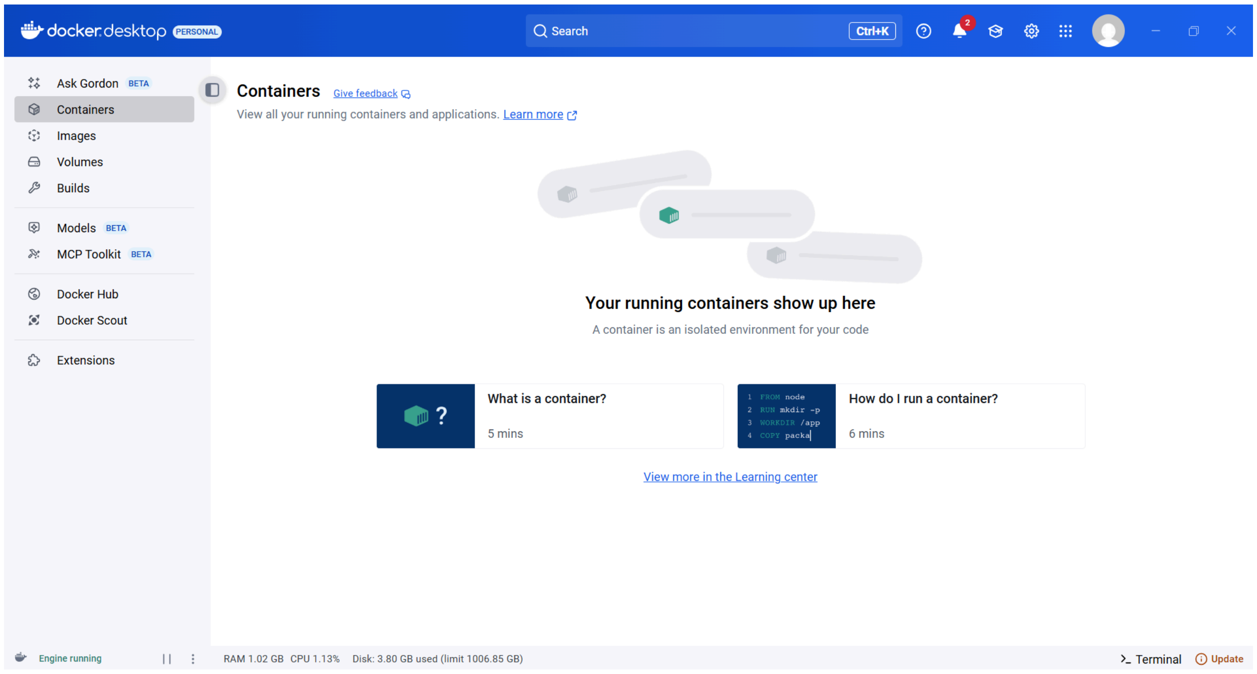
Airflow (2.4.2) on Windows + Docker - Manual

1) Prerequisites

1. Install **Docker Desktop for Windows** (with WSL2 backend enabled).

   https://www.docker.com/products/docker-desktop/



View all your running containers and applications.

Your running containers show up here

A container is an isolated environment for your code

```
FROM node
RUN mkdir -p
WORKDIR /app
COPY packa
```

RAM 1.02 GB CPU 1.13% Disk: 3.80 GB used (limit 1006.85 GB)

2. Open **PowerShell** (as a normal user is fine).

3. Create a working folder:
   ```
   mkdir airflow
   cd airflow
   ```

   This will be the working directory for Airflow.

2) Get the Docker Compose file

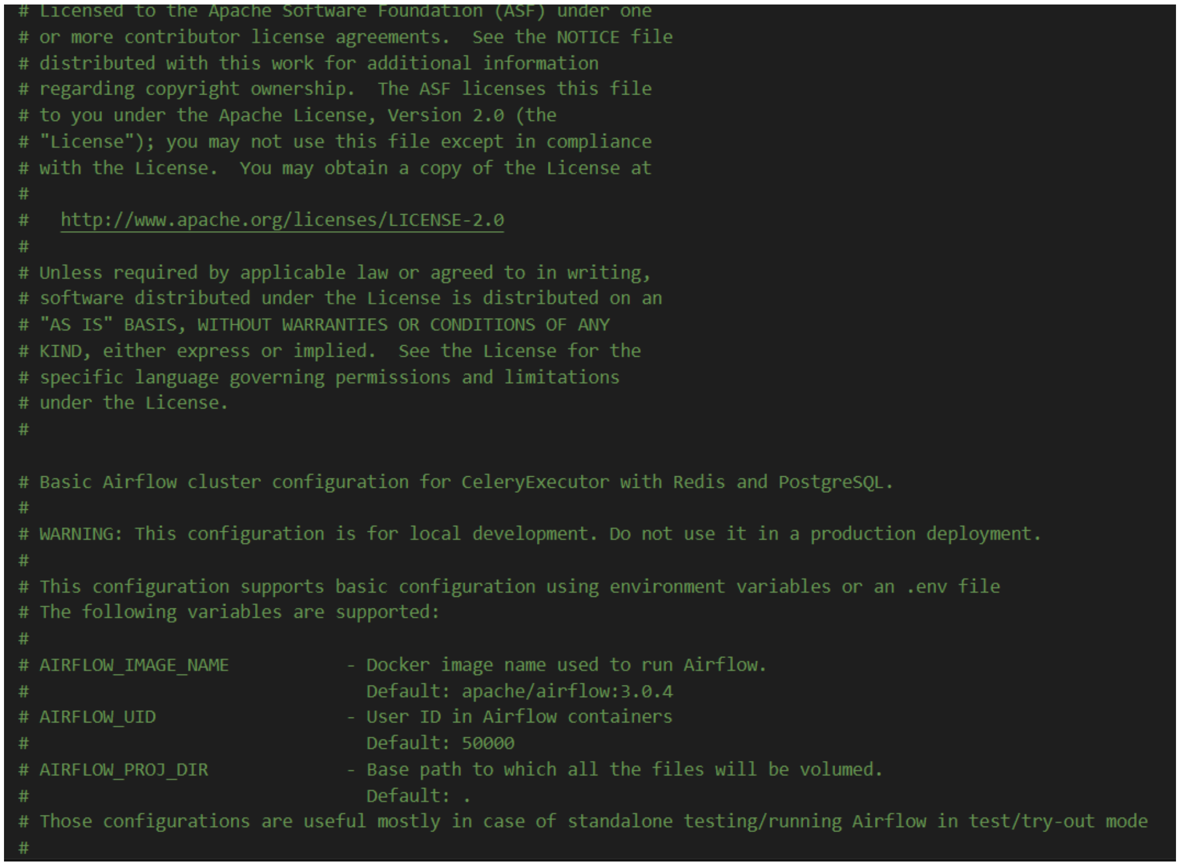
- Use the **CeleryExecutor** example for Airflow **2.4.2** (Postgres + Redis).

- **Important edits we made** to avoid errors:
    - Removed the **airflow-apiserver** service (Airflow 2.4.2 doesn't have the api-server command).
    - Removed any depends_on: airflow-apiserver (e.g., in airflow-worker).
    - Fixed the **duplicate healthcheck key** (YAML cannot define the same key twice in one block).

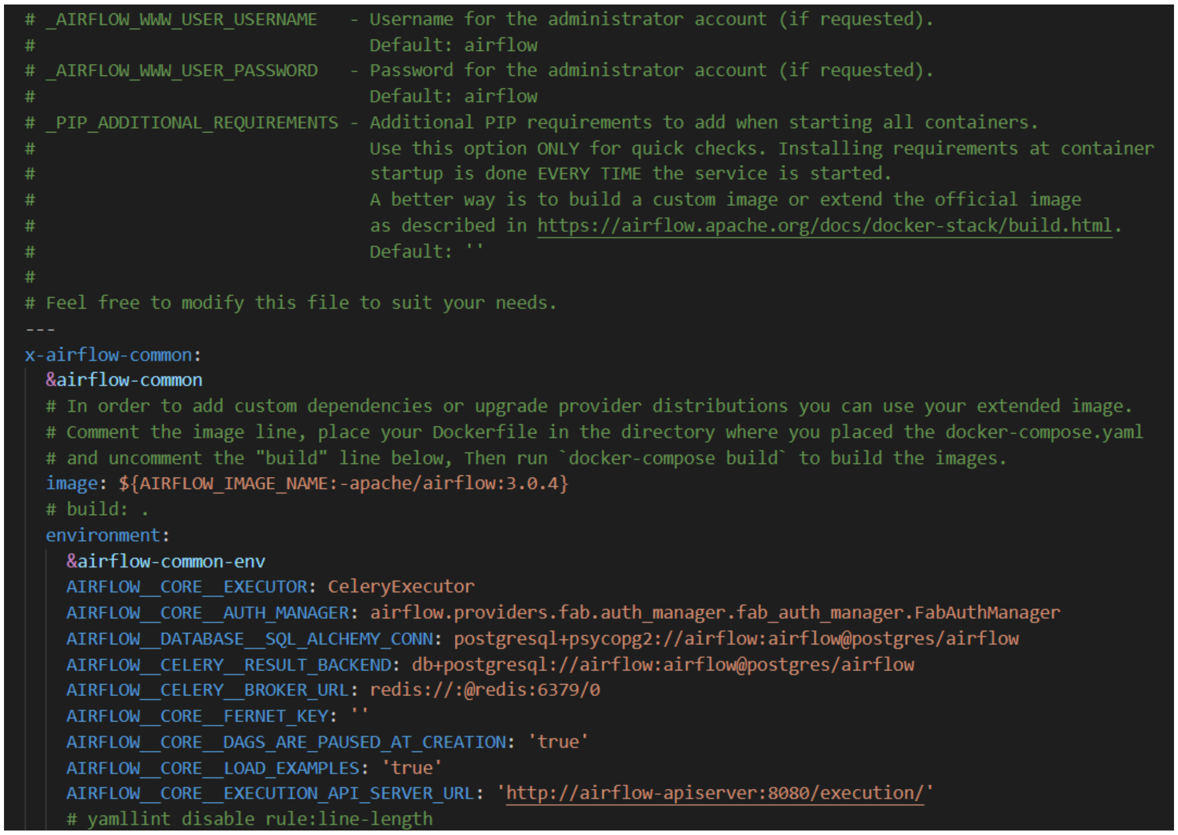
Ensured the **webserver** has:

- ports:
  - "8080:8080"

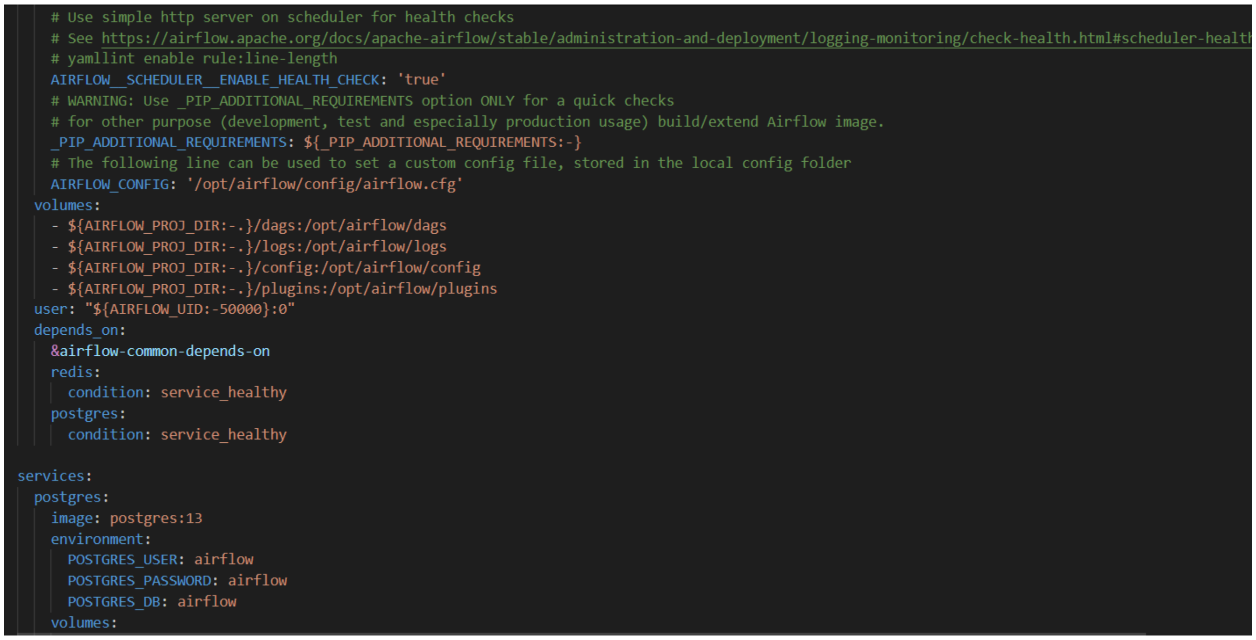
YAML file :



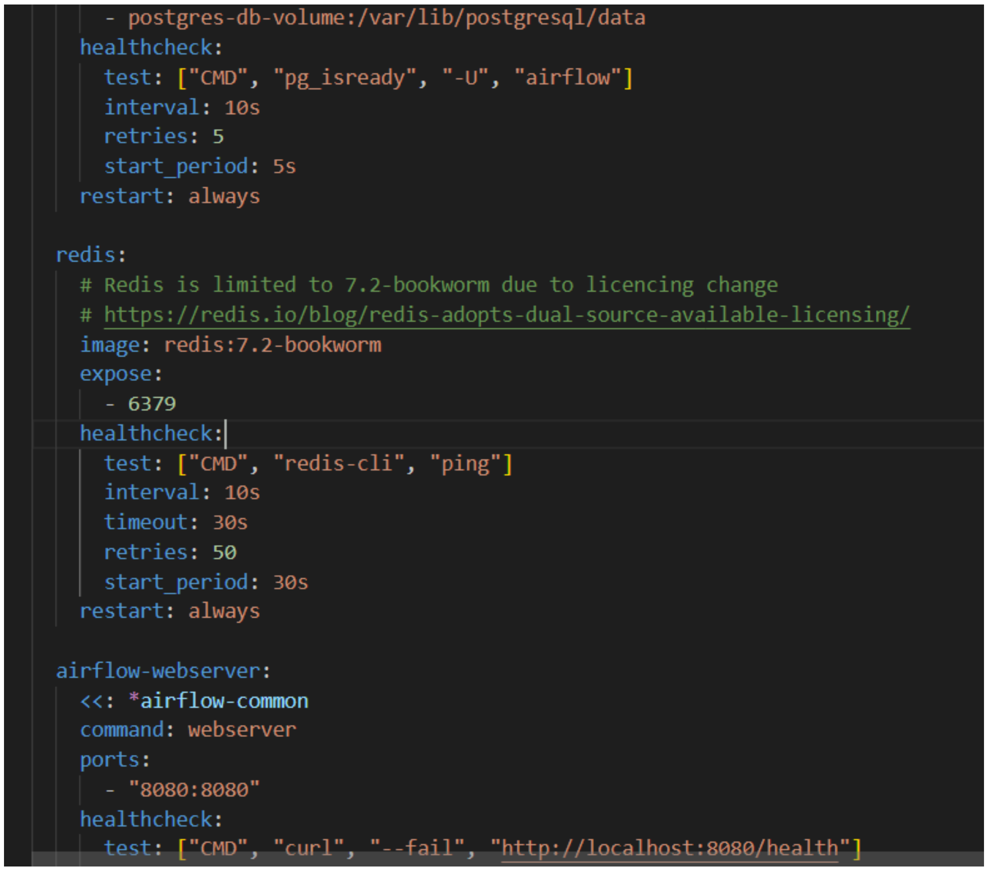
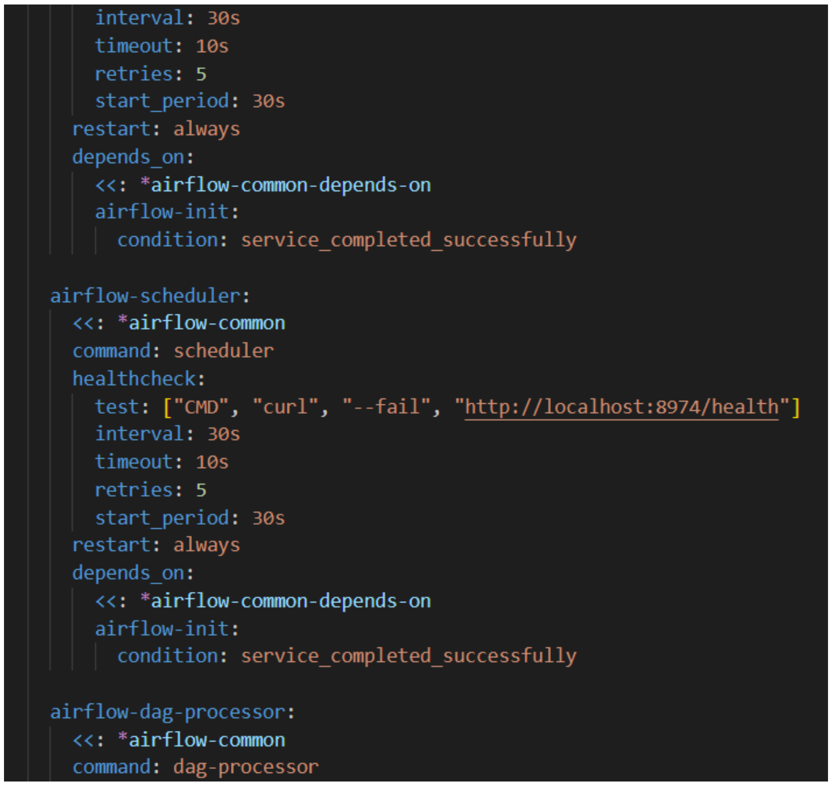
```
# Licensed to the Apache Software Foundation (ASF) under one
# or more contributor license agreements.  See the NOTICE file
# distributed with this work for additional information
# regarding copyright ownership.  The ASF licenses this file
# to you under the Apache License, Version 2.0 (the
# "License"); 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
#
```

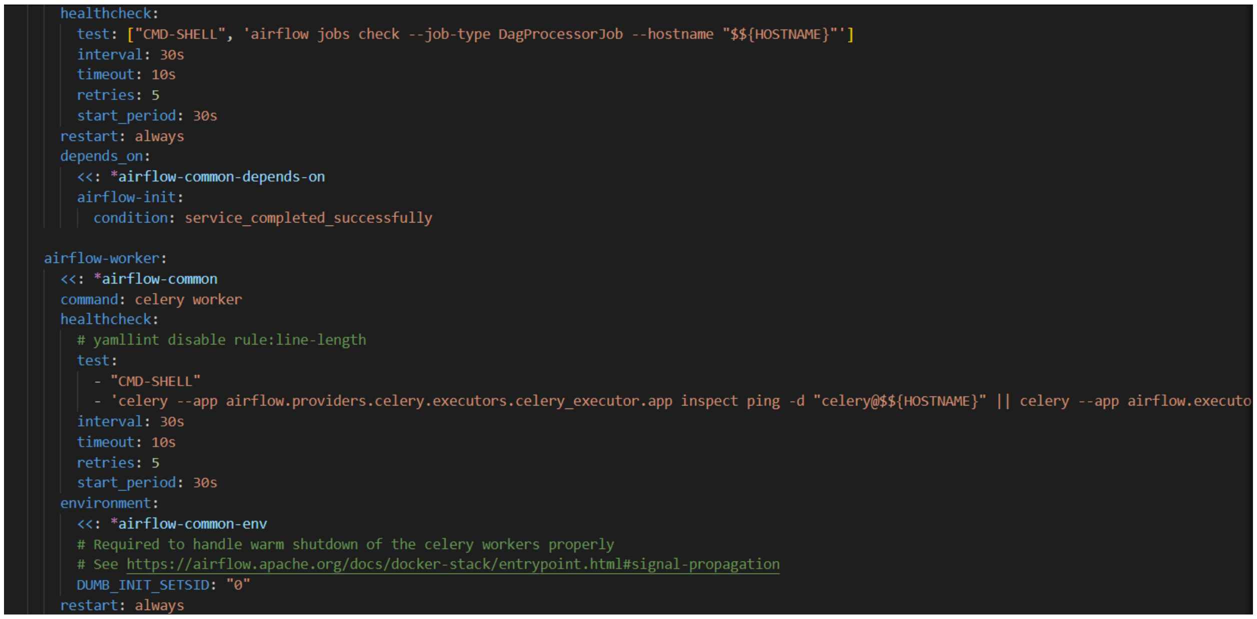


```yaml
# _AIRFLOW_WWW_USER_USERNAME   - Username for the administrator account (if requested).
#                                Default: airflow
# _AIRFLOW_WWW_USER_PASSWORD   - Password for the administrator account (if requested).
#                                Default: airflow
# _PIP_ADDITIONAL_REQUIREMENTS - 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
```

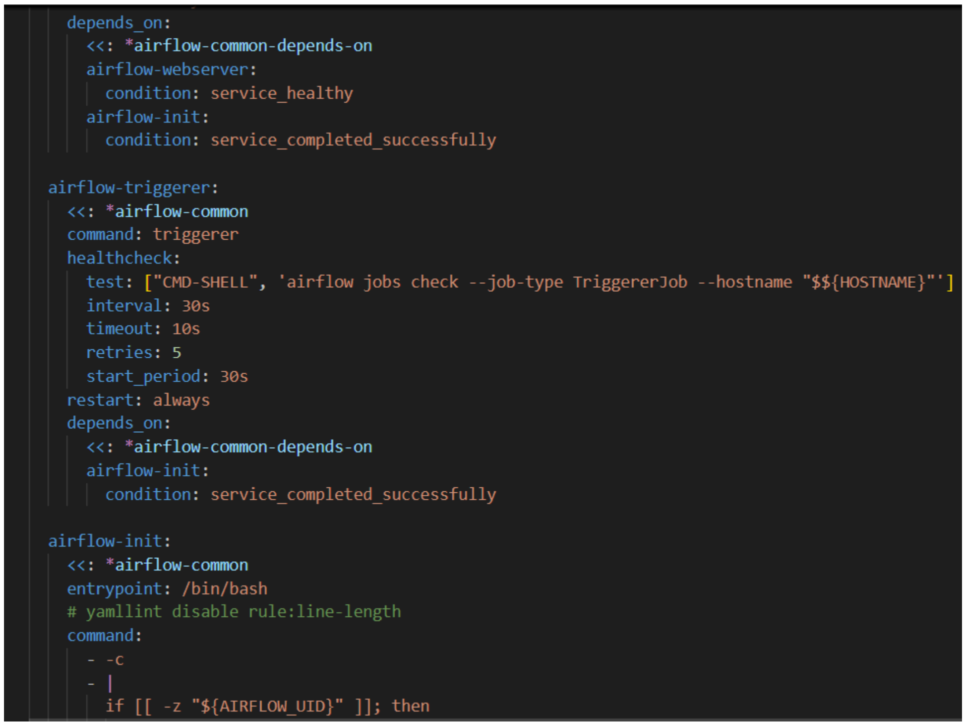


```yaml
    # Use simple http server on scheduler for health checks
    # See https://airflow.apache.org/docs/apache-airflow/stable/administration-and-deployment/logging-monitoring/check-health.html#scheduler-healtl
    # yamllint enable rule:line-length
    AIRFLOW__SCHEDULER__ENABLE_HEALTH_CHECK: 'true'
    # 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:
```

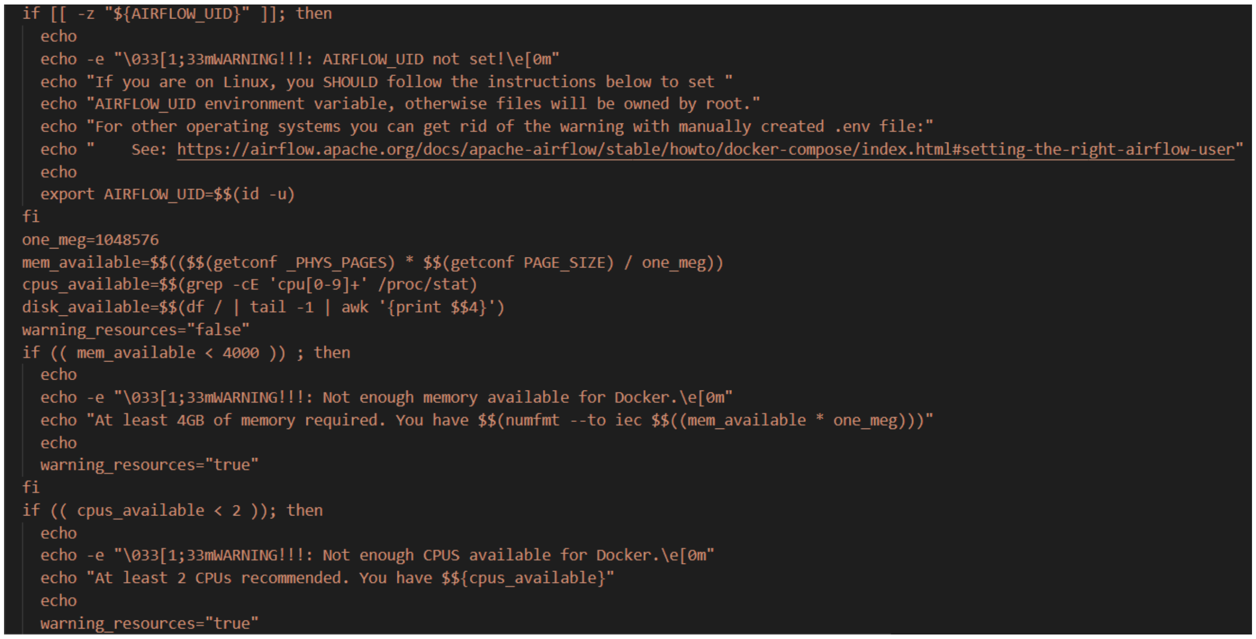

```yaml
      - postgres-db-volume:/var/lib/postgresql/data
    healthcheck:
      test: ["CMD", "pg_isready", "-U", "airflow"]
      interval: 10s
      retries: 5
      start_period: 5s
    restart: always

  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-webserver:
    <<: *airflow-common
    command: webserver
    ports:
      - "8080:8080"
    healthcheck:
      test: ["CMD", "curl", "--fail", "http://localhost:8080/health"]
```

```yaml
      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
```



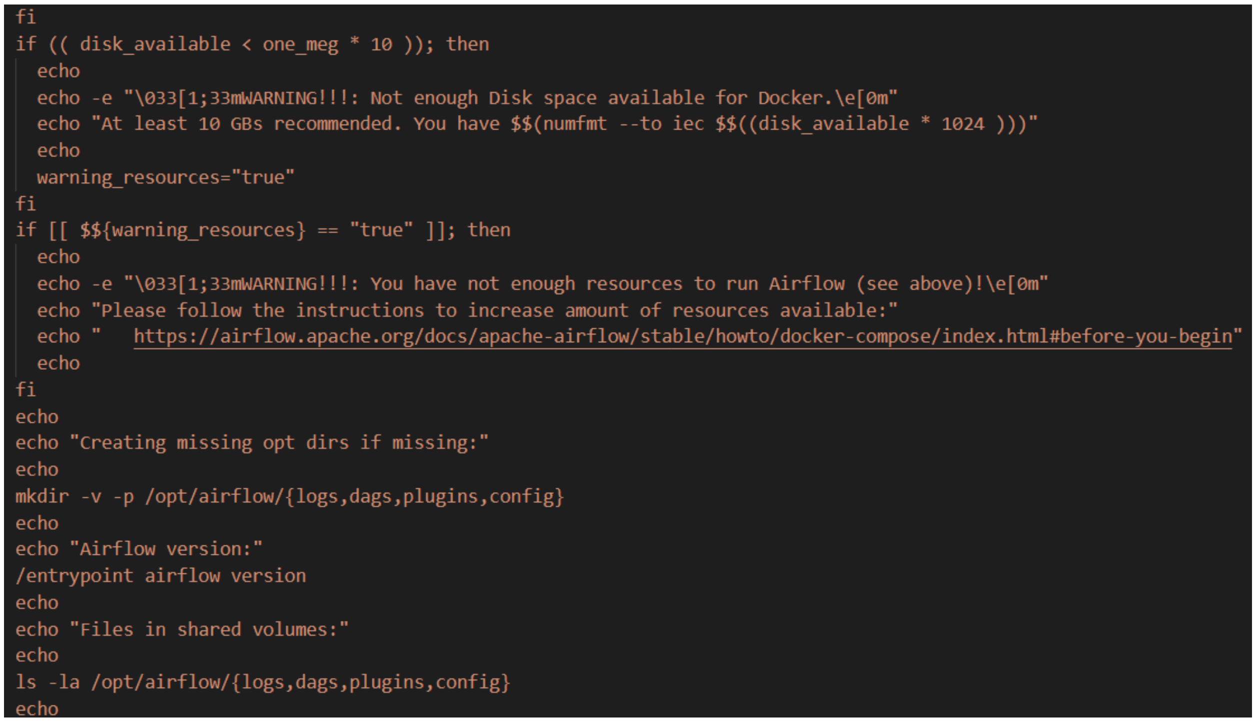
```
    healthcheck:
      test: ["CMD-SHELL", 'airflow jobs check --job-type DagProcessorJob --hostname "$${HOSTNAME}"']
      interval: 30s
      timeout: 10s
      retries: 5
      start_period: 30s
    restart: always
    depends_on:
      <<: *airflow-common-depends-on
      airflow-init:
        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.executo
      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
```



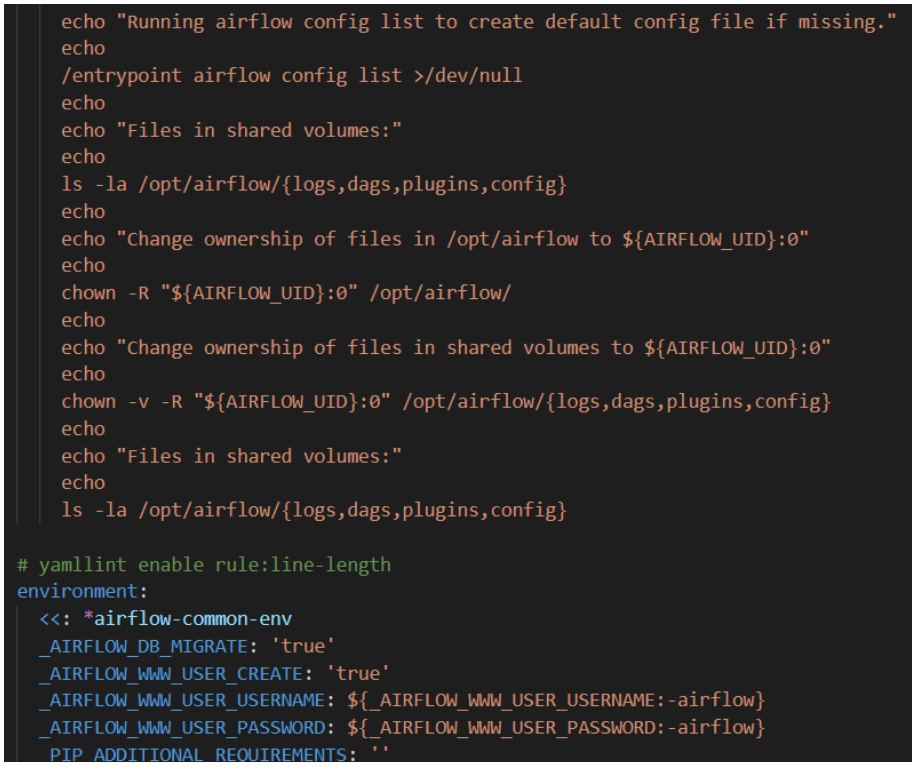
```
    depends_on:
      <<: *airflow-common-depends-on
      airflow-webserver:
        condition: service_healthy
      airflow-init:
        condition: service_completed_successfully

  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
```



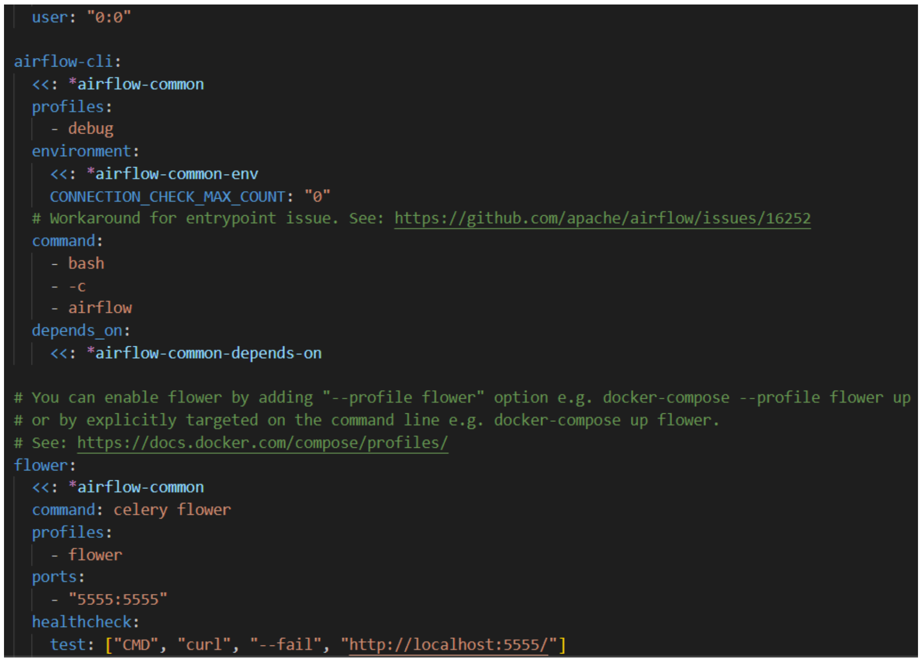
```
if [[ -z "${AIRFLOW_UID}" ]]; then
  echo
  echo -e "\033[1;33mWARNING!!!: AIRFLOW_UID not set!\e[0m"
  echo "If you are on Linux, you SHOULD follow the instructions below to set "
  echo "AIRFLOW_UID environment variable, otherwise files will be owned by root."
  echo "For other operating systems 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"
```



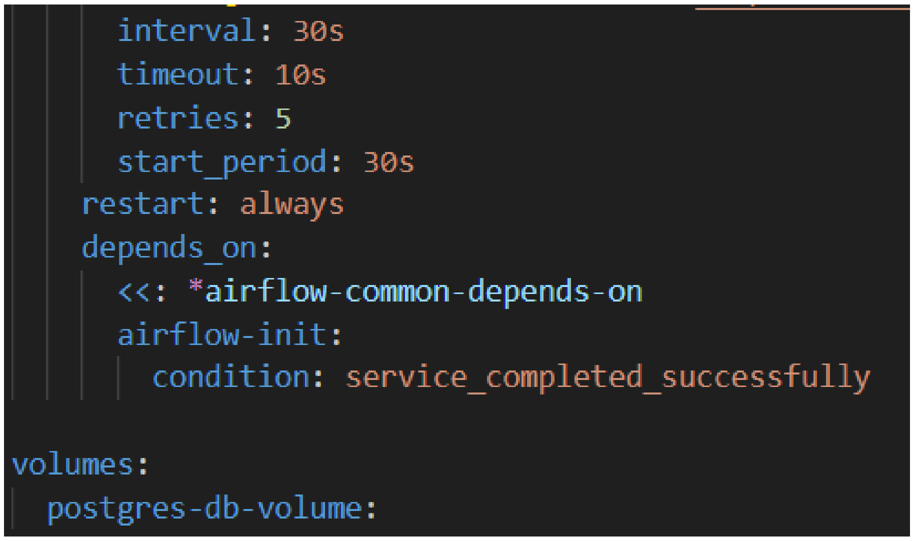
```
fi
if (( disk_available < one_meg * 10 )); then
  echo
  echo -e "\033[1;33mWARNING!!!: Not enough Disk space available for Docker.\e[0m"
  echo "At least 10 GBs recommended. You have $$(numfmt --to iec $$((disk_available * 1024 )))"
  echo
  warning_resources="true"
fi
if [[ $${warning_resources} == "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:
```

3) (One-time) Initialize Airflow

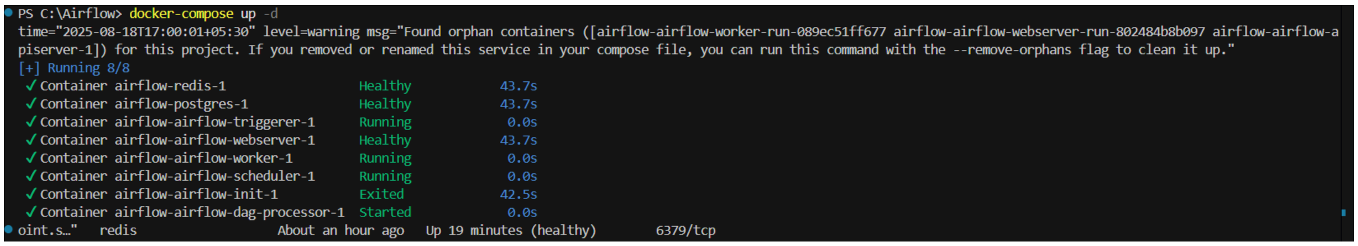
This creates the database schema, default roles, and the admin user if configured:

```
docker-compose up airflow-init
```

You should see the airflow-init container **exit successfully**.
(If you forget this step, webserver logs complain: “You need to initialize the database. Please run airflow db init.”)

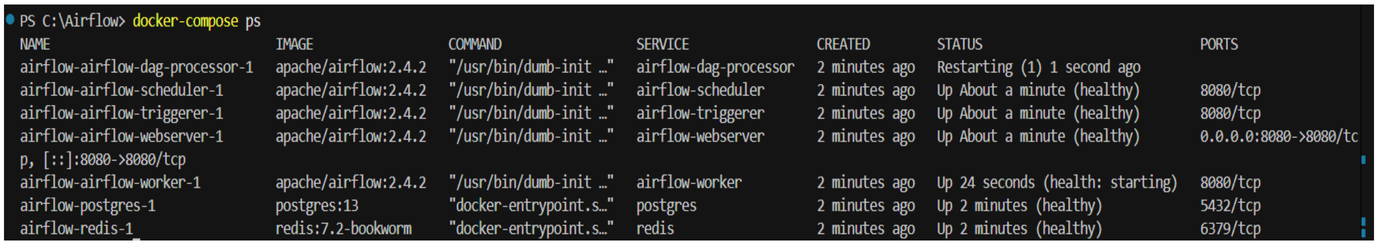
4) Start the Airflow stack

```
docker-compose up -d
```


```
PS C:\Airflow> docker-compose up -d
time="2025-08-18T17:00:01+05:30" level=warning msg="Found orphan containers ([airflow-airflow-worker-run-089ec51ff677 airflow-airflow-webserver-run-802484b8b097 airflow-airflow-a
piserver-1]) for this project. If you removed or renamed this service in your compose file, you can run this command with the --remove-orphans flag to clean it up."
[+] Running 8/8
 ✓ Container airflow-redis-1                  Healthy          43.7s
 ✓ Container airflow-postgres-1               Healthy          43.7s
 ✓ Container airflow-airflow-triggerer-1      Running           0.0s
 ✓ Container airflow-airflow-webserver-1      Healthy          43.7s
 ✓ Container airflow-airflow-worker-1         Running           0.0s
 ✓ Container airflow-airflow-scheduler-1      Running           0.0s
 ✓ Container airflow-airflow-init-1           Exited           42.5s
 ✓ Container airflow-airflow-dag-processor-1  Started           0.0s
oint.s…"   redis                  About an hour ago   Up 19 minutes (healthy)       6379/tcp
```

Check status:

```
docker-compose ps
```



```
PS C:\Airflow> docker-compose ps
NAME                              IMAGE                 COMMAND                  SERVICE                 CREATED         STATUS                             PORTS
airflow-airflow-dag-processor-1   apache/airflow:2.4.2  "/usr/bin/dumb-init …"   airflow-dag-processor   2 minutes ago   Restarting (1) 1 second ago
airflow-airflow-scheduler-1       apache/airflow:2.4.2  "/usr/bin/dumb-init …"   airflow-scheduler       2 minutes ago   Up About a minute (healthy)        8080/tcp
airflow-airflow-triggerer-1       apache/airflow:2.4.2  "/usr/bin/dumb-init …"   airflow-triggerer       2 minutes ago   Up About a minute (healthy)        8080/tcp
airflow-airflow-webserver-1       apache/airflow:2.4.2  "/usr/bin/dumb-init …"   airflow-webserver       2 minutes ago   Up About a minute (healthy)        0.0.0.0:8080->8080/tc
p, [::]:8080->8080/tcp
airflow-airflow-worker-1          apache/airflow:2.4.2  "/usr/bin/dumb-init …"   airflow-worker          2 minutes ago   Up 24 seconds (health: starting)   8080/tcp
airflow-postgres-1                postgres:13           "docker-entrypoint.s…"   postgres                2 minutes ago   Up 2 minutes (healthy)             5432/tcp
airflow-redis-1                   redis:7.2-bookworm    "docker-entrypoint.s…"   redis                   2 minutes ago   Up 2 minutes (healthy)             6379/tcp
```

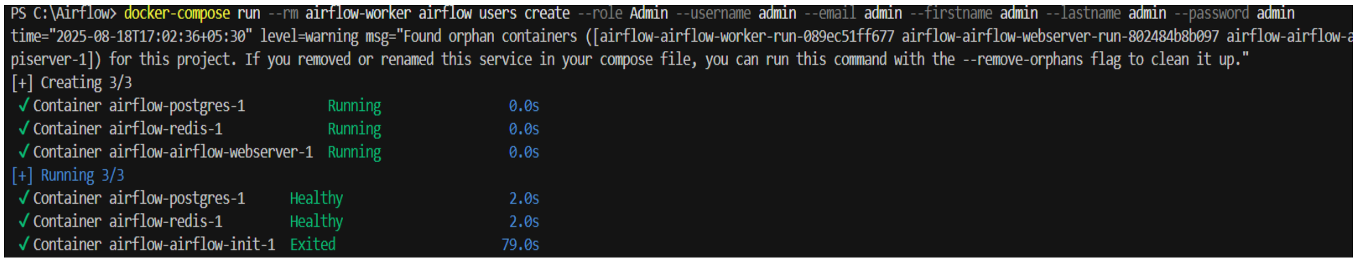
You should see these services **running/healthy**:

- `airflow-webserver`
- `airflow-scheduler`
- `airflow-worker`
- `airflow-triggerer`
- `postgres`
- `redis`
- (`airflow-dag-processor` may restart; it's optional and not critical.)

5) Create an Admin user (PowerShell syntax!)

If you didn't set admin creds via env vars, create one **on a single line** (no `\` in PowerShell):

```
docker-compose run --rm airflow-worker airflow users create --role
Admin --username admin --email admin --firstname admin --lastname
admin --password admin
```

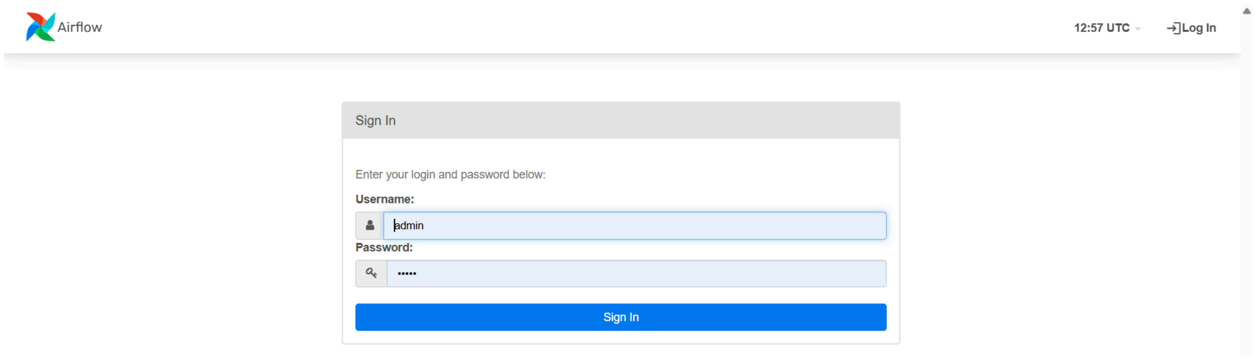


```
PS C:\Airflow> docker-compose run --rm airflow-worker airflow users create --role Admin --username admin --email admin --firstname admin --lastname admin --password admin
time="2025-08-18T17:02:36+05:30" level=warning msg="Found orphan containers ([airflow-airflow-worker-run-089ec51ff677 airflow-airflow-webserver-run-802484b8b097 airflow-airflow-a
piserver-1]) for this project. If you removed or renamed this service in your compose file, you can run this command with the --remove-orphans flag to clean it up."
[+] Creating 3/3
 ✓Container airflow-postgres-1          Running                 0.0s
 ✓Container airflow-redis-1             Running                 0.0s
 ✓Container airflow-airflow-webserver-1 Running                 0.0s
[+] Running 3/3
 ✓Container airflow-postgres-1     Healthy                      2.0s
 ✓Container airflow-redis-1        Healthy                      2.0s
 ✓Container airflow-airflow-init-1 Exited                      79.0s
```

Note: The multiline example with \ is for bash. In PowerShell, use one line (or PowerShell's backtick ` for line continuation).

6) Open the Airflow UI

- Go to: http://localhost:8080
- Log in with: **admin / admin** (or whatever you created).



Sign In

Enter your login and password below:

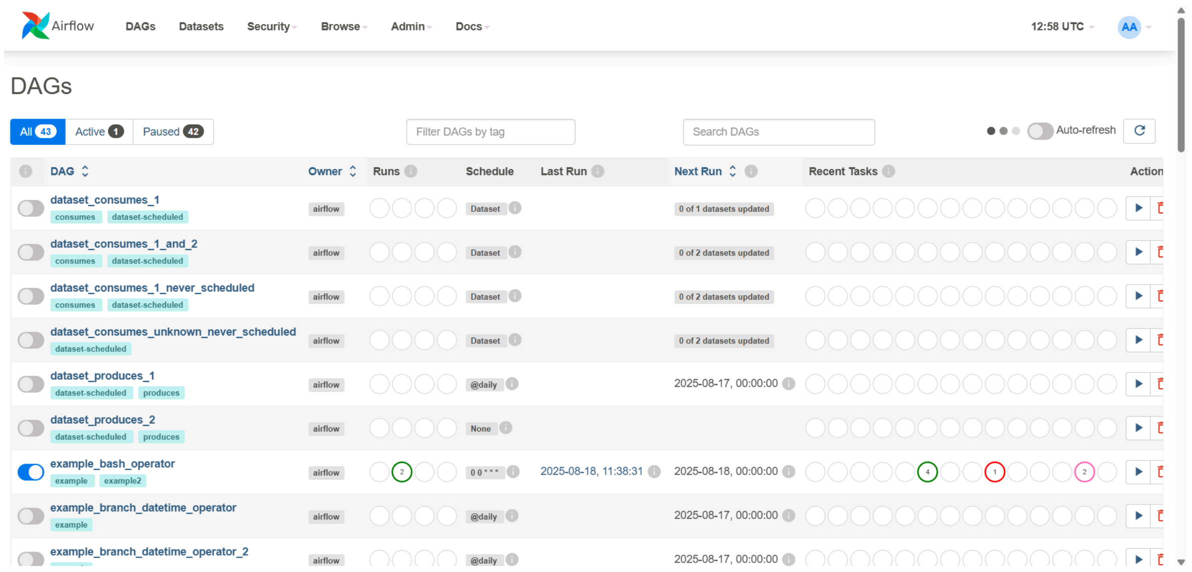
Username: admin

Password: •••••

Sign In

7) Verify with a test DAG

- In the UI, unpause any example DAG (if enabled) or drop a simple DAG file into C:\Airflow\dags.
- Trigger it and view task logs to confirm the worker is running tasks.



DAGs

All 43 | Active 1 | Paused 42

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Action
dataset_consumes_1 (consumes, dataset-scheduled)	airflow		Dataset		0 of 1 datasets updated		
dataset_consumes_1_and_2 (consumes, dataset-scheduled)	airflow		Dataset		0 of 2 datasets updated		
dataset_consumes_1_never_scheduled (consumes, dataset-scheduled)	airflow		Dataset		0 of 2 datasets updated		
dataset_consumes_unknown_never_scheduled (dataset-scheduled)	airflow		Dataset		0 of 2 datasets updated		
dataset_produces_1 (dataset-scheduled, produces)	airflow		@daily		2025-08-17, 00:00:00		
dataset_produces_2 (dataset-scheduled, produces)	airflow		None				
example_bash_operator (example, example2)	airflow	2	0 0 * * *	2025-08-18, 11:38:31	2025-08-18, 00:00:00	4 1 2	
example_branch_datetime_operator (example)	airflow		@daily		2025-08-17, 00:00:00		
example_branch_datetime_operator_2	airflow		@daily		2025-08-17, 00:00:00		

Troubleshooting we hit (and fixes)

1. **argument GROUP_OR_COMMAND: invalid choice: 'api-server'**
   - Cause: Compose file referenced **airflow-apiserver** with command api-server (only in newer Airflow).
   - Fix: **Remove** the airflow-apiserver service and **remove** any depends_on: airflow-apiserver.
2. **line N: mapping key "healthcheck" already defined**
   - Cause: Duplicate healthcheck key in the same YAML block.
   - Fix: Keep only one healthcheck section per service.
3. **Web UI ERR_EMPTY_RESPONSE / webserver unhealthy**
   - Cause: Database wasn't initialized yet.
   - Fix: Run docker-compose up airflow-init, then docker-compose up -d.
4. **PowerShell parsing errors on --role etc.**
   - Cause: Using bash line continuations (\) in PowerShell.
   - Fix: Run the whole command on **one line**.
5. **Orphan containers warning**
   - Cause: Old containers from a previous compose definition.
   - Fix:

```
docker-compose down --remove-orphans
docker-compose up -d
```