Projekt 2 (5 pkt)

Celem projektu jest stworzenie aplikacji webowej hostowanej w chmurze o architekturze mikroserwisowej.

1. Stwórz aplikację webową w wybranej przez siebie technologii lub wykorzystaj aplikację stworzoną na potrzeby poprzedniego projektu, tematyka aplikacji jest dowolna (może to być np. aplikacja umożliwiająca prosty chat grupowy) aplikacja powinna:
    - Posiadać przynajmniej 4 endpointy GET oraz 4 endpointy POST (w tym jeden POST i GET do przesłania/odbierania plików multimedialnych)
2. Architektura mikroserwisowa – Podział
    - Frontend – oddzielny niezależny moduł hostowany niezależnie od backendu w dowolny sposób
    - Backend - podzielony na kilka mikroserwisowych modułów, które będą wdrożone za pomocą AWS Fargate:
        i. Moduł autoryzacji - Wykorzystuje AWS Cognito do zarządzania rejestracją, logowaniem
        ii. Moduł odpowiedzialny za główną logikę aplikacji np. zarządzanie czatami - odpowiada za tworzenie czatów, pobieranie listy czatów oraz zarządzanie historią wiadomości. Moduł powinien mieć dedykowaną bazę danych np. do przechowania czatów
        iii. Moduł obsługi przesyłania plików odpowiedzialny za przesyłania i przetwarzania plików (np. obrazków przesyłanych w ramach wiadomości). Moduł powinien mieć dedykowaną bazę danych np. do przechowywania metadanych przesyłanych plików. Pliki powinny być przechowywane w S3
        iv. Moduł notyfikacji odpowiedzialny za generowanie i wysyłanie powiadomień do użytkowników, np. o nowych wiadomościach, aktualizacjach czatów lub innych zdarzeniach. Moduł powinien mieć dedykowaną bazę danych np. do przechowania historii powiadomień. Mechanizm powiadomień powinien zostać zrealizowany z użyciem AWS SNS
3. Każdy z modułów wdrażanych na AWS Fargate musi wspierać automatyczne skalowanie. W ramach rozwiązania należy skonfigurować mechanizmy umożliwiające dynamiczne zwiększanie lub zmniejszanie liczby uruchomionych zadań (tasks) lub replik kontenerów. Minimalna liczba uruchomionych zadań dla każdego modułu powinna wynosić dwie identyczne repliki, co gwarantuje redundancję i ciągłość działania systemu.
4. Każdy z modułów powinien użyć niezależnej bazy danych, w aplikacji powinny zostać wykorzystane przynajmniej dwa serwisy bazodanowe oferowane przez AWS np. Amazon RDS, Amazon DynamoDB, Amazon DocumentDB, Amazon Keyspaces, Amazon Neptune, Amazon Timestream, Amazon ElastiCache, Amazon QLDB, Amazon MemoryDB
5. Utwórz konfigurację Dockera dla każdego z modułów
6. Cała konfiguracja infrastruktury powinna być wykonana w Terraform