

# OPEN AI x ChatGPT x LineBot 串接完整全紀錄



人氣格鬥士 吳佳鑫

2022/12/17 • 閱讀時間約 18 分鐘



程式原碼: <https://github.com/JokerWuXin/ChatGpt-LineBot>

步驟:

S1.至OPEN AI 取得 API KEY

S2.至LineBot 取得 Channel access token 、Channel secret 並 進行相關設定

S3.至Vercel 申請免費伺服器並連結 GitHub原碼

S4.GitHub上傳原碼(index.py 、chatgpt.py 、prompt.py 、requirements.txt 、vercel.json)

S5.至Vercel進行伺服器設定(此處主要為原碼與環境變數，也就是將S1 & S2 的api key 與 權杖 進行設定，取得進行 Webhook用的網址)，成功後可看到 Hello World 字眼

S6.至LineBot 進行 WebHook設定，完成後即可在Line上進行測試

詳細步驟流程:

## 取得OPEN API KEY

首先至OPEN AI 取得 API KEY，詳細可參閱 [OPENAI 聊天機器人](#) 文章

## LineBot Channel access token 、Channel secret

取得 LineBot Channel access token 、Channel secret 後 至 [Line BOT 控制台](#) 申請 LineBot Key

申請方式，可參照 [LineBOt 建置影片](#)

其中要注意，此處為

Allow bot to join group chats <sup>?</sup>	Enabled
Auto-reply messages <sup>?</sup>	Disabled
Greeting messages <sup>?</sup>	Enabled

點擊後方 Edit 可進到中文頁面修改

### 回應設定

可配合帳號的經營目的，設定聊天及自動訊息的回應方式。

#### 回應功能

聊天

☐

可透過聊天與好友互動。

加入好友的歡迎訊息

☒

當用戶將本帳號加為好友時，可自動傳送訊息內容。  
[開啟「加入好友的歡迎訊息」設定頁面](#)

Webhook

☒

當用戶傳送訊息給本帳號或將本帳號加為好友時，從LINE平台傳送Webhook事件至Webhook網址。  
[開啟Messaging API的設定頁面](#)

自動回應訊息

☐

可使用事先設定好的訊息內容進行自動回覆。  
[開啟自動回應訊息的設定頁面](#)

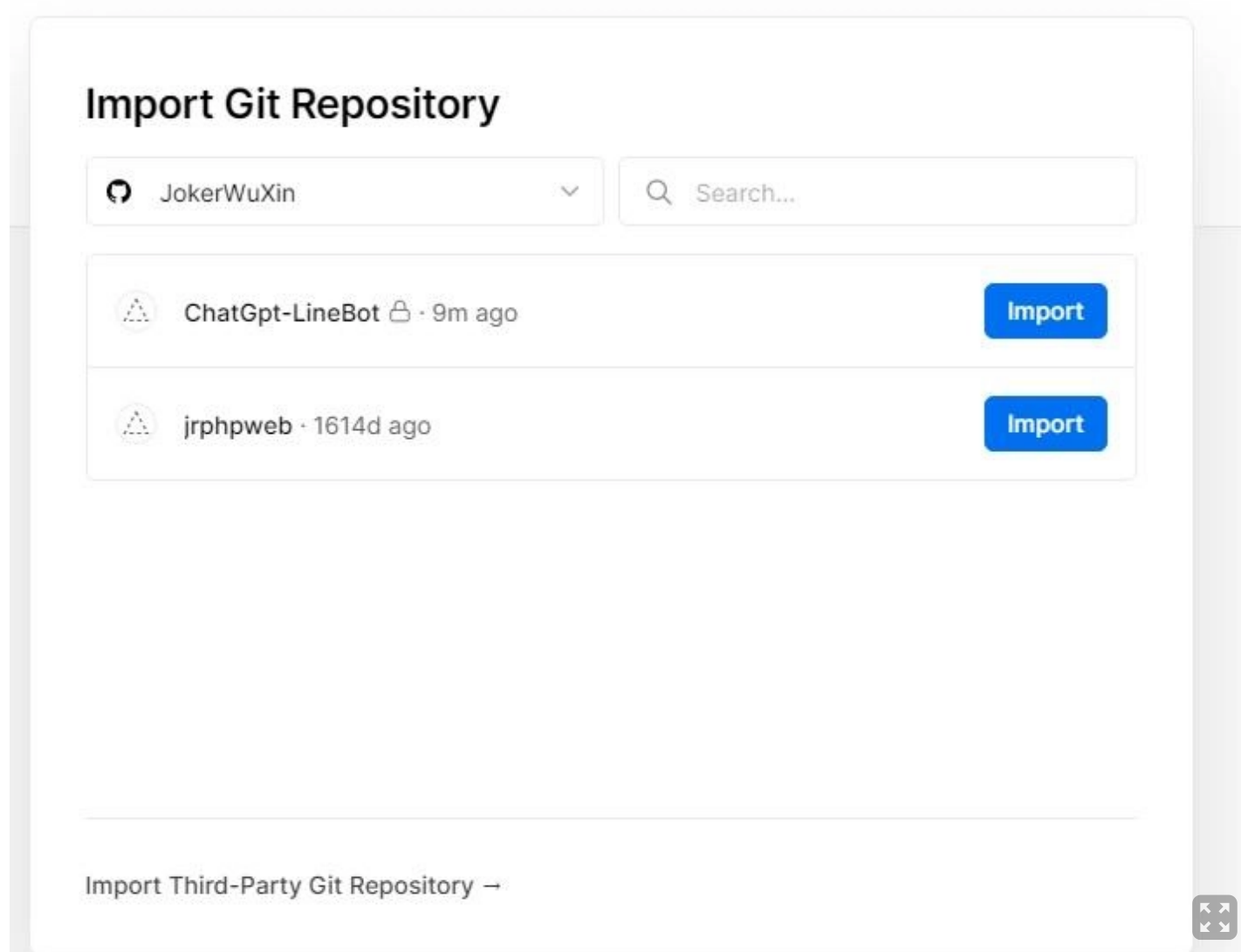
設定完成後 至 [vercel](#) 申請免費伺服器

## Vercel 匯入 GitHub 原碼

申請完成後，匯入 GitHub 原碼

# Let's build something new.

To deploy a new Project, import an existing Git Repository or get started with one of our Templates.



若沒 GitHub 則需至 [GitHub官網](#)註冊

GitHub上傳原碼 可在[官網下載工具使用](#)

## 原碼部分

index.py 可參考 [LINE Messaging API SDK for Python](#)

```

from flask import Flask, request, abort
from linebot import LineBotApi, WebhookHandler
from linebot.exceptions import InvalidSignatureError
from linebot.models import MessageEvent, TextMessage, TextSendMessage
from api.chatgpt import ChatGPT

import os

line_bot_api = LineBotApi(os.getenv("LINE_CHANNEL_ACCESS_TOKEN"))
line_handler = WebhookHandler(os.getenv("LINE_CHANNEL_SECRET"))

working_status = os.getenv("DEFALUT_TALKING", default = "true").lower() == "true"

app = Flask(__name__)
chatgpt = ChatGPT()

# domain root
@app.route('/')
def home():
    return 'Hello, World!'

@app.route("/webhook", methods=['POST'])
def callback():
    # get X-Line-Signature header value
    signature = request.headers['X-Line-Signature']

    # get request body as text
    body = request.get_data(as_text=True)
    app.logger.info("Request body: " + body)

    # handle webhook body
    try:
        line_handler.handle(body, signature)
    except InvalidSignatureError:
        abort(400)

    return 'OK'

@line_handler.add(MessageEvent, message=TextMessage)
def handle_message(event):
    global working_status

    if event.message.type != "text":
        return

    working_status = True

    if working_status:
        chatgpt.add_msg(f"Human:{event.message.text}?\n")
        reply_msg = chatgpt.get_response().replace("AI:", "", 1)
        chatgpt.add_msg(f"AI:{reply_msg}\n")
        line_bot_api.reply_message(
            event.reply_token,
            TextSendMessage(text=reply_msg))

if __name__ == "__main__":
    app.run()

```

chatgpt.py 可參考 [OPEN AI 聊天機器人Chat範例](#)

```
from api.prompt import Prompt
import os
import openai

openai.api_key = os.getenv("OPENAI_API_KEY")

class ChatGPT:
    def __init__(self):
        self.prompt = Prompt()
        self.model = os.getenv("OPENAI_MODEL", default = "text-davinci-003")
        self.temperature = float(os.getenv("OPENAI_TEMPERATURE", default = 0))
        self.frequency_penalty = float(os.getenv("OPENAI_FREQUENCY_PENALTY", default = 0))
        self.presence_penalty = float(os.getenv("OPENAI_PRESENCE_PENALTY", default = 0.6))
        self.max_tokens = int(os.getenv("OPENAI_MAX_TOKENS", default = 240))

    def get_response(self):
        response = openai.Completion.create(
            model=self.model,
            prompt=self.prompt.generate_prompt(),
            temperature=self.temperature,
            frequency_penalty=self.frequency_penalty,
            presence_penalty=self.presence_penalty,
            max_tokens=self.max_tokens
        )
        return response['choices'][0]['text'].strip()

    def add_msg(self, text):
        self.prompt.add_msg(text)
```

prompt.py 可參考 vercel 官網提供的 [python flask](#) 範例

```
import os

chat_language = os.getenv("INIT_LANGUAGE", default = "zh")

MSG_LIST_LIMIT = int(os.getenv("MSG_LIST_LIMIT", default = 20))

LANGUAGE_TABLE = {
    "zh": "嗨！",
    "en": "Hi!"
}

class Prompt:
    def __init__(self):
        self.msg_list = []

        self.msg_list.append(f"AI:{LANGUAGE_TABLE[chat_language]}")

    def add_msg(self, new_msg):
        if len(self.msg_list) >= MSG_LIST_LIMIT:
            self.remove_msg()

        self.msg_list.append(new_msg)

    def remove_msg(self):
        self.msg_list.pop(0)

    def generate_prompt(self):
        return '\n'.join(self.msg_list)
```

此外，根據vercel官網提供的 python flask建置，還須注意

Python Dependencies

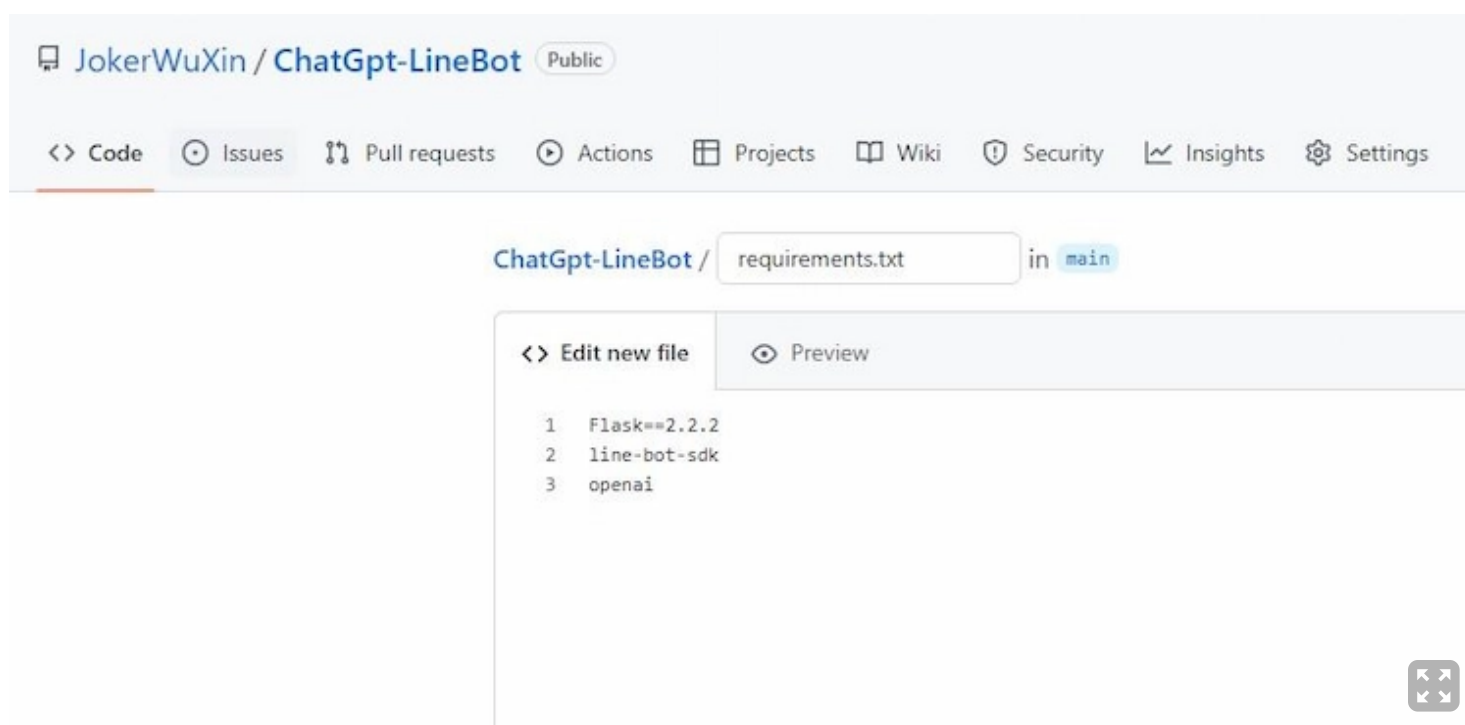
You can install dependencies for your Python projects by defining them in requirements.txt or a Pipfile with corresponding Pipfile.lock.

requirements.txt

```
Flask==2.2.2
```

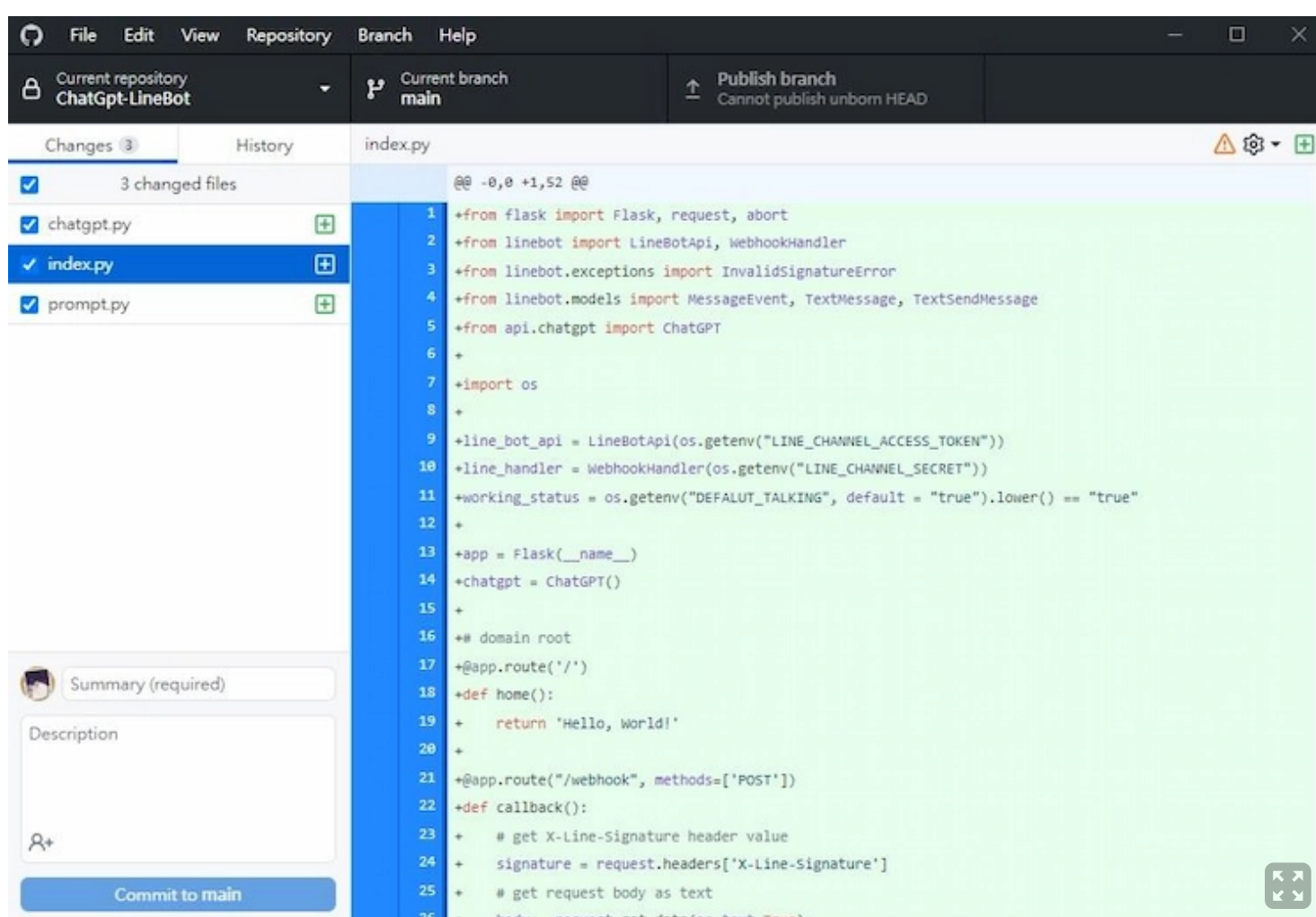
這件事，將版本與相關的SDK建立 requirements.txt

```
Flask==2.2.2
line-bot-sdk
openai
```

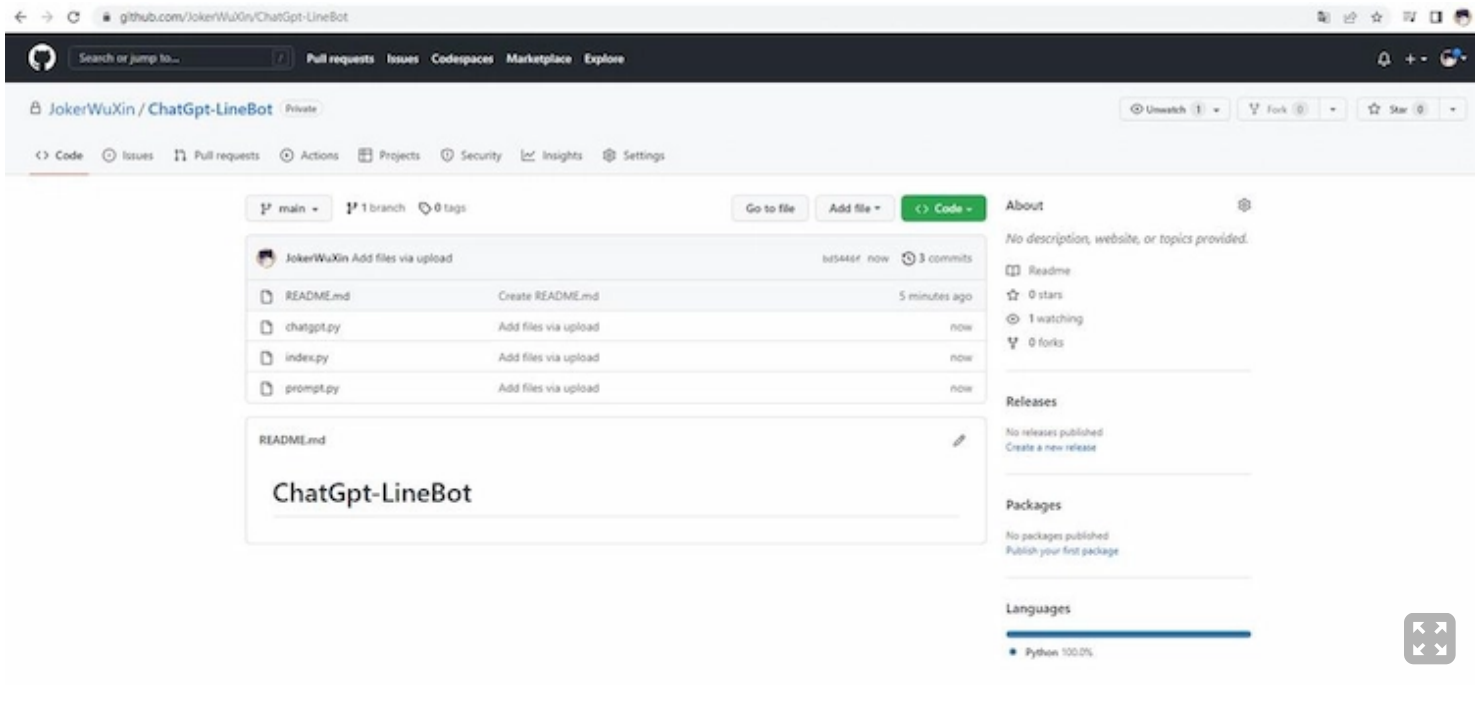


還有，vercel.json 用來，指定路徑進入點的

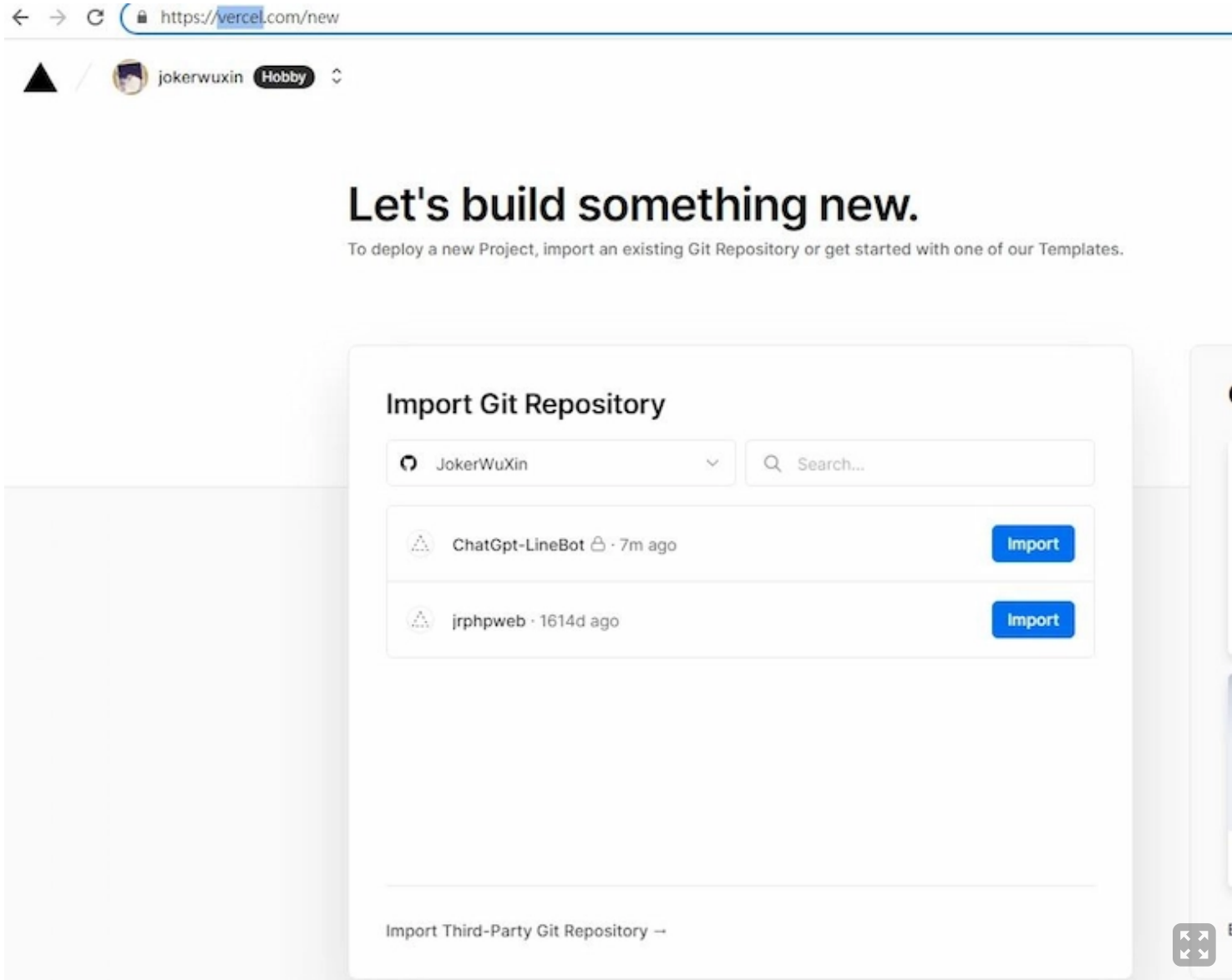
```
{
  "builds": [
    {
      "src": "api/index.py",
      "use": "@vercel/python"
    }
  ],
  "routes": [
    {
      "src": "/(.*)",
      "dest": "api/index.py"
    }
  ]
}
```



完成後，透過 GitHub 工具上傳

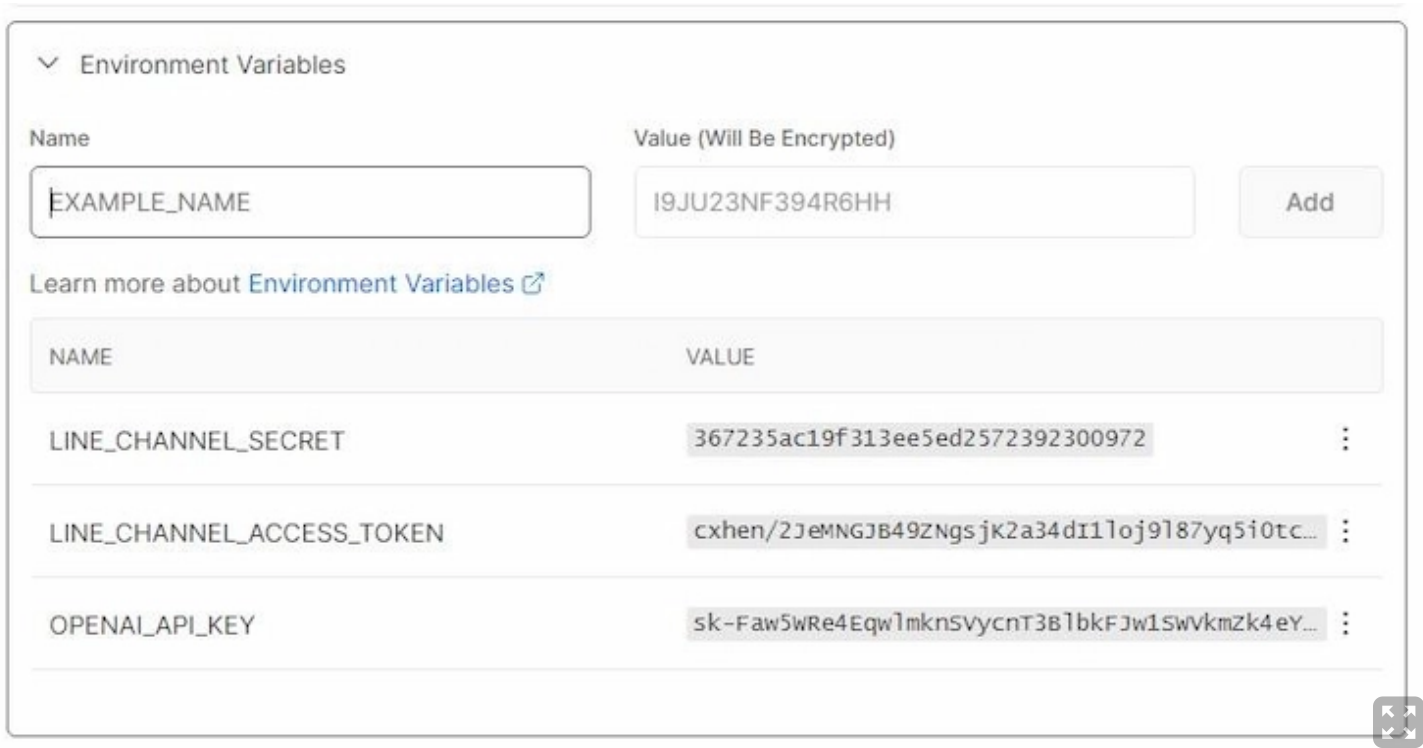
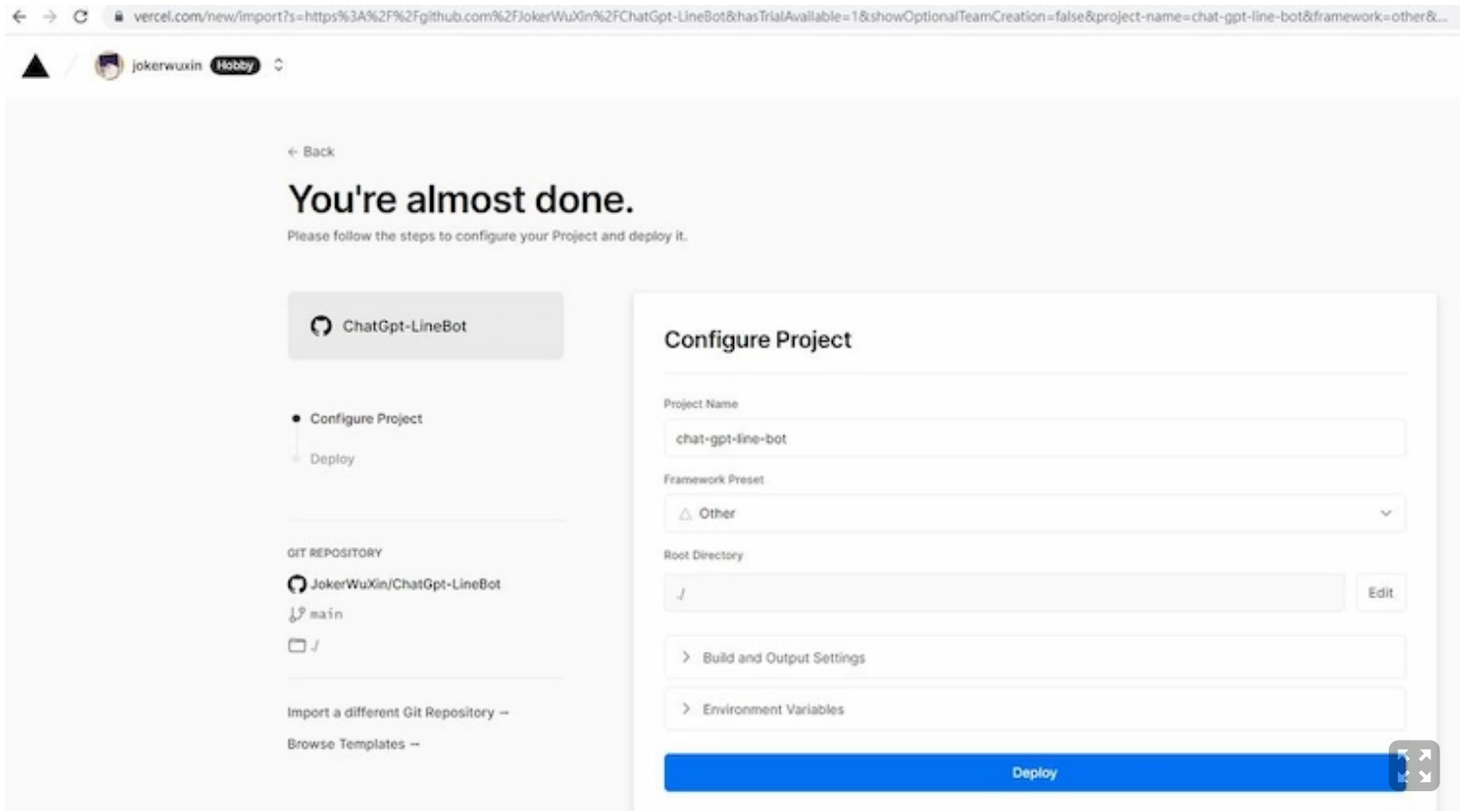


上傳完成後，返回 vercel 進行 import

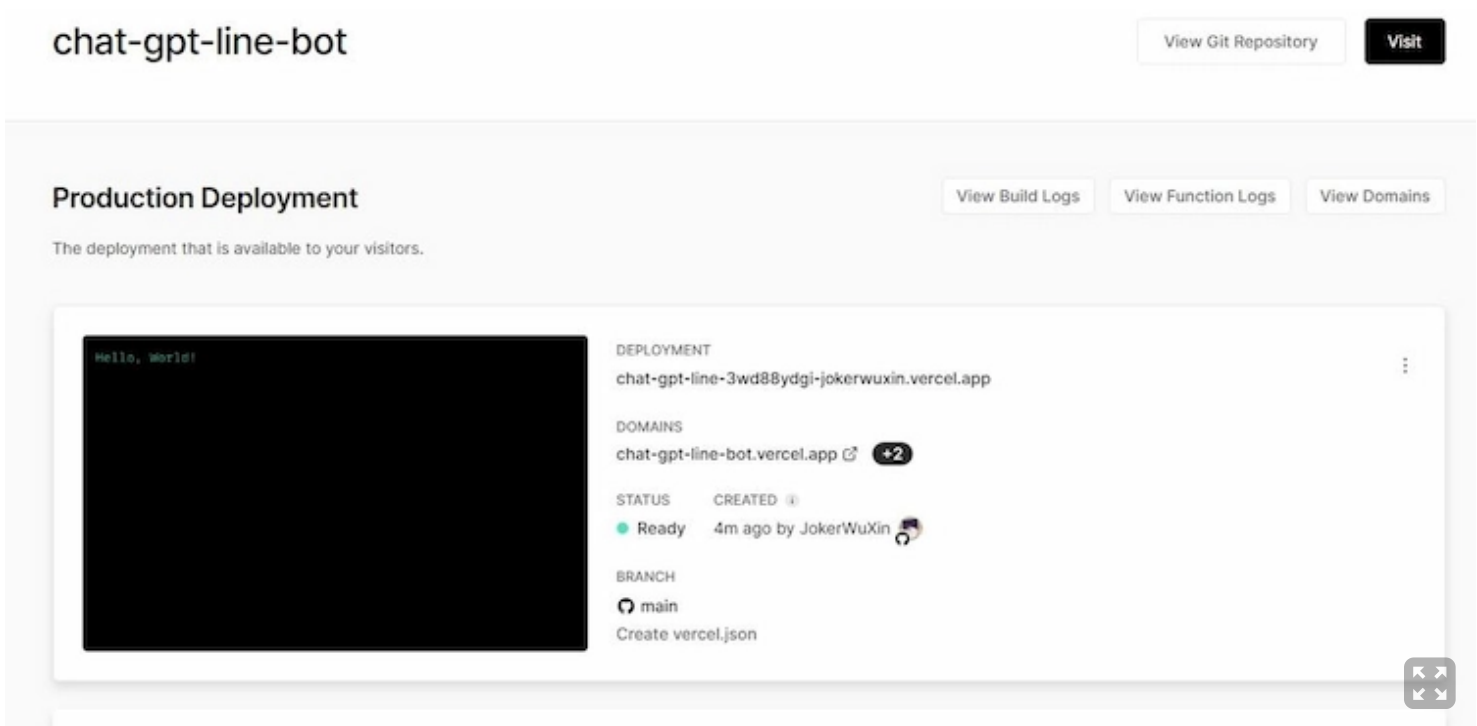




# Vercel Environment Variables 環境設定

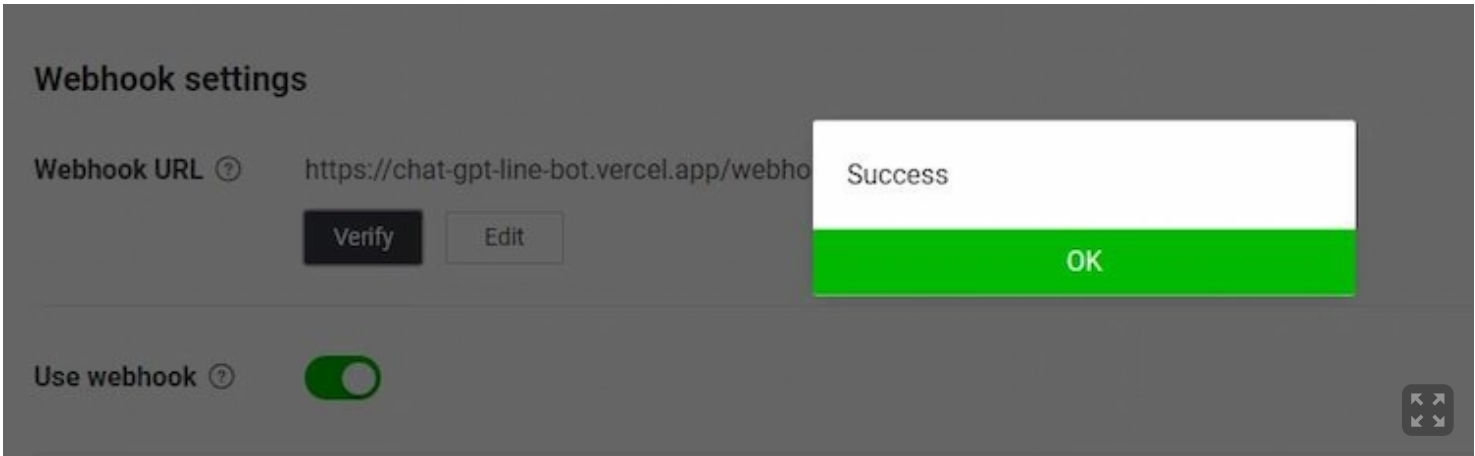


點擊 Environment Variables 進行環境設定，將OPEN AI & LineBot API KEY 進行輸入設定，完成後 點擊 Deploy



# LineBot Webhook 設定

完成後，複製 **DOMAINS** 路徑，返回 LineBot後台

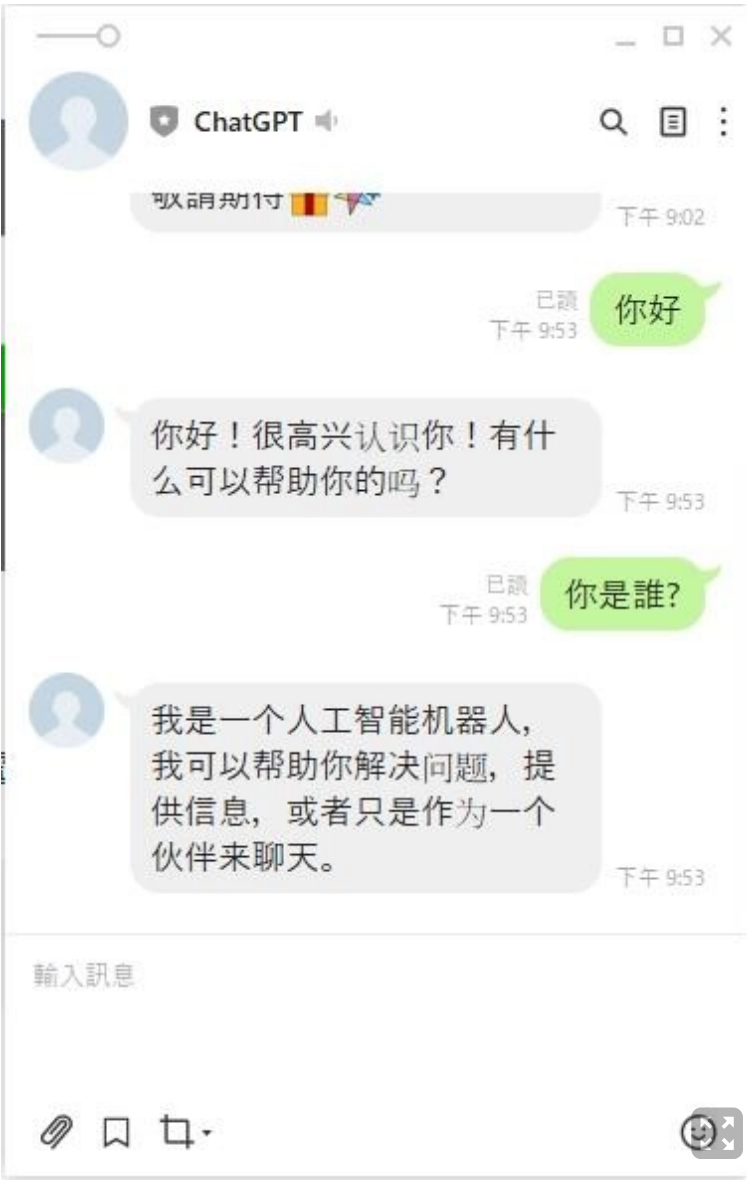


將路徑貼上，並在路徑後方加上 webhook，原因是我們在 index.py處

```
@app.route("/webhook", methods=['POST'])
```

將路徑引到此處與LineBot透過post方式傳輸資料

至此就可以進行測試囉~



# QA

## chatgpt 與 openai gpt3 有何差異

 chatgpt 與 openai gpt3.0 有何差異 



CHATGPT 是由 OpenAI 开发的一种语言模型，它被训练来预测用户输入之后的可能输出。它   可以被用来生成自然语言文本，例如聊天机器人的回复或文章的自动生成。

OpenAI GPT-3 是一种大型语言模型，也是由 OpenAI 开发的。它与 CHATGPT 相似，也可以用来生成自然语言文本，但它的能力要比 CHATGPT 更强大。GPT-3 可以被用来做很多不同的任务，包括文本生成、翻译、问答等等。它也可以用来解决许多不同的 NLP（自然语言处理）任务，例如语音识别、文本分类、情感分析等等。GPT-3 是目前为止最大的语言模型之一，具有非常强大的语言生成能力。



## chatgpt 有api嗎

 chatgpt 有api嗎 



是的，CHATGPT 有 API（应用程序编程接口）可供开发人员使用。通过 API，开发人员可以   使用 CHATGPT 的功能，并将其集成到自己的应用程序中。

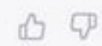
要使用 CHATGPT 的 API，您需要注册 OpenAI 的 API 服务，然后获取 API 密钥。您还需要安装 OpenAI 的 Python 库，这样就可以在 Python 中调用 CHATGPT 的 API 了。

CHATGPT 的 API 与其他常见的 API 类似，可以通过发送 HTTP 请求来访问 CHATGPT 的功能。您可以通过 API 调用来生成自然语言文本，例如聊天机器人的回复或文章的自动生成。您也可以使用 API 来训练 CHATGPT 来执行特定的任务，例如情感分析或文本分类。





如果您想要使用 CHATGPT 的 API，可以通过以下步骤来开始：



1. 注册 OpenAI 的 API 服务，并获取 API 密钥。这样您就可以访问 OpenAI 的 API 了。
2. 安装 OpenAI 的 Python 库。您可以通过运行以下命令来安装库：

```
pip install openai
```

3. 使用 API 密钥和 Python 库来访问 CHATGPT 的功能。您可以使用以下代码来调用 CHATGPT 的 API：

```
import openai
```

```
openai.api_key = "YOUR_API_KEY"
```

```
response = openai.Completion.create(  
engine="chatbot",  
prompt="Hello, how are you today?",  
max_tokens=20  
)
```

```
print(response.text)
```

在这个例子中，我们使用 API 密钥和 Python 库来访问 CHATGPT 的 API。我们使用 Completion.create() 方法来生成一个自然语言文本，并将文本作为响应返回。

有关 CHATGPT 的 API 的更多信息，可以参考 OpenAI 的 API 文档。



## 使用ChatGPT API

若要使用 ChatGPT 作為 Line機器人的串接API

則須將 chatgpt.py 的

```
def get_response(self):  
    response = openai.Completion.create(  
        model=self.model,  
        prompt=self.prompt.generate_prompt(),  
        temperature=self.temperature,  
        frequency_penalty=self.frequency_penalty,  
        presence_penalty=self.presence_penalty,  
        max_tokens=self.max_tokens  
    )  
    return response['choices'][0]['text'].strip()
```

透過ChatGPT說的進行合併修改

```
import openai

openai.api_key = "YOUR_API_KEY"

response = openai.Completion.create(
    engine="chatbot",
    prompt="Hello, how are you today?",
    max_tokens=20
)

print(response.text)
```

不過經過測試，這裡有點有趣



如果您想要使用 CHATGPT 的 API 进行自然语言生成，可以将代码中的 `self.model` 设置为 `"chatbot"`。这样就可以使用 CHATGPT 进行自然语言生成了。

此外，您还可以使用其他参数来控制 CHATGPT 生成的文本。例如，您可以使用 `temperature` 参数来控制文本的创造性和随机性。您也可以使用 `frequency_penalty` 和 `presence_penalty` 参数来控制文本的多样性。

最后，您还可以使用 `max_tokens` 参数来限制生成的文本的最大长度。

具体来说，可以将代码改为如下所示：

```
def init(self):
    self.prompt = Prompt()
    self.model = "chatbot"
    self.temperature = float(os.getenv("OPENAI_TEMPER
```






依上述所言，只要改 `model` 就應該可以

```
def __init__(self):
    self.prompt = Prompt()
    self.model = "chatbot"

    self.temperature = float(os.getenv("OPENAI_TEMPERATURE", default = 0))
    self.frequency_penalty = float(os.getenv("OPENAI_FREQUENCY_PENALTY", default = 0))
    self.presence_penalty = float(os.getenv("OPENAI_PRESENCE_PENALTY", default = 0.6))
    self.max_tokens = int(os.getenv("OPENAI_MAX_TOKENS", default = 240))

    def get_response(self):
        response = openai.Completion.create(
            model=self.model,
            prompt=self.prompt.generate_prompt(),
            temperature=self.temperature,
            frequency_penalty=self.frequency_penalty,
            presence_penalty=self.presence_penalty,
            max_tokens=self.max_tokens
        )
        return response['choices'][0]['text'].strip()
```







通过您提供的代码，可以看出您已经成功地使用了 CHATGPT 的 API 进行自然语言生成。


在调用 `openai.Completion.create()` 方法时，您已经指定了模型引擎为 "chatbot"，因此会使用 CHATGPT 生成文本。

此外，您还使用了 `temperature`、`frequency_penalty`、`presence_penalty` 和 `max_tokens` 参数来控制 CHATGPT 生成的文本。

在返回文本时，您使用了 `response['choices'][0]['text'].strip()` 语句，这是正确的。这个语句会访问 `response` 对象的 `choices` 列表的第一个元素（即第一个 `Choice` 对象），并返回其 `text` 属性的值。

综上，您的代码可以正常使用 CHATGPT 的 API 进行自然语言生成。如果您还有其他问题，请随时联系我。





But



所以

```
import openai

def generate_text():
    openai.api_key = os.environ["OPENAI_API_KEY"]
    prompt = "What is the weather like today?"
    model = "chatbot"
    temperature = 0.7
    frequency_penalty = 0
    presence_penalty = 0.6
    max_tokens = 240

    model 就不能使用 "chatbot"
```



您好，根据您提供的代码，您正在使用 OpenAI API 的 Completion 接口来生成文本。

在使用 OpenAI API 时，您需要指定要使用的模型。根据 OpenAI API 的文档，目前可用的模型有：

- text-davinci-002
- text-babbage-001
- text-curie-001
- text-curie-002
- text-curie-003

因此，在您的代码中，`model` 变量应当设置为上述模型之一，例如 "text-curie-002"。

如果您使用的是名为 "chatbot" 的模型，那么会导致错误，因为 OpenAI API 目前并没有提供名为 "chatbot" 的模型。





因此～

根據 [官方技術文件](#)

目前就是只能用 OpenAI GPT-3 玩～

後記:

經過我的騷擾跟測試，這機器人改口了～



笑死~

本紀錄參考:

- Line 官方提供的 python flask 製作 linebot 的 sample code [line/line-bot-sdk-python](#)
- Vercel 官方提供的 python runtime Flask 範例 [Deploy an example with Flask](#)
- [memochou1993/gpt-ai-assistant](#) node.js 版本串接 vercel 範例
- [Lanznx/HealthLineBot](#) python Django 版本串接 vercel 範例
- [howarder3/GPT-Linebot-python-flask-on-vercel](#) python Flask 版本串接 vercel 範例

# OpenAI

# ChatGPT

# python

# LineBot

# Vercel

# API



贊助支持創作者，成為他繼續創作的動力吧！

贊助

[上一篇](#)

[AI 動漫產生器](#)

[下一篇](#)

[Python 基礎](#)



**吳佳鑫**

社會觀察者、社會企業家、教育創新者、遊戲人、媒體人、廚師，小人物與您分享眼見耳聞的生活大小事。 Anything will understand slightly a spot, life more color spots

追蹤

留言 1

[查看全部](#)

張

凱森張 2022/12/24

非常感謝，照著做很順利的成功了！