Przygotowali:
148913
Arkadiusz Żak
Michał Mazur
Jonatan Kozdęba-Łojek
Karol Hamielec

Czym jest Chmura?

Chmura to grupa połączonych ze sobą serwerów tworzących jeden ekosystem, które udostępniają swoje zasoby na rzecz różnorakich usług.

Czym jest aplikacja chmurowa/ internetowa?

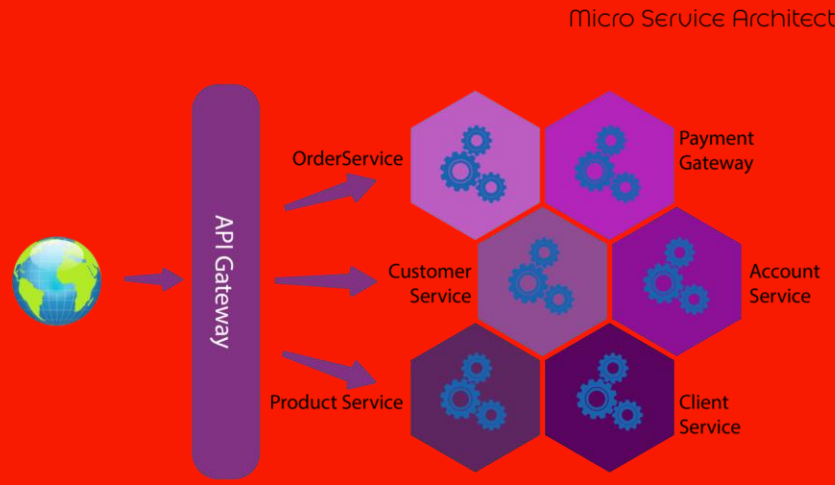
- Aplikacja Chmurowa to program który korzysta z zasobów środowiska chmurowego,
  w przeciwieństwie do tradycyjnych aplikacji,
  które działają lokalnie na maszynie klienta.

Jakie są cechy aplikacji chmurowej?

- **Aplikacje chmurowe:**
- Są spakowane w kontenery, które są odizolowane od reszty środowiska.
- Są dynamicznie zarządzane; chmura przydziela im zasoby w zależności od potrzeb, przez co optymalizuje koszta utrzymania.
- Wykorzystują mikroserwisy.

Czym są Mikroserwisy?

Architektura oparta na mikroserwisach to sposób tworzenia aplikacji, które w przeciwieństwie do klasycznych aplikacji monolitycznych, są złożone z wielu niezależnych od siebie modułów, zwanych serwisami.



Jakie cele stawiają sobie wzorce projektowe wykorzystywane w Chmurach?

- Dostępność: zasoby powinny być nieprzerwanie dostępne
- Zarządzanie danymi: dystrybucja danych powinna być łatwa, co pozwala na lepszą skalowalność.
- Asynchroniczna komunikacja: usługi komunikują się asynchronicznie, co również ułatwia skalowalność.
- Zarządzanie i monitorowanie: powinien istnieć sposób na monitorowanie zasobów i zarządzanie nimi.

- Wydajność i skalowalność: aplikacja powinna pozostać wydajna jednocześnie dbając o skalowalność.
- Odporność: aplikacja powinna być odporna na awarie, żeby utrzymać dostępność.
- Bezpieczeństwo: aplikacja powinna być przygotowana na potencjalne ataki.

Jakie są fałszywe założenia dotyczące tworzenia aplikacji w chmurach?

- Sieć jest niezawodna
- Opóźnienie jest zerowe
- Szerokość pasma jest nieskończona
- Sieć jest bezpieczna
- Topologia się nie zmienia
- Jest jeden administrator.
- Koszt transportu wynosi zero.
- Sieć jest jednorodna

Jakie wzorce projektowe metotologie są wykorzystywane w chmurze?

Mikroserwisy

Abstract Factory

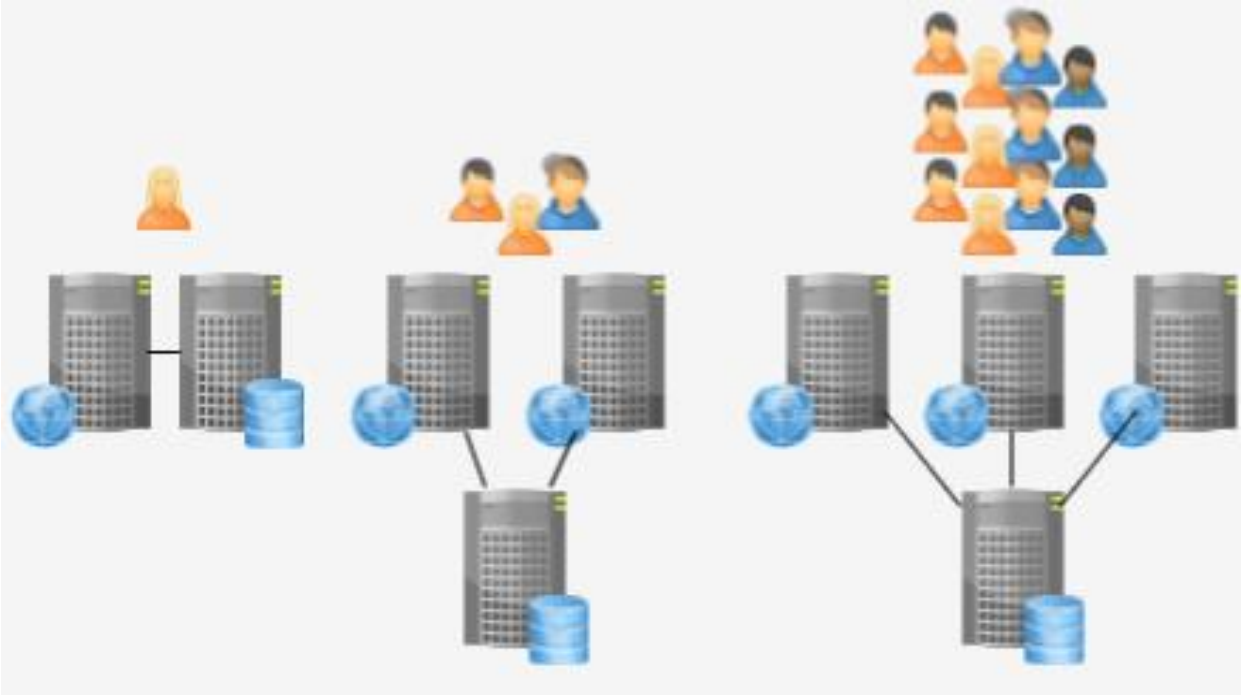
Twelve-Factor

API Gateway

Serwer konfiguracyjny

Circuit breaker

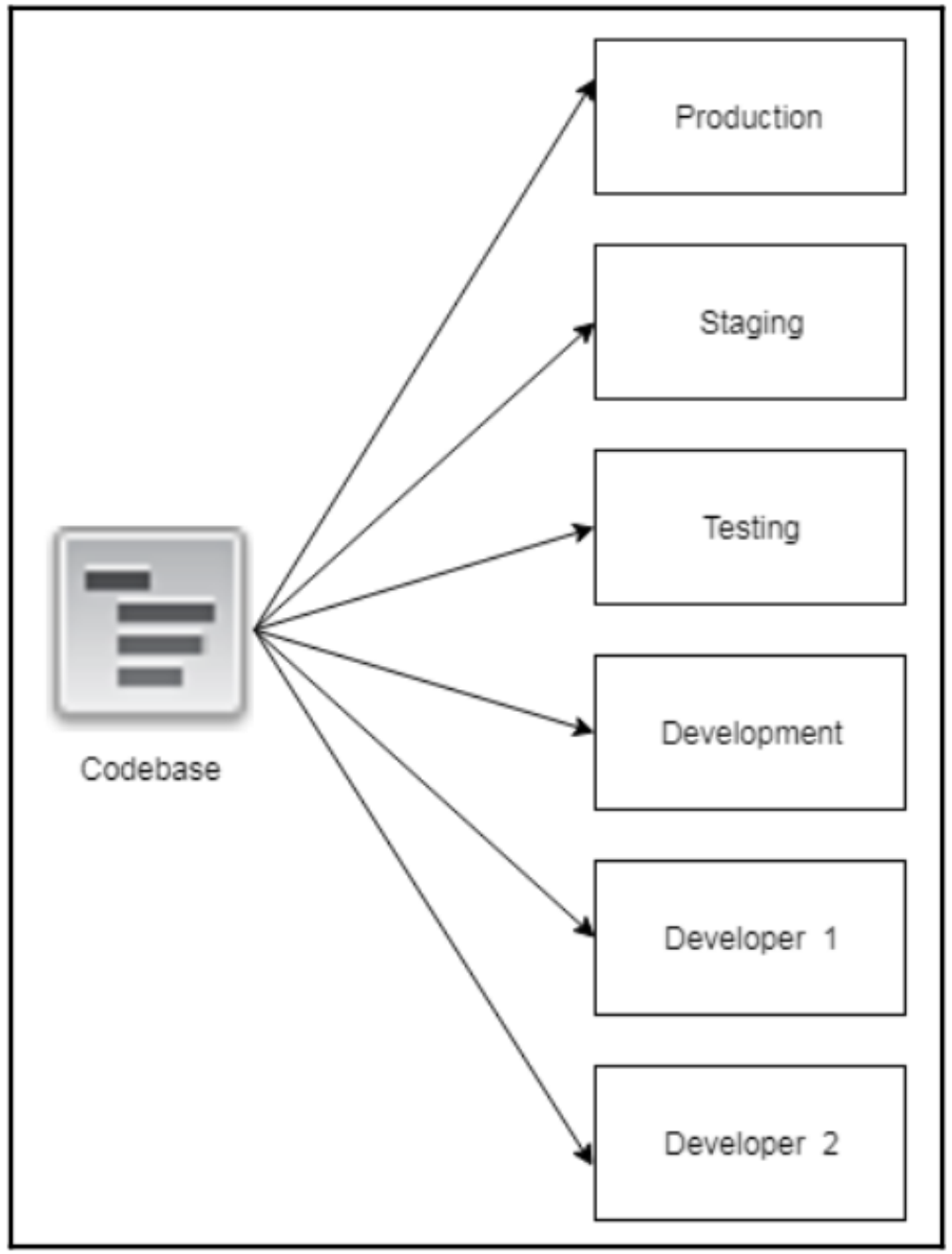
Tworzenie aplikacji chmurowych



Metodologia Twelve-factor

- Baza kodu
- Zależności
- Konfiguracja
- Usługi wspierające
- Build, release, run
- Procesy
- Port-binding
- Współbieżność
- Dyspozycyjność
- Parytet dev/prod
- Logi
- Procesy administracyjne

Baza kodu


Production
Staging
Testing
Development
Developer 1
Developer 2
Codebase

Zależność

- Każda zależność, z której korzysta projekt, musi być zadeklarowana i odizolowana od kodu. W tym przypadku możemy użyć niektórych narzędzi do zarządzania pakietami (takich jak Maven, grundle i npm).

Konfiguracja

Konfiguracja aplikacji to wszystko, co różni się pomiędzy różnymi deploymentami. Mogą to być:

- Hosty, porty i poświadczenia dostępu do bazy danych
- Schemat bazy danych
- Ustawienia cache
- Hosty, porty i dane uwierzytelniające dla dostępu do kolejek wiadomości

Usługi wspierające

- Usługi wspierające to zewnętrzne usługi używane przez aplikację, takie jak baza danych, usługa wiadomości, repozytorium plików czy usługa poczty elektronicznej.

Build
Release
Run

Proces przekształcania bazy kodowej w danym środowisku należy podzielić na trzy ściśle oddzielne etapy:

- **Build:** Kompiluje i generuje pakiet wykonywalny na przykład generując archiwum WAR.
- **Release:** Stosuje konfigurację aplikacji zawartą w pakiecie wykonywalnym. Powstaje release, będący połączeniem pakietu wygenerowanego w build z konfiguracją aplikacji.
- **Run (lub runtime):** Inicjalizuje aplikację w określonym środowisku.

Procesy

- Metodyka dwunastu czynników podkreśla, że wszystkie procesy lub komponenty aplikacji muszą być bezstanowe i niczego nie współdzielić, czyli nie powinny przechowywać informacji. Dlatego w aplikacji zdekomponowanej na mikroserwisy, każdy mikroserwis nie może przechowywać informacji w pamięci ani korzystać z pamięci podręcznej serwera.

Przypisanie portów

- Aplikacje internetowe uruchamiane były kiedyś web-kontenerach. Na przykład aplikacja PHP może być uruchomiona wewnątrz Apache HTTPD lub aplikacja Javy może być uruchamiana wewnątrz Tomcata.
- W nowoczesnych aplikacjach chmurowych aplikacje powinny być samowystarczalne I nie polegać na wstrzykiwaniu w inne webservery. Aplikacje powinny udostępniąć swoje usługi poprzez związanie z portem I obsługiwać zapytania np. HTTP poprzez nasłuchiwanie na danym porcie.
- Podczas rozwijania aplikacji w środowisku deweloperskim programista używa poprostu bezpośredniego URL, aby połaczyć się z aplikacją, a w środowisku produkcyjnym za poprawny routing odpowiada inna warstwa infrastruktury (load-balancer, service mesh, reverse proxy)

Współbieżność

- Aplikacje powinny brać przykład z UNIXowego podejścia do zarządzania procesami. Korzystając z tego modelu, programista może obsłużyć różnego typu obciążenie, przypisując konkretnym zadaniom dedykowane procesy. Na przykład: zapytania HTTP mogą być obsługiwane przez jeden proces, natomiast zajmujące dużo czasu zadania powinny być wykonywane przez inny proces.
- Zależnie od przyjętej technologii jednostkami współbieżnymi mogą być wątki lub podprocesy

Logi

- Aplikacja dwunastoczynnikowa nigdy nie zajmuje się routingiem ani przechowywaniem swoich logów. Nie powinien próbować zapisywać ani zarządzać plikami logów. Zamiast tego każdy uruchomiony proces zapisuje swój strumień zdarzeń, niebuforowany, na stdout. Podczas rozwoju lokalnego programista będzie wyświetlał ten strumień na pierwszym planie swojego terminala, aby obserwować zachowanie aplikacji.
- W środowisku produkcyjnym logi aplikacji będą zbierane przez zaawansowany system analizy logów taki jak ELK (Elasticsearch, Kibana, Logstash)

Parytet Dev/prod

- Utrzymuj rozwój, etapy i produkcję tak podobne, jak to możliwe
- Historycznie rzecz biorąc, istniały znaczne luki między środowiskiem produkcyjnym a testowym. Wynikały one min. Z:
  - Długiego czasu wdrożenia (aktualizacje raz na kwartał)
  - Osobnego zespołu wdrożeniowego I osobnego zespołu deweloperskiego
  - Odrębnego zestawu narzędzi (Nginx vs Apache, SqlLite vs Mysql)
- W nowoczesnej aplikacji:
  - produkcja aktualizowana kilka razy w tygodniu
  - Zespół utrzymujący aplikacje I ją rozwijający jest tym samym zespołem
  - Technologie używane w środowiskach testowych I produkcyjnych powinny być do siebie zbliżone (prawie takie same)

Procesy administracyjne

- Jednorazowe procesy administracyjne powinny być uruchamiane w identycznym środowisku, jak zwykłe, długotrwałe procesy aplikacji. Działają w oparciu o wydanie, używając tej samej bazy kodu i konfiguracji, co każdy proces uruchamiany w odniesieniu do tego wydania. Kod administratora musi być dostarczany z kodem aplikacji, aby uniknąć problemów z synchronizacją.

Jednorazowość (disposability)

- Procesy aplikacji dwunastu czynników są jednorazowe, co oznacza, że można je uruchomić lub zatrzymać w dowolnym momencie
- Procesy powinny dążyć do minimalizacji czasu uruchamiania. W idealnym przypadku proces zajmuje kilka sekund od momentu wykonania polecenia uruchomienia do momentu, gdy proces jest gotowy do przyjmowania żądań lub zadań. Krótki czas uruchamiania zapewnia większą elastyczność procesu wydawania i skalowania
- Procesy zamykają się płynnie po otrzymaniu sygnału SIGTERM od menedżera procesów. W przypadku procesu internetowego płynne zamknięcie uzyskuje się poprzez zaprzestanie nasłuchiwania na porcie usługi

Rejestr usług

Baza danych zarejestrowanych usług

Rejestruje się tam każdy tworzony mikroserwis

Po usunięciu mikroserwisu jest on usuwany z rejestru

Klient mikroserwisu uzyskuje dostęp do rejestru usług

Rejestr sprawdza stan usługi jej dostępność oraz podaje klientowi jej lokalizację

Serwer konfiguracji

- Udostępnia wszystkie potrzebne dane I właściwości danej usługi potrzebne do działania
- Właściwości usługi uzuskiwane są z miejsca okreslonego przez scieżkę podaną przy rejestracji mikrousługi
- Uzyskiwane informacje zapisywane są w pamięci
- Ścieżka jest kontrolowana przez serwer wersji (np. Git,subversion)

Wzorzec wyłącznika

- Zapobiega wielokrotnym próbą uruchomienia operacji w sytuacja gdy prawdopodobieństwo niepowodzenia jest wysokie
- Sprawdza czy błąd jest rozwiązany
- Działa jak serwer proxy który wysyła żadanie do operacji lub natychmiast zwraca wyjątek
- Jeżeli operacja się nie powiedzie licznik niepowodzeń jest automatycznie zwiększany

3 różne stany

- Zamknięty - serwer proxy wysyła żadanie do operacji
- Otwarty – po nadejściu żadania zwraca wyjątek
- Półotwarty - przekazywana jest ograniczona liczba żądań. Jeżeli zadania powiądą się wtyedy serwer przechodzi w stan zamknięty, jeżeli nie w otwarty

3 różne stany