# Kurs DevOps

Lista 1

15 i 16 października 2025

Jako że zazwyczaj w dużych grupach znajduje się co najmniej jeden użytkownik Windowsa, to informuje się, iż zadania należy wykonywać z linii poleceń, o ile nie powiedziano inaczej w treści.

**Zadanie 1.** (3 p.)
Przygotuj (w formie dokumentu elektronicznego[1]) nieabstrakcyjny projekt skalowalnego systemu (ang. Non-abstract Large System Design) podobnie do przykładu z [rozdziału 12, 1]. Jeśli jakieś wymogi/ograniczenia nakładane przez HW są istotne, a nie są podane, to przyjmij sensowne założenia. Wybierz jeden z poniższych scenariuszy:

- Aplikacja do rozliczeń bankowych – masz tabelę z listą klientów i stanem ich konta, strumień zleceń przelewów do wykonania oraz strumień potwierdzeń otrzymania wyniku o realizacji przelewu. Zakładamy, że użytkownik zleca wykonanie przelewu przez zewnętrzną aplikację (np. sklep internetowy) dostajemy wtedy prośbę o przelew, którą trzeba zweryfikować (czy użytkownik ma odpowiednio dużo środków), następnie przenieść kwotę między kontami (zakładamy, że oba są w naszej bazie) i wysłać rezultat przelewu do zewnętrznej aplikacji. Oczekujemy, że zewnętrzna aplikacja potwierdzi nam odebranie rezultatu, jeśli nie, to ponawiamy wysłanie rezultatu. Docelowe obciążenie i niezawodność:
  - 10 tys. przelewów na sekundę
  - weryfikacja przelewu w 5 sekund w 95% przypadków
  - weryfikacja przelewu w ciągu 10 min w 99,9% przypadków
  - 99,999% dostępności możliwości przelewu
  - 99,9999% niezawodności w księgowaniu stanu konta na korzyść klienta
  - 100% niezawodności w księgowaniu stanu konta na niekorzyść klienta (czyli nie możemy zaksięgować nic na niekorzyść)
  - ponawianie prób wysłania rezultatu przelewu przez 2 następne dni
- System obsługi telefonii – który posiada tabelę z ostatnią lokalizacją każdego telefonu komórkowego (wraz z saldem) oraz strumień rozpoczynanych rozmów telefonicznych i wysyłanych SMS-ów (zakładamy, że z naszej perspektywy to jest to samo). Dla każdej prośby o nawiązanie komunikacji sprawdzamy w bazie lokalizację odbiorcy i zmniejszamy saldo nadawcy o 1 (jeśli było niezerowe). Jeśli nadawca był w stanie pokryć koszt, to wysyłamy odpowiedź z lokalizacją odbiorcy. Jeśli nie, to zwracamy błąd. Docelowe obciążenie i niezawodność:
  - 10 tys. nawiązań komunikacji na sekundę
  - opóźnienie odpowiedzi mniejsze niż 500ms w 95% przypadków
  - opóźnienie odpowiedzi mniejsze niż 1s w 99,9% przypadków
  - 99,999% dostępności systemu
  - 99,99% niezawodności w aktualizacji salda[2]

[1] Pamiętaj o wysłaniu go poprzez SKOS-a.

[2] Klient się raczej nie zorientuje, jak policzymy mu dodatkową minutę, a i dla nas dodatkowa minuta to nie jest duży koszt.

- System strumieniowania filmów – posiadający bazę z filmami (dużymi, rzędu setek megabajtów) i strumień próśb o wyświetlenie danego materiału. Każda prośba inicjuje wysyłanie pliku we fragmentach, przy czym jest ono rozłożone w czasie równym długości filmu. Docelowe obciążenie i niezawodność:
  - 500 nowych wyświetleń na sekundę
  - opóźnienie rozpoczęcia transmicji mniejsze niż 5s w 95% przypadków
  - w 95% przypadków, nie przekraczamy 20MB zużycia pamięci na buforowanie najbliższego fragmentu filmu
  - w 99,99% przypadków kolejny fragment filmu jest dostępny w momencie, gdy wyświetlanie poprzedniego zostanie zakończone.

**Zadanie 2.** 🐻
Załóż, że aktualnie prywatna chmura, którą zarządza Twój zespół, ma 10 tys. rdzeni, z czego 5 tys. przypada na weryfikację dostarczeń deweloperów do poszczególnych mikroserwisów (połowa z tego na testy, a połowa na statyczną analizę przy pomocy dwóch różnych narzędzi), a 5 tys. na weryfikację łączną mikroserwisów i wydawanie 2 produktów. Aktualnie zespół deweloperów liczy 100 osób.

W ramach zbierania informacji o zapotrzebowaniu mocy obliczeniowej na przyszły rok udało Ci się dowiedzieć:

- planowane jest zatrudnienie 10 nowych deweloperów;
- zostanie wprowadzony jeden nowy produkt (składający się z już istniejących mikroserwisów);
- chodzą plotki, że jeden z klientów jest zirytowany luką w zabezpieczeniach, która została niedawno odkryta w jednym z waszych produktów.

Wyestymuj ile rdzeni w chmurze będzie potrzeba w przyszłym roku.

**Zadanie 3.** 🐻
Metryki oraz wskaźniki są podstawą pracy DevOps i ogółem korporacji, trzeba jednak pamiętać, by je dobrze sformułować, bowiem zła formuła może spowodować, że będziemy optymalizowali coś czego optymalizować nie chcemy (zgodnie z zasadą "Optymalizujesz dokładnie to, co mierzysz."). Dla poniższych przykładów wskaźników wymyśl, w jaki sposób najłatwiej je zoptymalizować:

- liczba zgłoszonych błędów (minimalizacja),
- liczba zgłoszonych elementów potencjalnie zagrażających BHP (maksymalizacja),
- stosunek godzin poświęconych na implementację danej funkcjonalności, względem początkowej estymaty (optymalizacja w kierunku 1.0)
- stosunek wstępnej estymaty czasu potrzebnego na implementację do czasu z estymaty pogłębionej (optymalizacja w kierunku 1.0)

**Zadanie 4.**
Opisz czym jest *Chaos Monkey* i czemu możemy chcieć go używać[3] [4].

**Zadanie 5.**
Czym jest „zarządzanie zmianami" (ang. Change Management) wprowadzone przez Kurta Lewina?

**Zadanie 6.**
Omów „John Kotter's 8-Step Process for Leading Change". Jak uważasz, czemu te kroki są potrzebne i co się stanie, jeśli jakiś pominiemy/nie uda się go zrealizować?

**Zadanie 7.**
Czym jest SLO[1]? Czemu nie chcemy, by jakaś metryka miała cel 100%?

[3] `https://netflixtechblog.com/netflix-chaos-monkey-upgraded-1d679429be5d?gi=7243927a6f23`
[4] `https://github.com/netflix/chaosmonkey`

# Bibliografia

[1] Betsy Beyer i in. *The Site Reliability Workbook: Practical Ways to Implement SRE*. 1st. O'Reilly Media, Inc., 2018. ISBN: 1492029505. URL: `https://sre.google/workbook/table-of-contents/`.

[2] Fay Chang i in. "Bigtable: A Distributed Storage System for Structured Data". W: *7th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 2006, s. 205–218.