

Przypadek przekrojowy

Niezgodne wiadomości, niepełne wdrożenie i odzyskanie danych

mgr inż. Jakub Woźniak

Zarządzanie Systemami Rozproszonymi · Semestr zimowy 2026/27



- API zapisuje zamówienie i outbox w jednej transakcji; 202 oznacza przyjęcie.
- Publisher może powtórzyć publikację; worker deduplikuje skutki po order_id.
- SLO: 99,9% przyjętych zamówień zrealizowanych poprawnie do 60 s.
- Wiadomości po 3 nieudanych dostarczeniach trafiają do DLQ.

Scenariusz dydaktyczny; format komunikatów i pomiary zdefiniowane na kolejnych slajdach.

Ostatnie 5 minut	Pomiar
Napływ	2 zamówienia/s; przyjęto 600
Realizacja	300 ukończonych, 300 w DLQ
API	Brak wzrostu 5xx; p95 = 150 ms
Worker	3 stare Pody gotowe; CPU 20%
Cel odtworzenia	RTO 45 min; RPO 5 min

Decyzja 1

Kontynuować rollout, dodać repliki czy wstrzymać zmianę? Wybierz dwa następne odczyty i uzasadnij ich wartość.

```
// API v1
{"schema": 1, "order_id": "o-101", "amount": "12.50"}

// API v2
{"schema": 2, "order_id": "o-102", "total_cents": 1250}

// Log workera v1
event=unsupported_schema schema=2 order_id=o-102
action=abandon delivery_count=2
```

- API v2 obsługuje 50% ruchu; stary worker zna tylko schema 1.
- Worker v2 ma czytać oba formaty; test kontraktu potwierdza tę własność.

```
$ kubectl get pods -l app=worker
NAME          READY   STATUS
worker-v1-a   1/1     Running
worker-v1-b   1/1     Running
worker-v1-c   1/1     Running
worker-v2-a   0/1     ImagePullBackOff

# Event
Failed to pull image: manifest unknown
```

- Deklaracja wskazuje digest, którego nie ma w docelowym rejestrze.
- Istnieje zweryfikowany obraz v2 pod innym digestem.

Decyzja 2

Jak skorygować stan oczekiwany i powstrzymać powstawanie kolejnych nieobsługiwanych wiadomości?

Działanie	Co zmienia	Czego nie naprawia
Cofnięcie API do v1	Nowe komunikaty mają schema 1	Istniejących komunikatów schema 2
Wdrożenie kompatybilnego workera v2	Obsługa obu formatów	Automatycznego powrotu wiadomości z DLQ
Dodanie workerów v1	Więcej starego kodu	Niezanego formatu
Ręczna zmiana w klastrze	Chwilowy stan	Trwałej deklaracji GitOps

Argo CD ma selfHeal. Zmiany awaryjne muszą uwzględnić działanie kontrolera.

Dane

300 wiadomości do ponowienia. Nowy worker może obsłużyć 12/s; bezpieczny limit bazy 10/s. Napływ nowych: 2/s. Przyjęto łączny limit obsługi 8/s.

- Dostępna przepustowość replay: $8 - 2 = 6/s$.
- Minimalny czas w modelu: $300 / 6 = \mathbf{50\ s}$.
- Najpierw mała próbka, kontrola zgodności i deduplikacji, potem limitowany replay.

Stałe tempo, brak dodatkowych błędów. Rzeczywisty czas obejmuje przygotowanie i weryfikację.

10:05 API cofnięte do v1; nowe zamówienia realizowane

10:06 ręczny skrypt porządkowy:

```
DELETE FROM orders WHERE status = 'pending';
```

```
rows_affected = 40
```

10:08 wykrycie brakujących zamówień

Od DELETE: 240 nowych poprawnych zamówień.

Backup bazy + pełne archiwum WAL dostępne.

Replika zawiera ten sam DELETE.

Decyzja 3

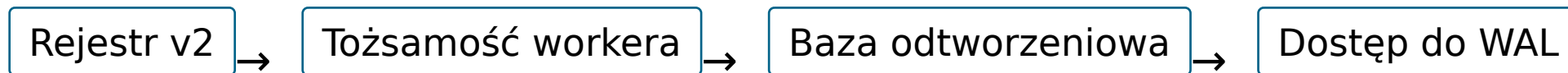
Jak odzyskać 40 rekordów i zachować 240 nowych? Czy wolno teraz uruchomić pełny replay DLQ?

Krok	Warunek rozpoczęcia	Dowód zakończenia
Izolowany PITR	Znany moment błędnej transakcji	Odzyskany stan sprzed DELETE
Porównanie	Lista brakujących order_id	Różnice i zależne rekordy
Kontrolowane scalenie	Zweryfikowane późniejsze skutki	40 odzyskanych; 240 nowych zachowanych
Replay	Spójne dane i bezpieczne skutki	Brak powtórnej realizacji

Odtwarzanie rekordów i ponawianie skutków zewnętrznych to odrębne decyzje.

Od początku zakłócenia	Czas
Wykrycie i zebranie danych	8 min
Izolowane odtworzenie	12 min
Porównanie i scalenie danych	10 min
Wdrożenie, replay i walidacja	8 min
Razem — model sekwencyjny	38 min

- Cel 45 min → 7 min zapasu.
- Przy braku osoby uprawnionej do scalenia danych zapas może zniknąć.
- RPO ocenić według odzyskanych zapisów, nie daty najstarszego backupu.



- Sprawdzić dostęp do istniejącego obrazu, kluczy i kopii.
- Rola workera do odbioru nie musi pozwalać narzędziu naprawczemu na replay.
- Tymczasowe uprawnienia operatora: ograniczony zakres, zapis użycia i wycofanie.

Wariant dydaktyczny	Dodatkowy koszt	Wpływ
Baza restore na 4 h	$4 \times 0,50 \text{ EUR} = 2 \text{ EUR}$	Umożliwia porównanie danych
Podwojenie workerów	Większy koszt obliczeń	Nie usuwa błędu formatu
Stała baza zapasowa	Koszt całomiesięczny	Sama replikuje błędny DELETE

- Porównać rzeczywisty mechanizm ochrony, a nie wyłącznie liczbę replik.

Stawki umowne; pominięto storage i transfer.

Obszar	Warunek
Wiadomości	Obsługiwane formaty; DLQ rozliczona
Dane	Brak utraty nowych zamówień i podwójnych skutków
Wdrożenie	Git, klaster i zweryfikowany digest zgodne
Usługa	Realizacja wróciła do celu; obserwacja stabilności
Proces	Oś czasu i działania z właścicielami

- Późne wykonanie nie usuwa wcześniejszego naruszenia SLO.

Do obrony

Jedna decyzja ograniczająca wpływ, jedna przywracająca zgodność oraz jeden dowód poprawności danych.

- Który fakt odrzucił hipotezę braku CPU?
- Która kontrola wykryłaby brak obrazu przed promocją API?
- Która zmiana organizacyjna zapobiegłaby niekontrolowanemu DELETE?
- Jakie koszty i ryzyka pozostają po odzyskaniu usługi?

- Kubernetes — Debug running Pods
- Argo CD — Automated Sync Policy
- PostgreSQL — Continuous archiving and PITR
- Google SRE — Managing incidents
- Google SRE — Implementing SLOs

Dokumentacja sprawdzona: 26.09.2026. Dane przypadków: przykłady dydaktyczne.