Form | Custom Upload Files Form and Send to Webhook

🧾 Overview

To enable secure and scalable file uploads via a Webflow form, we implemented a fully custom uploader that:

- Uses **pre-signed S3 URLs** generated from a **Supabase Edge Function**.
- Uploads files directly from the browser to S3.
- Submits form data (including the file's S3 key and public URL) to an **n8n webhook**.
- Lets the webhook return a **302 redirect**, which the browser correctly follows.

AWS S3

🔐 AWS IAM Configuration

Ensure the IAM user (or role) has the following policy to support pre-signed uploads/downloads:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:PutObject", "s3:GetObject"],
      "Resource": "arn:aws:s3:::pueblo-private/empleos-resume/*"
    }
  ]
}
```

🪪 CORS Configuration for S3 Bucket

Your S3 bucket (e.g. `pueblo-private`) must have a **CORS policy** that allows:

- `PUT` requests (for uploads)
- `GET` requests (if accessing via signed download URLs from the browser)
- Preflight `OPTIONS` requests

🛠️ How to Configure

1. Go to the **S3 bucket** in the AWS console (e.g. `pueblo-private`)
2. Click **Permissions**
3. Scroll to **Cross-origin resource sharing (CORS)**
4. Paste the following JSON:

```
[
  {
    "AllowedHeaders": ["*"],
    "AllowedMethods": ["GET", "PUT", "HEAD"],
    "AllowedOrigins": ["*"],
    "ExposeHeaders": ["ETag"],
    "MaxAgeSeconds": 3000
  }
]
```

⚙️ Supabase Edge Function Setup

📁 Folder & File Structure

Your Supabase project directory should look like this:

```
supabase/
├── functions/
│   ├── upload.ts        # POST - generates pre-signed upload URL
│   ├── download.ts      # GET - generates pre-signed download URL
├── import_map.json      # Required for Deno imports
└── .env                 # Stores AWS credentials securely
```

📤 Uploading Files with Supabase Edge Functions

🔧 POST Edge Function (`upload.ts`)

This function handles **pre-signed S3 upload URL** generation:

```
import { serve } from "std/server";
import { S3Client, PutObjectCommand } from "@aws-sdk/client-s3";
import { getSignedUrl } from "@aws-sdk/s3-request-presigner";

const REGION = "us-east-1";
const BUCKET_NAME = "pueblo-private";
const UPLOAD_FOLDER = "empleos-resume/";
const EXPIRATION_SECONDS = 600;

serve(async (req) => {
  const headers = {
    "Content-Type": "application/json",
    "Access-Control-Allow-Origin": "*",
    "Access-Control-Allow-Methods": "POST, OPTIONS",
    "Access-Control-Allow-Headers": "Content-Type"
  };

  if (req.method === "OPTIONS") {
    return new Response(null, { status: 204, headers });
  }

  try {
    const { filename, contentType } = await req.json();
    if (!filename || !contentType) {
      return new Response(JSON.stringify({ error: "Missing filename or content
Type" }), {
        status: 400,
        headers
      });
    }

    const key = `${UPLOAD_FOLDER}${crypto.randomUUID()}-${filename}`;

    const s3Client = new S3Client({
      region: REGION,
      credentials: {
        accessKeyId: Deno.env.get("AWS_ACCESS_KEY_ID") || "",
        secretAccessKey: Deno.env.get("AWS_SECRET_ACCESS_KEY") || ""
      }
    });

    const command = new PutObjectCommand({
      Bucket: BUCKET_NAME,
      Key: key,
      ContentType: contentType
    });

    const url = await getSignedUrl(s3Client, command, {
      expiresIn: EXPIRATION_SECONDS
    });

    return new Response(JSON.stringify({ url, key }), {
      status: 200,
      headers
    });
  } catch (error) {
    console.error("Upload pre-sign error:", error);
    return new Response(JSON.stringify({ error: "Failed to generate upload UR
L" }), {
      status: 500,
      headers
    });
  }
});
```

📥 Downloading Files Securely with Signed URLs

🔧 GET Edge Function (`download.ts`)

This function returns a **temporary signed URL** for downloading a file.

```
import { serve } from "std/server";
import { S3Client, GetObjectCommand } from "@aws-sdk/client-s3";
import { getSignedUrl } from "@aws-sdk/s3-request-presigner";

const REGION = "us-east-1";
const BUCKET_NAME = "pueblo-private";
const EXPIRATION_SECONDS = 600;

serve(async (req) => {
  const url = new URL(req.url);
  const key = url.searchParams.get("key");

  const headers = {
    "Content-Type": "application/json",
    "Access-Control-Allow-Origin": "*"
  };

  if (!key) {
    return new Response(JSON.stringify({ error: "Missing required parameter: k
ey" }), {
      status: 400,
      headers
    });
  }

  try {
    const s3Client = new S3Client({
      region: REGION,
      credentials: {
        accessKeyId: Deno.env.get("AWS_ACCESS_KEY_ID") || "",
        secretAccessKey: Deno.env.get("AWS_SECRET_ACCESS_KEY") || ""
      }
    });

    const command = new GetObjectCommand({
      Bucket: BUCKET_NAME,
      Key: key,
      ResponseContentDisposition: `attachment; filename="${key.split("/").pop
()}"`
    });

    const presignedUrl = await getSignedUrl(s3Client, command, {
      expiresIn: EXPIRATION_SECONDS
    });

    return new Response(JSON.stringify({ url: presignedUrl }), {
      status: 200,
      headers
    });
  } catch (error) {
    console.error("Error generating pre-signed URL:", error);
    return new Response(JSON.stringify({ error: "Failed to generate pre-signed
URL" }), {
      status: 500,
      headers
    });
  }
});
```

⚙️ Supabase Edge Runtime `import_map.json`

Both Edge Functions require this import map for dependencies:

```
{
  "imports": {
    "std/": "https://deno.land/std@0.177.0/",
    "std/server": "https://deno.land/std@0.177.0/http/server.ts",
    "@aws-sdk/client-s3": "npm:@aws-sdk/client-s3@3.440.0",
    "@aws-sdk/s3-request-presigner": "npm:@aws-sdk/s3-request-presigner
@3.440.0"
  }
}
```

Deploy Functions to Supabase

After creating your functions (`upload.ts`, `download.ts`) and `import_map.json` inside the `supabase/functions/` folder, deploy them using:

```
supabase functions deploy upload
supabase functions deploy download
```

These deploy your functions to your Supabase Cloud project and make them available at:

```
https://<project-ref>.functions.supabase.co/upload
https://<project-ref>.functions.supabase.co/download
```

Set Environment Variables (Secrets)

To access AWS credentials securely from your Edge Functions:

In Supabase Cloud:

1. Go to your **Supabase project**.
2. Navigate to **Functions → Settings → Environment Variables**.
3. Add the following variables:

```
AWS_ACCESS_KEY_ID=your-access-key
AWS_SECRET_ACCESS_KEY=your-secret-key
```

Disable JWT Requirement for Public Access

By default, Supabase requires a **JWT token** to invoke Edge Functions. Since your upload and download endpoints are meant to be **public**, you must disable that check.

To make the function public:

1. Go to **Functions → [Your Function]** in the Supabase Dashboard.
2. Click the **gear icon** ⚙️ for function settings.
3. Set **"Verify JWT"** to **"off"**.

Repeat this for both `upload` and `download`.

> 🔐 This allows unauthenticated users (like browsers uploading files) to access the function.

🌐 Webflow Form Integration (Short Version)

To handle file uploads in Webflow:

🧠 How It Works

- User clicks the styled div.
- JavaScript triggers the hidden input.
- After upload, the form is submitted via JS and redirected.

✅ Hidden File Input

Manually added inside the form:

```
<input type="file" id="upload-resume" name="resume" style="display: none;">
```

✅ Custom Upload Button (Div Block)

Created in Webflow as a styled `Div Block` with this attribute:

```
<div data-upload-trigger="upload-btn">Subir Resume</div>
```

🧠 Final JavaScript Uploader with Redirect Handling

This script does everything: upload → submit → redirect.

```
<script>
document.addEventListener("DOMContentLoaded", () => {
  const trigger = document.querySelector('[data-upload-trigger="upload-btn"]');
  const fileInput = document.getElementById("upload-resume");
  const form = document.querySelector("form");
  const submitButton = form.querySelector('input[type="submit"], button[type="submit"]');

  let fileToUpload = null;
  let uploadedKey = "";
  let uploadedUrl = "";
```

```
if (!trigger || !fileInput || !form || !submitButton) {
  console.error("Missing required elements");
  return;
}

form.setAttribute("action", "javascript:void(0)");
form.setAttribute("novalidate", "true");

trigger.addEventListener("click", (e) => {
  e.preventDefault();
  fileInput.click();
});

fileInput.addEventListener("change", () => {
  const file = fileInput.files[0];
  if (!file) return;

  if (file.size > 25 * 1024 * 1024) {
    alert("Archivo demasiado grande. Máximo 25MB.");
    fileInput.value = "";
    return;
  }

  fileToUpload = file;
  trigger.innerText = "Archivo seleccionado ✔️";
  console.log("📁 File ready:", file.name, file.type);
});

submitButton.addEventListener("click", async (e) => {
  e.preventDefault();

  if (!fileToUpload) {
    alert("Por favor selecciona un archivo.");
    return;
  }
```

```
    submitButton.disabled = true;
    submitButton.value = "Por favor espere...";

    try {
      console.log("📡 Obteniendo URL firmada...");
      const presignRes = await fetch("https://fmafwossstvdymuvwmaa.supabas
e.co/functions/v1/presign-url-empleos", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({
          filename: fileToUpload.name,
          contentType: fileToUpload.type,
        }),
      });

      const { url, key } = await presignRes.json();
      console.log("✅ URL recibida:", { url, key });

      console.log("⬆️ Subiendo archivo a S3...");
      const uploadRes = await fetch(url, {
        method: "PUT",
        headers: { "Content-Type": fileToUpload.type },
        body: fileToUpload,
      });

      if (!uploadRes.ok) throw new Error("Fallo al subir a S3");

      uploadedKey = key;
      uploadedUrl = url.split("?")[0];
      console.log("✅ Subido:", uploadedUrl);

      const formData = new FormData(form);
      formData.delete(fileInput.name); // remove raw file input
      formData.append("resume_key", uploadedKey);
      formData.append("resume_url", uploadedUrl);
      console.log("🚀 Enviando al webhook mediante redirección...");

      // 🔁 Build a real HTML form to submit and follow redirect
      const redirectForm = document.createElement("form");
      redirectForm.method = "POST";
      redirectForm.action = "https://n8n.melodev.com/webhook/pueblo-empleo
s";

      for (let [key, value] of formData.entries()) {
        const input = document.createElement("input");
        input.type = "hidden";
        input.name = key;
        input.value = value;
        redirectForm.appendChild(input);
      }

      document.body.appendChild(redirectForm);
      redirectForm.submit(); // 🔀 Browser will follow redirect

    } catch (err) {
      console.error("❌ Error:", err);
      alert("Hubo un error. Intenta de nuevo.");
      submitButton.disabled = false;
      submitButton.value = "Someter";
    }
   });
});
</script>
```