


Techniczny SM

Kate Prokopiuk



 *Wystąpienie będzie nagrywane*

The slide features four decorative shapes in the corners: a green quarter-circle in the top-left, an orange quarter-circle in the top-right, a green quarter-circle in the bottom-left, and an orange quarter-circle in the bottom-right. The main text is centered in a large, bold, black font.

**Jeśli nas słuchają to
zazwyczaj nie rozumieją ...**

~ prof. Jerzy Bralczyk



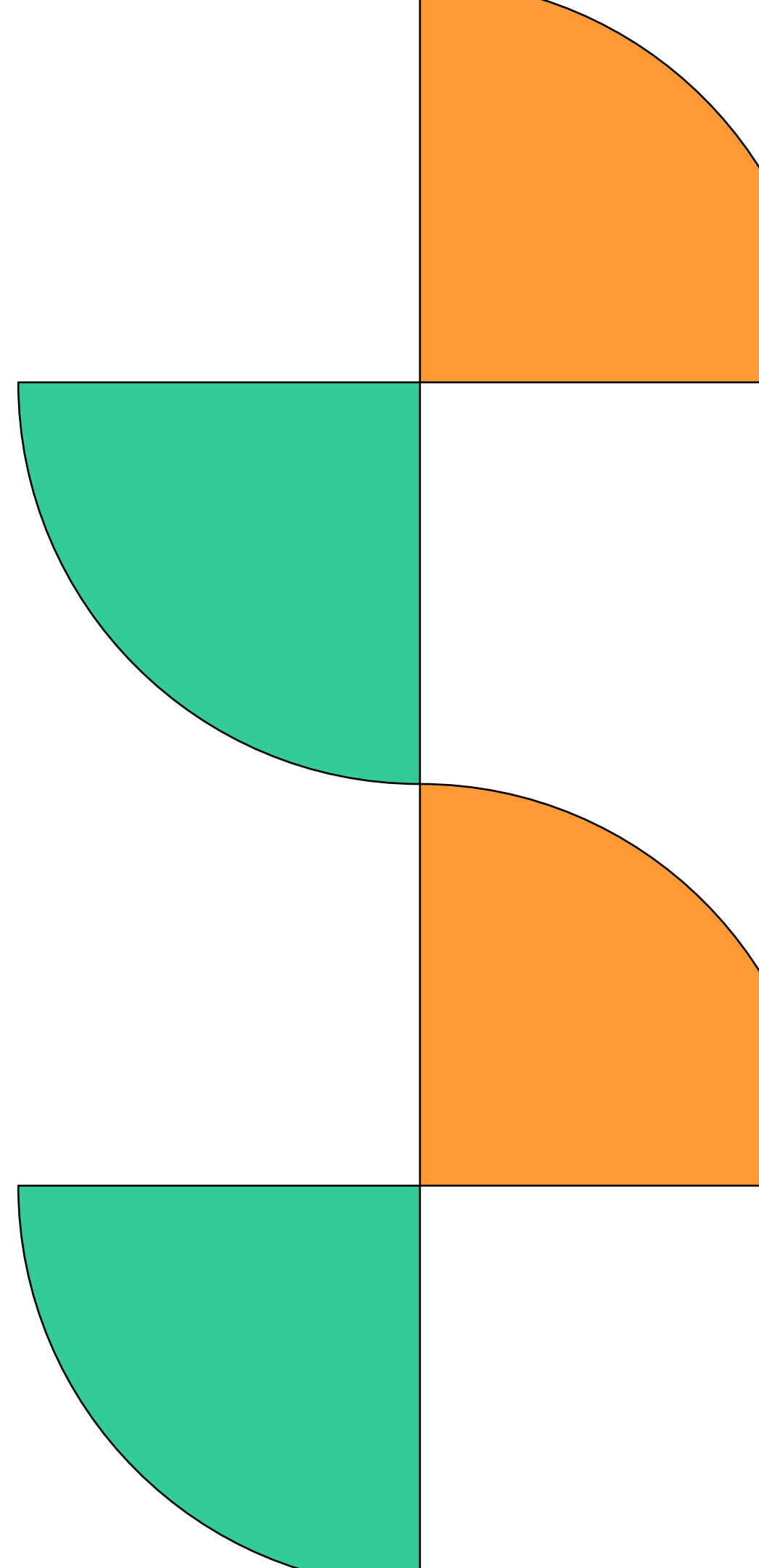
O mnie

10 lat doświadczenia

kręta droga

od Project Managera do Engineering Managera

Dlaczego?



Business case 1

Dział Security zgłosił podatność systemu do jednego z zespołów aplikacyjnych. Okazuje się, że zespół używa libki, która jest 3 major wstecz.

“PO: To chyba jakaś formalność?”

A: Trzeba podpisać libkę i przetestować zmiany na developie. To co się rozsypie, będziemy musieli przerobić.

B: No, ale CR musi przejść 2 approve, a Tomka nie ma.

C: To ja Wam zrobię CR i będziemy gotowi puścić MR do infry.

A: Jak build przejdzie testy to zmergują nam to do maina.”

Business case 1

craftmanship

CI/CD

“PO: To chyba jakaś formalność?”

A: Trzeba podpisać libkę i przetestować zmiany na developie. To co się rozsypie, będziemy musieli przerobić.

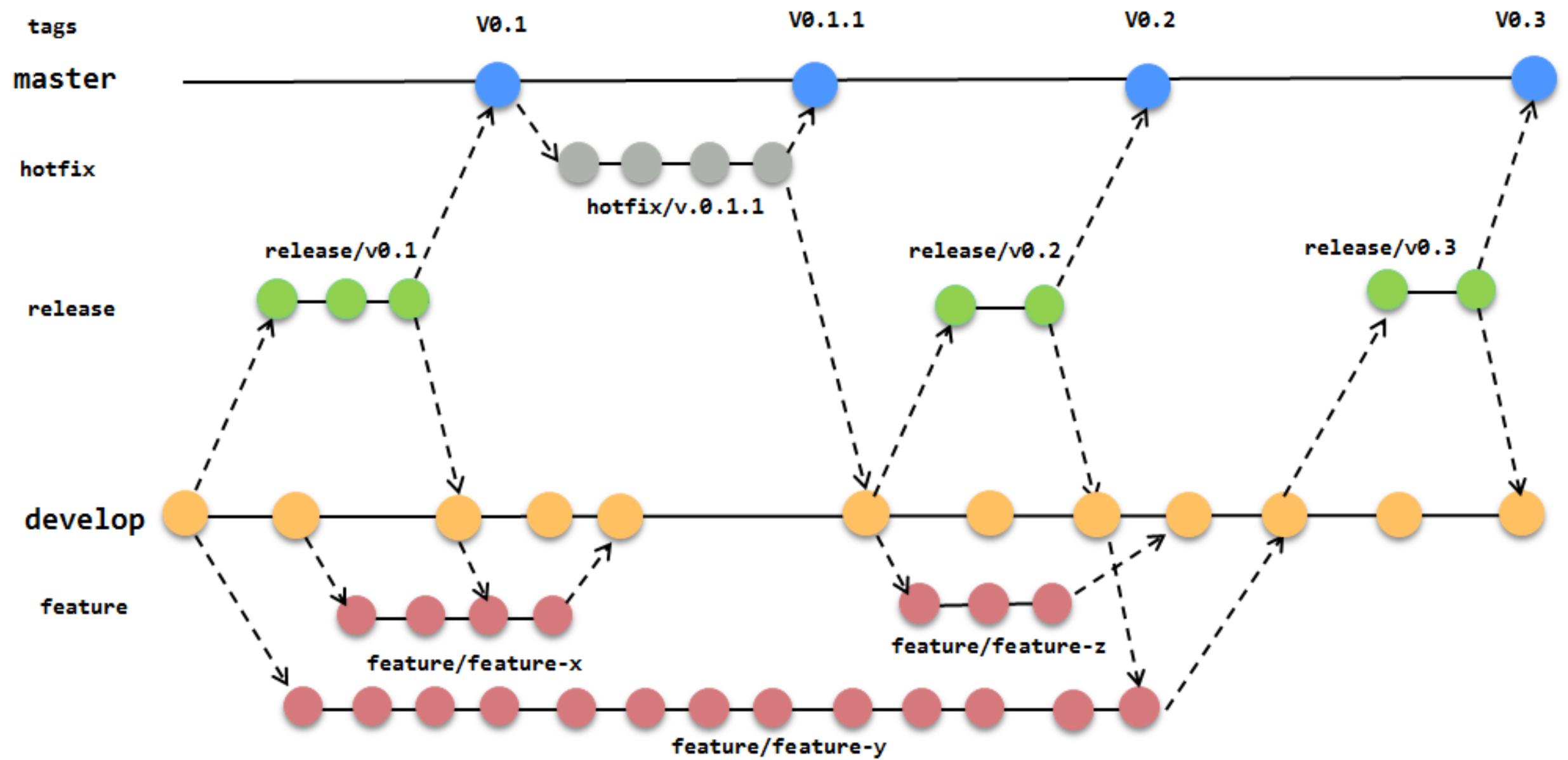
B: No, ale CR musi przejść 2 approve, a Tomka nie ma.

C: To ja Wam zrobię CR i będziemy gotowi puścić MR do infry.

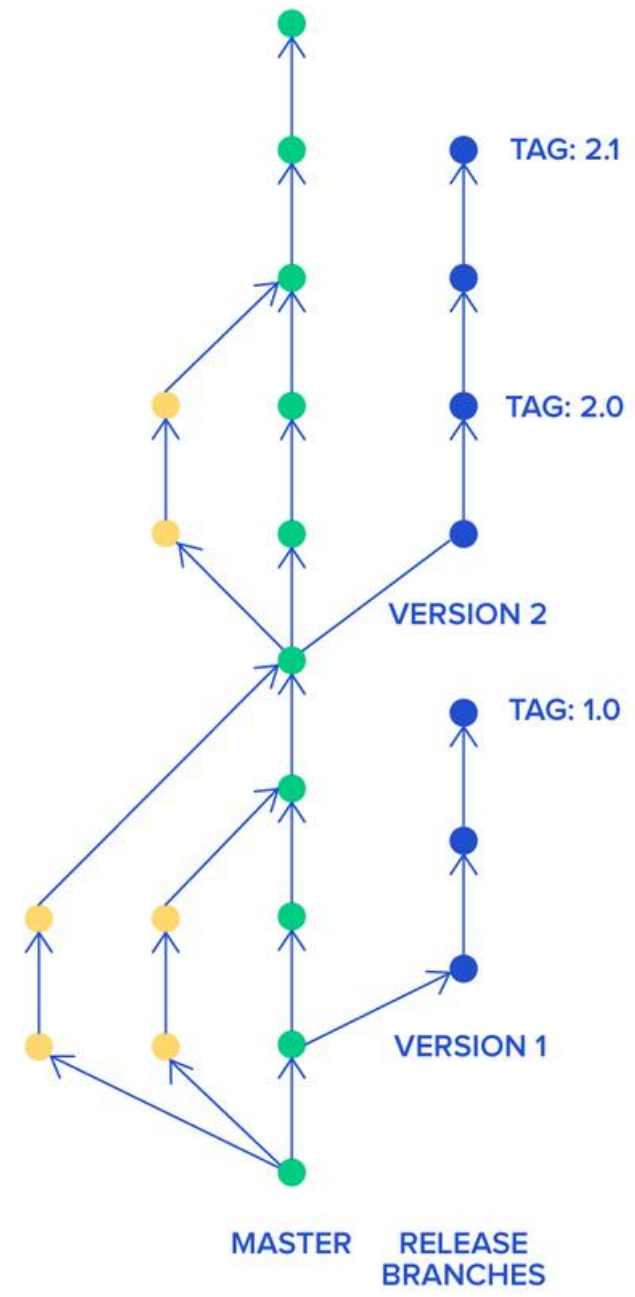
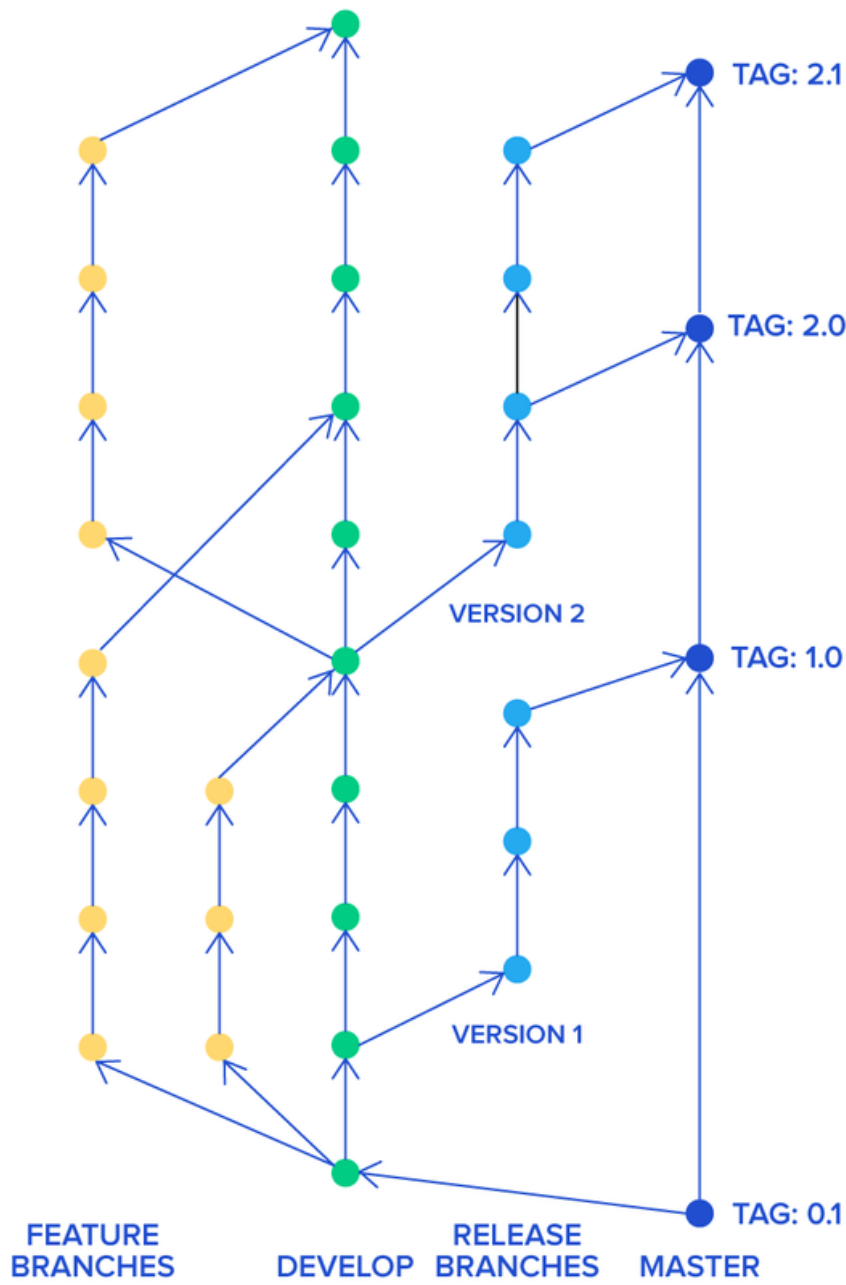
A: Jak build przejdzie testy to zmergują nam to do maina.”

kontrola wersji Git

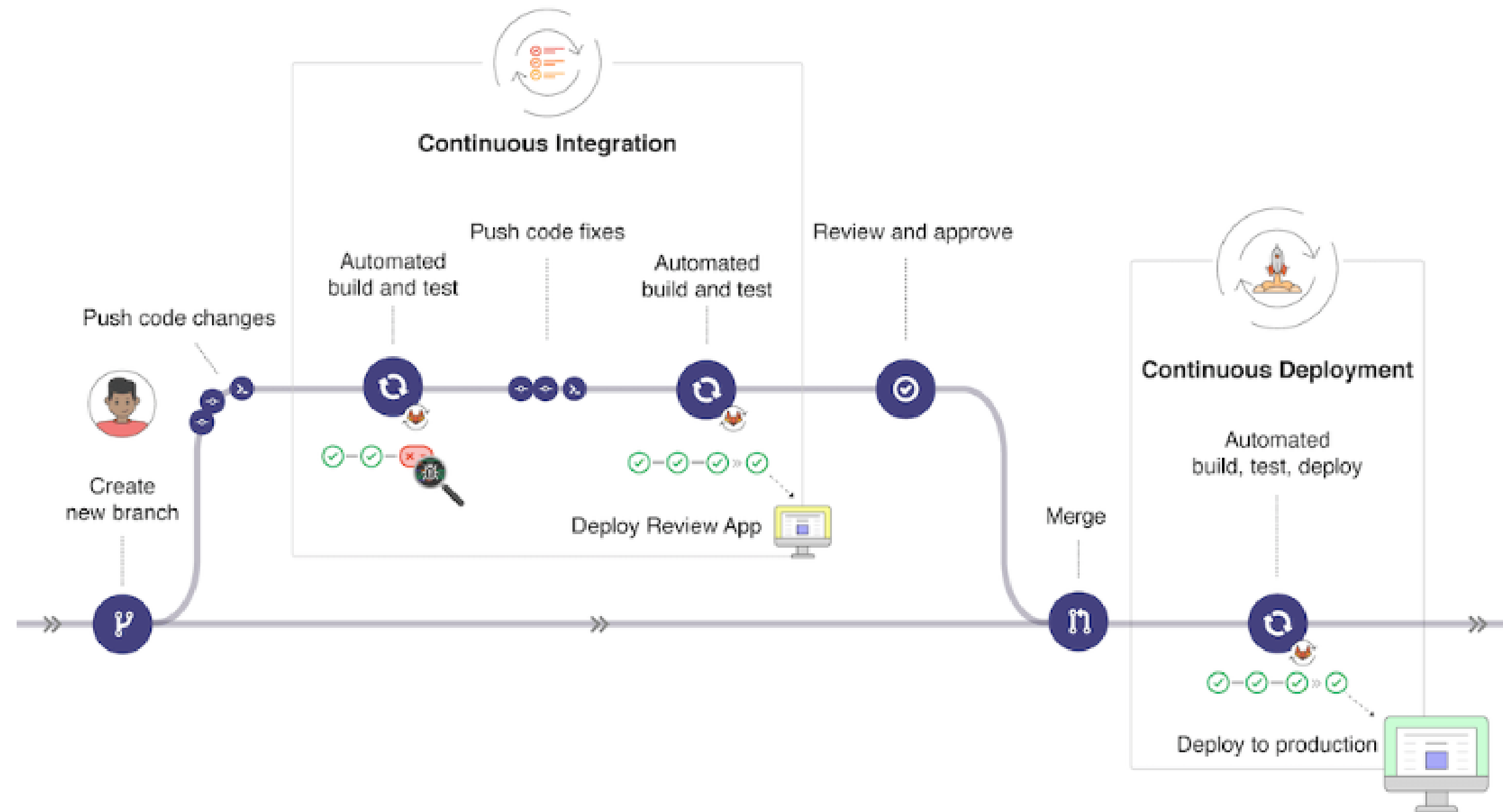
kontrola wersji



git kontra trunk

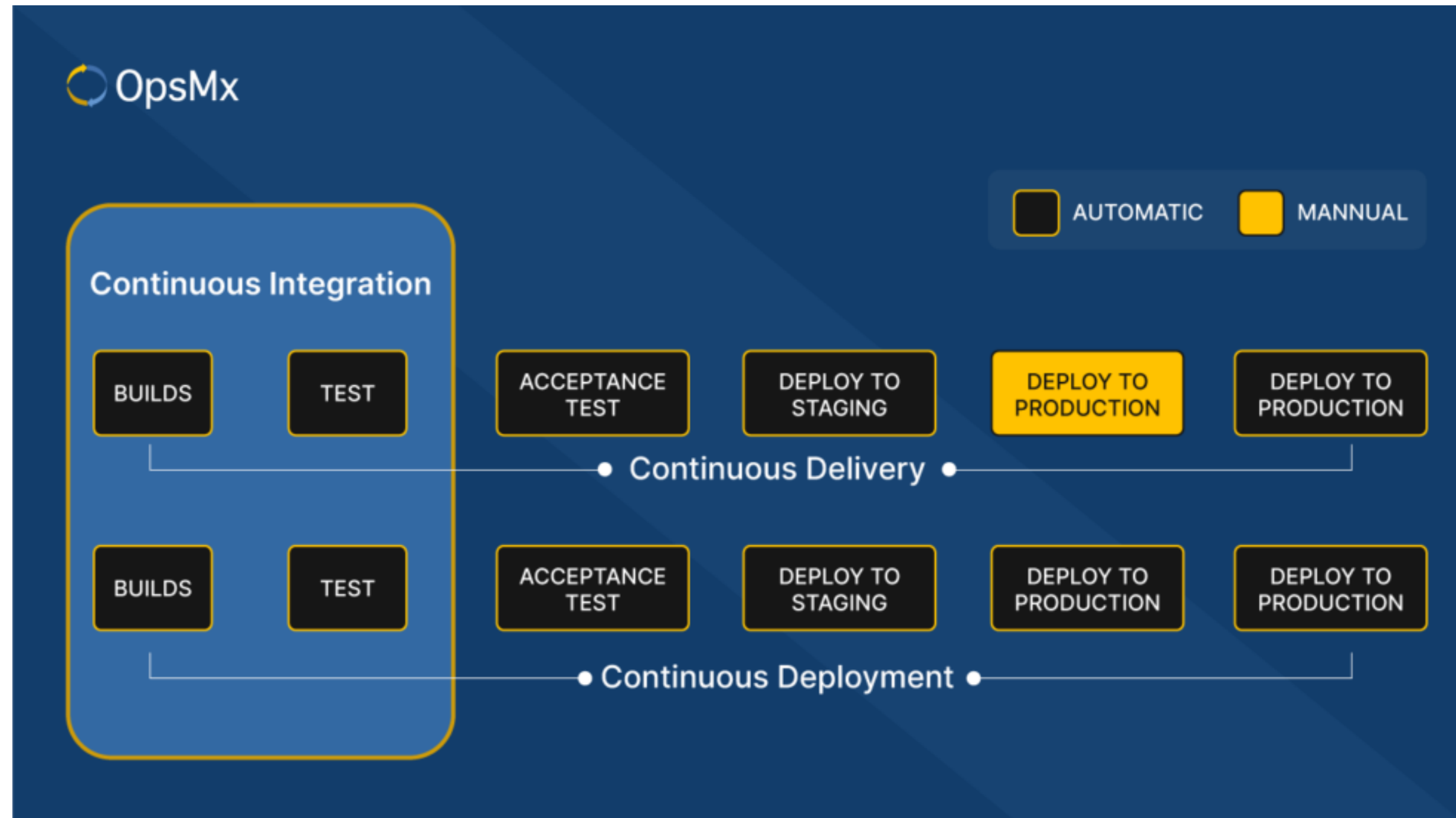


CI/CD



CI/CD Pipeline Diagram. Image credit: [GitLab](#)

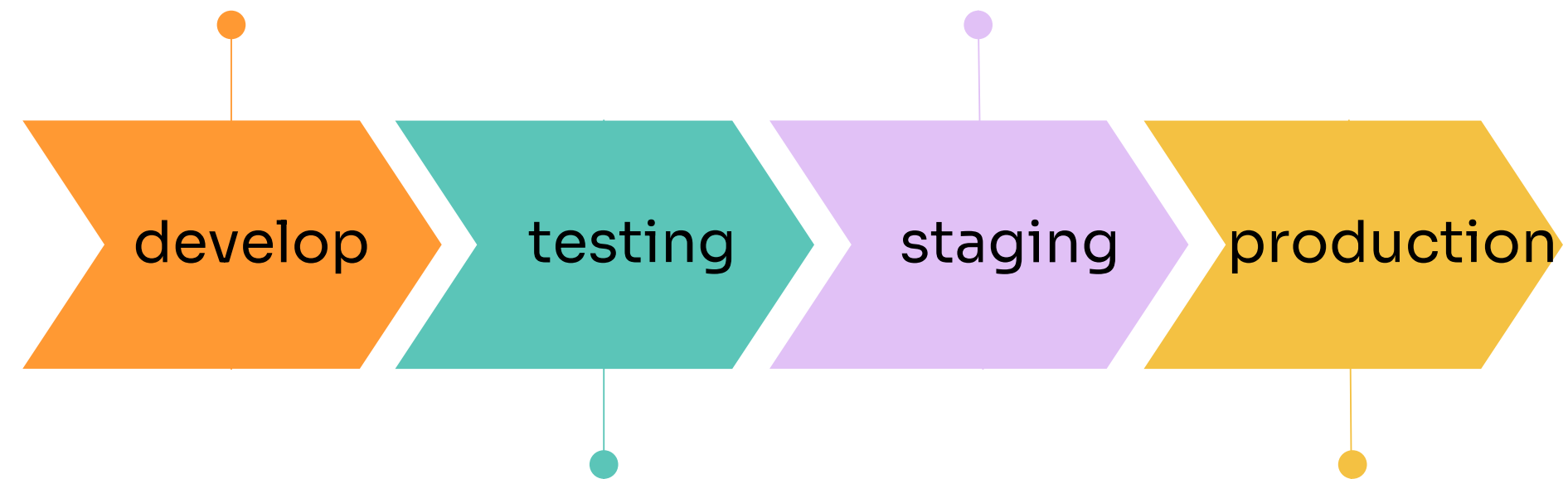
CI/CD



Środowiska aplikacji

- używane przez programistów
- brak danych

- używane przez testerów i użytkowników
- ograniczone dane systemowe
- poprawki błędów



- używane przez testerów
- brak danych
- poprawki błędów

- używane komercyjnie przez użytkowników

craftmanship

Zbiór praktyk deweloperskich

- używanie gotowych bibliotek
- code review
- wtyczki do analizy kodu i określonych standardów
- “czysty kod”
- samo-dokumentujący się kod
- uwzględnia testy

Business Case 2

Zespół podgrywał nową funkcjonalność, która wymagała przerwy serwisowej w e-commerce. Aplikacja wstała i użytkownicy już zaczęli składać nowe zamówienia.. 8 godziny później

“A: Słuchajcie dostaliśmy monit, że bazka padła 6 godzin po release.

PO: Co? To co teraz zrobimy?

B: Musimy podegrać wersję poprzednią. Spoko. Mamy backup, ale gap to jakieś z 3h.

PO: A co z zamówieniami, które w międzyczasie wleciały?

A: Puścimy joba z kolejki by spuścił dane do bazki.”

Business Case 2

observability

“A: Słuchajcie dostaliśmy monit, że bazka padła 6 godzin po release.

PO: Co? To co teraz zrobimy?

B: Musimy podegrać wersję poprzednią. Spoko. Mamy backup, ale gap to jakieś z 3h.

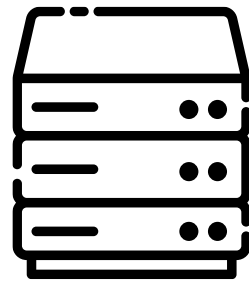
PO: A co z zamówieniami, które w międzyczasie wleciały?

A: Puścimy joba z kolejki by spuścił dane do bazki.”

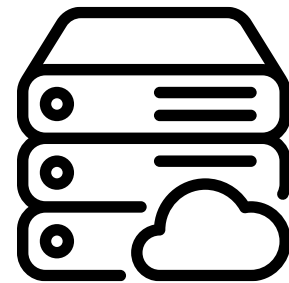
architektura
systemu

infrastruktura

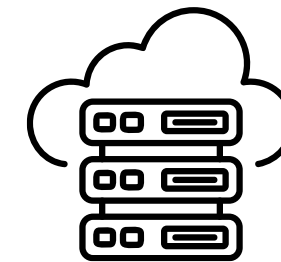
infrastruktura



On - premise



Hybrid



Cloud

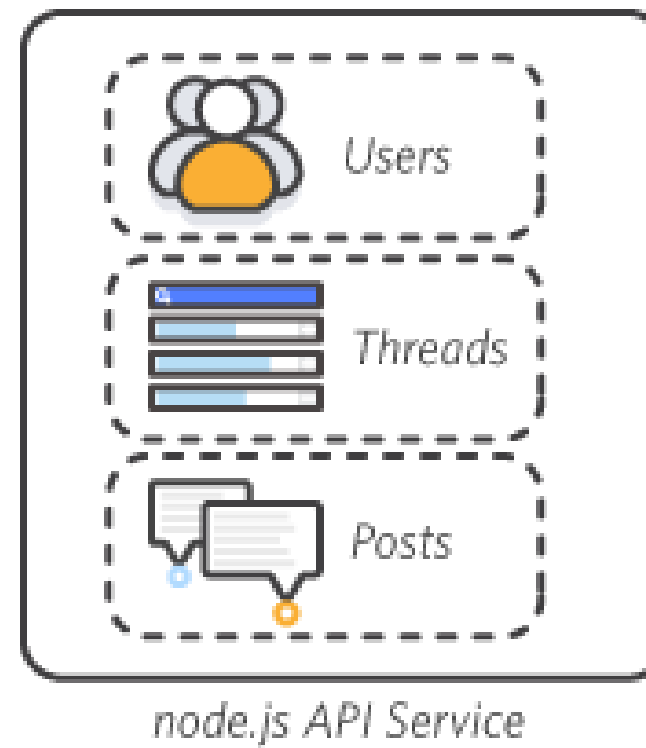
observability

= Monitoring + Logging + Tracing

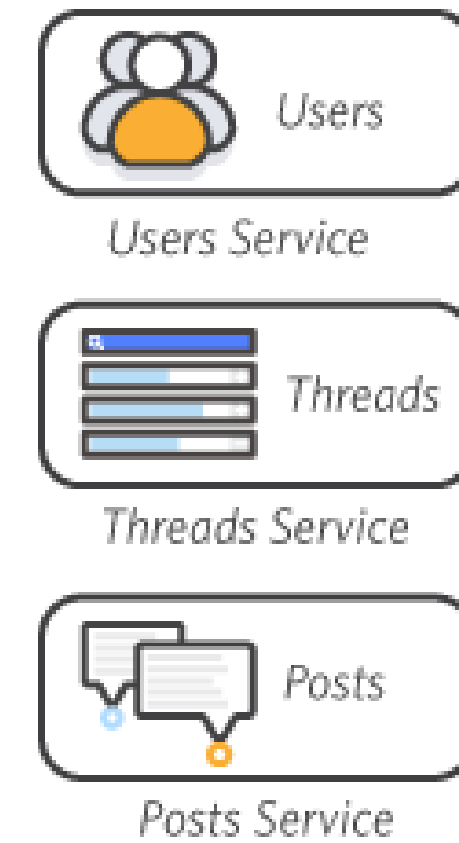
- komponenty systemowe
- eventy
- logi
- aplikacje

architektura

1. MONOLITH



2. MICROSERVICES



Business case 3

PO dostał info, że wykryto atak hakerski w nocy. Kilka osób postawiono na nogi.

“PO: Co się stało w nocy?”

A: Mieliśmy DoSa tej nocy.

PO: A konkretniej?

A: Typiarz pingał nam po 1 end-poincie.

B: Basia, nie ma stresa - szybko spróbkowaliśmy ruch i zbanowaliśmy typa.

PO: Kamień z serca, co z tym zrobimy na przyszłość?”

Business case 3

cyberbezpieczeństwo

“PO: Co się stało w nocy?”

A: Mieliliśmy DoSa tej nocy.

PO: A konkretniej?

A: Typiarz pingał nam po 1 end-poincie.

B: Basia, nie ma stresa - szybko spróbkowaliśmy ruch i zbanowaliśmy typa.

PO: Kamień z serca, co z tym zrobimy na przyszłość?”

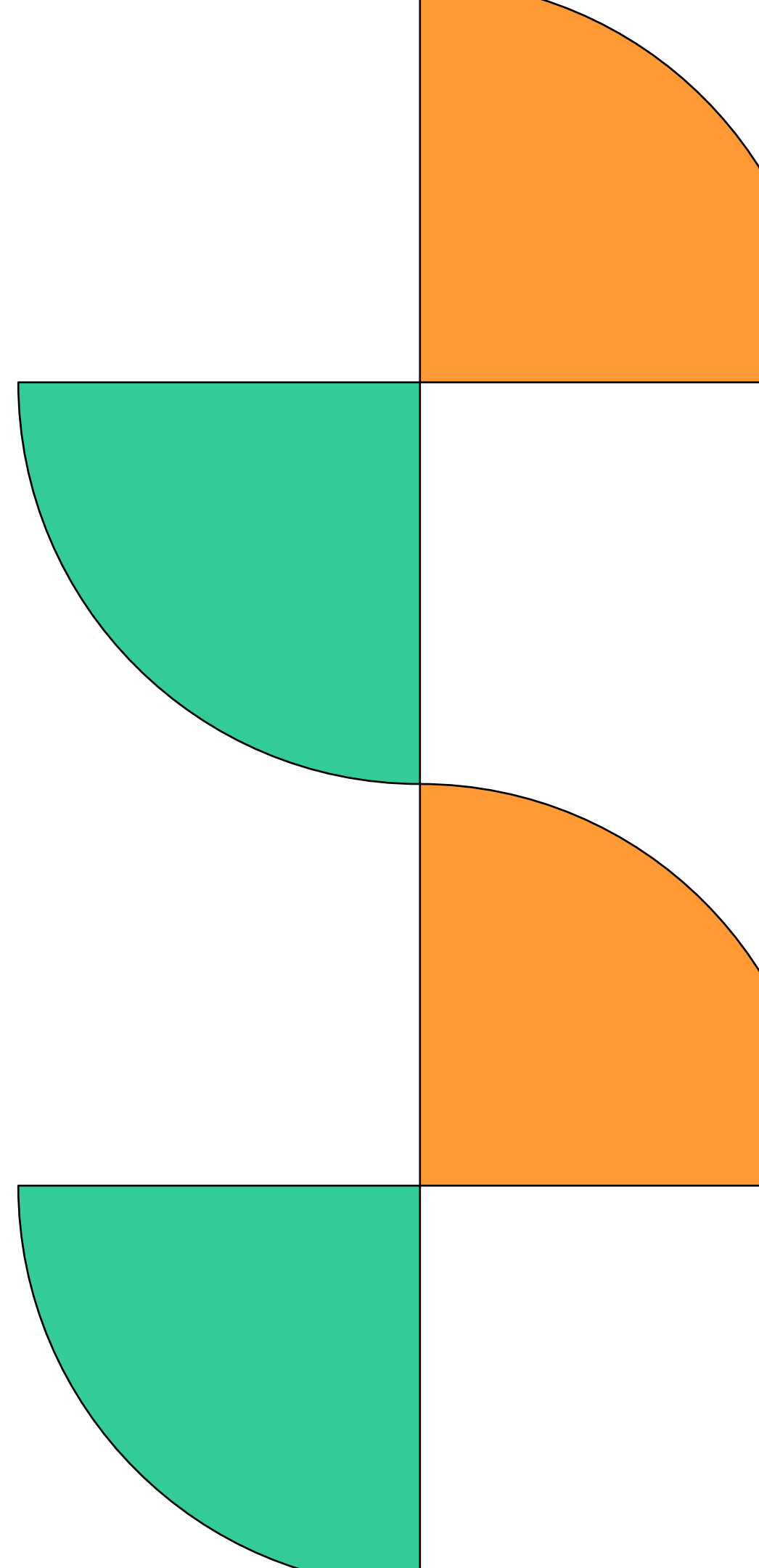
observability
infrastruktura

cyberbezpieczeństwo

- skanery bezpieczeństwa
- pen testy
- dobrze ułożona infrastruktura i system monitoringu i alertowania
- dobrze spisane testy /postawione integracje na pipeline

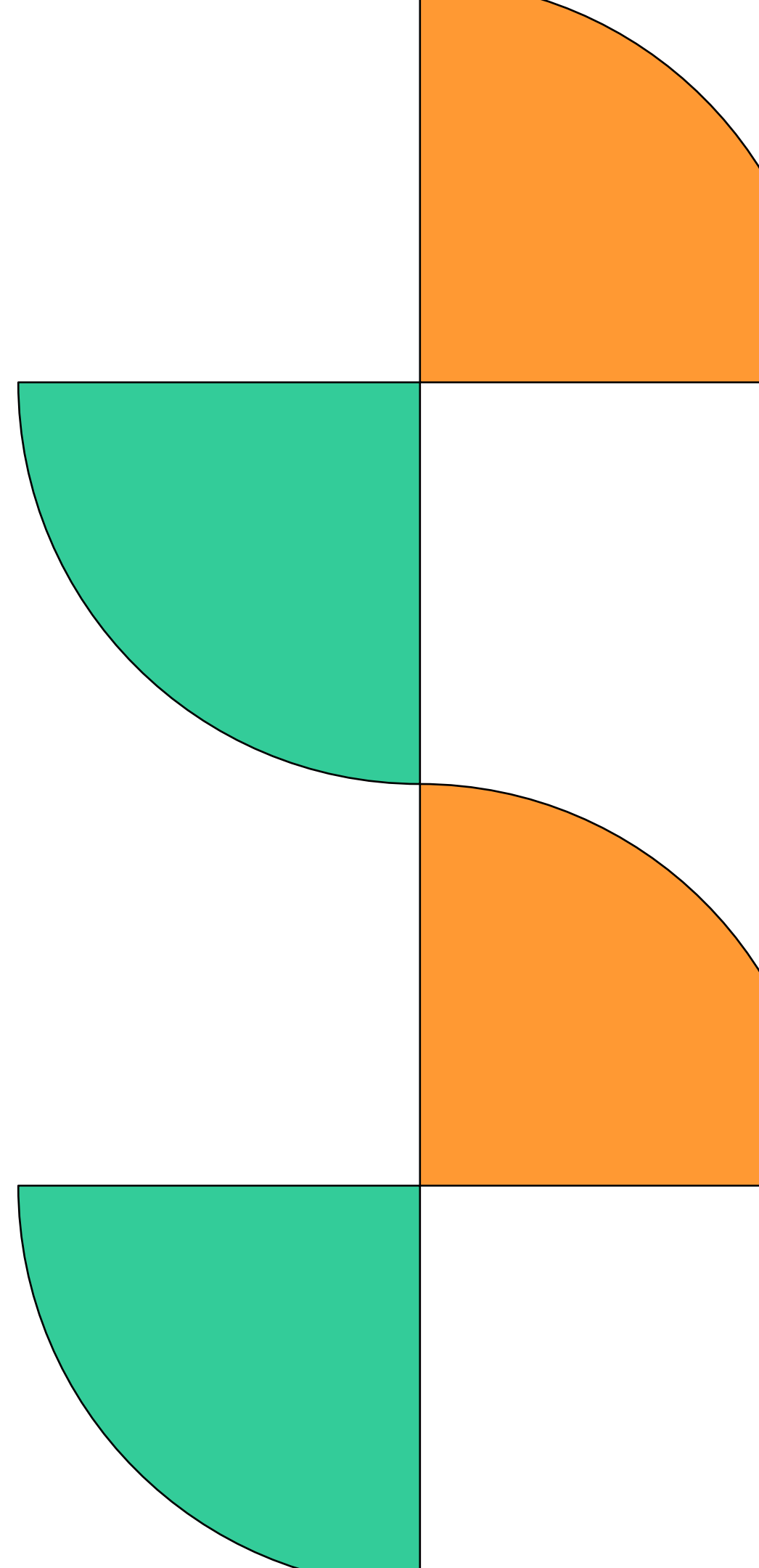
15 aspektów, które musisz znać

1. SDLC
2. języki programowania i frameworki
3. budowa aplikacji
4. środowiska aplikacji
5. zaniedbania jakości kodu
6. testowanie kodu
7. rodzaje architektury
8. rodzaje infrastruktury
9. programer craftsmanship
10. jak powstaje kod
11. kultura devops
12. automatyzacja
13. cyberbezpieczeństwo
14. utrzymanie aplikacji
15. agile i technologia



15 aspektów, które musisz znać

1. SDLC
2. języki programowania i frameworki
3. budowa aplikacji
4. środowiska aplikacji
5. zaniedbania jakości kodu
6. testowanie kodu
7. rodzaje architektury
8. rodzaje infrastruktury
9. programer craftsmanship
10. jak powstaje kod
11. kultura devops
12. automatyzacja
13. cyberbezpieczeństwo
14. utrzymanie aplikacji
15. agile i technologia



Gdzie mnie znaleźć?



LinkedIn

[kate prokopiuk](#)

E-mail

kontakt@technicznysm.pl