On-Prem Medallion Data Platform – Local Docker Guide

Stack: MinIO (S3), Kafka + Zookeeper, Spark 3.5, Iceberg + Nessie 0.104, Trino 447, ClickHouse 24.3 (optional), Airflow 2.9.3, Superset 3.0, Postgres 13.

Goal: Bronze → Silver → Gold lakehouse on a single laptop/VM.

---

0 Prerequisites

Software	Version tested
Docker Engine	≥ 24
Docker Compose	plugin (comes with Engine)
8 GB RAM free	min
30 GB disk	min

```
# verify
docker --version  &&  docker compose version
```

---

1 Project layout

```
medallion-data-platform/
├─ airflow/
│  └─ dags/                      # drop DAG python files here
├─ configs/
│  ├─ trino/
│  │   └─ catalog/iceberg.properties
│  └─ spark/spark-defaults.conf   # optional tuning
├─ docker/
│  └─ airflow/Dockerfile          # custom Airflow image
├─ docker-compose.yml             # full stack
└─ README.md
```

Create the folders:

```
mkdir -p airflow/dags configs/trino/catalog docker/airflow
```

---

2 Custom Airflow image (providers pre-baked)

`docker/airflow/Dockerfile`:

```
FROM apache/airflow:2.9.3-python3.11
ARG AIRFLOW_VERSION=2.9.3
ARG PYTHON_VERSION=3.11
RUN set -ex \
 && curl -Lf "https://raw.githubusercontent.com/apache/airflow/constraints-$
{AIRFLOW_VERSION}/constraints-${PYTHON_VERSION}.txt" -o /tmp/constraints.txt \
 && pip install --no-cache-dir --constraint /tmp/constraints.txt \
      apache-airflow-providers-apache-kafka \
      apache-airflow-providers-apache-spark
```

Build once:

```
docker build -t custom-airflow:2.9.3 docker/airflow
```

---

3 Trino Iceberg catalog

`configs/trino/catalog/iceberg.properties`:

```
connector.name=iceberg

# Nessie
iceberg.catalog.type=nessie
iceberg.nessie-catalog.uri=http://nessie:19120/api/v1
iceberg.nessie-catalog.default-warehouse-dir=s3a://silver-curated

# MinIO (S3)
hive.s3.endpoint=http://minio:9000
hive.s3.aws-access-key=minio
hive.s3.aws-secret-key=minio123
hive.s3.path-style-access=true
hive.s3.ssl.enabled=false
```

> **Tip:** keep lines **exact**; wrong keys break startup.

4 docker-compose.yml (core services)

```
version: "3.9"   # optional legacy warning

networks:
  dataplane:

volumes:
  minio_data:
  clickhouse_data:
  postgres_data:
  nessie_data:
  superset_data:
  airflow_logs:

services:

  # Zookeeper + Kafka
  zookeeper:
    image: confluentinc/cp-zookeeper:7.6.0
    environment:
      ZOOKEEPER_CLIENT_PORT: 2181
    networks: [dataplane]
    ports: ["2181:2181"]

  kafka:
    image: confluentinc/cp-kafka:7.6.0
    depends_on: [zookeeper]
    networks: [dataplane]
    ports:
      - "9092:9092"  # internal
      - "9093:9093"  # host
    environment:
      KAFKA_BROKER_ID: 1
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP:
PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka:9092,PLAINTEXT_HOST://
localhost:9093
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1

  # MinIO (object store)
  minio:
    image: minio/minio:latest
    command: server /data --console-address ":9001"
```

```
    environment:
      MINIO_ROOT_USER: minio
      MINIO_ROOT_PASSWORD: minio123
    networks: [dataplane]
    ports:
      - "9000:9000"
      - "9001:9001"
    volumes:
      - minio_data:/data

  # Nessie catalog backend
  nessie:
    image: ghcr.io/projectnessie/nessie:0.104.2-java
    networks: [dataplane]
    ports: ["19120:19120"]
    volumes:
      - nessie_data:/opt/nessie/store

  # Spark master + worker
  spark-master:
    image: bitnami/spark:3.5.0
    environment:
      SPARK_MODE: master
    networks: [dataplane]
    ports:
      - "7077:7077"
      - "8082:8080"

  spark-worker:
    image: bitnami/spark:3.5.0
    depends_on: [spark-master]
    environment:
      SPARK_MODE: worker
      SPARK_MASTER_URL: spark://spark-master:7077
    networks: [dataplane]

  # Trino
  trino:
    image: trinodb/trino:447
    networks: [dataplane]
    ports: ["8081:8080"]
    depends_on: [minio, nessie]
    volumes:
      - ./configs/trino/catalog:/etc/trino/catalog:ro

  # Postgres for Airflow
  postgres:
    image: postgres:13
```

```
    environment:
      POSTGRES_USER: airflow
      POSTGRES_PASSWORD: airflow
      POSTGRES_DB: airflow
    networks: [dataplane]
    volumes:
      - postgres_data:/var/lib/postgresql/data

  # Airflow (init job)
  airflow-init:
    image: custom-airflow:2.9.3
    depends_on: [postgres]
    networks: [dataplane]
    environment:
      AIRFLOW__DATABASE__SQL_ALCHEMY_CONN: postgresql+psycopg2://
airflow:airflow@postgres/airflow
      AIRFLOW__CORE__LOAD_EXAMPLES: "False"
    volumes:
      - ./airflow/dags:/opt/airflow/dags
      - airflow_logs:/opt/airflow/logs
    entrypoint: >
      bash -e -c "airflow db upgrade && airflow users create --username admin --
password admin --firstname Admin --lastname User --role Admin --email
admin@example.com"
    restart: "no"

  airflow-webserver:
    image: custom-airflow:2.9.3
    depends_on: [airflow-init]
    networks: [dataplane]
    ports: ["8080:8080"]
    environment:
      AIRFLOW__DATABASE__SQL_ALCHEMY_CONN: postgresql+psycopg2://
airflow:airflow@postgres/airflow
      AIRFLOW__CORE__EXECUTOR: LocalExecutor
      AIRFLOW__CORE__LOAD_EXAMPLES: "False"
    volumes:
      - ./airflow/dags:/opt/airflow/dags
      - airflow_logs:/opt/airflow/logs
    command: webserver

  airflow-scheduler:
    image: custom-airflow:2.9.3
    depends_on: [airflow-init]
    networks: [dataplane]
    environment:
      AIRFLOW__DATABASE__SQL_ALCHEMY_CONN: postgresql+psycopg2://
airflow:airflow@postgres/airflow
```

```
      AIRFLOW__CORE__EXECUTOR: LocalExecutor
    volumes:
      - ./airflow/dags:/opt/airflow/dags
      - airflow_logs:/opt/airflow/logs
    command: scheduler

  # Superset BI
  superset:
    image: apache/superset:3.0.2
    depends_on: [trino]
    networks: [dataplane]
    ports: ["8088:8088"]
    environment:
      SUPERSET_SECRET_KEY: "TH1sIsASecret_ChangeMe"
      DATABASE_URL: sqlite:////var/lib/superset/superset.db
    volumes:
      - superset_data:/var/lib/superset
    command: >
      bash -e -c "superset db upgrade && superset fab create-admin --username
admin --firstname Uzair --lastname User --email admin@example.com --password
admin || true && superset init && superset run -p 8088 -h 0.0.0.0"
```

---

5 Bootstrap sequence

```
# 1. Build Airflow image (once)
docker build -t custom-airflow:2.9.3 docker/airflow

# 2. Launch full stack
docker compose up -d      # first run ~5 min

# 3. Create three buckets in MinIO
open http://localhost:9001  (minio/minio123)
bronze-raw   |   silver-curated   |   gold-marts
```

---

6 Smoke tests

6.1 Spark → Iceberg → Nessie

```
echo -e "id,name\n1,alice\n2,bob" > sample.csv
mc alias set minio http://localhost:9000 minio minio123
mc cp sample.csv minio/bronze-raw/customer/
```

Spark shell:

```
docker exec -it medallion-data-platform-on-minio-spark-master-1 spark-shell \
  --packages org.apache.iceberg:iceberg-spark-runtime-3.5_2.12:1.5.2 \
  --conf spark.sql.catalog.nessie=org.apache.iceberg.spark.SparkCatalog \
  --conf spark.sql.catalog.nessie.warehouse=s3a://silver-curated \
  --conf spark.sql.catalog.nessie.uri=http://nessie:19120/api/v1

val df = spark.read.option("header","true").csv("s3a://bronze-raw/customer")
df.writeTo("nessie.demo.customer").using("iceberg").create()
:quit
```

6.2 Trino query

```
docker exec -it $(docker compose ps -q trino) trino -e "SELECT * FROM
demo.customer;"
```

---

7 Superset connection

1. Browse http://localhost:8088 (admin / admin)
2. **Data ➜ Database Connections ➜ +** *SQLAlchemy URI*: `trino://@trino:8080`
3. Create dataset on `demo.customer`, build a chart.

---

8 Airflow DAG skeleton

`airflow/dags/bronze_to_silver_customer.py`:

```
from airflow import DAG
from airflow.providers.apache.spark.operators.spark_submit import
SparkSubmitOperator
from datetime import datetime

default_args = {"owner": "uzair", "retries": 0}

dag = DAG(
    "bronze_to_silver_customer",
    start_date=datetime(2025, 7, 1),
    schedule_interval="@daily",
    catchup=False,
    default_args=default_args,
)
```

```
SparkSubmitOperator(
    task_id="spark_csv_to_iceberg",
    dag=dag,
    application="/opt/airflow/dags/jobs/csv_to_iceberg.py",
    conn_id="spark_default",
    packages="org.apache.iceberg:iceberg-spark-runtime-3.5_2.12:1.5.2",
    application_args=[
        "s3a://bronze-raw/customer/sample.csv",
        "nessie.demo.customer",
        "s3a://silver-curated",
        "http://nessie:19120/api/v1",
    ],
    conf={
        "spark.sql.catalog.nessie": "org.apache.iceberg.spark.SparkCatalog",
        "spark.sql.catalog.nessie.uri": "http://nessie:19120/api/v1",
        "spark.sql.catalog.nessie.warehouse": "s3a://silver-curated",
        "spark.hadoop.fs.s3a.endpoint": "http://minio:9000",
        "spark.hadoop.fs.s3a.path.style.access": "true",
        "spark.hadoop.fs.s3a.access.key": "minio",
        "spark.hadoop.fs.s3a.secret.key": "minio123",
    },
)
```

Turn it on in Airflow UI and watch tasks succeed.

---

9 Troubleshooting crib sheet

Error message	Fix
`iceberg.properties does not contain connector.name`	File missing or wrong mount → ensure first line `connector.name=iceberg`.
`...Invalid configuration property hive.metastore.uri`	Leftover Hive keys → use the exact Nessie keys shown above.
Trino exits after provider install	Inline comments in properties; remove trailing comments.
Airflow `/entrypoint: exec: airflow: not found`	Avoid runtime pip install; use custom image as shown.
Superset `--username` flags not found	Keep entire admin-create command on **one line** inside `bash -c "..."`.

---

10 Next enhancements

- Wire Kafka streaming job into Bronze bucket.
- Add ClickHouse if you need ultra-fast aggregates.
- Enable MinIO versioning for time-travel.
- Swap LocalExecutor → CeleryExecutor for multi-node Airflow.
- Push stack to k8s (Helm) when ready for cluster.

Happy data-engineering!\ — Uzair’s local Medallion stack