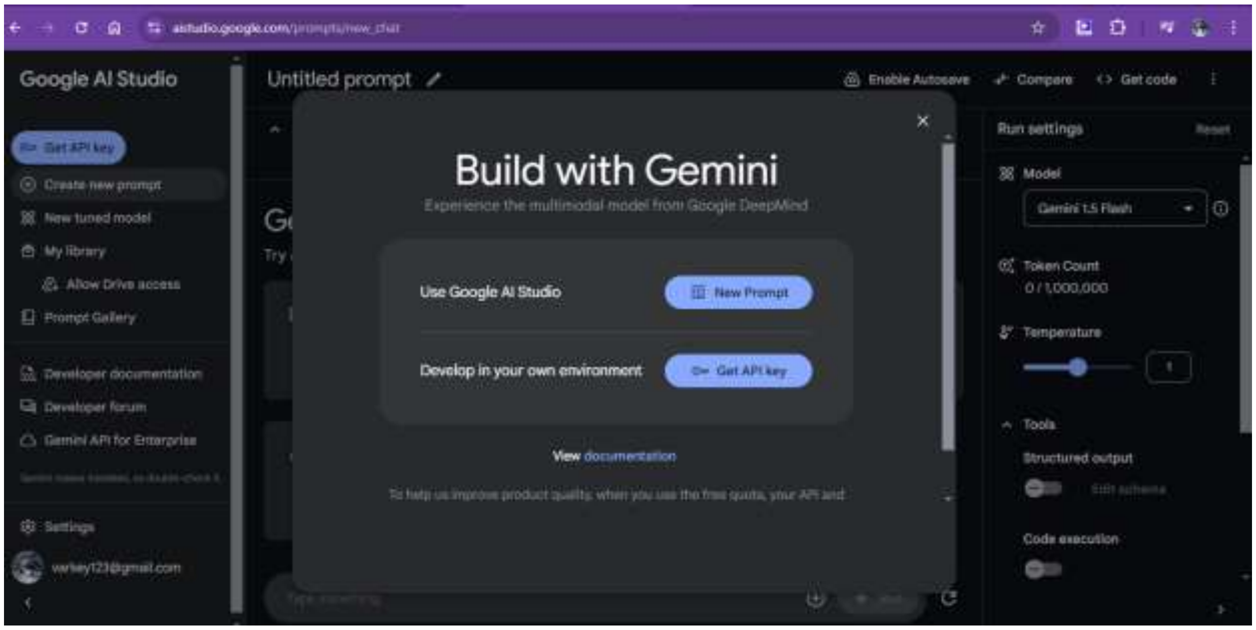
The very first step on all the projects mentioned in this note is to get the API key and store it in the environment variable.

This is place from where we can get the key - https://aistudio.google.com/prompts/new_chat



Build with Gemini

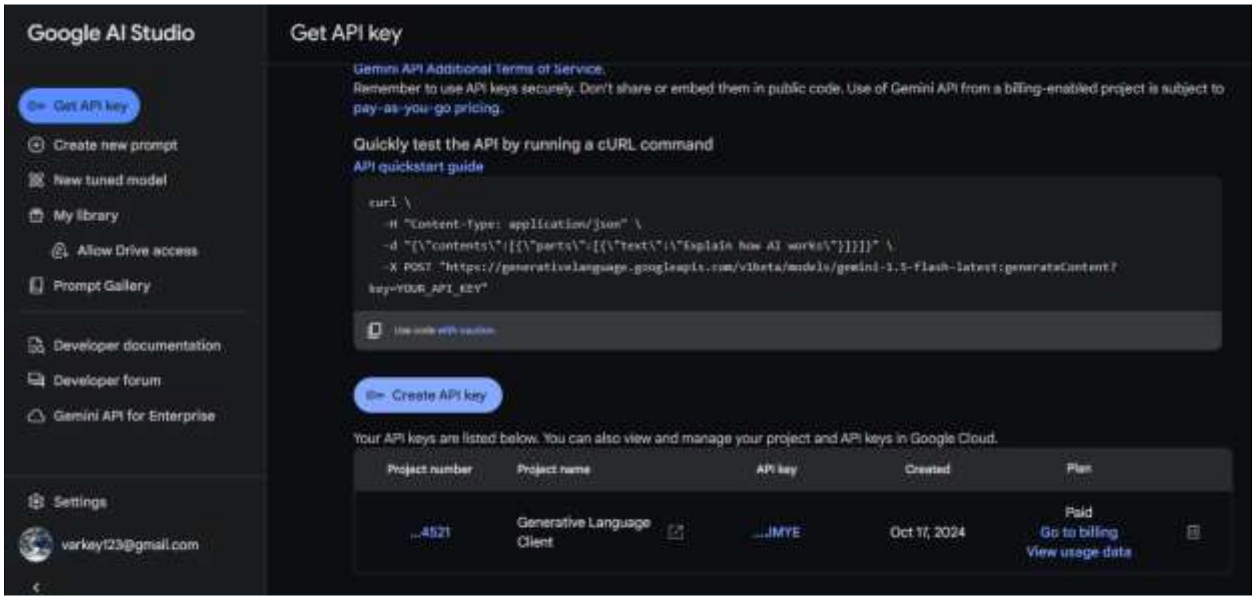
Experience the multimodal model from Google DeepMind

Use Google AI Studio	New Prompt
Develop in your own environment	Get API key

View documentation

To help us improve product quality, when you use the free quota, your API and

I have taken pay as you go model . Its important that we need to purchase the key for minimum usage



Get API key

Gemini API Additional Terms of Service.
Remember to use API keys securely. Don't share or embed them in public code. Use of Gemini API from a billing-enabled project is subject to pay-as-you-go pricing.

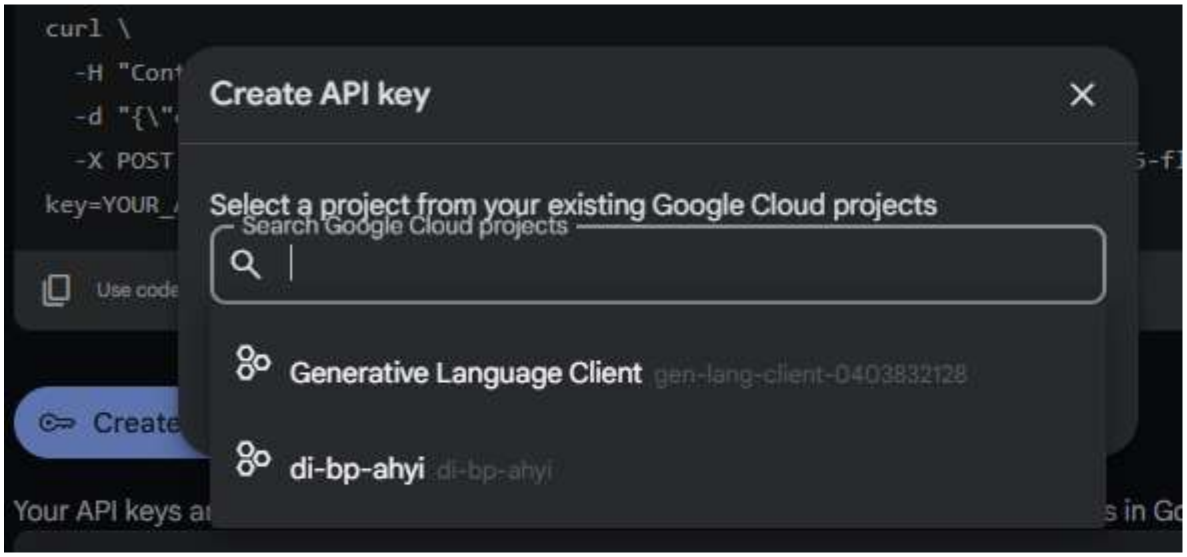
Quickly test the API by running a cURL command
API quickstart guide

```
curl \
  -H "Content-Type: application/json" \
  -d "{\"contents\":[{\"parts\":[{\"text\":\"Explain how AI works\"}]}]}" \
  -X POST "https://generativelanguage.googleapis.com/v1beta/models/gemini-1.5-flash-latest:generateContent?key=YOUR_API_KEY"
```

Your API keys are listed below. You can also view and manage your project and API keys in Google Cloud.

Project number	Project name	API key	Created	Plan
...4521	Generative Language Client	...JMYE	Oct 17, 2024	Paid Go to billing View usage data

Please note we need a project name from google cloud to create a key if you click on create API key it will show a prompt as below



```
curl \
  -H "Cont
  -d "{\"
  -X POST
key=YOUR_
```

Create API key

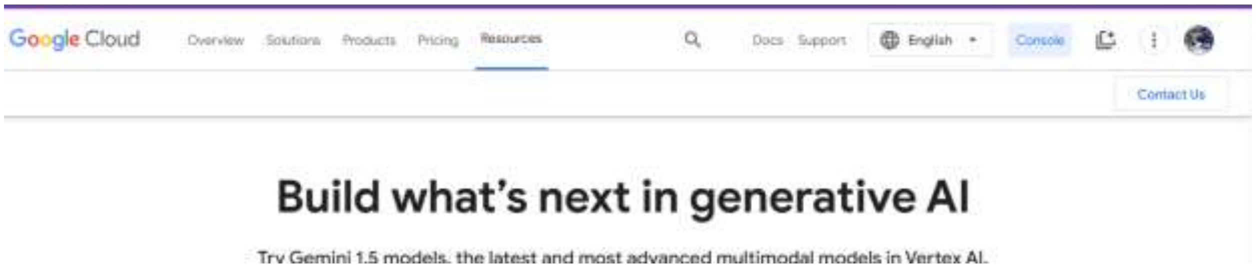
Select a project from your existing Google Cloud projects

Search Google Cloud projects

Generative Language Client gen-lang-client-0403832128

di-bp-ahyi di-bp-ahyi

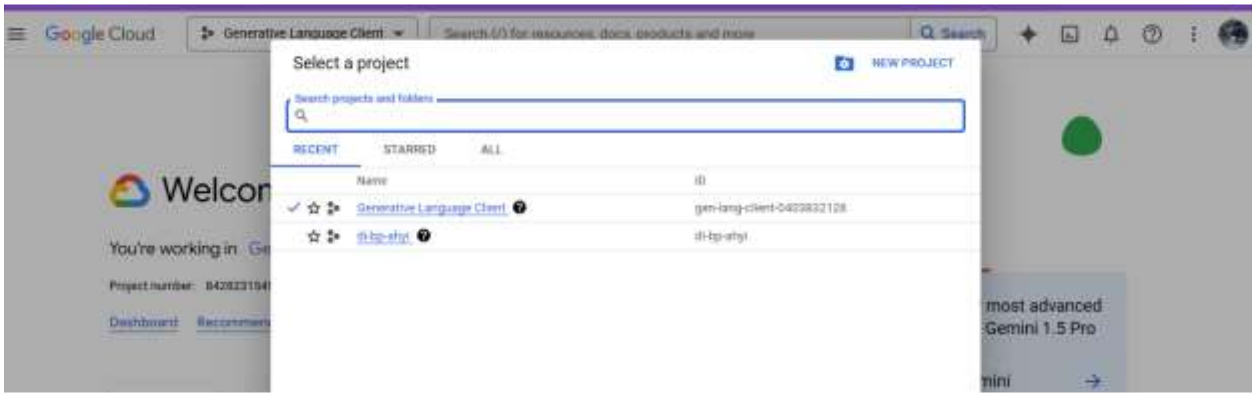
Here we need to go to google cloud page and click on console



Build what's next in generative AI

Try Gemini 1.5 models, the latest and most advanced multimodal models in Vertex AI.

Then we will find a window were we can create a new project. Just give a name so that it reflects on the place while we create a api key.



Select a project — NEW PROJECT

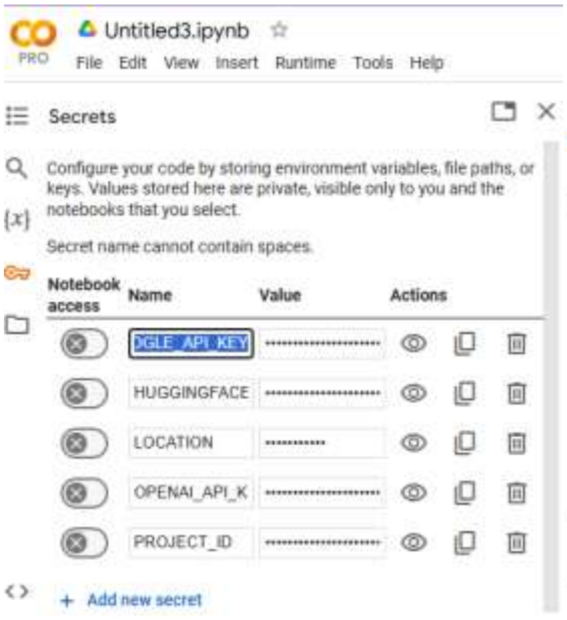
Search projects and folders

RECENT — STARRED — ALL

Name	ID
Generative Language Client	gen-lang-client-0403832128
di-bp-ahyi	di-bp-ahyi

Next what we want is to communicate with the GEMINI models using python with the help of API Key which we have created above.

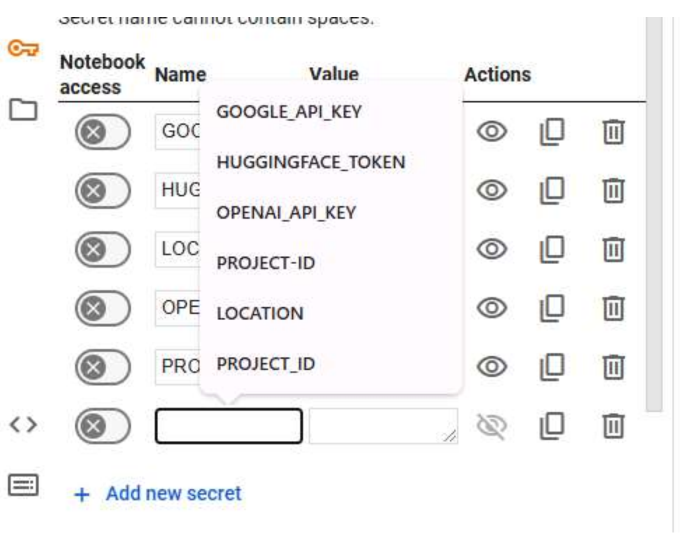
For that lets put the key here on google colab by clicking Add new Secret button and give the name and value.



Configure your code by storing environment variables, file paths, or keys. Values stored here are private, visible only to you and the notebooks that you select.

Secret name cannot contain spaces.

Notebook access	Name	Value	Actions
	OGLE_API_KEY	••••••••••••••••••••	
	HUGGINGFACE	••••••••••••••••••••	
	LOCATION	••••••••••	
	OPENAI_API_K	••••••••••••••••••••	
	PROJECT_ID	••••••••••••••••••••	



Secret name cannot contain spaces.

Notebook access	Name	Value	Actions
	GOC		
	HUG		
	LOC		
	OPE		
	PRO		

GOOGLE_API_KEY
HUGGINGFACE_TOKEN
OPENAI_API_KEY
PROJECT-ID
LOCATION
PROJECT_ID

Our next step is installing the Python SDK for the Google Gemini which is contained in this particular package. ie Google generative AI.

```
!pip install -q -U google-generativeai
```

we will go ahead and install this package.
Now this package is installed.
Along with this we are going to import some more important libraries ...

```
import pathlib
import textwrap
import google.generativeai as genai
from IPython.display import display
from IPython.display import Markdown
def to_markdown(text):
  text = text.replace("•", "  *")
  return Markdown(textwrap.indent(text, "> ", predicate=lambda _: True))
```

we are going to import google. generativeai as genai which will be responsible in calling various Gemini models. Along with this we will be using IPython.display for displaying the entire response that we will be getting from the Gemini models. Then another function that is markdown. Also We are going to replace all . with * . Then we are going to display the entire text over here.

In order to call our API key, there are two ways. First of all, we need to set your environment variable. In this case for Google Colab we have already created our secret key so lets call that

```
# Used to securely store your API key
from google.colab import userdata
```

once we import this, the user data will be responsible in calling that particular API.

```
# Or use `os.getenv('GOOGLE_API_KEY')` to fetch an environment variable.
GOOGLE_API_KEY = userdata.get("GOOGLE_API_KEY")
genai.configure(api_key=GOOGLE_API_KEY)
```

This in short, we will be calling this particular API key with this value. Once we get it, we will be storing this in GOOGLE_API_KEY. After that we have to use this genai.configure() function to configure .

Next we want to know which are the list of models? For that run the below lines of code.

```
for m in genai.list_models():
  if "generateContent" in m.supported_generation_methods:
    print(m.name)
```

Here in most of our projects we will be exploring with all these models.
Just try these lines of code

```
model = genai.GenerativeModel("gemini-1.5-flash")
```

```
%%time
response = model.generate_content("What is the meaning of life?")
```

We can see inside the model there is a function which is called as generate _content. By this we will be able to take some input in the form of text and generate whatever content that we really want. once we execute the above lines of code we will be able to see the response.

Lets go ahead and display the response using the below line of code.

```
to_markdown(response.text)
```

Please note if the question we ask is against humanity like may be how to rob a bank or kill some one or how to insult someone then the API will fail to return a result, This is because of safety reasons. We can use the following - GenerateContentResponse.prompt_feedback
to see why it was blocked ?? (In our case as we wont be able to see any output as we got the response so its not blocked..)

```
response.prompt_feedback
```

to check the setting we can give the command

```
model
```

(Sorry it was showing empty.. I think we need to set it)

```
genai.GenerativeModel(
  model_name='models/gemini-1.5-flash',
  generation_config={},
  safety_settings={},
  tools=None,
  system_instruction=None,
  cached_content=None
)
```

Gemini can generate multiple possible responses for a single prompt. These possible responses are called candidates, we can review them to select the most suitable one as the response.

View the response candidates with GenerateContentResponse.candidates:

```
response.candidates
```

By default, the model returns a response after completing the entire generation process. We can also stream the response as it is being generated, and the model will return chunks of the response as soon as they are generated.

To stream responses, use GenerativeModel.generate_content(..., stream=True).

```
%%time
response = model.generate_content("What is the meaning of life?", stream=True)
```

now lets display the above response chunk by chunk

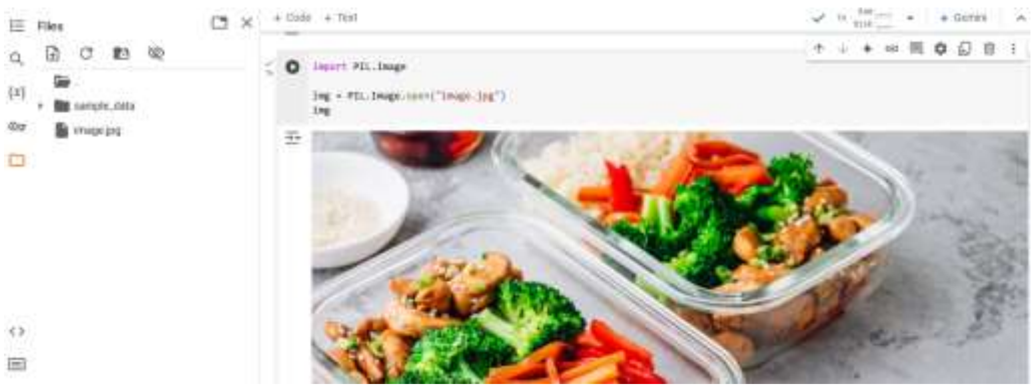
```
for chunk in response:
  print(chunk.text)
  print("_" * 80)
```

Next if we want to Generate text from image and text inputs
Please note the GenerativeModel.generate_content API is designed to handle multimodal prompts and returns a text output. So Let's include an image using the below command

```
!curl -o image.jpg https://t0.gstatic.com/licensed-image?q=tbn:ANd9GcQ_Kevbk21QBRy-PgB4kQpS79brbmmEG7m3VOTShAn4PecDU5H5UxrJxE3Dw1JiaG17V88QIol19-3TM2wCHw
```

first of all, what we are doing by using this curl command is we are downloading the images. Once the image is downloaded we will go ahead and display this image.

```
import PIL.Image
img = PIL.Image.open("image.jpg")
img
```



```
import PIL.Image
img = PIL.Image.open("image.jpg")
img
```

Now in our local there will be something called as image.jpg. Next We're just going to call this and we will try to display it. Lets use the **gemini-1.5-flash** model and pass the image to the model with generate_content. And again for this we are importing plot image okay.

```
model = genai.GenerativeModel("gemini-1.5-flash")
```

So here you can see this is my entire image okay guys.

```
response = model.generate_content(
    [
        "Write a short, engaging blog post based on this picture. It should include a description of the meal in the photo and talk about my journey meal prepping.",
        img,
    ],
    stream=True,
)
response.resolve()
```

We can see how accurately the model was able to detect each and every object. And it was able to give you the answer.

```
to_markdown(response.text)
```