

Slack API 를 이용한 자동화 업무

Slack 서비스

- Slack은 비즈니스 커뮤니케이션을 위한 클라우드 기반의 팀 협업 도구
- 2013년에 스튜어트 버터필드 등에 의해 출시되었으며, 이메일을 대체할 수 있는 효과적인 대안으로 빠르게 인기를 얻음
- Slack은 개별 회사나 조직의 팀원들이 실시간으로 소통할 수 있는 환경을 제공

Slack is your productivity platform

Slack is a new way to communicate with your team. It's faster, better organized, and more secure than email.

<https://slack.com/>



- (실습) 슬랙 가입
- (실습) 워크 스페이스 및 채널 생성

(다른 사용자 초대 건너뛰기)

- 워크 스페이스 이름 : **python_slack**
- 채널 이름 : **python-slack-test**

슬랙 API 서비스

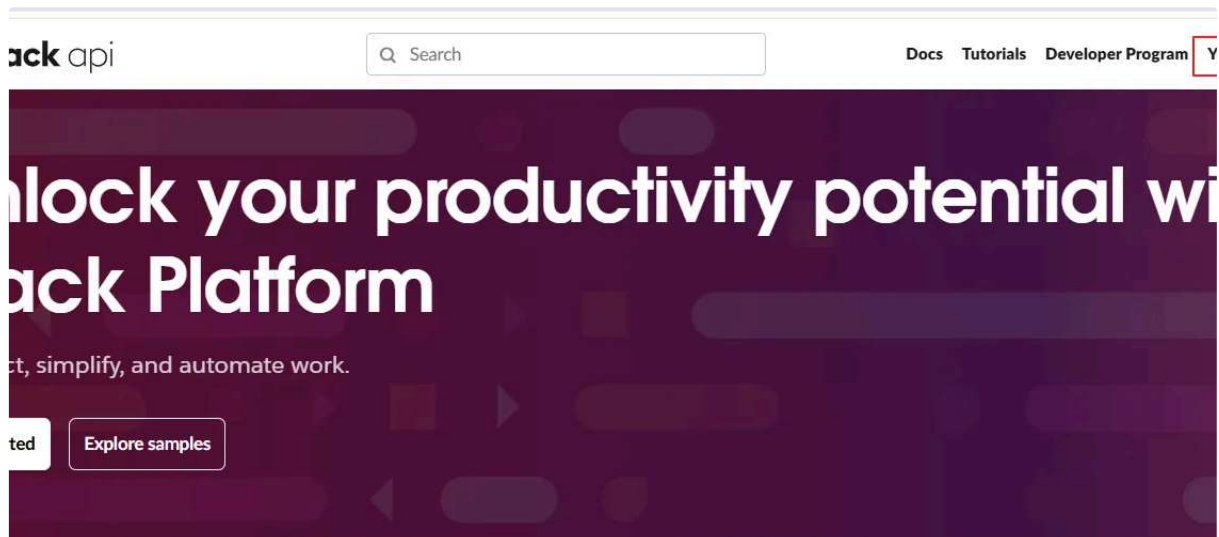
- 슬랙 로그인에 되어 있는 상태면 슬랙 API 서비스에도 자동 로그인

Unlock your productivity potential with Slack Platform

Connect, simplify, and automate work.

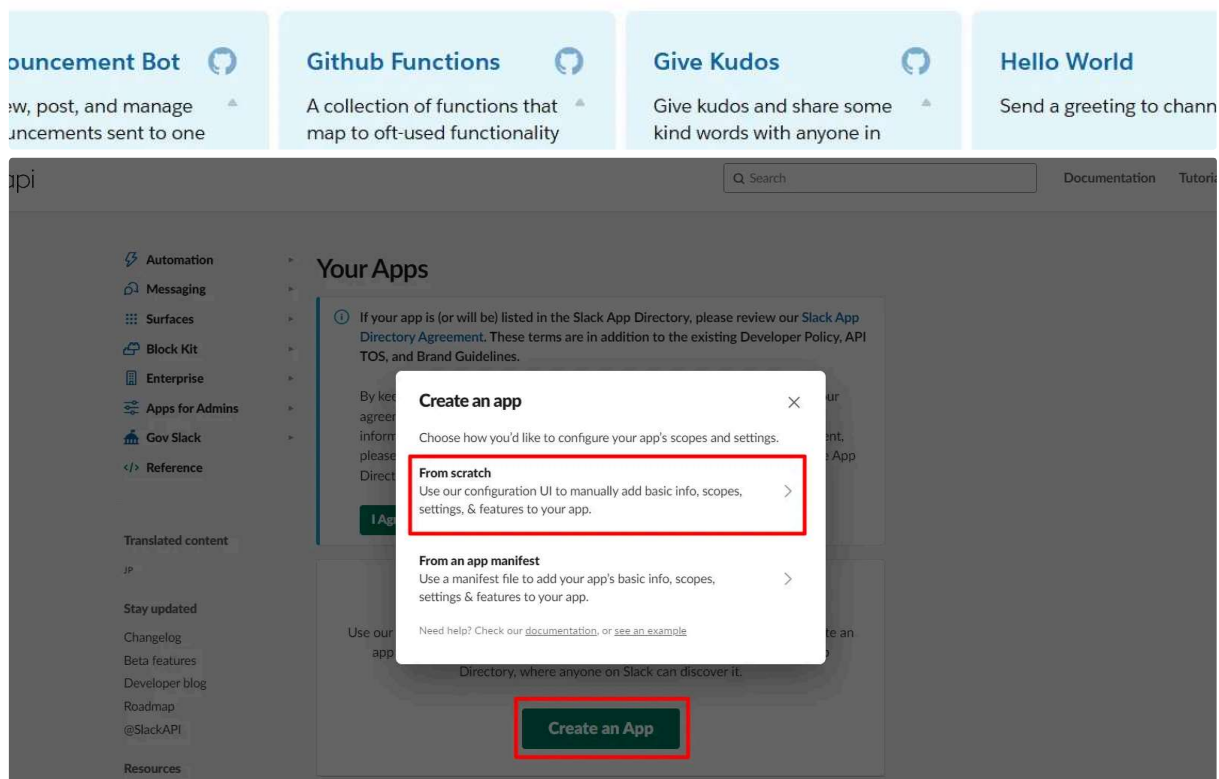
 <https://api.slack.com/>

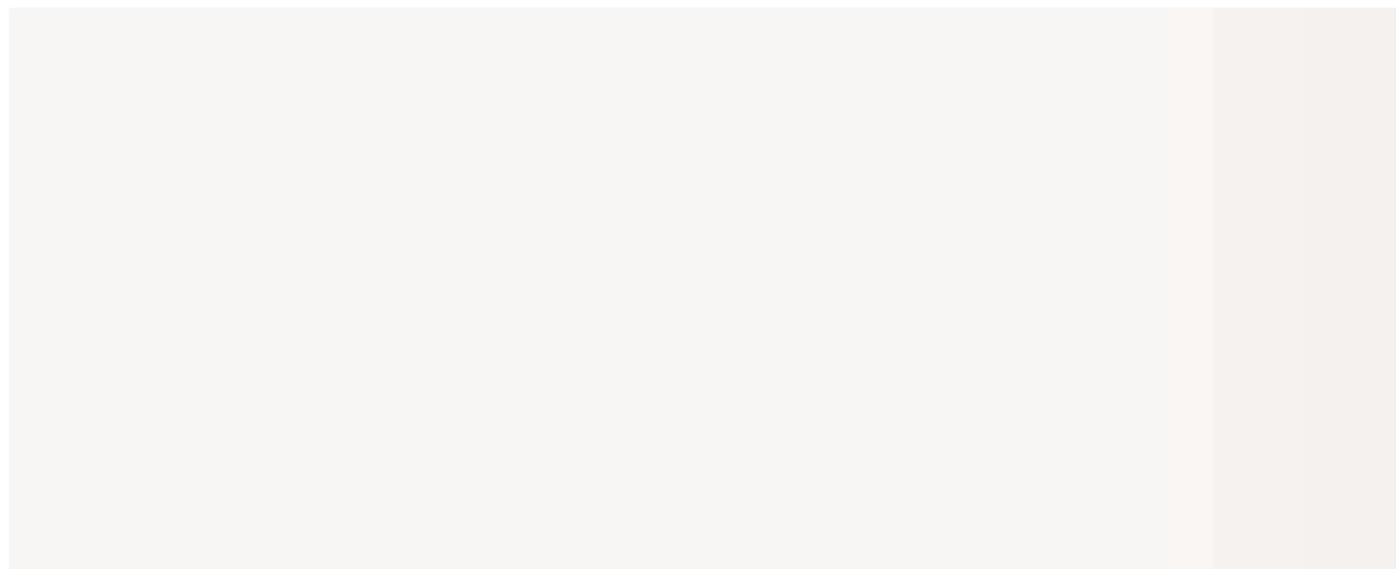
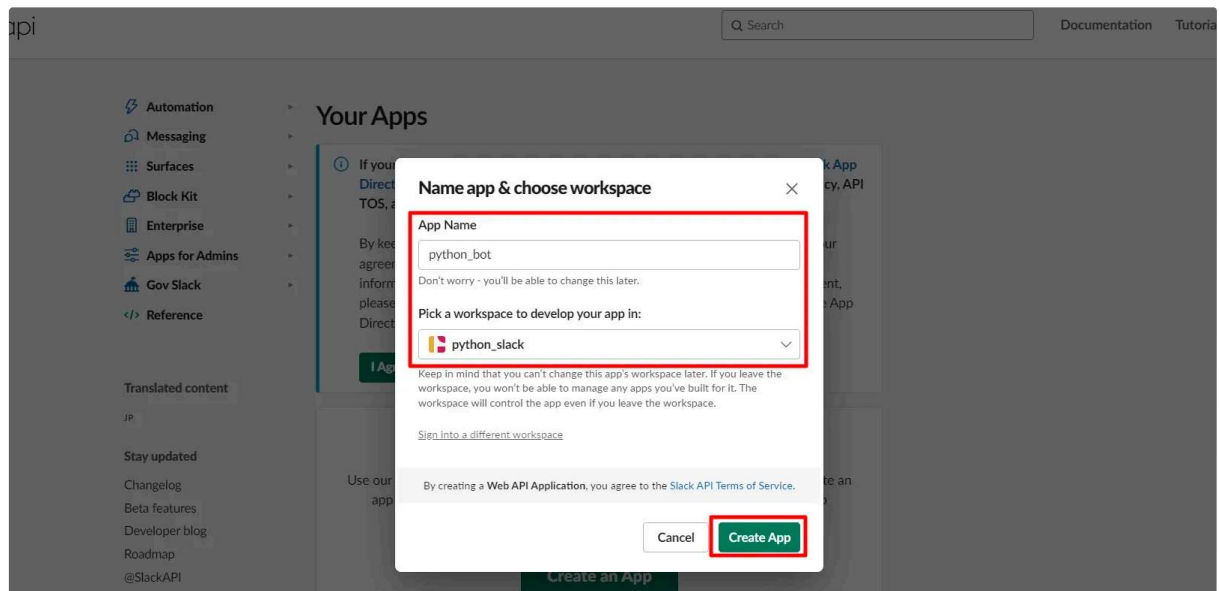




Inspired

Get started with samples and tutorials for common use cases.





이 앱은 python_slack 워크스페이스의 멤버가 만들었습니다.



python_bot에서 python_slack Slack 워크스페이스
에 액세스하기 위해 권한을 요청합니다.

python_bot은(는) 어디에 게시해야 하나요?

python_bot에는 앱으로 게시할 채널이 필요합니다.

python-slack-test

취소

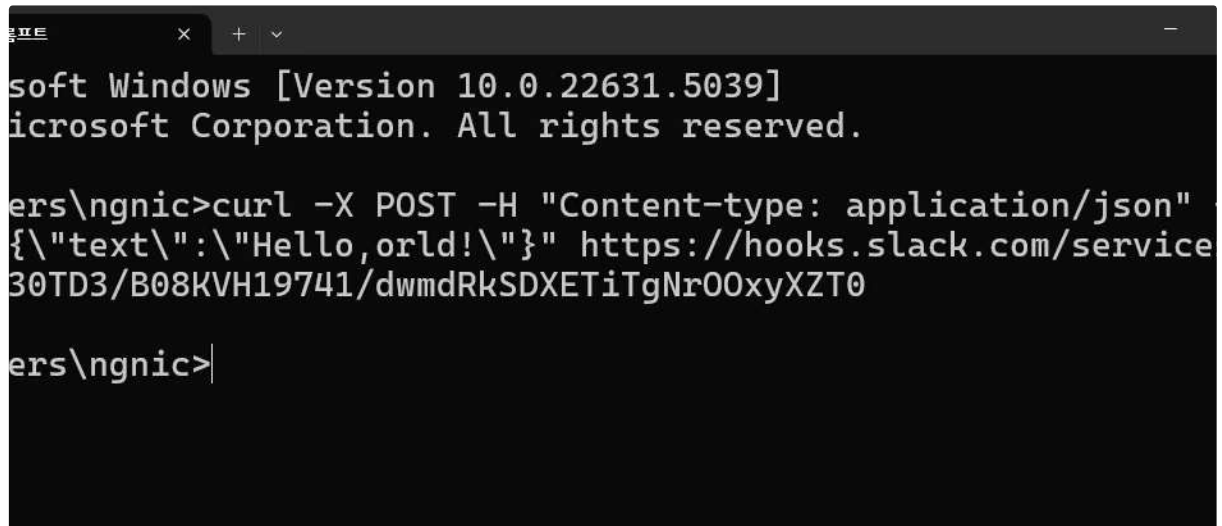
허용

Incoming Webhooks 메시지 전송

윈도우에서 확인

```
curl -X POST -H "Content-type: application/json" --data "{ \"text\": \"Hello,orld!\" }" https://hooks.slack.com/services/T07AXE30TD3/B088VNA1QKD/VaNCswJxtq27MW701D
```

뒤쪽 https... 부분은 여러분의 Slack 키로 복사



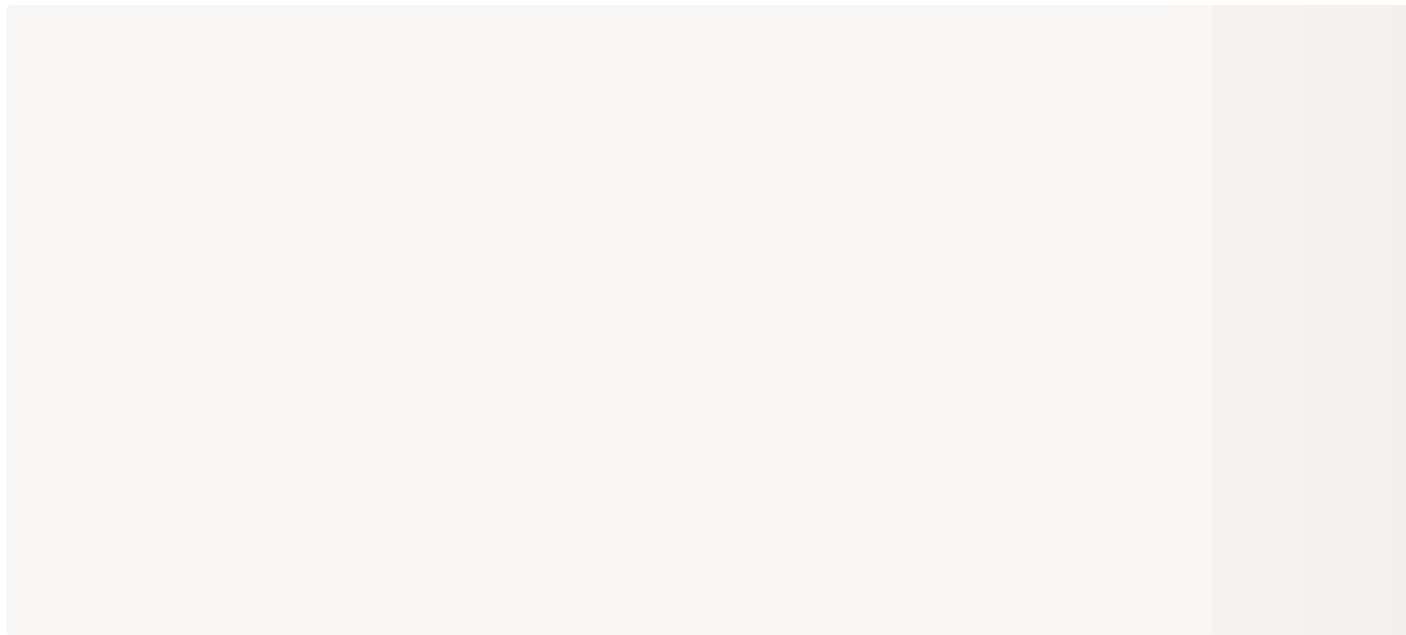
```
Microsoft Windows [Version 10.0.22631.5039]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ngnic>curl -X POST -H "Content-type: application/json"
{"text": "Hello,orld!"} https://hooks.slack.com/service
30TD3/B08KVH19741/dwmdRkSDXETiTgNr00xyXZT0

C:\Users\ngnic>
```

Webhook URL 방법을 이용한 슬랙 채널 메시지 전송

```
import requests slack_url = "Slack Webhook URL" #여러분의 Slack API 주소
def sendSlackWebHook(strText): headers = { "Content-type": "application/json" } data = { "text": strText } res = requests.post(slack_url, headers=headers, json=data) if res.status_code == 200: return "OK" else: return "Error" print(sendSlackWebHook("파이썬 자동화 슬랙 메시지 테스트"))
```



Scopes

A Slack app's capabilities and permissions are governed by the [scopes](#) it requests.

Bot Token Scopes

Scopes that govern what your app can access.

OAuth Scope	Description	
incoming-webhook	Post messages to specific channels in Slack	
Add an OAuth Scope		

Search

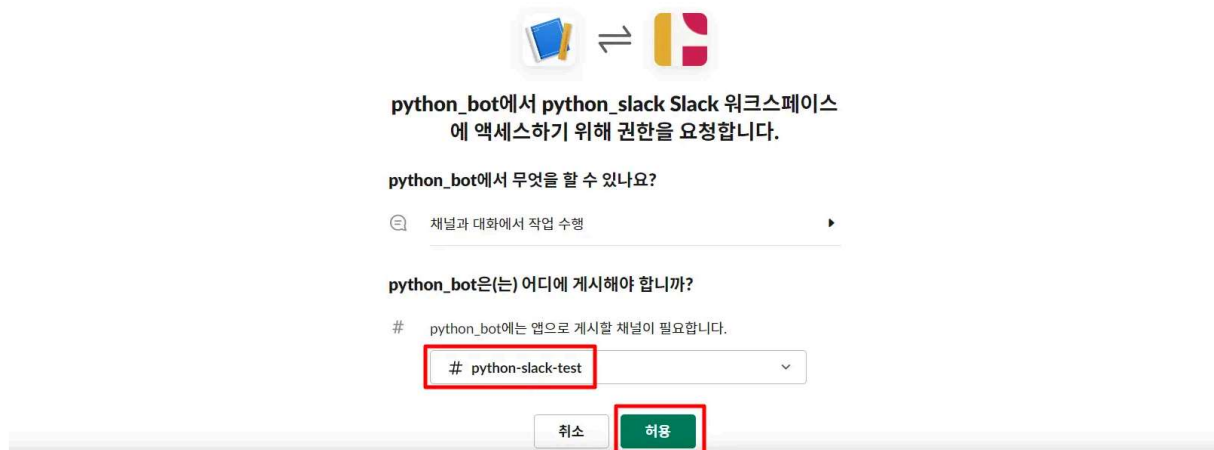
Documentation Tutorials

changed the permission scopes your app uses. Please [reinstall your app](#) for these changes to take effect (and if your app is listed in the Slack App Directory, you'll need to resubmit it a

Bot Token Scopes

Scopes that govern what your app can access.

OAuth Scope	Description	
incoming-webhook	Post messages to specific channels in Slack	
chat:write	Send messages as @python_bot	
files:write	Upload, edit, and delete files as python_bot	
Add an OAuth Scope		



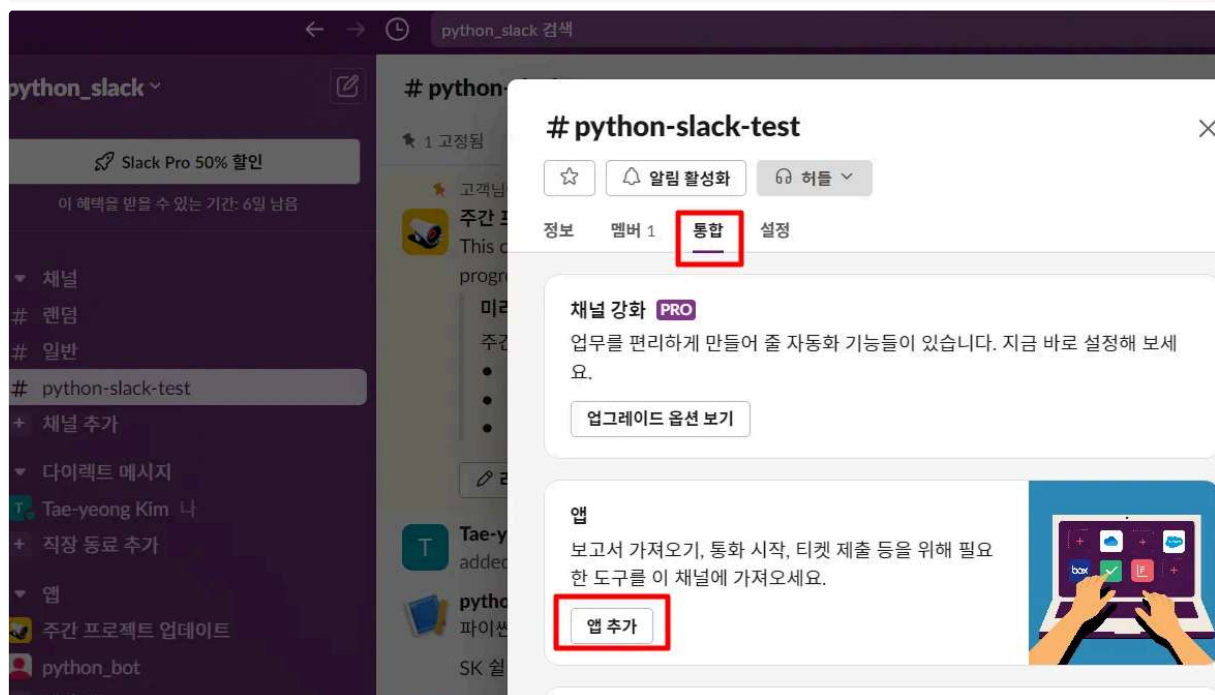
The image shows a Slack workspace setup screen. At the top, there are two icons: a blue book icon and a red square icon, connected by a double-headed arrow. Below this, the text reads: "python_bot에서 python_slack Slack 워크스페이스에 액세스하기 위해 권한을 요청합니다." (Requesting permissions for python_bot to access the python_slack Slack workspace). The next question is "python_bot에서 무엇을 할 수 있나요?" (What can python_bot do?). Below this is a search bar with the text "채널과 대화에서 작업 수행" (Perform work in channels and conversations). The next question is "python_bot은(는) 어디에 게시해야 할까요?" (Where should python_bot post?). Below this is a text input field with the text "# python-slack-test" and a dropdown arrow. At the bottom, there are two buttons: "취소" (Cancel) and "허용" (Allow). The "허용" button is highlighted with a red box.

Python Slack API

- slack_sdk 라이브러리 설치

```
pip install slack_sdk
```

```
from slack_sdk import WebClient from slack_sdk.errors import SlackApiError
```



☰ 앱 디렉터리 보기


python-slack-test에 앱 추가

Q 이름 또는 카테고리(예: 생산성, 판매)로 검색

업무를 완료하는 데 도움이 되는 앱 추가
 회의 일정 관리, 문서 공유 등을 Slack에서 바로 수행할 수 있습니다. [앱 제안 보기](#)



워크스페이스에서(1)

 python_bot

추가

Slack 메시지 전송 자동화

- 간단한 메시지 전송 구현 코드

```
from slack_sdk import WebClient from slack_sdk.errors import SlackApiError
# Slack API 토큰과 메시지를 보낼 채널 설정 SLACK_API_TOKEN = "your-slack-api-token" SLACK_CHANNEL = "your-channel-id" def send_message(channel, text):
# WebClient 인스턴스 생성 client = WebClient(token=SLACK_API_TOKEN) try: # 채널에 메시지 전송 response = client.chat_postMessage( channel=channel, text=text ) # 응답 출력 print("Message sent successfully:", response["message"]["text"]) except SlackApiError as e: # 에러 처리 print("Error sending message:", e.response["error"]) # 메시지 전송 함수 호출 send_message(SLACK_CHANNEL, "Hello, this is a test message from Slack API!")
```

1. 필요한 라이브러리 импорт

- slack_sdk 라이브러리의 WebClient 클래스와 SlackApiError 예외를 импорт


```
from slack_sdk import WebClient from slack_sdk.errors import SlackApiError
```

2. Slack API Token과 채널 설정:

```
SLACK_API_TOKEN = "your-slack-api-token" SLACK_CHANNEL = "#your-channel-name"
```

- **SLACK_API_TOKEN**: 이 토큰은 Slack 애플리케이션에서 생성되며, 사용자 또는 봇이 Slack API에 접속하여 인증을 받고 메시지를 보내는 등의 작업을 수행할 때 사용. 이 토큰은 보안을 유지해야 하는 중요한 정보.
- **SLACK_CHANNEL**: 메시지를 보낼 Slack 채널의 이름. 여기서는 예시로 "#your-channel-name"을 사용했으나, 실제 채널명으로 변경 필요. 채널명은 메시지가 전송될 대상을 지정

3. Slack 메시지 전송 함수:

```
def send_slack_message(message): try: # WebClient 인스턴스를 생성하고, 생성자에 Slack API 토큰을 전달합니다. client = WebClient(token=SLACK_API_TOKEN) # chat_postMessage 메소드를 사용하여 메시지를 Slack 채널에 전송합니다. response = client.chat_postMessage( channel=SLACK_CHANNEL, text=message ) # 응답을 출력하여 메시지 전송 성공 여부를 확인합니다. print(f"Slack message sent: {response['message']['text']}") except SlackApiError as e: # 메시지 전송 중 발생한 에러를 캐치하고, 에러 메시지를 출력합니다. print(f"Error sending message to Slack: {e.response['error']}")
```

- **WebClient**: **slack_sdk** 라이브러리의 **WebClient** 클래스는 Slack API에 접근하여 다양한 API 호출을 수행.
- **chat_postMessage** 메소드: 이 메소드는 지정된 **channel** 에 **text** 로 전달된 메시지를 전송. 메소드 호출이 성공하면, Slack API로부터 메시지에 대한 응답을 받아와서 처리 가능. 메시지 전송 실패 시 **SlackApiError** 예외가 발생하며, 이는 에러 처리 블록에서 캐치되어 에러 상세 출력.

WebClient 클래스

- `WebClient` 클래스는 `slack_sdk` 라이브러리에서 제공하는 주요 클래스 중 하나
- Python에서 Slack API에 접근하여 다양한 작업을 수행하는 데 사용
- 이 클래스를 사용하면 Slack의 Web API 메소드를 호출하여 메시지 보내기, 채널 관리, 사용자 정보 조회, 파일 업로드 등의 작업 수행 가능

다양한 API 메소드 지원

- WebClient 클래스는 Slack의 거의 모든 Web API 메소드를 호출할 수 있도록 메소드를 제공
- 메시지 보내기(`chat_postMessage`), 파일 업로드(`files_upload`), 사용자 정보 조회(`users_info`), 채널 정보 관리(`conversations_info`) 등

1. 인스턴스 생성:

- `WebClient` 클래스는 Slack API 토큰을 인자로 받아 인스턴스를 생성
- 이 토큰은 API 요청을 인증하는 데 사용

```
from slack_sdk import WebClient client = WebClient(token="your-slack-api-token")
```

2. 메시지 보내기:

- `chat_postMessage` 메서드를 사용하여 특정 채널에 메시지를 보낼 수 있음
- 필수 인자는 채널 ID(또는 이름)와 메시지 내용

```
response = client.chat_postMessage(channel="#general", text="Hello, world!")
```

chat_postMessage() 메서드

- `chat_postMessage()` 메서드는 Slack의 WebClient API를 사용하여 특정 채널에 메시지를 보내는 핵심 기능
- `channel` : 메시지를 보낼 채널의 ID나 이름

- **text**: 전송할 메시지의 텍스트

```
response = client.chat_postMessage(channel="#general", text="Hello, world!")
```

응답 처리

- 메소드 호출 성공 시, Slack API는 다양한 정보를 담고 있는 **response** 객체를 반환
- 이 객체는 API 호출이 성공했는지 여부(**ok** 필드), 메시지가 전송된 채널과 시간스탬프, 메시지의 고유 ID 등을 포함
- 에러가 발생하면 **SlackApiError** 예외가 발생하며, 이 예외를 처리하여 에러의 원인을 파악 가능

```
try: response = client.chat_postMessage(channel="#general", text="Hello, world!")
    print("Message sent successfully, response:", response)
except SlackApiError as e: print(f"Error sending message: {e.response['error']}")
```

Slack 파일 업로드 및 메시지 전송 자동화

- 간단한 파일 업로드 및 메시지 전송 구현 코드

```

from slack_sdk import WebClient from slack_sdk.errors import SlackApiError
# Slack API 토큰과 메시지를 보낼 채널 설정 SLACK_API_TOKEN = "your-slack-api-token"
SLACK_CHANNEL = "your-channel-id" # 채널 접근 후 URL 뒤에서 확인 가능
def upload_file(channel, file_path, message): # WebClient 인스턴스 생성
    client = WebClient(token=SLACK_API_TOKEN)
    try: # 파일을 Slack 채널에 업로드 하고, 해당 파일에 메시지를 추가합니다.
        response = client.files_upload_v2(channel=channel, file=file_path, initial_comment=message)
        # 업로드 성공 메시지 출력
        print("File uploaded successfully:", response["file"]["name"])
    except SlackApiError as e: # 에러 처리
        print("Error uploading file:", e.response["error"])
# 파일 업로드 및 메시지 전송 함수 호출
upload_file(SLACK_CHANNEL, "path/to/yourfile.pdf", "Here is the file you requested!")

```

files_upload_v2() 메서드

- `files_upload_v2()` 함수는 Slack의 웹 API를 사용하여 파일을 Slack 채널이나 개인 메시지로 업로드하는 데 사용되는 메서드
- `files_upload()`의 후속 버전으로서, 더 나은 성능과 안정성을 제공하기 위해 개선됨

기능

- Slack 채널 또는 개인 메시지에 파일 업로드
- 초기 메시지와 함께 파일을 전송할 수 있으며, 이는 파일 업로드와 동시에 전달
- 큰 파일을 업로드할 때 더 나은 성능과 안정성을 제공

파라미터

- `channel`: 파일을 업로드할 채널의 ID 또는 콤마로 구분된 채널의 ID 목록.
`files_upload_v2()`에서는 채널을 지정하는 데 `channel` 파라미터를 사용하는 것을 권장.
- `file`: 업로드할 파일의 경로
- `initial_comment`: 파일과 함께 전송할 메시지
- `filename`: 업로드할 파일의 이름을 명시적으로 지정. 지정하지 않으면 원본 파일 이름을 사용.
- `title`: 파일에 제목 추가

```
from slack_sdk import WebClient from slack_sdk.errors import SlackApiError
SLACK_API_TOKEN = "your-slack-api-token" SLACK_CHANNEL = "C1234567890" cli
ent = WebClient(token=SLACK_API_TOKEN) try: response = client.files_upload
_v2( channel=SLACK_CHANNEL, file="path/to/yourfile.pdf", initial_comment
="Here is the document we discussed.", filename="document.pdf", title="Imp
ortant Document" ) print("File uploaded successfully:", response["file"]
["name"]) except SlackApiError as e: print("Error uploading file:", e.resp
onse["error"])
```

11시 15분까지

```
from slack_sdk import WebClient from slack_sdk.errors import SlackApiError
```