

# CHATBOT DEMO

## *with Slack*

Revised @ 2021.09.25

# 목차

01

## SETUP

슬랙 설치와 기본적인  
셋업을 알아본다

02

## BASICS

챗봇의 기본적인 기능을  
사용해본다

03

## SLACK "BLOCKS"

슬랙의 "block"  
기능에 대해 알아본다

04

## DEMO : MOVIE BOT

영화정보를 불러오는  
챗봇 데모를 알아본다



01

SETUP

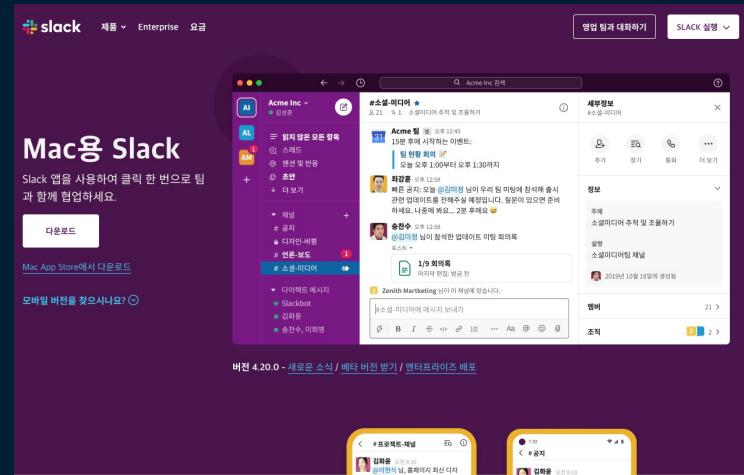
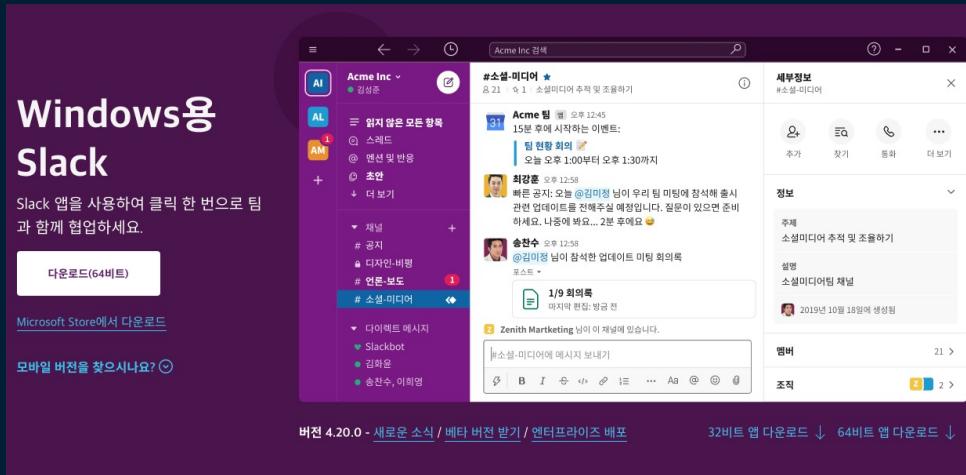
# 슬랙 설치

MAC OS:

URL : <https://slack.com/intl/ko-kr/downloads/mac?geocode=ko-kr>

WINDOWS:

URL : <https://slack.com/intl/ko-kr/downloads/windows?geocode=ko-kr>



# 워크스페이스 생성

Side Bar가 없는 경우

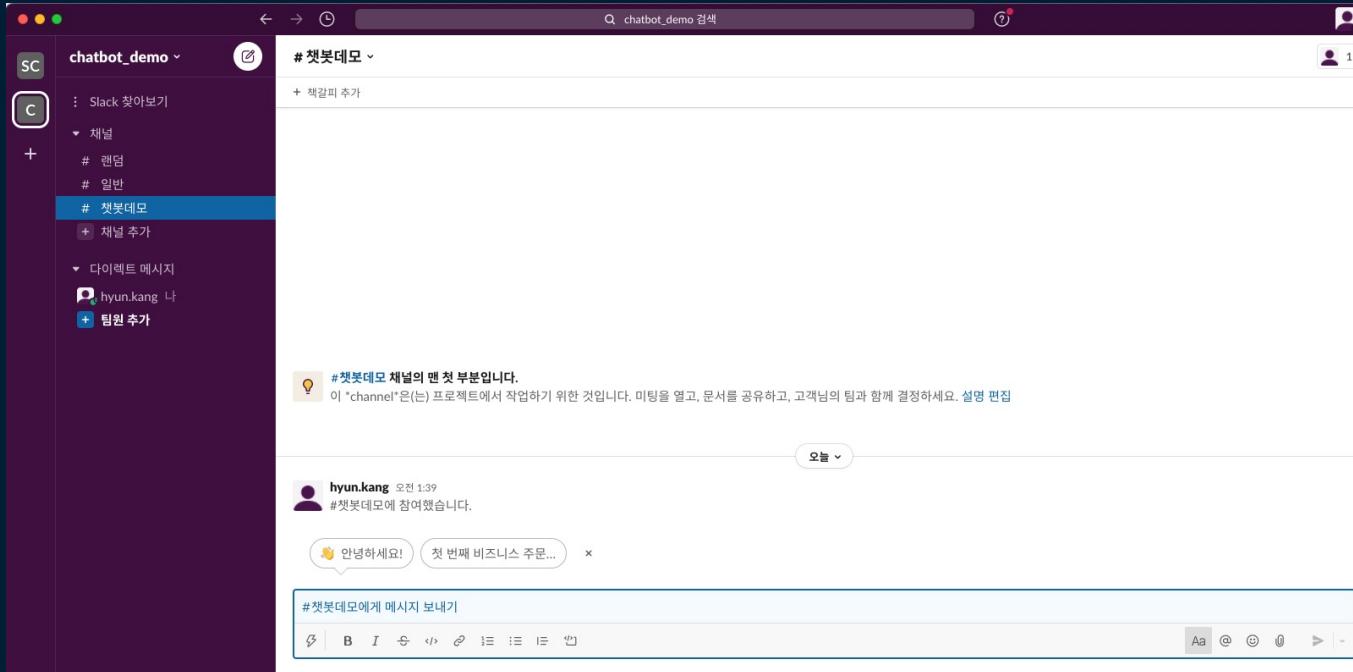
The screenshot shows the Slack desktop application window. At the top, there's a search bar with the text "slack chatbot 검색". Below it is a header bar with tabs for "slack chatbot" and "SC". The main workspace is titled "# 일반". A message from "myChatBot" is visible: "고객님의 워크스페이스에서는 현재 Slack 무료 버전을 사용하고 있습니다. [클릭 보기](#)". On the left, there's a sidebar with sections like "slack chatbot에 사용자 초대", "환경설정", "설정 및 관리", and "도구". A "워크스페이스 추가" button is highlighted with a red box. A dropdown menu appears over this button, containing three items: "다른 워크스페이스에 로그인", "새 워크스페이스 생성", and "워크스페이스 찾기". The "새 워크스페이스 생성" option is also highlighted with a red box.

Side Bar가 있는 경우

This screenshot shows the same Slack interface but with a sidebar on the left. The sidebar contains sections such as "Slack 소개하기", "채널", "# 일반", "워크스페이스", "설정", and "도구". The "워크스페이스" section is expanded, showing a list of existing workspaces. One workspace, "워크스페이스 생성", is highlighted with a red box. The main workspace area shows a message from "myChatBot": "+슬랙-봇-생성 채널이 맨 첫 부분입니다. 이 channel(한글) 프로젝트에서 적극하기 위한 것입니다. 미동을 알고, 문서를 공유하고, 고객님의 팀과 함께 결정하세요. [클릭 편집](#)". Below this, another message from "myChatBot" says "Hello, World!".

# 워크스페이스 생성

## 완성 화면:



# API 세팅

## 1. SLACK API 홈페이지 접속

The screenshot shows the Slack API homepage with a dark background. At the top, there's a navigation bar with a search bar, documentation links, and a tab labeled "Your Apps" which is currently selected. On the left, there's a sidebar with various API-related links: "Start learning", "Authentication", "Surfaces", "Block Kit", "Interactivity", "Messaging", "APIs" (which is expanded to show "Overview", "Connection protocols", "Usage guides", and several specific API endpoints like "Events API", "Web API", "Pagination", etc.), and "Reference" sections for "Methods" and "Event types". The main content area is titled "Your Apps" and features a large button labeled "Create an App". Below this button, there's a message encouraging users to build amazing apps using Slack's APIs.

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

# API 세팅

## 2. SLACK APP 생성

The screenshot shows the Slack API documentation page with a sidebar on the left containing various API-related links. A central modal window titled 'Create an app' is displayed, prompting the user to choose how to configure their app's scopes and settings. Two options are shown: 'From scratch' and 'From an app manifest'. The 'From scratch' option is highlighted with a red rectangular box. Below the modal, there is a note about getting help through documentation or examples.

slack api

Documentation Tutorials Your Apps

Start learning

Authentication

Surfaces

Block Kit

Interactivity

Messaging

APIs

Overview

Connection protocols

Overview

Using the Events API over...

Intro to Socket Mode

Socket Mode implement...

Usage guides

Using the Web API

Pagination

Rate limits

Slack Connect

Status API

Calls API

Presence & status

Reference

Methods

Event types

Search

Your Apps

Build something amazing.

Use our APIs to build an app that makes people's working lives better. You can create an app.

Create an app

Choose how you'd like to configure your app's scopes and settings.

**From scratch**

Use our configuration UI to manually add basic info, scopes, settings, & features to your app.

**From an app manifest BETA**

Use a manifest file to add your app's basic info, scopes, settings & features to your app.

Need help? Check our [documentation](#) or [see an example](#)

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

# API 세팅

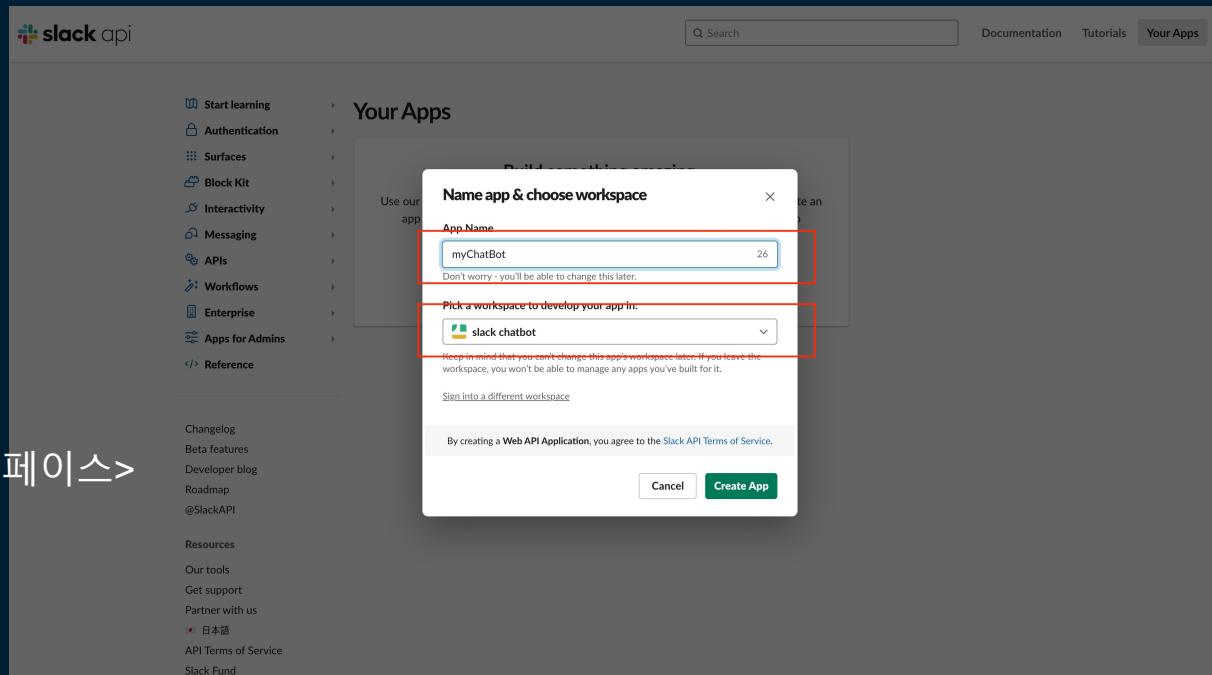
## 3. SLACK APP 생성

APP NAME :

<임의의 이름>

WORKSPACE :

<이전 단계에서 생성한 워크스페이스>



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

# API 세팅

## 4. BOTS 세팅

The screenshot shows the Slack API Settings page. On the left, there's a sidebar with navigation links like 'Basic Information', 'Collaborators', 'Socket Mode', 'Install App', 'App Manifest (BETA)', 'Manage Distribution', 'Features', 'App Home', 'Org Level Apps', 'Incoming Webhooks', 'Interactivity & Shortcuts', 'Slash Commands', 'Workflow Steps', 'OAuth & Permissions', 'Event Subscriptions', 'User ID Translation', 'Beta Features', 'Where's Bot User', 'Submit to App Directory', 'Review & Submit', 'Slack ❤️', 'Help', 'Contact', and 'Policies'. The main content area has a search bar at the top right and tabs for 'Documentation', 'Tutorials', and 'Your Apps' (which is selected). Below the tabs, the title 'Building Apps for Slack' is followed by a sub-section titled 'Add features and functionality'. This section contains several cards: 'Incoming Webhooks', 'Interactive Components', 'Slash Commands', 'Event Subscriptions', 'Bots' (which is highlighted with a red box), and 'Permissions'. At the bottom, there's a button for 'Install your app' and two buttons for 'Discard Changes' and 'Save Changes'.

Basic Information

Collaborators

Socket Mode

Install App

App Manifest BETA

Manage Distribution

Features

App Home

Org Level Apps

Incoming Webhooks

Interactivity & Shortcuts

Slash Commands

Workflow Steps

OAuth & Permissions

Event Subscriptions

User ID Translation

Beta Features

Where's Bot User

Submit to App Directory

Review & Submit

Slack ❤️

Help

Contact

Policies

Search

Documentation

Tutorials

Your Apps

### Building Apps for Slack

Create an app that's just for your workspace (or build one that can be used by any workspace) by following the steps below.

#### Add features and functionality

Choose and configure the tools you'll need to create your app (or review all [our documentation](#)).

**Building an internal app locally or behind a firewall?**  
To receive your app's payloads over a WebSockets connection, enable Socket Mode for your app.

**Incoming Webhooks**  
Post messages from external sources into Slack.

**Interactive Components**  
Add components like buttons and select menus to your app's interface, and create an interactive experience for users.

**Slash Commands**  
Allow users to perform app actions by typing commands in Slack.

**Event Subscriptions**  
Make it easy for your app to respond to activity in Slack.

**Bots**  
Allow users to interact with your app through channels and conversations. +

**Permissions**  
Configure permissions to allow your app to interact with the Slack API.

Install your app

Discard Changes

Save Changes

<https://app.slack.com/app-settings/T025T7XTD8C/A025T83H9E0/distribute>

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

# API 세팅

## 4. BOTS 세팅

The screenshot shows the Slack API App Home page for an app named "myChatBot". The left sidebar contains navigation links for "Settings" (Basic Information, Collaborators, Socket Mode, Install App, App Manifest BETA, Manage Distribution) and "Features" (App Home, Org Level Apps, Incoming Webhooks, Interactivity & Shortcuts, Slash Commands, Workflow Steps, OAuth & Permissions, Event Subscriptions, User ID Translation, Beta Features, Where's Bot User). The "App Home" link is highlighted with a blue background. The main content area is titled "App Home" and includes sections for "Where people find your app on Slack" (describing the Home, Messages, and About tabs), "First, assign a scope to your bot token" (explaining what a bot token is and how scopes govern capabilities), and "Show Tabs" (describing how to turn on Home and Message tabs). A red box highlights the "Review Scopes to Add" button.

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

# API 세팅

## 4. BOTS 세팅

### BOT TOKEN SCOPES:

1. calls:read
2. channels:read
3. chat:write
4. app\_mentions:read

The screenshot shows the Slack API Scopes configuration page. At the top, there is a message: "You can now show tabs on App Home. Manage which tabs your user sees in your app's home. [Go to App Home](#)". Below this, the "Scopes" section is titled "Bot Token Scopes". It says, "A Slack app's capabilities and permissions are governed by the scopes it requests." Under "Bot Token Scopes", it says, "Scopes that govern what your app can access." A table lists four OAuth Scopes:

OAuth Scope	Description
calls:read	View information about ongoing and past calls
channels:read	View basic information about public channels in a workspace
chat:write	Send messages as myChatBot
app_mentions:read	View messages that directly mention myChatBot in conversations that the app is in

Each row has a trash icon. Below the table is a button "Add an OAuth Scope". The "User Token Scopes" section is also shown, stating "You haven't added any OAuth Scopes for your User token." There is a search bar "Add permission by Scope or API method..." and a "Add an OAuth Scope" button. A note at the bottom says, "Scopes define the API methods an app is allowed to call, which information and capabilities are available on".

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

# API 세팅

## 4. BOTS 세팅

The screenshot shows the Slack API OAuth & Permissions page for an app named "myChatBot". The left sidebar lists various settings and features, with "OAuth & Permissions" selected. The main content area displays the "OAuth Tokens for Your Workspace" section, which includes a button labeled "Install to Workspace" (which is highlighted with a red box). Below this is the "Redirect URLs" section, which states "You haven't added any Redirect URLs" and includes "Add New Redirect URL" and "Save URLs" buttons. A note at the bottom says "You can now show tabs on App Home". The top navigation bar includes links for Documentation, Tutorials, and Your Apps.

slack api

myChatBot

Settings

- Basic Information
- Collaborators
- Socket Mode
- Install App
- App Manifest BETA
- Manage Distribution

Features

- App Home
- Org Level Apps
- Incoming Webhooks
- Interactivity & Shortcuts
- Slash Commands
- Workflow Steps
- OAuth & Permissions**
- Event Subscriptions
- User ID Translation
- Beta Features
- Where's Bot User

Submit to App Directory

Review & Submit

Slack ❤️  
Help  
Contact

Search

Documentation Tutorials Your Apps

### OAuth & Permissions

#### OAuth Tokens for Your Workspace

These OAuth Tokens will be automatically generated when you finish connecting the app to your workspace. You'll use these tokens to authenticate your app.

Install to Workspace

#### Redirect URLs

You haven't added any Redirect URLs

Add New Redirect URL

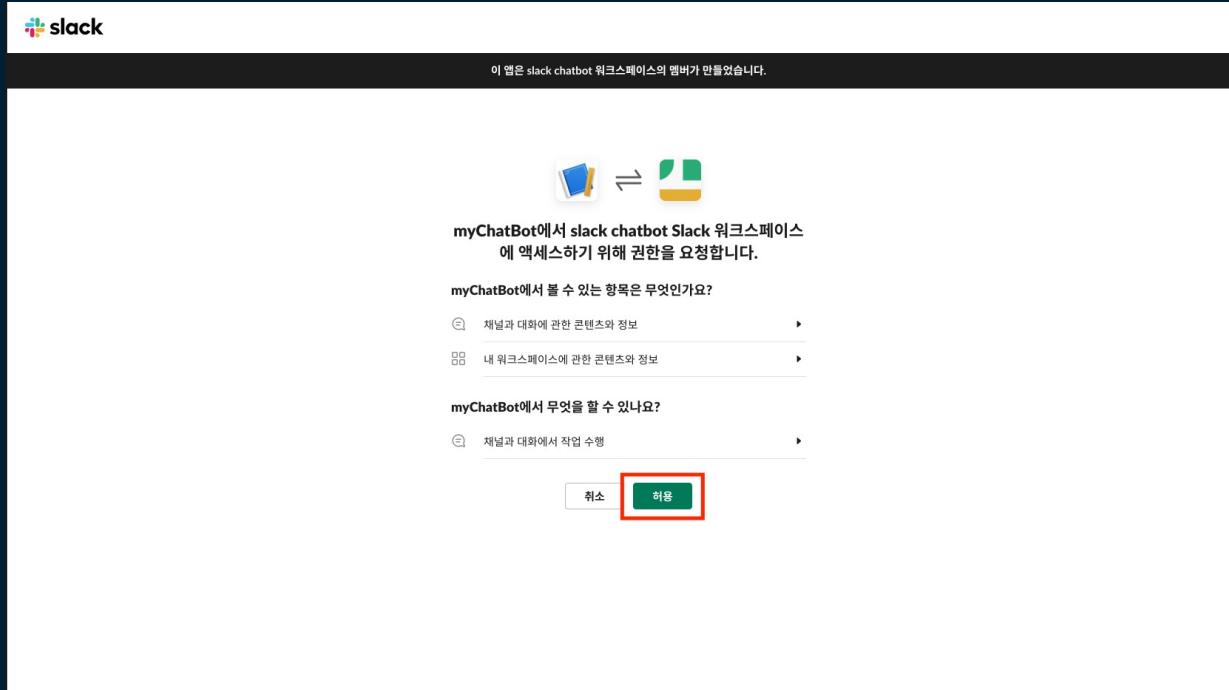
Save URLs

You can now show tabs on App Home  
Manage which tabs your user sees in your app's home. [Go to App Home](#)

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

# API 세팅

## 4. BOTS 세팅



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

# API 세팅

## 4. BOTS 세팅

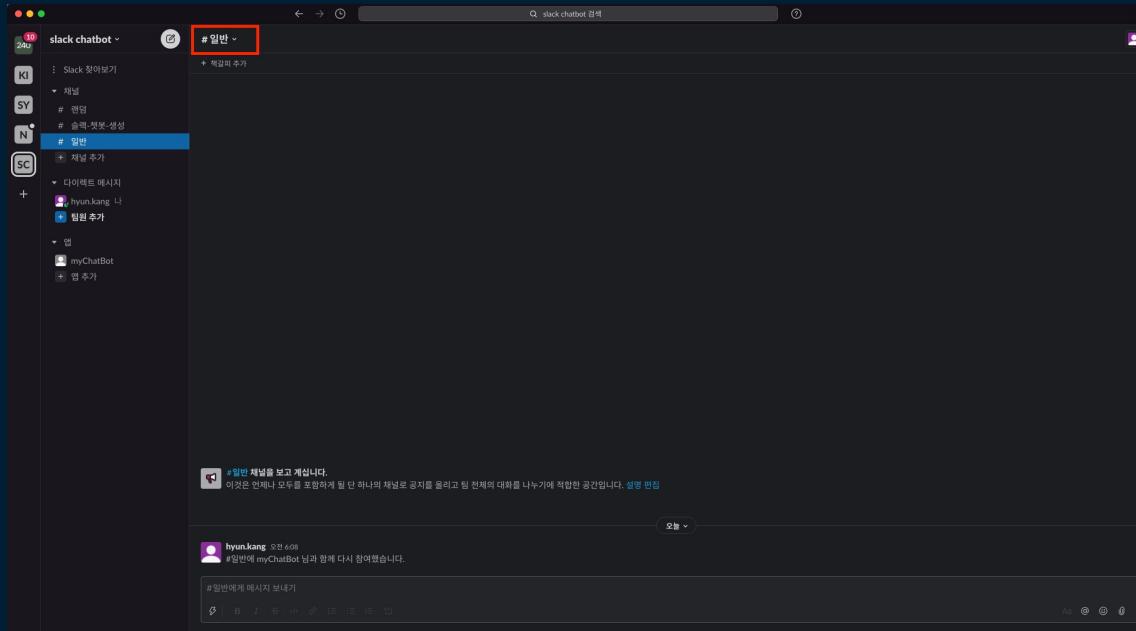
Bot User OAuth Token 저장

The screenshot shows the Slack App Settings interface. On the left, a sidebar lists various settings like Basic Information, Collaborators, and Features. The 'OAuth & Permissions' tab is selected, highlighted by a blue bar. In the main content area, there's a section titled 'OAuth Tokens for Your Workspace' which contains a single token entry: 'Bot User OAuth Token' with the value 'xoxb-2197269931284-2191078061234-uOanNpBwTDm3gqlHS0YgqYt'. A red box highlights this token entry. Below it is a 'Copy' button. Further down, there's a 'Reinstall to Workspace' button. Another section titled 'Redirect URLs' is shown below, with a note that no redirect URLs have been added yet.

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

# API 세팅

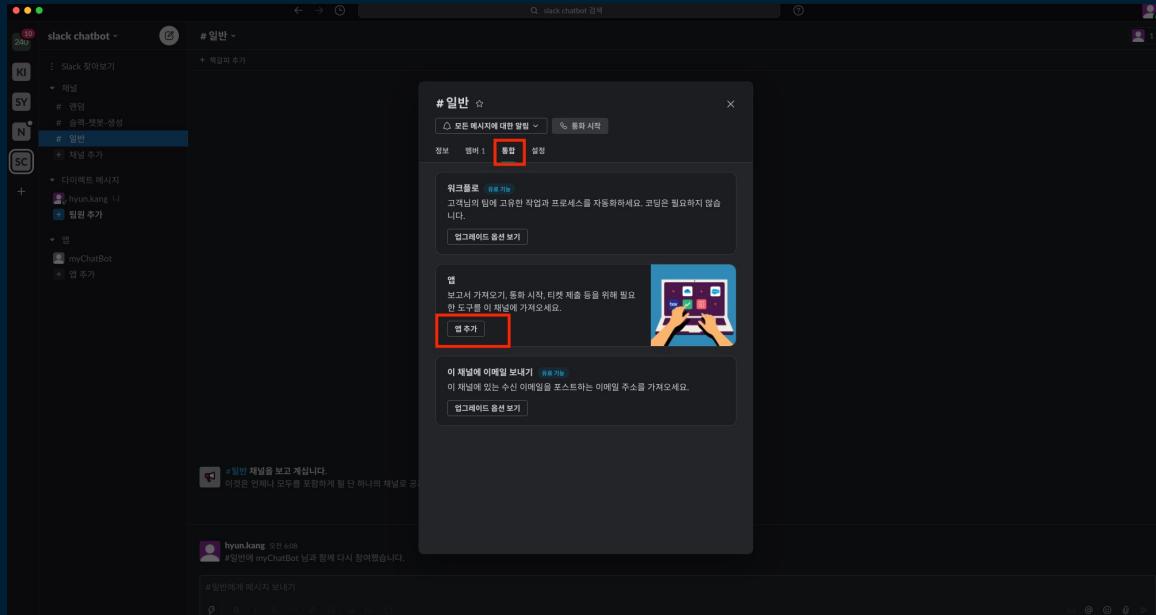
## 5. Slack App 추가



# API 세팅

## 5. Slack App 추가

1. 통합 탭 클릭
2. 앱 추가
3. 생성한 앱 선택 후 추가





**02**

# **BASICS**

# 라이브러리 설치

URL:

[https://drive.google.com/drive/folders/18aAVylX4ku9hefhgJU4KKWpJzfcLUKhK  
?usp=sharing](https://drive.google.com/drive/folders/18aAVylX4ku9hefhgJU4KKWpJzfcLUKhK?usp=sharing)

```
$ pip install -r requirements.txt
```

# 환경변수 설정

## REFERENCE : .env

```
◀ ▶ .env x
1 #.env 파일
2
3 # 환경변수를 독립적으로 관리하는 파일
4
5 # Development Setting
6 # 슬랙에서 발급받은 OAuth Tokens을 환경변수로 지정
7
8 SLACK_TOKEN = BOT_USER_OAUTH_TOKEN
9 |
```

# 통신모듈 설정

REFERENCE : **handler.py**

```
1 # handler.py
2 # 슬랙과 통신하는 모듈을 정의하는 코드
3
4 import os
5
6 # pip install python-dotenv
7 from dotenv import load_dotenv
8
9 # pip install slack_sdk
10 from slack_sdk import WebClient
11 from slack_sdk.errors import SlackApiError
12
```

# 메시지 보내기

## REFERENCE : 1\_simple\_pose.py

```
1  # 1_simple_post.py
2  # 생성한 슬랙 봇을 통해 메시지를 보내는 예제
3
4  # 생성한 슬랙 handler 불러오기
5  from handler import slack_handler
6
7  # 메인 함수
8  if __name__ == "__main__":
9      slack_handler.post_slack_message(message="Hello, World!", channel="#일반")
```

## 생성된 query

```
{'ok': True, 'channel': 'C026AN94Q2U', 'ts': '1624228999.001200', 'message': {'bot_id': 'B025T8AULNQ', 'type': 'message', 'text': 'Hello, World!', 'user': 'U025M2A1T6W', 'ts': '1624228999.001200', 'team': 'T025T7XTD8C', 'bot_profile': {'id': 'B025T8AULNQ', 'deleted': False, 'name': 'myChatBot', 'updated': 1624224709}, 'app_id': 'A025T83H9E0', 'image_36': 'https://slack-edge.com/80588/img/plugins/app/bot_36.png', 'image_48': 'https://slack-edge.com/80588/img/plugins/app/bot_48.png', 'image_72': 'https://slack-edge.com/80588/img/plugins/app/service_72.png'}, 'team_id': 'T025T7XTD8C'}} everywhere for classes, files, tool windows, actions, and settings.
```

## 결과



myChatBot 웹 오전 7:29

Hello, World!

# 플라스크 서버 설정

## REFERENCE: 2\_auto\_response.py

```
1 import json
2
3 from flask import Flask, request, make_response
4
5 from handler import slack_handler
6
7 # 플라스크 인스턴스 생성
8 app = Flask(__name__)
9
```

# 플라스크 서버 설정

## REFERENCE: 2\_auto\_response.py

```
12 @app.route('/', methods=['POST'])
13 def default_listener():
14     # 슬랙에서 보낸 request 데이터를 json으로 파싱한다.
15     slack_event = json.loads(request.data)
16     print(slack_event)
17
18     # 인자 중 challenge가 있으면 해당 인자의 값을 반환한다.
19     # slack api specification. 참고:https://api.slack.com/
20     if "challenge" in slack_event:
21         return make_response(slack_event["challenge"], 200, {"content_type": "application/json"})
22
23     # slack에서 발생한 event를 통한 request에 대한 핸들링
24     if "event" in slack_event:
25         event_type = slack_event["event"]["type"]
26         # bot_mention일 경우에 대한 핸들링
27         if event_type == 'app_mention':
28             try:
29                 # 멘션을 남긴 채널 읽어오기
30                 channel = slack_event['event']['channel']
31                 # 유저가 멘션과 함께 남긴 텍스트 읽어오기
32                 user_query = slack_event['event']['blocks'][0]['elements'][0]['elements'][1]['text']
33
34                 slack_handler.post_slack_message(message=user_query, channel=channel)
35
36                 # 정상적으로 완료했음에 대한 http response
37                 return make_response("response made :)", 200, )
38             except IndexError:
39                 # 멘션은 했지만 텍스트는 남기지 않은 경우에 대한 에러.
40                 # do nothing
41                 pass
42
43             # 그 외 event에 대한 핸들링: 404 error
44             msg = f"[{event_type}] cannot find event handler"
45             return make_response(msg, 404, {"X-Slack-No-Retry": 1 })
```

# Ngrok 설정

The screenshot shows the Ngrok homepage. At the top, there's a navigation bar with links for 'How it works', 'Pricing', 'Enterprise solutions', 'Docs', 'Download', 'Login', and 'Sign up'. Below the navigation, there's a section titled 'Public URLs for SSH access'. It includes a sub-section with the text: 'Spend more time programming. One command for an instant, secure URL to your localhost server through any NAT or firewall.' A blue button labeled 'Get started for free' is visible. To the right of this text is a screenshot of a terminal window showing the output of the command: \$ ./ngrok http 3000 ngrokby @inconschreivable. The terminal also displays session details: Session Status online, Account Kate Libby (Plan: Pro), Web Interface http://127.0.0.1:4443, Forwarding https://katesapp.ngrok.io -> localhost, and Forwarding https://katesapp.ngrok.io -> localhost. Below the terminal screenshot, there are logos for Slack, GitHub, SendGrid, Twilio, and Atlassian. A note says 'As well as Amazon Web Services and many more.' At the bottom of the page, there's a dark blue footer with the text 'TAKE ADVANTAGE OF A POWERFUL LOCAL INSPECTOR' and a description of the Ngrok Inspector feature. A screenshot of the Ngrok Inspector interface is shown, along with a 'Documentation' link and an 'Ask a question' button.

1. <https://ngrok.com/>

2. 다운로드

3. Ngrok 실행 파일을 코드와 같은  
폴더로 이동

# Ngrok 실행

## 1. 작성한 플라스크 서버를 로컬서버에서 실행

```
→ codes python3 autoResponse.py
 * Serving Flask app 'autoResponse' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 845-649-508
```

## 2. Ngrok을 통해 로컬과 외부 인터넷 연결

```
→ codes ./ngrok http 5000
```

## 3. Forwarding URL 복사

```
ngrok by @inconshreveable                                     (ctrl+c to quit)

Session Status      online
Account            KORguy (Plan: Free)
Version             2.3.40
Region              United States (us)
Web Interface      http://127.0.0.1:4040
Forwarding          http://ffd948d685a0.ngrok.io -> http://localhost:5000
Forwarding          https://ffd948d685a0.ngrok.io -> https://localhost:5000

Connections        ttl     opn     rtt1    rtt5     p50     p90
                    0       0      0.00    0.00    0.00    0.00
```

# SLACK API 추가 설정

슬랙에서 봇을 멘션하면 플라스크 서버로 request 보내는 환경 설정

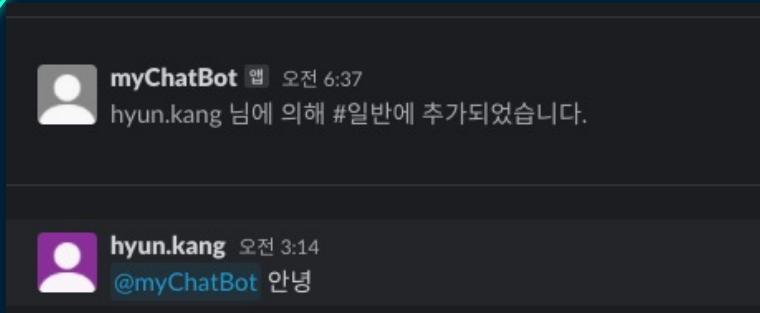
1. Your apps > event subscriptions
2. Enable events
3. Subscribe to bot events > app\_mention
4. Request url :  
<forwarding\_url\_from\_ngrok>

The screenshot shows the Slack API configuration interface for the 'myChatBot' app. The left sidebar lists various settings like Basic Information, Collaborators, and Features (with 'Event Subscriptions' highlighted). The main content area is titled 'Event Subscriptions'. It has two main sections: 'Enable Events' (with a green 'On' toggle switch) and 'Request URL' (containing the value 'https://my.app.com/slack/action-endpoint'). Below these is a note about the new event authorization format. The 'Subscribe to bot events' section contains a table with one row for 'app\_mention' (Description: 'Subscribe to only the message events that mention your app or bot', Required Scope: 'app\_mentions:read'). At the bottom are 'Discard Changes' and 'Save Changes' buttons.

Event Name	Description	Required Scope
app_mention	Subscribe to only the message events that mention your app or bot	app_mentions:read

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

# 자동응답 테스트



```
{
  "token": "fd54wao3UbSiugxNiVAWXq0W",
  "team_id": "T025T7XTD8C",
  "api_app_id": "A025T83H9E0",
  "event": {
    "client_msg_id": "9b9bbabe-c2df-4552-894a-2a35baf93730",
    "type": "app_mention",
    "text": "<@U025M2A1T6W> 안녕",
    "user": "U025E9D5KSS",
    "ts": "1624299241.000300",
    "team": "T025T7XTD8C",
    "blocks": [
      {
        "type": "rich_text",
        "block_id": "D2m",
        "elements": [
          {
            "type": "rich_text_section",
            "elements": [
              {
                "type": "user",
                "user_id": "U025M2A1T6W"
              },
              {
                "type": "text",
                "text": "안녕"
              }
            ]
          }
        ]
      }
    ],
    "channel": "C026AN94Q2U",
    "event_ts": "1624299241.000300"
  },
  "type": "event_callback",
  "event_id": "Ev02632VGUGZ",
  "event_time": 1624299241,
  "authorizations": [
    {
      "enterprise_id": None,
      "team_id": "T025T7XTD8C",
      "user_id": "U025M2A1T6W",
      "is_bot": True,
      "is_enterprise_install": False
    }
  ],
  "is_ext_shared_channel": False,
  "event_context": "2-app_mention-T025T7XTD8C-A025T83H9E0-C026AN94Q2U"
}
```

# Handler 추가

## REFERENCE : updated\_handler.py

```
# 레이아웃을 포함한 메시지를 보내는 메소드
def post_layout_message(self, channel: str, blocks: list):
    try:
        response = self.client.chat_postMessage(channel=channel, blocks=blocks)
        print(response)
    except SlackApiError as e:
        assert e.response["ok"] is False
        assert e.response["error"]
        print(f"ERROR: {e.response['error']}")
```

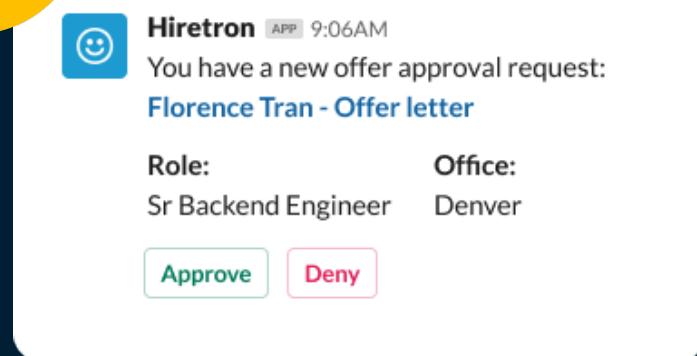
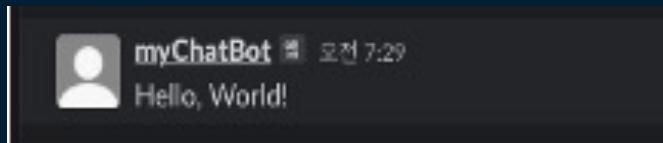


03

BLOCKS

# Slack Block이란.

단순한 텍스트를 넘어 이미지나 링크, 투표 등을 포함한 레이아웃에 맞춰진 응답 또한 가능.



# Block Types

Block type	Available in surfaces		
Actions	Modals	Messages	Home tabs
Context	Modals	Messages	Home tabs
Divider	Modals	Messages	Home tabs
File	Messages		
Header	Modals	Messages	Home tabs
Image	Modals	Messages	Home tabs
Input	Modals	Messages	Home tabs
Section	Modals	Messages	Home tabs

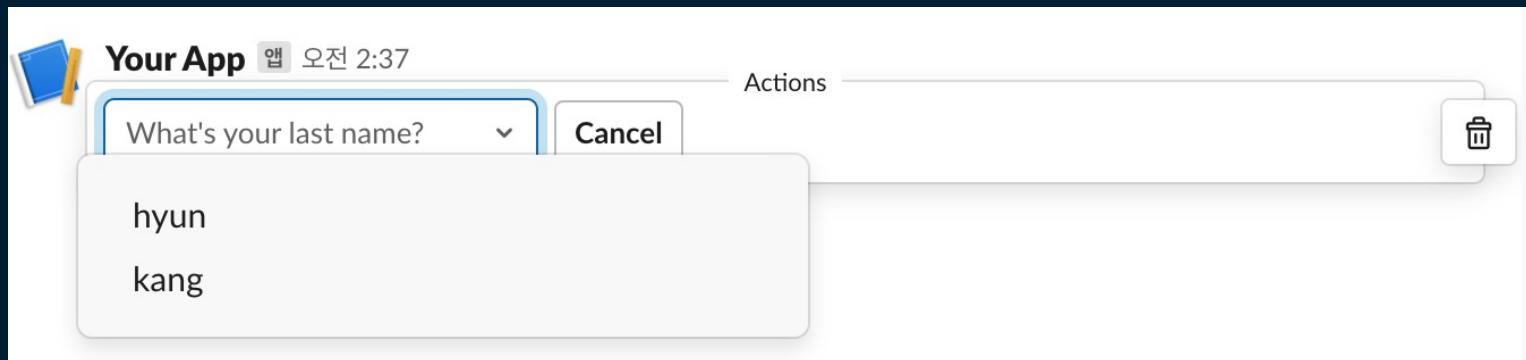
# 1. Action Type

사용자와 상호작용을 할 수 있는 버튼 등을 포함한 블록

Field	Type	Required?	Description
<code>type</code>	String	Yes	The type of block. For an actions block, <code>type</code> is always <code>actions</code> .
<code>elements</code>	Object[]	Yes	An array of interactive <a href="#">element objects</a> - buttons, select menus, overflow menus, or date pickers. There is a maximum of 5 elements in each action block.
<code>block_id</code>	String	No	A string acting as a unique identifier for a block. If not specified, a <code>block_id</code> will be generated. You can use this <code>block_id</code> when you receive an interaction payload to <a href="#">identify the source of the action</a> . Maximum length for this field is 255 characters. <code>block_id</code> should be unique for each message and each iteration of a message. If a message is updated, use a new <code>block_id</code> .

# 1. Action Type

사용자와 상호작용을 할 수 있는 버튼 등을 포함한 블록



# 2. Context Type

텍스트와 이미지를 포함한 블록

Field	Type	Required?	Description
<code>type</code>	String	Yes	The type of block. For a context block, <code>type</code> is always <code>context</code> .
<code>elements</code>	Object[]	Yes	An array of <code>image</code> elements and <code>text objects</code> . Maximum number of items is 10.
<code>block_id</code>	String	No	A string acting as a unique identifier for a block. If not specified, one will be generated. Maximum length for this field is 255 characters. <code>block_id</code> should be unique for each message and each iteration of a message. If a message is updated, use a new <code>block_id</code> .

## 2. Context Type

텍스트와 이미지를 포함한 블록



# 3. Divider Type

블록들을 나누는 평행선 블록

Field	Type	Required?	Description
<code>type</code>	String	Yes	The type of block. For a divider block, <code>type</code> is always <code>divider</code> .
<code>block_id</code>	String	No	A string acting as a unique identifier for a block. If not specified, one will be generated. Maximum length for this field is 255 characters. <code>block_id</code> should be unique for each message and each iteration of a message. If a message is updated, use a new <code>block_id</code> .

# 3. Divider Type

블록들을 나누는 평행선 블록



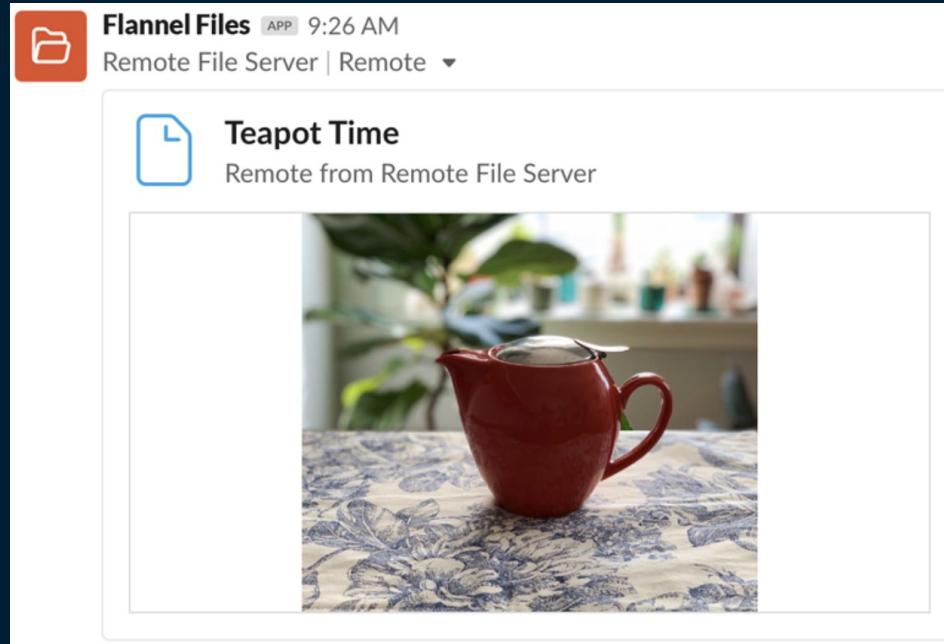
# 4. File Type

Remote File을 포함한 블록

Field	Type	Required?	Description
<code>type</code>	String	Yes	The type of block. For a file block, <code>type</code> is always <code>file</code> .
<code>external_id</code>	String	Yes	The external unique ID for this file.
<code>source</code>	String	Yes	At the moment, <code>source</code> will always be <code>remote</code> for a remote file.
<code>block_id</code>	String	No	A string acting as a unique identifier for a block. If not specified, one will be generated. Maximum length for this field is 255 characters. <code>block_id</code> should be unique for each message and each iteration of a message. If a message is updated, use a new <code>block_id</code> .

# 4. File Type

Remote File을 포함한 블록



# 5. Header Type

일반 텍스트보다 크고, 볼드체의 텍스트를 포함한 블록

Field	Type	Required?	Description
<code>type</code>	String	Yes	The type of block. For this block, type will always be <code>header</code> .
<code>text</code>	Object	Yes	The text for the block, in the form of a <code>plain_text</code> text object. Maximum length for the <code>text</code> in this field is 150 characters.
<code>block_id</code>	String	No	A string acting as a unique identifier for a block. If not specified, one will be generated. Maximum length for this field is 255 characters. <code>block_id</code> should be unique for each message and each iteration of a message. If a message is updated, use a new <code>block_id</code> .

# 5. Header Type

일반 텍스트보다 크고, 볼드체의 텍스트를 포함한 블록

HEADER TEXT

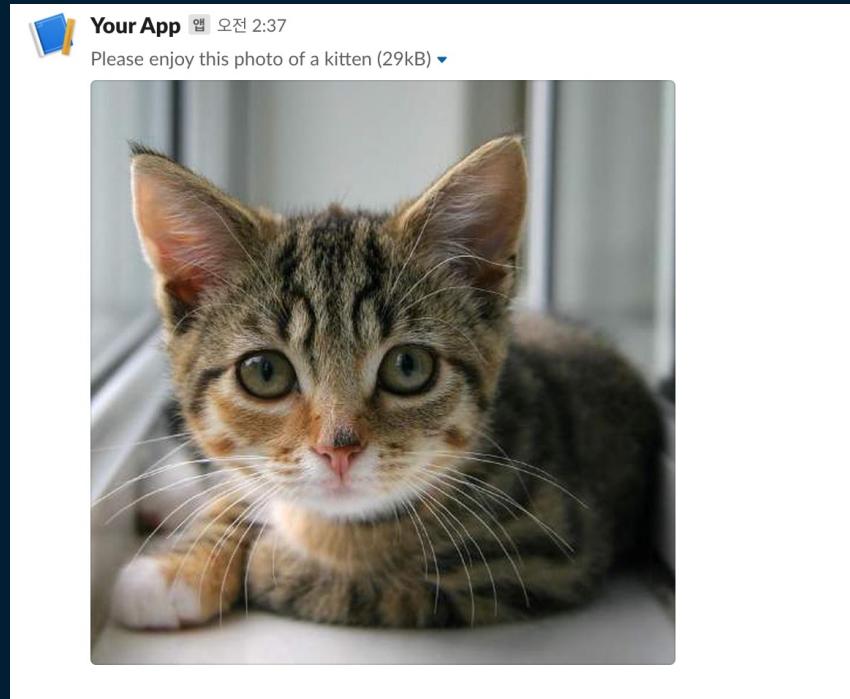
# 6. Image Type

## 이미지를 포함한 블록

Field	Type	Required?	Description
<code>type</code>	String	Yes	The type of block. For an image block, <code>type</code> is always <code>image</code> .
<code>image_url</code>	String	Yes	The URL of the image to be displayed. Maximum length for this field is 3000 characters.
<code>alt_text</code>	String	Yes	A plain-text summary of the image. This should not contain any markup. Maximum length for this field is 2000 characters.
<code>title</code>	Object	No	An optional title for the image in the form of a <code>text object</code> that can only be of <code>type: plain_text</code> . Maximum length for the <code>text</code> in this field is 2000 characters.
<code>block_id</code>	String	No	A string acting as a unique identifier for a block. If not specified, one will be generated. Maximum length for this field is 255 characters. <code>block_id</code> should be unique for each message and each iteration of a message. If a message is updated, use a new <code>block_id</code> .

# 6. Image Type

이미지를 포함한 블록



# 7. Input Type

사용자의 인풋을 받을 수 있는 블록

Field	Type	Required?	Description
<code>type</code>	String	Yes	The type of block. For an input block, <code>type</code> is always <code>input</code> .
<code>label</code>	Object	Yes	A label that appears above an input element in the form of a <code>text</code> object that must have <code>type</code> of <code>plain_text</code> . Maximum length for the <code>text</code> in this field is 2000 characters.
<code>element</code>	Object	Yes	An plain-text input element, a checkbox element, a radio button element, a select menu element, a multi-select menu element, or a datepicker.
<code>dispatch_action</code>	Boolean	No	A boolean that indicates whether or not the use of elements in this block should dispatch a <code>block_actions</code> payload. Defaults to <code>false</code> .
<code>block_id</code>	String	No	A string acting as a unique identifier for a block. If not specified, one will be generated. Maximum length for this field is 255 characters. <code>block_id</code> should be unique for each message or view and each iteration of a message or view. If a message or view is updated, use a new <code>block_id</code> .
<code>hint</code>	Object	No	An optional hint that appears below an input element in a lighter grey. It must be a a <code>text</code> object with a <code>type</code> of <code>plain_text</code> . Maximum length for the <code>text</code> in this field is 2000 characters.
<code>optional</code>	Boolean	No	A boolean that indicates whether the input element may be empty when a user submits the modal. Defaults to <code>false</code> .

# 7. Input Type

사용자의 인풋을 받을 수 있는 블록



## 피드백

×

텍스트

작성해주세요.

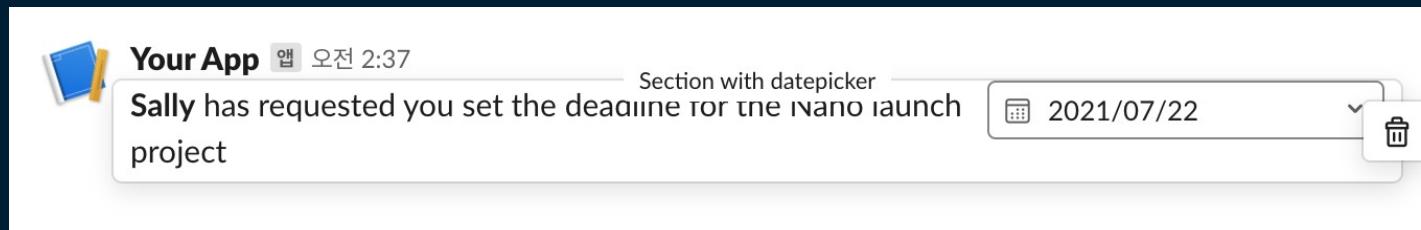
# 8. Section Type

가장 유연한 블록으로 이미지나 텍스트를 자유자재로 배치 가능한 블록

Field	Type	Required?	Description
<code>type</code>	String	Yes	The type of block. For a section block, type will always be <code>section</code> .
<code>text</code>	Object	Preferred	The text for the block, in the form of a <code>text object</code> . Maximum length for the <code>text</code> in this field is 3000 characters. This field is not required if a valid array of <code>fields</code> objects is provided instead.
<code>block_id</code>	String	No	A string acting as a unique identifier for a block. If not specified, one will be generated. You can use this <code>block_id</code> when you receive an interaction payload to <a href="#">identify the source of the action</a> . Maximum length for this field is 255 characters. <code>block_id</code> should be unique for each message and each iteration of a message. If a message is updated, use a new <code>block_id</code> .
<code>fields</code>	Object[]	Maybe	Required if no <code>text</code> is provided. An array of <code>text objects</code> . Any text objects included with <code>fields</code> will be rendered in a compact format that allows for 2 columns of side-by-side text. Maximum number of items is 10. Maximum length for the <code>text</code> in each item is 2000 characters. <a href="#">Click here for an example</a> .
<code>accessory</code>	Object	No	One of the available <code>element objects</code> .

# 8. Section Type

가장 유연한 블록으로 이미지나 텍스트를 자유자재로 배치 가능한 블록



# \* Rich Text

Text Type 을 마크다운으로 좀 더 자유로운 텍스트 또한 가능

```
1 | {  
2 |   "type": "section",  
3 |   "text": {  
4 |     "type": "mrkdwn",  
5 |     "text": "New Paid Time Off request from <example.com|Fred Enriquez>\n\n<https://example.com|View request>"  
6 |   }  
7 | }
```



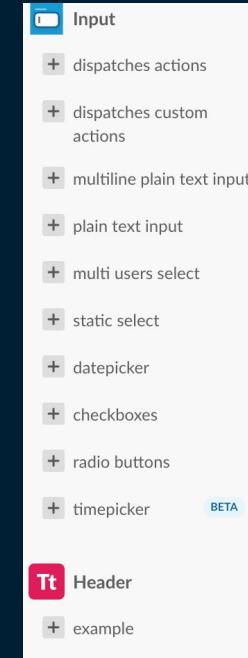
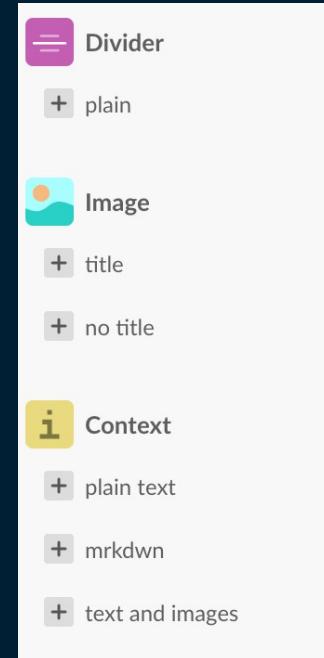
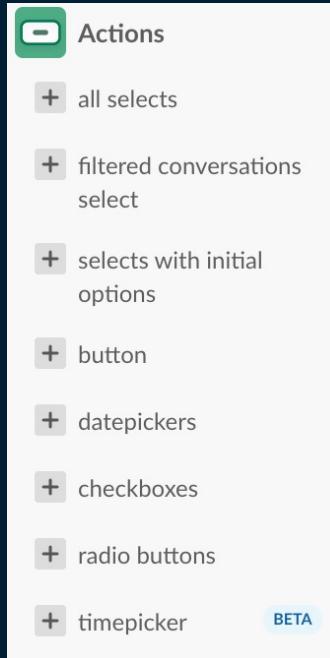
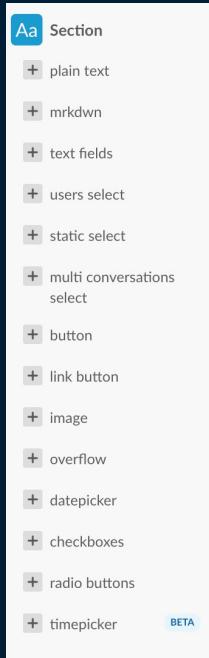
Your App 앱 오전 2:37

New Paid Time Off request from Fred Enriquez

[View request](#)

# GUI Block Building Kit

단순 클릭으로 block 자동 포맷 생성





04

DEMO:  
MOVIE BOT

# 영화 정보 챗봇

## 개요:

챗봇을 통해 현재 상영중인 영화에 대한 정보를 불러오고 이를 슬랙 인터페이스를 통해 유저에게 전달해주는 업무를 진행한다.

## 단계:

1. 네이버 영화로부터 현재 상영작에 대한 정보 불러오기
2. 슬랙 블록으로 변환하여 유저에게 보여주기
3. 사용자의 위치 정보 받기
4. 위치 정보를 기반으로 예약 페이지에 대한 링크 생성

# 크롤링

영화 상영정보 확인하기

<https://movie.naver.com/movie/running/current.nhn>

The screenshot shows the Naver Movie website's interface for checking movie showtimes. The left sidebar has a red header '상영작·예정작' (Running/Electric Movies). The main content area displays a grid of movie posters for four films: '발신제한' (Rating Restricted), '인 더 하이츠' (In the Heights), '크루엘라' (Cruella), and '킬러의 보디ガ드 2' (Killer's Bodyguard 2). Each movie card includes its title, rating, number of screenings, and opening date.

영화 제목	평점	상영장 수	개봉일
발신제한	★ ★ ★ ★ ☆	6.83	2021.06.23
인 더 하이츠	★ ★ ★ ★ ☆	8.57	2021.06.30
크루엘라	★ ★ ★ ★ ☆	9.26	2021.05.26
킬러의 보디ガ드 2	★ ★ ★ ☆ ☆	7.43	2021.06.23

# 크로링

## REFERENCE : 1\_crawling.py

```
1 # 1_crawling.py
2 # 네이버 실시간 영화 정보를 불러오는 코드를 작성한다.
3
4 # pip install beautifulsoup4
5 import requests
6 from bs4 import BeautifulSoup
7
8 URL = "https://movie.naver.com/movie/running/current.nhn"
9
10 # 영화 정보를 불러와 포맷에 맞춰 return 해주는 함수
11 def get_current_movies():
12     # 1. URL로부터 html을 파싱해오기
13     req = requests.get(URL)
14     html = req.text
15     soup = BeautifulSoup(html, 'html.parser')
16
17 # 메인함수
18 if __name__ == "__main__":
19     get_current_movies()
```

```
> <!DOCTYPE html>
> <html lang="ko">
>   <div style="display: none;">...</div>
>   <head>...</head>
>   <body>
>     <div id="wrap" class="basic">
>       <!-- GNB -->
>       <script type="text/javascript">...</script>
>       <script type="text/javascript">...</script>
>       <!-- skip navigation -->
>       <div id="u_skip">...</div>
>       <!-- //skip navigation -->
>       <div class="gnb_container">...</div>
>       <!-- //GNB -->
>       <!-- header -->
>       <div id="header" style="bottom: 0px;">...</div>
>       <script type="text/javascript">...</script>
>       <!-- //header -->
>       <!-- container -->
>     <div id="container">
>       <!-- 프로모션 레이어 -->
>       <div class="ly_promo" id="thumb_promotion_layer">...
>     </div>
>       <!-- 상단 현재 상영장 챔피언 리스트 -->
>       <div class="spot">...</div>
>       <!-- //상단 현재 상영장 챔피언 리스트 -->
>       <script type="text/template" id="_TopAreaThumbnailExpan...
>       <script type="text/javascript">...</script>
>       <!-- content -->
>     <div id="content">
>       <div class="article">
>         <div class="obj_section">
>           <h3 class="tit_t1">...</h3>
>           <div class="view_sorting">...</div>
>           <div class="tab_t1">...</div>
>           <h4 class="blind">상영영화</h4>
>           <div class="lst_wrap">
>             <ul class="lst_detail_t1">
>               <li>...</li> == $0
>               <li>...</li>
>             </ul>
>           </div>
>         </div>
>       </div>
>     </div>
>   </body>
> </html>
```

크롤링

1. F12 혹은 우클릭 > 페이지 소스 검사
  2. 영화들 정보가 있는 부분 찾기

=> lst\_detail\_t1 이라는 ul 안에 li 형태로 영화들 정보가 담겨져 있음

# 크롤링

NAVER 영화

영화홈

상영작·예정작

현재 상영영화

개봉 예정영화

예고편

영화랭킹

예매

평점·리뷰

다운로드

인디극장

상영작 예매 순위

원하시는 영화를 선택해주세요. 회자기 되고 있는 최신영화 정보를 한 눈에 확인하실 수 있습니다.

665 x 171 영화제

15 발신자제한

네이버 ★★★★☆ ★ 6.83 참여 1,682명 | 예매율 18.59%

개요 드라마, 스릴러 94분 | 2021.06.23 개봉

감독 김창주

출연 조우진, 이재인, 진경

예매하기 포토보기 예고편 메이킹

12 인더 하이츠

네이버 ★★★★★ 8.57 참여 342명 | 예매율 14.34%

개요 유지컬 | 142분 | 2021.06.30 개봉

감독 존 추

출연 앤소니 라모스, 멜리사 바레사, 코리 호킨스, 레슬리 그레이스, 스테파니 비트리즈, 린-마누엘 미란다

예매하기 포토보기 예고편 메이킹

Elements Application Console Sources Network Performance Styles Computed Layout

Overflow: hidden; padding: 15px 0 15px;

Copy

Copy element

Copy outerHTML

Copy selector

Copy JS path

Copy styles

Copy XPath

Copy full XPath

=> #content > div.article > div:nth-child(1) > div.lst\_wrap > ul > li:nth-child(1)

# 크롤링

## REFERENCE : 1\_crawling.py

```
10 # 영화 정보를 불러와 포맷에 맞춰 return 해주는 함수
11 def get_current_movies():
12     # 1. URL로부터 html을 파싱해오기
13     req = requests.get(URL)
14     html = req.text
15     soup = BeautifulSoup(html, 'html.parser')
16     # 2. 영화 부분 추출하기
17     lis = soup.select('#content > div.article > div:nth-of-type(1) > div.lst_wrap > ul > li')
```

=> **#content > div.article > div:nth-child(1) > div.lst\_wrap > ul > li:nth-child(1)**

\*\* nth-child를 nth-of-type으로 바꿔줘야 함

\*\* 모든 li를 불러오기 위해 마지막 nth-child는 삭제

# 크롤링

## REFERENCE:1\_crawling.py

```
# 3. 필요한 정보만 추출
movies = []
for idx, li in enumerate(lis):
    temp = {}
    temp['title'] = li.select('dl > dt > a')[0].text
    temp['score'] = float(li.select('dl > dd.star > dl.info_star > dd > div > a > span.num')[0].text)
    detail = li.select('.link_txt') # 개요, 감독, 출연
    temp['genre'] = [x.text for x in detail[0].select('a')]
    temp['director'] = [x.text for x in detail[1].select('a')]
    # 애니메이션처럼 배우 정보가 없는 경우가 있음
    if len(detail) > 2:
        temp['actors'] = [x.text for x in detail[2].select('a')]
    movies.append(temp)
```

# 크롤링

## REFERENCE: 1\_crawling.py

```
1 # 1_crawling.py
2 # 네이버 실시간 영화 정보를 불러오는 코드를 작성한다.
3
4 # pip install beautifulsoup4
5 import requests
6 from bs4 import BeautifulSoup
7
8 URL = "https://movie.naver.com/movie/running/current.nhn"
9
10 # 영화 정보를 불러와 딕셔너리 형태에 맞춰 리스트로 return 해주는 함수
11 def get_current_movies():
12     # 1. URL로부터 html을 파싱해오기
13     req = requests.get(URL)
14     html = req.text
15     soup = BeautifulSoup(html, 'html.parser')
16     # 2. 영화 부분 추출하기
17     lis = soup.select('#content > div.article > div:nth-of-type(1) > div.lst_wrap > ul > li')
18     # 3. 필요한 정보만 추출
19     movies = []
20     for idx, li in enumerate(lis):
21         temp = {}
22         temp['title'] = li.select('dl > dt > a')[0].text
23         temp['score'] = float(li.select('dl > dd.star > dl.info_star > dd > div > a > span.num')[0].text)
24         detail = li.select('.link_txt') # 개요, 감독, 출연
25         temp['genre'] = [x.text for x in detail[0].select('a')]
26         temp['director'] = [x.text for x in detail[1].select('a')]
27         # 애니메이션처럼 배우 정보가 없는 경우가 있음
28         if len(detail) > 2:
29             temp['actors'] = [x.text for x in detail[2].select('a')]
30         movies.append(temp)
31     return movies
32
33 # 메인함수
34 if __name__ == "__main__":
35     movies = get_current_movies()
36     print(movies[0])
```



```
{
    'title': '보이스',
    'score': 8.55,
    'genre': [
        '범죄',
        '액션'
    ],
    'director': [
        '김선',
        '김곡'
    ],
    'actors': [
        '변요한',
        '김무열',
        '김희원',
        '박명훈'
    ]
}
```

# 블록 구상

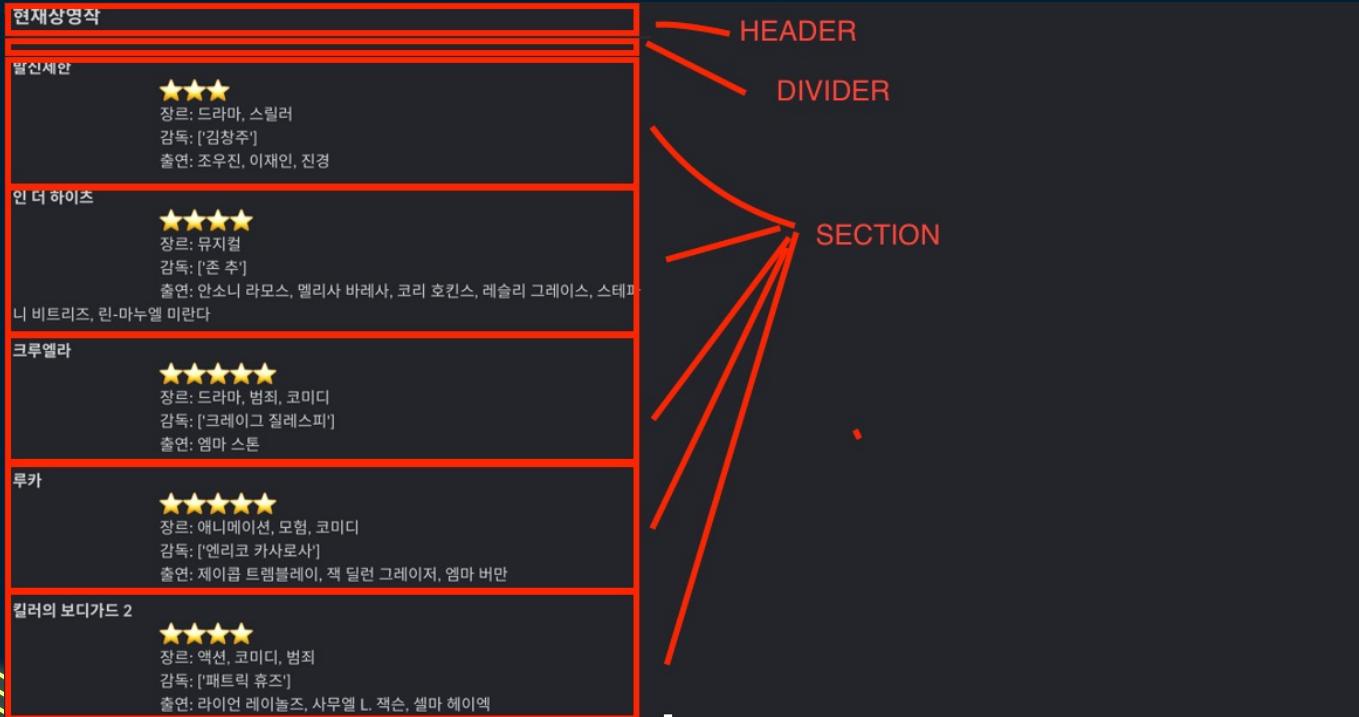
1. <https://app.slack.com/block-kit-builder/>
2. 빈 블록 생성

The screenshot shows the Slack Block Kit Builder interface. On the left, there's a sidebar with various block types: Section, plain text, mrkdwn, text fields, users select, static select, multi conversations select, button, and link button. The 'Section' block is currently selected. In the center, there's a message preview window titled 'Your App' with a timestamp '오전 2:37'. The message content is '0| 메시지를 완전히 표시할 수 없습니다.' (0| You cannot fully display the message). Above the message preview are three dropdown menus: 'Message Preview' (set to 'Desktop'), 'Desktop' (set to 'Light'), and 'Copy Payload' (button), 'Clear Blocks' (button), and 'slack chatbot' (dropdown). To the right of the message preview are 'Templates' (button), 'Payload' (selected tab), and 'Actions Preview' (button). The 'Payload' tab displays the following JSON code:

```
1 {  
2   "blocks": []  
3 }
```

# 블록 구상

## 블록 레이아웃 구상



# 블록 생성

## Block Kit Builder로 블록 포맷 확인

The screenshot shows the Slack Block Kit Builder interface. On the left, there's a sidebar with various block types: plain text, mrkdwn, text fields, users select, static select, multi conversations select, button, link button, image, overflow, datepicker, checkboxes, and radio buttons. The main area displays a message preview from 'Your App' at 2:37 PM. The message contains a header block with the text "This is a header block". Below it is a mrkdwn section block with the text "This is a mrkdwn section block", followed by bolded text ("this is bold"), crossed-out text ("this is crossed out"), and a link ("this is a link"). To the right, there are tabs for 'Templates' and 'Payload'. The 'Payload' tab shows the generated JSON code:

```
1 {
2   "blocks": [
3     {
4       "type": "header",
5       "text": {
6         "type": "plain_text",
7         "text": "This is a header block",
8         "emoji": true
9       }
10    },
11    {
12      "type": "divider"
13    },
14    {
15      "type": "section",
16      "text": {
17        "type": "mrkdwn",
18        "text": "This is a mrkdwn section block :ghost: *this is bold*, and ~this is crossed out~, and <https://google.com>this is a link"
19      }
20    }
21  ]
22 }
```

=> 좌측에서 header, divider, section (mrkdwn) 클릭한 결과

# 블록 생성

## 우측 블록 리스트 확인

```
1 {
2   "blocks": [
3     {
4       "type": "header",
5       "text": {
6         "type": "plain_text",
7         "text": "This is a header block",
8         "emoji": true
9       }
10      },
11      {
12        "type": "divider"
13      },
14      {
15        "type": "section",
16        "text": {
17          "type": "mrkdwn",
18          "text": "This is a mrkdwn section block :ghost: *this is bold*, and ~this is crossed out~, and
<https://google.com!this is a link>"
19        }
20      }
21    ]
22 }
```

# 블록 생성

```
1 {
2   "blocks": [
3     {
4       "type": "header",
5       "text": {
6         "type": "plain_text",
7         "text": "This is a header block",
8         "emoji": true
9       }
10    },
11    {
12      "type": "divider"
13    },
14    {
15      "type": "section",
16      "text": {
17        "type": "mrkdwn",
18        "text": "This is a mrkdwn section block :ghost: *this is bold*, and ~this is crossed out~, and
<https://google.com|this is a link>"
19      }
20    }
21  ]
22 }
```

## Block 1:

- Type: header
- Text:
  - Type: plain\_text
  - Text: string
  - Emoji: (optional)

## Block 2:

- Type: divider

## Block 3:

- Type: section
- Text:
  - Type: mrkdwn
  - Text: string

# 블록 생성

## 블록 담을 리스트 생성

```
# 블록 담을 리스트  
block = []
```

# 블록 생성

## 헤더 블록 쌓기

1. 빈 dictionary 생성
2. Key 별로 value 지정 ex. key: type => value: header
3. Dictionary 내 dictionary가 있는 경우 마찬가지로 dictionary 생성 후 키 지정
4. 블록 리스트에 추가

Block 1:

- Type: header
- Text:
  - Type: plain\_text
  - Text: string
  - Emoji: (optional)



```
# 헤더 블록
header = {}
header['type'] = 'header'
header['text'] = {}
header['text']['type'] = 'plain_text'
header['text']['text'] = '현재상영작'
block.append(header)
```

# 블록 생성

## Divider 블록 쌓기

1. 빈 dictionary 생성
2. Key 별로 value 지정
3. 블록 리스트에 추가

Block 2:  
• Type: divider



```
# divider 블록
divider = {}
divider['type'] = 'divider'
block.append(divider)
```

# 블록 생성

## Section 블록 쌓기

1. 영화 object별로 빈 dictionary 생성
2. Key-value pair 지정
3. Markdown 포맷에 맞게 영화 object 값들 지정
4. 블록 리스트에 추가

Block 3:

- Type: section
- Text:
  - Type: mrkdwn
  - Text: string

# 블록 생성

## Section 블록 쌓기

### 영화 Object

```
{  
  'title': '보이스',  
  'score': 8.55,  
  'genre': [  
    '범죄',  
    '액션'  
,  
  'director': [  
    '김선',  
    '김곡'  
,  
  'actors': [  
    '변요한',  
    '김무열',  
    '김희원',  
    '박명훈'  
,  
  ]  
}
```

\*\*발신제한\*\*  
:star::star::star::star::star:  
장르 : 드라마, 스릴러  
감독: 김창주  
출연: 조우진, 이재인, 진경

□ 발신제한  
□ ★★★★★★  
◎ 장르 : 드라마, 스릴러  
◎ 감독: 김창주  
◎ 출연: 조우진, 이재인, 진경

좌: markdown string  
우: 아웃풋

\*StackEdit에서 Markdown 확인 가능

# 블록 생성

## Section 블록 쌓기

```
**{movie['title']}**
{"star:" * (movie['score']/2)}
장르 : {'.'join(movie['genre'])}
감독 : {'.'join(movie['director'])}
출연 : {'.'join(movie['actors'])}
```

## 변수로 치환

```
{movie['title']}
{"★" * (movie['score']/2)}
장르 : {'.'join(movie['genre'])}
감독 : {'.'join(movie['director'])}
출연 : {'.'join(movie['actors'])}
```

# 블록 생성

## Section 블록 쌓기

```
# 영화 블록
for movie in movies:
    temp = {}
    temp['type'] = 'section'
    temp['text'] = {}
    temp['text']['type'] = 'mrkdwn'
    stars = ":star:" * round(movie['score']/2)
    temp['text']['text'] = f"*{movie['title']}*\\n \
                            {stars}\\n \
                            장르: {' , '.join(movie['genre'])}\\n \
                            감독: {movie['director']}\\n \
                            출연: {' , '.join(movie['actors'])}\\n"
    block.append(temp)
```

# 블록 생성

## REFERENCE : 2\_display\_movies.py

```
55 def build_blocks(movies: list):
56     # 블록 담을 리스트
57     block = []
58     # 헤더 블록
59     header = {}
60     header['type'] = 'header'
61     header['text'] = {}
62     header['text']['type'] = 'plain_text'
63     header['text']['text'] = '현재상영작'
64     block.append(header)
65     # divider 블록
66     divider = {}
67     divider['type'] = 'divider'
68     block.append(divider)
69     # 영화 블록
70     for movie in movies:
71         temp = {}
72         temp['type'] = 'section'
73         temp['text'] = {}
74         temp['text']['type'] = 'mrkdn'
75         stars = ":star:" * round(movie['score']/2)
76         temp['text']['text'] = f"*{movie['title']}*\n \
77                                     {stars}\n \
78                                     장르: {', '.join(movie['genre'])}\n \
79                                     감독: {movie['director']}\n \
80                                     출연: {', '.join(movie['actors'])}\n"
81         block.append(temp)
82     return block
83
```

# 서버 설정

## REFERENCE: 2\_display\_movies.py

```
1 # 2_display_movies.py
2 # 크롤링한 영화 정보를 포맷에 맞춰 슬랙에 보내주는 역할
3
4 import json
5
6 # pip install Flask
7 from flask import Flask, request, make_response
8 from handler import slack_handler
9
10 crawling = __import__('1_crawling')
```

# 서버 설정

## REFERENCE : 2\_display\_movies.py

```
25      # slack에서 발생한 event를 통한 request에 대한 핸들링
26      if "event" in slack_event:
27          event_type = slack_event["event"]["type"]
28          # bot_mention일 경우에 대한 핸들링
29          if event_type == 'app_mention':
30              try:
31                  # 멘션을 남긴 채널 읽어오기
32                  channel = slack_event['event']['channel']
33                  # 유저가 멘션과 함께 남긴 텍스트 읽어오기
34                  user_query = slack_event['event']['blocks'][0]['elements'][0]['elements'][1]['text']
35
36                  if user_query.strip() == '/영화정보':
37                      movies = crawling.get_current_movies()
38                      block = build_blocks(movies)
39                      slack_handler.post_layout_message(blocks=block, channel=channel)
40
41                  # 정상적으로 완료했음에 대한 http response
42                  return make_response("response made :)", 200, )
43
44          except IndexError:
45              # 멘션은 했지만 텍스트는 남기지 않은 경우에 대한 에러.
46              # do nothing
47              pass
```

# 서버 설정

## REFERENCE : 2\_display\_movies.py

```
55  def build_blocks(movies: list):
56      # 블록 담을 리스트
57      block = []
58      # 헤더 블록
59      header = {}
60      header['type'] = 'header'
61      header['text'] = {}
62      header['text']['type'] = 'plain_text'
63      header['text']['text'] = '현재상영작'
64      block.append(header)
65      # divider 블록
66      divider = {}
67      divider['type'] = 'divider'
68      block.append(divider)
69      # 영화 블록
70      for movie in movies:
71          temp = {}
72          temp['type'] = 'section'
73          temp['text'] = {}
74          temp['text']['type'] = 'mrkdwn'
75          stars = ":star:" * round(movie['score']/2)
76          temp['text']['text'] = f"*{movie['title']}*\n \
77                                      {stars}\n \
78                                      장르: {'', '.join(movie['genre'])}\n \
79                                      감독: {movie['director']}\n \
80                                      출연: {'', '.join(movie['actors'])}\n"
81          block.append(temp)
82      return block
```

# 서버 수정

결과:

@myChatBot /영화정보

오늘

myChatBot 오전 4:38

현재상영작

보이스

★★★★★ 8.55  
장르: 범죄, 액션  
감독: 김선, 김곡  
출연: 변요한, 김무열, 김희원, 박명훈

극장판 포켓몬스터: 정글의 아이, 코코

★★★★★ 8.94  
장르: 애니메이션, 모험  
감독: 아지마 태초오  
출연: 마츠모토 리카, 오오타니 이쿠에, 하야시바라 메구미, 미키 신이치로

상치와 텐 릴즈의 전설

★★★★★ 6.6  
장르: 액션, 모험, 판타지  
감독: 데스틴 크리튼  
출연: 시무 리우, 양조위, 아콰피나

기적

★★★★★ 9.23  
장르: 드라마  
감독: 이장훈  
출연: 박정민, 이성민, 윤아, 이수경

# SLACK API 설정

## User Interaction 추가

The screenshot shows the Slack API Settings interface for an app named "myChatBot". The left sidebar has sections for "Settings" (Basic Information, Collaborators, Socket Mode, Install App, App Manifest, Manage Distribution) and "Features" (App Home, Org Level Apps, Incoming Webhooks, **Interactivity & Shortcuts**, Slash Commands, Workflow Steps, OAuth & Permissions, Event Subscriptions, User ID Translation, Beta Features, Where's Bot User). The "Interactivity & Shortcuts" section is selected and highlighted with a red box. It contains a toggle switch labeled "On" (also highlighted with a red box), a "Request URL" input field containing "https://1467ae3ac2c6.ngrok.io/user-select" (also highlighted with a red box), and a "Shortcuts" section with a table header for Name, Location, and Callback ID, and a "Create New Shortcut" button. At the bottom, there are "Discard Changes" and "Save Changes" buttons.

⇒ Request URL: <Forwarding URL> + "/user-select"  
⇒ ex. <https://ec22-118-33-100-66.ngrok.io/user-select>

# Interaction 추가

## REFERENCE : info.json

```
1  {
2      "서울" : {
3          "code": "01",
4          "theaters" : {
5              "CGV강남" : "0056",
6              "CGV강변" : "0001",
7              "CGV건대입구" : "0229",
8              "CGV구로" : "0010",
9              "CGV대학로" : "0063",
10             "CGV동대문" : "0252",
11             "CGV등촌" : "0230",
12             "CGV명동" : "0009",
13             "CGV명동역 씨네라이브러리" : "00105",
14             "CGV목동" : "0011",
15             "CGV미아" : "0057",
16             "CGV불광" : "0030"
17         },
18     },
19     "경기" :{
20         "code" : "02"
21     },
22     "인천" :{
23         "code" : "03"
24     },
25     "강원" :{
26         "code" : "04"
27     },
28     "대전/충청" :{
29         "code" : "05"
30     },
31     "대구" :{
32         "code" : "06"
33     },
34     "부산/울산" :{
35         "code" : "07"
36     },
37     "경상" :{
38         "code" : "08"
39     },
40     "광주/전라/제주" :{
41         "code" : "09"
42     }
43 }
```



# Interaction 추가

## REFERENCE : 3\_user\_interaction.py

```
30     # slack에서 발생한 event를 통한 request에 대한 핸들링
31     if "event" in slack_event:
32         event_type = slack_event["event"]["type"]
33         # bot_mention일 경우에 대한 핸들링
34         if event_type == 'app_mention':
35             try:
36                 # 멘션을 남긴 채널 읽어오기
37                 channel = slack_event['event']['channel']
38                 # 유저가 멘션과 함께 남긴 텍스트 읽어오기
39                 user_query = slack_event['event']['blocks'][0]['elements'][0]['elements']
40
41                 if user_query.strip() == '/영화정보':
42                     movies = crawling.get_current_movies()
43                     block = build_blocks(movies)
44                     slack_handler.post_layout_message(blocks=block, channel=channel)
45
46                 if user_query.strip() == '/상영정보':
47                     block = build_select_block()
48                     slack_handler.post_layout_message(blocks=block, channel=channel)
49
50             # 정상적으로 완료했음에 대한 http response
51             return make_response("response made :)", 200, )
52         except IndexError:
53             # 멘션은 했지만 텍스트는 남기지 않은 경우에 대한 에러.
54             # do nothing
55             pass
56         # 그 외 event에 대한 핸들링: 404 error
57         msg = f"[{event_type}] cannot find event handler"
58         return make_response(msg, 404, {"X-Slack-No-Retry": 1 })
59
60
61     # 그 외 request 핸들링: 404 error
62     return make_response("No Slack request events", 404, {"X-Slack-No-Retry": 1 })
```

# Interaction 추가

## REFERENCE: 3\_user\_interaction.py

```
93 def build_select_block():
94     # 블록 담을 리스트
95     block = []
96     # 헤더 블록
97     header = {}
98     header['type'] = 'header'
99     header['text'] = {}
100    header['text'][ 'type'] = 'plain_text'
101    header['text'][ 'text'] = '지역을 선택해주세요.'
102    block.append(header)
103    # divider 블록
104    divider = {}
105    divider['type'] = 'divider'
106    block.append(divider)
107    # 지역 블록
108    options = json.load(open('info.json', 'rb'), encoding='CP949')
109    input_block = {}
110    input_block[ 'type'] = "input"
111    input_block[ 'element'] = {}
112    input_block[ 'element'][ 'type'] = 'static_select'
113    input_block[ 'element'][ 'placeholder'] = {}
114    input_block[ 'element'][ 'placeholder'][ 'type'] = "plain_text"
115    input_block[ 'element'][ 'placeholder'][ 'text'] = '지역을 선택해주세요.'
116    input_block[ 'element'][ 'options'] = []
117    for area in options:
118        temp = {}
119        temp[ 'text'] = {}
120        temp[ 'text'][ 'type'] = "plain_text"
121        temp[ 'text'][ 'text'] = area
122        temp[ 'value'] = options[area][ 'code']
123        input_block[ 'element'][ 'options'].append(temp)
124    input_block[ 'element'][ 'action_id'] = 'static_select-action'
125    input_block[ 'label'] = {}
126    input_block[ 'label'][ 'type'] = 'plain_text'
127    input_block[ 'label'][ 'text'] = "지역"
128    block.append(input_block)
129    print(block)
130    return block
```

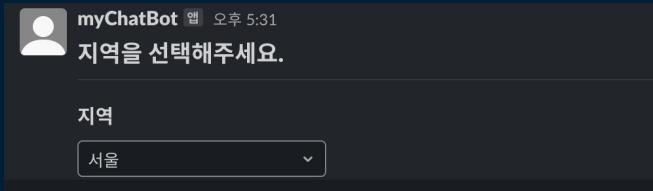
# Interaction 추가

REFERENCE: 3\_user\_interaction.py

```
18     @app.route('/user-select', methods=['POST', 'GET'])
19     def handle_interaction():
20         slack_event = json.loads(request.form['payload'])
21         return make_response("response made :)", 200, )
```

# Interaction 추가

Interaction이 생겼을 때 설정한 url로 날아오는 request 확인



```
{  
    'type': 'block_actions',  
    'user': {  
        'id': 'U025BPD5K8E',  
        'username': 'hyun.kang',  
        'name': 'hyun.kang',  
        'team_id': 'T0257XTXDBC'  
    },  
    'api_app_id': 'A025783H9P50',  
    'token': 'fd54wao3UD81uqNtVANXg0W',  
    'container': {  
        'type': 'message',  
        'message_ts': '1625301082.006800',  
        'channel_id': 'C026AN94Q2U',  
        'is_ephemeral': False  
    },  
    'trigger_id': '2238142057762.2197269931284.23d09184d1ccff05fb95c7406a0balc5',  
    'team': {  
        'id': 'T0257XTXDBC',  
        'domain': 'the-slack-chatbot',  
        'enterprise': None,  
        'is_enterprise_install': False,  
        'channel': {  
            'id': 'C026AN94Q2U',  
            'name': '일반'  
        },  
        'message': {  
            'bot_id': 'B025T8AULNQ',  
            'type': 'message',  
            'text': '지역을 선택해주세요.',  
            'user': 'U025M2A1TSW',  
            'ts': '1625301082.006800',  
            'team': 'T0257XTXDBC',  
            'blocks': [  
                {  
                    'type': 'header',  
                    'block_id': 'B030',  
                    'text': {  
                        'type': 'plain_text',  
                        'text': '지역을 선택해주세요.',  
                        'emoji': True  
                    },  
                    'divider': {  
                        'block_id': 'x8FwA',  
                    },  
                    'input': {  
                        'block_id': 'yefw5',  
                        'label': {  
                            'type': 'plain_text',  
                            'text': '지역',  
                            'emoji': True  
                        },  
                        'optional': False,  
                        'dispatch_action': False,  
                        'static_select': {  
                            'type': 'static_select',  
                            'action_id': 'static_select-action',  
                            'placeholder': {  
                                'type': 'plain_text',  
                                'text': '지역을 선택해주세요.',  
                                'emoji': True  
                            }  
                        }  
                    }  
                }  
            ]  
        }  
    }  
}
```

```
,  
    'text': {  
        'type': 'plain_text',  
        'text': '경기',  
        'emoji': True  
    },  
    'value': '08'  
},  
{  
    'text': {  
        'type': 'plain_text',  
        'text': '광주(전라)제주',  
        'emoji': True  
    },  
    'value': '09'  
}  
]  
}  
}  
},  
'state': {  
    'values': {  
        'yefw5': {  
            'static_select-action': {  
                'type': 'static_select',  
                'selected_option': {  
                    'text': '서울',  
                    'type': 'plain_text',  
                    'text': '서울',  
                    'emoji': True  
                },  
                'value': '01'  
            }  
        }  
    }  
},  
'response_url': 'https://hooks.slack.com/actions/T0257XTXDBC/A025783H9P50/yefw5',  
'actions': [  
    {  
        'type': 'static_select',  
        'action_id': 'static_select-action',  
        'block_id': 'yefw5',  
        'select_ted_option': {  
            'text': {  
                'type': 'plain_text',  
                'text': '서울',  
                'emoji': True  
            },  
            'value': '01'  
        },  
        'placeholder': {  
            'type': 'plain_text',  
            'text': '지역을 선택해주세요.',  
            'emoji': True  
        },  
        'action_ts': '1625301311.497506'  
    }  
]
```

# Interaction 추가

## Interaction Handling 추가

REFERENCE: 3\_user\_interaction.py

```
14     areaCode = ''
15     theaterCode = ''
16
18     @app.route('/user-select', methods=['POST', 'GET'])
19     def handle_interaction():
20         slack_event = json.loads(request.form['payload'])
21         # 만약 지역 선택 블록에서 interaction request가 온 경우
22         if slack_event['actions'][0]['placeholder']['text'] == '지역을 선택해주세요.':
23             area = slack_event['actions'][0]['selected_option']['text']['text']
24             areaCode = slack_event['actions'][0]['selected_option']['value']
25             block = build_theater_block(area)
26             slack_handler.post_layout_message(blocks=block, channel=slack_event['channel']['id'])
27
28         return make_response("response made :)", 200, )
```

# Interaction 추가

## Interaction Handling 추가

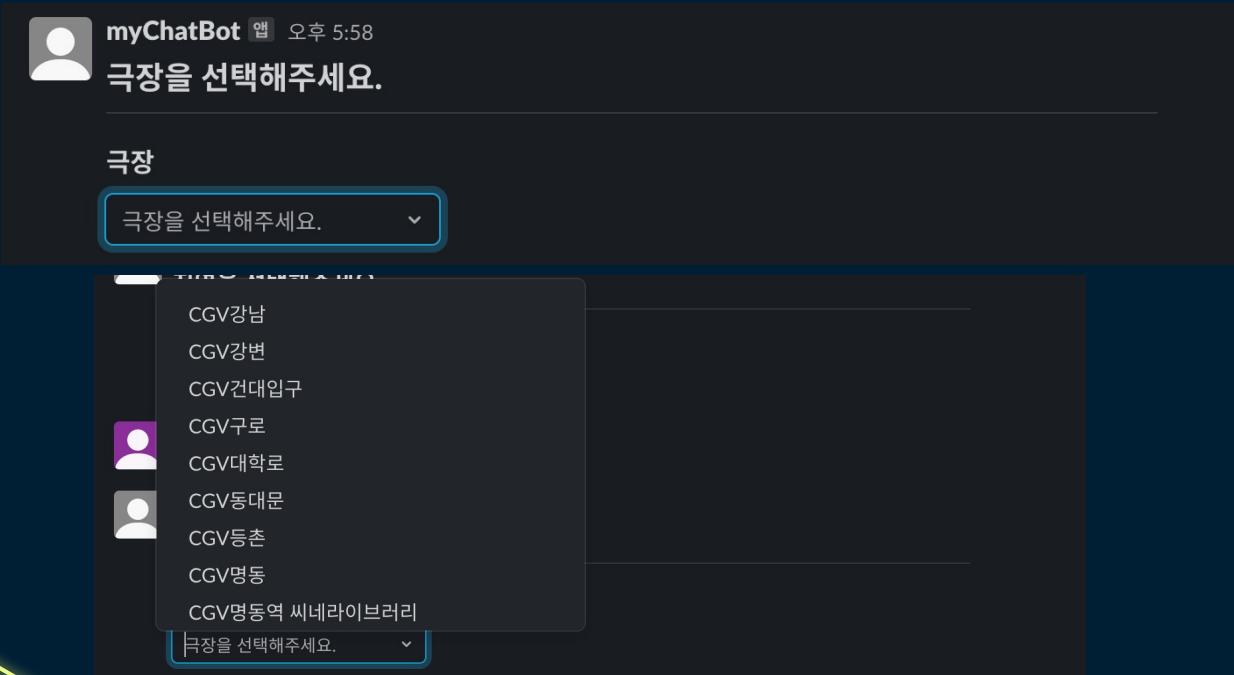
### REFERENCE : 3\_user\_interaction.py

```
141 def build_theater_block(area):
142     # 블록 담을 리스트
143     block = []
144     # 헤더 블록
145     header = {}
146     header['type'] = 'header'
147     header['text'] = {}
148     header['text'][type] = 'plain_text'
149     header['text'][text] = '극장을 선택해주세요.'
150     block.append(header)
151     # 디바이저 블록
152     divider = {}
153     divider[type] = 'divider'
154     block.append(divider)
155     # 영화관 블록
156     options = json.load(open('info.json', 'rb'), encoding='CP949')
157     input_block = {}
158     input_block[type] = "input"
159     input_block[element] = {}
160     input_block[element][type] = 'static_select'
161     input_block[element][placeholder] = {}
162     input_block[element][placeholder][type] = "plain_text"
163     input_block[element][placeholder][text] = '극장을 선택해주세요.'
164     input_block[element][options] = []
165     print(area)
166     for theater, value in options[area]['theaters'].items():
167         temp = {}
168         temp['text'] = {}
169         temp['text'][type] = "plain_text"
170         temp['text'][text] = theater
171         temp[value] = value
172         input_block[element][options].append(temp)
173     input_block[element][action_id] = 'static_select-action'
174     input_block[label] = {}
175     input_block[label][type] = 'plain_text'
176     input_block[label][text] = "극장"
177     block.append(input_block)
178
179 return block
```

# Interaction 추가

## Interaction Handling 추가

결과:



# Interaction 추가

## Interaction Handling 추가

### REFERENCE : 3\_user\_interaction.py

```
18 @app.route('/user-select', methods=['POST', 'GET'])
19 def handle_interaction():
20     slack_event = json.loads(request.form['payload'])
21     print(slack_event)
22     # 만약 지역 선택 블록에서 interaction request가 온 경우
23     if slack_event['actions'][0]['placeholder']['text'] == '지역을 선택해주세요.':
24         area = slack_event['actions'][0]['selected_option']['text']['text']
25         areaCode = slack_event['actions'][0]['selected_option']['value']
26         block = build_theater_block(area)
27         slack_handler.post_layout_message(blocks=block, channel=slack_event['channel']['id'])
28
29     # 만약 극장 선택 블록에서 interaction request가 온 경우
30     if slack_event['actions'][0]['placeholder']['text'] == '극장을 선택해주세요.':
31         theaterCode = slack_event['actions'][0]['selected_option']['value']
32         print(theaterCode)
33
34     return make_response("response made :)", 200, )
```

# Interaction 추가

## Interaction Handling 추가

REFERENCE : 3\_user\_interaction.py

```
31
32     # 만약 극장 선택 블록에서 interaction request가 온 경우
33     if slack_event['actions'][0]['placeholder']['text'] == '극장을 선택해주세요.':
34         date = datetime.datetime.now().strftime("%Y%m%d")
35         global theaterCode
36         theaterCode = slack_event['actions'][0]['selected_option']['value']
37         slack_handler.post_slack_message(channel=slack_event['channel']['id'], message=f'http://www.cgv.co.kr/reserve/\n    show-times/?areacode={areaCode}&theaterCode={theaterCode}&date={date}' )
38
39
```

# Interaction 추가

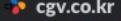
myChatBot 웹 오후 11:31  
지역을 선택해주세요.

지역

극장을 선택해주세요.

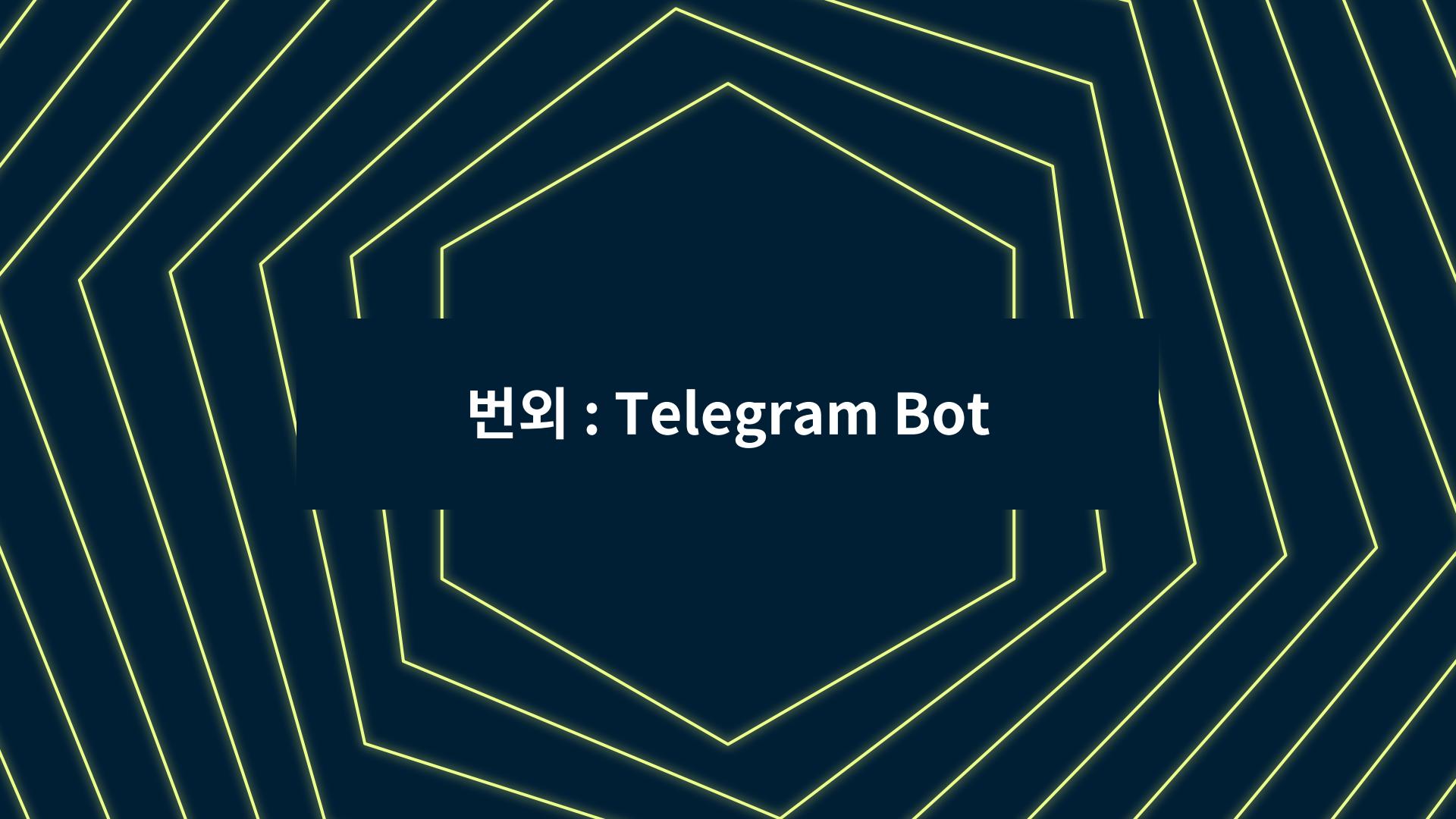
극장

myChatBot 웹 오후 11:48  
<http://www.cgv.co.kr/reserve/show-times/?areacode=&theaterCode=0230&date=20210703> (편집됨)

 [cgv.co.kr](http://www.cgv.co.kr)  
영화 그 이상의 감동. CGV

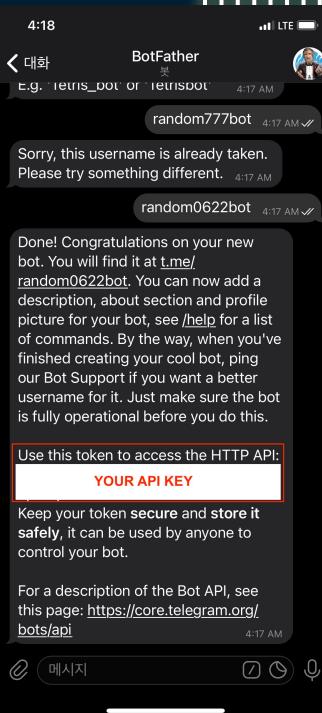
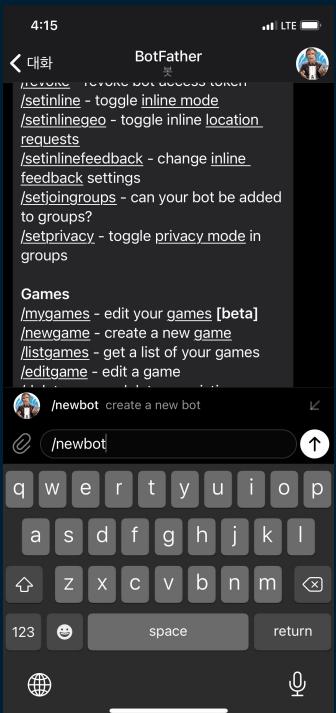
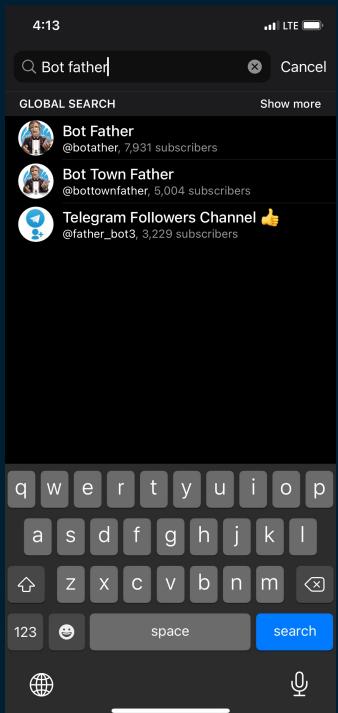
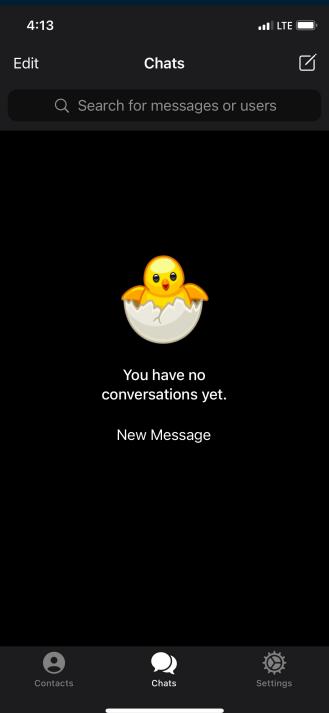
CGV는 선진화된 관람문화와 최고의 서비스로 고객에게 잊을 수 없는 감동을 선사합니다.  
CGV홈페이지를 통해 영화 예매뿐만 아니라 그 이상의 서비스와 감동을 전달하고, 다양한 즐거움과 특별한 경험을 제공하고자 합니다. (100kB) ▾





번외 : Telegram Bot

# SETUP



# 자동 응답하기

## REFERENCE : credentials.py

```
1 bot_token = "1804380674:AAGJr0aqeXCWvDHee-hpKUqTrINexSDWHRE"
2 URL = "https://4c4a061df74e.ngrok.io"
```

## REFERENCE : telegramBot.py

```
1 from flask import Flask, request
2 # pip install python-telegram-bot
3 # pip install telegram
4 import telegram
5 from credentials import bot_token, URL
6
7 TOKEN = bot_token
8 bot = telegram.Bot(token=TOKEN)
9
10 app = Flask(__name__)
11
12 @app.route('/setwebhook', methods=['GET', 'POST'])
13 def set_webhook():
14     # 텔레그램 봇과 연결 세팅
15     s = bot.setWebhook(f'{URL}/{TOKEN}')
16     if s:
17         return "webhook setup ok"
18     else:
19         return "webhook setup failed"
20
21 if __name__ == '__main__':
22     app.run(debug=True)
```

# 자동 응답하기

결과:



=> <본인 url>/setwebhook 들어가면 webhook setup ok 라고 나와야 함.

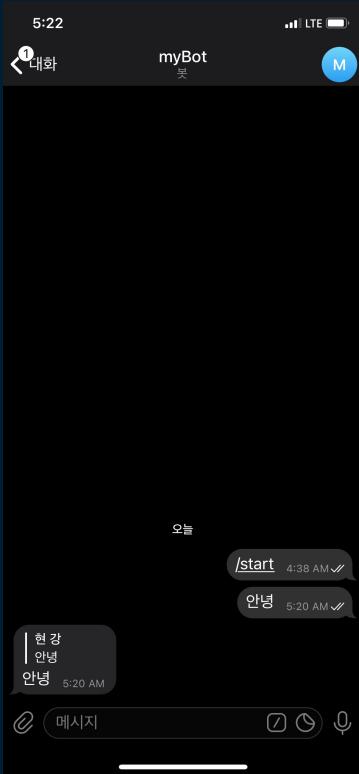
# 자동 응답하기

## REFERENCE : telegramBot.py

```
22 # 실제로 텔레그램에서 request를 보내는 URL
23 # 텔레그램의 경우 <URL>/<TOKEN> 으로 http request 전송
24 @app.route(f'/{TOKEN}', methods=['POST'])
25 def autoResponse():
26     # request를 json 형식으로 받은 다음 Telegram object로 변환
27     update = telegram.Update.de_json(request.get_json(force=True), bot)
28
29     chat_id = update.message.chat.id # request를 발생시킨 채팅방의 id
30     msg_id = update.message.message_id # request를 발생시킨 메시지의 id
31
32     # 유저가 보낸 메시지 읽기 (텔레그램은 utf-8를 사용하기에 서버에서 사용하는 unicode와 호환을 위해 utf-8으로 인코딩을 한 후 디코드)
33     text = update.message.text.encode('utf-8').decode()
34
35     try:
36         # 유저가 보낸 메시지에 대한 답장으로 유저 메시지 반복하기
37         bot.sendMessage(chat_id=chat_id, text=text, reply_to_message_id=msg_id)
38     except Exception:
39         # 에러가 발생했을 때
40         bot.sendMessage(chat_id=chat_id, text="문제가 발생하였습니다 :(, reply_to_message_id=msg_id)
41
42     return "ok"
```

# 자동 응답하기

결과:



# 자동 응답하기

## REFERENCE : telegramBot.py

# 채팅에 추가하기

## REFERENCE : telegramBot.py

```
# 실제로 텔레그램에서 request를 보내는 URL
# 텔레그램의 경우 <URL>/<TOKEN> 으로 http request 전송
@app.route(f'/{TOKEN}', methods=['POST'])
def autoResponse():
    # request를 json 형식으로 파싱
    resp = request.get_json(force=True)

    if 'message' in resp.keys():
        msgtext = resp["message"]["text"]
        sendername = resp["message"]["from"]["first_name"]
        chat_id = resp["message"]["chat"]["id"]
    elif 'channel_post' in resp.keys():
        msgtext = resp["channel_post"]["text"]
        sendername = resp["channel_post"]["chat"]["username"]
        chat_id = resp["channel_post"]["chat"]["id"]
    try:
        # 유저가 보낸 메시지에 대한 답장으로 유저 메시지 반복하기
        bot.sendMessage(chat_id=chat_id, text=msgtext)
    except Exception:
        # 에러가 발생했을 때
        bot.sendMessage(chat_id=chat_id, text="문제가 발생하였습니다 :( ")
    return "ok"
```

# 이미지 보내기

## REFERENCE : telegramImageBot.py

```
 1  from flask import Flask, request
 2
 3 # pip install python-telegram-bot
 4 # pip install telegram
 5 import telegram
 6 from credentials import bot_token, URL
 7
 8 TOKEN = bot_token
 9 bot = telegram.Bot(token=TOKEN)
10
11 app = Flask(__name__)
12
13 # 텔레그램 봇과 url 연동
14 @app.route('/setwebhook', methods=['GET', 'POST'])
15 def set_webhook():
16     s = bot.setWebhook(f'{URL}/{TOKEN}')
17     if s:
18         return "webhook setup ok"
19     else:
20         return "webhook setup failed"
21
22 # 실제로 텔레그램에서 request를 보내는 URL
23 # 텔레그램의 경우 <URL>/<TOKEN> 으로 http request 전송
24 @app.route(f'/{TOKEN}', methods=['POST'])
25 def autoResponse():
26     # request를 json 형식으로 받은 다음 Telegram object로 변환
27     update = telegram.Update.de_json(request.get_json(force=True), bot)
28
29     chat_id = update.message.chat.id # request를 발생시킨 채팅방의 id
30     msg_id = update.message.message_id # request를 발생시킨 메시지의 id
31
32     # 유저가 보낸 메시지 읽기 (텔레그램은 utf-8를 사용하기에 서버에서 사용하는 unicode와 호환을 위해 utf-8으로 인코딩을 한 후 디코드)
33     text = update.message.text.encode('utf-8').decode()
34
35     try:
36         # 이미지 보내기
37         bot.send_photo(chat_id=chat_id, photo='https://telegram.org/img/t_logo.png')
38     except Exception:
39         # 에러가 발생했을 때
40         bot.sendMessage(chat_id=chat_id, text="문제가 발생하였습니다 :(, reply_to_message_id=msg_id)
41
42     return "ok"
43
44 if __name__ == '__main__':
45     app.run(debug=True)
```