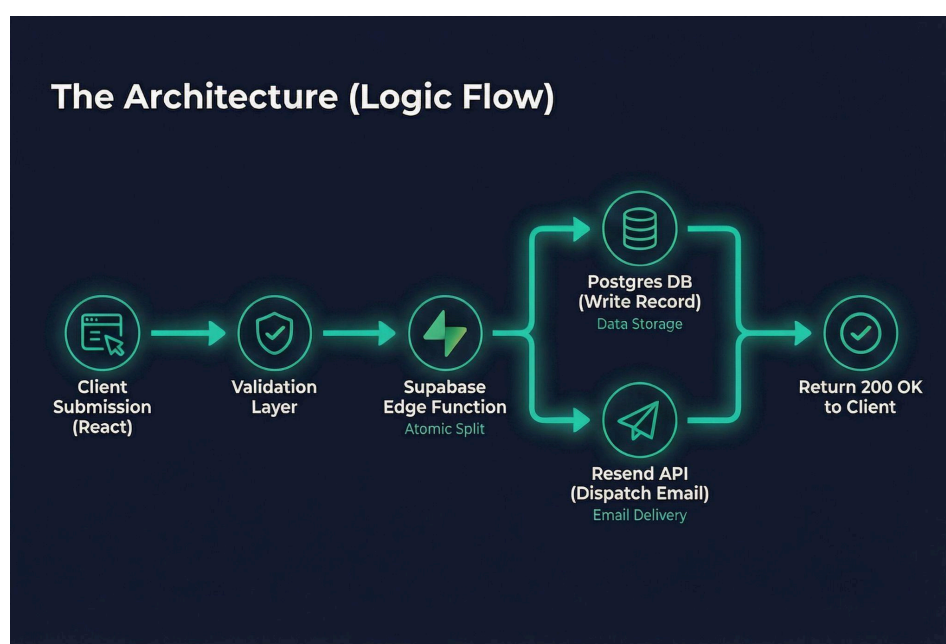


The "Hypergravity" Deployment Protocol

Migrating Bolt.new to the Hypergravity Stack

SUMMARY

- **Subject:** Zero-to-One Migration (Bolt.new → Production)
- **Version:** v1.0 (EU-Compliant)
- **Difficulty:** Medium (Requires CLI / Git)
- **Source:** Bolt.new (Prototype)
- **Target Stack:** Vite + Supabase + Netlify + Resend
- **Est. Time:** 120 Minutes



1. THE PROBLEM

Standard prototyping platforms (e.g., Bolt.new) excel at speed but fail at asset ownership and cost scaling. Moreover, they are hardly adaptable for custom integrations. Moving a project to production often incurs an immediate "Idle Tax" (\$30-\$200/month for servers/databases) regardless of user traffic.

Furthermore, default configurations often make it almost impossible to comply with GDPR data residency requirements, creating potential legal liability for EU-based ventures.

The Objective: Migrate the codebase to a serverless architecture that ensures **\$0 fixed costs, 100% IP ownership**, and **EU-Data Compliance**. We pay only for execution time, not uptime.

2. THE ARCHITECTURE (LOGIC FLOW)

The "Hypergravity" stack decouples the frontend from the backend logic to maximize cache hits and minimize database compute time.

Workflow: Client Submission (React) → Validation Layer → Supabase Edge Function

1. → **Postgres DB** (Write Record)
2. → **Resend API** (Dispatch Email)
3. → Return 200 OK to Client.

3. THE CODE / SCRIPT INSERT

The critical component of the "Hypergravity" stack is the Supabase Edge Function. This function replaces the traditional Node.js backend server, allowing us to handle logic efficiently without paying for idle server time.

It manages the workflow sequentially to ensure data integrity: first securing the record in the database, and then dispatching the email notification.

3.1 The Edge Function Source Code

File Path: supabase/functions/send-contact-email/index.ts

```
// PATH: supabase/functions/send-contact-email/index.ts

import { serve } from "https://deno.land/std@0.168.0/http/server.ts";
import { createClient } from "https://esm.sh/@supabase/supabase-js@2";

serve(async (req) => {
  try {
```

```
// 1. Setup & Parsing
const { name, email, message } = await req.json();

// 2. Initialize Auth Context
const supabase = createClient(
  Deno.env.get("SUPABASE_URL") ?? "",
  Deno.env.get("SUPABASE_SERVICE_ROLE_KEY") ?? ""
);

// 3. Insert into DB (Action A)
const { error: dbError } = await supabase
  .from("contact_submissions")
  .insert({ name, email, message });

if (dbError) {
  console.error("DB Error:", dbError);
  return new Response(JSON.stringify({ error: dbError }), { status: 500 });
}

// 4. Send Email via Resend (Action B)
const res = await fetch("https://api.resend.com/emails", {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
    "Authorization": `Bearer ${Deno.env.get("RESEND_API_KEY")}`,
  },
  body: JSON.stringify({
    from: "System <onboarding@yourdomain.com>",
    to: ["admin@yourdomain.com"],
    subject: `New Lead: ${name}`,
    html: `<p>${message}</p>`,
  }),
});

if (!res.ok) {
  const errorText = await res.text();
  console.error("Resend API Error:", errorText);
}

// 5. Success Response
return new Response("Action Complete", { status: 200 });

} catch (err) {
  return new Response(JSON.stringify({ error: err.message }), { status: 400 });
}
});
```

3.2 Environment Configuration

For the system to function, you must define the following secret keys. Without these, the code above will fail to send emails or save data.

A. Supabase Edge Function Secrets: Run these commands in your terminal or add them via the Supabase Dashboard (Settings > Edge Functions > Management).

- RESEND_API_KEY: Your API Key from Resend.com (starts with re_).
- SUPABASE_URL: Your project URL (found in Settings > API).
- SUPABASE_SERVICE_ROLE_KEY: Critical. Do not use the "anon" key here. You must use the service_role key (found in Settings > API) so the script has permission to write to the database bypassing Row Level Security (RLS) if necessary.

B. Frontend Variables (Netlify): Add these in Netlify under Site Settings > Environment Variables.

- VITE_SUPABASE_URL: Your project URL.
- VITE_SUPABASE_ANON_KEY: The public "anon" key. This allows your React frontend to initialize the connection.

Security Warning: Never expose the SUPABASE_SERVICE_ROLE_KEY in your Netlify frontend variables. It gives full administrative access to your database. It belongs only in the Supabase backend environment.

3.3 Terminal Deployment Commands

Execute these commands in your project root folder (where the supabase folder is located).

1. Login to Supabase CLI: Link your local environment to your remote project.

```
Bash
npx supabase login
npx supabase link --project-ref your-project-id
```

(Note: Replace your-project-id with the reference string found in your Supabase Dashboard URL).

2. Set Production Secrets: Push your API keys to the live Supabase environment securely.

```
Bash
npx supabase secrets set RESEND_API_KEY=re_12345...
npx supabase secrets set SUPABASE_SERVICE_ROLE_KEY=eyJhb...
```

3. Deploy the Edge Function: This pushes your index.ts code to the global edge network.

```
npx supabase functions deploy send-contact-email
```

4. Verification: Once deployed, the CLI will output a URL (e.g., `https://[project-ref].supabase.co/functions/v1/send-contact-email`). Copy this URL for use in your React frontend.

4. IMPLEMENTATION STEPS

Step 1: The Extraction (Local Environment)

Action: Export your project from Bolt.new (via .zip download or GitHub Sync). Sanitization:

1. Unzip and open the folder in VS Code.
2. Run `npm install` to finalize dependencies.
3. Clean Up: Move purely UI elements into `src/components` and ensure `src/index.css` contains the standard Tailwind directives (`@tailwind base;`, etc.).
4. Version Control: Initialize a git repository (`git init`) to establish asset ownership.

Step 2: The Database Core (Supabase)

Action: Create a new project at supabase.com. Schema Setup:

1. Go to the Table Editor and create a new table named `contact_submissions`.
2. Columns:
 - a. `id` (int8, primary key)
 - b. `created_at` (timestampz, default: `now()`)
 - c. `name` (text)
 - d. `email` (text)
 - e. `message` (text)
3. RLS Policies: Enable Row Level Security (RLS). Create a policy that allows "Enable insert for public/anon key" so your Edge Function can write to it.

Step 3: Frontend Integration (Connecting React)

This step connects your form to the Edge Function.

Action: Locate your Contact Form component (e.g., `Contact.tsx`) and replace the submit handler with this logic:

```
const handleSubmit = async (e: React.FormEvent) => {
  e.preventDefault();

  // 1. Prepare Data
  const formData = {
    name: (e.target as any).name.value,
    email: (e.target as any).email.value,
```

```
message: (e.target as any).message.value
};

// 2. Call Edge Function
// Replace with the URL you got from the CLI in Chapter 3
const FUNCTION_URL = import.meta.env.VITE_SUPABASE_FUNCTION_URL;

try {
  const response = await fetch(FUNCTION_URL, {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify(formData),
  });

  if (response.ok) {
    alert("Message sent successfully!");
    // Optional: Reset form here
  } else {
    alert("Failed to send message.");
  }
} catch (error) {
  console.error("Error:", error);
  alert("Network error.");
}
};
```

Step 4: Deployment Configuration (Netlify)

Config: Create a file named `netlify.toml` in your project root.

```
[build]
command = "npm run build"
publish = "dist"
```

Environment Variables: In your Netlify Dashboard (Site Settings > Environment Variables), add:

1. `VITE_SUPABASE_FUNCTION_URL`: The full URL from Chapter 3 (e.g., `https://xyz.supabase.co/functions/v1/send-contact-email`).
2. `VITE_SUPABASE_ANON_KEY`: Your public anon key.

Deploy: Commit your changes and push to GitHub (`git push`). Connect the repo to Netlify to trigger the auto-build.

Step 5: The "Germany Patch" (EU Compliance)

Critical for operating in the DACH region to minimize DSGVO liability.

1. **Data Residency (Supabase):** When creating the project, you must select Region: Central EU (Frankfurt). Do not select US East.
2. **Processors (Netlify):** Go to Team Settings > General > Data Processing Agreement (DPA). Digitally sign this to activate EU-U.S. Data Privacy Framework protections.

3. **Transmission (Resend)**: In your Resend Dashboard (Settings > Domains), verify the Region tag is Ireland (eu-west-1).
4. **Consent Logic (Frontend)**: Your form must include a mandatory checkbox: "Ich habe die Datenschutzerklärung zur Kenntnis genommen. Ich stimme zu, dass meine Angaben zur Kontaktaufnahme gespeichert werden."

Step 6: Validation (Definition of Done)

1. Email Check: Submit the live form. Did admin@yourdomain.com receive the lead?
2. Database Check: Does the row appear in the Supabase Table Editor?
3. Network Check: In Chrome DevTools, does the request return 200 OK?