The Cutover

Technical breakdown for the Build Note “The Move to the Mini” · Michael Gaff · November 2026

Why laptops make bad servers

A laptop is designed to be closed, carried, updated and put to sleep. Every one of those is a failure mode for a machine that must answer at 6:45 in the morning. The trigger here was ordinary: an overnight operating system update restarted the machine, and it came back with its network interface off. Nothing crashed and no data was lost. The machine was unreachable for three days because the role did not match the hardware.

The inventory

Before moving anything, write down everything the old host does. Four lists cover it. Scheduled jobs: every launch agent, its schedule, its interpreter and its log path. Files: everything the system writes, split into code (lives in git), configuration (the laptop was the source of truth) and runtime state (lives only on production). Listeners: every service and the port it binds, including which bind to localhost only. Secrets: which keys each job needs and where they come from, by name only. The fourth list is the one teams skip and the one that breaks the week after.

Cutover day

Treat the old host as read only from the start of the day. Install runtimes on the new host first, pinned to explicit paths rather than whatever the shell finds. Move code through git, configuration by copy with a diff, and runtime state last, once the old jobs are stopped, so nothing is written in two places. Bring services up in dependency order: data stores, then the application server, then schedulers, then anything that talks to the outside world. A tunnel whose routes all point at localhost moves every hostname at once when its connector moves, which is convenient and also a single switch that must be flipped deliberately.

Hardening an unattended host

Automatic login, so a power event does not strand the machine at a password prompt. Sleep disabled. Automatic restart after power loss. Operating system updates on a schedule you choose, not one the vendor chooses. A watchdog on a different machine that checks the new host from outside, because a host cannot report its own silence.

The week after

Expect quiet failures, not loud ones. Four classes showed up. Wrong environment: a scheduled task configured to run in a cloud sandbox instead of on a machine failed nightly with no access to any of the files it needed. Moved addresses: a service took a new port during the move and its clients kept calling the old one, so it looked dead while running fine. Permission changes: the new operating system denied the system Python under the background scheduler, so jobs pinned to it stopped while the same scripts ran by hand. Stale waits: a page blocked on a mail scan that the new architecture made unnecessary, turning a fast endpoint into a 25 second timeout. The fix for that last class was a stale while revalidate layer: return the last good answer instantly and refresh in the background.

The verification probe

After the cutover, enumerate every endpoint every page calls and hit each one, recording status, latency, payload size and the freshest timestamp in the response. Write the results to a file and run it again after each fix. Two cautions. Separate endpoints that only accept writes from real failures, or the report fills with false alarms. And treat old timestamps as questions, not verdicts: confirm that something still writes a field before calling it stale.

Lessons

Separate duty from desk: production belongs on a machine nobody types on. Plan for the week after the cutover, because that is where the work is. Pin interpreters and paths explicitly. Keep a written inventory, and keep it current, because the next migration starts from it.

Architecture and process only. No personal data, credentials, account details or private addresses are included.