Introduction to DevOps & DevOps Philosophy

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)

Introduction round

 Current role, background and ambitions

 Hobbies / interests / life outside of work

 What do you hope to learn?

Introduction to DevOps & DevOps Philosophy

Learning objectives

O

Define the work of the **operations team** in the traditional sense, understand its tasks and the role in the software development lifecycle.

P

Discuss the **philosophy** beyond DevOps, emphasizing collaboration, automation, continuous improvement, and high efficiency in development and operations teams.

Schedule

Intro + theory

Introduction round + what is the operations team

b

Exercise

Manual deployments

D

Debrief + theory

Debrief exercise + challenges

b

Exercise

Thought experiment

q

DevOps theory

Terms and if enough time small exercises + mini quiz

Schedule

Intro + theory

Introduction round + what is the operations team

b

Exercise

Manual deployments

D

Debrief + theory

Debrief exercise + challenges

b

Exercise

Thought experiments

q

Debrief + wrap up

Debrief exercise + mini quiz

Operations team

- 
  Responsible for maintaining IT infrastructure
- J
  Handles server setup, deployment and maintenance
- 
  Manages network configurations and security
- I
  Works separately from development teams
- B
  Reactive approach to issues

Deploying software

- Receive code from developers
- Manually configure servers
- Install dependencies and packages
- Deploy code to production environment

Exercise

Manually deploy a basic application with a frontend and backend to experience the work of the operations team first-hand.

Debrief - Steps taken

- Server setup and configuration
- Installed dependencies manually
- Configured environment variables
- Deployed frontend and backend code
- Started application services

Challenges

- Time-consuming and error-prone
- Lack of standardization
- Minimal collaboration with developers
- Difficulty in reproducing environments

Exercise

Thought experiment: scaling up

Debrief

d

Scaling up manually makes it very complex and time-consuming

Imaging having to manage all this

r

Can you feel the headache yet?

What is DevOps?

- DevOps = Development + Operations
- A cultural movement, not just tools
- Unifies development and operation teams
- Focuses on collaboration and communication
- Enhances efficiency and quality

DevOps principles

- Collaboration
- Shared goals and responsibilities
- Automation
- Continuous Integration/Continuous Delivery (CI/CD
- Infrastructure as Code (IaC)
- Monitoring and logging
- Continuous feedback

Benefits of the software lifecycle

- 
  Faster time to market
- m
  Improved quality
- X
  Enhanced collaboration and breaking down silos
- B
  Increased reliability
- h
  Automation reduces manual tasks
- R
  Continuous improvement

Philosophy behind DevOps

- n — People over processes and tools
- B — Shared responsibility and ownership
- b — Lean and agile principles
- P — Embracing failure as learning
- R — Customer-centric focus
- D — Culture of trust and transparency

Business value of DevOps

- Faster delivery meets market demands
- Optimized resources reduce expenses
- Improved quality, less errors due to automation
- Customer satisfaction
- Competitive advantage
- Risk mitigation and early detection of issues
- Scalability

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

Benefits of automation

- b — Consistency and standardized processes
- M — Faster delivery cycles
- k — Reduced errors

Continuous improvement and feedback loops

- Measure performance
- Collect user feedback
- Iterative improvements to implement changes on feedback
- Tools like Prometheus, Grafana, ELK stack

Benefits of continuous improvement and feedback loops

Higher quality

Proactive issue resolution

Informed decision-making

Collaboration between development and operations

X

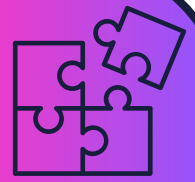
Break down silos to foster cross-team interaction

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

Improved communication

Faster issue resolution

Increased innovation

m

R

A

The three ways of DevOps

First way - flow (systems thinking)

- Optimize the entire system

Second way - feedback loops

- Amplify feedback for continuous improvement

Third way - continuous learning and experimentation

- Foster a culture of innovation and learning

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

- Deployment frequency
- 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

Deployment strategies

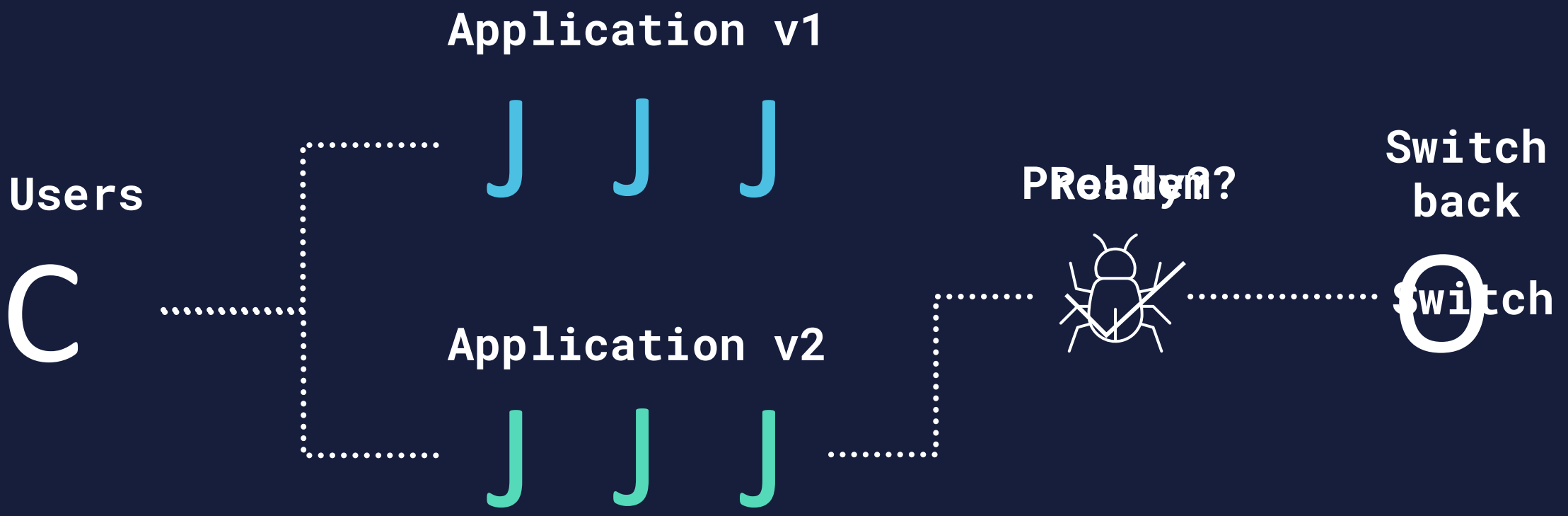
E — Blue/green deployment

 Canary releases

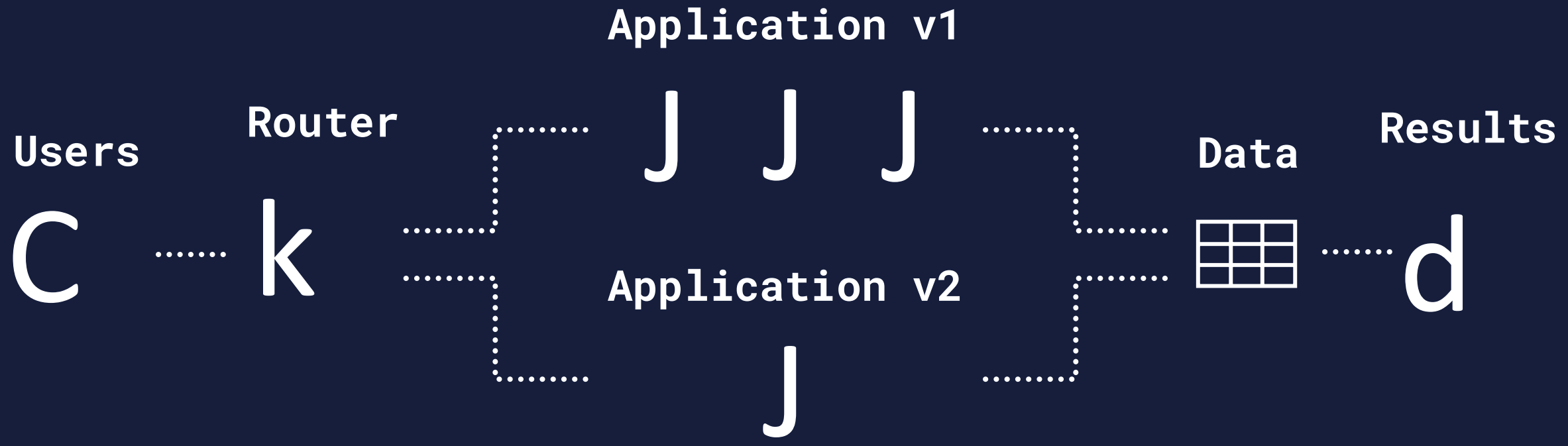
Y — Rolling updates

U — Feature toggles

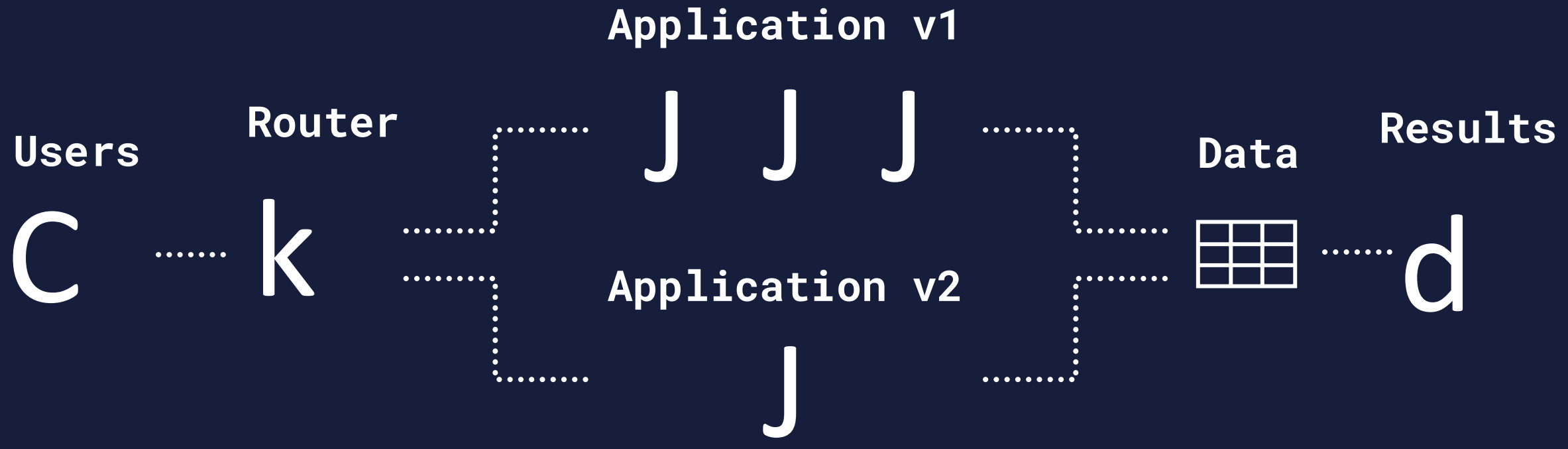
Blue/green deployment



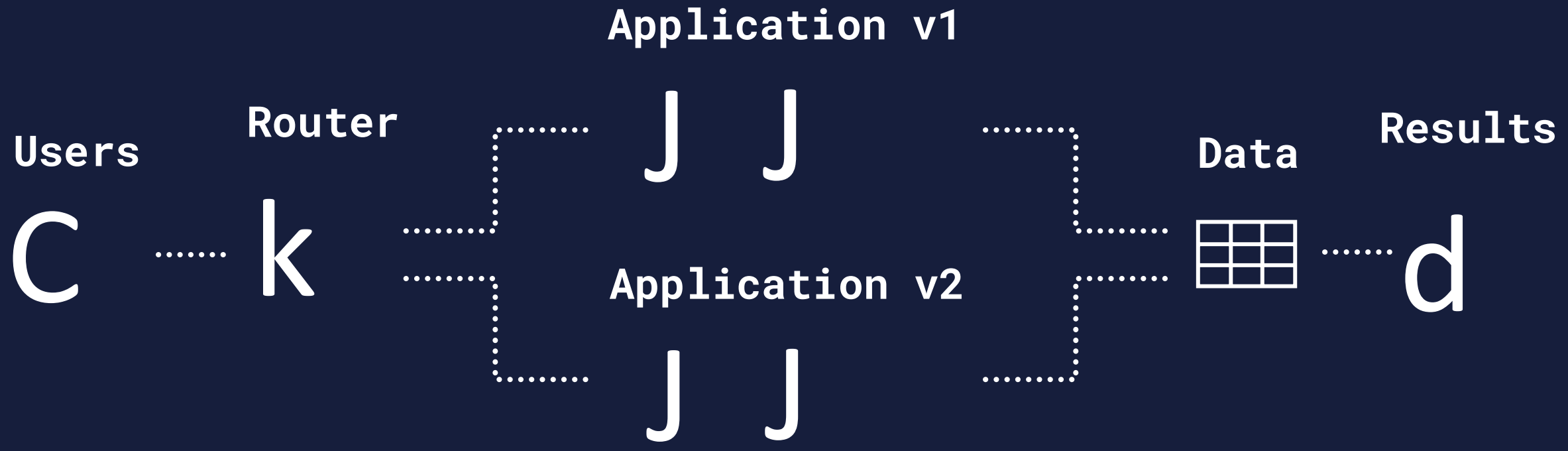
Canary releases



Rolling update



Rolling update



Rolling update

Rolling update

Users

Router

Application v2

Data

Results

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

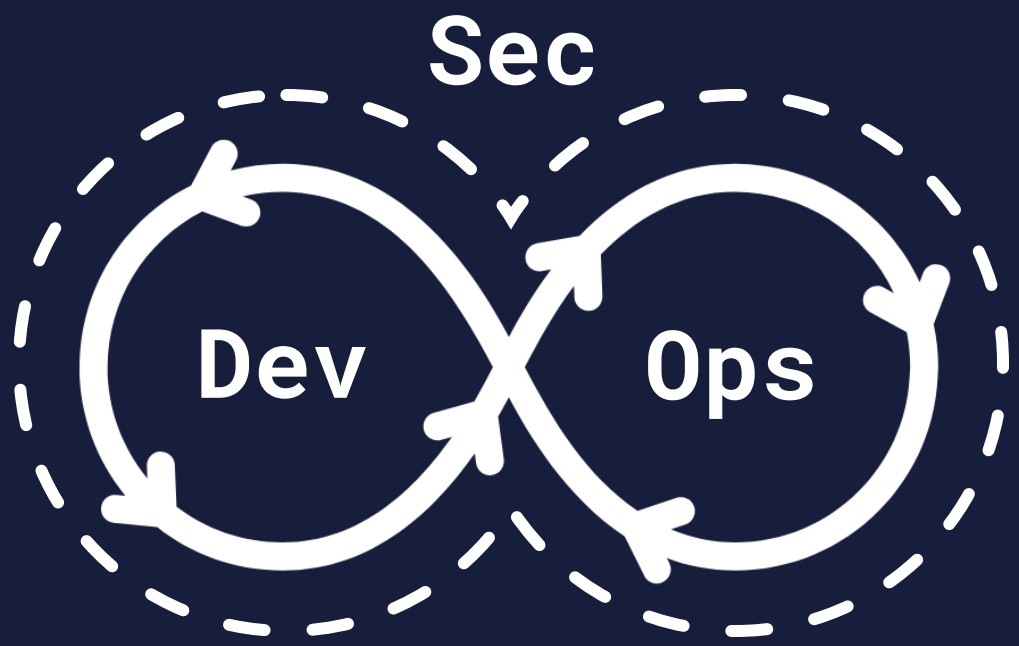
- Integrating security into DevOps

Principles:

- Shift-left security
- Automation of security checks

Tools:

- Snyk
- OWASP ZAP



What is the primary goal of DevOps in the software development lifecycle?

{A} To eliminate the need for testing by automating deployments

{B} To separate development and operations teams to increase specialization

{C} To outsource operations tasks to third-party vendors

{D} To unify development and operations teams for enhanced collaboration and efficiency

What is the primary goal of DevOps in the software development lifecycle?

{A} To eliminate the need for testing by automating deployments

{B} To separate development and operations teams to increase specialization

{C} To outsource operations tasks to third-party vendors

{D} **To unify development and operations teams for enhanced collaboration and efficiency**

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)

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

Next up:

Introduction to CI/CD

Questions or suggestions?

maaikejvp@gmail.com

See you tomorrow!

Introduction to CI/CD

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 **Azure DevOps** environment and its tools that support CI/CD processes.

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

So, with CI we can achieve...

- m — Higher quality
- — Shorter time to market, more releases
- k — Higher flexibility
- g — Less effort / time / money needed for integration

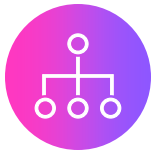
Group exercise

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

h

Tests need to be automated

i

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

When the build fails, fix it immediately

N

Everyone commits the code at least daily, preferably more often

M

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

- Lower chance of errors
- Less effort / time / money for releasing
- Easy to reproduce deployment due to automatic management of the environments (and therefore the exact same environments)
- Deployments can be done more often and are less risky
- 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

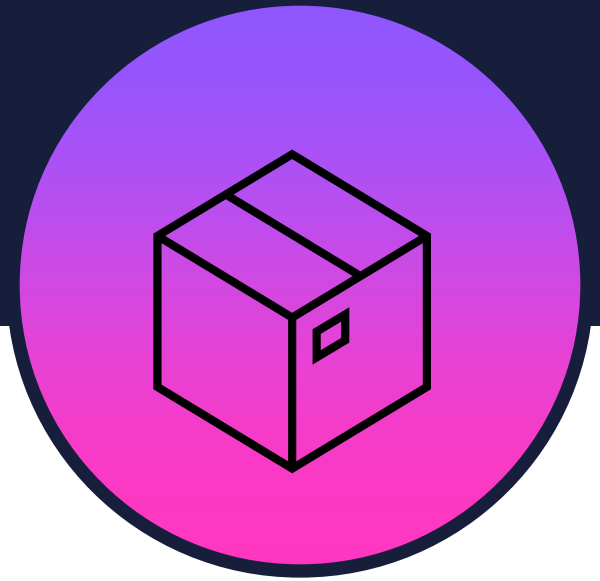
All benefits from CI and Continuous Delivery

Short TTM

S

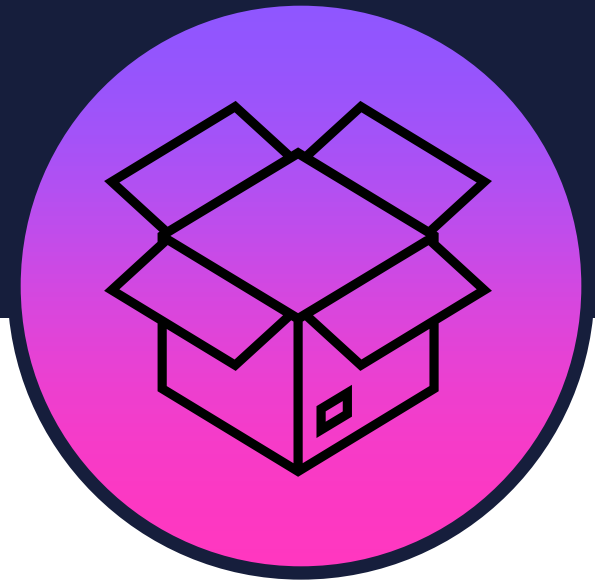
Early feedback from users

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

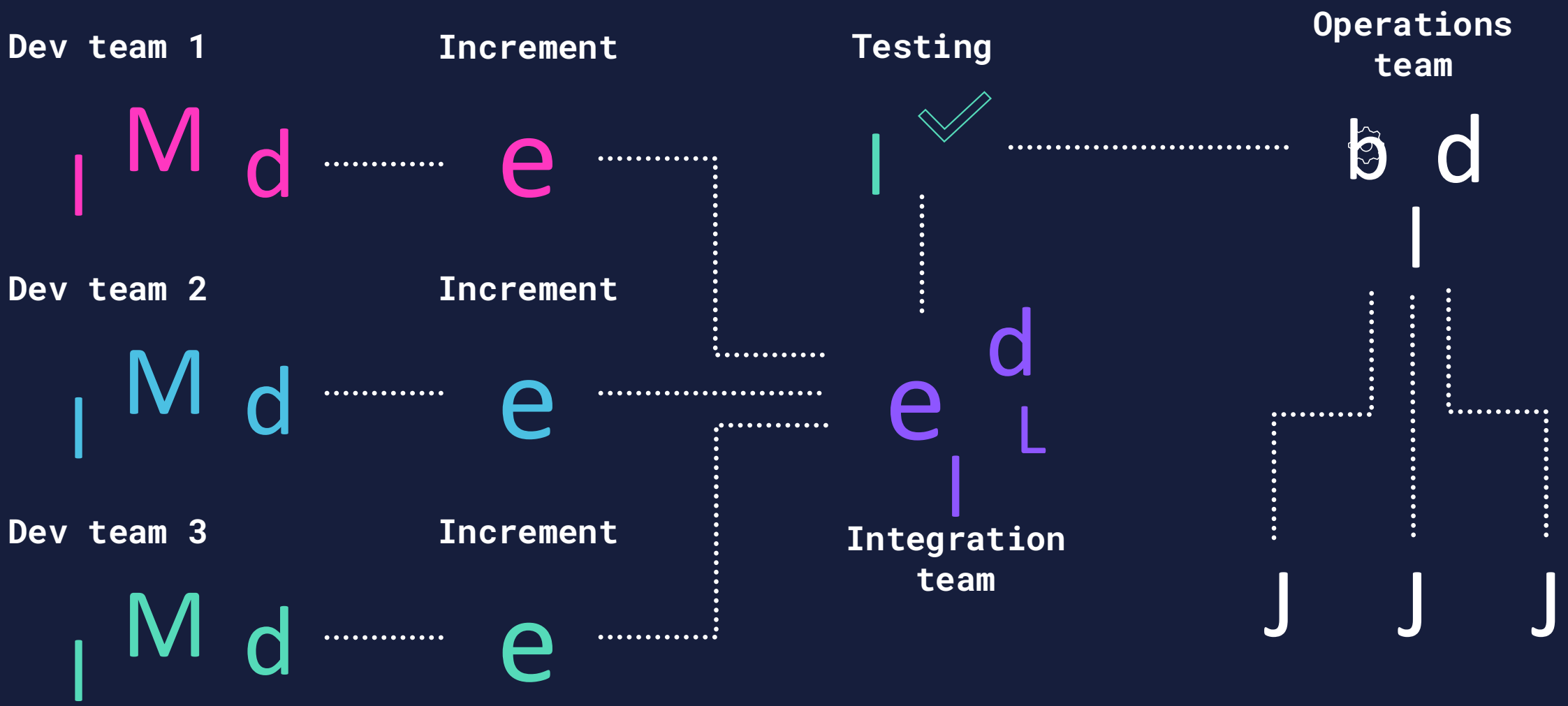
Exercise

What is the situation without CI/CD?

Tools to use:

- draw.io

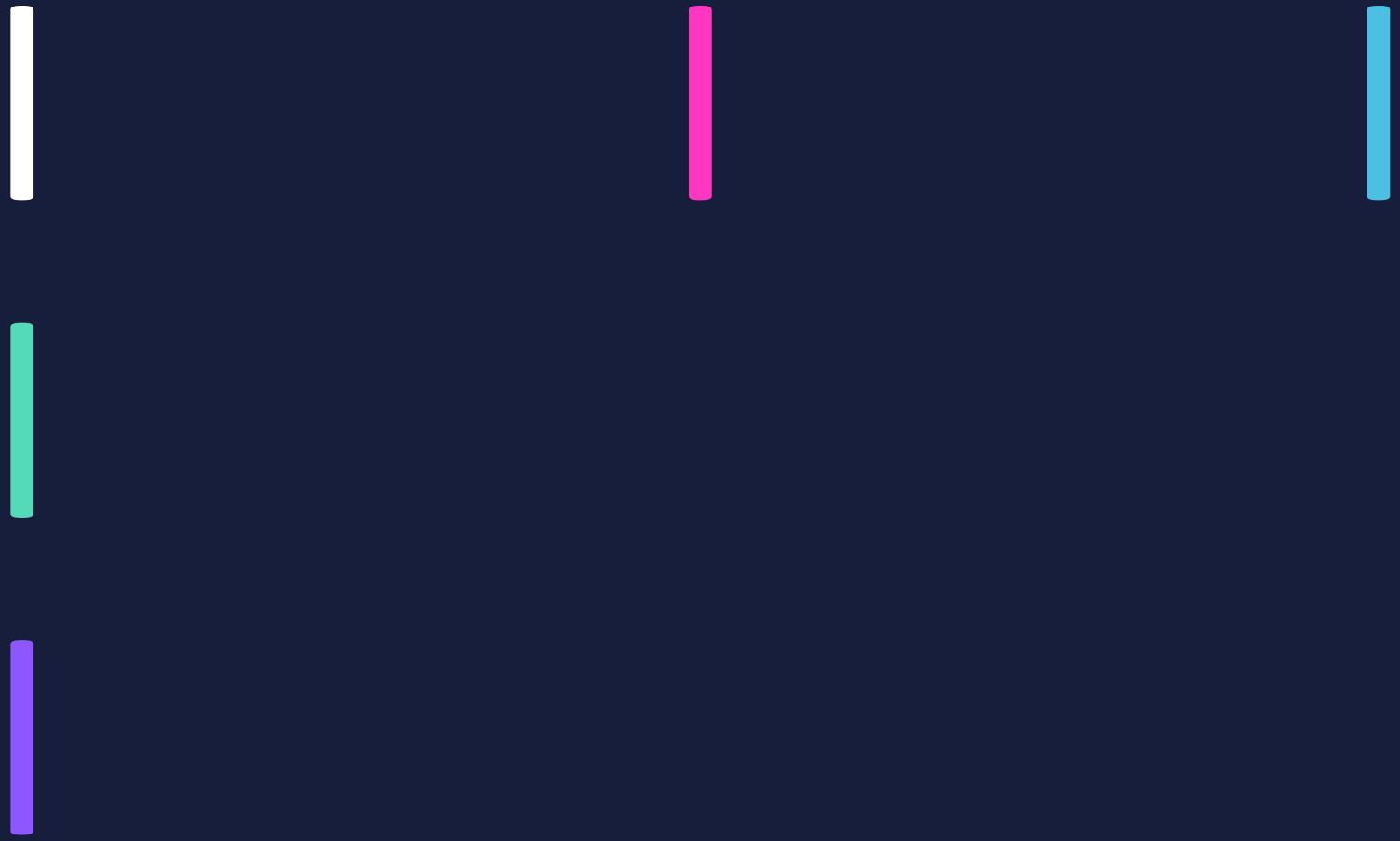
Without continuous delivery

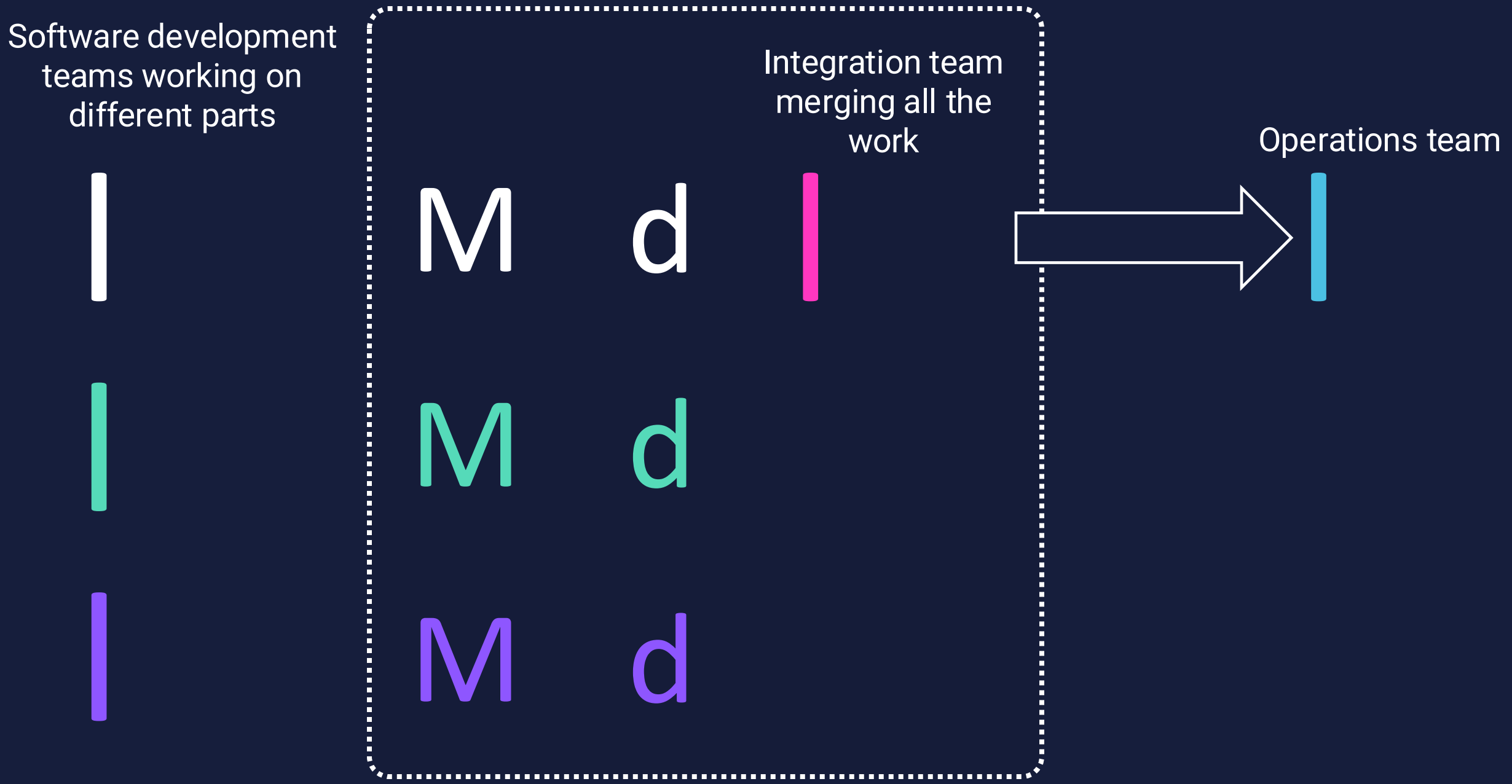


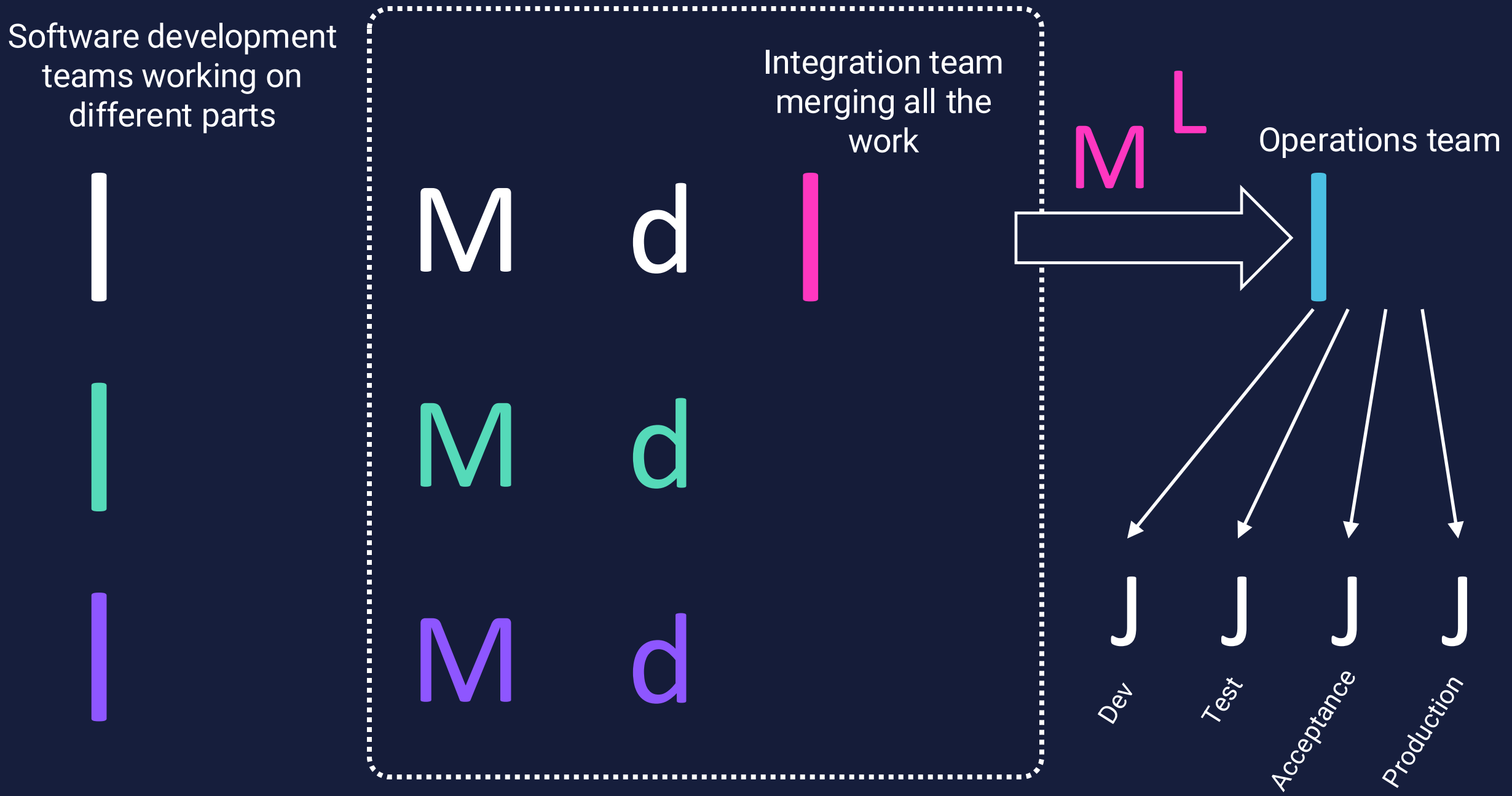
Software development teams working on different parts

Integration team merging all the work

Operations team

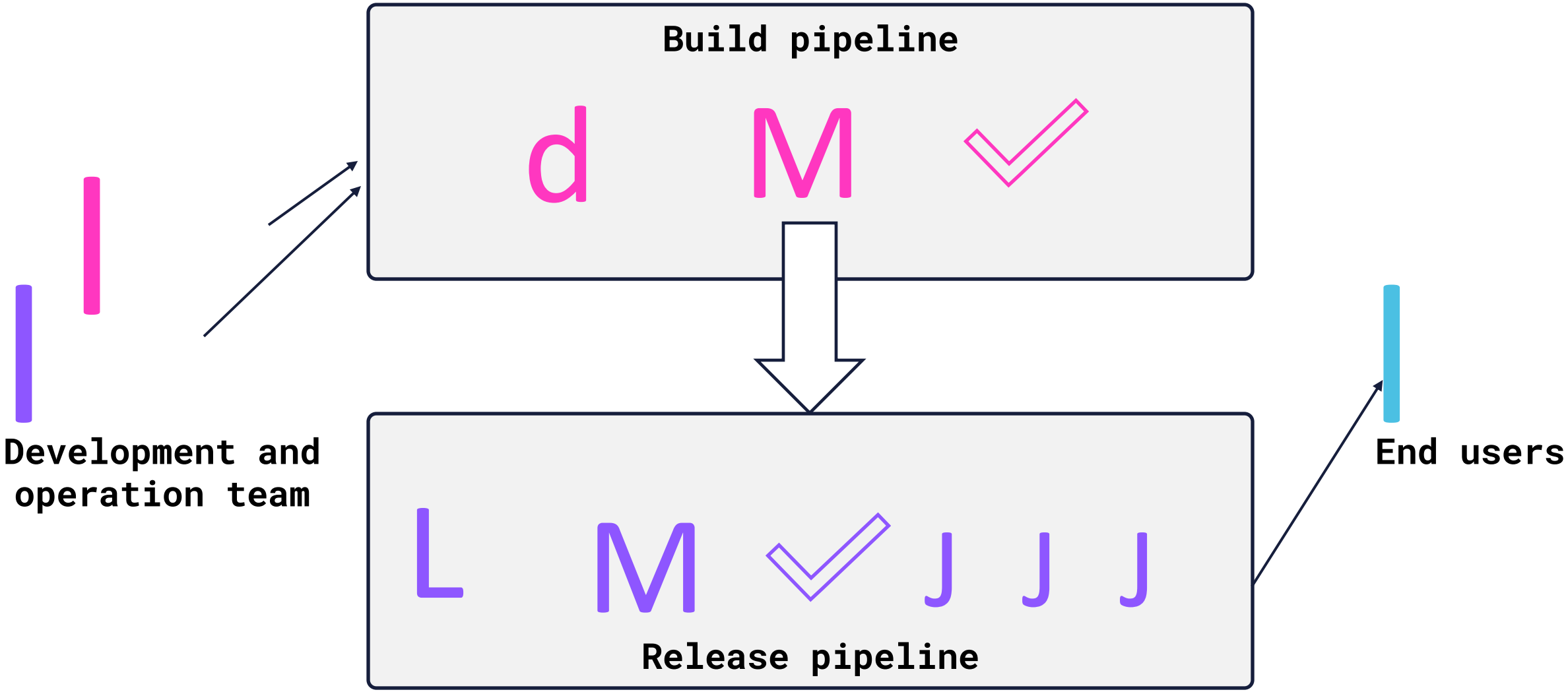



Software development teams working on different parts
M
d
M
d
M
d
Integration team merging all the work
Operations team

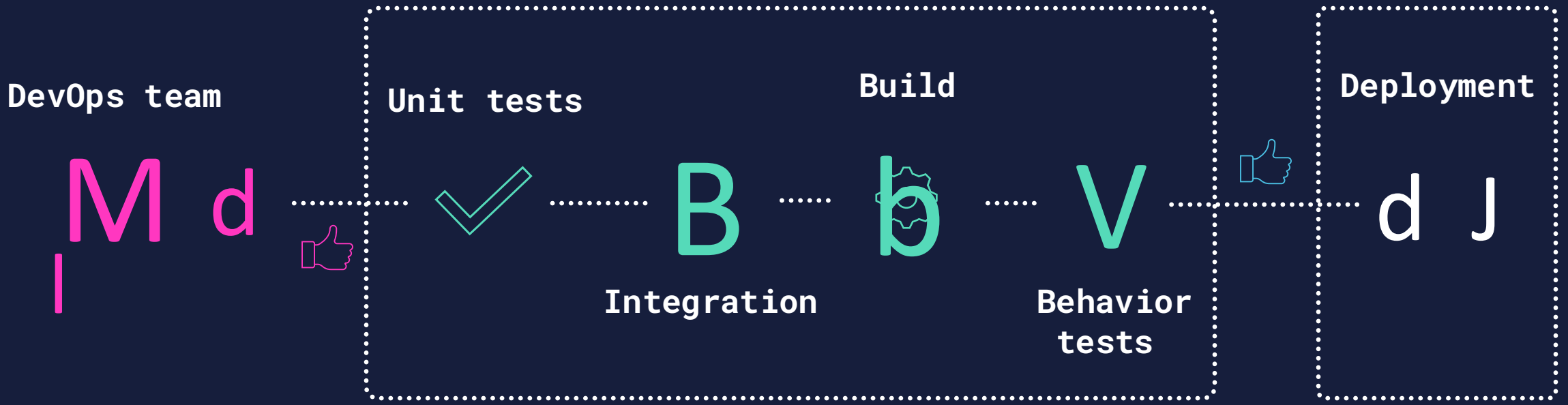

Software development teams working on different parts
Integration team merging all the work
Operations team
Dev
Test
Acceptance
Production

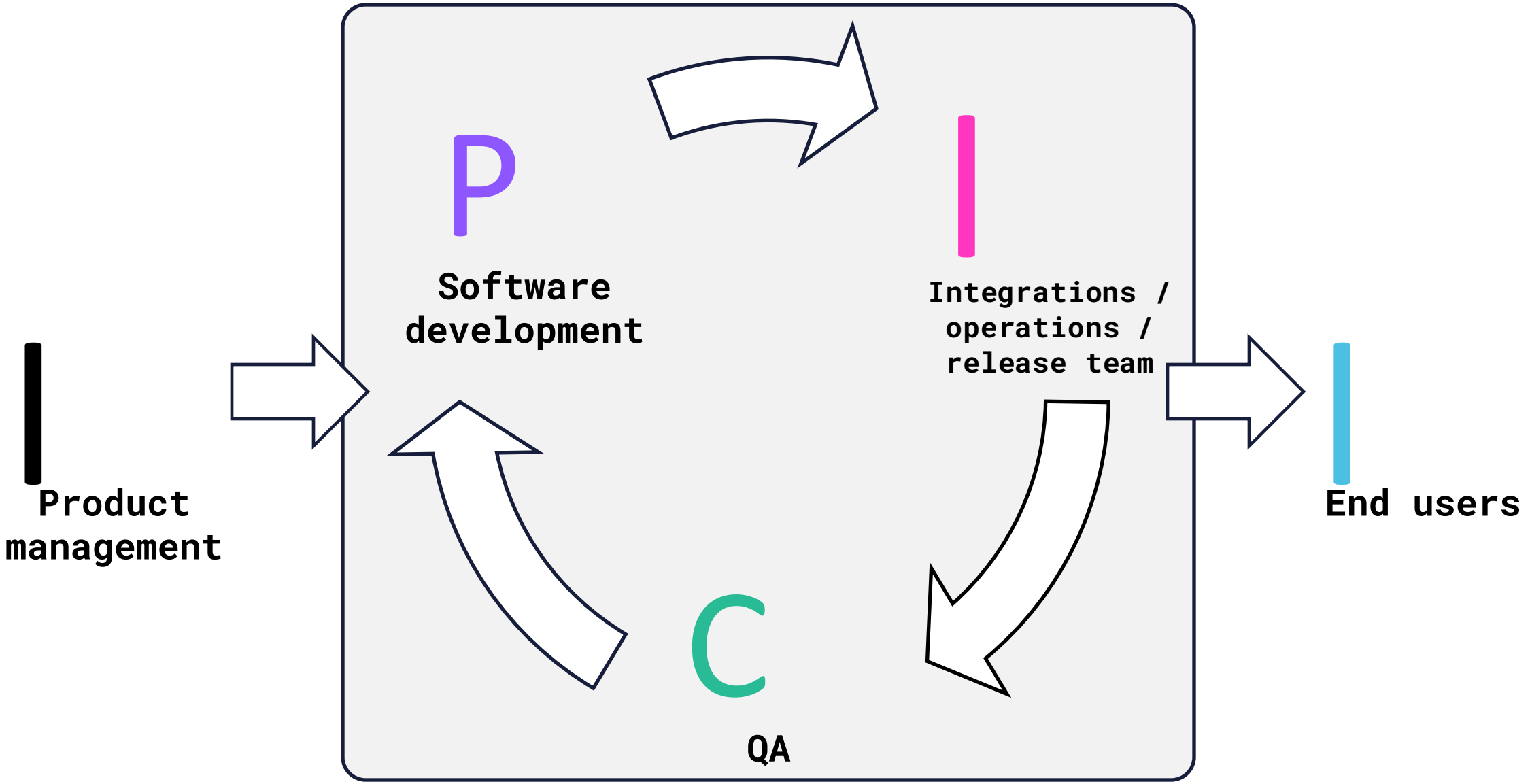
Exercise

Draw a CI/CD pipeline


Build pipeline
Development and operation team
Release pipeline
End users

With continuous delivery




P
Software development
I
Integrations / operations / release team
Product management
End users
C
QA

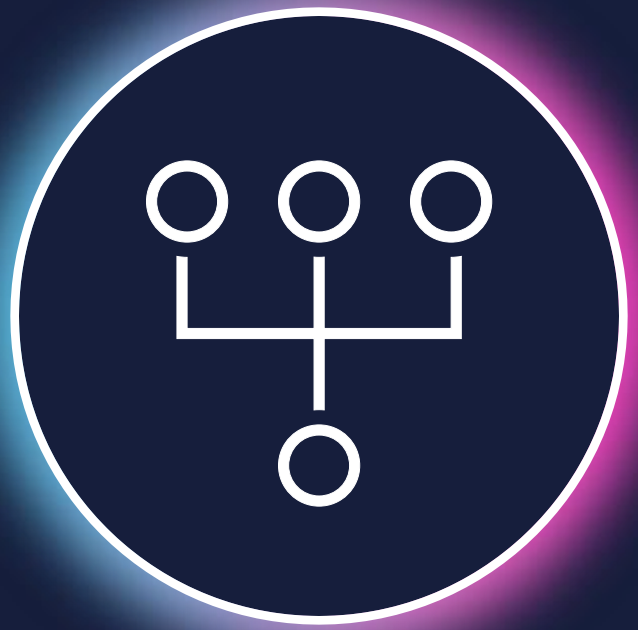
Must-have ingredients

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

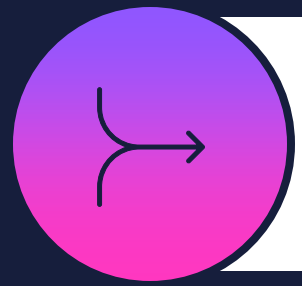
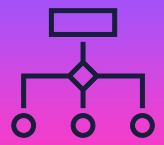
Enables going back to previous versions of the code

Helps merging different versions of the code

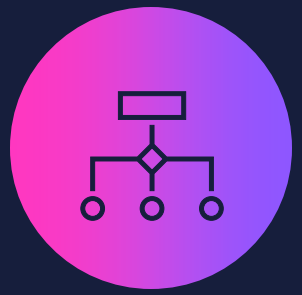
Why do we need source control management?

- **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
-  — Merge asap
- k — Pull often
- e — Do code reviews
-  — Work with branches
- 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

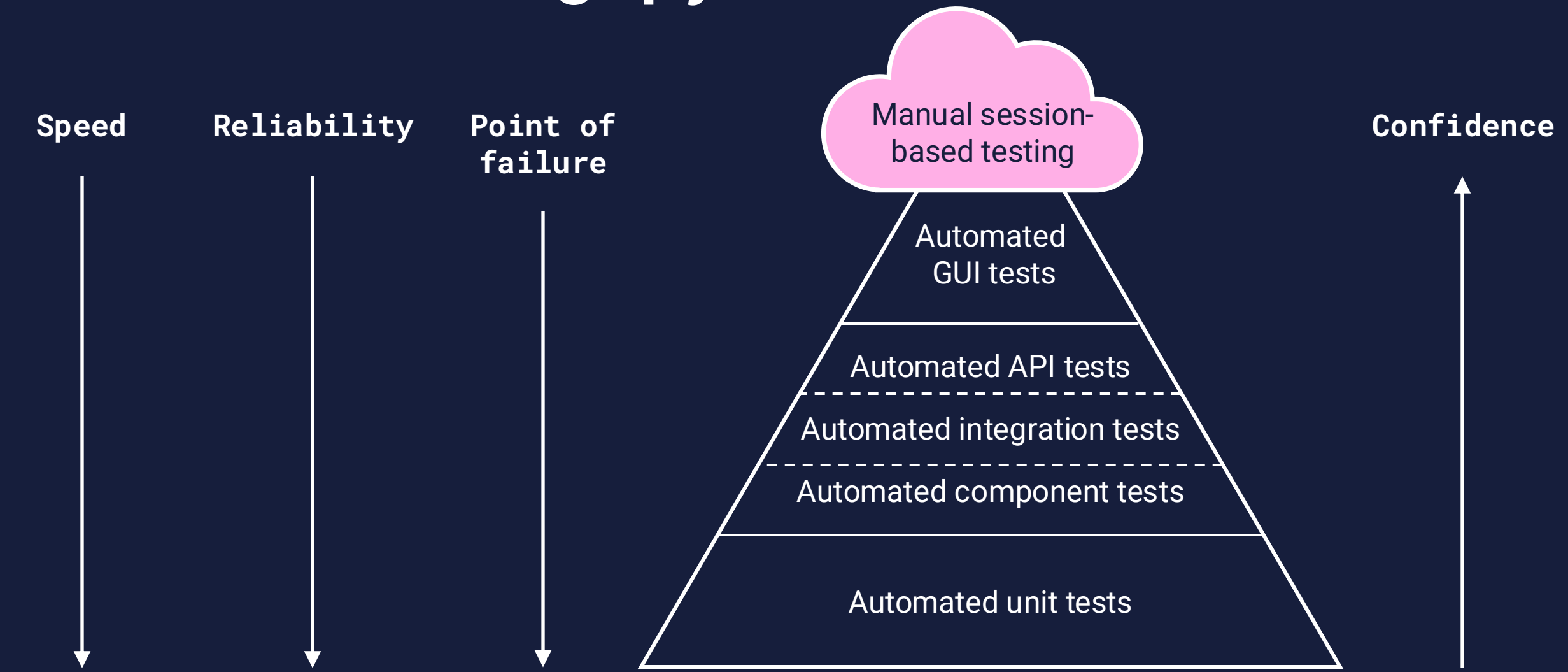
One topic per group:

- 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



Exercise

Deploy a node.js Express app with Azure DevOps

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

Questions or suggestions?

maaikejvp@gmail.com

www.biltup.com