# Licensed to the Apache Software Foundation (ASF) under one

# or more contributor license agreements.  See the NOTICE file

# distributed with this work for additional information

# regarding copyright ownership.  The ASF licenses this file

# to you under the Apache License, Version 2.0 (the

# "License"); you may not use this file except in compliance

# with the License.  You may obtain a copy of the License at

#

#   http://www.apache.org/licenses/LICENSE-2.0

#

# Unless required by applicable law or agreed to in writing,

# software distributed under the License is distributed on an

# "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY

# KIND, either express or implied.  See the License for the

# specific language governing permissions and limitations

# under the License.

#

# Basic Airflow cluster configuration for CeleryExecutor with Redis and PostgreSQL.

#

# WARNING: This configuration is for local development. Do not use it in a production deployment.

#

# This configuration supports basic configuration using environment variables or an .env file

# The following variables are supported:

#

# AIRFLOW\_IMAGE\_NAME           - Docker image name used to run Airflow.

#                                Default: apache/airflow:2.8.1

# AIRFLOW\_UID                  - User ID in Airflow containers

#                                Default: 50000

# AIRFLOW\_PROJ\_DIR             - Base path to which all the files will be volumed.

#                                Default: .

# Those configurations are useful mostly in case of standalone testing/running Airflow in test/try-out mode

#

# \_AIRFLOW\_WWW\_USER\_USERNAME   - Username for the administrator account (if requested).

#                                Default: airflow

# \_AIRFLOW\_WWW\_USER\_PASSWORD   - Password for the administrator account (if requested).

#                                Default: airflow

# \_PIP\_ADDITIONAL\_REQUIREMENTS - Additional PIP requirements to add when starting all containers.

#                                Use this option ONLY for quick checks. Installing requirements at container

#                                startup is done EVERY TIME the service is started.

#                                A better way is to build a custom image or extend the official image

#                                as described in https://airflow.apache.org/docs/docker-stack/build.html.

#                                Default: ''

#

# Feel free to modify this file to suit your needs.

---

x-airflow-common:

  &airflow-common

  # In order to add custom dependencies or upgrade provider packages you can use your extended image.

  # Comment the image line, place your Dockerfile in the directory where you placed the docker-compose.yaml

  # and uncomment the "build" line below, Then run `docker-compose build` to build the images.

  image: ${AIRFLOW\_IMAGE\_NAME:-apache/airflow:2.8.1}

  # build: .

  environment:

    &airflow-common-env

    AIRFLOW\_\_CORE\_\_EXECUTOR: CeleryExecutor

    AIRFLOW\_\_DATABASE\_\_SQL\_ALCHEMY\_CONN: postgresql+psycopg2://airflow:airflow@postgres/airflow

    AIRFLOW\_\_CELERY\_\_RESULT\_BACKEND: db+postgresql://airflow:airflow@postgres/airflow

    AIRFLOW\_\_CELERY\_\_BROKER\_URL: redis://:@redis:6379/0

    AIRFLOW\_\_CORE\_\_FERNET\_KEY: ''

    AIRFLOW\_\_CORE\_\_DAGS\_ARE\_PAUSED\_AT\_CREATION: 'true'

    AIRFLOW\_\_CORE\_\_LOAD\_EXAMPLES: 'true'

    AIRFLOW\_\_API\_\_AUTH\_BACKENDS: 'airflow.api.auth.backend.basic\_auth,airflow.api.auth.backend.session'

    # yamllint disable rule:line-length

    # Use simple http server on scheduler for health checks

    # See https://airflow.apache.org/docs/apache-airflow/stable/administration-and-deployment/logging-monitoring/check-health.html#scheduler-health-check-server

    # yamllint enable rule:line-length

    AIRFLOW\_\_SCHEDULER\_\_ENABLE\_HEALTH\_CHECK: 'true'

    # WARNING: Use \_PIP\_ADDITIONAL\_REQUIREMENTS option ONLY for a quick checks

    # for other purpose (development, test and especially production usage) build/extend Airflow image.

    \_PIP\_ADDITIONAL\_REQUIREMENTS: ${\_PIP\_ADDITIONAL\_REQUIREMENTS:-psycopg2-binary}

  volumes:

    - ${AIRFLOW\_PROJ\_DIR:-.}/dags:/opt/airflow/dags

    - ${AIRFLOW\_PROJ\_DIR:-.}/logs:/opt/airflow/logs

    - ${AIRFLOW\_PROJ\_DIR:-.}/config:/opt/airflow/config

    - ${AIRFLOW\_PROJ\_DIR:-.}/plugins:/opt/airflow/plugins

    - ${AIRFLOW\_PROJ\_DIR:-.}/data:/opt/airflow/data

  user: "${AIRFLOW\_UID:-50000}:0"

  depends\_on:

    &airflow-common-depends-on

    redis:

      condition: service\_healthy

    postgres:

      condition: service\_healthy

services:

  postgres:

    image: postgres:13

    environment:

      POSTGRES\_USER: airflow

      POSTGRES\_PASSWORD: airflow

      POSTGRES\_DB: airflow

    volumes:

      - postgres-db-volume:/var/lib/postgresql/data

    healthcheck:

      test: ["CMD", "pg\_isready", "-U", "airflow"]

      interval: 10s

      retries: 5

      start\_period: 5s

    restart: always

  redis:

    image: redis:latest

    expose:

      - 6379

    healthcheck:

      test: ["CMD", "redis-cli", "ping"]

      interval: 10s

      timeout: 30s

      retries: 50

      start\_period: 30s

    restart: always

  airflow-webserver:

    <<: \*airflow-common

    command: webserver

    ports:

      - "8080:8080"

    healthcheck:

      test: ["CMD", "curl", "--fail", "http://localhost:8080/health"]

      interval: 30s

      timeout: 10s

      retries: 5

      start\_period: 30s

    restart: always

    depends\_on:

      <<: \*airflow-common-depends-on

      airflow-init:

        condition: service\_completed\_successfully

  airflow-scheduler:

    <<: \*airflow-common

    command: scheduler

    healthcheck:

      test: ["CMD", "curl", "--fail", "http://localhost:8974/health"]

      interval: 30s

      timeout: 10s

      retries: 5

      start\_period: 30s

    restart: always

    depends\_on:

      <<: \*airflow-common-depends-on

      airflow-init:

        condition: service\_completed\_successfully

  airflow-worker:

    <<: \*airflow-common

    command: celery worker

    healthcheck:

      # yamllint disable rule:line-length

      test:

        - "CMD-SHELL"

        - 'celery --app airflow.providers.celery.executors.celery\_executor.app inspect ping -d "celery@$${HOSTNAME}" || celery --app airflow.executors.celery\_executor.app inspect ping -d "celery@$${HOSTNAME}"'

      interval: 30s

      timeout: 10s

      retries: 5

      start\_period: 30s

    environment:

      <<: \*airflow-common-env

      # Required to handle warm shutdown of the celery workers properly

      # See https://airflow.apache.org/docs/docker-stack/entrypoint.html#signal-propagation

      DUMB\_INIT\_SETSID: "0"

    restart: always

    depends\_on:

      <<: \*airflow-common-depends-on

      airflow-init:

        condition: service\_completed\_successfully

  airflow-triggerer:

    <<: \*airflow-common

    command: triggerer

    healthcheck:

      test: ["CMD-SHELL", 'airflow jobs check --job-type TriggererJob --hostname "$${HOSTNAME}"']

      interval: 30s

      timeout: 10s

      retries: 5

      start\_period: 30s

    restart: always

    depends\_on:

      <<: \*airflow-common-depends-on

      airflow-init:

        condition: service\_completed\_successfully

  airflow-init:

    <<: \*airflow-common

    entrypoint: /bin/bash

    # yamllint disable rule:line-length

    command:

      - -c

      - |

        if [[ -z "${AIRFLOW\_UID}" ]]; then

          echo

          echo -e "\033[1;33mWARNING!!!: AIRFLOW\_UID not set!\e[0m"

          echo "If you are on Linux, you SHOULD follow the instructions below to set "

          echo "AIRFLOW\_UID environment variable, otherwise files will be owned by root."

          echo "For other operating systems you can get rid of the warning with manually created .env file:"

          echo "    See: https://airflow.apache.org/docs/apache-airflow/stable/howto/docker-compose/index.html#setting-the-right-airflow-user"

          echo

        fi

        one\_meg=1048576

        mem\_available=$$(($$(getconf \_PHYS\_PAGES) \* $$(getconf PAGE\_SIZE) / one\_meg))

        cpus\_available=$$(grep -cE 'cpu[0-9]+' /proc/stat)

        disk\_available=$$(df / | tail -1 | awk '{print $$4}')

        warning\_resources="false"

        if (( mem\_available < 4000 )) ; then

          echo

          echo -e "\033[1;33mWARNING!!!: Not enough memory available for Docker.\e[0m"

          echo "At least 4GB of memory required. You have $$(numfmt --to iec $$((mem\_available \* one\_meg)))"

          echo

          warning\_resources="true"

        fi

        if (( cpus\_available < 2 )); then

          echo

          echo -e "\033[1;33mWARNING!!!: Not enough CPUS available for Docker.\e[0m"

          echo "At least 2 CPUs recommended. You have $${cpus\_available}"

          echo

          warning\_resources="true"

        fi

        if (( disk\_available < one\_meg \* 10 )); then

          echo

          echo -e "\033[1;33mWARNING!!!: Not enough Disk space available for Docker.\e[0m"

          echo "At least 10 GBs recommended. You have $$(numfmt --to iec $$((disk\_available \* 1024 )))"

          echo

          warning\_resources="true"

        fi

        if [[ $${warning\_resources} == "true" ]]; then

          echo

          echo -e "\033[1;33mWARNING!!!: You have not enough resources to run Airflow (see above)!\e[0m"

          echo "Please follow the instructions to increase amount of resources available:"

          echo "   https://airflow.apache.org/docs/apache-airflow/stable/howto/docker-compose/index.html#before-you-begin"

          echo

        fi

        mkdir -p /sources/logs /sources/dags /sources/plugins

        chown -R "${AIRFLOW\_UID}:0" /sources/{logs,dags,plugins}

        exec /entrypoint airflow version

    # yamllint enable rule:line-length

    environment:

      <<: \*airflow-common-env

      \_AIRFLOW\_DB\_MIGRATE: 'true'

      \_AIRFLOW\_WWW\_USER\_CREATE: 'true'

      \_AIRFLOW\_WWW\_USER\_USERNAME: ${\_AIRFLOW\_WWW\_USER\_USERNAME:-airflow}

      \_AIRFLOW\_WWW\_USER\_PASSWORD: ${\_AIRFLOW\_WWW\_USER\_PASSWORD:-airflow}

      \_PIP\_ADDITIONAL\_REQUIREMENTS: ''

    user: "0:0"

    volumes:

      - ${AIRFLOW\_PROJ\_DIR:-.}:/sources

  airflow-cli:

    <<: \*airflow-common

    profiles:

      - debug

    environment:

      <<: \*airflow-common-env

      CONNECTION\_CHECK\_MAX\_COUNT: "0"

    # Workaround for entrypoint issue. See: https://github.com/apache/airflow/issues/16252

    command:

      - bash

      - -c

      - airflow

  # You can enable flower by adding "--profile flower" option e.g. docker-compose --profile flower up

  # or by explicitly targeted on the command line e.g. docker-compose up flower.

  # See: https://docs.docker.com/compose/profiles/

  flower:

    <<: \*airflow-common

    command: celery flower

    profiles:

      - flower

    ports:

      - "5555:5555"

    healthcheck:

      test: ["CMD", "curl", "--fail", "http://localhost:5555/"]

      interval: 30s

      timeout: 10s

      retries: 5

      start\_period: 30s

    restart: always

    depends\_on:

      <<: \*airflow-common-depends-on

      airflow-init:

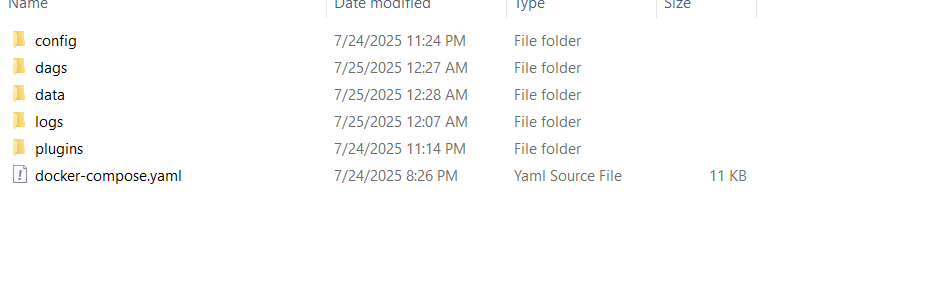
        condition: service\_completed\_successfully

volumes:

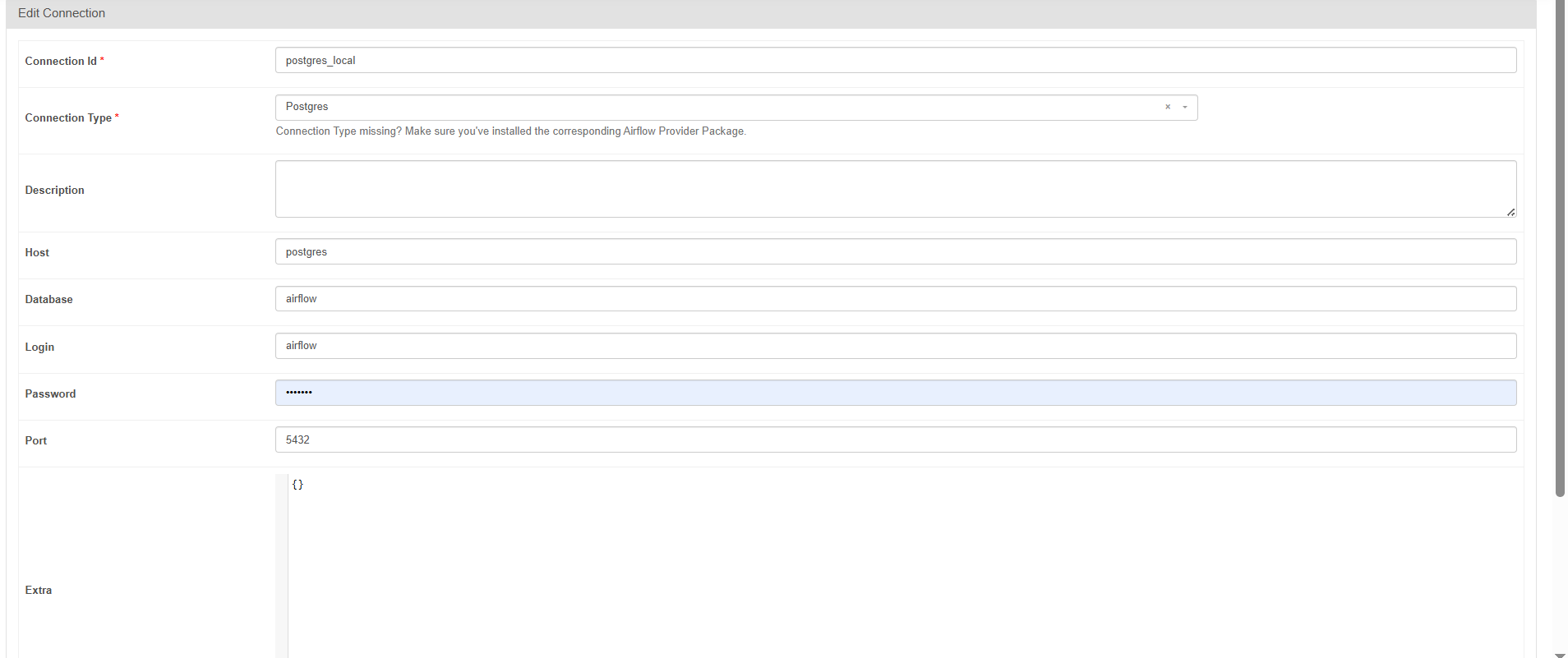
  postgres-db-volume:

Above was the docker-compose.ymal file.

* Tạo folder có chứa dags, plugins, logs và file ymal

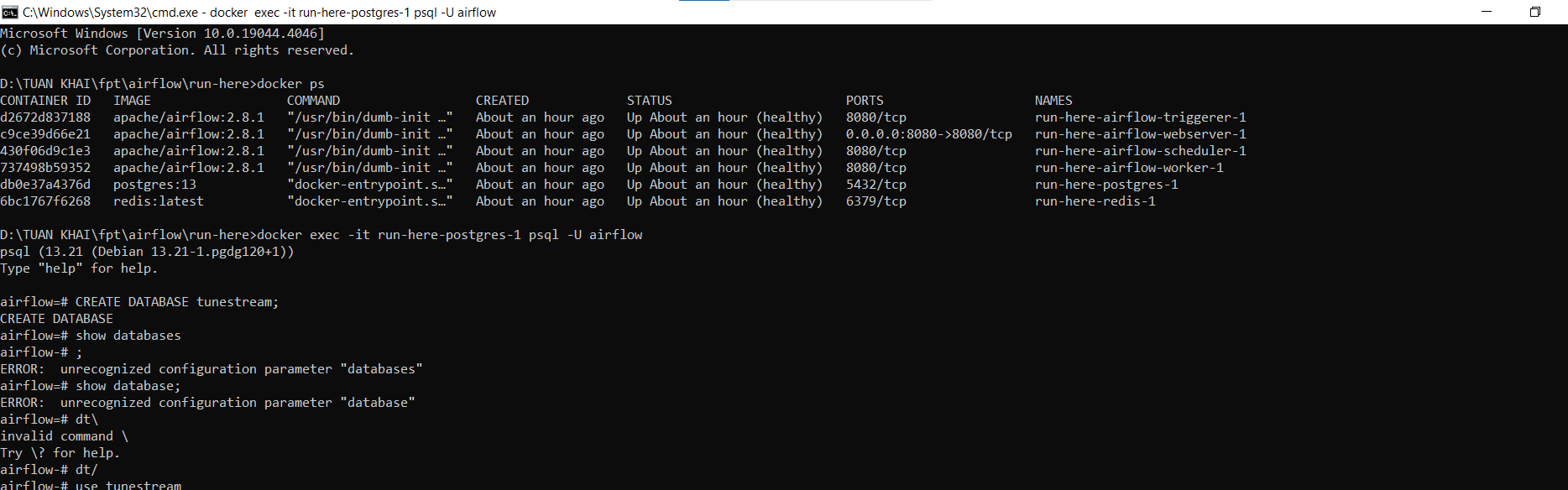


* Docker-compose init và up ở trong folder đó
* Chạy 8080 vào airflow UI. Dags sẽ tự load từ dags folder.
* Connection như sau



Để database, login và password là airflow vì trong ymal file tạo postgres cũng khởi tạo như vậy.

* Kết nối psql trong terminal để tạo bảng bằng code trong file create\_table.sql



* Got one table sorted out

