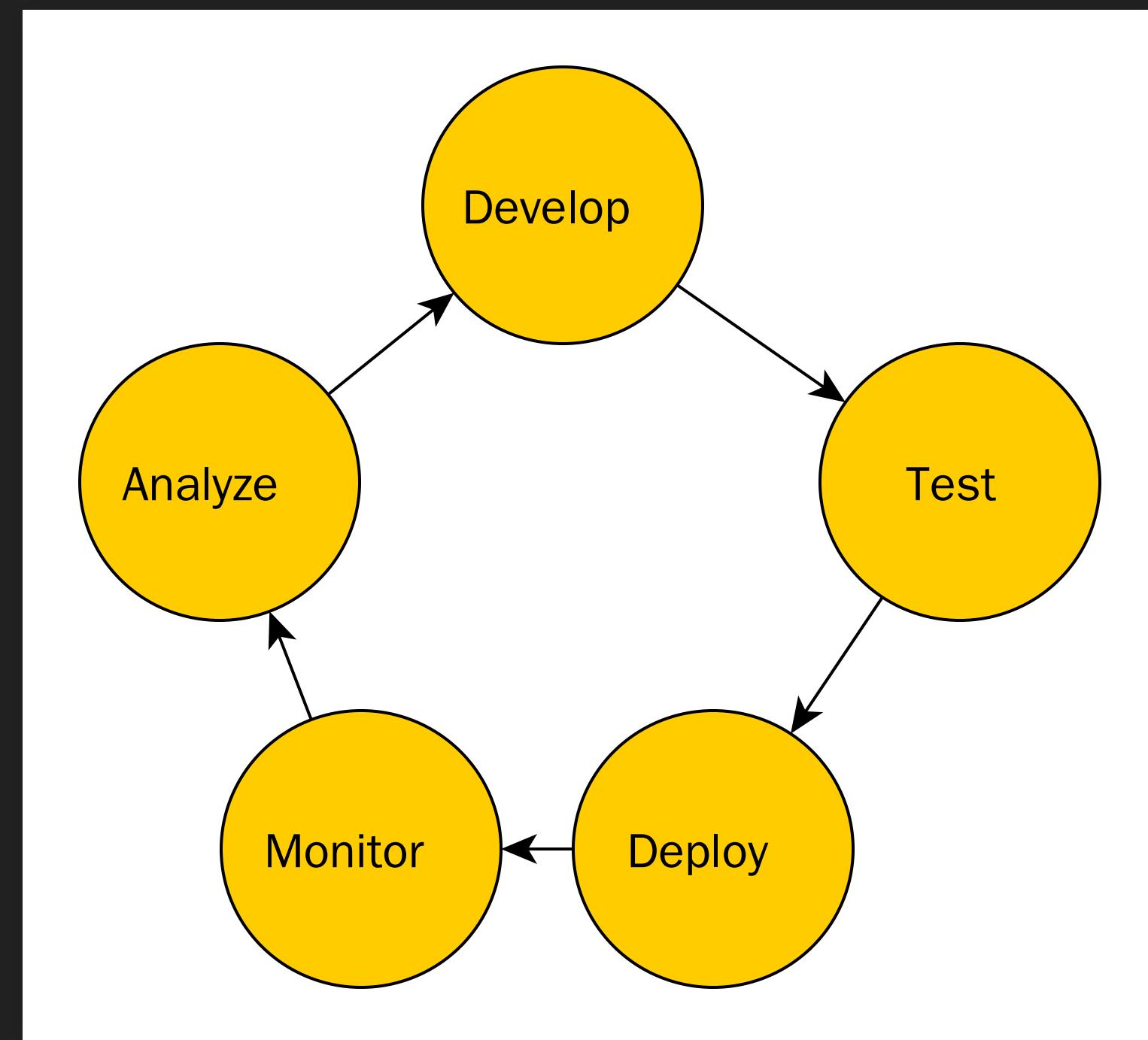


CONTINUOUS.*



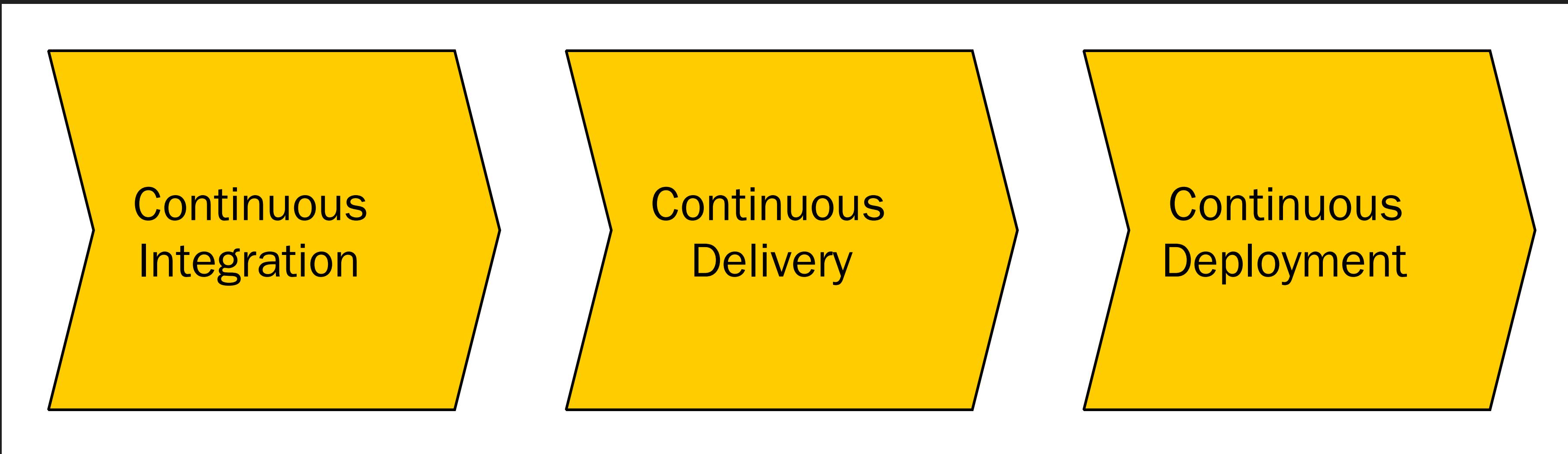
Dlaczego jest ważne

- zrozumieć jak buduje się obecnie oprogramowanie

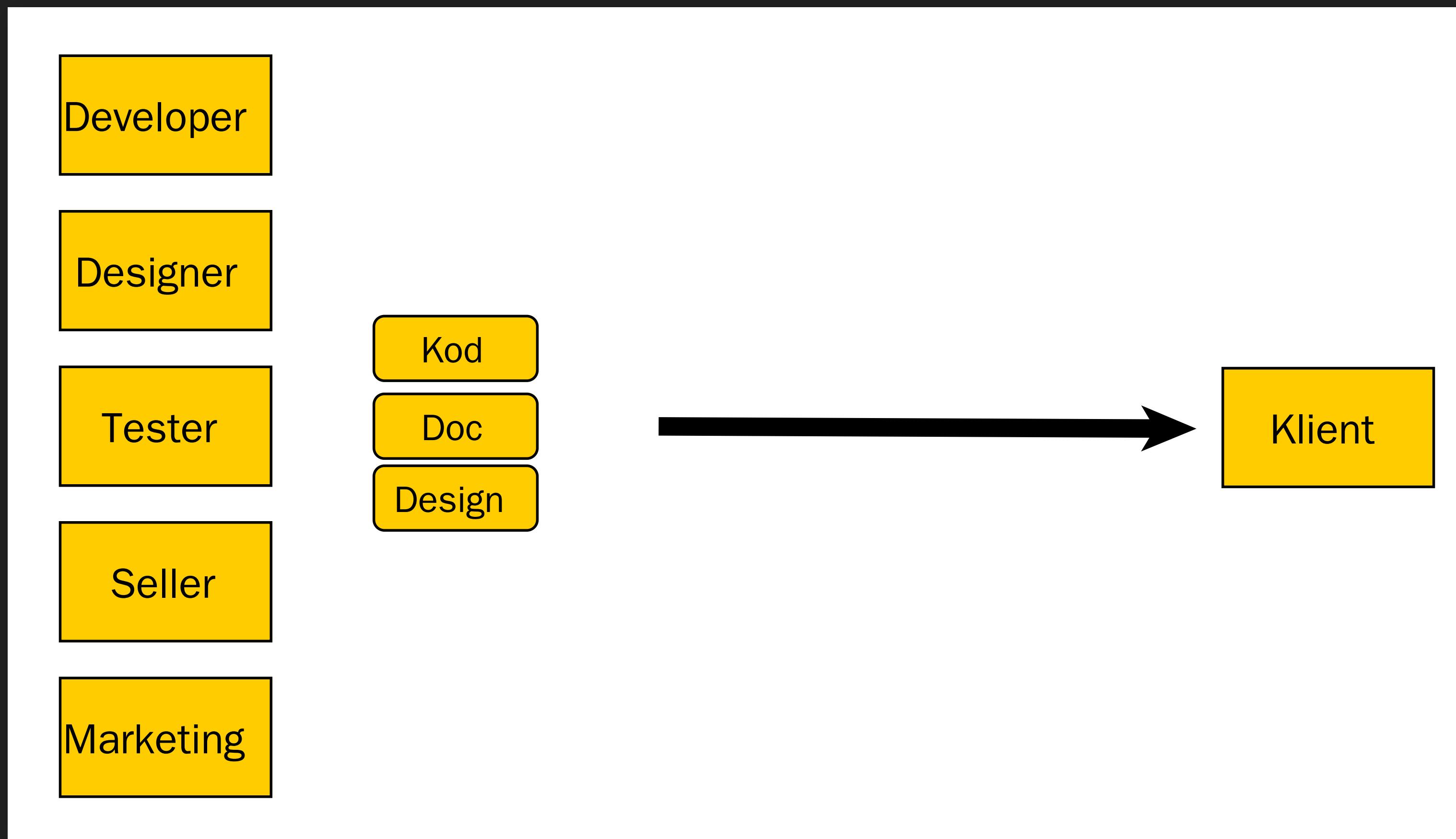
Procesy ciągłej

- Integracji (1)
- Dostarczania (1)
- Deploymentu (2)

CONTINUOUS.*



CONTINUOUS.*



CEL

- Release bez ekscytacji
- Release bez zawału serca
- Automatyzacja
- Bez u mnie to działa!
- Szybsze dostarczenie feature do klienta

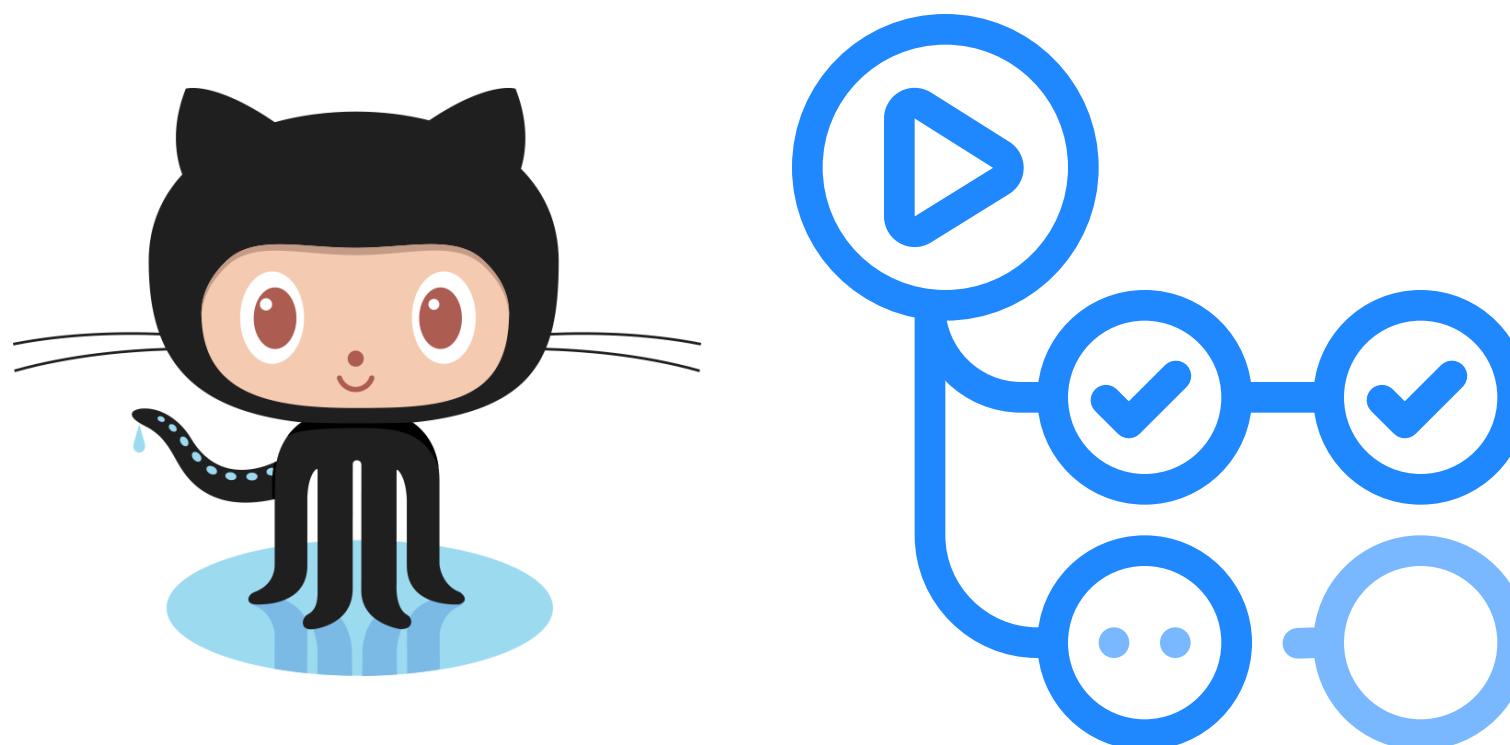
CEL

- Wiemy w jakim stanie jest nasze oprogramowanie

Praktyki

- Automatyzacji
- Standaryzacji
- Wykorzytanie narzędzi
- Dostosowania oprogramowania
- Praca z ludźmi

Plan zajęć



Ciągła integracja

- Każda zmiana (commit) zwalidowana
przetestowana
- Przygotowany pakiet do instalacji

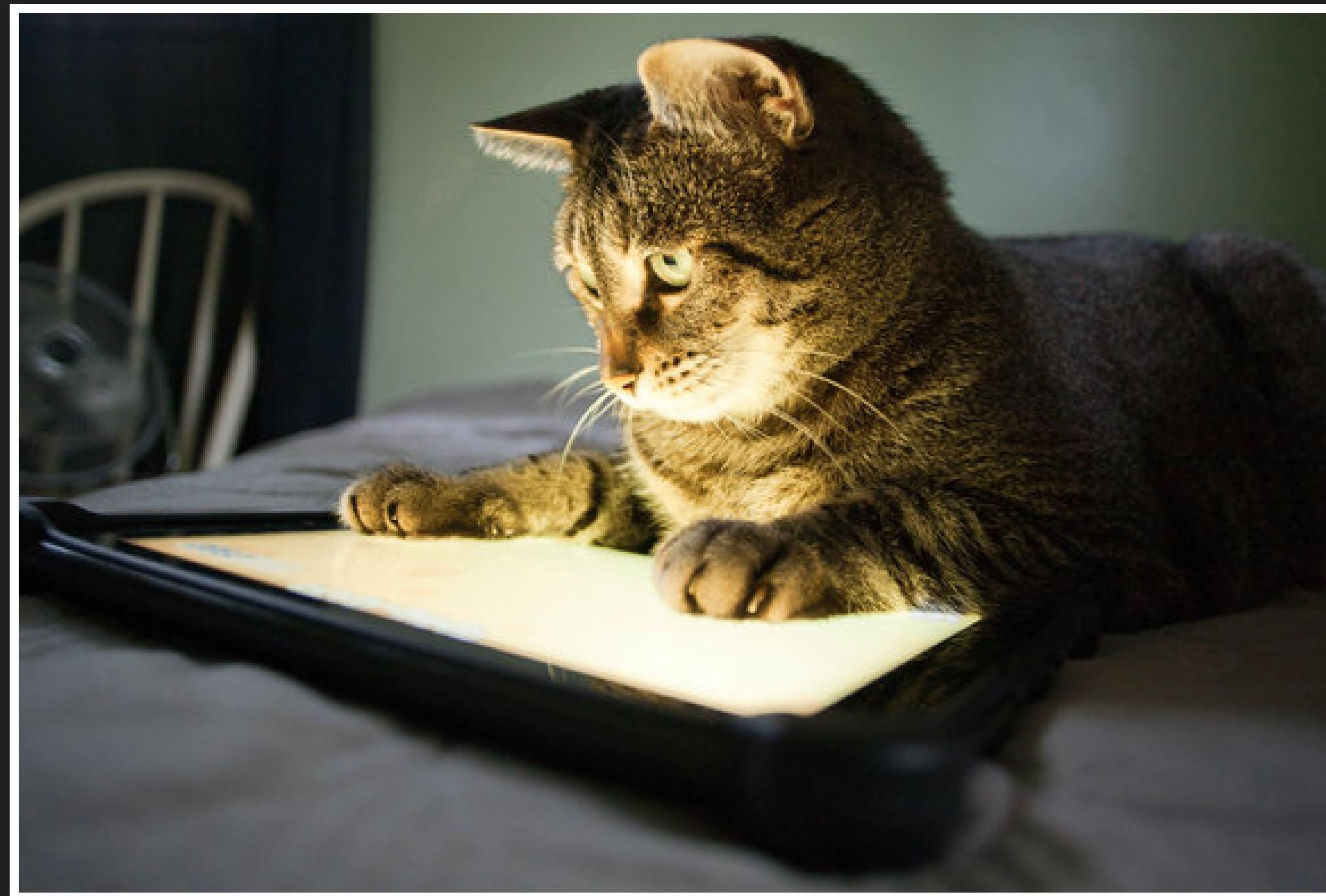
Ciągła integracja



Ciągła dostarczanie

- Oprogramowanie gotowe do instalacji

Ciągła deployment



Oprogramowania w rękach klienta

Gdzie się zaczyna CI a CD?

Continuous
Integration

Continuous
Delivery

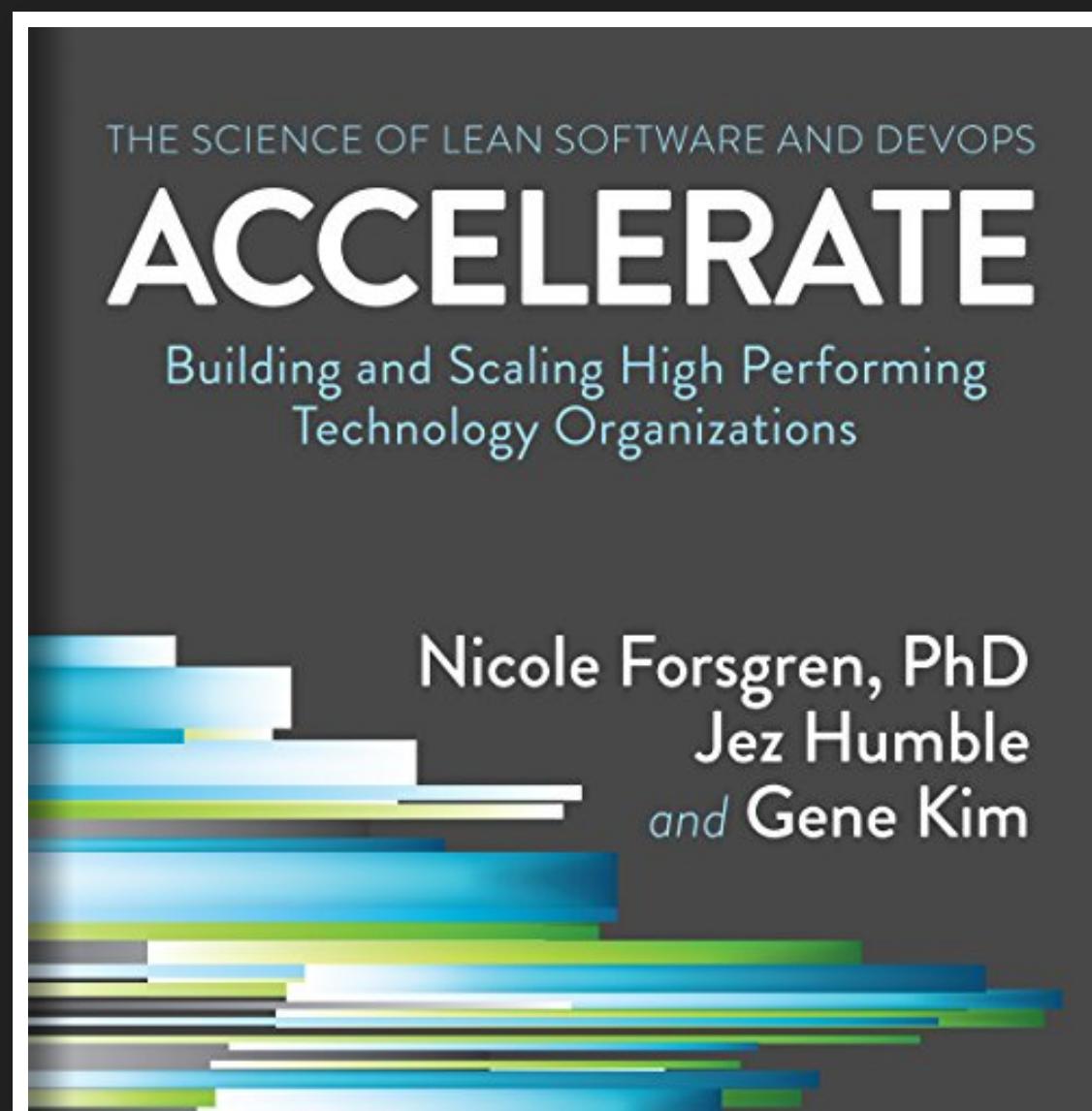
Continuous
Deployment

Najlepsze firmy

- N zmian do produkcji dziennie
- Release -> Business
- Release -> Klient
- Amazon w 2013, miał deployment nowego kodu co 11.6 sekundy

Co rozróżnia wiodące firmy od innych?

- Lead Time
- **Deployment Frequency**
- Mean Time to Recovery
- Change to Fail



Patrz książki DevOps Handbook i Accelerate

Kto pracuje

- Deweloperzy
- Test Inżynierowie
- Inżynierowie DevOps

AUTOMATYZACJA

Czego się nauczyliśmy

- Ludzie nie radzą sobie z powtarzanymi zadaniami
- Czym później wykryty błąd, tym więcej kosztuje

Niebezpieczeństwa

- Rabbit hole
- Za wcześnie*

[*]

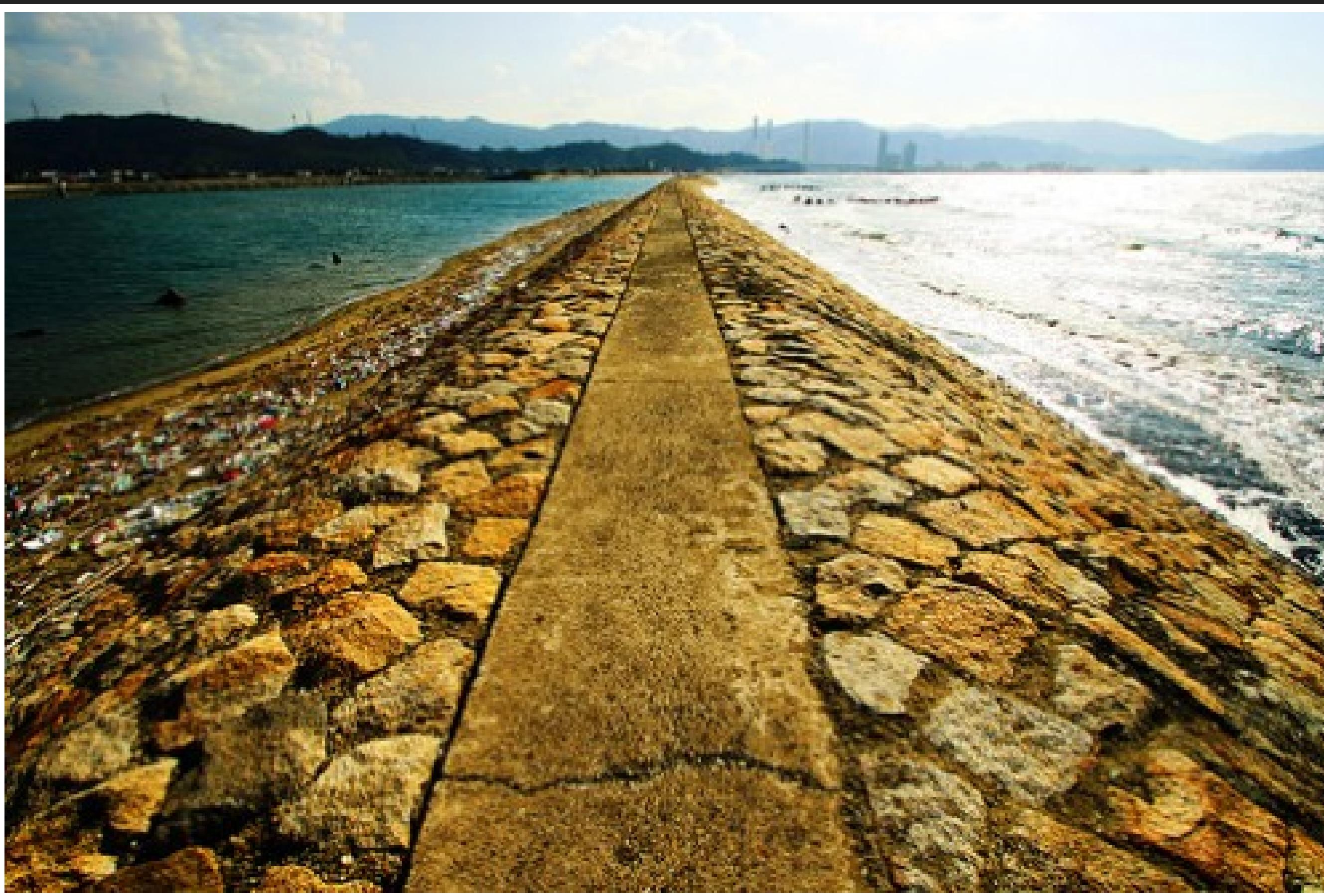
Kroki

1. Dobre notatki - praktycznie z instrukcjami do copy & paste.
2. Skryty
3. Automatyzacja

Automate Everything!



Złoty Środek.



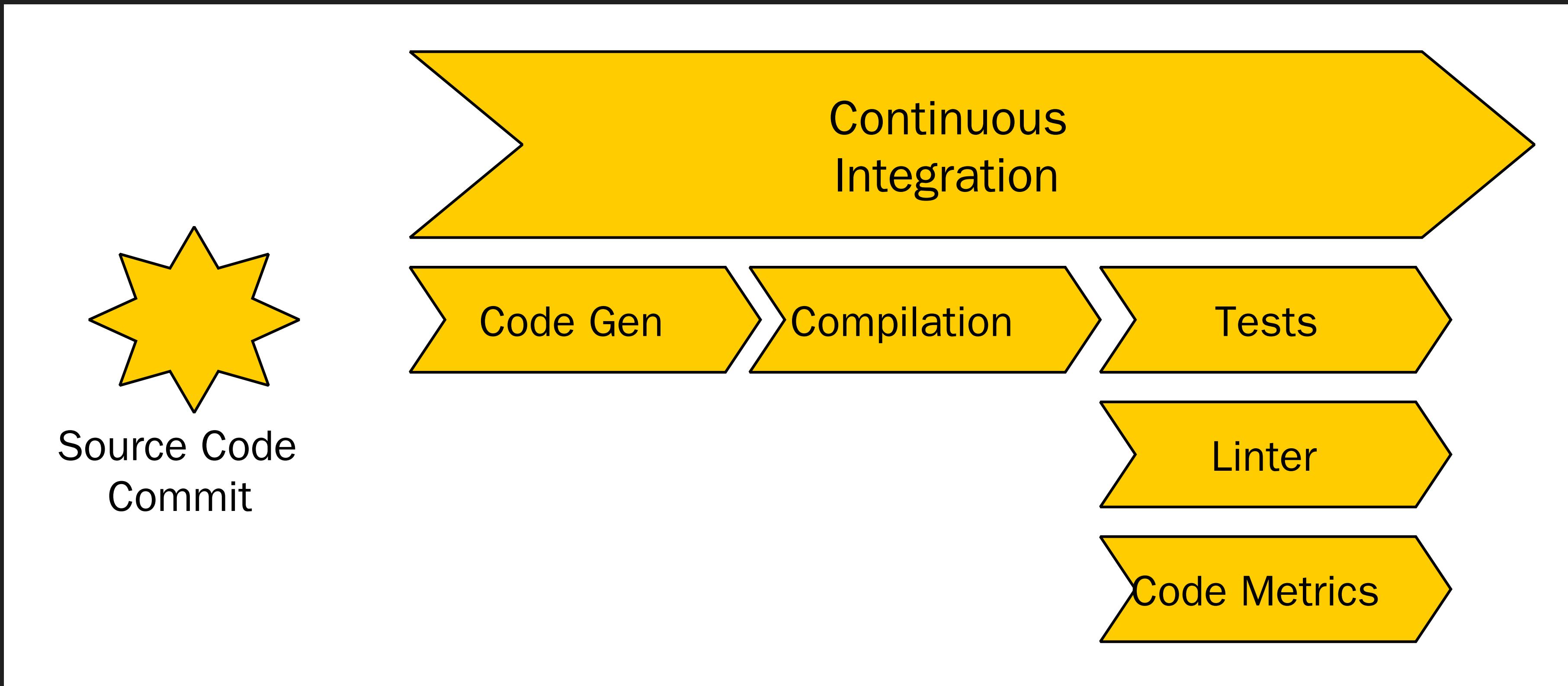
Warto się zapoznać

- Książka "Automate the Boring Stuff with Python"
[\(online\)](#)
- Wykłady MIT [Missing Semester](#)

Ciągła Integracja



Ciągła Integracja

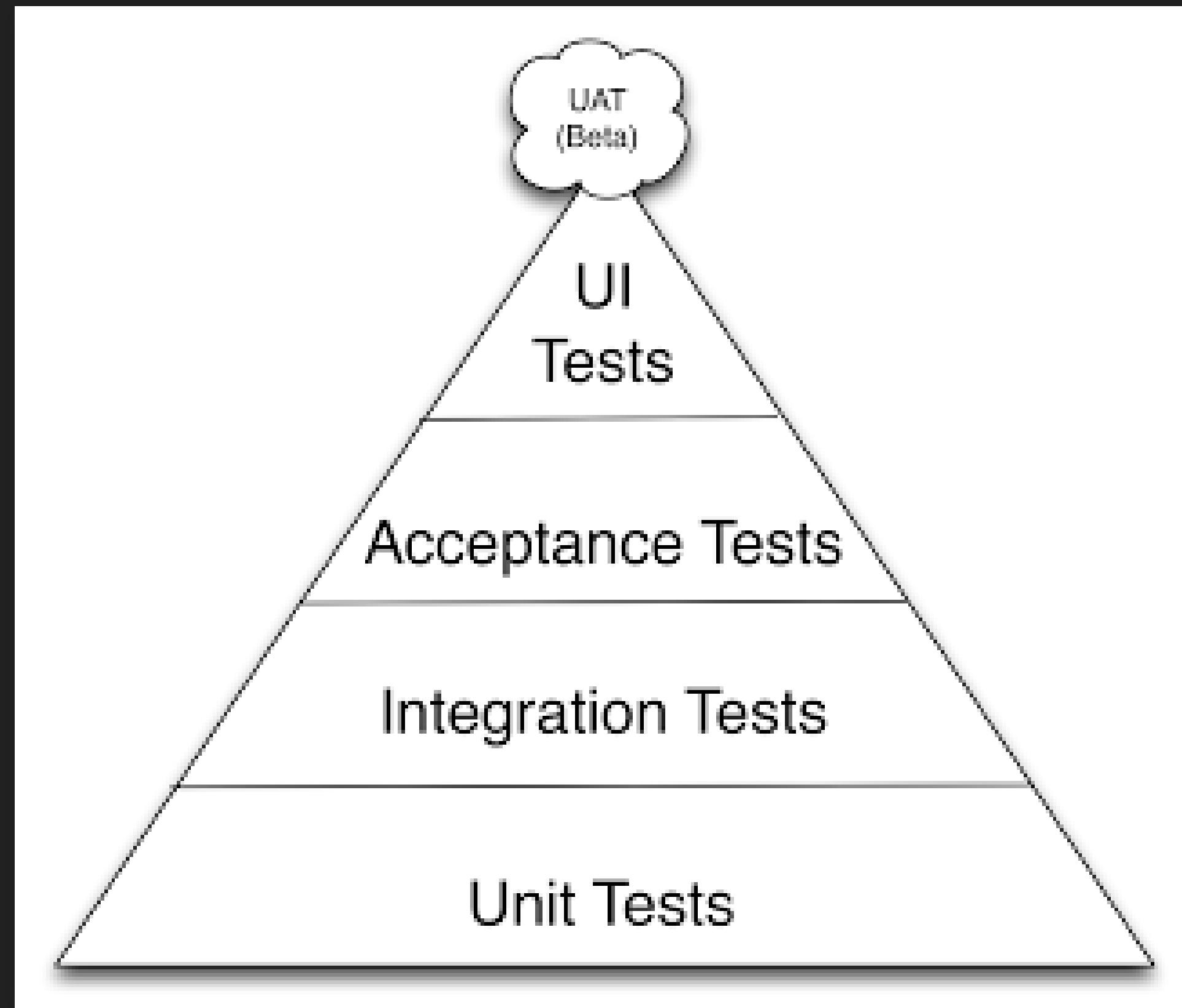


Niektóre kroki mogą być manualne (jako step albo PR)!

DECYZJE

- konwencja dla git-a
- jedno czy wiele repozytorium
- metryki, konwencje nazewnictwo ...
- dokumentacja ...

Rola testera



Integruj stabilne testy i najszybsze w pierwszej kolejności!

Zabójca CI/CD

- niestabilne testy i skrypty
- wooooolne testy*

* - można przenieść je do testów uruchamianych nocą

Co się stanie jak ci/cd nie działa?

DZIAŁAJĄCY CI/CD JEST
##najwyższym priorytetem

Kilka pomysłów

- Retaliation
- Sygnał

CI zaczyna się na maszynie dev

CI zaczyna się na maszynie dev

- tool wspólne dla automatyzacji i dev
- testy (albo ich część) do lokalnego testowania
bardzo szybkie
- skrypt do uruchomiania komponentu *

* patrz: docker-compose

Testy

- Testy Eksploracyjne
- Testy Weryfikacyjne Automatyczna

Testy - Automatyczne

- Pasy bezpieczeństwa/Spadochron
- Weryfikacja czy czegoś nie zepsuliśmy
- Fuzzing

Testy Eksporacyjne

- Manualne
- Może być wspierany przez narzędzia
(curl, postman, selenium, tcpdump, consola dev)

Patrz : [Jak można zarobić 15k na Bug Bounty](#)

Testy Eksploracyjne

W domu: James Bach on testing in an agile software development team.

Metryki

- Zdrowie (build health)
- Pokrycie (test coverage)
- test reproducibility (☠☠☠)
- czas działania testów
(unit, integration, black, white box, ...) !!!!!

Metryki

Jenkins

Suisse Stop-tabac dev

Back to Dashboard Status Changes Workspace Build Now Delete Project Configure Set Next Build Number Duplicate Code Coverage Report SLOCCount Git Polling Log

Project Stop-tabac dev

CI build

Coverage Report Workspace Recent Changes Latest Test Result (no failures)

Test Result Trend

count

#90 #171 #210 #263 #345 #426 #438 #977

(just show failures) enlarge

Build History (trend)

- Last build (#977), 3 min 17 sec ago
- Last stable build (#977), 3 min 17 sec ago
- Last successful build (#977), 3 min 17 sec ago

RSS for all RSS for failures

Permalinks

Code Coverage

Classes	45%	Conditionals	74%	Files	45%	Lines	28%	Packages	88%
---------	-----	--------------	-----	-------	-----	-------	-----	----------	-----

100%
80%
70%
60%
50%
40%
30%
20%
10%
0%

#171 #210 #263 #345 #426 #438 #977

Classes Conditionals Files Lines Packages

SLOCCount Trend

lines

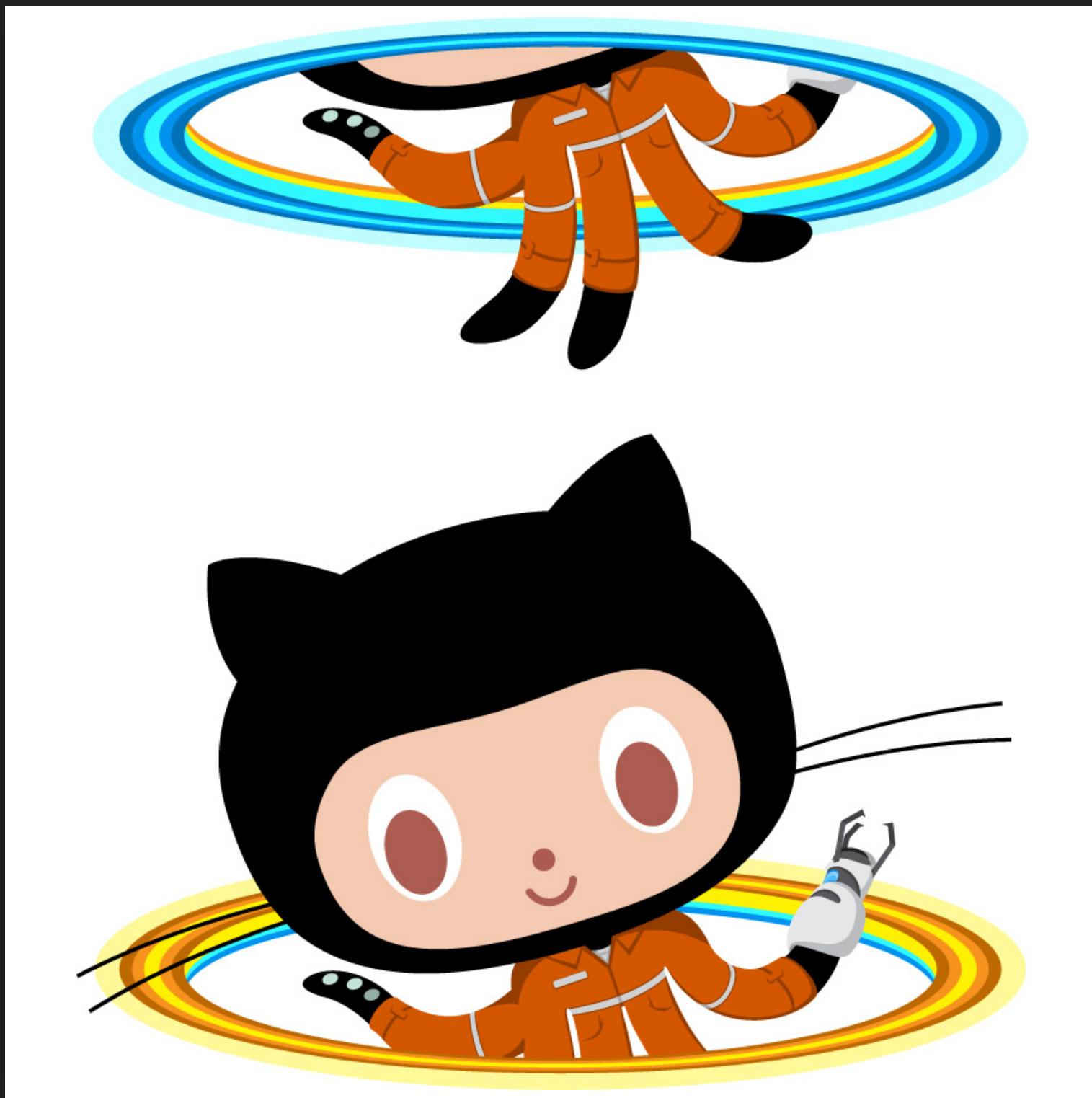
8.000
6.000
4.000
2.000
0

Jeśli wystąpi błąd

- email
- slack / skype / chat / Discord

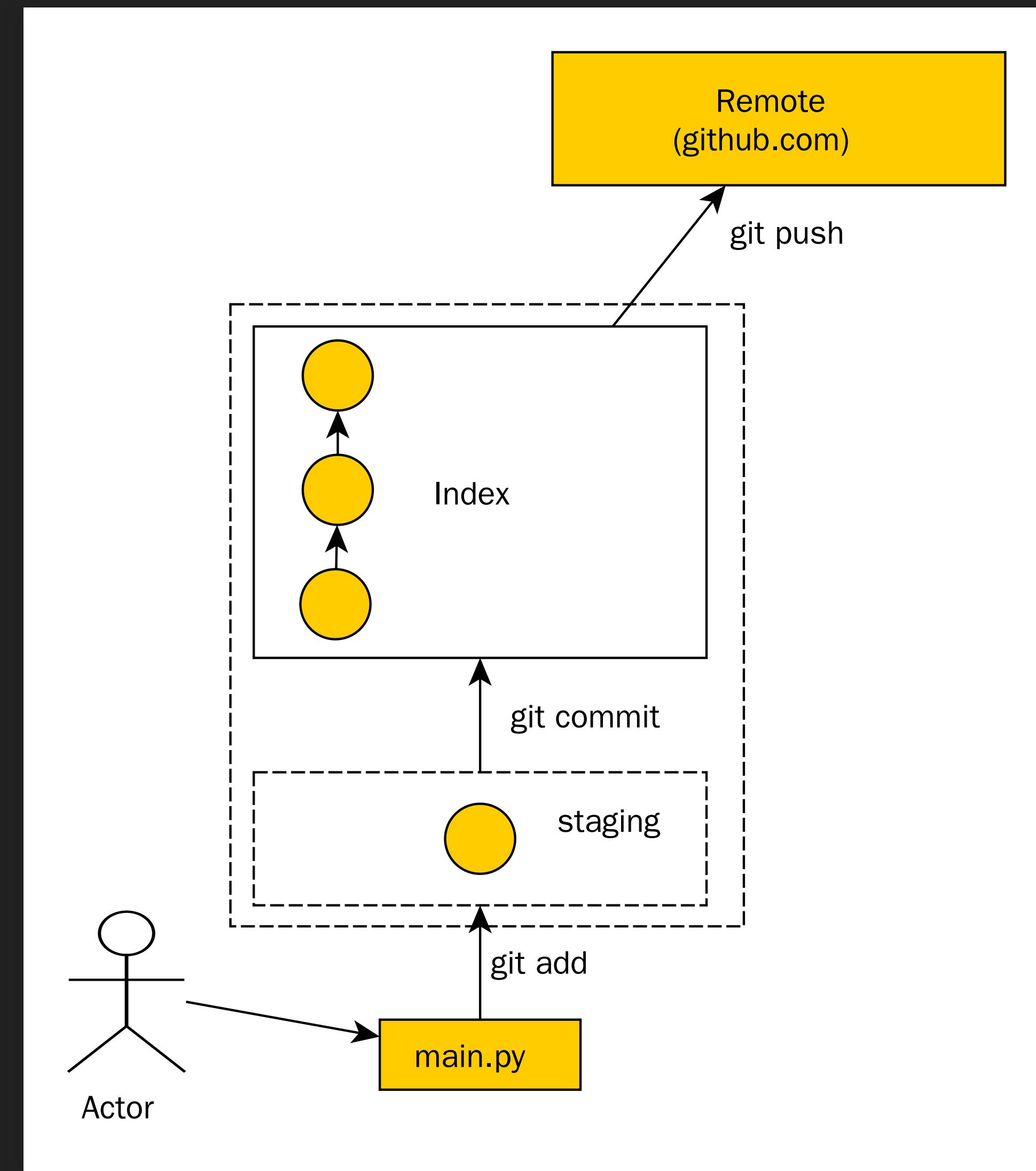
Dlaczego Git?

- Najpopularniejszy SCM
- Github
- Mnóstwo narzędzi
- Alternatywa: mercurial



<https://octodex.github.com/>

Git z centralnym repo



Zauważ: są inne modele wykorzystania git-a, patrz, linux kernel

Git

- klasycznie - master / develop
- prosty: master + krótko żyjące feature branches
- [github flow](#) (rekomendowany)
- ... wiele innych
- złożony - [git-flow](#)

Git

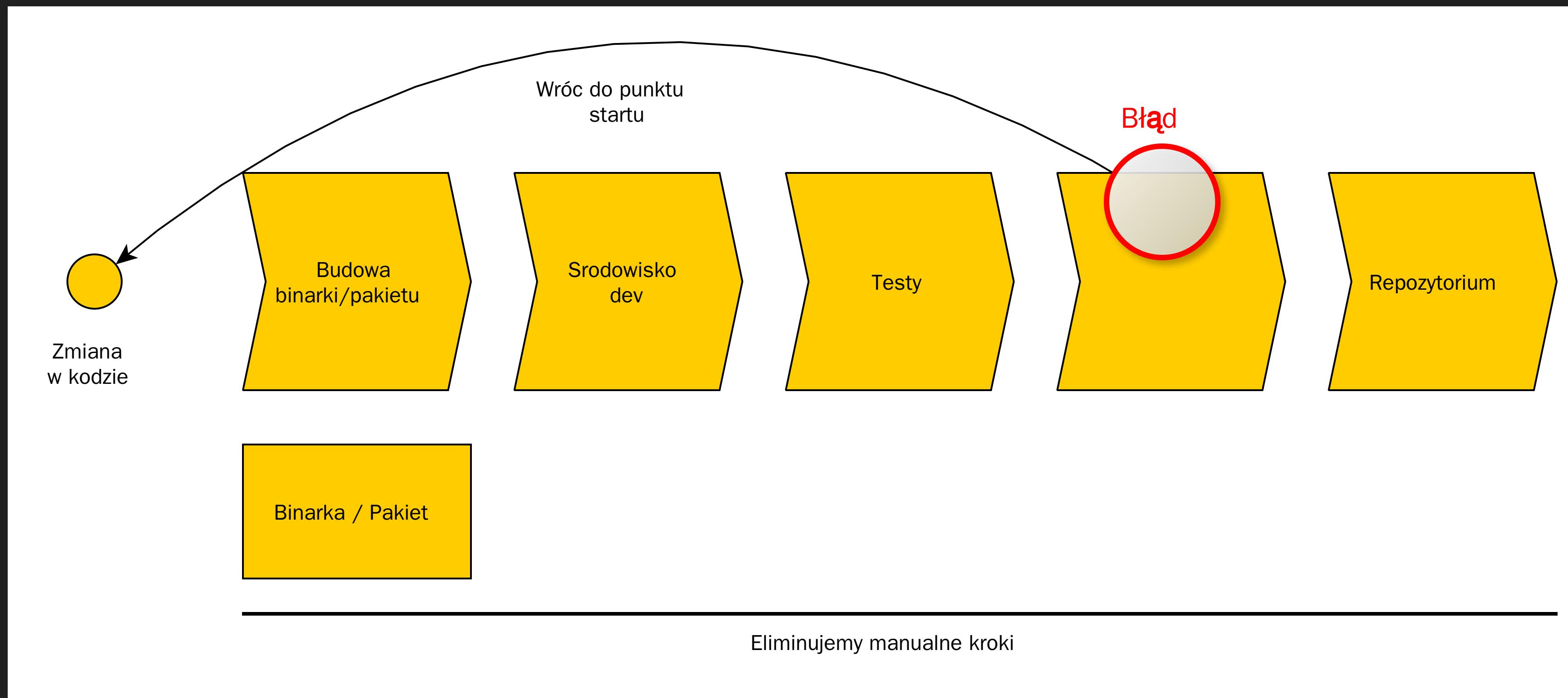
- 1 repo dla 1 produktu (wielu komponentów)
- 1 repo dla wszystkich komponentów

CONTINUOUS DELIVERY



Przygotowanie do Release-u / Deployment do

ZASADY



ZASADY

- proces Release, Delivery... Deployment musi być powtarzalny i solidny
- automatyzacja wszystkiego
- jeśli coś jest trudne, rób to częściej i zautomatyzuj
- wszystko w systemie kontroli wersji

The Practice of Cloud System Administration: DevOps and SRE Practices for WS, Volume 2

ZASADY

- DONE = w rękach klienta
- wszyscy współdzielą odpowiedzialność za proces
- ciągle poprawiaj proces / kod / narzędzia

PRAKTYKI

- zbuduj binarkę / pakiet raz
- używaj tej samej binarki pakietu w całym procesie
- proste testy (smoke tests) w docelowej platformie / środowisku

PRAKTYKI

- zweryfikuj monitoring w docelowej platformie / środowisku
- jeśli cokolwiek zawiodło, zacznij od początku

CI | CD

Niezbędne narzędzie wprowadzenie kultury DevOps w
firmie

The DevOps Handbook

PAKIET: DOCKER



Przejdźmy do tablicy.

PAKIET

- "u mnie działa"
- dependency hell

KOSZMAR

- A -> B -> C 1.0 and A-> C 2.0 
- biblioteki aplikacji
- biblioteki natywne
- ...

PAKIET: PAKIETY

- deployment z githuba
- pip
- deb/rpm
- fat packages (apt, runtime, ...), e.g., dh-virtualenv

PAKIET: WIRTUALNE MASZYNY

Plusy:

- ultimate fat packages
- wszystko od bibliotek aplikacji do natywnych

Minusy:

- duży rozmiar
- wolno się budują

PAKIET: DOCKER

Plusy:

- coś jak VM ale szybkie w tworzeniu i deploymentu
- proste API
- Podejście społeczności Dockera

Minusy:

- izolacja gościa od hosta

PAKIET: DOCKER

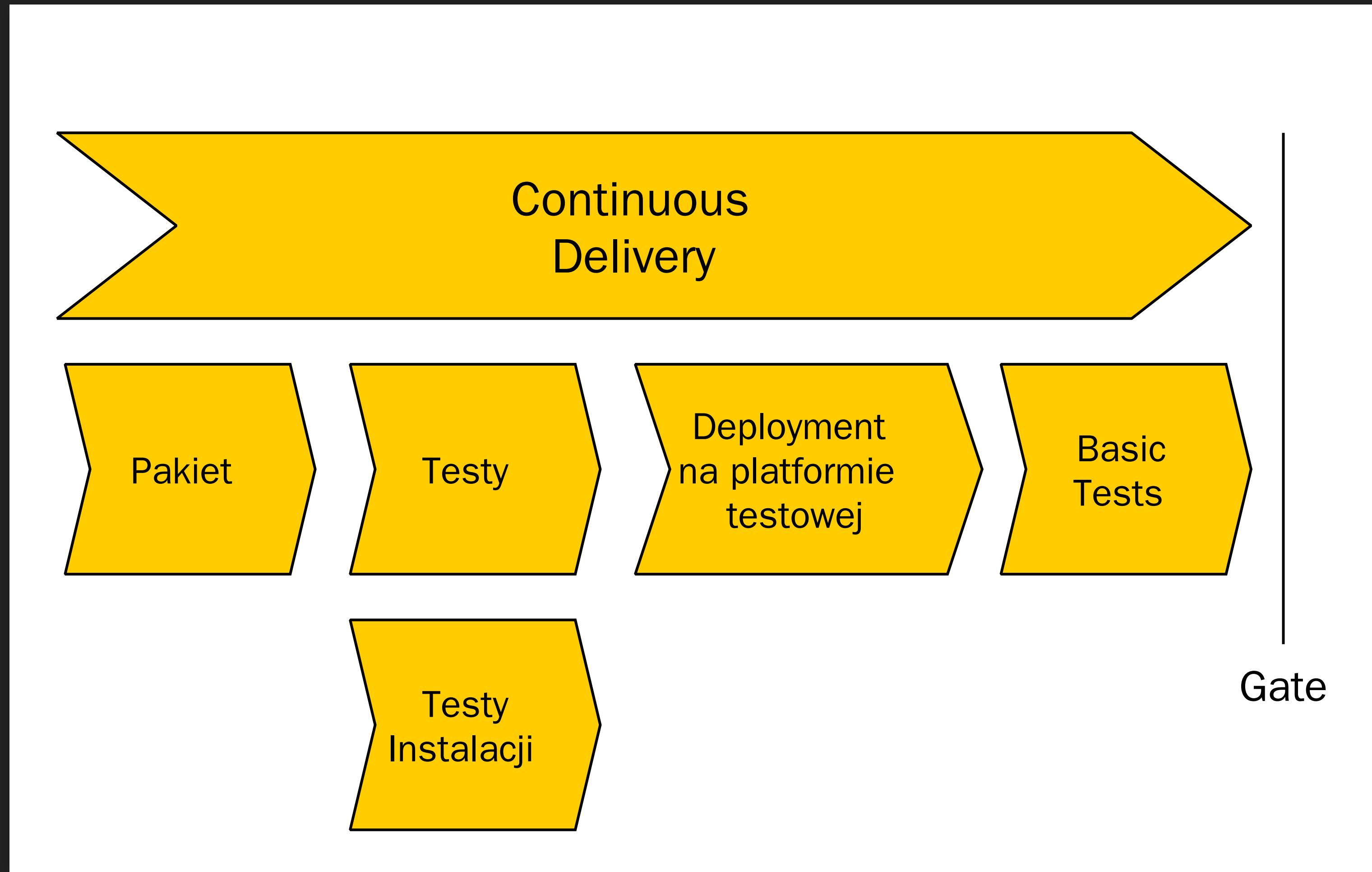
Platformy aplikacji w oparciu o Dockera:

- Kubernetes

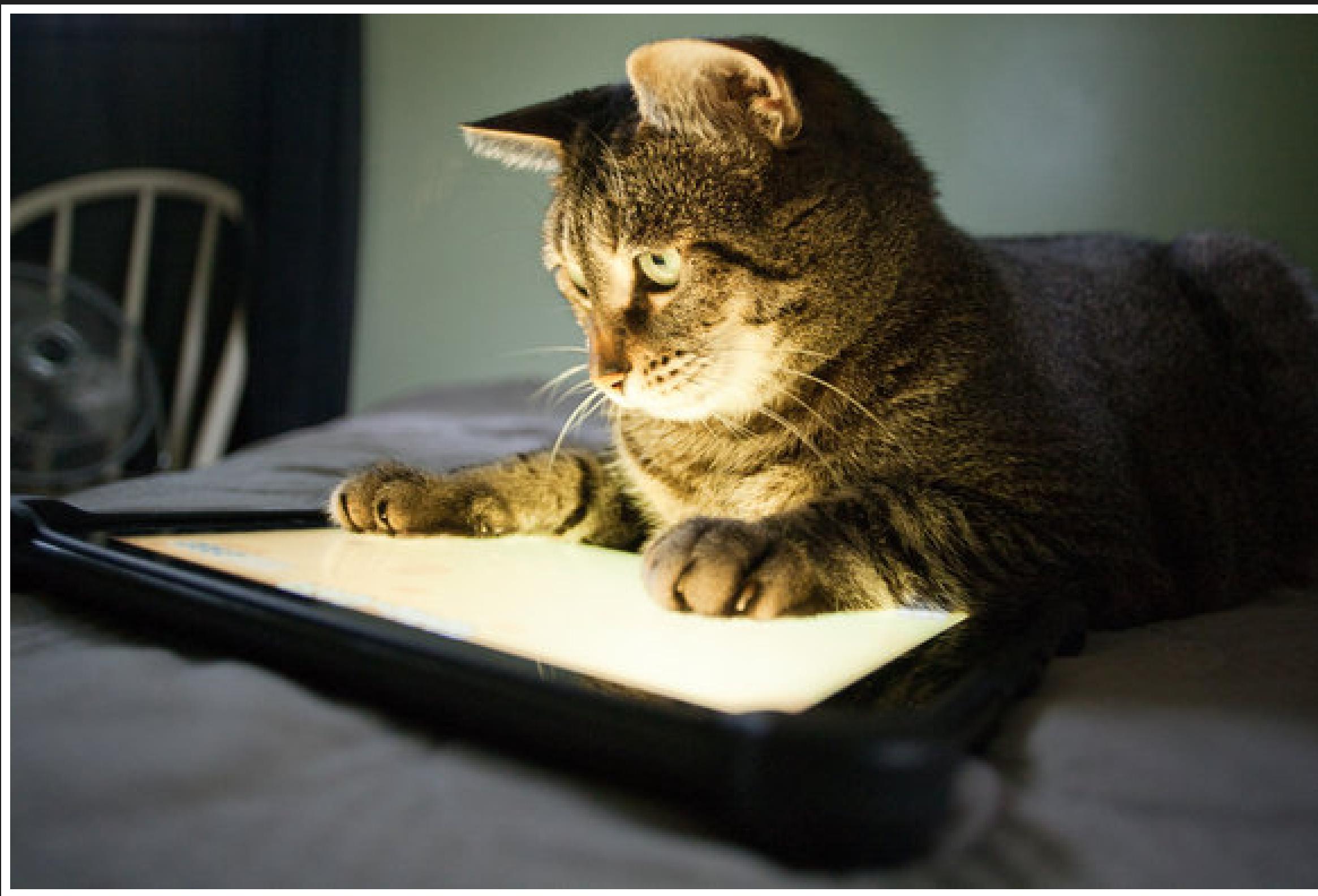
CONTINUOUS DEPLOYMENT



CONTINUOUS DEPLOYMENT



CONTINUOUS DEPLOYMENT



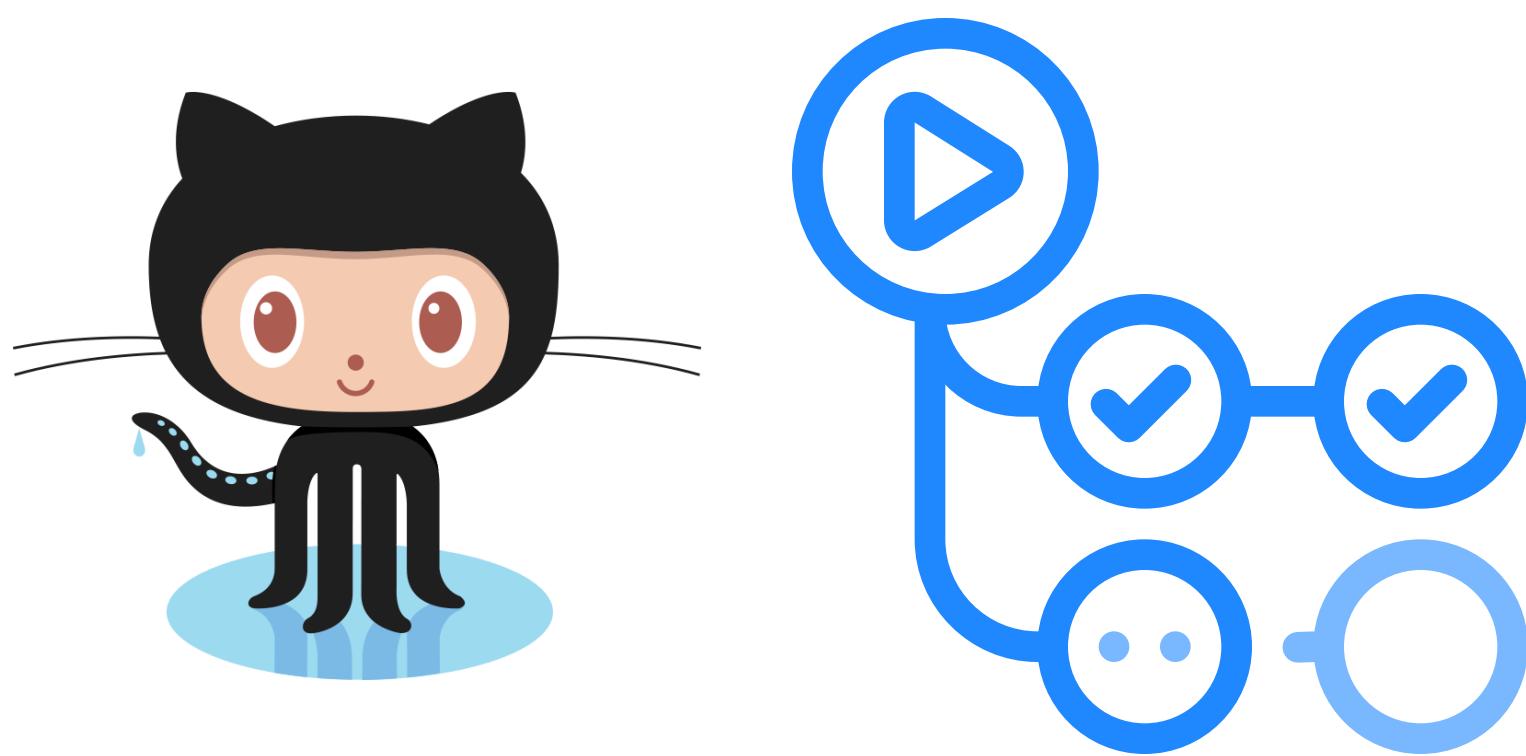
Szczęśliwy klient

STRACH



Push to production

Plan zajęć 1: deployment

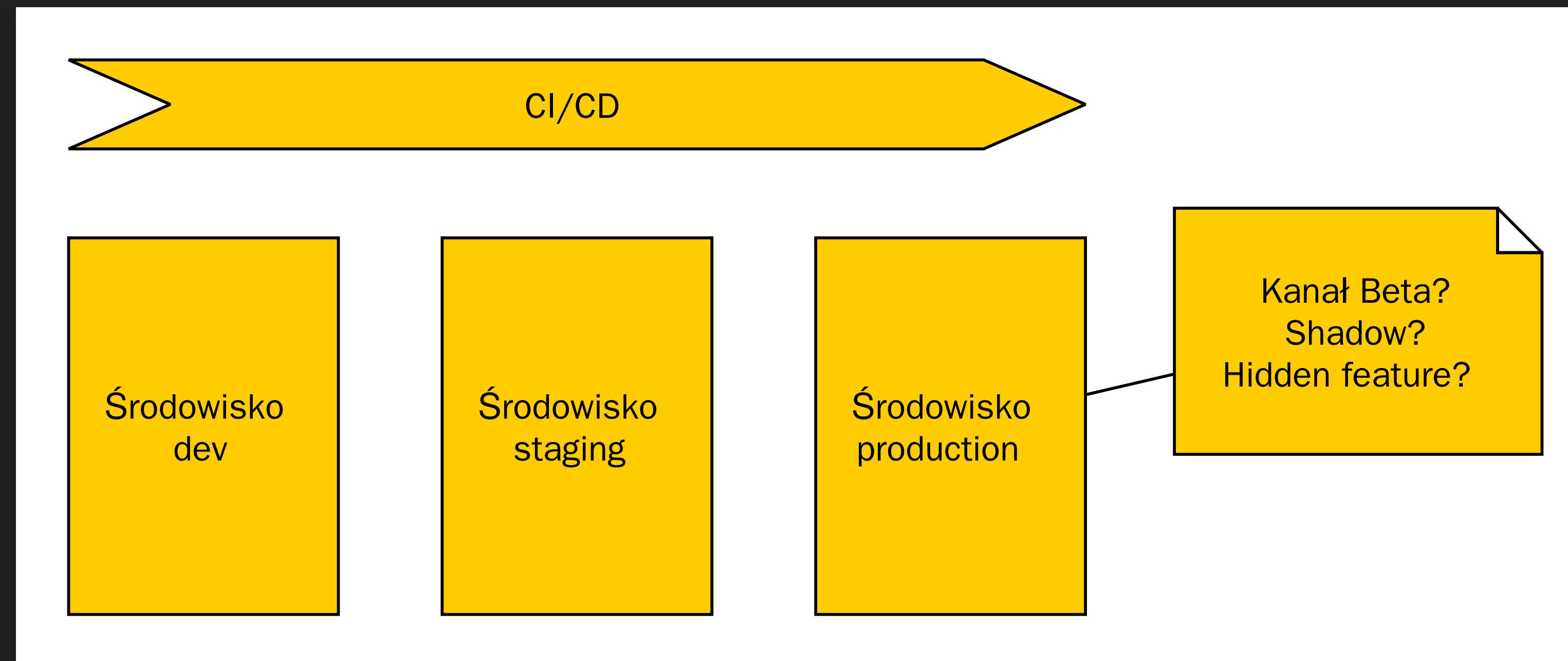


+ [statuscake](#)

Plan zajęć 2: deployment



Środowiska



Deployment

- Konfiguracja środowiska
- Przygotowania i rollout środowiska
 - infrastructure as-a-code (ansible / salt / chef)
 - Kubernetes and Terraform
- Zmiany w danych w produkcji
- ...

CONTINUOUS DEPLOYMENT

- System Tesing
- Load and Performance testing
- User Acceptance Tests
- Smoke tests
- Monitoring

ROLLING UPDATES

- jeden za drugim...

[presentation strona 35](#)

CANARY

- pierwsza grupa serwerów z nowych oprogramowaniem
- jeśli wystąpią błędy zatrzymujemy release

TOGGLING FEATURES

- każda funkcjonalność ma flagę
- flagi możemy szybko ustawiać albo usuwać

HIDDEN FEATURES

- nowe funkcjonalności nie widoczne dla użytkownika
- działające w tle
- weryfikuje nowego kodu w warunkach produkcyjnych

METRICS

- UP/Down
- User history in app (CR1, CR2, CR3, CR4)
- Analytics and Business KPIs, e.g., revenue
- A/B Testing
- Performance
- Request rate (count), Error rate (count), and Duration of requests. (**RED**)

OBSERVABILITY: PROMETHEUS



OBSERVABILITY: GRAFANA



JEŚLI METRYKI TO ALERTY

```
ALERT ProductionAppServiceInstanceDown
  IF up { environment = "production", app =~ ".+" } == 0
  FOR 4m
  ANNOTATIONS {
    summary = "Instance of {{$labels.app}} is down",
    description = " Instance {{$labels.instance}} of app
  }
```

JEŚLI METRYKI TO ALERTY

- pagerduty
- opsgenie

OBSERVABILITY

- logging
- tracing

PYTANIA?

PODSUMOWANIE

DZIĘKUJE ZA UWAGĘ