درس مبانی اینترنت اشیاء

گزارش تمرین ۲

فهرست

بخش ۱

اصطلاح Hot Path و Cold Path

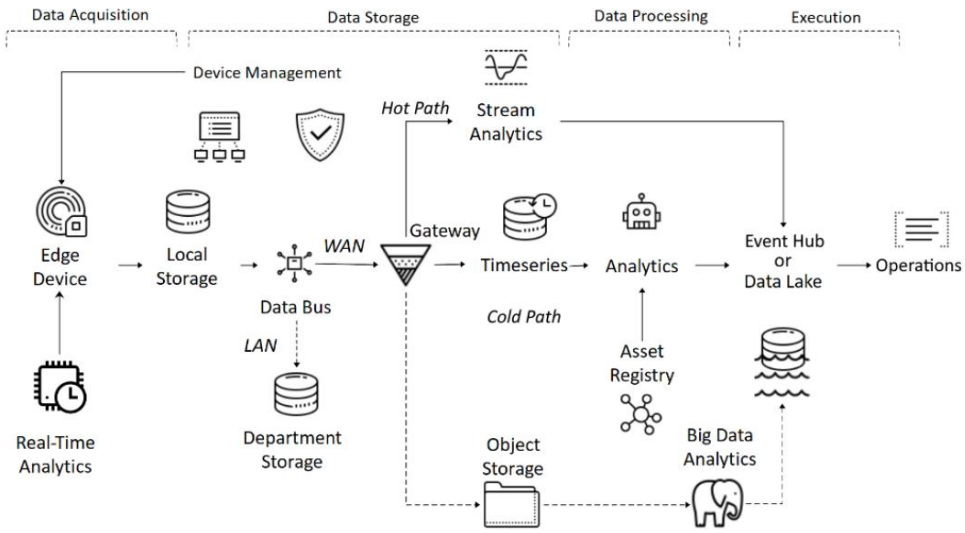


Figure 7.3 – End-to-end data flow on the IIoT

در طول انتقال داده، اطلاعات دریافتی از حسگرها از منابع داده جمع‌آوری و به‌صورت موقت ذخیره می‌شود تا در صورت بروز مشکل در اتصال، از از دست رفتن داده جلوگیری شود. این داده‌ها می‌توانند به‌صورت داده‌های سری زمانی مانند یک رویداد، داده‌های نیمه‌ساختار مانند لاگ‌ها یا فایل‌های باینری، یا داده‌های کاملاً بدون ساختار مانند تصاویر باشند. داده‌های سری زمانی و رویدادها معمولاً به‌صورت مکرر جمع‌آوری می‌شوند (از هر ثانیه تا هر چند دقیقه یک‌بار).

- **Hot Path**:

داده‌ها می‌توانند بلافاصله با استفاده از تحلیل‌های جریان داده[1] پردازش شوند که به این مسیر، «مسیر داغ» یا Hot Path گفته می‌شود. این کار با استفاده از یک پلتفرم ساده‌ی موتور قوانین[2] انجام می‌گیرد که بر اساس آستانه یا آستانه هوشمند[3] عمل می‌کند.

- **Cold Path**:

تحلیل‌های پیشرفته، از جمله دوقلوهای دیجیتال[4]، یادگیری ماشین[5]، یادگیری عمیق[6] و تحلیل‌های مبتنی بر داده یا مبتنی بر فیزیک، می‌توانند حجم زیادی از داده‌ها (از حدود ده دقیقه تا یک ماه داده) که از حسگرهای مختلف دریافت شده‌اند را پردازش

[1] Data-stream Analytics
[2] Rule Engine
[3] Smart Threshold
[4] Digital Twins
[5] Machine Learning
[6] Deep Learning

کنند. این داده‌ها در یک مخزن واسط ذخیره می‌شوند که به آن «مسیر سرد» یا Cold Path گفته می‌شود. این نوع تحلیل‌ها معمولاً توسط یک زمان‌بند[7] یا بر اساس در دسترس بودن داده‌ها فعال می‌شوند و به منابع محاسباتی زیادی نیاز دارند، از جمله سخت‌افزارهای اختصاصی مانند CPU و GPU.

Azure refers to cold paths and hot paths. In a hot path, data is processed immediately. In a cold path, data is stored and processed later. Normally, we use data streams for hot paths and micro-batch analytics for cold paths.

منبع:

صفحه ۱۷۱ کتاب Hands-On Industrial Internet of Things

معایب و مزایا استفاده از راهکارهای متن‌باز وon-premises

ده‌ها دلیل وجود دارد که ممکن است بخواهیم از راهکارهای متن‌باز و on-premises استفاده کنیم:

- **بازده سرمایه‌گذاری**[8]: ممکن است بودجه ما برای توجیه یک سرمایه‌گذاری بزرگ کافی نباشد.
- **فناوری**[9]: ممکن است بخواهیم از فناوری‌هایی استفاده کنیم که به‌طور کامل وابسته به یک تأمین‌کننده خاص نباشند.
- **حریم خصوصی**[10]: ممکن است نخواهیم داده‌ها را از کشور خارج کنیم.
- **داده‌های سایه**[11]: ممکن است به نسخه‌ای از داده‌ها روی پلتفرم قدیمی خود نیاز داشته باشیم، یا برای آزمایش یا نسخه پشتیبان.
- **یکپارچه‌سازی**[12]: ممکن است در حال پیاده‌سازی یک پلتفرم بین‌ابری باشیم، که نیاز به یک لایه یکپارچه‌سازی دارد.
- **تجربه**[13]: ممکن است در حال توسعه اولین پلتفرم اینترنت صنعتی اشیا (IIoT) باشیم، بنابراین می‌خواهیم کوچک شروع کنیم.
- **محصول**[14]: ممکن است بخواهیم محصول خود را تنها با استفاده از زیرساخت ابری توسعه دهیم.
- **مالکیت فکری**[15]: ممکن است به فضای کوچکی برای اجرای تحلیل‌ها نیاز داشته باشیم تا از دارایی فکری خود محافظت کنیم.
- **یکپارچه‌سازی با سامانه‌های قدیمی**[16]: ممکن است پلتفرم قدیمی‌ای را توسعه داده باشیم که بخواهیم آن را با پلتفرم اینترنت اشیای صنعتی خود یکپارچه کنیم.

---

[7] Scheduler
[8] Return on Investment
[9] Technology
[10] Privacy
[11] Data Shadowing
[12] Integration
[13] Experience
[14] Product
[15] Intellectual Property
[16] Legacy Integration

- **اتصال**[17]: ممکن است محدودیت‌هایی در اتصال داشته باشیم و بخواهیم ترافیک را کنترل کنیم.

معایب این روش:

- **هزینه‌های نگهداری و زیرساخت**[18]: استفاده از سرورهای داخلی نیاز به خرید سخت‌افزار، برق، سیستم خنک‌کننده، فضای فیزیکی و نیروی متخصص دارد. این موارد می‌تواند در بلندمدت هزینه‌برتر از راهکارهای ابری باشد.
- **مقیاس‌پذیری محدود**[19]: اگر نیاز به افزایش ظرفیت پردازش یا ذخیره‌سازی داشته باشیم، زیرساخت داخلی به‌سرعت به محدودیت می‌رسد. در حالی که در رایانش ابری می‌توان به‌صورت خودکار منابع را افزایش داد.
- **سرعت پیاده‌سازی پایین**[20]: نصب و راه‌اندازی سیستم‌های درون‌سازمانی زمان‌بر است، در حالی که با راهکارهای ابری می‌توان تنها با چند کلیک، محیط موردنیاز را در دسترس داشت.
- **محدودیت در دسترسی از راه دور**[21]: زیرساخت‌های داخلی ممکن است برای کاربران راه دور (مثل تیم‌های توزیع‌شده یا کار از خانه) دسترسی آسان و امن نداشته باشند، در حالی که فضای ابری طراحی‌شده برای دسترسی از هر کجاست.
- **ریسک امنیتی داخلی**[22]: در نبود استانداردهای امنیتی قوی و مدیریت صحیح، سیستم‌های داخلی می‌توانند در معرض تهدیدات بیشتری مانند حملات داخلی، بدافزار، یا اشتباهات انسانی باشند.
- **نیاز به تخصص فنی**[23]: برای نگهداری و پشتیبانی از راهکارهای درون‌سازمانی نیاز به تیم فنی مجرب است. در صورتی که در فضای ابری بسیاری از این مسئولیت‌ها توسط ارائه‌دهنده خدمات انجام می‌شود.
- **به‌روزرسانی و مدیریت نرم‌افزار**[24]: در راهکارهای داخلی، مسئولیت به‌روزرسانی سیستم‌عامل، نرم‌افزارها، وصله‌های امنیتی و غیره بر عهده سازمان است که زمان‌بر و پرهزینه است.
- **کاهش تمرکز بر کسب‌وکار**[25]: زمان و منابع صرف‌شده برای مدیریت زیرساخت داخلی ممکن است سازمان را از تمرکز بر اهداف اصلی و توسعه محصول دور کند.
- **محدودیت در قابلیت‌های پیشرفته**[26]: بسیاری از قابلیت‌های پیشرفته مانند یادگیری ماشین، هوش مصنوعی، تحلیل بلادرنگ و مقیاس‌پذیری پویا در فضای ابری در دسترس هستند ولی در سیستم‌های داخلی به‌سختی پیاده‌سازی می‌شوند.
- **بازیابی در مواقع بحران**[27]: در راهکارهای ابری، بازیابی داده‌ها معمولاً سریع‌تر و با هزینه کمتر انجام می‌شود. ولی در سیستم‌های داخلی، اگر نسخه پشتیبان مناسبی تهیه نشده باشد، بازیابی ممکن است بسیار دشوار باشد.

---

[17] Connectivity
[18] Maintenance Cost
[19] Limited Scalability
[20] Slower Deployment
[21] Limited Remote Access
[22] Internal Security Risks
[23] Need for In-House Expertise
[24] Manual Updates
[25] Less Focus on Core Business
[26] Lack of Advanced Features
[27] Disaster Recovery

منبع:

صفحه ۱۸۸ کتاب Hands-On Industrial Internet of Things

بهبود امنیت MQTT

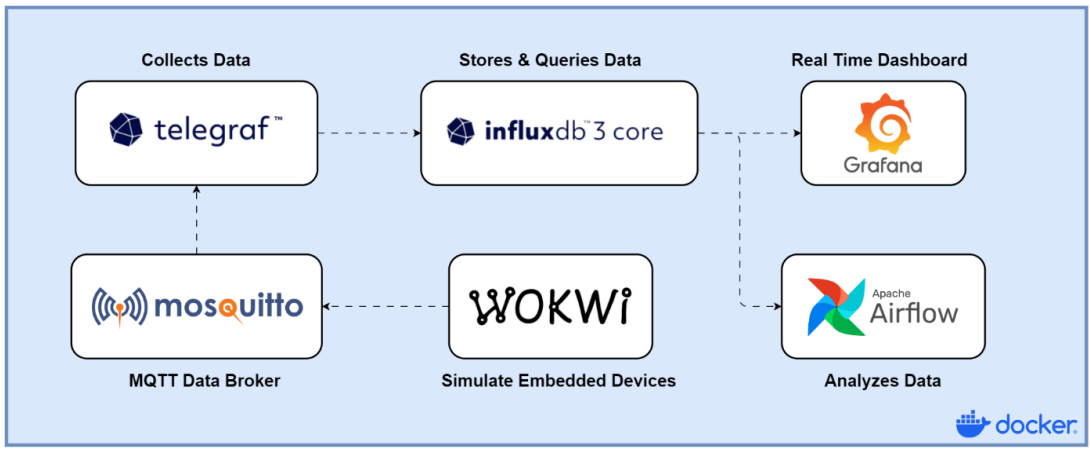
- **استفاده از MQTTS (TLS):** پروتکل MQTTS از TLS برای رمزنگاری داده‌ها در طول مسیر استفاده می‌کند. این روش، امکان شنود و دستکاری داده‌ها را به شدت کاهش می‌دهد.
- **گواهی‌های X.509 برای احراز هویت:** در AWS IoT Core و سایر پلتفرم‌ها، استفاده از گواهی X.509 برای شناسایی دستگاه‌ها و اعتبارسنجی آن‌ها توصیه می‌شود. گواهی‌ها شامل کلید عمومی و خصوصی هستند که برای رمزنگاری و امضا استفاده می‌شوند
- **سیاست‌های محدود دسترسی (IAM Policies):** سیاست‌های دسترسی مانند `iot:Connect`، `iot:Publish`، `iot:Subscribe` و `iot:Receive` باید فقط برای منابع خاص مجاز باشند. استفاده از `iot:*` به شدت توصیه نمی‌شود چرا که منجر به افزایش سطح حمله می‌شود.
- **پیکربندی TLS در کلاینت Node-RED:** برای اتصال ایمن، باید گواهی‌های ریشه (CA) را به کلاینت‌های MQTT (مانند Node-RED) اضافه کرد تا ارتباط TLS به درستی برقرار شود.
- **ایزوله‌سازی شبکه فایروال (DMZ):** قرار دادن دستگاه‌های edge در ناحیه DMZ با دو فایروال (یکی در برابر اینترنت و یکی بین edge و شبکه صنعتی) باعث محدود شدن حملات می‌شود.

منبع:

فصل ۵ کتاب Hands-On Industrial Internet of Things

بخش ۲

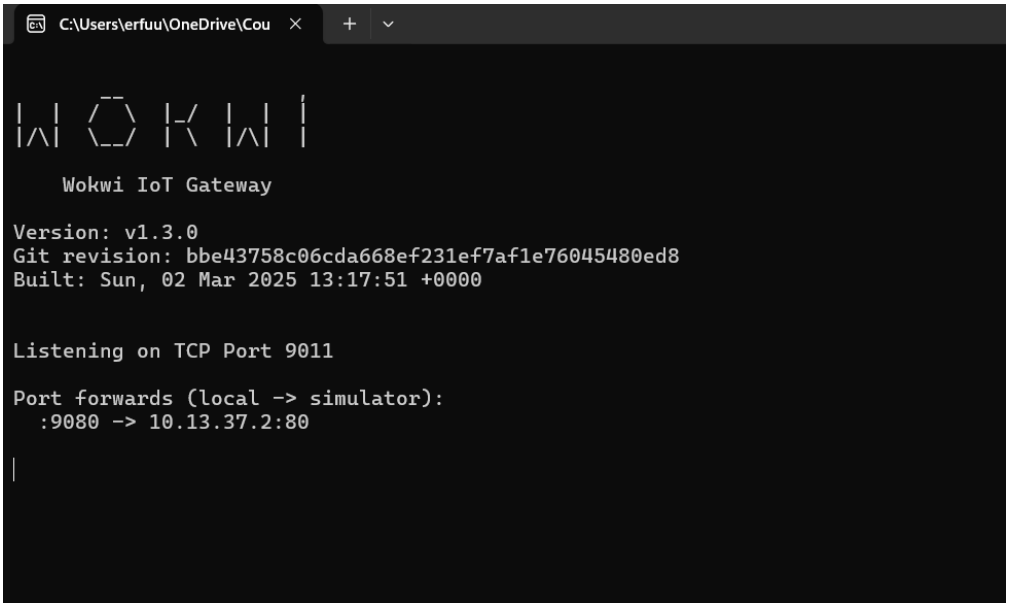
نمودار کلی معماری



پیکربندی ابزار شبیه‌سازی Wokwi

wokwi.toml

```
[wokwi]
version = 1
firmware = '.pio/build/esp32dev/firmware.bin'
elf = '.pio/build/esp32dev/firmware.elf'

[net]
gateway="ws://localhost:9011"
```



پیکربندی بروکر داده Mosquitto

mosquitto.conf

```
persistence true
persistence_location /mosquitto/data/

log_dest file /mosquitto/log/mosquitto.log
log_dest stdout

listener 1883

allow_anonymous true
```

docker-composer.yaml

```yaml
 mosquitto:
    image: eclipse-mosquitto:latest
    ports:
      - "${MOSQUITTO_PORT:-1883}:1883"
    volumes:
      - ./mosquitto/config:/mosquitto/config:rw
      - ./mosquitto/data:/mosquitto/data:rw
      - ./mosquitto/log:/mosquitto/log:rw
    networks:
      - iot-ex2-net
```

پیکربندی ابزار جمع‌آوری داده Telegraf

telegraf.conf

```
[agent]
  interval = "${TELEGRAF_COLLECTION_INTERVAL}"
  round_interval = true
  metric_batch_size = 1000
  metric_buffer_limit = 10000
  collection_jitter = "0s"
  flush_interval = "${TELEGRAF_COLLECTION_INTERVAL}"
  flush_jitter = "0s"
  precision = ""
  hostname = ""
  omit_hostname = false

[[outputs.influxdb_v2]]
  urls = ["http://${INFLUXDB_HOST}:${INFLUXDB_HTTP_PORT}"]
  token = "${INFLUXDB_TOKEN}"
  organization = "${INFLUXDB_ORG}"
```

```
  bucket = "${INFLUXDB_BUCKET}"

[[inputs.mqtt_consumer]]
  servers = ["tcp://${MOSQUITTO_HOST}:${MOSQUITTO_PORT}"]
  topics = [
    "sensors/greenhouse/dht22/#"
  ]
  topic_tag = "topic"
  client_id = "telegraf"
  data_format="json"

  [[inputs.mqtt_consumer.topic_parsing]]
    topic = "sensors/greenhouse/dht22/#"
    measurement = "measurement/_/_/_"
    tags = "_/location/sensorType/sensorID"
```

docker-composer.yaml

```
  telegraf:
    image: telegraf:latest
    volumes:
      - ./telegraf/telegraf.conf:/etc/telegraf/telegraf.conf:rw
    environment:
      - INFLUXDB_HOST=${INFLUXDB_HOST}
      - INFLUXDB_HTTP_PORT=${INFLUXDB_HTTP_PORT}
      - INFLUXDB_TOKEN=${INFLUXDB_TOKEN}
      - INFLUXDB_ORG=${INFLUXDB_ORG}
      - INFLUXDB_BUCKET=${INFLUXDB_BUCKET}
      - MOSQUITTO_HOST=${MOSQUITTO_HOST}
      - MOSQUITTO_PORT=${MOSQUITTO_PORT}
      - TELEGRAF_COLLECTION_INTERVAL=${TELEGRAF_COLLECTION_INTERVAL}
      - HOSTNAME=telegraf
    depends_on:
      - influxdb
      - mosquitto
    networks:
      - iot-ex2-net
```

پیکربندی پایگاه‌داده InfluxDB

docker-composer.yaml

```
influxdb:
    image: influxdb:2
    ports:
      - "${INFLUXDB_HTTP_PORT:-8086}:8086"
    environment:
```

```
      - DOCKER_INFLUXDB_INIT_MODE=setup
      - DOCKER_INFLUXDB_INIT_USERNAME=${INFLUXDB_INIT_USERNAME}
      - DOCKER_INFLUXDB_INIT_PASSWORD=${INFLUXDB_INIT_PASSWORD}
      - DOCKER_INFLUXDB_INIT_ADMIN_TOKEN=${INFLUXDB_TOKEN}
      - DOCKER_INFLUXDB_INIT_ORG=${INFLUXDB_ORG}
      - DOCKER_INFLUXDB_INIT_BUCKET=${INFLUXDB_BUCKET}
    volumes:
      - ./influxdb/data:/var/lib/influxdb2:rw
      - ./influxdb/config:/etc/influxdb2:rw
    networks:
      - iot-ex2-net
```

پیکربندی ابزار دیداری‌سازی Grafana

docker-composer.yaml

```
  grafana:
    image: grafana/grafana:latest
    ports:
      - "${GRAFANA_PORT:-3000}:3000"
    volumes:
      - ./grafana/data:/var/lib/grafana
    environment:
      - GF_SECURITY_ADMIN_USER=${GRAFANA_ADMIN_USER}
      - GF_SECURITY_ADMIN_PASSWORD=${GRAFANA_ADMIN_PASSWORD}
    depends_on:
      - influxdb
    networks:
      - iot-ex2-net

networks:
  iot-ex2-net:
    external: true
```

پیکربندی ابزار تحلیل Airflow

docker-composer.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:3.0.0
# 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:3.0.0}
  # build: .
  environment:
    &airflow-common-env
    AIRFLOW__API_AUTH__JWT_SECRET: VYssn3BNaiZkdWGIFrDV1w==
    AIRFLOW__WEBSERVER__SECRET_KEY: aj0F0i3dHIZyAdFr72xHMQ==
    AIRFLOW__CORE__INTERNAL_API_SECRET_KEY: aj0F0i3dHIZyAdFr72xHMQ==
    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: bFHppMvwV4trOTG-seHhJkYOpjJqGaXtJc38E4cO-Ho=
    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/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:-}
    # 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
    networks:
      - iot-ex2-net

  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
  networks:
    - iot-ex2-net

airflow-apiserver:
  <<: *airflow-common
  command: api-server
  ports:
    - "8080:8080"
  environment:
    <<: *airflow-common-env
  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
  networks:
    - iot-ex2-net

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
  environment:
    <<: *airflow-common-env
  depends_on:
    <<: *airflow-common-depends-on
    airflow-init:
      condition: service_completed_successfully
  networks:
    - iot-ex2-net
```

```
  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
    networks:
      - iot-ex2-net

  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
    networks:
      - iot-ex2-net

  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
    networks:
      - iot-ex2-net

  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 /opts/airflow/{logs,dags,plugins,config}
        chown -R "${AIRFLOW_UID}:0" /opts/airflow/{logs,dags,plugins,config}
        exec /entrypoint airflow version
    # yamllint enable rule:line-length
    environment:
      <<: *airflow-common-env
```

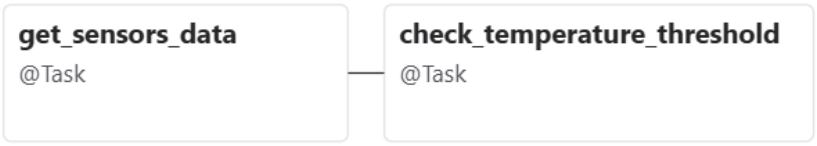
```
      _AIRFLOW_DB_MIGRATE: 'true'
      _AIRFLOW_WWW_USER_CREATE: 'true'
      _AIRFLOW_WWW_USER_USERNAME: ${_AIRFLOW_WWW_USER_USERNAME:-airflow}
      _AIRFLOW_WWW_USER_PASSWORD: ${_AIRFLOW_WWW_USER_PASSWORD:-airflow}
      _PIP_ADDITIONAL_REQUIREMENTS: ''
    user: "0:0"
    networks:
      - iot-ex2-net

  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
    networks:
      - iot-ex2-net

  # 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
    networks:
      - iot-ex2-net

networks:
  iot-ex2-net:
    external: true

volumes:
  postgres-db-volume:
```

کد Hot Path DAG

```
import logging
import pendulum
import pandas as pd

from airflow.decorators import dag, task
from airflow.providers.influxdb.hooks.influxdb import InfluxDBHook

TABLE = 'sensors'
BUCKET = 'iot-ex2'
TIME_RANGE = '1m'
ALLOWED_RANGE = (15, 35)

@dag(
    schedule="*/1 * * * *",
    start_date=pendulum.datetime(2025, 1, 1, tz="UTC"),
    catchup=False,
    tags=["hotpath", "influxdb2", "temperature"],
)
def temperature_hotpath():

    @task()
    def get_sensors_data():
        influx_hook = InfluxDBHook(conn_id="influxdb")

        flux_query = '''
        from(bucket: "{BUCKET}")
        |> range(start: -{TIME_RANGE})
        |> filter(fn: (r) => r._measurement == "{TABLE}")
        |> filter(fn: (r) => r._field == "t")
        |> sort(columns: ["_time"])
```

```
        '''

        df = influx_hook.query_to_df(flux_query)

        # Convert Timestamp columns to strings to make XCom-safe
        for col in df.columns:
            if pd.api.types.is_datetime64_any_dtype(df[col]):
                df[col] = df[col].astype(str)

        if "_value" not in df.columns or "_time" not in df.columns:
            raise ValueError("Expected '_value' and '_time' in query result")

        return df[["_value", "_time", "sensorID"]].rename(columns={"_value":
"t"}).to_dict(orient="records")

    @task()
    def check_temperature_threshold(rows: list[dict]):
        for row in rows:
            temp = row.get("t")
            ts = row.get("_time")
            sensor_id = row.get("sensorID", "unknown")

            if temp is None:
                logging.warning(f"Missing temperature value for sensor
{sensor_id} at {ts}")
                continue

            if not (ALLOWED_RANGE[0] <= temp <= ALLOWED_RANGE[1]):
                logging.warning(f"sensor {sensor_id} = {temp}°C at {ts} (out of
range)")
            else:
                logging.info(f"sensor {sensor_id} = {temp}°C at {ts} (within
range)")

    data = get_sensors_data()
    check_temperature_threshold(data)

temperature_hotpath()
```



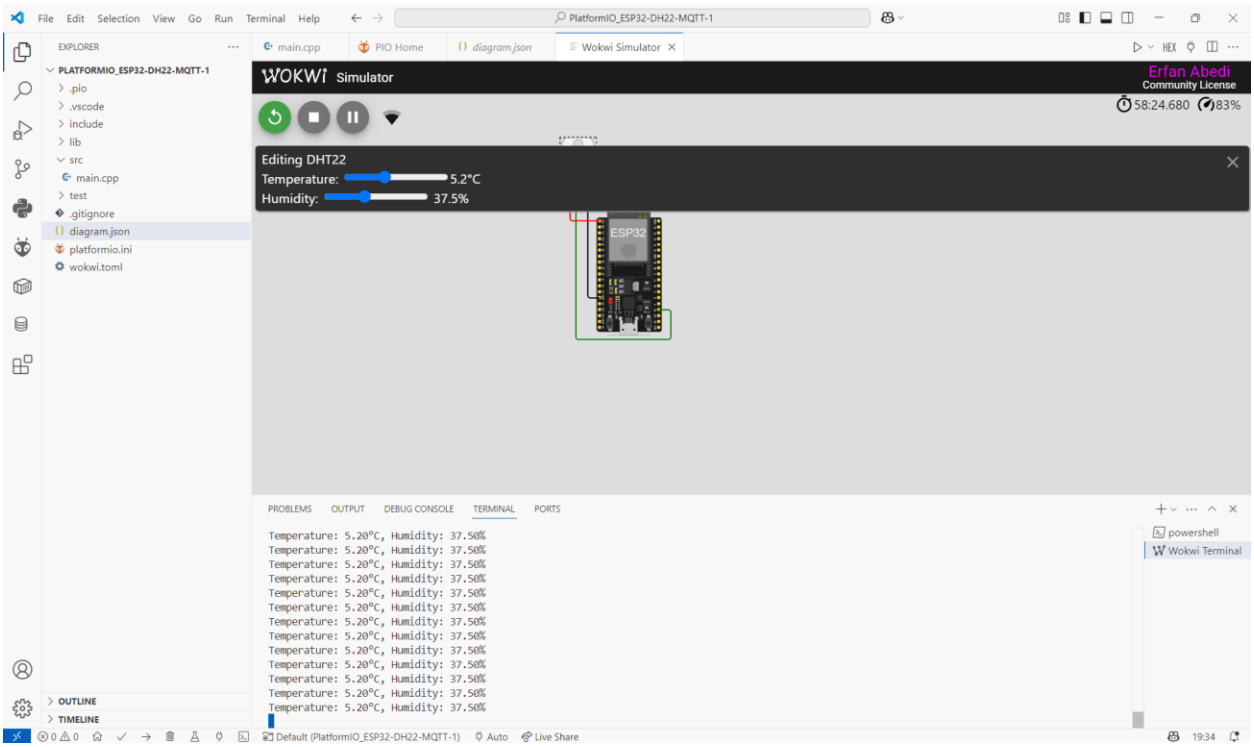
کد Cold Path DAG

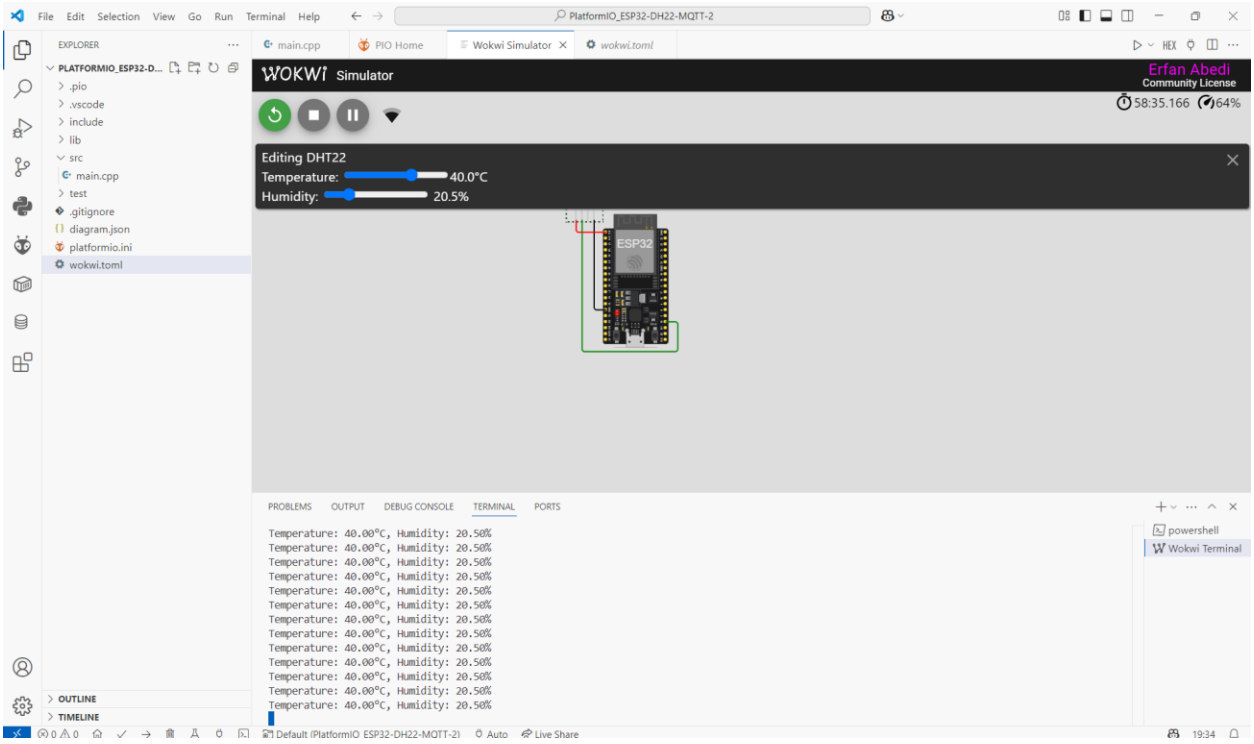
```
import logging
import numpy as np
import pandas as pd
import pendulum

from airflow.decorators import dag, task
from airflow.providers.influxdb.hooks.influxdb import InfluxDBHook

TABLE = 'sensors'
BUCKET = 'iot-ex2'
TIME_RANGE = '1h'
SENSOR_ID_1 = '1'
SENSOR_ID_2 = '2'

@dag(
    schedule="0 * * * *",
    start_date=pendulum.datetime(2025, 1, 1, tz="UTC"),
    catchup=False,
    tags=["coldpath", "influxdb2", "taskflow"],
)
def temperature_cold_path():

    @task()
    def get_sensor_data(sensor_id: str) -> list[dict]:
        import pandas as pd
        influx_hook = InfluxDBHook(conn_id="influxdb")

        flux_query = f'''
        from(bucket: "{BUCKET}")
        |> range(start: -{TIME_RANGE})
        |> filter(fn: (r) => r._measurement == "{TABLE}")
        |> filter(fn: (r) => r["sensorID"] == "{sensor_id}")
        |> filter(fn: (r) => r._field == "t")
        |> sort(columns: ["_time"])
        '''
```

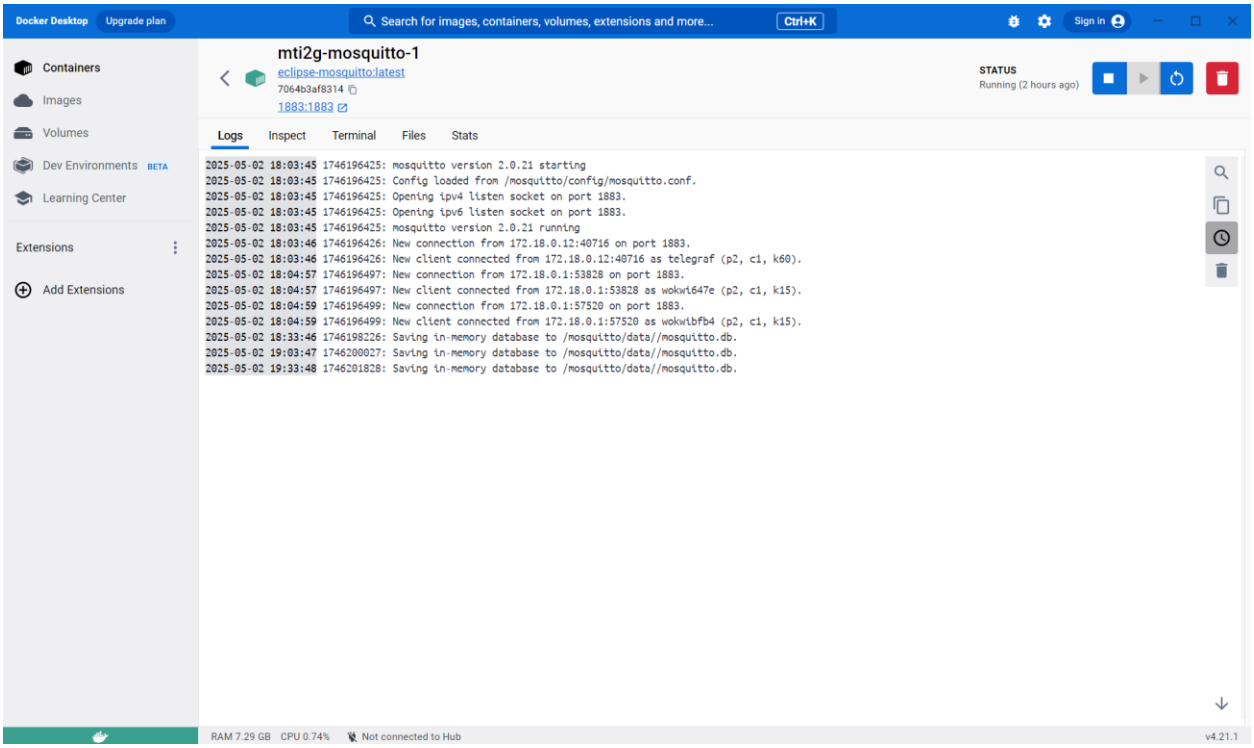
```
        df = influx_hook.query_to_df(flux_query)

        # Ensure datetime columns are XCom-safe
        for col in df.columns:
            if pd.api.types.is_datetime64_any_dtype(df[col]):
                df[col] = df[col].astype(str)

        # Keep only necessary columns for comparison
        if "_value" not in df.columns or "_time" not in df.columns:
            raise ValueError("Expected '_value' and '_time' in query result")

        return df[["_value", "_time"]].rename(columns={"_value":
"t"}).to_dict(orient="records")


    @task()
    def compare_and_detect(sensor_a_data: list[dict], sensor_b_data: list[dict])
-> None:
        values_a = [row["t"] for row in sensor_a_data]
        values_b = [row["t"] for row in sensor_b_data]

        if not values_a or not values_b:
            logging.warning("One or both sensor data lists are empty.")
            return

        mean_a = np.mean(values_a)
        mean_b = np.mean(values_b)

        p95_a = np.percentile(values_a, 95)
        p95_b = np.percentile(values_b, 95)

        diff_means = abs(mean_a - mean_b)
        diff_p95s = abs(p95_a - p95_b)

        logging.info(f"Mean A: {mean_a:.2f}, Mean B: {mean_b:.2f}, Diff:
{diff_means:.2f}")
        logging.info(f"P95 A: {p95_a:.2f}, P95 B: {p95_b:.2f}, Diff:
{diff_p95s:.2f}")

        if diff_means > diff_p95s:
            logging.warning(f"Difference in means ({diff_means:.2f}) exceeds
difference in P95s ({diff_p95s:.2f})")
        else:
```

```
            logging.info(f"Difference in means ({diff_means:.2f}) is within
expected variation ({diff_p95s:.2f})")


    data_a = get_sensor_data(SENSOR_ID_1)
    data_b = get_sensor_data(SENSOR_ID_2)
    compare_and_detect(data_a, data_b)

temperature_cold_path()
```

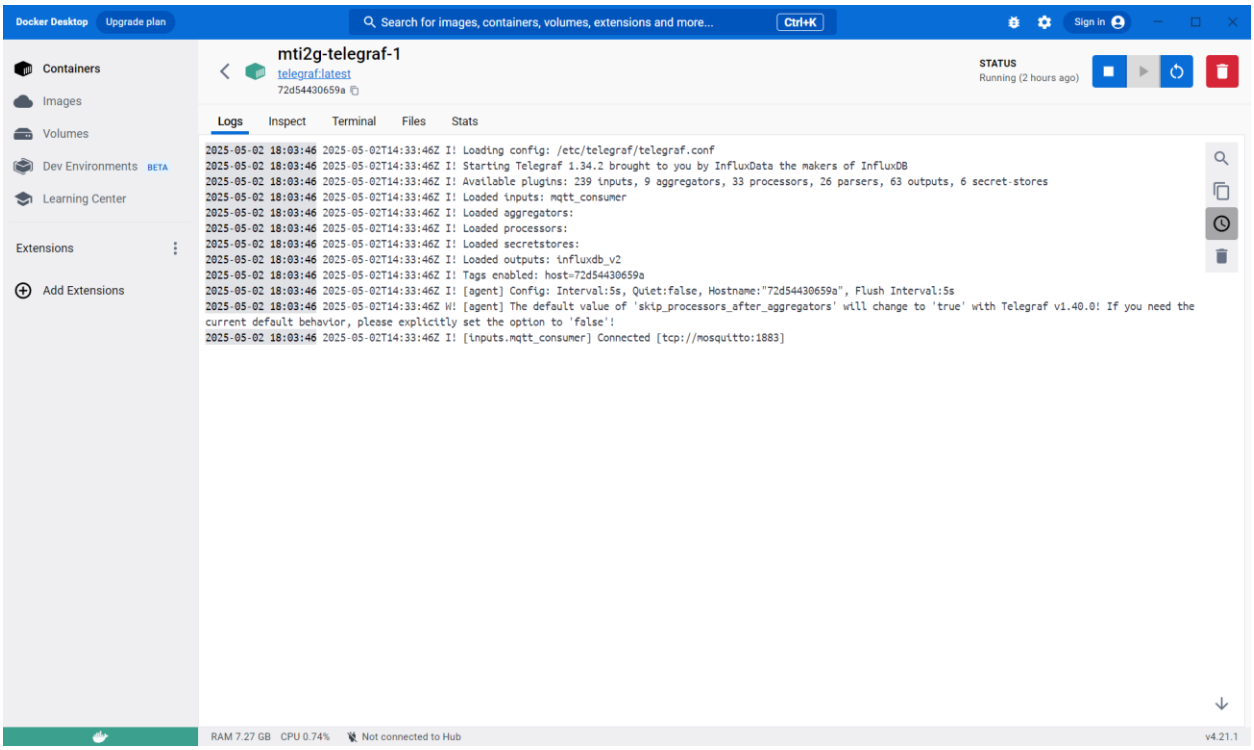
شبیه‌سازی و ارسال داده

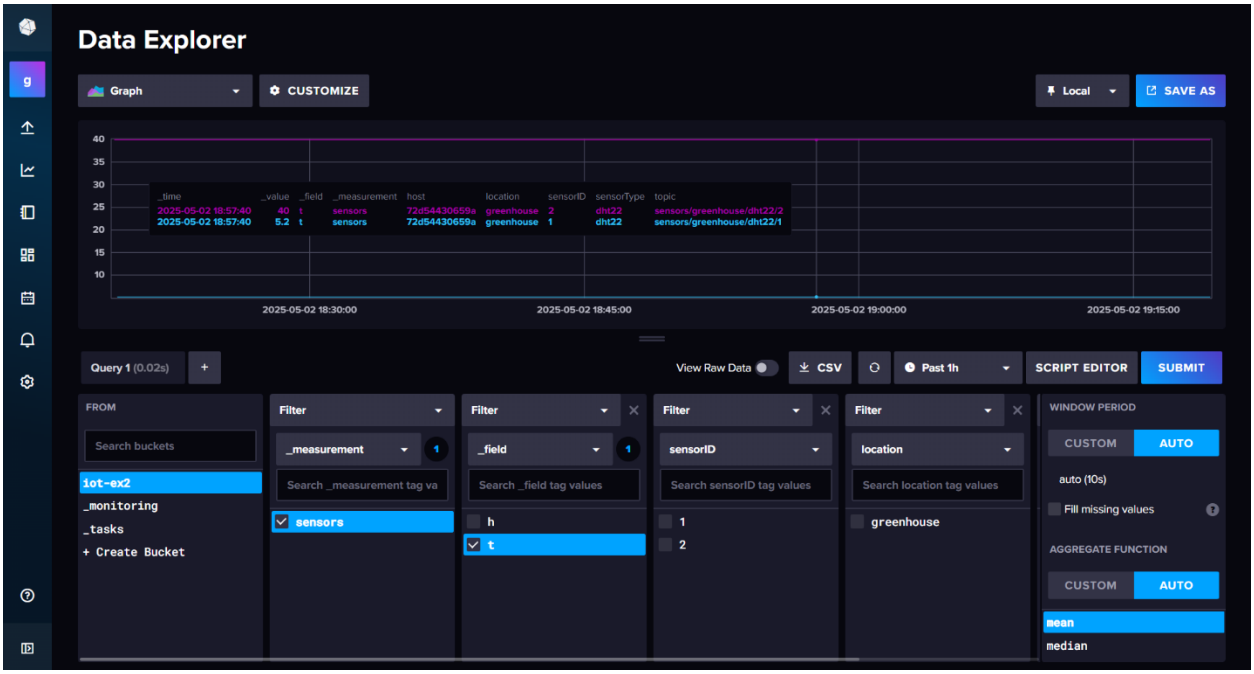



File Edit Selection View Go Run Terminal Help
PlatformIO_ESP32-DH22-MQTT-2
EXPLORER
PLATFORMIO_ESP32-D...
.pio
.vscode
include
lib
src
main.cpp
test
.gitignore
diagram.json
platformio.ini
wokwi.toml
main.cpp
PIO Home
Wokwi Simulator
wokwi.toml
WOKWI Simulator
Erfan Abedi
Community License
58:35.166
64%
Editing DHT22
Temperature: 40.0°C
Humidity: 20.5%
ESP32
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Temperature: 40.00°C, Humidity: 20.50%
powershell
Wokwi Terminal
OUTLINE
TIMELINE
Default (PlatformIO ESP32-DH22-MQTT-2)
Auto
Live Share
19:34

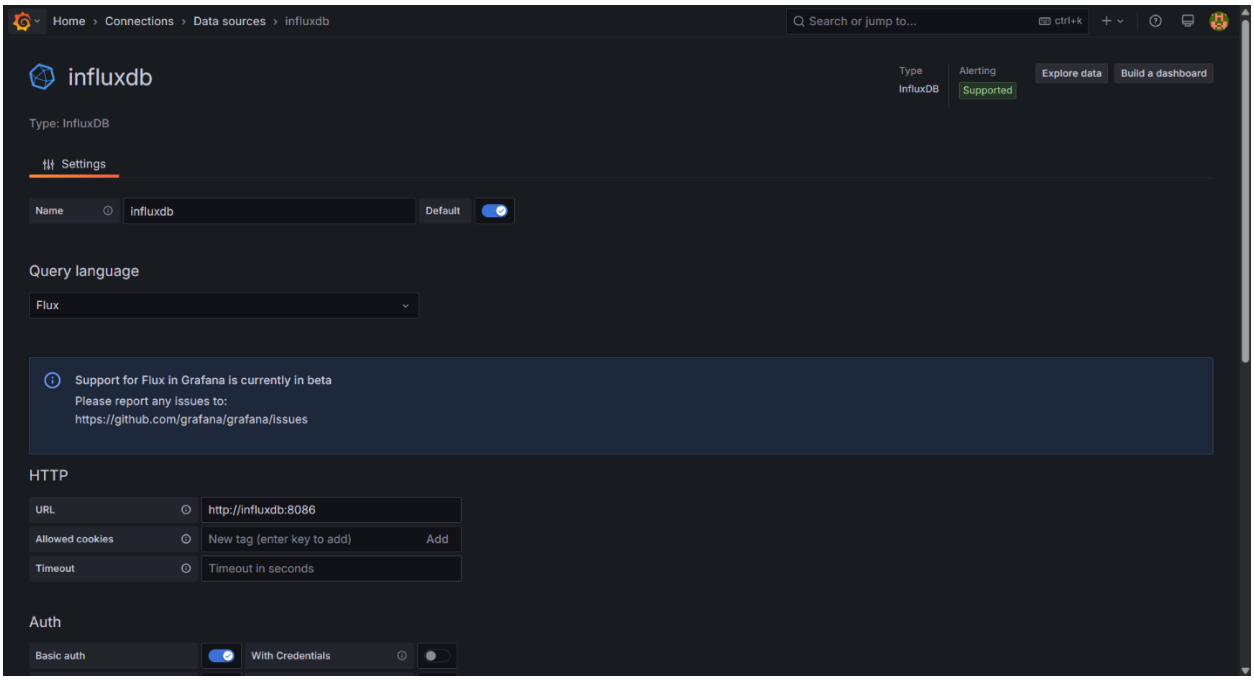

Docker Desktop
Upgrade plan
Search for images, containers, volumes, extensions and more...
Ctrl+K
Sign in
Containers
Images
Volumes
Dev Environments BETA
Learning Center
Extensions
Add Extensions
mti2g-mosquitto-1
eclipse-mosquitto:latest
7064b3af8314
1883:1883
STATUS
Running (2 hours ago)
Logs Inspect Terminal Files Stats
2025-05-02 18:03:45 1746196425: mosquitto version 2.0.21 starting
2025-05-02 18:03:45 1746196425: Config loaded from /mosquitto/config/mosquitto.conf.
2025-05-02 18:03:45 1746196425: Opening ipv4 listen socket on port 1883.
2025-05-02 18:03:45 1746196425: Opening ipv6 listen socket on port 1883.
2025-05-02 18:03:45 1746196425: mosquitto version 2.0.21 running
2025-05-02 18:03:46 1746196426: New connection from 172.18.0.12:40716 on port 1883.
2025-05-02 18:03:46 1746196426: New client connected from 172.18.0.12:40716 as telegraf (p2, c1, k60).
2025-05-02 18:04:57 1746196497: New connection from 172.18.0.1:53828 on port 1883.
2025-05-02 18:04:57 1746196497: New client connected from 172.18.0.1:53828 as wokwi647e (p2, c1, k15).
2025-05-02 18:04:59 1746196499: New connection from 172.18.0.1:57520 on port 1883.
2025-05-02 18:04:59 1746196499: New client connected from 172.18.0.1:57520 as wokwibfb4 (p2, c1, k15).
2025-05-02 18:33:46 1746198226: Saving in-memory database to /mosquitto/data//mosquitto.db.
2025-05-02 19:03:47 1746200027: Saving in-memory database to /mosquitto/data//mosquitto.db.
2025-05-02 19:33:48 1746201828: Saving in-memory database to /mosquitto/data//mosquitto.db.
RAM 7.29 GB CPU 0.74%
Not connected to Hub
v4.21.1

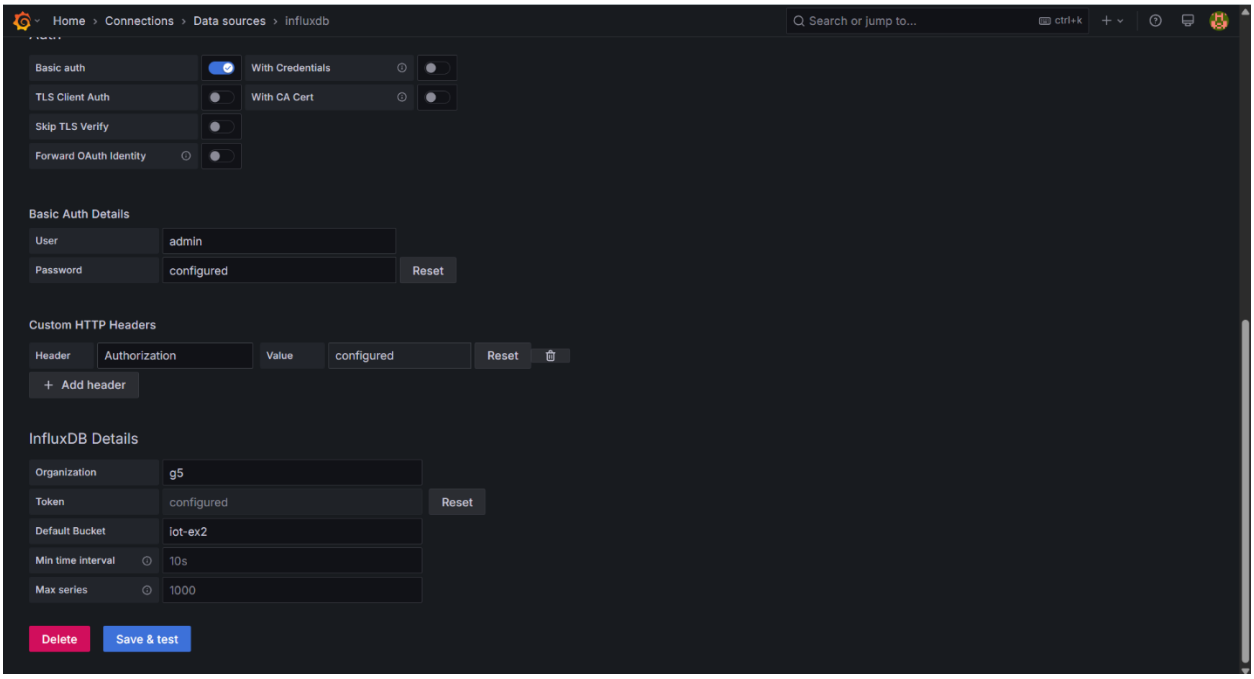
دریافت داده و انتقال آن به پایگاه‌داده

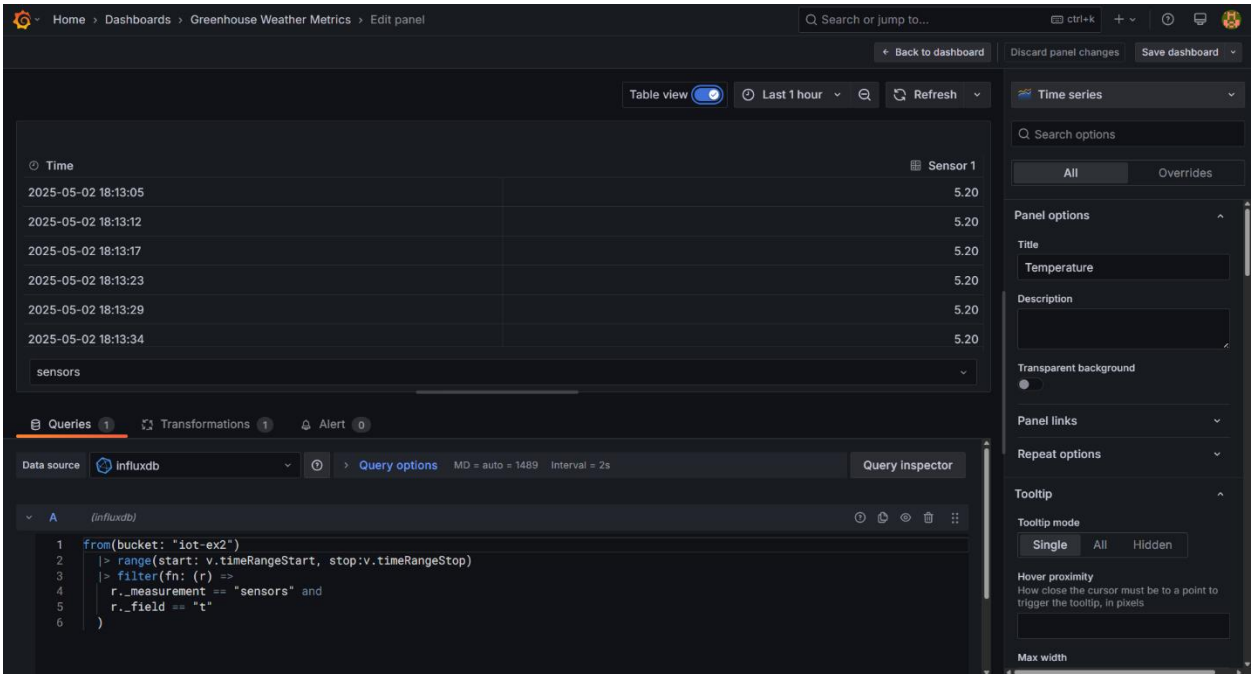


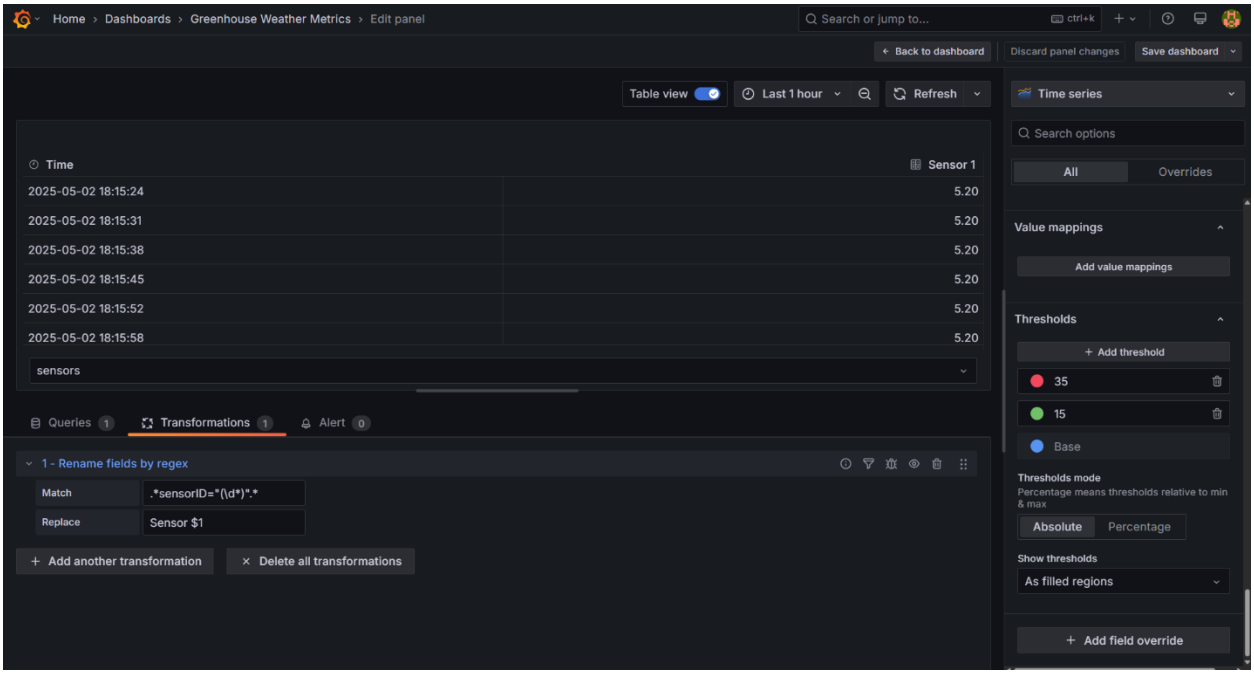


دیداری‌سازی داده در نمودارهای داشبورد

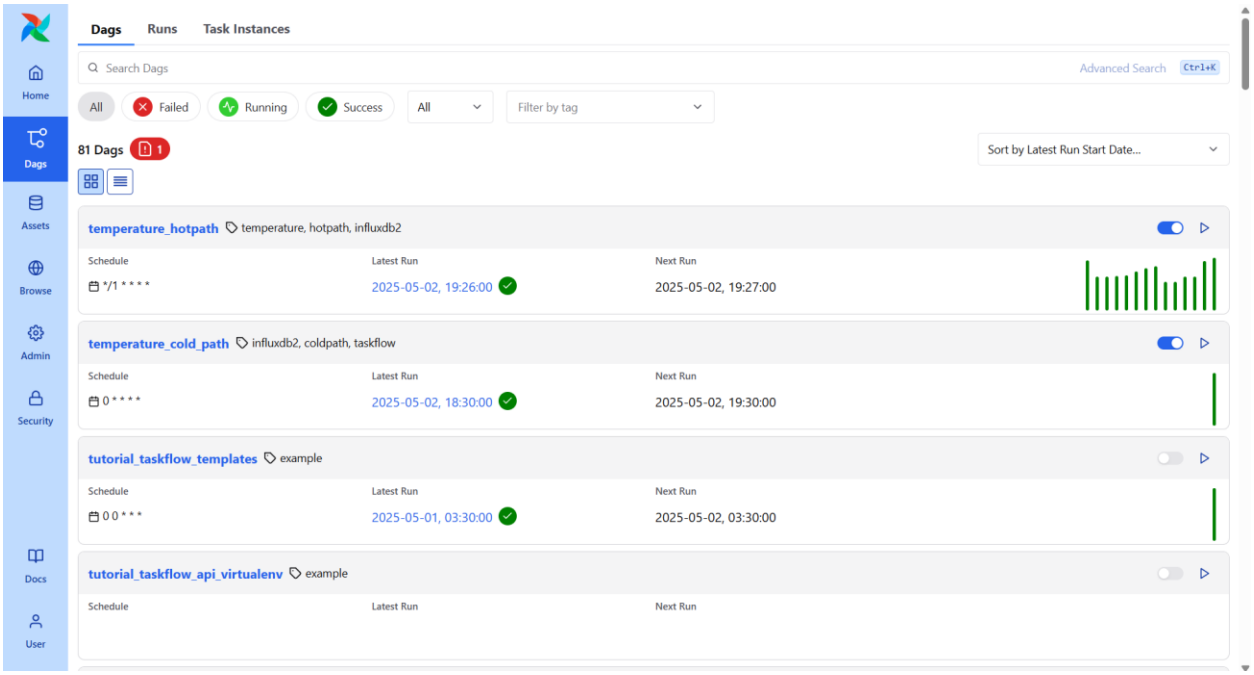


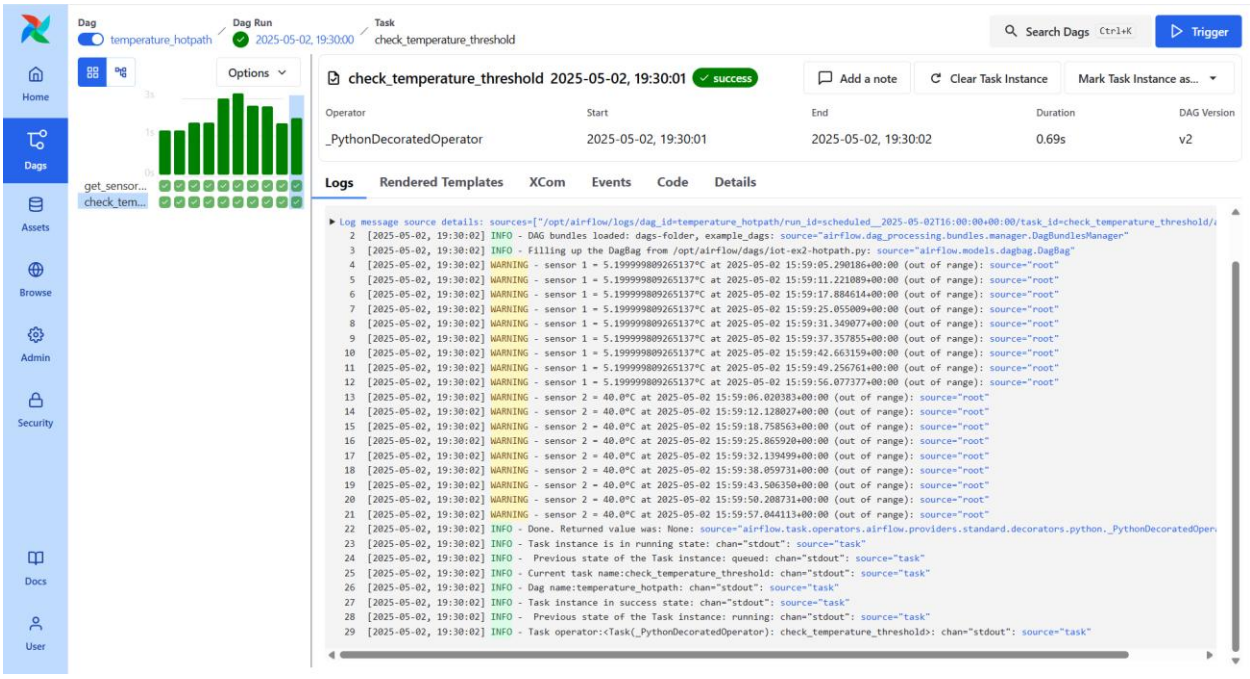


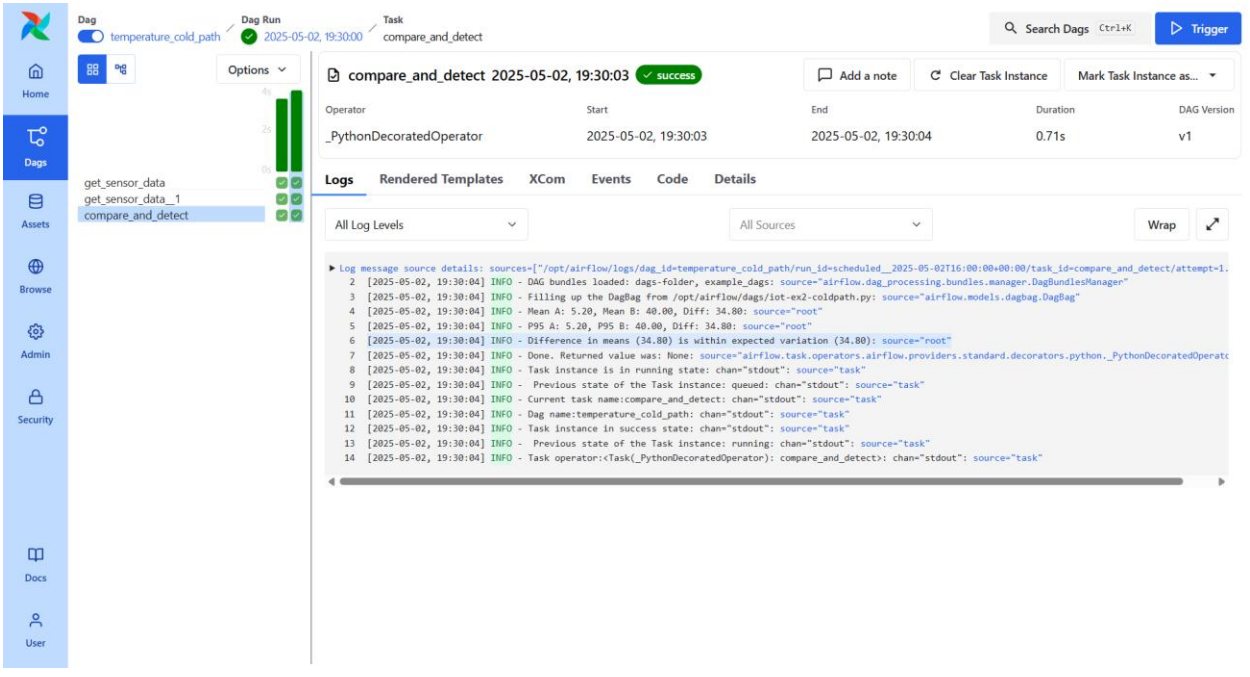

Home › Dashboards › Greenhouse Weather Metrics › Edit panel
Search or jump to... ctrl+k
Back to dashboard
Discard panel changes
Save dashboard
Table view
Last 1 hour
Refresh
Time series
Search options
All
Overrides
Time
Sensor 1
2025-05-02 18:13:05 5.20
2025-05-02 18:13:12 5.20
2025-05-02 18:13:17 5.20
2025-05-02 18:13:23 5.20
2025-05-02 18:13:29 5.20
2025-05-02 18:13:34 5.20
sensors
Queries 1
Transformations 1
Alert 0
Data source
influxdb
Query options
MD = auto = 1489
Interval = 2s
Query inspector
A
(influxdb)
from(bucket: "iot-ex2")
|> range(start: v.timeRangeStart, stop:v.timeRangeStop)
|> filter(fn: (r) =>
r._measurement == "sensors" and
r._field == "t"
)
Panel options
Title
Temperature
Description
Transparent background
Panel links
Repeat options
Tooltip
Tooltip mode
Single
All
Hidden
Hover proximity
How close the cursor must be to a point to trigger the tooltip, in pixels
Max width


Home › Dashboards › Greenhouse Weather Metrics › Edit panel
Search or jump to... ctrl+k
Back to dashboard
Discard panel changes
Save dashboard
Table view
Last 1 hour
Refresh
Time series
Search options
All
Overrides
Time
Sensor 1
2025-05-02 18:15:24 5.20
2025-05-02 18:15:31 5.20
2025-05-02 18:15:38 5.20
2025-05-02 18:15:45 5.20
2025-05-02 18:15:52 5.20
2025-05-02 18:15:58 5.20
sensors
Queries 1
Transformations 1
Alert 0
1 - Rename fields by regex
Match
.*sensorID="(\d*)".*
Replace
Sensor $1
Add another transformation
Delete all transformations
Value mappings
Add value mappings
Thresholds
Add threshold
35
15
Base
Thresholds mode
Percentage means thresholds relative to min & max
Absolute
Percentage
Show thresholds
As filled regions
Add field override

گزارش ناهنجاری و تولید هشدار




Dag
temperature_hotpath
Dag Run
2025-05-02, 19:30:00
Task
check_temperature_threshold
Search Dags Ctrl+K
Trigger
Home
Dags
Assets
Browse
Admin
Security
Docs
User
Options
get_sensor...
check_tem...
check_temperature_threshold 2025-05-02, 19:30:01
success
Add a note
Clear Task Instance
Mark Task Instance as...
Operator
Start
End
Duration
DAG Version
_PythonDecoratedOperator
2025-05-02, 19:30:01
2025-05-02, 19:30:02
0.69s
v2
Logs
Rendered Templates
XCom
Events
Code
Details
Log message source details: sources=["/opt/airflow/logs/dag_id=temperature_hotpath/run_id=scheduled__2025-05-02T16:00:00+00:00/task_id=check_temperature_threshold/
[2025-05-02, 19:30:02] INFO - DAG bundles loaded: dags-folder, example_dags: source="airflow.dag_processing.bundles.manager.DagBundlesManager"
[2025-05-02, 19:30:02] INFO - Filling up the DagBag from /opt/airflow/dags/iot-ex2-hotpath.py: source="airflow.models.dagbag.DagBag"
[2025-05-02, 19:30:02] WARNING - sensor 1 = 5.1999998092651377°C at 2025-05-02 15:59:05.290186+00:00 (out of range): source="root"
[2025-05-02, 19:30:02] WARNING - sensor 1 = 5.1999998092651377°C at 2025-05-02 15:59:11.221089+00:00 (out of range): source="root"
[2025-05-02, 19:30:02] WARNING - sensor 1 = 5.1999998092651377°C at 2025-05-02 15:59:17.884614+00:00 (out of range): source="root"
[2025-05-02, 19:30:02] WARNING - sensor 1 = 5.1999998092651377°C at 2025-05-02 15:59:25.055009+00:00 (out of range): source="root"
[2025-05-02, 19:30:02] WARNING - sensor 1 = 5.1999998092651377°C at 2025-05-02 15:59:31.349077+00:00 (out of range): source="root"
[2025-05-02, 19:30:02] WARNING - sensor 1 = 5.1999998092651377°C at 2025-05-02 15:59:37.357855+00:00 (out of range): source="root"
[2025-05-02, 19:30:02] WARNING - sensor 1 = 5.1999998092651377°C at 2025-05-02 15:59:42.663159+00:00 (out of range): source="root"
[2025-05-02, 19:30:02] WARNING - sensor 1 = 5.1999998092651377°C at 2025-05-02 15:59:49.256761+00:00 (out of range): source="root"
[2025-05-02, 19:30:02] WARNING - sensor 1 = 5.1999998092651377°C at 2025-05-02 15:59:56.077377+00:00 (out of range): source="root"
[2025-05-02, 19:30:02] WARNING - sensor 2 = 40.0°C at 2025-05-02 15:59:06.020383+00:00 (out of range): source="root"
[2025-05-02, 19:30:02] WARNING - sensor 2 = 40.0°C at 2025-05-02 15:59:12.128027+00:00 (out of range): source="root"
[2025-05-02, 19:30:02] WARNING - sensor 2 = 40.0°C at 2025-05-02 15:59:18.758563+00:00 (out of range): source="root"
[2025-05-02, 19:30:02] WARNING - sensor 2 = 40.0°C at 2025-05-02 15:59:25.865920+00:00 (out of range): source="root"
[2025-05-02, 19:30:02] WARNING - sensor 2 = 40.0°C at 2025-05-02 15:59:32.139499+00:00 (out of range): source="root"
[2025-05-02, 19:30:02] WARNING - sensor 2 = 40.0°C at 2025-05-02 15:59:38.059731+00:00 (out of range): source="root"
[2025-05-02, 19:30:02] WARNING - sensor 2 = 40.0°C at 2025-05-02 15:59:43.506350+00:00 (out of range): source="root"
[2025-05-02, 19:30:02] WARNING - sensor 2 = 40.0°C at 2025-05-02 15:59:50.208731+00:00 (out of range): source="root"
[2025-05-02, 19:30:02] WARNING - sensor 2 = 40.0°C at 2025-05-02 15:59:57.044113+00:00 (out of range): source="root"
[2025-05-02, 19:30:02] INFO - Done. Returned value was: None: source="airflow.task.operators.airflow.providers.standard.decorators.python._PythonDecoratedOper
[2025-05-02, 19:30:02] INFO - Task instance is in running state: chan="stdout": source="task"
[2025-05-02, 19:30:02] INFO - Previous state of the Task instance: queued: chan="stdout": source="task"
[2025-05-02, 19:30:02] INFO - Current task name:check_temperature_threshold: chan="stdout": source="task"
[2025-05-02, 19:30:02] INFO - Dag name:temperature_hotpath: chan="stdout": source="task"
[2025-05-02, 19:30:02] INFO - Task instance in success state: chan="stdout": source="task"
[2025-05-02, 19:30:02] INFO - Previous state of the Task instance: running: chan="stdout": source="task"
[2025-05-02, 19:30:02] INFO - Task operator:<Task(_PythonDecoratedOperator): check_temperature_threshold>: chan="stdout": source="task"


Dag
temperature_cold_path
Dag Run
2025-05-02, 19:30:00
Task
compare_and_detect
Search Dags Ctrl+K
Trigger
Home
Dags
Assets
Browse
Admin
Security
Docs
User
Options
get_sensor_data
get_sensor_data__1
compare_and_detect
compare_and_detect 2025-05-02, 19:30:03
success
Add a note
Clear Task Instance
Mark Task Instance as...
Operator
Start
End
Duration
DAG Version
_PythonDecoratedOperator
2025-05-02, 19:30:03
2025-05-02, 19:30:04
0.71s
v1
Logs
Rendered Templates
XCom
Events
Code
Details
All Log Levels
All Sources
Wrap
Log message source details: sources=["/opt/airflow/logs/dag_id=temperature_cold_path/run_id=scheduled__2025-05-02T16:00:00+00:00/task_id=compare_and_detect/attempt=1.
[2025-05-02, 19:30:04] INFO - DAG bundles loaded: dags-folder, example_dags: source="airflow.dag_processing.bundles.manager.DagBundlesManager"
[2025-05-02, 19:30:04] INFO - Filling up the DagBag from /opt/airflow/dags/iot-ex2-coldpath.py: source="airflow.models.dagbag.DagBag"
[2025-05-02, 19:30:04] INFO - Mean A: 5.20, Mean B: 40.00, Diff: 34.80: source="root"
[2025-05-02, 19:30:04] INFO - P95 A: 5.20, P95 B: 40.00, Diff: 34.80: source="root"
[2025-05-02, 19:30:04] INFO - Difference in means (34.80) is within expected variation (34.80): source="root"
[2025-05-02, 19:30:04] INFO - Done. Returned value was: None: source="airflow.task.operators.airflow.providers.standard.decorators.python._PythonDecoratedOperato
[2025-05-02, 19:30:04] INFO - Task instance is in running state: chan="stdout": source="task"
[2025-05-02, 19:30:04] INFO - Previous state of the Task instance: queued: chan="stdout": source="task"
[2025-05-02, 19:30:04] INFO - Current task name:compare_and_detect: chan="stdout": source="task"
[2025-05-02, 19:30:04] INFO - Dag name:temperature_cold_path: chan="stdout": source="task"
[2025-05-02, 19:30:04] INFO - Task instance in success state: chan="stdout": source="task"
[2025-05-02, 19:30:04] INFO - Previous state of the Task instance: running: chan="stdout": source="task"
[2025-05-02, 19:30:04] INFO - Task operator:<Task(_PythonDecoratedOperator): compare_and_detect>: chan="stdout": source="task"