OWN YOUR INFRASTRUCTURE

TGP // FIELD MANUAL 01

THE ENGINE ROOM

Turn dead laptops into always-on servers that quietly run your websites, your automations, and your entire operation. No cloud landlord. No monthly bill. Hardware you already own.

RAFAEL HUNNA // The Grind Protocol

The Engine Room: Field Manual 01.

CONTENTS

What's Inside

CHAPTER 00

Why a Dead Laptop Beats the Cloud

Before you spend a dollar, understand what you already own.

Everyone selling you a "home server" wants you to buy a Raspberry Pi, then a case, then a power supply, then a microSD card, then an SSD because the microSD died. Everyone selling you cloud wants $50 a month, forever, for a machine you never touch and never own.

Meanwhile there is a laptop in a drawer in your house right now that is faster than both.

An old laptop is the most underrated server on earth. Here is why it wins:

WHAT IT HAS	WHY IT MATTERS
`Built-in battery`	It is its own backup power supply. The lights flicker, the Pi dies, the cloud charges you for "high availability." Your laptop just keeps running on battery like nothing happened.
`Real storage + RAM`	A typical old laptop has 8GB of RAM and a real hard drive or SSD. Renting that in the cloud runs about $50/month. You already paid for it once.
`Desktop-grade I/O`	Real USB, real networking, a real keyboard and screen for setup. With a Pi you buy all of that separately.
`It is free`	The cheapest server is the one already sitting in your closet.

> THE NUMBER
>
> A shared cloud machine with 8GB RAM and 4 cores runs around **$50 a month**. That is **$600 a year**, every year. Your old laptop has those specs already. Over five years, that drawer laptop just saved you roughly **$3,000**.

This manual turns that drawer laptop into what I call an **Engine Room**: a quiet, always-on machine that hosts your websites, runs your automations, stores your files, and works while you sleep. One laptop is a server. Three laptops is infrastructure. A room full of them, owned outright, is leverage nobody can shut off or raise the price on.

Let's build it.

CHAPTER 01

Pick the Right Machine

You need less than you think. A cracked screen is not a dealbreaker.

A server has no human staring at it, so most of what makes a laptop "good" does not matter here. A broken screen, a dead trackpad, missing keys, ugly stickers: none of it counts. Once it is running, you control it from another computer and the laptop lives in a closet with the lid shut.

The only specs that matter

PART	AIM FOR	NOTES
RAM	8GB	Enough to run several services at once. Do not buy more until something actually runs out. 4GB still works for light jobs.
Storage	SSD if possible	The single biggest speed upgrade for old hardware. A 256GB SSD is cheap and transforms a sluggish laptop.
CPU	Any Intel i3/i5 from the last decade	Server work is mostly waiting, not crunching. Almost anything qualifies.
Ethernet port	Nice to have	A wired connection is more reliable than Wi-Fi for an always-on box. Wi-Fi still works fine.

FIELD NOTE

The best server in your house is the laptop you stopped using because the screen cracked or the battery got weak. It still boots, it still networks, and you were never going to look at that screen again anyway. That is a free server hiding as junk.

One quick upgrade worth doing

If the laptop still has a spinning hard drive, swapping in a cheap SSD is the one upgrade that pays for itself in speed. It is usually a few screws on the bottom panel and a five-minute job. If it already has an SSD, skip it and move on.

BATTERY WARNING

If the battery is visibly swollen or puffy, remove it before you do anything else and run the laptop on the wall charger only. A swollen lithium battery is a fire risk, especially in a closet

> running 24/7. A laptop with no battery still works as a server; it just loses its built-in backup power.

CHAPTER 02

Wipe It and Prep the Hardware

Clean slate, then two BIOS settings that make it behave like a real server.

Step 1: Rescue anything you care about

You are about to erase this laptop completely. Copy off any photos, documents, or files first. Once the install starts, everything on the drive is gone.

Step 2: Two BIOS settings that matter

Turn the laptop on and tap the BIOS key during boot (usually `F2`, `F10`, `Del`, or `Esc`, the screen often tells you). Look for two things:

- **Restore on AC Power Loss → set to "Power On."** This means if the power goes out and comes back, the laptop turns itself back on automatically instead of sitting there dark. Essential for a server you are not standing next to.
- **Boot Order → put USB first.** So you can boot from the install USB you are about to make.

> FIELD NOTE
>
> Not every old laptop BIOS has the "Restore on AC Power Loss" option. If yours doesn't, no problem, the built-in battery already covers short outages, which is exactly the advantage a laptop has over a desktop or a Pi.

Save and exit. Now we make the install drive.

CHAPTER 03

Install the Server Brain (Ubuntu)

Ubuntu Server is free, rock-solid, and runs lean because it has no desktop to slow it down.

We use **Ubuntu Server** because it is the most widely supported, best-documented free server operating system on the planet. Any problem you ever hit, someone has already solved and posted the answer. It runs without a graphical desktop, which means more of your laptop's power goes to actual work.

Step 1: Make a bootable USB

1. On any working computer, download the **Ubuntu Server LTS** image from ubuntu.com (the "LTS" version is the long-term, stable one).
2. Download **balenaEtcher** (free, works on Mac, Windows, Linux).
3. Plug in any USB stick 4GB or larger. Open Etcher, select the Ubuntu file, select your USB stick, click Flash. Wait a few minutes.

Step 2: Boot the laptop from the USB

Plug the USB into the old laptop and power it on. Because you set USB first in the boot order, it boots into the Ubuntu installer.

Step 3: Walk through the installer

The installer asks simple questions. The defaults are almost all correct. The choices that matter:

PROMPT	WHAT TO DO
`Storage`	Choose "use entire disk." This wipes the laptop drive and gives the whole thing to the server.
`Server name`	Pick anything. It is how the server shows up on your network, e.g. `engine-room-01`.
`Username + password`	Choose a username and a strong password. Write them down. You log in with these.
`Install OpenSSH server`	**Check this box.** It lets you control the server remotely. We cover it in Chapter 05. Saves you a step.

Let it finish, remove the USB when prompted, and reboot. You now have a server. It boots to a black screen with a login prompt, that is correct and exactly what you want.

> **YOU JUST DID THE HARD PART**
>
> That black login screen is a real Linux server, the same kind that runs a huge chunk of the internet. From here on it only gets easier.

CHAPTER 04

The Closed-Lid Trick

The one setting that separates a laptop-server pro from everyone else.

By default, when you close a laptop lid it goes to sleep. That is a disaster for a server, you want to close the lid, tuck it in a closet, and have it keep running forever. One config change fixes this permanently.

Log into the laptop (or SSH in, see next chapter) and open the power-management config:

```
# open the login manager config in a text editor
sudo nano /etc/systemd/logind.conf
```

Find these lines, remove the # in front of them, and set them to `ignore`:

```
HandleLidSwitch=ignore
HandleLidSwitchExternalPower=ignore
HandleLidSwitchDocked=ignore
```

Save (in nano: `Ctrl+O`, `Enter`, then `Ctrl+X`). Then apply it:

```
sudo systemctl restart systemd-logind
```

THAT'S IT

Close the lid. The laptop keeps running. You can now stand it on its edge, slide it onto a shelf, or stack it with others. The Engine Room takes up almost no space.

FIELD NOTE

Heat is the enemy of a closed laptop running 24/7. Leave a little air gap around it, keep it off carpet and out of direct sun, and blow the dust out of the vents every few months. A cool server is a server that lasts years.

CHAPTER 05

Control It From Another Room

Once SSH is on, you never touch the laptop again. You run it from your main computer.

SSH (Secure Shell) lets you type commands into the server from any other computer on your network. This is how every professional manages servers, not by sitting in front of them.

Step 1: Make sure SSH is running

If you checked "Install OpenSSH server" during setup, it is already on. If not, log into the laptop directly and run:

```
sudo apt update
sudo apt install -y openssh-server
sudo systemctl enable --now ssh
```

Step 2: Find the server's address

On the laptop, run:

```
ip a
```

Look for an address that starts with `192.168.` or `10.`, that is your server's address on your home network. Write it down, for example `192.168.1.42`.

Step 3: Connect from your main computer

On your Mac or Windows machine, open Terminal (Mac) or PowerShell (Windows) and type:

```
ssh yourusername@192.168.1.42
```

Enter your password and you are in. You are now controlling the server from your couch. Close the laptop, put it away, and forget about it.

> FIELD NOTE
>
> Set a "DHCP reservation" in your home router so the server always gets the same address. Otherwise the address can change after a reboot and you have to look it up again. Search your router brand plus "DHCP reservation" for the two-minute steps.

CHAPTER 06

One-Click Apps with CasaOS

A friendly dashboard that turns "Linux commands" into "click to install."

Everything so far has been the command line. Now we put a clean web dashboard on top so installing apps is as easy as an app store. That tool is **CasaOS**, and it is free.

Install it with one line

SSH into your server and run:

```
curl -fsSL https://get.casaos.io | sudo bash
```

When it finishes, it prints a web address (your server's IP). Open that in any browser on your network. You get a clean dashboard where you can:

- See your server's health: CPU, memory, storage, all at a glance.
- Install apps from an app store with one click, media servers, file storage, automation tools, and more.
- Manage files through a web interface instead of commands.

WHY THIS MATTERS

CasaOS is the moment the Engine Room stops feeling technical. From here, anyone in the family can look at the dashboard and understand what is running. That is how a one-person setup becomes a family-run machine.

FIELD NOTE

CasaOS installs Docker underneath, which is the standard way professionals package and run apps in clean, isolated containers. You do not need to learn Docker to use CasaOS, but know that it is there doing the heavy lifting.

CHAPTER 07

Reach It From Anywhere, Safely

Access your Engine Room from your phone across town, without opening your home network to the world.

So far the server only works inside your house. To reach it from anywhere, the wrong way is to "port forward," which pokes a hole straight through your router and invites the entire internet to knock. Do not do that. There are two safe, free ways instead.

Option A: Tailscale (private, easiest, recommended)

Tailscale builds a private encrypted network between your devices, as if they were all in the same room, no matter where they are. Install on the server:

```
curl -fsSL https://tailscale.com/install.sh | sh
sudo tailscale up
```

Install the Tailscale app on your phone and laptop too, sign in with the same account, and every device can now reach the server privately. Nobody else can see it. This is the safest option and the one I use.

Option B: Cloudflare Tunnel or playit.gg (for public sites)

If you are hosting a website or service that the public needs to reach (a real visitor-facing site), a tunnel service like **Cloudflare Tunnel** or **playit.gg** exposes just that one service to the internet through their secure pipe, without opening your home network. Use this only for things meant to be public.

> SECURITY RULE
>
> Private stuff (files, automations, dashboards) goes through Tailscale. Only genuinely public things (a customer-facing website) get a public tunnel. Keep that line clean and you stay safe.

CHAPTER 08

What to Actually Run

The server is built. Here is what to put in it to make it earn its keep.

An empty server is just a warm laptop. Here is the lineup that turns it into a real Engine Room, roughly in order of usefulness.

RUN THIS	WHAT IT DOES FOR YOU
`n8n`	A free, self-hosted automation tool. Connect apps, move data, send messages, run workflows on a schedule. This is the workhorse of the Engine Room, the thing that does jobs for you while you sleep.
`A website`	Host your own sites directly. No monthly hosting bill, full control, deploy as many as your laptop can handle.
`Nextcloud`	Your own private Google Drive. Files, photos, calendar, contacts, synced across all your devices, stored on hardware you own.
`Jellyfin`	Your own private Netflix. Stream your movies and music to any screen in the house.
`Uptime Kuma`	Watches your other services and pings your phone if anything goes down. A night watchman for the Engine Room.
`A backup job`	Automatic scheduled backups of everything important, so a dead drive never costs you data.

THE REAL PLAY

The single highest-value thing on this list is **n8n**. A self-hosted automation engine running on free hardware means every repetitive task in your life or business can be handed to a machine that costs you nothing per month to run. That is the difference between owning tools and renting them.

CHAPTER 09

Build a Fleet

One laptop is a server. Several laptops, each with a job, is infrastructure you own outright.

Once you have built one, building the next is an afternoon. The smart move is not one overloaded laptop, it is a few laptops, each with a clear role, so if one dies the others keep going.

A simple way to split the jobs

MACHINE	JOB
`engine-room-01`	Automations (n8n) and your dashboards. The brain.
`engine-room-02`	Websites and public-facing services. The storefront.
`engine-room-03`	Files, backups, and media (Nextcloud, Jellyfin). The vault.

Give each one the same setup from Chapters 03 to 07, a clear name, and a single main job. Now you have a small data center, assembled from hardware that was headed for a landfill, running on power that costs a few dollars a month total.

> FIELD NOTE
>
> This is where a family or a crew becomes powerful. Everyone has an old laptop somewhere. Pool five of them into one room and you have built infrastructure that small businesses pay hundreds a month to rent, except you own it, and nobody can shut it off or raise the price.

CHAPTER 10

Keep It Alive

Five habits that keep the Engine Room secure and running for years.

---

1. Lock the front door (SSH keys)

Passwords can be guessed. SSH keys cannot, practically speaking. Once comfortable, switch from password login to key-based login. Search "Ubuntu SSH key setup" for the steps, it is a one-time, ten-minute upgrade that makes your server dramatically harder to break into.

2. Turn on the firewall

```
sudo ufw allow OpenSSH
sudo ufw enable
```

This blocks everything except what you explicitly allow. Add a line for each service you intentionally run.

3. Keep it patched automatically

```
sudo apt update && sudo apt upgrade -y
sudo apt install -y unattended-upgrades
```

The second command sets up automatic security updates so the server patches itself.

4. Back up before you need to

A backup you set up today is the one that saves you in six months. Automate it (Chapter 08), and occasionally test that you can actually restore from it.

5. Watch the heat and dust

Every few months, blow out the vents and confirm the machine is cool to the touch. Dust is what kills closed laptops over time. Two minutes of maintenance buys you years of uptime.

> THE ONE RULE
>
> If it is private, never expose it directly to the internet. Reach it through Tailscale (Chapter 07). Most home-server disasters come from skipping this one rule.

APPENDIX

The Command Cheat Sheet

Tape this next to the Engine Room.

```
# --- CONNECT ---
ssh user@192.168.1.42         # log into the server
ip a                          # find the server's address

# --- KEEP LID CLOSED ---
sudo nano /etc/systemd/logind.conf
# set HandleLidSwitch=ignore, then:
sudo systemctl restart systemd-logind

# --- INSTALL THE DASHBOARD ---
curl -fsSL https://get.casaos.io | sudo bash

# --- REMOTE ACCESS (PRIVATE) ---
curl -fsSL https://tailscale.com/install.sh | sh
sudo tailscale up

# --- SECURITY ---
sudo ufw allow OpenSSH
sudo ufw enable
sudo apt update && sudo apt upgrade -y

# --- HEALTH CHECK ---
htop                          # live CPU/memory (sudo apt install htop)
df -h                         # disk space used/free
sudo systemctl status ssh     # is SSH running?
```

THE BIGGER PLAY

You don't rent your power anymore.

You just built infrastructure out of hardware the world threw away. Websites, automations, storage, all running on machines you own, for a few dollars of electricity a month. That is the whole idea behind The Grind Protocol: stop renting the tools that run your life and start owning the machine.

WHAT'S NEXT

This manual gets the Engine Room running. The next step is what you **run inside it**, the automations and systems that actually make money while you sleep. That is what we build at TGP.

More builds, templates, and systems:

rafaelhunna.com

RAFAEL HUNNA // The Grind Protocol