🚀 PulseStream AI – Week 1 Action Plan

🎯 Goal of the Week

Set up the environment, build the ingestion + streaming backbone, and connect it to the warehouse with orchestration in place.

📅 Day 1 – Project Setup & Environment

Objective: Prepare your workspace and Docker-based development environment.

Tasks

- Create a main folder: `PulseStream_AI/`
- Inside it, initialize subfolders:

```
ingestion/
processing/
airflow/
dbt/
api/
docker/
docs/
```

- Set up Python virtual environment (`venv` or `conda`).
- Install essentials:

```
pip install kafka-python pandas fastapi uvicorn
```

- Install **Docker Desktop**.
- Pull Kafka & Zookeeper images:

```
docker pull wurstmeister/kafka
docker pull wurstmeister/zookeeper
```

- Verify Docker + Kafka run locally.

Outcome: Local Kafka cluster running and Python env ready. **Checkpoint:** Run `docker ps` — Kafka and Zookeeper containers should be up.

📅 Day 2 – Data Ingestion with Kafka

Objective: Fetch live news data and publish it to Kafka.

Tasks

- Register at NewsAPI.org and get an API key.
- Write a simple Python producer in `ingestion/producer.py` :
  - Fetch headlines every 10 seconds.
  - Publish each article JSON into topic `raw_news_feed` .
- Test by reading messages with a simple consumer:

```
kafka-console-consumer --topic raw_news_feed --bootstrap-server localhost:9092
```

Outcome: News data continuously streaming into Kafka. **Checkpoint:** Console shows JSON articles arriving live.

⚙️ Day 3 – Stream Processing & Enrichment

Objective: Consume Kafka messages, clean and enrich them.

Tasks

- In `processing/consumer.py` :
  - Consume from `raw_news_feed` .
  - Remove duplicates.
  - Add sentiment score using a simple model (e.g., `textblob`).
  - Publish to `cleaned_news_feed` .

```
pip install textblob
```

- Test enriched messages via console consumer.

Outcome: Cleaned + sentiment-enriched messages flow to `cleaned_news_feed`. **Checkpoint:** JSON now includes `sentiment` key.

❄ Day 4 – Snowflake Integration (Local Test)

Objective: Store processed data into a Snowflake table.

Tasks

- Get Snowflake trial account or connect to an existing workspace.
- Install connector:

```
pip install snowflake-connector-python
```

- Create a table:

```
CREATE TABLE news_feed (title STRING, sentiment FLOAT, source STRING, published_at TIMESTAMP);
```

- Write `processing/load_to_snowflake.py`:
  - Read from `cleaned_news_feed`.
  - Insert records into Snowflake.

Outcome: Data from stream stored in Snowflake table. **Checkpoint:** Run SQL query `SELECT COUNT(*) FROM news_feed;` — see rows increasing.

□ Day 5 – DBT & Transformations

Objective: Prepare basic transformations and analytics table.

Tasks

- Install DBT and initialize project:

```
pip install dbt-snowflake
dbt init pulsestream_dbt
```

- Configure connection in `profiles.yml`.
- Create models:
  - `stg_news_raw.sql` → cleans data.
  - `news_sentiment_summary.sql` → aggregates average sentiment by source/category.
- Run and test:

```
dbt run
dbt test
```

Outcome: Clean & summarized tables ready in Snowflake. **Checkpoint:** Query `SELECT * FROM news_sentiment_summary;`

□ Day 6 – Airflow Orchestration

Objective: Automate ingestion → transformation.

Tasks

- Install Airflow (via Docker Compose easiest):

```
curl -LfO 'https://airflow.apache.org/docs/apache-airflow/stable/docker-compose.yaml'
docker-compose up airflow-init
docker-compose up
```

- Create a simple DAG `airflow/dags/news_ingestion_dag.py`:
  - Task 1: Call producer script.
  - Task 2: Run DBT models.
  - Task 3: Log summary.

Outcome: Airflow web UI up at `localhost:8080` showing successful DAG runs. **Checkpoint:** All tasks turn green.

⚡ Day 7 – FastAPI Mock & Mini Demo

Objective: Build and expose a simple API endpoint for recommendations (static for now).

Tasks

- Create `api/app.py`:

```
from fastapi import FastAPI
app = FastAPI()

@app.get('/recommendations')
def get_recs(user_id: str):
    return {"user_id": user_id, "recommendations": ["AI News", "Tech Trends"]}
```

- Run locally:

```
uvicorn api.app:app --reload
```

- Test endpoint in browser: `http://127.0.0.1:8000/recommendations?user_id=tan_103`

Outcome: Working API endpoint responding instantly. **Checkpoint:** JSON output visible in browser.

✅ End-of-Week Milestone

By the end of the first week, you will have:

Component	Status
Local Kafka ingestion	✅ Working
Stream processing	✅ Sentiment-enriched data
Snowflake connection	✅ Tested
DBT transformation	✅ Basic models built
Airflow orchestration	✅ DAG scheduled
FastAPI	✅ Mock recommendations API
Docker environment	✅ Configured

🚀 Next Week Preview

Week 2 will focus on:

- Building the real recommendation model
- Deploying FastAPI + Airflow to the cloud (Azure/AWS)
- Automating daily retraining and adding a feedback loop

Would you like me to turn this into a **trackable checklist format (Markdown + progress boxes)** so you can paste it in Notion or GitHub README?