

Outils

- Configurer le MCP GitLab sur code-server

Configurer le MCP GitLab sur code-server

1. Objectif

Configurer et authentifier le serveur MCP GitLab dans OpenAI Codex, exécuté dans un conteneur Docker `code-server` hébergé sur Ubuntu Server, avec un navigateur Windows.

L'authentification utilise OAuth 2.0 et un tunnel SSH pour rendre accessible le callback OAuth de Codex sans exposer de ports supplémentaires sur Internet.

Architecture

Windows

|

└─ Chrome / navigateur

| └─ http://127.0.0.1:23721

|

└─ Tunnel SSH

|

| Connexion au domaine SSH, port 3307

|



Box Internet

|

└─ Redirection TCP 3307

|



Ubuntu Server

|

└─ Serveur SSH

|

└─ Docker : 127.0.0.1:23720



Tableau des ports

Port	Emplacement	Utilisation
3307	Box Internet	Port SSH public
23720	Ubuntu Server	Port Docker exposé uniquement en localhost
23721	Conteneur Docker	Port d'écoute de socat
23721	Windows	Port local du tunnel SSH
23722	Conteneur Docker	Listener OAuth de Codex

Sécurité : seul le port SSH 3307 doit être accessible depuis Internet. Les ports 23720 , 23721 et 23722 ne doivent pas être exposés sur la box.

2. Configuration initiale (une seule fois)

2.1. Configurer Docker Compose

Dans le service code-server du fichier docker-compose.yml , vérifier la présence de :

```
services:
  code-server:
```

```
ports:
  - "8218:8080"
  - "127.0.0.1:23720:23721"
```

Le port `23720` est accessible uniquement depuis la boucle locale du serveur Ubuntu.

Le port `23721` correspond au port interne sur lequel `socat` écoutera.

Vérifier également que le volume de configuration Codex est persistant :

```
volumes:
  - /home/darkmat/.docker/config/code-server/codex:/home/coder/.codex
```

Si la configuration Docker a été modifiée, appliquer les changements depuis le dossier contenant le fichier Compose :

```
docker compose up -d code-server
```

Docker Compose recrée le conteneur si sa configuration a changé.

2.2. Configurer Codex

Depuis le terminal intégré de `code-server`, ouvrir :

```
nano ~/.codex/config.toml
```

Ajouter les paramètres suivants au niveau racine du TOML, avant toute section `[section]` :

```
# MCP OAuth callback
mcp_oauth_callback_port = 23722
mcp_oauth_callback_url = "http://127.0.0.1:23721/callback"
```

Explications :

- `mcp_oauth_callback_port` : port réel utilisé par Codex pour écouter le retour OAuth.
- `mcp_oauth_callback_url` : adresse communiquée à GitLab pour rediriger le navigateur après authentification.

Codex peut ajouter automatiquement un identifiant à la fin du chemin `/callback`.

Vérifier également la déclaration du serveur MCP GitLab :

```
[mcp_servers.gitlab-http]  
url = "https://gitlab.darkmat.fr/api/v4/mcp"
```

Si le MCP GitLab n'existe pas encore, il peut être créé avec :

```
codex mcp add gitlab-http \  
--url "https://gitlab.darkmat.fr/api/v4/mcp"
```

Important : utiliser le transport HTTP natif de Codex plutôt que `mcp-remote`.

3. Procédure d'authentification OAuth

À utiliser lors de la première connexion ou lorsqu'une nouvelle authentification interactive GitLab est nécessaire.

Étape 1 — Lancer socat dans code-server

Ouvrir un premier terminal dans `code-server` :

```
socat \  
TCP-LISTEN:23721,bind=0.0.0.0,reuseaddr,fork \  
TCP:127.0.0.1:23722
```

Cette commande :

- Écoute sur le port `23721` du conteneur.
- Accepte les connexions arrivant depuis le réseau Docker.
- Redirige les connexions vers `127.0.0.1:23722`, utilisé par Codex.
- Permet de traiter plusieurs connexions grâce à `fork`.

Conserver ce terminal ouvert pendant toute l'authentification.

Étape 2 — Lancer le tunnel SSH depuis Windows

Ouvrir PowerShell sur Windows :

```
ssh -N -p 3307 -L 127.0.0.1:23721:127.0.0.1:23720 darkmat@darkmat.fr
```

Remplacer `<DOMAINE_SSH>` par le nom de domaine pointant vers la box Internet.

Paramètres :

Option	Signification
<code>-N</code>	N'exécute aucune commande distante
<code>-p 3307</code>	Utilise le port SSH public
<code>-L</code>	Crée une redirection de port local
<code>127.0.0.1:23721</code>	Port local utilisé sur Windows
<code>127.0.0.1:23720</code>	Destination sur Ubuntu Server

Conserver également cette fenêtre PowerShell ouverte.

Étape 3 — Vérifier le tunnel SSH

Dans un second PowerShell Windows :

```
Test-NetConnection 127.0.0.1 -Port 23721
```

Résultat attendu :

```
ComputerName    : 127.0.0.1
RemotePort      : 23721
TcpTestSucceeded : True
```

Ce résultat confirme que le port local est joignable. Il ne garantit pas, à lui seul, que l'ensemble de la chaîne OAuth fonctionne.

Étape 4 — Lancer l'authentification MCP GitLab

Ouvrir un second terminal dans `code-server` :

```
codex mcp login gitlab-http
```

Codex affiche une URL d'autorisation GitLab :

Authorize `gitlab-http` by opening this URL in your browser:

`https://gitlab.darkmat.fr/oauth/authorize?...`

Ouvrir cette URL dans le navigateur Windows.

S'authentifier sur GitLab puis autoriser l'application OAuth.

Après validation, le navigateur est redirigé vers une adresse de ce type :

`http://127.0.0.1:23721/callback/<identifiant>?code=...`

Ne pas partager cette URL complète : elle contient un code d'autorisation temporaire.

Étape 5 — Confirmer la réussite

Le navigateur doit afficher :

Authentication complete. You may close this window.

Dans le terminal `code-server` :

Successfully logged in to MCP server 'gitlab-http'.

Ces deux messages confirment que le callback OAuth a été reçu et que Codex a terminé l'authentification.

4. Vérifier le fonctionnement du MCP GitLab

4.1. Vérifier la configuration

Dans `code-server` :

`codex mcp list`

Puis :

```
codex mcp get gitlab-http
```

Le MCP doit être activé et déclaré avec l'URL :

```
https://gitlab.darkmat.fr/api/v4/mcp
```

4.2. Vérifier depuis Codex

Démarrer une nouvelle session interactive :

```
codex
```

Puis saisir :

```
/mcp
```

Vérifier que `gitlab-http` apparaît avec un statut `connected` et que des outils sont disponibles.

4.3. Tester un appel réel

Demander à Codex :

```
Utilise le MCP GitLab pour lister les projets auxquels  
mon compte GitLab a accès.
```

Si Codex utilise les outils MCP GitLab et retourne les projets, la connexion est opérationnelle.

4.4. Supprimer l'ancienne configuration mcp-remote

Si l'ancienne connexion `gitlab` basée sur `mcp-remote` est encore présente :

```
codex mcp list
```

Une fois la connexion `gitlab-http` vérifiée, supprimer uniquement l'ancienne :

```
codex mcp remove gitlab
```

Ne pas supprimer `gitlab-http`.

5. Nettoyage après authentification

Lorsque l'authentification a réussi :

1. Fermer la fenêtre de callback dans Chrome.
2. Arrêter le tunnel SSH Windows avec `Ctrl+C`.
3. Arrêter `socat` avec `Ctrl+C`.

Les identifiants OAuth sont normalement conservés par Codex et peuvent être réutilisés pour les connexions suivantes.

Le tunnel SSH et `socat` ne sont nécessaires que lorsqu'une authentification interactive OAuth doit à nouveau être effectuée.

Un simple redémarrage de Codex ne nécessite donc normalement pas de relancer ces deux services.

6. Réauthentification — Procédure rapide

Si le MCP GitLab nécessite une nouvelle authentification :

1. Dans code-server, lancer socat :

```
socat \  
TCP-LISTEN:23721,bind=0.0.0.0,reuseaddr,fork \  
TCP:127.0.0.1:23722
```

2. Dans PowerShell Windows, ouvrir le tunnel SSH :

```
ssh -N -p 3307 -L 127.0.0.1:23721:127.0.0.1:23720 darkmat@<DOMAINE_SSH>
```

3. Dans code-server, lancer le login GitLab :

```
codex mcp login gitlab-http
```

4. Dans le navigateur Windows :

- Ouvrir l'URL d'autorisation GitLab.
- Valider l'accès OAuth.
- Attendre `Authentication complete`.

5. Vérifier :

```
codex mcp list
```

Puis `/mcp` dans une nouvelle session Codex.

6. Arrêter le tunnel SSH et socat.

7. Dépannage

Erreur ERR_CONNECTION_REFUSED dans Chrome

Si le navigateur est redirigé vers :

```
http://127.0.0.1:38317/callback/...
```

Codex utilise probablement un port OAuth dynamique.

Vérifier :

```
mcp_oauth_callback_port = 23722  
mcp_oauth_callback_url = "http://127.0.0.1:23721/callback"
```

Ces paramètres doivent figurer au niveau racine de `config.toml`.

Relancer ensuite l'authentification.

Erreur sur 127.0.0.1:23721

Vérifier successivement :

Sur Windows :

```
Test-NetConnection 127.0.0.1 -Port 23721
```

Sur Ubuntu Server :

```
docker port code-server
```

Vérifier que Docker expose :

```
23721/tcp -> 127.0.0.1:23720
```

Dans code-server :

```
command -v socat  
ss -lntp
```

Vérifier que `socat` écoute sur `23721` et que Codex écoute sur `127.0.0.1:23722` pendant l'authentification.

Erreur Address already in use

Un autre processus utilise déjà l'un des ports.

Vérifier dans `code-server` :

```
ss -lntp | grep -E '23721|23722'
```

Sous Windows :

```
Get-NetTCPConnection -LocalPort 23721 -ErrorAction SilentlyContinue
```

Fermer l'ancien tunnel SSH ou le relais `socat` s'ils sont encore actifs, sans interrompre un service non identifié.

MCP GitLab : failed (0 tools)

Ce message signifie que Codex n'a pas pu initialiser les outils du serveur.

Vérifier :

```
codex mcp list
codex mcp get gitlab-http
```

Causes possibles :

- Authentification OAuth expirée ou invalide.
- Endpoint GitLab MCP indisponible.
- Problème réseau entre le conteneur et GitLab.
- Permissions GitLab insuffisantes.
- Problème de compatibilité MCP.

Si nécessaire, refaire la procédure OAuth complète.

Le tunnel SSH ne démarre pas

Vérifier que le port SSH `3307` est accessible et que la redirection de la box est correcte.

Pour obtenir davantage d'informations :

```
ssh -v -N -p 3307 -L 127.0.0.1:23721:127.0.0.1:23720 darkmat@<DOMAINE_SSH>
```

Vérifier également que le serveur SSH autorise les redirections TCP (`AllowTcpForwarding`).

8. Rappels de sécurité

- Ne jamais exposer les ports OAuth `23720`, `23721` ou `23722` sur Internet.
- Conserver le mapping Docker sur `127.0.0.1:23720`.
- Utiliser SSH comme transport chiffré entre Windows et Ubuntu.
- Préférer l'authentification SSH par clé.
- Ne pas partager les URL OAuth complètes contenant un paramètre `code`.
- Protéger les identifiants OAuth conservés par Codex.
- Éviter d'autoriser inutilement des opérations d'écriture GitLab via les outils MCP.

9. Références

- [Documentation GitLab — GitLab MCP server](#)
- [Documentation OpenAI — MCP dans Codex](#)
- [Configuration Codex — Paramètres OAuth](#)
- [Documentation socat](#)

En résumé : Docker et Codex se configurent une seule fois. Pour toute nouvelle authentification interactive, il suffit de lancer `socat`, d'ouvrir le tunnel SSH Windows, puis d'exécuter `codex mcp login gitlab-http`. Aucun autre port n'est à ouvrir sur la box Internet.