Outils IA pour Linux

Guide Complet & RefCard

Synthese - Installation - Trucs & Astuces
Variables d'environnement - Commandes CLI - Lexique IA

Ce document presente en detail les outils d'intelligence artificielle disponibles pour Linux, tels que recenses par TecMint. Pour chaque outil, vous trouverez : une synthese, les instructions d'installation detaillees, les variables d'environnement, les commandes CLI avec options, des trucs et astuces, une refcard, ainsi que les liens de reference. Un lexique complet sur l'IA est inclus en fin de document.

Table des matieres

1. ONLYOFFICE Docs - Suite Bureautique avec IA

SYNTHESE

ONLYOFFICE Docs est une suite bureautique open-source offrant des applications web, desktop et mobiles avec des capacites IA integrees. Elle est compatible avec les formats Microsoft Office (DOCX, XLSX, PPTX) et propose plus de 40 connecteurs pour l'integration de plateformes tierces. Le plugin IA universel supporte ChatGPT, Claude, Gemini, Mistral, DeepSeek, Ollama et GPT4All, permettant la generation de texte, la traduction, la correction grammaticale, la generation d'images et le chat IA directement dans les documents.

INSTALLATION

Plusieurs methodes d'installation sont disponibles :

```
# Snap (toutes distributions)
sudo snap install onlyoffice-desktopeditors

# Debian/Ubuntu - depuis le depot officiel
mkdir -p -m 700 ~/.gnupg
gpg --no-default-keyring --keyring gnupg-ring:/tmp/onlyoffice.gpg \
    --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys CB2DE8E5
chmod 644 /tmp/onlyoffice.gpg
sudo mv /tmp/onlyoffice.gpg /usr/share/keyrings/onlyoffice.gpg
echo "deb [signed-by=/usr/share/keyrings/onlyoffice.gpg] \
    https://download.onlyoffice.com/repo/debian squeeze main" | \
    sudo tee /etc/apt/sources.list.d/onlyoffice.list
sudo apt update && sudo apt install onlyoffice-desktopeditors

# Fedora/RHEL (RPM)
sudo yum install https://download.onlyoffice.com/install/desktop/editors/\
    linux/onlyoffice-desktopeditors.x86_64.rpm

# Flatpak
flatpak install flathub org.onlyoffice.desktopeditors

# Docker (version serveur collaboratif)
docker run -i -t -d -p 80:80 onlyoffice/documentserver
```

VARIABLES D'ENVIRONNEMENT

Les variables suivantes peuvent etre configurees pour le serveur Docker :

```
# Cle JWT pour securiser les communications API
JWT_SECRET=ma_cle_secrete_jwt

# Configuration de la base de donnees
DB_TYPE=postgres
DB_HOST=localhost
DB_PORT=5432
DB_NAME=onlyoffice
DB_USER=onlyoffice
DB_PWD=motdepasse

# Configuration AMQP (RabbitMQ)
AMQP_URI=amqp://guest:guest@localhost

# Activer le mode debug
DEBUG=true
```

```
# Exporter les variables
export JWT_SECRET DB_TYPE DB_HOST DB_PORT DB_NAME
```

TRUCS & ASTUCES

-> Activer le plugin IA : Plugins > Gestionnaire de plugins > Rechercher 'AI' > Installer
-> Raccourci rapide : clic droit sur du texte selectionne pour acceder aux fonctions IA (reformuler, traduire, resumer)
-> Connecter Ollama en local : dans les parametres du plugin IA, choisir 'Ollama' et pointer vers http://localhost:11434
-> Pour utiliser Claude : configurer la cle API Anthropic dans les parametres du plugin
-> Mode hors-ligne : connecter GPT4All ou Ollama pour une utilisation sans Internet
-> Raccourcis clavier utiles : Ctrl+Shift+A (tout selectionner), Ctrl+K (inserer lien), Alt+H (aide)
-> Exporter en PDF : Fichier > Telecharger en > PDF. Ajouter un mot de passe dans les options d'export

COMMANDES CLI (Docker / Serveur)

```
# Demarrer le serveur de documents
docker run -d --name onlyoffice -p 80:80 onlyoffice/documentserver

# Verifier les logs
docker logs onlyoffice

# Acceder au shell du conteneur
docker exec -it onlyoffice bash

# Redemarrer les services internes
supervisorctl restart all

# Verifier l'etat de sante
curl -s http://localhost/healthcheck | jq .

# Sauvegarder la configuration
docker cp onlyoffice:/etc/onlyoffice ./backup_config/
```

LIENS DE REFERENCE

Site officiel : https://www.onlyoffice.com/
Documentation : https://helpcenter.onlyoffice.com/
GitHub : https://github.com/ONLYOFFICE/DocumentServer
Plugin IA : https://www.onlyoffice.com/blog/ai-plugin
API : https://api.onlyoffice.com/
Forum : https://forum.onlyoffice.com/

2. Opera One - Navigateur avec IA Integree

SYNTHESE

Opera One est un navigateur web moderne integrant une IA basee sur les modeles Google Gemini. Il propose les 'Tab Islands' (onglets groupes) pour organiser la navigation, et un assistant IA accessible directement depuis la barre d'outils. L'IA peut analyser le contenu de l'onglet courant ou de plusieurs onglets simultanement, traiter des fichiers (images, videos), et repondre par synthese vocale. Gratuit et sans creation de compte obligatoire.

INSTALLATION

```
# Debian/Ubuntu - depot officiel Opera
wget -qO- https://deb.opera.com/archive.key | \
    sudo gpg --dearmor -o /usr/share/keyrings/opera-browser.gpg
echo "deb [signed-by=/usr/share/keyrings/opera-browser.gpg] \
    https://deb.opera.com/opera-stable/ stable non-free" | \
    sudo tee /etc/apt/sources.list.d/opera-stable.list
sudo apt update && sudo apt install opera-stable

# Fedora/RHEL
sudo rpm --import https://rpm.opera.com/rpmrepo.key
sudo tee /etc/yum.repos.d/opera.repo <<'EOF'
[opera]
name=Opera packages
type=rpm-md
baseurl=https://rpm.opera.com/rpm
gpgcheck=1
gpgkey=https://rpm.opera.com/rpmrepo.key
enabled=1
EOF
sudo dnf install opera-stable

# Snap
sudo snap install opera

# Flatpak
flatpak install flathub com.opera.Opera
```

VARIABLES D'ENVIRONNEMENT

Opera utilise les variables Chromium standard :

```
# Desactiver l'acceleration GPU (si problemes graphiques)
export OPERA_ENABLE_GPU=0

# Forcer le backend Wayland
export OPERA_WAYLAND=1

# Definir un proxy
export http_proxy=http://proxy:8080
export https_proxy=http://proxy:8080

# Repertoire de profil personnalise
export OPERA_PERSONALDIR=$HOME/.config/opera-custom

# Desactiver le sandbox (debug uniquement)
export OPERA_NO_SANDBOX=1
```

TRUCS & ASTUCES

-> Acceder a l'IA : cliquer sur 'Ask AI' en haut a droite, ou utiliser le raccourci Ctrl+/
-> Tab Islands : glisser un onglet sur un autre pour creer un groupe. Renommer et colorer le groupe via clic droit
-> Analyse multi-onglets : ouvrir plusieurs pages de comparaison, puis demander a l'IA de les synthetiser
-> Entree vocale : cliquer sur l'icone micro dans le panneau IA pour dicter vos questions
-> Mode flux de travail : sidebar gauche pour acceder rapidement a Telegram, WhatsApp, Messenger
-> Bloqueur de pubs integre : Parametres > Protection de la vie privee > Bloquer les publicites
-> VPN gratuit integre : Parametres > VPN > Activer (limites de bande passante)
-> Raccourcis : Ctrl+Shift+E (barre laterale), F11 (plein ecran), Ctrl+Shift+N (navigation privee)

COMMANDES CLI

```
# Lancer Opera depuis le terminal
opera --new-window https://example.com

# Mode navigation privee
opera --private

# Desactiver l'acceleration materielle
opera --disable-gpu

# Ouvrir avec un profil specifique
opera --user-data-dir=/tmp/opera-profile

# Mode kiosque (plein ecran sans barre)
opera --kiosk https://example.com

# Activer les flags experimentaux
opera --enable-features=FeatureName

# Afficher la version
opera --version

# Debug avec DevTools ouvert
opera --auto-open-devtools-for-tabs
```

LIENS DE REFERENCE

Site officiel : https://www.opera.com/one
Telechargement : https://www.opera.com/download
Blog IA Opera : https://blogs.opera.com/news/
Documentation : https://help.opera.com/
Communaute : https://forums.opera.com/
Changelog : https://blogs.opera.com/desktop/changelog/

3. Smartcat - Plateforme de Traduction IA

SYNTHESE

Smartcat est une plateforme de traduction professionnelle utilisant des reseaux de neurones et une technologie IA adaptive. Elle supporte plus de 280 langues et permet la traduction de documents, sites web et logiciels. La plateforme propose des memoires de traduction, des glossaires, des flux de travail collaboratifs et une selection automatique du meilleur moteur de traduction. Accessible via navigateur avec extensions disponibles.

INSTALLATION & ACCES

```
# Smartcat est une application web - pas d'installation locale
# Acceder via : https://www.smartcat.com/

# Extension Chrome pour traduction de pages web
# -> Chrome Web Store > Rechercher "Smartcat"

# API CLI via curl pour automatisation
# Inscription gratuite requise pour obtenir une cle API

# Installer l'outil CLI (Node.js)
npm install -g @smartcat/cli

# Ou utiliser l'API REST directement
curl -X GET "https://smartcat.com/api/integration/v2/project/list" \
    -H "Authorization: Basic $(echo -n 'account_id:api_key' | base64)"
```

VARIABLES D'ENVIRONNEMENT

```
# Identifiant de compte Smartcat
export SMARTCAT_ACCOUNT_ID="votre_account_id"

# Cle API pour l'automatisation
export SMARTCAT_API_KEY="votre_api_key"

# URL de base de l'API (defaut: production)
export SMARTCAT_API_URL="https://smartcat.com/api/integration/v2"

# Langue source par defaut
export SMARTCAT_SOURCE_LANG="fr"

# Langues cibles par defaut (separees par des virgules)
export SMARTCAT_TARGET_LANGS="en,de,es"

# Repertoire de travail pour les fichiers
export SMARTCAT_WORKDIR="$HOME/translations"
```

TRUCS & ASTUCES

-> Plan gratuit : 2000 mots/mois en traduction IA. Suffisant pour des tests et petits projets
-> Memoire de traduction : importer vos fichiers TMX existants pour ameliorer la qualite des traductions
-> Glossaire : creer un glossaire de termes techniques pour garantir la coherence terminologique
-> Formats supportes : DOCX, XLSX, PPTX, PDF, HTML, JSON, XLIFF, PO, SRT (sous-titres), YAML
-> Pretranduction : activer la pretranduction automatique pour accelerer le travail des traducteurs humains
-> Webhook : configurer des callbacks pour etre notifie quand une traduction est terminee
-> Selection de moteur : Smartcat choisit automatiquement entre Google, DeepL, et ses propres modeles
-> Revue collaborative : inviter des relecteurs pour valider les traductions avant publication

COMMANDES CLI & API

```
# Lister les projets
curl -s -H "Authorization: Basic $AUTH" \
    "$SMARTCAT_API_URL/project/list" | jq .

# Creer un projet de traduction
curl -X POST "$SMARTCAT_API_URL/project" \
    -H "Authorization: Basic $AUTH" \
    -H "Content-Type: application/json" \
    -d '{
        "name": "Mon Projet",
        "sourceLanguage": "fr",
        "targetLanguages": ["en", "de"],
        "workflowStages": ["Translation"]
    }'

# Uploader un fichier a traduire
curl -X POST "$SMARTCAT_API_URL/project/{id}/document" \
    -H "Authorization: Basic $AUTH" \
    -F "file=@document.docx"

# Telecharger la traduction terminee
curl -s "$SMARTCAT_API_URL/document/export?documentId={id}" \
    -H "Authorization: Basic $AUTH" -o translated.docx

# Obtenir les statistiques d'un projet
curl -s "$SMARTCAT_API_URL/project/{id}/statistics" \
    -H "Authorization: Basic $AUTH" | jq .
```

LIENS DE REFERENCE

Site officiel : https://www.smartcat.com/
API Docs : https://developers.smartcat.com/
Blog : https://www.smartcat.com/blog/
Aide : https://help.smartcat.com/
Tarifs : https://www.smartcat.com/pricing/
Marketplace : https://www.smartcat.com/marketplace/

4. PhotoGlimmer - Retouche Photo IA Locale

SYNTHESE

PhotoGlimmer est un editeur d'images leger utilisant l'IA (Mediapipe et OpenCV) pour ameliorer l'eclairage des personnes dans les photos tout en preservant l'arriere-plan. Entierement local (aucune connexion Internet requise), il garantit la confidentialite des donnees. L'IA detecte automatiquement les sujets humains et permet un ajustement selectif de la luminosite via des curseurs intuitifs. Compatible avec les grands fichiers images.

INSTALLATION

```
# AppImage (toutes distributions Linux)
wget https://github.com/codecliff/PhotoGlimmer/releases/latest/\
    download/PhotoGlimmer-x86_64.AppImage
chmod +x PhotoGlimmer-x86_64.AppImage
./PhotoGlimmer-x86_64.AppImage

# Debian/Ubuntu (.deb)
wget https://github.com/codecliff/PhotoGlimmer/releases/latest/\
    download/photoglimmer_amd64.deb
sudo dpkg -i photoglimmer_amd64.deb
sudo apt install -f   # installer les dependances manquantes

# Depuis les sources (Python)
git clone https://github.com/codecliff/PhotoGlimmer.git
cd PhotoGlimmer
pip install -r requirements.txt
python3 main.py

# Dependances principales
pip install mediapipe opencv-python-headless PyQt5
```

VARIABLES D'ENVIRONNEMENT

```
# Forcer l'utilisation du CPU (si pas de GPU)
export MEDIAPIPE_GPU=0

# Limiter l'utilisation memoire OpenCV
export OPENCV_IO_MAX_IMAGE_PIXELS=178956970

# Repertoire temporaire pour le traitement
export TMPDIR=/tmp/photoglimmer

# Niveau de log (DEBUG, INFO, WARNING, ERROR)
export PHOTOGLIMMER_LOG_LEVEL=INFO

# Backend Qt (xcb pour X11, wayland pour Wayland)
export QT_QPA_PLATFORM=xcb
```

TRUCS & ASTUCES

-> Traitement local uniquement : vos photos ne quittent jamais votre machine, ideal pour la vie privee
-> Grands fichiers : PhotoGlimmer gere les images haute resolution sans probleme de memoire
-> Curseur de luminosite : ajuster finement l'eclairage du premier plan sans toucher au fond
-> Plusieurs personnes : l'IA detecte et traite chaque personne independamment
-> Sauvegarde : exporter en PNG pour conserver la qualite maximale, JPEG pour un fichier leger
-> AppImage portable : copier le fichier sur une cle USB pour l'utiliser sur n'importe quel Linux
-> Raccourci bureau : creer un .desktop pour lancer facilement l'AppImage depuis le menu
-> Si la detection echoue : essayer avec une photo mieux eclairee ou recadrer le sujet

COMMANDES CLI

```
# Lancer depuis l'AppImage
./PhotoGlimmer-x86_64.AppImage

# Lancer depuis les sources
cd PhotoGlimmer && python3 main.py

# Lancer avec debug
python3 main.py --verbose

# Creer un raccourci bureau
cat > ~/.local/share/applications/photoglimmer.desktop << 'EOF'
[Desktop Entry]
Type=Application
Name=PhotoGlimmer
Exec=/chemin/vers/PhotoGlimmer-x86_64.AppImage
Icon=photoglimmer
Categories=Graphics;Photography;
Comment=Retouche photo IA locale
EOF

# Extraire l'AppImage (pour inspection ou modification)
./PhotoGlimmer-x86_64.AppImage --appimage-extract
```

LIENS DE REFERENCE

GitHub : https://github.com/codecliff/PhotoGlimmer
Releases : https://github.com/codecliff/PhotoGlimmer/releases
MediaPipe : https://mediapipe.dev/
OpenCV : https://opencv.org/
Issues : https://github.com/codecliff/PhotoGlimmer/issues

5. Cursor - Editeur de Code avec IA

SYNTHESE

Cursor est un editeur de code base sur un fork de Visual Studio Code, avec des capacites IA natives pour le developpement logiciel. Il comprend l'integralite du codebase et propose l'autocompletion predictive, un chat IA pour poser des questions sur le code, la correction d'erreurs dans le terminal, et un mode Agent pour les taches autonomes. Il supporte GPT-4, Claude Sonnet, Gemini et d'autres modeles. Tous les parametres, extensions et themes VS Code sont importables.

INSTALLATION

```
# Telecharger l'AppImage depuis le site officiel
wget https://downloader.cursor.sh/linux/appImage/x64 -O cursor.AppImage
chmod +x cursor.AppImage
./cursor.AppImage

# Alternative : .deb pour Debian/Ubuntu
wget https://downloader.cursor.sh/linux/deb/x64 -O cursor.deb
sudo dpkg -i cursor.deb

# Alternative : extraction manuelle
wget https://downloader.cursor.sh/linux/appImage/x64 -O cursor.AppImage
chmod +x cursor.AppImage
./cursor.AppImage --appimage-extract
cd squashfs-root && ./cursor

# Creer un lien symbolique
sudo ln -s /chemin/vers/cursor.AppImage /usr/local/bin/cursor

# Importer les parametres VS Code au premier lancement
# -> L'assistant propose automatiquement l'import
```

VARIABLES D'ENVIRONNEMENT

```
# Cle API OpenAI (pour GPT-4)
export OPENAI_API_KEY="sk-..."

# Cle API Anthropic (pour Claude)
export ANTHROPIC_API_KEY="sk-ant-..."

# Cle API Google (pour Gemini)
export GOOGLE_API_KEY="AIza..."

# Configuration du proxy
export HTTP_PROXY="http://proxy:8080"
export HTTPS_PROXY="http://proxy:8080"

# Repertoire d'extensions
export CURSOR_EXTENSIONS_DIR="$HOME/.cursor/extensions"

# Desactiver la telemetrie
export CURSOR_TELEMETRY=0

# Forcer le rendu GPU/Software
export CURSOR_FORCE_GPU=1
export CURSOR_DISABLE_GPU=1

# Taille du contexte IA (nombre de fichiers indexes)
```

```
export CURSOR_CONTEXT_SIZE=50
```

TRUCS & ASTUCES

-> Importer VS Code : au premier lancement, importer extensions, themes et raccourcis en un clic
-> Autocompletion IA : taper du code et attendre les suggestions predictives (Tab pour accepter)
-> Chat IA (Ctrl+L) : poser des questions sur le code, demander des explications ou des refactorisations
-> Commande inline (Ctrl+K) : selectionner du code et decrire la modification souhaitee en langage naturel
-> Mode Agent (Ctrl+I) : laisser l'IA effectuer des taches complexes de maniere autonome sur le projet
-> Correction terminal : quand une commande echoue dans le terminal, Cursor propose une correction IA
-> Contexte du projet : Cursor indexe tout le codebase - les reponses IA tiennent compte de l'ensemble du projet
-> @-references : dans le chat, utiliser @filename pour referencer un fichier specifique dans votre question
-> .cursorrules : creer ce fichier a la racine du projet pour definir des regles IA specifiques
-> Plan gratuit : 2000 completions + 50 requetes lentes par mois. Plans Pro a 20$/mois

COMMANDES CLI & RACCOURCIS

```
# Ouvrir un projet
cursor /chemin/vers/projet

# Ouvrir un fichier specifique a une ligne
cursor fichier.py:42

# Ouvrir dans une nouvelle fenetre
cursor --new-window /chemin/vers/projet

# Comparer deux fichiers
cursor --diff fichier1.py fichier2.py

# Installer une extension depuis le terminal
cursor --install-extension ms-python.python

# Lister les extensions installees
cursor --list-extensions

# Desinstaller une extension
cursor --uninstall-extension extension.id

# Mode portable (sans installation)
./cursor.AppImage --user-data-dir /tmp/cursor-portable

# === RACCOURCIS CLAVIER IA ===
# Ctrl+L         -> Ouvrir le chat IA
# Ctrl+K         -> Commande inline (edit IA)
# Ctrl+I         -> Mode Agent
# Tab            -> Accepter la suggestion IA
# Esc            -> Rejeter la suggestion IA
# Ctrl+Shift+P   -> Palette de commandes
```

LIENS DE REFERENCE

Site officiel : https://cursor.sh/
Documentation : https://docs.cursor.com/
Changelog : https://changelog.cursor.sh/
Tarifs : https://cursor.sh/pricing
Forum : https://forum.cursor.com/
GitHub : https://github.com/getcursor/cursor

VS Code base : https://code.visualstudio.com/

6. Claude Code - Assistant CLI d'Anthropic

SYNTHESE

Claude Code est l'outil CLI officiel d'Anthropic permettant d'utiliser Claude directement dans le terminal. C'est un agent de programmation agentic qui comprend l'ensemble du codebase, peut editer des fichiers, executer des commandes, naviguer dans le code, creer des commits Git et des pull requests. Il utilise Claude Opus 4 et Sonnet 4 comme modeles. Fonctionne comme un assistant de developpement complet capable de gerer des taches complexes de bout en bout.

INSTALLATION

```
# Installation via npm (methode recommandee)
npm install -g @anthropic-ai/claude-code

# Verification de l'installation
claude --version

# Prerequis
# - Node.js >= 18
# - Cle API Anthropic (ou abonnement Claude Max/Team/Enterprise)

# Premiere utilisation : authentification
claude
# -> Suit le flux d'authentification OAuth ou cle API

# Mise a jour
npm update -g @anthropic-ai/claude-code
```

VARIABLES D'ENVIRONNEMENT

```
# Cle API Anthropic (obligatoire si pas OAuth)
export ANTHROPIC_API_KEY="sk-ant-api03-..."

# Modele a utiliser (defaut: claude-sonnet-4-20250514)
export CLAUDE_MODEL="claude-opus-4-20250514"

# Desactiver la telemetrie
export CLAUDE_CODE_DISABLE_TELEMETRY=1

# Proxy HTTP/HTTPS
export HTTP_PROXY="http://proxy:8080"
export HTTPS_PROXY="http://proxy:8080"

# Repertoire de configuration
export CLAUDE_CONFIG_DIR="$HOME/.claude"

# Niveau de log
export CLAUDE_CODE_LOG_LEVEL="debug"

# Desactiver les mises a jour automatiques
export CLAUDE_CODE_DISABLE_AUTO_UPDATE=1

# Forcer un mode de permission
# Options: default, plan, bypassPermissions
export CLAUDE_CODE_PERMISSION_MODE="default"

# URL API personnalisee (pour proxy ou relais)
export ANTHROPIC_BASE_URL="https://custom-api.example.com"
```

```
# Budget max de tokens par session
export CLAUDE_CODE_MAX_TOKENS=100000
```

TRUCS & ASTUCES

-> Fichier CLAUDE.md : creer un fichier CLAUDE.md a la racine du projet pour donner des instructions persistantes a Claude
-> Mode plan : lancer 'claude --plan' pour que Claude analyse d'abord et propose un plan avant d'agir
-> Commandes slash : /help, /compact (compresser le contexte), /clear, /cost, /model, /permissions
-> Historique : Claude Code retient le contexte de la conversation, utiliser /compact si le contexte est trop long
-> Multi-fichiers : Claude peut editer plusieurs fichiers simultanement et lancer des tests pour verifier
-> Git integre : Claude peut creer des branches, commiter, pousser et creer des PR directement
-> Mode pipe : echo 'question' | claude pour une utilisation non-interactive dans des scripts
-> Reprendre une session : claude --resume pour reprendre la derniere conversation
-> Permissions : Claude demande confirmation avant les actions risquees (suppression, push, etc.)
-> SDK : utiliser le SDK TypeScript @anthropic-ai/claude-code pour l'integration dans des outils custom
-> Sous-agents : Claude Code peut lancer des sous-agents pour paralleliser les recherches dans le code
-> Memory : Claude Code dispose d'une memoire persistante dans ~/.claude/projects/ entre les sessions

COMMANDES CLI & RACCOURCIS

```
# Lancer Claude Code dans le repertoire courant
claude

# Lancer avec un prompt initial
claude "explique moi ce projet"

# Mode non-interactif (pour scripts)
claude -p "corrige le bug dans main.py"

# Mode plan (analyse sans modification)
claude --plan

# Reprendre la derniere session
claude --resume

# Utiliser un modele specifique
claude --model claude-opus-4-20250514

# Mode pipe (stdin -> stdout)
echo "genere un Dockerfile pour une app Python" | claude

# Afficher les couts de la session
# (dans une session) taper: /cost

# === COMMANDES SLASH (dans une session) ===
# /help             -> Aide complete
# /compact          -> Compresser le contexte pour economiser des tokens
# /clear            -> Effacer la conversation
# /model            -> Changer de modele en cours de session
# /cost             -> Afficher le cout cumule
# /permissions      -> Gerer les permissions d'outils
# /bug              -> Signaler un bug
# Ctrl+C            -> Annuler l'action en cours (sans quitter)
# Ctrl+C x2         -> Quitter la session
# Esc               -> Annuler l'edition en cours
```

```
# === FICHIERS DE CONFIGURATION ===
# ~/.claude/settings.json    -> Config globale
# .claude/settings.json      -> Config par projet
# CLAUDE.md                  -> Instructions pour Claude (racine projet)
# .claude/commands/           -> Commandes slash personnalisees
```

LIENS DE REFERENCE

Site officiel :	https://claude.ai/claude-code
Documentation :	https://docs.anthropic.com/en/docs/claude-code
npm :	https://www.npmjs.com/package/@anthropic-ai/claude-code
GitHub Issues :	https://github.com/anthropics/claude-code/issues
API Anthropic :	https://docs.anthropic.com/en/docs/api
Tarifs :	https://www.anthropic.com/pricing
Changelog :	https://docs.anthropic.com/en/docs/claude-code/changelog

7. Gemini CLI - Assistant CLI de Google

SYNTHESE

Gemini CLI est l'interface en ligne de commande officielle de Google pour interagir avec les modeles Gemini (2.5 Pro, Flash, etc.) directement depuis le terminal. Open-source et gratuit, il offre un agent de codage agentic capable de lire/editer des fichiers, executer des commandes shell, faire des recherches Google, et interagir avec les API Google. Il supporte le contexte de 1 million de tokens de Gemini et peut analyser des projets entiers.

INSTALLATION

```
# Installation via npm
npm install -g @anthropic-ai/gemini-cli
# OU depuis le depot Google
npm install -g @google/gemini-cli

# Verification
gemini --version

# Prerequis
# - Node.js >= 18
# - Compte Google (authentification gratuite)

# Premiere utilisation
gemini
# -> Authentification via navigateur (OAuth Google)

# Alternative : cle API Gemini
export GEMINI_API_KEY="AIza..."
gemini

# Mise a jour
npm update -g @google/gemini-cli
```

VARIABLES D'ENVIRONNEMENT

```
# Cle API Google Gemini
export GEMINI_API_KEY="AIza..."

# Projet Google Cloud (optionnel, pour Vertex AI)
export GOOGLE_CLOUD_PROJECT="mon-projet-id"

# Region Vertex AI
export GOOGLE_CLOUD_LOCATION="europe-west1"

# Modele par defaut
export GEMINI_MODEL="gemini-2.5-pro"

# Proxy HTTP
export HTTP_PROXY="http://proxy:8080"
export HTTPS_PROXY="http://proxy:8080"

# Repertoire de configuration
export GEMINI_CONFIG_DIR="$HOME/.gemini"

# Desactiver la telemetrie
export GEMINI_DISABLE_TELEMETRY=1
```

```
# Activer le mode sandbox (securite)
export GEMINI_SANDBOX=1

# Theme de couleurs (dark/light)
export GEMINI_THEME="dark"
```

TRUCS & ASTUCES

-> GEMINI.md : creer un fichier GEMINI.md a la racine du projet pour les instructions persistantes
-> Gratuit : 60 requetes/minute avec un compte Google gratuit (modele Gemini 2.5 Pro)
-> Contexte 1M tokens : Gemini peut analyser des projets entiers grace a sa fenetre de contexte massive
-> Google Search integre : Gemini peut chercher sur Google pour completer ses reponses avec des infos recentes
-> Extensions : systeme de plugins pour ajouter des fonctionnalites (MCP compatible)
-> Multi-modal : peut analyser des images, des screenshots et des diagrammes en plus du code
-> Mode sandbox : utiliser le flag --sandbox pour executer les commandes dans un conteneur isole
-> Memoire : les sessions sont sauvegardees dans ~/.gemini/ pour pouvoir etre reprises
-> Commandes slash : /help, /clear, /model, /tools pour gerer la session
-> Thinking : Gemini 2.5 Pro utilise le 'thinking' (reflexion) pour les taches complexes

COMMANDES CLI

```
# Lancer Gemini CLI interactif
gemini

# Prompt initial
gemini "analyse ce projet et resume l'architecture"

# Mode non-interactif
echo "genere des tests unitaires pour utils.py" | gemini

# Specifier un modele
gemini --model gemini-2.5-flash

# Mode sandbox (execution isolee)
gemini --sandbox

# Configurer les outils disponibles
gemini --tools shell,read,write,google_search

# === COMMANDES SLASH ===
# /help       -> Aide
# /clear      -> Nouvelle conversation
# /model      -> Changer de modele
# /tools      -> Lister les outils actifs
# /stats      -> Statistiques d'utilisation
# /save       -> Sauvegarder la conversation
# /load       -> Charger une conversation
# Ctrl+C      -> Annuler

# === FICHIERS DE CONFIGURATION ===
# ~/.gemini/settings.json   -> Config globale
# GEMINI.md                 -> Instructions projet
# .gemini/                  -> Config locale du projet
```

LIENS DE REFERENCE

GitHub :	https://github.com/google-gemini/gemini-cli
Google AI Studio :	https://aistudio.google.com/
API Gemini :	https://ai.google.dev/
Documentation :	https://ai.google.dev/gemini-api/docs
Modeles :	https://ai.google.dev/gemini-api/docs/models
Tarifs :	https://ai.google.dev/pricing

8. Windsurf - IDE IA (ex Codeium)

SYNTHESE

Windsurf (anciennement Codeium) est un IDE de programmation avec IA integree, base sur un fork de VS Code. Il propose Cascade, un agent IA agentic qui comprend le codebase entier et peut effectuer des modifications multi-fichiers. Windsurf offre l'autocompletion IA (Supercomplete), un chat contextuel, et un mode Flows qui combine la collaboration humain-IA. Supporte de nombreux modeles dont GPT-4o, Claude Sonnet, et des modeles proprietaires. Plan gratuit genereux disponible.

INSTALLATION

```
# Telecharger le .deb depuis le site officiel
wget https://windsurf-stable.codeium.com/linux/x64/latest -O windsurf.deb
sudo dpkg -i windsurf.deb

# Alternative : telechargement direct
# -> https://codeium.com/windsurf/download

# Snap (si disponible)
sudo snap install windsurf --classic

# Lancer apres installation
windsurf

# Importer les parametres VS Code
# -> Propose automatiquement au premier lancement

# Mise a jour : automatique via le gestionnaire integre

# Extension Codeium pour VS Code (alternative)
# Si vous preferez rester sur VS Code :
code --install-extension Codeium.codeium
```

VARIABLES D'ENVIRONNEMENT

```
# Cle API Codeium (generee depuis le compte)
export CODEIUM_API_KEY="votre_cle"

# Proxy entreprise
export HTTP_PROXY="http://proxy:8080"
export HTTPS_PROXY="http://proxy:8080"

# Desactiver la telemetrie
export WINDSURF_TELEMETRY=0

# Repertoire de configuration
export WINDSURF_USER_DATA_DIR="$HOME/.windsurf"

# Forcer le rendu GPU/Software
export WINDSURF_DISABLE_GPU=1

# Backend d'affichage (X11/Wayland)
export WINDSURF_OZONE_PLATFORM="wayland"

# Chemin extensions personnalise
export WINDSURF_EXTENSIONS_DIR="$HOME/.windsurf/extensions"

# Configuration du modele IA prefere
```

```
export WINDSURF_DEFAULT_MODEL="claude-sonnet"
```

TRUCS & ASTUCES

-> Cascade (Ctrl+Shift+L) : l'agent IA agentic qui peut modifier plusieurs fichiers et executer des commandes
-> Supercomplete : autocompletion multi-lignes predictive, plus avancee que Copilot (Tab pour accepter)
-> Mode Write vs Chat : Write modifie le code directement, Chat explique et repond aux questions
-> Flows : combine les suggestions IA avec vos propres actions pour un workflow fluide
-> Contexte automatique : Windsurf indexe tout le projet, pas besoin de specifier les fichiers manuellement
-> Plan gratuit genereux : completions illimitees + credits Cascade + acces aux modeles premium
-> Importer VS Code : toutes les extensions, themes et raccourcis VS Code sont compatibles
-> .windsurfrules : fichier de configuration a la racine du projet pour les instructions IA
-> Terminal integre : les erreurs dans le terminal sont detectees et une correction IA est proposee
-> Multi-modeles : choisir entre GPT-4o, Claude Sonnet, et les modeles Codeium selon la tache

COMMANDES CLI

```
# Ouvrir un projet
windsurf /chemin/vers/projet

# Ouvrir un fichier a une ligne specifique
windsurf fichier.py:42

# Nouvelle fenetre
windsurf --new-window /chemin/vers/projet

# Comparer deux fichiers
windsurf --diff fichier1.py fichier2.py

# Installer une extension
windsurf --install-extension ms-python.python

# Lister les extensions
windsurf --list-extensions

# Mode portable
windsurf --user-data-dir /tmp/windsurf-portable

# === RACCOURCIS CLAVIER IA ===
# Ctrl+Shift+L    -> Ouvrir Cascade (agent IA)
# Ctrl+L          -> Chat IA contextuel
# Ctrl+I          -> Commande inline IA
# Tab             -> Accepter la suggestion
# Esc             -> Rejeter la suggestion
# Ctrl+Shift+P    -> Palette de commandes
# Ctrl+K          -> Quick action IA

# === FICHIERS DE CONFIGURATION ===
# .windsurfrules          -> Instructions IA pour le projet
# ~/.windsurf/settings/   -> Config globale
```

LIENS DE REFERENCE

Site officiel :	https://codeium.com/windsurf
Telechargement :	https://codeium.com/windsurf/download
Documentation :	https://docs.codeium.com/
Blog :	https://codeium.com/blog
Tarifs :	https://codeium.com/pricing
Discord :	https://discord.gg/codeium
Extension VS Code :	https://marketplace.visualstudio.com/items?itemName=Codeium.codeium

9. Aider - Programmation en Binome avec IA

SYNTHESE

Aider est un outil CLI open-source de programmation en binome avec l'IA. Il se connecte a votre depot Git et permet de modifier le code via des conversations en langage naturel. Aider comprend la structure du projet, edite les fichiers directement, et cree des commits Git automatiquement. Il supporte plus de 30 modeles (GPT-4o, Claude, Gemini, DeepSeek, Ollama). Classe #1 sur le benchmark SWE Bench de resolution de bugs.

INSTALLATION

```
# Installation via pip (recommandee)
pip install aider-chat

# Ou avec pipx (environnement isole)
pipx install aider-chat

# Via Homebrew (macOS/Linux)
brew install aider

# Verification
aider --version

# Mise a jour
pip install --upgrade aider-chat

# Prerequis : au moins une cle API
export OPENAI_API_KEY="sk-..."     # pour GPT-4
export ANTHROPIC_API_KEY="sk-ant-..." # pour Claude
# OU utiliser Ollama pour le mode local
```

VARIABLES D'ENVIRONNEMENT

```
# Cle API OpenAI
export OPENAI_API_KEY="sk-..."

# Cle API Anthropic (Claude)
export ANTHROPIC_API_KEY="sk-ant-..."

# Cle API Google (Gemini)
export GOOGLE_API_KEY="AIza..."

# Cle API DeepSeek
export DEEPSEEK_API_KEY="sk-..."

# Cle API Groq (inference rapide)
export GROQ_API_KEY="gsk_..."

# URL de base OpenAI (pour proxy/compatible)
export OPENAI_API_BASE="http://localhost:11434/v1"

# Modele par defaut
export AIDER_MODEL="claude-sonnet-4-20250514"

# Desactiver les commits automatiques
export AIDER_AUTO_COMMITS=0

# Mode sombre/clair
```

```
export AIDER_DARK_MODE=1

# Fichier de configuration
export AIDER_CONFIG="$HOME/.aider.conf.yml"

# Desactiver l'envoi de statistiques
export AIDER_ANALYTICS=false
```

TRUCS & ASTUCES

-> Commits automatiques : Aider cree un commit Git a chaque modification, facilitant le rollback si necessaire
-> Fichier .aider.conf.yml : configurer les preferences par defaut (modele, dark mode, auto-commit, etc.)
-> Ajouter des fichiers : /add fichier.py pour ajouter un fichier au contexte de la conversation
-> Mode architecte : --architect utilise un modele puissant pour planifier et un modele rapide pour coder
-> Ollama local : aider --model ollama/llama3 pour utiliser des modeles locaux sans cle API
-> Mode lint : Aider detecte les erreurs de syntaxe et les corrige automatiquement apres chaque edit
-> Map du repo : Aider construit automatiquement une carte du repository pour le contexte IA
-> Voice : aider --voice pour dicter les modifications par la voix
-> Historique : les conversations sont sauvegardees dans .aider.chat.history.md
-> Mode watch : aider --watch surveille les fichiers et repond aux commentaires # AI dans le code

COMMANDES CLI

```
# Lancer aider avec Claude
aider --model claude-sonnet-4-20250514

# Lancer avec GPT-4o
aider --model gpt-4o

# Utiliser Ollama (local)
aider --model ollama/llama3

# Mode architecte (plan + execution)
aider --architect

# Ajouter des fichiers au contexte
aider fichier1.py fichier2.py

# Desactiver les auto-commits
aider --no-auto-commits

# Mode sombre
aider --dark-mode

# Mode vocal
aider --voice

# Mode watch (surveille les commentaires AI)
aider --watch

# === COMMANDES SLASH (dans une session) ===
# /add fichier.py     -> Ajouter un fichier au contexte
# /drop fichier.py    -> Retirer un fichier du contexte
# /ls                 -> Lister les fichiers dans le contexte
# /diff               -> Voir les modifications en cours
# /undo               -> Annuler le dernier commit
# /clear              -> Effacer la conversation
```

```
# /tokens             -> Afficher l'utilisation de tokens
# /model nom          -> Changer de modele
# /help               -> Aide complete
# /run commande       -> Executer une commande shell
# /test commande      -> Lancer les tests
# /lint               -> Lancer le linter
# /commit             -> Forcer un commit
# /quit               -> Quitter
```

LIENS DE REFERENCE

Site officiel : https://aider.chat/
GitHub : https://github.com/Aider-AI/aider
Documentation : https://aider.chat/docs/
Leaderboard : https://aider.chat/docs/leaderboards/
Configuration : https://aider.chat/docs/config.html
PyPI : https://pypi.org/project/aider-chat/

10. llm - CLI Universel pour LLM

SYNTHESE

llm est un outil CLI cree par Simon Willison (createur de Datasette) permettant d'interagir avec n'importe quel LLM depuis le terminal. Il supporte OpenAI, Anthropic, Gemini, Ollama, et des dizaines d'autres via un systeme de plugins. Il inclut un systeme de templates, une base SQLite pour l'historique des conversations, et la gestion des embeddings. Ideal pour le scripting et l'automatisation IA en ligne de commande.

INSTALLATION

```
# Installation via pip
pip install llm

# Ou avec pipx
pipx install llm

# Ou avec Homebrew
brew install llm

# Configurer une cle API
llm keys set openai      # -> entrer la cle
llm keys set anthropic   # -> entrer la cle

# Installer des plugins pour d'autres modeles
llm install llm-claude-3    # Claude
llm install llm-gemini      # Gemini
llm install llm-ollama       # Ollama (local)
llm install llm-gpt4all     # GPT4All (local)
llm install llm-mistral     # Mistral

# Verification
llm --version
llm models                # Lister les modeles disponibles
```

VARIABLES D'ENVIRONNEMENT

```
# Cle API OpenAI
export OPENAI_API_KEY="sk-..."

# Cle API Anthropic
export ANTHROPIC_API_KEY="sk-ant-..."

# Cle API Gemini
export GEMINI_API_KEY="AIza..."

# Cle API Mistral
export MISTRAL_API_KEY="..."

# Modele par defaut
export LLM_DEFAULT_MODEL="claude-sonnet-4-20250514"

# Repertoire de donnees (SQLite, logs, config)
export LLM_USER_PATH="$HOME/.llm"

# URL OpenAI personnalisee (proxy, local)
export OPENAI_API_BASE="http://localhost:11434/v1"
```

TRUCS & ASTUCES

-> Pipe Unix : cat fichier.py | llm 'explique ce code' -> ideal pour le scripting
-> Templates : llm --system 'Tu es un expert Python' 'question' pour les instructions systeme
-> Historique SQLite : llm logs -> voir tout l'historique, llm logs -q 'mot' pour chercher
-> Conversation : llm chat -m claude-3-opus pour un mode conversation continue
-> Sauvegarder un template : llm templates save expert --system 'Tu es un expert...'
-> Embeddings : llm embed -m ada-002 -c 'texte' pour generer des vecteurs d'embedding
-> Plugins : plus de 50 plugins disponibles pour tous les fournisseurs de LLM
-> Aliases : llm aliases set c3s claude-3-sonnet -> raccourci pour les noms de modeles
-> JSON output : llm 'liste 5 langages' --json pour forcer une sortie JSON structuree

COMMANDES CLI

```
# Requete simple
llm "Qu'est-ce que le machine learning ?"

# Choisir un modele
llm -m claude-sonnet-4-20250514 "explique les monades"

# Pipe depuis un fichier
cat error.log | llm "explique cette erreur et propose une solution"

# Instruction systeme
llm --system "Tu es un DBA PostgreSQL" "optimise cette requete"

# Mode conversation
llm chat -m gpt-4o

# Voir l'historique
llm logs
llm logs --count 10         # 10 derniers
llm logs -q "python"        # rechercher

# Lister les modeles
llm models
llm models -q claude        # filtrer

# Gestion des cles
llm keys set openai
llm keys set anthropic

# Templates
llm templates save reviewer --system "Tu es un code reviewer strict"
llm -t reviewer "def add(a,b): return a+b"

# Embeddings
llm embed -m text-embedding-3-small -c "texte a encoder"
llm similar -d ma_collection "recherche semantique"

# Plugins
llm plugins                 # lister les plugins installes
llm install llm-claude-3   # installer un plugin
```

LIENS DE REFERENCE

Site officiel : https://llm.datasette.io/
GitHub : https://github.com/simonw/llm
Plugins : https://llm.datasette.io/en/stable/plugins/directory.html
PyPI : https://pypi.org/project/llm/
Blog auteur : https://simonwillison.net/
Documentation : https://llm.datasette.io/en/stable/

11. GitHub Copilot CLI - IA dans le Terminal

SYNTHESE

GitHub Copilot CLI (integre a la commande gh) permet d'utiliser l'IA de GitHub Copilot directement dans le terminal. Il peut suggerer des commandes shell, expliquer des commandes complexes, et generer des commandes Git/gh a partir de descriptions en langage naturel. Necessite un abonnement GitHub Copilot (gratuit pour les etudiants et l'open-source).

INSTALLATION

```
# Prerequis : GitHub CLI (gh)
sudo apt install gh          # Debian/Ubuntu
sudo dnf install gh          # Fedora
brew install gh              # Homebrew

# Authentification GitHub
gh auth login

# Installer l'extension Copilot
gh extension install github/gh-copilot

# Verification
gh copilot --help

# Mise a jour
gh extension upgrade gh-copilot

# Activer dans le shell (aliases)
eval "$(gh copilot alias -- zsh)"   # pour zsh
eval "$(gh copilot alias -- bash)"  # pour bash
# -> Cree les aliases 'ghcs' (suggest) et 'ghce' (explain)
```

VARIABLES D'ENVIRONNEMENT

```
# Token GitHub (normalement gere par gh auth)
export GITHUB_TOKEN="ghp_..."

# Proxy HTTP
export HTTP_PROXY="http://proxy:8080"
export HTTPS_PROXY="http://proxy:8080"

# Editeur par defaut pour gh
export GH_EDITOR="vim"

# Repertoire de configuration gh
export GH_CONFIG_DIR="$HOME/.config/gh"

# Desactiver la telemetrie
export GH_NO_TELEMETRY=1

# Hote GitHub Enterprise (si applicable)
export GH_HOST="github.enterprise.com"
```

TRUCS & ASTUCES

-> ghcs (suggest) : decrire ce que vous voulez faire et Copilot genere la commande shell
-> ghce (explain) : coller une commande complexe et Copilot l'explique en langage naturel
-> Types de suggestions : ghcs -t shell (commandes shell), -t git (commandes git), -t gh (commandes gh)
-> Execution directe : apres une suggestion, choisir 'Execute' pour l'executer immediatement
-> Copier en presse-papier : choisir 'Copy' pour copier la commande suggeree
-> Gratuit pour OSS : les mainteneurs de projets open-source populaires ont acces gratuit a Copilot
-> Alias dans .zshrc/.bashrc : ajouter eval "$(gh copilot alias -- zsh)" pour les aliases permanents
-> Combiner avec pipe : ghcs fonctionne aussi dans des contextes de pipe et de scripting

COMMANDES CLI

```
# Suggerer une commande shell
gh copilot suggest "trouver les fichiers modifies cette semaine"
# Raccourci avec alias :
ghcs "trouver les fichiers modifies cette semaine"

# Suggerer une commande git
gh copilot suggest -t git "annuler le dernier commit sans perdre les changements"
ghcs -t git "squash les 3 derniers commits"

# Suggerer une commande gh
gh copilot suggest -t gh "creer une issue avec le label bug"
ghcs -t gh "lister les PR ouvertes assignees a moi"

# Expliquer une commande
gh copilot explain "find . -name '*.log' -mtime +30 -exec rm {} \;"
ghce "awk '{sum+=$1} END {print sum}' fichier.txt"

# Expliquer une commande git complexe
ghce "git rebase -i --autosquash HEAD~5"

# Configurer les aliases dans le shell
# Ajouter dans ~/.zshrc ou ~/.bashrc :
# eval "$(gh copilot alias -- zsh)"
# eval "$(gh copilot alias -- bash)"

# Verifier la version
gh copilot --version
```

LIENS DE REFERENCE

GitHub Copilot : https://github.com/features/copilot
Documentation : https://docs.github.com/en/copilot/using-github-copilot/using-github-copilot-in-the-command-line
Extension : https://github.com/github/gh-copilot
GitHub CLI : https://cli.github.com/
Tarifs : https://github.com/features/copilot/plans
Blog : https://github.blog/tag/github-copilot/

12. Open Interpreter - Code Interpreter Local

SYNTHESE

Open Interpreter est un outil open-source qui permet aux LLM d'executer du code localement sur votre machine. Il fonctionne comme le Code Interpreter de ChatGPT mais sans les limitations de sandbox. Il peut executer du Python, JavaScript, Shell, et plus. Il manipule des fichiers, navigue sur le web, et controle le systeme. Supporte GPT-4, Claude, Gemini, et les modeles locaux.

INSTALLATION

```
# Installation via pip
pip install open-interpreter

# Ou avec pipx
pipx install open-interpreter

# Verification
interpreter --version

# Premiere utilisation (configure la cle API)
interpreter
# -> Demande la cle API au premier lancement

# Mode local avec Ollama (pas de cle API)
interpreter --local

# Mise a jour
pip install --upgrade open-interpreter
```

VARIABLES D'ENVIRONNEMENT

```
# Cle API OpenAI
export OPENAI_API_KEY="sk-..."

# Cle API Anthropic
export ANTHROPIC_API_KEY="sk-ant-..."

# Cle API Google
export GOOGLE_API_KEY="AIza..."

# Modele par defaut
export INTERPRETER_MODEL="claude-sonnet-4-20250514"

# Desactiver la confirmation avant execution
export INTERPRETER_AUTO_RUN=true

# URL API personnalisee (Ollama, LM Studio, etc.)
export OPENAI_API_BASE="http://localhost:11434/v1"

# Limiter les depenses max
export INTERPRETER_MAX_BUDGET=5.00

# Repertoire de configuration
export INTERPRETER_CONFIG_DIR="$HOME/.interpreter"

# Mode safe (demande confirmation)
export INTERPRETER_SAFE_MODE=true
```

TRUCS & ASTUCES

-> ATTENTION : Open Interpreter execute du code reel sur votre machine - utiliser avec prudence !
-> Mode safe : par defaut, il demande confirmation avant chaque execution de code
-> --auto-run : desactiver les confirmations (uniquement pour des taches de confiance)
-> Mode local : interpreter --local pour utiliser Ollama sans envoyer de donnees au cloud
-> Python : excellent pour l'analyse de donnees, graphiques matplotlib, manipulation de CSV/Excel
-> System prompt : interpreter --system_message 'instructions' pour personnaliser le comportement
-> Profils : sauvegarder des configurations dans ~/.interpreter/profiles/
-> Vision : interpreter peut analyser des images et des screenshots
-> Budget : utiliser --max_budget 5.00 pour limiter les couts API
-> Fichier de config : ~/.interpreter/config.yaml pour les parametres permanents

COMMANDES CLI

```
# Lancer en mode interactif
interpreter

# Mode local (Ollama)
interpreter --local

# Choisir un modele
interpreter --model claude-sonnet-4-20250514
interpreter --model gpt-4o

# Auto-run (pas de confirmation)
interpreter --auto_run

# Limiter le budget
interpreter --max_budget 5.00

# Custom system message
interpreter --system_message "Tu es un data scientist expert"

# Mode conversation continue
interpreter --conversation

# === DANS UNE SESSION ===
# Simplement decrire ce que vous voulez :
# "Analyse le fichier data.csv et cree un graphique"
# "Telecharge cette page web et resume le contenu"
# "Renomme tous les fichiers .jpeg en .jpg dans ce dossier"
# "Cree un script Python qui..."

# reset -> Nouvelle conversation
# Ctrl+C -> Annuler l'execution en cours
```

LIENS DE REFERENCE

Site officiel : https://openinterpreter.com/
GitHub : https://github.com/OpenInterpreter/open-interpreter
Documentation : https://docs.openinterpreter.com/
PyPI : https://pypi.org/project/open-interpreter/
Discord : https://discord.gg/openinterpreter

13. Ollama - LLM Locaux en CLI

SYNTHESE

Ollama est l'outil de reference pour executer des grands modeles de langage localement sur votre machine. Il telecharge, gere et execute des modeles comme Llama 3, Mistral, Gemma, Phi, Code Llama, et des dizaines d'autres. Il expose une API REST compatible OpenAI sur localhost:11434 et fonctionne avec CPU ou GPU (NVIDIA/AMD). Ideal pour la confidentialite, le travail hors-ligne et les couts zero.

INSTALLATION

```
# Installation rapide (script officiel)
curl -fsSL https://ollama.ai/install.sh | sh

# Verification
ollama --version

# Demarrer le serveur (si pas auto-demarre)
ollama serve

# Via Docker
docker run -d -v ollama:/root/.ollama -p 11434:11434 \
    --name ollama ollama/ollama

# Docker avec GPU NVIDIA
docker run -d --gpus=all -v ollama:/root/.ollama -p 11434:11434 \
    --name ollama ollama/ollama

# Telecharger un premier modele
ollama pull llama3.1
ollama pull mistral
ollama pull codellama
```

VARIABLES D'ENVIRONNEMENT

```
# Adresse d'ecoute du serveur (defaut: 127.0.0.1)
export OLLAMA_HOST="0.0.0.0:11434"

# Repertoire des modeles (defaut: ~/.ollama/models)
export OLLAMA_MODELS="/data/ollama/models"

# Nombre de GPU a utiliser (defaut: auto)
export OLLAMA_NUM_GPU=1

# Limiter la VRAM utilisee (en Go)
export OLLAMA_GPU_MEMORY="8G"

# Forcer le mode CPU uniquement
export OLLAMA_NO_GPU=1

# Nombre de threads CPU
export OLLAMA_NUM_THREADS=8

# Temps avant dechargement du modele (defaut: 5m)
export OLLAMA_KEEP_ALIVE="30m"

# Taille maximale de la queue de requetes
export OLLAMA_MAX_QUEUE=10
```

```
# Niveau de log (debug, info, warn, error)
export OLLAMA_DEBUG=1

# Proxy HTTP
export HTTP_PROXY="http://proxy:8080"
export HTTPS_PROXY="http://proxy:8080"

# Origines CORS autorisees
export OLLAMA_ORIGINS="http://localhost:3000,http://localhost:8080"
```

TRUCS & ASTUCES

-> Modeles populaires : llama3.1 (general), codellama (code), mistral (rapide), phi3 (leger), gemma2 (Google)
-> GPU NVIDIA : Ollama detecte automatiquement CUDA. Verifier avec: ollama ps
-> Quantization : les modeles sont quantifies (Q4_0, Q5_K_M, etc.) pour tourner sur du materiel standard
-> API OpenAI compatible : pointer vos outils vers http://localhost:11434/v1 comme base URL OpenAI
-> Modelfile : creer des modeles personnalises avec des system prompts, temperature, etc.
-> Multi-modal : ollama run llava pour un modele qui comprend les images
-> RAM requise : 8 Go pour les modeles 7B, 16 Go pour 13B, 32 Go pour 30B+
-> Serveur LAN : OLLAMA_HOST=0.0.0.0 pour exposer sur le reseau local
-> Combiner avec Aider/llm : utiliser Ollama comme backend pour les autres outils CLI IA
-> Open WebUI : docker run ... ghcr.io/open-webui/open-webui pour une interface web pour Ollama

COMMANDES CLI

```
# Telecharger un modele
ollama pull llama3.1
ollama pull mistral
ollama pull codellama:13b    # variante specifique

# Lancer une conversation
ollama run llama3.1
ollama run mistral

# Requete en une ligne
ollama run llama3.1 "Explique le theoreme de Bayes"

# Mode pipe
echo "Resume ce texte" | ollama run llama3.1

# Lister les modeles installes
ollama list

# Voir les modeles en cours d'execution
ollama ps

# Supprimer un modele
ollama rm codellama

# Copier/renommer un modele
ollama cp llama3.1 mon-modele

# Creer un modele personnalise (Modelfile)
cat > Modelfile << 'EOF'
FROM llama3.1
SYSTEM Tu es un assistant Python expert.
```

```
PARAMETER temperature 0.3
PARAMETER num_ctx 8192
EOF
ollama create python-expert -f Modelfile

# Afficher les infos d'un modele
ollama show llama3.1

# API REST
curl http://localhost:11434/api/generate -d '{
    "model": "llama3.1",
    "prompt": "Bonjour, comment vas-tu ?"
}'

# API compatible OpenAI
curl http://localhost:11434/v1/chat/completions -d '{
    "model": "llama3.1",
    "messages": [{"role": "user", "content": "Hello"}]
}'

# Demarrer le serveur manuellement
ollama serve

# Arreter le serveur
systemctl stop ollama    # si installe en service
```

LIENS DE REFERENCE

Site officiel :	https://ollama.ai/
GitHub :	https://github.com/ollama/ollama
Bibliotheque :	https://ollama.ai/library
API Docs :	https://github.com/ollama/ollama/blob/main/docs/api.md
Modelfile :	https://github.com/ollama/ollama/blob/main/docs/modelfile.md
Discord :	https://discord.gg/ollama
Blog :	https://ollama.ai/blog

Annexe A : Lexique de l'Intelligence Artificielle

IA (Intelligence Artificielle)
Discipline informatique visant a creer des systemes capables de realiser des taches necessitant normalement l'intelligence humaine : comprehension du langage, reconnaissance d'images, prise de decision.

LLM (Large Language Model)
Grand modele de langage entraine sur d'enormes corpus de texte. Exemples : GPT-4, Claude, Gemini, Llama, Mistral. Capables de generer du texte, coder, traduire.

NLP (Natural Language Processing)
Traitement automatique du langage naturel. Sous-domaine de l'IA permettant aux machines de comprendre, interpreter et generer du langage humain.

Transformer
Architecture de reseau de neurones introduite en 2017 (Google). Base de tous les LLM modernes. Utilise le mecanisme d'attention pour traiter les sequences.

Token
Unite de base du traitement textuel par les LLM. Un token represente environ 3/4 d'un mot en anglais. Les modeles ont une limite de contexte en tokens.

Prompt
Instruction ou question donnee a un modele IA. La qualite du prompt influence directement la qualite de la reponse (d'ou le 'prompt engineering').

Prompt Engineering
Art de formuler des instructions optimales pour obtenir les meilleures reponses d'un LLM. Techniques : few-shot, chain-of-thought, role-playing.

Fine-tuning
Processus d'ajustement d'un modele pre-entraine sur un jeu de donnees specifique pour ameliorer ses performances sur une tache particuliere.

RAG (Retrieval-Augmented Generation)
Technique combinant la recherche de documents pertinents avec la generation de texte par LLM. Permet de reduire les hallucinations en ancrant les reponses.

Hallucination
Phenomene ou un modele IA genere des informations factuellement incorrectes mais presentees avec confiance. Probleme majeur des LLM actuels.

Inference
Phase d'utilisation d'un modele entraine pour generer des predictions ou des reponses. Par opposition a l'entrainement (training).

GPU (Graphics Processing Unit)
Processeur graphique utilise pour accelerer les calculs de l'IA. Les GPU NVIDIA (CUDA) sont les plus utilises pour l'entrainement et l'inference.

CUDA
Plateforme de calcul parallele de NVIDIA permettant d'utiliser les GPU pour l'entrainement et l'inference des modeles IA.

ONNX (Open Neural Network Exchange)
Format ouvert pour representer des modeles de machine learning. Permet la portabilite entre frameworks (PyTorch, TensorFlow).

API (Application Programming Interface)
Interface permettant a un programme d'interagir avec un service IA. Exemple : envoyer un prompt a GPT-4 via l'API OpenAI et recevoir une reponse.

Embedding
Representation numerique (vecteur) d'un texte, image ou autre donnee dans un espace multidimensionnel. Permet la recherche semantique.

Vector Database
Base de donnees optimisee pour stocker et rechercher des embeddings. Exemples : Chroma, Pinecone, Weaviate, Milvus, pgvector.

Quantization
Technique de compression des modeles IA en reduisant la precision des poids (ex: float32 -> int8). Reduit la taille et accelere l'inference.

GGUF / GGML
Formats de fichiers pour les modeles quantifies, utilises par llama.cpp et Ollama. Permettent d'executer des LLM sur du materiel grand public.

Diffusion (Stable Diffusion)
Modele generatif d'images par processus de diffusion. Genere des images a partir de descriptions textuelles (text-to-image).

Computer Vision
Sous-domaine de l'IA permettant aux machines d'interpreter des images et videos. Applications : detection d'objets, segmentation, OCR.

OCR (Optical Character Recognition)
Reconnaissance optique de caracteres. Technologie IA pour extraire du texte a partir d'images ou de documents scannes.

Machine Learning (ML)
Sous-ensemble de l'IA ou les algorithmes apprennent a partir de donnees sans etre explicitement programmes. Types : supervise, non supervise, par renforcement.

Deep Learning
Sous-ensemble du ML utilisant des reseaux de neurones profonds (plusieurs couches). Performant pour l'image, le son et le texte.

Neural Network (Reseau de neurones)
Modele mathematique inspire du cerveau humain. Compose de couches de neurones artificiels connectes par des poids ajustables.

Open-Source IA
Modeles et outils IA dont le code source est librement accessible. Exemples : Llama (Meta), Mistral, Stable Diffusion, Whisper (OpenAI).

Edge AI
Execution de modeles IA directement sur l'appareil local (telephone, PC) plutot que dans le cloud. Avantages : latence, confidentialite, hors-ligne.

Agent IA
Systeme IA autonome capable d'executer des taches complexes en enchainant plusieurs actions : recherche, code, execution, verification.

Temperature
Parametre controlant la creativite des reponses d'un LLM. Valeur basse (0.1) = deterministe et precis. Valeur haute (1.0) = creatif et varie.

Context Window
Nombre maximum de tokens qu'un LLM peut traiter en une seule requete. GPT-4 : 128k tokens. Claude : 200k tokens. Gemini : 1M tokens.

Annexe B : RefCard - Aide-Memoire Rapide

ONLYOFFICE Docs

```
Installation : snap install onlyoffice-desktopeditors
Docker       : docker run -d -p 80:80 onlyoffice/documentserver
Plugin IA    : Plugins > Gestionnaire > AI Plugin
Modeles IA   : ChatGPT, Claude, Gemini, Mistral, Ollama, GPT4All
Formats      : DOCX, XLSX, PPTX, PDF, ODF
Cle env      : JWT_SECRET (securisation API serveur)
```

Opera One

```
Installation : apt install opera-stable | snap install opera
Lancer       : opera [URL]
Mode prive   : opera --private
IA           : Ctrl+/ ou bouton 'Ask AI'
Tab Islands  : Glisser onglet sur onglet
VPN gratuit  : Parametres > VPN > Activer
Kiosque      : opera --kiosk URL
```

Smartcat

```
Acces        : https://www.smartcat.com/ (navigateur)
CLI          : npm install -g @smartcat/cli
Auth env     : SMARTCAT_ACCOUNT_ID, SMARTCAT_API_KEY
Langues      : 280+ supportees
Formats      : DOCX, XLSX, JSON, XLIFF, PO, SRT, YAML
API base     : https://smartcat.com/api/integration/v2
Gratuit      : 2000 mots/mois
```

PhotoGlimmer

```
Installation : AppImage ou .deb depuis GitHub
Lancer       : ./PhotoGlimmer-x86_64.AppImage
Sources      : git clone github.com/codecliff/PhotoGlimmer
Deps Python  : mediapipe, opencv-python, PyQt5
Traitement   : 100% local, aucune connexion requise
Env          : QT_QPA_PLATFORM=xcb|wayland
```

Cursor (Editeur IA)

```
Installation : AppImage ou .deb depuis cursor.sh
Lancer       : cursor [chemin|fichier:ligne]
Chat IA      : Ctrl+L
Edit inline  : Ctrl+K (selectionner + decrire)
Agent mode   : Ctrl+I
Accepter IA  : Tab  |  Rejeter : Esc
Diff         : cursor --diff f1.py f2.py
Extensions   : cursor --install-extension id
Env          : OPENAI_API_KEY, ANTHROPIC_API_KEY
Config       : .cursorrules (racine projet)
Gratuit      : 2000 completions + 50 requetes/mois
```

Claude Code (CLI Anthropic)

```
Installation : npm install -g @anthropic-ai/claude-code
Lancer       : claude
Prompt direct: claude "question"
Non-interactif: claude -p "instruction"
Mode plan    : claude --plan
Reprendre    : claude --resume
Env          : ANTHROPIC_API_KEY, CLAUDE_MODEL
Config       : CLAUDE.md (racine projet), ~/.claude/
Slash        : /compact /clear /cost /model /help
Raccourcis   : Ctrl+C (annuler), Ctrl+C x2 (quitter), Esc (annuler edit)
```

Gemini CLI (CLI Google)

```
Installation : npm install -g @google/gemini-cli
Lancer       : gemini
Prompt direct: gemini "question"
Sandbox      : gemini --sandbox
Env          : GEMINI_API_KEY, GOOGLE_CLOUD_PROJECT
Config       : GEMINI.md (racine projet), ~/.gemini/
Gratuit      : 60 req/min avec compte Google
Contexte     : 1M tokens (projet entier)
Slash        : /help /clear /model /tools /stats
```

Windsurf (IDE IA)

```
Installation : .deb depuis codeium.com/windsurf
Lancer       : windsurf [chemin|fichier:ligne]
Cascade      : Ctrl+Shift+L (agent agentic)
Chat IA      : Ctrl+L
Inline edit  : Ctrl+I | Ctrl+K
Accepter     : Tab  |  Rejeter : Esc
Config       : .windsurfrules (racine projet)
Env          : CODEIUM_API_KEY
Gratuit      : completions illimitees + credits
```

Aider (Binome IA)

```
Installation : pip install aider-chat | brew install aider
Lancer       : aider (dans un depot Git)
Modele       : aider --model claude-sonnet-4-20250514
Local        : aider --model ollama/llama3
Architecte   : aider --architect
Voice        : aider --voice
Env          : OPENAI_API_KEY, ANTHROPIC_API_KEY, AIDER_MODEL
Slash        : /add /drop /ls /diff /undo /run /test /commit
```

llm (CLI Universel)

```
Installation : pip install llm | brew install llm
Requete      : llm "question"
Modele       : llm -m claude-sonnet-4-20250514 "question"
Pipe         : cat fichier | llm "explique"
Chat         : llm chat -m gpt-4o
```

```
Historique   : llm logs | llm logs -q "mot"
Templates    : llm -t template_name "prompt"
Plugins      : llm install llm-claude-3 | llm plugins
Cles         : llm keys set openai | llm keys set anthropic
```

GitHub Copilot CLI

```
Installation : gh extension install github/gh-copilot
Aliases      : eval "$(gh copilot alias -- zsh)"
Suggerer     : ghcs "description" | ghcs -t git "description"
Expliquer    : ghce "commande complexe"
Types        : -t shell, -t git, -t gh
Env          : GITHUB_TOKEN (via gh auth login)
```

Open Interpreter

```
Installation : pip install open-interpreter
Lancer       : interpreter
Local        : interpreter --local
Modele       : interpreter --model claude-sonnet-4-20250514
Auto-run     : interpreter --auto_run
Budget       : interpreter --max_budget 5.00
Env          : OPENAI_API_KEY, ANTHROPIC_API_KEY, INTERPRETER_MODEL
```

Ollama (LLM Locaux)

```
Installation : curl -fsSL https://ollama.ai/install.sh | sh
Telecharger  : ollama pull llama3.1 | ollama pull mistral
Lancer       : ollama run llama3.1
Lister       : ollama list | ollama ps
Supprimer    : ollama rm modele
Custom       : ollama create nom -f Modelfile
Serveur      : ollama serve (port 11434)
Env          : OLLAMA_HOST, OLLAMA_MODELS, OLLAMA_NUM_GPU
API REST     : curl localhost:11434/api/generate
API OpenAI   : curl localhost:11434/v1/chat/completions
```

Annexe C : Liens de Reference Globaux

Sites officiels des outils

ONLYOFFICE	https://www.onlyoffice.com/
Opera One	https://www.opera.com/one
Smartcat	https://www.smartcat.com/
PhotoGlimmer	https://github.com/codecliff/PhotoGlimmer
Cursor	https://cursor.sh/
Claude Code	https://docs.anthropic.com/en/docs/claude-code
Gemini CLI	https://github.com/google-gemini/gemini-cli
Windsurf	https://codeium.com/windsurf
Aider	https://aider.chat/
llm	https://llm.datasette.io/
GitHub Copilot CLI	https://docs.github.com/en/copilot
Open Interpreter	https://openinterpreter.com/
Ollama	https://ollama.ai/

Modeles IA compatibles

OpenAI (GPT-4, ChatGPT)	https://platform.openai.com/
Anthropic (Claude)	https://www.anthropic.com/
Google (Gemini)	https://ai.google.dev/
Meta (Llama)	https://llama.meta.com/
Mistral AI	https://mistral.ai/
Ollama (LLM local)	https://ollama.ai/
GPT4All (LLM local)	https://gpt4all.io/

Bibliotheques IA utilisees

MediaPipe (Google)	https://mediapipe.dev/
OpenCV	https://opencv.org/
TensorFlow	https://www.tensorflow.org/
PyTorch	https://pytorch.org/
Hugging Face	https://huggingface.co/

Ressources d'apprentissage IA

Papers With Code	https://paperswithcode.com/
Arxiv (recherche IA)	https://arxiv.org/list/cs.AI/recent
Fast.ai (cours gratuits)	https://www.fast.ai/
Hugging Face Learn	https://huggingface.co/learn
Google ML Crash Course	https://developers.google.com/machine-learning/crash-course

Communautes

Reddit r/LocalLLaMA	https://www.reddit.com/r/LocalLLaMA/
Reddit r/artificial	https://www.reddit.com/r/artificial/
Hacker News	https://news.ycombinator.com/
Stack Overflow IA	https://stackoverflow.com/questions/tagged/artificial-intelligence