🐬 Docker — Images prêtes à l'emploi

Objectifs

À la fin de cette page, tu seras capable de :

- Identifier les applications couramment hébergées avec Docker en entreprise
- Déployer une application selfhosted en quelques minutes
- Lire et adapter un `docker-compose.yml` minimal pour chaque service
- Comprendre pourquoi le selfhosting est une alternative aux services cloud payants

---

5 notions-clés

1. **Selfhosting** — Héberger soi-même une application sur son propre serveur plutôt que d'utiliser un service cloud tiers
2. **Image officielle** — Image maintenue par l'éditeur du logiciel lui-même, généralement plus fiable et mieux documentée
3. **Reverse proxy** — Serveur (Nginx, Traefik) qui reçoit les requêtes et les redirige vers le bon conteneur selon le nom de domaine
4. **Portainer** — Interface web de gestion Docker, indispensable en entreprise pour administrer les conteneurs sans CLI
5. **Persistance** — Toutes ces applications ont besoin de volumes : les données (fichiers, bases de données, config) doivent survivre aux mises à jour

---

Pourquoi héberger ses propres services ?

Les entreprises et les administrateurs systèmes utilisent Docker pour héberger des alternatives à des services cloud payants :

Service cloud	Alternative selfhosted
Google Drive / OneDrive	**Nextcloud**
Slack / Teams	**Mattermost** / **Rocket.Chat**
GitHub / GitLab.com	**Gitea** / **Forgejo**
Notion / Confluence	**Wiki.js** / **BookStack**
Jira	**Plane** / **Taiga**
LastPass / 1Password	**Vaultwarden**
Datadog / New Relic	**Grafana + Prometheus**

Avantages :

- 💰 Coût réduit (pas d'abonnement par utilisateur)
- 🔒 Données hébergées sur son infrastructure (souveraineté)
- 🔧 Configuration fine selon les besoins
- 🌱 Empreinte carbone maîtrisée (mutualisation sur ses propres serveurs)

Prérequis typique : un serveur Linux (physique ou VM), Docker installé, un nom de domaine.

Comment lire les exemples de cette page

Chaque service propose un `docker-compose.yml` minimal. Pour l'utiliser :

```
mkdir nom-du-service && cd nom-du-service
# Copier le docker-compose.yml ci-dessous dans ce dossier
# Créer un .env si nécessaire
docker compose up -d
```

Les données sont toujours stockées dans un volume nommé ou un dossier `./data/` local.

🗂️ Gestion Docker — Portainer

Portainer est une interface web qui permet de gérer Docker entièrement depuis un navigateur. C'est l'outil numéro 1 des admins qui gèrent plusieurs serveurs Docker.

Fonctionnalités : gérer les conteneurs, images, volumes, réseaux, stacks Compose, logs, statistiques — sans jamais ouvrir un terminal.

```yaml
services:
  portainer:
    image: portainer/portainer-ce:latest
    container_name: portainer
    restart: unless-stopped
    ports:
      - "9443:9443"    # Interface HTTPS
      - "9000:9000"    # Interface HTTP (si HTTPS non disponible)
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock   # accès au moteur Docker
      - portainer_data:/data

volumes:
  portainer_data:
```

Accès : **https://localhost:9443** (ignorer l'alerte certificat auto-signé)
🐬 Hub : portainer/portainer-ce

⚠️ La ligne `/var/run/docker.sock:/var/run/docker.sock` donne à Portainer un accès complet au moteur Docker. À ne jamais exposer sur Internet sans authentification forte.

☁️ Stockage de fichiers — Nextcloud

Nextcloud est l'alternative open source à Google Drive / OneDrive. Il offre : stockage de fichiers, partage, synchronisation, agenda, contacts, visioconférence, et des centaines d'applications complémentaires.

```yaml
services:
  db:
    image: mariadb:10.11
    container_name: nextcloud_db
    restart: unless-stopped
    environment:
      MYSQL_ROOT_PASSWORD: ${DB_ROOT_PASS}
      MYSQL_DATABASE: nextcloud
      MYSQL_USER: nextcloud
      MYSQL_PASSWORD: ${DB_PASS}
    volumes:
      - db_data:/var/lib/mysql

  nextcloud:
    image: nextcloud:28
    container_name: nextcloud_app
    restart: unless-stopped
    ports:
      - "8080:80"
    environment:
      MYSQL_HOST: db
      MYSQL_DATABASE: nextcloud
      MYSQL_USER: nextcloud
      MYSQL_PASSWORD: ${DB_PASS}
      NEXTCLOUD_ADMIN_USER: admin
      NEXTCLOUD_ADMIN_PASSWORD: ${ADMIN_PASS}
    volumes:
      - nextcloud_data:/var/www/html
    depends_on:
      - db

volumes:
  db_data:
  nextcloud_data:
```

Accès : **http://localhost:8080**
🐬 Hub : nextcloud

💬 Communication — Mattermost

Mattermost est l'alternative open source à Slack. Il permet la messagerie d'équipe en temps réel avec des canaux, des messages directs, le partage de fichiers et des intégrations.

```yaml
services:
  db:
    image: postgres:15
    container_name: mattermost_db
    restart: unless-stopped
    environment:
      POSTGRES_DB: mattermost
      POSTGRES_USER: mmuser
      POSTGRES_PASSWORD: ${DB_PASS}
```

```
    volumes:
      - db_data:/var/lib/postgresql/data

  mattermost:
    image: mattermost/mattermost-team-edition:latest
    container_name: mattermost_app
    restart: unless-stopped
    ports:
      - "8065:8065"
    environment:
      MM_SQLSETTINGS_DRIVERNAME: postgres
MM_SQLSETTINGS_DATASOURCE: >
  postgres://mmuser:${DB_PASS}@db:5432/mattermost?sslmode=disable
    volumes:
      - mattermost_data:/mattermost/data
      - mattermost_logs:/mattermost/logs
    depends_on:
      - db

volumes:
  db_data:
  mattermost_data:
  mattermost_logs:
```

Accès : **http://localhost:8065**
🐬 Hub : mattermost/mattermost-team-edition

📋 Gestion de projet — Plane

Plane est l'alternative open source à Jira / Linear. Il permet de gérer des projets avec des issues, des sprints, des cycles et des feuilles de route.

```
services:
  db:
    image: postgres:15
    restart: unless-stopped
    environment:
      POSTGRES_DB: plane
      POSTGRES_USER: plane
      POSTGRES_PASSWORD: ${DB_PASS}
    volumes:
      - db_data:/var/lib/postgresql/data

  redis:
    image: redis:7
    restart: unless-stopped

  plane:
    image: makeplane/plane-frontend:latest
    restart: unless-stopped
    ports:
      - "3000:3000"
    environment:
```

```
      DATABASE_URL: postgresql://plane:${DB_PASS}@db:5432/plane
      REDIS_URL: redis://redis:6379
    depends_on:
      - db
      - redis

volumes:
  db_data:
```

💡 Plane a une installation un peu plus complexe (plusieurs services : frontend, backend, worker). Consulter la documentation officielle sur github.com/makeplane/plane pour l'installation complète.

Accès : **http://localhost:3000**

📚 Documentation & Wiki — Wiki.js

Wiki.js est un wiki moderne et élégant. Il supporte Markdown, l'éditeur visuel, la gestion des droits, la recherche full-text et des dizaines de modules.

```yaml
services:
  db:
    image: postgres:15
    container_name: wikijs_db
    restart: unless-stopped
    environment:
      POSTGRES_DB: wiki
      POSTGRES_USER: wikijs
      POSTGRES_PASSWORD: ${DB_PASS}
    volumes:
      - db_data:/var/lib/postgresql/data

  wiki:
    image: ghcr.io/requarks/wiki:2
    container_name: wikijs_app
    restart: unless-stopped
    ports:
      - "3000:3000"
    environment:
      DB_TYPE: postgres
      DB_HOST: db
      DB_PORT: 5432
      DB_NAME: wiki
      DB_USER: wikijs
      DB_PASS: ${DB_PASS}
    depends_on:
      - db

volumes:
  db_data:
```

Accès : **http://localhost:3000**
🐬 Hub : requarks/wiki

📚 Documentation — BookStack

BookStack est une plateforme de documentation organisée en **livres** → **chapitres** → **pages**. Idéale pour la documentation technique interne d'une équipe ou d'une entreprise.

```yaml
services:
  db:
    image: mysql:8.0
    container_name: bookstack_db
    restart: unless-stopped
    environment:
      MYSQL_ROOT_PASSWORD: ${DB_ROOT_PASS}
      MYSQL_DATABASE: bookstack
      MYSQL_USER: bookstack
      MYSQL_PASSWORD: ${DB_PASS}
    volumes:
      - db_data:/var/lib/mysql

  bookstack:
    image: lscr.io/linuxserver/bookstack:latest
    container_name: bookstack_app
    restart: unless-stopped
    ports:
      - "6875:80"
    environment:
      APP_URL: http://localhost:6875
      DB_HOST: db
      DB_DATABASE: bookstack
      DB_USERNAME: bookstack
      DB_PASSWORD: ${DB_PASS}
    volumes:
      - bookstack_data:/config
    depends_on:
      - db

volumes:
  db_data:
  bookstack_data:
```

Accès : **http://localhost:6875** — Compte par défaut : `admin@admin.com` / `password`
🐬 Hub : linuxserver/bookstack

🔐 Gestionnaire de mots de passe — Vaultwarden

Vaultwarden est une implémentation légère et compatible avec **Bitwarden** (client officiel). Chaque employé peut l'utiliser via l'extension navigateur ou l'application mobile Bitwarden, en pointant vers le serveur interne.

```yaml
services:
  vaultwarden:
    image: vaultwarden/server:latest
    container_name: vaultwarden
    restart: unless-stopped
    ports:
      - "8083:80"
    volumes:
      - vw_data:/data
    environment:
      ADMIN_TOKEN: ${ADMIN_TOKEN}   # token pour accéder à /admin
      SIGNUPS_ALLOWED: "false"      # désactiver les inscriptions publiques

volumes:
  vw_data:
```

Accès : **http://localhost:8083**
Admin : **http://localhost:8083/admin**
🐬 Hub : vaultwarden/server

🔁 Automatisation — n8n

n8n est un outil d'automatisation de workflows (comme Zapier, mais selfhosted). Il permet de connecter des APIs, des bases de données, des services web et d'automatiser des tâches répétitives, via une interface visuelle de type "nœuds".

```yaml
services:
  n8n:
    image: n8nio/n8n:latest
    container_name: n8n
    restart: unless-stopped
    ports:
      - "5678:5678"
    environment:
      N8N_BASIC_AUTH_ACTIVE: "true"
      N8N_BASIC_AUTH_USER: admin
      N8N_BASIC_AUTH_PASSWORD: ${N8N_PASS}
    volumes:
      - n8n_data:/home/node/.n8n

volumes:
  n8n_data:
```

Accès : **http://localhost:5678**
🐬 Hub : n8nio/n8n

📊 Monitoring — Uptime Kuma

Uptime Kuma surveille la disponibilité de tes services (sites web, APIs, serveurs) et t'envoie des alertes (email, Slack, Telegram...) quand quelque chose tombe. Interface moderne et claire.

```yaml
services:
  uptime-kuma:
    image: louislam/uptime-kuma:latest
    container_name: uptime-kuma
    restart: unless-stopped
    ports:
      - "3001:3001"
    volumes:
      - kuma_data:/app/data

volumes:
  kuma_data:
```

Accès : **http://localhost:3001**
🐋 Hub : louislam/uptime-kuma

📊 Monitoring avancé — Grafana + Prometheus

Prometheus collecte des métriques (CPU, RAM, réseau, requêtes...) depuis des services. **Grafana** les affiche sous forme de tableaux de bord visuels. Ensemble, ils forment le standard du monitoring en entreprise.

```yaml
services:
  prometheus:
    image: prom/prometheus:latest
    container_name: prometheus
    restart: unless-stopped
    volumes:
      - ./prometheus.yml:/etc/prometheus/prometheus.yml
      - prometheus_data:/prometheus
    ports:
      - "9090:9090"

  grafana:
    image: grafana/grafana:latest
    container_name: grafana
    restart: unless-stopped
    ports:
      - "3000:3000"
    environment:
      GF_SECURITY_ADMIN_PASSWORD: ${GRAFANA_PASS}
    volumes:
      - grafana_data:/var/lib/grafana
    depends_on:
      - prometheus

volumes:
```

```
  prometheus_data:
  grafana_data:
```

Accès Grafana : **http://localhost:3000** — Identifiant : `admin` / `${GRAFANA_PASS}`
🐳 Hub : grafana/grafana

Hébergement Git — Gitea

Gitea est un serveur Git léger (alternative à GitLab). Il permet d'héberger ses propres dépôts Git avec interface web, gestion des utilisateurs, pull requests, issues et CI/CD basique.

```yaml
services:
  db:
    image: mysql:8.0
    container_name: gitea_db
    restart: unless-stopped
    environment:
      MYSQL_ROOT_PASSWORD: ${DB_ROOT_PASS}
      MYSQL_DATABASE: gitea
      MYSQL_USER: gitea
      MYSQL_PASSWORD: ${DB_PASS}
    volumes:
      - db_data:/var/lib/mysql

  gitea:
    image: gitea/gitea:latest
    container_name: gitea_app
    restart: unless-stopped
    ports:
      - "3000:3000"    # Interface web
      - "2222:22"      # SSH pour les push Git
    environment:
      GITEA__database__DB_TYPE: mysql
      GITEA__database__HOST: db:3306
      GITEA__database__NAME: gitea
      GITEA__database__USER: gitea
      GITEA__database__PASSWD: ${DB_PASS}
    volumes:
      - gitea_data:/data
    depends_on:
      - db

volumes:
  db_data:
  gitea_data:
```

Accès : **http://localhost:3000**
🐳 Hub : gitea/gitea

💻 VS Code dans le navigateur — code-server

code-server est VS Code accessible depuis un navigateur web. Idéal pour coder depuis n'importe quel poste (y compris une tablette) ou pour donner accès à un environnement de développement à des stagiaires.

```yaml
services:
  code-server:
    image: lscr.io/linuxserver/code-server:latest
    container_name: code-server
    restart: unless-stopped
    ports:
      - "8443:8443"
    environment:
      PASSWORD: ${CODE_PASSWORD}
      SUDO_PASSWORD: ${SUDO_PASSWORD}
    volumes:
      - code_data:/config
      - ./projets:/config/workspace   # dossier projets accessible dans VS Code

volumes:
  code_data:
```

Accès : **http://localhost:8443**
🐬 Hub : linuxserver/code-server

🗺️ Récapitulatif

Application	Catégorie	Image	Port défaut
Portainer	Gestion Docker	`portainer/portainer-ce`	9443
Nextcloud	Stockage fichiers	`nextcloud`	8080
Mattermost	Messagerie d'équipe	`mattermost/mattermost-team-edition`	8065
Plane	Gestion de projet	`makeplane/plane-frontend`	3000
Wiki.js	Documentation	`requarks/wiki`	3000
BookStack	Documentation	`linuxserver/bookstack`	6875
Vaultwarden	Mots de passe	`vaultwarden/server`	8083
n8n	Automatisation	`n8nio/n8n`	5678
Uptime Kuma	Monitoring simple	`louislam/uptime-kuma`	3001
Grafana	Monitoring avancé	`grafana/grafana`	3000
Gitea	Hébergement Git	`gitea/gitea`	3000
code-server	IDE en ligne	`linuxserver/code-server`	8443

💡 **Ressource** : Le site awesome-selfhosted.net répertorie des centaines d'applications selfhosted par catégorie. C'est la référence de la communauté.