Florida Flood Gauge Monitor

Documentation

Paul Fishwick and Claude Code

February 2026

1 Overview

The Florida Flood Gauge Monitor is a web application that displays real-time water level data from over 500 USGS monitoring stations across Florida. Users can explore an interactive map, view 7-day historical charts for individual gauges, and generate AI-powered narratives explaining water level behavior.

2 Architecture

The application uses a three-tier architecture that keeps API keys secure while serving a static frontend from GitHub Pages.

2.1 System Overview

```
User's Browser
     |
     v
GitHub Pages (static site)
     |
     |--- GET ---> USGS Water Services API (public, no key)
     |
     |--- POST --> Cloudflare Worker (serverless proxy)
                         |
                         |--- API key added server-side
                         v
                   Google Gemini API
```

2.2 Frontend (GitHub Pages)

The static site is built with:

- **React + TypeScript** – UI framework
- **Vite** – Build tool and dev server
- **Leaflet** – Interactive map rendering via OpenStreetMap
- **Recharts** – Time series chart visualization
- **vite-plugin-pwa** – Progressive Web App support (service worker, manifest, offline caching)

The frontend calls the USGS API directly (public, no key required) and routes all Gemini requests through the Cloudflare Worker proxy.

2.3 Serverless API Proxy (Cloudflare Worker)

A Cloudflare Worker acts as a secure intermediary between the frontend and the Google Gemini API. This pattern is essential because:

1. **API key security** – The Gemini API key is stored as an encrypted secret on Cloudflare, never exposed in browser code or source repositories.
2. **No server management** – The Worker runs on Cloudflare's edge network. There is no server to maintain, patch, or scale.
3. **Free tier** – Cloudflare Workers provide 100,000 requests/day at no cost.
4. **Reusable** – The same Worker (or pattern) can serve multiple projects that need the same API key.

The Worker is deployed at `api-proxy.metaphorz.workers.dev` and exposes a single endpoint:

- `POST /api/story` – Accepts `prompt` and `systemInstruction` as JSON, forwards to Gemini, returns the generated text.

The Worker source code is located at `../api-proxy/api-proxy/src/index.ts` relative to the main project.

2.3.1 Why Not Use `VITE_` Environment Variables?

Vite automatically inlines any environment variable prefixed with `VITE_` into the JavaScript bundle at build time. This means a variable like `VITE_GEMINI_API_KEY` becomes a plain string visible to anyone who inspects the deployed JavaScript. Removing the `VITE_` prefix prevents Vite from inlining the value, but then the frontend cannot access it either. The serverless proxy solves this by keeping the key entirely server-side.

2.3.2 Managing the Worker

The Worker is managed using the `wrangler` CLI:

```
# Deploy updates
cd api-proxy/api-proxy && npx wrangler deploy

# Set or rotate the API key
echo "new_key_here" | npx wrangler secret put GEMINI_API_KEY

# View live logs
npx wrangler tail
```

2.4 Progressive Web App (PWA)

The application is configured as a PWA, allowing users to install it on mobile devices and desktops:

- **Service Worker** – Generated by `vite-plugin-pwa` during production builds. Precaches static assets for fast loading.
- **Web App Manifest** – Defines the app name ("Florida Flood Gauge Monitor"), theme color, and icons for home screen installation.

- **Runtime Caching** – USGS API responses are cached with a `NetworkFirst` strategy (5-minute expiration), so recent data is available even on flaky connections.
- **Icons** – Custom gauge/water icons at 192x192 and 512x512 pixels.

To install: visit the deployed site in Chrome and click the install icon in the address bar, or on mobile Safari use Share → "Add to Home Screen."

2.5 Data Flow

1. On page load, the frontend fetches 7 days of gauge height data from the USGS Water Services API for all active Florida monitoring sites.
2. Data is parsed, sorted chronologically, and rendered as map markers.
3. When a user clicks a marker, a 7-day chart is displayed.
4. When the user clicks "Story," three requests (summary, standard, detailed) are sent to the Cloudflare Worker, which adds the API key and forwards them to Gemini.
5. Responses are cached client-side so switching between detail levels is instant.

3 Features

3.1 Interactive Map

The main view displays all active USGS flood gauges in Florida as blue markers on an OpenStreetMap base layer, as shown in Figure 1. Users can pan and zoom to explore gauge locations across the state.

3.2 Gauge Chart

Clicking on any gauge marker opens a bottom panel displaying a 7-day historical chart of water level changes, as shown in Figure 2. The y-axis shows water level change in feet, and the x-axis shows the date range. A "Story" button is available to generate an AI explanation.

3.3 AI-Generated Story

The Story feature uses Google Gemini to generate a narrative explanation of the water level data, as shown in Figure 3. Three detail levels are available:

- **Summary** – One to two sentences
- **Standard** – A single paragraph
- **Detailed** – Two to three paragraphs

All three levels are fetched concurrently and cached, so switching between them is instant.

4 Setup

4.1 Prerequisites

- Node.js (v18+)
- npm

4.2 Installation

```
npm install
```

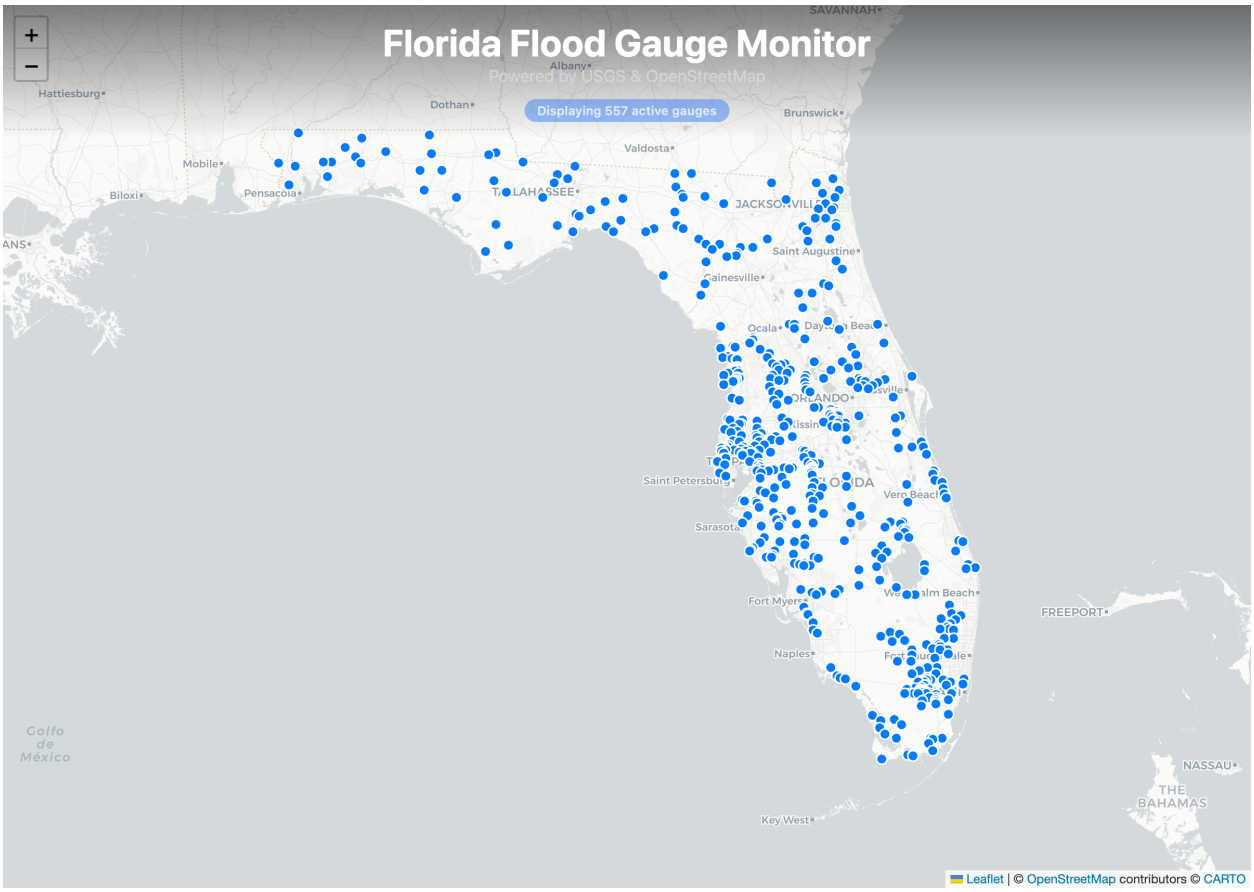

Figure 1: Main map view showing 557 active USGS flood gauges across Florida.

4.3 API Key Configuration

The Gemini API key is stored securely on the Cloudflare Worker, not in the application code. The Story feature works automatically on the deployed site without any local key configuration.

For developers who wish to modify the Worker or rotate the key:

1. Obtain a free key at https://aistudio.google.com
2. Store it in `~/.env` (home directory, outside the project):

   ```
   GEMINI_API_KEY=your_key_here
   ```

3. Set it on the Worker:

   ```
   cd api-proxy/api-proxy
   echo "your_key_here" | npx wrangler secret put GEMINI_API_KEY
   ```

Important: Never use the `VITE_` prefix for secret keys. Vite inlines `VITE_`-prefixed variables into the JavaScript bundle, making them visible to all users.

4.4 Running Locally

```
./start
```

This script kills any existing process on port 5173, starts the Vite dev server, and opens the browser.

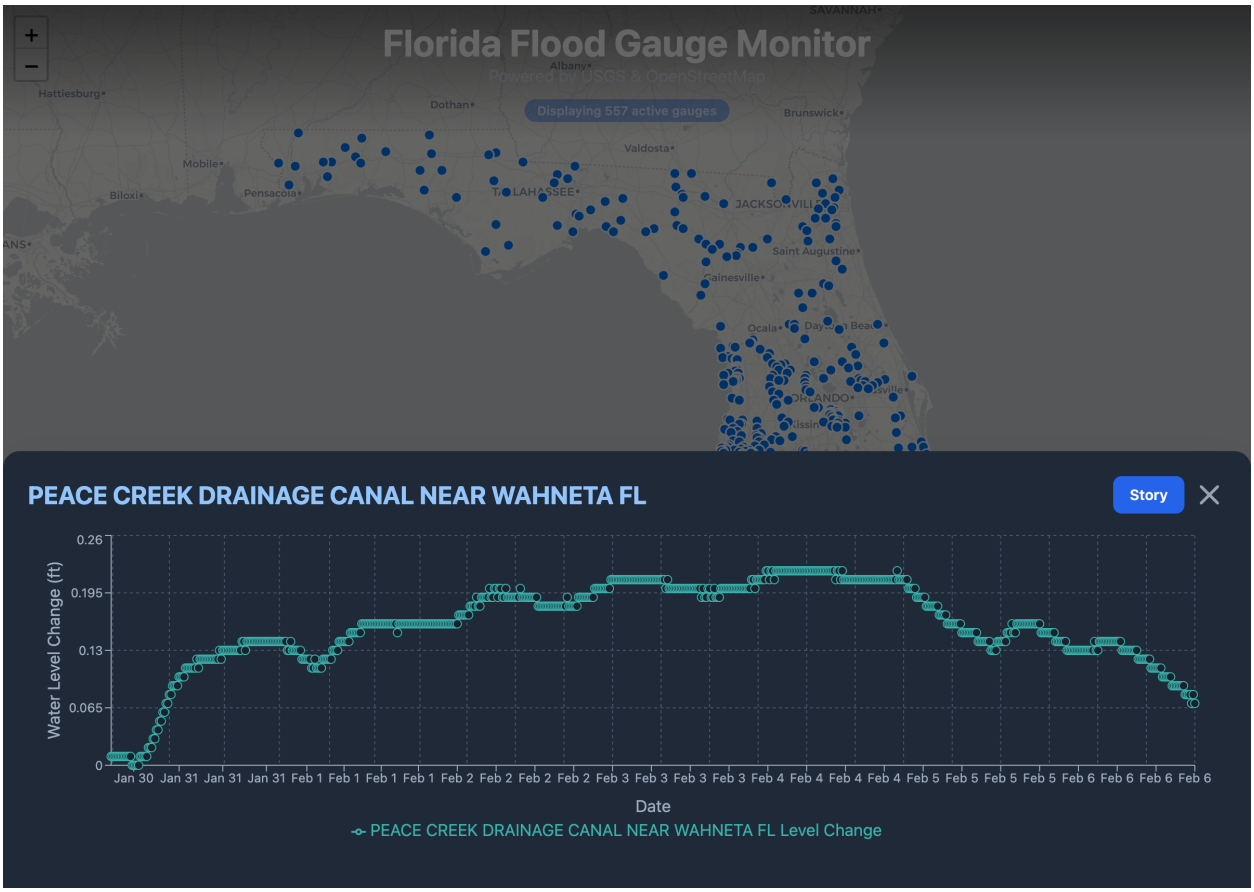

Figure 2: Historical water level chart for a selected gauge, with the Story button visible.

4.5 Deploying

```
./deploy
```

This script builds the production bundle (with PWA assets), verifies no secrets leaked into the output, clears the `gh-pages` cache, and publishes to GitHub Pages.

5 License

This project is released under the MIT License.

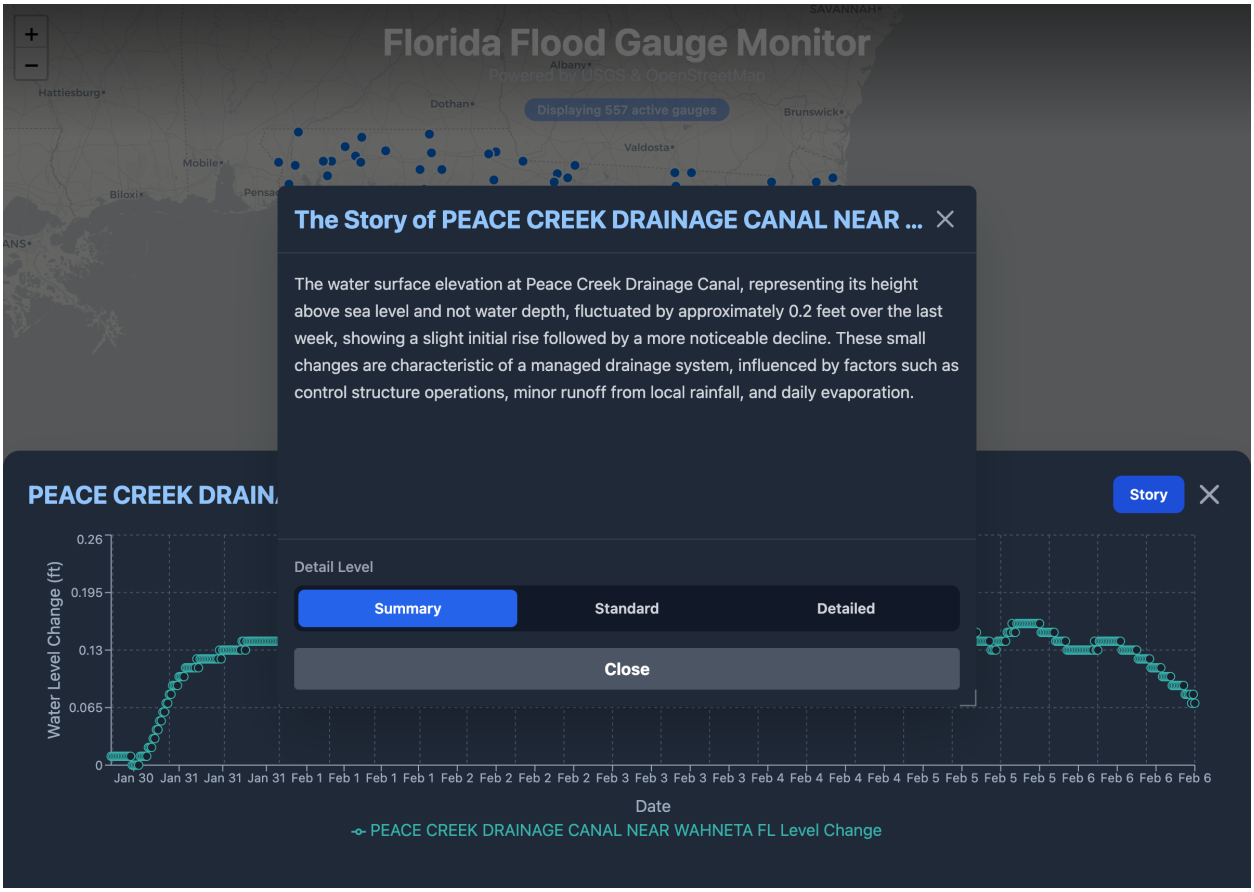

Figure 3: AI-generated story modal explaining water level behavior for a selected gauge.