

# Social-Media Analytics on the Medallion Platform

A high-level delivery plan that turns your working infra (Kafka → Spark → Iceberg → Trino → Superset) into a production-grade pipeline for X / Instagram / TikTok data.

## 0 Foundations (already done )

✓ Component	Notes
✓ <b>Kafka</b> (internal : 9092, host : 9093)	Single broker with PLAINTEXT + HOST listener
✓ <b>MinIO</b> buckets	bronze-raw/, silver-curated/, gold-marts/
✓ <b>Spark 3.5</b> with Nessie Iceberg catalog	nessie.uri=http://nessie:19120/api/v1
✓ <b>Trino 447</b> → Iceberg	Catalog iceberg
✓ <b>Superset 3.0</b> with sqlalchemy-trino 0.5.0	Dashboard UI

No more infra stories—everything below is data & application.

## 1 Product Vision

*"Marketing wants minute-level engagement KPIs and hashtag sentiment trend lines for all active campaigns."*

- **Freshness SLA:** < 2 min end-to-dashboard.
- **History:** Keep raw forever; Silver + Gold for 400 days.
- **Scalability target:** 10 M social events / hr ( $\approx$  2 K msg/s).

## gh repo clone uzair14137/Medallion-Data-Platform-on-MinIO2 Roadmap & Sprints (2-week cadence)

| Sprint | Primary Goal | Key Deliverables | Acceptance / DoD | |-----|-----|-----|-----| |  
**0 (2 d)** | Project bootstrap | • Repo scaffolding (src/, dags/, charts/) • Git branches: dev, main | CI pipeline green | | **1** | Ingestion MVP | • Python FastAPI collector for X/Twitter → Kafka social\_raw • Topic schema in Confluent Schema Reg. • Airflow DAG ingest\_raw hourly cron | Offsets < 10 s behind | | **2** | Bronze → Silver streaming | • Spark StructuredStreaming job bronze\_to\_silver (JSON → Iceberg) • nessie.social.posts table partitioned by date\_hour | Data appears in Silver within SLA | | **3** |

Enrichment & ClickHouse metrics | • User lookup enrichment (followers, verified)• ClickHouse MV  
likes\_last\_5m | SELECT returns correct counts | | 4 | Gold marts + Superset dashboards | • Iceberg  
view gold.social.campaign\_perf • Superset dashboard “Real-Time Engagement” with 6 charts |  
Stakeholder sign-off | | 5 | Hardening & observability | • Prom/ Grafana alerts: lag, DAG failures• Retry/  
dead-letter topic | All dashboards stay green for 72 h |

## 3 Detailed Backlog (next two sprints)

### Sprint 1 – Ingestion MVP

1. **Story S1-1** Twitter API connector (search/recent by hashtag)
2. Use tweepy v4, polling every 15 s.
3. **S1-2** Dockerise collector
4. Build collector-social:0.1 image; env vars for API keys, Kafka brokers.
5. **S1-3** Kafka topic setup script
6. kafka-topics --create ... --configs cleanup.policy=compact,retention.ms=604800000.
7. **S1-4** Schema Registry setup (optional)
8. **S1-5** Airflow DAG ingest\_raw.py
9. KubernetesPodOperator or DockerOperator.

### Sprint 2 – Bronze → Silver

1. **S2-1** Spark job skeleton
2. Read from kafka, write to nessie.social.posts\_raw. Demo run on cluster mode.
3. **S2-2** Parse + basic transformations
4. created\_at to UTC, extract hashtags array, language.
5. **S2-3** Unit tests with spark-testing-base.
6. **S2-4** Airflow DAG bronze\_to\_silver.py (Streaming Add Jar operator).

## 4 Data Model (Silver → Gold)

Layer	Table / View	Partitions	Notes
Silver	social.posts	ds_hour	Clean, enriched posts
Silver	social.hashtags	ds_hour, hashtag	Exploded array
Gold	social.campaign_perf	daily	campaign_id, metrics



```
--topic social_raw \
--partitions 3 \
--replication-factor 1 \
--config cleanup.policy=delete \
--config retention.ms=604800000 # 7 days for replay
```

## 2 Producer skeleton (Python + FastAPI)

```
# src/collector/twitter_producer.py
import os, json, asyncio, tweepy, aiokafka

BOOTSTRAP = os.getenv("KAFKA_BROKERS", "kafka:9092")
TOPIC      = os.getenv("TOPIC", "social_raw")

async def main():
    producer = aiokafka.AIOKafkaProducer(bootstrap_servers=BOOTSTRAP)
    await producer.start()
    try:
        for tweet in tweepy.Cursor(api.search_tweets, q="#marketing",
tweet_mode="extended").items():
            await producer.send_and_wait(TOPIC,
json.dumps(tweet._json).encode())
    finally:
        await producer.stop()

if __name__ == "__main__":
    asyncio.run(main())
```

Build as image `collector-social:0.1` and run via Airflow `DockerOperator`.

## 3 Spark Structured-Streaming job (Bronze writer)

```
# src/jobs/bronze_to_minio.py
from pyspark.sql import SparkSession, functions as F

spark = (SparkSession.builder
        .appName("bronze-social")
        .getOrCreate())

raw_df = (spark.readStream
        .format("kafka")
        .option("kafka.bootstrap.servers", "kafka:9092")
        .option("subscribe", "social_raw")
        .option("startingOffsets", "earliest")
        .load())
```

```

json_df = (raw_df
    .select(F.col("value").cast("string").alias("json"))
    .withColumn("ingest_ts", F.current_timestamp()))

checkpoint = "s3a://bronze-raw/_checkpoints/social_raw/"

(json_df
    .writeStream
    .format("json") # or parquet
    .option("path", "s3a://bronze-raw/social/")
    .option("checkpointLocation", checkpoint)
    .trigger(processingTime="30 seconds")
    .start()
    .awaitTermination())

```

Package with `zipapp` or `--py-files`, submit via

```

spark-submit --packages org.apache.iceberg:iceberg-spark-runtime-3.5_2.12:1.5.2
\
    bronze_to_minio.py

```

## 4 Airflow DAG snippet

```

from airflow import DAG
from airflow.providers.apache.spark.operators.spark_submit import
SparkSubmitOperator
from airflow.utils.dates import days_ago

dag = DAG(
    "bronze_to_minio", start_date=days_ago(1), schedule_interval="@once",
    catchup=False)

write_bronze = SparkSubmitOperator(
    task_id="write_bronze", dag=dag,
    application="/opt/airflow/jobs/bronze_to_minio.py",
    name="bronze-social",
    conf={
        "spark.hadoop.fs.s3a.endpoint": "http://minio:9000",
        "spark.hadoop.fs.s3a.access.key": "minio",
        "spark.hadoop.fs.s3a.secret.key": "minio123",
        "spark.hadoop.fs.s3a.path.style.access": "true",
    },
    packages="org.apache.iceberg:iceberg-spark-runtime-3.5_2.12:1.5.2")

```

## 5 Validation

Check	Command
Kafka lag	<code>kafka-consumer-groups --bootstrap-server kafka:9092 --describe --group &lt;spark-group&gt;</code>
Files in bucket	<code>`mc ls minio/bronze-raw/social/`</code> <code>head`</code>
Query raw in Spark	<code>spark.read.json("s3a://bronze-raw/social/").count()</code>

(this week)

1. Approve sprint scope
2. Generate Twitter dev keys (marketing)
3. Merge Compose changes (HOST listener, spark-defaults) to `main`
4. Start **Sprint 1-Day 1** stand-up Monday 09:00