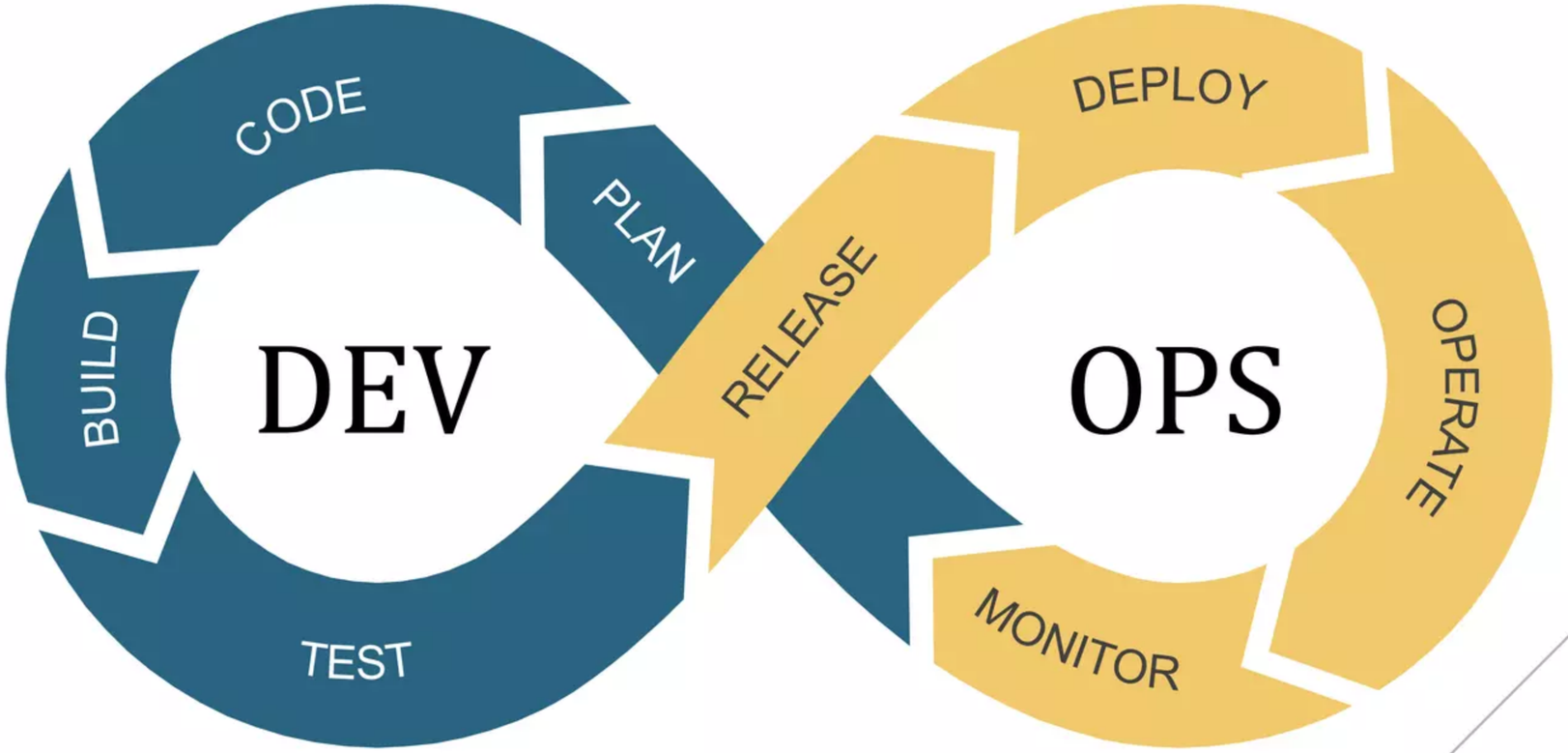
DevOps



Presented by: Pradeep Patel

Pradeep, as an experienced professional primarily focuses on Software Deliveries.

Being a software developer himself during his early career, he is well aware of the day to day struggles of software Delivery teams

Through this presentation he is sharing his experiences of how working with DevOps has helped improve deliveries and made life much easier for teams

About the Presenter

PMI PMP™

SCRUM @SCALE Certified Practitioner

Microsoft CERTIFIED Technology Specialist

Microsoft CERTIFIED Professional Developer

What is DevOps

DevOps is not a tool or a software, it's a culture that you can adopt for continuous improvement. It will help you to bring your Developer Team and Operations Team on the same page, allowing them to work together with ease.

DevOps: Foundation Principles (Three Ways)

01

Flow : The principles of Flow, which accelerate the delivery of work from Development to Operations to our customers

02

Feedback : The principles of Feedback, which enable us to create ever safer system of work

03

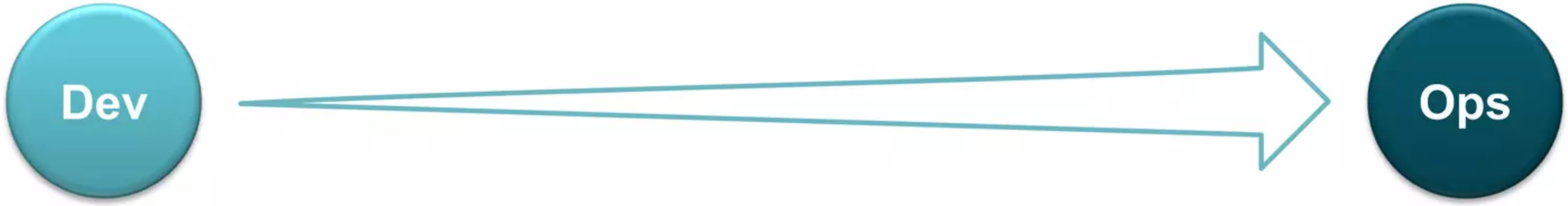
Continual Learning and Experimentation : which foster a high-trust culture and a scientific approach to organizational improvement as part of our daily work

First Way

FLOW

The first principle/way says we need to accelerate the work from development to operation, and then to our customers.

The First Way: Flow



- Understand the flow of work
- Always seek to increase flow
- Never pass a known defect downstream
- Never allow local optimization to cause global degradation
- Achieve a profound understanding of the system

A goal of The First Way is to have work flow quickly from left to right.

Continuous Integration (CI)

Continuous integration is a development practice that requires developers to integrate code into a shared repository on a daily basis.

Each check-in is validated by

- An automated build
- Automated unit, integration and acceptance tests

Integrating regularly in production-like environments makes it easier to quickly detect and locate conflicts and errors.

Continuous Delivery

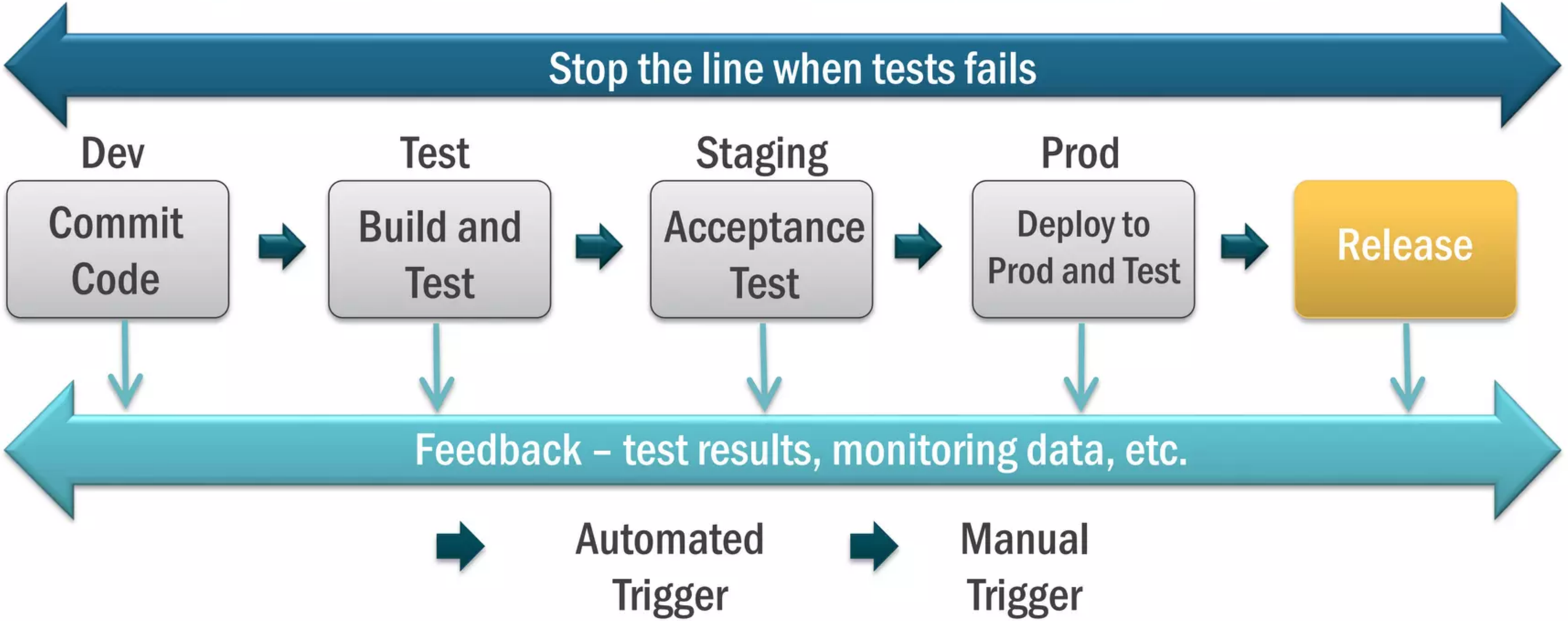
Continuous delivery is a methodology that focuses on making sure software is always in a releasable state throughout its lifecycle.

- Extends continuous integration
- Provides fact, automated feedback on the production-readiness of systems
- Prioritizes keeping software deployable over working on new features
- Enables push-button deployments on demand
- Reduces deployment risks and enables quicker user feedback

Integrating regularly in production-like environments makes it easier to quickly detect and locate conflicts and errors.

Continuous Delivery Cont..

Automated tests in production-like environments assure the code and environment operation as designed and are always in a deployable state.



Deployment is the installation of a specified version of software to a given environment (e.g., promoting a new build into production).

Continuous Deployment

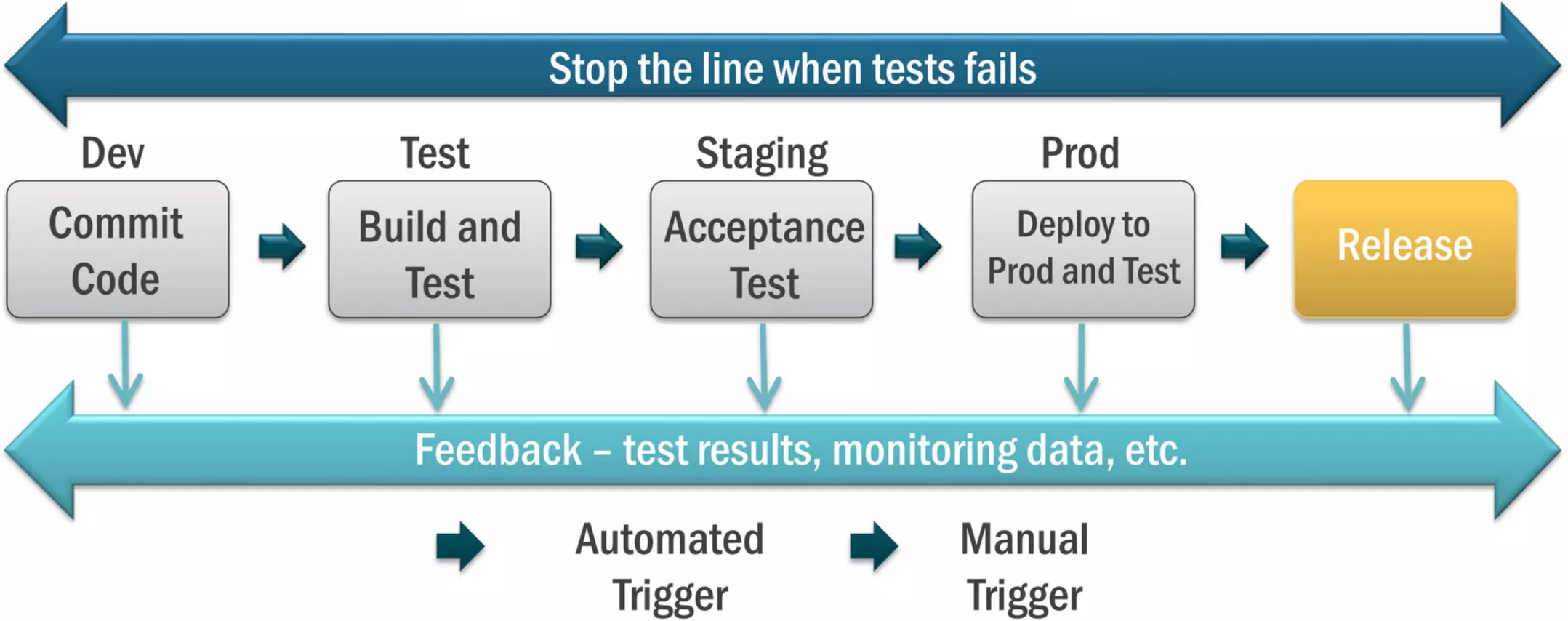
Continuous deployment is a set of practices that enable every change that passes automated tests to be automatically deployed to production.

- Removes the manual step in the Continuous Delivery pipeline
- Results in multiple deployments per day

Continuous deployment may not be practical or possible for companies constrained by regulatory or other requirements.

Continuous Deployment Cont..

Code deployed into production may be invisible to customers, but features can be run and tested by internal staff.



Release is the process or event of making a feature (or set of features) available to a segment of customers.

First Way -in Action

- **Visualize/Measure the Work:** Configured issue tracking software (JIRA) with a Kanban board. Every unit of work is represented as a card on a virtual board. This enables to see the amount of work required to be done have to do in a visual way

- **Limit Work in Progress (WIP) :** If we have less WIP, the quality will go up; if the quality is higher, the work will have fewer defects, and work with fewer defects is likely to not come back (a feature poorly implemented comes back with a bug, for example). If work doesn't come back, the flow is improved.

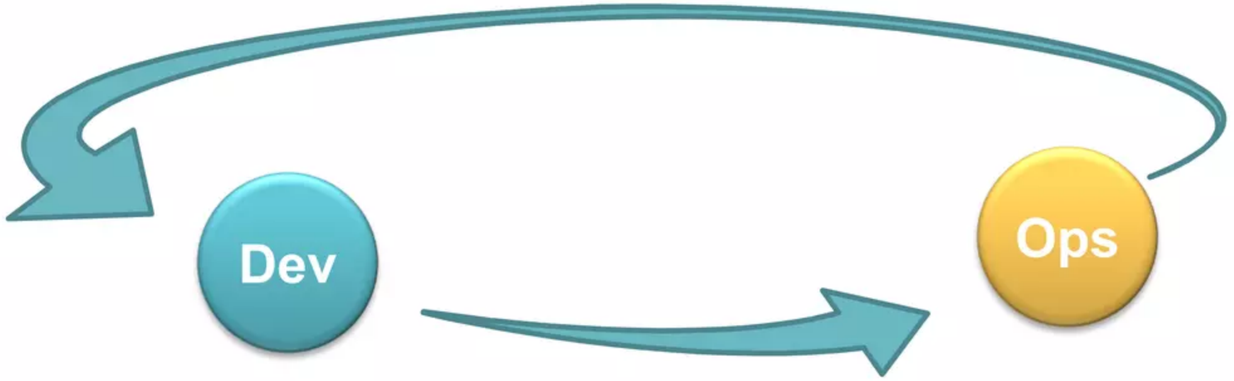
- **Tech/Automation Tools and Practices :** implement continuous build, integration, and then deployment practices. These implementations that happen over a span of few months, will lead to continuous improvement.

Second Way

FEEDBACK

The second way enables the constant flow of feedback from right to left in the stages of our value stream. It requires that we amplify feedback to prevent problems from happening again, and enable faster detection.

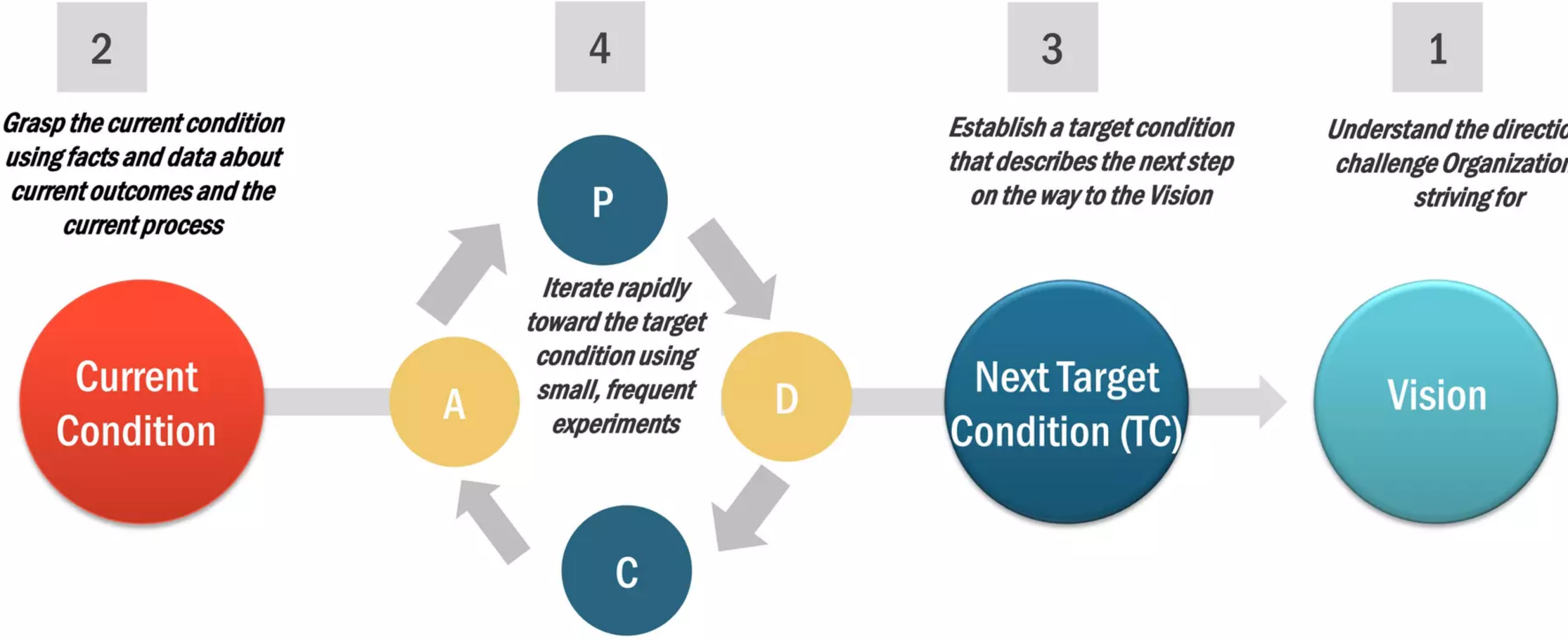
The Second Way: Feedback



- Understand and respond to the needs of all customers – both internal and external
- Shorten and amplify all feedback loops
- Create and embed knowledge where needed

A goal of the Second Way is to shorten and amplify right to left feedback loops so necessary corrections can be continuity made.

Improvement Kata – Four Steps



A kata is a practice pattern used to develop a set of skills. Leaders at any and every level of an organization can use the improvement kata to practice continuous improvement.

Second Way(Kata) in Action

After a developer completes the assigned work, he/she creates a pull-request.

This unit of work has to be reviewed by someone else and be approved before it goes back to the master/trunk stream of development.

This is a peer-review because the work is validated by a member of the team, and not someone higher in the company hierarchy

To ensure quality close to the source itself, it should be mandated that whoever breaks a build is responsible for fixing it as soon as possible.

This way, bad code is prevented from being pulled from someone else or spreading into other branches of development.

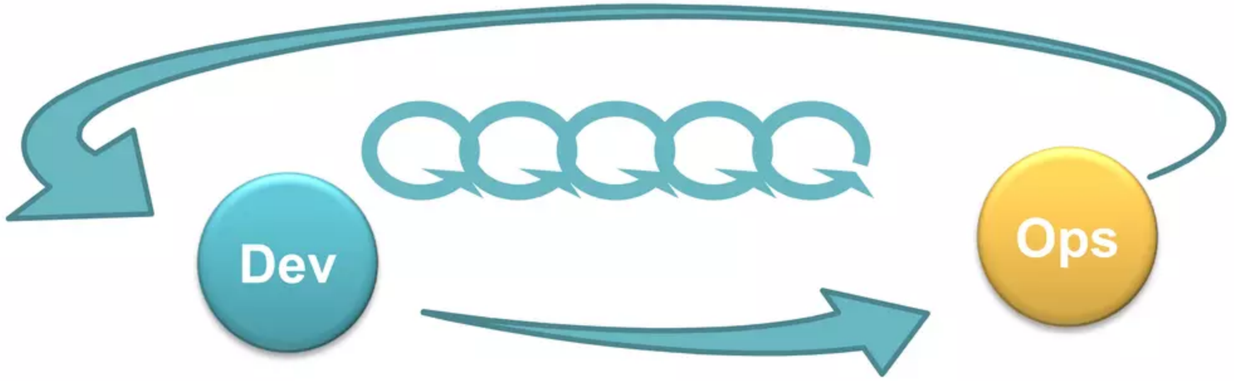
Pair-programming, an extreme implementation of the peer-review can be used, once the masters peer review

Third Way

CONTINUAL LEARNING & EXPERIMENTATION

The third way enables the creation of a high-trust culture that supports a dynamic, disciplined, and scientific approach to experimentation and risk-taking, facilitating the creation of organizational learning, both from our successes and failures.

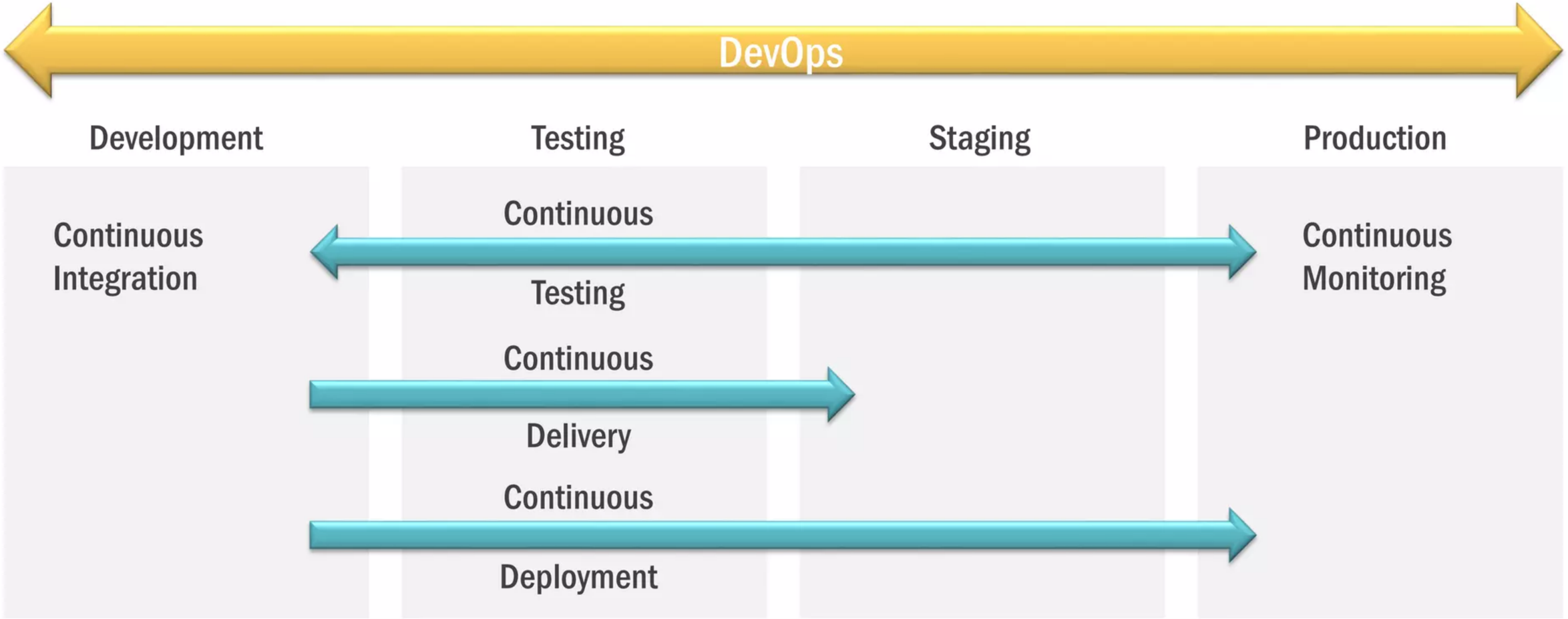
Third Way : Continual learning & Experimentation



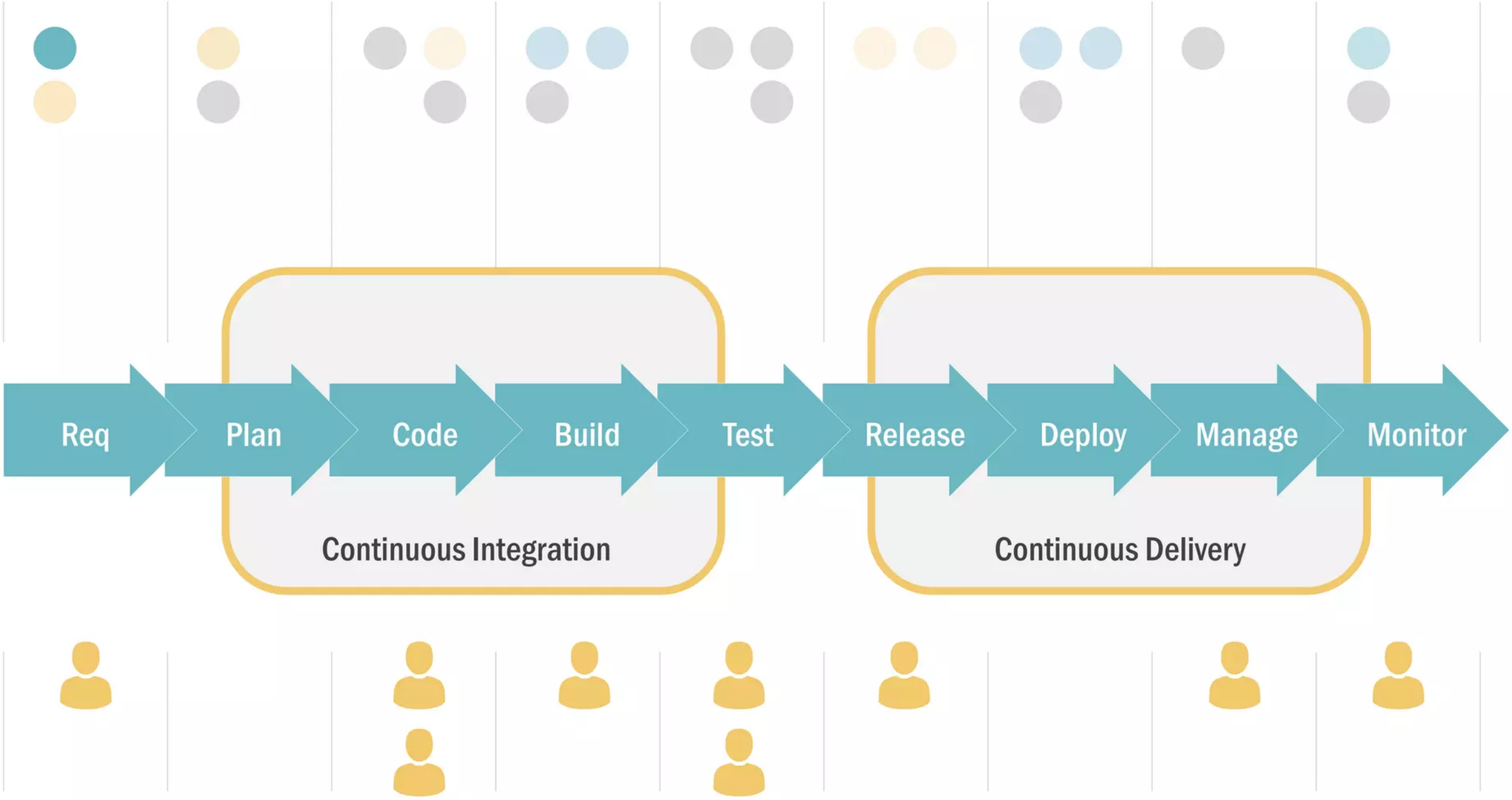
Goal is to create a culture that fosters two things:

- Continual experimentation, taking risks and leaning from failure
- Understanding that repetition and practice is the prerequisite to mastery

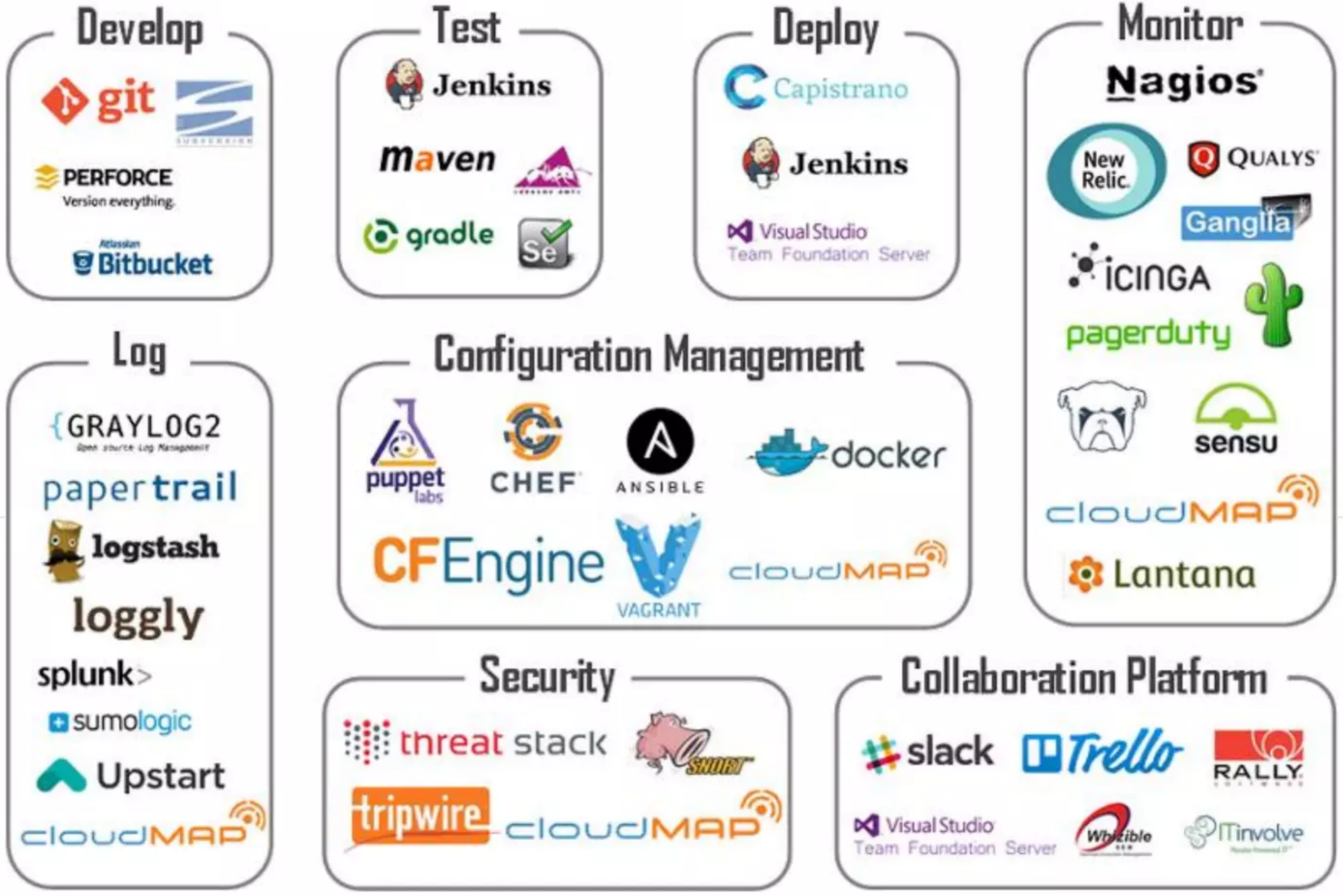
Continuous Delivery

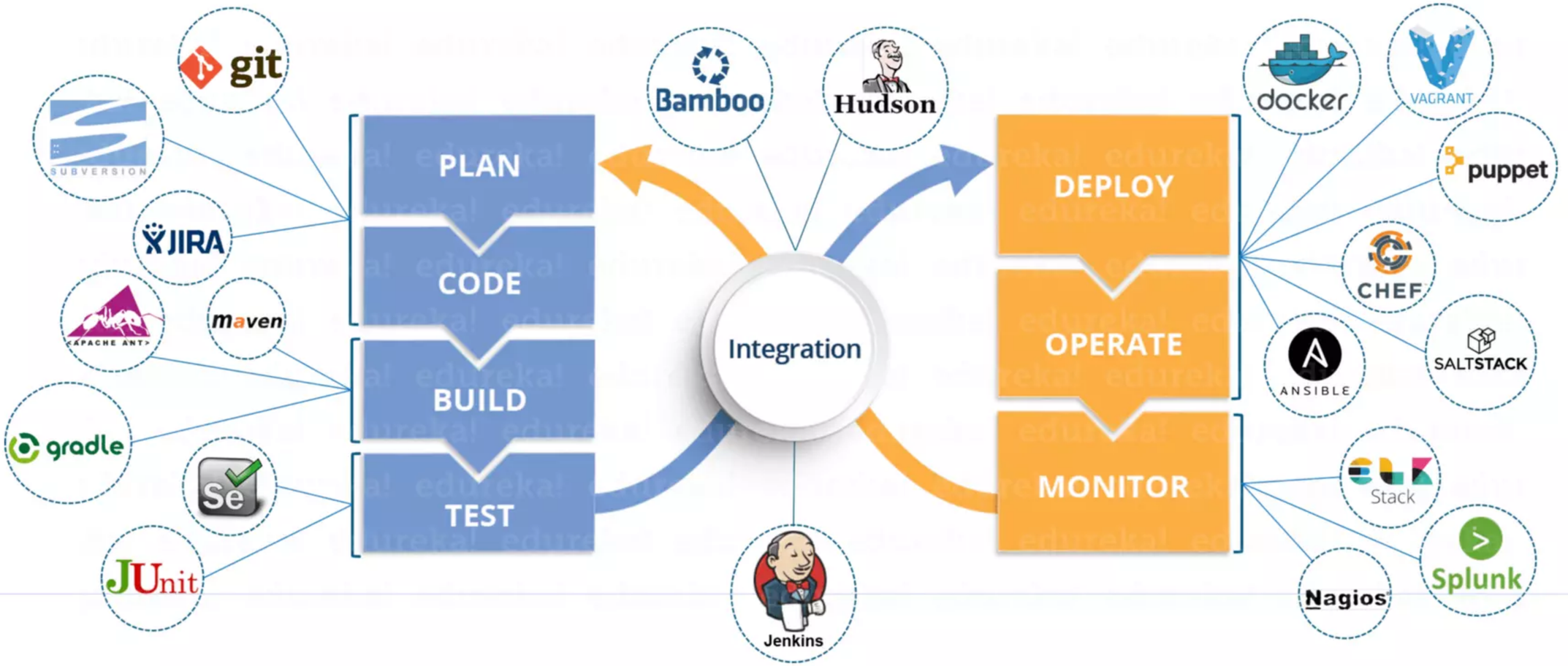


Continuous delivery requires collaboration between Dev and Ops.


Req
Plan
Code
Build
Test
Release
Deploy
Manage
Monitor
Continuous Integration
Continuous Delivery

DevOps Tools





These are some Basic Tools for the teams just starting with DevOps

Tools available for DevOps

Got Questions ?

Connect with me

Email: pradeeppatel2k17@gmail.com

LinkedIn:https://www.linkedin.com/in/pradeeppatelpmp/