Readme

GDPR Risk Pipeline

A reproducible project that fetches, processes, validates, and forecasts GDPR policy updates on an hourly basis using Apache Airflow, Python, and Prophet.

📋 Prerequisites

- **Operating System:** macOS or Linux
- **Git:** to clone the repository
- **Python 3.8+:** with venv module
- **pip:** Python package manager
- **Apache Airflow 2.7.x**
- **Prophet:** for time-series forecasting
- **Jupyter Notebook or VS Code:** for editing scripts
- **Optional:** lsof, pkill, pgrep for troubleshooting

🚧 Installation & Troubleshooting Guide

Follow these steps carefully. Pitfalls and solutions are noted.

1. 1. Clone the repository

```
git clone https://github.com/yourusername/gdpr-ccpa-risk-pipeline.git
cd gdpr-ccpa-risk-pipeline
```

- *Pitfall: If cd fails, check your clone path.*

2. 2. Create & activate venv

```
python3 -m venv venv
source venv/bin/activate
```

- *Pitfall: Ensure venv exists and you're in project root.*

3. 3. Install dependencies

```
pip install --upgrade pip
pip install apache-airflow==2.7.1 prophet requests beautifulsoup4
pandas lxml
```

- *Pitfall: On macOS, run xcode-select --install if build fails.*

4. 4. Configure Airflow

```
export AIRFLOW_HOME="$HOME/.../airflow_home"
export AIRFLOW__CORE__DAGS_FOLDER="$HOME/.../dags"
```

- *Pitfall: Mismatched AIRFLOW_HOME leads to missing DAGs.*

5. 5. Initialize Airflow DB

```
airflow db init
```

- *Pitfall: Type y if prompted.*

6. 6. Create Admin user

```
airflow users create \
--username admin \
--firstname Admin \
--lastname User \
--role Admin \
--email you@example.com \
--use-random-password
```

- *Pitfall: Include --email and --role flags.*

🛠️ Step-by-Step Setup & Execution

Scripts, DAG definition, and local execution flow.

7. 1. fetch_policy_data.py snippet

```
# scripts/fetch_policy_data.py
import os, requests
from datetime import datetime

def fetch_policy_data():
    url = "https://edpb.europa.eu/news/news_en"
    resp = requests.get(url)
    resp.raise_for_status()
    ts = datetime.utcnow().strftime("%Y%m%dT%H%M%SZ")
    fn = f"data/raw/edpb_news_{ts}.json"
    with open(fn, "w") as f:
        f.write(resp.text)
    print(f"Fetched raw data to {fn}")
```

- scripts/__init__.py — package marker
- scripts/process_policy_data.py — cleaning & aggregation
- scripts/validate_policy_data.py — sanity checks
- scripts/forecast_policy_trends.py — forecasting logic

8. 2. DAG definition snippet

```
# dags/gdpr_ccpa_risk_pipeline.py
from airflow import DAG
from airflow.operators.python import PythonOperator
```

```
from datetime import datetime

from scripts.fetch_policy_data import fetch_policy_data
from scripts.process_policy_data import process_policy_data
from scripts.validate_policy_data import validate_policy_data
from scripts.forecast_policy_trends import forecast_policy_trends

with DAG("gdpr_ccpa_risk_pipeline", schedule_interval="@hourly",
catchup=False) as dag:
    t1 = PythonOperator(task_id="fetch",
python_callable=fetch_policy_data)
    t2 = PythonOperator(task_id="process",
python_callable=process_policy_data)
    t3 = PythonOperator(task_id="validate",
python_callable=validate_policy_data)
    t4 = PythonOperator(task_id="forecast", python_callable=lambda:
forecast_policy_trends(periods=7))
    t1 >> t2 >> t3 >> t4
```

📥 Download & Inspect Input/Output

- python scripts/process_policy_data.py
- ls data/processed/cleaned_policies.csv
- head data/processed/cleaned_policies.csv
- cat data/raw/edpb_news_<timestamp>.json
- python scripts/forecast_policy_trends.py
- ls data/forecasts
- head data/forecasts/forecast_*.csv

✅ Forecast Validation Notebook

Use validate_forecast.ipynb to load raw, processed, and forecast data; plot actual vs. forecasted trends; check confidence intervals; and validate model assumptions.

🔮 Next Steps

- Integrate LLM (T5) to classify policy update severity.
- Build dashboards in Power BI/Tableau.