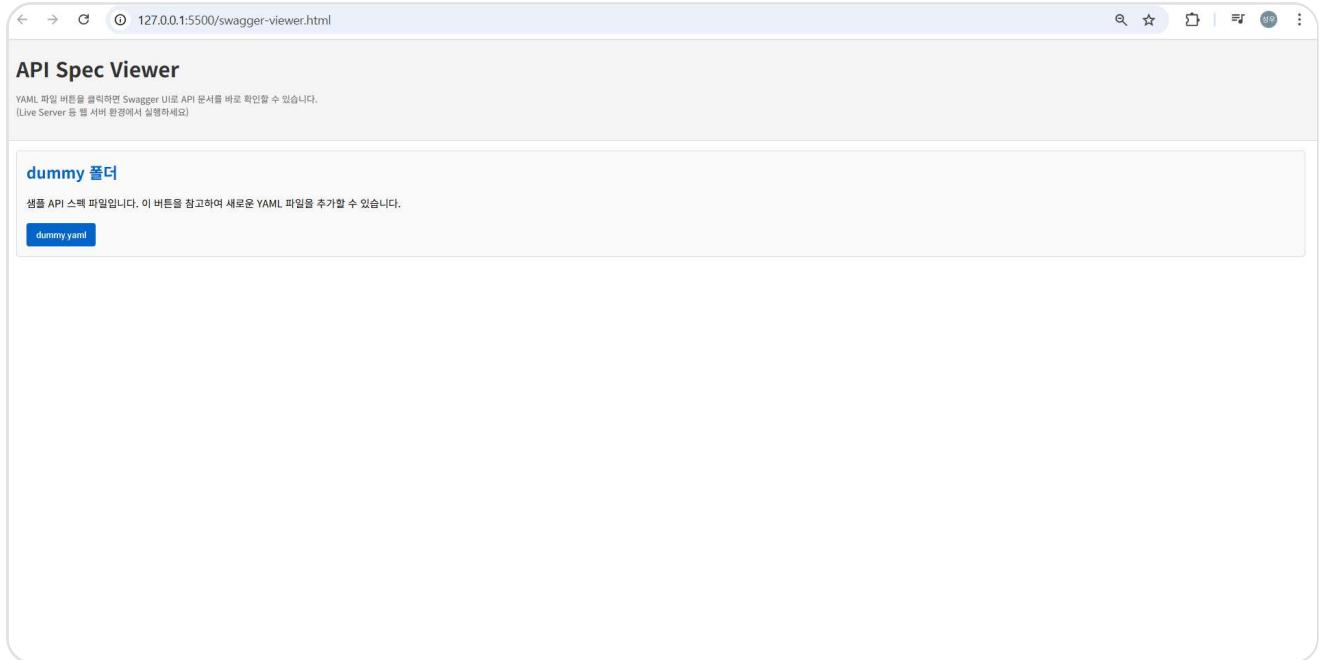


API(최성우)

본 문서는 임시 페이지입니다. 상세 API 명세는 [sungwoo-fullstack](#) 브랜치의 [swagger-viewer.html](#)을 웹 서버 환경(Live Server 등)에서 실행하여 확인하실 수 있습니다.



A screenshot of the Swagger UI showing a sample API. The title is 'Sample API 1.0.0 OAS 3.0' with the URL http://127.0.0.1:5500/api/spec/dummy/dummy.yaml. It says '이것은 샘플 API 문서입니다.' Below is a 'Servers' dropdown set to 'https://api.example.com/v1 - Production server'. The main area shows a 'default' section with a 'GET /users' operation. The description is '모든 사용자 목록을 반환합니다'. It has a 'Parameters' section with 'No parameters' and a 'Responses' section. There is a 'Try it out' button in the top right of the operation panel.

Backend API Integration Guide

본 문서는 프론트엔드에서 백엔드 API 연동이 필요한 지점을 정리한 문서입니다.

***"테스트 & 실행" 기능은 프론트엔드 시뮬레이션이므로 제외**되었습니다.

```

---
## Data Models
### Workspace
```typescript
interface Workspace {
 id: string;
 name: string;
 description?: string;
 createdAt: Date;
 functionCount: number; // 백엔드에서 계산
 invocations24h: number; // 백엔드에서 집계
 errorRate: number; // 백엔드에서 계산
}
```
```
FunctionConfig
```typescript
interface FunctionConfig {
  id: string;
  workspaceId: string;
  name: string;
  description?: string;
  runtime: string;           // "Python 3.12"
  memory: number;            // 128 | 256 | 512 | 1024 (MB)
  timeout: number;           // 1~900 (seconds)
  httpMethods: string[];     // ['GET', 'POST', 'PUT', 'DELETE', 'PATCH']
  environmentVariables: Record<string, string>; // Key-Value pairs
  code: string;               // Base64 Encoded Python source code
  invocationUrl: string | null; // 함수 호출 URL(배포 완료 후 생성)
  status: 'active' | 'disabled';
  lastModified: Date;
  lastDeployed?: Date;
  invocations24h: number;     // 백엔드에서 집계
  errors24h: number;          // 백엔드에서 집계
  avgDuration: number;        // 백엔드에서 계산 (ms)
}
```
```

```

중요 필드:

- `code`: **Base64 인코딩 필수** (따옴표/개행 문자 파싱 예러 방지)
- `invocationUrl`: 함수 배포 완료 시 생성, 미완료 시 `null`

ExecutionLog

```

```typescript
interface ExecutionLog {
 id: string;
 functionId: string;
 timestamp: Date;
 status: 'success' | 'error';
 duration: number; // ms
}

```

```
statusCode: number;
requestBody?: any;
responseBody?: any;
logs: string[];
level: 'info' | 'warn' | 'error';
}
```
---
```

API Endpoints by User Action

1 워크스페이스 생성

UI: `/` (Landing Page) → "워크스페이스 만들기" 버튼

Request:

```http

POST /api/workspaces

Content-Type: application/json

```
{
 "name": "Production",
 "description": "Production environment functions" // optional
}
```

```

Response (201 Created):

```json

```
{
 "id": "ws-abc123",
 "name": "Production",
 "description": "Production environment functions",
 "createdAt": "2025-11-30T12:00:00Z",
 "functionCount": 0,
 "invocations24h": 0,
 "errorRate": 0
}
```

```

Frontend Action:

- 생성 후 `/workspaces/{id}`로 이동

- `setCurrentWorkspaceId(id)` 호출

2 워크스페이스 목록 조회

UI: `/` (Landing Page) - 워크스페이스 카드 목록 표시

Request:

```http

GET /api/workspaces

```

Response (200 OK):

```json

```
[
 {
 "id": "ws-1",
```

```
"name": "Production",
"description": "Production environment functions",
"createdAt": "2025-12-01T00:00:00Z",
"functionCount": 1,
"invocations24h": 8420,
"errorRate": 0.2
}
]
...

```

### ### 3 워크스페이스 수정

**\*\*UI\*\*:** `/workspaces/{workspaceId}/settings` → Settings 탭 → "변경사항 저장" 버튼

**\*\*Request\*\*:**

```
```http
PATCH /api/workspaces/{workspaceId}
Content-Type: application/json
{
  "name": "Updated Name",
  "description": "Updated description"
}
...```

```

****Response**** (200 OK):

```
```json
{
 "id": "ws-1",
 "name": "Updated Name",
 "description": "Updated description",
 "createdAt": "2025-12-01T00:00:00Z",
 "functionCount": 1,
 "invocations24h": 8420,
 "errorRate": 0.2
}
...```

```

**\*\*Validation\*\*:**

- `name`은 필수 (빈 문자열 불가)

---

### ### 4 워크스페이스 삭제

**\*\*UI\*\*:** `/workspaces/{workspaceId}/settings` → Danger Zone → "워크스페이스 삭제" 버튼

**\*\*Request\*\*:**

```
```http
DELETE /api/workspaces/{workspaceId}
...```

```

****Response**** (204 No Content)

****Frontend Action**:**

- 해당 워크스페이스의 모든 함수도 삭제 처리
- 현재 워크스페이스인 경우 `setCurrentWorkspaceId(null)`
- `/`로 이동

****Confirmation**:** 삭제 전 confirm 다이얼로그 필요

5 함수 생성

****UI**:**

- `/workspaces/{workspaceId}` → "+ 함수 만들기" 버튼
- `/workspaces/{workspaceId}/functions` → "새 함수" 버튼
- `/workspaces/{workspaceId}/functions/new` → "함수 만들기" 버튼

****Request**:**

```http

POST /api/workspaces/{workspaceId}/functions

Content-Type: application/json

{

```
"name": "user-authentication",
"description": "Handles user login and token generation",
"runtime": "Python 3.12",
"memory": 256,
"timeout": 30,
"httpMethods": ["POST"],
"environmentVariables": {
 "JWT_SECRET": "secret-key",
 "TOKEN_EXPIRY": "3600"
},
```

"code":

```
"ZGVmIGhhbmRsZXIoZXZlbnQsIGNvbnRleHQpOgogiCAgcmV0dXJuIHsnc3RhdHVzQ29kZSc6IDIwMCwgJ2
JvZHknOiAnSGVsbg8nfQ=="
```

}

...

**\*\*중요\*\*:** `code` 필드는 **Base64 인코딩된 Python 소스 코드**입니다.

**\*\*Response\*\* (201 Created):**

```json

```
{
  "id": "fn-xyz789",
  "workspaceId": "ws-1",
  "name": "user-authentication",
  "description": "Handles user login and token generation",
  "runtime": "Python 3.12",
  "memory": 256,
  "timeout": 30,
  "httpMethods": ["POST"],
  "environmentVariables": {
    "JWT_SECRET": "****"
  },
  "code":
```

"ZGVmIGhhbmRsZXIoZXZlbnQsIGNvbnRleHQpOgogiCAgcmV0dXJuIHsnc3RhdHVzQ29kZSc6IDIwMCwgJ2
JvZHknOiAnSGVsbg8nfQ==",

"invocationUrl": null,

"status": "active",

```
"lastModified": "2025-11-30T12:00:00Z",
"lastDeployed": null,
"invocations24h": 0,
"errors24h": 0,
"avgDuration": 0
}
```

```

#### \*\*주의\*\*:

- `invocationUrl`은 배포 완료 전까지 **null**
- `code`는 S3에 저장, 응답에는 Base64 문자열 포함

#### \*\*Validation\*\*:

- `name`: 필수
- `httpMethods`: 최소 1개 필요
- `code`: Base64 디코딩 가능 여부 검증

#### \*\*Frontend Action\*\*:

- `code` 필드: 전송 전 `btoa()` 또는 Base64 라이브러리로 인코딩
- 수신 후 `atob()`로 디코딩하여 에디터에 표시
- 생성 후 `/workspaces/{workspaceId}/functions/{functionId}`로 이동
- 워크스페이스의 `functionCount` 증가

---

### ### 6 함수 목록 조회

\*\*UI\*\*: `/workspaces/{workspaceId}/functions` - 함수 목록 테이블

#### \*\*Request\*\*:

```
```http
GET /api/workspaces/{workspaceId}/functions
```

```

\*\*Response\*\* (200 OK):

```
```json
[
  {
    "id": "fn-1",
    "workspaceId": "ws-1",
    "name": "user-authentication",
    "description": "Handles user login and token generation",
    "runtime": "Python 3.12",
    "memory": 256,
    "timeout": 30,
    "httpMethods": ["POST"],
    "environmentVariables": {
      "JWT_SECRET": "secret-key",
      "TOKEN_EXPIRY": "3600"
    },
    "code": "ZGVmIGhhbmRsZXIoZXZlbnQsIGNvbnRleHQpOgogICAgcmV0dXJuIHsnc3RhHVzQ29kZSc6IDlwMCwgJ2JvZHknOiAnSGVsbg8nfQ==",
    "invocationUrl": "https://api.example.com/invoke/fn-1",
    "status": "active"
  }
]
```

```

```
"lastModified": "2025-11-28T00:00:00Z",
"lastDeployed": "2025-11-28T00:00:00Z",
"invocations24h": 8420,
"errors24h": 15,
"avgDuration": 145
}
]
```

```

Frontend Features:

- 검색 기능 (클라이언트 사이드)
- 테이블 표시: Name, Runtime, Status, Invocations(24h), Last Modified
- `code` 필드는 목록에서는 표시 안 함 (상세 조회 시에만)

7 함수 상세 조회

UI: `/workspaces/{workspaceId}/functions/{functionId}` - Overview/Logs/Code 탭

Request:

```http

GET /api/workspaces/{workspaceId}/functions/{functionId}

```

Response (200 OK):

```json

```
{
 "id": "fn-1",
 "workspaceId": "ws-1",
 "name": "user-authentication",
 "description": "Handles user login and token generation",
 "runtime": "Python 3.12",
 "memory": 256,
 "timeout": 30,
 "httpMethods": ["POST"],
 "environmentVariables": {
 "JWT_SECRET": "secret-key",
 "TOKEN_EXPIRY": "3600"
 },
 "code": "ZGVmIGhhbmRsZXIoZXZlbnQsIGNvbnRleHQpOgogiCAgcmV0dXJuIHsnc3RhdHVzQ29kZSc6IDlwMCwgJ2JvZHknOiAnSGVsbg8nfQ==",
 "invocationUrl": "https://api.example.com/invoke/fn-1",
 "status": "active",
 "lastModified": "2025-11-28T00:00:00Z",
 "lastDeployed": "2025-11-28T00:00:00Z",
 "invocations24h": 8420,
 "errors24h": 15,
 "avgDuration": 145
}
```

```

Display Tabs:

- **Overview**: 엔드포인트 (`invocationUrl`), 설정, 메트릭
- **Logs**: 실행 로그 조회 (별도 API)
- **Code**: `code` 필드를 Base64 디코딩하여 Monaco Editor에 표시
- **Test**: ~~프론트엔드 시뮬레이션 (API를 필요로)~~

Frontend Action:

- `code` 필드: `atob()`로 디코딩 후 표시

8 함수 상태 토글 (활성/비활성)

UI: `/workspaces/{workspaceId}/functions` → 함수 드롭다운 메뉴 → "활성화" / "비활성화"

Request:

```http

PATCH /api/workspaces/{workspaceId}/functions/{functionId}

Content-Type: application/json

```
{
 "status": "disabled" // or "active"
}
```

```

Response (200 OK):

```json

```
{
 "id": "fn-1",
 "status": "disabled",
 "lastModified": "2025-11-30T12:00:00Z",
 ...
}
```

```

9 함수 삭제

UI:

- `/workspaces/{workspaceId}/functions` → 드롭다운 메뉴 → "삭제"
- `/workspaces/{workspaceId}/functions/{functionId}` → "함수 삭제" 버튼

Request:

```http

DELETE /api/workspaces/{workspaceId}/functions/{functionId}

```

Response (204 No Content)

Backend Action:

- DynamoDB에서 함수 레코드 삭제
- S3에서 코드 파일 삭제 (`s3://bucket/{workspaceId}/{functionId}.py`)

Frontend Action:

- 워크스페이스의 `functionCount` 감소
- 상세 페이지에서 삭제 시 `/workspaces/{workspaceId}/functions`로 이동

Confirmation: 삭제 전 confirmダイ얼로그 필요

10 함수 실행 로그 조회

UI: `/workspaces/{workspaceId}/functions/{functionId}` → "로그" 탭

Request:

```
```http
GET /api/workspaces/{workspaceId}/functions/{functionId}/logs?limit=100
````
```

****Query Parameters** (MVP 간소화):**

- `limit`: 조회할 로그 수 (기본 100, 최대 1000)

****Response** (200 OK):**

```
```json
```

```
{
 "logs": [
 {
 "id": "log-1",
 "functionId": "fn-1",
 "timestamp": "2025-11-30T12:05:00Z",
 "status": "success",
 "duration": 142,
 "statusCode": 200,
 "requestBody": {
 "username": "john@example.com"
 },
 "responseBody": {
 "token": "jwt.token.here"
 },
 "logs": ["Processing authentication request", "Token generated successfully"],
 "level": "info"
 }
],
 "total": 1
}
````
```

****MVP 페이지네이션**:**

- 최신 `limit` 개만 반환 (간단한 구현)
- 추후 확장 시 `nextToken` 방식으로 변경 가능

****Display Columns**:**

- Timestamp
- Status (Badge)
- Duration (ms)
- Status Code

API 연동이 불필요한 영역

✗ "테스트 & 실행" 탭

****UI**:** `/workspaces/{workspaceId}/functions/{functionId}` → "테스트 & 실행" 탭

****이유**:** 프론트엔드에서 `invokeFunction()` 메서드로 시뮬레이션

- 100~500ms 랜덤 지연
- 90% 성공률 시뮬레이션
- ExecutionLog 생성 (메모리)
- 백엔드 API 호출 없음

****실제 함수 실행****은 사용자가 `invocationUrl`로 직접 HTTP 요청을 보냅니다.

백엔드 구현 시 고려사항

1. 집계 데이터 계산

- `functionCount` : 워크스페이스별 활성 함수 개수
- `invocations24h` : 최근 24시간 호출 횟수
- `errors24h` : 최근 24시간 에러 횟수
- `errorRate` : $(\text{errors} / \text{invocations}) * 100$
- `avgDuration` : 평균 실행 시간 (ms)

2. 코드 저장 전략 (S3 + DynamoDB)

S3 버킷 구조:

```  
s3://functions-code-bucket/  
└─ {workspaceId}/  
 └─ {functionId}.py # Base64 디코딩된 Python 파일  
```

DynamoDB 테이블 (단일 테이블 설계):

```

Table: FaaSData

PK (Partition Key): String

SK (Sort Key): String

Attributes: JSON (모든 필드)

레코드 예시:

1. 워크스페이스:

PK: "WS#{workspaceId}"

SK: "METADATA"

Attributes: { name, description, createdAt, ... }

2. 함수:

PK: "WS#{workspaceId}"

SK: "FN#{functionId}"

Attributes: { name, runtime, memory, invocationUrl, codeS3Key, ... }

3. 로그:

PK: "FN#{functionId}"

SK: "LOG#{timestamp}#[logId]"

Attributes: { status, duration, statusCode, ... }

```

장점:

- 단일 테이블로 관계 표현 가능

- 워크스페이스별 함수 조회: PK = "WS#ws-1", SK begins_with "FN#"

- 함수별 로그 조회: PK = "FN#fn-1", SK begins_with "LOG#"

3. Base64 인코딩/디코딩 처리

프론트엔드 (JavaScript):

```javascript

// 전송 전

const encoded = btoa(pythonCode); // 인코딩

// 수신 후

const decoded = atob(response.code); // 디코딩

```

백엔드 (Python):

```
```python
import base64
수신 후
decoded_code = base64.b64decode(request_data['code']).decode('utf-8')
S3에 저장 (원본 Python 코드)
s3.put_object(Bucket='bucket', Key=f'{ws_id}/{fn_id}.py', Body=decoded_code)
응답 전
encoded_code = base64.b64encode(python_code.encode('utf-8')).decode('utf-8')
```

```

4. 에러 응답 포맷

```
```json
{
 "error": {
 "code": "VALIDATION_ERROR",
 "message": "Function name is required",
 "details": {
 "field": "name"
 }
 }
}
```

```

5. 환경 변수 처리

- `environmentVariables`는 민감 정보 포함 가능
- MVP에서는 평문 저장/조회 (추후 암호화 및 마스킹 고도화 필요)

URL 라우팅 매핑

```
```
/
 # 워크스페이스 목록 조회
/workspaces/{id} # 워크스페이스 대시보드
/workspaces/{id}/functions # 함수 목록 조회
/workspaces/{id}/functions/new # 함수 생성 폼
/workspaces/{id}/functions/{fnId} # 함수 상세 조회
 └─ Overview 탭 # 함수 정보 표시 (invocationUrl 포함)
 └─ Test 탭 # ❌ 프론트엔드 시뮬레이션
 └─ Logs 탭 # 로그 조회
 └─ Code 탭 # Base64 디코딩 후 표시
/workspaces/{id}/settings # 워크스페이스 수정/삭제
```

```

MVP 구현 체크리스트

필수 구현

- [] Base64 인코딩/디코딩 (code 필드)
- [] invocationUrl 필드 추가
- [] 단일 DynamoDB 테이블 설계
- [] S3 코드 저장 경로 설계
- [] 간단한 로그 조회 (limit만)

추후 확장

- [] nextToken 기반 페이지네이션
- [] 환경 변수 암호화 및 마스킹
- [] 메트릭 실시간 집계 (CloudWatch)
- [] 함수 배포 자동화 (invocationUrl 생성)