Plik docker-compose.yml

version: '3.8' # Określa wersję Docker Compose

services:

redis:

image: redis:latest # Pobiera najnowszy obraz Redis

container\_name: redis # Nazwa kontenera

ports:

- "6379:6379" # Mapowanie portu Redis (6379)

restart: always # Kontener restartuje się w razie awarii

airflow:

image: apache/airflow:latest # Pobiera najnowszy obraz Airflow

container\_name: airflow # Nazwa kontenera

depends\_on:

- redis # Airflow uruchomi się dopiero po Redisie

environment:

- AIRFLOW\_\_CORE\_\_EXECUTOR=CeleryExecutor # Ustawienie Airflow na używanie Celery

- AIRFLOW\_\_CELERY\_\_BROKER\_URL=redis://redis:6379/0 # Airflow używa Redisa do kolejkowania zadań

- AIRFLOW\_\_CELERY\_\_RESULT\_BACKEND=db+sqlite:///airflow.db # SQLite jako backend dla wyników (można zamienić na PostgreSQL)

- AIRFLOW\_\_METRICS\_\_STATSD\_ON=True # Włączenie metryk Prometheus w Airflow

- AIRFLOW\_\_METRICS\_\_STATSD\_HOST=prometheus # Adres Prometheusa do zbierania metryk

- AIRFLOW\_\_METRICS\_\_STATSD\_PORT=9125 # Port Prometheusa dla metryk

ports:

- "8080:8080" # UI Airflow dostępne na localhost:8080

volumes:

- ./airflow/dags:/opt/airflow/dags # Współdzielenie folderu DAG-ów

restart: always # Kontener restartuje się w razie awarii

command: >

bash -c "airflow db init && airflow webserver & airflow scheduler" # Inicjalizacja bazy danych i uruchomienie webservera oraz schedulera

prometheus:

image: prom/prometheus:latest # Pobiera najnowszy obraz Prometheusa

container\_name: prometheus # Nazwa kontenera

volumes:

- ./prometheus.yml:/etc/prometheus/prometheus.yml # Ładowanie konfiguracji Prometheusa

ports:

- "9090:9090" # UI Prometheusa dostępne na localhost:9090

restart: always # Kontener restartuje się w razie awarii

grafana:

image: grafana/grafana:latest # Pobiera najnowszy obraz Grafany

container\_name: grafana # Nazwa kontenera

depends\_on:

- prometheus # Grafana uruchomi się dopiero po Prometheusie

ports:

- "3000:3000" # UI Grafany dostępne na localhost:3000

restart: always # Kontener restartuje się w razie awarii

Plik prometheus.yml

global:

scrape\_interval: 15s # Metryki będą pobierane co 15 sekund

scrape\_configs:

- job\_name: 'airflow'

metrics\_path: '/admin/metrics'

static\_configs:

- targets: ['airflow:8080'] # Airflow udostępnia metryki na porcie 8080

Struktura projektu:  
Obraz zawierający tekst, zrzut ekranu, Czcionka, czarne

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Uruchomienie pliku docker compose:  
Obraz zawierający tekst, Czcionka, zrzut ekranu, Grafika

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

(Flaga -d uruchamia kontenery w tle)

**Dostęp do aplikacji**:

* **Airflow**: http://localhost:8080
* **Prometheus**: http://localhost:9090
* **Grafana**: http://localhost:3000

Zatrzymanie starych kontenerów utworzonych z pliku compose:  
Obraz zawierający Czcionka, tekst, Grafika, czarne

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Flaga --volumes usuwa stare dane PostgreSQL

GRAFANA logowanie:

login: admin

hasło: admin

AIRFLOW logowanie:  
login: airflow

hasło: airflow

Logowanie się do kontenera Airflow:  
*docker compose exec airflow bash*

Sprawdzenie dostępnych użytkowników:

Obraz zawierający Czcionka, tekst, Grafika, typografia

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Utworzenie użytkownika:

airflow users create \

--username airflow \

--firstname Admin \

--lastname User \

--role Admin \

--email admin@example.com \

--password airflow