

Lambda Functions i GitOps

jak szybko budować nowoczesne aplikacje

Marcin Witalis & Filip Stróżik, SAP
01 Grudzień, 2020

PUBLIC



Agenda



Hack Your Career

HA YO
CKUR
CAREER



<https://hackyourcareer.github.io/>
<https://www.facebook.com/Hack.your.Career/>

SAP Labs Poland

Różnorodność domen

E-commerce, social commerce, cloud platform, open source,...

Development: Go, Java, Angular, Cloud Native solutions, ...

Możliwość nauki i rozwoju:

Platformy do edukacji on-line, certyfikacje, łatwy dostęp do ekspertów na całym świecie

> 350 pracowników

Różnorodność specjalizacji
Product teams, project teams, innovation projects teams, support teams

Jedno z 21 centrów SAP's Labs Network



Filíp



W SAP Gliwice pracuję od 2019 roku jako Software Developer Intern, gdzie pomagam rozwijać ideę Serverless oraz GitOps, w projekcie Kyma.
Na co dzień pracuję z takimi technologiami jak Go czy Kubernetes.
Jestem fanem rozwiązań chmurowych, yerba mate i dobrej muzyki.



Dołączyłem do SAP Gliwice w 2019 roku jako Senior Software Developer.
Pracuję przy projekcie Open Source Kyma opartym na platformie Kubernetes i zajmuję się rozwijaniem funkcjonalności serverless.
Mam wieloletnie doświadczenie w tworzeniu oprogramowania opartego na rozproszonej architekturze.
Jestem entuzjastą Linux'a i filozofii Unix oraz Open Source.
Interesuje mnie programowanie funkcyjne, inżynieria oprogramowania, cloud oraz klasyczne gry fabularne (RPG).

Marcin

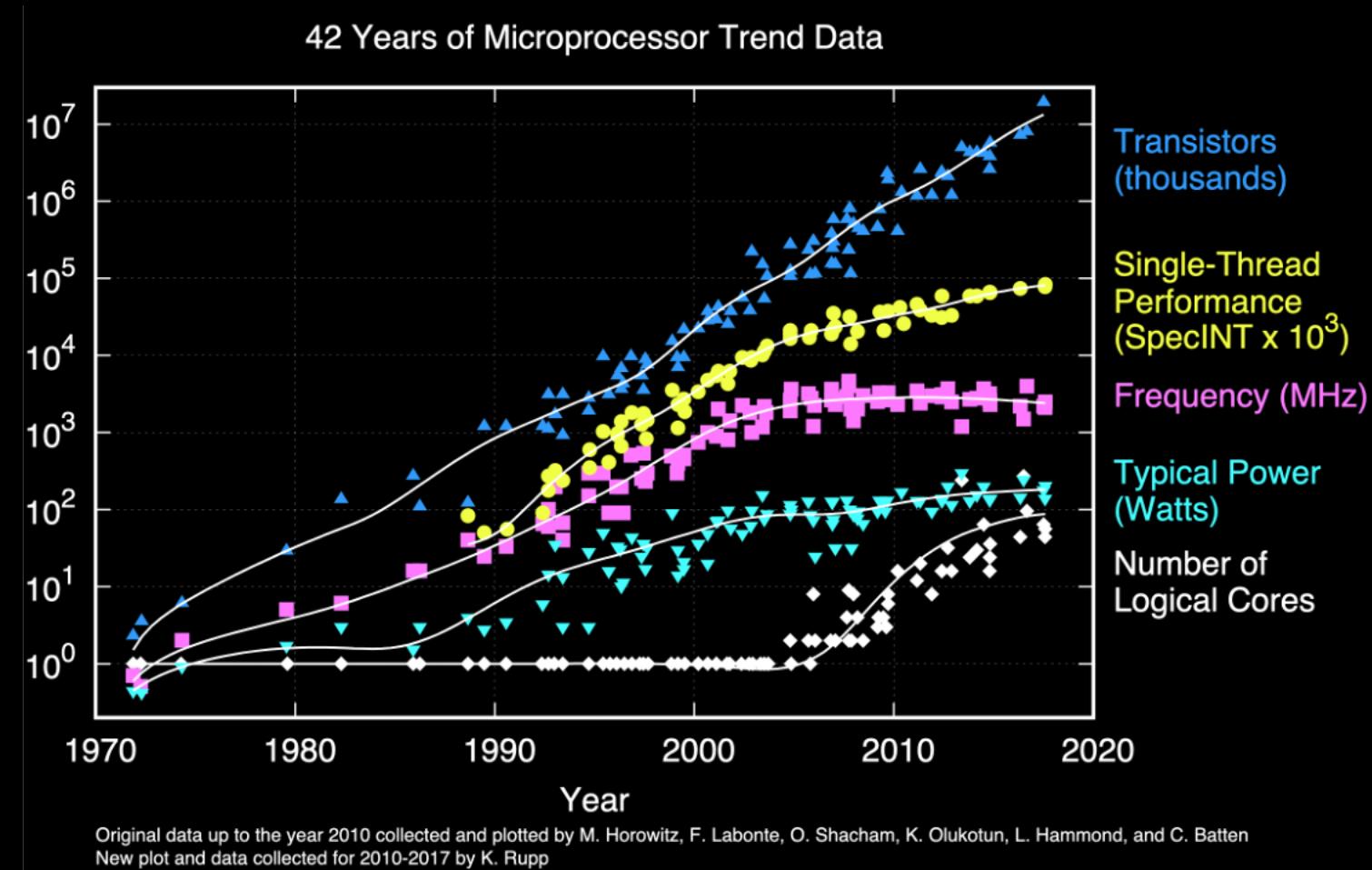
GitOps

#2

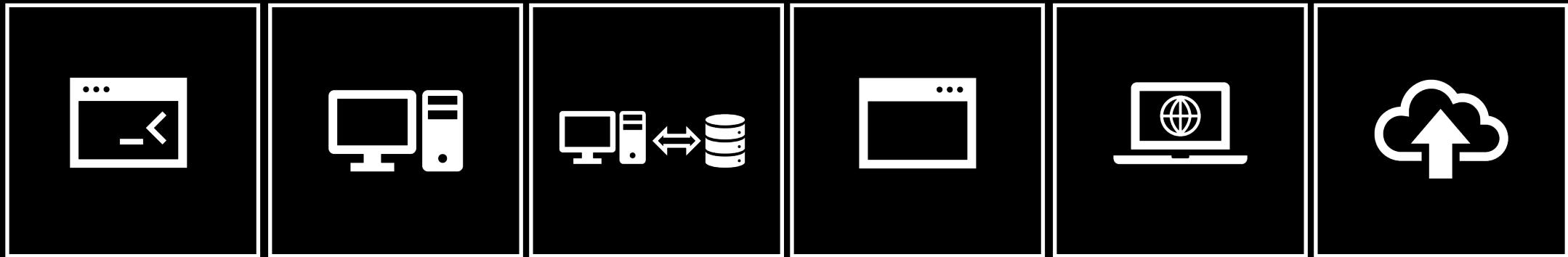
Plan:

1. Ewolucja sprzętu i aplikacji
2. Cloud computing
3. GitOps
4. Demo
5. Podsumowanie

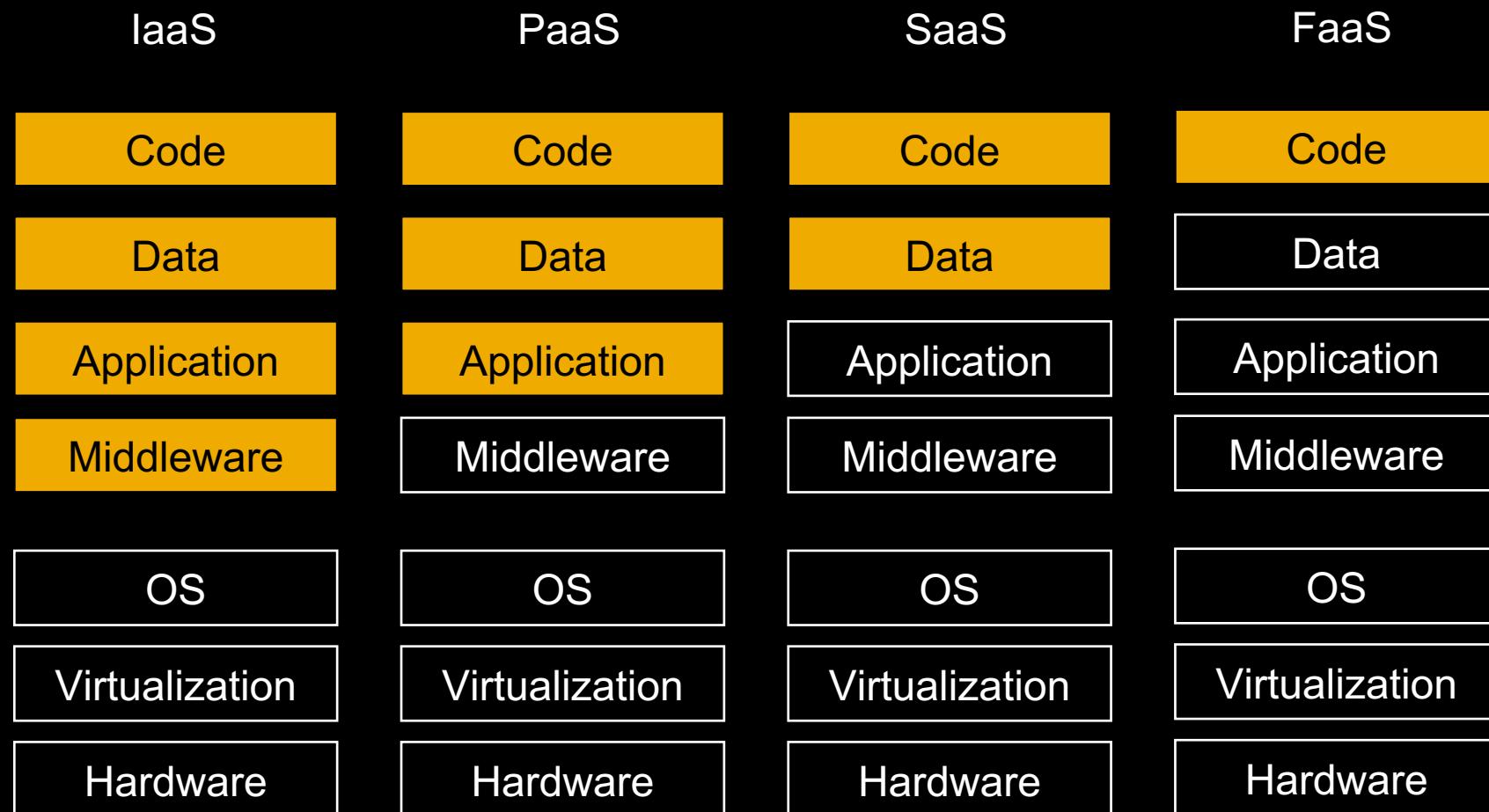
Ewolucja sprzętu



Ewolucja infrastruktury i aplikacji



Warstwy świadczenia usług



Infrastructure as a Service

- Pozwalają na zdalne zarządzanie (tworzenie i modyfikowanie) infrastruktury sieciowej oraz fizycznych zasobów obliczeniowych.
- Usługa wystawia wysokopoziomowe API
- Może mieć deklaratywny i imperatywny charakter
- Wykorzystuje wirtualizację

Code

Data

Application

Middleware

OS

Virtualization

Hardware



Infrastructure as a Service - korzyści

- Skalowalność
- Redukcja kosztów i czasu.
- Delegowanie podstawowych czynności administracyjnych na dostawcę
- Bezpieczeństwo i stabilność

Code

Data

Application

Middleware

OS

Virtualization

Hardware

Infrastructure as a code

- Brak manualnych operacji
- Infrastruktura opisywana w kodzie w sposób deklaratywny lub imperatywny
- Umożliwia automatyzację
- Zwiększa produktywność
- Eliminuje możliwość powstania tzw. “Snowflake” servers





imgflip.com

Platform as a Service

- Agreguje infrastrukturę oraz dodatkowe narzędzia i usługi (BI, DB etc.)
- Usługobiorca zajmuje się wyłącznie rozwijanymi przez siebie aplikacjami i serwisami, usługodawca dostarcza całą resztą
- Użytkownikami są zazwyczaj developerzy

Code

Data

Application

Middleware

OS

Virtualization

Hardware



Platform as a Service - korzyści

- Zmniejsza złożoność projektów
- Zapewnia szybkość developowania i testowania
- Zapewnia wsparcie dla wielu platform
- Brak konieczności instalacji i konfiguracji

Code

Data

Application

Middleware

OS

Virtualization

Hardware

DevOps

- Devops = Software Development + IT Operations
- Zespół praktyk pozwalających na jak najszybsze dostarczenie nowych funkcjonalności przy jednoczesnym zachowaniu najwyższych norm jakości
- Nie jest rolą a kulturą
- Automatyzacja
- Wersjonowanie konfiguracji i aplikacji
- CI/CD

Kubernetes

- “Kubernetes to **przenośna**, rozszerzalna platforma oprogramowania **open-source** służąca do zarządzania zadaniami i serwisami uruchamianymi w kontenerach, która umożliwia **deklaratywną konfigurację i automatyzację**”
- Standaryzacja
- Kubernetes nie jest tradycyjnym, zawierającym wszystko systemem PaaS (*Platform as a Service*)
- Środowiskiem pozwalającym uruchomić aplikacje



The road so far

- Globalne usługi, które możemy konsumować lub dostarczać
- Wysoka skalowalność
- Przenaszalność
- Time-to-market
- Deklaratywny charakter
- Automatyzacja

Wszystko jest kodem

“I don't even see the code.

All I see is blonde, brunette, redhead.

Hey uh, you want a drink?”

Cypher - Matrix

VCS jest
jedynym
źródłem
prawdy

GitOps – początek legendy

“calling all **kubernetes fans** - super fast CICD and all the **gitops** goodness from **kubecon** - packed into one little blog post with props going out to [...]”

Alexis Richardson

“GitOps: versioned CI/CD on top of declarative infrastructure. Stop scripting and start shipping”

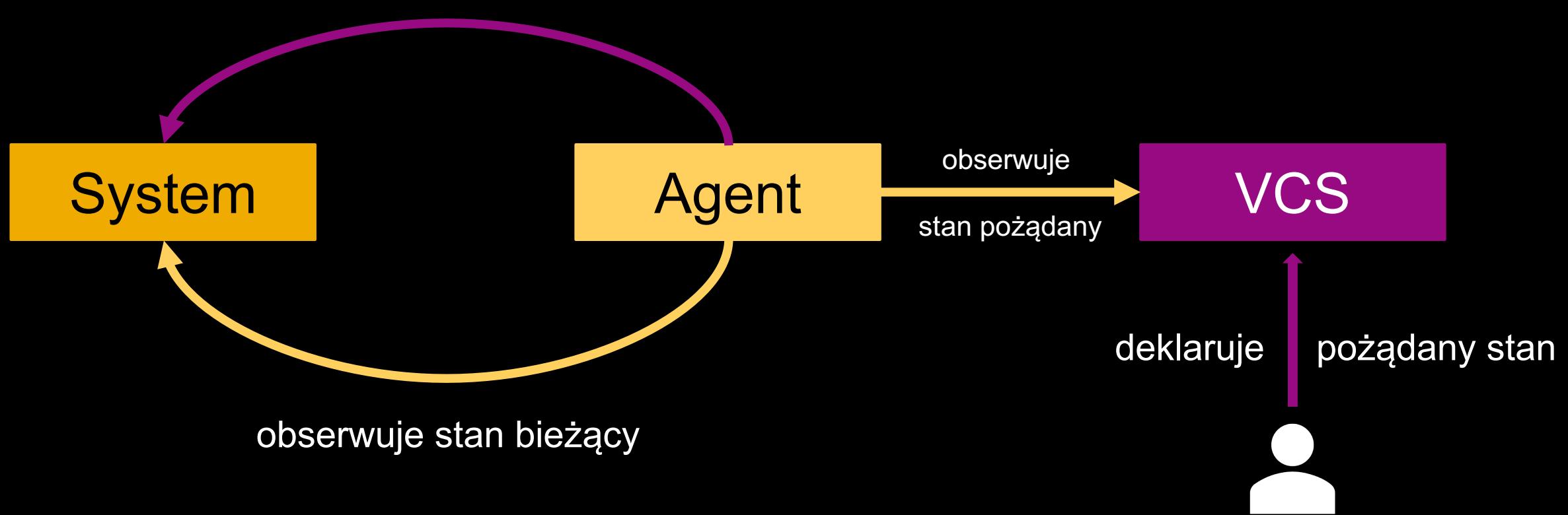
Kelsey Hightower

GitOps to pattern

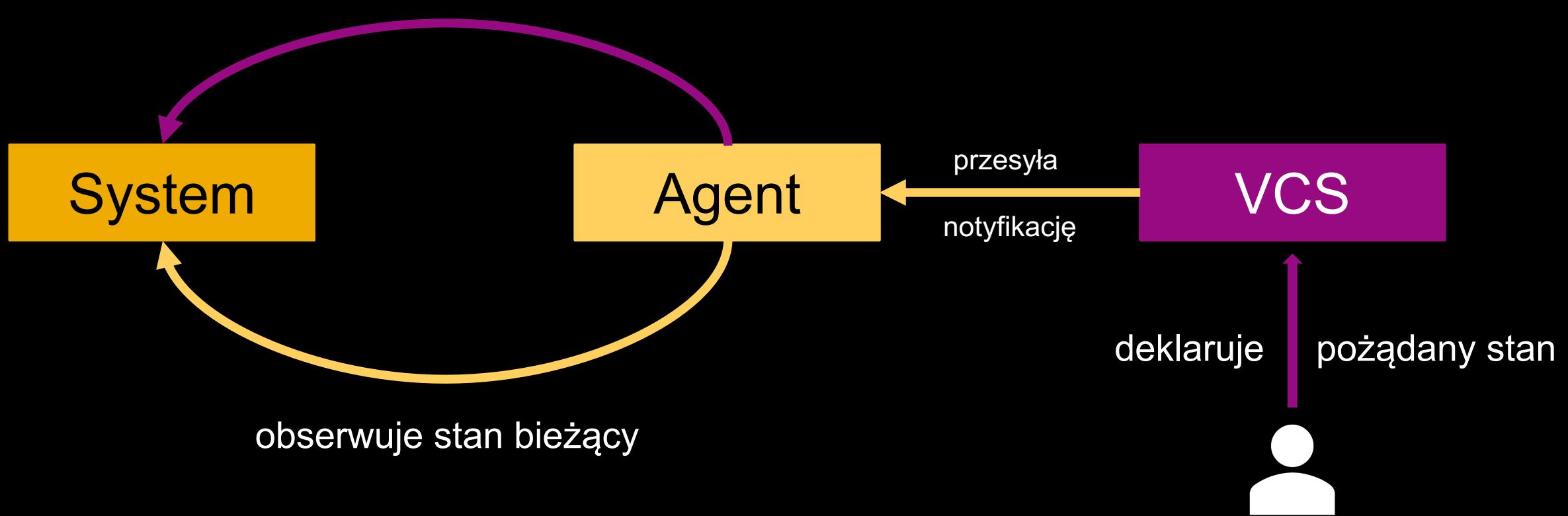
GitOps – building blocks

- System (np: kubernetes)
- Version Control (np: git, mercurial etc.)
- Agent (np: flux)

GitOps



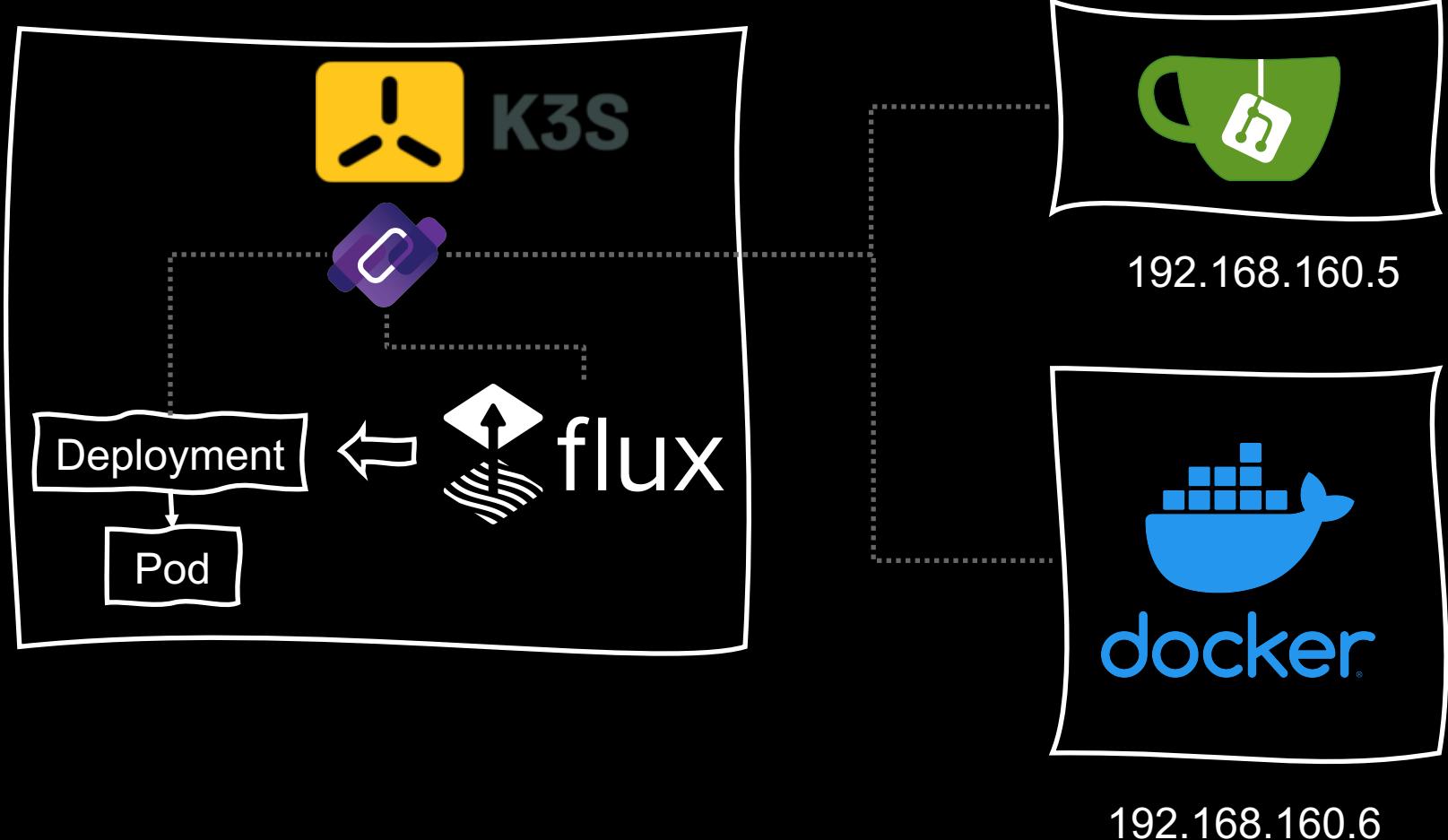
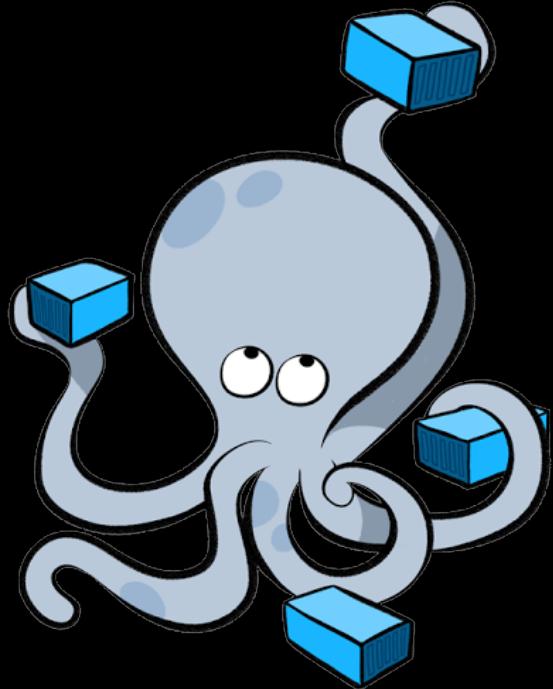
GitOps



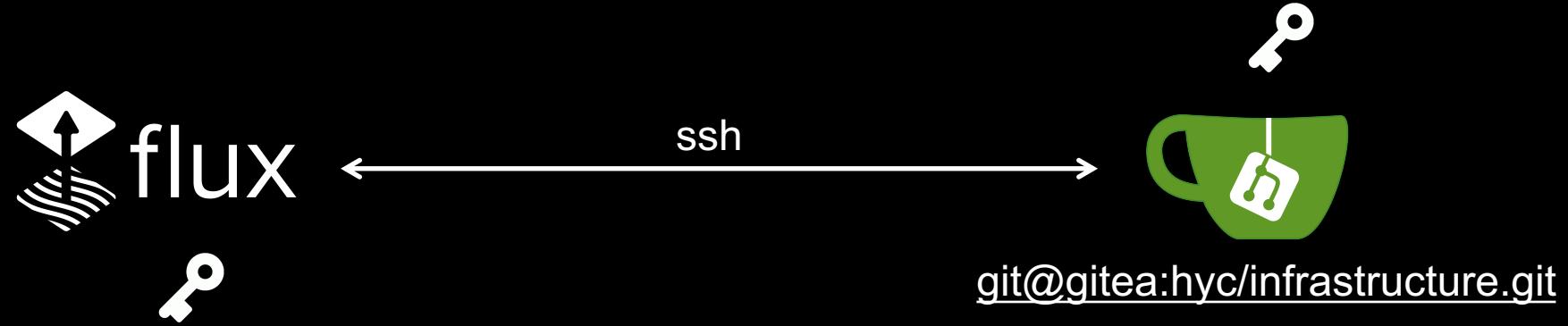
Środowisko developerskie - charakterystyka

- Pełen **GitOps Workflow** przy użyciu Flux'a
- Pełna kontrola
- Reużywalność kodu
- **Batteries included** – brak zewnętrznych dependencji

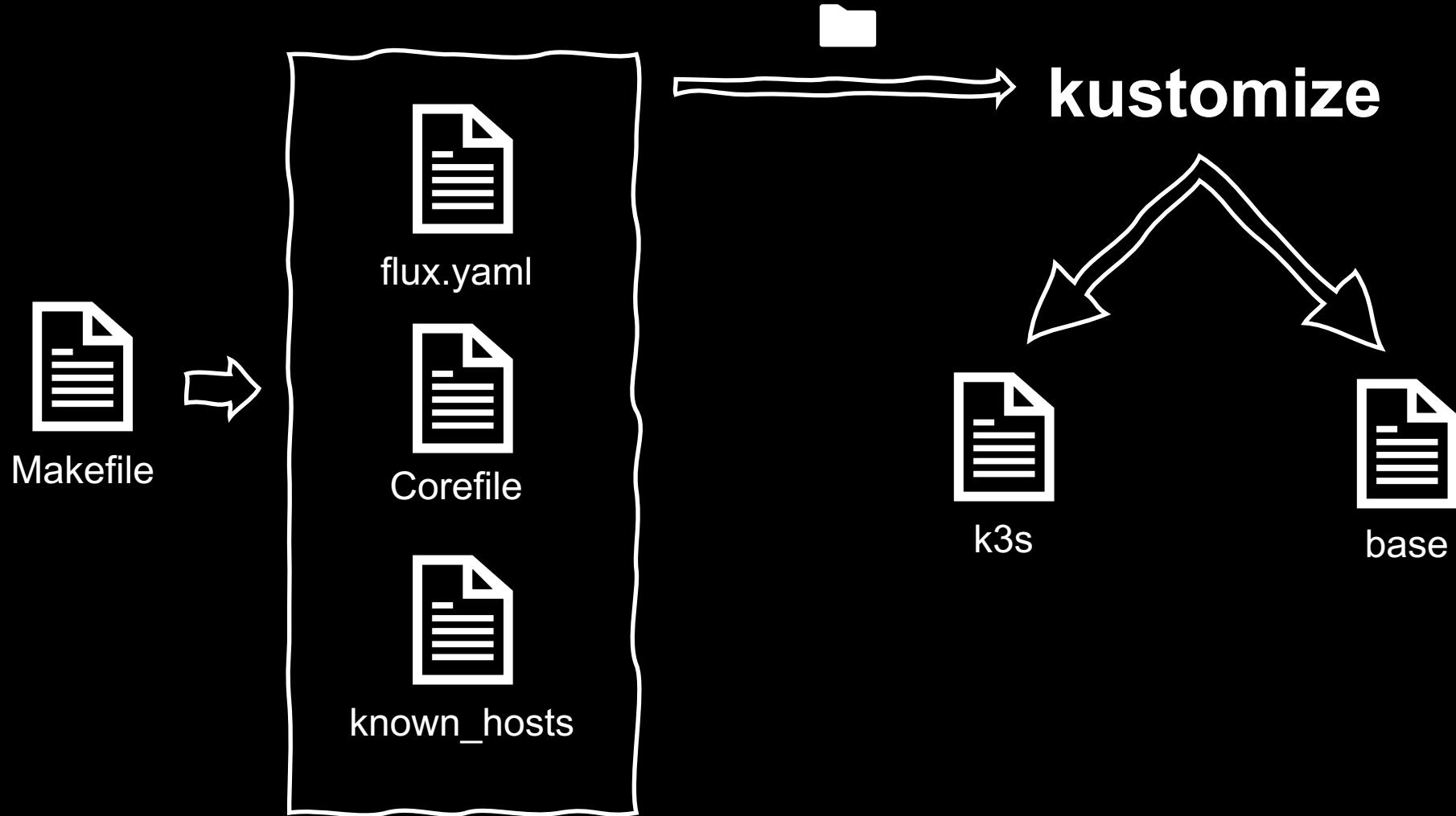
Środowisko developerskie - infrastruktura



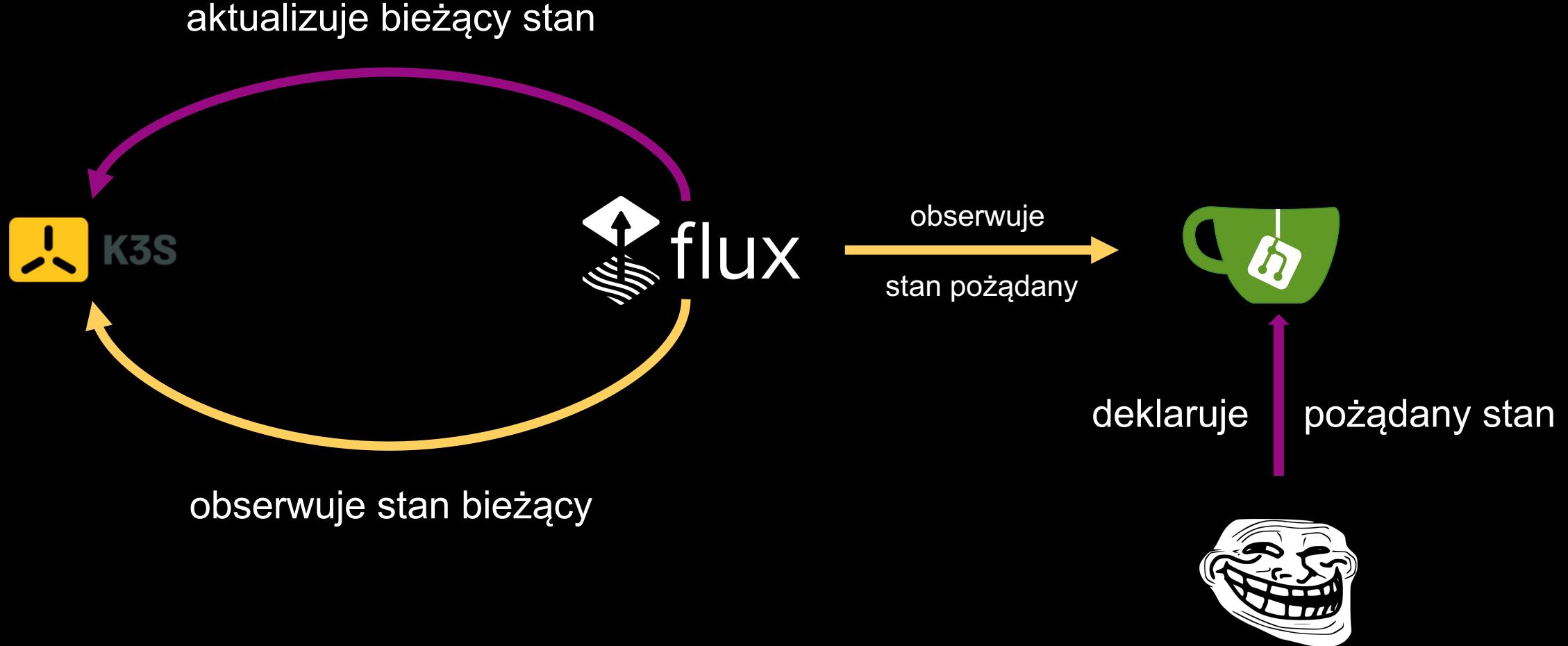
Środowisko developerskie – infrastruktura



Środowisko developerskie - flow



Środowisko developerskie



Instalacja agenta

```
fluxctl install \
--git-user=admin123 \
--git-email=me@test.you \
--git-path=infrastructure \
--git-url=git@gitea:hyc/infrastructure.git
```

Pobieranie klucza publicznego

fluxctl identity --k8s-fwd-ns flux

Wymuszenie synchronizacji

fluxctl sync --k8s-fwd-ns flux

Wylistowanie obiektów obserwowanych przez agenta

fluxctl list-workloads --k8s-fwd-ns flu

GitOps TIME

Flux'a śledzenie registry

```
fluxctl policy \
-w <ns>:<kind>/<name> \
--tag-all='2.*' \
--automate \
--k8s-fwd-ns <ns>
```

GitOps - podsumowanie

- Deklaratywna natura
- Przyjazny developerom
- Zwiększa produktywność
- Zapewnia transparencję (kto, co, kiedy)
- Zwiększa bezpieczeństwo

GitOps - dlaczego

- Skalowalny
- Porządkuje proces wprowadzania zmian w systemie
- Ułatwia śledzenie zmian (kto, kiedy, czemu)
- Przewidywalny
- Prosty w koncepcji

Serverless

#3

Anegdotka

Function as a service (FaaS)

Code

Data

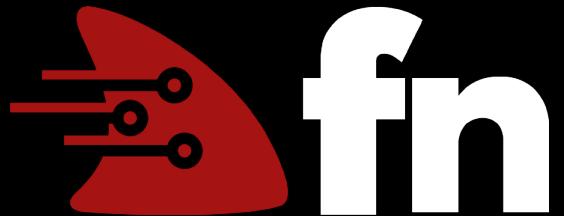
Application

Middleware

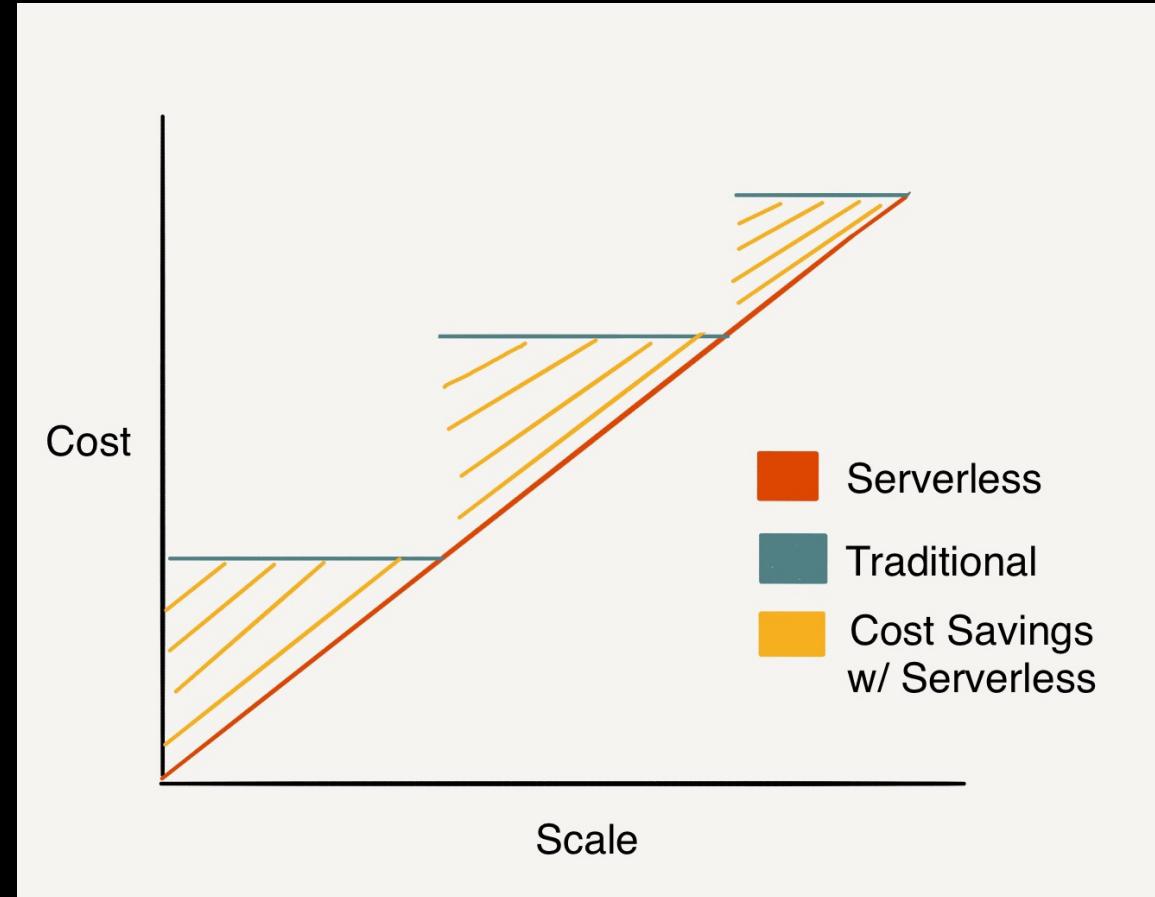
OS

Virtualization

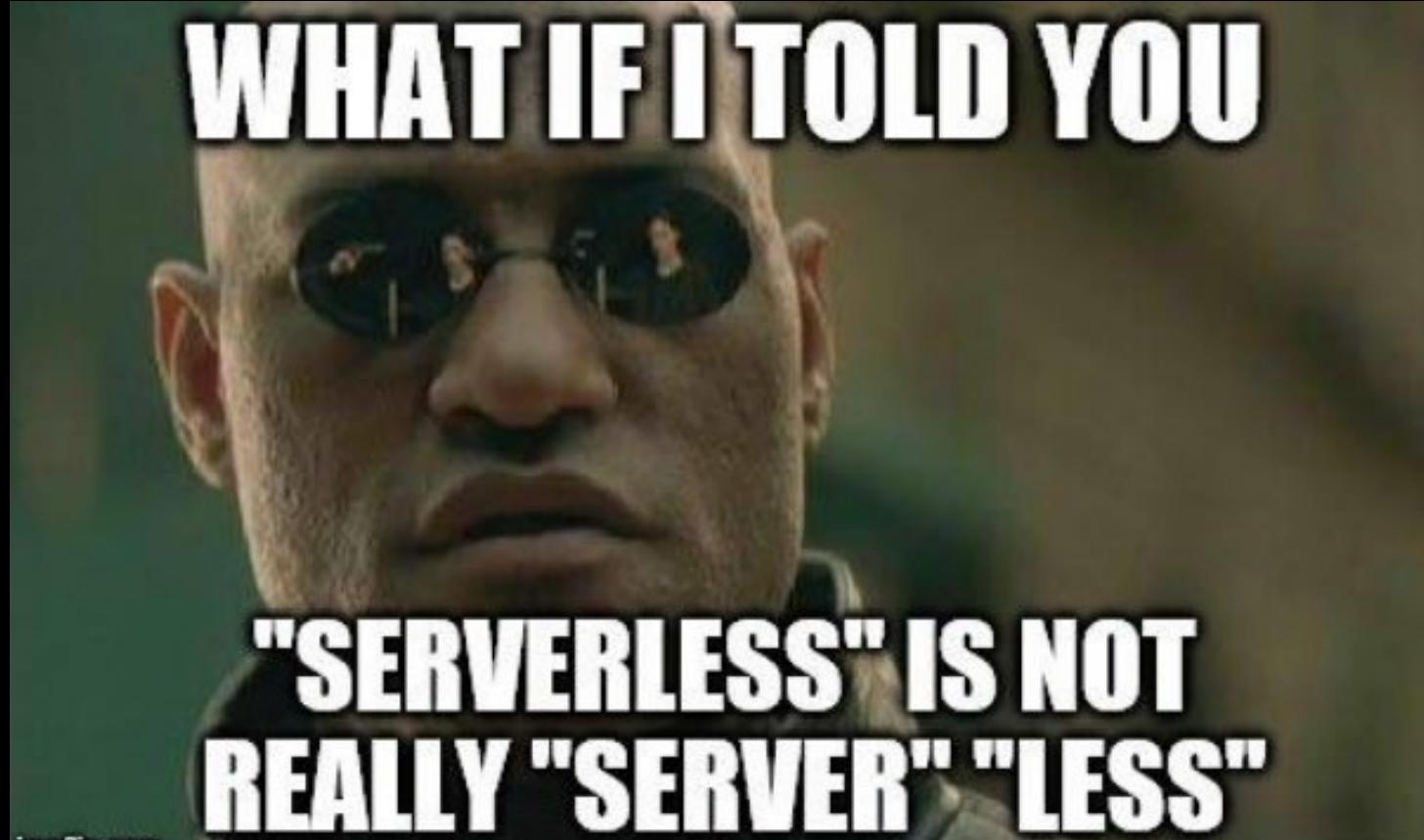
Hardware



Serverless – skalowanie a koszty

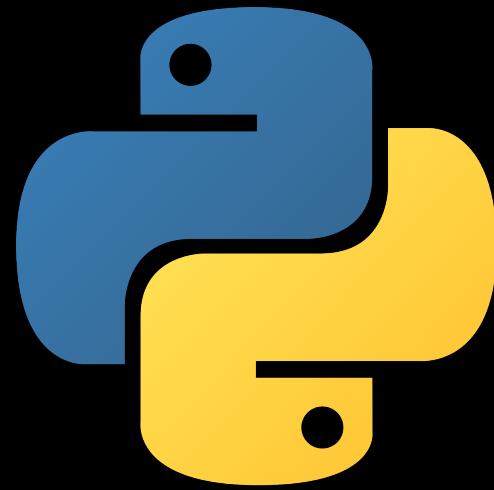


Serverless – zero administracji



Serverless – czym jest?

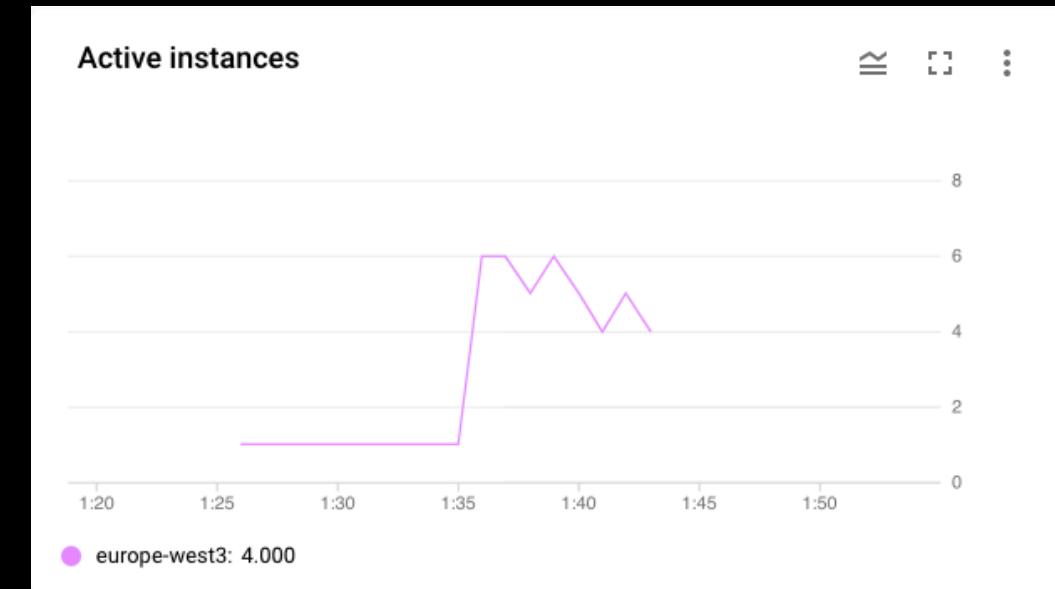
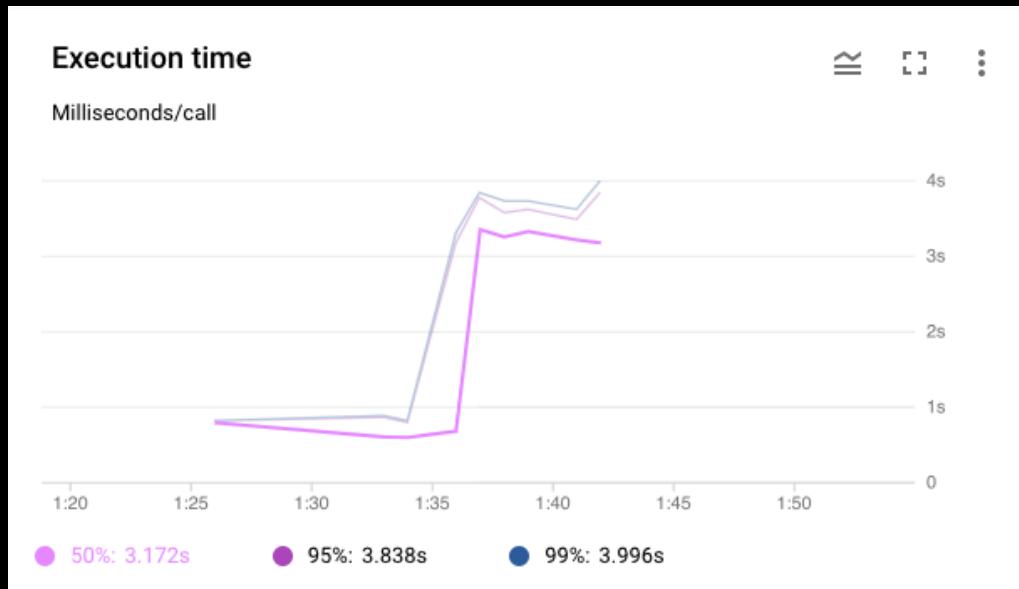
- Sposób traktowania fragmentów kodu, reprezentujących mniejszą funkcjonalność (funkcji), jako serwisów
- Funkcje wywoływane są poprzez otrzymanie zapytania (eventu)
- Skalowanie funkcji następuje w pełni automatycznie



DEMO



Przykładowe metryki



Serverless – kiedy użyć?

- Możliwość wyodrębnienia logiki biznesowej
- Kod jest samowystarczalny
- Duże wahania obciążenia w czasie (Black Friday, dzień singla)

Serverless – kiedy NIE użyć?

- Długi czas konfiguracji serwera przed gotowością na odebranie zapytania
- Konieczność komunikacji z wieloma serwisami
- Funkcjonalność nie daje się wyodrębnić do pojedynczego handlera
- Wymagana jest przenośność kodu
- Automatyzacja / DevOps
- CI / CD

Serverless – podsumowanie

Q&A

#4

Aktualne oferty pracy:

- <https://jobs.sap.com/search/?locationsearch=gliwice>
[\(https://url.sap/3vywdy\)](https://url.sap/3vywdy)

Śledź nas na:

- <https://hackyourcareer.github.io/>
- <https://www.facebook.com/Hack.your.Career/>

Dziękujemy
za uwagę

