Airflow Installation

To install airflow we need docker-compose file that's available in official airflow documentation.

Follow this steps to install configure and test airflow

https://airflow.apache.org/docs/apache-airflow/stable/howto/docker-compose/index.html

1. Create a folder called “airflow”
2. Past docker compose file
3. Docker compose up -d
4. Airflow should run on port 8080
5. Check all the volume and port details on the docker compose file

Testing airflow

1. Open airflow ui using port 8080
2. User & Pass: airflow
3. Check the terminal (configs dags logs plugins) - folders are available
4. Create an simple dag on the dags folder (scheduled task)
5. Verify the dags are visible on the ui - and run it - check logs

—----------

Docker compose file

docker-compose.yaml

```
# 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:3.0.4
# 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 distributions 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:3.0.4}
  # build: .
  environment:
    &airflow-common-env
    AIRFLOW__CORE__EXECUTOR: CeleryExecutor
    AIRFLOW__CORE__AUTH_MANAGER:
airflow.providers.fab.auth_manager.fab_auth_manager.FabAuthManager
    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__CORE__EXECUTION_API_SERVER_URL:
'http://airflow-apiserver:8080/execution/'
    # 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/log
ging-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:-}
    # The following line can be used to set a custom config file, stored in the local config
folder
    AIRFLOW_CONFIG: '/opt/airflow/config/airflow.cfg'
  volumes:
    - ${AIRFLOW_PROJ_DIR:-.}/dags:/opt/airflow/dags
    - ${AIRFLOW_PROJ_DIR:-.}/logs:/opt/airflow/logs
    - ${AIRFLOW_PROJ_DIR:-.}/config:/opt/airflow/config
    - ${AIRFLOW_PROJ_DIR:-.}/plugins:/opt/airflow/plugins
  user: "${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:
    # Redis is limited to 7.2-bookworm due to licencing change
```

```
  # https://redis.io/blog/redis-adopts-dual-source-available-licensing/
  image: redis:7.2-bookworm
  expose:
    - 6379
  healthcheck:
    test: ["CMD", "redis-cli", "ping"]
    interval: 10s
    timeout: 30s
    retries: 50
    start_period: 30s
  restart: always

airflow-apiserver:
  <<: *airflow-common
  command: api-server
  ports:
    - "8080:8080"
  healthcheck:
    test: ["CMD", "curl", "--fail", "http://localhost:8080/api/v2/version"]
    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-dag-processor:
    <<: *airflow-common
    command: dag-processor
    healthcheck:
      test: ["CMD-SHELL", 'airflow jobs check --job-type DagProcessorJob --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-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-apiserver:
        condition: service_healthy
      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
    export AIRFLOW_UID=$$(id -u)
  fi
  one_meg=1048576
  mem_available=$$(($$(getconf _PHYS_PAGES) * $$(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
  echo
  echo "Creating missing opt dirs if missing:"
  echo
  mkdir -v -p /opt/airflow/{logs,dags,plugins,config}
  echo
  echo "Airflow version:"
  /entrypoint airflow version
  echo
  echo "Files in shared volumes:"
  echo
  ls -la /opt/airflow/{logs,dags,plugins,config}
  echo
  echo "Running airflow config list to create default config file if missing."
  echo
  /entrypoint airflow config list >/dev/null
  echo
  echo "Files in shared volumes:"
  echo
  ls -la /opt/airflow/{logs,dags,plugins,config}
  echo
  echo "Change ownership of files in /opt/airflow to ${AIRFLOW_UID}:0"
  echo
  chown -R "${AIRFLOW_UID}:0" /opt/airflow/
  echo
  echo "Change ownership of files in shared volumes to ${AIRFLOW_UID}:0"
  echo
  chown -v -R "${AIRFLOW_UID}:0" /opt/airflow/{logs,dags,plugins,config}
  echo
  echo "Files in shared volumes:"
  echo
  ls -la /opt/airflow/{logs,dags,plugins,config}
```

```
    # 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"

  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
    depends_on:
      <<: *airflow-common-depends-on

  # 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:
```

Docker compose up -d