Introduction to CI/CD

Maaike van Putten
Software developer & trainer
www.biltup.com

Overview

Session 1

Introduction to DevOps & DevOps Philosophy

Session 2

Introduction to CI/CD (Continuous Integration / Continuous Delivery)

Session 3

Implementing CI

Learning objectives

P Understand the concepts of **Continuous Integration** (CI) and **Continuous Deployment** (CD) and how they are the core of the **DevOps environment.**

r Be familiar with the environment that supports CI/CD processes.

Schedule

Intro + theory

More on DevOps +
What is CI/CD?

Exercise

Visualizing the
pipeline

**Debrief +
theory**

Debrief exercise +
Deployment
strategies

Exercise

Thought exercise

**Debrief +
challenges**

Debrief + challenges
Concepts + exercise
if enough time +
mini quiz

O

Key Practices DevOps

Automation

- Automate repetitive tasks to free up human resources

- Implement CI/CD pipelines to streamline code integration and deployment

- Use automation tools such as Jenkins, GitHub Actions, GitLab CI/CD, CircleCi, Azure DevOps

Continuous improvement and feedback loops
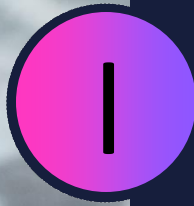
I Measure performance

G Collect user feedback

♾ Iterative improvements to implement changes on feedback

🔧 Tools like Prometheus, Grafana, ELK stack

Benefits of continuous improvement and feedback loops
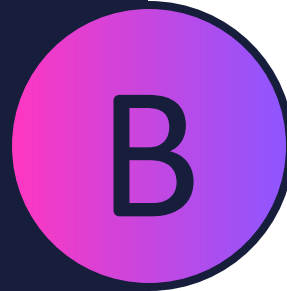
R Higher quality

R Proactive issue resolution

e Informed decision-making
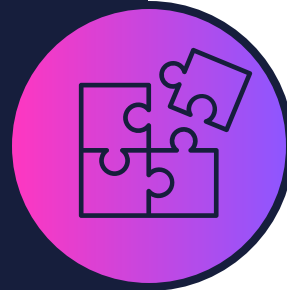
Collaboration between development and operations

X Break down silos to foster cross-team interaction

B Shared goals by aligning objectives and KPIs

O Cross-functional teams that blend skills and expertise

Collaborative tools: Slack, Jira, Confluence

Benefits of collaboration between development and operations

S Improved communication

M Faster issue resolution

t Increased innovation

CALMS framework

(n) **Culture >>** collaboration and trust

(h) **Automation >>** streamline processes

(n) **Lean >>** eliminate waste

(l) **Measurement >>** data-driven decisions

(U) **Sharing >>** knowledge and success stories

DevOps maturity models

Levels of maturity:

- Initial (Ad Hoc)
- Managed
- Defined
- Measured
- Optimized

DORA metrics

j Deployment frequency

N Lead time for changes

Mean time to recovery (MTTR)

Change failure rate

Value stream mapping (VSM)

- What is VSM?
    - Visualizing the flow from idea to delivery

- Identifying bottlenecks
- Improving processes

DevOps toolchains and ecosystems

Tool categories:

- Planning and collaboration

- Source code management

- Continuous integration/delivery

- Monitoring and logging

Integrations and ecosystems

Exercise

Thought experiment: incorporating DevOps

What is Continuous Integration?

The integration of developers' code in a shared codebase multiple times per day

Not working on separate branches per team for too long, but same repository, so work is integrated continuously

Not that new: idea already existed in the 90s

Compiling and automated tests of the code should happen after every commit to central repo to detect problems early

Benefits of continuous integration

Integration is less work

Problems are detected early

Less bugs and errors

Higher flexibility, thanks to transparent and visible process

Shorter time to market

Group exercise

What do you need to successfully implement CI?

What doors does successfully implemented CI open?

- Make groups of 3-4

- Research for 10 minutes

- Present your findings in 1-2 minutes

Minimum requirements CI

Well set up source control management system where all the code is present

Tests need to be automated

Process of compiling and testing needs to be automated, commit should trigger this process

When the build fails, fix it immediately

Everyone commits the code at least daily, preferably more often

Everyone can see the status of the code and see all the code

What is Continuous Delivery?

- The possibility to release software to production any time using automated processes

- Automating all the steps necessary for deployment in a pipeline

Benefits of Continuous Delivery

(X) Lower chance of errors

(g) Less effort / time / money for releasing

(h) Easy to reproduce deployment due to automatic management of the environments (and therefore the exact same environments)

(j) Deployments can be done more often and are less risky

(T) Get feedback on new features earlier because TTM is lower

Group exercise

What do you need to successfully implement Continuous Delivery?

What doors does successfully implemented CD open?

- Make groups of 3-4

- Research for 10 minutes

- Present your findings in 1-2 minutes

Minimum requirements Continuous Delivery

- Well implemented CI

- Automated build pipeline

- Automated release pipeline that can be triggered manually, with optionally some manual (quality) checks

- DoD should contain release: feature isn't completed until it's been released

- Transparency: access for everyone to the pipeline, the process and the latest build

- **Bonus:** DevOps environment

- **Bonus:** IaC = infrastructure as code, management and creation of the environments using scripts, to make sure the environments are the same and to rule out unexpected behavior due to environmental differences

What is Continuous Deployment?

The **releasing of software** to production **after every change** to the codebase using **automated processes**

Benefits of Continuous Deployment

B All benefits from CI and Continuous Delivery

⏱ Short TTM

S Early feedback from users

Continuous Delivery vs. Continuous Deployment

Continuous Delivery

Software could potentially be released to production at any given moment
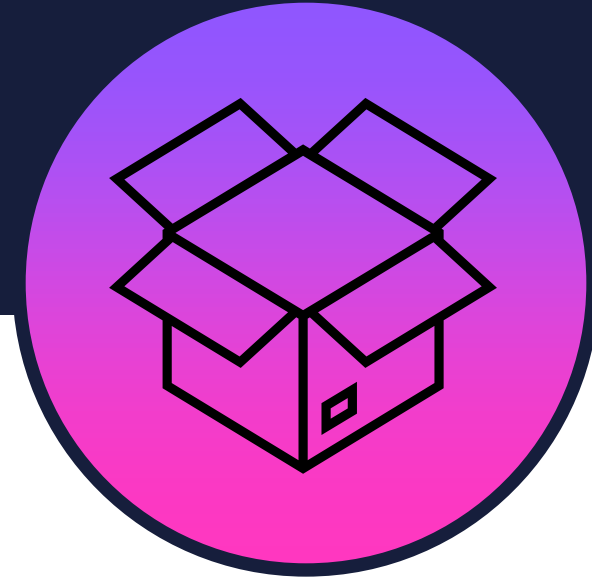
Continuous Deployment

Software is being released to production after every change to the codebase

Minimum requirements Continuous Deployment

b Continuous Integration

Continuous Delivery

a Rollback strategy

Situation without CI/CD

Without continuous delivery
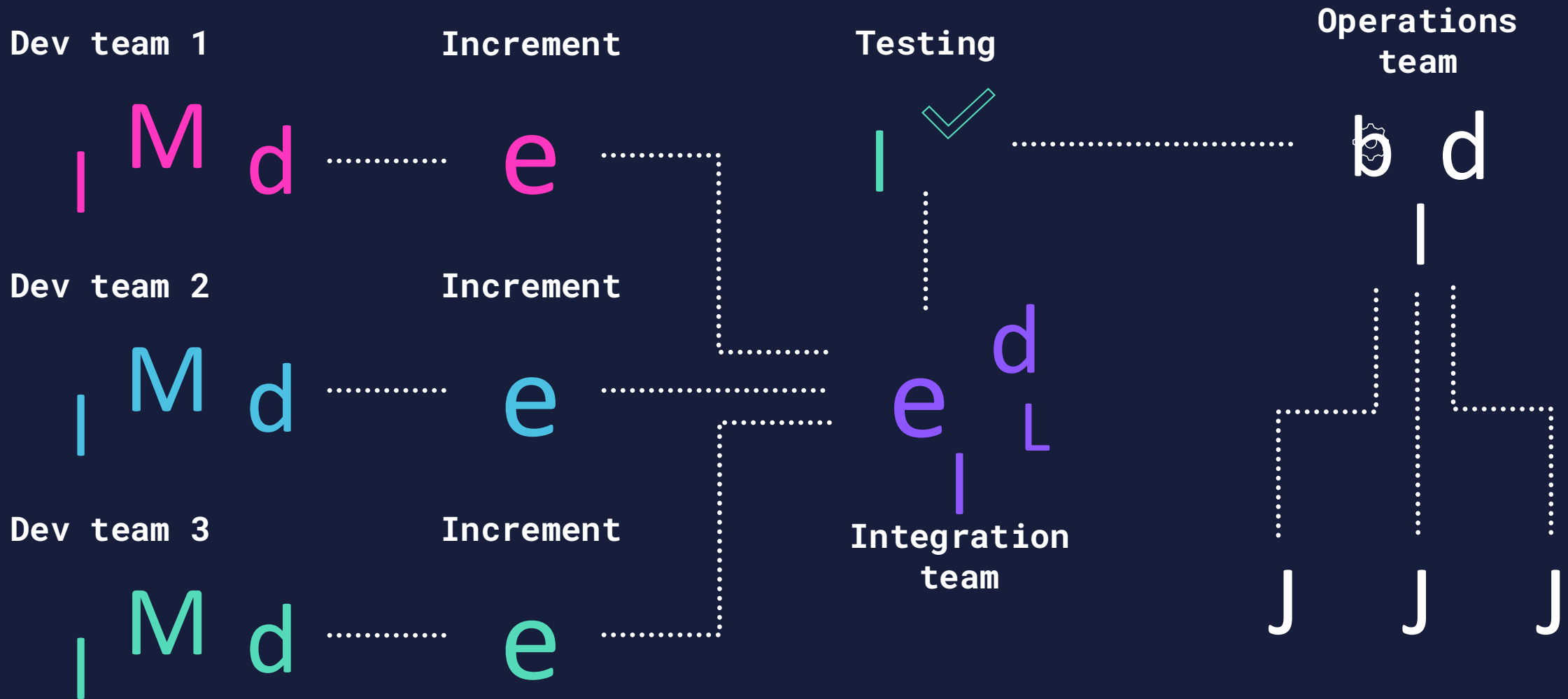
Dev team 1 — Increment — Testing — Operations team

Dev team 2 — Increment

Dev team 3 — Increment — Integration team

Software development
teams working on
different parts

Integration team
merging all the
work

Operations team

Software development teams working on different parts

Integration team merging all the work

Operations team

M d |

M d

M d

Software development teams working on different parts

Integration team merging all the work

M L

Operations team

M d I

M d

M d

J J J J

Dev   Test   Acceptance   Production
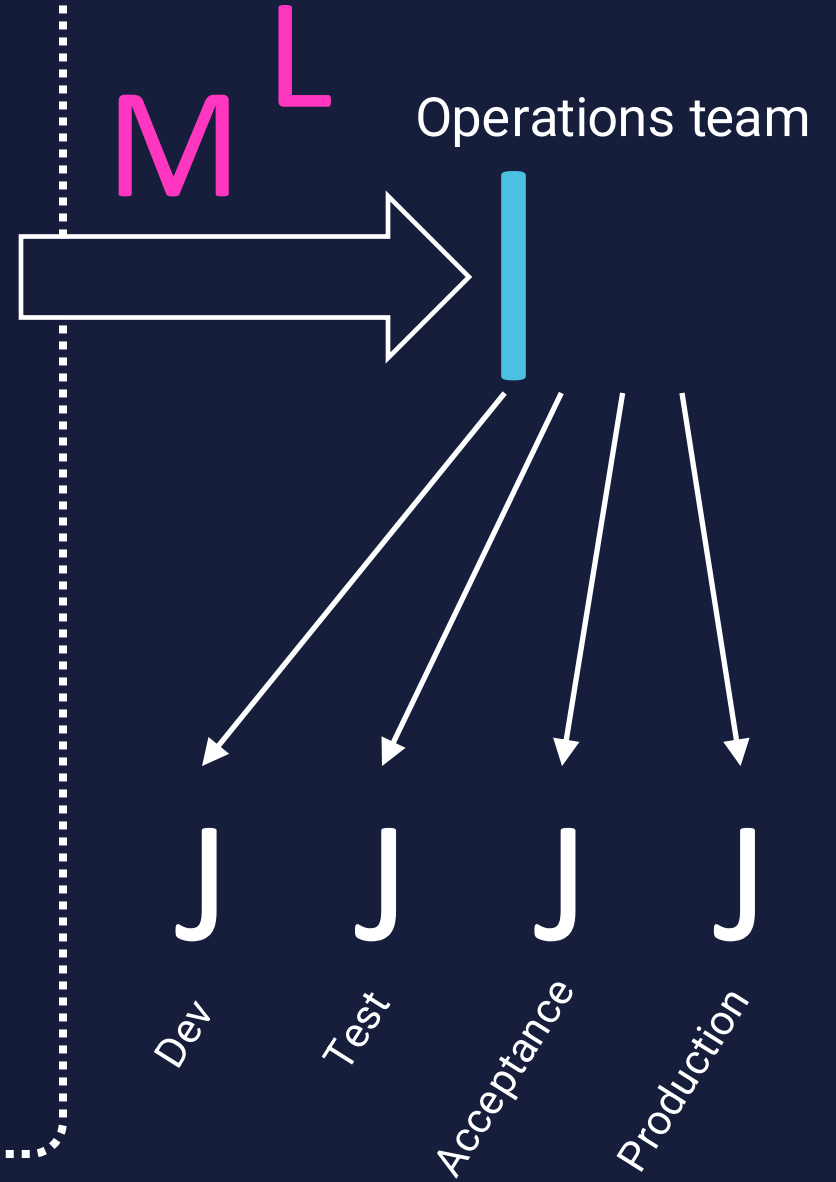
Exercise

Drawing a CI/CD pipeline

With continuous delivery

DevOps team

Unit tests

Build

Deployment

M d

I

✓ B b V

Integration

Behavior tests

d J

Product management → **P** Software development → **I** Integrations / operations / release team → **End users**

C QA
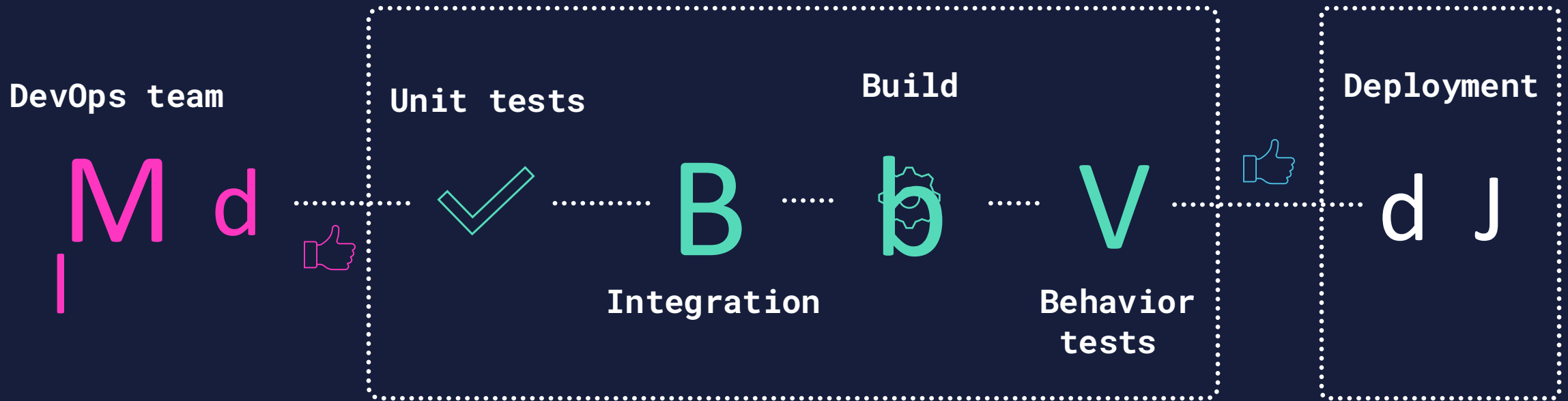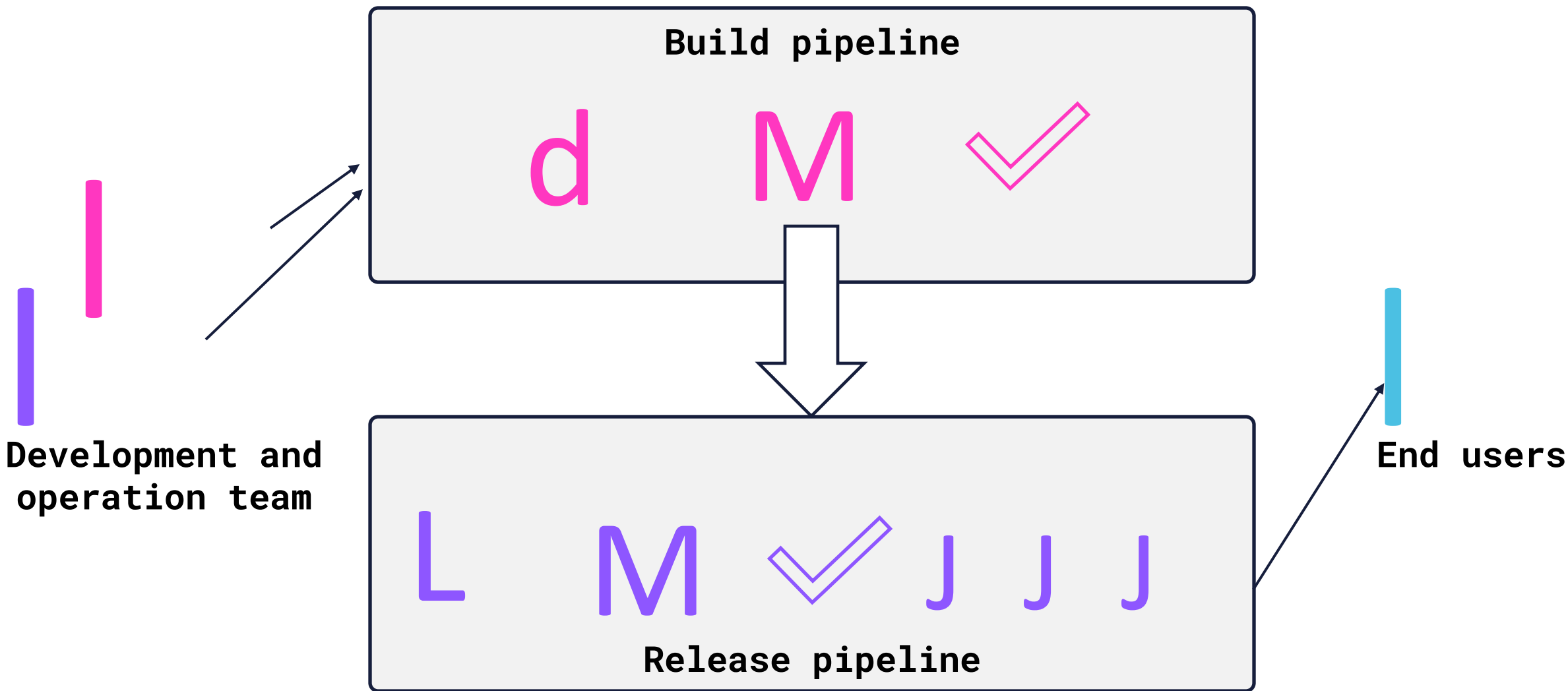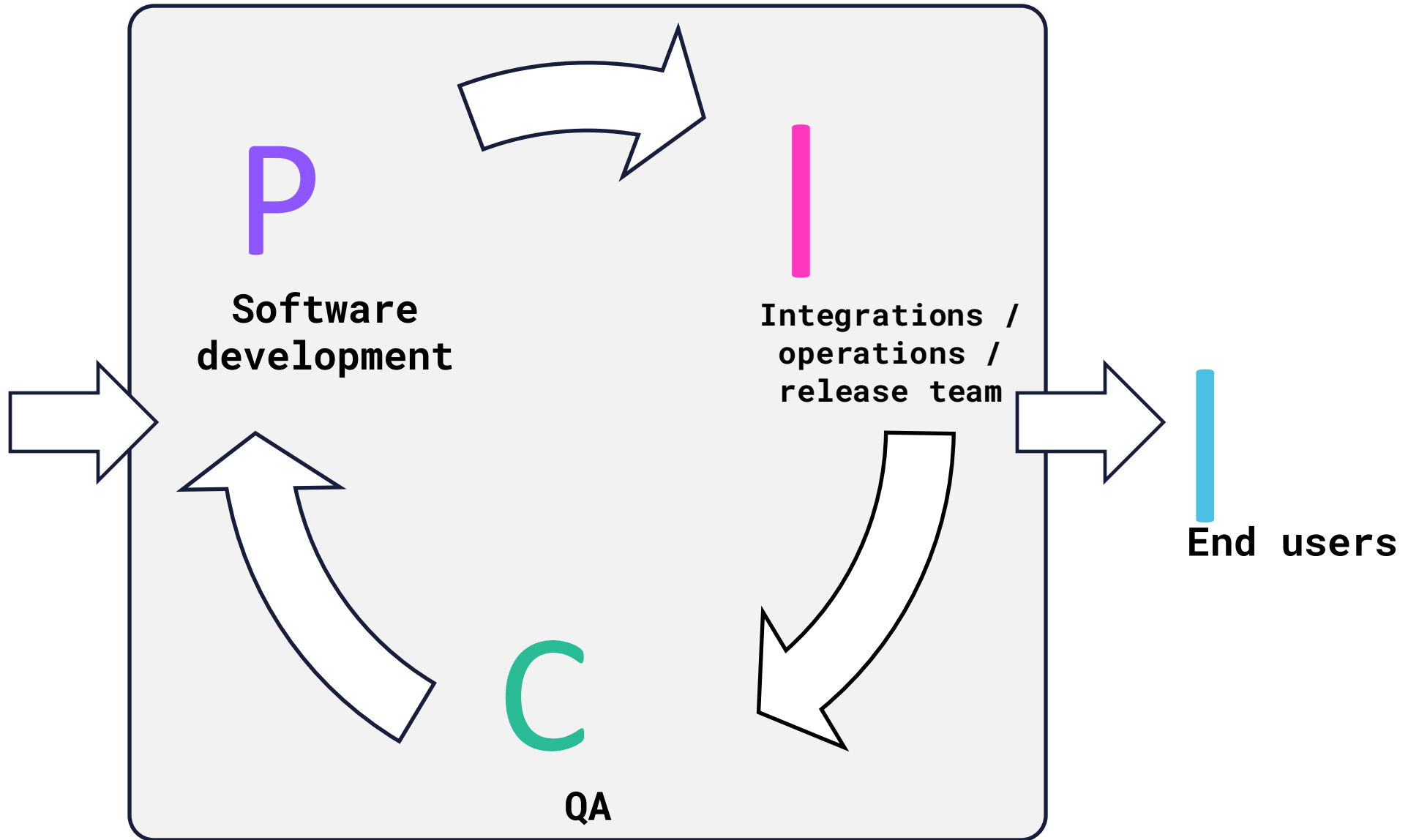
Must-have ingredients
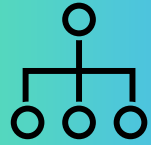
Central codebase on a source control system

Automated tests

Putting all steps to deployment together in a pipeline

Source control management

Keeps track of all previous versions of the code
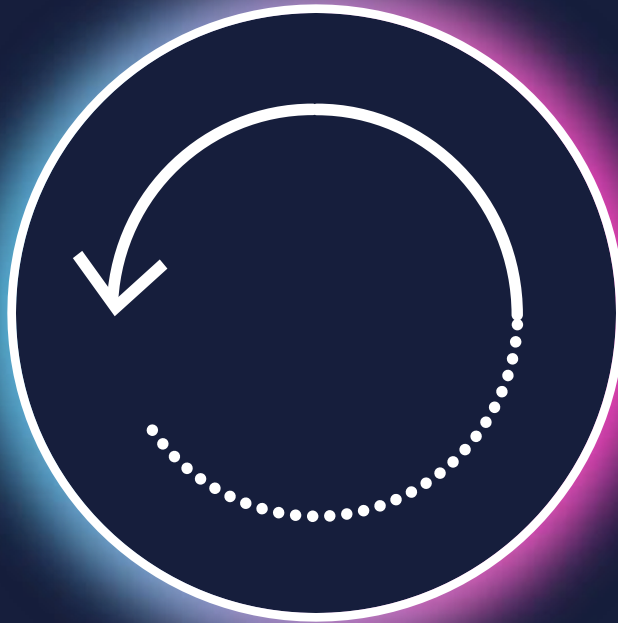
Enables going back to previous versions of the code
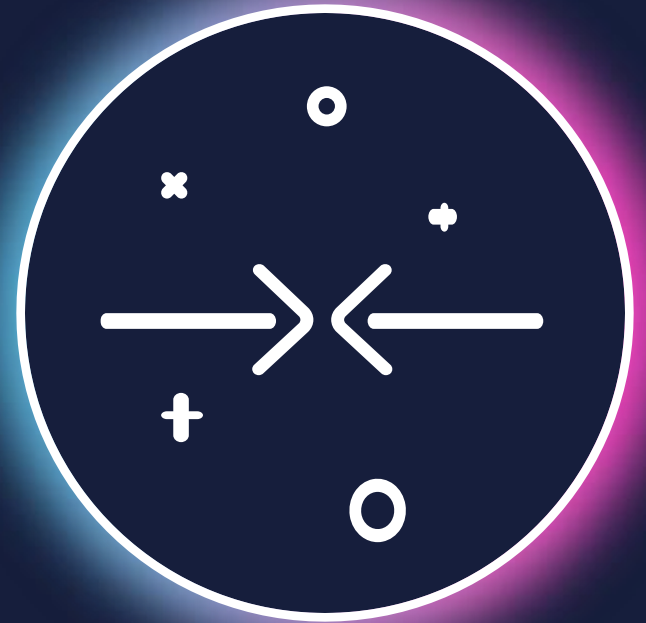
Helps merging different versions of the code

Why do we need source control management?

l Developers are working together on the same code base and will be editing the same files at the same time

p SCM avoids overriding each others work and breaking the software

a Whenever issues arise, you can easily go back to an older version that was stable

Best practices SCM

i Commit often

k Pull often

⋔ Work with branches

↳→ Merge asap

e Do code reviews
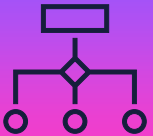
B Have detailed agreements about the process

What do we need for CI?
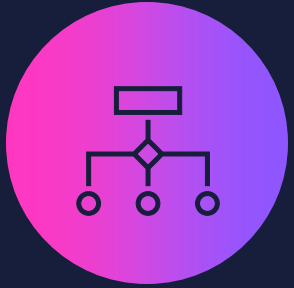
Branching strategy

A Single repo

U Different collaboration and communication at the IT department to make sure all the code will be compatible

Testing

Many kinds of testing:

- Unit testing
- Functional testing
- Non-functional testing
- BDD
- TDD

Unit testing

- Validates small code pieces (units) with code written only for testing

- Reduces risk of changes breaking functionality elsewhere

- Can be automated and run with every code update

- Units should be tested independently (using stubs, mocks, fakes, etc.)

- Tools: Junit, TestNG, and others

Exercise

- Split up in three groups

One topic per group:

- BDD
- TDD
- Functional and non-functional testing

Answer the following questions:

- What is it?
- Why do you want to use it?
- What technology and tools are there to help you do this?
- What tools are available to help you automate this?
- Show an example of the type of test
- When do you want to do this test?

The testing pyramid

Speed

Reliability

Point of failure

Confidence

Manual session-based testing

Automated GUI tests

Automated API tests

Automated integration tests

Automated component tests

Automated unit tests
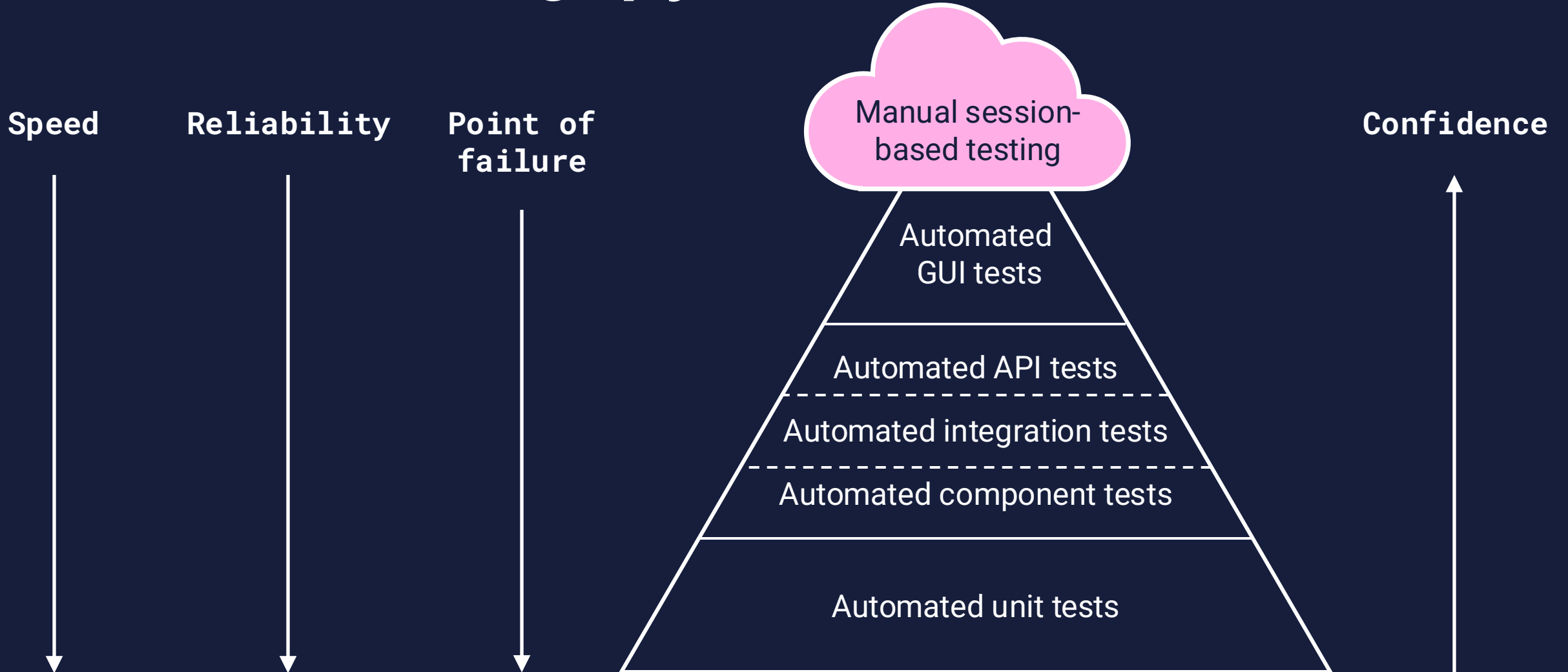
Deployment strategies

E Blue/green deployment
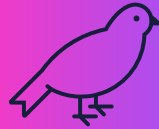
🐦 Canary releases

Y Rolling updates

U Feature toggles

Blue/green deployment

Application v1

Users

J J J

C

Application v2

J J J

Problem? Ready?

Switch back

Switch

Canary releases

Application v1

Router   Results

Users   Data

C ···· k   J J J   ⊞ ···· d

Application v2

J

Rolling update

Users

Router

Application v1

Data

Results

C ····· k ····· J J J ····· ⊞ ····· d

Application v2

J

Rolling update

Users

Router

Application v1

J J

Application v2

J J

Data

Results

C k d

Rolling update

Users

Router

Application v1

Data

Results

C

k

J

Application v2

J J J

d

Rolling update

Users

Router

Application v2

Data

Results

C .... k .... J J J J .... ⊞ ...... d

Feature toggles

- Let you adjust how a system works without needing to rewrite any code

- Allows features on or off as needed

- Useful for :
  - Testing new features on a specific user group
  - Quickly turning off a feature that causes issues
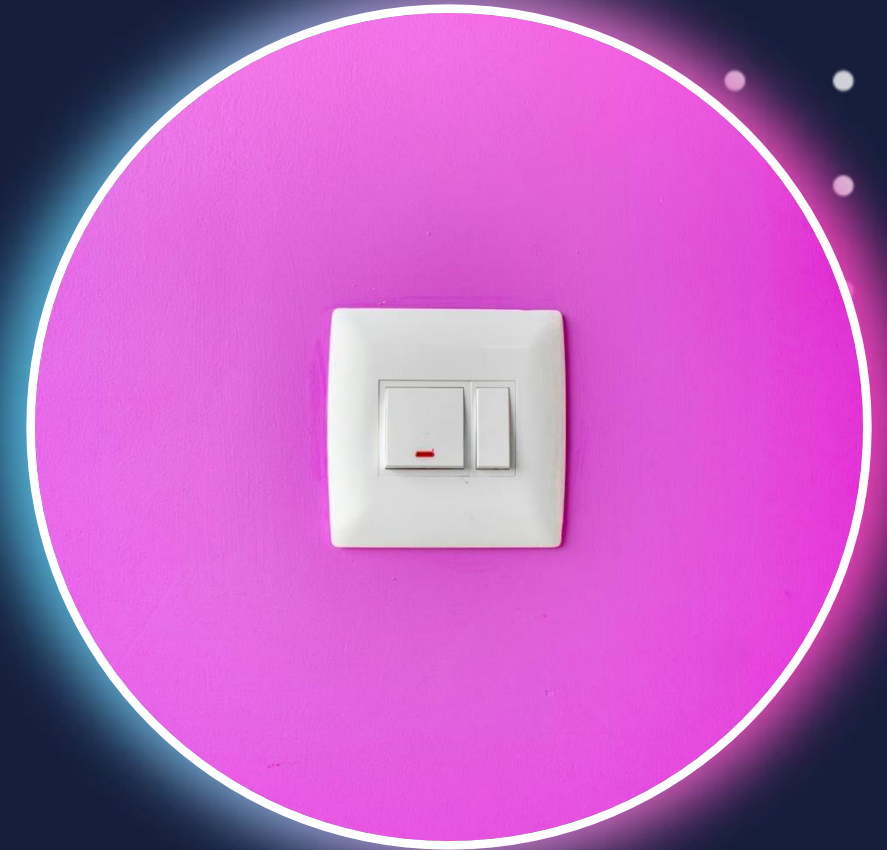
Exercise

Choosing a deployment strategy for our application

IaC and immutable infrastructure

IaC: Infrastructure as code
- Managing infrastructure with code

Immutable infrastructure
- No changes after deployment

Tools:
- Terraform
- Ansible
- CloudFormation

GitOps

- Git as the single source of truth

Principles:

- Declarative descriptions
- Automated deployments

Tools:

- Flux

- Argo CD

Site reliability engineering (SRE)

- Applying software engineering to operations

Principles:
- Reliability
- Scalability
- Efficiency

Key concepts:
- Service Level Objectives (SLOs)
- Error Budgets

Security in DevOps: DevSecOps

- Integrating security into DevOps

Principles:

- Shift-left security

- Automation of security checks

Tools:

- Snyk

- OWASP ZAP
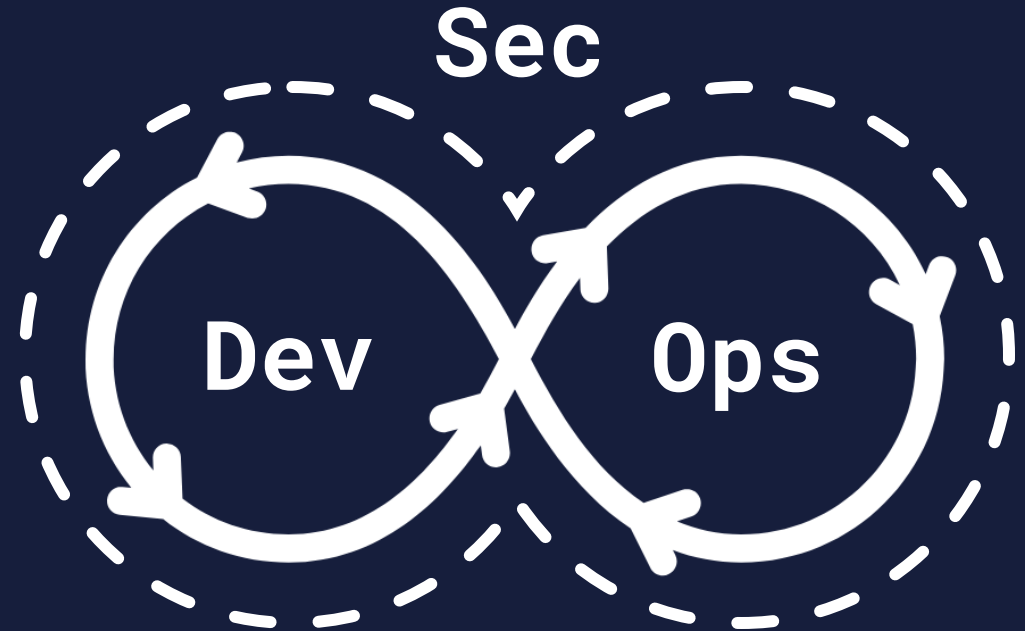
Which of the following is NOT one of the Three Ways of DevOps?

{A} **Rigorous compliance enforcement**

{B} Amplify feedback loops

{C} Continuous learning and experimentation

{D} Flow (systems thinking)

In the CALMS framework, what does the 'L' stand for?

{A} Lean

{B} Leadership

{C} Learning

{D} Lifecycle

In the CALMS framework, what does the 'L' stand for?

{A} **Lean**

{B} Leadership

{C} Learning

{D} Lifecycle

What is the purpose of Value Stream Mapping (VSM) in a DevOps context?

{A} To create a hierarchy of team responsibilities

{B} To design the user interface for applications

{C} To visualize and analyze the flow of work to identify bottlenecks

{D} To map network infrastructure for security purposes

What is the purpose of Value Stream Mapping (VSM) in a DevOps context?

{A} To create a hierarchy of team responsibilities

{B} To design the user interface for applications

{C} **To visualize and analyze the flow of work to identify bottlenecks**

{D} To map network infrastructure for security purposes

Which deployment strategy involves running two identical production environments where one is live and the other is on standby, allowing for quick rollbacks?

{A} Rolling update

{B} Blue/Green deployment

{C} Canary release

{D} Feature toggle deployment

Which deployment strategy involves running two identical production environments where one is live and the other is on standby, allowing for quick rollbacks?

{A} Rolling update

{B} **Blue/Green deployment**

{C} Canary release

{D} Feature toggle deployment

What does CI/CD stand for in the context of DevOps?

{A} Continuous Inspection / Continuous Delivery

{B} Continuous Improvement / Continuous Development

{C} Continuous Integration / Continuous Deployment

{D} Continuous Innovation / Continuous Design

What does CI/CD stand for in the context of DevOps?

{A} Continuous Inspection / Continuous Delivery

{B} Continuous Improvement / Continuous Development

{C} Continuous Integration / Continuous Deployment

{D} Continuous Innovation / Continuous Design

What is the primary goal of Continuous Integration (CI)?

{A} To merge all developer working copies to a shared mainline several times a day

{B} To automate the deployment process to production environments

{C} To perform security audits automatically

{D} To integrate customer feedback into the development process

What is the primary goal of Continuous Integration (CI)?

{A} **To merge all developer working copies to a shared mainline several times a day**

{B} To automate the deployment process to production environments

{C} To perform security audits automatically

{D} To integrate customer feedback into the development process

What is the main difference between Continuous Delivery and Continuous Deployment?

{A} Continuous Delivery requires manual testing; Continuous Deployment is fully automated

{B} Continuous Delivery automates the release process; Continuous Deployment does not

{C} Continuous Deployment automatically deploys to production; Continuous Delivery requires a manual approval

{D} Continuous Deployment is for development environments; Continuous Delivery is for production

What is the main difference between Continuous Delivery and Continuous Deployment?

{A} Continuous Delivery requires manual testing; Continuous Deployment is fully automated

{B} Continuous Delivery automates the release process; Continuous Deployment does not

{C} **Continuous Deployment automatically deploys to production; Continuous Delivery requires a manual approval**

{D} Continuous Deployment is for development environments; Continuous Delivery is for production

In the context of CI/CD, what is a 'pipeline'?

{A} A physical conduit for transporting data

{B} A network protocol for secure communication

{C} A method for compressing data before transmission

{D} A sequence of automated processes that deliver software from version control to users

In the context of CI/CD, what is a 'pipeline'?

{A} A physical conduit for transporting data

{B} A network protocol for secure communication

{C} A method for compressing data before transmission

{D} A sequence of automated processes that deliver software from version control to users

Which practice helps ensure that new code changes do not break the existing functionality in a CI process?

{A} Automated testing (unit tests, integration tests)

{B} Manual code review without testing

{C} Ignoring test failures

{D} Delaying integration until all features are complete

Which practice helps ensure that new code changes do not break the existing functionality in a CI process?

{A} Automated testing (unit tests, integration tests)

{B} Manual code review without testing

{C} Ignoring test failures

{D} Delaying integration until all features are complete

Next up:

Implementing CI

Questions or suggestions?

maaikejvp@gmail.com

See you tomorrow!