CI/CD i automatyzacja procesów deweloperskich

definicja problemu i strategie

mgr inż. Jakub Woźniak

Politechnika Poznańska
Wydział Informatyki i Telekomunikacji

Agenda Wykładu

Ewolucja: Od Skryptów do Platform Engineeringu

Era DevOps (2015-2020):

- "You build it, you run it".
- Każdy zespół pisze własne pipeline'y (Jenkinsfile).
- Problem: Kognitywne przeciążenie deweloperów.

Era Platform Engineering (2025):

- Traktowanie platformy jako produktu.
- **IDP (Internal Developer Platform):** Backstage, Port.
- *Golden Paths*: Prekonfigurowane szablony usług.

Kluczowa zmiana

Przesuwamy ciężar z *konfiguracji imperatywnej* na *abstrakcję platformową*.

Jak mierzyć sukces? Metryki DORA w systemach rozproszonych

Metryka	Definicja	Wyzwanie w skali mikroserwisów
Deployment Frequency	Częstotliwość wdrożeń	Agregacja wdrożeń 100+ serwisów vs Monolit
Lead Time for Changes	Czas od commitu do prod	Zależności "Train Release Model"
Change Failure Rate	% awaryjnych wdrożeń	Kaskada błędów między serwisami
Mean Time to Recovery	Średni czas naprawy	Distributed Tracing (Root Cause Analysis)

Architektura Repozytorium: Monorepo vs Polyrepo

Polyrepo (Wiele repozytoriów)

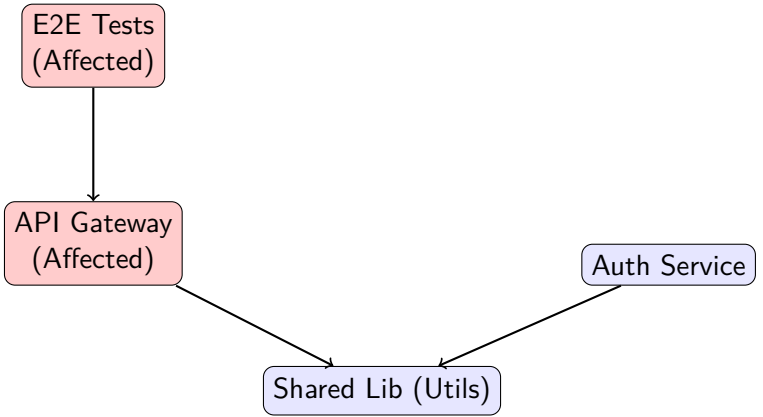
- Izolacja usług.
- Niezależne cykle wydawnicze.
- Wada: "Dependency Hell" przy aktualizacji bibliotek współdzielonych.

Monorepo (Jedno repozytorium)

- Atomowe zmiany (API + Klient w jednym commicie).
- Ujednolicone wersjonowanie.
- Wada: Czas budowania rośnie wykładniczo bez odpowiednich narzędzi.

Inteligentne Budowanie: Dependency Graph Analysis

Nie budujemy wszystkiego (‘build all‘). Budujemy tylko to, co się zmieniło (‘build affected‘).



Zmiana w "API Gateway" wymaga przebudowania tylko API i testów E2E.

Hermetic Builds Remote Caching

Hermetyczność: Proces budowania zależy *wyłącznie* od zadeklarowanych wejść (kod, narzędzia). Brak dostępu do sieci i ‘/usr/bin‘.

Remote Caching (Zdalne buforowanie):

1. Oblicz Hash $= H(SourceFiles + EnvVars + CompilerFlags)$.
2. Sprawdź w Cache (S3/Redis).
3. **Hit:** Pobierz artefakt (sekundy).
4. **Miss:** Buduj i wyślij do cache (minuty).

Efekt skali

Jeśli kolega z zespołu już zbudował bibliotekę ‘common-math‘, Twój build pobierze gotowy plik binarny.

Paradygmat GitOps: Push vs Pull

CIOps (Push Model - np. Jenkins)

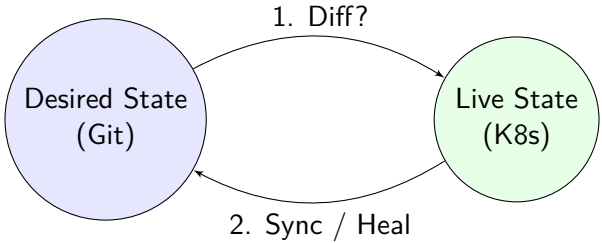
- CI ma ‘kubectl apply‘.
- CI ma "klucze do królestwa".
- **Configuration Drift:** CI nie wie, jeśli ktoś zmienił coś ręcznie na klastrze.

GitOps (Pull Model - np. ArgoCD)

- Agent działa *wewnątrz* klastra.
- Git jako jedyne źródło prawdy (SSOT).
- **Reconciliation Loop:** Ciągłe porównywanie stanu Git ze stanem klastra.

Pętla Rekoncyliacji (The Reconciliation Loop)

GitOps to implementacja teorii sterowania w inżynierii oprogramowania.



Pętla Rekoncyliacji (The Reconciliation Loop) - kont.

Przykład obiektu ArgoCD Application:

```
apiVersion: argoproj.io/v1alpha1
kind: Application
spec:
  source:
    repoURL: https://github.com/my-org/infra.git
    path: k8s/overlays/prod
  destination:
    server: https://kubernetes.default.svc
  syncPolicy:
    automated:
      prune: true       # ńUsu zasoby, których nie ma w Git
      selfHeal: true    # Napraw, śjeli śkto łzmieni ęrcznie
```

Zarządzanie Sekretami w GitOps

Nie możemy trzymać haseł jawnym tekstem w Git. Dwa podejścia:

1. **Szyfrowanie w Git (Sealed Secrets):**
   - Szyfrujemy kluczem publicznym klastra.
   - Tylko kontroler w klastrze ma klucz prywatny.
   - Plik w repozytorium jest bezpieczny.
2. **Zewnętrzny Skarbiec (External Secrets Operator):**
   - Sekrety w AWS Secrets Manager / HashiCorp Vault.
   - W Git tylko referencja (wskaźnik).
   - Operator pobiera sekret i wstrzykuje go jako Kubernetes Secret.

Deployment vs Release

Kluczowe rozróżnienie

Deployment (Wdrożenie): Instalacja nowej wersji oprogramowania na infrastrukturze (czynność techniczna).
Release (Wydanie): Udostępnienie funkcjonalności użytkownikom (decyzja biznesowa).

W systemach rozproszonych unikamy "Big Bang Deployments" (wszyscy naraz). Stosujemy **Progressive Delivery**.

Strategie Wdrożeń

Strategia	Mechanizm	Zalety	Wady
Rolling Update	Stopniowa wymiana Podów	Standard w K8s, brak downtime	Trudny rollback, brak testów na grupie
Blue/Green	Dwa środowiska, przełącznik LB	Błyskawiczny rollback	Koszt (2x zasoby)
Canary	Przekierowanie % ruchu	Minimalny "Blast Radius"	Złożona konfiguracja (Service Mesh)

Argo Rollouts: Canary Analysis

Rozszerzenie standardowego ‘Deployment‘ w Kubernetes. Pozwala na automatyczną analizę metryk.

```
apiVersion: argoproj.io/v1alpha1
kind: Rollout
spec:
  strategy:
    canary:
      steps:
      - setWeight: 20
      - pause: {duration: 5m}
      - analysis:
          templates:
          - templateName: success-rate # źSprawd Prometheus
      - setWeight: 50
      - pause: {duration: 10m}
```

Jeśli ‘success-rate‘ spadnie poniżej 99%, nastąpi **automatyczny rollback**.

Problem Stanu w CI/CD

Aplikację (Stateless) łatwo cofnąć (‘git revert‘). Bazy danych (Stateful) nie da się cofnąć bez utraty danych transakcyjnych.

Wzorzec Expand-Contract (Parallel Change):

1. **Expand:** Dodaj nową kolumnę/tabelę. Stara nadal istnieje. (Baza obsługuje v1 i v2).
2. **Deploy:** Wdróż aplikację v2 (pisze do nowej, czyta z nowej, fallback do starej).
3. **Migrate:** Przepisz stare dane (Backfill).
4. **Contract:** Usuń starą kolumnę w kolejnym sprincie.

Zasada: Baza danych musi być zawsze kompatybilna wstecz o N-1 wersji.

Kubernetes StatefulSets

Dla baz danych (Postgres, Cassandra, Kafka) na K8s używamy ‘StatefulSet‘.

Różnice w aktualizacji:

- Pods mają stałą tożsamość (‘kafka-0‘, ‘kafka-1‘).
- Aktualizacja jest sekwencyjna (od końca: 2 -> 1 -> 0).
- CI/CD musi czekać na ‘Ready‘ check każdego węzła, aby zachować kworum klastra (np. Raft consensus).

Podsumowanie Wykładu

1. **Kryzys skali:** Tradycyjne CI (Jenkins) nie skaluje się dla 100+ mikroserwisów. Platform Engineering i IDP to odpowiedź.
2. **Build Engineering:** Wymaga podejścia grafowego (Bazel/Nx) i zdalnego cache'owania.
3. **GitOps:** To standard operacyjny. Klaster sam naprawia swój stan (Reconciliation Loop).
4. **Progressive Delivery:** Odseparowanie wdrożenia od wydania. Automatyczna analiza Canary zmniejsza ryzyko awarii.
5. **Bazy danych:** Są najtrudniejszym elementem. Wymagają migracji Zero-Downtime (Expand-Contract).

Pytania i dyskusja

Dziękuję za uwagę.