
Point de théorie · hébergement

# Où faire tourner votre application ?

## Du VPS au serverless

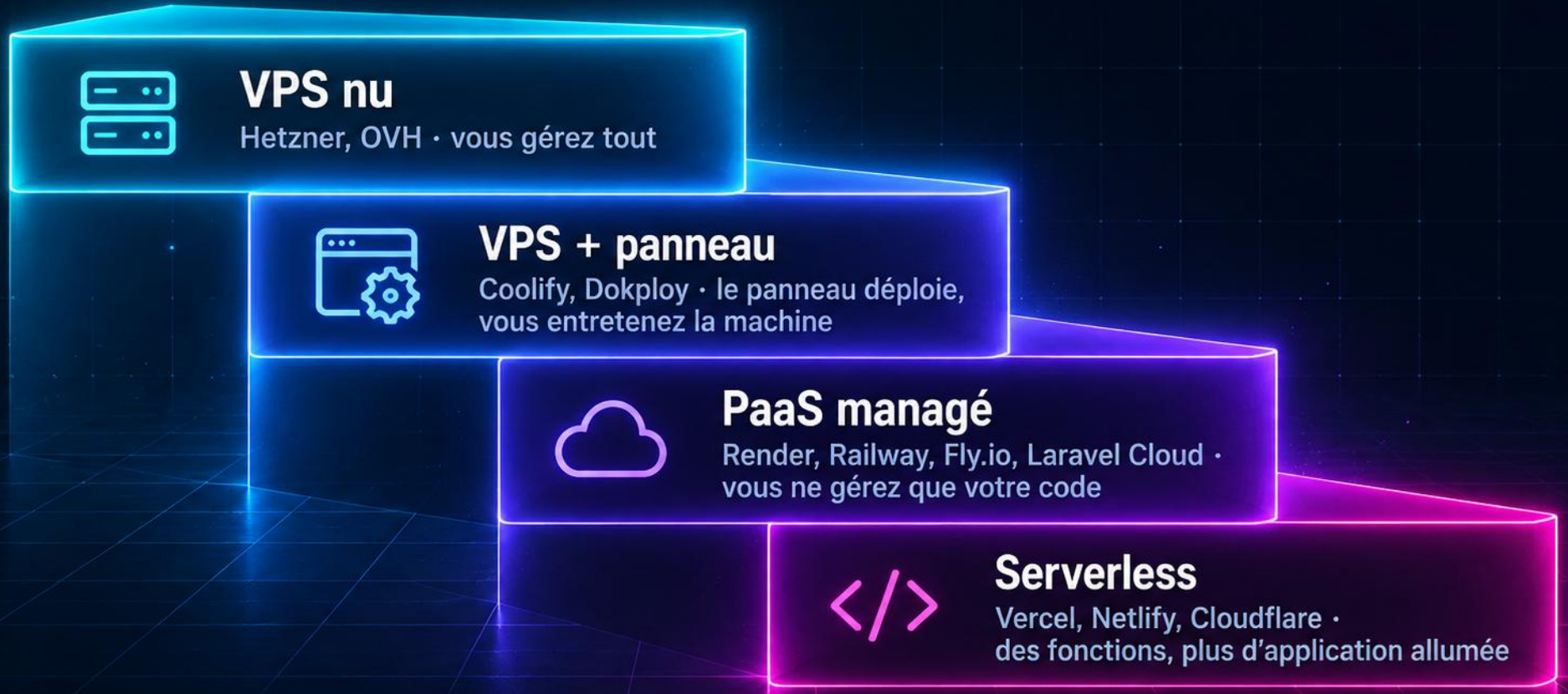
# Qui s’occupe de la machine, et jusqu’où ?

**Votre chat :** un backend, une base, des WebSockets

**Quelque** part, une machine exécute ce code

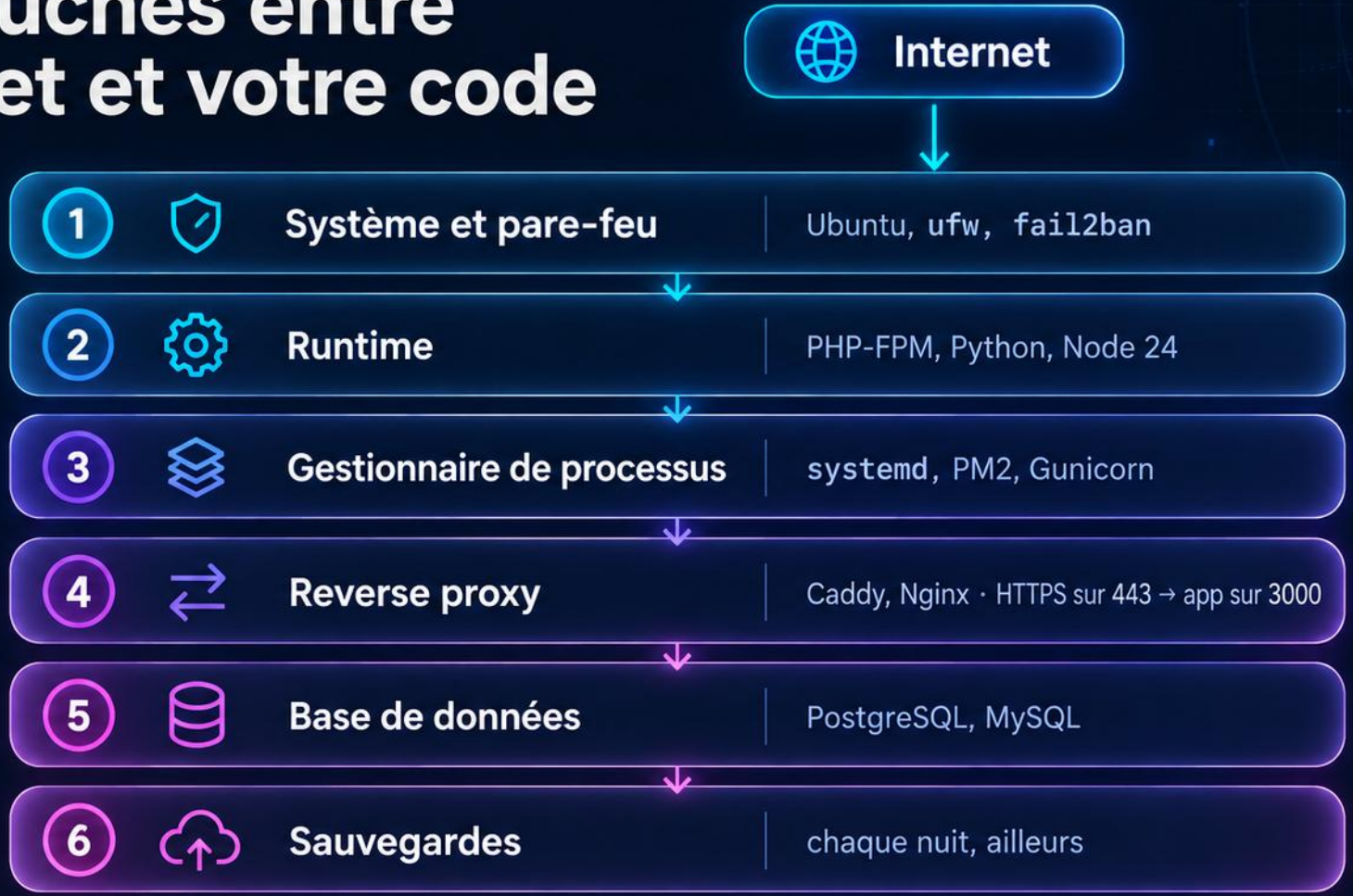
Un hébergeur, ce n'est pas un prix : c'est un partage des tâches entre vous et lui

# Quatre marches, du serveur nu aux fonctions




Plus haut : plus de contrôle, plus de travail. Plus bas : moins de travail, les règles de la plateforme.

# Six couches entre Internet et votre code



# Une soirée pour l'installer, une heure par mois pour l'entretenir

## En échange :

- **Un processus qui reste allumé ·**
  WebSockets, tâches de nuit, variables en mémoire
- **Un disque ·**
  fichiers et base au même endroit
- **Un prix fixe ·**
  4 à 10 € par mois, quel que soit le trafic
- **Le contrôle ·**
  tout installer, tout lire

# Le VPS avec le confort d'un PaaS

- Le panneau tourne sur votre serveur
- Il lit votre dépôt GitHub, construit un conteneur, obtient le HTTPS, crée la base, redéploie à chaque push
- La machine reste la vôtre : mises à jour, disque, sécurité

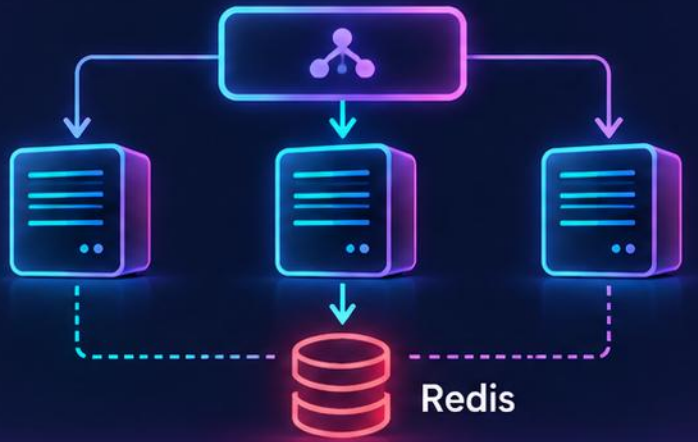
# Une machine plus grosse, ou plusieurs machines

## Vertical

plus de RAM, plus de CPU · simple, cher, plafonné

## Horizontal

un répartiteur de charge, trois serveurs · résilient, complexe




⚠ Le piège : la session du serveur A n'existe pas sur le serveur B

**Dès qu'il y a plusieurs machines, tout état vit en dehors d'elles.**

# Le serveur n’existe qu’au moment de la requête

- **Vous déployez des fonctions,** pas une application
- **Une requête arrive : la plateforme** lance une instance, répond, la détruit plus tard
- **Restaurant ouvert 24 h/24** contre food truck qui vient à la commande
- **Mille utilisateurs à midi :** mille instances, sans rien configurer

# Le build une fois, la requête à chaque visite

**Build** · à chaque push : dépendances, compilation, fichiers statiques vers le CDN, fonctions prêtes

le temps passe →

**Requête** · à chaque visite : la page statique vient du CDN, l'appel d'API réveille une fonction

**Entre deux requêtes, rien ne tourne**

# À froid, à chaud, puis plus rien

# Pas de `listen()`, pas de port, pas de processus qui attend

```
// app/api/messages/route.js
export async function GET() {
  const messages = await sql`SELECT * FROM messages ORDER BY id DESC LIMIT 50`;
  return Response.json(messages);
}
```

Le fichier exporte des fonctions. La plateforme décide quand les appeler.

# Une fonction est faite pour répondre vite et disparaître

**Vercel, plan gratuit, septembre 2026 :**


Les chiffres montent chaque année. La nature des limites ne change pas.

# Du code de VPS dans une fonction

**Un compteur en mémoire**
`1, 2, 1, 3, 1, 1`

→ Redis

**Un fichier écrit sur le disque**
`disparu avec l'instance`

→ `stockage d'objets`

**Un traitement de vingt minutes**
`coupé à 504`

→ `file de tâches + worker`

Même règle que le scaling horizontal : **sortir l'état de la machine.**

# Une connexion longue demande un processus qui dure

- Un WebSocket reste ouvert des heures, une fonction vit quelques secondes
- Vercel l'expérimente depuis 2026 ; en pratique : un service tiers (Pusher, Ably) ou un serveur allumé
- Pour un chat, cette question passe avant le prix

# Quatre questions

1. **Une connexion reste-t-elle ouverte ?** → processus allumé 
2. **Traitements longs ou fichiers ?** → worker + stockage d'objets 
3. **Trafic prévisible ?** → prix fixe · **imprévisible ?** → serverless 
4. **Qui fera la maintenance ?** → personne : PaaS ou Coolify 

# Dans le rapport : la formule, la marche, ce que vous gérez, ce que vous avez écarté


Trois phrases par question du point précédent suffisent.