

OLD FILE STRUCTURE

fraud-detection

```
|  
|   └── infrastructure  
|       |   └── docker  
|       |   └── terraform  
|       |   └── k8s  
|       └── configs  
|           |   └── airflow  
|           |   └── spark  
|           |   └── superset  
|           └── logging  
|  
|   └── data  
|       |   └── raw  
|       |   └── processed  
|       └── sample  
|  
|   └── ingestion  
|       |   └── python-producer  
|       └── connectors  
|  
|   └── streaming  
|       |   └── jobs  
|       └── spark-submit  
|  
|   └── ml  
|       |   └── training  
|       |   └── model  
|       └── serving
```

```
|  
|   └── airflow  
|       |   └── dags  
|       └── scripts  
|  
|   └── superset  
|       └── dashboards  
|  
|   └── monitoring  
|       |   └── grafana  
|       └── prometheus  
|  
|   └── tests  
|       |   └── unit  
|       |   └── integration  
|       └── streaming-tests  
|  
└── scripts  
  
"D:\data_engineering\fraud_detection\old\ingestion\python-producer\producer.py"  
  
import time  
  
import json  
  
import pandas as pd  
  
from confluent_kafka import Producer  
  
import yaml  
  
import os  
  
  
# Load config  
  
config_path = os.path.join(os.path.dirname(__file__), "config.yaml")  
  
with open(config_path, "r") as f:  
  
    config = yaml.safe_load(f)  
  
print("Loaded config:", config)
```

```
CSV_PATH = config["data"]["csv_path"]
KAFKA_BOOTSTRAP = config["kafka"]["bootstrap_servers"]
TOPIC = config["kafka"]["topic"]
RATE = config["producer"]["rows_per_second"]

# Producer config
producer_conf = {
    "bootstrap.servers": KAFKA_BOOTSTRAP,
    "client.id": "creditcard-producer"
}

producer = Producer(producer_conf)

def delivery_report(err, msg):
    if err is not None:
        print(f" Delivery failed: {err}")
    else:
        print(f" Delivered to {msg.topic()} [{msg.partition()}]")

def stream_csv():
    df = pd.read_csv(CSV_PATH)

    print(f" Streaming {len(df)} rows to Kafka topic -> {TOPIC}")

    for _, row in df.iterrows():
        producer.produce(
            topic=TOPIC,
            value=json.dumps(row.to_dict()),
            callback=delivery_report
        )
```

```
producer.poll(0)
time.sleep(1 / RATE)

producer.flush()
print("Stream complete!")

if __name__ == "__main__":
    stream_csv()

"D:\data_engineering\fraud_detection\old\ingestion\python-producer\config.yaml"
kafka:
    bootstrap_servers: "localhost:29092"
    topic: "creditcard-transactions"
data:
    csv_path: "../../data/raw/creditcard.csv"
producer:
    rows_per_second: 50

"D:\data_engineering\fraud_detection\old\ingestion\python-producer\requirements.txt"
pandas
confluent-kafka
pyyaml
```

```
"D:\data_engineering\fraud_detection\old\streaming\jobs\clean_stream.py"

from pyspark.sql import SparkSession

from pyspark.sql.functions import from_json, col, current_timestamp

from pyspark.sql.types import StructType, StructField, DoubleType, IntegerType

# ----

# 1. SCHEMA

# ----

schema = StructType([
    StructField("Time", DoubleType(), True),
] + [
    StructField(f"V{i}", DoubleType(), True) for i in range(1, 29)
] + [
    StructField("Amount", DoubleType(), True),
    StructField("Class", IntegerType(), True)
])

# ----

# 2. SPARK SESSION

# ----

spark = SparkSession.builder \
    .appName("CreditCardFraudCleaner") \
    .config(
        "spark.jars.packages",
        "org.apache.spark:spark-sql-kafka-0-10_2.12:3.4.1,"
        "net.snowflake:snowflake-jdbc:3.13.30,"
        "net.snowflake:spark-snowflake_2.12:2.12.0-spark_3.4"
    ) \
    .getOrCreate()
```

```
spark.sparkContext.setLogLevel("WARN")

# -----
# 3. READ STREAM FROM KAFKA
# -----

df_raw = spark.readStream \
    .format("kafka") \
    .option("kafka.bootstrap.servers", "localhost:29092") \
    .option("subscribe", "creditcard-transactions") \
    .option("startingOffsets", "latest") \
    .load()

df_json = df_raw.selectExpr("CAST(value AS STRING) AS json_str")

df_parsed = df_json.select(
    from_json(col("json_str"), schema).alias("data")
).select("data.*")

df_cleaned = df_parsed.fillna(0).withColumn("ingest_ts", current_timestamp())

# -----
# 4. SNOWFLAKE CONFIG (CORRECT)
# -----

sfOptions = {
    "sfURL": "https://rpjxzuz-wg57165.snowflakecomputing.com",
    "sfUser": "RAMAKRISHNAM",
    "sfPassword": "Ramakrishna118143",
    "sfDatabase": "FRAUD_DETECTION_DB",
    "sfSchema": "RAW",
    "sfWarehouse": "COMPUTE_WH",
    "sfRole": "ACCOUNTADMIN"
}
```

```
}

# -----
# 5. FOREACH BATCH → WRITE TO SNOWFLAKE

# -----
def write_to_snowflake(batch_df, batch_id):

    print(f"📌 Writing batch {batch_id} to Snowflake...")

    batch_df.write \
        .format("snowflake") \
        .options(**sfOptions) \
        .option("dbtable", "CREDIT_CLEANED") \
        .mode("append") \
        .save()

# -----
# 6. START STREAMING

# -----
query = df_cleaned.writeStream \
    .foreachBatch(write_to_snowflake) \
    .outputMode("append") \
    .start()

query.awaitTermination()

"D:\data_engineering\fraud_detection\old\spark_conf\java.security"
policy.allowSystemPropertyOverrides=true

"D:\data_engineering\fraud_detection\jars\kafka-clients-3.4.1.jar"
"D:\data_engineering\fraud_detection\jars\spark-sql-kafka-0-10_2.12-3.4.1.jar"
```

```
"D:\data_engineering\fraud_detection\data\raw\creditcard.csv"
```

```
"D:\data_engineering\fraud_detection\infrastructure\docker\docker-compose.dev.yml"  
version: "3.9"
```

```
services:
```

```
zookeeper:
```

```
  image: confluentinc/cp-zookeeper:7.4.1
```

```
  environment:
```

```
    ZOOKEEPER_CLIENT_PORT: 2181
```

```
  ports:
```

```
    - "2181:2181"
```

```
kafka:
```

```
  image: confluentinc/cp-kafka:7.4.1
```

```
  depends_on:
```

```
    - zookeeper
```

```
  environment:
```

```
    KAFKA_BROKER_ID: 1
```

```
    KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
```

```
    KAFKA_LISTENERS: PLAINTEXT://0.0.0.0:9092
```

```
    KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka:9092
```

```
    KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
```

```
  ports:
```

```
    - "9092:9092"
```

```
kafka-ui:
```

```
  image: provectuslabs/kafka-ui:latest
```

```
  ports:
```

```
    - "8082:8080"
```

environment:

```
KAFKA_CLUSTERS_0_NAME: local  
KAFKA_CLUSTERS_0_BOOTSTRAPSERVERS: kafka:9092
```

depends_on:

- kafka

postgres:

image: postgres:15

environment:

```
POSTGRES_USER: airflow  
POSTGRES_PASSWORD: airflow  
POSTGRES_DB: airflow
```

ports:

- "5432:5432"

airflow:

image: apache/airflow:2.7.1

depends_on:

- postgres

environment:

```
AIRFLOW__CORE__EXECUTOR: LocalExecutor
```

```
AIRFLOW__DATABASE__SQL_ALCHEMY_CONN:
```

postgresql+psycopg2://airflow:airflow@postgres/airflow

```
AIRFLOW__CORE__FERNET_KEY: "dGhpc2IzYXZlcmbGx5bG9uZ2ZlcmbGtIeQ=="
```

ports:

- "8080:8080"

command: >

```
bash -c "
```

```
airflow db init &&
```

```
airflow users create --username admin --password admin --firstname Ram --lastname Krishna --  
role Admin --email admin@example.com &&
```

```
airflow webserver
```

"

superset:

build:

context: ../../superset

dockerfile: Dockerfile

container_name: superset

depends_on:

- postgres

environment:

SUPERSET_SECRET_KEY: "mysecret"

SUPERSET_DATABASE_URI: "postgresql+psycopg2://airflow:airflow@postgres:5432/airflow"

ADMIN_USERNAME: admin

ADMIN_PASSWORD: admin

ADMIN_FIRSTNAME: Ram

ADMIN_LASTNAME: Krishna

ADMIN_EMAIL: admin@example.com

ports:

- "8088:8088"

volumes:

- ../../superset/superset_config.py:/app/pythonpath/superset_config.py

command: >

bash -c "

superset db upgrade &&

superset fab create-admin --username admin --password admin --firstname Ram --lastname Krishna --email admin@example.com &&

superset init &&

```
gunicorn --bind 0.0.0.0:8088 --workers 4 superset.app:create_app()
```

```
"
```

NEW FILE STRUCTURE

```
fraud-detection
|
└── 1_data_ingestion/
    ├── producers/          # your CSV → Kafka code (move python-producer here)
    ├── loaders/            # NEW: Kafka → Snowflake loader
    ├── schemas/            # Kafka, Snowflake schemas
    └── utils/              # shared ingestion utils
|
└── 2_stream_processing/
    ├── spark_jobs/         # existing Spark Structured Streaming jobs
    ├── realtime_ml/        # NEW: ML model for real-time predictions
    ├── consumers/           # Kafka consumers for real-time dashboards
    └── alerts/              # Email/SMS/Slack alert logic
|
└── 3_batch_processing/
    ├── snowflake_queries/   # NEW: SQL transformations
    ├── ml_training/         # move your ml/training here
    ├── ml_models/           # move your ml/model here
    ├── batch_jobs/          # Airflow jobs or Python scripts
    └── dashboards/          # Streamlit or Superset dashboards
|
└── streamlit_app/         # NEW home for Streamlit
    ├── app.py               # Your Streamlit UI
    └── pages/               # multi-page streamlit
```

```
| └── .env           # Snowflake credentials (local only)
|
|   └── infrastructure/
|       ├── docker/      # SAME (your current docker compose)
|       ├── k8s/
|       ├── terraform/
|       └── configs/
|           ├── airflow/
|           ├── spark/
|           ├── logging/
|           └── superset/    # we will keep old superset but not use now
|
|   └── data/
|       ├── raw/
|       ├── processed/
|       └── sample/
|
|   └── airflow/
|       ├── dags/        # your airflow DAG stays same
|       └── scripts/
|
└── scripts/          # helper scripts
```

Got it — **you are NOT starting a new project**, you want to **re-organize the project into a real-world architecture with three logical layers, without losing any existing code.**

I will NOT make you repeat anything.

I will map **exactly what work you already built → where it should go now**, cleanly.

 Your New Real-World 3-Layer Architecture

You want:

1 Data Ingestion Layer

- read CSV → Kafka → Snowflake
- push real-time stream to Kafka
- ALSO write to Snowflake for warehousing
- ML preprocessing (light)
- send clean stream for real-time ML

2 Stream Processing Layer (Real-time ML + Insights)

- read from Kafka
- Spark Streaming or Python consumer
- run ML predictions live
- send alerts / results to Kafka or database
- real-time dashboards (Streamlit)

3 Batch Processing Layer (Snowflake → ML → Visualization)

- nightly / hourly batch jobs
- Snowflake SQL transformations
- ML training + evaluation
- dashboards for historical trends

Perfectly real-world.

Now I'll give you the **updated file structure** and **exactly where your existing code goes**.

📁 NEW FILE STRUCTURE (WITHOUT LOSING OLD WORK)

Below is the **new structure**, and I will tell where to move each of your current folders.

fraud-detection

|

 |— 1_data_ingestion/

 | |— producers/ # your CSV → Kafka code (move python-producer here)

 | |— loaders/ # NEW: Kafka → Snowflake loader

 | |— schemas/ # Kafka, Snowflake schemas

 | └— utils/ # shared ingestion utils

|

 |— 2_stream_processing/

 | |— spark_jobs/ # existing Spark Structured Streaming jobs

 | |— realtime_ml/ # NEW: ML model for real-time predictions

 | |— consumers/ # Kafka consumers for real-time dashboards

```
|   └── alerts/           # Email/SMS/Slack alert logic  
|  
|  
└── 3_batch_processing/  
    |   ├── snowflake_queries/    # NEW: SQL transformations  
    |   ├── ml_training/         # move your ml/training here  
    |   ├── ml_models/          # move your ml/model here  
    |   ├── batch_jobs/         # Airflow jobs or Python scripts  
    |   └── dashboards/         # Streamlit or Superset dashboards  
  
|  
  
└── streamlit_app/        # NEW home for Streamlit  
    |   ├── app.py             # Your Streamlit UI  
    |   ├── pages/             # multi-page streamlit  
    |   └── .env               # Snowflake credentials (local only)  
  
|  
  
└── infrastructure/  
    |   ├── docker/            # SAME (your current docker compose)  
    |   ├── k8s/  
    |   ├── terraform/  
    |   └── configs/  
        |       ├── airflow/  
        |       ├── spark/  
        |       └── logging/  
    |   └── superset/          # we will keep old superset but not use now  
  
|  
  
└── data/  
    |   ├── raw/  
    |   ├── processed/  
    |   └── sample/  
  
|  
  
└── airflow/
```

```
|   |-- dags/           # your airflow DAG stays same  
|   |-- scripts/  
|  
|   |-- scripts/       # helper scripts
```

NOW — EXACTLY WHERE YOUR EXISTING FILES GO

Your current CSV → Kafka code

ingestion/python-producer

 Move to:

1_data_ingestion/producers/

Your existing Kafka → Spark → Snowflake code

streaming/jobs

 Move to:

2_stream_processing/spark_jobs/

Your ML training code

ml/training

 Move to:

3_batch_processing/ml_training/

Your saved ML model

ml/model

 Move to:

3_batch_processing/ml_models/

Your Superset dashboard folder

Keep it untouched, but you don't need to use Superset anymore.

WHAT NEW FILES YOU MUST ADD (VERY FEW)

1. Kafka → Snowflake Loader

Create:

1_data_ingestion/loaders/kafka_to_snowflake.py

This script will write stream to Snowflake.

★ 2. Real-time ML Predictor

Create:

2_stream_processing/realtime_ml/predict_fraud.py

Uses your trained model to predict in real-time.

★ 3. Streamlit App Folder

Create:

streamlit_app/app.py

streamlit_app/.env

You already have app.py — I will fix it next.

■ WHAT YOU DO NOT TOUCH

- ✓ Your Kafka
- ✓ Your Airflow
- ✓ Your Snowflake structure
- ✓ Your streaming DAGs
- ✓ Your ML training code
- ✓ Docker compose

Everything remains — just placed in correct folders.

■ NOW : WHAT YOU NEED FROM ME NEXT

Tell me which one you want to do next:

Option A → Fix Streamlit .env + working Snowflake connection

I will give corrected app.py + working connection instantly.

Option B → Build Streamlit real-time dashboard (Kafka consumer)

Live updates from Kafka.

Option C → Build Kafka → Snowflake loader script

Fully production ready.

Option D → Build Spark real-time ML prediction job

Using your ML model.

Option E → Setup Airflow DAG for batch ML training

Daily training + storing results.

❓ Choose A / B / C / D / E

(You can pick multiple, we'll go step-by-step — no repeating work)

You said:

```
fraud-detection | └── 1_data_ingestion/ | └── producers/ # your CSV → Kafka code (move  
python-producer here) | └── loaders/ # NEW: Kafka → Snowflake loader | └── schemas/ # Kafka,  
Snowflake schemas | └── utils/ # shared ingestion utils | └── 2_stream_processing/ | └──  
spark_jobs/ # existing Spark Structured Streaming jobs | └── realtime_ml/ # NEW: ML model for  
real-time predictions | └── consumers/ # Kafka consumers for real-time dashboards | └── alerts/ #  
Email/SMS/Slack alert logic | └── 3_batch_processing/ | └── snowflake_queries/ # NEW: SQL  
transformations | └── ml_training/ # move your ml/training here | └── ml_models/ # move your  
ml/model here | └── batch_jobs/ # Airflow jobs or Python scripts | └── dashboards/ # Streamlit or  
Superset dashboards | └── streamlit_app/ # NEW home for Streamlit | └── app.py # Your  
Streamlit UI | └── pages/ # multi-page streamlit | └── .env # Snowflake credentials (local only) | └──  
infrastructure/ | └── docker/ # SAME (your current docker compose) | └── k8s/ | └──  
terraform/ | └── configs/ | └── airflow/ | └── spark/ | └── logging/ | └── superset/ # we will  
keep old superset but not use now | └── data/ | └── raw/ | └── processed/ | └── sample/ | └──  
airflow/ | └── dags/ # your airflow DAG stays same | └── scripts/ | └── scripts/ # helper scripts  
for this file structure what are my old files should i move to new folders ?
```

ChatGPT said:

Great — now that you've chosen the **new real-world file structure**, here is the **exact list of what to move**, from your **existing project** → into the **new folders**, with ZERO code rewriting and NO duplication.

I'll tell you **exactly which old folder goes where, one by one**.

Final Mapping: OLD → NEW Folder Structure

Below is your **new architecture** (3 big layers):

1_data_ingestion/

2_stream_processing/

3_batch_processing/

streamlit_app/

infrastructure/

data/

airflow/

scripts/

Now let's map your **current folders** into this structure.

1. DATA INGESTION (Kafka + Snowflake)

→ This will store everything related to **reading CSV + writing to Kafka + writing to Snowflake**.

Move from OLD → NEW:

A. OLD: ingestion/python-producer/

 **MOVE TO → 1_data_ingestion/producers/**

This includes:

- CSV → Kafka producer
 - Any producer utility code
 - Your existing ingestion Python scripts
-

B. OLD: ingestion/connectors/

 **MOVE TO → 1_data_ingestion/utils/**

(or if some files are specific to Snowflake loaders: 1_data_ingestion/loaders/)

These are the helper functions, configs, connection utils.

C. NEW (EMPTY NOW): 1_data_ingestion/loaders/

This is where you will later add **Kafka → Snowflake loader** (Spark or Python Snowpipe).

 No need to add anything now — we will create it step-by-step later.

D. NEW: 1_data_ingestion/schemas/

You will store:

- Kafka schema (if using JSON schema / AVRO / Protobuf)
- Snowflake table definitions
- Column metadata

Right now this folder is empty — later we add schemas.

2. STREAM PROCESSING (Spark Real-Time + Real-Time ML)

Your current Spark code lives in `streaming/`.

Move from OLD → NEW:

A. OLD: `streaming/jobs/`

 **MOVE TO → 2_stream_processing/spark_jobs/**

These are:

- Spark Structured Streaming jobs
 - ETL transformations
 - Any Kafka → Spark → Kafka pipeline
-

B. OLD: `streaming/spark-submit/`

 **MOVE TO → 2_stream_processing/spark_jobs/spark-submit/**
(or keep the same name inside `spark_jobs`)

This folder contains:

- Spark-submit scripts
 - Deployment scripts
-

C. NEW: `2_stream_processing/realtimeml/`

Nothing to move yet.

Later you will put:

- Real-time ML model
 - Model inference code (Spark UDF, Python consumer, etc.)
-

D. NEW: `2_stream_processing/consumers/`

You will store:

- Kafka consumer that reads real-time data for dashboard
- Websocket updates
- Alerts consumer

(No files now — we will build it.)

3. BATCH PROCESSING (Snowflake + ML Training + Dashboards)

Your old ML and superset work will go here.

Move from OLD → NEW:

A. OLD: ml/training/

📍 **MOVE TO → 3_batch_processing/ml_training/**

B. OLD: ml/model/

📍 **MOVE TO → 3_batch_processing/ml_models/**

These include:

- training notebook
 - preprocessing
 - saved model.pkl
-

C. OLD Superset Dashboards

You already have:

superset/dashboards/

📍 **MOVE TO →**

3_batch_processing/dashboards/superset/

Even though you won't use Superset now, you're **preserving your work**.

D. NEW: 3_batch_processing/snowflake_queries/

Later will store:

- SQL transformations
 - DBT-style logic
 - Materialized views
 - Batch aggregations
-

E. NEW: 3_batch_processing/batch_jobs/

You will put:

- Airflow batch scripts
- Batch ML inference
- Snowflake → ML prediction

We can migrate later.

4. STREAMLIT

This is your new UI replacing Superset.

OLD: nothing to move

→ this is newly created:

streamlit_app/

 app.py

 pages/

 .env

This stays separate — perfect.

5. INFRASTRUCTURE (NO CHANGES)

Your entire infrastructure/ folder stays as is.

- ✓ Docker
- ✓ Terraform
- ✓ K8s
- ✓ configs

You DON'T move anything here.

6. DATA (NO CHANGES)

Your data/raw, data/processed, etc. remain untouched.

7. AIRFLOW (NO CHANGES)

airflow/dags and airflow/scripts stay where they are.

Summary Table — What to Move

OLD FOLDER

NEW LOCATION

ingestion/python-producer 1_data_ingestion/producers/

ingestion/connectors 1_data_ingestion/utils/

streaming/jobs 2_stream_processing/spark_jobs/

OLD FOLDER	NEW LOCATION
streaming/spark-submit	2_stream_processing/spark_jobs/spark-submit/
ml/training	3_batch_processing/ml_training/
ml/model	3_batch_processing/ml_models/
superset/dashboards	3_batch_processing/dashboards/superset/

Everything else stays where it is.

What we will do next

After you move these files, we will do:

✓ Step 1 — Fix Streamlit .env

(SNOWFLAKE_ACCOUNT error)

✓ Step 2 — Streamlit reading real snowflake tables

✓ Step 3 — Dashboard pages

✓ Step 4 — Add real-time predictions

✓ Step 5 — Clean project structure

✓ Step 6 — Setup batch jobs (Snowflake + Airflow)

Tell me when you've moved the folders

Once you confirm:

"I moved the folders"

I will:

- Adjust your imports
- Fix paths
- Fix your app.py
- Make Streamlit connect to Snowflake
- Start the real visualization dashboard