Setup Guide: AI Cement Plant Optimization Platform

Master the deployment of a state-of-the-art, end-to-end AI platform on Google Cloud and Firebase. This definitive guide provides the complete, refined framework for real-time Cement Plant performance monitoring and the automated delivery of intelligent, prescriptive operational intelligence, guaranteeing optimized output across all dynamic operating conditions.

I. Google Cloud Project & Core Setup

First, set up your core GCP project, data storage, and enable necessary services.

1. Create Project and Enable APIs

1. **Create a New GCP Project:**
    - Create a new project in the Google Cloud Console. Use a unique ID, such as `genai-cement-optimizer-hack`. We will refer to this as `<PROJECT-ID>`.
2. **Enable Required APIs:**
    - Run the following commands in your local terminal (ensure `gcloud` CLI is installed and configured to your `<PROJECT-ID>`):

```
 # Core Services
gcloud services enable compute.googleapis.com
gcloud services enable cloudstorage.googleapis.com
gcloud services enable bigquery.googleapis.com

# AI/ML Services
gcloud services enable aiplatform.googleapis.com

# Deployment Services (Cloud Run, Artifact Registry, Firebase)
gcloud services enable cloudbuild.googleapis.com
gcloud services enable run.googleapis.com
gcloud services enable artifactregistry.googleapis.com
gcloud services enable firebase.googleapis.com
```

2. Google Cloud Storage (GCS) - Data and Image Staging

> The **raw telemetry data** is staged in GCS, then transformed and made available for Vertex AI via BigQuery. The **raw image data** is also staged in GCS, then transformed and made available for Vertex AI via BigQuery.

1. **Create Data Bucket:**
    - Navigate to **Cloud Storage**. Click **CREATE BUCKET**.
    - Name it uniquely (e.g., `cement-optimizer-data-<PROJECT-ID>`).
    - Select a **Region** (e.g., `us-central1`). **Record this region**, as all subsequent services (BigQuery, Vertex AI, Cloud Run) must use it for optimal performance.
    - Upload your raw data file (`synthetic_cement_plant_data_v4.csv`) to this bucket.
2. **Create Few-Shot Image Folders (for Gemini):**
    - In the same bucket (or a new image-specific bucket), create the following folders, which hold the example images required for **Few-Shot Prompting** in the Gemini vision services:
        - ** `kiln/` ** (for Kiln image analysis examples)
        - ** `conveyor/` ** (for Conveyor Belt image analysis examples)
    - Upload the relevant example images (e.g., `kiln_operating_normal_1.jpg`, `conveyor_belt_normal.jpg`) into their respective folders.

II. BigQuery - Data Ingestion and Transformation

This prepares the data for the two target KPIs: `clinker_free_lime_pct` and `kiln_specific_thermal_energy`.

1. **Create Dataset:**
    - Navigate to **BigQuery**. Click your project name, then **CREATE DATASET**.
    - Name it: `cement_data` and select the **same Region** as your GCS bucket.
2. **Create Table from GCS (Raw Data):**
    - Create a table named `raw_plant_telemetry` by ingesting your CSV file from GCS, using **Auto detect** for the schema.
3. **Create the Training Data View (Lagged Features):**
    - This view calculates the necessary features, such as lagged sensor readings.
    - Click **COMPOSE NEW QUERY** and run your SQL script.
    - The resulting View, ** `cement_data.training_data_view` **, will be the data source for both AutoML models.

```
 CREATE OR REPLACE VIEW `cement_data.training_data_view` AS
SELECT
```

```
    *, -- All original columns
    LAG(raw_meal_lsf_ratio, 300) OVER (ORDER BY timestamp) AS raw_meal_lsf_ratio_lag_300
    -- Include all other necessary lagged feature calculations here (e.g., ~5 hours of lag)
FROM
    `cement_data.raw_plant_telemetry`
WHERE
    -- Filter out initial rows where lag values are NULL
    raw_meal_lsf_ratio_lag_300 IS NOT NULL;
```

III. Vertex AI - Model Training and Deployment

You will train two models (one for each KPI) and deploy them to their own dedicated endpoints for low-latency predictions.

1. AutoML Training Job (Repeat for 2 Models)

1. **Create Vertex AI Dataset:** (Repeat once for each KPI model)
    - Navigate to **Vertex AI** $\rightarrow$ **Datasets**. Click **CREATE**.
    - **Name:** E.g., `clinker_free_lime_pct_dataset`.
    - **Data Source:** Use the BigQuery View URI: `bq://<PROJECT-ID>.cement_data.training_data_view`.
2. **Start Training:** (Repeat once for each KPI model)
    - Open the dataset and click **TRAIN NEW MODEL**.
    - **Training Method:** **AutoML**.
    - **Model Objective:** **Regression**.
    - **Target column:** Select the target KPI (e.g., `clinker_free_lime_pct` for the first model, then `kiln_specific_thermal_energy` for the second).
    - **Optimization objective:** Minimize **Root Mean Squared Error (RMSE)**.
    - Set a reasonable **Training budget** (e.g., 4-6 compute hours).

2. Model Deployment to Endpoint (2 Endpoints)

You must create a separate, dedicated endpoint for each of your **two trained models**.

1. **Deploy Model:** (Repeat for both trained models)
    - Go to **Vertex AI** $\rightarrow$ **Models**. Select a trained model. Go to the **Deploy & test** tab.
    - Click **Deploy to endpoint**.
    - **Endpoint name:** Use a clear, descriptive name (e.g., `clinker-free-lime-pct-endpoint`).
    - **Machine type:** Select a suitable type (e.g., `n1-standard-2`).
    - **Min/Max compute nodes:** Set both to **1**.

- Click **Deploy**.

2. **Record Endpoint IDs:**
   - Go to **Vertex AI** $\rightarrow$ **Online Prediction** $\rightarrow$ **Endpoints**.
   - **Copy the unique Endpoint ID** for each of your **two deployed models**. These IDs are critical configuration variables for your backend logic.

IV. Cloud Run - Backend Services Deployment (5 Services)

You are deploying **five** distinct services in Cloud Run to handle the two-part optimization and two-part image analysis. All services must use the **same region** as your data.

1. Service Account IAM Setup

Your Cloud Run services need permission to interact with Vertex AI, Gemini, and GCS.

1. **Identify Service Account (SA):** The default SA is typically the Compute Engine default: `PROJECT_NUMBER-compute@developer.gserviceaccount.com`.
2. **Grant Roles:** Go to **IAM & Admin** $\rightarrow$ **IAM** and grant the following roles to this Service Account:
   - **Vertex AI User** (`roles/aiplatform.user`) - Allows the services to call the AutoML endpoints and Gemini via Vertex AI.
   - **Storage Object Viewer** (`roles/storage.objectViewer`) - Allows the Image Analysis services to read the few-shot images from your GCS bucket.

2. Deploy Cloud Run Services

You will run the `gcloud run deploy` command **five times**, once for each distinct microservice.

#	Service Name	Purpose	Notes
1	`clinker-kpi-predictor`	Predicts `clinker_free_lime_pct`.	Calls the first AutoML Endpoint.
2	`energy-kpi-predictor`	Predicts `kiln_specific_thermal_energy`.	Calls the second AutoML Endpoint.
3	`optimization-recommender`	Receives 2 KPI predictions, uses **Gemini** to provide operational recommendations.	Single model handles both KPIs.

#	Service Name	Purpose	Notes
4	`kiln-image-analyzer`	Uses **Gemini** (Vision) for Kiln image defect analysis.	Reads from GCS `kiln/` folder.
5	`conveyor-image-analyzer`	Uses **Gemini** (Vision) for Conveyor Belt image defect analysis.	Reads from GCS `conveyor/` folder.

Deployment Command Template (Repeat 5 times, changing the service name):

```
gcloud run deploy <SERVICE-NAME> \
  --source . \
  --region us-central1 \
  --platform managed \
  --allow-unauthenticated \
  --min-instances 0 \
  --max-instances 2
```

3. Record All Service URLs

- After each deployment, **copy the Service URL** from the terminal output. These **five URLs** are the API endpoints that your frontend will call.

V. Firebase Hosting - Frontend Deployment

The application frontend is a static HTML/JS file hosted via Firebase.

1. Initialize Firebase Project

1. **Log In and Initialize:**

    ```
    firebase login
    firebase init
    ```

2. **Follow Prompts:**
    - Select **Hosting: Configure files for Firebase Hosting**.
    - Choose **Use an existing project** and select your GCP project ID.
    - **Public directory?** Type `public`.
    - **Configure as a single-page app?** Type `N` (No).

2. Prepare and Deploy Frontend

1. **Update Cloud Run URLs in `index.html`:**
    - In your local `public/index.html` file, update the JavaScript logic to call the **five Cloud Run Service URLs** you recorded in the previous step. You will have separate POST requests for each service.
2. **Deploy the Application:**

    ```
    firebase deploy --only hosting
    ```

3. **Verify and Test:** The output will provide the final **Hosting URL** (e.g., `https://[Your-Project-ID].web.app`).

VI. Technology Stack Summary

This solution leverages a modern, serverless, and decoupled architecture on Google Cloud Platform and Firebase, utilizing specialized services for each phase of the MLOps pipeline and application serving.

GCP/Firebase Service	Category	Role in the Solution
Cloud Run	Serverless Compute	Hosts the **five backend microservices**: 2 KPI Predictors, 1 Gemini Recommender, 2 Gemini Image Analyzers. Provides automatic scaling from zero.
Vertex AI (AutoML)	Machine Learning	Used for **training the two Regression models** that predict the critical KPIs.
Vertex AI (Generative AI)	Generative AI	Powers the **Optimization Recommender** and the **Kiln/Conveyor Image Analyzers** using multimodal Few-Shot prompting.
Vertex AI (Endpoints)	Machine Learning	Provides **managed, low-latency deployment endpoints** to serve the predictions from the two trained AutoML models.
BigQuery	Data Warehouse	**Stores, manages, and transforms** the plant telemetry data, creating the training data view with lagged features.

GCP/Firebase Service	Category	Role in the Solution
Cloud Storage (GCS)	Storage	Serves as the **central staging area** for raw data, model artifacts, and Few-Shot images.
Firebase Hosting	Frontend Hosting	Provides **fast, secure, and globally distributed hosting** for the web application frontend.