

VPS Setup Guide

A step-by-step guide to hosting your own sites on a VPS with Coolify: from choosing a server to live, monitored, backed-up sites behind Cloudflare. Every command, setting and common error, explained.

Any VPS (OVHcloud example)

Ubuntu LTS

Coolify

Traefik

Nuxt / Node + pnpm

Railpack

GitHub Actions

Cloudflare

Cloudflare R2

Uptime Kuma

Umami

Domain email

Contents

0	Before you start	Placeholders, your setup sheet, architecture
1	Choosing the server	Provider, plan, VPS vs Vercel, capacity
2	Ordering the VPS	Options, billing, terms of service
3	Connecting with SSH	Keys, shortcut, lost port, rescue mode
4	Securing the server	Updates, SSH port & hardening, firewall, fail2ban, swap
5	Installing & configuring Coolify	Install, GitHub App, cleanup, API, notifications
6	The domain	Buying or transferring, DNS records, subdomains
7	Deploying a Nuxt site	Static & SSR, pnpm, CI/CD with GitHub Actions
8	Cloudflare	CDN, firewall, nameservers, DNSSEC, SSL, bots
9	Professional email	MX Plan, Roundcube, DKIM, DMARC
10	Monitoring & maintenance	Uptime Kuma, Umami, backups, logs, updates
11	Extras	Minecraft, game servers, self-hosted apps
12	Next steps checklist	The whole setup, in order
A	Quick reference	Most-used commands
B	Troubleshooting	Common errors and their fixes
C	Glossary	The words used in this guide

How to use this guide

- Every command was run on a real server (Ubuntu 26.04, Coolify v4.4, October 2026). Menu labels move between versions: if a button has a slightly different name, look for the closest match.
- `UPPERCASE` values and `example.com` are placeholders. Chapter 0 lists them; fill in your own setup sheet there first.
- Follow the chapters in order the first time. Chapters 10–12 and the appendices are for the long run.
- When something breaks, start with appendix B: every entry is an error met during a real setup.

0 Before you start

The placeholders used in this guide, and a sheet to fill in as you go.

Placeholders

In this guide	Replace with
<code>YOUR_SERVER_IP</code>	your VPS's public IPv4 address (from the provider's welcome email)
<code>49222</code>	the custom SSH port you choose (any free port between 1024 and 65535)
<code>ubuntu</code>	the default user your provider creates (<code>ubuntu</code> on OVH; sometimes <code>root</code> or <code>debian</code>)
<code>example.com</code>	your domain
<code>coolify.example.com</code> , <code>status.</code> , <code>stats.</code>	subdomains for the Coolify dashboard, Uptime Kuma and Umami
<code>my-coolify</code>	the name of your GitHub App (must be unique on GitHub)
<code>your-user/your-repo</code>	your GitHub repository
<code>NS1</code> / <code>NS2.ns.cloudflare.com</code>	the two nameservers Cloudflare assigns you
<code>YOUR-WEBSITE-ID</code>	the website ID Umami gives you
<code>coolify-backups</code>	your R2 bucket name
<code>ssh-vps</code> / <code>ssh vps</code>	your SSH shortcut (ch. 3)

Your setup sheet

Fill this in as you go and keep it somewhere safe. Never write passwords or secret keys here: they belong in a password manager.

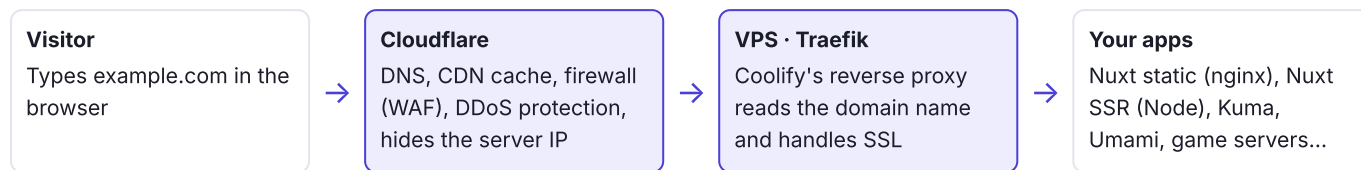
Item	Value
Provider / plan	
Location / region	
Operating system	
Public IPv4	
SSH	user ____ · port ____ · key <code>~/.ssh/id_ed25519</code> · shortcut ____
Hostname	
Coolify dashboard	<code>https://coolify.____</code> (DNS only in Cloudflare)
Domain / registrar / expiry	
Nameservers	
Status page / analytics	<code>https://status.____</code> · <code>https://stats.____</code>
Email	
GitHub App	
Backups	R2 bucket ____ · schedule ____

Things that expire

Coolify API token (30 days): _____ · domain: _____

Monthly cost

How the pieces fit together



Email takes a separate path: the MX records point to your email host's servers, not to the VPS. App databases can live on a hosted service like Supabase. `coolify.example.com` skips Cloudflare's proxy (DNS only) so GitHub Actions can reach the deploy API (ch. 8). Backups leave the server every night for Cloudflare R2 (ch. 10).

One IP, many sites

Every domain and subdomain points to the same IP. When a request arrives, Traefik reads the requested name (the HTTP `Host` header, or SNI for HTTPS) and forwards it to the right container. Websites use ports 80/443. Other services, like Minecraft on 25565, use their own ports on the same IP.

1 Choosing the server

Why a self-managed VPS on OVH, and how much it can hold.

Managed vs unmanaged

A **managed** VPS (e.g. Ultrahost) costs more because the host maintains the system for you. An **unmanaged** VPS (OVH, Hetzner, Contabo) is cheaper for the same resources, but you handle security and updates. **Coolify** brings back most of the convenience: deploy from GitHub, automatic SSL, logs and databases in a web dashboard.

Providers compared (October 2026)

Provider	Strength	Note
OVHcloud (used in this guide)	Good specs per euro, free anti-DDoS, daily backup included, European data centers	Support is average; most stock was sold out except the UK region
Hetzner	Best price/performance in the EU	The cheap CX line was out of stock; prices rose 30–37% in April 2026
Contabo	Most RAM per euro	Mixed reviews on performance and support
DigitalOcean / Vultr	Great docs and tooling	About 2–3× more expensive for the same specs
Ultrahost (managed)	They manage the server	Only 1 GB RAM for \$7; you pay for the support

VPS or Vercel?

Site	Where	Why
Your own portfolio / experiments	Vercel (free)	The Hobby plan is allowed for personal use and has a global CDN
Client sites	VPS	Vercel's free plan forbids commercial use. The VPS is already paid, and you can bill clients a monthly hosting fee
ERP, webhooks, background jobs	VPS	No serverless limits (timeouts, cold starts); runs non-stop
Client with heavy, worldwide traffic	Vercel Pro (billed to the client)	Automatic scaling

Serverless in one sentence

Your code runs only when a request arrives, on the provider's servers, then shuts down. It's great for zero maintenance, but it brings cold starts, time limits, and no memory between requests. On the VPS, the same `Nuxt server/api` routes run inside one Node server that's always on.

Capacity on VPS-3 (12 GB RAM)

Reserve about **4.5 GB** (system + Coolify ~1.5 GB, one build ~2 GB, safety margin ~1 GB). That leaves about **7.5 GB for your apps**.

What	RAM (approx.)	Disk (approx.)
Ubuntu + Docker + Coolify	1–1.5 GB	5–8 GB

Nuxt static site (nginx)	~5–10 MB	a few MB to ~100 MB
Nuxt SSR site (low traffic)	100–250 MB	200 MB–1 GB (image)
One <code>nuxt build</code> running (temporary)	1–2 GB	build cache 1–5 GB
Minecraft vanilla / Paper	3–4 GB	1–5 GB per world
Uptime Kuma / Umami (+ Postgres) / n8n	~100 / ~300 / ~400 MB	small

Realistic mix: 15 SSR sites plus as many static sites as you want, or a few sites plus Minecraft plus tools. Supabase runs outside the VPS, so it uses none of this RAM. Don't self-host Supabase here: it needs 2–3 GB on its own.

2 Ordering the VPS

The choices to make on the order page (OVH as the example), and the account details.

Option	Choice	Why
Plan	4 GB RAM minimum; 8–12 GB for SSR apps + tools + a game server	Coolify and one build need ~3.5 GB on their own (ch. 1)
Image	Ubuntu 26.04 (not Fedora)	LTS: years of security updates, and what every tutorial and Coolify expect
Location	The region closest to your visitors	Lower latency; for European clients, an EU region also keeps data in the EU
Commitment	No commitment	Test first. A 12-month commitment saves ~15% but renews automatically
Backups	Included daily backup only	Snapshot (€0.30) is optional; skip Premium backup and extra storage
Account type	Individual, or Company	Pick "Company" once you have a registered business, so invoices carry your company ID
Billing address	Your real address	VAT depends on your country: a billing address outside the EU usually means no EU VAT

Paying from outside the provider's country

Some banks block foreign online payments by default: enable **international online payments** on your card first. Expect a small currency-conversion fee.

Terms of service: what matters

- **Backups are your responsibility.** OVH isn't liable for lost data. Keep copies *outside* OVH.
- **To stop a monthly service**, turn off auto-renew **before the provider's cutoff** (on OVH: the 19th of the month, 11pm Paris time).
- **No refunds** for a month that has started. Domain orders waive the 14-day withdrawal right (it's a personalized product).
- OVH can **suspend without notice** in an emergency (security risk, illegal content, spam). You still pay during a suspension.
- Forbidden: spam, crypto mining, DDoS, game bots, Black-Hat SEO, attacks on other systems. Game servers are allowed.
- **Outgoing email on port 25 is monitored.** Don't send email from the VPS; use the MX Plan or a service like Resend or Brevo.
- Professional use requires **liability insurance**, and you must pass OVH's terms on to clients you host.
- Price increases come with **30 days' notice**, and you can then cancel without penalty.

3 Connecting with SSH

Logging into the server from your computer's terminal, and getting back in when you can't.

First connection

OVH emails you the IPv4 address, the user (`ubuntu`) and a temporary password. Always use the IPv4 address.

```
ssh ubuntu@YOUR_SERVER_IP          # first time, port 22, before changing it
```

Set up key-based login (on your computer)

```
ssh-keygen -t ed25519                # press Enter at each question
ssh-copy-id -p 49222 ubuntu@YOUR_SERVER_IP  # copies your public key to the server
```

Shortcuts (do this once)

Option 1, an SSH config entry, which also works with `scp` and `rsync` :

```
cat >> ~/.ssh/config <<'EOF'
Host vps
    HostName YOUR_SERVER_IP
    User ubuntu
    Port 49222
    IdentityFile ~/.ssh/id_ed25519
    IdentitiesOnly yes
EOF
chmod 600 ~/.ssh/config
```

Option 2, a shell alias (in `~/.bashrc`):

```
alias ssh-vps="ssh -o IdentitiesOnly=yes -i ~/.ssh/id_ed25519 -p 49222 ubuntu@YOUR_SERVER_IP"
```

"Too many authentication failures"

Your computer offers every stored key one after another, and the server disconnects before it gets to the right one.

`IdentitiesOnly yes` fixes it for keys. When the server expects a **password** (rescue mode), skip keys entirely:

```
ssh -o IdentitiesOnly=yes -i ~/.ssh/id_ed25519 -p 49222 ubuntu@YOUR_SERVER_IP
# password login (rescue mode):
ssh -o PubkeyAuthentication=no -o PreferredAuthentications=password root@YOUR_SERVER_IP
```

Reading the error

What you see	Meaning
<code>Connection refused</code>	The server is up and the packet arrived, but nothing listens on that port (or fail2ban rejected you).
Hangs, then times out	A firewall silently drops the packet (ufw blocks that port), or the server is off.
<code>Too many authentication failures</code>	Your client tried too many keys. See above.
<code>REMOTE HOST IDENTIFICATION HAS CHANGED</code>	The server's key changed. Expected in rescue mode or after a reinstall. Otherwise, stop and investigate.

Forgot which SSH port you set?

- 1 **known_hosts** records non-default ports as `[IP]:port`: `grep "YOUR_SERVER_IP" ~/.ssh/known_hosts`
- 2 **Shell history**: `history | grep -E "ssh|ufw|Port"`
- 3 **Scan your own server** (takes a few minutes): `nmap -p- --open YOUR_SERVER_IP`. "Connection refused" means nothing is listening on that port; an SSH error means you found it.
- 4 **Try port 22**. If SSH lost its custom port (ch. 4), it falls back to 22. If ufw blocks 22, this hangs.
- 5 **Rescue mode** (below). On VPS-3 2027 the Boot menu only offers Reboot and Rescue; there is no KVM console there.

Rescue mode: when SSH won't let you in

The VPS boots a temporary Linux from OVH. Your real disk is untouched and can be mounted and repaired. **Your sites are offline** for the 10–15 minutes this takes.

What the mount commands do

`mount` plugs your server's disk in at `/mnt` like an external drive. The `--bind` loop copies the live `/dev`, `/proc`, `/sys` and `/run` into it, which tools like `systemctl` and `sshd -t` need. `chroot /mnt` makes that disk your `/`, so every command edits your real server. `exit` + `umount -R` undo it all before rebooting.

- 1 **OVH panel** → your VPS → **Boot** ... → **Reboot in rescue mode** (about 3 minutes).
- 2 OVH emails a **link** (not the password). Open it while logged in to OVH to see the temporary root password once.
- 3 Connect. The rescue system has its own host key and listens on port 22:

```
ssh -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null \
-o PubkeyAuthentication=no -o PreferredAuthentications=password \
root@YOUR_SERVER_IP
```

- 4 Find your disk. `sda` (~3 GB) is the rescue system; `sdb1` (98.9 GB) is your server:

```
lsblk
```

- 5 Mount it and step inside it:

```
mount /dev/sdb1 /mnt
for d in dev proc sys run; do mount --bind /$d /mnt/$d; done
chroot /mnt                # "/" is now your real server's disk
```

- 6 Inspect and fix (example: the SSH port):

```
grep -rn "49222" /etc 2>/dev/null          # where the port is configured
grep -rin "port" /etc/ssh/sshd_config /etc/ssh/sshd_config.d/
sed -i 's/^#Port 49222/Port 49222/' /etc/ssh/sshd_config
mkdir -p /run/sshd && sshd -t && echo OK
systemctl disable ssh.socket && systemctl enable ssh.service
```

- 7 Leave and unmount cleanly:

```
exit
umount -R /mnt
```

- 8 OVH panel: check **Boot = LOCAL**, then **Reboot my VPS**. Otherwise it boots into rescue again.
- 9 Back on your computer, clear the rescue host key, then connect normally:

```
ssh-keygen -R YOUR_SERVER_IP  
ssh-keygen -R "[YOUR_SERVER_IP]:49222"  
ssh-vps
```

4 Securing the server

Run these once, right after the first login. Paste a command's output into the chat if something looks wrong.

1 · Update the system and enable automatic security updates

```
sudo apt update && sudo apt upgrade -y
sudo apt install -y unattended-upgrades fail2ban
sudo reboot
```

Let security-update reboots happen at a quiet hour: edit `/etc/apt/apt.conf.d/50unattended-upgrades` and set

```
Unattended-Upgrade::Automatic-Reboot "true";
Unattended-Upgrade::Automatic-Reboot-Time "04:00";      # server clock (check: timedatectl)
```

2 · Change the SSH port

The trap: a port that disappears after a reboot

Ubuntu 24.04 and later start SSH through `ssh.socket`. If the `Port` line in `sshd_config` is still commented out (`#Port 49222`), a custom port set another way only lasts until the next reboot. After it, SSH listens on 22, ufw blocks 22, and nothing listens on your port: you're locked out and need rescue mode (ch. 3). The procedure below makes the port permanent.

```
sudo ufw allow 49222/tcp                # 1. open the new port FIRST
sudo sed -i 's/^#\?Port .*/Port 49222/' /etc/ssh/sshd_config
grep -n "^Port" /etc/ssh/sshd_config   # 2. must show: Port 49222 (no #)
sudo sshd -t && echo OK                 # 3. config has no errors
sudo systemctl disable --now ssh.socket # 4. let sshd read the port itself
sudo systemctl enable --now ssh.service
sudo ss -tlnp | grep sshd              # 5. must show :49222
```

- 1 Keep this session open. In a **second terminal**, run `ssh-vps`.
- 2 Only when that works, run `sudo reboot` once and connect again, to prove the port survives a reboot.

3 · SSH hardening

Files in `/etc/ssh/sshd_config.d/` override the main config, and **the first value read wins**. Cloud images ship `50-cloud-init.conf`, which can turn password login back on, so put your rules in a file read before it:

```
sudo tee /etc/ssh/sshd_config.d/00-hardening.conf > /dev/null <<'EOF'
PasswordAuthentication no
PermitRootLogin prohibit-password
KbdInteractiveAuthentication no
EOF
sudo sshd -t && sudo systemctl restart ssh
sudo sshd -T | grep -E "(port|passwordauthentication|permitrootlogin|pubkeyauthentication)"
```

Expected: `port 49222`, `passwordauthentication no`, `permitrootlogin prohibit-password`, `pubkeyauthentication yes`. Test from a second terminal before closing the first.

Not "PermitRootLogin no"

Coolify manages this server by logging in as **root with a key** (ch. 5). `no` breaks Coolify's server connection and deploys. `prohibit-password` allows root only with a key, never a password.

4 · Firewall (ufw)

```
sudo ufw default deny incoming
sudo ufw default allow outgoing
sudo ufw allow 49222/tcp          # SSH: essential
sudo ufw allow 80/tcp            # HTTP
sudo ufw allow 443/tcp           # HTTPS
sudo ufw enable
sudo ufw status verbose
```

Docker bypasses ufw

Ports published by containers (Coolify's 8000/6001/6002, Minecraft's 25565...) stay reachable from the internet even if ufw doesn't allow them. ufw protects SSH and the system. For Coolify, rely on a strong password plus 2FA and always use the HTTPS domain. To block a Docker port completely, use OVH's network firewall (in the panel, under your IP).

5 · fail2ban on the custom SSH port

```
sudo tee /etc/fail2ban/jail.local > /dev/null <<'EOF'
[sshd]
enabled = true
port = 49222
backend = systemd
maxretry = 5
bantime = 1h
EOF
sudo systemctl restart fail2ban
sudo fail2ban-client status sshd
# banned yourself? connect from another network (phone hotspot), then:
sudo fail2ban-client unban --all
```

6 · Swap file (safety net for memory spikes)

Swap is disk space used as overflow memory. It's slower than RAM, but it means a spike slows things down instead of Linux killing an app. It uses 4 GB of disk.

```
sudo fallocate -l 4G /swapfile
sudo chmod 600 /swapfile
sudo mkswap /swapfile
sudo swapon /swapfile
echo '/swapfile none swap sw 0 0' | sudo tee -a /etc/fstab
echo 'vm.swappiness=10' | sudo tee /etc/sysctl.d/99-swappiness.conf # swap only as a last resort
sudo sysctl -p /etc/sysctl.d/99-swappiness.conf
free -h
```

5 Installing & configuring Coolify

Coolify is a free, self-hosted alternative to Vercel. It deploys from GitHub, manages domains and SSL, and runs databases and services.

Coolify or Dokploy?

Both do the same job and use about 600 MB of RAM. Coolify has a larger community and 280 one-click apps. Dokploy has a cleaner UI, volume backups with restore from the UI, and built-in multi-server support. Coolify was chosen; switching later means redeploying everything.

Install

```
curl -fsSL https://cdn.coollabs.io/coolify/install.sh | sudo bash
```

Right after the install

Open `http://YOUR_SERVER_IP:8000` and **create the admin account immediately**. The first person to open that page becomes admin.

Fix: "ssh: connect to host host.docker.internal port 22: Connection refused"

Coolify manages "this machine" over SSH, as `root`, on port 22 by default. Since SSH listens on 49222:

- 1 On the onboarding screen, choose **This machine**. If the error appears, skip onboarding.
- 2 **Servers** → **localhost** → **General**: Port `49222`, User `root` → Save → **Validate Server**.
- 3 If it still fails, check that root can log in with a key:

```
sudo sshd -T | grep permitrootlogin          # must be prohibit-password, not no
sudo sed -i 's/^PermitRootLogin no/PermitRootLogin prohibit-password/' /etc/ssh/sshd_config.d/00-
hardening.conf
sudo sshd -t && sudo systemctl restart ssh
```

- 4 Check that Coolify's key is authorized:

```
sudo cat /root/.ssh/authorized_keys
sudo cat /data/coolify/ssh/keys/*.pub | sudo tee -a /root/.ssh/authorized_keys # only if missing
```

Instance settings (Settings → General)

- 1 DNS first: an A record `coolify` → `YOUR_SERVER_IP` in Cloudflare, **DNS only** (ch. 8). Check it with `dig +short coolify.example.com`.
- 2 URL: `https://coolify.example.com` · Instance timezone: yours (e.g. `Europe/Paris`) → Save. Only save the URL once the DNS resolves, or the SSL certificate request fails.
- 3 Use `https://coolify.example.com` from now on, never `http://IP:8000` (unencrypted). Enable **2FA** on your profile.

Connect GitHub (Sources → + Add → GitHub App)

Field	Value
Name	<code>my-coolify</code> (must be unique on GitHub)
Organization	empty (personal account)
System wide	unchecked

Advanced "Self-hosted / Enterprise
GitHub"

leave the defaults: `https://github.com`, `https://api.github.com`, user
`git`, port `22` (GitHub's SSH port, not yours)

On GitHub: create the app, then install it on **Only select repositories** (or All).

Docker access for the ubuntu user

`permission denied ... /var/run/docker.sock` means `ubuntu` isn't in the `docker` group. Use `sudo docker ...`, or once:

```
sudo usermod -aG docker $USER && newgrp docker # new SSH sessions pick it up
```

Membership of the docker group is effectively root access. Fine on your own VPS; never give it to someone you don't trust.

Server settings (Servers → localhost)

Where	Setting
General / Advanced	Concurrent builds: 1 , so two builds never compete for RAM
Docker Cleanup	Frequency <code>0 0 * * *</code> (daily, 00:00 UTC; runs in seconds, no downtime) · Trigger: run on every schedule · Unused volumes: Keep · Unused networks: Keep · Application images: Keep retained images (needed for Rollback)
Sentinel	Enable: CPU/RAM/disk graphs under Metrics, and disk alerts
Analytics	Traffic analytics: off . Umami and Cloudflare already cover traffic, and behind Cloudflare Traefik mostly sees Cloudflare's IPs

Never set cleanup to remove volumes

A volume counts as "unused" whenever its container is stopped. A stopped Minecraft server or Uptime Kuma would lose its data at the next cleanup.

API access and tokens (for CI/CD)

- 1 **Settings → API access: on.** Leave the IP allowlist **empty**: GitHub's runners use thousands of changing IPs. The "open to every source" warning is expected; every request still needs a token.
- 2 **Keys & Tokens → API Tokens → create**, permission **deploy** only. Copy it immediately (shown once). If it leaks, all anyone can do is redeploy your app.
- 3 Keep **one** token and revoke duplicates. "Last used" shows whether the pipeline uses it.
- 4 The token **expires after 30 days** (write the date in your setup sheet). Before then: create a new one, paste it into GitHub's `COOLIFY_TOKEN` environment secret (ch. 7), revoke the old one.

Notifications

Notifications (main sidebar) → **Telegram**: bot token from `@BotFather`, chat ID from `@userinfobot` → **Send test** → enable. Turn on: deployment failed, backup failed, scheduled task failed, server unreachable, server disk usage. Set the disk alert threshold around **80%**.

Updates

What	Where	Impact
Coolify	Settings → Update / Auto update (a button also appears in the sidebar). If auto-update has a cron, use a quiet time, e.g. <code>0 3 * * 0</code>	Dashboard restarts; sometimes the proxy too

Traefik (proxy)	Servers → localhost → Proxy → Configuration → change <code>image: 'traefik:v3.x'</code> → Save → Restart Proxy . Read the changelog first	Every site offline for a few seconds
Ubuntu	SSH: <code>sudo apt update && sudo apt upgrade -y</code> ; reboot only if <code>/var/run/reboot-required</code> exists	Reboot = everything down 1–2 min

Is Coolify running?

```

sudo docker ps --format "table {{.Names}}\t{{.Status}}" # coolify, -db, -redis, -realtime, -proxy:
"Up (healthy)"
sudo docker logs --tail 50 coolify
curl -I http://localhost:8000 # 200 or 302 = dashboard up
sudo docker restart coolify coolify-realtime coolify-db coolify-redis
curl -fsSL https://cdn.coollabs.io/coolify/install.sh | sudo bash # repair / update without losing
data

```

Coolify runs as Docker containers and starts automatically after a reboot. There's nothing to "launch".

6 The domain

The domain stays registered at a registrar (OVH in the examples). Once Cloudflare manages its DNS (chapter 8), every record is added in Cloudflare, never at the registrar.

Transferring a domain (example: Namecheap → OVH)

- 1 Check the **60-day rule**: no transfer within 60 days of registration or a previous transfer.
- 2 Namecheap → Domain List → Manage → Sharing & Transfer: **Domain Lock OFF**, then request the **Auth/EPP code** (emailed to the domain's contact).
- 3 **Screenshot every DNS record** (Advanced DNS). Namecheap's DNS stops working once the domain leaves.
- 4 OVH may show "Transfer locked" for a while after unlocking (cached). Wait 30–60 min, or use OVH's transfer page directly.
- 5 Order the transfer on the **European** OVH site (same account as the VPS) and enter the auth code. The US site is a separate company with separate accounts.
- 6 Click the **approval link** in the email to finish in hours instead of 5 days. Don't re-lock the domain meanwhile.

Order option	Choice
Free hosting (Start 100M) + 1 email account	Take it: the free email address is useful; ignore the 100 MB hosting
DNS Anycast (€1.09/year)	Skip: it only speeds up the DNS lookup by milliseconds
Withdrawal-right waiver checkbox	Required for domains (personalized product, no refund)

A transfer usually costs about one year of renewal and adds one year on top of the existing time. The time already paid is never lost. Already have a domain at a registrar you like? Skip the transfer: only the DNS needs to move to Cloudflare.

Records that point the domain to the VPS

Type	Name	Target	Cloudflare proxy
A	@	YOUR_SERVER_IP	Proxied
A	www	YOUR_SERVER_IP (or CNAME → example.com)	Proxied
A	coolify	YOUR_SERVER_IP	DNS only (ch. 8)
A	status	YOUR_SERVER_IP	Proxied
A	stats	YOUR_SERVER_IP	Proxied
A	*	YOUR_SERVER_IP	optional wildcard: every subdomain works automatically

After adding OVH's free hosting

OVH points @ / www to the hosting (51.91.236.255) and adds AAAA records. Change the A records to the VPS and delete the AAAA records for @ and www, or IPv6 visitors land on the wrong server. Never delete the MX, SPF, DKIM, SRV, mail/imap/smtp/pop3/autoconfig/autodiscover records: those run your email.

Adding a subdomain: the right order

Subdomains are free and practically unlimited. A subdomain is just a DNS record, so it's created where the DNS lives (Cloudflare), not at the registrar.

- 1 **Cloudflare** → example.com → DNS → Records → Add record: type `A`, name `status` (just the word), IPv4 `YOUR_SERVER_IP`, same proxy setting as the other web records → Save.
- 2 **Don't open the URL yet.** Wait until public DNS answers: `dig +short status.example.com @1.1.1.1`.
- 3 **Coolify** → the resource → **Domains** → **Add Domain**: protocol `https`, the domain, the app's **internal port** (Kuma `3001`, Umami `3000`), Search indexing **Noindex** for admin pages, www redirect "No redirect". Leave Path empty; never click "Regenerate Hostname".
- 4 **Restart** the resource so Traefik picks up the domain and requests the certificate. The Domains page should show **DNS matches**.
- 5 Now open it in the browser.

Why the order matters: negative DNS caching

If you open the URL before the record exists, your browser, your computer and your router cache the answer "this name doesn't exist" for up to ~30 minutes, even after the record is created. Firefox then shows `NS_ERROR_UNKNOWN_HOST` while everyone else sees the site. Fix: `about:networking#dns` → Clear DNS Cache, `sudo resolvectl flush-caches`, restart Firefox, or test on your phone using mobile data. Coolify's "DNS is not configured" warning in the meantime means the record doesn't exist yet.

Typical subdomains: `coolify.` (dashboard), `status.` (Uptime Kuma), `stats.` (Umami), `demo-client.` (client previews before go-live), `mc.` (Minecraft, DNS only). Testing without a domain: Coolify's `xxxx.YOUR_SERVER_IP.sslip.io` addresses expose your IP and are often plain HTTP, so replace them before sharing anything.

7 Deploying a Nuxt site

From a GitHub repo to a live HTTPS site. Pushes to `main` deploy only after GitHub Actions has checked the code (CI/CD, below).

Create the resource

- 1 **Projects** → **+ Add** (e.g. `my-site`) → environment `production`.
- 2 **+ New** → **Private Repository (with GitHub App)** → `my-coolify` → choose the repo and branch `main`.
- 3 Build pack: **Railpack** (the successor to Nixpacks). If a build fails with it, try Nixpacks. "Static" is only for repos that already contain built HTML; "Dockerfile/Docker Compose" are for custom setups.

Static site (`nuxt generate`)

Field	Value
Output type / Site type	Static
Web server	<code>nginx:alpine</code>
Base directory / Publish directory	<code>/ · /.output/public</code>
Install command	empty (Railpack runs <code>pnpm install</code> itself)
Build command	<code>pnpm nuxt generate</code> (or <code>pnpm generate</code> if the script exists)
Start command	empty (nginx serves the files)
Port (exposes)	<code>80</code>
Watch paths	empty (the grey <code>src/pages/**</code> is only an example)

SSR site (API routes + Supabase)

Field	Value
Build command	<code>pnpm build</code>
Start command	<code>node .output/server/index.mjs</code>
Port (exposes)	<code>3000</code> . A wrong port shows as Exited / Bad Gateway / "no available server"
Environment Variables	your secrets (Supabase URL and keys), same names as your local <code>.env</code> , never committed. A value read at build time (e.g. by a static site) needs Available during build

pnpm requirements

- Commit `pnpm-lock.yaml` . Without it, Railpack falls back to npm.
- Pin the version in `package.json` : `"packageManager": "pnpm@11.22.0"` (check yours with `pnpm -v`). CI reads it too.

```
"scripts": {
  "build": "nuxt build",
  "dev": "nuxt dev",
  "generate": "nuxt generate",
  "preview": "nuxt preview",
  "postinstall": "nuxt prepare",
  "lint": "eslint .",
  "typecheck": "nuxt typecheck"
}
```

`nuxt typecheck` needs `vue-tsc` and `typescript` as dev dependencies, or it prompts and fails in CI.

Domains (resource → Domains)

- 1 Replace the sslip.io address with `https://example.com` and add `https://www.example.com`. Always use `https://`, or no certificate is issued.
- 2 Check **DNS matches** on both, and that "Redirect HTTP to HTTPS" is Enabled.
- 3 Optional: on the www row ⚙️, set Domain redirect → example.com (better for SEO).
- 4 "Changes pending" means nothing is applied yet: Actions → Redeploy.

CI/CD: deploy only when the checks pass



If CI fails, nothing deploys and the current version stays online. Pull requests run CI but never deploy.

- 1 **Coolify** → the app → **Advanced** → **Deployment** → **Auto deploy: Manual deployments only**. Otherwise Coolify deploys on every push, before CI finishes.
- 2 **Coolify** → the app → **Webhooks** → **Deploy webhook**: copy the URL (`https://coolify.example.com/api/v1/deploy?uuid=...&force=false`).
- 3 **Coolify** → API token with deploy permission (ch. 5).
- 4 **GitHub** → repo → Settings → Secrets and variables → Actions → **Secrets** tab. Yours live in the environment named `Coolify`: `COOLIFY_WEBHOOK` (the URL) and `COOLIFY_TOKEN` (the token). Environment secrets are only visible to a job that declares `environment: Coolify`.
- 5 Add `.github/workflows/ci.yml` to the repo (in your code, not in Coolify), commit, push.

```

name: ci

on:
  push:
    branches: [main]
  pull_request:

concurrency:
  group: ${ github.workflow }}-${ github.ref }}
  cancel-in-progress: true

permissions:
  contents: read

jobs:
  ci:
    runs-on: ubuntu-latest
    timeout-minutes: 10
    steps:
      - name: Checkout
        uses: actions/checkout@v7
      - name: Install pnpm
        uses: pnpm/action-setup@v6 # reads "packageManager" from package.json
      - name: Install node
        uses: actions/setup-node@v7
        with:
          node-version: 22 # match the Node version Coolify builds with
          cache: pnpm
      - name: Install dependencies
        run: pnpm install --frozen-lockfile
      - name: Lint
        run: pnpm run lint
      - name: Typecheck
        run: pnpm run typecheck
      - name: Build
        run: pnpm run build

  deploy:
    needs: ci
    if: github.event_name == 'push' && github.ref == 'refs/heads/main'
    runs-on: ubuntu-latest
    timeout-minutes: 5
    environment: Coolify # exact name, capital C: gives access to its secrets
    steps:
      - name: Trigger Coolify deploy
        run: |
          curl -fsS -X POST "${ secrets.COOLIFY_WEBHOOK }}" \
            -H "Authorization: Bearer ${ secrets.COOLIFY_TOKEN }"

```

Check it worked: GitHub → Actions → both jobs green; Coolify → Deployment Logs shows a new deployment; the token's "Last used" updates. To see Coolify's error message in a failing run, temporarily change `-fsS` to `-sS`. You can still click **Deploy** in Coolify by hand at any time.

Reading a failed build

The error is almost always in the last 20–30 lines of the deployment log (Deployments → the deploy → Show Debug Logs). For a CI failure, open the red step in GitHub → Actions.

Log message	Cause	Fix
Command "generate" not found	no <code>generate</code> script	Build command <code>pnpm nuxt generate</code> , or add the script

<code>ERR_PNPM_OUTDATED_LOCKFILE</code>	lockfile doesn't match package.json	run <code>pnpm install</code> locally, commit <code>pnpm-lock.yaml</code> , push
Blank page / 404 on a static site	<code>nuxt build</code> ran instead of <code>generate</code>	set the build command to generate
Bad Gateway / app Exited	wrong port, or the app crashed	3000 for SSR, 80 for static; check the resource's Runtime Logs

CI/CD errors (empty secrets, 403, 405) are in appendix B.

8 Cloudflare

Free CDN, firewall (WAF), DDoS protection and bot blocking, plus it hides the VPS IP. The domain stays registered at OVH; only DNS moves to Cloudflare.

1 · Add the site

Search bots: Allow · **Agent bots** (ChatGPT, Claude, Perplexity citing you): Allow · **Training bots:** your choice (Block protects your texts and 3D work) · Bot Preference Sync: on

Import DNS records: Automatic. Then compare carefully with OVH: the scan misses records.

2 · Proxy status for each record

Records	Proxy	Why
@, www, status, stats, *, other web subdomains	Proxied	CDN + firewall + hidden IP
coolify	DNS only	GitHub Actions must reach the deploy API without being challenged as a bot. HTTPS still works through Coolify's own certificate. Protected by password + 2FA
mail, imap, smtp, pop3, autoconfig, autodiscover	DNS only	the proxy only handles web traffic; it would break email
MX, SPF (TXT), DKIM, SRV, DMARC	(not proxiable)	email
mc (Minecraft) and any game server	DNS only	not HTTP; players couldn't connect

Records the automatic import often misses: add them by hand, with values copied from your old DNS zone (without the trailing dot):

Type	Name	Content
A	coolify	YOUR_SERVER_IP (DNS only)
CNAME	ovhmo-selector-1._domainkey	copy from your old zone (DNS only), DKIM
CNAME	ovhmo-selector-2._domainkey	copy from your old zone (DNS only), DKIM
SRV	_autodiscover._tcp	priority 0 · weight 0 · port 443 · target mailconfig.ovh.net

The DKIM names above are OVH MX Plan's; other email hosts (Google, Proton, Zoho...) give you their own DKIM records. OVH hosting markers like "1|www.example..." and "3|welcome" TXT records are harmless; delete or keep them.

3 · DNSSEC and nameservers: the safe order

- 1 OVH → example.com → General information → **DNSSEC: turn off**. If it's on when you switch, the domain stops resolving. Check that no DS record is published anymore (e.g. on dnsviz.net).
- 2 OVH → DNS servers → **Use my own DNS**: enter NS1.ns.cloudflare.com → Add → NS2.ns.cloudflare.com → Add → Apply configuration. Leave "Associated IP" empty. Don't mix with OVH's servers.
- 3 Wait for **Active** in Cloudflare. Check: dig +short NS example.com → the 2 cloudflare.com servers.
- 4 **Re-enable DNSSEC via Cloudflare** (DNS → Settings → Enable DNSSEC), only once the domain is Active. In OVH → DS records, replace any old entry with Cloudflare's values: Key tag, Flag 257 (KSK), Algorithm 13, Public key (copy it in full with Cloudflare's copy button). Cloudflare shows "DNSSEC: Success" within hours.

From now on

All DNS changes are made in Cloudflare. The OVH DNS zone is no longer used. Records added at OVH are ignored.

4 • Settings

Where	Setting
SSL/TLS → Overview	Full (strict) . Otherwise you risk an infinite redirect loop ("too many redirects")
SSL/TLS → Edge Certificates	Always Use HTTPS: on · Automatic HTTPS Rewrites: on · Minimum TLS: 1.2
Security → Bots (or Security → Settings → Bot traffic)	Bot Fight Mode: Off (see below)
Speed → Settings → Protocol	HTTP/3 (with QUIC): on
Speed → Settings → Content	Brotli is always on now (no toggle)

Why Bot Fight Mode is off

It challenges anything that looks automated: GitHub Actions calling the deploy API (403 "Managed Challenge"), Uptime Kuma's checks from the VPS (false "down" alerts), external monitors and webhooks. On the free plan it can't be bypassed with a rule. You keep Cloudflare's DDoS protection and WAF managed rules without it.

5 • Reading what Cloudflare blocked

Security → Events lists every blocked or challenged request. Expand one: the **Service** field names the feature responsible (Bot fight mode, Managed rules, Custom rule, Browser Integrity Check...), and "Export event JSON" shows `ruleId` and `source`. **Analytics → Traffic** (sampled logs) shows everything that got through, including bots probing `/.git/config`: harmless as long as `curl -sI https://example.com/.git/config` returns 404.

Optional WAF rule (Security → WAF → Custom rules): Block when URI Path contains `/.git` or `/.env` or `/wp-`. It cuts scanner noise and never affects real visitors.

6 • Verify

```
curl -sI https://example.com | grep -i -E "server|cf-ray" # server: cloudflare + cf-ray = proxied
dig +short example.com @NS1.ns.cloudflare.com             # Cloudflare IPs (104.21.x / 172.67.x)
resolvectl flush-caches                                  # if your computer still shows an old
answer
```

`curl` on a proxied site can return **403** with ~5 KB of HTML: that's Cloudflare's bot challenge for curl, not an error on your server. Browsers pass.

9 Professional email

Many registrars include a small mailbox with the domain (example: OVH MX Plan). Never send email from the VPS itself; it lands in spam.

Create the mailbox

- 1 OVH → Web Cloud → Emails (MX Plan) → example.com → **Emails** tab → Create an email address.
- 2 Account name `contact` → `contact@example.com`, with a strong password saved in a password manager.
- 3 The "device" step only shows setup guides, so any choice is fine → Confirm. Activation takes a few minutes (see "Ongoing jobs").
- 4 Read mail via **Webmail** → **Roundcube** (link on the MX Plan page). Bookmark it.
- 5 Test both directions: send from Proton/Gmail to `contact@`, then reply from Roundcube.

Mail apps (phone, Thunderbird...)

Protocol	Server	Port	Security
IMAP (incoming)	<code>ssl0.ovh.net</code>	993	SSL/TLS
SMTP (outgoing)	<code>ssl0.ovh.net</code>	465	SSL/TLS

Username = the full email address. Confirm the values shown in the OVH panel if a client refuses them.

Deliverability: SPF, DKIM, DMARC

- **SPF** (TXT on `@`): usually created by your email host; check it was imported
- **DKIM**: the two `ovhmo-selector-...._domainkey` CNAMEs (DNS only). The MX Plan "Diagnostic" shows whether DKIM is OK.
- **DMARC**: add it yourself in Cloudflare:

```
Type: TXT      Name: _dmarc
Content: v=DMARC1; p=quarantine; rua=mailto:contact@example.com
```

Check existing redirections (OVH: Emails → Redirections) so no mail silently forwards somewhere you don't expect.

10 Monitoring & maintenance

Know when something breaks before a client tells you, and keep the server healthy.

Uptime Kuma: is everything up?

- 1 Coolify → project → + **New** → **Service** → **Uptime Kuma** (the default template, **not** MySQL).
- 2 Domains → Add Domain: `https://status.example.com`, port `3001`, Noindex → Save → **Restart** (ch. 6 for the DNS order).
- 3 Open the URL and **create the admin account at once** (the first visitor becomes admin). Database: **SQLite**: one file in the volume, lightest, easy to back up. MariaDB only pays off with hundreds of monitors.
- 4 Settings → Monitor History: keep ~30 days so the SQLite file stays small.

Monitor	Type	Target
My site	HTTP(s)	<code>https://example.com</code>
Umami	HTTP(s)	<code>https://stats.example.com</code>
Coolify	HTTP(s)	<code>https://coolify.example.com</code>
Minecraft	TCP Port	<code>YOUR_SERVER_IP</code> : <code>25565</code> (only if it runs constantly)

Per monitor: interval **60 s**, **Retries 3** (so one slow response doesn't alert), timeout 30–48 s, tick **Certificate Expiry** and **Domain Name Expiry** notifications, never tick "Ignore TLS/SSL errors". Proxy, HTTP options and Authentication stay default.

Alerts: Settings → Notifications → Setup Notification → Telegram (`@BotFather` token + `@userinfobot` chat ID) → Test → **Default enabled** + apply on all existing monitors.

Status page (optional): Status Pages → New: slug `my-site`, add public monitors only (not Coolify). To show it at the root URL: Settings → General → Entry Page.

Who watches the watcher?

Kuma runs on the same VPS. If the server dies, Kuma dies with it and says nothing. Add one free external monitor (UptimeRobot or Better Stack) on `https://status.example.com`.

Umami: who visits?

- 1 Coolify → + **New** → **Service** → **Umami** (the template **with PostgreSQL**; Coolify generates the DB password and APP_SECRET).
- 2 Domains → `https://stats.example.com`, port `3000`, Noindex.
- 3 Environment Variables → `TRACKER_SCRIPT_NAME` = `app.js` (Runtime: available in the container). Ad blockers block the default `script.js`; avoid names with "stats", "track" or "analytics". Restart, then check `https://stats.example.com/app.js` shows JavaScript.
- 4 Log in with the default `admin` / `umami` and **change the password immediately** (Settings → Profile).
- 5 Settings → Websites → Add website (name + domain) → Edit → **Tracking code** gives the website ID.

In `nuxt.config.ts` (the website ID isn't secret: it's visible in the page source, so you can write it directly instead of using `process.env`):

```
export default defineNuxtConfig({
  app: {
    head: {
      script: [{
        src: 'https://stats.example.com/app.js',
        defer: true,
        'data-website-id': 'YOUR-WEBSITE-ID',
        'data-domains': 'example.com,www.example.com', // only these hostnames count; localhost is
        ignored
      }],
    },
  },
})
```

- **One website per project.** A website isn't tied to a domain: the ID in the script decides. One ID can cover several subdomains (filter by hostname), and `data-domains` stops other sites from sending data with your ID.
- **Verify:** DevTools → Network → `app.js` and `send` (POST `/api/send`) both 200 → visit appears in **Realtime** within seconds.
- **Exclude yourself:** on your site, DevTools Console → `localStorage.setItem('umami.disabled', 1)` (once per browser).
- **Reading the data:** identical sessions at the same minute (e.g. US / Chrome / Windows, 3 views each) are bots running a real browser. Channel "Paid ads" means the URL carried ad parameters (`gclid`, `fbclid`...): bots again, since you run no ads.
- **Growth reports:** *UTM* splits visits by tagged links (`?utm_source=instagram&utm_medium=bio`); *Revenue* sums `umami.track('checkout', {revenue, currency})` events; *Attribution* shows which channel led to a goal like `umami.track('contact-submit')`.
- **Back up** Umami's PostgreSQL (see "Backups to Cloudflare R2" below).

Backups to Cloudflare R2

A backup on the same server protects you from mistakes, not from losing the server (hack, disk failure, suspended account). Copies must leave the VPS. R2 gives 10 GB, 1 M writes and 10 M reads free per month, and downloads are free.

R2 line item	Meaning	Typical small setup
Storage (10 GB free)	average data stored over the month; then \$0.015/GB	1–5 GB with retention set
Class A (1 M free)	writes: uploads, listing, copying	a few thousand a month
Class B (10 M free)	reads: downloads, existence checks	almost none (restores only)

There is no spending cap

R2 can't stop at the free tier. Keep the bucket **private** (no public `r2.dev` URL), set **retention** on every backup, and add a **budget alert** at \$0.50 (Billing → Billable Usage, or the R2 page). Want a hard cap instead? Backblaze B2 is also S3-compatible and has "Caps & Alerts".

- 1 **Cloudflare → R2 → Add R2 subscription** (\$0 due; a card is pre-authorized). **Create bucket** `coolify-backups`: location Automatic or a hint near your server, storage class **Standard** (the free tier doesn't apply to Infrequent Access), public access off.
- 2 **Manage API Tokens → Create Account API token** (tied to the account, not your login): permission **Object Read & Write, only** that bucket, TTL Forever. Copy the Access Key ID, Secret Access Key (shown once) and the S3 endpoint into a password manager.

- 3 **Coolify → S3 Storage → + Add:** protocol `https`, Host `ACCOUNT_ID.r2.cloudflarestorage.com` (no `https://`, nothing after `.com`), Port and Path empty, Bucket `coolify-backups`, Region `auto`, the two keys → **Validate Connection & Continue**.
- 4 **Coolify's own database** (projects, settings, env vars): Settings → Backup → frequency `0 3 * * *` → **Enable S3** (the purple button means it's still off) → retention below → **Back Up Now**.
- 5 **App databases** (e.g. Umami's PostgreSQL): the service → Backups → **Add Backup → Database backup** → same schedule, S3 on, same retention.
- 6 **Volumes** (e.g. Uptime Kuma, a Minecraft world): **Add Backup → Storage backup** → the volume. Archive behavior: **stop containers during the archive** for anything with a live database file (SQLite) or a game world, otherwise the copy can be corrupt. Kuma: daily `30 3 * * *`; a game world: weekly, at a time nobody plays.
- 7 **Check:** each backup's Executions show Success, and R2 → bucket → **Objects** has files under `data/coolify/backups/` (`coolify/`, `databases/`, `volumes/`) with a size above 0. The R2 overview's storage numbers lag by hours; the Objects list is live.

Retention setting	Value	Why
Local backups to keep	7	fast restores after a mistake, without filling the VPS disk
S3 backups to keep	14 (or 30)	survive losing the server; time to notice a problem days later
S3 maximum storage	e.g. 2 GB	safety cap well under the free 10 GB
Days to keep / max storage = 0	no limit by that rule	fine when "backups to keep" is set
Missing backup alert	~26 h / 2 days (check the unit)	a notification when a nightly run is skipped

Don't rename, move or delete files in the bucket by hand: Coolify tracks them by path for retention and restores. Test a restore once (e.g. restore a database backup into a test copy) and write the steps down.

Where to find logs

What	Where
A build (why a deploy failed)	Coolify → resource → Deployment Logs (Show Debug Logs); CI failures: GitHub → Actions
A running site or app	Coolify → resource → Runtime Logs
Domain routing / SSL issues	Coolify → Servers → localhost → Proxy → Logs, or <code>sudo docker logs coolify-proxy --tail 30</code>
Real CPU/RAM graphs	Servers → localhost → Sentinel / Metrics
Blocked / challenged requests	Cloudflare → Security → Events
Machine health, reboots, rescue	OVH panel: usage graphs, operations history. It can't see inside the VPS

```

sudo docker ps                # container names and how long they've been up
sudo docker logs -f --tail 100 <name>    # one container, live
sudo journalctl -f             # system, live
sudo journalctl -p err -b       # errors since boot
sudo journalctl -u ssh --since today    # SSH attempts (bot noise is normal)
sudo fail2ban-client status sshd        # banned IPs

```

Press Ctrl + C to leave any live (`-f`) view.

RAM: read `available`, not the OVH graph

```
free -h
      total        used        free      shared  buff/cache   available
Mem:   11Gi       1.7Gi       426Mi       241Mi         9.8Gi         9.7Gi
```

OVH's graph showed 53%, then 99% after a build. The real usage was 1.7 GB. Linux fills free RAM with disk cache (`buff/cache`) and releases it instantly when an app needs memory. Rule of thumb: everything's fine as long as **available** stays above 2 GB.

```
sudo docker stats --no-stream --format "table {{.Name}}\t{{.MemUsage}}" # per container
```

Disk

```
df -h / # keep Use% under ~80%
sudo docker system df # images, containers, volumes, build cache + reclaimable
sudo apt install ncdu && sudo ncdu /
# cleanup (does NOT touch volumes = your data, unless you add --volumes)
sudo docker system prune -af
sudo docker builder prune -af
```

Maintenance windows

Restarting Traefik, rebooting Ubuntu or updating Coolify takes sites offline briefly. Do it when traffic is lowest (check Umami; for a local audience, usually 2–6 AM), and for client sites, tell them in advance ("maintenance Tuesday 3:00–3:15 AM").

- 1 Check an OVH automated backup exists (VPS page → Backup), or take a snapshot.
- 2 Keep a way back in: one SSH session open; rescue mode as the fallback (ch. 3).
- 3 Update (ch. 5, "Updates"), then confirm Uptime Kuma is all green within a few minutes.

Monthly routine

- ☐ `df -h /` and `free -h` look healthy
- ☐ `sudo apt update && sudo apt upgrade -y` (security updates also run automatically)
- ☐ Update Coolify and, if needed, Traefik, during a low-traffic window
- ☐ Check that backups ran, and restore one to a test copy now and then
- ☐ Glance at Uptime Kuma, GitHub Actions runs, and `sudo fail2ban-client status sshd`
- ☐ Coolify API token: renew if it expires within the next 2 weeks

11 Extras

Other things the VPS can run alongside your sites. Always keep 4.5 GB of RAM free for the system, Coolify and builds.

Minecraft server (Docker Compose in Coolify)

Coolify → + New → Docker Compose (empty). Don't give it a domain (it isn't a website). In Cloudflare, use

A mc → YOUR_SERVER_IP with **DNS only**.

```
services:
  minecraft:
    image: itzg/minecraft-server
    ports:
      - "25565:25565"
    environment:
      EULA: "TRUE"
      TYPE: "PAPER"           # lighter than VANILLA, same gameplay
      VERSION: "LATEST"
      MEMORY: "3G"
      MOTD: "My server"
      DIFFICULTY: "normal"
      ENABLE_WHITELIST: "TRUE"
      WHITELIST: "YourPseudo,FriendPseudo"
      OPS: "YourPseudo"
    volumes:
      - minecraft-data:/data
    deploy:
      resources:
        limits:
          memory: 3500M       # Java heap + ~0.5–1 GB
      restart: unless-stopped
volumes:
  minecraft-data:
```

```
sudo ufw allow 25565/tcp
sudo docker ps --format "table {{.Names}}\t{{.Status}}" | grep -i minecraft # "Up 3 days" = no
nightly restarts
sudo docker exec -it <name> rcon-cli      # console: whitelist add Pseudo, op Pseudo...
```

Back up the world outside OVH

A provider's included backup often keeps only the last 24 h. Add a Coolify **Storage backup** of the `minecraft-data` volume to R2 (ch. 10): weekly, containers stopped during the archive, fewer copies kept (e.g. 4) since worlds are larger. Coolify's Docker cleanup never touches the volume as long as "Unused volumes: Keep" stays set.

Docker cleanup at 00:00 UTC doesn't restart running containers. A server that kicks everyone at the same hour every night is usually a host's scheduled restart, a backup that stops the server, or (when hosted at home) the ISP's daily reconnection. More than 4–6 GB doesn't help vanilla: the CPU becomes the limit first. Avoid deploying sites while people play. Heavy modpacks (6–10 GB) are better on a separate server.

Other game servers

Game	RAM	Verdict
Half-Life / CS 1.6 (GoldSrc)	100–300 MB	✓ very light
HL2 DM, CS:Source, L4D2	0.5–1 GB	✓
TF2 / Garry's Mod	1–3 GB	✓

Terraria	0.5–1 GB	✓
Valheim / Project Zomboid	2–4 GB	⚠ small groups
Rust, Palworld, ARK, Satisfactory	8–16 GB	✗ too heavy
CS2	2–4 GB, but ~60 GB disk	✗ disk

Install Steam servers with LinuxGSM (linuxgsm.com), or use a web panel like Pterodactyl/Pelican. Source and GoldSrc games use UDP 27015: `sudo ufw allow 27015/udp`. Pick a server region near your players: every 1,000 km adds roughly 10 ms.

Useful self-hosted apps (one click in Coolify)

App	What for	RAM
Uptime Kuma	alerts when a site goes down (ch. 10)	~100 MB
Umami / Plausible	privacy-friendly analytics, a dashboard per client (ch. 10)	~300 MB with Postgres
n8n	automations (forms, Shopify, WhatsApp...), handy for the ERP	300–500 MB
Directus / Strapi / Payload	headless CMS so clients edit their own content	300–500 MB
Invoice Ninja	quotes and invoices	~300 MB
Vaultwarden	password manager for you and client access credentials	very light
MinIO / Gitea / WireGuard	S3 storage, Git server, VPN	light
TeamSpeak / Mumble	voice chat for gaming	~50 MB

12 Setup checklist

The whole setup in order. Tick as you go.

Server

- ☐ VPS ordered, setup sheet started (ch. 0, 2)
- ☐ SSH key, shortcut, custom port made permanent and tested after a reboot (ch. 3–4)
- ☐ SSH hardening file: no passwords, root by key only (ch. 4)
- ☐ Automatic security updates with a quiet reboot time, ufw, fail2ban, swap (ch. 4)

Coolify

- ☐ Installed, admin account created at once, server validated on your SSH port (ch. 5)
- ☐ Dashboard on `https://coolify.example.com` (DNS only), 2FA on
- ☐ GitHub App connected; Docker cleanup set to keep volumes and retained images
- ☐ Notifications (Telegram): deploy failed, backup failed, disk usage; Sentinel on

Domain, Cloudflare, email

- ☐ Records created in Cloudflare; nameservers switched; DNSSEC re-enabled via Cloudflare (ch. 6, 8)
- ☐ SSL Full (strict); Bot Fight Mode off; `server: cloudflare` in curl
- ☐ Mailbox created and tested both ways; SPF, DKIM, DMARC (ch. 9)

Sites and deploys

- ☐ First site deployed with HTTPS on the root domain and www (ch. 7)
- ☐ CI/CD: auto deploy off, API token (deploy only), environment secrets, workflow pushed, both jobs green
- ☐ Token renewal date written down (30 days)

Monitoring and backups

- ☐ Uptime Kuma with Telegram alerts, plus one external monitor watching Kuma (ch. 10)
- ☐ Umami with a renamed tracker script and your own visits excluded
- ☐ R2 bucket, budget alert, Coolify S3 storage; backups of Coolify, every database and every important volume, with retention
- ☐ One restore tested and written down

Going further

- ☐ Performance and SEO in the code: `@nuxtjs/seo`, `@nuxt/image` (WebP/AVIF), 3D models with Draco/Meshopt and KTX2 textures; test on pagespeed.web.dev
- ☐ Conversion tracking: `umami.track('contact-submit')` and UTM-tagged links
- ☐ Optional WAF rule against `/.git`, `/.env`, `/wp-` scanners
- ☐ For client work: the client owns the domain and any paid accounts; you bill a monthly hosting fee that covers monitoring, backups, analytics and updates in maintenance windows

A Quick reference

The commands you'll use most.

Connect

```
ssh-vps          # or: ssh vps
ssh -o IdentitiesOnly=yes \
  -i ~/.ssh/id_ed25519 \
  -p 49222 ubuntu@YOUR_SERVER_IP
```

Health

```
free -h
df -h /
uptime
sudo docker stats --no-stream
sudo docker system df
```

Containers

```
sudo docker ps
sudo docker logs -f --tail 100 <name>
sudo docker restart <name>
sudo docker exec -it <name> sh
sudo docker logs coolify-proxy --tail 30
```

Cleanup

```
sudo docker system prune -af
sudo docker builder prune -af
```

System

```
sudo apt update && sudo apt upgrade -y
ls /var/run/reboot-required
sudo reboot
sudo journalctl -f
sudo journalctl -p err -b
timedatectl
```

SSH & security

```
sudo ss -tlnp | grep sshd
sudo sshd -t && echo OK
sudo sshd -T | grep -E
"^(port|password|permitroot)"
sudo systemctl restart ssh
sudo ufw status verbose
sudo fail2ban-client status sshd
sudo fail2ban-client unban --all
```

DNS checks (from your computer)

```
dig +short example.com
dig +short NS example.com
dig +short sub.example.com @1.1.1.1
curl -sI https://example.com
sudo resolvectl flush-caches
# Firefox: about:networking#dns
# Chrome: chrome://net-internals/#dns
```

Coolify

```
curl -I http://localhost:8000
sudo docker logs --tail 50 coolify
curl -fsSL https://cdn.coollabs.io/\
coolify/install.sh | sudo bash
```

Rescue mode (ch. 3)

```
lsblk
mount /dev/sdb1 /mnt
for d in dev proc sys run; do
  mount --bind /$d /mnt/$d; done
chroot /mnt
# ... fix ...
exit && umount -R /mnt
```

After rescue (your computer)

```
ssh-keygen -R YOUR_SERVER_IP
ssh-keygen -R "[YOUR_SERVER_IP]:49222"
```

B Troubleshooting

Errors met during a real setup, and how they were fixed.

SSH and the server

Symptom	Cause	Fix
Too many authentication failures	SSH tries every stored key	<code>IdentitiesOnly yes</code> / <code>-o IdentitiesOnly=yes -i key</code> ; for password logins <code>-o PubkeyAuthentication=no</code>
Connection refused on your SSH port after <code>sudo reboot</code>	<code>Port</code> line still commented; <code>ssh.socket</code> came back on 22, which ufw blocks (port 22 hangs)	Rescue mode → uncomment the port → disable <code>ssh.socket</code> , enable <code>ssh.service</code> (ch. 3–4)
SSH suddenly times out or is refused	fail2ban banned your IP	try from your phone's hotspot; then <code>sudo fail2ban-client unban --all</code>
REMOTE HOST IDENTIFICATION HAS CHANGED	rescue system's own host key	expected; <code>-o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null</code> , then <code>ssh-keygen -R</code> afterwards
unknown command "sudo" for "ovhcloud"	that's the OVHcloud CLI (<code>>>></code> prompt), not your server	close it; use SSH or rescue mode
permission denied ... <code>docker.sock</code>	<code>ubuntu</code> isn't in the docker group	<code>sudo docker ...</code> or <code>sudo usermod -aG docker \$USER</code>
Coolify: <code>host.docker.internal</code> port 22: Connection refused	SSH isn't on port 22	Servers → localhost → Port 49222 → Validate
Coolify can't validate the server after hardening	<code>PermitRootLogin no</code>	use <code>prohibit-password</code> (ch. 4)
<code>ufw delete</code> → "non-existent rule"	ufw was never enabled	run the full ufw block (ch. 4)
OVH RAM graph at 53% → 99%	Linux disk cache after builds	none needed: check <code>available</code> in <code>free -h</code>

Domains, DNS and Cloudflare

Symptom	Cause	Fix
OVH shows "Transfer locked" after unlocking	cached status	wait 30–60 min / use the direct transfer page
Site shows OVH hosting instead of yours	A/AAAA still point to the free hosting	A → VPS IP, delete AAAA for <code>@ / www</code>
Coolify: "DNS is not configured for ..."	the record doesn't exist yet, or was added at OVH	add the A record in Cloudflare ; check with <code>dig +short ... @1.1.1.1</code>
Firefox <code>NS_ERROR_UNKNOWN_HOST</code> / Server Not Found, while DNS matches	negative DNS cache: you visited before the record existed	<code>about:networking#dns</code> → Clear, <code>sudo resolvectl flush-caches</code> , restart Firefox, or use mobile data
Proxy log: <code>no valid A records found</code> (ACME)	certificate requested before the record existed; Traefik didn't retry	<code>sudo docker restart coolify-proxy</code> ; if it fails again, grey cloud → restart → orange

<code>curl -sI</code> returns <code>HTTP/2 403</code> , ~5 KB HTML	Cloudflare's bot challenge for curl	nothing wrong; browsers pass
"Too many redirects"	Cloudflare SSL mode isn't Full (strict)	SSL/TLS → Full (strict)
curl shows <code>server: nginx</code>	old DNS cache, bypassing Cloudflare	wait ~1 h, <code>resolvectl flush-caches</code>
Email stopped after moving to Cloudflare	mail records were proxied	set them to DNS only
Domain stopped resolving after the DNS switch	old DNSSEC key still published	remove the DS at OVH; re-add Cloudflare's once Active

Builds, CI/CD and apps

Symptom	Cause	Fix
Command <code>"generate"</code> not found / <code>ERR_PNPM_OUTDATED_LOCKFILE</code>	missing script / stale lockfile	ch. 7 table
Deploy job: <code>curl -X GET ""</code> , (3) URL rejected	secrets empty: they're <i>environment</i> secrets and the job didn't declare the environment	add <code>environment: Coolify</code> to the deploy job
Deploy job: <code>error: 403</code>	Cloudflare Bot Fight Mode challenged the GitHub runner (Security → Events: <code>bot_fight_mode</code>)	Bot Fight Mode off and/or <code>coolify</code> record DNS only
Deploy job: <code>error: 405</code>	Coolify's deploy endpoint wants POST	<code>curl -X POST</code>
Deploy job: <code>401</code> (after ~30 days)	API token expired or wrong	new deploy-only token → update <code>COOLIFY_TOKEN</code>
Push doesn't deploy any more	auto deploy is off by design	check GitHub → Actions; or click Deploy in Coolify
Editor: <code>Cannot find name 'process'</code> in <code>nuxt.config.ts</code>	Node types not picked up	<code>pnpm nuxt prepare</code> ; reference <code>.nuxt/tsconfig.node.json</code> in <code>tsconfig</code> ; restart Zed's language server; or <code>import process from 'node:process'</code>
Umami shows no data	site not deployed / env var empty at build / visiting <code>www.</code> not in <code>data-domains</code> / blocker	view source for <code>app.js</code> + ID; write the ID directly; add <code>www.</code> ; check <code>send</code> = 200
Brotli toggle missing	Cloudflare enables it for everyone now	nothing to do

C Glossary

Plain-language definitions of the terms in this guide.

Term	Meaning
VPS	A virtual server: your own slice of a physical machine, with fixed CPU, RAM and disk, and full control (root).
Managed / unmanaged	Whether the host maintains the system for you (managed) or you do it yourself (unmanaged).
vCore	A virtual CPU core, shared with neighbours, so performance can vary slightly.
RAM / disk	RAM is fast working memory, cleared on reboot. The disk is long-term storage for the OS, images, files and logs.
Cache (buff/cache)	Free RAM that Linux uses to keep recent files; given back instantly when apps need it.
Swap	Disk space used as overflow memory. Slower, but it prevents crashes when RAM runs out.
SSH / SSH key	Encrypted remote terminal. A key pair (private on your PC, public on the server) replaces the password.
ssh.socket / ssh.service	Two ways Ubuntu starts SSH: on demand through a socket (default on 26.04) or as an always-running service that reads its port from <code>sshd_config</code> .
Rescue mode / chroot	Bootting a temporary system from the provider to repair your disk; <code>chroot</code> makes that disk act as <code>/</code> so commands change your real server.
Firewall (ufw)	Decides which ports accept incoming connections.
fail2ban	Temporarily bans IPs that fail to log in too many times.
Docker / container / image / volume	Docker runs apps in isolated containers built from images. Volumes hold data that must survive redeloys.
Coolify	Self-hosted dashboard that builds and deploys your apps with Docker.
Reverse proxy (Traefik)	Receives all web traffic on 80/443 and forwards each request to the right app based on the domain; handles SSL.
SNI	Lets the browser announce the domain at the start of an HTTPS connection, so one IP can serve many HTTPS sites.
Build pack (Railpack / Nixpacks)	Detects your project type and builds the Docker image automatically.
Static vs SSR	Static = pre-generated HTML served by nginx. SSR = a Node server renders pages on each request.
Serverless	Code that runs only when called, on the provider's servers: no maintenance, but cold starts and time limits.
CI/CD	Continuous integration / deployment: every push is checked automatically (lint, types, build) and deployed only if the checks pass.
GitHub Actions / workflow / job	GitHub's automation: a workflow file in <code>.github/workflows/</code> defines jobs (<code>ci</code> , <code>deploy</code>) that run on GitHub's machines (runners).
Secret / environment secret	An encrypted value (token, URL) a workflow can use without it appearing in the code. Environment secrets are only given to jobs that declare that environment.

API token	A key that lets a program call Coolify's API. Scoped to "deploy" so a leak can only trigger a redeploy.
Registrar	The company where the domain is registered and renewed (OVH).
Nameservers (NS)	The servers that answer DNS questions for the domain (Cloudflare).
A / AAAA / CNAME	Point a name to an IPv4 / an IPv6 address / another name.
MX / SPF / DKIM / DMARC	Where email is delivered / who may send it / signatures that prove it's genuine / what to do with fakes.
SRV	Points a service (e.g. mail autodiscovery, Minecraft) to a host and port.
TTL / propagation	How long a DNS answer is cached. Changes appear as caches expire: minutes to hours.
Negative caching	Caching the answer "this name doesn't exist". Why a new subdomain can stay invisible on your computer for ~30 min.
Wildcard (<code>*</code>)	A DNS record matching every subdomain that isn't defined explicitly.
DNSSEC / DS record	Cryptographic signing of DNS answers. The DS record at the registrar must match the DNS provider's key.
CDN / WAF / DDoS protection	Servers worldwide caching your files / a web firewall filtering malicious requests / absorbing traffic floods.
Proxied / DNS only	In Cloudflare: traffic passes through Cloudflare (orange) or goes straight to your server (grey).
Bot Fight Mode / Managed Challenge	Cloudflare features that test visitors that look automated; scripts like curl or GitHub runners fail the test (403).
SSL Full (strict)	Cloudflare ↔ server traffic is also encrypted, with a valid certificate on the server (from Coolify).
sslip.io	Free service where <code>anything.IP.sslip.io</code> resolves to that IP, for temporary test URLs.
Uptime Kuma / Sentinel	Kuma checks your sites from outside the apps and alerts when one is down; Sentinel is Coolify's agent for CPU/RAM/disk metrics.
Umami / UTM	Privacy-friendly analytics counting visits through a small script; UTM tags (<code>?utm_source=...</code>) label links so you know which one brought a visit.
R2 / S3 / retention	R2 is Cloudflare's S3-compatible file storage (10 GB free); retention is how many old backups are kept before the oldest is deleted.
Maintenance window	A planned low-traffic time for updates and restarts.