

Running Spiti — a step-by-step guide

Written for someone who has never opened a “terminal.” If a step ever mentions a word you don’t know, it’s explained right there. You cannot break anything by trying.

What changed in this version (for Hicham)

This is the same Spiti, grown. You receive the **whole product**, not a patch file: run it exactly as before. Everything below is new, and it all works the moment you sign in. Nothing from the first version was removed or changed.

- **Two new roles at the brand.** A **Client Director** (the maison's most senior person) now manages who holds power at the house: they appoint and remove Brand Approvers and Zone Managers. They see a guest **count** per event and never a guest list. A **Zone Manager** oversees a group of stores: they judge nominations across their whole zone and appoint the store managers under them, their zone only. Both are on the sign-in role-picker (Irène Lacroix and Priyanka Rao) so you can walk in as each.
- **Create events from the screen.** A **New event** button (as OC Operations) opens a form for a new production, including a **guest cap** and a **nomination deadline**. Past the deadline, new nominations are refused. Approve past the cap and the overflow goes to the waitlist automatically.
- **Approve many at once.** An approver can now tick several nominees and approve, decline, or request-info on all of them in one click. Each one is still decided and recorded individually.
- **A hotel catalog.** Room blocks now pick a hotel from a filterable catalog (by category, country, city, neighborhood) instead of typing a name.
- **A restaurant / experience portal.** Restaurants and activity vendors get their own scoped view, the same way the hotel and transport companies already do, seeing only what they need to set a table or run an experience.
- **Import and export the guest list.** Bring a list in from a spreadsheet, or export the list you can see (never fields you can't).
- **More message lanes.** The concierge can now message the hotel and the brand on separate, walled lanes, on top of the guest and internal lanes that already existed.

- **A tidier menu.** An event's inner pages stay out of the left menu until you open that event, so the menu is never cluttered.

For your technical team: the server setup, backups, and scaling notes now live in

`DEPLOYMENT.md` (also in this folder). This `RUN.md` stays the first-timer's guide;

`DEPLOYMENT.md` is for whoever runs your servers.

Spiti is your portal. This guide takes you from the files in this folder to a live web address your team and vendors can sign into — and then walks the first-time secret setup so **you** own everything from day one.

Part 1 — Where to run it (pick one)

Spiti is a small, standard web application. It runs anywhere that can run Node.js. Three good, current options are below, with **today's real prices (checked on each provider's own pricing page, July 2026)**. All three deploy straight from a code folder — no “terminal” required.

Platform	What you'll pay	Why you'd pick it	Terminal needed?
Render ★ <i>recommended</i>	\$0 to try (Hobby workspace) → \$7/month for an always-on web service (Starter), + \$6/month if you later move to their managed database.	The gentlest click-through: connect a folder, it detects Node, it deploys. Predictable flat pricing — no surprise “usage” bill.	No
Railway	\$5/month (Hobby), which <i>includes</i> \$5 of usage — a small app like this typically stays inside that.	Fastest, friendliest dashboard; databases one click away if you grow. Bills by usage, so a quiet month costs less.	No
Fly.io	Usage-based; a tiny always-on machine runs in the low single dollars/month.	Most control and global regions (useful if you want EU data residency for Paris). Slightly more technical.	A little

Prices verified July 2026 on render.com/pricing and railway.com/pricing.

Cloud prices change — glance at the provider's pricing page before you commit. Re-check

MADE BY KALDUNE

yearly.

Our recommendation: Render, on the free Hobby workspace to start. It's the simplest to stand up, the pricing is a flat number you can predict, and you can upgrade the one web service to \$7/month for always-on when you go live. The rest of this guide uses Render; Railway is nearly identical if you prefer it.

Spiti stores its data in a single file (`data/spiti.db`) that lives with the app, so **you do not need a separate database to get running**. When you outgrow that, RUN.md's Part 5 explains the one setting to change.

Part 2 — Put the code somewhere Render can see it (GitHub)

Render deploys from **GitHub** — a free website that stores code folders (called “repositories,” or “repos”). You'll make a free account and upload this folder once. After that, Render watches it.

1. Go to **github.com** and click **Sign up**. Make a free account.
2. Once signed in, click the + in the top-right corner → **New repository**.
3. Give it a name — `spiti-portal` is fine. Choose **Private** (only you can see it). Click **Create repository**.
4. On the next page, find the link that says “**uploading an existing file**” (it's in the small grey text). Click it.
5. Open the folder these files came in on your computer. Select **everything** inside it and drag it into the GitHub upload box. Wait for the files to finish uploading.
 -  **Do not upload a file called `.env` if one exists.** The included `.gitignore` is set up to keep secrets out automatically, but if your computer shows a `.env`, leave it out. The safe one to upload is `.env.example` (no real secrets in it) — that one is fine and helpful.
6. Click the green **Commit changes** button. Your code now lives in GitHub.

Part 3 — Deploy it on Render

1. Go to **render.com** and click **Get Started**. Sign up — choose “**Sign in with GitHub**” so Render can see the repo you just made.
2. In the Render dashboard, click **New +** (top right) → **Web Service**.

3. Render shows your GitHub repos. Find `spiti-portal` and click **Connect**.
4. Render reads the project and fills most fields in for you. Check these:
 - **Name:** `spiti` (this becomes part of your web address).
 - **Region:** pick the one closest to your team. (*For Paris/EU data, choose **Frankfurt**.*)
 - **Build Command:** `npm install`
 - **Start Command:** `npm start`
 - **Instance Type:** **Free** to try, or **Starter (\$7/mo)** so it never sleeps. (Free “sleeps” after 15 minutes idle and takes ~30 seconds to wake.)
5. Scroll down to **Environment Variables** — this is Part 4. Do that now, before you click Create.

Part 3.5 — Add the brand fonts (one-time, 10 minutes)

Spiti is designed in four typefaces (Cormorant Garamond, Inter, Newsreader, IBM Plex Mono). The app never fetches fonts from the internet at runtime. That's deliberate, so it works on a locked-down network. The trade-off: the four font files need to live **with** the app, and they're not included in the upload (they're large binaries). **If you skip this, the portal still works perfectly — it just renders in your computer's built-in serif instead of the Spiti lettering.** Ten minutes now makes the brand render correctly for everyone.

1. Go to **gwfh.mranftl.com** (a free site that hands you ready-made font files — no design software needed).
2. Search each of these four names in turn and download them: **Cormorant Garamond**, **Inter**, **Newsreader**, **IBM Plex Mono**. (Pick the "Modern Browsers / woff2" option if it asks.)
3. You'll end up with a set of `.woff2` files. Rename them to exactly these twelve names:

cormorant-garamond-300.woff2	cormorant-garamond-400.woff2
cormorant-garamond-500.woff2	cormorant-garamond-300italic.woff2
cormorant-garamond-400italic.woff2	inter-400.woff2
inter-500.woff2	inter-600.woff2
newsreader-400.woff2	newsreader-400italic.woff2
ibm-plex-mono-400.woff2	ibm-plex-mono-500.woff2

4. In your GitHub repo (from Part 2), open the `public/fonts` folder, click **Add file** → **Upload files**, and drag all twelve in. Commit. Render will redeploy automatically and the brand fonts now render for everyone, on any network. (*The folder already contains a `README.md` repeating these exact steps and the license note — all four fonts are free to use and self-host.*)

(These fonts are all free/open-license. If you'd rather we prepare the twelve files for you, just ask in the chat and we'll hand them over ready to drop in.)

Part 4 — Set your secrets (first-time setup — do this once)

This is the most important part, and it’s the lesson from before: secrets never live in the code. You paste them into Render’s own secure box, and only you can see them.

Still on the “New Web Service” screen, find **Environment Variables** and click **Add Environment Variable** for each row below. The file `.env.example` in your folder lists and explains every one; here are the ones that matter to get running:

Key (type exactly)	Value to paste	What it is
<code>SESSION_SECRET</code>	a long random string — see the tip below	Signs login sessions so they can’t be forged.
<code>APP_URL</code>	leave blank for now; after the first deploy, set it to your real address (e.g. <code>https://spiti.onrender.com</code>)	Used to build sign-in links in emails.
<code>EMAIL_PROVIDER</code>	<code>dev</code>	Keeps sign-in links on-screen until you connect real email (Part 6).
<code>FLIGHT_PROVIDER</code>	<code>simulated</code>	The flight feed is a realistic simulation, clearly labelled in the app.

Tip — making a `SESSION_SECRET`: open a password generator (or your password manager’s “generate password” button), make one about 40–60 characters long, and paste it in. Any long random string works. Don’t reuse it anywhere else.

Now click **Create Web Service**. Render will build and start Spiti — this takes a couple of minutes the first time. When it finishes, the top of the page shows your live address, like `https://spiti.onrender.com`.

One quick follow-up: copy that address, go to **Environment** in the left menu, set `APP_URL` to it, and click **Save Changes**. Render restarts with the correct link address. Done.

Part 5 — Open it and sign in

1. Click your new address (e.g. `https://spiti.onrender.com`). You'll see the Spiti sign-in door.
2. To walk through the whole product, use the “**walk in as any role**” buttons — they sign you straight in as Marielle (Operations), Tomás (Concierge), an Approver, a Store Manager, the Hotel, the Transport company, and there's a button to **view the guest itinerary** on your phone.
3. To try the **real** sign-in: type an email into the top box. Because `EMAIL_PROVIDER` is `dev`, the secure link is shown on the next screen (in a live setup with email connected, it arrives in the inbox instead).

The portal comes pre-loaded with a full example event, the *LV Cruise 2027 Preview* in Paris, with named guests, a hotel block at the Crillon, and cars timed to real-looking flights, so every screen is populated from the first moment. (It's clearly labelled illustrative sample data; you replace it with your own real events.)

See the signature feature work: sign in as **Tomás (Concierge)**, open **Mr. Aaron Vance**, go to the **Flights** tab, and press “**Delay 90 min.**” Then open the **Transport** and **Hotel** tabs, and view the **guest itinerary** — the pickup time, the hotel arrival, and the guest's day have all re-timed themselves from that one press. Nobody re-typed anything.

Part 6 — Connecting real email later (on a domain you own)

This is the other hard lesson from before: **your sign-in email must run on a domain your company owns** — never a former agency's mailbox. When you're ready:

1. Make a free account at **Postmark** (postmarkapp.com) or **SendGrid**.
2. In their dashboard, add and **verify your own domain** (e.g. `octravel.com`) — they walk you through adding a few DNS records at your domain registrar. This proves the domain is yours and lets email send *from* it.
3. Copy your **API key** from Postmark/SendGrid.
4. Back in Render → **Environment**, change these:
 - `EMAIL_PROVIDER` → `postmark` (or `sendgrid`)
 - `EMAIL_API_KEY` → paste your key
 - `EMAIL_FROM` → e.g. `concierge@octravel.com` (on your verified domain)

5. Save. Sign-in links now arrive as real email, from *your* domain, on *your* account. If one email provider ever has an outage, you switch to another on the same owned domain — no one can lock you out.

Connecting the **live flight feed** works the same way: make a **FlightAware AeroAPI** or **Cirium** account, set `FLIGHT_PROVIDER` to `flightaware` (or `cirium`) and paste `FLIGHT_API_KEY`. Until you do, the simulation runs and is labelled “simulated” everywhere it appears — nothing is ever presented as live when it isn’t.

Part 7 — You own all of it

- **The code** is yours, in your own GitHub repo. No agency holds it.
- **The secrets** are in your own Render account, and you can change any of them without touching the code. Rotate them any time (and we recommend rotating every provider key now, at handover, so nothing anyone else has seen stays live).
- **The domain** for email and sign-in is yours.
- **The data** is in your own database file (or your own managed database when you scale). It’s exportable; nothing traps it.

When you’ve done this

Come back to this chat, paste what you see (the address, or any message on the screen) **and I’ll walk you through it from there.** If a step didn’t look like this guide described, tell me exactly what you saw and we’ll sort it together.