OpenAI Codex *mémo*

Les commandes, raccourcis et fichiers à connaître pour **OpenAI Codex**, l'agent de développement d'OpenAI (CLI v0.157, septembre 2026). Voir aussi la fiche **OpenAI Codex** et la synthèse **des avis**.

Famille Agents de développement · **Éditeur** OpenAI · **Licence** Apache-2.0 (CLI) · application propriétaire
Note du wiki 13/15 · **Commandes vérifiées le** 25 septembre 2026
Fiche, notes et avis aitoolswiki.viellard.fr/outils/agents-de-developpement/openai-codex

Installation

`curl -fsSL https://chatgpt.com/codex/install.sh | sh`	macOS, Linux (installateur officiel ; relancer la même commande pour mettre à jour)
`powershell -ExecutionPolicy ByPass -c "irm https://chatgpt.com/codex/install.ps1 | iex"`	Windows PowerShell
`npm install -g @openai/codex` `brew install --cask codex`	npm ou Homebrew

`codex login`
`cd mon-projet`
`codex`
	Se connecter (compte ChatGPT recommandé), puis lancer dans un projet

Inclus dans les forfaits ChatGPT (le forfait gratuit se limite à GPT-6 Luna dans l'application de bureau) ou facturé au token avec une clé d'API. Autres surfaces : extension VS Code (compatible Cursor et Windsurf), mode Codex de l'application de bureau ChatGPT (`codex app`) et chatgpt.com/codex.

Lancer une session

`codex`	Session interactive (TUI).
`codex "explique ce dépôt"`	Session interactive avec un premier prompt.
`codex -i capture.png "corrige ce rendu"`	Joint une ou plusieurs images au premier message.
`codex -m gpt-6-luna`	Choisit le modèle : **`gpt-6-sol`** (défaut), **`gpt-6-luna`**, **`gpt-6-astra`**.
`codex -C services/api`	Démarre dans un autre dossier de travail.

Lancer une session (suite)

`codex --add-dir ../lib`	Autorise l'écriture dans un dossier supplémentaire.
`codex -s read-only -a on-request`	Choisit le sandbox et la politique d'approbation (voir plus bas).
`codex --search`	Recherche web en direct au lieu du cache.
`codex -p rapide`	Ajoute le profil **`~/.codex/rapide.config.toml`** à la configuration.
`codex --oss --local-provider ollama`	Modèle local via Ollama ou LM Studio.
`codex -c model_reasoning_effort=high`	Surcharge ponctuelle d'une clé de configuration.
`codex resume` / **`codex resume --last`**	Sélecteur de sessions, ou reprise de la dernière du dossier (`--all` pour tous les dossiers).
`codex fork --last`	Crée une nouvelle session à partir de la précédente, sans toucher à l'original.
`codex app .`	Ouvre le dossier dans l'application de bureau ChatGPT (mode Codex).
`codex login --device-auth`	Connexion par code, sans navigateur (serveur distant, SSH).
`codex login status`	Affiche le mode d'authentification ; code 0 si connecté.
`codex update`	Met la CLI à jour.
`codex doctor`	Diagnostic de l'installation, de la config, de l'authentification et de git.
`codex features list`	Liste les feature flags et leur état.

Scripts et CI

`codex exec "corrige les tests qui échouent"`	Tâche non interactive (alias : codex e)
`cat consignes.md | codex exec --json -o resultat.md -`	Lire la consigne sur stdin, événements JSONL, dernier message dans un fichier

Lancer une session (suite)

`codex exec --output-schema schema.json "liste les dépendances obsolètes"`	Imposer la forme de la réponse finale avec un schéma JSON
`codex exec --sandbox workspace-write --ephemeral "mets à jour le changelog"`	CI : écriture limitée au dépôt, session non enregistrée sur disque
`codex exec resume --last "ajoute aussi les tests"`	Reprendre la dernière exécution avec une consigne de suivi
`printenv OPENAI_API_KEY | codex login --with-api-key`	Authentification par clé d'API

Tâches dans le cloud

`codex cloud`	sélecteur interactif
`codex cloud exec --env ENV_ID "corrige l'issue 42"`	envoie une tâche
`codex cloud exec --env ENV_ID --attempts 3 "…"`	best-of-N, de 1 à 4 tentatives
`codex cloud list --json`	tâches récentes
`codex apply TASK_ID`	applique le diff dans le dépôt local

Commandes slash

Taper `/` pour afficher la liste, ou `/` suivi de quelques lettres pour filtrer. Les skills activées y apparaissent aussi ; `$nom` appelle une skill directement.

Démarrer un projet

`/init`	Génère un **`AGENTS.md`** de départ dans le dossier courant.
`/permissions`	Change ce que Codex peut faire sans demander (Auto, Read Only…).
`/model`	Choisit le modèle et l'effort de raisonnement.
`/personality`	Style des réponses : plus concis, plus explicatif ou plus collaboratif.
`/mcp`	Liste les outils MCP disponibles (**`/mcp verbose`** pour le détail).
`/skills`	Parcourt et utilise les skills.

Commandes slash (suite)

`/plugins`	Parcourt les plugins installés et installables.
`/apps`	Insère un connecteur dans le prompt (`$app-slug`).
`/hooks`	Consulte, approuve ou désactive les hooks.
`/memories`	Active ou coupe l'utilisation et la création de mémoires.
`/import`	Importe la configuration, les projets et les conversations de Claude Code ou de Cursor.

Pendant une tâche

`/plan` *[tâche]*	Passe en mode plan : Codex propose un plan avant d'implémenter.
`/goal`	Fixe, modifie, met en pause, reprend ou efface un objectif persistant.
`/fast`	Active ou coupe le niveau de service rapide, si le modèle le propose.
`/compact`	Résume la conversation pour libérer du contexte.
`/side` (alias `/btw`)	Discussion annexe éphémère, sans perturber le fil principal.
`/mention`	Joint un fichier ou un dossier à la conversation.
`/ide`	Ajoute le contexte de l'IDE : fichiers ouverts, sélection.
`/ps` / `/stop`	Affiche ou arrête les terminaux d'arrière-plan.
`/approve`	Autorise une nouvelle tentative après un refus de la revue automatique.
`/experimental`	Active des fonctions expérimentales (proxy réseau, empêcher la veille…).

Commandes slash (suite)

Travailler en parallèle

`/agent` (alias `/subagents`)	Passe d'un fil de sous-agent à l'autre pour suivre ou reprendre son travail.
`/fork`	Copie la conversation dans une nouvelle pour tester une autre piste.
`/app`	Continue la session dans l'application de bureau.
`/worktree`	Dans l'application : lance la conversation dans un nouveau git worktree.
`/cloud`	Dans l'application : exécute la conversation dans le cloud.

Les sous-agents se lancent en le demandant : « lance un agent par point, attends-les tous et résume ».

Avant de livrer

`/diff`	Affiche le diff git, fichiers non suivis compris.
`/review`	Revue sans toucher au working tree : branche de base, changements non commités, commit ou consignes libres.
`/copy`	Copie la dernière réponse terminée (aussi `Ctrl+O`).

Entre deux sessions

`/new`	Nouvelle conversation dans la même session de la CLI.
`/clear`	Efface le terminal et repart d'une conversation vierge.
`/resume`	Reprend une conversation enregistrée.
`/rename`	Renomme la conversation.
`/archive`	Archive la session et quitte (la transcription est conservée).
`/delete`	Supprime définitivement la session et quitte.
`/quit` ou `/exit`	Quitte la CLI.

Commandes slash (suite)

En cas de problème

`/status`	Modèle, politique d'approbation, dossiers accessibles en écriture, contexte restant.
`/usage`	Consommation de tokens du compte et limites.
`/debug-config`	Diagnostic des couches de configuration et des contraintes imposées.
`/feedback`	Envoie des logs aux mainteneurs.
`/logout`	Déconnecte le compte (machine partagée).

Personnaliser l'interface

`/statusline` (barre d'état), `/title` (titre du terminal), `/theme` (coloration syntaxique), `/keymap` (raccourcis), `/vim` (mode Vim), `/raw` (défilement brut pour copier), `/pets` (animal de compagnie du terminal).

Raccourcis clavier

Raccourci	Effet
`@`	Cherche un fichier du projet et insère son chemin.
`!commande`	Exécute une commande shell avec le sandbox et les approbations en cours.
`Entrée` pendant que Codex travaille	Injecte une consigne dans le tour en cours.
`Tab` pendant que Codex travaille	Met en file un prompt, une commande slash ou une commande shell pour le tour suivant.
`Esc Esc` sur un champ vide	Modifie le message précédent et crée une nouvelle branche de la conversation à partir de là.
`↑` / `↓`	Rappelle les brouillons précédents.
`Ctrl+R`	Recherche dans l'historique des prompts.
`Ctrl+O`	Copie la dernière réponse terminée.

Raccourcis clavier (suite)

Raccourci	Effet
o dans une demande d'approbation	Ouvre le fil du sous-agent qui la demande.
Ctrl+C	Quitte la session.

Les raccourcis se remappent avec **/keymap** ou dans *[tui.keymap]* de **config.toml**.

Sandbox et approbations

Combinaison	Ce qui passe sans demander	Pour
`-s read-only -a on-request`	Lectures ; toute modification est soumise à approbation.	Explorer, planifier, code inconnu.
`-s workspace-write -a on-request` (preset Auto)	Lectures, éditions et commandes dans le dossier de travail. Codex demande pour sortir du dossier ou accéder au réseau.	Usage courant.
`-s workspace-write -a never`	Tout ce que le sandbox autorise ; ce qui exigerait une approbation échoue.	CI et tâches non surveillées.
`-s danger-full-access`	Fichiers et réseau sans limite.	À éviter : préférer `--add-dir`.
`--yolo`	Tout, sans sandbox ni approbation.	VM ou conteneur isolés uniquement.

Le réseau est coupé par défaut. Pour l'ouvrir en **workspace-write** :

```
[sandbox_workspace_write]
network_access = true
```

Ajouter **features.network_proxy** pour restreindre les destinations à une liste de domaines. La politique **untrusted** a été retirée : pour un projet non fiable, déclarer **trust_level** = `"untrusted"` dans *[projects."/chemin"]* (voir plus bas). **codex** sandbox -- *<commande>* teste une commande sous la même politique que l'agent.

Fichiers de configuration

Fichier	Rôle
~/.codex/config.toml	Réglages utilisateur : modèle, sandbox, MCP, feature flags.
.codex/config.toml	Réglages du projet, chargés seulement si le projet est approuvé.

Fichiers de configuration (suite)

Fichier	Rôle
~/.codex/*<profil>***.config.toml**	Profil chargé avec `-p` *<profil>*.
~/.codex/AGENTS.md	Instructions personnelles pour tous les projets (**AGENTS.override.md** pour une surcharge temporaire).
AGENTS.md / **AGENTS.override.md**	Instructions du projet, lues de la racine jusqu'au dossier courant (32 Kio au total par défaut).
.agents/skills/*<nom>***/SKILL.md**	Skill de projet (**~/.agents/skills/** pour une skill personnelle).
.codex/agents/*<nom>***.toml**	Agent personnalisé du projet (**~/.codex/agents/** pour un agent personnel).
.codex/hooks.json	Hooks du projet (**~/.codex/hooks.json** ou tables *[hooks]* dans **config.toml**).
~/.codex/rules/	Règles **execpolicy** : autoriser, demander ou bloquer une commande.
/etc/codex/config.toml, **requirements.toml**	Configuration système et contraintes imposées par l'administrateur.

Ordre de priorité : options de ligne de commande > **.codex/config.toml** du projet (le plus proche du dossier courant gagne) > profil > **~/.codex/config.toml** > configuration gérée dans le cloud > **/etc/codex/config.toml** > valeurs par défaut.

Fichiers de configuration (suite)

Exemple de config.toml

```
model = "gpt-6-sol"
model_reasoning_effort = "medium"   # low, medium, high,
xhigh, max, ultra selon le modèle
approval_policy = "on-request"
sandbox_mode = "workspace-write"
web_search = "cached"               # "live" (= --search),
"indexed" ou "disabled"
personality = "pragmatic"           # ou "friendly",
"none"
review_model = "gpt-6-sol"          # modèle utilisé par
/review

[agents]
default_subagent_model = "gpt-6-luna"
max_concurrent_threads_per_session = 4

[features]
memories = true                     # expérimental

[projects."/chemin/vers/projet-inconnu"]
trust_level = "untrusted"           # chaque commande
demande une approbation
```

Serveurs MCP

codex mcp add openai-docs --url https://developers.openai.com/mcp	Serveur distant en HTTP streamable
codex mcp add db --env DATABASE_URL=postgres://... -- npx -y *<paquet-mcp>*	Serveur local en stdio (tout ce qui suit -- est la commande à lancer)
codex mcp add sentry --url https://mcp.sentry.dev/mcp --bearer-token-env-var SENTRY_TOKEN	Jeton lu dans une variable d'environnement
Gérer	
codex mcp list	
codex mcp get openai-docs --json	
codex mcp login *<nom>*	OAuth, serveurs HTTP qui le proposent
codex mcp remove db	

Les serveurs sont enregistrés dans **~/.codex/config.toml**.

Skills, plugins et agents personnalisés

$skill-creator **$skill-installer** linear	Dans une session : créer ou installer une skill

Skills, plugins et agents personnalisés (suite)

Plugins (skills + connecteurs + config MCP)	
`codex plugin marketplace add owner/repo`	ou URL Git, SSH, dossier local
`codex plugin list --available --json`	
`codex plugin add mon-plugin@ma-marketplace`	
`codex plugin remove mon-plugin@ma-marketplace`	
`codex plugin marketplace upgrade`	

Agent personnalisé, par exemple `.codex/agents/reviewer.toml` :

```
name = "reviewer"
description = "Relit un diff pour trouver bugs, failles et tests manquants."
developer_instructions = "Ne modifie aucun fichier. Classe les problèmes par gravité avec les références de fichiers."
model = "gpt-6-sol"
model_reasoning_effort = "high"
sandbox_mode = "read-only"
```

Les agents intégrés sont `default`, `worker` et `explorer`. Un agent personnalisé du même nom les remplace.

Hooks

Exemple de `.codex/hooks.json` : contrôler chaque commande shell avant son exécution et lancer une vérification à la fin de chaque tour.

```
{
  "hooks": {
    "PreToolUse": [
      {
        "matcher": "Bash",
        "hooks": [{ "type": "command", "command": "python3 \"$(git rev-parse --show-toplevel)/.codex/hooks/check_bash.py\"", "statusMessage": "Vérification de la commande" }]
      }
    ],
    "Stop": [
      {
        "hooks": [{ "type": "command", "command": "\"$(git rev-parse --show-toplevel)/scripts/verify.sh\"", "timeout": 120 }]
      }
    ]
  }
}
```

Événement	Moment
`SessionStart` / `SessionEnd`	Début ou reprise, puis fin de la session principale.
`UserPromptSubmit`	À l'envoi d'un prompt.

Hooks (suite)

Événement	Moment
`PreToolUse` / `PostToolUse`	Avant et après un appel d'outil.
`PermissionRequest`	Quand Codex demande une approbation.
`PreCompact` / `PostCompact`	Autour de la compaction du contexte.
`SubagentStart` / `SubagentStop`	Démarrage et fin d'un sous-agent.
`Stop`	Fin d'un tour.
`Interrupt`	Interruption d'un tour en cours.

Un hook nouveau ou modifié ne s'exécute qu'après approbation dans `/hooks`. Les hooks peuvent aussi appeler un outil MCP (`"type": "mcp_tool"`).

Revue de code

`/review`	Menu de revue dans la session.
`codex review --uncommitted`	Revue des changements indexés, non indexés et non suivis.
`codex review --base main`	Revue de la branche par rapport à `main`.
`codex review --commit <SHA>`	Revue d'un commit précis.
`codex review "vérifie la gestion des erreurs"`	Revue guidée par des consignes libres.

Sur GitHub

Prérequis : Codex Cloud configuré pour le dépôt et revue de code activée dans les réglages Codex, avec revue automatique de chaque PR en option.

Commentaire sur la PR	Effet
`@codex review`	Revue de la PR, limitée aux problèmes P0 et P1.
`@codex review for issues in the database migration`	Revue centrée sur un point précis.
`@codex security review`	Security Review, plus approfondie (research preview).
`@codex <question ou tâche>`	Pose une question ou lance une tâche.

Les règles propres au dépôt vont dans une section `## Code Review Rules` de l'`AGENTS.md` le plus proche du code concerné.

Conseils d'utilisation

- **Effort `medium` par défaut avec Sol** : `xhigh`, `max` et `ultra` sont lents et épuisent vite les quotas. Les réserver aux problèmes vraiment difficiles.
- **Luna pour les sous-agents** d'exploration, de tests ou de triage. Paralléliser surtout les tâches de lecture : plusieurs agents qui écrivent en même temps créent des conflits.
- **Fixer le périmètre et le critère de fin** : les modèles récents ont tendance à ajouter des vérifications non demandées. Préciser par exemple « un seul test pour ce cas, pas d'infrastructure de test supplémentaire ».
- **`/plan` puis `/goal` pour les tâches longues** : valider le plan avant de lancer un objectif persistant.
- **Commiter souvent et relire toute opération git destructrice** : des utilisateurs ont vu l'agent s'écarter de consignes claires sur ce type d'opération.
- **Préférer `--add-dir` à `danger-full-access`** et réserver `--yolo` à une VM ou un conteneur jetable.
- **Tenir la CLI à jour** (`codex update`) : la version 0.149.0 a corrigé deux évasions de sandbox en août 2026.
- **Garder `AGENTS.md` court et placé au plus près du code** : au-delà de 32 Kio, les instructions sont tronquées (`project_doc_max_bytes`).
- **Surveiller `/status` et `/usage`** : les sous-agents et les efforts élevés multiplient la consommation.
- **Venir de Claude Code ?** `/import` reprend la configuration, les projets et les conversations. Les skills au format `SKILL.md` se réutilisent telles quelles dans `.agents/skills/`.

Sources (documentation officielle)

- Documentation – référence des commandes de la CLI
- Documentation – bases de la configuration
- Documentation – approbations et sécurité
- Documentation – AGENTS.md
- Documentation – skills
- Documentation – sous-agents
- Documentation – hooks
- Documentation – revue des PR GitHub
- README du dépôt openai/codex