Chapter 4: GSuite MCP server

Work Smarter, Not Harder: AI Brings GSuite to Life!

4.1 Introduction

The GSuite Model Context Protocol (MCP) is a convenient tool that allows your AI assistant to handle your Gmail, Google Calendar, Google Maps, etc through simple, natural language commands. Talking about the Gmail automation specifically, whether you need to check your inbox, send emails, or organize your schedule, Gmail MCP makes it easy without the need for manual navigation.
Though in this chapter, we will demonstrate just Gmail integration. Rest of the integrations are also similar.
Using GSuite MCP, your assistant can perform tasks like listing recent emails, conducting advanced searches, sending new messages, modifying email labels, viewing upcoming calendar events, creating new events with attendees, and updating existing ones.

4.1.1 GSuite MCP tools (Gmail)

GSuite MCP for Gmail offers a suite of tools that empower your AI assistant to manage your Gmail inbox efficiently using natural language commands. Here's a concise overview of the available tools:

1. **list_emails** : Retrieve recent emails from your inbox with optional filtering.
   Example: "List the latest 5 unread emails."
2. **search_emails** : Perform advanced email searches using Gmail's query syntax.

Example: "Search for emails from 'client@example.com' with attachments."

3. **send_email**: Compose and send new emails, including support for CC and BCC fields.
   Example: "Send an email to 'team@example.com' with the subject 'Meeting Update' and CC 'manager@example.com'."

4. **modify_email**: Update email labels to archive, trash, or mark emails as read/unread.
   Example: "Mark the email from 'newsletter@example.com' as read and archive it.

4.1.1 Real-World Applications

Amit, a customer support lead, starts his day by checking for any urgent client emails that need immediate attention. To streamline his workflow, he turns to GSuite MCP for Gmail.
He prompts his assistant:

- *"List the latest emails from the inbox."*
- *"Search for emails from 'client@example.com' received this week."*
- *"Send an email to 'client@example.com' with the subject 'Follow-up on Support Ticket' and include the latest update in the body. CC 'manager@example.com'."*
- *"Mark the email from 'client@example.com' with the subject 'Issue Resolved' as read and archive it."*

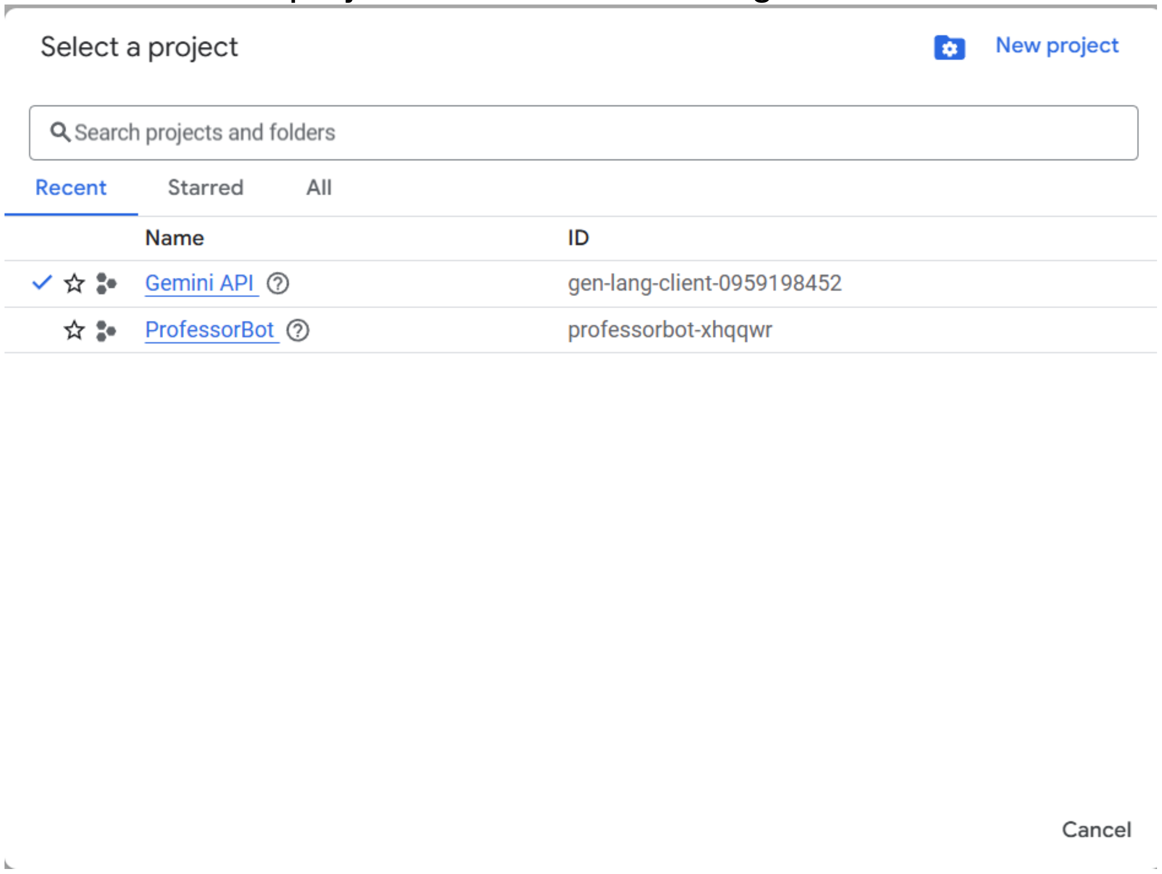
Within minutes, the assistant retrieves the necessary emails, sends the follow-up message, and organizes the inbox accordingly. Amit efficiently manages his communications, ensuring prompt responses and a well-organized inbox.

4.2 Step-by-Step Installation

4.2.1 Pre-requisites

1. Install Node.js version 14 or higher.

2. Go to the Google Cloud Console (https://console.cloud.google.com/).
3. Create a new project or select an existing one.

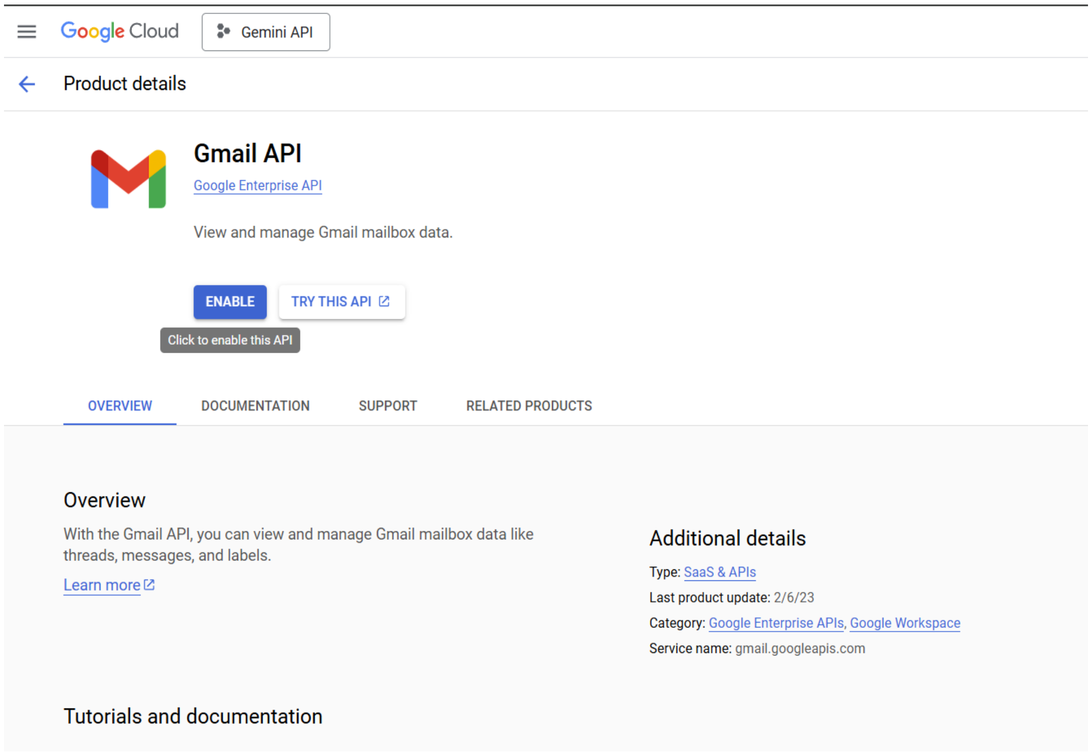


	Name	ID
✓ ☆	Gemini API	gen-lang-client-0959198452
☆	ProfessorBot	professorbot-xhqqwr

Project list in Console

4. Navigate to **"APIs & Services" > "Library"**. Search for and enable the Gmail API.

Note : *If you wish to use other GSuite MCPs like Google Calendar or Google Maps, search for those APIs as well and enable similarly*



View and manage Gmail mailbox data.

Overview

With the Gmail API, you can view and manage Gmail mailbox data like threads, messages, and labels.

Learn more

Additional details

Type: SaaS & APIs

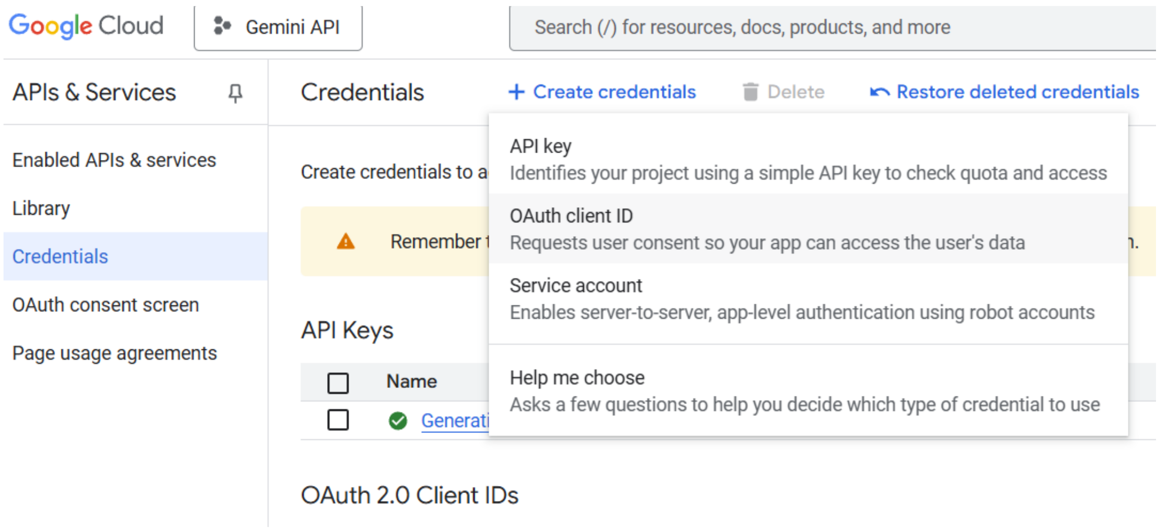
Last product update: 2/6/23

Category: Google Enterprise APIs, Google Workspace

Service name: gmail.googleapis.com

Enable Gmail API

5. **Setup OAuth 2.0 credentials:** Go to the Credentials page in the API library & click on *Create Credentials and select OAuth client ID*



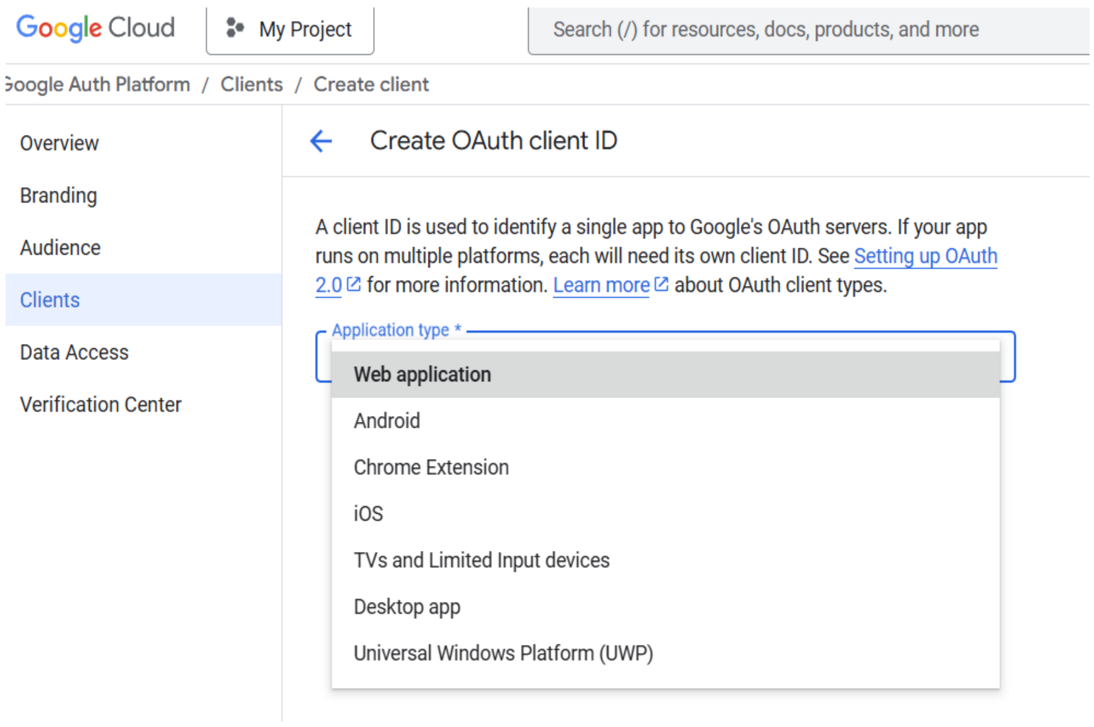
API key
Identifies your project using a simple API key to check quota and access

OAuth client ID
Requests user consent so your app can access the user's data

Service account
Enables server-to-server, app-level authentication using robot accounts

Help me choose
Asks a few questions to help you decide which type of credential to use

Create OAuth client ID


Google Cloud
My Project
Search (/) for resources, docs, products, and more
Google Auth Platform / Clients / Create client
Overview
Branding
Audience
Clients
Data Access
Verification Center

Create OAuth client ID

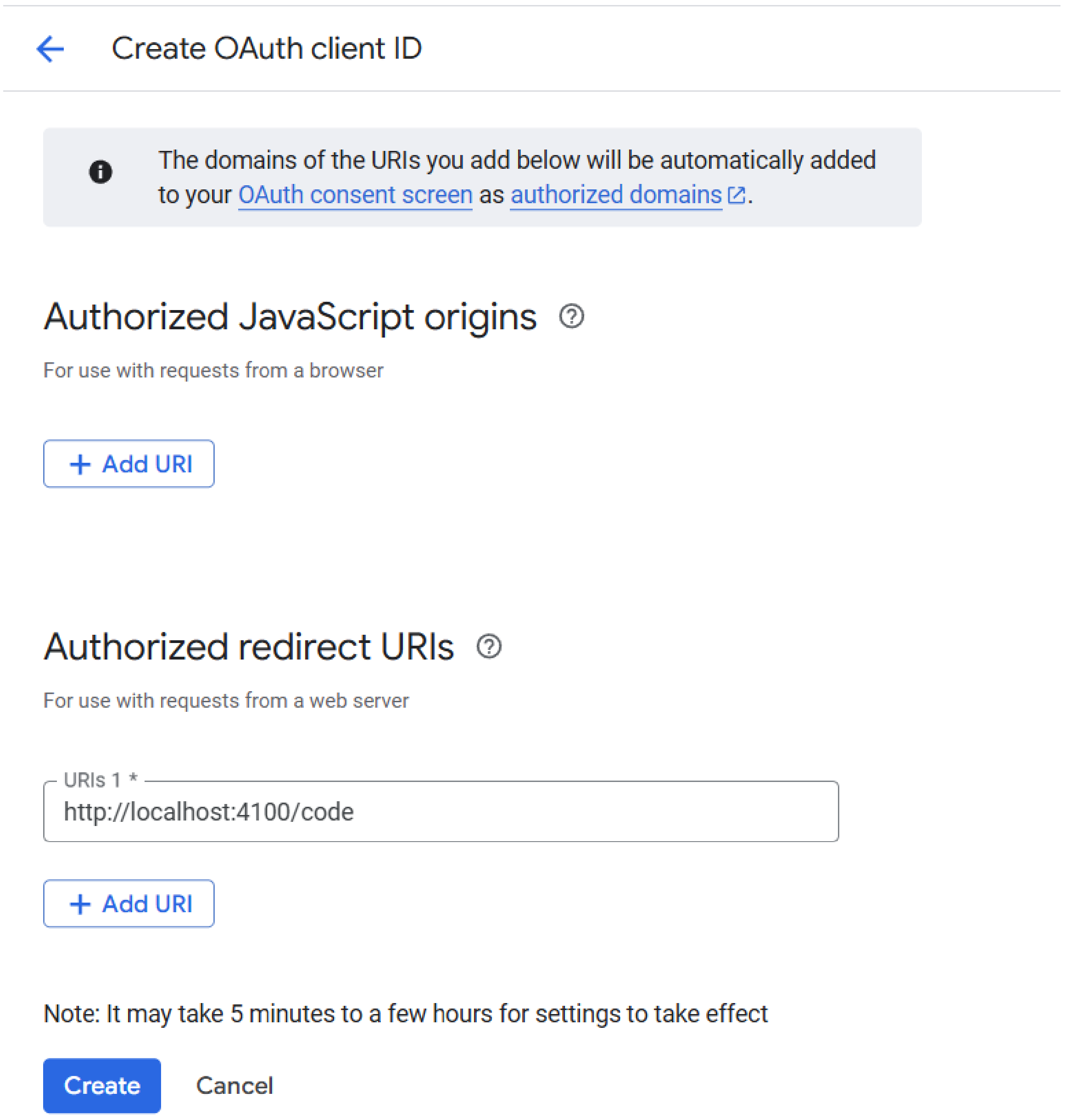
A client ID is used to identify a single app to Google's OAuth servers. If your app runs on multiple platforms, each will need its own client ID. See Setting up OAuth 2.0 for more information. Learn more about OAuth client types.

Application type *

- Web application
- Android
- Chrome Extension
- iOS
- TVs and Limited Input devices
- Desktop app
- Universal Windows Platform (UWP)

Select Web Application from the dropdown.

6. Set **"Authorized redirect URIs"** to include: http://localhost:4100/code


Create OAuth client ID

The domains of the URIs you add below will be automatically added to your OAuth consent screen as authorized domains.

Authorized JavaScript origins

For use with requests from a browser

Authorized redirect URIs

For use with requests from a web server

URIs 1 *
http://localhost:4100/code

Note: It may take 5 minutes to a few hours for settings to take effect

Redirect URI

7. Note down the Client ID and Client Secret. We will need these details in the later stage.

4.2.2 Steps

1. Clone and Install the MCP Server (run the below commands in Command-Line, cmd)

```
git clone https://github.com/epaproditus/google-workspace-mcp-server.git
```

```
cd google-workspace-mcp-server
npm install
```

2. **Create OAuth Credentials -** In the root directory, create a file named credentials.json with the following content:

```
{
  "web": {
    "client_id": "YOUR_CLIENT_ID",
    "client_secret": "YOUR_CLIENT_SECRET",
    "redirect_uris": ["http://localhost:4100/code"],
    "auth_uri": "https://accounts.google.com/o/oauth2/auth",
    "token_uri": "https://oauth2.googleapis.com/token"
  }
}
```

credentials.json

3. Obtain Refresh Token by running the below command inside the repo

   ```
   node get-refresh-token.js
   ```

   4. This will open your browser for Google OAuth authentication and request for some permissions. Approve them and save the credentials to **token.json** :

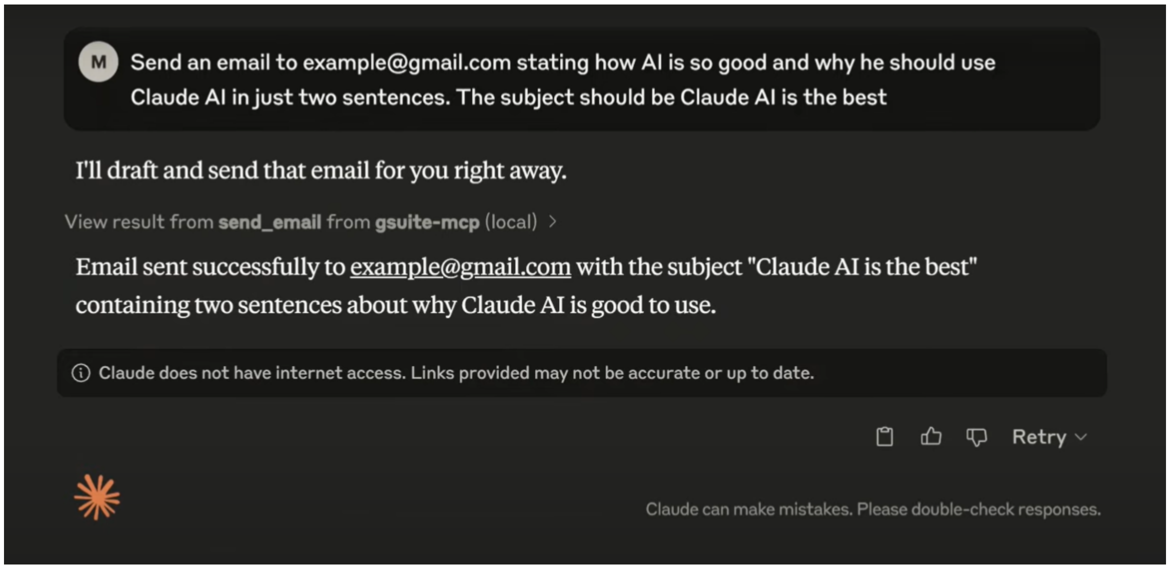
5. Add the server to Claude config json file

```
{
  "mcpServers": {
    "google-workspace": {
      "command": "node",
      "args": ["/path/to/google-workspace-server/build/index.js"],
      "env": {
        "GOOGLE_CLIENT_ID": "your_client_id",
        "GOOGLE_CLIENT_SECRET": "your_client_secret",
        "GOOGLE_REFRESH_TOKEN": "your_refresh_token"
      }
    }
  }
```

```
}
```

claude_desktop_config.json

6. Restart Claude (end from Task Manager). On restarting, you should the MCP tools in Claude message section



Send an email to example@gmail.com stating how AI is so good and why he should use Claude AI in just two sentences. The subject should be Claude AI is the best

I'll draft and send that email for you right away.

View result from **send_email** from **gsuite-mcp** (local)

Email sent successfully to example@gmail.com with the subject "Claude AI is the best" containing two sentences about why Claude AI is good to use.

Claude does not have internet access. Links provided may not be accurate or up to date.

And that’s it, you’ve now added Gmail MCP in your Claude AI. Similarly, you can add these settings to Cursor.ai, Windsurf or Cline and let AI operate your Gmail account. Also, if you wish to use other GSuite services (like Google Calendar etc.), this MCP server should support it too.