

Travailler avec les APIs

Travailler avec les APIs

Utiliser les interfaces chatbot des LLMs (OpenAi, Anthropic, ...) ne scale pas. Même si les fenêtres de contexte sont de plus en plus grandes (1M de token pour chatGPT 4.1), il y a, à un moment, une limite à la taille des documents que l'on peut traiter.

Dans cette session nous allons mettre en place un environnement de travail qui permette une utilisation programmatique des LLM via leur APIs.

Nous utiliserons

- github.dev
- groq
- les LLM open sources disponibles sur groq

Github.dev permet de lancer un environnement de calcul, une instance, à partir d'une repo github. le principe est archi simple, aller sur une repo github.com (par exemple ...), changer dans l'url le .com en .dev. Vous avez accès à un éditeur de code, vscode, dans votre navigateur ainsi qu'à un terminal qui permet d'exécuter vos scripts. C'est puissant et gratuit!

Groq est un service d'inférence rapide. En ouvrant un compte sur groq, vous avez la possibilité de créer des clés API pour interagir avec des [modèles](#) open sources comme Llama 4, Deepseek R1, QwQ ou Gemma 2. Le nombre de token offert devrait être assez large pour la session d'aujourd'hui. C'est puissant et gratuit!

Déroulé

- commencer par github : créer une repo sur github, ajouter un script simple, instancier github.dev, lancer le terminal et exécuter le script!
- Sur groq.com, ouvrir un compte, créer une clé API, explorer le playground.
- associer les documents sources a la repo github
- faire écrire les codes d'interactions avec les API à un LLM (on peut utiliser chatGPT, claude, Qwen etc en mode chatBOT)

Github

Etape 1: créer la repo

- se connecter ou ouvrir un compte sur github.com
- créer une nouvelle repo
 - sur l'onglet "repositories", cliquer sur "New repository"
 - renseigner
 - nom de la repo
 - repo PUBLIC (important)
 - ajouter un readme
 - un .gitignore (choisir python)
 - pas besoin de licence

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner *

 SkatAI ▾

Repository name *

/ social

✓ social is available.

Great repository names are short and memorable. Need inspiration? How about **congenial-journey** ?

Description (optional)

Decodage des usages des AI



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Initialize this repository with:



Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: Python ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None ▾

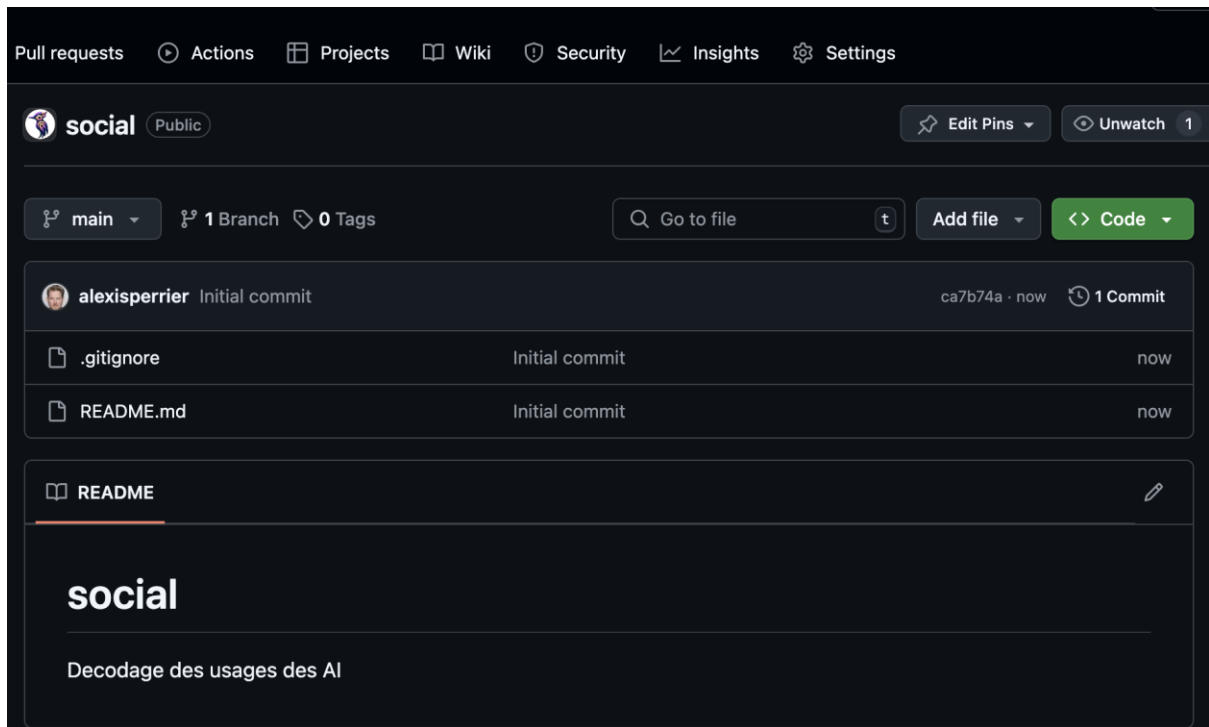
A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set  main as the default branch. Change the default name in SkatAI's [settings](#).

 You are creating a public repository in the SkatAI organization.

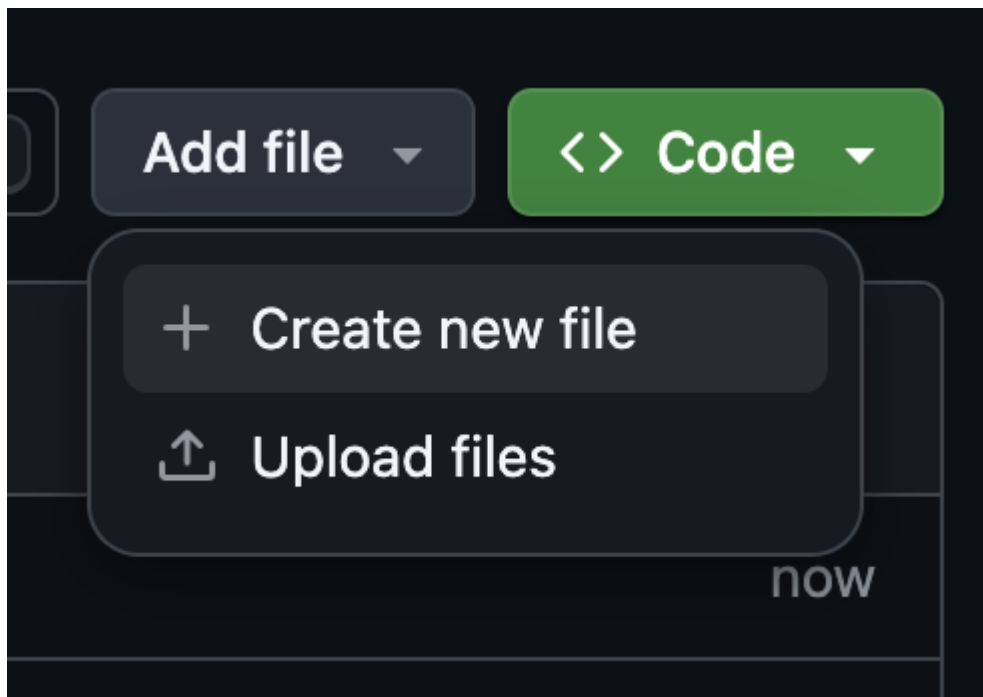
Create repository

Vous obtenez cet écran



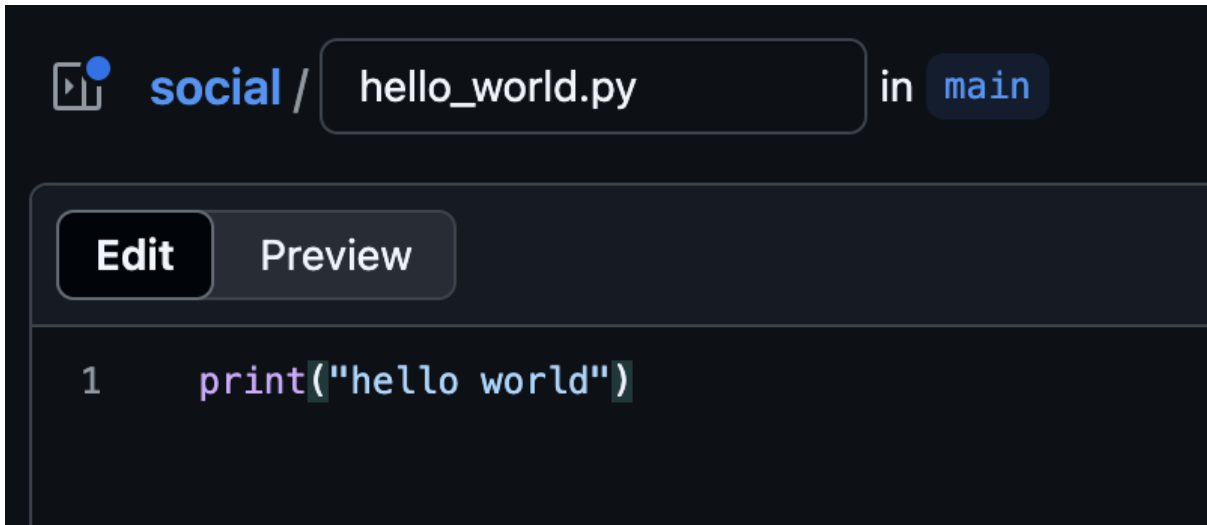
Etape 2: ajouter un script

Github.dev a besoin d'un script pour pouvoir être lancé. On va créer un simple script python hello world (toujours commencer par hello world)



dans l'espace de saisie

- nommer le fichier : hello_world.py
- ajouter le code python print("hello world")



```
social / hello_world.py in main
Edit Preview
1 print("hello world")
```

Git ?

Le processus pour ajouter du code a une repo suit les étapes

1. modifier le code (c'est ce qu'on vient de faire)
2. ajouter le code :
 - `git add .`
 - Attention il y a un point `.` après le `add`
3. décrire la modification :
 - `git commit -m "un message de description du travail"`
4. si on travaille en dehors de github.com: pousser la nouvelle version du code :
 - `git push`
5. pour récupérer le code quand on démarre de zéro :
 - `git pull`

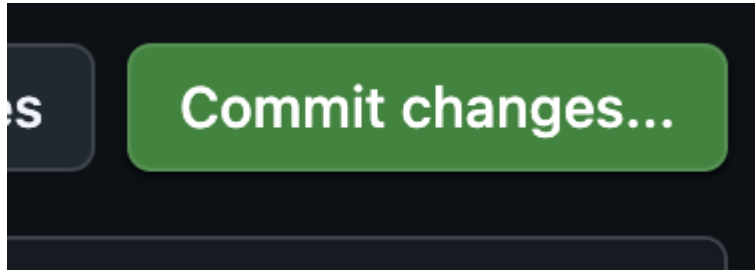
Le monde tourne sur ces 4 commandes!

Quand on travaille sur github.com comme nous venons de le faire il y a seulement l'étape de git commit.

Avec d'autres commandes, (git branch, git merge, git rebase), cela permet à des équipes de développeurs ou d'AI de collaborer sur une base de code commune. C'est super puissant.

Retour a hello world

Une fois fichier nommé et le code écrit, on clique sur commit changes pour



Puis continuer

- attention à bien sélectionner : `Commit directly to the main branch`

Commit changes

Commit message

Create `hello_world.py`

Extended description

Add an optional extended description...

Commit Email

alexis.perrier@gmail.com

☒ Commit directly to the main branch

☐ Create a **new branch** for this commit and start a pull request
[Learn more about pull requests](#)

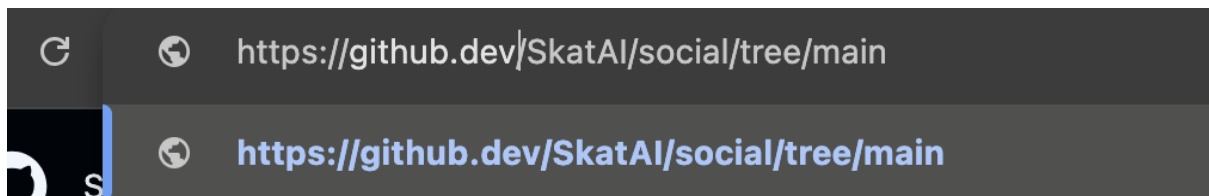
Cancel

Commit changes

Github.dev

On a maintenant une base de code super simple mais qui permet de lancer la machine github.dev

Allez dans la barre de navigation
changez le ".com" en .dev



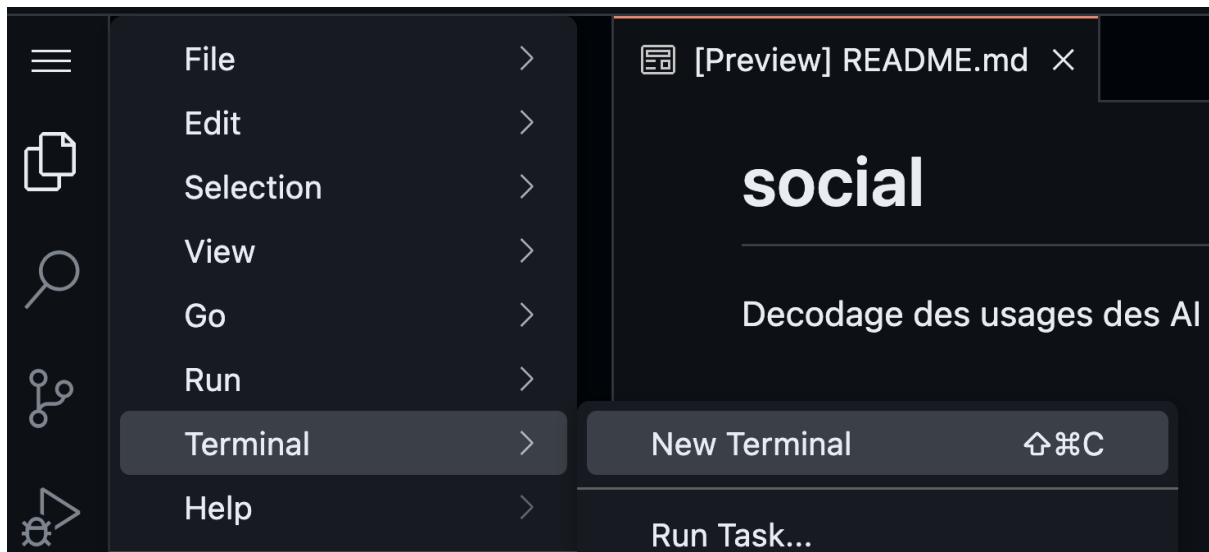
Vous avez un environnement de développement complet

Prenez quelques minutes pour explorer la barre de menu de gauche
Si vous travaillez déjà sur vscode c'est exactement la même interface!

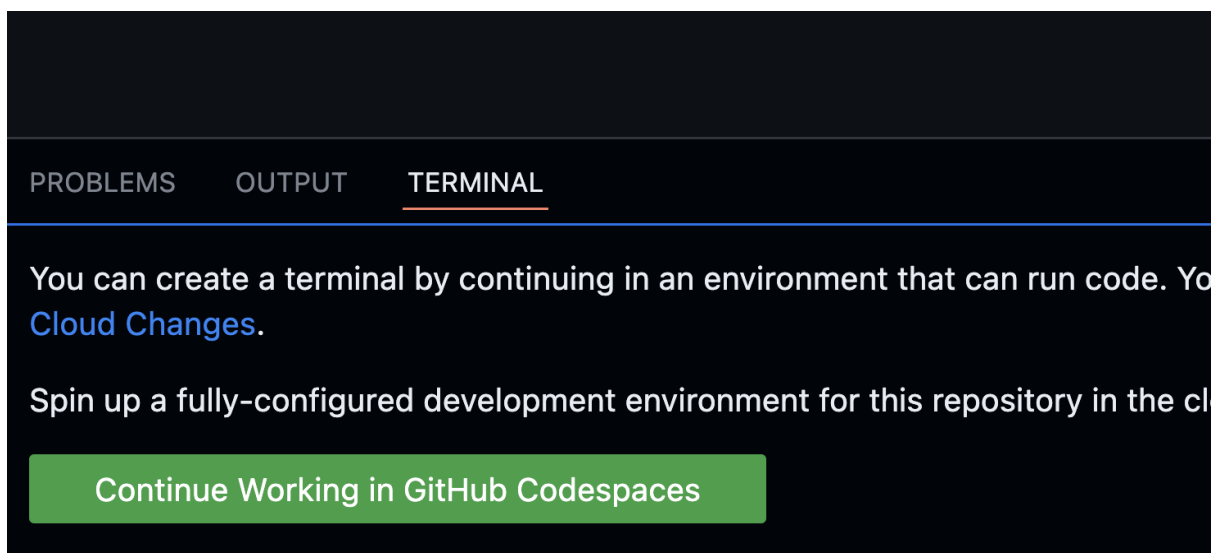
Lancer une machine virtuelle

Pour exécuter le code `hello_world.py` il faut une machine, une instance. sur github ça s'appelle un **codespace**.

On crée et lance cette instance en ouvrant un terminal



Choisir l'option Continue working in GitHub Code Space



Un menu déroulant apparaît avec les différents types de machine disponibles

Select the instance type for your codespace ✓ Usage paid f

2 cores, 8 GB RAM, 32 GB storage

4 cores, 16 GB RAM, 32 GB storage

8 cores, 32 GB RAM, 64 GB storage

16 cores, 64 GB RAM, 128 GB storage

Choisissez la plus puissante

et la! Hyperspace!



Dans le terminal,

```
> python hello_world.py
```

```
👋 Welcome to Codespaces! You are on our default image.
- It includes runtimes and tools for Python, Node.js, Docker, and more. See the
- Want to use a custom image instead? Learn more here: https://aka.ms/configure

🔍 To explore VS Code to its fullest, search using the Command Palette (Cmd/Ctrl +

📝 Edit away, run your app as usual, and we'll automatically make it available for

@alexisperrier → /workspaces/social (main) $ python hello_world.py
hello world
@alexisperrier → /workspaces/social (main) $ █
```

Re - BAZINGA!

Le terminal

Dans ce terminal, exécutez les commandes suivantes

- ls -al
- pwd
- python --version
- cat hello_world.py
- git log

Un nouveau fichier

Ajoutez un nouveau fichier, nommé bonjour.py avec le code
`print("Bonjour tout le monde")`

Le code est modifié dans l'instance de travail, mais pas sur github.

Dans le terminal faites

`git status`

```
Date: Mon Apr 28 09:35:54 2025 +0200

Initial commit
@alexisperrier → /workspaces/social (main) $ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    bonjour.py

nothing added to commit but untracked files present (use "git add" to track)
@alexisperrier → /workspaces/social (main) $
```

ce qui indique que le fichier bonjour.py est untracked. Il est juste dans l'environnement de développement mais pas encore enregistré dans la repo github

dans l'ordre, exécutez

- `git add .`
- `git commit -m "wip"`
- `git push`

puis `git status` pour vérifier que le code dans le Codespace et le code sur github sont bien synchronisés.

allez sur votre repo dans github et vérifiez que le nouveau fichier y apparaît bien

Groq.com

- Allez sur groq.com et ouvrez un compte
- DEV CONSOLE en haut à droite

Vous pouvez créer un compte via "se logger avec github" ou en renseignant votre email. Un email de création de compte vous est envoyé si vous n'avez pas déjà de compte.

Playground

Une fois le compte créé, allez sur le [playground](#) et expérimentez avec des prompts et des modèles différents

Puis créez une clé API

Gestion des clés API

Important!

Ne jamais pousser les clés API sur github. Elles seront détectées par des bots.

La bonne façon de gérer des clés API

- créer un fichier `.env`
- y écrire les clés
- ajouter le fichier `.env` à votre `.gitignore`
- dans le code python utiliser la librairie `dotenv` pour charger les clés

Une première requête

Dans chatGPT rédiger un prompt pour envoyer une requête vers un modèle dispo via l'API groq

Par exemple

Je travaille dans une repo `github.dev` en python et avec groq et llama 4

Ecrit le code pour

- charger la clé API à partir du fichier `.env` en utilisant `dotenv`
- faire une requête a groq
- afficher le résultat

Écrit aussi le fichier de requirements pour installer les librairies

- Copier coller le code dans un fichier `groq.py`
- copier coller les noms des librairies dans un fichier `requirements.txt`

puis dans le terminal

```
pip install -r requirements.txt
```

et

```
python groq.py
```

Si le modèle n'est pas disponible, allez dans le playground et choisissez un autre nom de modèle.

Llama 4 Scout 17B 16E

Search Models...

Alibaba Cloud

QwQ 32B

DeepSeek / Meta

DeepSeek R1 Distill Llama 70B

Google

Gemma 2 Instruct

Groq

Compound Beta

Compound Beta Mini

Hugging Face

Distil Whisper

Meta

Llama 3.1 8B

QwQ 32B

qwq-qwq-32b

Copy model id

PRICING	Input Token	Output Token
	\$0.29	\$0.39
	3.45M / \$1	2.56M / \$1

LIMITS	Requests
	30 / minute
	1k / day

RELEASED	March 5, 2025
----------	---------------

Charger le corpus

Il y a 2 façons de charger les fichiers de corpus

- soit donner accès directement au google doc
- soit ajouter les documents sous forme texte ou pdf dans la repo.
 - créer un repertoire data

Pour commencer on peut demander au LLM de simplement lire le document et le décrire

Puis de lui demander une analyse plus approfondie

Automatiser l'analyse

- Lire les documents
- Pour chaque document analyser, extraire

- Sauver les resultats
- Sauver le code: git add., git commit, git push

Iterate

Insert appropriate <rinse and repeat> meme