


MARCH 2025

Cloud & DevOps

Your journey has begun!

Agenda

- 1 From user story to production**
Feature Journey
- 2 So... DevOps you say?**
What does really mean "DevOps"
- 3 Let's think about it**
Group Exercise
- 4 How we solve it @ Deloitte**
C. A. L. M. S.
- 5 Keeping on track**
Misconceptions about DevOps
- 6 At the end of the day**
People, Process & Technology



From User Story to Production

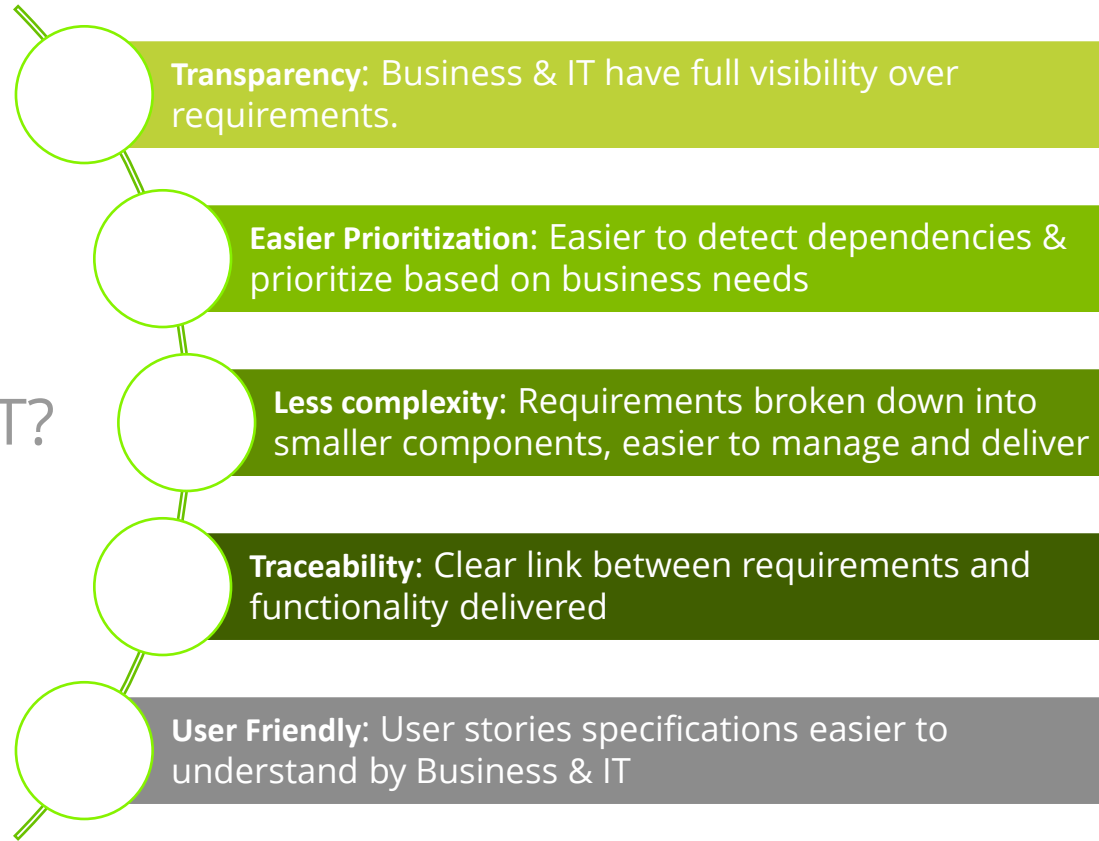
The Journey

Delivering high quality software is not a straight line...



Requirements (1/1)

Requirements should be easily understood by everybody and the benefit of it



SO WHAT?

Project Management



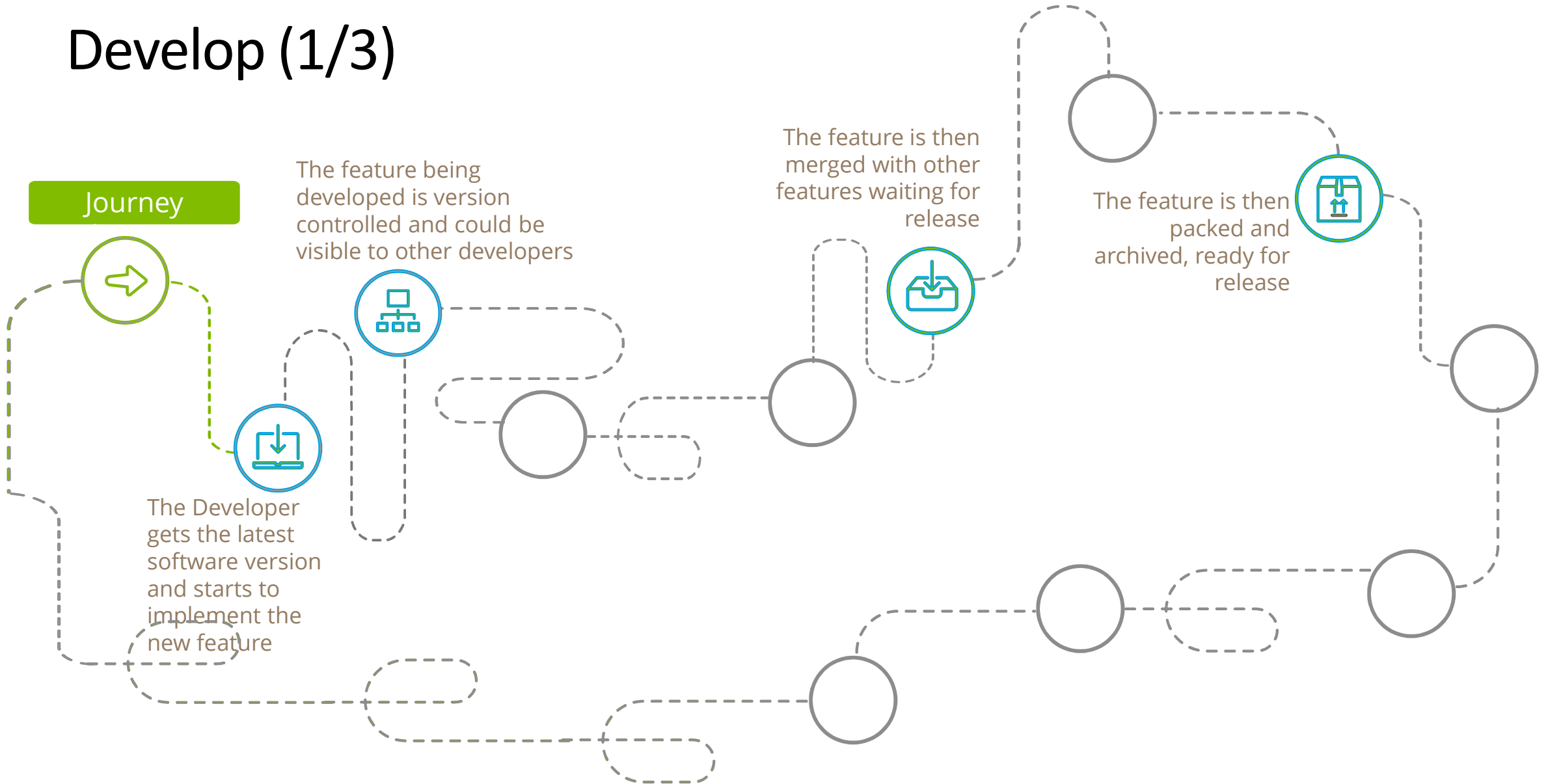
Design Requirements



Documentation

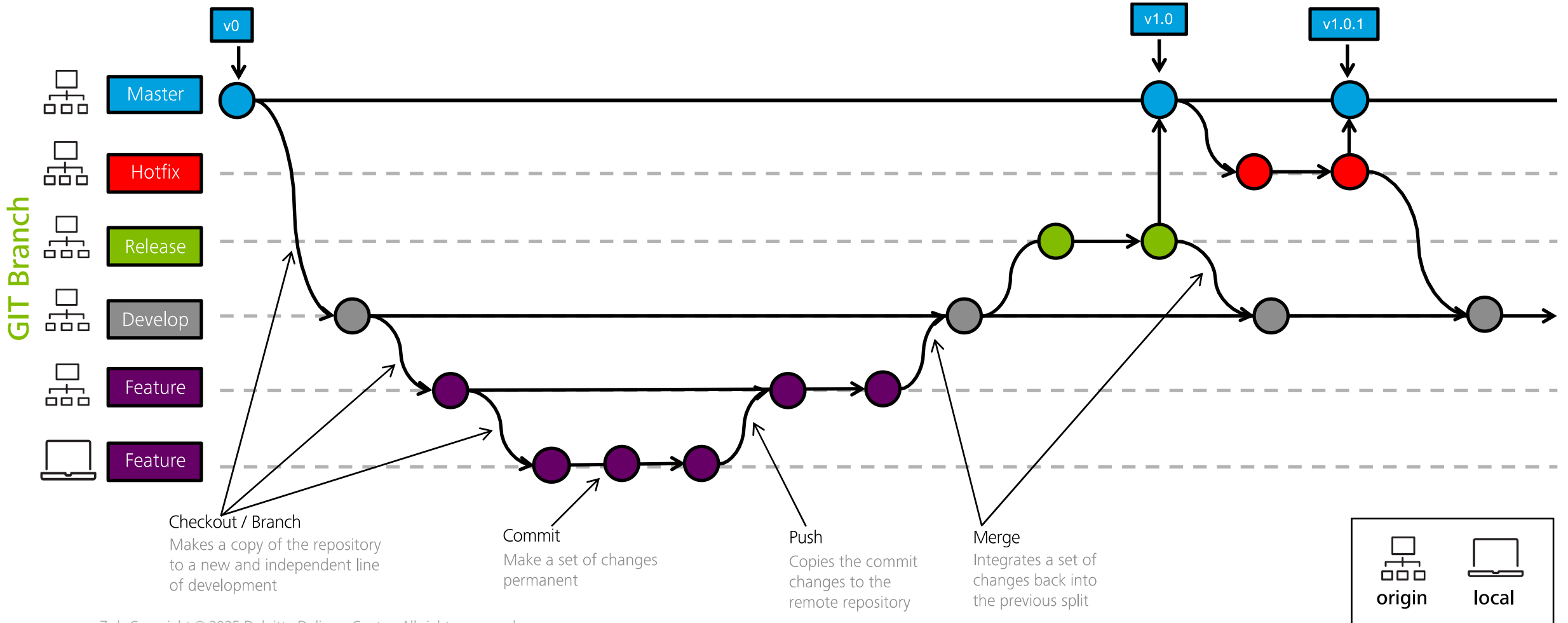


Develop (1/3)



Develop (2/3)

Understanding GIT, a Version Control System



Develop (3/3)

A recipe for best dealing with your code



Source Code Management



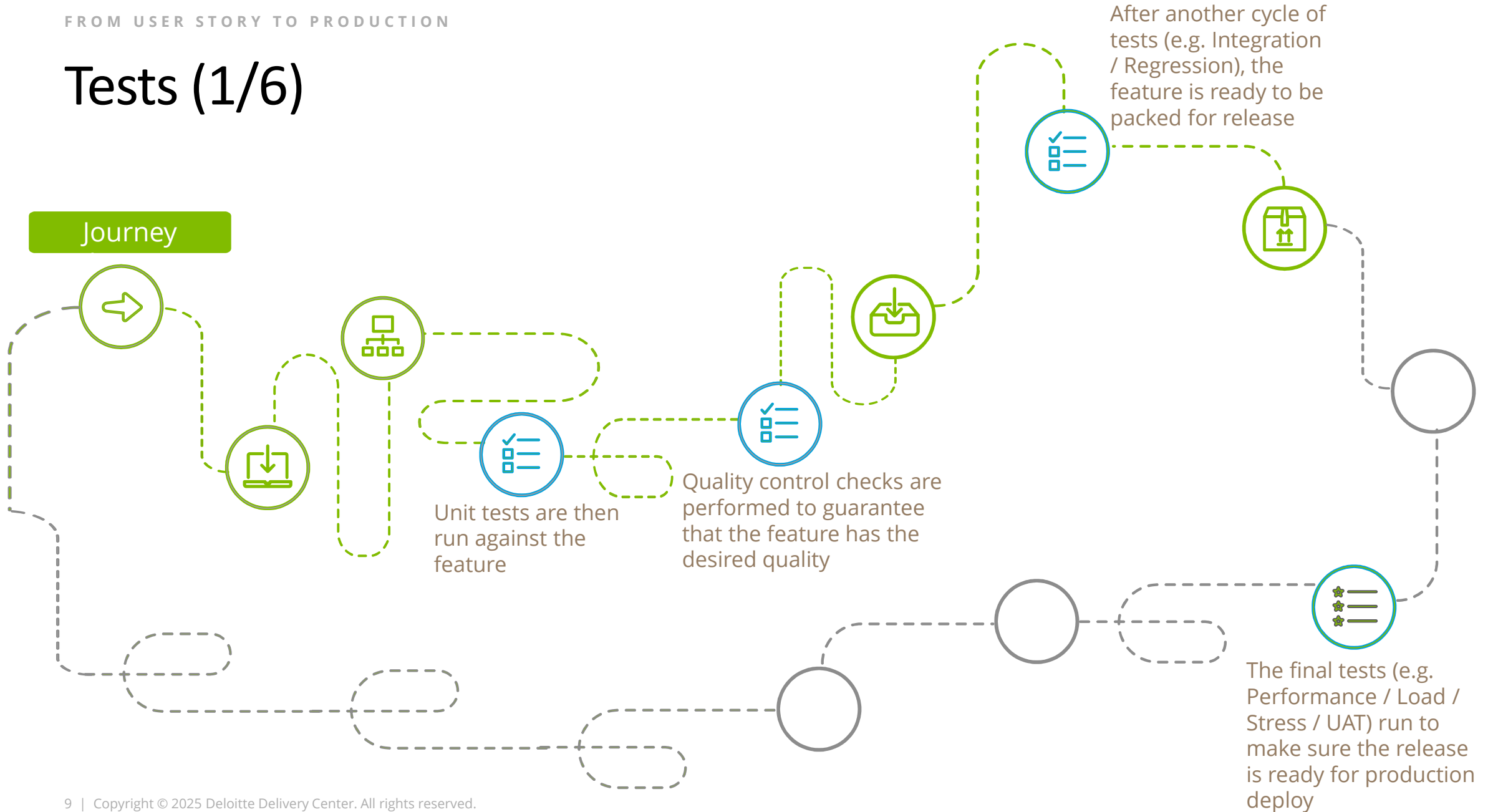
Version Control System



Build

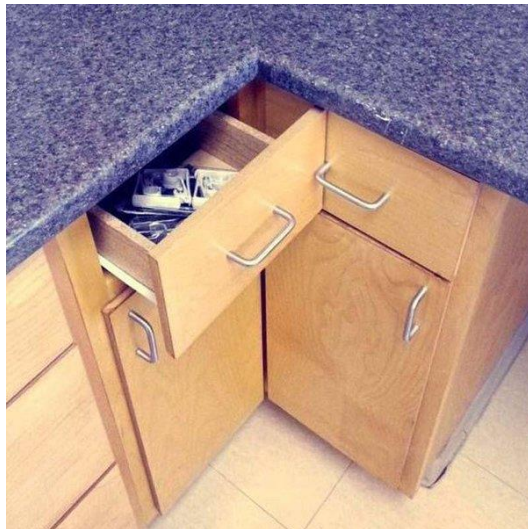


Tests (1/6)



Tests (2/6)

Why tests are important?



Tests (3/6)

Why tests are important?

- In April 2015, Bloomberg terminal in London crashed due to software glitch affected more than 300,000 traders on financial markets. It forced the government to postpone a 3bn pound debt sale.
- The hole in the ozone layer over Antarctica remained undetected for a long period of time because the data analysis software used by NASA in its project to map the ozone layer had been designed to ignore values that deviated greatly from expected measurements.
- In 2015 fighter plane F-35 fell victim to a software bug, making it unable to detect targets correctly.
- China Airlines Airbus A300 crashed due to a software bug on April 26, 1994, killing 264 innocent lives.
- In 1985, Canada's Therac-25 radiation therapy machine malfunctioned due to software bug and delivered lethal radiation doses to patients, leaving 3 people dead and critically injuring 3 others.
- For nine hours in January 1990 no AT&T customer could make a long-distance call. The problem was the software that controlled the company's long-distance relay switches—software that had just been updated. AT&T wound up losing \$60 million in charges that day—a very expensive bug.

Tests (4/6)

Test types

Unit tests

Testing individual methods and functions of the classes, components or modules – executed during build process.

Integration tests

Testing if different modules or services work well together.

End-to-end tests

Replicates a user behavior with the software in a complete application environment.

Acceptance tests

Formal tests executed to verify if a system satisfies its business requirements.

Performance tests

Checks the behaviors of the system when it is under significant load.

Smoke tests

Check basic functionality of the application. They are meant to be quick to execute, and their goal is to give you the assurance that the major features of your system are working as expected.

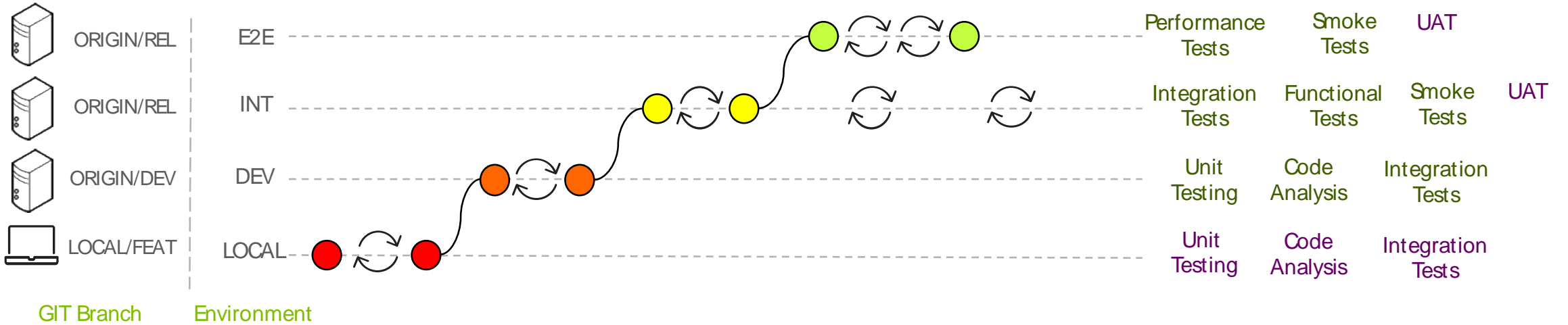
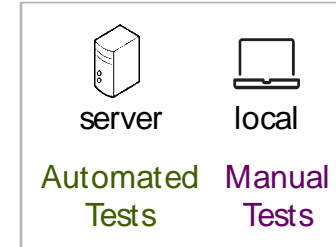
Cloud Environment Tests

Having a Cloud infrastructure, we need to test our own IaC– Chaos Engineering



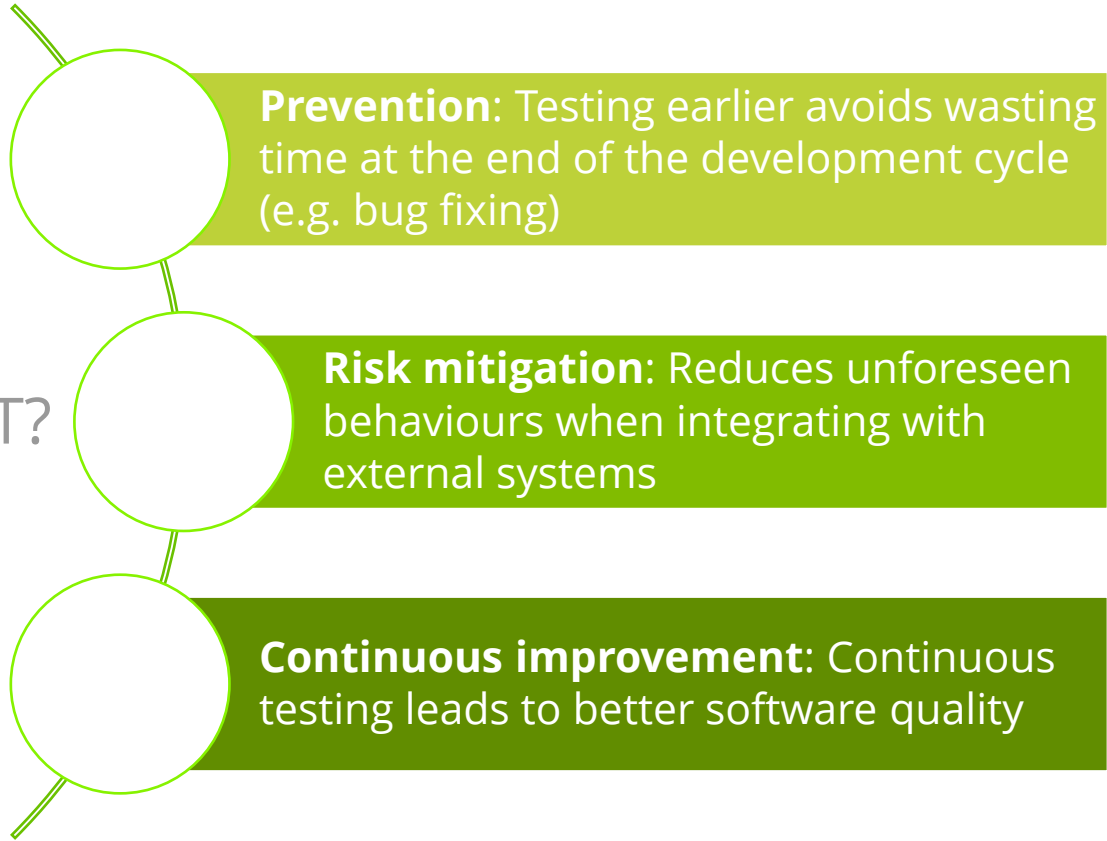
Tests (5/6)

Test types



Tests (6/6)

SO WHAT?



Unit Testing

JUnit.net



Code Analysis



Functional Testing



Load / Performance Testing



Chaos Engineering



Deploy (1/5)



Deploy (2/5)

Recreate

Version A is terminated then version B is rolled out.

Ramped

Version B is slowly rolled out and replacing version A.

Blue/Green

Version B is released alongside version A, then the traffic is switched to version B.

Canary

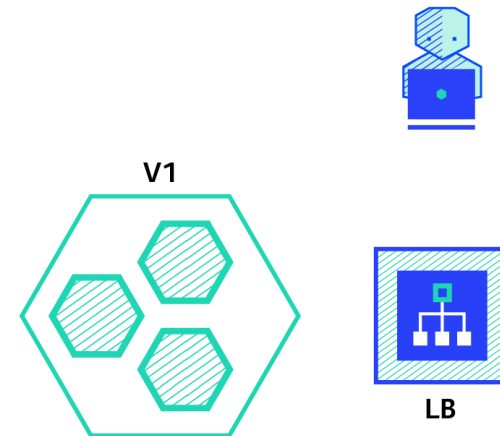
Version B is released to a subset of users, then proceed to a full rollout.

A/B Testing

Version B is released to a subset of users under specific condition.

Shadow

Version B receives real-world traffic alongside version A and doesn't impact the response.



Deploy (3/5)

Continuous integration

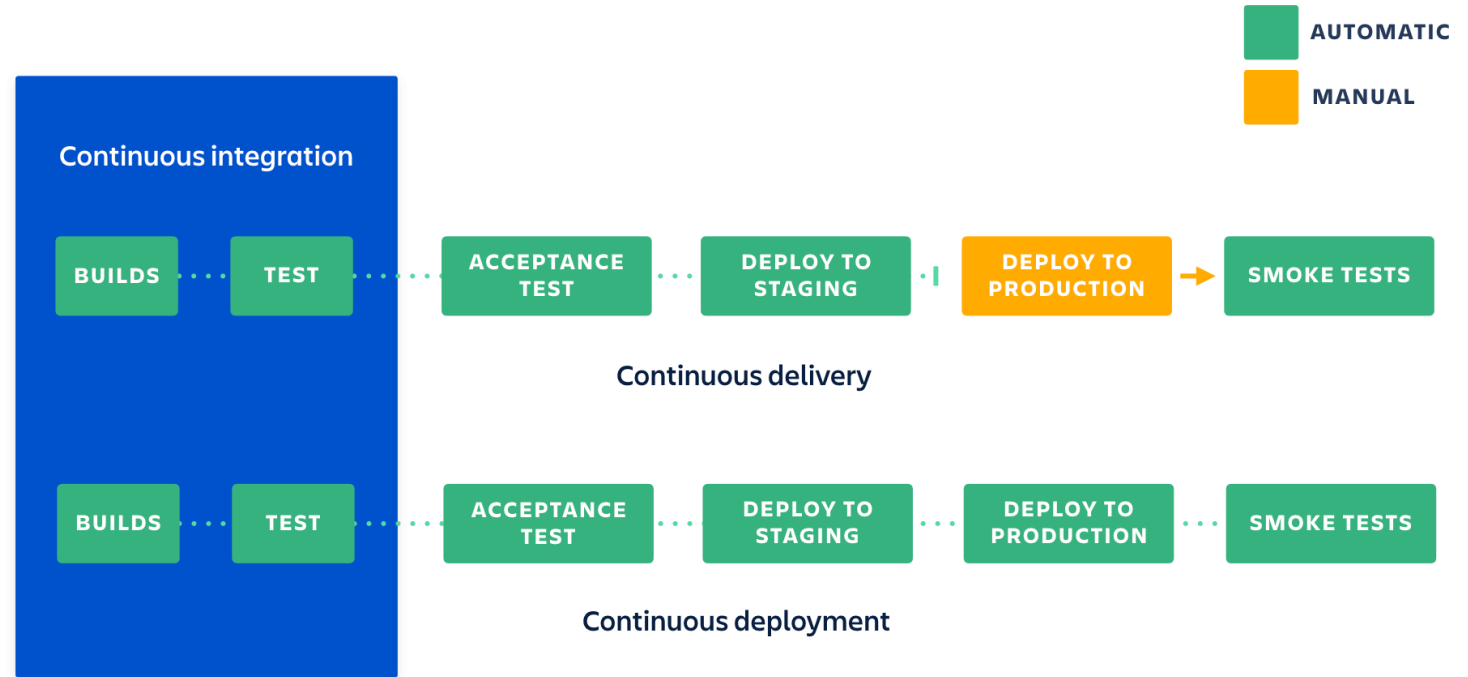
- Developers **merge their changes** to main branch as often as possible;
- Changes are validated by creating a build and running **automated unit tests** against the build.

Continuous delivery

- **Automate the release** process;
- **Release to production** requires a human approval.

Continuous deployment

- **Automate the release** process even for production environment;



Deploy (4/5)

From the Monolith

- Big “chunk” of code
- Unable to release new features independently
- Application will be 100% unavailable
- Requires more team coordination

To More distributed application

- Having your application more “spitted” / smaller modules / features
- Minor Blast radius
- Deploy independently
- Less costs

Serverless



Amazon Elastic Kubernetes Service (Amazon EKS)



AWS Fargate



AWS Lambda



Azure AKS



Azure Container Instance



Azure Function



GCP Kubernetes Engine



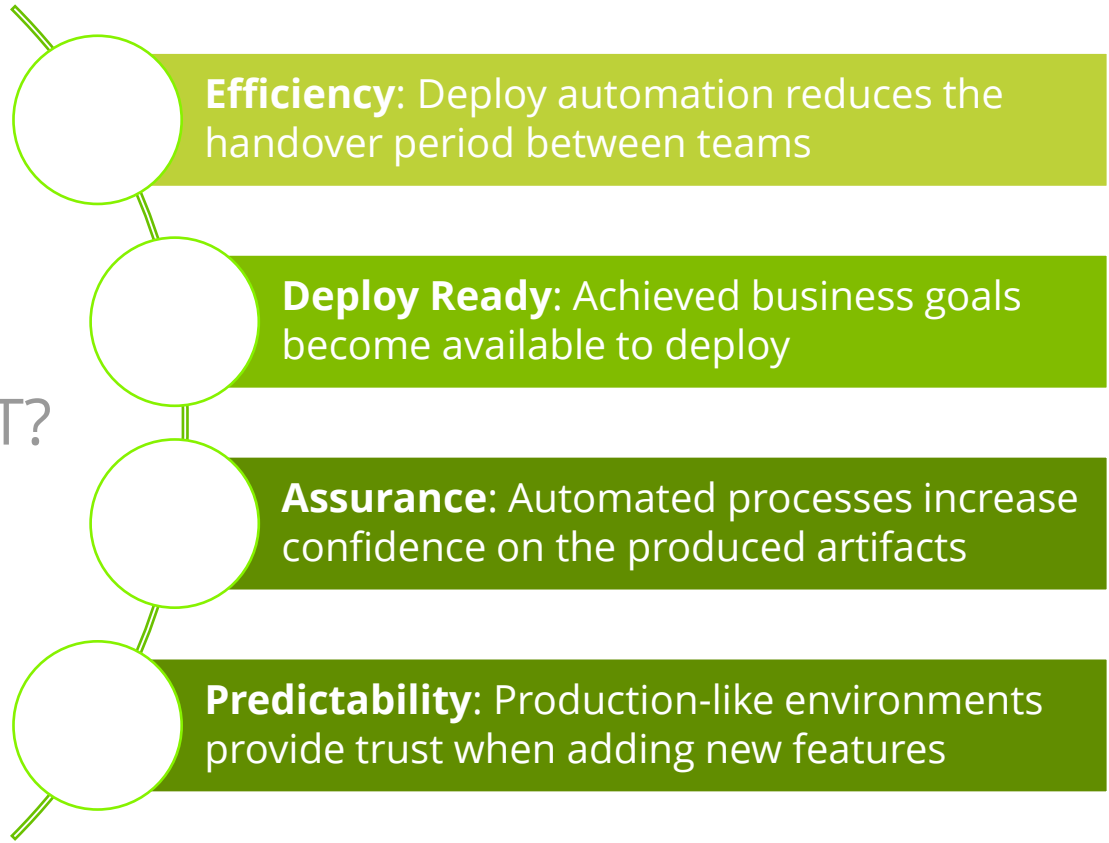
GCP Cloud Run



GCP Cloud Function

Deploy (5/5)

SO WHAT?



Continuous Integration



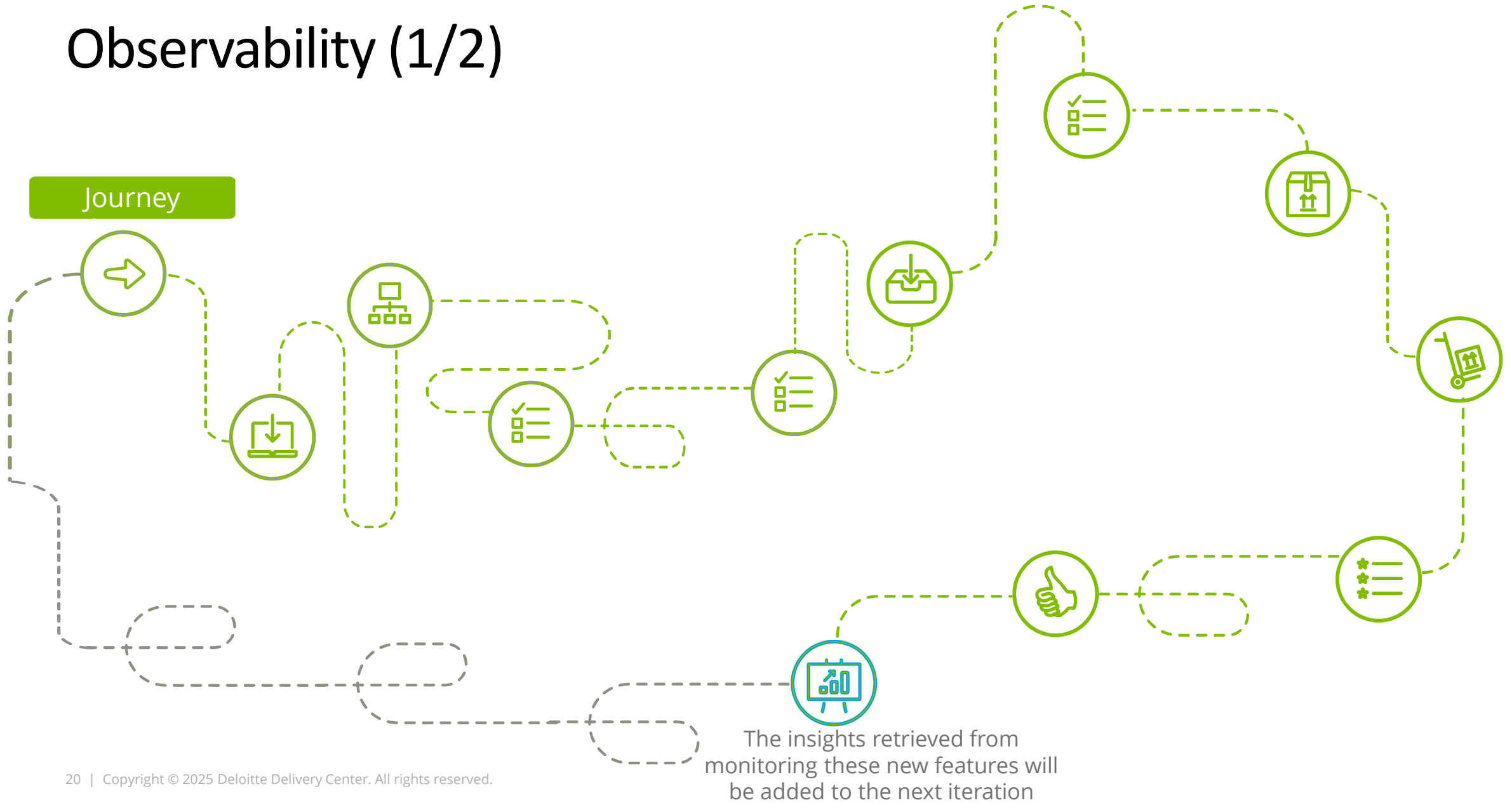
Configuration Management



Repository Manager



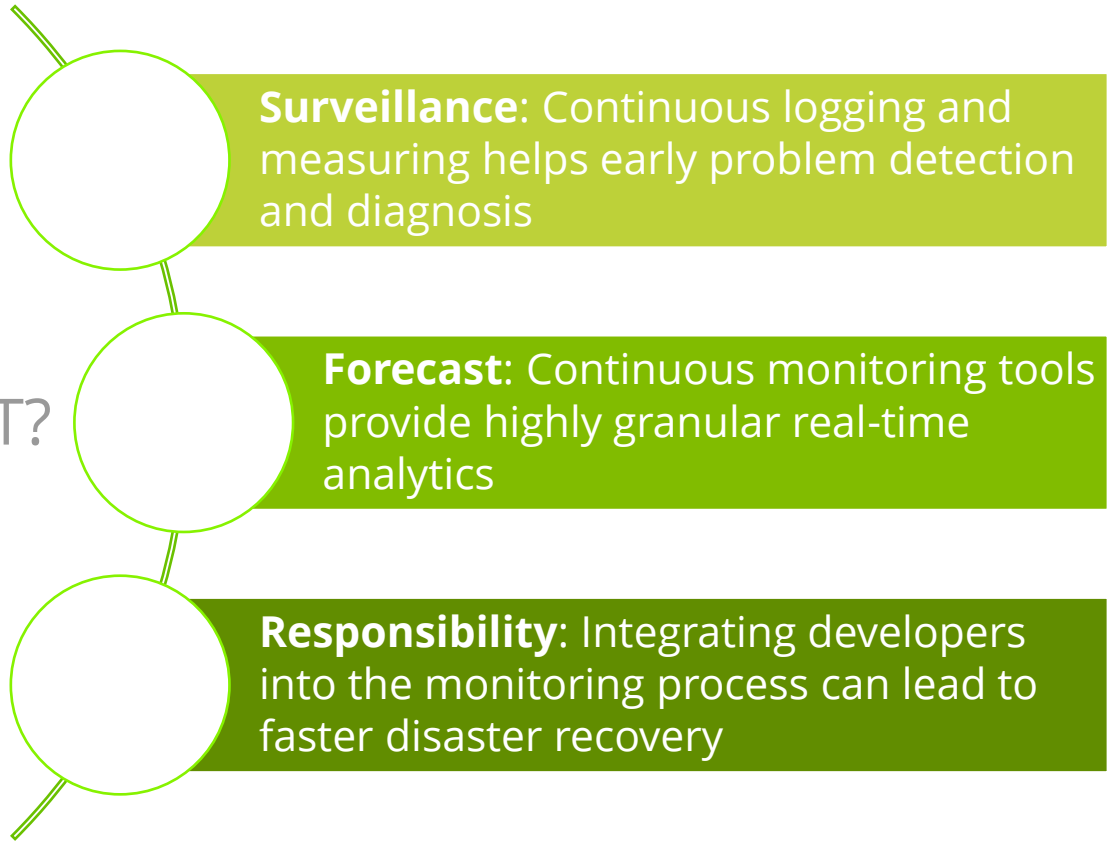
Observability (1/2)



Journey

The insights retrieved from monitoring these new features will be added to the next iteration

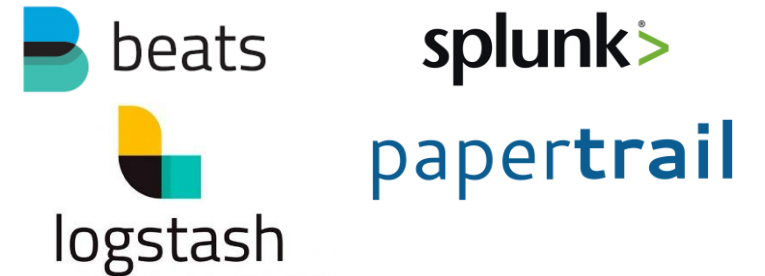
Observability (2/2)



Continuous Monitoring

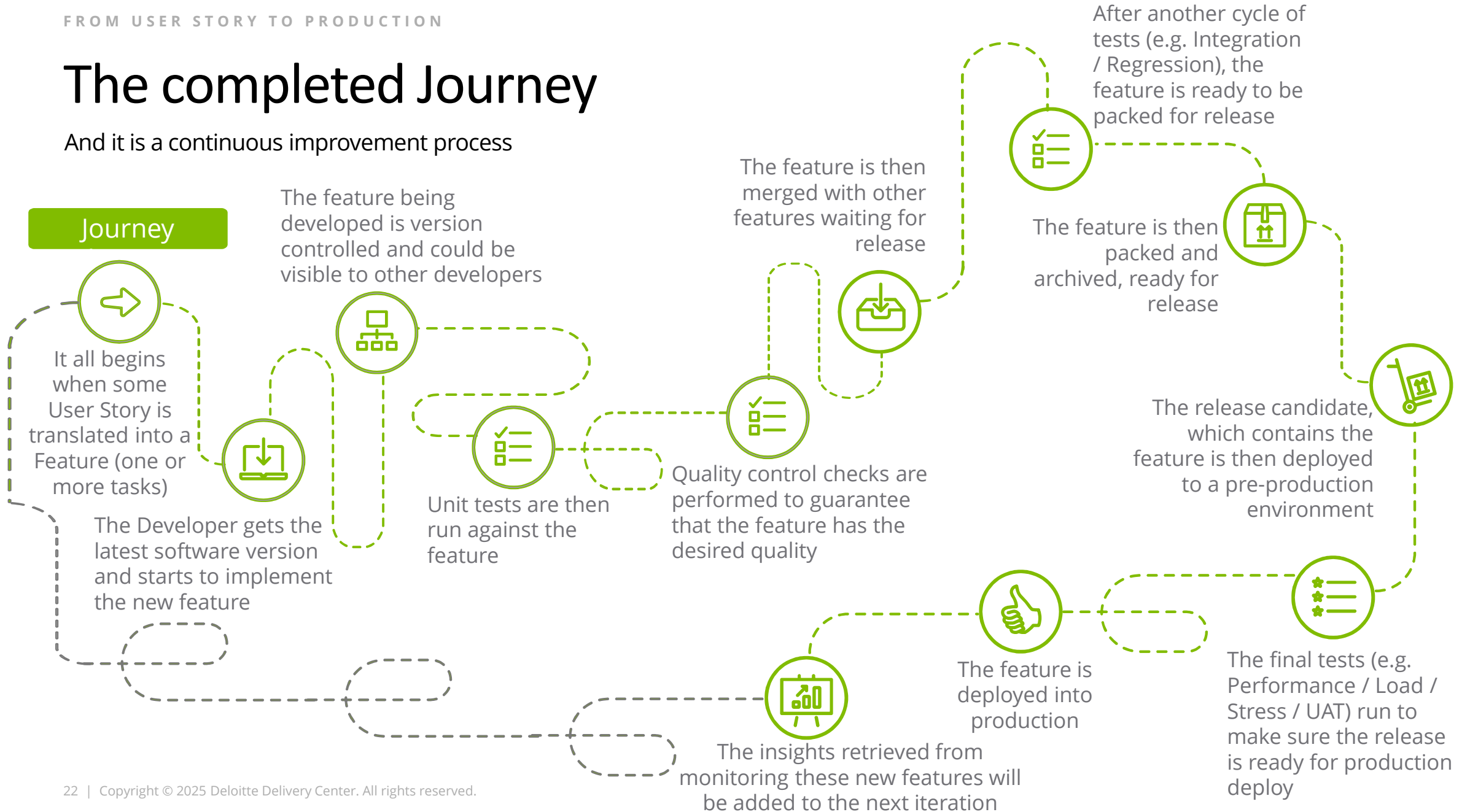


Logging



The completed Journey

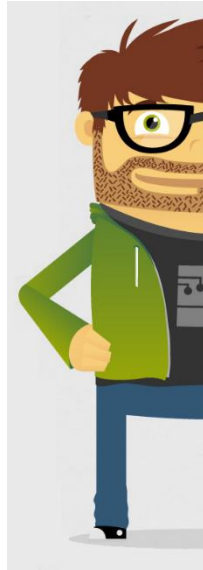
And it is a continuous improvement process





So... DevOps you say?

What does the word “DevOps” mean?



DEvelopers



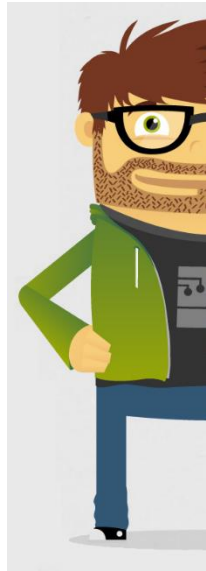
OPerations



DEVOPS

YOU JUST MERGE THE TWO WORDS! RIGHT?!

What does the word “DevOps” mean?



DEvelopers



OPerations

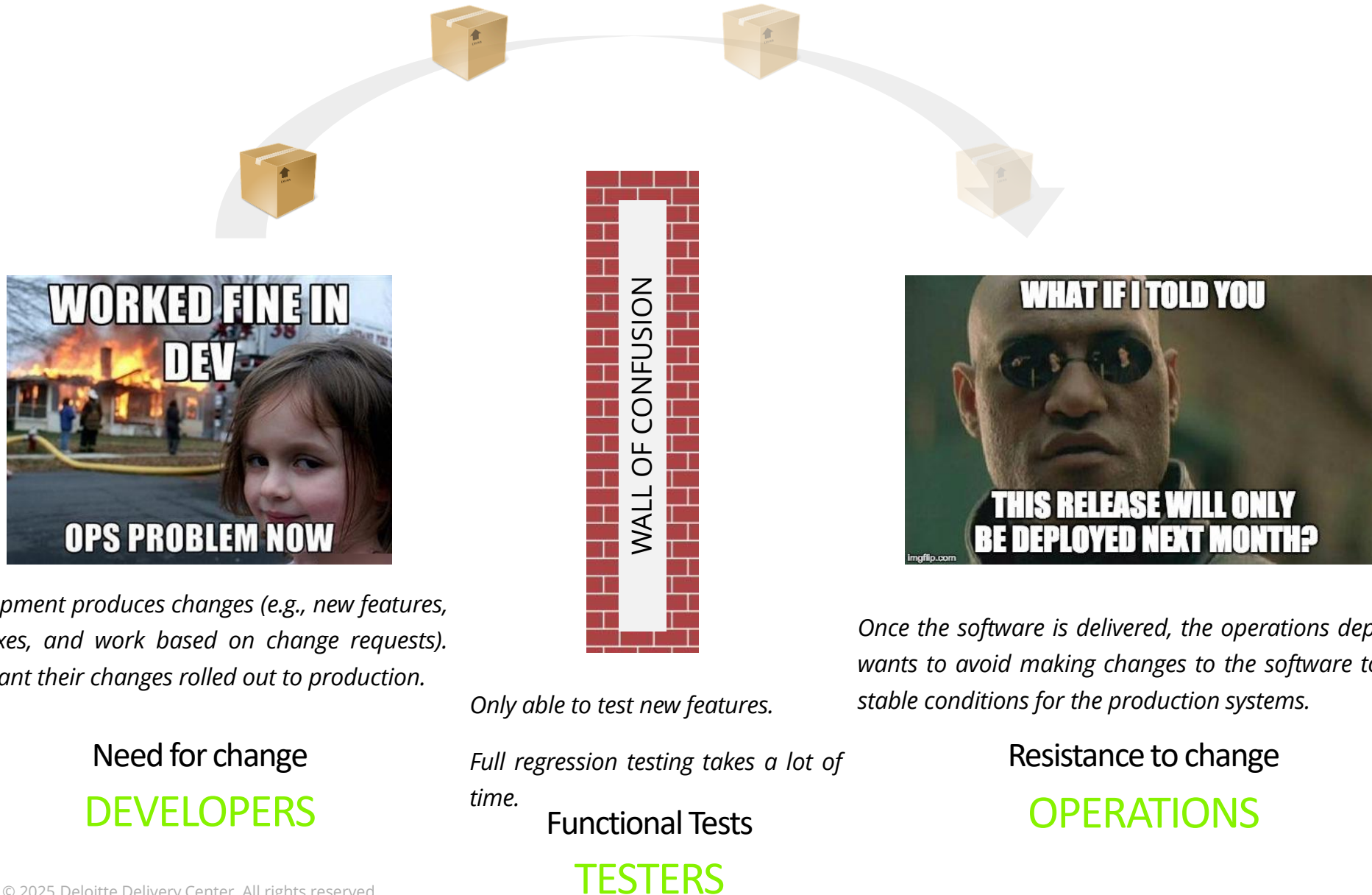


DEVOPS

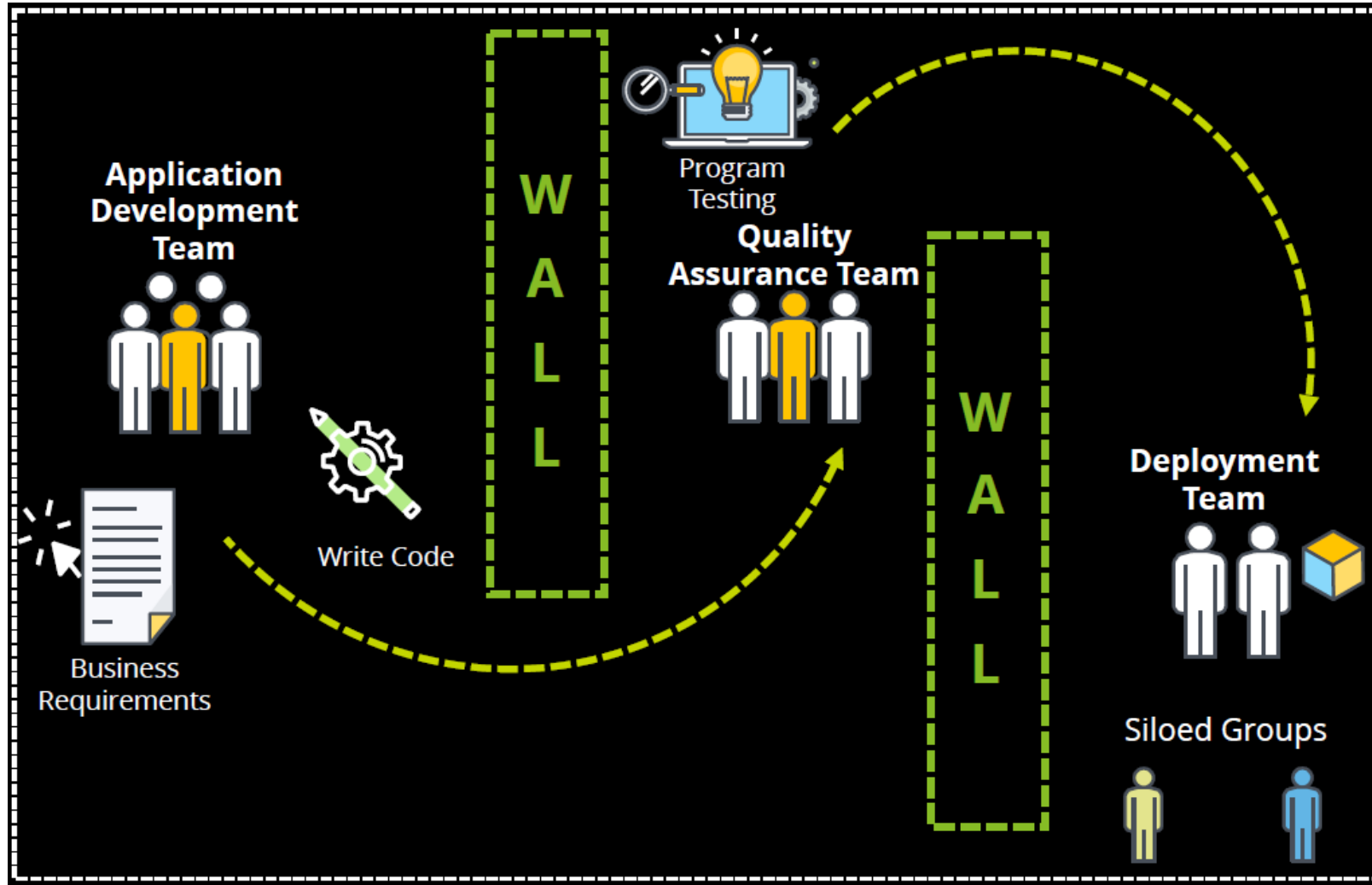
WELL... HERE IS THE PROBLEM...

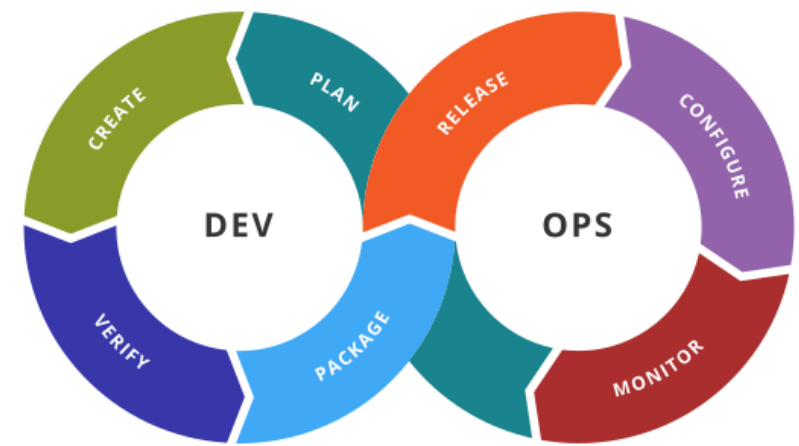
Can you guess what the
problem is?

SO... DEVOPS YOU SAY?



The Conflict





Let's think about it

Groups of 4/5

Scenario:

You are part of a development team that is facing frequent delays in software delivery.

Group tasks: Identify the **bottlenecks**: What are the possible main points of delay in the current process?

Propose DevOps solutions: How can DevOps practices help resolve these bottlenecks?

10 min – pick 2 groups to share the results



How we solve it
@Deloitte?

So Keep

A framework to measure the adoption of DevOps into your organization.

C

Adapt your Culture

A

Automate what you can

L

Simple, lean, focus on value

M

Measure everything

S

Share what matters / Collaborate

How we do it @ Deloitte?



Culture

DevOps isn't a simple process or approach for software development – It's a cultural change.

Increasing **collaboration** by reducing team silos;

Go from function-based teams to **product oriented teams**.



What we propose:

- “You build-it , you run-it” approach.
- Shift-left approach for test earlier
- Build cross-functional teams
- Create resources for business involvement

How we do it @ Deloitte?



Automation

Automate as much as you can in order to reduce repeatable tasks prone to manual errors due to human intervention.

It helps creating reliable systems.



What we propose:

- Leverage current DevOps tools
- Evaluate different DevOps tools in order to fill gaps
- Implement Continuous Delivery;
- Implement Continuous Testing;
- Implement Continuous Integration.

How we do it @ Deloitte?



Simple, lean, focus on value

Keep your features small, in order to deliver it faster, with quality.

Break up the monolith, in order to ship self-contained piece of software instead of a big-chunk.

Stay focus on what bring most value.



What we propose:

- Continuous deployment
- Release management
- Change mind-set from "Achieving perfection" to "Good-enough"

How we do it @ Deloitte?



Measure

We need to stay focus, constantly learning, in order to improve our skills and processes.

How long did it take from development to production? Is it acceptable?

How often recurring bugs occur? Is it to frequent?

How long does it take to recover from a system crash ? Is it fast?



What we propose:

- Deployment frequency
- Change lead-time
- Mean time to recover
- Continuous monitor
- Change rate failure
- Technical debt

How we do it @ Deloitte?



Share

Share knowledge and keep communication channels open.

Learn and share.

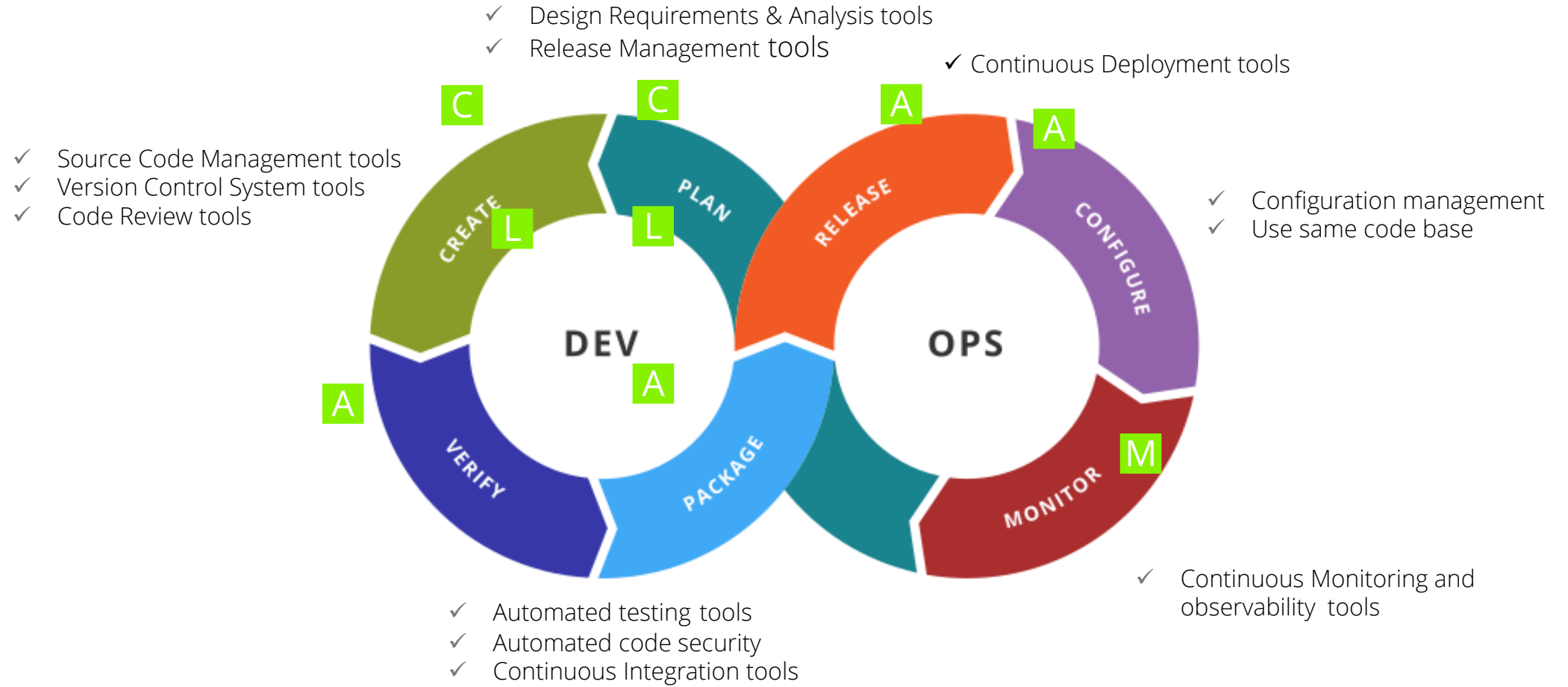


What we propose:

- Easy communication channels
- Bridging instead of dividing
- Accelerate on Onboarding processes

The Repeatable Process

Where to apply CALMS framework.





Keeping on track

Misconceptions

#1 DevOps is Only for Development and Operations teams.

DevOps is for every one.

- Usually, the term DevOps is incorrectly (only) associated with “Programmers” on the Developers side, and “Sys Admins” on the Operations side;
- Even though they could be considered the main “stakeholders”, others should be considered: QA engineers, database administrators, network technicians, security engineers, etc.

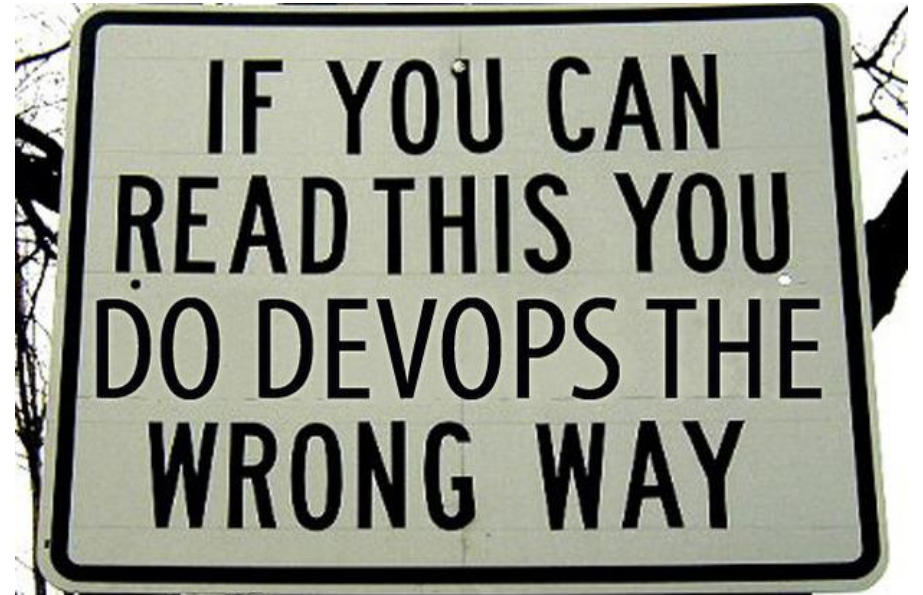


Misconceptions

#2 DevOps has an end state or goal

It's a continuous improvement process.

- In DevOps, there is no such thing as a “one-size-fits-all-devops-way”;
- There is no “right way” or “wrong way”;
- It must be the solution that fits best the organization.

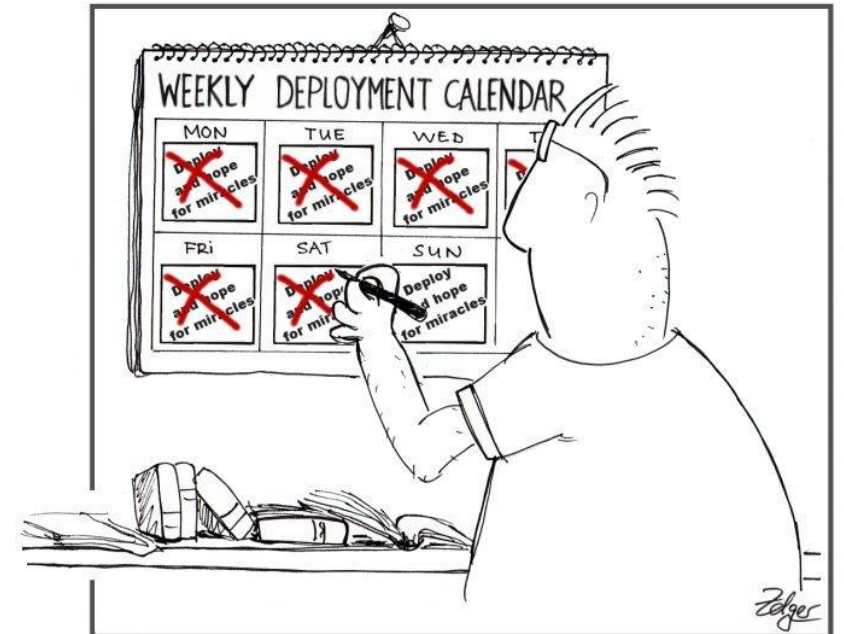


Misconceptions

#3 DevOps is only successful for new / green field applications

DevOps is suitable for all types of projects where you can continuously keep improving it.

- “10 deploys per day” – John Allspaw & Paul Hammond @ Velocity 2009
- DevOps focus on people and software development... So remember it's technology-agnostic...
- And how about mainframe core applications?



Misconceptions

#4 DevOps it's all about tools

DevOps is a culture.

- Tools help you achieve your goals, but they are not the only ingredient.
- Being a cultural movement, DevOps is technology-agnostic.
- You cannot “buy” DevOps.

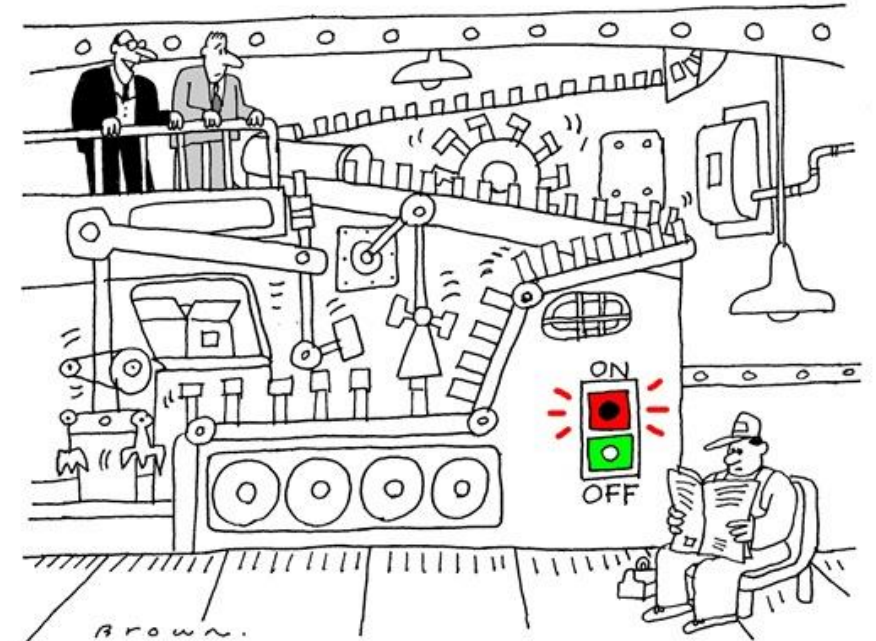


Misconceptions

#5 DevOps is all about automation

DevOps is team work.

- One of DevOps purposes is to improve how people work together.
- The automation is just a way of creating a more efficient workflow by freeing up a person by automating some repetitive tasks.



"We're almost fully automated now."

Misconceptions

#6 DevOps is a job title

DevOps is a culture.

- Maybe you are a System Admin that can code;
- Maybe you are a Developer that have some SysAdmin skills;
- Is it your tech skills that will identify a solution, or is your capacity as a problem solver / team player?

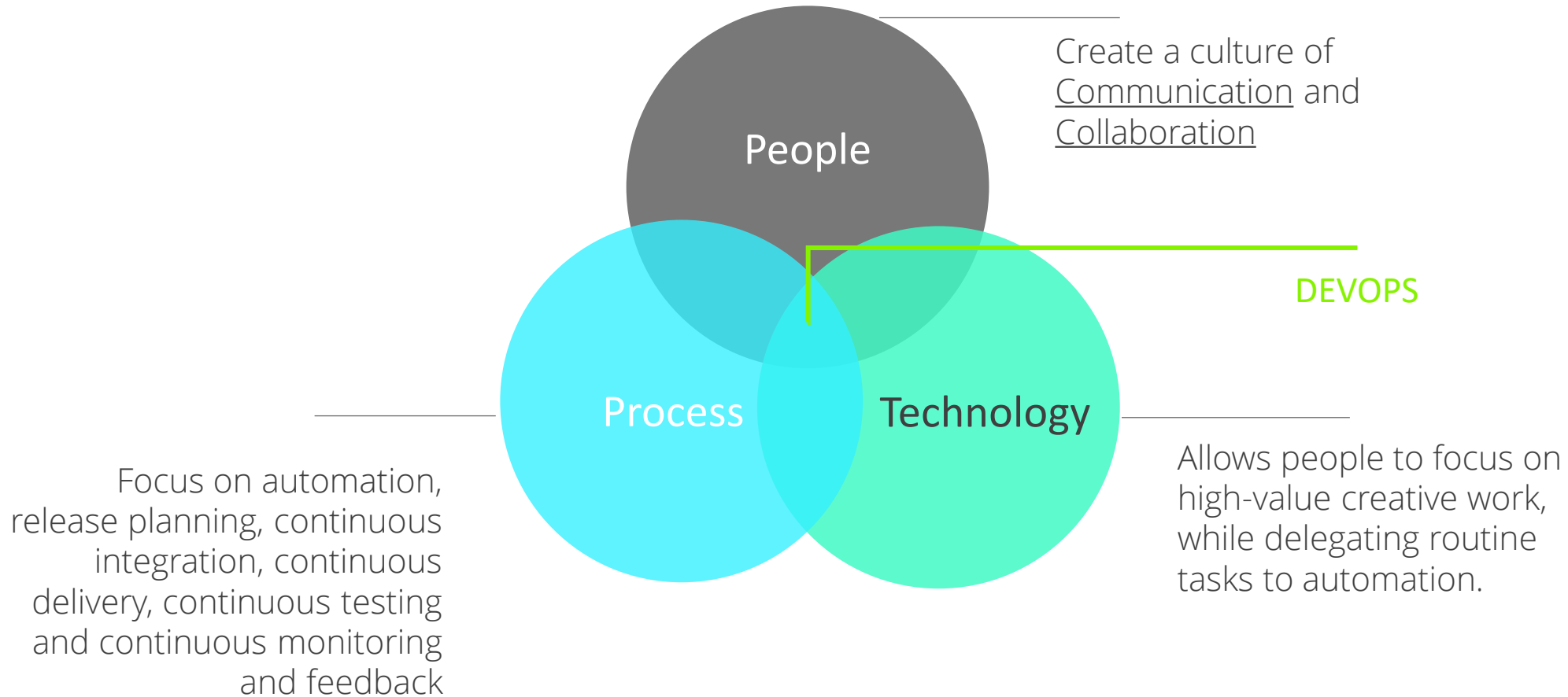




At the end of the day...

... It's all about

An interception of three sub sets



DevOps Practitioner Skills

What makes a good DevOps Practitioner



- Collaboration, soft skills, and communication
- **Customer-first mindset**
- Source control systems
- Testing
- DevOps tools and technologies, enabling CI/CD/CM
- Cloud and infrastructure
- Security training

Stories of Success

Amazon

You build it, you run it

- Developers are given operational responsibilities, which enhanced the quality of the services, both from a customer and a technology point of view;
- All engineers MUST develop web service APIs to share all data internally across the entire organization. No exceptions;
- Teams MUST communicate with each other using the APIs;
- It doesn't matter the technology, as long as they have service interfaces that can be externalizable;
- "Anyone who doesn't do this will be fired."

Netflix

Simian Army and the automating failure

- Automated processes to achieve consistency and efficiency;
- Simian Army – Series of tools that continuously and in an automated way, test the environments. Some example tools:
 - Chaos Monkey – Randomly disable production instances to ensure no customer impact in case of failure
 - Latency Monkey – Artificial delays (with monitoring) to simulate service degradation
 - Conformity Monkey – Finds instances that don't adhere to best-practices and shut them down;
 - Security Monkey – Finds security violations and terminates the instances
 - Chaos Gorilla – Like Monkey Chaos but an entire Amazon availability zone.

Sony Pictures Entertainment

From months to minutes

- Manual processes and other hurdles typically resulted in a months-long delay between completion of software development and delivery;
- Automated cloud delivery system composed of open source tools and SaaS solutions;
- Continuous delivery model, DMG has cut down its months-long delivery time to just minutes.



Thank you.

Fábio João

Tech Manager

Contact: fjoao@deloitte.pt

This publication contains general information only, and none of the member firms of Deloitte Touche Tohmatsu Limited, its member firms, or their related entities (collective, the "Deloitte Network") is, by means of this publication, rendering professional advice or services. Before making any decision or taking any action that may affect your business, you should consult a qualified professional adviser. No entity in the Deloitte Network shall be responsible for any loss whatsoever sustained by any person who relies on this publication.

As used in this document, "Deloitte" means Deloitte Consulting LLP, a subsidiary of Deloitte LLP. Please see www.deloitte.com/us/about for a detailed description of the legal structure of Deloitte USA LLP, Deloitte LLP and their respective subsidiaries. Certain services may not be available to attest clients under the rules and regulations of public accounting.

Copyright © 2025 Deloitte Deliver Center
All rights reserved. Member of Deloitte Touche Tohmatsu Limited

Deloitte Event | Tech Feedback Survey



Nome do Evento: Workshop Cloud & DevOps U Lusófona
Data do Evento: 19/03/2025

Deloitte.