SELF-HOSTING

The Self-Hosting Survival Guide

Escape SaaS subscriptions: set up your own cloud with Nextcloud, Pi-hole, and more

by Rook

First Edition · February 2026

Table of Contents

The Self-Hosting Survival Guide

Stop renting your digital life. Own it.

CHAPTER 01

Why Self-Host?

▸ The Maths Don’t Lie

Let’s add up what a typical developer pays monthly for cloud services:

SERVICE	MONTHLY COST
Google One (200GB)	£2.49
Notion (Plus)	£7.50
Slack (Pro)	£5.75
LastPass Premium	£2.50
Spotify Premium	£10.99
Google Analytics 360	Free tier, but your data is the product
Email (Google Workspace)	£4.14
Photo storage (Google/iCloud)	£2.49
Total	**~£36/month → £432/year**

A decent mini PC costs £150. A domain costs £10/year. Electricity runs about £3/month. That's £196 for your first year — and under £50/year after that. You break even in six months, then you're saving roughly £380 every year thereafter.

But it's not just about the money.

▸ Data Ownership

Every photo you upload to Google Photos trains their AI models. Every note in Notion lives on someone else's server, subject to their terms of service, their outages, and their pricing whims. When you self-host, your data lives on hardware you control. No one can pull the rug.

▸ The Learning Opportunity

Self-hosting will teach you more about networking, Linux, security, and infrastructure than any online course. It's hands-on, it's real, and it's the kind of knowledge that makes you genuinely better at your job.

▸ When NOT to Self-Host

Let's be honest. Don't self-host if:

- **You need five-nines uptime for a business** — pay for managed services.
- **Email is mission-critical** — self-hosted email is a nightmare (more on that later).

- **You won’t maintain it** — an unpatched server is worse than no server.
- **You’re doing it to avoid paying £3/month** — your time has value.

Self-hosting is brilliant when you treat it as a hobby that saves money. It’s a disaster when you treat it as a chore you resent.

CHAPTER 02

Hardware Options

You don’t need a rack server. You need something that runs Linux and doesn’t cost much.

▸ Repurpose an Old Laptop or Desktop

That ThinkPad gathering dust in your drawer? Perfect. Any machine from the last 8–10 years with 4GB+ RAM and an SSD can run a dozen Docker containers comfortably. Laptops have a bonus: a built-in UPS (the battery). Downsides: power consumption is higher than purpose-built options (typically 30–60W idle).

▸ Raspberry Pi

The Pi 5 (8GB) costs around £75 and sips about 5W. It handles lightweight services beautifully — Vaultwarden, Uptime Kuma, Pi-hole, Home Assistant, and a reverse proxy will all run fine.

Where it struggles: anything CPU-intensive. Nextcloud is usable but sluggish with multiple users. Immich’s machine learning features will crawl. Photo and video transcoding is essentially a non-starter.

Verdict: Great for your first few services. You’ll outgrow it.

▸ Mini PCs (Best Bang for Buck)

This is the sweet spot. A second-hand or refurbished mini PC gives you proper x86 performance at low power draw. Look for:

- **Lenovo ThinkCentre M720q/M920q** — Intel 8th/9th gen, widely available refurbished for £80–£130
- **Dell OptiPlex Micro 3060/5060** — same era, same price bracket
- **Beelink/MinisForum** — new, AMD Ryzen options from £150–£250

Aim for: quad-core CPU, 16GB RAM (upgrade yourself — it’s cheaper), 256GB+ SSD. This setup idles at 8–15W and handles everything in this guide without breaking a sweat.

▸ VPS Providers (Ranked)

If you'd rather not have hardware at home — or you want a publicly accessible server without fiddling with port forwarding — a VPS is the way.

1. **Hetzner** — The gold standard. €4.51/month gets you 2 vCPUs, 4GB RAM, 40GB SSD in Falkenstein or Helsinki. Unbeatable price-to-performance.
2. **Oracle Cloud Free Tier** — Genuinely free. 4 ARM Ampere cores, 24GB RAM. The catch: Oracle's UI is awful, provisioning sometimes fails, and there's always a nagging worry they'll pull the free tier.
3. **Contabo** — Cheap specs on paper (8GB RAM for €6/month), but storage is HDD-based and network performance is inconsistent. Fine for non-critical services.
4. **Vultr/DigitalOcean** — More expensive ($6–$12/month for comparable specs) but reliable with excellent documentation. Good if you value polish.

My recommendation: Hetzner for a VPS, or a refurbished mini PC at home if you have a stable internet connection. Choose one and start.

CHAPTER 03

The Foundation Stack

Before you install anything fun, you need a solid foundation. Every service in this guide builds on these five pillars.

▸ Linux: Ubuntu Server 24.04 LTS

I'm recommending Ubuntu Server because it has the largest community, the most tutorials, and the widest compatibility. Debian purists: you'll be fine with Debian 12, the commands are nearly identical.

Flash the ISO to a USB drive with Balena Etcher or Rufus, boot from it, and follow the installer. Choose "minimised" when asked — you don't need a desktop environment.

After installation, SSH in and run:

```
sudo apt update && sudo apt upgrade -y
sudo apt install -y curl git htop
```

That's your base. Everything else runs in containers.

▸ Docker & Docker Compose

Docker is the single most important tool in self-hosting. It lets you run isolated services without dependency conflicts, and tear them down cleanly when you're done.

Install Docker (the official way — don't use Ubuntu's ancient snap package):

```
curl -fsSL https://get.docker.com | sh
sudo usermod -aG docker $USER
```

Log out and back in. Verify:

```
docker run hello-world
docker compose version
```

Docker Compose (now built into Docker as `docker compose`) lets you define multi-container setups in a single YAML file. This is how you'll deploy every service in this guide.

A basic `docker-compose.yml` structure:

```
services:
  myapp:
    image: someimage:latest
    container_name: myapp
    restart: unless-stopped
    ports:
      - "8080:80"
    volumes:
      - ./data:/app/data
    environment:
      - TZ=Europe/London
```

Key concepts: - `restart: unless-stopped` — survives reboots, but stays down if you manually stop it - **Volumes** — persist data outside the container. Without them, your data vanishes when the container updates - **Environment variables** — configure the app without modifying files inside the container

▸ Reverse Proxy: Caddy

You'll run multiple services, each on a different port. A reverse proxy maps `nextcloud.yourdomain.com` to `localhost:8080`, `photos.yourdomain.com` to `localhost:2283`, and so on.

Use Caddy. It's simpler than Nginx, and handles SSL certificates automatically via Let's Encrypt with zero configuration.

```
services:
  caddy:
    image: caddy:2
    container_name: caddy
    restart: unless-stopped
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./Caddyfile:/etc/caddy/Caddyfile
      - caddy_data:/data
      - caddy_config:/config

volumes:
  caddy_data:
  caddy_config:
```

Your `Caddyfile`:

```
nextcloud.yourdomain.com {
    reverse_proxy nextcloud:80
}

photos.yourdomain.com {
    reverse_proxy immich-server:2283
}
```

That's it. Caddy fetches and renews SSL certificates automatically. No `certbot` cron jobs, no Nginx config nightmares.

Alternative: If you prefer a GUI, Nginx Proxy Manager provides a web interface for the same job. It works, but I find the Caddyfile approach faster once you're comfortable with it.

▸ SSL with Let's Encrypt

If you're using Caddy, you already have this. Point your domain's DNS A record at your server's IP address, and Caddy handles the rest.

For home servers behind a router: you'll need to forward ports 80 and 443 to your server's local IP. If your ISP uses CGNAT (common with some providers), you'll need either a VPS with a tunnel (Cloudflare Tunnel works well and is free) or a VPN like Tailscale to access your services remotely.

▸ Backups: The 3-2-1 Rule

3 copies of your data, on 2 different media types, with 1 offsite. This isn't optional. Self-hosting without backups is just data loss waiting to happen.

Restic is my pick for most people. It's fast, supports encryption, deduplication, and can push to local drives, SFTP, S3, or Backblaze B2.

```
# Install
sudo apt install -y restic

# Initialise a repository (e.g., to Backblaze B2)
export B2_ACCOUNT_ID="your-id"
export B2_ACCOUNT_KEY="your-key"
restic -r b2:your-bucket-name init

# Back up your Docker volumes
restic -r b2:your-bucket-name backup /home/user/docker/

# Automate with cron
# 0 3 * * * restic -r b2:your-bucket-name backup /home/user/docker/ --tag nightly
```

BorgBackup is the alternative — slightly faster for large datasets, but less flexible with remote backends.

Backblaze B2 charges $0.005/GB/month. For 100GB of backups, that's $0.50/month. There's no excuse.

CHAPTER 04

Service-by-Service Replacement Guide

Here's the meat of it. For each service: what it replaces, a working Docker Compose snippet, a difficulty rating, and the gotchas nobody tells you about.

All snippets assume you're placing each service in its own directory with its own `docker-compose.yml`.

▸ Google Drive → Nextcloud

Difficulty: ▯▯ Moderate **Replaces:** Google Drive, Dropbox, OneDrive

```
services:
  nextcloud:
    image: nextcloud:29
    container_name: nextcloud
    restart: unless-stopped
    volumes:
      - ./data:/var/www/html
    environment:
      - POSTGRES_HOST=nextcloud-db
      - POSTGRES_DB=nextcloud
      - POSTGRES_USER=nextcloud
      - POSTGRES_PASSWORD=changeme_use_a_real_password
    depends_on:
      - nextcloud-db

  nextcloud-db:
    image: postgres:16-alpine
    container_name: nextcloud-db
    restart: unless-stopped
    volumes:
      - ./db:/var/lib/postgresql/data
    environment:
      - POSTGRES_DB=nextcloud
      - POSTGRES_USER=nextcloud
      - POSTGRES_PASSWORD=changeme_use_a_real_password
```

Gotchas: Use PostgreSQL, not SQLite (the default) — SQLite will buckle under any real usage. Mobile and desktop sync clients are solid. The web UI feels dated compared to Google Drive but it's functional. Enable server-side encryption if you're storing sensitive files. Run `occ` maintenance commands periodically via cron.

▸ Gmail → Stalwart Mail

Difficulty: ▯▯▯▯▯ Expert — Proceed with caution **Replaces:** Gmail, Outlook, any hosted email

Let me be brutally honest: **self-hosting email is the hardest thing in this guide.** Your IP will probably be blacklisted by default. SPF, DKIM, and DMARC records must be perfect. Major providers (Gmail, Outlook) will silently drop your emails into spam for weeks until you build reputation. If email deliverability matters to you, use a paid service like Fastmail (£4/month) or Migadu.

Still want to try? **Stalwart Mail** is the modern choice — it's a single Rust binary that handles SMTP, IMAP, and JMAP with built-in spam filtering.

```
services:
  stalwart:
    image: stalwartlabs/mail-server:latest
    container_name: stalwart
    restart: unless-stopped
    ports:
      - "25:25"
      - "465:465"
      - "993:993"
      - "4190:4190"
    volumes:
      - ./data:/opt/stalwart-mail
```

Gotchas: You absolutely need a VPS with a clean IP and a proper rDNS (reverse DNS) record. Most home ISPs block port 25. Set up SPF, DKIM, and DMARC DNS records before sending a single email. Test with mail-tester.com. Expect a month of babysitting before it works reliably.

▸ Google Photos → Immich

Difficulty: ▯▯ Moderate **Replaces:** Google Photos, iCloud Photos

Immich is the standout self-hosted project of the past two years. It looks and feels like Google Photos — mobile auto-upload, facial recognition, maps, memories, the lot.

```
services:
  immich-server:
    image: ghcr.io/immich-app/immich-server:release
    container_name: immich-server
    restart: unless-stopped
    volumes:
      - ./upload:/usr/src/app/upload
    environment:
      - DB_HOSTNAME=immich-db
      - DB_USERNAME=postgres
      - DB_PASSWORD=changeme
      - DB_DATABASE_NAME=immich
      - REDIS_HOSTNAME=immich-redis
    depends_on:
      - immich-db
      - immich-redis

  immich-machine-learning:
    image: ghcr.io/immich-app/immich-machine-learning:release
    container_name: immich-ml
    restart: unless-stopped
    volumes:
      - ./model-cache:/cache

  immich-redis:
    image: redis:7-alpine
    container_name: immich-redis
    restart: unless-stopped

  immich-db:
    image: tensorchord/pgvecto-rs:pg16-v0.2.0
    container_name: immich-db
    restart: unless-stopped
    volumes:
      - ./db:/var/lib/postgresql/data
    environment:
      - POSTGRES_PASSWORD=changeme
      - POSTGRES_USER=postgres
      - POSTGRES_DB=immich
```

Gotchas: The machine learning container wants 2–4GB of RAM for facial recognition. First-time indexing of a large photo library will peg your CPU for hours — let it run overnight. Mobile apps (iOS/Android) are excellent. The project is pre-1.0 and breaking changes happen, so read release notes before updating.

▶ Notion → Outline

Difficulty: □□□ Moderate-Hard **Replaces:** Notion, Confluence, Google Docs (for wikis/notes)

```
services:
  outline:
    image: docker.getoutline.com/outlinewiki/outline:latest
    container_name: outline
    restart: unless-stopped
    environment:
      - DATABASE_URL=postgres://outline:changeme@outline-db:5432/outline
      - REDIS_URL=redis://outline-redis:6379
      - URL=https://wiki.yourdomain.com
      - SECRET_KEY=generate-a-64-char-hex-string
      - UTILS_SECRET=generate-another-64-char-hex-string
      - FILE_STORAGE=local
    depends_on:
      - outline-db
      - outline-redis

  outline-db:
    image: postgres:16-alpine
    container_name: outline-db
    restart: unless-stopped
    volumes:
      - ./db:/var/lib/postgresql/data
    environment:
      - POSTGRES_USER=outline
      - POSTGRES_PASSWORD=changeme
      - POSTGRES_DB=outline

  outline-redis:
    image: redis:7-alpine
    container_name: outline-redis
    restart: unless-stopped
```

Gotchas: Outline requires an authentication provider (OIDC). You'll need something like Authentik or Keycloak, which adds complexity. **AppFlowy** is a simpler alternative with a local-first approach if you don't need multi-user collaboration.

▶ Slack → Matrix/Element

Difficulty: ▯▯▯ Moderate-Hard **Replaces:** Slack, Discord, Teams

Matrix is the protocol; Element is the client; Synapse is the server. It's federated, meaning you can chat with anyone on any Matrix server.

```
services:
  synapse:
    image: matrixdotorg/synapse:latest
    container_name: synapse
    restart: unless-stopped
    volumes:
      - ./data:/data
    environment:
      - SYNAPSE_SERVER_NAME=yourdomain.com
      - SYNAPSE_REPORT_STATS=no
```

Gotchas: Generate the config first with `docker run --rm -v ./data:/data matrixdotorg/synapse:latest generate`. Synapse (Python) is resource-hungry — consider **Conduit** or **Dendrite** (both written in Rust/Go) for lighter-weight alternatives. Federation is brilliant but burns RAM. For a private family/team server, disable federation to save resources.

▶ LastPass → Vaultwarden

Difficulty: ▯ Easy **Replaces:** LastPass, 1Password, Bitwarden

The easiest win on this list. Vaultwarden is an unofficial Bitwarden-compatible server that uses a fraction of the resources.

```
services:
  vaultwarden:
    image: vaultwarden/server:latest
    container_name: vaultwarden
    restart: unless-stopped
    volumes:
      - ./data:/data
    environment:
      - DOMAIN=https://vault.yourdomain.com
      - SIGNUPS_ALLOWED=false
```

Gotchas: Set `SIGNUPS_ALLOWED=true` initially to create your account, then switch to `false`. Use the official Bitwarden browser extensions and mobile apps — they work perfectly with Vaultwarden. **Back this up religiously.** If you lose your vault, you lose everything.

▶ Google Analytics → Umami

Difficulty: ▯ Easy **Replaces:** Google Analytics, Fathom, Plausible

```
services:
  umami:
    image: ghcr.io/umami-software/umami:postgresql-latest
    container_name: umami
    restart: unless-stopped
    environment:
      - DATABASE_URL=postgresql://umami:changeme@umami-db:5432/umami
    depends_on:
      - umami-db

  umami-db:
    image: postgres:16-alpine
    container_name: umami-db
    restart: unless-stopped
    volumes:
      - ./db:/var/lib/postgresql/data
    environment:
      - POSTGRES_DB=umami
      - POSTGRES_USER=umami
      - POSTGRES_PASSWORD=changeme
```

Gotchas: Umami is privacy-focused and GDPR-compliant out of the box — no cookie banner needed. Default login is `admin` / `umami` — change it immediately. **Plausible** is the alternative; it's polished but its self-hosted version requires more resources.

▸ Spotify → Navidrome

Difficulty: ▯ Easy **Replaces:** Spotify, Apple Music (for your own music library)

```
services:
  navidrome:
    image: deluan/navidrome:latest
    container_name: navidrome
    restart: unless-stopped
    ports:
      - "4533:4533"
    volumes:
      - ./data:/data
      - /path/to/your/music:/music:ro
    environment:
      - ND_SCANSCHEDULE=1h
      - ND_LOGLEVEL=info
```

Gotchas: You need your own music files (FLAC, MP3, etc.). Navidrome supports the Subsonic API, so you can use apps like Symfonium (Android) or play:Sub (iOS). It won't replace Spotify's discovery features — it replaces Spotify's playback of music you already own.

▸ YouTube Subscriptions → Invidious + FreeTube

Difficulty: ▯▯ Moderate **Replaces:** YouTube (frontend)

Invidious is a privacy-respecting YouTube frontend. FreeTube is the desktop client.

```
services:
  invidious:
    image: quay.io/invidious/invidious:latest
    container_name: invidious
    restart: unless-stopped
    environment:
      - INVIDIOUS_CONFIG=channel_threads:1,feed_threads:1,db__dbname:invidious,db_
    depends_on:
      - invidious-db

  invidious-db:
    image: postgres:16-alpine
    container_name: invidious-db
    restart: unless-stopped
    volumes:
      - ./db:/var/lib/postgresql/data
    environment:
      - POSTGRES_DB=invidious
      - POSTGRES_USER=kemal
      - POSTGRES_PASSWORD=changeme
```

Gotchas: YouTube actively breaks Invidious. Expect occasional downtime while maintainers patch around API changes. Import your YouTube subscriptions via OPML. For desktop viewing, **FreeTube** is excellent and doesn't need a server — it connects directly to YouTube or your Invidious instance.

▸ Smart Home → Home Assistant

Difficulty: □□ Moderate **Replaces:** Google Home, Alexa, SmartThings

```
services:
  homeassistant:
    image: ghcr.io/home-assistant/home-assistant:stable
    container_name: homeassistant
    restart: unless-stopped
    network_mode: host
    volumes:
      - ./config:/config
    environment:
      - TZ=Europe/London
```

Gotchas: `network_mode: host` is needed for device discovery (mDNS, UPnP). Home Assistant has a steep initial learning curve but an incredibly active community. The companion mobile app provides location tracking and notifications. For Zigbee/Z-Wave devices, you'll need a USB coordinator (Sonoff Zigbee 3.0 dongle is about £12).

CHAPTER 05

Security Hardening

A server exposed to the internet without security hardening will be compromised. It's not a question of if — it's when. These steps are mandatory.

▸ Firewall (UFW)

```
sudo apt install -y ufw
sudo ufw default deny incoming
sudo ufw default allow outgoing
sudo ufw allow 22/tcp    # SSH
sudo ufw allow 80/tcp    # HTTP
sudo ufw allow 443/tcp   # HTTPS
sudo ufw enable
```

That's it. Only open ports you actually need. If you add a service on port 8080 that's behind your reverse proxy, you don't need to open 8080 — traffic flows through Caddy on 443.

▸ Fail2ban

Fail2ban watches log files and bans IPs that show malicious behaviour (brute-force SSH attempts, repeated failed logins).

```
sudo apt install -y fail2ban
sudo systemctl enable fail2ban
sudo systemctl start fail2ban
```

The default config protects SSH out of the box. For additional services, add custom jail configurations in `/etc/fail2ban/jail.local`.

▸ SSH Key-Only Access

Password authentication over SSH is an open invitation. Disable it.

```
# On your local machine, generate a key (if you haven't)
ssh-keygen -t ed25519

# Copy it to your server
ssh-copy-id user@your-server-ip

# On the server, edit SSH config
sudo nano /etc/ssh/sshd_config
```

Set these values:

```
PasswordAuthentication no
PubkeyAuthentication yes
PermitRootLogin no
```

Then restart SSH:

```
sudo systemctl restart sshd
```

Test in a new terminal before closing your current session. If you lock yourself out, you'll need physical access or a VPS console.

▸ Automatic Security Updates

```
sudo apt install -y unattended-upgrades
sudo dpkg-reconfigure -plow unattended-upgrades
```

This automatically installs security patches. For Docker containers, you'll manage updates separately (see Maintenance section).

▸ VPN Access with WireGuard

Rather than exposing every service to the internet, put them behind WireGuard. Only your VPN-connected devices can reach them.

```
sudo apt install -y wireguard
```

Or use **Tailscale** — it's WireGuard under the hood but handles NAT traversal and key management automatically. Free for up to 100 devices. Genuinely brilliant for home servers.

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

Now your server is accessible via a Tailscale IP (e.g., `100.x.y.z`) from any device on your Tailnet. No port forwarding, no dynamic DNS, no exposed ports.

▸ Monitoring with Uptime Kuma

Know when things break before your users do.

```
services:
  uptime-kuma:
    image: louislam/uptime-kuma:1
    container_name: uptime-kuma
    restart: unless-stopped
    volumes:
      - ./data:/app/data
    ports:
      - "3001:3001"
```

Add monitors for each service. Set up notifications via Telegram, Discord, or email. Uptime Kuma is lightweight and gorgeous — it's the one monitoring tool everyone should run.

CHAPTER 06

Maintenance & Monitoring

Self-hosting isn't "set it and forget it." Budget 30 minutes a week for maintenance. Here's how to keep things running.

▸ Update Strategy

OS updates: Handled by `unattended-upgrades` for security patches. Run `sudo apt update && sudo apt upgrade` manually every couple of weeks for non-security updates.

Docker containers: Use Watchtower for automatic updates, or update manually:

```
docker compose pull
docker compose up -d
```

My recommendation: Don't auto-update everything. Use Watchtower selectively for low-risk services (Uptime Kuma, Vaultwarden). For complex services like Nextcloud or Immich, read release notes and update manually. Breaking changes are real.

▸ Monitoring

For most home setups, **Uptime Kuma** plus Docker's built-in tools is sufficient:

```
# Check running containers
docker ps

# View logs for a specific container
docker logs nextcloud --tail 50

# Resource usage
docker stats
```

If you want proper metrics, **Grafana + Prometheus** is the industry standard, but it's overkill for a home server. Consider **Dozzle** for a simple real-time Docker log viewer:

```
services:
  dozzle:
    image: amir20/dozzle:latest
    container_name: dozzle
    restart: unless-stopped
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock:ro
    ports:
      - "9999:8080"
```

▸ When Things Break

Docker containers are disposable. That's the beauty. Your debugging workflow:

1. **Check logs:** `docker logs <container> --tail 100`
2. **Check if it's running:** `docker ps -a` (shows stopped containers too)
3. **Restart it:** `docker compose restart <service>`
4. **Nuclear option:** `docker compose down && docker compose up -d`
5. **Check disk space:** `df -h` — full disks are the #1 silent killer
6. **Check RAM:** `free -h` — OOM kills are the #2 silent killer

If a container won't start after an update, pin the previous version in your `docker-compose.yml` (e.g., `image: nextcloud:28` instead of `nextcloud:latest`) and wait for a fix.

▸ Community Resources

You're not alone. These communities are active, helpful, and full of people solving the exact problems you'll encounter:

- **r/selfhosted** — The main hub. Search before posting; your question has probably been answered.
- **r/homelab** — More hardware-focused, great for build inspiration.
- selfh.st — Weekly newsletter covering new self-hosted tools.
- awesome-selfhosted — The canonical list of self-hosted software.
- Matrix/Discord servers for specific projects (Immich, Home Assistant, etc.)

CHAPTER 07

Appendix

▸ Monthly Cost Comparison: SaaS vs Self-Hosted

SERVICE	SAAS MONTHLY	SELF-HOSTED MONTHLY
Cloud Storage (200GB)	£2.49 (Google One)	£0 (included)
Email	£4.14 (Google Workspace)	£0*
Photo Backup	£2.49 (Google One / iCloud)	£0 (included)
Notes/Wiki	£7.50 (Notion Plus)	£0 (included)
Team Chat	£5.75 (Slack Pro)	£0 (included)
Password Manager	£2.50 (LastPass)	£0 (included)
Analytics	“Free” (you’re the product)	£0 (included)
Music Streaming	£10.99 (Spotify)	£0 (BYO music)
Smart Home Hub	£0–£10 (subscriptions)	£0 (included)
Monitoring	£0–£30	£0 (included)
Subtotal (services)	**~£36–£46/month**	**£0**
Hardware (amortised)	—	~£6/month (£150 over 24 months)
Electricity	—	~£3/month
Domain	—	~£0.80/month (£10/year)
Offsite Backup (B2)	—	~£0.40/month
Total	**~£36–£46/month**	**~£10/month**
Annual	**£432–£552**	**~£120 (year 1), ~£50 thereafter**

*Email self-hosting is free in hosting costs but expensive in time and frustration. Honestly, just pay for Fastmail.

▶ Docker Compose Cheat Sheet

```
# Start services in background
docker compose up -d

# Stop services
docker compose down

# View logs (follow mode)
docker compose logs -f

# View logs for one service
docker compose logs -f nextcloud

# Restart a service
docker compose restart nextcloud

# Pull latest images
docker compose pull

# Pull and recreate
docker compose pull && docker compose up -d

# List running containers
docker ps

# Enter a container's shell
docker exec -it nextcloud bash

# Check resource usage
docker stats

# Clean up unused images/volumes
docker system prune -a
docker volume prune

# See all volumes
docker volume ls

# Back up a volume to tar
docker run --rm -v myvolume:/data -v $(pwd):/backup alpine tar czf /backup/myvolum
```

▸ **Recommended Communities**

COMMUNITY	PLATFORM	FOCUS
r/selfhosted	Reddit	General self-hosting
r/homelab	Reddit	Hardware & infrastructure
r/homeassistant	Reddit	Smart home
selfh.st	Newsletter	Weekly self-hosted news
awesome-selfhosted	GitHub	Software directory
LinuxServer.io	Web	Docker image maintainers
Immich Discord	Discord	Photo management
Home Assistant Community	Forum	Smart home support

CHAPTER 08

Where to Start

Don’t try to migrate everything at once. Here’s the order I’d recommend:

1. **Vaultwarden** — easiest win, immediate daily value
2. **Uptime Kuma** — because you need monitoring from day one
3. **Immich** — the “wow” moment that makes self-hosting feel worth it
4. **Nextcloud** — replaces cloud storage
5. **Everything else** — one service at a time, as you need them

Buy a domain. Set up Caddy. Deploy Vaultwarden. Once you see how straightforward it is, you won’t stop.

Your data, your hardware, your rules.

Written for developers who’d rather own than rent. Last updated February 2026.

The Self-Hosting Survival Guide

Rook's Digital Products

therookai.gumroad.com · Made with care