

Gitea

- [Présentation et Initialisation](#)
- [Templates](#)
- [Configurer une clé SSH](#)

Présentation et Initialisation

“ **Gitea** est un service de développement logiciel tout-en-un, facile à utiliser et auto-hébergé. Il inclut l'hébergement Git, la révision de code, la collaboration en équipe, un registre de paquets et des fonctionnalités d'intégration continue et de déploiement continu (CI/CD). Il est similaire à GitHub, Bitbucket et GitLab.

À l'origine, **Gitea** est issu d'un fork de *Gogs*, mais la quasi-totalité du code a été remaniée.

Objectifs - Déploiement Docker Compose

Résumé : Rédiger une stack Gitea auto-hébergée, pensée pour tourner derrière un Apache déjà en place sur un serveur, avec accès Git en SSH.

Prérequis :

- PostgreSQL comme base de données (conteneur dédié, réseau interne isolé)
- Le serveur SSH intégré de Gitea exposé sur le port hôte **2222** (le port 22 est déjà pris par le sshd système)
- Un accès HTTP publié uniquement sur `127.0.0.1:3000`, car **Apache** (natif sur l'hôte) fait déjà reverse-proxy/TLS pour les autres vhosts de ce serveur et gèrera aussi celui-ci
- Domaine cible : `git.mondomain.ovh` (email admin : placeholder `admin@mondomain.ovh`, à changer)
- Inscription publique désactivée (comptes créés par l'admin uniquement)
- Gitea Actions et mailer désactivés pour le moment (documentés en option, pas activés)
- Toute la configuration versionnable en clair, tous les secrets externalisés dans un `.env` non commité

Architecture

Internet

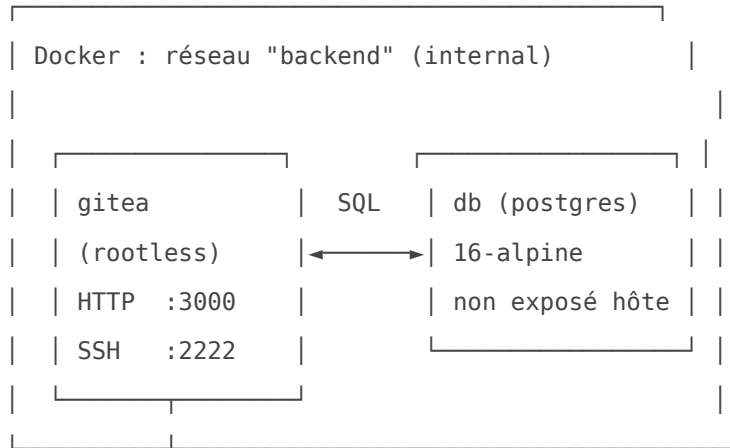
| 80/443 (TLS géré par Apache, certs Certbot existants)



```
| Apache (natif, sur l'hôte) |  
| vhost git.mondomain.ovh   |
```



| proxy_pass http://127.0.0.1:3000



| port 2222 publié sur toutes les interfaces



Clients Git (ssh://git@git.mondomain.ovh:2222/...)

Installation

1. Préparer les configuration

Nous proposons le docker-compose et le modèle app.ini par défaut.

Copier les deux fichiers dans votre dossier pour les configurer.

```
cd gitea/  
cp <from Wiki>.env.exemple .env  
cp <from Wiki>docker-compose.yaml  
cp <from Wiki>app.ini app.ini  
chmod 600 .env
```

Le template de la configuration App est accessible [ici](#), sans aucun secret.

Le template de la configuration docker-compose est accessible [ici](#), sans aucun secret.

Le template de la configuration .env est accessible [ici](#), sans aucun secret.

2. Configurer les variables & les secrets

Générer un mot de passe PostgreSQL fort :

```
openssl rand -base64 32
```

Générer les trois secrets internes de Gitea (aucun conteneur ne tourne encore, on utilise l'image pour exécuter juste la commande) :

```
docker run --rm gitea/gitea:1.27-rootless gitea generate secret SECRET_KEY
docker run --rm gitea/gitea:1.27-rootless gitea generate secret INTERNAL_TOKEN
docker run --rm gitea/gitea:1.27-rootless gitea generate secret JWT_SECRET
```

Reporter chaque valeur générée dans le `.env` ou directement dans le fichier `app.ini` (`POSTGRES_PASSWORD`, `GITEA_SECRET_KEY`, `GITEA_INTERNAL_TOKEN`, `GITEA_OAUTH2_JWT_SECRET`).

App.ini c'est le fichier dans lequel l'entrypoint Gitea écrit directement les secrets au démarrage (mot de passe DB, `SECRET_KEY`, `INTERNAL_TOKEN`, `JWT_SECRET`). Il permet également de configurer Gitea pour les logs et la gestion des inscriptions des comptes etc...

Avant le premier démarrage vérifier dans les 3 fichiers (`docker-compose.yml` / `.env` / `app.ini`) :

- Les secrets sont remplacés par ceux générés précédemment
- Le domaine `git.mondomain.ovh` et l'email admin placeholder (`admin@mondomain.ovh`) sont remplacés

3. Démarrer la stack

```
docker compose up -d db
docker compose ps          # attendre que "db" soit "healthy"
docker compose up -d gitea
docker compose logs -f gitea
```

4. Créer le premier compte administrateur

L'inscription publique étant désactivée, le premier compte se crée en ligne de commande :

```
docker compose exec gitea gitea admin user create \
  --username admin \
  --password 'UN_MOT_DE_PASSE_FORT' \
  --email admin@mondomain.ovh \
  --admin \
  --must-change-password=false
```

Vérifier en local avant de brancher Apache :

```
curl -I http://127.0.0.1:3000/
```

Templates

“ Ci dessous les templates des 3 fichiers de configurations utile pour la stack Gitea en docker compose.

docker-compose.yaml

```
name: gitea

services:
  db:
    image: postgres:16-alpine
    container_name: gitea-db
    restart: unless-stopped
    security_opt:
      - no-new-privileges:true
    cap_drop:
      - ALL
    cap_add:
      # Requis uniquement par postgres pour initdb / changement d'UID au démarrage.
      - CHOWN
      - DAC_OVERRIDE
      - SETUID
      - SETGID
    environment:
      POSTGRES_DB: ${POSTGRES_DB:-gitea}
      POSTGRES_USER: ${POSTGRES_USER:-gitea}
      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD:?La variable POSTGRES_PASSWORD doit être définie dans .env}
    volumes:
      - gitea_db_data:/var/lib/postgresql/data
    networks:
      - backend
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U ${POSTGRES_USER:-gitea} -d ${POSTGRES_DB:-gitea}"]
      interval: 10s
```

```

    timeout: 5s
    retries: 5
    start_period: 10s
logging:
    driver: json-file
    options:
        max-size: "10m"
        max-file: "3"
deploy:
    resources:
        limits:
            cpus: "1.0"
            memory: 512M

gitea:
    # Image "rootless" : le conteneur ne tourne jamais en root, pas de chown au démarrage.
    # 🔍 Vérifier régulièrement la dernière version stable sur
    hub.docker.com/r/gitea/gitea/tags
    image: gitea/gitea:1.27-rootless
    container_name: gitea
    restart: unless-stopped
    # cap_drop:
    #   - ALL
    environment:
        # --- Non-secret, utile de garder ici plutôt que de dupliquer dans app.ini ---
        GITEA__database__DB_TYPE: postgres
        GITEA__database__HOST: db:5432
        GITEA__database__NAME: ${POSTGRES_DB:-gitea}
        GITEA__database__USER: ${POSTGRES_USER:-gitea}
        # --- Secrets injectés depuis .env, fusionnés dans app.ini par l'entrypoint Gitea ---
        # (mécanisme officiel "environment-to-ini", ne modifie pas le fichier monté sur disque)
        GITEA__database__PASSWD: "${POSTGRES_PASSWORD:?La variable POSTGRES_PASSWORD doit être
définie dans .env}"
        GITEA__security__SECRET_KEY: "${GITEA_SECRET_KEY:?Générez la valeur avec 'gitea generate
secret SECRET_KEY'}"
        GITEA__security__INTERNAL_TOKEN: "${GITEA_INTERNAL_TOKEN:?Générez la valeur avec 'gitea
generate secret INTERNAL_TOKEN'}"
        GITEA__oauth2__JWT_SECRET: "${GITEA_OAUTH2_JWT_SECRET:?Générez la valeur avec 'gitea
generate secret JWT_SECRET'}"
    volumes:

```

```

- ./data:/var/lib/gitea
# ⚠ Pas de ":ro" ici : l'entrypoint Gitea (environment-to-ini) écrit directement dans
ce
# fichier au démarrage pour y fusionner les valeurs GITEA__* (dont les secrets). C'est
donc
# config/app.ini (gitignoré, non versionné) qui reçoit le résultat – voir
config/app.ini.example
# pour le template versionné sans secret.
- ./config/app.ini:/etc/gitea/app.ini
ports:
- "3000:3000"
- "2222:2222"
networks:
- backend
- web
depends_on:
db:
condition: service_healthy
healthcheck:
test: ["CMD", "wget", "--no-verbose", "--tries=1", "--spider",
"http://127.0.0.1:3000/api/healthz"]
interval: 30s
timeout: 10s
retries: 3
start_period: 30s

networks:
# Réseau isolé : "db" n'y a accès à rien d'autre, et n'est jamais routable vers l'extérieur.
backend:
internal: true
# Réseau non-interne : nécessaire uniquement pour que les ports publiés de "gitea"
(3000/2222)
# soient réellement mappés sur l'hôte (Docker ne publie pas de port pour un conteneur qui
# n'est connecté qu'à un réseau "internal: true"). "db" n'y est volontairement pas connecté.
web:

volumes:
gitea_data:
gitea_db_data:

```

App.ini

```
; ▲ Fichier généré/complété au runtime – NE PAS VERSIONNER (voir .gitignore).
; Ce fichier est monté en écriture dans le conteneur : au démarrage, l'entrypoint Gitea
; (environment-to-ini) y écrit directement les valeurs GITEA__section__CLE définies dans
; docker-compose.yml/.env (mot de passe DB, SECRET_KEY, INTERNAL_TOKEN, JWT_SECRET...).
; Après le premier démarrage, ce fichier contient donc des secrets en clair.
; Le template versionné (sans secret) se trouve dans config/app.ini.example – c'est lui
; qu'il faut modifier pour changer la configuration fonctionnelle, puis recopier ici.
APP_NAME = Gitea
RUN_MODE = prod
RUN_USER = git
WORK_PATH = /var/lib/gitea

[repository]
ROOT = /var/lib/gitea/git/repositories
DEFAULT_PRIVATE = private
DEFAULT_BRANCH = main
DISABLE_HTTP_GIT = false

[server]
PROTOCOL = http
DOMAIN = git.mondomain.ovh
ROOT_URL = https://git.mondomain.ovh/
HTTP_ADDR = 0.0.0.0
HTTP_PORT = 3000
; Serveur SSH intégré à Gitea (pas de passthrough OpenSSH de l'hôte).
START_SSH_SERVER = true
SSH_DOMAIN = git.mondomain.ovh
; Port annoncé aux utilisateurs (clone URLs) et port d'écoute interne du conteneur.
; Correspond au mapping "2222:2222" du docker-compose (le port hôte 22 reste au sshd système).
SSH_PORT = 2222
SSH_LISTEN_PORT = 2222
LFS_START_SERVER = true
OFFLINE_MODE = true
LFS_JWT_SECRET =

[database]
DB_TYPE = postgres
HOST = db:5432
```

```
NAME = gitea
USER = gitea
PASSWD = CHANGE_ME

; NAME / USER / PASSWD sont fournis par les variables d'environnement GITEA__database__*
; (voir docker-compose.yml). Ne pas les redéfinir ici pour éviter toute divergence.

[security]
; Empêche l'assistant d'installation web de s'afficher : la config est déjà complète.
INSTALL_LOCK = true
; SECRET_KEY / INTERNAL_TOKEN fournis via l'environnement (GITEA__security__*).
COOKIE_SECURE = true
REVERSE_PROXY_LIMIT = 1
; Apache tourne nativement sur le même hôte : seul 127.0.0.1 est une source de confiance
; pour les en-têtes X-Forwarded-*.
REVERSE_PROXY_TRUSTED_PROXIES = 127.0.0.1
; Un admin de dépôt ne peut pas éditer les hooks Git côté serveur (surface de RCE réduite).
DISABLE_GIT_HOOKS = true
DISABLE_WEBHOOKS = false
MIN_PASSWORD_LENGTH = 12
PASSWORD_COMPLEXITY = lower,upper,digit,spec
INTERNAL_TOKEN =
SECRET_KEY =

[service]
; Comptes créés uniquement par un administrateur (voir README) : pas d'auto-inscription.
DISABLE_REGISTRATION = true
REQUIRE_SIGNIN_VIEW = true
REGISTER_EMAIL_CONFIRM = false
ENABLE_NOTIFY_MAIL = false
DEFAULT_KEEP_EMAIL_PRIVATE = true
DEFAULT_ALLOW_CREATE_ORGANIZATION = true
NO_REPLY_ADDRESS = noreply.git.mondomain.ovh

[openid]
ENABLE_OPENID_SIGNIN = false
ENABLE_OPENID_SIGNUP = false

[session]
PROVIDER = db
COOKIE_NAME = gitea_session
```

```
COOKIE_SECURE = true
SESSION_LIFE_TIME = 86400

[log]
MODE = console, file
LEVEL = Info
ROOT_PATH = /var/lib/gitea/log

[api]
; Réduit la surface d'information exposée publiquement ; réactivable si besoin d'intégrations
API.
ENABLE_SWAGGER = false
MAX_RESPONSE_ITEMS = 50

[other]
SHOW_FOOTER_VERSION = false
SHOW_FOOTER_TEMPLATE_LOAD_TIME = false

[mailer]
; Désactivé pour le moment (pas de réinitialisation de mot de passe par email).
; Pour l'activer : ENABLED = true puis ajouter PROTOCOL/SMTP_ADDR/SMTP_PORT/USER
; et injecter GITEA__mailer__PASSWD via l'environnement (jamais en clair ici).
ENABLED = false

[actions]
; Gitea Actions (CI/CD) désactivé pour cette installation minimale.
ENABLED = false

[metrics]
; Désactivé par défaut : à activer uniquement derrière une authentification/IP restreinte.
ENABLED = false

[oauth2]
JWT_SECRET =
```

.env.exemple

```
# Copier ce fichier vers .env puis renseigner de vraies valeurs :
#   cp .env.exemple .env && chmod 600 .env
#
```

```
# ⚠ .env ne doit JAMAIS être commité (voir .gitignore). Toutes les valeurs ci-dessous  
# sont des secrets ou quasi-secrets.
```

```
# --- PostgreSQL ---
```

```
POSTGRES_DB=gitea
```

```
POSTGRES_USER=gitea
```

```
# Générer un mot de passe fort, par exemple : openssl rand -base64 32
```

```
POSTGRES_PASSWORD=CHANGE_ME
```

```
# --- Secrets internes Gitea ---
```

```
# Générer chaque valeur avec le binaire Gitea, par exemple :
```

```
# docker run --rm gitea/gitea:1.27-rootless gitea generate secret SECRET_KEY
```

```
# docker run --rm gitea/gitea:1.27-rootless gitea generate secret INTERNAL_TOKEN
```

```
# docker run --rm gitea/gitea:1.27-rootless gitea generate secret JWT_SECRET
```

```
GITEA_SECRET_KEY=CHANGE_ME
```

```
GITEA_INTERNAL_TOKEN=CHANGE_ME
```

```
GITEA_OAUTH2_JWT_SECRET=CHANGE_ME
```

Configurer une clé SSH

Ce guide explique comment configurer une clé SSH pour cloner, pousser et gérer tous vos dépôts Gitea sans mot de passe.

1. Une seule clé suffit pour tous vos dépôts

En SSH, la clé publique est rattachée à votre **compte utilisateur Gitea**, pas à un dépôt en particulier. Une fois ajoutée une seule fois dans **Paramètres** → **Clés SSH/GPG**, elle vous donne accès en SSH à :

- tous les dépôts dont vous êtes propriétaire,
- tous les dépôts d'organisation où vous avez des droits (lecture/écriture selon vos permissions),
- avec les mêmes droits que ceux définis dans l'interface web (une clé SSH ne donne pas plus de droits qu'un compte n'en a déjà).

Inutile donc de créer une clé par dépôt.

2. Générer une clé SSH (si vous n'en avez pas déjà une)

```
ssh-keygen -t ed25519 -C "votre_email@example.com" -f ~/.ssh/id_ed25519_gitea
```

- `ed25519` : algorithme moderne, recommandé (plus court et au moins aussi sûr que RSA).
- Une **passphrase** est demandée : renseignez-en une (voir §6, bonnes pratiques).
- Si vous avez déjà une clé SSH que vous souhaitez réutiliser, passez directement à l'étape 3 avec `~/.ssh/id_ed25519` (ou le nom de votre clé existante) à la place de `~/.ssh/id_ed25519_gitea` dans la suite de ce guide.

3. Ajouter la clé publique à votre compte Gitea

Afficher la clé publique à copier :

```
cat ~/.ssh/id_ed25519_gitea.pub
```

Dans l'interface web de Gitea :

1. **Paramètres du compte** (icône profil, en haut à droite) → **Clés SSH/GPG**
2. **Ajouter une clé** → coller le contenu de `id_ed25519_gitea.pub`
3. Donner un nom explicite à la clé (ex : `laptop-perso`, `poste-bureau`) — utile pour la révoquer précisément plus tard sans casser l'accès de vos autres machines
4. Valider

4. Configurer le client SSH (`~/.ssh/config`)

Le serveur SSH intégré de Gitea écoute sur le port **2222** (et non 22). Un alias dans `~/.ssh/config` évite d'avoir à le préciser à chaque commande :

```
Host git.mondomain.ovh
  HostName git.mondomain.ovh
  Port 2222
  User git
  IdentityFile ~/.ssh/id_ed25519_gitea
  IdentitiesOnly yes
```

```
chmod 600 ~/.ssh/config
```

5. Tester la connexion

```
ssh -T git@git.mondomain.ovh
```

Réponse attendue :

```
Hi <votre_nom_utilisateur>! You've successfully authenticated, but Gitea does not provide
shell access.
```

C'est normal : Gitea confirme l'authentification mais ne donne volontairement pas de shell.

Sans l'alias `~/.ssh/config` configuré à l'étape 4 :

```
ssh -T -p 2222 git@git.mondomain.ovh
```

6. Cloner, pousser, committer

Cloner un nouveau dépôt (avec l'alias configuré) :

```
git clone git@git.mondomain.ovh:USER/REP0.git
```

Sans alias :

```
git clone ssh://git@git.mondomain.ovh:2222/USER/REP0.git
```

Basculer un dépôt existant (cloné en HTTP) vers SSH :

```
git remote set-url origin git@git.mondomain.ovh:USER/REP0.git
```

Ensuite, `git add`, `git commit`, `git push`, `git pull` fonctionnent normalement : l'authentification est entièrement gérée par la clé SSH, plus besoin de nom d'utilisateur ni de mot de passe/token.

7. Bonnes pratiques (DevSecOps)

- **Une clé par appareil**, pas une clé unique copiée sur toutes vos machines : en cas de perte ou de compromission d'un appareil, vous pouvez révoquer uniquement sa clé sans couper l'accès des autres.
- **Toujours une passphrase** sur la clé privée, combinée à `ssh-agent` pour ne pas la retaper à chaque commande :

```
eval "$(ssh-agent -s)"  
ssh-add ~/.ssh/id_ed25519_gitea
```

- La clé **privée** (`id_ed25519_gitea`, sans `.pub`) ne doit jamais quitter votre machine : vérifier `chmod 600 ~/.ssh/id_ed25519_gitea` et `chmod 700 ~/.ssh`.
- Nommez explicitement chaque clé ajoutée dans Gitea (étape 3) : l'interface affiche aussi la date de dernière utilisation, utile pour repérer une clé oubliée/inutilisée à révoquer.
- **Révoquer immédiatement** une clé en cas de perte/vol de l'appareil correspondant : Paramètres → Clés SSH/GPG → supprimer la clé concernée.

8. Dépannage

Symptôme	Cause probable	Solution
Permission denied (publickey)	Clé non ajoutée dans Gitea, ou mauvaise clé utilisée	<code>ssh -vT git@git.mondomain.ovh</code> pour voir quelle clé est proposée ; vérifier qu'elle correspond bien à celle ajoutée dans Gitea

Symptôme	Cause probable	Solution
Permission denied (publickey) malgré une clé correcte	Plusieurs clés dans l'agent SSH, la mauvaise est essayée en premier	Ajouter <code>IdentitiesOnly yes</code> dans <code>~/.ssh/config</code> (voir §4)
<code>ssh: connect to host ... port 22: Connection refused</code>	Le port <code>2222</code> n'est pas utilisé (alias non configuré ou oublié dans l'URL)	Utiliser l'alias <code>~/.ssh/config</code> , ou <code>ssh://git@git.mondomain.ovh:2222/...</code>
Timeout / <code>Connection refused</code> sur le port <code>2222</code>	Port fermé côté pare-feu ou conteneur non démarré	<code>sudo ufw status</code> côté serveur ; <code>docker compose ps</code> (le port <code>2222</code> doit apparaître)
<code>Bad permissions</code> / clé ignorée par SSH	Droits trop ouverts sur <code>~/.ssh</code> ou la clé privée	<code>chmod 700 ~/.ssh && chmod 600 ~/.ssh/id_ed25519_gitea</code>
<code>WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!</code> qui revient à chaque redémarrage du conteneur	Clés hôte SSH du serveur (pas de votre poste) illisibles par le conteneur — voir §9	Corriger les permissions de <code>data/data/ssh/</code> côté serveur (voir §9)

Clé refusé après redémarrage du service

Côté serveur : la clé hôte change à chaque redémarrage du conteneur

Symptôme : l'avertissement *"REMOTE HOST IDENTIFICATION HAS CHANGED"* réapparaît après chaque redémarrage de `gitea`, même en supprimant l'entrée dans `known_hosts` et en faisant confiance à la nouvelle clé à chaque fois — le fingerprint proposé change encore la fois suivante.

Cause : les fichiers de clés hôte SSH persistées (`data/data/ssh/gitea.rsa`, `ssh_host_ecdsa_key`, `ssh_host_ed25519_key` ...) appartiennent à `root:root` en mode `600` au lieu de `1000:1000` (l'UID `git` du conteneur `rootless`). Gitea échoue silencieusement à charger ces clés persistées — visible dans les logs :

```
docker compose logs gitea | grep -i -A1 "Adding SSH host key"
# [I] Adding SSH host key: /var/lib/gitea/data/ssh/gitea.rsa
# [E] Failed to set Host Key. open /var/lib/gitea/data/ssh/gitea.rsa: permission denied
```

Le serveur SSH démarre quand même, mais avec **une clé hôte générée en mémoire, différente à chaque redémarrage du conteneur** — d'où l'avertissement qui revient sans cesse, même après avoir fait confiance à la clé précédente : elle n'est jamais la même deux fois.

Ce cas survient typiquement après une restauration/migration manuelle où le dossier `ssh/` a été copié avec `sudo` séparément du reste des données (voir aussi `BACKUP.md`, §7, sur la correspondance UID à respecter côté serveur).

Correction, côté serveur :

```
sudo chown -R 1000:1000 data/data/ssh
docker compose restart gitea
```

```
# Vérifier que le chargement réussit désormais (plus de ligne [E])
docker compose logs gitea | grep -i -A1 "Adding SSH host key"
```

Puis côté client, une dernière fois (la clé hôte restera stable à partir de maintenant) :

```
ssh-keygen -f ~/.ssh/known_hosts -R "[git.mondomain.ovh]:2222"
ssh -T -p 2222 git@git.mondomain.ovh
```