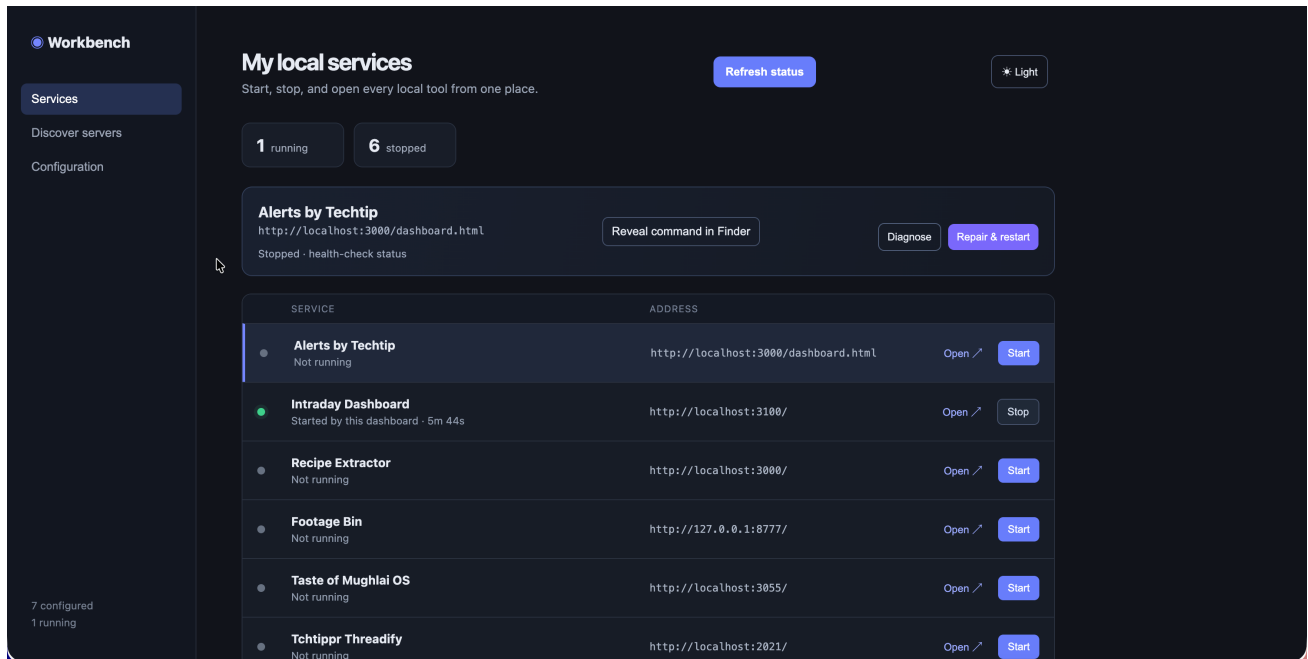


Local Server Dashboard

A visual quick-start guide for Mac users building browser-based tools.



Built from Asif's personal Local Control workflow. This is a build-your-own guide, not a downloadable dashboard installer.

Understand the three layers

Browser dashboard → Local Node helper → Approved app launchers

The browser displays controls. The helper runs on your computer and manages configured launchers. Your applications keep their own servers and dependencies.

LOCAL ONLY: Bind the helper to 127.0.0.1. Do not publish or tunnel its command-control endpoint.

Best for: creators with personal tools, AI-assisted app builders, developers switching demos, and learners organizing experiments.

Start with one service

A short path from scattered launchers to a working dashboard.

1. Inventory your app

Write down its name, real launcher path, project directory, browser URL, and reliable stop method. Confirm the existing launcher works.

2. Prepare your project

Install a supported Node.js LTS release. Check `node --version` and `npm --version`. Create a dedicated Local Control folder and open it in your coding assistant.

3. Build with the prompt

Use the brief on the next page. Ask for implementation, tests, and a README. Test with a disposable sample service before connecting real projects.

4. Configure one app

Use an absolute launcher path and the exact URL. Assign a distinct port. Review optional stop commands: never stop an unverified process just because it occupies a port.

5. Test the complete lifecycle

Run `npm start` in the generated dashboard folder if its README defines that script. Open `http://127.0.0.1:4610`. Test Start → wait → Open → Stop → Refresh.

6. Expand gradually

Review discovered launchers individually. Add a Mac launcher only after the helper works. Keep paths stable; moving the folder may require rebuilding that launcher.

SAVE THIS SERVICE CARD: Name · Launcher path · URL/port · Verified stop method · Health check · Log location

A JSON configuration sample is in the accompanying article. Your generated implementation may use a different schema; follow its README.

Build prompt for Codex

A specification to adapt—not a promise of identical generated code.

Paste this into a local project chat. Supply your operating system, launcher inventory, and screenshot if you want a similar interface.

Build a local server control dashboard for my Mac in this project folder.
Use a browser UI and a small Node.js helper bound to 127.0.0.1 only.
Use a configurable dashboard port, default 4610. Do not expose it to the LAN.

Show configured services with name, URL, launcher path, status, Start, Stop, Open, Refresh, and Reveal in Finder. Add light/dark mode, configuration editing, and manual-review discovery of launcher scripts. Use JSON configuration and validate every field. Execute only approved launchers; never automatically run a discovered script.

Track process ownership and separate reachable URLs from managed processes. Verify app identity and process ownership before stopping anything. Never kill an arbitrary process because it occupies the expected port. Detect port conflicts, preserve logs, and show starting and failed states. Handle child processes, helper restarts, and stale PID records safely. Restrict browser requests with Host/Origin checks and CSRF protection. Do not accept arbitrary shell commands from unauthenticated requests.

Add Diagnose. Make Repair a confirmed, documented cleanup/restart action, not an automatic rewrite of application code. Do not delete project files. Create a Mac launcher that avoids duplicate helper instances. Keep existing apps unchanged. Test with a disposable sample service first. Test missing files, slow startup, occupied ports, external processes, helper restart, and graceful stopping. Provide README setup instructions, configuration examples, known limitations, and a rollback procedure.

IMPORTANT: This brief asks for stronger safeguards than Asif's current personal version. Have the assistant demonstrate the tests before you trust Start, Stop, or Repair.

Your launch-day checklist

A dashboard should make local work clearer, not conceal failures.

- Each URL identifies the intended app, not merely an HTTP response.
- Ports are distinct, or collisions fail visibly rather than shifting silently.
- Stop verifies ownership and handles child processes safely.
- Externally started apps are not presented as dashboard-owned.
- Helper restarts do not make stale PID records dangerous.
- Logs are available for failed or slow launches.
- Only approved commands run; discovered scripts require review.
- Loopback binding and browser request protections are tested.

If something goes wrong

Symptom	First check
Wrong app opens	Saved URL, port collision, and app-specific health identity.
Start does nothing	Launcher exists; runtime is available; inspect its logs.
Stop leaves app running	Child process ownership and the app's documented stop method.
Discovery misses an app	Spotlight indexing or unusual launcher; add it manually.
Launcher breaks after a move	Absolute paths and rebuilt Mac launcher.

“Repair” should mean explained cleanup and restart. It is not a magic fix for missing dependencies or broken application code. Closing a browser tab does not necessarily stop its server.

Official references

[Node.js download](#)
[Codex resources](#)
[Vite port / strictPort](#)
[Python local HTTP server](#)