Edge Functions | WeWeb Documentation

|  |  |
| --- | --- |
| URL source | <https://docs.weweb.io/workflows/actions/supabase/invoke-edge-function.html> |
| Date de scraping | 2025-04-08T15:57:02.564048+00:00 |

## Edge Functions ​

Edge Functions are mini-apps that run specific tasks when triggered by events in your app.  
For example:  
Send a welcome email when a customer signs up. Process Stripe payments and update database access levels during subscription purchases.  
They run on Supabase's servers, "on the edge," close to your users, ensuring speed, reliability, and security. Key benefits include:  
Secure sensitive data : Store API keys and credentials safely on the server, away from the frontend. Connect to external services : Centralize and simplify interactions with third-party tools like Stripe (payments), SendGrid (emails), OpenAI (AI), or Twilio (SMS) and more, while improving performance and reliability. Handle heavy tasks : Offload resource-intensive work like image processing, data transformation, PDF generation, or batch operations. Automate workflows : Trigger actions based on user activity (e.g., welcome emails), database changes (e.g., low stock alerts), or scheduled tasks (e.g., daily reports).

## Pre-requisites ​

To use Edge Functions in WeWeb, you'll need to create them on your local machine first, then upload them to Supabase. After that, you can invoke them in your WeWeb workflows using the Invoke Edge function action.  
Before starting, ensure you have:  
The latest version of the Supabase CLI to create and upload functions from your computer. Docker Desktop for a secure, isolated environment. A text editor like Visual Studio Code for writing and editing functions.

## Process overview ​

Assuming you have all of the above, the process to invoke a Supabase Edge Function in your WeWeb app will go as follows:  
Create a project folder on your computer where you run Supabase locally . Create the function and edit it. Deploy the function to your Supabase project . Invoke the Supabase function in your WeWeb app.  
Let's go through this step by step.

## Setup Your Workspace ​

Create a folder (e.g., my\_project ) and open it in Visual Studio Code: Open the terminal (command window) in VS Code:

### Images

<https://docs.weweb.io/assets/edge-open-repo.BsuAxZ94.gif>

<https://docs.weweb.io/assets/edge-folder-terminal.DQjF6IqQ.png>

## Set up Supabase locally ​

Type supabase init in the terminal. This creates a new Supabase workspace in your folder. Say yes when VS Code asks to set up some helpful tools (for Deno, which runs code).  
This creates a bunch of new files in the my\_project folder. Some files are for VS Code, and others are for the local Supabase setup.

### Images

<https://docs.weweb.io/assets/edge-supabase-init.B1OBqatm.png>

## Create a function ​

Next, to create an Edge function in your project, run the command supabase functions new your-function-name .  
This creates a new file where you'll write your function.  
  
The index.ts file comes with an Edge function boilerplate that you can use as a starting point to write your own functions.

### Images

<https://docs.weweb.io/assets/edge-functions-new.COiUHqzn.png>

## Test and deploy ​

Open Docker Desktop. Run supabase start to test your function locally. WARNING Ensure Docker Desktop is running. Otherwise, you'll encounter errors like: In the terminal, run supabase functions deploy your-function-name to deploy the function and then enter your Supabase project ID (found in project settings). Check Supabase dashboard to confirm upload, and that's it!

### Images

<https://docs.weweb.io/assets/edge-docker-error.Dx8dVJ_c.png>

<https://docs.weweb.io/assets/edge-deploy.Br11KWMV.png>

<https://docs.weweb.io/assets/edge-functions-in-supabase.Crw3Wh3H.png>

## Invoke the function ​

In WeWeb, you can call Supabase Edge functions by:  
Selecting the Invoke an Edge function action in a workflow. Entering the function name (e.g., your-name-function ). Configuring the request: Method : Choose the appropriate HTTP method (e.g., POST , GET ). Authorization headers : Include a valid token to authenticate the request. Body (for POST ): Include a JSON object (e.g., { "name": "Joyce" } ).  
When tested, the response will match the format defined in the Supabase Edge function, such as:

### Images

<https://docs.weweb.io/assets/edge-function-invoke.Bpz0gqEi.png>

<https://docs.weweb.io/assets/edge-function-response.DCEAZ4K-.png>

## CORS ​

CORS (Cross-Origin Resource Sharing) is a safety feature that checks if your app is allowed to request data from different places. You might see CORS errors when connecting to Supabase edge functions.  
To resolve CORS issues, simply add this object to your edge function:

### Code

Langage: unknown

// At the top of your edge function file (index.ts):  
const corsHeaders = {  
 'Access-Control-Allow-Origin': '\*',  
 'Access-Control-Allow-Headers': 'authorization, content-type'  
}  
//Inside your edge function's response handler (where you return data to the frontend), modify your response to include these CORS headers:  
  
// This is where you modify the response to include CORS headers  
return new Response(JSON.stringify(data), {  
 headers: { ...corsHeaders, 'Content-Type': 'application/json' }  
})