LEARN

The "Zero-Python" Linux Mint Setup

A native, offline-capable development environment — no Python, no Node.js, no Snap, if you can avoid it.

Abstract: Modern development environments are increasingly bloated with heavy runtime dependencies, mandatory package managers like `npm` or `pip`, and sandboxed formats like Snap. This guide outlines a clean, native setup on Linux Mint Cinnamon focused on offline capability, system stability, and avoiding Python and Node.js dependencies wherever a native alternative exists.

1. Core package management: Nala instead of bare apt

For native .deb package management, **Nala** (gitlab.com/volian/nala) gives a much cleaner, more readable interface than plain apt, with parallel downloads and clearer dependency output.

- **Install:** `sudo apt install nala`
- **Usage:** from here on, `sudo nala install <package>` and `sudo nala upgrade`.
- **Rule of thumb:** always prefer a native .deb via Nala first. If it's not in the repos, look for a standalone .AppImage. Use Flatpak only as a last resort. Avoid Snap packages — Mint's own philosophy agrees here, and Snap's forced auto-updates and slower cold-starts aren't worth it for dev tooling.

2. The heavy lifting: FPC and Lazarus

For native desktop applications, Free Pascal (FPC) and the Lazarus IDE remain hard to beat for rapid, cross-platform development with zero runtime dependencies — compiled binaries just run, no interpreter, no venv, no node_modules.

- **Setup:** `sudo nala install fpc lazarus-ide`
- **Offline advantage:** once installed, Lazarus produces static binaries. No internet needed at build time — the project keeps compiling indefinitely, immune to a registry going down or a package being yanked years later.

3. Web and scripting editor: VSCodium

Lazarus handles native desktop apps; for vanilla HTML/CSS/JS, PHP, and Bash you still want a capable editor. **VSCodium** (vscodium.com) gives you VS Code's editing experience without Microsoft's telemetry/marketplace lock-in.

- **Install:** grab the .deb from the VSCodium releases page (github.com/VSCodium/vscodium/releases), then `sudo dpkg -i <file>.deb`.

- **Keep it light:** minimize extensions. Rely on native syntax highlighting over heavy language-server extensions that quietly pull in their own Python/Node runtime in the background — the whole point of this setup is avoiding exactly that.

4. Local web server stack

Testing PHP backends and vanilla JS frontends needs a local, offline server.

- **Setup:** `sudo nala install apache2 php libapache2-mod-php`
- **Testing:** point your browser at localhost, keep project folders under /var/www/html/ (or symlink them in from your home directory). This lets you test GET/POST flows and any custom encryption end to end before anything touches a live server.

Why this matters

None of this is dogma — Python and Node are fine tools when the job actually needs them. The point is a *default*: reach for the native, dependency-free option first, and only pull in a heavier runtime when there's a real reason to. A dev environment built this way survives longer, breaks less between distro upgrades, and keeps working when you're offline.