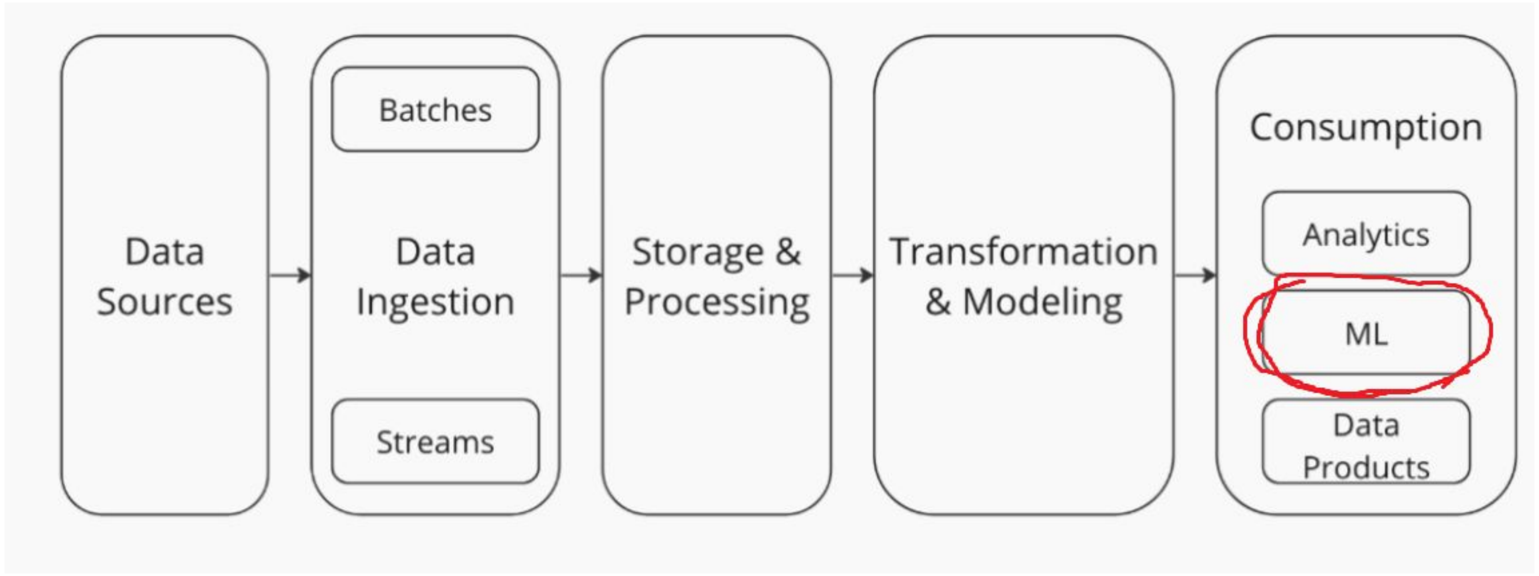
Deploying and Running Machine Learning Models in Google Cloud Platform

Once your ELT pipeline prepares data in BigQuery, you need to deploy your trained ML model to GCP and run predictions (inferences) on that data



What Does It Mean to Deploy and Run an ML Model?

Deployment: Saving and hosting a trained model in the cloud so it's accessible for predictions.

Inference: Using the model to predict outcomes on new data

Types of Inference:

- **Batch Inference**: Process a large dataset at once (e.g., predict for all new BQ rows daily).
- **Real-Time Inference**: Predict for single or small inputs via an API (e.g., user submits data, gets a prediction).

Saving your model

A trained model (e.g., scikit-learn, XGBoost) must be saved in a format for deployment.

Use Python’s **pickle** or **joblib** to save models (e.g., scikit-learn’s RandomForestClassifier).

```python
import pickle
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier().fit(X_train, y_train)
with open('model.pkl', 'wb') as f:
    pickle.dump(model, f)
```

pickle and joblib

Pickle: Part of Python's standard library (import pickle).

A general-purpose serialization tool for **any Python object** (e.g., lists, dictionaries, custom classes, ML models), as long as the object is pickleable (i.e., doesn't contain unserializable elements like file handles or database connections).

Designed for flexibility across all Python objects, not optimized specifically for ML models.

joblib: An external library (pip install joblib), built on top of pickle but optimized for **machine learning workflows**, especially for libraries like scikit-learn.

- Focuses on efficiently handling large numerical data structures (e.g., NumPy arrays, which are common in ML models like scikit-learn's RandomForestClassifier).
- Often used in ML because it's faster and produces smaller files for models with large arrays.

Implementation

```python
import pickle
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier().fit(X_train, y_train)
with open('model.pkl', 'wb') as f:
    pickle.dump(model, f)
with open('model.pkl', 'rb') as f:
    model = pickle.load(f)
```

```python
import joblib
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier().fit(X_train, y_train)
joblib.dump(model, 'model.joblib')
model = joblib.load('model.joblib')
```

Upload your model

Upload model.pkl (or joblib) to Cloud Storage (gs://your-bucket/model.pkl) for access during deployment.

Deployment options

Vertex AI (Managed ML Platform):

- Upload model to Vertex AI, deploy to an endpoint (real-time) or run batch predictions.
- Ideal for production, auto-scaling, and BigQuery integration. (according to Google)
- Example: Import model.pkl to Vertex AI, deploy to endpoint, query BQ for data.

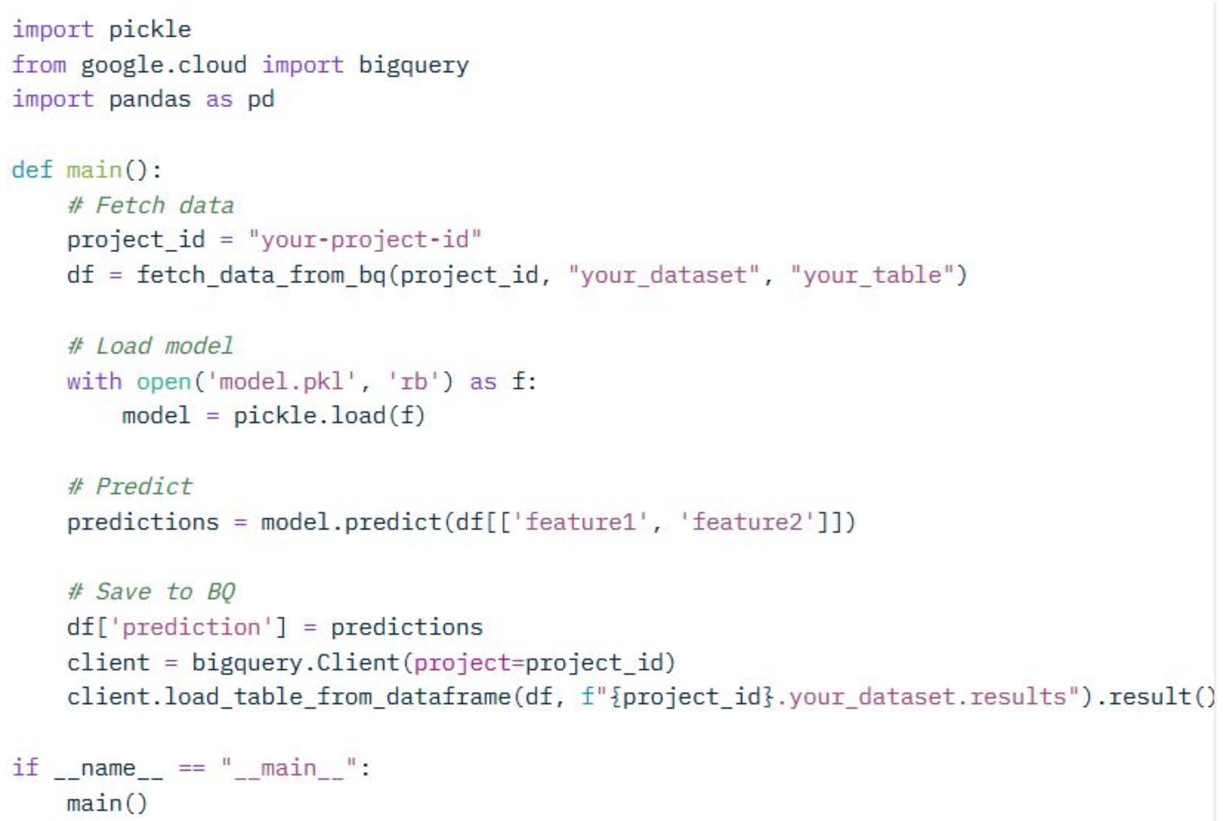
Cloud Run Jobs (Batch Inference):

- Run a containerized script to process BQ data and save predictions.
- Example: Query BQ, load model.pkl, predict, save to BQ or Storage.
- Great for scheduled tasks (e.g., daily predictions).

Cloud Run Services with FastAPI

- Host model in a FastAPI app, expose an API endpoint for predictions.
- Can Query BQ on-demand based on user input.
- Example: GET /predict?user_id=123 fetches BQ data and returns predictions.

What your app/script/model will probably need



```python
import pickle
from google.cloud import bigquery
import pandas as pd

def main():
    # Fetch data
    project_id = "your-project-id"
    df = fetch_data_from_bq(project_id, "your_dataset", "your_table")

    # Load model
    with open('model.pkl', 'rb') as f:
        model = pickle.load(f)

    # Predict
    predictions = model.predict(df[['feature1', 'feature2']])

    # Save to BQ
    df['prediction'] = predictions
    client = bigquery.Client(project=project_id)
    client.load_table_from_dataframe(df, f"{project_id}.your_dataset.results").result()

if __name__ == "__main__":
    main()
```