PROOF OF CONCEPT: LM with Google Drive

CONCEPT

1. **Centralize data** stored in Google Drive (Excel, PDFs, etc.) without requiring users to manually open files.
2. **Enable natural language queries** and provide answers based on file contents.
3. **Manage access permissions** to control data authorization.

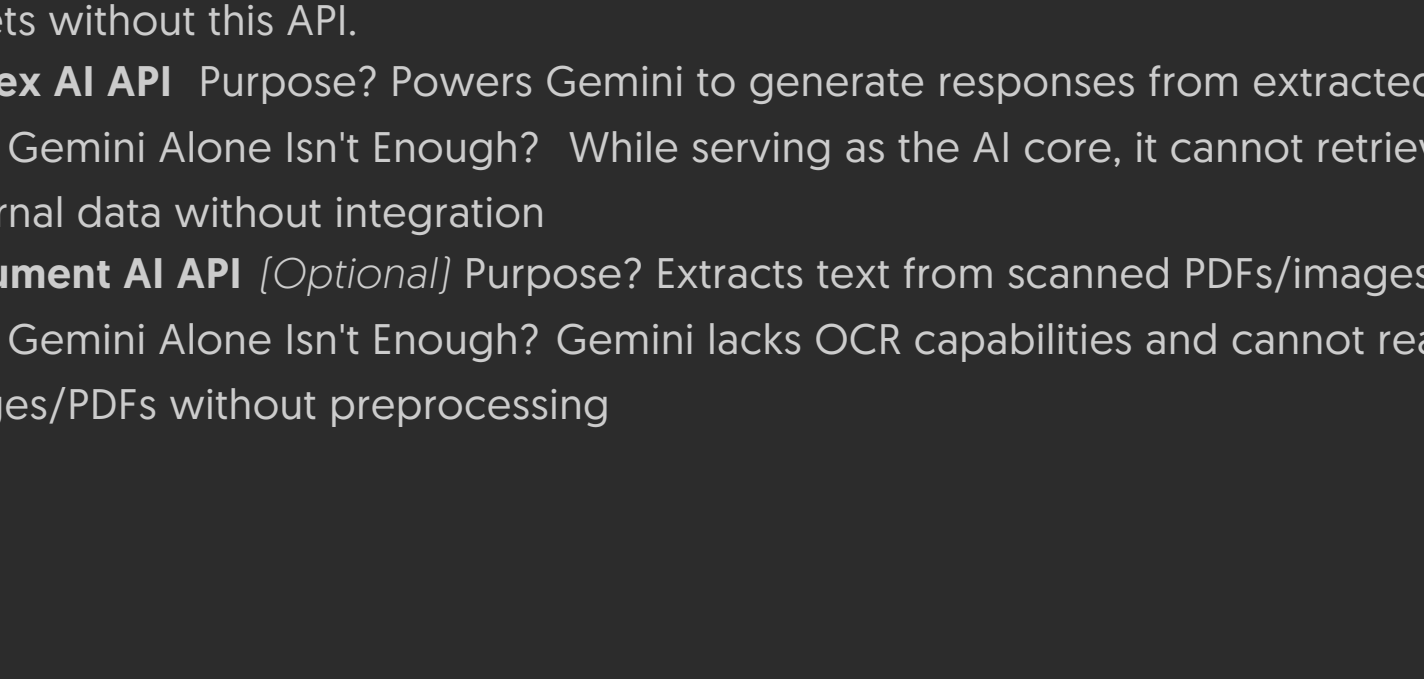
SOLUTION & ARCHITECTURE

The **Google Drive API, Google Sheets API, Vertex AI API, and optionally Document AI API** serve specific functions in your project. You cannot use only Gemini (Vertex AI API) because it lacks direct access to your Google Drive files. Here's why each API is essential:

Why You Need All 4 APIs (Key Functions)

1. ***Google Drive API*** *¿Purpose?* Reads folder/file structures in DriveWhy Gemini Alone Isn't Enough? Gemini cannot browse your Drive independently - this API finds relevant files
2. **Google Sheets API** ¿Purpose? Extracts spreadsheet data (values, formulas, formatting) Why Gemini Alone Isn't Enough? Gemini only processes plain text and cannot access Sheets without this API.
3. **Vertex AI API** Purpose? Powers Gemini to generate responses from extracted data Why Gemini Alone Isn't Enough? While serving as the AI core, it cannot retrieve external data without integration
4. **Document AI API** *(Optional)* Purpose? Extracts text from scanned PDFs/images (OCR) Why Gemini Alone Isn't Enough? Gemini lacks OCR capabilities and cannot read text in images/PDFs without preprocessing

Gemini's capabilities expand with API integration.



Practical Example

User Question: *"What were Q1 2024 sales according to 'Sales.xlsx'?"*

Workflow:

1. **Drive API**: Finds Sales.xlsx in your folders.
2. **Sheets API**: Extracts numerical data from the file.
3. **Vertex AI (Gemini)**: Analyzes data and generates the answer.
4. **Document AI** *(if PDF is scanned)*: Converts images to text.

→ **Without Drive/Sheets APIs**, Gemini would respond: *"I don't have access to your files."*

What Happens If You Use Only Gemini?

- **Critical limitation**: Gemini is a language model—**not a data-access tool**.
- **It cannot**:
  - List files in Drive.
  - Open/read Sheets, Docs, or PDFs.
  - Run structured queries (e.g., filter Excel data).

Alternative (Less Efficient) Workarounds

1. **Manual copy-paste**: Extract data and paste into Gemini's prompt.
   - *Problem*: Impractical for dynamic/multiple files.
2. **Google Apps Script alone**: Read simple files without APIs.
   - *Limits*: No OCR, 6-minute runtime cap.

Conclusion:

- The **API combination is mandatory** because:
  - Gemini is "blind" to your files without Drive/Sheets APIs.
  - Each API handles a unique task (data access → parsing → AI analysis).
- **Start simple** (Drive + Vertex AI for text files), then scale with permissions/OCR.

Steps to Activate APIs in Google Cloud Console

Access Google Cloud Console

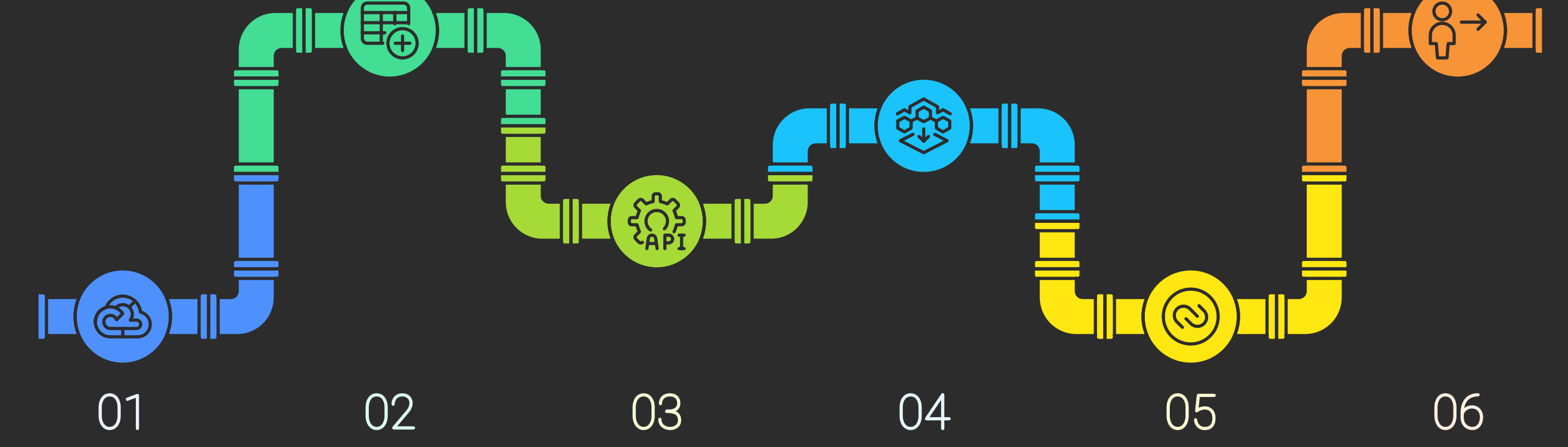
- Go to console.cloud.google.com
- Sign in with your Google account (the one linked to Google Drive)
- **Create a New Project**
  - Click the project selector (top-left dropdown) → "New Project"
  - Name: Drive-AI-Integration
  - Click "Create"
- **Enable APIs**
  - In the left menu, navigate to **APIs & Services > Library**
  - Search for and enable these APIs one by one:
- **Google Drive API** Function Search Term: Access Drive files"Google Drive API"**Google Sheets API** Function Search Term: Read spreadsheet data"Google Sheets API"**Vertex AI API** Function Search Term: Use AI models (Gemini)"Vertex AI API"**Cloud Document AI API** (Optional for Phase 3) Process scanned PDFs"Cloud Document AI API"

- **Enable Vertex AI for Gemini**
  - Go to **Vertex AI > Language Models** in the menu
  - Click **"Enable API"** (if not already active)
  - Accept the terms of use
- **Configure Credentials**
  - Navigate to **APIs & Services > Credentials**
  - Click **"+ CREATE CREDENTIALS" > "API Key"**
  - Copy and save the generated key (you'll use it in Apps Script)
- **Set Up OAuth Consent Screen**
  - Under **Credentials**, click **"Configure Consent Screen"**
  - User Type: **External** (Internal requires Google Workspace)
  - App Name: Connectivism_AI-Drive
  - In **Google Auth Platform / Data Access**:
    - Click **"Add or Remove Scopes"**
    - Manually add these authorization scopes:

```
https://www.googleapis.com/auth/drive.readonly
https://www.googleapis.com/auth/cloud-platform
```

- Add your email under **Test Users**

Activating APIs in Google Cloud Console



Configuration in Apps Script

Here's how to properly create, open, and configure your project in Google Apps Script to integrate it with Google Drive and Vertex AI:

Step 1: Accessing Apps Script

1. **Open your browser** and go to: https://script.google.com *(Sign in with the same Google account connected to your Drive files - e.g., xxxxxxxx@gmail.com)*

2. **Create a new project**:
   - Click **"+ New project"** (top-left corner)
   - Name it: Connectivism_test1
   - A blank editor will open with a default file named Code.gs

Step 2: Configure the Project Structure

Your project requires these minimum files:

appsscript.json (Configuration Manifest)

1. Access the manifest editor:
   - Click on the **"Manifest editor"** (left menu > gear icon)
2. If the file isn't visible:
   - Go to **View > Show manifest file**
   - If still not appearing, create a new script file named exactly:appsscript.json

Code.gs (Main Script File)

1. This file is created automatically
2. Recommended actions:
   - **Delete the default template code**
   - **Replace its contents** with your custom integration script

```
const FOLDER_ID = "YOUR_DRIVE_FOLDER_ID";  // Replace with actual ID
const VERTEX_AI_KEY = "YOUR_VERTEX_AI_API_KEY"; // From Google Cloud Console

// Main function to process questions
function processQuestion(question) {
  try {
    // 1. Find relevant files in Drive
    const files = findRelevantFiles(question);

    // 2. Extract content (PDFs, Sheets, etc.)
    const context = extractContent(files);

    // 3. Query Vertex AI (Gemini)
    return queryGemini(context, question);

  } catch (error) {
    return ❌ Error: ${error.message};
  }
}

// Helper function to search for files
function findRelevantFiles(query) {
  const folder = DriveApp.getFolderById(FOLDER_ID);
  const files = folder.getFiles();
  const results = [];

  while (files.hasNext()) {
    const file = files.next();
    if (file.getName().toLowerCase().includes(query.toLowerCase())) {
      results.push(file);
    }
  }
  return results;
}

// Helper function to extract content
function extractContent(files) {
  return files.map(file => {
    switch (file.getMimeType()) {
      case MimeType.PDF:
        return extractPDFText(file);
      case MimeType.GOOGLE_SHEETS:
        return extractSheetData(file);
      default:
        return Unprocessable content: ${file.getName()};
    }
  }).join('\n\n');
}

// Function to extract text from PDFs (basic version)
function extractPDFText(file) {
  // Basic implementation for testing
  return "Sample text extracted from PDF"; // Improve with Document AI in Phase 3
}

// Function to extract data from Google Sheets
function extractSheetData(file) {
  const ss = SpreadsheetApp.openById(file.getId());
  return ss.getSheets()[0].getDataRange().getValues()
    .map(row => row.join(', '))
    .join('\n');
}

// Connection to Vertex AI (Gemini)
function queryGemini(context, question) {
  const endpoint =
https://us-central1-aiplatform.googleapis.com/v1/projects/your-project/locations/us-central1/
publishers/google/models/gemini-pro:streamGenerateContent?key=${VERTEX_AI_KEY};

  const prompt = `
  CONTEXT (extracted from documents):
  ${context}

  USER QUESTION:
  ${question}

  INSTRUCTIONS:
  - Respond only with verified data from the context
  - Use clear and concise format
  - If no relevant information exists, say "No data found"
  `;

  const options = {
    method: 'post',
    contentType: 'application/json',
    payload: JSON.stringify({
      contents: [{
        parts: [{ text: prompt }]
      }]
    })
  };

  const response = UrlFetchApp.fetch(endpoint, options);
  const data = JSON.parse(response.getContentText());
  return data.candidates[0].content.parts[0].text;
}
```

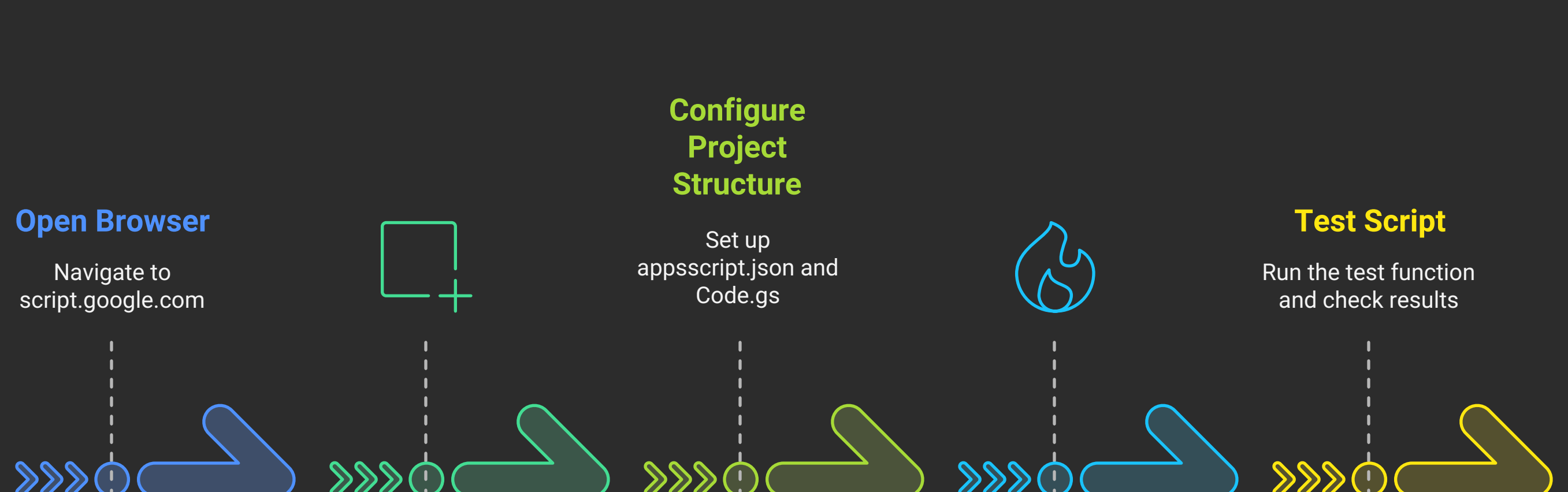
How to Test It?

Create a temporary test function (at the end of the file):

```
function test() {
  const response = processQuestion("Example question");
  Logger.log(response); // View results in Executions
}
```

- To Run test():
  1. **Select the test function** from the dropdown menu in the script editor
  2. **Click** the ▶ **Run** button

Setting Up and Testing a Google Apps Script Project



Step 3: Manifest Configuration (appsscript.json)

Edit the file to match this configuration:

json

```json
{
  "timeZone": "Europe/Warsaw",
  "dependencies": {
    "enabledAdvancedServices": [
      {
        "userSymbol": "Drive",
        "serviceId": "drive",
        "version": "v3"
      }
    ]
  },
  "oauthScopes": [
    "https://www.googleapis.com/auth/drive.readonly",
    "https://www.googleapis.com/auth/cloud-platform"
  ],
  "runtimeVersion": "V8"
}
```

Key explanation:

- oauthScopes: The permissions you will ask the user for..
- runtimeVersion: Use the V8 engine (modern and fast).
- enabledAdvancedServices: Enable the Drive API directly from Apps Script.

Step 4: Run for the First Time

1. Click on "Run" (▶ in the top menu)..
2. Authorise the permissions:
   - You will be prompted to review the scopes.
   - Click "Review Permissions" > "Advanced" > "Go to [Project Name]".
   - Check the boxes and accept.

Running and Authorizing the Script