1. Project Title: Electricity Consumption
2. Project Objective: Create monthly predictions for residential electricity consumption in New Jersey.
3. Learning Objective:
   1. Practice building time series forecasting models
   2. Practice ML Ops – batch data collection, storage, processing, predicting, visualization, monitoring
4. Data:
   1. NJ Residential Electricity Consumption Data: <https://www.eia.gov/opendata/browser/electricity/retail-sales?frequency=monthly&data=sales;&facets=stateid;sectorid;&stateid=NJ;&sectorid=RES;&start=2025-01&end=2025-05&sortColumn=period;&sortDirection=desc>;
   2. Weather Data (historical or forecasts): <https://www.weather.gov/documentation/services-web-api#/> (use get/stations after finding out stationid in NJ?)
5. Plan (chatgpt):

Perfect — thanks for clarifying 👍. Since you’re already strong on **modeling** but new to **MLOps**, the best path is to keep the project **lean and hands-on**, without getting lost in too much infrastructure.

I’ll simplify the plan so you can build confidence step by step, while still practicing the MLOps workflow.

**🔹 Simplified Project Plan: NJ Residential Electricity Forecasting**

**0. Goal**

* Forecast **monthly electricity consumption** in NJ using past consumption + weather.
* Practice: **batch data ingestion → storage → model training → predictions → dashboard → monitoring**.

**1. Tools to Use (Simpler Stack)**

* **Python** (core)
* **Pandas, scikit-learn, statsmodels, XGBoost** → modeling
* **Requests** → API calls
* **Parquet/CSV** → storage (no database needed yet)
* **Prefect** → lightweight workflow scheduling (instead of Airflow)
* **MLflow** → experiment tracking & saving models
* **Streamlit** → simple dashboard
* **Evidently AI** → basic monitoring (drift, errors)
* **GitHub** → version control

*(Later, you can replace Parquet with a database, or Prefect with Airflow, etc.)*

**2. Project Structure (simplified)**

nj-electricity-forecast/

├─ data/ # local storage (raw & processed)

│ ├─ raw/

│ ├─ processed/

│ └─ forecasts/

├─ notebooks/ # EDA and scratch

├─ src/

│ ├─ data\_ingest.py

│ ├─ preprocess.py

│ ├─ train\_model.py

│ ├─ predict.py

│ ├─ monitor.py

│ └─ dashboard.py

├─ flows/

│ └─ monthly\_batch.py # Prefect workflow

└─ requirements.txt

**3. Step-by-Step Plan**

**Step 1 – Data Ingestion**

* Write Python scripts to:
  + Pull **electricity consumption** from your API (monthly values).
  + Pull **weather** data from API (monthly average temp, HDD/CDD, precipitation).
* Save each raw API response as a **CSV/Parquet file** under data/raw/.
* # pseudo-code
* data = requests.get(api\_url).json()
* pd.DataFrame(data).to\_csv("data/raw/electricity\_2025\_08.csv")

**Step 2 – Preprocessing**

* Clean and merge data into a single table:
  + Columns: month, consumption, avg\_temp, hdd, cdd, precip, ...
* Add **lag features** (last month’s consumption, last year’s same month).
* Save processed table under data/processed/processed.parquet.

**Step 3 – Modeling**

* Start with a **baseline**:
  + Naïve forecast = last year’s same month.
* Add a simple **SARIMAX** or **XGBoost** model using weather + lags.
* Split data (train/test), evaluate with MAE/MAPE.
* Save the **trained model** with MLflow.

**Step 4 – Prediction**

* Load the latest model from MLflow.
* Predict **next 1–12 months** electricity usage.
* Save results as a CSV under data/forecasts/.
  + Columns: month, forecast, lower\_ci, upper\_ci, model\_version.

**Step 5 – Workflow Automation (Prefect)**

* Create one Prefect flow flows/monthly\_batch.py that runs:
  1. Ingest new data from APIs.
  2. Update processed dataset.
  3. Run predictions with the latest model.
  4. Save new forecast file.
* Schedule it to run once a month.

**Step 6 – Dashboard (Streamlit)**

* Build a simple Streamlit app to show:
  + Actual vs forecast line chart.
  + Error metrics.
  + Latest forecast table.
* Example:
* st.line\_chart(df[["actual", "forecast"]])

**Step 7 – Monitoring**

* Each time new **actual consumption** is available:
  + Compare against past forecasts.
  + Compute metrics (MAPE, RMSE).
  + Generate an Evidently report (HTML).
* Store monitoring reports in data/monitoring/.

**4. Execution Timeline (Simplified)**

* **Week 1**: Ingest data, preprocess, create features.
* **Week 2**: Train baseline + 1 advanced model (SARIMAX/XGBoost). Log in MLflow.
* **Week 3**: Automate with Prefect, save predictions, build Streamlit dashboard.
* **Week 4**: Add monitoring (Evidently), track errors.

**5. Stretch Goals (Once Basics Work)**

* Use **Docker** to containerize your project.
* Store data in a **Postgres DB** instead of CSV.
* Deploy Streamlit app online (e.g., Streamlit Cloud).
* Add a **CI pipeline** (GitHub Actions).

👉 With this setup, you’ll cover the **MLOps lifecycle** (ingest → preprocess → train → predict → automate → monitor) but without heavy infrastructure.