Complete Portfolio Website Runbook

Astro + Cloudflare Stack

Finalized Version 2.0 - With Comprehensive Troubleshooting

January 2026

Locked Tech Stack:
• Astro 5.x (Static Site Generator with Content Collections)
• Cloudflare Pages (Hosting with Git Integration)
• Cloudflare Access (Preview Deployment Gating)
• Cloudflare Pages Functions (Serverless Contact Form)
• Cloudflare Turnstile (Bot Protection)
• Cloudflare Registrar (Domain at Cost)
• GitHub (Version Control + PR Previews)
• Pagefind (Static Search)
• Resend (Outbound Email API)

Total Monthly Cost: $0 (excluding domain ~$10.46/year for .com)
Target Timeline: 3-6 months for full implementation

Table of Contents

Part 1: Development Environment Setup

Install Node.js (LTS Version)

Node.js powers Astro's build process. **Important:** Cloudflare Pages Build Image v3 defaults to **Node.js 22.16.0**. Install Node.js 22.x locally to match the build environment.

Windows Installation:

- Visit https://nodejs.org/en/download
- Download Node.js 22.x LTS (.msi installer)
- Run installer with "Add to PATH" checked
- Verify: Open PowerShell and run `node -v`

macOS Installation:

```
brew install node@22
```

Linux (Ubuntu/Debian):

```
curl -fsSL https://deb.nodesource.com/setup_22.x | sudo -E bash -
sudo apt-get install -y nodejs
```

Install and Configure Git

Git tracks code changes and connects to GitHub for automatic deployments.

Windows: Download from https://git-scm.com/download/win

macOS: Pre-installed. Run `git --version` to verify.

Linux: `sudo apt-get install git-all`

Configure your identity (run in any terminal - VS Code, PowerShell, etc.):

```
git config --global user.name "Your Name"
git config --global user.email "your.email@example.com"
git config --global init.defaultBranch main
```

Note: The --global flag applies to your user profile; the folder you run it from doesn't matter. The init.defaultBranch setting affects NEW repos created after setting it.

Install VS Code with Astro Extension

- Download from https://code.visualstudio.com
- Install the **Astro** extension by astro-build (official)
- Recommended: Prettier, ESLint, Material Icon Theme

Pin Node Version (After Creating Project)

> **IMPORTANT:** You cannot create .node-version until the project exists. Create the Astro project first (Part 3), then place .node-version in the project root next to package.json.

File location example: `C:\dev\portfolio\.node-version`

File contents:

```
22
```

Commit it so Cloudflare can use it:

```
git add .node-version
git commit -m "Pin Node version"
git push
```

Part 2: Account Creation and Two-Factor Authentication

Create GitHub Account with 2FA

- [ ] Sign up at https://github.com
- [ ] Verify email address
- [ ] Enable 2FA: Settings > Password and authentication > Enable two-factor authentication
- [ ] Choose TOTP app (Authy, 1Password, Google Authenticator)
- [ ] **Download and securely store recovery codes**

Create Cloudflare Account with 2FA

- [ ] Sign up at https://dash.cloudflare.com
- [ ] Verify email
- [ ] Enable 2FA: Profile icon > My Profile > Authentication

Create Resend Account

- [ ] Sign up at https://resend.com/signup
- [ ] Verify email address
- [ ] Domain verification comes later in Part 9

Part 3: Astro Project Scaffolding

Create New Astro Project

Open terminal (VS Code integrated terminal recommended) and run:

```
cd C:\dev
npm create astro@latest portfolio
cd portfolio
npm install
npm run dev
```

Note: When npm run dev shows "watching for file changes...", that's normal - the server is running. Open http://localhost:4321/ in your browser. Stop the server with Ctrl+C.

Wizard prompts (recommended choices):

- Template: Empty (for full control) or Blog (for head start)
- TypeScript: Strict (recommended)
- Install dependencies: Yes
- Initialize git: Yes

Install Required Packages

IMPORTANT: Use `astro-pagefind` (community package), NOT @astrojs/pagefind (does not exist).

```
npm install @astrojs/mdx @astrojs/sitemap
npm install astro-pagefind pagefind
```

Configure astro.config.mjs

Location: `C:\dev\portfolio\astro.config.mjs` (same level as package.json)

```
import { defineConfig } from 'astro/config';
import mdx from '@astrojs/mdx';
import sitemap from '@astrojs/sitemap';
import pagefind from 'astro-pagefind';

export default defineConfig({
  site: 'https://yourdomain.com',  // Update after domain purchase
  output: 'static',
  integrations: [
    mdx(),
    sitemap(),
    pagefind(),  // Must be LAST integration
  ],
  markdown: {
    shikiConfig: {
      theme: 'github-dark',
      wrap: true,
    },
  },
});
```

Create Content Directories

Create these directories (even if empty initially) to prevent build warnings:

```
mkdir src\content\blog
mkdir src\content\portfolio
mkdir src\content\studies
mkdir src\content\certifications
```

Optional: Add .gitkeep files to track empty directories:

```
ni src\content\blog\.gitkeep -ItemType File
ni src\content\portfolio\.gitkeep -ItemType File
ni src\content\studies\.gitkeep -ItemType File
ni src\content\certifications\.gitkeep -ItemType File
```

Configure Content Collections (Astro 5.x)

> **CRITICAL:** For Astro 5.x, create `src/content.config.ts` (NOT src/content/config.ts). It's a sibling of src/pages/, not inside src/content/.

File structure:

```
src/
  pages/
  content/
  content.config.ts   <-- HERE (sibling of pages)
```

Create `src/content.config.ts`:

```
import { defineCollection } from 'astro:content';
import { glob } from 'astro/loaders';
import { z } from 'astro/zod';

const blog = defineCollection({
  loader: glob({ pattern: '**/*.{md,mdx}', base: './src/content/blog' }),
  schema: z.object({
    title: z.string(),
    description: z.string().optional(),
    pubDate: z.coerce.date(),
    updatedDate: z.coerce.date().optional(),
    tags: z.array(z.string()).optional(),
    draft: z.boolean().default(false),
    disclaimer: z.boolean().default(false),
  }),
});

const portfolio = defineCollection({
  loader: glob({ pattern: '**/*.{md,mdx}', base: './src/content/portfolio' }),
  schema: z.object({
    title: z.string(),
    description: z.string(),
    technologies: z.array(z.string()),
    date: z.coerce.date(),
    github: z.string().url().optional(),
    featured: z.boolean().default(false),
    archived: z.boolean().default(false),
  }),
});
```

```
const studies = defineCollection({
  loader: glob({ pattern: '**/*.{md,mdx}', base: './src/content/studies' }),
  schema: z.object({
    title: z.string(),
    topic: z.string(),
    date: z.coerce.date(),
    tags: z.array(z.string()).optional(),
  }),
});

const certifications = defineCollection({
  loader: glob({ pattern: '**/*.{md,mdx}', base: './src/content/certifications' }),
  schema: z.object({
    title: z.string(),
    vendor: z.string().optional(),
    date: z.coerce.date(),
    proof: z.string().url().optional(),
    demo: z.string().optional(),
  }),
});

export const collections = { blog, portfolio, studies, certifications };
```

Part 4: GitHub Repository Setup

Create Repository and Push

- [ ] Create new repo on GitHub (portfolio or your-name)
- [ ] Visibility: Public (recommended) or Private
- [ ] Do NOT initialize with README

```
git remote add origin https://github.com/YOUR-USERNAME/portfolio.git
git branch -M main
git add .
git commit -m "Initial commit: Astro portfolio setup"
git push -u origin main
```

Set Up Dev/Main Branch Model

```
git checkout -b dev
git push -u origin dev
```

Branch Workflow:

- `main` = Production deployments (your live site)
- `dev` = Preview deployments (testing before live)
- Feature branches = Created from dev for specific changes

Note on Private Repos: Making your GitHub repo private is safe - Cloudflare Pages can still deploy as long as the GitHub integration retains access. Tradeoff: recruiters can't view source code, but your deployed site remains public. After switching to private, trigger a small commit to confirm Pages still builds.

Part 5: Cloudflare Pages Deployment

CRITICAL: You must create a **Pages** project, NOT a Workers project. If you see "deploy command required" or "npx wrangler deploy" in build logs, you're in the wrong flow.

Connect GitHub to Cloudflare Pages

[ ] Log into Cloudflare Dashboard (dash.cloudflare.com)

[ ] Navigate to **Workers & Pages > Create > Pages > Connect to Git**

[ ] Click + Add account and select your GitHub account

[ ] Click Install & Authorize

[ ] Select your portfolio repository

Build Settings:

Setting	Value
Project name	portfolio (becomes portfolio-xxx.pages.dev)
Production branch	main
Framework preset	Astro
Build command	npm run build
Build output directory	dist

[ ] Click Save and Deploy

Troubleshooting: "Missing entry-point" Error

If deployment fails with "Missing entry-point to Worker script" and logs show "npx wrangler deploy", you accidentally created a Workers Builds project instead of Pages.

Fix: Delete the project and create a new one using Pages > Connect to Git flow.

Part 6: Custom Domain and DNS Configuration

Purchase Domain via Cloudflare Registrar

Cloudflare sells domains at **wholesale cost** with no markup. .com domains are ~$10.46/year.

[ ] Navigate to https://domains.cloudflare.com/

[ ] Search for your desired domain name

[ ] Select TLD (.com recommended)

[ ] Complete checkout

[ ] **Enable DNSSEC** (one-click enable, requires manual action)

Configure Apex Domain for Pages

[ ] Workers & Pages > Your project > Custom domains tab

[ ] Click Set up a custom domain

[ ] Enter apex domain: example.com (no www)

Add www Subdomain with Redirect

[ ] Add www.example.com as another custom domain

[ ] Create redirect: Rules > Redirect Rules > Create rule

- When: Hostname equals www.example.com
- Then: Redirect to https://example.com${http.request.uri.path}
- Status: 301 (Permanent)

Part 7: Cloudflare Access for Preview Gating

Enable Access on Pages Project

[ ] Workers & Pages > Your project > Settings > General

[ ] Under "Access Policy", click Enable access policy

IMPORTANT: This only protects preview deployments. Production remains PUBLIC.

Configure One-Time PIN (OTP) Authentication

UI Note: The location has moved. Look under: Integrations > Identity providers > Add new > One-time PIN

[ ] Go to Cloudflare Zero Trust Dashboard (one.dash.cloudflare.com)

[ ] Integrations > Identity providers > Add new > One-time PIN

[ ] Save configuration

OTP Behavior: There is no static code to look up. A new code is generated per login attempt and emailed. If expired, request a new code. Add noreply@notify.cloudflare.com to email allowlists.

Create Access Policy

[ ] Zero Trust > Access > Applications > Your Pages app > Policies

[ ] Include: Emails = admin@yourdomain.com, guest@clientdomain.com

Part 8: Turnstile Bot Protection

Create Turnstile Widget

[ ] Cloudflare Dashboard > Turnstile > Add widget

[ ] Widget name: Portfolio Contact Form

[ ] Hostname Management:

IMPORTANT: Add your *actual* Pages hostname (e.g., portfolio-7ub.pages.dev), not the example "portfolio.pages.dev". Also add your custom domain when you have it. Use hostname only - no https://, no paths.

[ ] Widget Mode: Managed (recommended)

[ ] **Copy and securely store Site Key and Secret Key**

Token Validation Rules

Property	Value
Token expiration	300 seconds (5 minutes)
Reusability	Single-use only
Siteverify endpoint	https://challenges.cloudflare.com/turnstile/v0/siteverify

Development Testing Keys

Sitekey	Behavior
1x00000000000000000000AA	Always passes (visible)
2x00000000000000000000AB	Always fails (visible)
1x00000000000000000000BB	Always passes (invisible)

Part 9: Email Provider Setup (Resend)

Free Tier Limits

Limit	Value
Emails/day	100
Emails/month	3,000
Domains	1 per team
Rate limit	2 requests/second

Domain Verification

- [ ] Resend Dashboard > Domains > Add Domain
- [ ] Enter yourdomain.com
- [ ] Add DNS records Resend provides to Cloudflare DNS
- [ ] Wait 5-15 minutes, then click Verify DNS Records

Create API Key

- [ ] Resend > API Keys > Create API Key
- [ ] Permission: **Sending access** (least privilege for contact form)
- [ ] Restrict to your verified domain if possible
- [ ] **Copy API key immediately - shown only once**

Part 10: Pages Functions for Contact Form

Create Functions Directory

> **CRITICAL:** The `functions/` directory must be at project root, NOT inside src/.

```
portfolio/
  functions/           <-- At root, NOT in src/
    api/
      contact.js
```

Create Contact Form Function

Create `functions/api/contact.js`:

```
export async function onRequestPost(context) {
  const { request, env } = context;

  try {
    const formData = await request.formData();
    const name = formData.get('name');
    const email = formData.get('email');
    const message = formData.get('message');
    const turnstileToken = formData.get('cf-turnstile-response');

    if (!name || !email || !message) {
      return Response.json({ error: 'All fields required' }, { status: 400 });
    }

    // Validate Turnstile token
    const turnstileResponse = await fetch(
      'https://challenges.cloudflare.com/turnstile/v0/siteverify',
      {
        method: 'POST',
        headers: { 'Content-Type': 'application/x-www-form-urlencoded' },
        body: new URLSearchParams({
          secret: env.TURNSTILE_SECRET_KEY,
          response: turnstileToken,
          remoteip: request.headers.get('CF-Connecting-IP'),
        }),
      }
    );

    const turnstileResult = await turnstileResponse.json();
    if (!turnstileResult.success) {
      return Response.json({ error: 'Bot verification failed' }, { status: 400 });
    }

    // Send email via Resend
    const emailResponse = await fetch('https://api.resend.com/emails', {
      method: 'POST',
      headers: {
        'Authorization': `Bearer ${env.RESEND_API_KEY}`,
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({
        from: `Contact <noreply@${env.SENDER_DOMAIN}>`,
        to: [env.RECIPIENT_EMAIL],
```

```
        reply_to: email,
        subject: `Portfolio Contact: ${name}`,
        html: `<h2>New Contact</h2>
               <p><b>Name:</b> ${escapeHtml(name)}</p>
               <p><b>Email:</b> ${escapeHtml(email)}</p>
               <p><b>Message:</b></p>
               <p>${escapeHtml(message).replace(/\n/g, '<br>')}</p>`,
      }),
    });

    if (!emailResponse.ok) throw new Error('Failed to send');
    return Response.json({ success: true });

  } catch (error) {
    console.error('Contact error:', error);
    return Response.json({ error: 'Server error' }, { status: 500 });
  }
}

function escapeHtml(text) {
  const map = {'&':'&amp;','<':'&lt;','>':'&gt;','"':'&quot;',"'":"&#039;"};
  return String(text).replace(/[&<>"']/g, m => map[m]);
}
```

Configure _routes.json

Place in `public/_routes.json` (Astro copies to dist/):

```
{
  "version": 1,
  "include": ["/api/*"],
  "exclude": []
}
```

Set Environment Variables

Workers & Pages > Project > Settings > Variables and Secrets:

Variable	Type	Purpose
RESEND_API_KEY	Secret (encrypted)	Resend API authentication
TURNSTILE_SECRET_KEY	Secret (encrypted)	Server-side validation
SENDER_DOMAIN	Text (plaintext)	Domain verified with Resend
RECIPIENT_EMAIL	Text (plaintext)	Your inbox for submissions

SENDER_DOMAIN: The exact domain you verified with Resend (e.g., yourdomain.com). Function builds: noreply@${SENDER_DOMAIN}
RECIPIENT_EMAIL: Where contact submissions go (your Gmail or forwarding address).

Local Development Setup

Create `.dev.vars` at project root (add to .gitignore!):

```
RESEND_API_KEY=re_xxxxxxxxxxx
TURNSTILE_SECRET_KEY=0x...
```

```
SENDER_DOMAIN=yourdomain.com
RECIPIENT_EMAIL=you@gmail.com
```

Add to .gitignore:

```
.dev.vars
```

Build and run locally:

```
npm run build
npx wrangler pages dev dist
```

Part 11: Pagefind Static Search

Create Search Component

Note: Use lowercase `components` folder (not Components) to avoid Windows casing issues.

Create `src/components/Search.astro`:

```
---
import PagefindSearch from 'astro-pagefind/components/Search';
---

<PagefindSearch id="search" className="pagefind-ui" uiOptions={{ showImages: false }} />

<style is:global>
  .pagefind-ui {
    --pagefind-ui-scale: 1;
    --pagefind-ui-primary: #034ad8;
    --pagefind-ui-text: #393939;
    --pagefind-ui-background: #ffffff;
    --pagefind-ui-border: #eeeeee;
  }
</style>
```

Import Note: If you name your component Search.astro, alias the import as PagefindSearch to avoid "Import declaration conflicts with local declaration" errors.

Verify Build Output

After `npm run build`, you should see "Pagefind indexed X pages" and `dist/pagefind/` directory should exist.

"No files found" warnings: Normal if content folders are empty. Warnings disappear after adding .md files.

Part 12: Cloudflare Email Routing (Inbound)

- [ ] Cloudflare Dashboard > Email > Email Routing > Get Started
- [ ] Add destination (your Gmail)
- [ ] Click verification link
- [ ] Create custom address: contact@yourdomain.com > Forward to your inbox

Limitations: Forward-only. Cannot send FROM custom address. Replies come from your personal address.

Part 13: Web Analytics

> **Note:** Web Analytics is NOT auto-enabled. Manual enablement required.

- [ ] Workers & Pages > Project > Settings > Metrics > Enable Web Analytics

Features: 6-month retention, no cookies, GDPR-friendly, Core Web Vitals

Part 14: Security Headers

Create `public/_headers` (no extension). Verify dist/_headers exists after build.

```
/*
  X-Frame-Options: DENY
  X-Content-Type-Options: nosniff
  Referrer-Policy: strict-origin-when-cross-origin
  Permissions-Policy: accelerometer=(), camera=(), microphone=()
  Content-Security-Policy: default-src 'self'; script-src 'self' https://challenges.cloudflare.com https://static
```

Note: _headers does NOT apply to Functions responses. Set headers in function code.

Part 16: Deployment Workflow

Daily Development Cycle

```
# 1. Create feature branch
git checkout dev
git pull origin dev
git checkout -b feature/new-blog-post

# 2. Make changes and commit
git add .
git commit -m "Add new blog post"
git push -u origin feature/new-blog-post

# 3. Create PR: feature/new-blog-post -> dev
# 4. Review on Access-gated preview URL
# 5. Merge to dev

# 6. When ready for production: PR dev -> main
# 7. Merge to main triggers production deployment
```

Review on Preview URL: Find the preview deployment URL in Cloudflare Pages deployments. It should prompt for Access login (OTP) if Access policy is enabled.

Backup Checkpoint Before Major Changes

```
# Commit and push (primary backup)
git add -A
git commit -m "Checkpoint before Part 11"
git push

# Optional: Create tag
git tag pre-part11-2026-01-31
git push --tags

# Optional: Portable backup bundle
git bundle create ..\portfolio-backup.bundle --all
```

Part 17: Rollback Procedures

Revert Bad Production Deployment

Via Dashboard (instant, no rebuild):

Workers & Pages > Project > Deployments > Find working deployment > Rollback

Via Git (triggers rebuild):

```
git revert HEAD
git push origin main
```

Disable Functions Temporarily

```
# Option 1: Empty _routes.json
{ "version": 1, "include": [], "exclude": ["/*"] }

# Option 2: Rename directory
mv functions functions_disabled
git add . && git commit -m "Disable functions" && git push
```

Part 18: Pricing and Limits Reference

Service	Free Limit
Pages Static Requests	Unlimited
Pages Bandwidth	Unlimited
Pages Builds/month	500
Functions Requests/day	100,000
Functions CPU time	10ms per request
Access Users	50 seats
Turnstile	Unlimited
Email Routing	Unlimited forwarding
Web Analytics	6-month retention
Resend Emails/month	3,000 (100/day)

Total: $0/month (excluding domain ~$10.46/year)

Part 19: Final Verification Checklist

Accounts & Security

- [ ] GitHub 2FA enabled, recovery codes stored
- [ ] Cloudflare 2FA enabled
- [ ] Resend account + domain verified

Project Setup

- [ ] .node-version file (content: 22)
- [ ] Content config at src/content.config.ts (Astro 5.x)
- [ ] Content directories created (even if empty)
- [ ] astro-pagefind installed (NOT @astrojs/pagefind)

Cloudflare Pages

- [ ] Created via Pages > Connect to Git (NOT Workers)
- [ ] Build output: dist
- [ ] Environment variables set

Domain & Security

- [ ] Domain purchased, DNSSEC enabled
- [ ] Apex domain configured
- [ ] www redirect to apex
- [ ] _headers in public/ (NOT repo root)
- [ ] _routes.json in public/

Access & Turnstile

- [ ] Access enabled for previews
- [ ] OTP configured via Integrations > Identity providers
- [ ] Turnstile hostnames include actual Pages domain

Contact Form

- [ ] functions/api/contact.js at project ROOT
- [ ] .dev.vars in .gitignore
- [ ] Local test: npx wrangler pages dev dist
- [ ] End-to-end test successful

Appendix A: Comprehensive Troubleshooting Q&A;

Git / Windows Setup

Q: Where do I run git commands after installing Git for Windows?

Run in any terminal that can see Git: VS Code Integrated Terminal (recommended), Windows Terminal, or PowerShell. The --global flag applies to your user profile, so the folder doesn't matter.

Q: Where do I create .node-version?

You can't create it until the project exists. Create the Astro project first (Part 3), then place .node-version at project root next to package.json. Example: C:\dev\portfolio\.node-version with content: 22

Astro Development

Q: npm run dev "stalls" at "watching for file changes..."

That's normal - the dev server is running. Open http://localhost:4321/ in your browser. Stop with Ctrl+C.

Q: Where is src/content.config.ts located?

It's a sibling of pages under src (not inside content folder):
src/pages/
src/content/
src/content.config.ts <-- HERE

Q: Build warns "No files found matching **/*.{md,mdx}"

Normal if content folders are empty. Warnings disappear after you add .md/.mdx files.

Cloudflare Pages Deployment

Q: Build succeeds but deployment fails with "Missing entry-point to Worker script"

You're in Workers Builds flow, not Pages. The runbook requires Cloudflare Pages which publishes dist/ directly without "npx wrangler deploy". Fix: Delete project and create new one via Workers & Pages > Create > Pages > Connect to Git.

Q: Build warns "base directory does not exist" for content folders

Create the content directories (even empty) and optionally add .gitkeep files:
mkdir src\content\blog
ni src\content\blog\.gitkeep -ItemType File
Then commit and push.

Cloudflare Access / OTP

Q: Can't find OTP settings where runbook says "Settings > Authentication > Login methods"

UI moved. In Zero Trust, go to: Integrations > Identity providers > Add new > One-time PIN. If Access login already offers "email me a code", it may already be working.

Q: Can I retrieve or "know what the OTP is"?

No static code exists. A new code is generated per login attempt and emailed. If expired, request new code. Check spam; allowlist noreply@notify.cloudflare.com.

Turnstile

Q: What hostname do I add - portfolio.pages.dev or portfolio-7ub.pages.dev?

Add your ACTUAL Pages hostname (e.g., portfolio-7ub.pages.dev), not the example. Also add custom domain when ready. Use hostname only - no https://, no paths.

Appendix A: Troubleshooting Q&A; (continued)

Resend API

Q: API key permissions - "Sending access" or "Full access"?

Use "Sending access" for a contact form (least privilege). Full access only if you need admin operations.

Q: "All domains" vs specific domain for API key?

Restrict the key to your verified sending domain if possible (better security).

Environment Variables

Q: How do "Text/Secret" in UI map to "Plaintext/Encrypted" in runbook?

Secret (encrypted) = RESEND_API_KEY, TURNSTILE_SECRET_KEY
Text (plaintext) = SENDER_DOMAIN, RECIPIENT_EMAIL

Q: What values for SENDER_DOMAIN and RECIPIENT_EMAIL?

SENDER_DOMAIN: Exact domain verified with Resend. Function builds: noreply@${SENDER_DOMAIN}
RECIPIENT_EMAIL: Inbox that receives submissions (your Gmail or forwarding address)

Local Development

Q: Where do I create .dev.vars?

At project root (same level as package.json): C:\dev\portfolio\.dev.vars
Add to .gitignore so it's never committed.

Q: .dev.vars shows as untracked in git status

Add to .gitignore: .dev.vars
Verify file is named .dev.vars (not .dev.vars.txt): Get-ChildItem -Force .dev.vars*
If ever committed: git rm --cached .dev.vars && git commit

Q: Wrangler fails with "Top-level return cannot be used inside an ECMAScript module"

Your contact.js has a mismatched brace causing return to be outside a function. Replace escapeHtml with correct version:
function escapeHtml(text) { const map = {'&':'&'...}; return String(text).replace(...); }
Syntax check: node --check .\functions\api\contact.js

Pagefind / Search

Q: "File name differs only in casing: src/components vs src/Components"

Windows casing mismatch. Two-step rename:
Rename-Item .\src\Components components_tmp
Rename-Item .\src\components_tmp components
Then search/fix imports: Select-String -Path .\src***.astro -Pattern "Components/"

Q: "Import declaration conflicts with local declaration of 'Search'"

Name collision. Alias the import:
import PagefindSearch from 'astro-pagefind/components/Search';
<PagefindSearch ... />

Git Workflow

Q: PR says "Nothing to compare. dev and feature/... are identical."

Your feature branch has no unique commits. Either you never committed on it, or committed to dev instead.
Diagnose: git log --oneline origin/dev..origin/feature/new-blog-post
If empty, make commits on feature branch and push.

Q: Merge conflict: "README.md deleted in HEAD and modified in commit"

You're mid-merge with conflicts.
Abort: git merge --abort
Or resolve: git rm README.md && git add -A && git commit -m "Resolve merge"

Appendix B: File Structure Reference

```
portfolio/
■■■ public/
■   ■■■ _headers              # Security headers (HERE, not root)
■   ■■■ _routes.json          # Function routing (HERE, not root)
■   ■■■ favicon.svg
■   ■■■ images/
■■■ functions/                 # MUST be at root, NOT in src/
■   ■■■ api/
■       ■■■ contact.js
■■■ src/
■   ■■■ components/           # Lowercase (avoid Components casing issue)
■   ■   ■■■ Navigation.astro
■   ■   ■■■ Search.astro
■   ■■■ layouts/
■   ■   ■■■ BaseLayout.astro
■   ■■■ pages/
■   ■   ■■■ index.astro
■   ■   ■■■ resume.astro
■   ■   ■■■ portfolio/
■   ■   ■■■ blog/
■   ■   ■■■ studies/
■   ■   ■■■ certifications/
■   ■   ■■■ archive/
■   ■■■ content/
■   ■   ■■■ blog/
■   ■   ■■■ portfolio/
■   ■   ■■■ studies/
■   ■   ■■■ certifications/
■   ■■■ styles/
■   ■■■ content.config.ts      # HERE (sibling of pages, NOT in content/)
■■■ astro.config.mjs
■■■ package.json
■■■ .node-version              # Pin Node version (22)
■■■ .dev.vars                  # Local secrets (MUST be in .gitignore)
■■■ .gitignore
```

Appendix C: Environment Variables Reference

Variable	Example	Type	Purpose
RESEND_API_KEY	re_abc123...	Secret	Resend API
TURNSTILE_SECRET_KEY	0x4AAA...	Secret	Server validation
SENDER_DOMAIN	yourdomain.com	Text	Email from domain
RECIPIENT_EMAIL	you@gmail.com	Text	Inbox for submissions

Appendix D: Quick Command Reference

Development

```
npm run dev                    # Start dev server (localhost:4321)
npm run build                  # Build for production
npx wrangler pages dev dist    # Test with Functions locally
node --check file.js           # Syntax check JavaScript
```

Git Workflow

```
git checkout -b feature/name   # Create feature branch
git add -A                     # Stage all changes
git commit -m "message"        # Commit
git push -u origin branch      # Push new branch
git log --oneline dev..branch  # Compare branches
```

Troubleshooting

```
npx wrangler pages deployment tail    # View function logs
Get-ChildItem -Force .dev.vars*       # Check file exists (Windows)
Select-String -Path src/**/*.astro -Pattern "text"  # Search files
```

Backup

```
git tag checkpoint-name        # Create tag
git push --tags                # Push tags
git bundle create backup.bundle --all  # Portable backup
```

--- End of Runbook ---

Version 2.0 - With Comprehensive Troubleshooting Q&A;

Verified January 2026