

supabase -

목차

>3/26일	2
>4/17일	2

>3/26일

>4/17일

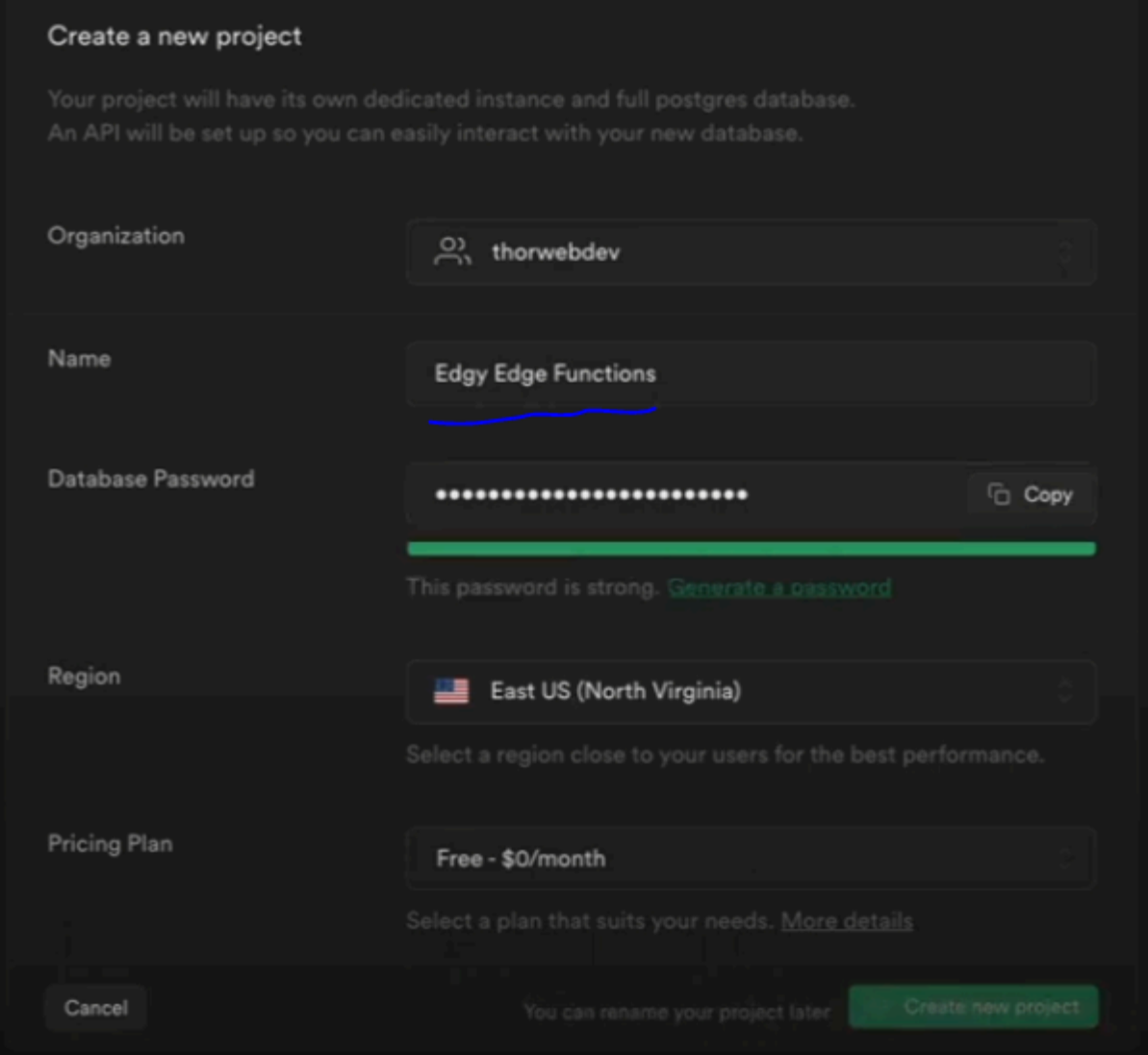
>Deno Edge Functions Setup

라이언트에서 하면 위험한 일	→ Edge Function 이 대신 함
DB에 원가 넣기/가공하기	가능
외부 API 호출 (비밀키 필요)	가능
Webhook 처리 (결제 완료 등)	가능
사용자 맞춤 응답	가능

딱 한 줄로 정리하면:

Supabase Edge Function은 앱이 못하거나 하면 위험한 일을 대신 해주는
똑똑한 서버 코드야! 💡

<https://www.youtube.com/watch?v=uZ6R9p3mgWM&list=PL5S4mPUpp40u1D3o1UW8Eq1IYKpUbK50b&index=1>



The image shows the 'Create a new project' form in the Supabase dashboard. The form is dark-themed and contains the following fields and options:

- Organization:** A dropdown menu showing 'thorwebdev'.
- Name:** A text input field containing 'Edgy Edge Functions'.
- Database Password:** A password input field with a strength indicator (a green bar) and a 'Copy' button.
- Region:** A dropdown menu showing 'East US (North Virginia)'.
- Pricing Plan:** A dropdown menu showing 'Free - \$0/month'.

At the bottom of the form, there is a 'Cancel' button, a note 'You can rename your project later', and a 'Create new project' button.

Edge Functions | Supabase

supabase.com/dashboard/project/egktwydismdtwqccip/functions/new

throwdev Free / Edgy Edge Function Connect Enable branching Feedback Chat

Edge Functions

MANAGE Functions Secrets

Edge Functions > Create new edge function

FILES + Add File index.ts

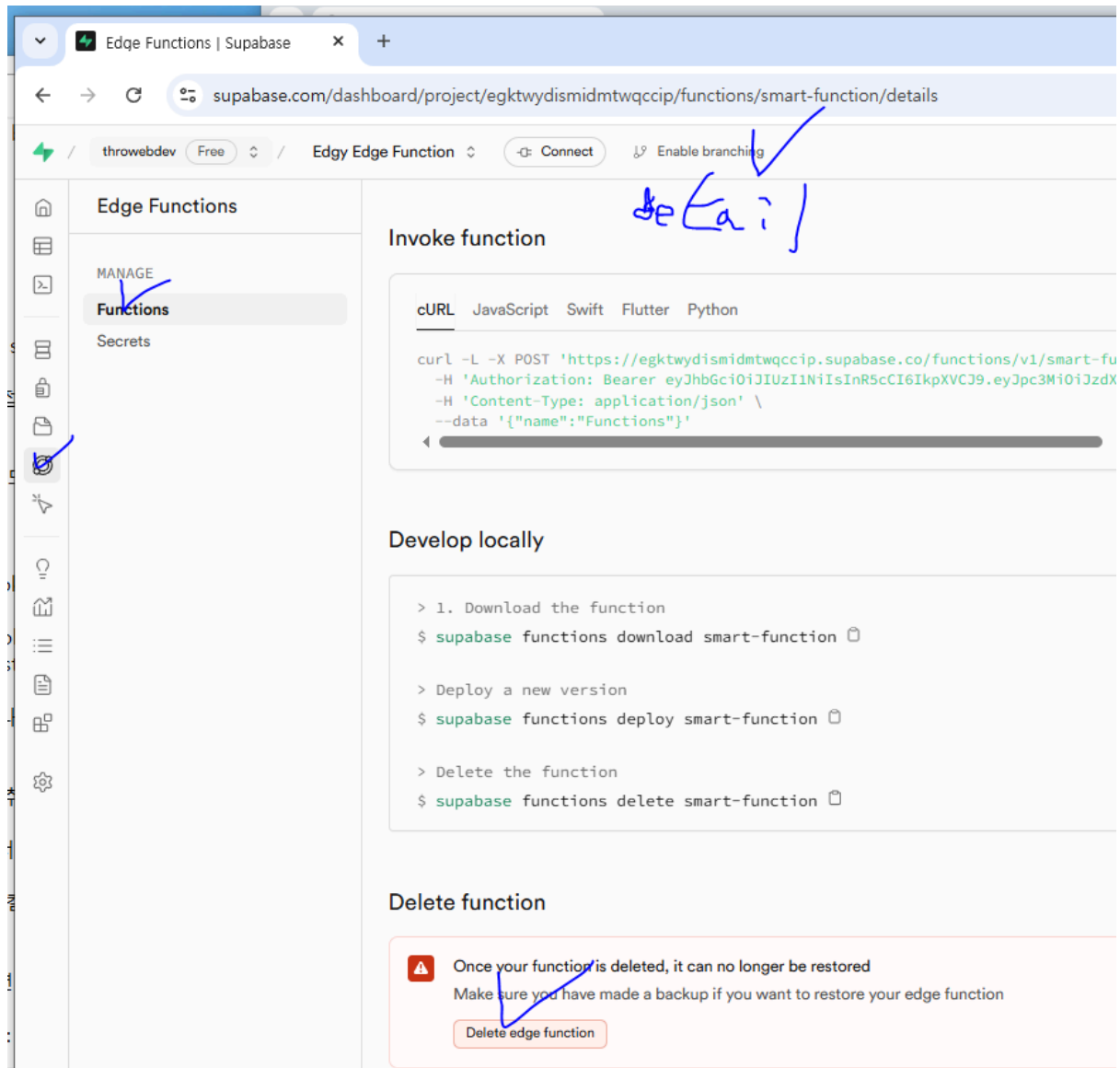
```
1 // Setup type definitions for built-in Supabase Runtime APIs
2 import "jsr:@supabase/functions-js/edge-runtime.d.ts";
3 interface reqPayload {
4   name: string;
5 }
6
7 console.info("server started");
8
9 Deno.serve(async (req: Request) => {
10   const { name } = reqPayload = await req.json();
11   const data = {
12     message: `Hello ${name}!`,
13   };
14
15   return new Response(
16     JSON.stringify(data),
17     { headers: { 'Content-Type': 'application/json', 'connection': 'keep-alive' } }
18   );
19 });
```

Search templates

- ✓ Simple Hello World
Basic function that returns a JSON response
- Supabase Database Access
Example using Supabase client to query your database
- Supabase Storage Upload
Upload files to Supabase Storage
- Node Built-in API Example
Example using Node.js built-in crypto and http modules
- Express Server
Example using Express.js for routing

Function name smart-function Deploy function

detail 탭에서 삭제 할 수 있다.



💬 Supabase Edge Function에서 사용하는 언어는?

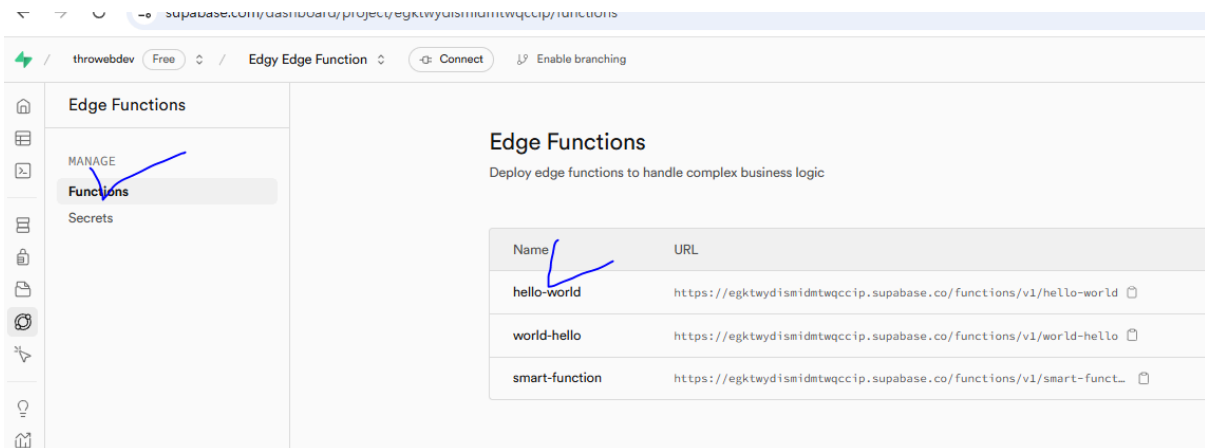
👉 TypeScript 또는 JavaScript (ESM 방식)

Deno 런타임에서 실행 가능한 방식으로 작성한 TypeScript/JavaScript만 가능해요.

🎯 어떤 스타일이여야 하나?

- ☒ **TypeScript** (기본적으로 가장 추천)
- ☒ **ESM** 스타일의 **JavaScript**
- ☒ **Node.js** 스타일 (`require`, `module.exports` 등) 은 안 됨

만든 함수 주소 접근하기



Supabase Edge Functions dashboard showing a list of deployed functions:

Name	URL
hello-world	https://egktwydisidmtwqccip.supabase.co/functions/v1/hello-world
world-hello	https://egktwydisidmtwqccip.supabase.co/functions/v1/world-hello
smart-function	https://egktwydisidmtwqccip.supabase.co/functions/v1/smart-funct...

```
// Setup type definitions for built-in Supabase Runtime APIs
```

```
import "jsr:@supabase/functions-js/edge-runtime.d.ts";
```

```
interface reqPayload {
```

```
    name: string;
  }

  console.info('server started');
```

```
Deno.serve(async (req: Request) => {

  const { name }: reqPayload = await req.json();

  const data = {

    message: `Hello ${name}!`,

  };

  return new Response(

    JSON.stringify(data),

    { headers: { 'Content-Type': 'application/json', 'Connection': 'keep-alive' } }

  );

});
```

썬더에서 확인하기

THUNDER CLIENT

New Request

Activity Collections Env

filter activity

POST egktywdismidt... just now

POST ftrtpjdzweflzdg... 2 days ago

GET localhost:3010... 2 days ago

GET curl -X POST "your... 6 days ago

POST https://egktywdismidtqwccip.supabase.co/functions/v1/world-h Send

Query Headers 4 Auth Body 1 Tests Pre Run

HTTP Headers

Accept */*

User-Agent Thunder Client (https://www.thundercli

Authorization Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6Ikp

Content-Type application/json

header value

Status: 200 OK Size: 30 Bytes Time: 2.04 s

Response Headers 15 Cookies Results Docs

```
1 {
2   "message": "Hello Functions!"
3 }
```

File Edit Selection View Go Run ...

reactwork

THUNDER CLIENT

New Request

Activity Collections Env

filter activity

POST egktywdismidt... just now

POST ftrtpjdzweflzdg... 2 days ago

GET localhost:3010... 2 days ago

GET curl -X POST "your... 6 days ago

POST https://egktywdismidtqwccip.supabase.co/functions/v1/hello-w Send

Query Headers 4 Auth Body 1 Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content

```
1 {
2   "name": "Functions"
3 }
```

Status: 200 OK Size: 30 Bytes Time: 2.09 s

Response Headers 15 Cookies Results I

```
1 {
2   "message": "Hello Functions!"
3 }
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

powershell - edgy-edge-fu

실행 서버 만들기

`npx local-web-server`

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Supabase Edge Function Test</title>
</head>
<body>
  <h1>Call Supabase Edge Function</h1>
  <input type="text" id="nameInput" placeholder="Enter your name" />
  <button onclick="callFunction()">Send</button>

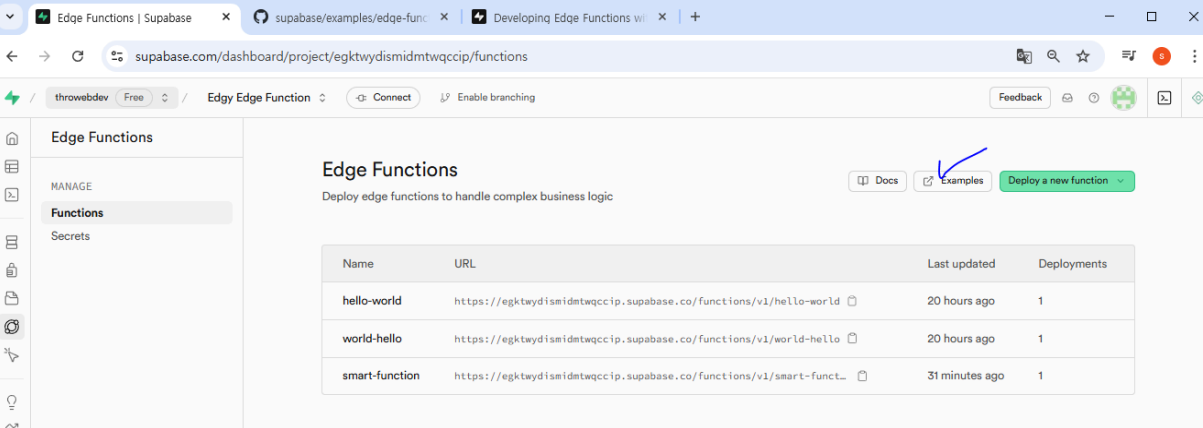
  <p id="responseArea"></p>

  <script>
    async function callFunction() {
      const name = document.getElementById("nameInput").value;
      const response = await fetch("/functions/v1/hello", {
        method: "POST",
        headers: {
          "Content-Type": "application/json"
        },
        body: JSON.stringify({ name })
      });

      const data = await response.json();
      document.getElementById("responseArea").innerText = data.message;
    }
  </script>
</body>
</html>
```

```
}  
</script>  
</body>  
</html>
```

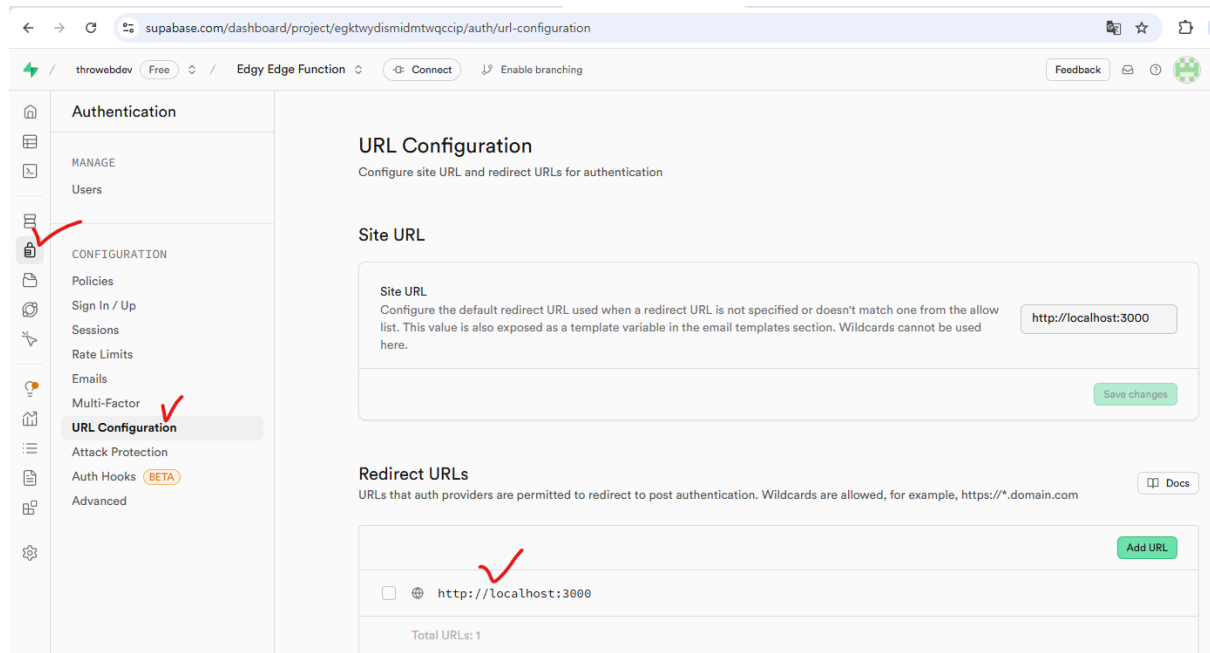
샘플을 얻고 싶으면.



The screenshot shows the Supabase Edge Functions dashboard. The left sidebar contains navigation links for 'MANAGE', 'Functions', and 'Secrets'. The main content area is titled 'Edge Functions' and includes a sub-header 'Deploy edge functions to handle complex business logic'. In the top right corner, there are buttons for 'Docs', 'Examples' (which has a blue checkmark drawn over it), and a green 'Deploy a new function' button. Below this, a table lists the deployed functions:

Name	URL	Last updated	Deployments
hello-world	https://egktwydismidmtwqccip.supabase.co/functions/v1/hello-world	20 hours ago	1
world-hello	https://egktwydismidmtwqccip.supabase.co/functions/v1/world-hello	20 hours ago	1
smart-function	https://egktwydismidmtwqccip.supabase.co/functions/v1/smart-funct...	31 minutes ago	1

이메일 인증시 리다이렉트 되는 페이지. 등록



edge function을 사용한 CRUD

CRUD 작업

데이터 조회

```
{
  "data": [
    {
      "id": "b8518664-bafc-4221-81cf-f861c026f107",
      "name": "s",
      "age": 3,
      "created_at": "2025-04-13T08:52:12.576449+00:00",
      "updated_at": "2025-04-13T08:52:12.576449+00:00"
    }
  ]
}
```

데이터 업데이트

데이터 삭제

```
create table users (
  id uuid primary key default uuid_generate_v4(),
  name varchar(100) not null,
  age integer not null,
  created_at timestamptz default now(),
  updated_at timestamptz default now()
);
```

```

import "jsr:@supabase/functions-js/edge-runtime.d.ts";
import { createClient } from 'jsr:@supabase/supabase-js@2';
// ✅ 환경 변수에서 Supabase 설정 가져오기
const supabaseUrl = Deno.env.get('SUPABASE_URL') ?? '';
const supabaseAnonKey = Deno.env.get('SUPABASE_ANON_KEY') ?? '';
const TABLE_NAME = 'users';
// 공통 CORS 헤더
const corsHeaders = {
  'Content-Type': 'application/json',
  'Access-Control-Allow-Origin': '*',
  'Access-Control-Allow-Methods': 'GET, POST, PUT, DELETE, OPTIONS',
  'Access-Control-Allow-Headers': 'Content-Type, Authorization'
};
Deno.serve(async (req) => {
  try {
    if (req.method === 'OPTIONS') {
      return new Response(null, {
        headers: corsHeaders,
        status: 204
      });
    }
    // ✅ 클라이언트에서 보낸 Authorization 헤더 받기
    const authHeader = req.headers.get('Authorization');
    // ✅ Supabase 클라이언트 생성 시 Authorization 헤더 설정
    const supabase = createClient(supabaseUrl, supabaseAnonKey, {
      global: {
        headers: {
          Authorization: authHeader
        }
      }
    });
    const { method } = req;
    if (method === 'GET') {
      const { data, error } = await supabase.from(TABLE_NAME).select('*');
      if (error) throw error;
      return new Response(JSON.stringify({
        data
      }), {
        headers: corsHeaders,
        status: 200
      });
    }
    if (method === 'POST') {
      const { name, age } = await req.json();
      const { data, error } = await supabase.from(TABLE_NAME).insert([
        {
          name,
          age
        }
      ]

```

```

    });
    if (error) throw error;
    return new Response(JSON.stringify({
      message: '데이터가 추가되었습니다.',
      data
    })), {
      headers: corsHeaders,
      status: 201
    });
  }
  if (method === 'PUT') {
    const { id, name, age } = await req.json();
    const { data, error } = await supabase.from(TABLE_NAME).update({
      name,
      age
    }).eq('id', id);
    if (error) throw error;
    return new Response(JSON.stringify({
      message: '데이터가 업데이트되었습니다.',
      data
    })), {
      headers: corsHeaders,
      status: 200
    });
  }
  if (method === 'DELETE') {
    const { id } = await req.json();
    const { data, error } = await supabase.from(TABLE_NAME).delete().eq('id', id);
    if (error) throw error;
    return new Response(JSON.stringify({
      message: '데이터가 삭제되었습니다.',
      data
    })), {
      headers: corsHeaders,
      status: 200
    });
  }
  return new Response(JSON.stringify({
    message: '지원하지 않는 메소드입니다.'
  })), {
    headers: corsHeaders,
    status: 405
  });
} catch (err) {
  return new Response(JSON.stringify({
    message: err?.message ?? String(err)
  })), {
    headers: corsHeaders,
    status: 500
  }
}

```

```
    });  
  }  
});
```


OPTIONS는 일반적인 HTTP 요청(GET, POST 등)과는 조금 다른 **사전 요청(Preflight Request)**입니다. CORS(교차 출처 리소스 공유)와 관련이 깊어요.

✅ **OPTIONS** 메서드란?

OPTIONS는 브라우저가 서버에게 **"이런 요청을 보내도 괜찮아?"**라고 미리 물어보는 용도로 사용되는 **사전 요청(preflight request)**입니다.

예: 이런 상황에서 쓰여요

클라이언트에서 다음과 같이 요청할 경우:

js

복사편집

```
fetch("https://api.example.com", {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
    "Authorization": "Bearer token"
  },
  body: JSON.stringify({ name: "fox" })
});
```

이때 브라우저는 서버에게 먼저 **OPTIONS** 요청을 보내요. 예를 들면:

http

복사편집

OPTIONS /api HTTP/1.1

Origin: https://your-frontend.com

Access-Control-Request-Method: POST

Access-Control-Request-Headers: Content-Type, Authorization

✓ 왜 쓰는 이유

보안 때문입니다.

브라우저는 다음 조건이 충족되지 않으면 실제 요청 전에 서버에게 허락을 먼저 구하는 방식으로 **OPTIONS** 요청을 보냅니다.

이런 조건이면 **Preflight** 발생:

- **Content-Type**이 **application/json**, **text/plain** 외의 값
- 커스텀 헤더 사용 (예: **Authorization**)
- 메서드가 **GET**, **POST**, **HEAD** 이외의 것 (예: **PUT**, **DELETE** 등)

✓ 그래서 **Edge Function**에서 **OPTIONS** 처리 꼭 해줘야 해요

ts

복사편집

```
if (req.method === 'OPTIONS') {
  return new Response(null, {
    status: 204, // 성공인데 내용 없음
    headers: corsHeaders,
  });
}
```

이걸 안 해주면 클라이언트 **fetch** 요청이 **CORS** 에러로 막히게 됩니다.

요약

구분	설명
OPTIONS	브라우저가 "진짜 요청 보내도 될지 물어보는 요청"
언제 발생?	CORS 정책 위반 가능성이 있을 때
왜 필요?	서버가 허용한 출처인지 확인하기 위해
처리 방법	204 No Content 로 CORS 헤더와 함께 응답

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>CRUD 작업</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      text-align: center;
      padding: 20px;
    }
    input, button {
      padding: 10px;
      font-size: 16px;
      margin-top: 10px;
    }
    button {
      background-color: #4CAF50;
      color: white;
      border: none;
      cursor: pointer;
    }
    button:hover {
      background-color: #45a049;
    }
    pre {
      text-align: left;
      white-space: pre-wrap;
      word-wrap: break-word;
    }
  </style>
</head>
<body>
  <h1>CRUD 작업</h1>

  <div>
    <input type="text" id="nameInput" placeholder="이름을 입력하세요">
    <input type="number" id="ageInput" placeholder="나이를 입력하세요">
```

```
<button onclick="createData()">추가</button>
</div>
```

```
<h2>데이터 조회</h2>
```

```
<button onclick="fetchData()">데이터 조회</button>
```

```
<pre id="dataResponse"></pre>
```

```
<h2>데이터 업데이트</h2>
```

```
<input type="text" id="updateId" placeholder="업데이트할 ID">
```

```
<input type="text" id="updateName" placeholder="새 이름">
```

```
<input type="number" id="updateAge" placeholder="새 나이">
```

```
<button onclick="updateData()">업데이트</button>
```

```
<h2>데이터 삭제</h2>
```

```
<input type="text" id="deleteId" placeholder="삭제할 ID">
```

```
<button onclick="deleteData()">삭제</button>
```

```
<script>
```

```
const apiUrl =
```

```
'https://egktwydismidmtwqccip.supabase.co/functions/v1/smart-function'; //
```

```
Supabase Edge Function URL
```

```
const accessToken =
```

```
'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSI6InJlZiI6ImVna3R3eWRpc21pZG10d3FjY2lwIiwicm9sZSI6ImFub24iLCJpYXQiOiJlE3NDQyNzAyMzgsImV4cCI6MjA1OTg0NjIzOH0.GMEGj0fxvFFglP_FoKIg3Fik7DqfAeJf5K80zJ4LSUA'; // 사용자가 인증한  
토큰을 입력하세요
```

```
async function createData() {
```

```
const name = document.getElementById('nameInput').value;
```

```
const age = document.getElementById('ageInput').value;
```

```
if (!name || !age) {
```

```
alert('이름과 나이를 모두 입력해주세요!');
```

```
return;
```

```
}
```

```
try {
```

```
const response = await fetch(apiUrl, {
```

```
method: 'POST',
```

```
headers: {
```

```
'Authorization': `Bearer ${accessToken}`, // 인증 헤더
```

```
'Content-Type': 'application/json',
```

```

    },
    body: JSON.stringify({ name, age }),
  });

  const data = await response.json();
  alert('데이터가 추가되었습니다.');
```

console.log(data);

```

} catch (error) {
  console.error('데이터 추가 오류:', error);
}
}

async function fetchData() {
  try {
    const response = await fetch(apiUrl, {
      method: 'GET',
      headers: {
        'Authorization': `Bearer ${accessToken}`, // 인증 헤더
      },
    });

    const data = await response.json();
    document.getElementById('dataResponse').textContent =
JSON.stringify(data, null, 2);
  } catch (error) {
    console.error('데이터 조회 오류:', error);
  }
}

async function updateData() {
  const id = document.getElementById('updateId').value;
  const name = document.getElementById('updateName').value;
  const age = document.getElementById('updateAge').value;

  if (!id || !name || !age) {
    alert('ID, 이름, 나이를 모두 입력해주세요!');
    return;
  }

  try {
    const response = await fetch(apiUrl, {
      method: 'PUT',

```

```

        headers: {
            'Authorization': `Bearer ${accessToken}`,
            'Content-Type': 'application/json',
        },
        body: JSON.stringify({ id, name, age }),
    });

    const data = await response.json();
    alert('데이터가 업데이트되었습니다. ');
    console.log(data);
} catch (error) {
    console.error('데이터 업데이트 오류:', error);
}
}

async function deleteData() {
    const id = document.getElementById('deleteId').value;

    if (!id) {
        alert('삭제할 ID를 입력해주세요!');
        return;
    }

    try {
        const response = await fetch(apiUrl, {
            method: 'DELETE',
            headers: {
                'Authorization': `Bearer ${accessToken}`,
                'Content-Type': 'application/json',
            },
            body: JSON.stringify({ id }),
        });

        const data = await response.json();
        alert('데이터가 삭제되었습니다. ');
        console.log(data);
    } catch (error) {
        console.error('데이터 삭제 오류:', error);
    }
}
</script>
</body>

```

</html>

로그인 관련 UID 정보 얻어오기

Supabase Edge Function에서 ****현재 로그인한 사용자의 UID(user ID)****를 가져오려면, 요청 헤더의 **Authorization** 토큰을 기반으로 Supabase 클라이언트를 생성한 뒤 `supabase.auth.getUser()`를 호출하면 됩니다.

아래는 전체 흐름과 코드 예시입니다. 📌

✅ 현재 로그인한 유저의 UID 가져오는 방법

🔧 1. Authorization 헤더로 Supabase 클라이언트 생성

ts

복사편집

```
import "jsr:@supabase/functions-js/edge-runtime.d.ts";
import { createClient } from 'jsr:@supabase/supabase-js@2';

const corsHeaders = {
  'Content-Type': 'application/json',
  'Access-Control-Allow-Origin': '*',
  'Access-Control-Allow-Methods': 'GET, POST, OPTIONS',
  'Access-Control-Allow-Headers': 'Content-Type, Authorization'
};

Deno.serve(async (req) => {
  // Preflight OPTIONS 요청 처리
  if (req.method === 'OPTIONS') {
    return new Response(null, {
      status: 204,
      headers: corsHeaders
    });
  }
});
```

```
}
```

```
// Supabase 클라이언트 생성
```

```
const supabase = createClient(  
  Deno.env.get('SUPABASE_URL') ?? '',  
  Deno.env.get('SUPABASE_ANON_KEY') ?? '',  
  {  
    global: {  
      headers: {  
        Authorization: req.headers.get('Authorization') ?? ''  
      }  
    }  
  }  
);
```

```
// 👉 현재 로그인한 유저 정보 가져오기
```

```
const {  
  data: { user },  
  error  
} = await supabase.auth.getUser();  
  
if (error || !user) {  
  return new Response(JSON.stringify({ message: 'Unauthorized' } ),  
    {  
      status: 401,  
      headers: corsHeaders  
    }  
  );  
}
```

```
// ✅ UID 가져오기
```

```
const uid = user.id;
```

```
    return new Response(JSON.stringify({ uid }), {
      headers: corsHeaders
    });
  });
});
```

✓ 결과

- 유저가 로그인된 상태로 요청했다면, `uid`가 JSON 응답으로 돌아옴:

json

복사편집

```
{
  "uid": "f3b7c4d4-xxxx-xxxx-xxxx-6e18431cb6a8"
}
```

⚠ 주의사항

1. 클라이언트(프론트엔드)에서 요청할 때 반드시 `Authorization` 헤더를 포함해야 합니다:

ts

복사편집

```
fetch('https://your-project.functions.supabase.co/my-function', {
  headers: {
    'Authorization': `Bearer ${access_token}`
  }
});
```

`access_token`은 `supabase.auth.getSession()` 등으로 얻을 수 있어요.

2. 익명 유저거나 토큰이 없는 경우 `user`는 `null`이 됩니다.

html GUI

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8" />
  <title>Supabase 로그인 + UID 확인</title>
</head>
<body>
  <h1>🔑 Supabase 로그인 및 UID 확인</h1>

  <div>
    <input id="email" type="email" placeholder="이메일" />
    <input id="password" type="password" placeholder="비밀번호" />
    <button id="loginBtn">로그인</button>
    <button id="logoutBtn">로그아웃</button>
  </div>

  <div style="margin-top:20px;">
    <button id="getIdBtn">✅ UID 가져오기</button>
  </div>

  <pre id="output" style="margin-top: 20px; background: #f5f5f5; padding: 1em;"></pre>

  <script type="module">
    import { createClient } from 'https://esm.sh/@supabase/supabase-js@2';

    const supabase = createClient(
      'https://egktwydismidmtwqccip.supabase.co',
      'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSI6InJlZiI6ImVna3R3eWRpc21pZG10d3FjY2lwIiwicm9sZSI6ImFub24iLCJpYXQiOiJlZ3NDQyNzAyMzgsImV4cCI6ImJlZiA1OTg0NjIzOH0.GMEGj0fxvfFglP_FoKIg3Fik7DqfAeJf5K80zJ4LSUA'
    );

    const emailInput = document.getElementById('email');
```

```

const passwordInput = document.getElementById('password');
const loginBtn = document.getElementById('loginBtn');
const logoutBtn = document.getElementById('logoutBtn');
const getUidBtn = document.getElementById('getUidBtn');
const output = document.getElementById('output');

loginBtn.addEventListener('click', async () => {
  const email = emailInput.value;
  const password = passwordInput.value;

  const { data, error } = await supabase.auth.signInWithPassword({
    email,
    password
  });

  if (error) {
    output.textContent = `❌ 로그인 실패: ${error.message}`;
  } else {
    output.textContent = `✅ 로그인 성공: ${data.user.email}`;
  }
});

logoutBtn.addEventListener('click', async () => {
  await supabase.auth.signOut();
  output.textContent = '🔒 로그아웃 되었습니다.';
});

getUidBtn.addEventListener('click', async () => {
  const { data: { session } } = await supabase.auth.getSession();

  if (!session) {
    output.textContent = '❗ 로그인 되어 있지 않습니다.';
    return;
  }

  const res = await
fetch('https://egktwydismidmtwqccip.supabase.co/functions/v1/sample-uuid', {
  method: 'GET',
  headers: {
    'Authorization': `Bearer ${session.access_token}`
  }
});
});

```

```
    const result = await res.json();
    output.textContent = JSON.stringify(result, null, 2);
  });
</script>
</body>
</html>
```

이메일 보내기

re_H9MRJg4X_LntauEbnrp3ngNTZwjYgoAR9

```
// File: send-email.ts
import { serve } from "https://deno.land/std@0.168.0/http/server.ts";
const RESEND_API_KEY = Deno.env.get('RESEND_API_KEY') ||
're_XDUraVrH_6zDj4cUcagtjLgNqz3hHk1dN'; // 환경변수에서 API 키 가져오기
serve(async (req) => {
  // CORS 설정
  const corsHeaders = {
    'Access-Control-Allow-Origin': '*',
    'Access-Control-Allow-Methods': 'POST, OPTIONS',
    'Access-Control-Allow-Headers': 'Content-Type, Authorization'
  };
  // Preflight 요청 처리
  if (req.method === 'OPTIONS') {
    return new Response(null, {
      headers: corsHeaders,
      status: 204
    });
  }
  // 인증 헤더 확인
  const authHeader = req.headers.get('Authorization');
  if (!authHeader) {
    return new Response(JSON.stringify({
      code: 401,
      message: 'Missing authorization header'
    }), {
      status: 401,
      headers: {
        ...corsHeaders,
        'Content-Type': 'application/json'
      }
    });
  }
  // 요청 처리
  try {
```



```

const { to, subject, html } = await req.json();
// 입력 검증
if (!to || !subject || !html) {
  return new Response(JSON.stringify({
    code: 400,
    message: 'All fields are required'
  })), {
    status: 400,
    headers: {
      ...corsHeaders,
      'Content-Type': 'application/json'
    }
  });
}
// Resend API로 이메일 전송
const resendResponse = await fetch('https://api.resend.com/emails', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
    'Authorization': `Bearer ${RESEND_API_KEY}`
  },
  body: JSON.stringify({
    from: 'no-reply@yourdomain.com',
    to,
    subject,
    html
  })
});
const data = await resendResponse.json();
if (!resendResponse.ok) {
  throw new Error(data.message || 'Failed to send email');
}
return new Response(JSON.stringify(data), {
  status: 200,
  headers: {
    ...corsHeaders,
    'Content-Type': 'application/json'
  }
});
} catch (error) {
  return new Response(JSON.stringify({
    code: 500,
    message: error.message || 'Internal server error'
  })), {
    status: 500,
    headers: {
      ...corsHeaders,
      'Content-Type': 'application/json'
    }
  }
}

```

```
});  
}  
});
```

html....

<https://www.resend.com/>

```
<!DOCTYPE html>  
<html lang="ko">  
<head>  
  <meta charset="UTF-8">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <title>이메일 전송 시스템</title>  
  <style>  
    body {  
      font-family: 'Arial', sans-serif;  
      max-width: 600px;  
      margin: 0 auto;  
      padding: 20px;  
      line-height: 1.6;  
    }  
    .form-group {  
      margin-bottom: 15px;  
    }  
    label {  
      display: block;  
      margin-bottom: 5px;  
      font-weight: bold;  
    }  
    input, textarea {  
      width: 100%;  
      padding: 8px;  
      border: 1px solid #ddd;  
      border-radius: 4px;
```

```
    box-sizing: border-box;
}
textarea {
    height: 150px;
    resize: vertical;
}
button {
    background-color: #4CAF50;
    color: white;
    padding: 10px 15px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    font-size: 16px;
}
button:hover {
    background-color: #45a049;
}
#response {
    margin-top: 20px;
    padding: 10px;
    border-radius: 4px;
}
.error {
    background-color: #ffebee;
    color: #c62828;
    border: 1px solid #ef9a9a;
}
.success {
    background-color: #e8f5e9;
    color: #2e7d32;
    border: 1px solid #a5d6a7;
}
.loading {
    background-color: #e3f2fd;
    color: #1565c0;
    border: 1px solid #90caf9;
}
</style>
</head>
<body>
    <h1>이메일 전송</h1>
```

```

<form id="emailForm">
  <div class="form-group">
    <label for="to">수신자 이메일:</label>
    <input type="email" id="to" required placeholder="example@domain.com">
  </div>

  <div class="form-group">
    <label for="subject">제목:</label>
    <input type="text" id="subject" required placeholder="이메일 제목">
  </div>

  <div class="form-group">
    <label for="message">내용:</label>
    <textarea id="message" required placeholder="이메일 내용을
입력하세요"></textarea>
  </div>

  <button type="submit" id="submitBtn">이메일 전송</button>
</form>

<div id="response"></div>

<script>
  document.getElementById('emailForm').addEventListener('submit', async (e)
=> {
    e.preventDefault();
    const responseEl = document.getElementById('response');
    const submitBtn = document.getElementById('submitBtn');

    responseEl.className = 'loading';
    responseEl.textContent = '처리 중...';
    submitBtn.disabled = true;

    try {
      const to = document.getElementById('to').value.trim();
      const subject = document.getElementById('subject').value.trim();
      const message = document.getElementById('message').value.trim();

      if (!to || !subject || !message) {
        throw new Error('모든 필드를 채워주세요');
      }
    }
  });

```

```

const res = await
fetch('https://egktwydismidmtwqccip.supabase.co/functions/v1/sample-email', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
    'Authorization': 'Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6ImVna3R3e
WRpc21pZG10d3FjY2lwIiwicm9sZSI6ImFub24iLCJpYXQiOi0jE3NDQyNzAyMzgsImV4cCI6MjA1OTg
0NjIzOH0.GMEGj0fxvfFglP_FoKIg3Fik7DqfAeJf5K80zJ4LSUA' // 필요한 경우 API 키
추가
  },
  body: JSON.stringify({
    to: to,
    subject: subject,
    html: message
  }),
});

const data = await res.json();

if (!res.ok) {
  throw new Error(data.message || '서버에서 오류가 발생했습니다');
}

responseEl.className = 'success';
responseEl.textContent = '이메일이 성공적으로 전송 되었습니다!';
document.getElementById('emailForm').reset();
} catch (error) {
  responseEl.className = 'error';
  responseEl.textContent = `오류: ${error.message}`;
} finally {
  submitBtn.disabled = false;
}
});
</script>
</body>
</html>

```

무료도메인

<https://www.freenom.com/en/index.html?lang=en>

<https://resend.com/>

도메인 등록 해야 한다.

<https://resend.com/domain>