??????????????
Apache / BI

Apache Superset

Apache Superset

?????????

Сборка образа

```
FROM apache/superset:latest

USER root

RUN pip install --no-cache-dir \
    psycopg2-binary \
    redis

USER superset
```

Дополнительные драйвера:

БД	Драйвер
MySQL	mysqlclient
MSSQL	pymssql
ClickHouse	clickhouse-connect
Oracle	cx_Oracle

```
docker build -t test/superset6_1 .
```

Но это не поехало. Нужно использовать инструкцию по сборке образа из git.

Делаем Dockerfile и добавляем нужные драйвера. В инструкции предлагается тег master однако с ним не поехало. Указал конкретную цифру версии.

```
# change this to apache/superset:5.0.0 or whatever version you want to build from;
# otherwise the default is the latest commit on GitHub master branch
FROM apache/superset:6.0.0

USER root

# Set environment variable for Playwright
ENV PLAYWRIGHT_BROWSERS_PATH=/usr/local/share/playwright-browsers
```

```
# Install packages using uv into the virtual environment
RUN . /app/.venv/bin/activate && \
    uv pip install \
    # install psycopg2 for using PostgreSQL metadata store - could be a MySQL package if using
that backend:
    psycopg2-binary \
    # add the driver(s) for your data warehouse(s), in this example we're showing for
Microsoft SQL Server:
    #pymssql \
    # package needed for using single-sign on authentication:
    Authlib \
    # openpyxl to be able to upload Excel files
    openpyxl \
    # Pillow for Alerts & Reports to generate PDFs of dashboards
    Pillow \
    # install Playwright for taking screenshots for Alerts & Reports. This assumes the feature
flag PLAYWRIGHT_REPORTS_AND_THUMBNAILS is enabled
    # That feature flag will default to True starting in 6.0.0
    # Playwright works only with Chrome.
    # If you are still using Selenium instead of Playwright, you would instead install here
the selenium package and a headless browser & webdriver
    playwright \
    && playwright install-deps \
    && PLAYWRIGHT_BROWSERS_PATH=/usr/local/share/playwright-browsers playwright install
chromium

# Switch back to the superset user
USER superset

CMD ["/app/docker/entrypoints/run-server.sh"]
```

Собираем образ

docker build -t my-superset:6 .

Скачивается git проекта

```
git clone --depth=1  https://github.com/apache/superset.git
```

URI:

postgresql://testuser:testpass@pgdataout:5432/testdb

python -m pip install psycopg2-binary

Агент

```
import psycopg2
from psycopg2 import sql


def execute_query(query: str, params: tuple = None):
    """
    Выполняет SQL-запрос к PostgreSQL и возвращает результат.

    :param query: SQL-запрос
    :param params: параметры запроса (для безопасной подстановки)
    :return: результат fetchall() или None
    """

    connection = None
    try:
        connection = psycopg2.connect(
            host="localhost",
            port=5432,
            database="your_database",
            user="your_user",
            password="your_password"
        )

        with connection.cursor() as cursor:
            cursor.execute(query, params)

            # Если запрос возвращает данные (SELECT)
```

```
            if cursor.description:
                result = cursor.fetchall()
                return result

            # Если это INSERT / UPDATE / DELETE
            connection.commit()
            return None

    except Exception as e:
        print(f"Ошибка выполнения запроса: {e}")
        if connection:
            connection.rollback()
        raise

    finally:
        if connection:
            connection.close()

rows = execute_query(
    "SELECT id, name FROM users WHERE age > %s",
    (18,)
)

print(rows)

execute_query(
    "INSERT INTO users (name, age) VALUES (%s, %s)",
    ("Ivan", 25)
)
```

HTML обертки

Обертка для прокручивания дашбордов

```
<!DOCTYPE html>
<html>
```

```
<head>
  <meta charset="UTF-8">
  <title>Dashboard Rotation</title>
  <style>
    body { margin: 0; }
    iframe {
      width: 100vw;
      height: 100vh;
      border: none;
    }
  </style>
</head>
<body>

<iframe id="dashboardFrame"></iframe>

<script>
  const dashboards = [
    "http://192.168.1.196/superset/dashboard/1/?standalone=1",
    "http://192.168.1.196/superset/dashboard/2/?standalone=1",
  ];

  let index = 0;
  const frame = document.getElementById("dashboardFrame");

  function rotate() {
    frame.src = dashboards[index];
    index = (index + 1) % dashboards.length;
  }

  rotate();
  setInterval(rotate, 15000); // 15 секунд
</script>

</body>
</html>
```

Обертка для переключения tab

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <title>Superset Dashboard Carousel</title>
  <style>
    body {
      margin: 0;
      background: #111;
    }
    iframe {
      width: 100vw;
      height: 100vh;
      border: none;
    }
  </style>
</head>
<body>

<iframe
  id="supersetFrame"
  src="http://192.168.1.191:8088/superset/dashboard/p/kgA6aQpyVpb/?standalone=1"
></iframe>

<script>
  const ROTATION_INTERVAL = 10000; // 10 секунд

  function startTabCarousel() {
    const iframe = document.getElementById("supersetFrame");

    iframe.onload = function () {
      const iframeDoc = iframe.contentWindow.document;

      setInterval(() => {
        const tabs = iframeDoc.querySelectorAll('[role="tab"]');
        const activeTab = iframeDoc.querySelector('[role="tab"][aria-selected="true"]');

        if (!tabs.length || !activeTab) return;

        const tabArray = Array.from(tabs);
```

```
        const currentIndex = tabArray.indexOf(activeTab);
        const nextIndex = (currentIndex + 1) % tabArray.length;

        tabArray[nextIndex].click();
      }, ROTATION_INTERVAL);
    };
  }

  startTabCarousel();
</script>


</body>
</html>
```

Директории

- mydata
- static
- superset_db_data
- superset_home

docker-compose.yml

```
services:
  superset:
    image: amancevice/superset:6.0.0
    container_name: superset_app_next
    restart: unless-stopped
    depends_on:
      - superset_db
      - redis
    environment:
      SUPERSET_SECRET_KEY:
"hj9uozuQOM2RTilH5Dwfdo7MqcslMaogDYMY6Xo9IyYS_mYqN5QQAsff9oSUToqvCc0GihnARc4IYQTi022rHw"
      SUPERSET_CONFIG_PATH: /etc/superset/superset_config.py
      DATABASE_DB: superset
      DATABASE_HOST: superset_db
```

```
      DATABASE_USER: superset
      DATABASE_PASSWORD: superset
      DATABASE_PORT: 5432
      REDIS_HOST: redis
    volumes:
      - ./superset_home:/app/superset_home
      - ./superset_config.py:/etc/superset/superset_config.py
    command:
      - "sh"
      - "-c"
      - |
    superset fab create-admin \
          --username admin \
          --firstname Admin \
          --lastname User \
          --email admin@example.com \
          --password admin
superset db upgrade &&
        superset init &&
        gunicorn --bind 0.0.0.0:8088 \
              --workers 4 \
              --timeout 120 \
              --limit-request-line 0 \
              --limit-request-field_size 0 \
              'superset.app:create_app()'
    networks:
      - superset_network

  superset_db:
    image: postgres:16
    container_name: superset_metadata_db_next
    restart: unless-stopped
    environment:
      POSTGRES_DB: superset
      POSTGRES_USER: superset
      POSTGRES_PASSWORD: superset
    volumes:
      - ./superset_db_data:/var/lib/postgresql/data
    networks:
      - superset_network
```

```
  redis:
    image: redis:7
    container_name: superset_redis_next
    restart: unless-stopped
    networks:
      - superset_network

  nginx:
    image: nginx:latest
    container_name: superset_nginx_next
    restart: unless-stopped
    ports:
      - "80:80"
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf:ro
      - ./static:/usr/share/nginx/html:ro
    depends_on:
      - superset
    networks:
      - superset_network

  pgdataout:
    image: postgres:16
    container_name: postgres_data_next
    environment:
      POSTGRES_DB: "testdb"
      POSTGRES_USER: "testuser"
      POSTGRES_PASSWORD: "testpass"
      PGDATA: "/var/lib/postgresql/data/pgdata"
    volumes:
      - ./mydata:/var/lib/postgresql/data
    ports:
      - "5433:5432"
    networks:
      - superset_network

networks:
  superset_network:
    driver: bridge
```

После первого запуска кусок

```
        superset fab create-admin \
          --username admin \
          --firstname Admin \
          --lastname User \
          --email admin@example.com \
          --password password
        superset db upgrade &&
        superset init &&
```

нужно убрать.

Или его сразу убрать, однако в первый раз запустить команды

```
docker compose run --rm superset superset db upgrade
docker compose run --rm superset superset fab create-admin ...
docker compose run --rm superset superset init
```

Описание сервисов

superset	Основной сервис
superset_db	Сервис хранения метаданных. Если не будет, то метаданные будут храниться в /var/lib/superset/superset.db При работе через docker после удаления контейнера все настройки будут сброшены.
redis	Кэширующий сервер.
nginx	Распределяющий прокси.
pgdataout	Не нужен для работы superset. Это для хранения данных дашбордов, указывается в виде dataset.

nginx.conf

```
events {}

http {
  upstream superset {
    server superset:8088;
  }

  server {
```

```
    listen 80;

    client_max_body_size 50M;

    # Статические файлы
    location /alterinfo/ {
            alias /usr/share/nginx/html/;
            index index.html;
        }

    location / {
      proxy_pass http://superset;
      proxy_set_header Host $host;
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_set_header X-Forwarded-Proto $scheme;
      proxy_redirect off;
    }

  }
}
```

superset_config.py

```
import os

#SECRET_KEY = os.environ.get("SUPERSET_SECRET_KEY")
SECRET_KEY =
"hj9uozuQOM2RTilH5Dwfdo7MqcslMaogDYMY6Xo9IyYS_mYqN5QQAsff9oSUToqvCc0GihnARc4IYQTi022rHw"

CACHE_CONFIG = {
    "CACHE_TYPE": "RedisCache",
    "CACHE_DEFAULT_TIMEOUT": 300,
    "CACHE_REDIS_HOST": "redis",
    "CACHE_REDIS_PORT": 6379,
}

CELERY_CONFIG = {
    "broker_url": "redis://redis:6379/0",
    "result_backend": "redis://redis:6379/0",
```

```
}

SQLALCHEMY_DATABASE_URI = (
    "postgresql+psycopg2://"
    "superset:superset@superset_db:5432/superset"
)

# Использование html в компоненте markdown
ESCAPE_MARKDOWN_HTML = False
HTML_SANITIZATION = False

# Обновление конфигурации безопасности в части источников подгрузки изображений. Добавлено
https
from superset.config import TALISMAN_CONFIG as TALISMAN_DEFAULTS
TALISMAN_CONFIG = TALISMAN_DEFAULTS.copy()
csp = TALISMAN_CONFIG.get("content_security_policy", {}).copy()
img_src = csp.get("img-src", [])
img_src += [
    "https:",
]
csp["img-src"] = img_src
TALISMAN_CONFIG["content_security_policy"] = csp
```

Базовый compose для старта в режиме разработки:

```
services:
  superset:
    image: apache/superset:latest
    container_name: superset
    restart: unless-stopped
    ports:
      - "8088:8088"
    environment:
      - SUPERSET_SECRET_KEY=change_key_here
    volumes:
      - ./superset_data:/app/superset_home
    command:
      - "sh"
      - "-c"
      - |
        superset fab create-admin \
          --username admin \
          --firstname Admin \
          --lastname User \
          --email admin@example.com \
          --password admin && \
        superset db upgrade && \
        superset init && \
        superset run -p 8088 --with-threads --reload --debugger --host 0.0.0.0
```

Но в стандартном не оказалось драйверов, странно...

Kafka

Kafka: ????????

Сообщение

- Ключ - метаданные для управления записью сообщения в разделы
- Схема - способ разобрать сообщение. Может быть JSON, XML, Apache Avro
- Смещение - точка считывания, добавляется Kafka

Тема (топик) - раздел - сообщение. Упорядочиваются в пределах раздела.

Производители

Потребители

Могут объединяться в группы. Чтение каждого раздела только одним членом группы. Принадлежность - какой раздел какому потребителю.

Брокер

Отдельный сервер Kafka - брокер. Объединяются в кластер, один из брокеров - контроллер. Если раздел нескольким брокерам, то происходит репликация. Основной брокер - ведущий. Механизмы репликации только в пределах одного кластера. MirrorMaker - репликация между кластерами.

broker.id	Целочисленный идентификатор, с 0, уникальный.
listeners	протокол://имя_хоста:порт Перечисляются через запятую. Имя хоста: конкретный ip - соответствующий интерфейс, 0.0.0.0 - все интерфейсы, не указан - интерфейс по умолчанию. Протокол: PLAINTEXT, SSL Если порт менее 1024 то Kafka от имени root,
log.dirs	Директории размещения логов.

Python: kafka-python

Kafka

?????????

Kafka с web интерфейсом:

```
services:
  kafka:
    image: apache/kafka:4.2.0
    container_name: kafka
    hostname: kafka
    restart: unless-stopped

    ports:
      - "9092:9092"   # INTERNAL (docker network)
      - "29092:29092" # EXTERNAL (LAN access)

    environment:
      # =========================
      # KRaft
      # =========================
      KAFKA_NODE_ID: 1
      KAFKA_PROCESS_ROLES: broker,controller
      KAFKA_CONTROLLER_QUORUM_VOTERS: 1@kafka:9093

      # =========================
      # Listeners
      # =========================
      KAFKA_LISTENERS: INTERNAL://:9092,EXTERNAL://:29092,CONTROLLER://:9093
      KAFKA_ADVERTISED_LISTENERS: INTERNAL://kafka:9092,EXTERNAL://192.168.1.195:29092
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP:
INTERNAL:PLAINTEXT,EXTERNAL:PLAINTEXT,CONTROLLER:PLAINTEXT
      KAFKA_INTER_BROKER_LISTENER_NAME: INTERNAL
      KAFKA_CONTROLLER_LISTENER_NAMES: CONTROLLER

      # =========================
      # Single-node safety
      # =========================
```

```
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
      KAFKA_TRANSACTION_STATE_LOG_REPLICATION_FACTOR: 1
      KAFKA_TRANSACTION_STATE_LOG_MIN_ISR: 1
      KAFKA_GROUP_INITIAL_REBALANCE_DELAY_MS: 0

      # Optional tuning
      KAFKA_AUTO_CREATE_TOPICS_ENABLE: "true"

    volumes:
      - ./kafka_data:/var/lib/kafka/data

  kafka-ui:
    image: provectuslabs/kafka-ui:latest
    container_name: kafka-ui
    restart: unless-stopped

    ports:
      - "8080:8080"

    environment:
      KAFKA_CLUSTERS_0_NAME: local
      KAFKA_CLUSTERS_0_BOOTSTRAPSERVERS: kafka:9092

    depends_on:
      - kafka
```

На https://kafka.apache.org/quickstart/ есть информация по запуску без docker

Проверка внутренними средствами

Создание темы

```
docker exec -it kafka /opt/kafka/bin/kafka-topics.sh \
  --create \
  --topic demo \
  --bootstrap-server kafka:9092 \
  --partitions 3 \
  --replication-factor 1
```

Запуск продюсера (сервер на 1.120)

```
docker exec -it kafka /opt/kafka/bin/kafka-console-producer.sh \
  --topic demo \
  --bootstrap-server 192.168.1.120:29092
```

Запуск консъюмера (сервер на 1.120)

```
docker exec -it kafka /opt/kafka/bin/kafka-console-consumer.sh \
  --topic demo \
  --from-beginning \
  --bootstrap-server 192.168.1.120:29092
```

.

Yandex DataLens

Yandex DataLens

?????????? ? ??????????

Установка

Склонировать git

```
git clone https://github.com/datalens-tech/datalens
```

Настраиваем файл настройки .env

```
DATALENS_BASE_URL=http://ipdockerserver:8080
FORCE_REBUILD_STATIC=1
```