BLUE FRONTIER LABS

SITES • PORTALS • SYSTEMS

THE BLUE FRONTIER LABS STACK

How I Build Websites

Every site I build is custom-built in code, not a template, and runs on a small set of cloud services, each doing one job well. None of them need a server of my own to maintain, so sites are fast, secure and cheap to run, and most small businesses stay on free or low-cost tiers.

The stack at a glance

Service	Its job	In one sentence
GitHub	Code storage and history	Holds every file of the site and a record of every change ever made.
Netlify	Hosting and deployment	Puts the site on the internet and republishes it automatically whenever the code changes.
Namecheap	Domain registration	Where your domain is registered, in your name, and pointed at your site.
Firebase (Google)	Logins and live data	Handles sign-in and stores data like bookings, clients and project records.
Supabase	Logins and relational data	An alternative to Firebase, built on a full SQL database, for data with lots of connections.
EmailJS	Email from forms	Sends a form submission straight to an inbox without a mail server.
Stripe	Payments	Takes card payments, deposits and invoices securely.
Shopify	Online stores	Runs product catalogs, carts, inventory, shipping and sales tax for businesses that sell goods.
UptimeRobot	Uptime monitoring	Checks the site around the clock and alerts me the moment it goes down.

GitHub where the code lives

What it is. GitHub is an online home for code. Each website gets its own repository (repo): a folder holding every page, image, style and script, plus a complete history of every change.

How it works. When I make a change, I save it as a commit, a snapshot with a short note like "Add favicon" or "Update pricing page." Commits are pushed to GitHub, which keeps them all in order.

Why I use it.

- **Nothing is ever lost.** Any earlier version of the site can be viewed or restored in minutes.
- **It drives publishing.** Netlify watches the repo and publishes each new push automatically.
- **Clear ownership.** The repo can be transferred to you or shared with another developer at any time. The site is never locked inside a proprietary builder.

Netlify putting the site on the internet

What it is. Netlify is the web host. It takes the files from GitHub and serves them to visitors from a global network of servers, so pages load quickly wherever people are.

How it works. Netlify is linked to the site's GitHub repo. Every time I push a change, Netlify rebuilds and republishes the site, usually in under a minute. Beyond plain hosting it provides:

- **Custom domain and HTTPS.** Your domain points to Netlify, which supplies the security certificate (the padlock) for free and renews it automatically.
- **Forms.** Simple contact and inquiry forms can be collected by Netlify directly.
- **Serverless functions.** Small pieces of back-end code that run only when called. I use these for anything that needs a secret key the public must never see, such as creating a Stripe payment.
- **Preview links.** Proposed changes get their own temporary address for review before going live.

Why I use it. No server to patch or monitor, automatic publishing, free HTTPS, and one-click rollback to any earlier version if something goes wrong.

Namecheap your domain name

What it is. Namecheap is the domain registrar, the company where a web address like yourbusiness.com is bought and renewed each year.

How it works. I register your domain at Namecheap in your name, or connect the one you already own, and point it at Netlify. Any email records for your business inbox stay in place.

Why I use it. Fair renewal prices, free privacy protection that keeps your personal contact details out of public domain records, and an account that belongs to you, not to me.

Firebase logins and live data

What it is. Firebase is Google's back-end platform. The two parts I use most are Authentication (secure sign-in) and Firestore (a database).

How it works. The website talks to Firebase directly from the visitor's browser. Authentication checks who someone is with email and password or a Google account. Firestore stores data as documents grouped in collections, such as a "clients" collection with one document per client. Security rules decide who can read or change what, so the public site can show availability while only a signed-in

owner can edit records.

What I build with it.

- Admin portals where an owner signs in to manage clients, proposals, projects, time and invoices
- Booking and inquiry records that update live on screen
- Customer accounts and member-only pages

Backups. Sites that store business records run on Firebase's paid Blaze plan, with scheduled daily and weekly backups of the database kept for up to 14 weeks. Any of them can be restored if data is lost or changed by mistake.

Why I use it. It is quick to set up, updates in real time, and scales automatically. Google handles the security and uptime, and a small site's usage costs very little.

Supabase the SQL alternative

What it is. Supabase does the same job as Firebase (sign-in, database, file storage) but is built on PostgreSQL, a traditional relational database that stores data in tables with rows and columns, like a set of linked spreadsheets.

How it works. As with Firebase, the site connects to Supabase from the browser, and row-level security policies control who can see each record. Because the data sits in related tables, it can answer questions like "every rental booked by this customer in July, with its payment status" in a single query. Supabase also stores uploaded files such as photos and PDFs.

Why I use it. A site uses one or the other, not both.

Choose Firebase when…	Choose Supabase when…
Data is simple and mostly independent records	Records are heavily linked (customers, orders, inventory, payments)
Live updates and speed of setup matter most	Reporting, filtering and exports matter
The project is small to mid-size	The data may need to move to another system later; standard SQL is portable

EmailJS email straight from the website

What it is. EmailJS sends emails from a website without a mail server. It connects to an existing email account, such as the business's Gmail or Outlook.

How it works. I design an email template in EmailJS, for example "New inquiry from {{name}}: {{message}}." When a visitor submits a form, the site hands the form fields to EmailJS, which fills in the template and sends it through the connected account. A second template can send the customer an automatic "Thanks, we got your request" confirmation.

Why I use it. Inquiries arrive in the owner's normal inbox within seconds, formatted the way they want. There is no mail server to configure, and emails come from the business's own address.

Stripe taking payments

What it is. Stripe is a payment processor. It accepts credit and debit cards, Apple Pay and Google Pay, and deposits the money into the business's bank account.

How it works. When a customer clicks "Book now" or "Pay deposit," a Netlify function holding Stripe's secret key asks Stripe for a checkout session. The customer pays on Stripe's own secure page, so card numbers never touch the website. Stripe then confirms the payment back to the site, which marks the booking paid.

Why I use it. Stripe carries the security and compliance burden of handling cards. There is no monthly fee, only a small charge per transaction, and it handles deposits, full payments, invoices, refunds and subscriptions from one dashboard.

Shopify online stores

What it is. Shopify is an e-commerce platform built for selling products. It runs the catalog, cart and checkout, and handles inventory, shipping labels and sales tax.

How it works. I set it up one of two ways, depending on the business:

- **A store inside your custom site.** I build the site as usual and connect Shopify's Buy Button or storefront, so products, cart and checkout run on Shopify while the site keeps its custom design and stays on Netlify.
- **A full Shopify store.** For larger catalogs, the whole store runs on Shopify with a theme I customize, and Shopify hosts it instead of Netlify.

Why I use it. Shopify takes care of the hard parts of retail: orders, stock counts, shipping rates, taxes and order emails, all manageable from the Shopify phone app. The store account is in your name, and product and order data can be exported at any time. Shopify charges a monthly plan plus card fees, billed to the business directly.

Stripe or Shopify?

Choose Stripe when…	Choose Shopify when…
You take bookings, deposits, invoices or service payments	You sell physical products
You have a handful of items or fixed prices	You have many products with sizes, colors or other options
You don't want a monthly fee	You need inventory tracking, shipping labels and sales tax

UptimeRobot watching the site

What it is. UptimeRobot is a monitoring service that visits your site every few minutes, around the clock.

How it works. If the site stops responding or its security certificate is about to expire, UptimeRobot alerts me by text and email right away, so I can fix it, usually before you or your customers notice.

Why I use it. Problems get caught in minutes instead of whenever someone happens to notice, and every site I host is watched the same way.

Why I build this way

Low running costs

Most services run on free or near-free tiers. Stripe charges only per payment, and Shopify's monthly plan applies only to sites with a store.

Fast

Pages are pre-built files served from a global network, not generated on a slow shared server.

Secure

No server to hack or forget to update. Card payments are handled by Stripe or Shopify, and secret keys stay in Netlify functions.

Reliable

Every piece is run by a company whose whole business is keeping it online, and any bad change can be rolled back in one click.

Owned by you

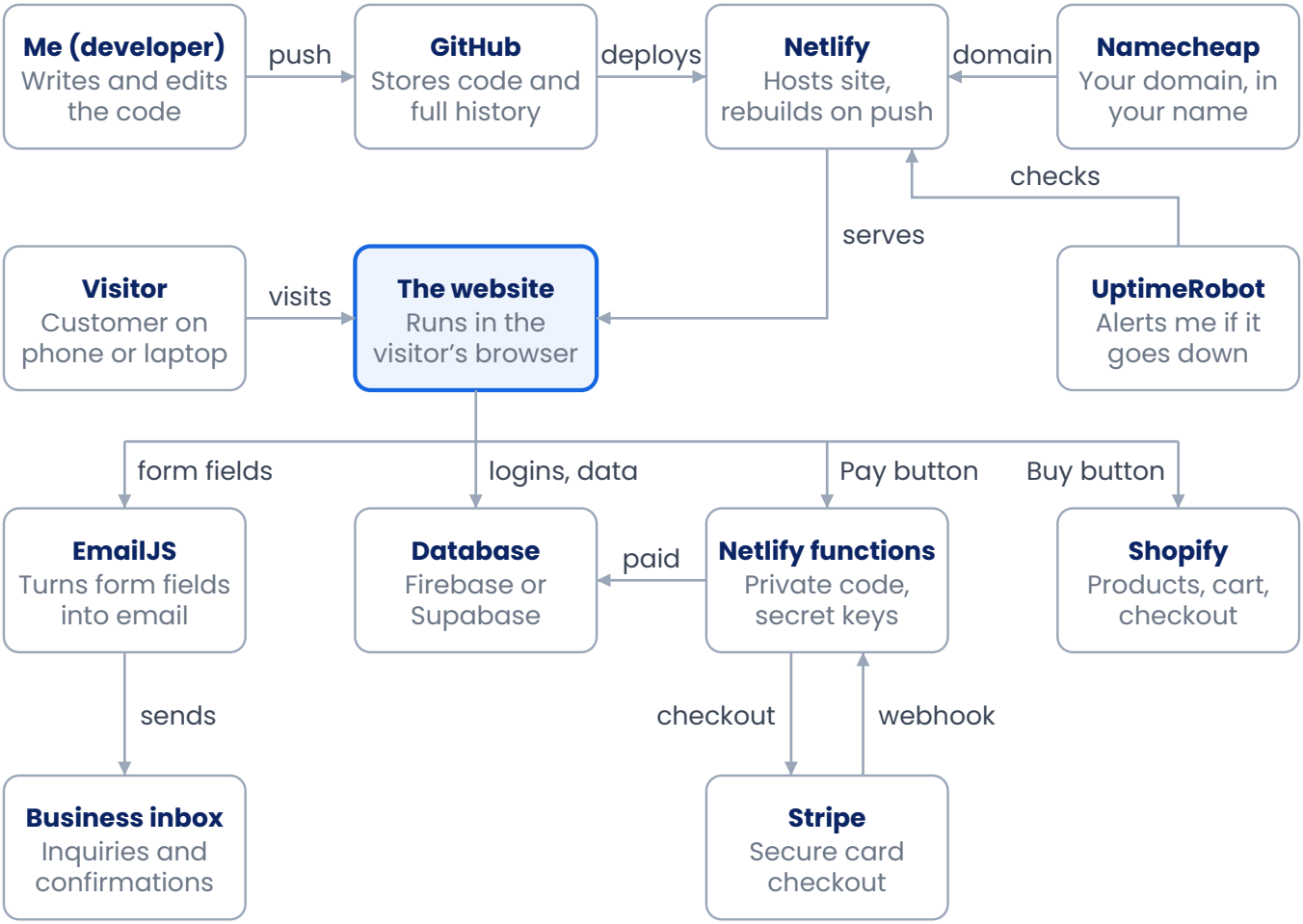
The code, domain, database, and Stripe or Shopify account can all sit in your name. Nothing is locked into me.

Grows with the business

A brochure site can later gain online booking, a store, a customer portal or payments by adding a service, without a rebuild.

How they work together

GitHub and Netlify get the site online. Once a visitor has it open, their browser talks to the other services directly, with Netlify functions guarding anything that needs a secret key.



Publishing a change

1. I edit the code and push it to GitHub.
2. Netlify rebuilds and publishes the site within about a minute.
3. Visitors see the new version on their next page load.

A customer books and pays

1. The customer fills out a booking form on the site.
2. The booking is saved to the database, and EmailJS emails the owner.
3. The customer pays the deposit on Stripe's checkout.
4. Stripe confirms the payment, and the booking is marked paid.
5. The owner sees it in their admin portal; the customer gets a confirmation.

A customer buys a product

1. The customer adds a product to the cart.
2. They pay on Shopify's secure checkout.
3. Shopify updates inventory and handles tax and shipping.
4. The owner prints a label; the customer gets a confirmation.