Utiliser un agent de code en local

- Morgan Bohn - morgan.bohn@unistra.fr
- Laurine Sottani - l.sottani@unistra.fr
- Equipe IA - SACRE - DNUM - Unistra

Partie 1 : Faire tourner un LLM en local

Ollama : Un moteur d'inférence

- Charge un modèle de LLM en mémoire.
- Exécute les requêtes.
- Voir https://ollama.com/

Utilisation de Ollama

- Installation : `curl -fsSL https://ollama.com/install.sh | sh`
- Commande : `ollama --help`
- Démarrage : `ollama serve`
- Vérification : `ollama -v`
- Voir les modèles disponibles sur le site officiel : https://ollama.com/search
- Télécharger un petit modèle : `ollama pull sorc/qwen3.5-instruct:0.8b`
- On liste : `ollama list`
- On l'interroge : `ollama run sorc/qwen3.5-instruct:0.8b`

- On joue avec les paramètres : `/show parameters`
  - Température : Niveau de créativité.
  - Top-k : Nombre de mots probables à garder avant de faire la réponse.
  - Top-p : Choisir le prochain mot uniquement parmi ceux qui couvrent un certain pourcentage de probabilité.
  - Repeat-penalty : Empêche le modèle de répéter les mêmes mots ou phrases trop souvent.

- On modifie la température :
  - `/set parameter temperature 0`
  - `Explique-moi en une phrase la gravité`
    * `/clear`
  - `/set parameter temperature 1`
  - `Explique-moi en une phrase la gravité`

- Créer un **`Modelfile`** puis `ollama create coder -f ./Modelfile`

```
FROM sorc/qwen3.5-instruct:0.8b

PARAMETER temperature 0.6
PARAMETER top_k 20
PARAMETER top_p 0.95
PARAMETER repeat_penalty 1

SYSTEM """
Role: Expert coding assistant.
Task: Write, debug, and explain code.
Rules:
1. Be concise. No fluff.
2. Use best practices.
3. Ask questions if unclear.
"""
```

- On teste : `ollama run coder` avec `Écris un script Hello World en python`

Partie 2 : Utiliser un agent de code local

OpenCode : Un agent de code local

- Installation : `curl -fsSL https://opencode.ai/install | bash`
- Commande : `ollama launch opencode`
- On choisit notre model **coder**
- On teste un prompt rapidement: `Écris un script Hello World en python`
- Il a créé un fichier ...
- .. mais lent sans carte graphique et modèle trop petit pour être efficace
- Pour la suite du TP, on va utiliser un modèle plus gros distant

Qwen 3.6 35B A3B

- API OpenAI Compatible
- Url : https://sandbox.ia.unistra.fr/api
- Token : sk-72595fbc0fae4b5bad6e3b4f417cbf7d
- Modèle : qwen3

- dans `~/.config/opencode/opencode.json`

```
{
  "$schema": "https://opencode.ai/config.json",
  "provider": {
    "owui": {
      "npm": "@ai-sdk/openai-compatible",
      "name": "sandbox",
      "options": {
        "baseURL": "https://sandbox.ia.unistra.fr/api"
      },
      "models": {
        "qwen3": {
          "name": "qwen3",
          "limit": {
            "context": 224000,
            "output": 32000
          }
        }
      }
    }
  }
}
```

- Redémarre OpenCode
- `/connect`, puis **sandbox**
- On saisit le token d'authentification : **sk-72595fbc0fae4b5bad6e3b4f417cbf7d**
- `/models`, puis **qwen3**
- On teste un prompt

Mode / agent

- On change de mode avec **Tab**
- **Plan** : en lecture seule, pour planifier avant exécution
- **Build** : pour écrire le code suite au plan
- On peut créer son propre mode/agent, par exemple pour du code review

- ~/.config/opencode/agents/review.md

```
---
description: Reviews code for quality and best practices
mode: primary
temperature: 0.1
tools:
  write: false
  edit: false
  bash: false
---

You are in code review mode. Focus on:

- Code quality and best practices
- Potential bugs and edge cases
- Performance implications
- Security considerations

Provide constructive feedback without making direct changes.
```

Etape 1 : Création d'un petit script

- On installe `python` si la commande n'est pas disponible
- On crée un dossier test et on lance opencode
- En mode plan : "Fait un plan pour créer un script python simple de calculatrice"
- En mode build : "Execute le plan"
- En mode build : "Teste le programme"
- Lancer le programme et tester

Etape 2 : La doc

- En mode build : "Fait le README du projet"
- On crée la doc **AGENTS.md** pour le llm avec la commande **/init**
- Dès qu'on rouvre opencode, il prend connaissance du **AGENTS.md**
- On ferme / on ouvre et on lui demande de quoi parle le projet

Etape 3 : intégration d'outils avec des skills

- Manuellement dans `~/.config/opencode/skills` ou dans `~/.agents/skills`
- Automatique via https://www.skills.sh/ et NodeJS
- Skills uv, ruff et ty
- `npx skills add https://github.com/astral-sh/claude-code-plugins -g -a opencode -all -y`
- `/skills` dans opencode pour les charger
- En mode build : "Intègre uv, ty et ruff"
- En mode build : "Lance ruff et ty"
- On teste et on met à jour notre AGENTS.md avec la commande **/init**

Etape 4 : Tests unitaires

- En mode plan : "Prépare le plan pour écrire les tests unitaires"
- En mode build : "Execute le plan"
- En mode review : "Fait une revue du code"
- En mode build : "Fait les correctifs indispensables"

Etape 5 : Interface web

- On installe le skill de streamlit
- `npx skills add https://github.com/streamlit/agent-skills --skill developing-with-streamlit -g -a opencode -all -y`
- En mode plan : "Prépare une interface streamlit pour notre calculatrice"
- Il va automatiquement charger le skill et préparer le plan de dev
- En mode build : "Exécute le plan"
- Il va automatiquement installer streamlit

- En mode review : "Fait une revue du code"
- En mode build : "Fait les correctifs indispensables"
- En mode build : "Donne-moi la commande pour lancer l'application streamlit"
- On teste et s'il y a une erreur on la lui donne pour qu'il corrige
- On teste le projet
- En mode build : "Mets à jour le README et le AGENTS"

Merci, des questions ?