Architektura Rozwiązania	
Nazwa produktu:	Zalacznik 6 POIR-NUBES-SYS-AR architektura systemu NUBES cz.1.pdf
Autor:	Autorzy ukryci

Historia rewizji:

Nr:	Data:	Opracował	Opis Zmian
1.0	2022-08-29	KACI	Inicjalizacja dokumentu, wprowadzenie
1.1	2022-09-14	MIPO	Dodanie diagramu architektury, opisów architektury, słownika pojęć oraz opisu przypadków użycia
1.2	2022-09-21	MIPO	Porządkowanie dokumentu, dodanie RabbitMQ do architektury, usunięcie Load Balancera przed Local Node, uzupełnienie słownika pojęć
1.3	2022-10-12	MIPO	Zmiany w architekturze, przyporządkowanie odpowiedzialności
1.4	2022-12-10	WLPL	Usuniete/ukryte wrazliwe dane.

SPIS TREŚCI:

1. Wprowadzenie

Celem projektu na obecnym etapie jest opracowanie projektu architektury IT rozwiązania chmurowego służącego do zbierania, analizowania i modelowania danych w ramach projektu “Innowacyjny system interkomowy” prowadzonego przez Ambient System.

2. Słownik pojęć

- GCP - Google Cloud Platform
- HA - skrót od High Availability, instancja HA to instancja powstała na potrzeby całodobowego dostępu do serwera i zapewnienie stałej pracy usługi, nawet w przypadku zawodzenia jednej z instancji w klastrze.
- ETL - skrót od Extract, Transform and Load - procesy pozyskania danych dla baz danych, szczególnie dla hurtowni danych. Stanowią nieodzowny element procesów z zakresu Business Intelligence.
- ELT - Extract, Load & Transform - w przeciwieństwie do ETL, dane są najpierw ładowane do hurtowni a finalnie transformowane na potrzeby analityki, a nie odwrotnie.
- REST API - Representational State Transfer + Application Programming Interface - API definiuje jak użytkownik może skomunikować się z systemem, reguły określające jak użytkownik może uzyskać dostęp do zasobów oraz w jakiej postaci je otrzymuje. Natomiast REST to styl architektury definiujący kształt API.
- Cloud-Connected - wersja Local Node + Master Cloud Application. Przeciwieństwem do Cloud-Connected jest wersja całkowicie pozbawiona połączenia z chmurą, które za pośrednictwem konfiguracji też jest możliwa.
- Sync Mechanism - mechanizm synchronizacji, aplikacja odpowiedzialna za synchronizowanie informacji pomiędzy Master Cloud Application(7) a Local Node(2)
- DAG - Directed acyclic graph - skierowany graf acykliczny, w przypadku DBT graf zależności procesu. Rysunek pokazujący jak przebiega dany job, proces.
- DBT - narzędzie do dewelopowania, testowania oraz deployowania procesów przetwarzania danych w języku SQL. Narzędzie posiadające interfejs webowy , możliwość wersjonowania kodu zapytań sql, automatyczne generowanie dokumentacji i wiele innych funkcjonalności wspomagających proces tworzenia procesów przetwarzających dane. (https://www.getdbt.com/)
- YAML - uniwersalny język do reprezentowania danych w ustrukturalizowany sposób. Często używany do tworzenia plików konfiguracyjnych.
- DWH - Data Warehouse - hurtownia danych.

3. Podsumowanie

W ramach projektu przeprowadzono warsztaty ..... mające na celu określenie wymagań funkcjonalnych oraz technologicznych dotyczących rozwiązania.

Efektem pracy są następujące produkty końcowe:

a. koncepcja architektury IT proponowanej do realizacji projektu z wykorzystaniem chmury GCP (Google Cloud Platform) składająca się z:

   i. opisu elementów składowych rozwiązania
   ii. diagramu funkcjonalnego

4. Opis elementów składowych rozwiązania

System podzielony jest na część działającą w chmurze Google Cloud oraz część działającą lokalnie. Lokalnie będą to dwie maszyny fizyczne, na pierwszej z nich będzie uruchomiony działający komplet aplikacji odpowiadających za działanie systemu, a na drugiej instancji zapasowy komplet, który wchodzi w życie tylko i wyłącznie w przypadku gdy instancja nr 1 zawodzi.

Każdy z numerów jest odzwierciedlony pod tym samym numerem na rysunku załączonym architektury załączonym do dokumentu.

1. **Urządzenia brzegowe (interkomy).**
   - Rejestrują i dostarczają (poprzez REST API) następujące rodzaje danych:
     i. nagrania, dane wektorowe, raw data, ewentualne metryki
     ii. wykryte zdarzenia (niezależnie od ich szczegółowości)
   - Przyjmują z Local Node (również poprzez REST API):
     i. dane konfiguracyjne
     ii. nowe modele ML do zdeploy'owania

2. **Local Master App**
   realizujące zaplanowane funkcje. Uruchomiona jako aplikacja skonteneryzowana. Aplikacje będą przyjmowały dane od Urządzeń brzegowych - staging danych metrycznych, wektorowych, wykrytych eventów, pełna konfiguracja urządzeń dla urządzeń brzegowych spiętych z tym node. Aplikacja webowa do zarządzania. Aplikacja odpowiada także za natychmiastowe wypychanie danych o wykrytych zdarzeniach lub danych konfiguracyjnych lokalnej instalacji do chmury. Jeżeli z różnych przyczyn nie powiedzie się wypchnięcie takich danych - dalsze próby automatycznie przejmuje Sync Mechanizm (6)
   - 2A - Backup Local Master App - instancja zapasowa, bliźniacza, wchodząca w życie tylko w przypadku, gdy pierwsza zawiedzie, następuje wtedy zmiana adresacji, tak by maszyna backup uzyskała adresację maszyny głównej

3. **Website -** front-end aplikacji local node

4. **RabbitMQ**
   Kolejka wiadomości obsługująca napływ informacji priorytetowych z urządzeń EDGE. Eventy wykryte, wektory danych. RabbitMQ jako skonteneryzowany.
   - 4A - Zapasowa instancja RabbitMQ

5. **Lokalna instancja bazy danych (PostgreSQL/ MongoDB)**
   przechowująca wszystkie dane Local Master App, nie skonteneryzowana. Uruchomiona fizycznie na tej samej maszynie co aplikacja. Relacyjna baza danych (mysql lub postgresql) - obejmujące dane które dalej możemy traktować jako stream data lub batch data
   - 5A - Replika w/w bazy danych na maszynie zapasowej

6. **Sync Mechanism (aplikacja),**
   który dba o synchronizację danych pomiędzy Local Node a Master Node w chmurze. Aplikacja skonteneryzowana. Mechanizm nie będzie działał w konfiguracji pozbawionej połączenia z chmurą. Ten mechanizm odpowiada tylko za wypychanie danych do chmury oraz rejestrowanie co jest zsynchronizowane a co nie.
   - 6A - Mechanizm posiada swój odpowiednik uruchomiony na Backup Local Node.
   - push desynchronized events data - przesyła ewentualnie niezsynchronizowane dane o wykrytych zdarzeniach
   - push configuration data - przesyła ewentualne niezsynchronizowane dane nt zmian w konfiguracji dotyczących local node lub podległych mu urządzeń brzegowych
   - push raw data - przesyła (niezsynchronizowane dane) - nagrania / dane wektorowe / ogólnie raw
7. **Master Cloud Application**
   - zestaw aplikacji i natywnych usług GCP realizujących opisane poniżej funkcje.
   - 7A - app - aplikacja “Matka” uruchomiona na Google Kubernetes Engine- w pełni skalowalne rozwiązanie, zarządzalne, deploy oparty o kontenery Docker, łatwo konfigurowalny z poziomu plików YAML.
   - 7B - Relacyjna baza danych uruchomiona w usłudze typu managed database - Cloud SQL. Przechowująca konfiguracje wszystkich Local Node-ów, konfigurację główną oraz wszelkie relacyjne dane dotyczące aplikacji, np. konta użytkowników, uprawnienia itp., a także przechowująca dane związane z funkcjonalnościami aplikacji chmurowej samej w sobie.
   - 7C - Cloud Scheduler - usługa serverless, harmonogram cron w Google Cloud. Wykorzystywany będzie do konfiguracji oraz wywoływania cyklicznych jobów w obrębie Master Cloud Application
   - 7D - Wszelkie pliki statyczne (obrazki, pliki javascript itp.) serwowane z poziomu Google Cloud Storage. Z uwagi na specyfikę tego typu storage (key-value store) - umożliwia natychmiastowy dostęp do danych z dowolnego miejsca na ziemi bez zbędnych opóźnień.
   - 7E - Pub/Sub - w pełni skalowalna usługa kolejkowania dostarczana przez GCP oferująca dodatkowo integrację z innymi komponentami jak np CF, BQ, etc.

- 7F - load balancing - usługa GCP odpowiadająca za przekierowanie ruchu zewnętrznego (od użytkowników lub z REST API), umożliwiająca rozkładanie ruchu, pozwalająca zatrzymać pozostałe podzespoły poza dostępem do internetu (z GKE tylko komunikacja poprzez Load Balancer), uruchomić kilka dodatkowych zabezpieczeń (np anty-ddos, firewall) oraz takich usług jak health check.
- 7G - logging & reporting - usługi GCP kompletnie obejmujące procesy związane ze zbieraniem logów oraz raportowaniem błędów. Do rozważenia czy w/w dane dotyczące zarówno local Node jak i Cloud nie będą zbierane do DWH celem ich dalszego przetwarzania bądź analizowania.

8. **Website**
   - front-end aplikacji cloud master
9. **Data Ingestion**
   - 9A - Cloud Storage - dane przekazywane jako pliki raw
   - 9B - Pub/Sub - dane dotyczące wydarzeń
   - 9C - Cloud storage - miejsce przechowywania nagrań, na potrzeby m.in. weryfikacji potencjalnych zdarzeń niebezpiecznych na rzecz przyszłego trenowania
10. **Data Processing**
    - jeżeli z uwagi na strukturę / typ danych pojawi się konieczność przetwarzania danych zanim trafią do DWH (BigQuery) wykorzystane będą zależnie od scenariusza następujące komponenty:
      - 10A - DataFlow - w sytuacji gdy dane będą dostarczane ciągle (streaming) - celem ich ciągłego / skalowalnego przetwarzania do zastosowania wykorzystane będzie narzędzie DataFlow - rozwiązaniu opartym o Apache Beam zarządzane przez GCP umożliwiającym skalowalne niemalże w nieskończoność równoległe przetwarzanie danych.
      - 10B - Cloud Functions - w sytuacji gdy dane będą dostarczane w sposób cykliczny (batch) do ewentualnego ich przetwarzania wykorzystane będą Cloud Funkcje - Usługa serverless od Google Cloud umożliwiająca wdrożenie z reguły jednej funkcji wykonującej jedną czynność w jednym z wielu dostępnych języków programowania. Usługa w pełni skalowalna, płatna w trybie Pay as you go.
11. **Data Warehouse**
    - przyjmujemy model pracy ELT co oznacza, że wszystkie dane będą dostarczane do DWH w swojej pierwotnej postaci bez jakichkolwiek transformacji. Może się okazać że od tej zasady pojawią się wyjątki - opisane w pkt 10. Wykorzystywane narzędzia w tej części BigQuery oraz DBT. Pierwsze do skalowalna analityczna baza danych sql, drugie to zewnętrzna usługa umożliwiająca wersjonowanie kodu, zarządzanie procesem zrównoleglania, ustawienia harmonogramu wykonywanych data pipelines, zarządzanie całym powstałym DAG oraz utrzymywaniem aktualnej dokumentacji procesów.
      - 11A - Stage - wydzielony schemat danych na BigQuery zarządzany przez DBT obejmujący (głównie) nieprzetworzone dane z poniższych strumieni danych:
        i. poprzez Data Ingestion
        ii. bezpośrednio z Cloud SQL

- 11B - Transformation - wydzielony schemat danych na BigQuery zarządzany przez DBT obejmujący dane częściowo przetworzone, wyczyszczone i uspójnione.
- 11C - DataMart wydzielony schemat danych na BigQuery zarządzany przez DBT obejmujący dane przygotowane do bezpośredniego wystawienia dla BI
- 11D - BI wydzielony schemat danych na BigQuery zarządzany przez DBT obejmujący dane podzielone na osobne obszary raportowanych faktów zdefiniowanych miar i wymiarów.

12. **Machine Learning**
    - 12A - Model Inference - przy wykorzystaniu AI Vertex przygotowanie danych trenowania
    - 12B - Model Training - przy wykorzystaniu AI Vertex trenowanie modeli z wykorzystaniem najnowszych danych
    - 12C - ML Ops - przy wykorzystaniu AI Vertex operacjonalizacja wszystkich procesów związanych z re-trenowaniem modeli.
13. **Data Studio**
    - rozwiązanie klasy BI posiadające większość niezbędnych funkcji potrzebnych do wizualizacji danych (tabele, tabele przestawne, wykresy, filtry, grafiki, zarządzanie dostępem, publikowanie, embeddowanie, itp.)
14. **Pub/Sub (model is ready)**
    - kolejka wiadomości otrzymująca od środowiska treningowego (ML) informację o tym, że model został wytrenowany prawidłowo oraz informację o tym gdzie model się znajduje (lokalizacja plików na Google Cloud Storage). Kolejka pracująca w trybie PUSH - wypycha informacje bezpośrednio pod API Master Cloud Application.

5. Diagram architektury rozwiązania

Załączniki:
POIR-NUBES-SYS-AR-architektura systemu NUBES.png
POIR-NUBES-SYS-AR-architektura systemu NUBES.svg

6. Przypadki użycia - odzwierciedlenie w architekturze.

- **REAKCJA NA ZDARZENIE Z ALGORYTMU ML WYKRYWANIA ZDARZEŃ**
  i. Rejestracja zdarzenia przez urządzenia EDGE (1)
  ii. Wysyłka przez MQTT informacji do RabbitMQ (4). Informacja trafia następnie do backendu Local Node (2)
  iii. Backend Local Node zapisuje informacje w lokalnej bazie danych (5A) jak i w logach. W przypadku wersji Cloud-Connected następuje również zapis w Google Cloud Logging (7G)
  iv. Backend Local Node podejmuje decyzję (wykrycie zdarzenia) na bazie odebranych informacji, zapisuje informację, dane wektorowe oraz event w lokalnej bazie danych (5A) oraz próbuje natychmiast poinformować Master Cloud Application o zaistniałym evencie, wykorzystując kanał komunikacji Google Cloud Pub Sub (7E)
     1. W przypadku próby nieudanej, próba wysyłki eventu do Master Cloud Application będzie się odbywała za pośrednictwem Sync Mechanism (6) w ustalonych odstępach czasowych.
  v. Local Node wysyła powiadomienia w ustalonych zewnętrznych kanałach komunikacji.
  vi. Master Cloud Application otrzymuję informację poprzez Pub/Sub (7E), który w trybie push wpycha informację do Rest API (7A+7F) , następnie informacja trafia do chmurowej instancji bazy danych Cloud SQL (7B).
  vii. Oprócz danych eventowych, Sync Mechanism (6) w ustalonych odstępach czasowych będzie wysyłał do hurtowni danych (9) w trybie batch niezsynchronizowane dane (5) wektorowe jak i informacje o eventach.

- **AUTORYZACJA WYMIANY DANYCH POMIĘDZY WARSTWAMI SYSTEMU**
  i. Cloud -> Node
     1. Cloud w razie potrzeby będzie komunikował się z Local Node (2) tylko poprzez REST API, autoryzacja poprzez token autoryzacyjny + autoryzacja pochodzenia zapytania (request origin) tak by tylko wylistowane adresy mogły się komunikować.
  ii. Node -> Cloud

1. Autoryzacja w oparciu o Service Accounts (https://cloud.google.com/iam/docs/service-accounts) z komponentami chmurowymi (Google Cloud Storage (9), Cloud Pub/Sub (7E) , Cloud Logging + Error reporting (7G))
2. W komunikacji dodatkowej przez REST API autoryzacja poprzez token autoryzacyjny + autoryzacja pochodzenia zapytania (request origin) tak by tylko wylistowane adresy mogły się komunikować.

iii. Node -> Edge
1. autoryzacja poprzez klucz autoryzacyjny przypisany do Local Node (2)

iv. Edge -> Node
1. Komunikacja przez REST API, autoryzacja poprzez klucze autoryzacyjne przypisywane do urządzenia na etapie dodawania urządzeń do Local Node.

- **ODCZYT PARAMETRÓW URZĄDZEŃ WARSTW NIŻSZYCH**
  i. Odczyt parametrów systemowych (np. zużycie dysku, pamięci lub parametrów konfiguracji algorytmów)
  1. Poprzez endpointy w REST API możliwe pobieranie bieżącego stanu Local Node (2) oraz poszczególnych urządzeń EDGE (1)

  ii. Z poziomu Local Node (2) oraz Master Cloud Application(7) jest możliwe pobranie całej konfiguracji Egda w celu edycji i upload-u do urządzenia.
  1. Master Cloud Application (7) poprzez Sync Mechanism (6) jest cały czas na bieżąco informowany o zmianach w konfiguracji na EDGE wywołanych z poziomu Local Node.
  2. Master Cloud Application (7) może poprzez REST API - bezpośredniej komunikacji z Local Node (2) pobrać bieżącą konfigurację, jak i tą samą drogą komunikacji (REST API) wstawić informację o update konfiguracji dowolnego urządzenia EDGE spiętego z Node

  iii. Z poziomu Clouda możliwe jest pobranie konfiguracji Node, edycja i upload do urządzenia
  1. Master Cloud Application (7) poprzez Sync Mechanism (6) jest cały czas na bieżąco informowany o zmianach w konfiguracji.
  2. Cloud może poprzez REST API - bezpośredniej komunikacji z Local Node (2) pobrać bieżącą konfigurację, jak i tą samą drogą komunikacji (REST API) wstawić informację o update konfiguracji Node

- **WYSYŁANIE PRÓBEK**
  i. Rejestracja zdarzenia przez urządzenia EDGE (1)
  ii. Wysyłka przez MQTT informacji do RabbitMQ (4). Informacja trafia następnie do backendu Local Node (2)
  iii. Backend zapisuje informacje w Local Database (5A) jak i w plikach logów. W przypadku wersji Cloud-Connected następuje również zapis w Google Cloud Logging (7G)

iv. Backend podejmuje decyzję (wykrycie zdarzenia) na bazie odebranych informacji, zapisuje informację, dane wektorowe oraz event w Local Database (5A) oraz próbuje natychmiast poinformować Master Cloud Application (7) o zaistniałym evencie, wykorzystując kanał komunikacji Google Cloud Pub/Sub (7E)
    1. W przypadku próby nieudanej, próba wysyłki eventu do Master Cloud Application (7) będzie się odbywała za pośrednictwem Sync Mechanism (6), w ustalonych odstępach czasowych. (6)
v. Master Cloud Application otrzymuję informację poprzez Pub/Sub (7E), który w trybie push wpycha informację do Rest API (7A+7F) , następnie informacja trafia do chmurowej instancji bazy danych Cloud SQL (7B).
vi. Oprócz danych eventowych, Sync Mechanism (6) w ustalonych odstępach czasowych będzie wysyłał do hurtowni danych (9) w trybie batch niezsynchronizowane dane (5) dane wektorowe jak i informacje o eventach.
vii. W przypadku konfiguracji bez połączenia z Master Cloud Application, dane są odkładane na Local Node(2) w Local Database(5) i nie są podejmowane próby dostarczania do chmury.

- **DOTRENOWANIE MODELU I WYPCHNIĘCIE GO NA NODE**
    i. Dane trafiają do Data Warehouse (11) poprzez mechanizmy Data Ingestion (9) oraz opcjonalnym przeprocesowaniu (Data Processing (10))
    ii. Następnie automatycznie (w określonych interwałach czasowych) lub manualnie z poziomu Master Cloud Application (7) model jest trenowany w chmurowym środowisku ML z użyciem Vertex AI (12B)
    iii. Po skończonym treningu pliku modelu są zapisywane na Google Cloud Storage (9A) oraz powstaje informacja do przekazania do Master Cloud Application (7) w kolejce wiadomości Cloud Pub/Sub (14)
    iv. Po otrzymaniu informacji, Master Cloud Application (7) zapisuje w bazie danych (7B) informację, że model jest gotowy oraz jego lokalizację na Google Cloud Storage (9A)
    v. Użytkownik Master Cloud Application (7), może z poziomu panelu zarządzania zdecydować o aktualizacji modelu w wybranym środowisku Local Node (2)
    vi. Backend Master Cloud Application (7) Komunikuje się poprzez REST API z Local Node (2), wstawiając informację o nowym modelu (jego lokalizacji w Cloud Storage (9A)) i chęci aktualizacji. Następnie Local Node (2) ściąga z określonej lokalizacji nową wersję modelu, podmienia go, i zwraca informację o tym, że podmiana modelu się odbyła pomyślnie.
    vii. Następuje przeładowanie modelu ML.
- **UPDATE-Y OPROGRAMOWANIA**
    i. Cloud -> Node
        1. Update oprogramowania Local Node (2)będzie polegał na ściągnięciu najnowszej wersji obrazu aplikacji z chmury (lub z dowolnego wskazanego miejsca np. inna wersja obrazu

umieszczona np na Google Cloud Storage), wdrożeniu i restarcie. Inicjacja update może odbywać się przez REST API.

2. Użytkownik w panelu Master Cloud Application (7) znajduje na liście na dashboardzie Local Node (2) który chce zupdatować o najnowszą wersję aplikacji.
3. Backend Master Cloud Application (7) poprzez REST API komunikuje się z Local Node. (4+2A/2B)
4. Local Node (2) startuje proces w systemie (skrypt bash), który zbuduje nową wersję aplikacji Local Node (2).

ii. Cloud -> Edge

1. Master Cloud Application (7) nie będzie posiadał bezpośredniej komunikacji z Edge (1), gdyż Edge nie będzie miał połączenia z internetem, jedynie połączenie z Local Node (2). Update Edge (1) będzie się odbywał zawsze za pośrednictwem Local Node (2).
2. W tym celu użytkownik w panelu Master Cloud Application (7) znajduje Edge (1) w którym chce wymusić update oprogramowania. Backend poprzez REST API wysyła informację do Local Node o chęci aktualizacji oprogramowania Edge.
3. Całość dalej opisana w kroku poniżej.

iii. Node -> Edge

1. Update Edge rozpoczyna się zawsze z poziomu Local Node (2) - niezależnie czy nastąpi z poziomu Master Cloud Application (7) czy z poziomu Local Node.
2. Local Node (2) ściąga z określonej lokalizacji w Google Cloud Storage (9A) paczkę z nowym oprogramowaniem Edge (1).
3. Na Edge (1) powstanie REST API do komunikacji, które przyjmuje paczkę z nowym oprogramowaniem i podmienia pliki i następnie restartuje Edge.
4. Local Node (2) wypycha paczkę ZIP z oprogramowaniem pod w/w REST API Edge (1).