## **Project Plan: Real-Time Disaster Alert & Impact Assessment System**

This document outlines the step-by-step plan to build your project. We will progress through each of these stages, and I will provide the necessary instructions, code, and configuration details for each one.

### **Phase 1: Foundational Setup & Data Ingestion**

* **Step 1: Google Cloud Project Setup**
  + Create a new Google Cloud Project.
  + Ensure a billing account is linked.
  + Enable all the necessary APIs for the services we'll use (Pub/Sub, Dataflow, BigQuery, Vertex AI, Cloud Run, etc.).
* **Step 2: Set Up Real-Time Messaging with Pub/Sub**
  + Create a Pub/Sub topic to receive raw disaster alerts from our data sources.
  + This will act as the central entry point for all incoming event data.
* **Step 3: Accessing Disaster Data APIs**
  + Obtain API keys or access to public data feeds from:
    - **USGS (United States Geological Survey):** For real-time earthquake data.
    - **NASA EONET (Earth Observatory Natural Event Tracker):** For a wide range of natural events (wildfires, storms, etc.).
    - **Google Geocoding API:** To convert latitude/longitude into human-readable addresses.
* **Step 4: Create a Data Ingestion Cloud Function**
  + Write and deploy a Python-based Cloud Function.
  + This function will be triggered on a schedule (e.g., every 5 minutes).
  + It will call the USGS and NASA APIs, fetch the latest disaster data, format it, and publish it to the Pub/Sub topic created in Step 2.

### **Phase 2: Data Processing & Enrichment**

* **Step 5: Prepare BigQuery for Data Storage**
  + Create a BigQuery dataset to hold our project tables.
  + Define the schema for our main "disaster events" table, which will store the enriched data.
  + Load a public dataset with demographic or infrastructure data (e.g., population density, locations of hospitals) into another BigQuery table for enrichment.
* **Step 6: Build the Streaming Dataflow Pipeline**
  + Develop a streaming pipeline in Python using the Apache Beam SDK.
  + This pipeline will:
    1. Read new disaster events from the Pub/Sub topic in real-time.
    2. Parse and validate the incoming data.
    3. Call the Google Geocoding API to get address details for the event's coordinates.
    4. Perform a lookup against our demographic/infrastructure table in BigQuery to enrich the event with data like local population density.
    5. Write the final, enriched data into our main BigQuery table.

### **Phase 3: Machine Learning & Visualization**

* **Step 7: Develop and Deploy an Impact Scoring Model on Vertex AI**
  + Prepare a training dataset using historical disaster and demographic data from BigQuery.
  + Train a simple machine learning model (e.g., a regression model) in a Vertex AI Notebook to predict an "impact score" based on factors like magnitude, population density, and proximity to critical infrastructure.
  + Deploy the trained model to a Vertex AI Endpoint to make it available for real-time predictions.
  + Update the Dataflow pipeline to call this ML model for each new event and add the impact score to the data written to BigQuery.
* **Step 8: Visualize Data with Looker Studio**
  + Connect Looker Studio to your BigQuery "disaster events" table.
  + Build an interactive dashboard that displays:
    - A map showing the real-time location of disaster events.
    - Tables and charts detailing the events and their impact scores.
    - Filters to slice the data by disaster type, date, and severity.
* **Step 9 (Optional): Create an Interactive Frontend with Streamlit & Cloud Run**
  + Develop a simple web application using the Streamlit library in Python.
  + This application will query BigQuery directly to display the disaster data in a user-friendly interface.
  + Package the Streamlit app into a Docker container and deploy it on Cloud Run for a scalable, serverless web frontend.

Let's begin with the first step. Are you ready to set up your Google Cloud project?