Wojciech Kaliszewski

Big data, inżynieria i analiza danych z wykorzystaniem języka Python

Grupa 2 Rok 2024/2025

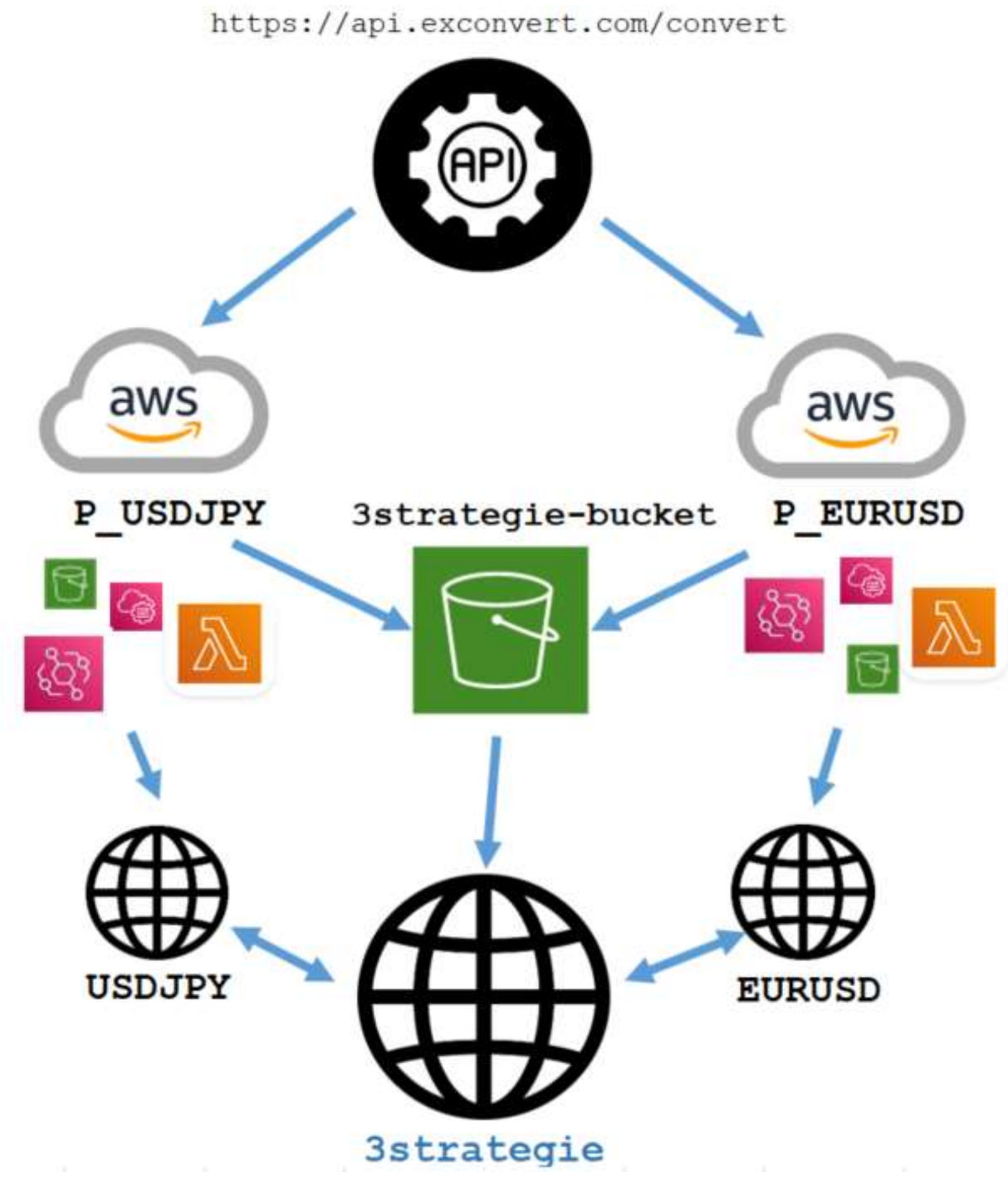
Projekt ‘3strategie’:

Symulacja i Wizualizacja Strategii Walutowych na AWS

Poniższy projekt wykorzystując usługi AWS pobiera kursy walut, przetwarza je i następnie symuluje grę trzema różnymi strategiami dla par walutowych EURUSD oraz USDJPY Ostatecznym efektem projektu jest dynamiczna strona internetowa hostowana na AWS S3, prezentująca wyniki symulacji strategii transakcyjnych.. Każda z par walutowych to oddzielny subprojekt opierający się na różnych rozwiązaniach (dalej nazywane P_EURUSD i P_USDJPY). Ich architektura i działanie zostaną opisane w dalszej części. W pierwszej kolejności pokazano ogólną architekturę od pobrania api do hostowania efektu 3strategie.html.

1. Schemat działania projektu:

Dane wejściowe pochodzące z api serwującego notowania walutowe są zachowane i przetwarzane przy pomocy bazy danych (P_USDJPY) lub bucketu S3 (P_USDJPY) oraz funkcji lambda (usługi AWS). Efekty analiz są przedstawiane na głównej stronie „3strategie" oraz podstronach EURUSD i USDJPY. Poniżej diagram architektury:



2. Działanie krok po kroku

1. **Harmonogram (AWS EventBridge):** Co 5 minut EventBridge wyzwala AWS Lambda.
2. **Pobieranie danych i analiza (AWS Lambda):**
    - Za pomocą projektów P_EURUSD i P_USDJPY pobierane są notowania z zewnętrznego API walutowego. Klucz API jest bezpiecznie przechowywany w **AWS Systems Manager Parameter Store**.
    - Funkcje lambda analizują świeżo pobrane dane i symulują działanie trzech strategii transakcyjnych dla każdej pary, śledząc otwarcie/zamknięcie pozycji i ich wyniki.
    - Wygenerowane wykresy strategii w postaci HTML są przesyłane do **bucketa S3** i następnie są pokazywane na stronie głównej „3strategie”
3. **Hosting strony („3strategie”):** Wygenerowane wykresy strategii w postaci HTML są przesyłane do **bucketa S3** i następnie dodawane do strony głównej „3strategie” Bucket S3 jest skonfigurowany jako statyczny hosting strony internetowej. Pliki HTML są publicznie dostępne.
4. **Podstrony USDJPY oraz EURUSD** Poza wykresami na główną stronę projekty P_EURUSD oraz P_USDJPY generują swoje strony EURUSD (za pomocą function URL w usłudze lambda) i USDJPY (za pomocą hostingu statycznej strony html z bucketu S3.

3. Schemat trzech stron HTML

Projekt składa się z głównej strony podsumowującej i dwóch podstron szczegółowych dla każdej pary walutowej.

- `summary_dashboard.html` **(Strona Główna)**
    - Prezentuje zagregowane podsumowanie wyników wszystkich strategii dla obu par walutowych (EURUSD i USDJPY).
    - Zawiera wykresy PnL (Profit and Loss) dla każdej strategii/pary.
    - Posiada linki nawigacyjne do szczegółowych podstron.

- `eurusd_dashboard.html` **(Podstrona EUR/USD – generowana przez P_EURUSD)**
  - Zawiera wykres notowań z 15 okresów dla pary EURUSD
  - Zawiera szczegółowe dane dotyczące wyników trzech strategii dla pary EURUSD (wykresy i tabele prezentujące wyniki każdej strategii)
  - Pozwala nawigować do strony głównej i dalej do podstrony USDJPY.
- `usdjpy_dashboard.html` **(Podstrona USD/JPY– generowana przez P_USDJPY)**
  - Zawiera wykres notowań z 15 okresów dla pary USDJPY
  - Zawiera szczegółowe dane dotyczące wyników trzech strategii dla pary USDJPY (wykresy i tabele prezentujące wyniki każdej strategii)
  - Pozwala nawigować do strony głównej i dalej do podstrony EURUSD.

4. Ocena szybkości i niezawodności projektu

- **Szybkość:**
  - **Generowanie danych/wykresów:** Funkcje Lambda są bardzo szybkie. Czas trwania funkcji (około 200-400 ms) jest bardzo krótki. Aktualizacja co 5 minut zapewnia bieżące dane.
  - **Dostarczanie strony:** S3 jako statyczny hosting jest szybki. Pliki są serwowane bezpośrednio z najbliższego punktu brzegowego AWS.
- **Niezawodność:**
  - **Wysoka dostępność:** Wszystkie użyte usługi AWS (Lambda, S3, EventBridge, Parameter Store) są z wysoce dostępne i odporne na awarie, ponieważ są zarządzane przez AWS i działają w wielu strefach dostępności.
  - **Odporność na błędy:** Kod Lambda jest wyposażony w obsługę błędów (`try-except`). Jeśli API zewnętrzne nie odpowie, funkcja zaloguje błąd i strona nadal będzie działać, pokazując ostatnie pomyślne dane.
  - **Skalowalność:** Rozwiązanie jest z natury skalowalne. Lambda automatycznie skaluje się, aby obsłużyć więcej wywołań.. S3 radzi sobie z dużą ilością ruchu.
  - **Ograniczenia:** Potencjalne problemy mogą wynikać z limitów zewnętrznego API walutowego lub sporadycznych "cold startów" Lambda (choć przy wywoływaniu co 5 minut, powinny być rzadkie).

5. Oszacowanie kosztów projektu (miesięcznie)

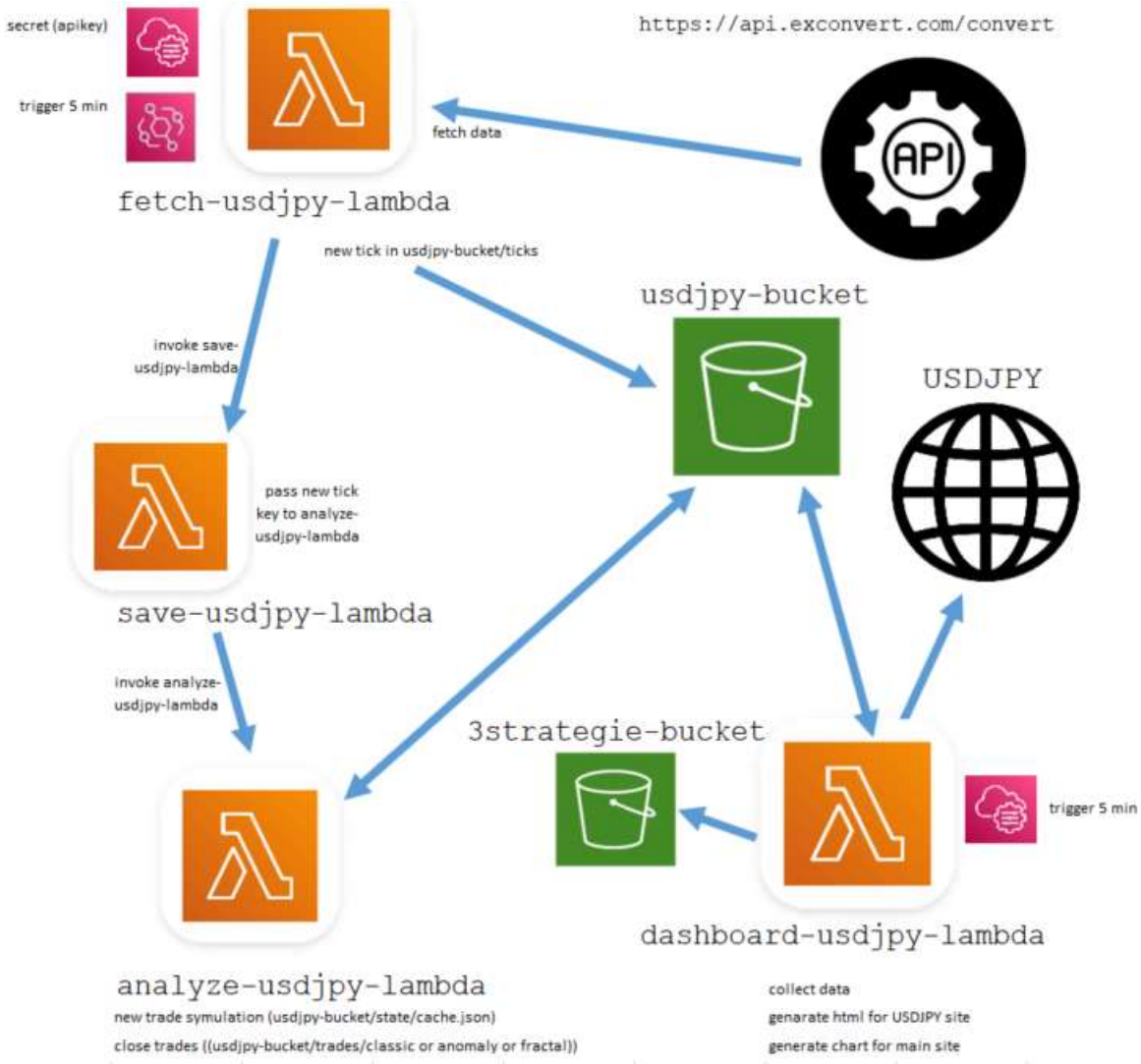
To rozwiązanie można zaliczyć do kategorii niskobudżetowych, ponieważ opiera się na obszernych pakietach **AWS Free Tier**. Aktualizacja i hostowanie wyników analizy nie powinna generować żadnego kosztu.

P_USDJPY

Projekt analizy strategii Forex dla USD/JPY (Architektura Serverless S3)

1. Przegląd Projektu

Projekt jest zautomatyzowanym systemem serverless do analizy kursu walutowego USD/JPY. System cyklicznie pobiera aktualne notowania, zapisuje je jako pliki JSON w buckecie S3, analizuje zgromadzone dane poprzez symulację trzech różnych strategii transakcyjnych, a następnie generuje i publikuje interaktywne pulpity nawigacyjne (dashboardy) w formie stron HTML, również przechowywanych w S3. Projekt nie wykorzystuje bazy danych ani konfiguracji VPC dla funkcji Lambda. Poniżej diagram architektury.



2. Działanie Krok po Kroku

System działa w oparciu o sekwencję zdarzeń i wywołań funkcji Lambda:

1. **Pobieranie Danych (`fetch-usdjpy-lambda`)**:
    - Funkcja AWS Lambda `fetch-usdjpy-lambda` jest cyklicznie uruchamiana przez Amazon EventBridge (CloudWatch Events) zgodnie z harmonogramem
    - Funkcja pobiera klucz API z AWS Systems Manager Parameter Store.
    - Wysyła zapytanie do zewnętrznego API (`https://api.exconvert.com/convert`) w celu uzyskania aktualnego kursu USD/JPY.
    - Pobrany kurs wraz z aktualnym znacznikiem czasu (UTC) jest zapisywany jako nowy plik JSON w buckecie S3 (`usdjpy-bucket`) w folderze `ticks/`. Nazwa pliku zawiera timestamp.
    - Funkcja aktualizuje również plik `state/cache.json` w tym samym buckecie, przechowujący listę kluczy do ostatnich 500 plików z notowaniami.
2. **Inicjacja Analizy (`save-usdjpy-lambda`)**:
    - Utworzenie nowego obiektu (pliku z notowaniem) w folderze `raw/usdjpy/` bucketa `usdjpy-bucket` automatycznie wyzwala funkcję Lambda `save-usdjpy-lambda`.
    - Głównym zadaniem tej funkcji jest asynchroniczne wywołanie (InvocationType: 'Event') kolejnej funkcji Lambda – `analyze-usdjpy-lambda`. Wywołanie odbywa się w osobnym wątku, aby `save-usdjpy-lambda` mogła szybko zakończyć działanie i przekazanie jej klucza (nazwa pliku) nowo utworzonego ticka.
3. **Analiza Danych i Symulacja Strategii (`analyze-usdjpy-lambda`)**:
    - Funkcja `analyze-usdjpy-lambda` jest wywoływana asynchronicznie przez `save-usdjpy-lambda` lub cyklicznie przez EventBridge (cron(1/5 * * * ? *)).
    - Odczytuje najnowszy plik z notowaniem (tick) oraz listę ostatnich notowań (do 100) z `state/cache.json` i odpowiednich plików w folderze `ticks/` w buckecie S3 (`usdjpy-bucket`).

- Na podstawie tych danych symuluje działanie trzech strategii handlowych (opisanych poniżej). Stan każdej otwartej pozycji (np. cena otwarcia, SL, TP) jest przechowywany w osobnym pliku JSON w folderze `state/` (np. `state/classic.json`).
- Zamknięte transakcje (wyniki) są zapisywane jako indywidualne pliki JSON w dedykowanych podfolderach w `trades/` (np. `trades/classic/`).

4. **Generowanie i Prezentacja Wyników (`dashboard-usdjpy-lambda`)**:
   - Funkcja `dashboard-usdjpy-lambda` jest cyklicznie uruchamiana przez EventBridge (cron(2/5 * * * ? *)).
   - Zbiera dane o ostatnich notowaniach (z `ticks/`) oraz wyniki wszystkich zamkniętych transakcji (z `trades/`) z bucketa S3 `usdjpy-bucket`.
   - Generuje dwie strony HTML:
     - Główny pulpit nawigacyjny USD/JPY (`index.html`) zapisywany jest w buckecie usdjpy-bucket. Zawiera wykres ostatnich notowań USD/JPY, zbiorczy wykres PnL (Profit and Loss) dla wszystkich strategii oraz szczegółowe tabele transakcji dla każdej z nich.
     - Wykres PnL (`usdjpy_pnl_chart_only.html`) zapisywany w buckecie "3strategie". Jest to uproszczona strona zawierająca wyłącznie wykres PnL, przeznaczona do osadzania.
   - Funkcja może również wysyłać powiadomienia email przez SES w przypadku wystąpienia określonych alertów (np. 3 wygrane/przegrane z rzędu – obecnie stosowane).

3. Kluczowe Komponenty Architektury

- **AWS Lambda**:
  - `fetch-usdjpy-lambda`: Pobiera kursy walut i zapisuje je do S3.
  - `save-usdjpy-lambda`: Wyzwalana przez S3, asynchronicznie uruchamia analizę.
  - `analyze-usdjpy-lambda`: Analizuje dane, symuluje strategie, zapisuje stany i wyniki transakcji do S3.

    - `dashboard-usdjpy-lambda`: Generuje i zapisuje strony HTML z wynikami do S3, obsługuje alerty email.
- **Amazon S3**:
    - Bucket `usdjpy-bucket`: Główny bucket roboczy, przechowuje surowe notowania (`ticks/`), stany otwartych pozycji (`state/`), historię zamkniętych transakcji (`trades/`) oraz listę notowań (`state/cache.json`). Część jego zawartości (`index.html`) jest publicznie dostępna.
    - Bucket `3strategie`: do generowania wykresu na stronę „3strategie” (`usdjpy_pnl_chart_only.html`).
- **Amazon EventBridge (CloudWatch Events)**: Używany do planowania cyklicznego uruchamiania funkcji `fetch-usdjpy-lambda`, i `dashboard-usdjpy-lambda`.
- **AWS Systems Manager Parameter Store**: Przechowuje bezpiecznie klucz API (`/currency-db/apikey`).
- **IAM (Identity and Access Management)**: Role IAM definiują uprawnienia dla funkcji Lambda do interakcji z S3, SSM, CloudWatch Logs, SES i innymi funkcjami Lambda.
- **Amazon SES (Simple Email Service)**: Używany przez `dashboard-usdjpy-lambda` do wysyłania alertów email.

4. Zaimplementowane Strategie Handlowe (w `analyze-usdjpy-lambda`)

Projekt symuluje trzy różne strategie forex dla USD/JPY:

1. **Strategia Klasyczna (RSI)**:
    - Sygnał KUPNA (Long): RSI(14) < 30.
    - Sygnał SPRZEDAŻY (Short): RSI(14) > 70.
    - Stop Loss (SL): 20 pipsów, Take Profit (TP): 30 pipsów.
2. **Strategia Anomalii (Z-score)**:
    - Sygnał KUPNA: Z-score logarytmicznych stóp zwrotu (z 50 okresów) <= -2.5.
    - Sygnał SPRZEDAŻY: Z-score >= +2.5.
    - SL: 15 pipsów, TP: 25 pipsów.
3. **Strategia Fraktal + SMA**:

- Sygnał KUPNA: Uformowanie niskiego fraktala (minimum z 5 świec, gdzie środkowa jest najniższa) ORAZ aktualna cena powyżej 50-okresowej prostej średniej kroczącej (SMA50).
- Sygnał SPRZEDAŻY: Uformowanie wysokiego fraktala (maksimum z 5 świec, gdzie środkowa jest najwyższa) ORAZ aktualna cena poniżej SMA50.
- SL: 12 pipsów, TP: 24 pipsy.

5. Prezentacja Wyników (Wykresy i Tabele)

Wygenerowane strony HTML (`index.html` w buckecie `3strategie`, generowane przez `dashboard-usdjpy-lambda`) zawierają:

- **Wykres Kursu USD/JPY**: Liniowy wykres przedstawiający ostatnie 15 notowań kursu, z etykietami czasu (HH:MM) na osi X i wartością kursu na osi Y (z dokładnością do 3 miejsc po przecinku).
- **Wykres PnL (Zysku/Straty)**: Liniowy wykres zbiorczy pokazujący skumulowany zysk/stratę (w pipsach) dla każdej z trzech strategii w funkcji czasu (data na osi X). Pozwala na wizualne porównanie efektywności strategii.
- **Tabele Transakcji**: Dla każdej strategii generowana jest osobna tabela zawierająca historię ostatnich transakcji (do 10 wyświetlanych, pobierane do 300). Kolumny w tabeli to: Czas Otwarcia, Cena Otwarcia, Kierunek (Long/Short), Poziom Stop Loss, Poziom Take Profit, Cena Zamknięcia, Wynik w Pipsach.

Druga strona HTML (`usdjpy_pnl_chart_only.html`) zawiera wyłącznie powyższy wykres PnL.

6. Ocena Szybkości i Niezawodności Projektu

- **Szybkość**:
  - **Pobieranie Danych**: Czas odpowiedzi zewnętrznego API (`api.exconvert.com`) jest kluczowy; funkcja `fetch-usdjpy-lambda`
  - **Funkcje Lambda**: Czasy wykonania funkcji Lambda są generalnie krótkie. Wykorzystanie pamięci 128MB jest minimalne. Asynchroniczne wywołanie

`analyze-usdjpy-lambda` przez `save-usdjpy-lambda` poprawia responsywność tej drugiej.

- **Operacje S3**: Odczyt/zapis małych plików JSON do S3 jest bardzo szybki. Odczyt wielu plików (np. 100 cen w `analyze-usdjpy-lambda`) może wprowadzić pewne opóźnienie, ale powinno być akceptowalne.

- **Niezawodność**:
  - **Zależności Zewnętrzne**: Projekt jest zależny od dostępności API `api.exconvert.com`.
  - **Obsługa Błędów**: Funkcje Lambda implementują logowanie błędów. `fetch-usdjpy-lambda` aktualizuje cache nawet przy błędzie pobrania z S3 (tworzy nowy cache). `analyze-usdjpy-lambda` zwraca błędy, jeśli tick lub cache są niedostępne.
  - **Usługi AWS**: Wysoka dostępność usług AWS (Lambda, S3, EventBridge, SSM, SES) stanowi podstawę niezawodności.
  - **Brak Tradycyjnej Bazy Danych**: Wykorzystanie S3 jako głównego magazynu danych upraszcza architekturę i eliminuje potrzebę zarządzania serwerem bazodanowym, ale transakcyjność i spójność danych między różnymi plikami (np. stan pozycji, logi transakcji) muszą być starannie zarządzane na poziomie logiki aplikacji.
  - **Limity AWS**: Należy monitorować limity dotyczące liczby obiektów S3, wywołań Lambda itp., choć dla tego projektu prawdopodobnie nie zostaną szybko osiągnięte.

7. Szacunkowe Koszty Projektu

Koszty będą generowane przez następujące usługi AWS (szacunki są przybliżone i zależą od rzeczywistego użycia):

- **AWS Lambda**:
  - Cztery funkcje Lambda (każda 128MB), uruchamiane co 5 minut (ok. 8640 razy miesięcznie każda), łaczna liczba wywołań to ok. 34 560 miesięcznie. Czas wykonania jest krótki.

Darmowy pakiet AWS Lambda (1 milion żądań i 400 000 GB-sekund miesięcznie) najprawdopodobniej pokryje całość zapotrzebowania, skutkując minimalnymi lub zerowymi kosztami.

- **Amazon S3**:
    - Darmowy limit S3 (np. 5GB standardowego przechowywania, 20 000 żądań GET, 2 000 żądań PUT miesięcznie) prawdopodobnie pokryje większość potrzeb. Koszty, jeśli wystąpią, będą minimalne (szacunkowo 0,5-0,7 USD).
- **Amazon EventBridge**: Koszty będą bardzo niskie lub zerowe, mieszcząc się w darmowym limicie.
- **AWS Systems Manager Parameter Store**: Pobieranie parametru `API_KEY`. Standardowe parametry są darmowe.
- **Amazon SES**: Koszty wysyłki e-maili (jeśli alerty są częste). Istnieje darmowy limit (np. 62 000 e-maili miesięcznie wysyłanych z EC2 lub Lambda).
- **CloudWatch Logs**: Przechowywanie logów. Darmowy limit (5GB/miesiąc) powinien być wystarczający.
- **Transfer Danych**: Może generować koszty, jeśli użytkownicy często pobierają strony HTML z S3 (publiczny dostęp).

Całkowity szacowany koszt miesięczny: Prawdopodobnie bardzo niski, potencjalnie **bliski $0-$1 USD**, jeśli użycie zmieści się w darmowych limitach AWS.

8. Potencjalne Problemy z Poprawnym Działaniem Projektu

- **Niezawodność API Zewnętrznego**: Jeśli `api.exconvert.com` będzie niedostępne, zwróci błędne dane lub zmieni format odpowiedzi, funkcja `fetch-usdjpy-lambda` zawiedzie.
- **Spójność Danych w S3**: Może wystąpić sytuacja, w której np. plik ticka zostanie zapisany, ale aktualizacja `cache.json` się nie powiedzie, prowadząc do niespójności. Ważny aspekt przy dalszym rozwijaniu projektu i odporności na takie scenariusz.
- **Obsługa Błędów i Ponowień**:
    - Asynchroniczne wywołanie `analyze-usdjpy-lambda` oznacza, że `save-usdjpy-lambda` nie wie, czy analiza zakończyła się sukcesem. W przypadku błędu w `analyze-usdjpy-lambda`, problem może nie zostać od razu

zauważony bez odpowiedniego monitoringu logów lub konfiguracji Dead Letter Queue (DLQ) dla funkcji Lambda.

- **Zarządzanie Stanem Strategii**: Przechowywanie stanu otwartej pozycji w pojedynczym pliku JSON w S3 jest proste, ale w przypadku uszkodzenia tego pliku może uniemożliwić poprawne zarządzanie pozycją. Trudno oszacować prawdopodobieńswo wystąpienia takiego zdarzenia.

9. Dodatkowe Ważne Aspekty Projektu

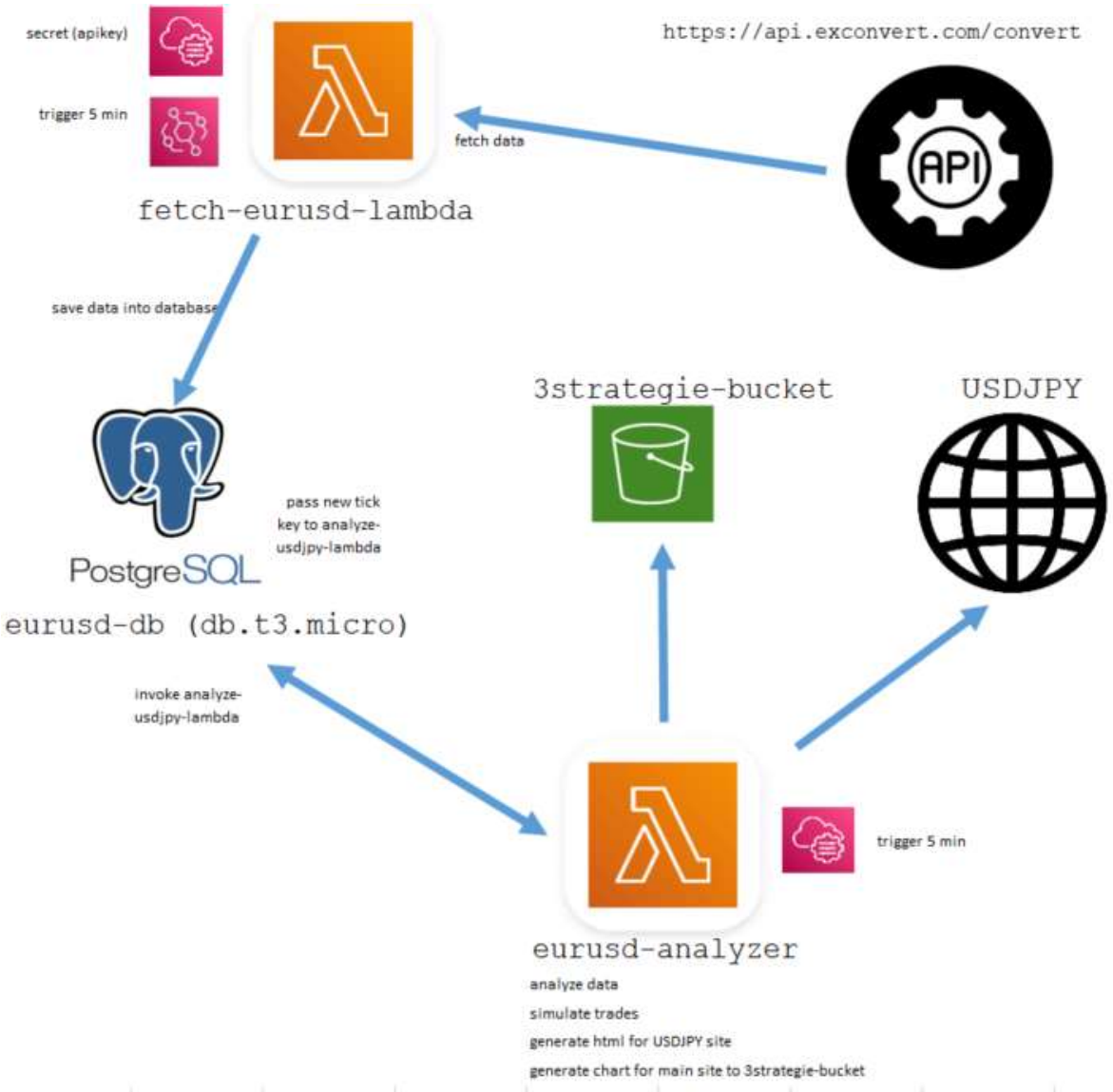
- **Architektura Serverless**: Całkowite oparcie na Lambda i S3 (bez RDS i VPC) znacząco upraszcza zarządzanie infrastrukturą i potencjalnie obniża koszty.
- **Bezpieczeństwo**:
  - Klucz API jest bezpiecznie przechowywany w SSM Parameter Store i pobierany z deszyfrowaniem.
  - Uprawnienia IAM są zdefiniowane dla funkcji Lambda.
  - Bucket `3strategie` oraz częściowo `usdjpy-bucket` są publicznie dostępne, co jest zamierzone dla dashboardów, ale wymaga świadomej konfiguracji.
- **Konwersja Czasu**: Funkcja `dashboard-usdjpy-lambda` zawiera logikę do konwersji czasu UTC na CEST dla wyświetlania, co jest ważne dla czytelności dashboardu dla użytkownika w tej strefie czasowej. Należy jednak pamiętać o ograniczeniach ręcznej implementacji obsługi stref czasowych, szczególnie w kontekście zmian czasu letniego/zimowego.

P_EUR/USD

Projekt analizy strategii Forex dla EUR/USD (PostgreSQL, VPC)

1. Przegląd Projektu

Projekt jest zautomatyzowanym systemem do analizy kursu walutowego EUR/USD, symulacji strategii handlowych i prezentacji wyników. System cyklicznie pobiera aktualne notowania, zapisuje je w bazie danych, przeprowadza analizy oparte na trzech różnych strategiach transakcyjnych, a następnie generuje i publikuje interaktywne pulpity nawigacyjne (dashboardy) w formie stron HTML. Poniżej diagram architektury:



2. Działanie Krok po Kroku

System działa w oparciu o zaplanowane zadania i interakcje między komponentami chmurowymi:

1. **Automatyczne Pobieranie Danych (`fetch_eurusd_lambda`)**:
    - Co 5 minut, usługa Amazon EventBridge (CloudWatch Events) uruchamia funkcję.
    - Funkcja wysyła zapytanie do zewnętrznego API (`https://api.exconvert.com/convert`) w celu uzyskania aktualnego kursu EUR/USD. Zapytanie jest autoryzowane za pomocą klucza API przechowywanego w Parameter Store.
    - Pobrany kurs wraz z aktualnym znacznikiem czasu (w strefie UTC) jest zapisywany w tabeli `eurusd_rates` w bazie danych PostgreSQL.
2. **Przechowywanie Danych (Baza Danych PostgreSQL)**:
    - Dane historyczne kursów oraz wyniki transakcji poszczególnych strategii są przechowywane w relacyjnej bazie danych PostgreSQL (`eurusd-db`) działającej na usłudze Amazon RDS. Baza danych jest instancją klasy `db.t3.micro`.
3. **Analiza Danych i Symulacja Strategii (`eurusd-analyzer`)**:
    - Co 5 minut, Amazon EventBridge uruchamia drugą funkcję AWS Lambda, `eurusd-analyzer`. Funkcja ta jest również dostępna publicznie poprzez Function URL.
    - Funkcja pobiera z bazy danych (`eurusd-db`) najnowsze (do 300) historyczne notowania EUR/USD.
    - Na podstawie tych danych, symuluje działanie trzech strategii handlowych (opisanych poniżej). Analiza obejmuje obliczanie wskaźników (np. RSI, Z-score, SMA), identyfikację sygnałów kupna/sprzedaży oraz otwieranie/zamykanie hipotetycznych transakcji. Wyniki transakcji (np. cena otwarcia/zamknięcia, zysk/strata w pipsach) są zapisywane w dedykowanych tabelach w bazie danych (`eurusd_trades`, `eurusd_anom_trades`, `eurusd_frac_trades`).

4. **Generowanie i Prezentacja Wyników (Strony HTML: główna + wykres na S3)**:
    - Po zakończeniu symulacji, funkcja `eurusd-analyzer` generuje dwie strony HTML:
        - Główny pulpit nawigacyjny (`eurusd_dashboard_index.html`): Zawiera wykres ostatnich notowań EUR/USD, zbiorczy wykres PnL (Profit and Loss – Zysk/Strata) dla wszystkich strategii oraz szczegółowe tabele transakcji dla każdej z nich.
        - Wykres PnL (`eurusd_pnl_chart_only.html`): Uproszczona strona zawierająca wyłącznie wykres PnL, przeznaczona do osadzania w innych miejscach (np. na stronie podsumowującej) wysyłany do bucket S3 w celu użycia go przez stronę „3strategie”

3. Kluczowe Komponenty Architektury

- **AWS Lambda**:
    - `fetch_eurusd_lambda`: Funkcja odpowiedzialna za pobieranie kursów walut. Wykorzystuje warstwę Lambda (`psycopg2-layer1`) do obsługi połączeń z PostgreSQL.
    - `eurusd-analyzer`: Funkcja realizująca analizę danych, symulację strategii i generowanie raportów HTML. Znajduje się w konfiguracji VPC, co pozwala jej na dostęp do bazy danych RDS w tej samej sieci. Ma uprawnienia do zapisu logów w CloudWatch, dostępu do VPC, zapisu obiektów w S3 oraz wysyłania e-maili przez SES.
- **Amazon RDS**: Baza danych PostgreSQL (nazwa: `eurusd-db`, klasa: `db.t3.micro`, region: `eu-central-1b`).
- **Amazon S3**: Bucket "3strategie" służący do hostowania wygenerowanych stron HTML z wynikami.
- **Amazon EventBridge (CloudWatch Events)**: Usługa używana do planowania cyklicznego uruchamiania obu funkcji Lambda (co 5 minut).
- **IAM (Identity and Access Management)**: Role IAM definiują uprawnienia dla funkcji Lambda, np. do zapisu logów, interakcji z S3 czy RDS.
-

4. Zaimplementowane Strategie Handlowe

Projekt symuluje trzy różne strategie forex:

1. **Strategia Klasyczna (RSI)**:
    - Sygnał KUPNA (Long): RSI(14) spada poniżej 30.
    - Sygnał SPRZEDAŻY (Short): RSI(14) wzrasta powyżej 70.
    - Stop Loss (SL): 20 pipsów, Take Profit (TP): 30 pipsów.
2. **Strategia Anomalii (Z-score + RSI)**:
    - Sygnał KUPNA: Z-score logarytmicznych stóp zwrotu (z 50 okresów) spada poniżej -2.5 ORAZ RSI(14) < 40.
    - Sygnał SPRZEDAŻY: Z-score wzrasta powyżej +2.5 ORAZ RSI(14) > 60.
    - SL: 15 pipsów, TP: 25 pipsów.
3. **Strategia Fraktal + SMA**:
    - Sygnał KUPNA: Uformowanie niskiego fraktala (minimum z 5 świec, gdzie środkowa jest najniższa) ORAZ aktualna cena powyżej 50-okresowej prostej średniej kroczącej (SMA50).
    - Sygnał SPRZEDAŻY: Uformowanie wysokiego fraktala (maksimum z 5 świec, gdzie środkowa jest najwyższa) ORAZ aktualna cena poniżej SMA50.
    - SL: 12 pipsów, TP: 24 pipsy.

5. Prezentacja Wyników (Wykresy i Tabele)

Wygenerowane strony HTML (`eurusd_dashboard_index.html`) zawierają:

- **Wykres Kursu EUR/USD**: Liniowy wykres przedstawiający ostatnie 15 notowań kursu, z etykietami czasu (HH:MM) na osi X i wartością kursu na osi Y (z dokładnością do 5 miejsc po przecinku).
- **Wykres PnL (Zysku/Straty)**: Liniowy wykres zbiorczy pokazujący skumulowany zysk/stratę (w pipsach) dla każdej z trzech strategii w funkcji czasu (data na osi X). Pozwala na wizualne porównanie efektywności strategii.
- **Tabele Transakcji**: Dla każdej strategii generowana jest osobna tabela zawierająca historię ostatnich 100 transakcji. Kolumny w tabeli to: Czas Otwarcia, Cena Otwarcia,

Kierunek (Long/Short), Poziom Stop Loss, Poziom Take Profit, Cena Zamknięcia, Wynik w Pipsach.

Druga strona HTML (`eurusd_pnl_chart_only.html`) zawiera wyłącznie powyższy wykres PnL.

6. Ocena Szybkości i Niezawodności Projektu

- **Szybkość**:
  - **Pobieranie Danych**: Czas odpowiedzi zewnętrznego API (`api.exconvert.com`) jest kluczowy. Funkcja `fetch_eurusd_lambda`
  - **Funkcje Lambda**: Czas wykonania funkcji Lambda jest zwykle krótki, ale może być obarczony "zimnym startem" (cold start), który wydłuża pierwsze uruchomienie po okresie nieaktywności. Uruchamianie co 5 minut powinno minimalizować częste zimne starty.
  - **Baza Danych**: Wydajność `db.t3.micro` jest ograniczona. Dla obecnego zakresu danych (pobieranie do 300 ostatnich notowań, zapisywanie pojedynczych transakcji) powinna być wystarczająca. Wzrost ilości danych lub złożoności zapytań może wymagać optymalizacji (np. indeksów).
  - **Generowanie HTML**: Odbywa się w pamięci funkcji Lambda i jest relatywnie szybkie dla ograniczonej liczby danych (np. 100 transakcji w tabelach).
- **Niezawodność**:
  - **Zależności Zewnętrzne**: Projekt jest zależny od dostępności i poprawności działania API `api.exconvert.com`. Awaria API uniemożliwi pobieranie nowych danych.
  - **Obsługa Błędów**: Obie funkcje Lambda używają bloków `try-except` do przechwytywania i logowania błędów, co zwiększa odporność na nieoczekiwane problemy i ułatwia diagnostykę.
  - **Usługi AWS**: Niezawodność opiera się na wysokiej dostępności usług AWS (Lambda, RDS, S3, EventBridge).
  - **Baza Danych**: Usługa RDS zapewnia automatyczne backupy i możliwość odtworzenia danych.

7. Szacunkowe Koszty Projektu

Koszty będą generowane przez następujące usługi AWS (szacunki są przybliżone):

- **AWS Lambda**:
  - `fetch_eurusd_lambda` i `eurusd-analyzer` obie uruchamiane co 5 minut (ok. 8640 razy miesięcznie każda).
  - AWS oferuje duży darmowy pakiet miesięczny dla Lambda (1 milion żądań i 400 000 GB-sekund). Prawdopodobnie użycie obu funkcji zmieści się w darmowym limicie.
- **Amazon RDS**:
  - Instancja `db.t3.micro` PostgreSQL. Koszt powinien wynieść zero.
  - Koszty przechowywania danych – prawdopodobnie brak.
  - Transfer danych (niewielki, jeśli nie ma dużego ruchu z internetu do bazy).
- **Amazon S3**:
  - Przechowywanie plików HTML (koszt znikomy, pliki są małe).
  - Żądania PUT (przez `eurusd-analyzer` co 5 minut) i GET (przez użytkowników). Darmowy limit dla S3 powinien być wystarczający.
- **Amazon EventBridge**: Zużycie zmieści się się w darmowym limicie.
- **CloudWatch Logs**: Przechowywanie logów. W darmowy limicie (5GB/miesiąc).
- **Transfer Danych**: Może generować koszty, jeśli dane są transferowane poza AWS (np. użytkownicy pobierający strony HTML). Publiczny dostęp do RDS również może generować koszty transferu danych, jeśli ruch jest znaczny.
- **EUC1-PublicIPv4:** praca w usłudze VPC (virtual private cloud) i użycie publicznego adresu IP w tym projekcie generuje 0,15 USD/dzień.

Całkowity szacowany koszt miesięczny: Prawdopodobnie w zakresie **$4-5 USD**, głównie zdominowany przez użycie instancji RDS w połączeniu z VPC.

8. Problem z Powiadomieniami Email

Problem pojawia się, gdy funkcja Lambda w VPC (bez własnego publicznego adresu IP) musi uzyskać dostęp do publicznego internetu, np. aby połączyć się z publicznym endpointem usługi SES. Dwa główne rozwiązania generują koszty:

1. **NAT Gateway**: Umożliwia funkcjom Lambda w prywatnych podsieciach VPC komunikację z internetem. Jest to rozwiązanie w pełni zarządzane i niezawodne, ale generuje stałe koszty godzinowe (ok. $0.045/godz. = ~$32/miesiąc w regionie eu-central-1) plus opłaty za przetworzone dane. Jest to często postrzegane jako "drogie" dla mniejszych projektów.
2. **VPC Endpoint dla SES**: Można utworzyć VPC Interface Endpoint dla usługi SES. Pozwala to na prywatną komunikację między funkcją Lambda w VPC a usługą SES, bez potrzeby wychodzenia do publicznego internetu (i bez potrzeby NAT Gateway *dla tej konkretnej komunikacji*). VPC Interface Endpointy również mają swoje koszty (godzinowe, ok. $0.01/godz. = ~$7.20/miesiąc za endpoint na strefę dostępności, plus opłaty za przetworzone dane). .

W przypadku projektu P_EURUSD zrezygnowano z powiadomień email.

9. Dodatkowe Ważne Aspekty Projektu

- **Bezpieczeństwo**:
  - **Zarządzanie Sekretami**: Klucz API jest w Parameter Store, ale hasło do bazy nie jest (jest bardzo silne). Obecnie jest poza Parameter Store ze względu na ograniczenia w ilość darmowych wywołań umieszczonych tam parametrów.
  - **Uprawnienia IAM**: Projekt wykorzystuje dedykowane role IAM dla funkcji Lambda, co jest dobrą praktyką.
- **Monitorowanie i Logowanie**:
  - Obie funkcje Lambda intensywnie wykorzystują logowanie do CloudWatch Logs, co jest kluczowe dla monitorowania działania i diagnozowania problemów.

- **Zarządzanie Zależnościami**:
  - Biblioteka `psycopg2` jest dostarczana jako warstwa Lambda (`psycopg2-layer1`). Należy dbać o jej aktualizacje.
- **Skalowalność**:
  - **Lambda**: Funkcje Lambda skalują się automatycznie w odpowiedzi na liczbę żądań.
  - **RDS**: Instancja `db.t3.micro` ma ograniczone zasoby. W miarę wzrostu ilości danych lub obciążenia, może być konieczne przeskalowanie do większej instancji.
  - **API Zewnętrzne**: Należy uwzględnić ewentualne limity żądań (rate limiting) narzucane przez API `api.exconvert.com`.