



# [스파르타코딩클럽] 웹개발 종합반 - 5주차



매 주차 강의자료 시작에 PDF파일을 올려두었어요!

## ▼ PDF 파일

## ▼ 단축키 모음

### ▼ 새로그침

- `F5`

### ▼ 저장

- Windows: `Ctrl` + `S`
- macOS: `command` + `S`

### ▼ 전체선택

- Windows: `Ctrl` + `A`
- macOS: `command` + `A`

### ▼ 잘라내기

- Windows: `Ctrl` + `X`
- macOS: `command` + `X`

### ▼ 콘솔창 줄바꿈

- `shift` + `enter`

### ▼ 코드정렬

- Windows: `Shift` + `Alt` + `F`
- macOS: `Shift` + `Option` + `F`

### ▼ 들여쓰기

- `Tab`
- 들여쓰기 취소 : `Shift` + `Tab`

### ▼ 주석

- Windows: `Ctrl` + `/`
- macOS: `command` + `/`

## [수업 목표]

1. Flask 프레임워크를 활용해 API를 만들 수 있다
2. [버킷리스트] 프로젝트를 생성해 API를 만들고 클라이언트에 연결한다
3. [팬명록] 프로젝트를 생성해 API를 만들고 클라이언트에 연결한다
4. AWS Elastic Beanstalk으로 직접 만든 웹 서비스를 배포한다

## [목차]

01. 5주차 오늘 배울 것
02. AWS 가입하기 및 보안설정하기
03. [버킷리스트] - 프로젝트 세팅

- 04. [버킷리스트] - 뼈대 준비하기
- 05. [버킷리스트] - POST 연습하기(버킷리스트 기록)
- 06. [버킷리스트] - GET 연습하기(버킷리스트 보여주기)
- 07. [팬명록] - 프로젝트 세팅
- 08. [팬명록] - 뼈대 준비하기
- 09. [팬명록] - 조각기능. 날씨 API 적용하기
- 10. [팬명록] - POST 연습하기(응원 등록하기)
- 11. [팬명록] - GET 연습하기(응원 보여주기)
- 12. og 태그
- 13. 내 프로젝트를 서버에 올리기
- 14. AWS Elastic Beanstalk으로 배포하기
- 15. 5주차 끝 & 숙제 설명



모든 토글을 열고 닫는 단축키

Windows : **Ctrl** + **alt** + **t**

Mac : **⌘** + **⌥** + **t**

## 01. 5주차 오늘 배울 것

- ▼ 1) 5주차 : 미니프로젝트3, 4, 배포하기

이번 주 완성본 1. 버킷리스트 → [결과물 링크](#)

이번 주 완성본 2. 팬명록 → [결과물 링크](#)



로컬 개발환경에서 두 개의 프로젝트를 더 연습해봅시다 그리고...



우리가 만든 프로젝트를 웹 서비스로 출시해보는 경험을 가질겁니다! 멋지지 않나요?

## 02. AWS 가입하기 및 보안설정하기

- ▼ 1) AWS 가입하기

- AWS 가입 링크 : ([링크](#))

- ▼ 추가 보안이 궁금하신 분들

- ▼ 1) AWS OTP 사용
- ▼ 2) 결제 알람 기능 사용

## 03. [버킷리스트] - 프로젝트 세팅



sparta → proejcts → 04.bucket 폴더에서 시작!

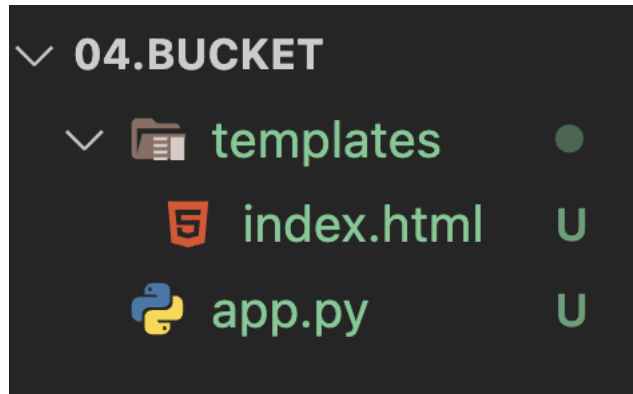
- ▼ 1) 문제 분석 - 완성작부터 보기!

- ▼ 2) 프로젝트 세팅 - Flask 폴더 구조 만들기

😎 templates 폴더 + app.py 만들기! 기억나시죠?

#### 👁 04. bucket 폴더 구조

```
---
prac
|— venv
|— app.py (서버)
|— templates
    |— index.html (클라이언트 파일)
```



😎 Flask는 만들 프로젝트의 폴더 구조가 정해져 있어요! 규칙을 지켜주세요!

- 폴더 안에 `app.py` 파일을 생성합니다!
- 폴더 안에 `templates` 폴더를 생성합니다!
- templates 폴더 안에 `index.html` 파일을 생성합니다!

😞 아무 이름이나 쓰면 안되나요? → 몇 가지 규칙이 있습니다!

- `templates` 폴더는 반드시 고정해야 합니다! Flask의 규칙이에요!
- `app.py` 는 변경해도 좋습니다만, 라이브러리 이름과 같은 것을 이름으로 사용하면 안돼요!
- `index.html` 은 변경해도 좋습니다만, 첫 페이지는 일반적으로 index.html을 사용해요!

🔥 준비가 끝났다면 화면 상단 `File > Open Folder` 로 04. bucket 폴더로 이동합니다!

### ▼ 3) 가상환경 생성, 패키지 설치하기

#### ▼ 1) 가상환경 생성, 활성화

- 화면 상단 `Terminal > New Terminal` 을 클릭!
- `python(맥 python3) -m venv venv` 입력 후 엔터!
- 터미널 오른쪽의 십자버튼을 클릭!
- (venv) 라고 뜨면 활성화까지 완료!

#### ▼ 2) 패키지 설치



이번에 필요한 패키지는 3개 : flask, pymongo, dnspython

- 화면 상단 `Terminal > New Terminal` 을 클릭!(열려있다면 그대로 진행!)

#### ▼ [코드스니펫] 패키지 설치코드

```
pip install flask pymongo dnspython
```

- 여러 개를 설치할 때는 띄어쓰기로 구분해요!

#### ▼ 📌 원하는 라이브러리를 확인해보려면?

- 터미널에서 `pip freeze` 를 입력하고 엔터!
- 터미널에서 flask, pymongo, dnspython을 찾아보죠!



제가 찾던 라이브러리 여기 있네요!

내용이 뭐가 많да구요?

우리가 사용하는 라이브러리가 다른 라이브러리를 필요로 해서 그것도 설치해주는 거예요!

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
zsh - projects  + -  [ ]  [X]  ^ x

[notice] A new release of pip available: 22.2.2 -> 22.3.1
[notice] To update, run: pip install --upgrade pip
• (venv) migdracios@Yungsangui-MacBookAir projects % pip freeze
click==8.1.3
dnspython==2.2.1
Flask==2.2.2
itsdangerous==2.1.2
Jinja2==3.1.2
MarkupSafe==2.1.1
pymongo==4.3.2
Werkzeug==2.2.2
○ (venv) migdracios@Yungsangui-MacBookAir projects %

```

## 04. [버킷리스트] - 뼈대 준비하기

### ▼ 1) 프로젝트 준비 - app.py 준비하기

#### ▼ [코드스니펫] 버킷리스트 - app.py

```

from flask import Flask, render_template, request, jsonify
app = Flask(__name__)

@app.route('/')
def home():
    return render_template('index.html')

@app.route("/bucket", methods=["POST"])
def bucket_post():
    sample_receive = request.form['sample_give']
    print(sample_receive)
    return jsonify({'msg': 'POST 연결 완료!'})

@app.route("/bucket", methods=["GET"])
def bucket_get():
    return jsonify({'msg': 'GET 연결 완료!'})

if __name__ == '__main__':
    app.run('0.0.0.0', port=5000, debug=True)

```

### ▼ 2) 프로젝트 준비 - index.html 준비하기

#### ▼ [코드스니펫] 버킷리스트 - index.html

```

<!DOCTYPE html>
<html lang="en">

```

```

<head>
<meta charset="UTF-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTWfSpd3yD65VohhpuuC0mLASjc"
crossorigin="anonymous"
/>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384-Mrcw6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
crossorigin="anonymous"
></script>

<link
href="https://fonts.googleapis.com/css2?family=Gowun+Dodum&display=swap"
rel="stylesheet"
/>

<title>인생 버킷리스트</title>

<style>
* {
font-family: "Gowun Dodum", sans-serif;
}
.mypic {
width: 100%;
height: 200px;

background-image: linear-gradient(
0deg,
rgba(0, 0, 0, 0.5),
rgba(0, 0, 0, 0.5)
),
url("https://images.unsplash.com/photo-1601024445121-e5b82f020549?ixid=MnwXmJA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8");
background-position: center;
background-size: cover;

color: white;

display: flex;
flex-direction: column;
align-items: center;
justify-content: center;
}
.mypic > h1 {
font-size: 30px;
}
.mybox {
width: 95%;
max-width: 700px;
padding: 20px;
box-shadow: 0px 0px 10px 0px lightblue;
margin: 20px auto;
}
.mybucket {
display: flex;
flex-direction: row;
align-items: center;
justify-content: space-between;
}

.mybucket > input {
width: 70%;
}
.mybox > li {
display: flex;
flex-direction: row;
align-items: center;
justify-content: center;

margin-bottom: 10px;
min-height: 48px;
}
.mybox > li > h2 {
max-width: 75%;
font-size: 20px;
font-weight: 500;
margin-right: auto;
margin-bottom: 0px;
}
.mybox > li > h2.done {
text-decoration: line-through;
}

```

```

    }
</style>
<script>
    $(document).ready(function () {
        show_bucket();
    });
    function show_bucket() {
        fetch('/bucket').then(res => res.json()).then(data => {
            // console.log(data)
        })
    }

    function save_bucket() {
        let formData = new FormData();
        formData.append("sample_give", "샘플데이터");

        fetch('/bucket', {method: "POST", body: formData}).then((response) => response.json()).then((data) => {
            alert(data["msg"]);
            window.location.reload();
        });
    }
}

</script>
</head>
<body>
<div class="mypic">
    <h1>나의 버킷리스트</h1>
</div>
<div class="mybox">
    <div class="mybucket">
        <input
            id="bucket"
            class="form-control"
            type="text"
            placeholder="이루고 싶은 것을 입력하세요"
        />
        <button onclick="save_bucket()" type="button" class="btn btn-outline-primary">기록하기</button>
    </div>
</div>
<div class="mybox" id="bucket-list">
    <li>
        <h2>👍 호주에서 스카이다이빙 하기</h2>
        <button onclick="done_bucket(5)" type="button" class="btn btn-outline-primary">완료!</button>
    </li>
    <li>
        <h2 class="done">👍 호주에서 스카이다이빙 하기</h2>
    </li>
    <li>
        <h2>👍 호주에서 스카이다이빙 하기</h2>
        <button type="button" class="btn btn-outline-primary">완료!</button>
    </li>
</div>
</body>
</html>

```

### ▼ 3) 프로젝트 준비 - mongoDB Atlas 창 띄워두기

#### ▼ [코드스니펫] mongoDB Atlas 주소

<https://cloud.mongodb.com/>

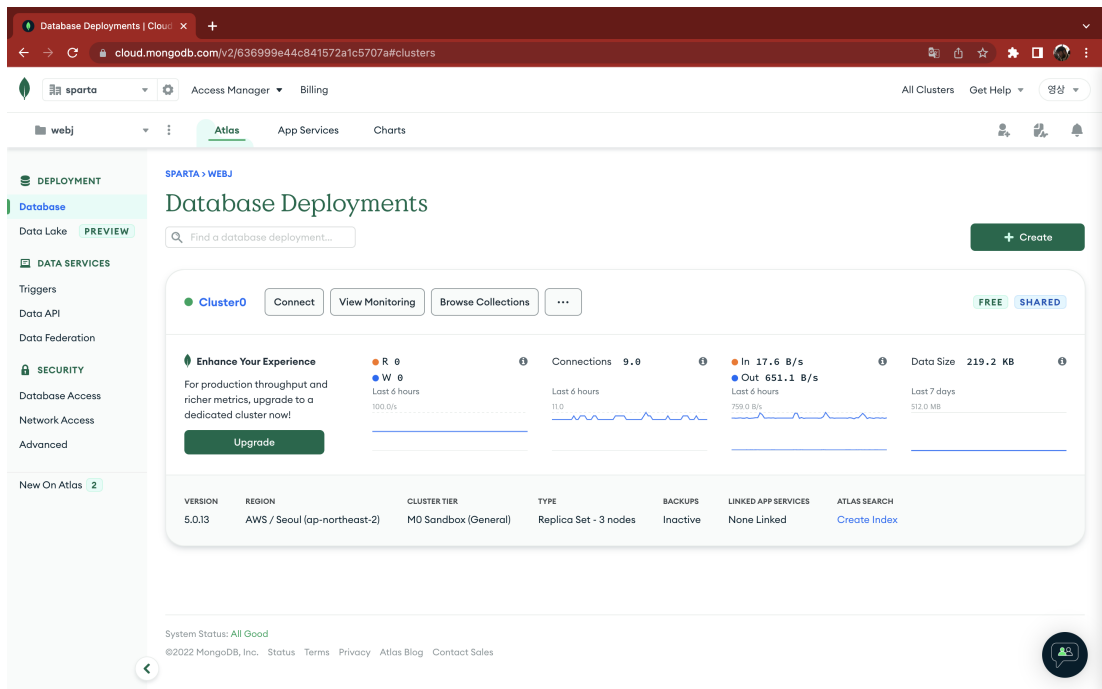
#### ▼ mongoDB 데이터베이스 들어가는 법



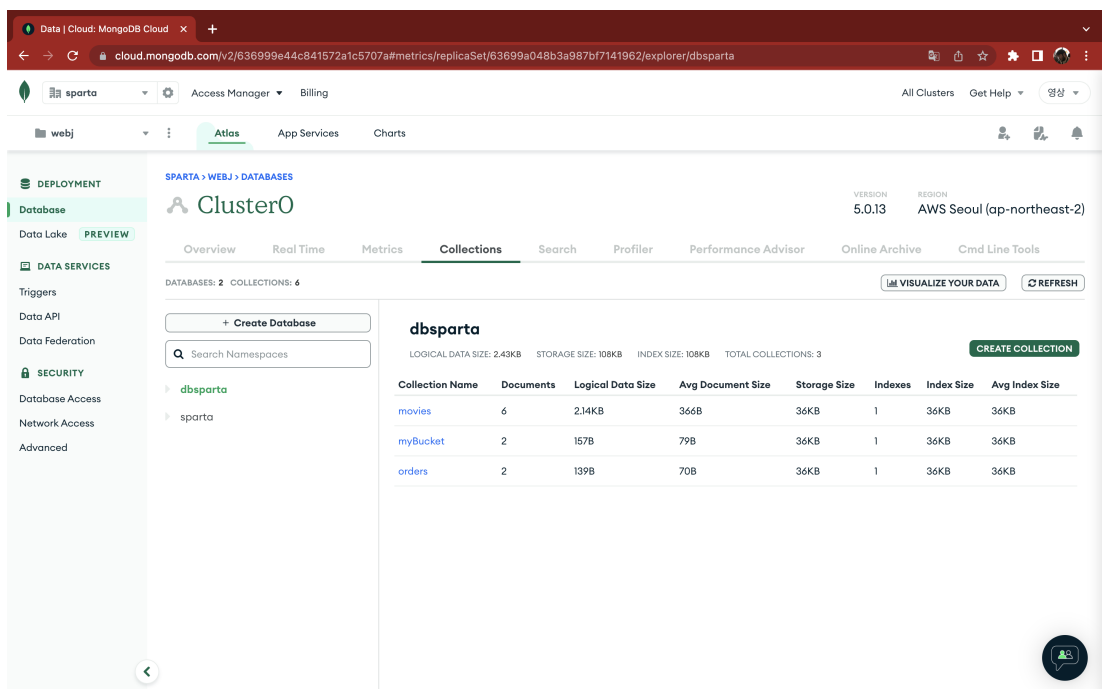
지난 주차에서 mongoDB Atlas! 기억나시죠?

- 아직 연결한 것은 없으니까 보이는 것은 없지만, 이따 확인하려면 미리 띄워둬야죠!

화면 왼쪽의 Database를 클릭한 뒤 **Cluster 0** 클릭해주세요!



화면 가운데의 여러 탭 중 **Collection** 탭을 클릭해주세요!



## 05. [버킷리스트] - POST 연습하기(버킷리스트 기록)

▼ 1) API 만들고 사용하기 - 버킷리스트 기록 API(Create → POST)



데이터 생성은 POST 방식! 기억나시죠?

### ▼ 1) 데이터 명세

- 1. 요청 정보 : URL= `/bucket` , 요청 방식 = `POST`
- 2. 클라(fetch) → 서버(flask) : `bucket`
- 3. 서버(flask) → 클라(fetch) : 메시지를 보냄 (버킷리스트 저장 완료!)

### ▼ 2) 클라이언트와 서버 연결 확인하기

[서버 코드 - `app.py`]

```
@app.route("/bucket", methods=["POST"])
def bucket_post():
    sample_receive = request.form['sample_give']
    print(sample_receive)
    return jsonify({'msg': 'POST 연결 완료!'})
```

[클라이언트 코드 - `index.html`]

```
function save_bucket() {
    let formData = new FormData();
    formData.append("sample_give", "샘플데이터");

    fetch('/bucket', {method: "POST", body: formData}).then((response) => response.json()).then((data) => {
        alert(data['msg'])
    });
}

<button onclick="save_bucket()" type="button" class="btn btn-outline-primary">기록하기</button>
```

### ▼ 3) 서버부터 만들기

데이터베이스에 연결해줍니다!

```
from pymongo import MongoClient

client = MongoClient('내 URL')
db = client.dbsparta
```

- bucket 정보를 받아서, 저장하면 되겠죠?
  - `bucket_receive = request.form['bucket_give']`
- 우리가 일전에 만들어둔 `dbtest.py` 에서 코드를 불러와봅시다!
- 잘 입력되었다는 표시를 해주기 위한 메시지를 바꿔봅시다!

```
@app.route("/bucket", methods=["POST"])
def bucket_post():
    bucket_receive = request.form['bucket_give']
    print("리시브", bucket_receive)
    doc = {
        'bucket' : bucket_receive
    }
    db.myBucket.insert_one(doc)
    return jsonify({'msg': '👉 버킷리스트가 저장되었습니다!'})
```

### ▼ 4) 클라이언트 만들기

- 이번엔 `bucket` 정보를 보내주면 되겠죠?
  - `let bucket = $('#bucket').val()`
- `formData`에 데이터를 넣고, 보내줍니다!
- 저장되면, 새로고침을 해서 다시 보여줄 준비를 합니다!



◦ `window.location.reload()`

```
function save_bucket() {
  let bucket = $("#bucket").val();

  let formData = new FormData();
  formData.append("bucket_give", bucket);

  fetch('/bucket', {method: "POST", body: formData,}).then((res) => res.json()).then((data) => {
    alert(data["msg"]);
    window.location.reload();
  });
}
```

#### ▼ 5) 완성 확인하기

- 데이터를 입력하고
- DB에 잘 들어갔는지 확인해보면 되겠죠?

```
{
  _id: ObjectId('636dbe5d517e5466b9e78f65')
  bucket: "미국에서 스카이다이빙 하기"
}
```

## 06. [버킷리스트] - GET 연습하기(버킷리스트 보여주기)

### ▼ 1) API 만들고 사용하기 - 버킷리스트 조회 API(Read → GET)



데이터 조회는 GET 방식! 기억나시죠?

#### ▼ 1) 데이터 명세

- 1. 요청 정보 : URL= `/bucket` , 요청 방식 = `GET`
- 2. 클라(fetch) → 서버(flask) : 없음
- 3. 서버(flask) → 클라(fetch) : 전체 버킷을 보여주기

#### ▼ 2) 클라이언트와 서버 연결 확인하기

[서버 코드 - `app.py`]

```
@app.route("/bucket", methods=["GET"])
def bucket_get():
    return jsonify({'msg': 'GET 연결 완료!'})
```

[클라이언트 코드 - `index.html`]

```
$(document).ready(function () {
  show_bucket();
});
function show_bucket() {
  fetch('/bucket').then(res => res.json()).then(data => {
    console.log(data)
    alert(data['msg'])
  })
}
```

#### ▼ 3) 서버부터 만들기

받을 것 없이 `myBucket` 에 주문정보를 담아서 내려주기만 하면 됩니다!

```
@app.route("/bucket", methods=["GET"])
def bucket_get():
    bucket_list = list(db.myBucket.find({}, {'_id': False}))
    return jsonify({'bucket_list': bucket_list})
```

#### ▼ 4) 클라이언트 만들기

- 버킷리스트는 말 그대로 리스트 형식이겠죠?
  - `forEach` 문으로 반복하면서 데이터를 뽑아냅니다!
- 가져온 데이터를 `temp_html`로 뼈대에 데이터를 담아 제이쿼리로 `append`!
- 기존 html 코드들을 지워줄 수 있도록 `empty()` 사용하기!

```
function show_bucket() {
    $('#bucket-list').empty()
    fetch('/bucket')
        .then(res => res.json())
        .then(data => {
            // console.log(data)
            data['bucket_list'].forEach((row) => {
                let bucket = row['bucket']
                let temp_html = `<li>
                    <h2>✔️ ${bucket}</h2>
                </li>`
                $('#bucket-list').append(temp_html)
            })
        })
}
```

#### ▼ 5) 완성 확인하기



**동작 테스트:** 화면을 새로고침 했을 때, DB에 저장된 리뷰가 화면에 올바르게 나타나는지 확인합니다.

명세

요청정보 /bucket 요청방식 get

클라 서버 없음

서버 클라 전체데이터 보여주기

#### ▼ 2) 전체 완성 코드 확인하기

##### ▼ [📄. 코드] app.py - 버킷리스트 서버

```
from flask import Flask, render_template, request, jsonify
from pymongo import MongoClient

client = MongoClient("내 URL")
db = client.dbsparta

app = Flask(__name__)

@app.route('/')
def home():
    return render_template('index.html')

@app.route("/bucket", methods=["POST"])
def bucket_post():
    bucket_receive = request.form['bucket_give']
    print("리시브", bucket_receive)
```

```

doc = {
    'bucket' : bucket_receive
}
db.myBucket.insert_one(doc)
return jsonify({'msg': ' 🍷 버킷리스트가 저장되었습니다! '})

@app.route("/bucket", methods=["GET"])
def bucket_get():
    bucket_list = list(db.myBucket.find({}, {'_id': False}))
    return jsonify({'bucket_list': bucket_list})

if __name__ == '__main__':
    app.run('0.0.0.0', port=5001, debug=True)

```

## ▼ [📁 코드] index.html - 버킷리스트 클라이언트

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
      rel="stylesheet"
      integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTWFspD3yD65VohhpuuC0mLASjC"
      crossorigin="anonymous"
    />
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script
      src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
      integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
      crossorigin="anonymous"
    ></script>

    <link
      href="https://fonts.googleapis.com/css2?family=Gowun+Dodum&display=swap"
      rel="stylesheet"
    />

    <title>인생 버킷리스트</title>

    <style>
      * {
        font-family: "Gowun Dodum", sans-serif;
      }
      .mypic {
        width: 100%;
        height: 200px;

        background-image: linear-gradient(
          0deg,
          rgba(0, 0, 0, 0.5),
          rgba(0, 0, 0, 0.5)
        ),
          url("https://images.unsplash.com/photo-1601024445121-e5b82f020549?ixid=MnwMjA3fDB8MHxwaG90by1wYwdlFhX8fGVufDB8fHx8");
        background-position: center;
        background-size: cover;

        color: white;

        display: flex;
        flex-direction: column;
        align-items: center;
        justify-content: center;
      }
      .mypic > h1 {
        font-size: 30px;
      }
      .mybox {
        width: 95%;
        max-width: 700px;
        padding: 20px;
        box-shadow: 0px 0px 10px 0px lightblue;
        margin: 20px auto;
      }
      .mybucket {
        display: flex;
        flex-direction: row;
        align-items: center;

```

```

        justify-content: space-between;
    }

    .mybucket > input {
        width: 70%;
    }
    .mybox > li {
        display: flex;
        flex-direction: row;
        align-items: center;
        justify-content: center;

        margin-bottom: 10px;
        min-height: 48px;
    }
    .mybox > li > h2 {
        max-width: 75%;
        font-size: 20px;
        font-weight: 500;
        margin-right: auto;
        margin-bottom: 0px;
    }
    .mybox > li > h2.done {
        text-decoration: line-through;
    }
</style>
<script>
$(document).ready(function () {
    show_bucket();
});
function show_bucket() {
    $('#bucket-list').empty()
    fetch('/bucket')
        .then(res => res.json())
        .then(data => {
            // console.log(data)
            data['bucket_list'].forEach((row) => {
                let bucket = row['bucket']
                let temp_html = `<li>
                                <h2>📌 ${bucket}</h2>
                                </li>`
                $('#bucket-list').append(temp_html)
            })
        })
    }

function save_bucket() {
    let bucket = $("#bucket").val();

    let formData = new FormData();
    formData.append("bucket_give", bucket);


    fetch('/bucket', {method: "POST", body: formData}).then((res) => res.json()).then((data) => {
        alert(data["msg"]);
        window.location.reload();
    });
}

</script>
</head>
<body>
<div class="mypic">
    <h1>나의 버킷리스트</h1>
</div>
<div class="mybox">
    <div class="mybucket">
        <input
            id="bucket"
            class="form-control"
            type="text"
            placeholder="이루고 싶은 것을 입력하세요"
        />
        <button onclick="save_bucket()" type="button" class="btn btn-outline-primary">기록하기</button>
    </div>
</div>
<div class="mybox" id="bucket-list">
    <li>
        <h2>📌 호주에서 스카이다이빙 하기</h2>
        <button onclick="done_bucket(5)" type="button" class="btn btn-outline-primary">완료!</button>
    </li>
    <li>
        <h2 class="done">📌 호주에서 스카이다이빙 하기</h2>
    </li>
    <li>
        <h2>📌 호주에서 스카이다이빙 하기</h2>
        <button type="button" class="btn btn-outline-primary">완료!</button>
    </li>
</div>

```

```
</div>
</body>
</html>
```

## 07. [팬명록] - 프로젝트 세팅

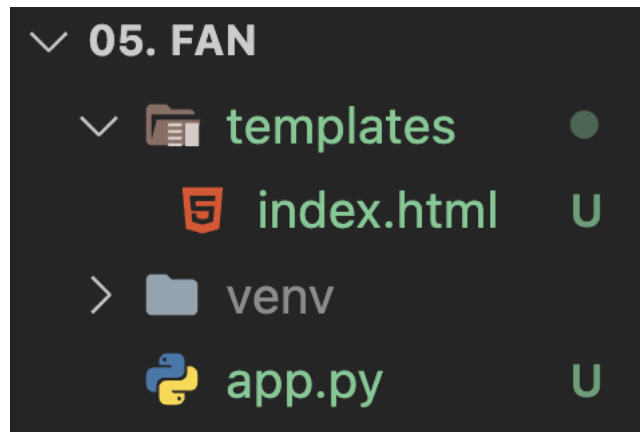
 sparta → proejcts → 05.fan 폴더에서 시작!


▼ 1) 문제 분석 - 완성작부터 보기!

▼ 2) 프로젝트 세팅 - Flask 폴더 구조 만들기


 templates 폴더 + app.py 만들기! 기억나시죠?

 05. fan 폴더 구조  
---  
prac  
|— venv  
|— app.py (서버)  
|— templates  
    |— index.html (클라이언트 파일)



 Flask는 만들 프로젝트의 폴더 구조가 정해져 있어요! 규칙을 지켜주세요!

- 폴더 안에 `app.py` 파일을 생성합니다!
- 폴더 안에 `templates` 폴더를 생성합니다!
- templates 폴더 안에 `index.html` 파일을 생성합니다!

 아무 이름이나 쓰면 안되나요? → 몇 가지 규칙이 있습니다!

- `templates` 폴더는 반드시 고정해야 합니다! Flask의 규칙이에요!
- `app.py` 는 변경해도 좋습니다만, 라이브러리 이름과 같은 것을 이름으로 사용하면 안돼요!

- `index.html` 은 변경해도 좋습니다만, 첫 페이지는 일반적으로 `index.html`을 사용해요!

🔥 준비가 끝났다면 화면 상단 `File > Open Folder` 로 05. fan 폴더로 이동합니다!

### ▼ 3) 가상환경 생성, 패키지 설치하기

#### ▼ 1) 가상환경 생성, 활성화

- 화면 상단 `Terminal > New Terminal` 을 클릭!
- `python(백 python3) -m venv venv` 입력 후 엔터!
- 터미널 오른쪽의 십자버튼을 클릭!
- `(venv)` 라고 뜨면 활성화까지 완료!

#### ▼ 2) 패키지 설치

🔥 이번에 필요한 패키지는 3개 : `flask`, `pymongo`, `dnspython`

- 화면 상단 `Terminal > New Terminal` 을 클릭!(열려있다면 그대로 진행!)

#### ▼ [코드스니펫] 패키지 설치코드

```
pip install flask pymongo dnspython
```

- 여러 개를 설치할 때는 띄어쓰기로 구분해요!
- ▼ 🐞 원하는 라이브러리를 확인해보려면?
  - 터미널에서 `pip freeze` 를 입력하고 엔터!
  - 터미널에서 `flask`, `pymongo`, `dnspython`을 찾아보죠!

😎 제가 찾던 라이브러리 여기 있네요!

내용이 뭐가 많да구요?

우리가 사용하는 라이브러리가 다른 라이브러리를 필요로 해서 그것도 설치해주는 거예요!

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
[notice] A new release of pip available: 22.2.2 -> 22.3.1
[notice] To update, run: pip install --upgrade pip
(migdracios@Yungsangui-MacBookAir projects) % pip freeze
click==8.1.3
dnspython==2.2.1
Flask==2.2.2
itsdangerous==2.1.2
Jinja2==3.1.2
MarkupSafe==2.1.1
pymongo==4.3.2
Werkzeug==2.2.2
(migdracios@Yungsangui-MacBookAir projects) %

```

## 08. [팬명록] - 뼈대 준비하기

### ▼ 1) 프로젝트 준비 - `app.py` 준비하기

#### ▼ [코드스니펫] 팬명록 뼈대 - `app.py`

```

from flask import Flask, render_template, request, jsonify
app = Flask(__name__)

@app.route('/')

```

```
def home():
    return render_template('index.html')

@app.route("/guestbook", methods=["POST"])
def guestbook_post():
    sample_receive = request.form['sample_give']
    print(sample_receive)
    return jsonify({'msg': 'POST 연결 완료!'})

@app.route("/guestbook", methods=["GET"])
def guestbook_get():
    return jsonify({'msg': 'GET 연결 완료!'})

if __name__ == '__main__':
    app.run('0.0.0.0', port=5000, debug=True)
```

## ▼ 2) 프로젝트 준비 - index.html 준비하기

### ▼ [코드스니펫] 팬명록 뼈대 - index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
      rel="stylesheet"
      integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTWfspd3yD65VohhpuuCOmLASjC"
      crossorigin="anonymous"
    />
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script
      src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
      integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
      crossorigin="anonymous"
    ></script>

    <title>초미니홈피 - 팬명록</title>

    <link
      href="https://fonts.googleapis.com/css2?family=Noto+Serif+KR:wght@200;300;400;500;600;700;900&display=swap"
      rel="stylesheet"
    />
    <style>
      * {
        font-family: "Noto Serif KR", serif;
      }
      .mypic {
        width: 100%;
        height: 300px;

        background-image: linear-gradient(
          0deg,
          rgba(0, 0, 0, 0.5),
          rgba(0, 0, 0, 0.5)
        ),
        url("https://cdn.topstarnews.net/news/photo/201807/456143_108614_510.jpg");
        background-position: center 30%;
        background-size: cover;

        color: white;

        display: flex;
        flex-direction: column;
        align-items: center;
        justify-content: center;
      }
      .mypost {
        width: 95%;
        max-width: 500px;
        margin: 20px auto 20px auto;

        box-shadow: 0px 0px 3px 0px black;
        padding: 20px;
      }
      .mypost > button {
        margin-top: 15px;
      }
```

```

.mycards {
  width: 95%;
  max-width: 500px;
  margin: auto;
}

.mycards > .card {
  margin-top: 10px;
  margin-bottom: 10px;
}
</style>
<script>
$(document).ready(function () {
  set_temp();
  show_comment();
});
function set_temp() {
  fetch("http://spartacodingclub.shop/sparta_api/weather/seoul").then((res) => res.json()).then((data) => {
    console.log(data)
  });
}
function save_comment() {
  let formData = new FormData();
  formData.append("sample_give", "샘플데이터");

  fetch('/guestbook', {method: "POST", body: formData}).then((res) => res.json()).then((data) => {
    //console.log(data)
    alert(data["msg"]);
  });
}
function show_comment() {
  fetch('/guestbook').then((res) => res.json()).then((data) => {
    alert(data["msg"])
  })
}
</script>
</head>
<body>
<div class="mypic">
  <h1>심센치(10cm) 팬명록</h1>
  <p>현재기온: <span id="temp">36</span>도</p>
</div>
<div class="mypost">
  <div class="form-floating mb-3">
    <input type="text" class="form-control" id="name" placeholder="url" />
    <label for="floatingInput">닉네임</label>
  </div>
  <div class="form-floating">
    <textarea
      class="form-control"
      placeholder="Leave a comment here"
      id="comment"
      style="height: 100px"
    ></textarea>
    <label for="floatingTextarea2">응원댓글</label>
  </div>
  <button onclick="save_comment()" type="button" class="btn btn-dark">
    댓글 남기기
  </button>
</div>
<div class="mycards" id="comment-list">
  <div class="card">
    <div class="card-body">
      <blockquote class="blockquote mb-0">
        <p>새로운 앨범 너무 멋져요!</p>
        <footer class="blockquote-footer">호빵맨</footer>
      </blockquote>
    </div>
  </div>
  <div class="card">
    <div class="card-body">
      <blockquote class="blockquote mb-0">
        <p>새로운 앨범 너무 멋져요!</p>
        <footer class="blockquote-footer">호빵맨</footer>
      </blockquote>
    </div>
  </div>
  <div class="card">
    <div class="card-body">
      <blockquote class="blockquote mb-0">
        <p>새로운 앨범 너무 멋져요!</p>
        <footer class="blockquote-footer">호빵맨</footer>
      </blockquote>
    </div>
  </div>
</div>

```



```
</body>
</html>
```

### ▼ 3) 프로젝트 준비 - mongoDB Atlas 창 띄워두기

#### ▼ [코드스니펫] mongoDB Atlas 주소

<https://cloud.mongodb.com/>

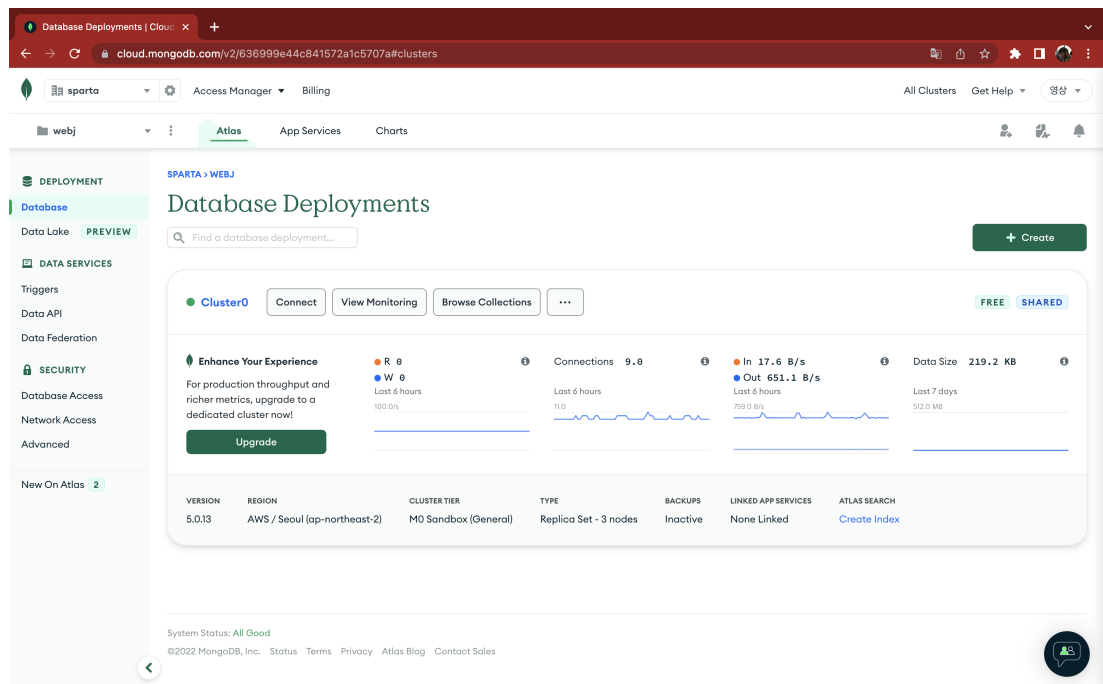
#### ▼ mongoDB 데이터베이스 들어가는 법



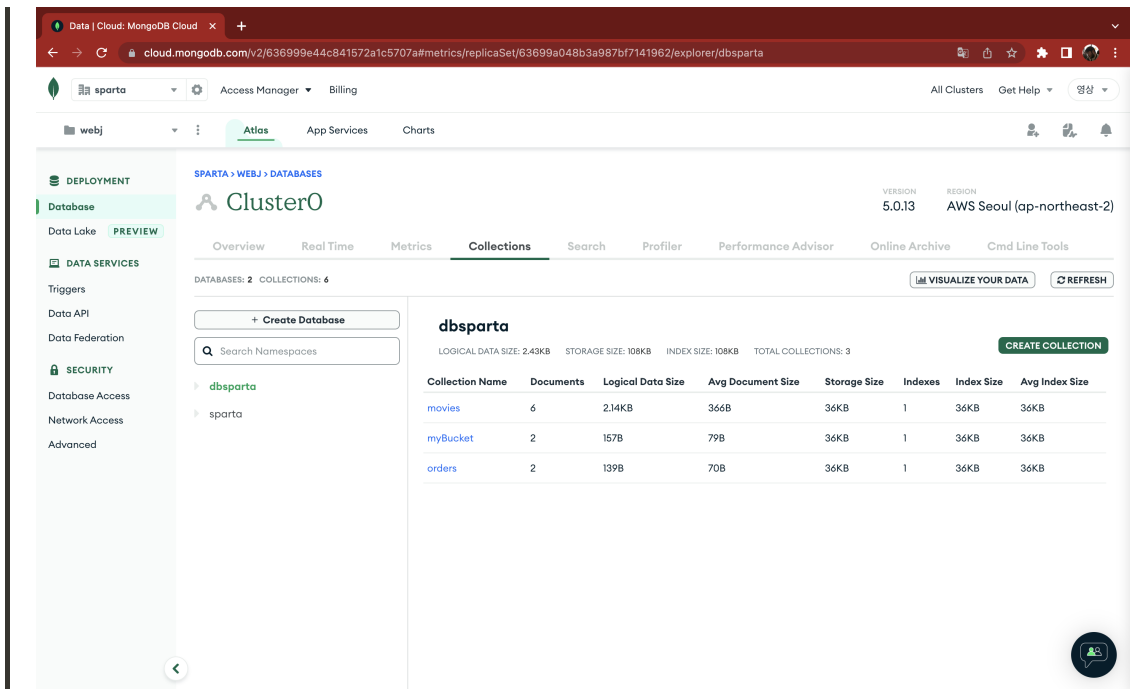
지난 주차에서 mongoDB Atlas! 기억나시죠?

- 아직 연결한 것은 없으니 보이는 것은 없지만, 이따 확인하려면 미리 띄워둬야죠!

화면 왼쪽의 Database를 클릭한 뒤 **Cluster 0** 클릭해주세요!



화면 가운데의 여러 탭 중 **Collection** 탭을 클릭해주세요!



## 09. [팬명록] - 조각기능. 날씨 API 적용하기

### ▼ 1) OpenAPI를 활용해 조각기능 구현하기

명세

#### ▼ Q. 문제

😎 스파르타코딩클럽에서 제공하는 서울 날씨 API를 적용해봅시다!

- OpenAPI를 가져다 쓰는 법을 우린 배웠었어요!
- 게다가 날씨 API도 어디선가 해보신 것 같지 않나요?

#### ▼ [코드스니펫] 서울 날씨 OpenAPI

```
"http://spartacodingclub.shop/sparta_api/weather/seoul"
```

#### ▼ [코드스니펫] index.html 날씨 API 빼대 코드

```
$(document).ready(function () {
  set_temp();
});
function set_temp() {
  fetch("http://spartacodingclub.shop/sparta_api/weather/seoul").then((res) => res.json()).then((data) => {
    console.log(data)
  });
}
```

#### ▼ 💡 힌트!

- 2주차 숙제에서 붙였었습니다!
- OpenAPI도 결국 API입니다!


- API 주소로 접속해볼까요?

우리가 데이터를 가져오는 JSON 형식!  
데이터를 읽는 것은 무엇이었죠?(Read → GET)

```
// http://spartacodingclub.shop/sparta_api/weather/seoul

{
  "city": "Seoul",
  "clouds": "0%",
  "icon": "http://openweathermap.org/img/w/01d.png",
  "temp": 10.26
}
```

#### ▼ A. 정답보기!

 OpenAPI도 결국 API입니다!

#### ▼ 1) 데이터명세

- 1.요청정보 : URL = "http://spartacodingclub.shop/sparta\_api/weather/seoul"  
요청방식 = GET
- 2.클라(fetch) → 서버(OpenAPI) : 없음
- 3.서버(OpenAPI) → 클라(fetch) : 서울 날씨 정보를 불러옴

#### ▼ 2) 클라이언트와 서버 연결 확인하기

```
$(document).ready(function () {
  set_temp();
  show_comment();
});
function set_temp() {
  fetch("http://spartacodingclub.shop/sparta_api/weather/seoul").then((res) => res.json()).then((data) => {
    console.log(data)
  });
}
```

개발자 도구를 열어 `console.log()` 가 우리가 접속하는 데이터가 잘 붙어서 오는지 확인해보세요!

```
▼ Object ⓘ
  city: "Seoul"
  clouds: "0%"
  icon: "http://openweathermap.org/img/w/01d.png"
  temp: 10.26
  ► [[Prototype]]: Object
```

### ▼ 3) 서버만들기?

👁️ OpenAPI는 이미 데이터베이스가 구축된 서버예요!

😎 우리는 가져다가 보여줄 클라이언트만 만들면 된다는 말씀!

### ▼ 4) 클라이언트 만들기

- 우리가 보여줄 값은 여러 개가 아닌 서울 지금의 날씨 딱 하나입니다!  
→ 반복문을 쓸 필요 없다는 사실!
- 가져온 데이터를 temp\_html로 뼈대에 담아 제이쿼리로 **append**!

```
function set_temp() {  
  fetch("http://spartacodingclub.shop/sparta_api/weather/seoul").then((res) => res.json()).then((data) => {  
    console.log(data)  
    $("#temp").text(data["temp"]);  
  });  
}
```

### ▼ 5) 완성 확인하기

👉 **동작 테스트:** 화면을 새로고침 했을 때, DB에 저장된 리뷰가 화면에 올바르게 나타나는지 확인합니다.

## 10. [팬명록] - POST 연습하기(응원 등록하기)

### ▼ 1) API 만들고 사용하기 - 응원 게시하기 API(Create → POST)

😎 데이터 생성은 POST 방식! 기억나시죠?

#### ▼ 1) 데이터명세

- 1.요청정보 : URL = **/guestbook**, 요청방식 = **GET**
- 2.클라(fetch) → 서버(flask) : **name**, **comment**
- 3.서버(flask) → 클라(fetch) : 메시지를 보냄 (응원 완료!)

#### ▼ 2) 클라이언트와 서버 연결 확인하기

[서버 코드 - **app.py**]

```
@app.route("/guestbook", methods=["POST"])  
def guestbook_post():  
    sample_receive = request.form['sample_give']  
    print(sample_receive)  
    return jsonify({'msg': 'POST 연결 완료!'})
```

[클라이언트 코드 - **index.html**]

```
function save_comment() {  
  let formData = new FormData();  
  formData.append("sample_give", "샘플데이터");  
  
  fetch('/guestbook', {method: "POST", body: formData}).then((response) => response.json()).then((data) => {  
    alert(data['msg'])  
  });  
}  
  
<button onclick="save_bucket()" type="button" class="btn btn-outline-primary">기록하기</button>
```

### ▼ 3) 서버부터 만들기

데이터베이스에 연결해줍시다!

```
from pymongo import MongoClient

client = MongoClient('내 URL')
db = client.dbsparta
```

- `name`, `comment` 정보를 받아서, 저장하면 되겠죠?
  - `name_receive = request.form['name_give']`
  - `comment_receive = request.form['comment_give']`
- 우리가 일전에 만들어둔 `dbtest.py` 에서 코드를 불러와봅시다!
- 잘 입력되었다는 표시를 해주기 위한 메시지를 바꿔봅시다!

```
@app.route("/guestbook", methods=["POST"])
def guestbook_post():
    name_receive = request.form["name_give"]
    comment_receive = request.form["comment_give"]

    doc = {
        'name': name_receive,
        'comment': comment_receive
    }

    db.guestbook.insert_one(doc)
    return jsonify({'msg': '응원 완료!'})
```

### ▼ 4) 클라이언트 만들기

- 이번엔 `name`, `comment` 정보를 보내주면 되겠죠?
- `formData`에 데이터를 넣고, 보내줍니다!
- 저장되면, 새로고침을 해서 다시 보여줄 준비를 합니다!
  - `window.location.reload()`

```
function save_comment() {
    let name = $("#name").val();
    let comment = $("#comment").val();

    let formData = new FormData();
    formData.append("name_give", name);
    formData.append("comment_give", comment);

    fetch('/guestbook', {method: "POST", body: formData}).then((res) => res.json()).then((data) => {
        //console.log(data)
        alert(data["msg"]);
        window.location.reload();
    });
}
```

### ▼ 5) 완성 확인하기

- 데이터를 입력하고
- DB에 잘 들어갔는지 확인해보면 되겠죠?

```
_id: ObjectId('637594c0a89dd4635b2aab49')
name: "팬123"
comment: "응원해요!"
```

명세

요청정보 guestbook 요청방식 post

## 11. [팬명록] - GET 연습하기(응원 보여주기)

▼ 1) API 만들고 사용하기 - 버킷리스트 조회 API(Read → GET)

😎 데이터 조회는 GET 방식! 기억나시죠?

▼ 1) 데이터 명세

- 1. 요청 정보 : URL= `/guestbook` , 요청 방식 = `GET`
- 2. 클라(fetch) → 서버(flask) : 없음
- 3. 서버(flask) → 클라(fetch) : 전체 응원 목록을 보여주기

▼ 2) 클라이언트와 서버 연결 확인하기

[서버 코드 - `app.py`]

```
@app.route("/guestbook", methods=["GET"])
def guestbook_get():
    return jsonify({'msg': 'GET 연결 완료!'})
```

[클라이언트 코드 - `index.html`]

```
$(document).ready(function () {
    set_temp()
    show_bucket()
});
function show_comment() {
    fetch('/guestbook').then(res => res.json()).then(data => {
        console.log(data)
        alert(data['msg'])
    })
}
```

▼ 3) 서버부터 만들기

받을 것 없이 `guestbook` 에 주문정보를 담아서 내려주기만 하면 됩니다!

```
@app.route("/guestbook", methods=["GET"])
def guestbook_get():
    comment_list = list(db.guestbook.find({}, {'_id': False}))
    return jsonify({'comments': comment_list})
```

▼ 4) 클라이언트 만들기

- 응원 목록을 `forEach` 문으로 반복하면서 데이터를 뽑아냅니다!
- 가져온 데이터를 `temp_html`로 뼈대에 데이터를 담아 제이쿼리로 `append`!
- 기존 html 코드들을 지워줄 수 있도록 `empty()` 사용하기!

```
function show_comment() {
    $("#comment-list").empty();
    fetch('/guestbook').then((res) => res.json()).then((data) => {
        //console.log(data)
        let comments = data['comments']
        comments.forEach((row) => {
            let name = row["name"];
            let comment = row["comment"];

            let temp_html = `<div class="card">
                <div class="card-body">
                    <blockquote class="blockquote mb-0">
                        <p>${comment}</p>
                        <footer class="blockquote-footer">${name}</footer>
                    </blockquote>
                </div>
            </div>`;
            $("#comment-list").append(temp_html);
        });
    });
}
```

#### ▼ 5) 완성 확인하기



**동작 테스트:** 화면을 새로고침 했을 때, DB에 저장된 리뷰가 화면에 올바르게 나타나는지 확인합니다.

#### ▼ 2) 전체 완성 코드 확인하기

##### ▼ [ 코드] app.py - 팬명록 서버

```
from flask import Flask, render_template, request, jsonify
app = Flask(__name__)

from pymongo import MongoClient
client = MongoClient("내 URL")
db = client.sparta

@app.route('/')
def home():
    return render_template('index.html')

@app.route("/guestbook", methods=["POST"])
def guestbook_post():
    name_receive = request.form["name_give"]
    comment_receive = request.form["comment_give"]

    doc = {
        'name': name_receive,
        'comment': comment_receive
    }

    db.guestbook.insert_one(doc)
    return jsonify({'msg': '응원 완료!'})

@app.route("/guestbook", methods=["GET"])
def guestbook_get():
    comment_list = list(db.guestbook.find({}, {'_id': False}))
    return jsonify({'comments': comment_list})

if __name__ == '__main__':
    app.run('0.0.0.0', port=5000, debug=True)
```

##### ▼ [ 코드] index.html - 팬명록 클라이언트

```
<!DOCTYPE html>
<html lang="en">
```

```

<head>
<meta charset="UTF-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTWfspd3yD65VohhpuuCOMLASjc"
crossorigin="anonymous"
/>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384-Mrcw6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
crossorigin="anonymous"
></script>

<title>초미니홈피 - 팬명록</title>

<link
href="https://fonts.googleapis.com/css2?family=Noto+Serif+KR:wght@200;300;400;500;600;700;900&display=swap"
rel="stylesheet"
/>
<style>
* {
font-family: "Noto Serif KR", serif;
}
.mypic {
width: 100%;
height: 300px;

background-image: linear-gradient(
0deg,
rgba(0, 0, 0, 0.5),
rgba(0, 0, 0, 0.5)
),
url("https://cdn.topstarnews.net/news/photo/201807/456143_108614_510.jpg");
background-position: center 30%;
background-size: cover;

color: white;

display: flex;
flex-direction: column;
align-items: center;
justify-content: center;
}

.mypost {
width: 95%;
max-width: 500px;
margin: 20px auto 20px auto;

box-shadow: 0px 0px 3px 0px black;
padding: 20px;
}

.mypost > button {
margin-top: 15px;
}

.mycards {
width: 95%;
max-width: 500px;
margin: auto;
}

.mycards > .card {
margin-top: 10px;
margin-bottom: 10px;
}
</style>
<script>
$(document).ready(function () {
set_temp();
show_comment();
});
function set_temp() {
fetch("http://spartacodingclub.shop/sparta_api/weather/seoul").then((res) => res.json()).then((data) => {
console.log(data)
$("#temp").text(data["temp"]);
});
}
function save_comment() {
let name = $("#name").val();
let comment = $("#comment").val();

```



```

    let formData = new FormData();
    formData.append("name_give", name);
    formData.append("comment_give", comment);

    fetch('/guestbook', {method: "POST", body: formData}).then((res) => res.json()).then((data) => {
      //console.log(data)
      alert(data["msg"]);
      window.location.reload();
    });
  }
  function show_comment() {
    $("#comment-list").empty();
    fetch('/guestbook').then((res) => res.json()).then((data) => {
      //console.log(data)
      let comments = data['comments']
      comments.forEach((row) => {
        let name = row["name"];
        let comment = row["comment"];

        let temp_html = `<div class="card">
          <div class="card-body">
            <blockquote class="blockquote mb-0">
              <p>${comment}</p>
              <footer class="blockquote-footer">${name}</footer>
            </blockquote>
          </div>
        </div>`;
        $("#comment-list").append(temp_html);
      });
    });
  }
</script>
</head>
<body>
  <div class="mypic">
    <h1>심센치(10cm) 팬명록</h1>
    <p>현재기온: <span id="temp">36</span>도</p>
  </div>
  <div class="mypost">
    <div class="form-floating mb-3">
      <input type="text" class="form-control" id="name" placeholder="url" />
      <label for="floatingInput">닉네임</label>
    </div>
    <div class="form-floating">
      <textarea
        class="form-control"
        placeholder="Leave a comment here"
        id="comment"
        style="height: 100px"
      ></textarea>
      <label for="floatingTextarea2">응원댓글</label>
    </div>
    <button onclick="save_comment()" type="button" class="btn btn-dark">
      댓글 남기기
    </button>
  </div>
  <div class="mycards" id="comment-list">
    <div class="card">
      <div class="card-body">
        <blockquote class="blockquote mb-0">
          <p>새로운 앨범 너무 멋져요!</p>
          <footer class="blockquote-footer">호행맨</footer>
        </blockquote>
      </div>
    </div>
    <div class="card">
      <div class="card-body">
        <blockquote class="blockquote mb-0">
          <p>새로운 앨범 너무 멋져요!</p>
          <footer class="blockquote-footer">호행맨</footer>
        </blockquote>
      </div>
    </div>
    <div class="card">
      <div class="card-body">
        <blockquote class="blockquote mb-0">
          <p>새로운 앨범 너무 멋져요!</p>
          <footer class="blockquote-footer">호행맨</footer>
        </blockquote>
      </div>
    </div>
  </div>
</body>
</html>

```

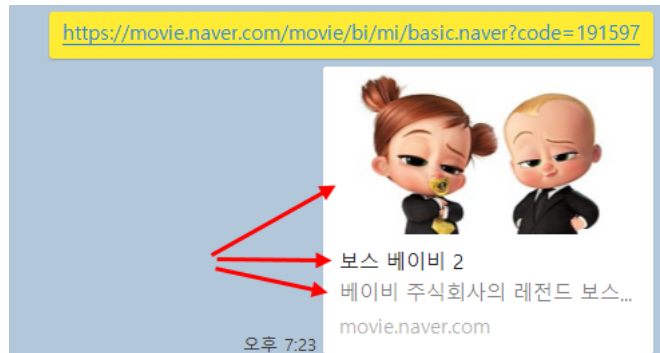
## 12. og 태그

### ▼ 1) og 태그 만들기

- 스파르타피아에서 배웠던 og:image, og:title, og:description 태그 기억하시나요?



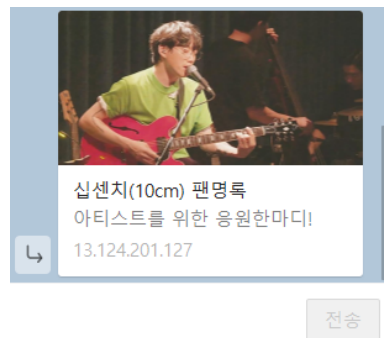
내 프로젝트도 카톡/페이스북/슬랙에 공유했을 때 예쁘게 나오도록,  
미리 꾸며봅시다!



- static 폴더 아래에 이미지 파일을 넣고, 각자 프로젝트 HTML의 <head>~</head> 사이에 아래 내용을 작성하면 og 태그를 개인 프로젝트에 사용할 수 있습니다.

### ▼ [코드스니펫] og태그 넣기

```
<meta property="og:title" content="내 사이트의 제목" />
<meta property="og:description" content="보고 있는 페이지의 내용 요약" />
<meta property="og:image" content="이미지URL" />
```



- 참고! 이미지를 바꿨는데 이전 og:image가 그대로 나와요!



그것은 페이스북/카카오톡 등에서 처음 것을 한동안 저장해놓기 때문입니다.

- 페이스북 og 태그 초기화 하기: <https://developers.facebook.com/tools/debug/>
- 카카오톡 og 태그 초기화 하기: <https://developers.kakao.com/tool/clear/og>

## 13. 내 프로젝트를 서버에 올리기

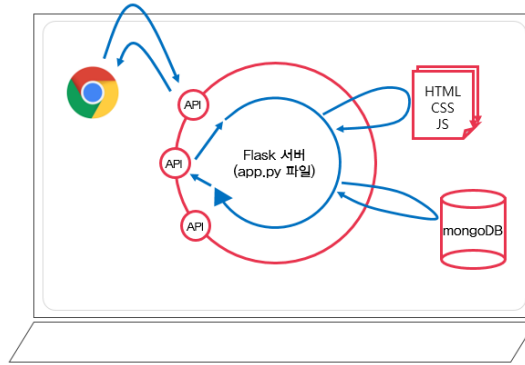
### ▼ 1) 웹서비스 런칭에 필요한 개념 소개

😎 이제 내가 만든 프로그램을 배포해봅시다. 웹 서비스를 런칭하는 거죠!

#### ▼ 1) 로컬 서버에서 클라우드 서버로!

로컬 개발 환경! 기억나시나요?

우리는 컴퓨터가 한 개니까 서버와 클라이언트를 하나의 컴퓨터에서 동작시켰어요!



😎 이젠 내 컴퓨터를 켜놓지 않아도 접근할 수 있는 웹 서비스를 목표로 가봅시다!

- 웹 서비스를 런칭하기 위해 클라이언트의 요청에 항상 응답해줄 수 있는 서버에 프로젝트를 실행시켜줄 거예요.
- 언제나 요청에 응답하려면,
  - 1) 컴퓨터가 **항상** 켜져있고 프로그램이 실행되어 있어야하고,
  - 2) 모두가 접근할 수 있는 공개 주소인 공개 IP 주소(Public IP Address)로 나의 웹 서비스에 **접근할 수 있도록** 해야해요.
- 서버는 그냥 컴퓨터라는거 기억나시죠? 외부 접속이 가능하게 설정한 다음에 내 컴퓨터를 서버로 사용할 수도 있어요.

#### ▼ 2) 컴퓨터 구매하기

😎 내 컴퓨터 대신 계속 켜놓을 컴퓨터를 구매해야합니다! 그걸 **서버** 라고 불러요!

이렇게 생긴 서버, 미디어에서 보신 적이 있나요?



🤔 이런 컴퓨터를 살 돈이 없대구요? 걱정 마세요, 빌릴거니깐요!

- 우리는 컴퓨터를 직접 사지 않을 겁니다! AWS라는 클라우드 서비스에서 **컴퓨터를 빌릴겁니다!**
- AWS 들어보신적 있으신가요? Amazon Web Service는 우리가 배포할 수 있도록 **컴퓨터를 빌려주는 곳이에요!**
- Amazon이 위 이미지와 같이 구축해놓은 데이터센터의 컴퓨터 한 개를 빌려서 우리도 배포를 해보는 거죠!

### ▼ 3) 업로드하기

- 마지막으로 내가 만든 코딩 파일들을 업로드하고 실행시켜두면 끝!

### ▼ 2) AWS Elastic Beanstalk

😎 너무너무 쉬운 AWS의 배포서비스를 사용해봐요!

- AWS는 컴퓨터를 빌려주는 서비스부터 배포와 관련된 정~말 많은 서비스를 제공해요!
- 그렇지만 배포는 초보자가 하기에 아주 쉽지는 않아요! 신경써야 하는 내용이 많고 복잡해서..

- 그래서 초보자를 위해서 **빠르게 똑딱 배포할 수 있는 서비스**도 제공한답니다!
- 그것이 바로 **AWS Elastic Beanstalk**!

Aws에서 제공하는 쉬운 배포서비스.

코드를 압축하여 배포까지 관리함.

배포링크, 코드 업데이트 기능까지 제공

## 14. AWS Elastic Beanstalk으로 배포하기

### ▼ 1) 배포 준비하기

[https://s3-us-west-2.amazonaws.com/secure.notion-static.com/31a855a4-c583-4e37-a85a-c5780e86037c/2022-11-19\\_16-30-03.mkv](https://s3-us-west-2.amazonaws.com/secure.notion-static.com/31a855a4-c583-4e37-a85a-c5780e86037c/2022-11-19_16-30-03.mkv)

### ▼ [코드스니펫] AWS Elastic Beanstalk

<https://ap-northeast-2.console.aws.amazon.com/elasticbeanstalk/home?region=ap-northeast-2#/welcome>

### ▼ [코드스니펫] 배포 명령어 모음

```
- 터미널 준비하기 -
mkdir deploy
cp app.py deploy/application.py
cp -r templates deploy/templates
pip freeze > deploy/requirements.txt
cd deploy

- application.py 세팅하기 -
application = app = Flask(__name__)
app.run()

- 패키지 설치하기 -
pip install awsebcli


- 보안 자격증명 -
eb init

- 초기 설정 -
eb create myweb

- 코드 수정 & 업데이트 -
eb deploy myweb
```

Pip install awsebcli


### ▼ 2) awsebcli 패키지 설치하기

 내 컴퓨터 터미널에서 배포를 해본다구요!

- 터미널을 열어봅시다!
- `pip install awsebcli` 입력하고 엔터!

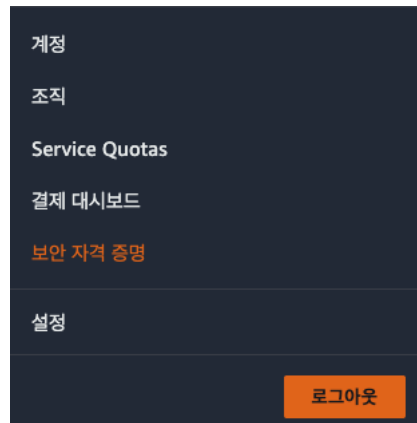
### ▼ 3) awsebcli로 AWS Elastic Beanstalk 배포하기

#### ▼ Eb init

 내 컴퓨터에서 로그인을 하려면 보안 자격이 필요합니다!

- AWS 콘솔에 들어가, 오른쪽 위 부분의 계정이름을 클릭합니다!

보안 자격 증명 을 클릭!



보안 자격 증명에서 액세스 키를 열고, 새 액세스 키 만들기 클릭!

### 보안 자격 증명

이 페이지를 사용하여 AWS 계정의 자격 증명을 관리합니다. AWS IAM(Identity and Access Management) 사용자에게 대한 자격 증명을 관리하려면 IAM 콘솔 을(를) 사용하십시오.

AWS 자격 증명 유형과 사용 방법에 대해 자세히 알아보려면 AWS 일반 참조의 [AWS 보안 자격 증명](#) 을(를) 참조하십시오.

▲ 비밀번호

▲ 멀티 팩터 인증(MFA)

▼ 액세스 키(엑세스 키 ID 및 보안 액세스 키)

엑세스 키를 사용하여 AWS CLI, PowerShell용 도구, AWS SDK 또는 직접 AWS API 호출을 통해 AWS를 프로그래밍 방식으로 호출합니다. 한 번에 최대 두 개의 액세스 키(활성 또는 비활성)를 가질 수 있습니다.

보호를 위해 비밀 키를 다른 사람과 공유해서는 안 됩니다. 또한 키를 자주 교체하는 것이 좋습니다.

비밀 키는 생성 시에만 보거나 다운로드할 수 있습니다. 기존 비밀 키를 잘못 보관한 경우 새 액세스 키를 생성하십시오. 자세히 알아보기

생성 완료	엑세스 키 ID	마지막 사용	마지막으로 사용한 리전	마지막으로 사용한 서비스
11월 17일 2022 11월 9일 2022	AKIAZDI7MS5KQJN4ZDMF AKIAZDI7MS5KFJR5LEMD	해당 사항 없음 2022-11-16 18:01 UTC+0900	해당 사항 없음 ap-northeast-2	해당 사항 없음 s3

새 액세스 키 만들기

루트 사용자 액세스 키는 전체 AWS 계정에 대한 무제한 액세스를 제공합니다. 장기 액세스 키가 필요한 경우 제한된 권한이 부여된 새 IAM 사용자를 생성하고 해당 사용자에게 대한 액세스 키를 루트 사용자 액세스 키 대신 생성하는 :

▲ CloudFront 키 페어

▲ X.509 인증서

▲ 계정 ID

엑세스 키 보기를 눌러 액세스 키 ID 와 보안 액세스 키 를 복사해서 메모장에 붙여주세요!

### 엑세스 키 만들기

✅ 액세스 키(엑세스 키 ID 및 보안 액세스 키)가 성공적으로 생성되었습니다.

새 액세스 키 ID와 보안 액세스 키가 포함된 키 파일을 지금 다운로드하십시오. 키 파일을 지금 다운로드하지 않으면 보안 액세스 키를 다시 가져올 수 없습니다.

보안을 유지하기 위해 보안 액세스 키를 안전하게 보관하고 다른 사람과 공유하지 마십시오.

▼ 액세스 키 숨기기

엑세스 키 ID: AKIAZDI7MS5KQJN4ZDMF

보안 액세스 키: AKIAZDI7MS5KFJR5LEMD

키 파일 다운로드

닫기

- 터미널에서 `pip install awscli` 를 입력하고 엔터!
- 터미널에서 `aws configure` 를 입력하고 엔터를 누르면! 아래와 같은 항목을 입력받습니다!


```
AWS Access Key ID [None]: 발급된 액세스 키 ID (복사 붙여넣기)
AWS Secret Access Key [None]: 발급된 비밀 액세스 키 (복사 붙여넣기)
Default region name [None]: ap-northeast-2 (서울)
Default output format [None]: json
```

- 전부 입력하면 보안 자격 증명 준비 끝!

Eb create fan


Eb deploy fan

## ▼ 2) 배포 프로젝트 준비하기

 AWS 공식문서 참조 ([링크](#))

- Elastic Beanstalk 배포를 위해 필요한 것이 있어요!
  - `application.py` ← `app.py`에서 이름을 바꿔줘야해요!
  - `templates`폴더와 `index.html`
  - `requirements.txt` ← 우리가 설치한 패키지들의 이름이 담긴 txt파일이에요!
  - **폴더 안에 venv가 있으면 안돼요!**

## ▼ 1) 폴더 세팅하기

 **sparta – deploy** 폴더에서 진행해봅시다!

### ▼ [코드스니펫] - `application.py` 배포 코드

```
from flask import Flask, render_template, request, jsonify
application = app = Flask(__name__)

from pymongo import MongoClient
client = MongoClient("내 URL")
db = client.sparta

@app.route('/')
def home():
    return render_template('index.html')

@app.route("/guestbook", methods=["POST"])
def guestbook_post():
    name_receive = request.form["name_give"]
    comment_receive = request.form["comment_give"]

    doc = {
        'name': name_receive,
        'comment': comment_receive
    }

    db.guestbook.insert_one(doc)
    return jsonify({'msg': '응원 완료!'})

@app.route("/guestbook", methods=["GET"])
def guestbook_get():
    comment_list = list(db.guestbook.find({}, {'_id': False}))
    return jsonify({'comments': comment_list})

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

### ▼ [코드스니펫] - index.html 배포 코드

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTWfSpd3yD65VohhpuuC0mLASjC"
crossorigin="anonymous"
/>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384-Mrcw6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
crossorigin="anonymous"
></script>

<title>초미니홈피 - 팬명록</title>

<link
href="https://fonts.googleapis.com/css2?family=Noto+Serif+KR:wght@200;300;400;500;600;700;900&display=swap"
rel="stylesheet"
/>
<style>
* {
font-family: "Noto Serif KR", serif;
}
.mypic {
width: 100%;
height: 300px;

background-image: linear-gradient(
0deg,
rgba(0, 0, 0, 0.5),
rgba(0, 0, 0, 0.5)
),
url("https://cdn.topstarnews.net/news/photo/201807/456143_108614_510.jpg");
background-position: center 30%;
background-size: cover;

color: white;

display: flex;
flex-direction: column;
align-items: center;
justify-content: center;
}

.mypost {
width: 95%;
max-width: 500px;
margin: 20px auto 20px auto;

box-shadow: 0px 0px 3px 0px black;
padding: 20px;
}

.mypost > button {
margin-top: 15px;
}

.mycards {
width: 95%;
max-width: 500px;
margin: auto;
}

.mycards > .card {
margin-top: 10px;
margin-bottom: 10px;
}
</style>
<script>
$(document).ready(function () {
set_temp();
show_comment();
});
function set_temp() {
fetch("http://spartacodingclub.shop/sparta_api/weather/seoul").then((res) => res.json()).then((data) => {
console.log(data)

```



```

        $("#temp").text(data["temp"]);
    });
}
function save_comment() {
    let name = $("#name").val();
    let comment = $("#comment").val();

    let formData = new FormData();
    formData.append("name_give", name);
    formData.append("comment_give", comment);

    fetch('/guestbook', {method: "POST", body: formData}).then((res) => res.json()).then((data) => {
        //console.log(data)
        alert(data["msg"]);
        window.location.reload();
    });
}
function show_comment() {
    $("#comment-list").empty();
    fetch('/guestbook').then((res) => res.json()).then((data) => {
        //console.log(data)
        let comments = data['comments']
        comments.forEach((row) => {
            let name = row["name"];
            let comment = row["comment"];

            let temp_html = `<div class="card">
                <div class="card-body">
                    <blockquote class="blockquote mb-0">
                        <p>${comment}</p>
                        <footer class="blockquote-footer">${name}</footer>
                    </blockquote>
                </div>
            </div>`;
            $("#comment-list").append(temp_html);
        });
    });
}
</script>
</head>
<body>
<div class="mypic">
<h1>심센치(10cm) 팬명록</h1>
<p>현재기온: <span id="temp">36</span>도</p>
</div>
<div class="mypost">
<div class="form-floating mb-3">
<input type="text" class="form-control" id="name" placeholder="url" />
<label for="floatingInput">닉네임</label>
</div>
<div class="form-floating">
<textarea
    class="form-control"
    placeholder="Leave a comment here"
    id="comment"
    style="height: 100px"
></textarea>
<label for="floatingTextarea2">응원댓글</label>
</div>
<button onclick="save_comment()" type="button" class="btn btn-dark">
    댓글 남기기
</button>
</div>
<div class="mycards" id="comment-list">
<div class="card">
<div class="card-body">
<blockquote class="blockquote mb-0">
<p>새로운 앨범 너무 멋져요!</p>
<footer class="blockquote-footer">호빵맨</footer>
</blockquote>
</div>
</div>
<div class="card">
<div class="card-body">
<blockquote class="blockquote mb-0">
<p>새로운 앨범 너무 멋져요!</p>
<footer class="blockquote-footer">호빵맨</footer>
</blockquote>
</div>
</div>
<div class="card">
<div class="card-body">
<blockquote class="blockquote mb-0">
<p>새로운 앨범 너무 멋져요!</p>
<footer class="blockquote-footer">호빵맨</footer>
</blockquote>
</div>
</div>

```

```

    </div>
  </div>
</body>
</html>

```

- 데이터베이스 연결 코드만 다시 적어주세요!

#### ▼ 1) application.py 설정( 그대로 파일을 가져다 쓴다면! )

- app.py를 application.py로 바꿔주세요!
- `application` 을 읽을 수 있도록 코드를 바꿔주세요!
- `app.run()` 내용을 지워주세요!

```

from flask import Flask, render_template, request, jsonify
application = app = Flask(__name__)

from pymongo import MongoClient
client = MongoClient("내 URL")
db = client.sparta

@app.route('/')
def home():
    return render_template('index.html')

@app.route("/guestbook", methods=["POST"])
def guestbook_post():
    name_receive = request.form["name_give"]
    comment_receive = request.form["comment_give"]

    doc = {
        'name': name_receive,
        'comment': comment_receive
    }

    db.guestbook.insert_one(doc)
    return jsonify({'msg': '응원 완료!'})

@app.route("/guestbook", methods=["GET"])
def guestbook_get():
    comment_list = list(db.guestbook.find({}, {'_id': False}))
    return jsonify({'comments': comment_list})

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

```

#### ▼ 2) requirements.txt 만들기



가상환경 폴더를 숨겼으니, 우리가 사용할 패키지 이름을 알려줘야 합니다!

#### ▼ [코드스니펫] requirements.txt 만들기 코드

```
pip freeze > requirements.txt
```

- 코드스니펫의 코드를 입력하고 엔터를 누르시면 파일이 생성됩니다! 신기하죠?

#### ▼ 3) 배포 초기 설정, 생성 및 배포

##### ▼ 1) 배포 초기 설정

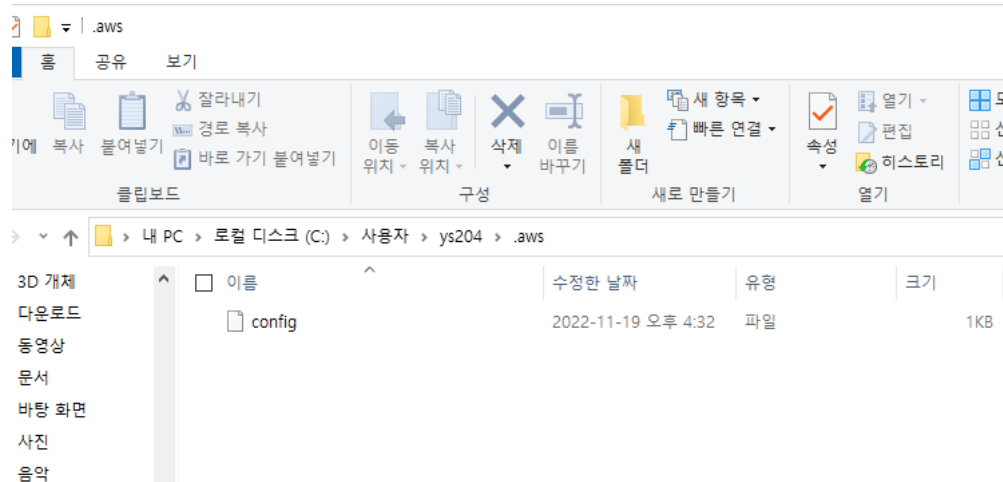


코드 한 줄로 초기설정, `eb init`

- 배포할 환경을 만들 초기 설정을 진행합니다!
- 터미널을 열어줍니다. `eb init` 을 입력하고 엔터!

▼ 📌 참조) 만약 액세스키를 입력하는데 오류가 발생한다면?

내 컴퓨터 → C 드라이브 → 사용자 → (사용자이름) → .aws 폴더에서 config, credential 파일을 삭제해주세요!



```
Select a default region : 10 (seoul)
Select an application to use : myweb [Create new Application]
It appears you are using Python. Is this correct? : Y
Select a platform branch : enter

---

Cannot setup CodeCommit because there is no Source Control setup, continuing with initialization
Do you want to set up SSH for your instances? : Y

Select a keypair : enter [Create new Keypair]
```

▼ 👁 상세히 보기!

터미널에서 `eb init` 을 입력하고 엔터를 누릅니다!

```
ts/05.fan/venv/bin/python3 -m pip install --upgrade pip'
(venv) Yungsangui-MacBookAir:05.fan migdracios$ eb init
```

배포할 서버가 있는 지역을 선택합니다

우리는 대한민국에 있으니, `10번 서울` 을 골라야겠죠!

```
Select a default region
1) us-east-1 : US East (N. Virginia)
2) us-west-1 : US West (N. California)
3) us-west-2 : US West (Oregon)
4) eu-west-1 : EU (Ireland)
5) eu-central-1 : EU (Frankfurt)
6) ap-south-1 : Asia Pacific (Mumbai)
7) ap-southeast-1 : Asia Pacific (Singapore)
8) ap-southeast-2 : Asia Pacific (Sydney)
9) ap-northeast-1 : Asia Pacific (Tokyo)
10) ap-northeast-2 : Asia Pacific (Seoul)
11) sa-east-1 : South America (Sao Paulo)
12) cn-north-1 : China (Beijing)
13) cn-northwest-1 : China (Ningxia)
14) us-east-2 : US East (Ohio)
```

```
22) at-south-1 : Africa (Cape Town)
(default is 3): 10
```

어플리케이션을 만듭니다!

엔터를 눌러 새로운 어플리케이션을 생성하세요!

```
7) [ Create new Application ]
(default is 7):
```

어플리케이션 이름을 입력합니다

이름 입력 없이 엔터를 누르면 폴더 이름으로 생성됩니다!

```
Enter Application Name
(default is "05.fan"): myWeb
```

우리가 사용하는 프로젝트가 파이썬인 것을 확인해네요! Y를 눌러주세요!

```
It appears you are using Python. Is this correct?
(Y/n): Y
```

파이썬 버전을 고르는 항목입니다. 엔터를 눌러 넘어가죠!

```
Select a platform branch.
1) Python 3.8 running on 64bit Amazon Linux 2
2) Python 3.7 running on 64bit Amazon Linux 2
(default is 1):
```

코드커밋이 없어 SSH인증 설정을 하겠다는 거예요! **Y를 눌러줍니다!**

```
Cannot setup CodeCommit because there is no Source Control setup, continuing with initialization
Do you want to set up SSH for your instances?
(Y/n): Y
```

키페어를 고르는 항목입니다. **엔터를 눌러주세요!**

```
4) [ Create new KeyPair ]
(default is 3): 4
```


키페어 이름을 적는 항목이에요! 기입하지 않으면 aws-eb로 만들어져요!

```
Type a keypair name.
(Default is aws-eb2): myKey
```

키페어 비밀번호를 입력하는 부분이에요! **엔터로 넘어갑니다!**

```
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
```

## ▼ 2) 배포 환경 생성 및 배포

 코드 한 줄로 배포완료 **eb create**

**eb create** **환경이름** 을 입력해주시고 엔터!

```
bash: eb create: command not found
(venv) Yungsangui-MacBookAir:05.fan migdracios$ eb create myWeb
```

생성까지 몇 분 걸려요!

이 화면이 터미널에서 뜬다면 완료!

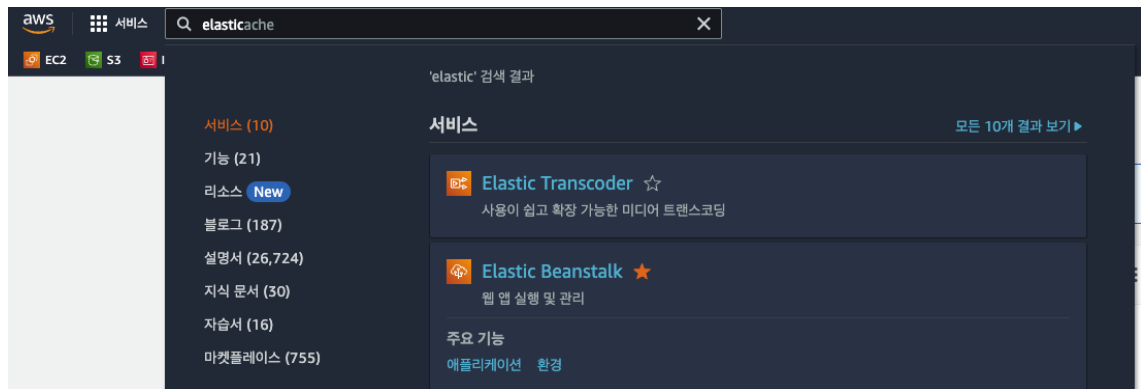
```

INVnyApCik06
2022-11-17 05:10:52 INFO Created CloudWatch alarm named: awseb-e-pcz3fptg
rmLow-FKB7KZYI4J63
2022-11-17 05:10:52 INFO Created CloudWatch alarm named: awseb-e-pcz3fptg
rmHigh-DILK3Q05MUSI
2022-11-17 05:11:02 INFO Instance deployment successfully generated a 'P
2022-11-17 05:11:04 INFO Instance deployment completed successfully.
2022-11-17 05:11:21 INFO Application available at myweb3333.eba-wbzhzgv5.
talk.com.
2022-11-17 05:11:21 INFO Successfully launched environment: myweb3333
(venv) Yungsangui-MacBookAir:deploy migdracios$ 

```

#### ▼ 4) 배포 환경 접속하기

AWS에 접속 → AWS 콘솔 연결하기 → Elastic Beanstalk



생성한 환경 이름 확인 → OK 표시가 되어있는 지 확인

<input type="radio"/>	myweb3333	<input checked="" type="radio"/>	myWeb33	2022-11-17 14:08:56 UTC+0900	2022-11-17 14:11:21 UTC+0900	myweb3333.eba-wbzhzgv5.ap- northeast- 2.elasticbeanstalk.com
-----------------------	-----------	----------------------------------	---------	------------------------------------	------------------------------------	--

- 화면에서 myWeb → 배포 주소 클릭!

코드 업데이트 적용하기

Aws console Elastic Bean

#### ▼ 5) 코드 업데이트 적용하기



코드 업데이트도 한 줄로 써주기만 하면 그대로 업데이트가 됩니다!

- 변경할 코드를 적고 저장해주세요!
- 터미널을 열어줍니다!
- 터미널에서 `eb depLoy 환경이름` 을 입력하고 엔터!
- 조금 기다리면 적용이 됩니다!

## ▼ 6) 주의사항

### ▼ 1) eb create를 할 때마다 서버를 구매한다는 사실!

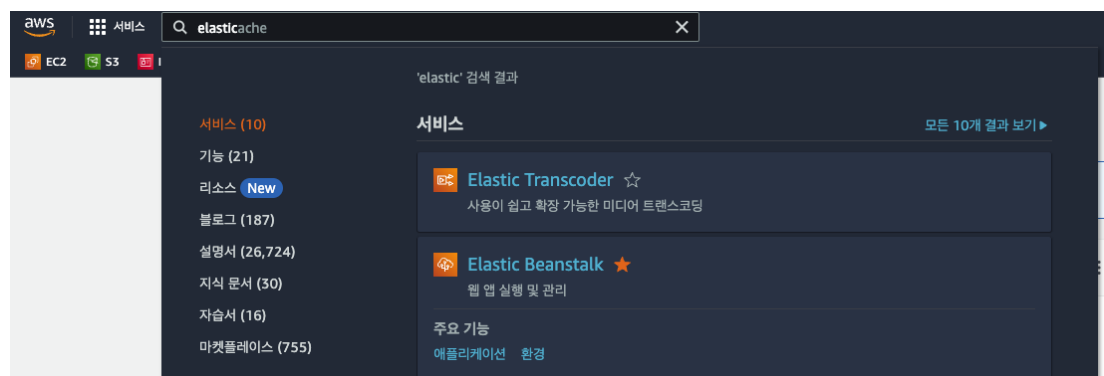
😎 Elastic Beanstalk은 eb create 환경이름 명령어를 입력할 때 마다 서버를 구매해요!

- 실수로 잘못 만들었다면, 서버를 여러 개를 빌린 것이 되는거예요!
- 안쓰는 서버를 빌릴 이유도 없고, **혹여나 과금이 될 수도 있기에!**

### ▼ 2) Elastic Beanstalk 환경 종료하기

- Elastic Beanstalk 환경을 종료하면, 배포 종료와 동시에 서버도 반납해요!

AWS → AWS 콘솔 접속하기 → Elastic Beanstalk에 들어가주세요!



종료하고 싶은 환경의 체크박스를 클릭하고, 오른쪽 위의 작업 → **환경 종료하기** 클릭!

Elastic Beanstalk > 환경								
모든 환경								
Q 표시 값과 일치하는 결과 필터링								
환경 이름 ▲	상태 ▼	애플리케이션 이름 ▼	생성 날짜 ▼	마지막 수정 ▼	URL ▼	실행 중인 버전 ▼	플랫폼 ▼	
<input type="radio"/> deploy(종료됨)	-	bucket-list	2022-11-15 23:32:16 UTC+0900	2022-11-17 14:05:26 UTC+0900	deploy.eba-rpd3bzaw.ap-northeast-2.elasticbeanstalk.com	app-221116_175835122619	Python 3.8 running on 64bit Amazon Linux 2	
<input type="radio"/> deploy0101(종료됨)	-	bucket-list	2022-11-16 17:44:24 UTC+0900	2022-11-17 14:05:44 UTC+0900	deploy0101.eba-rpd3bzaw.ap-northeast-2.elasticbeanstalk.com	app-221116_180116120295	Python 3.8 running on 64bit Amazon Linux 2	
<input type="radio"/> free10(종료됨)	-	deploy	2022-11-16 10:03:01 UTC+0900	2022-11-17 14:03:23 UTC+0900	free10.eba-3ku2hj4t.ap-northeast-2.elasticbeanstalk.com	app-221116_143654784537	Python 3.8 running on 64bit Amazon Linux 2	Supported
<input type="radio"/> free4(종료됨)	-	free-202212	2022-11-15 23:08:41 UTC+0900	2022-11-17 14:03:11 UTC+0900	free4.eba-2cj26e4p.ap-northeast-2.elasticbeanstalk.com	app-cccc3-221115_230839373424	Python 3.8 running on 64bit Amazon Linux 2	Supported
<input type="radio"/> myweb333(종료됨)	-	myWeb	2022-11-17 14:03:58 UTC+0900	2022-11-17 14:30:45 UTC+0900	myweb333.eba-cmm5m2yy.ap-northeast-2.elasticbeanstalk.com	app-221117_140349021063	Python 3.8 running on 64bit Amazon Linux 2	Supported
<input checked="" type="radio"/> myweb3333	OK	myWeb33	2022-11-17 14:08:56 UTC+0900	2022-11-17 14:41:43 UTC+0900	myweb3333.eba-wbzhgv5.ap-northeast-2.elasticbeanstalk.com	app-221117_144119535419	Python 3.8 running on 64bit Amazon Linux 2	Supported


종료할 **환경의 이름을 다시 한 번 입력** 하고 확인!

환경 종료 확인

✕

myweb3333을(를) 영구히 종료하시겠습니까? 이 작업은 실행 취소할 수 없습니다.

- 계층: 웹 서버
- 플랫폼: Python 3.8 running on 64bit Amazon Linux 2/3.4.1
- 버전: app-221117\_144119535419
- 마지막 수정: 2022-11-17 14:41:43 UTC+0900

 이 환경을 종료하면 연결된 리소스도 종료됩니다.

- URL - myweb3333.eba-wbzhgvs.ap-northeast-2.elasticbeanstalk.com이(가) 릴리스됩니다.
- 추가 리소스 - 이 환경과 연결된 모든 리소스도 종료됩니다.

확인을 위해 환경의 이름을 입력하십시오.


myweb3333

대/소문자 구분

취소

종료

## 15. 5주차 끝 & 숙제 설명

 **내 도메인을 제출해주세요!**

착착착 잘 따라오셨다면, 팬명록을 AWS에 잘 올려두셨을 거예요.

- 열심히 돌아가고 있는 나의 웹서비스를 제출하시면, 이번 숙제는 끝입니다~! 😊