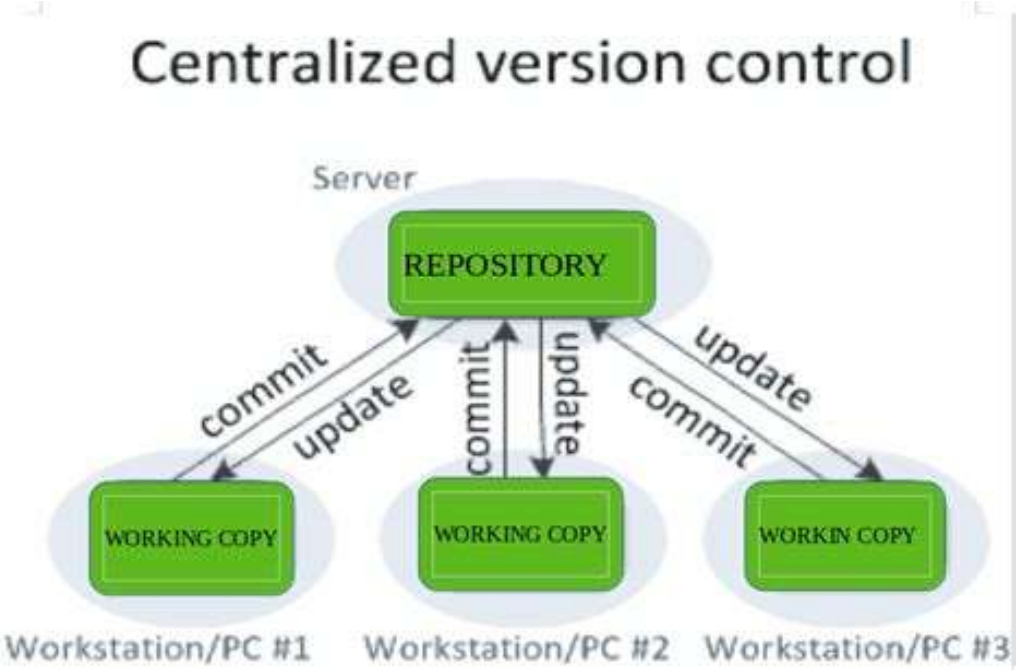




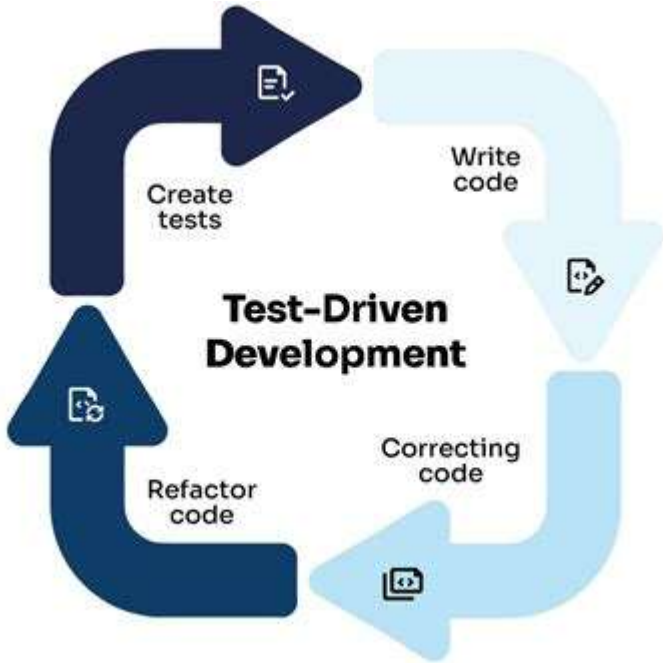
Infrastructure as Code

CI/CD, one touch build/deploy

Devops Practices

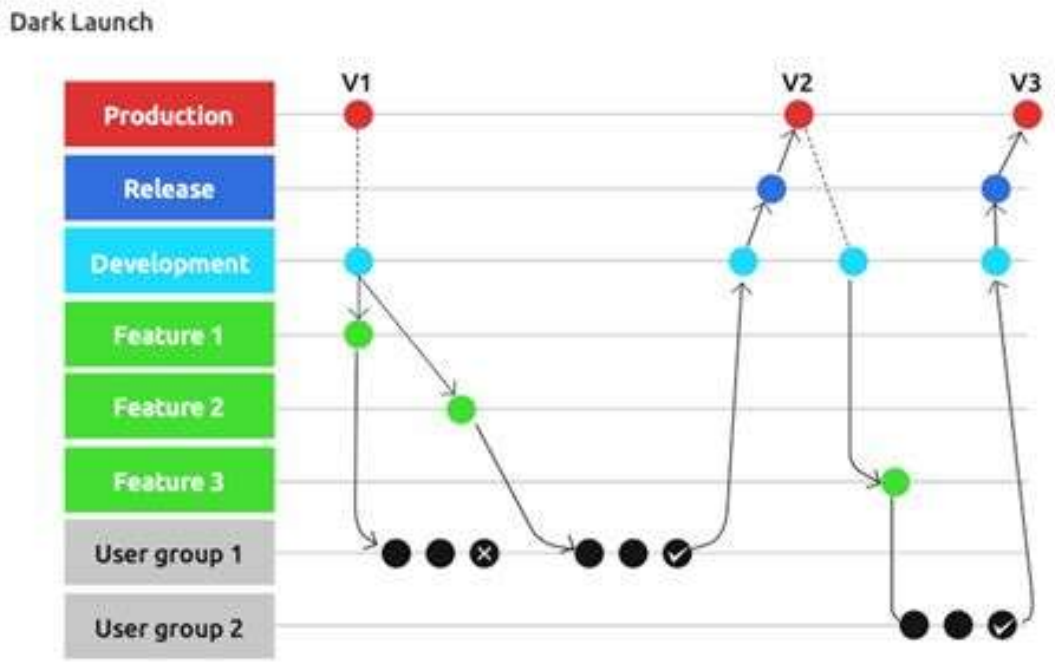


Version Control



Automated Testing

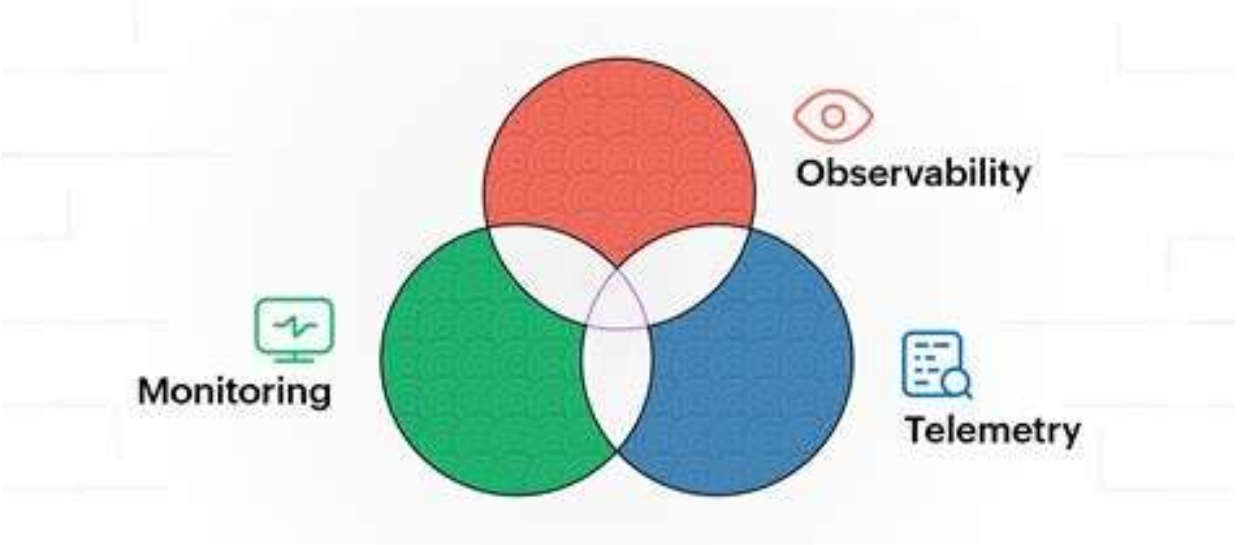
Devops Practices



Feature Flags

Dark Launches

Devops Practices

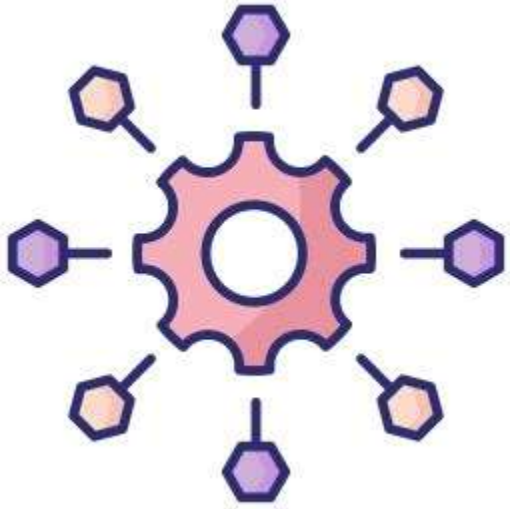


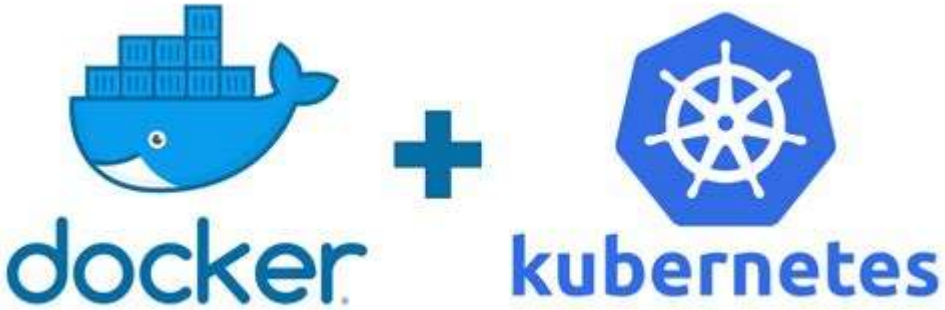
Monitoring and Observability

Devops Practices

Communication Tools

Devops Practices





Microservices Containers Cloud Native

Move on
Components & CI/CD Pipelines