Flask

Flask Basic

1. Install Flask

python 웹 프레임워크인 flask와 flask를 서버로 올리기위한 WSGI(Web Server Gateway Interface)를 설치합니다.

$ pip3 install flask
$ pip3 install gunicorn
---

2. Make Directories

flask는 기본적으로 아래와 같은 디렉토리 구조를 갖습니다. hello.py는 flask로 만든 서버의 request를 받기 위한 파일이며 static은 javascript나 image, css와 같은 정적인 파일들을 가지는 디렉토리이며, templates는 브라우져 화면에 보여주기 위한 html 코드 파일이 들어갑니다.

$ mkdir hello
$ mkdir hello/static
$ mkdir hello/templates
$ touch hello/hello.py
$ touch hello/templates/profile.html
$ tree

hello
├── hello.py
├── static
└── templates
    └── profile.html
---

3. hello.py

```
from flask import Flask, render_template, jsonify

app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello Flask"

# returns an HTML webpage
@app.route("/user/<username>")
def user(username):
    return render_template('profile.html', name=username)

# retruns a piece of data in JSON format
@app.route("/people")
def people():
```

```
    people = {"alice": 25, "jin":35}
    return jsonify(people)

if __name__ == "__main__":
    app.run(debug=True)
```

---

4. profile.html

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Flask Basic</title>
    </head>
    <body>

        {% if name %}
            <h1 style="color:red;">Hello, {{name}}</h1>
        {% else %}
            <h1>Hello noname!</h1>
        {% endif %}
        <button id="getData">Get Data</button>

        <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
        <script type="text/javascript">
            $(document).ready(function(){
                $("#getData").click(function(){
                    $.getJSON("/people", function(dat){
                        console.dir(dat)
                    })
                })
            })
        </script>
    </body>
</html>
```

---

5. run server

python3 명령을 통해서 web application을 실행합니다.

```
hello $ python3 hello.py
```

브라우져 URL을 http://localhost:5000/ 혹은 http://127.0.0.1:5000/ 로 입력하여 접속하면 Hello Flask 글자가 보이는 것을 확인하실수 있습니다.

위와 같이 python3 명령어를 통해서 실행 할수 있지만 gunicorn 으로 실행하면 부가적인 기능들이 있습니다. 아래 코드는 gunicorn을 이용하여 web application을 실행합니다. (gunicorn은 window를 지원하지 않습니다.)

```
hello $ gunicorn hello:app
```

---

ngrok

reference : https://ngrok.com/

ngrok은 웹서버를 로컬에서 실행을 할때는 외부에서 접속이 불가능하기 때문에 외부에서 접속할수 있도록 해주는 터널 프로그램입니다.

1. install

mac은 brew cask를 이용하여 설치할수 있습니다.

```
$ brew cask install ngrok
```

windows는 아래 사이트에서 파일을 다운 받아 설치할수 있습니다.

https://ngrok.com/download

2. excute

8000번 포트로 웹 서버가 실행되어 있는 상태 여야 합니다.

mac (terminal)

```
$ ngrok http 8000
```

windows (cmd)

```
$ ngrok.exe http 5000
```

위와 같이 ngrok을 실행하면 아래와 같이 나옵니다. 붉은색으로 표시되어 있는 부분을 브라우져 URL 입력칸에 넣고 페이지를 이동하면 Hello Flask 문자가 있는 페이지가 나오며 다른 기기에서 접속해도 페이지 접속이 가능합니다.

```
ngrok by @inconshreveable
(Ctrl+C to quit)

Session Status                online
Session Expires               7 hours, 59 minutes
Version                       2.2.8
Region                        United States (us)
Web Interface                 http://127.0.0.1:4040
Forwarding                    http://d31916ee.ngrok.io -> localhost:8000
Forwarding                    https://d31916ee.ngrok.io -> localhost:8000
```

브라우져에 127.0.0.1:4040 주소로 접속하면 서버 접속 로그를 ngrok의 대쉬보드를 확인하실수 있습니다.

Flask Slack Outgoing Webhook

여기에서는 flask 서버 어플리케이션을 만들고 ngrok을 이용하여 외부 망 접속이 가능하게 해준후 slack의 outgoing webhook을 이용하여 slack을 이용한 간단한 채팅 서비스를 구현하도록 하겠습니다.

1. Install forecastio

날씨 정보를 제공해주는 패키지인 python-forecastio 를 인스톨 해줍니다.

```
$ pip3 install python-forecastio
```

2. Make Directories

아래와 같이 bot 디렉토리 아래에 bot.py 파일과 lib 디렉토리를 만들고 lib 디렉토리 안에 forcast.py와 slack.py 파일을 만들어 줍니다. bot.py 파일을 client에서 request 요청을 처리하는 코드가 들어가고 lib 디렉토리의 forecast.py는 날씨를 예측해주는 코드가 들어가고 slack.py는 문자열을 slack으로 전송해주는 코드가 들어갑니다.

```
bot
├── bot.py
└── lib
    ├── forecast.py
    └── slack.py
```

3. bot.py

bot.py에는 /slack 경로로 post 방법으로 request요청에 대한 서버에서의 처리에 대한 코드가 들어갑니다. slack 서버에서 /slack 경로로 데이터를 보내면 text 데이터에 날씨라는 단어가 있으면 일기예보에대한 결과를 slack으로 보내줍니다.

```
from flask import Flask, request, Response

from lib.slack import send_slack
from lib.forecast import forecast

app = Flask(__name__)

# slack outgoing webhook
@app.route("/slack", methods=['POST'])
def slack():
    username = request.form.get('user_name')
    token = request.form.get('token')
    text = request.form.get('text')

    if "날씨" in text:
        summary = forecast()
```

```
        send_slack(summary)

    print(username, token, text)

    return Response(), 200

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

4. lib/forecast.py

FORECAST_TOKEN 의 값은 darksky(https://darksky.net/dev) 웹 사이트에서 회원가입을 하시면 token을 받으실수 있습니다.

```
import forecastio

FORECAST_TOKEN = "xxxxxxxxxxx"

def forecast(lat = 37.5124413, lng = 126.9540519):
    forecast = forecastio.load_forecast(FORECAST_TOKEN, lat, lng)
    byHour = forecast.hourly()
    return byHour.summary
```

FORECAST_TOKEN을 입력하는 방법은 위의 코드와 같이 직접 코드 내에 입력하는 방법도 있지만 환경변수로 저장후 사용도 가능합니다. 환경변수에 저장 후 사용하는 방법은 아래와 같습니다.

아래와 같이 터미널에서 export 명령을 통해 token 데이터를 저장합니다.

```
$ export FORECAST_TOKEN='xxxxxxxxxxx'
```

저장한 후에 위에서 FORECAST_TOKEN = "xxxxxxxxxxx" 코드 대신에 아래의 코드를 넣어 환경 변수에서 FORECAST_TOKEN 데이터를 가져와서 사용이 가능합니다.

```
FORECAST_TOKEN = os.environ.get('FORECAST_TOKEN')
```

환경변수로 등록해서 사용하면 코드 내에 token 데이터를 노출하지 않을수 있는 장점이 있습니다.

5. slack.py

slack.py는 일기예보 결과를 슬랙으로 보내주는 코드가 들어있습니다. WEBHOOK_URL은 slack manage의 custom integrations의 inocoming webhooks을 설정하여 받을수 있습니다.

```
import requests, json

WEBHOOK_URL = "https://hooks.slack.com/services/T1AE30QG6/xxxxxxxxx"

def send_slack(msg):
```

```
data = {
    "channel": "#webhook",
    "emoji": ":angry:",
    "msg": msg,
    "username": "날씨봇",
}
payload = {
    "channel": data["channel"],
    "username": data["username"],
    "icon_emoji": data["emoji"],
    "text": data["msg"],
}
response = requests.post(
    WEBHOOK_URL,
    data = json.dumps(payload),
)
```

---

6. run app & ngrok

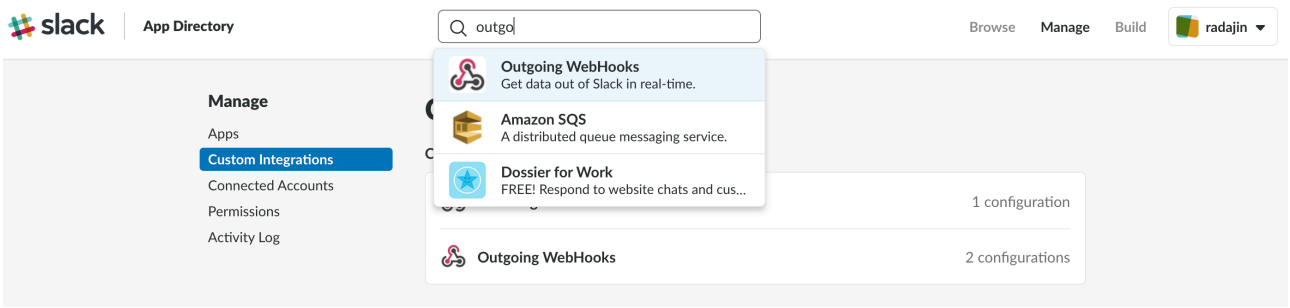
```
# run app
bot $ python3 bot.py

# mac
$ ngrok http 5000

# window
$ ngrok.exe http 5000
```

---

ngrok 외부 인터넷망 연결 주소 : **https://d31916ee.ngrok.io**

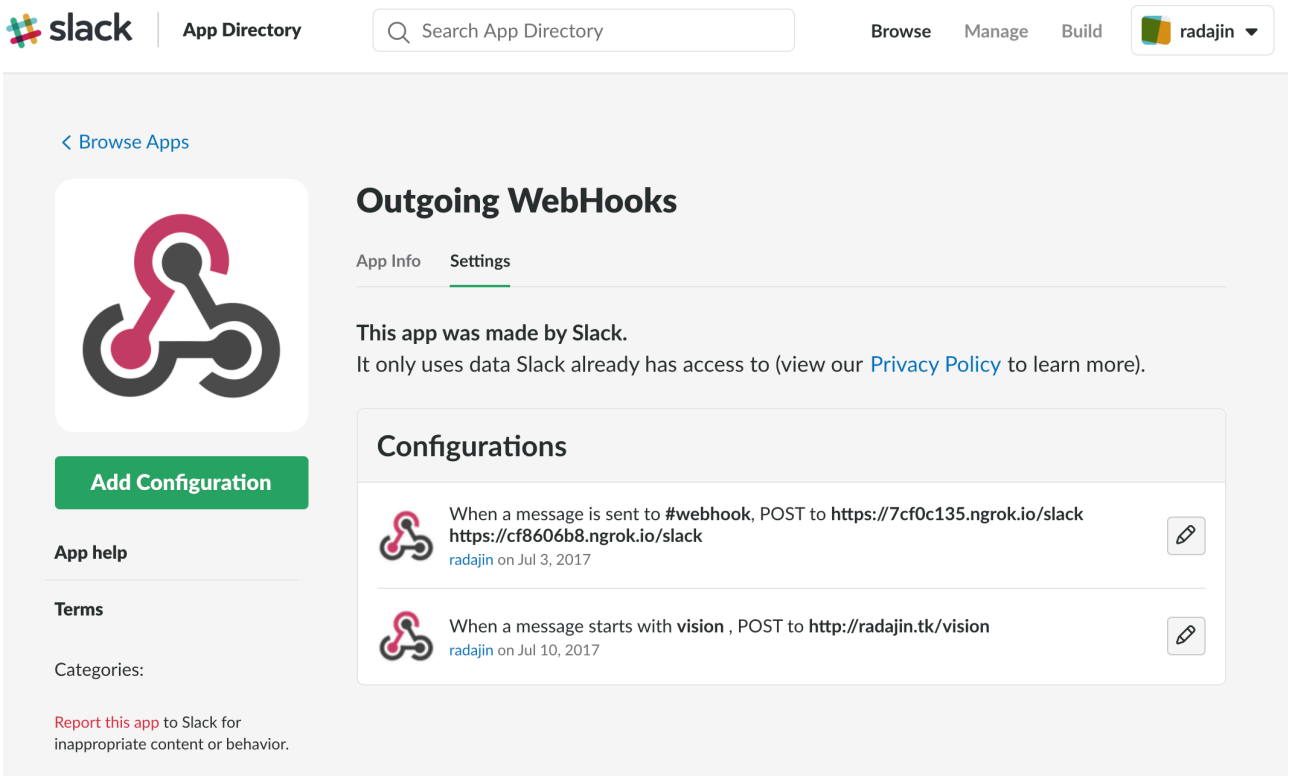
7. outgoing webhook 설정

(1) slack 의 manage 메뉴로 가서 outgoin webhooks을 검색하여 클릭합니다.

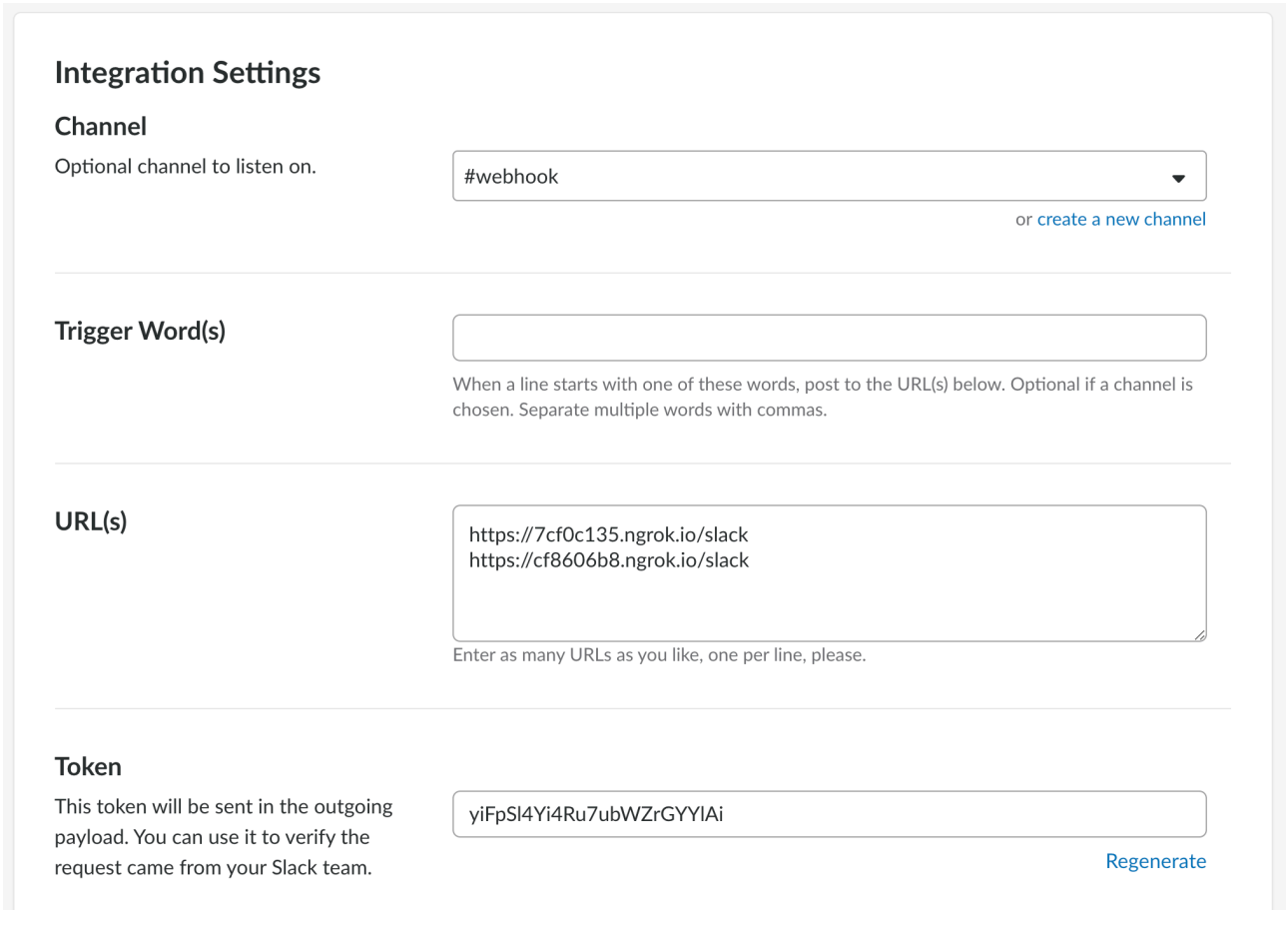


(2) outgoing webhooks 페이지의 좌측의 Add Configuration 버튼을 클릭합니다.

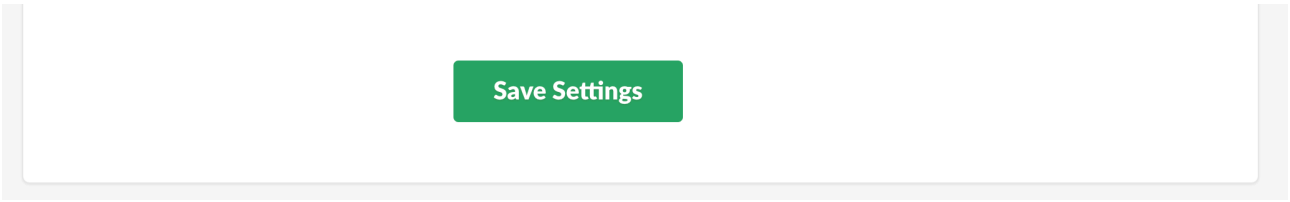
(3) 아래와 같이 채널선택, 트리거 단어, URL을 설정해줍니다.



트리거 단어는 메시지를 보낼때 가장 앞에 outgoing webhooks가 실행될 단어를 작성합니다.
URL은 슬랙에 작성된 메시지와 함께 전송할 URL을 작성합니다.

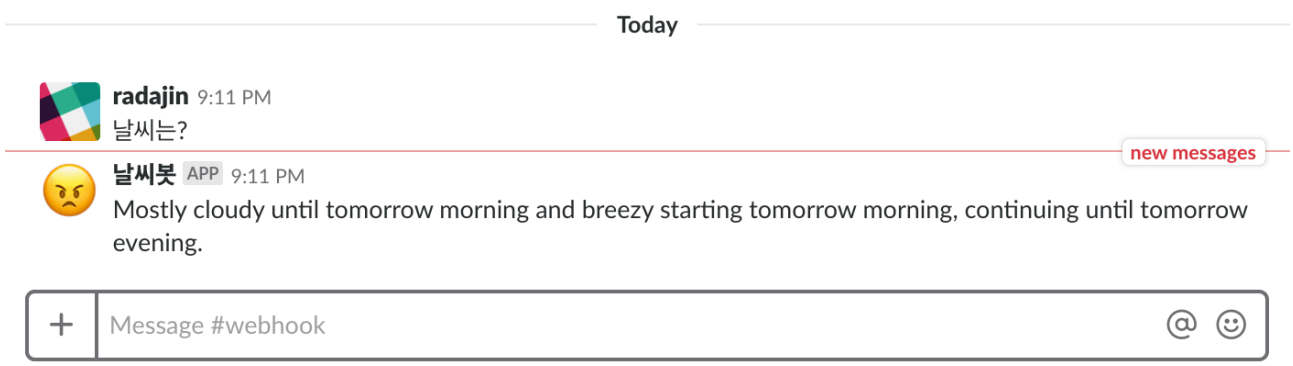


(4) 설정이 끝나면 페이지의 가장 하단에 있는 Save Settings를 클릭합니다.



8. 실행

슬랙에 "날씨는?" 이라고 입력해줍니다.



flask on ubuntu

reference : https://github.com/yyuu/pyenv/wiki/Common-build-problems

1. install 의존성 파일

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install -y make build-essential libssl-dev zlib1g-dev libbz2-dev libreadline-dev libsqlite3-dev wget curl llvm libncurses5-dev libncursesw5-dev
```

2. install pip & nginx

```
$ sudo apt-get install python-pip nginx
```

3. set nginx

nginx를 시작하고 127.0.0.1:8000 ip를 외부 IP로 매핑 시켜줌

```
$ sudo /etc/init.d/nginx start
$ sudo rm /etc/nginx/sites-enabled/default
$ sudo touch /etc/nginx/sites-available/flask_settings
$ sudo ln -s /etc/nginx/sites-available/flask_settings /etc/nginx/sites-enabled/flask_settings
$ sudo vi /etc/nginx/sites-enabled/flask_settings

------------------flask_settings--------------------
server {
        location / {
                proxy_pass http://127.0.0.1:8000;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
        }
}
----------------------------------------------------

$ sudo /etc/init.d/nginx restart
```

4. install flask & gunicorn

```
$ pip install flask gunicorn
```

5. excute flask app

```
$ gunicorn hello:app
```

6. aws open port 80