

Co każdy programista powinien wiedzieć budując serwis dla Kubernetesa?

Wojciech Barczyński
Director of Infrastructure

Codility_

High delivery performance

- Lead Time
- Deployment frequency
- Mean time to Recovery
- Change Fail Percent

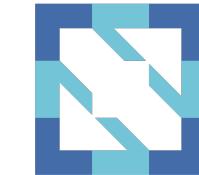


High delivery performance

- Sukces komercyjny
- ...
- Jak i zadowolenie pracowników



...



CLOUD NATIVE
COMPUTING FOUNDATION



argo

...

Nasz Cel

- **Lead time** – od otworzenia PR/ticketa do deploymentu
- **Deployment frequency** – X dziennie

Black (Blue) Box

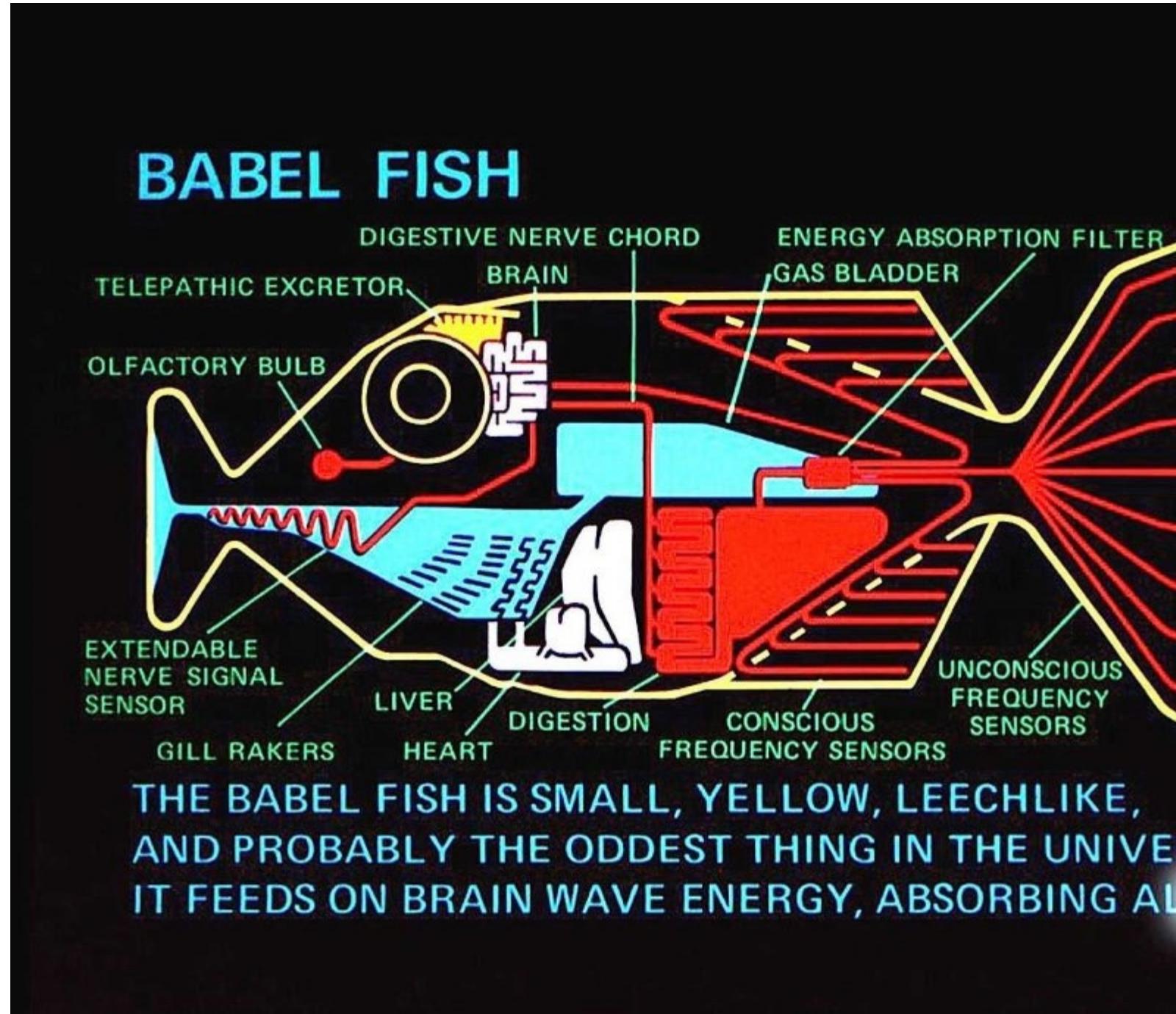
Infrastruktura
(prawie) niewidoczna.

Duża kontrola w
rękach dewelopera.



[https://en.wikipedia.org/wiki/File:Dr_Who_\(316350537\).jpg](https://en.wikipedia.org/wiki/File:Dr_Who_(316350537).jpg)

Wspólny język
dla
deploymentu
aplikacji



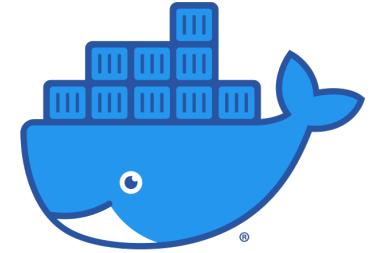
The way of Kubernetes



https://www.flickr.com/photos/bruno_brujah/

- W iteracjach
- Learn-as-you-go

Docker



1. Skorzystaj z linterów dla Dockera, np., hadolint
2. Dbaj o rozmiar obrazu: .dockerignore, budowanie multi-stage

```
RUN apt-get -y update \
    && apt-get install -y vim --no-install-recommended \
    && rm -rf /var/lib/apt/lists/*
```

3. Nie zostawiaj sekretów w Dockerze, wykorzystaj --build-arg i montowanie sekretów

Jak generować manifest Kubernetesa?

V1:

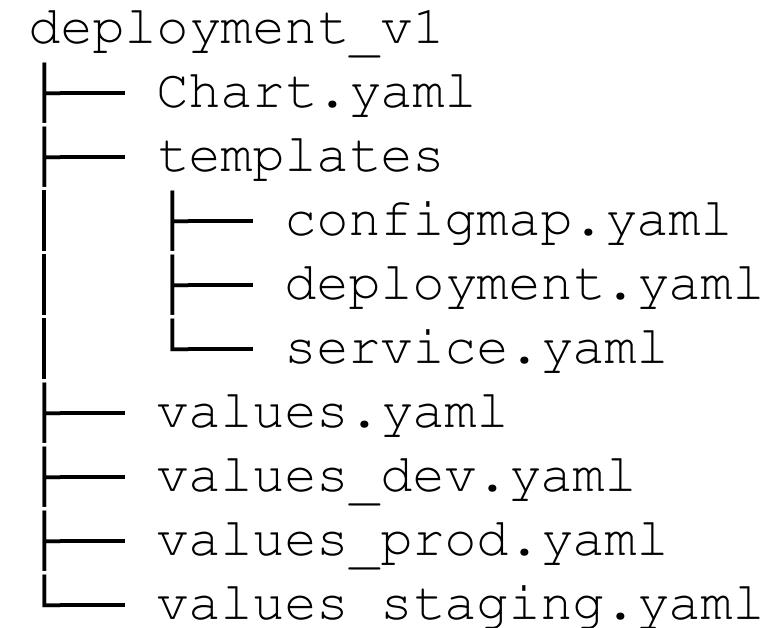
1. Helm jako proste narzędzie do template-owania
2. Aplikacja Kubernetes Deployment i Service z pomocą pipeline-u
3. Secrety manualnie oraz niektóre ConfigMapy

```
deployment_v1
├── Chart.yaml
└── templates
    ├── configmap.yaml
    ├── deployment.yaml
    └── service.yaml
├── values.yaml
├── values_dev.yaml
├── values_prod.yaml
└── values_staging.yaml
```

Jak generować manifest Kubernetes?

V1:

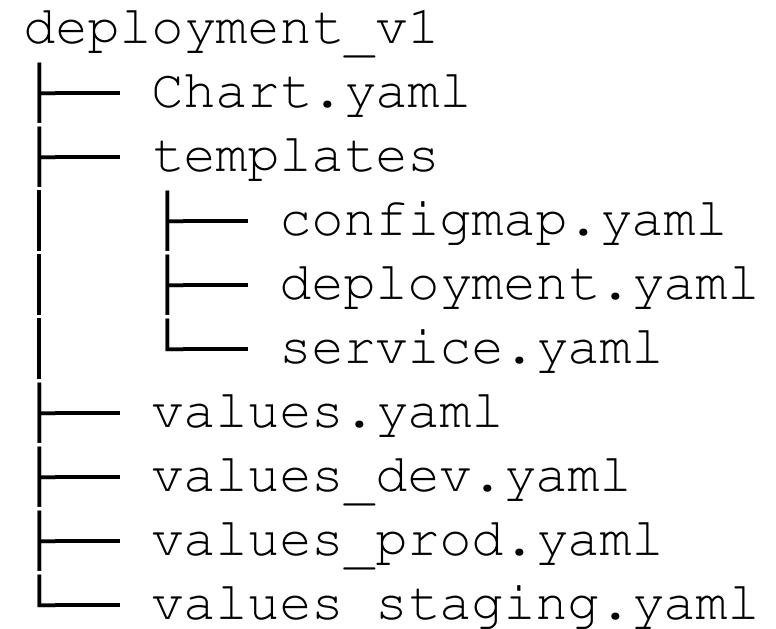
1. Minimalny template YAGNI
2. Generowany manifest K8S aplikowany bezpośrednio na klaster
lub
transitive repo



Jak generować manifest Kubernetes?

V2, V3, ... choose your advanture:

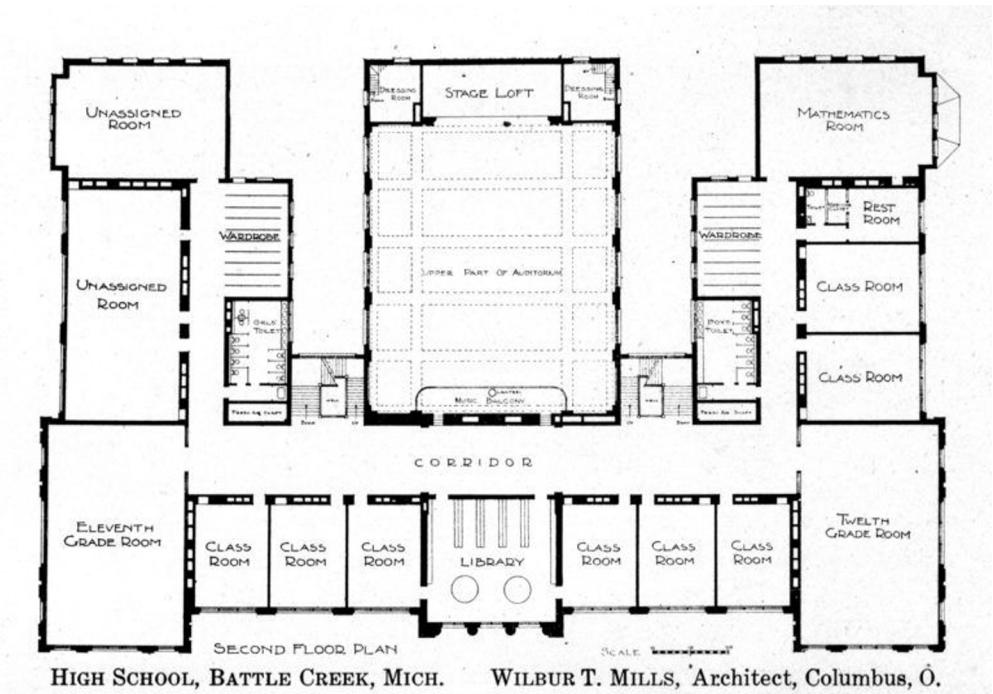
- Zarządzanie sekretami
- ArgoCD
- Strategie deploymentu



Konwencje a potem narzędzia

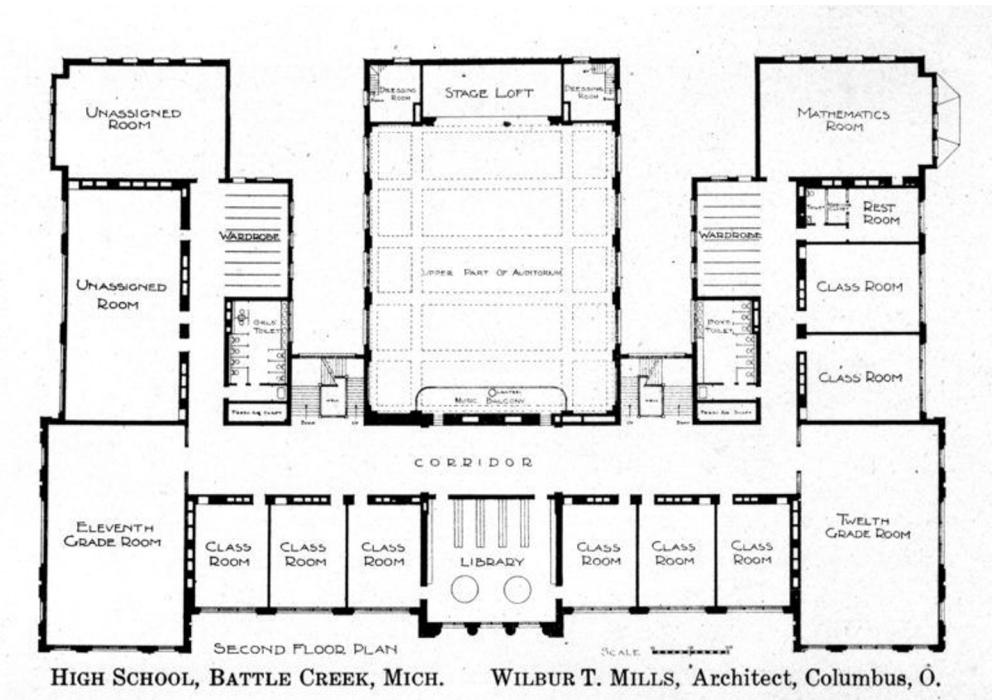
Nowe repozytorium?

- Copy&paste:
Makefile, .gitlab-ci.yml, README.md,
Dockerfile, helm, ...
- Zmień kilka linii
- Ready & Go

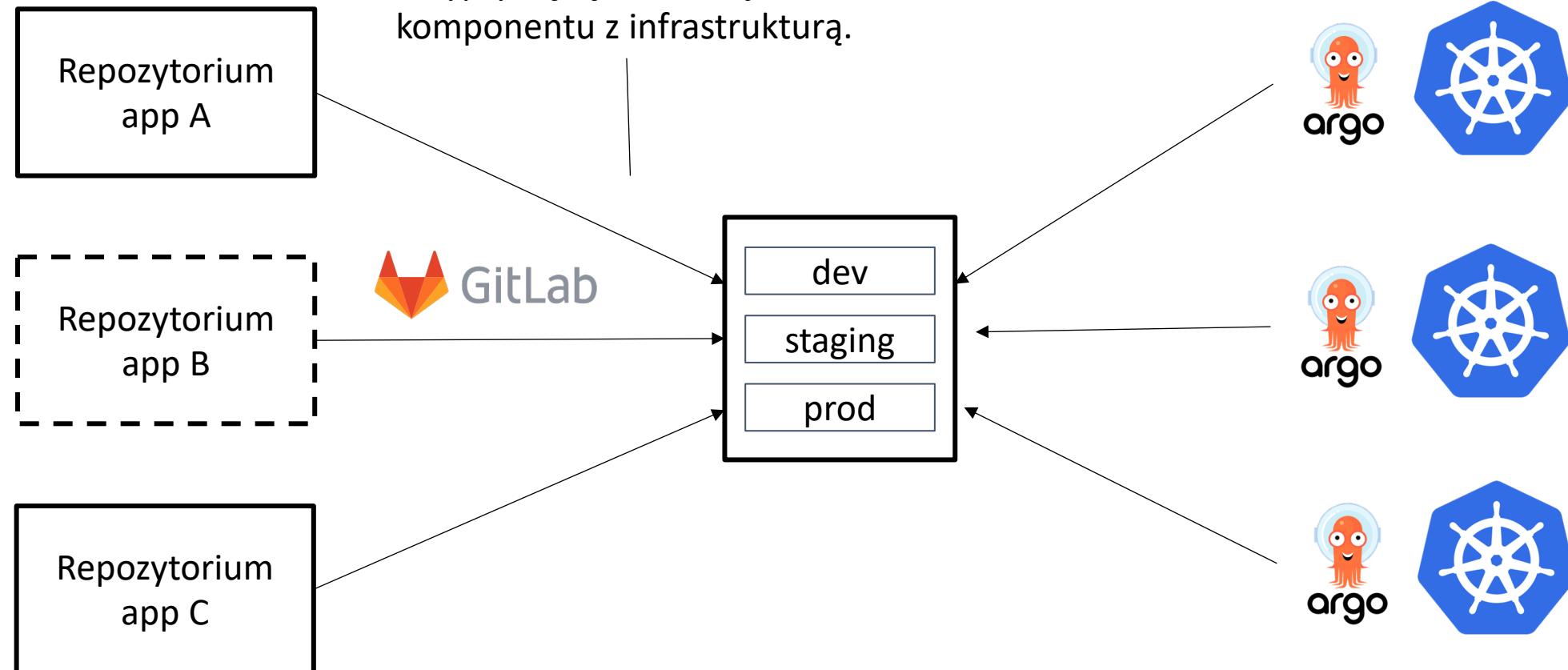


Konwencje a potem narzędzia

- Nie wprowadzamy powiązań między komponentami
- Na początku jedyne wspólne części będą skrypty dostarczających absolutnych podstaw
- Pozwalamy na pełną swobodę zmian



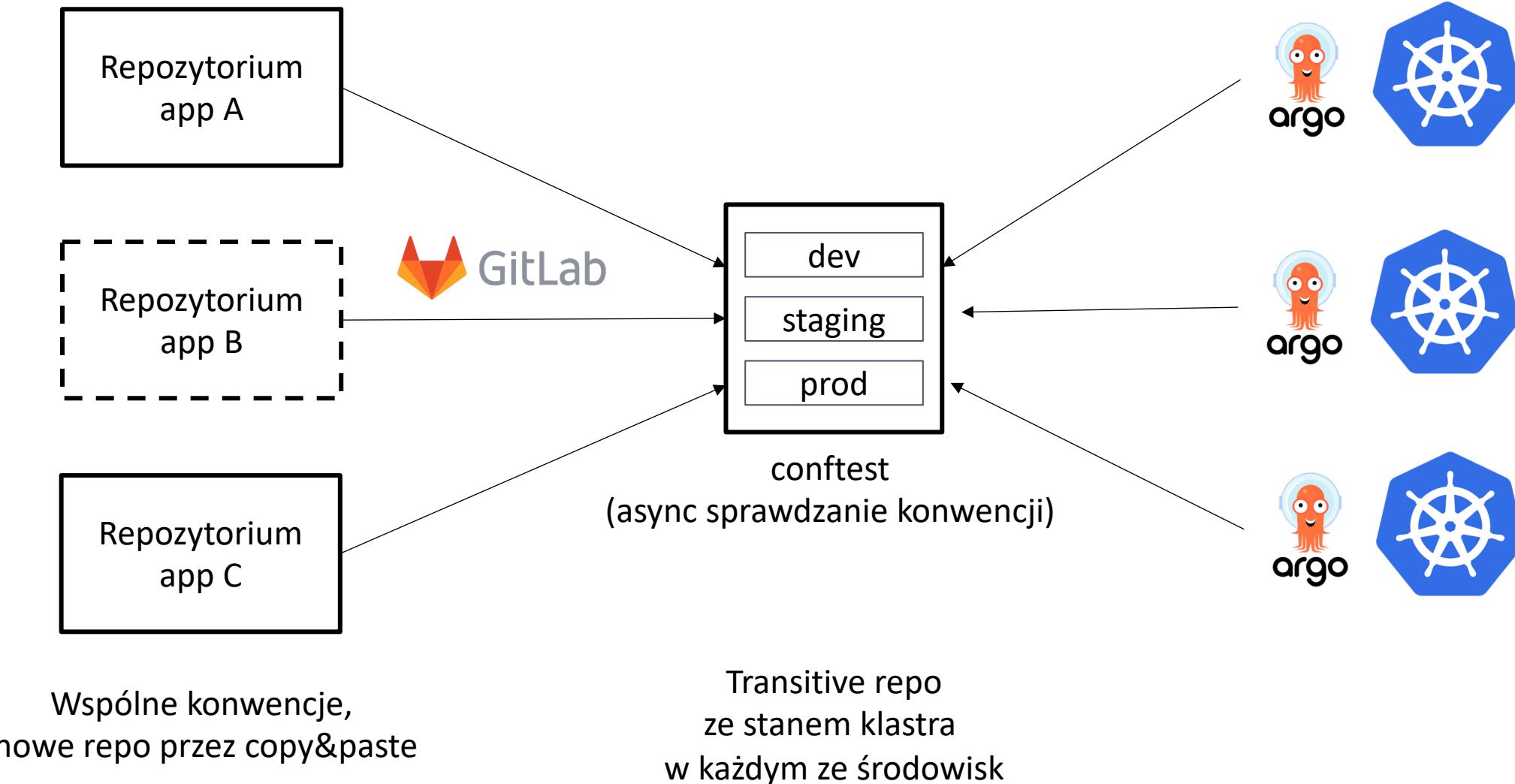
V1.5 Codility



Wspólne konwencje,
nowe repo przez copy&paste

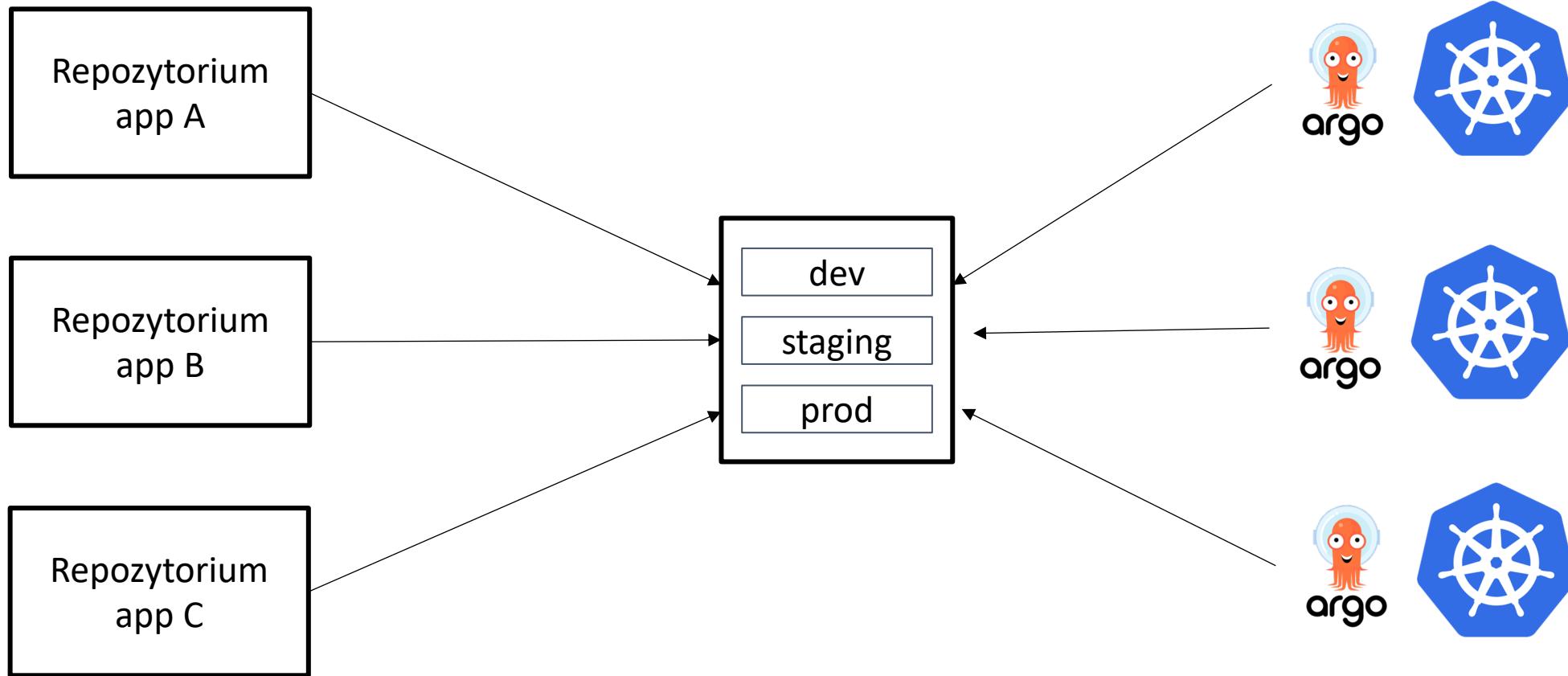
Transitive repo
ze stanem klastra
w każdym ze środowisk

V1.5 Codility

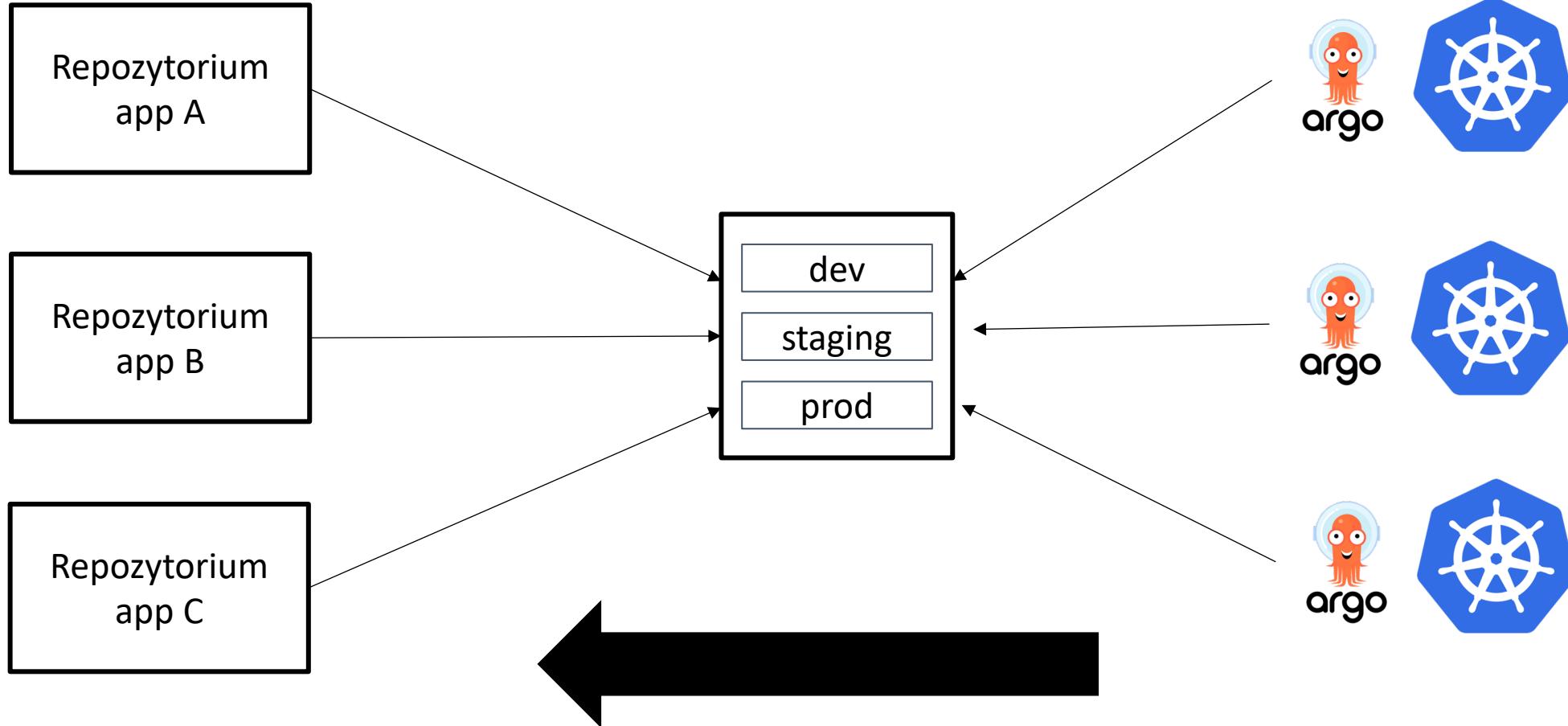
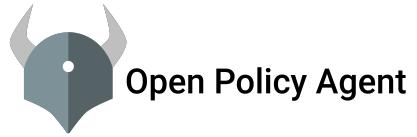


V1.7 Codility

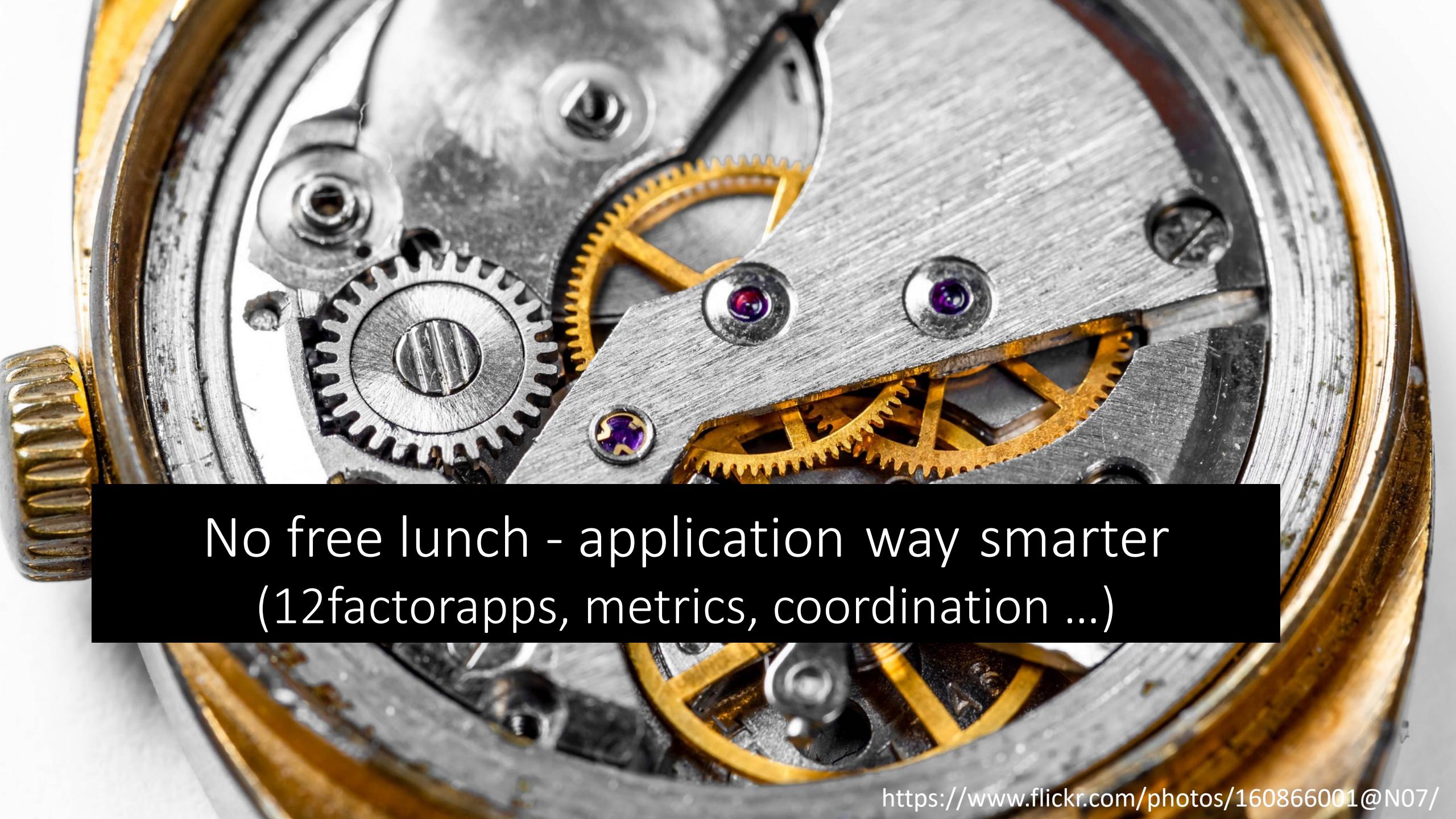
Środowiska dev na żądanie



V1.7+ Codility – więcej self service



Potem: kolejna iteracja analizy czy coś warto wyciągnąć jako wspólny skrypt.



No free lunch - application way smarter
(12factorapps, metrics, coordination ...)



12 factor apps

1. Wybieraj konfigurację przez zmienne środowiskowe
2. Loguj na stdout
(ustrukturyzowane, najlepiej json)
3. Skalowanie aplikacji przez dodanie nowego procesu
4. MTTR versus MTTF,
aka szybko wstajesz i się składasz (fail fast)
5. Zakładamy, że call do innego serwisu może timeout



12 factor apps

4. MTTR versus MTTF,
aka szybko wstajesz i się składasz (fail fast)
5. Zakładamy, że call do innego serwisu może timeout
6. Koordynacja jest po naszej stronie, np., wybór lidera czy właściwa kolejność wykonania operacji.

Observability

	Metrics	Logging	Tracing
CapEx	Medium	Low	High
OpEx	Low	High	Medium
Reaction	High	Medium	Low
Investigation	Low	Medium	High

Observability - monitoring

Serwuj dane monitoringowe na innym niż twoja aplikacja.

```
apiVersion: v1
kind: Service
metadata:
  name: my-app
  annotations:
    prometheus.io/scrape: 'true'
    prometheus.io/port:    '9090'
    prometheus.io/path:   'metrics/'
spec:
  selector:
    app: my-app
  ports:
    - port: 8080
      target_port: 8080
```



Prometheus

<https://prometheus.io/docs/practices/naming/>

Observability - monitoring

Dwa podejścia:

- RED
 - Rate Error Duration
- USE
 - Utilization Saturation and Errors

```
apiVersion: v1
kind: Service
metadata:
  name: my-app
  annotations:
    prometheus.io/scrape: 'true'
    prometheus.io/port: '9090'
    prometheus.io/path: 'metrics/'
spec:
  selector:
    app: my-app
  ports:
    - port: 8080
      target_port: 8080
```



Prometheus

Observability - logging

- Zmienna środowiskowa w deploymencie `LOG_LEVEL`
- Logi, najlepiej, JSON
- Wiele bibliotek, np., dla Golanga: [zap](#) lub [logrus](#)



Współpraca z K8S

- **Readiness probe**

- czy jestem gotowy?
 - czy moje powiązane serwisy odpowiadają?

- **Liveness probe**

- czy żyje?

```
readinessProbe:
```

```
  httpGet:
```

```
    path: /ready
```

```
    port: 8090
```

```
livenessProbe:
```

```
  httpGet:
```

```
    path: /healthz
```

```
    port: 8090
```

```
  initialDelaySeconds: 5
```

```
  periodSeconds: 5
```

Współpraca z K8S

- **Readiness probe**

- czy jestem gotowy?
- czy moje powiązane serwisy odpowiadają?

- **Liveness probe**

- czy żyje?

```
readinessProbe:
```

```
    httpGet:
```

```
        path: /ready
```

```
        port: 8090
```

```
livenessProbe:
```

```
    httpGet:
```

```
        path: /healthz
```

```
        port: 8090
```

```
    initialDelaySeconds: 5
```

```
    periodSeconds: 5
```

Nie musisz mieć o dnia zero na Kubernetesie.

Możesz po prostu zacząć od liveness.

Współpraca z K8S

Nieudane wywołanie

- **Readiness probe**
 - Kubernetes nie przekazuje requestów
(usuwa poda z endpointów dla serwisu)
- **Liveness probe**
 - kubectl ubije twój kontener i go zrestartuje

Zamykanie aplikacji przez K8S

1. Kubernetes wysyła SIGTERM
2. App powinna go obsłużyć
i od razu ustawić readiness z błędny kodem
3. Kubernetes przestaje przesyłać żądania
4. App powinna się zamknąć gracefully

```
func (s *Service) prepareShutdown(h Server) {
    signal.Notify(s.Stop, os.Interrupt, syscall.SIGTERM)
    <-s.Stop
    s.StatusNotReady()
    shutdown(h)
}
```

QoS – dlaczego k8s migruje mojego poda?

Klasy Qos:

- Guaranteed
- Burstable
- Best Effort

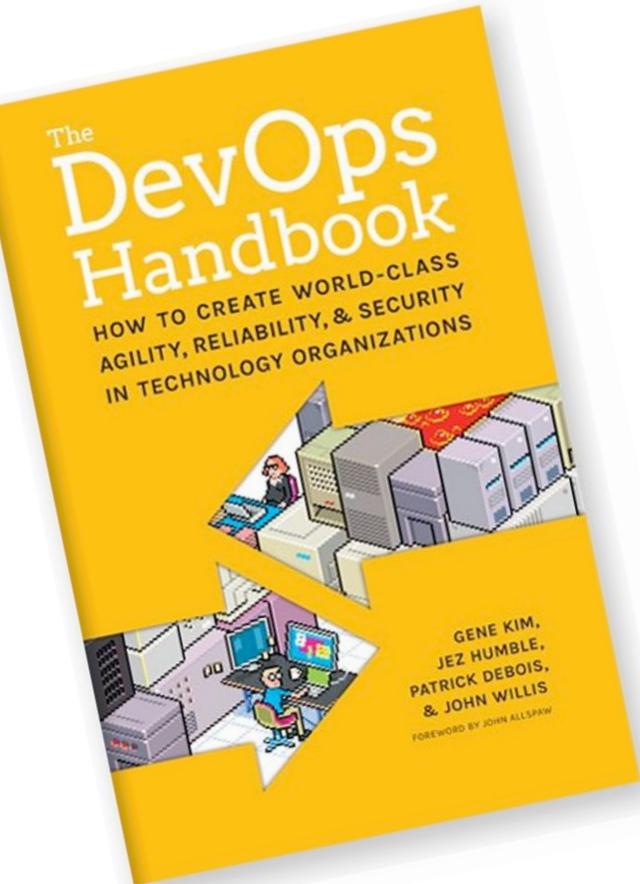
```
apiVersion: v1
kind: Pod
metadata:
  name: qos-demo
  namespace: qos-example
spec:
  containers:
  - name: qos-demo-ctr
    image: nginx
    resources:
      limits:
        memory: "200Mi"
        cpu: "700m"
      requests:
        memory: "200Mi"
        cpu: "700m"
```

Smoke test & Readiness/Liveness probes

Czy mój komponent działa?



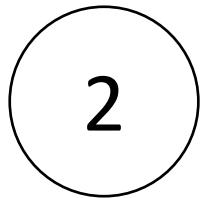
Warto przeczytać



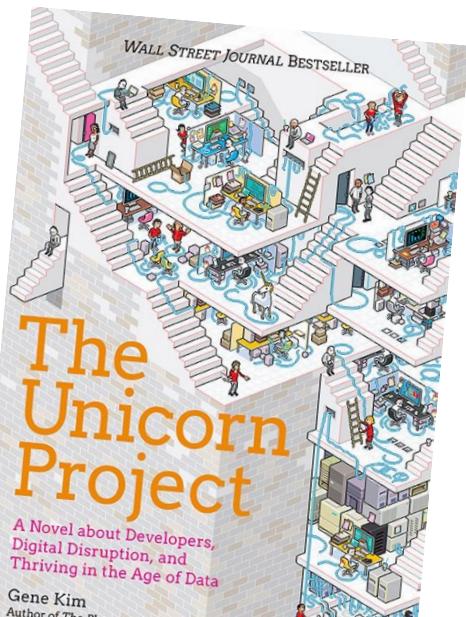
1

Konkretnie, zrób A, a potem B,
ponieważ...

(DevOps oznacza kulturę, a nie admina ☺)



Pierwsze 4 rozdziały najwięcej
wartości, szybko się czyta,
reszta jak macie czas.



3

Przypowieść,
nie jest to prosto
z mostu jak 1



Rabbit holes everywhere...

Podejście:

- Pierwsza iteracja, decyzja, dostarczamy
- Chcemy przejść jak najszybciej na cykl Patch Patch Patch

Alternatywne spojrzenie:

- Lean v1/v2*

* <https://katemats.com/blog/lean-software-development-build-v1s-and-v2s>



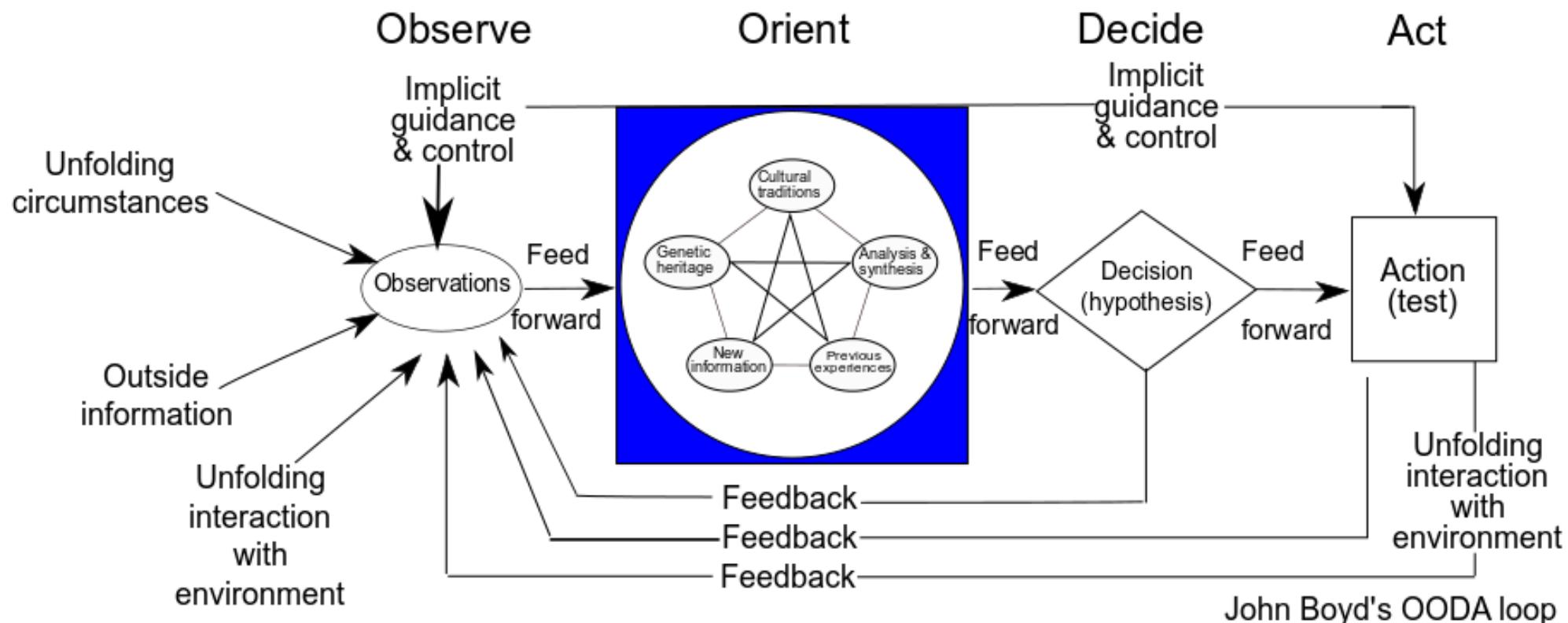
Thank you. Questions?

Backup

Migracja na Kubernetes / CloudNative

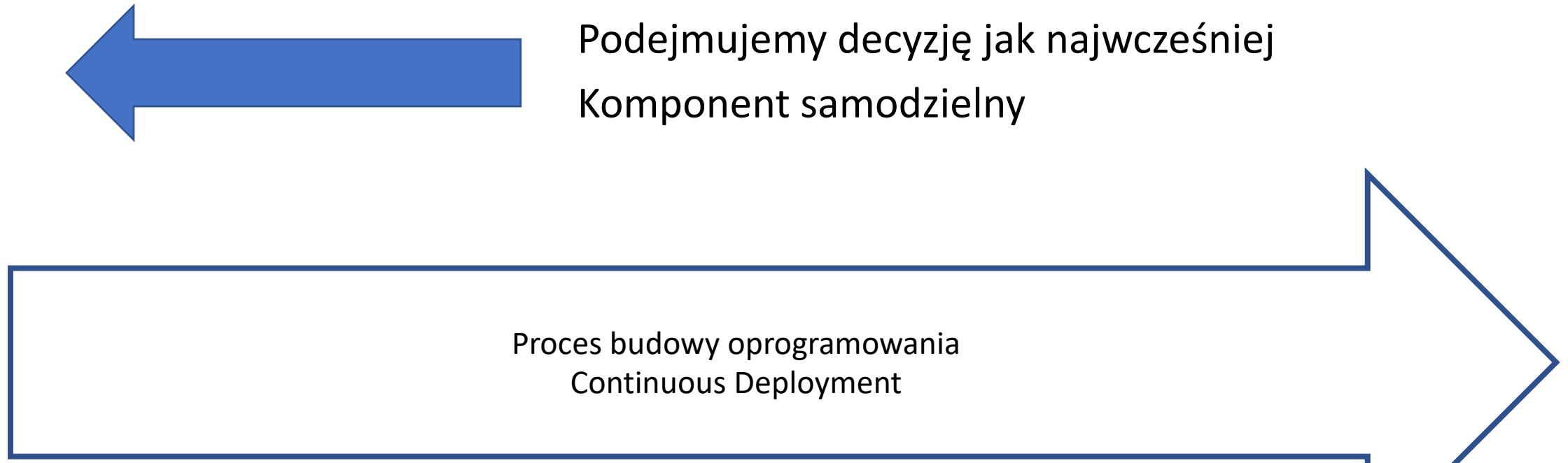
- Warto połączyć z projektem biznesowym
- Pamiętać o celu:
 - częste wydania i krótki cykl dla budowy funkcjonalności
 - autonomia zespołów, self-service

OODA



<https://upload.wikimedia.org/wikipedia/commons/3/3a/ODA.Boyd.svg>

Kubernetes



Jak największej szczegółów aplikacji jest przechowywana w repozytorium gita.

W jaki sposób?

