
hellowork
NOTRE JOB, VOUS AIDER À CHOISIR LE VÔTRE

Fête des démos **avec Gradio !**

hello work

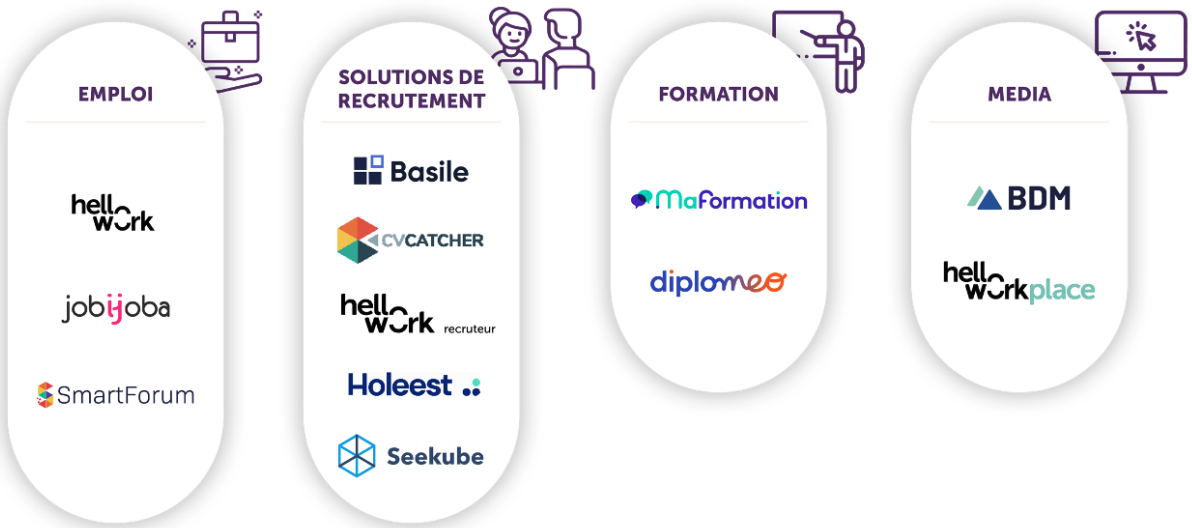
Hello ! 👋

Fabien Ric

Développeur > Product Manager > Data Scientist

- 👷 Script&Go (Evodia)
- ♠ Capgemini
- ✍️ MyScript
- 🪪 IDnow (AriadNEXT)
- 💼 HelloWork

HelloWork



4 millions d'utilisateurs par mois

services utilisés par **40 000** professionnels chaque mois

46 millions de mises en relations par an

500 collaborateurs

+ 33 % de croissance en 2022

La Data Science chez HW

Analyse des offres

Analyse des CVs

Algo de matching et recommandation

R&D transverse

Référentiels

Motivations

- Tester mon modèle avec d’autres données que mes jeux de tests
- Faire varier des paramètres et voir le résultat de manière interactive
- Faire utiliser mon modèle à une autre personne / une autre équipe
- Faire une démo du modèle en réunion sans montrer de code
- Construire une IHM rapidement
- …

Gradio

Librairie Python développée par HuggingFace.

Créer et déployer des IHMs autour d'un modèle ML en quelques lignes.

https://gradio.app/

https://github.com/gradio-app/gradio ⭐ **23.6 K**

```
pip install gradio
```

Python 3.8 minimum

Intégration : scripts Python, notebooks Jupyter, Colab etc., pages Web

Cas d’usages

Démos de modèles machine learning

Mais aussi:

- chatbot
- visualisation de données
- moteur de recherche
- outil d’annotation
- …

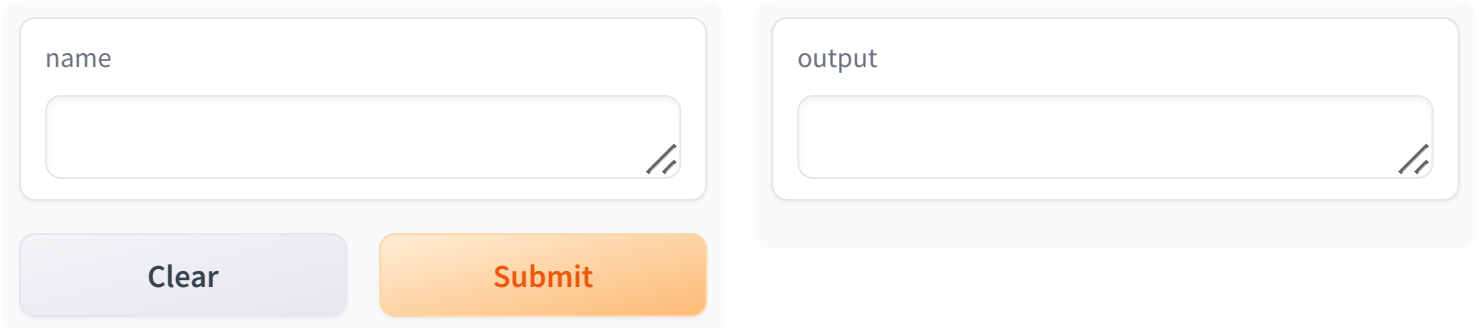
Fonctionnement général

Hello World

```
def greet(name):
    return "Hello " + name + "!"
```

Hello World

```
import gradio as gr

def greet(name):
    return "Hello " + name + "!"

demo = gr.Interface(fn=greet,
                    inputs="text",
                    outputs="text")

demo.launch()
```



Interface

Le coeur d'une application Gradio.

```
demo = gr.Interface(fn=greet,
                    inputs="text",
                    outputs="text")
```

- **fn**: la fonction à appeler lors du submit (ex: model.predict)
- **inputs**: la / les entrées de la fonction (*"text"*, *"image"*, *"number"*...)
- **outputs** : la / les sorties de la fonction (*"text"*, *"label"*, *"image"*...)

"text" est un raccourci pour le composant **gr.Textbox**

Composants ?

```
demo = gr.Interface(
  ner,
  inputs = [gr.Textbox(placeholder="Enter sentence here..."),
            gr.Radio(choices=["token", "entity"],
                     label="mode",
                     value="entity"),
            gr.Dropdown(choices=["job", "certif"],
                        multiselect=True,
                        label="model",
                        value="job")
            ]
  outputs=gr.HighlightedText()
)
```

Composants ?


text
Recherche Cow-Boy expérimenté et motivé pour attraper des bandits.
mode
token
entity
model
job
Clear
Submit


output
Recherche
Cow-Boy
JOBTITLE
expérimenté et motivé pour attraper des bandits.
Flag

Composants disponibles

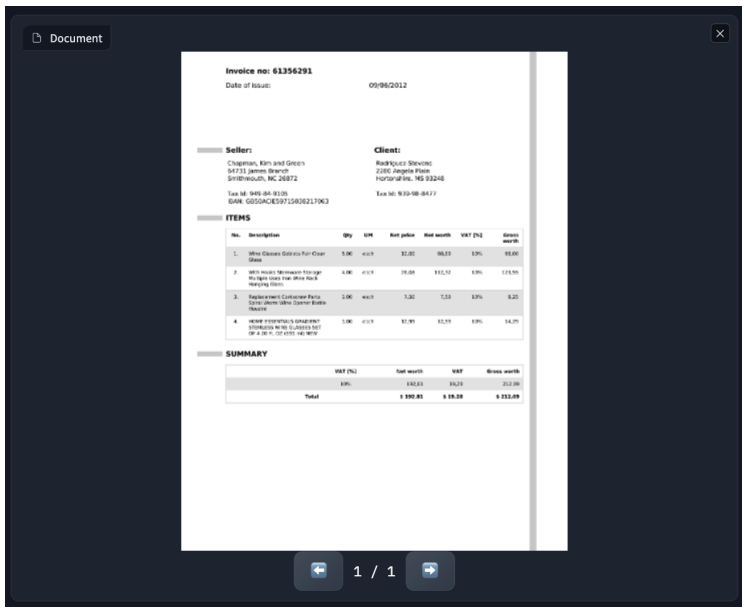
35+ composants :

- HighlightedText (NER)
- Label (classification)
- ChatBot
- DataFrame
- Slider
- Radio
- Dropdown
- Checkbox
- Image (upload ou visualisation)
- Audio
- Video
- ...

Custom Components

Nouveau en 4.0

Ecrire et publier ses propres composants (ex: lecteur PDF, map, ...)



Personnaliser l’interface

Pré-remplir avec des exemples :

```
demo = gr.Interface(
  ner,
  inputs = ["text", "checkbox"],
  outputs = "text",
  examples = [["example 1", True], ["example 2", False] ... ]
)
```

Changer le thème :

```
demo = gr.Interface(theme=gr.themes.Soft(), ...)
```

Autres formes d’interface

ChatInterface

Une interface de chatbot en 1 ligne :

```
import gradio as gr

def echo(message, history):
    return f"Echo {message}"

gr.ChatInterface(echo).launch()
```

TabbedInterface

Plusieurs onglets

```
gradio.TabbedInterface(interface_list, ···)
```

Blocks

Contrôle sur le layout et le comportement de l’application :

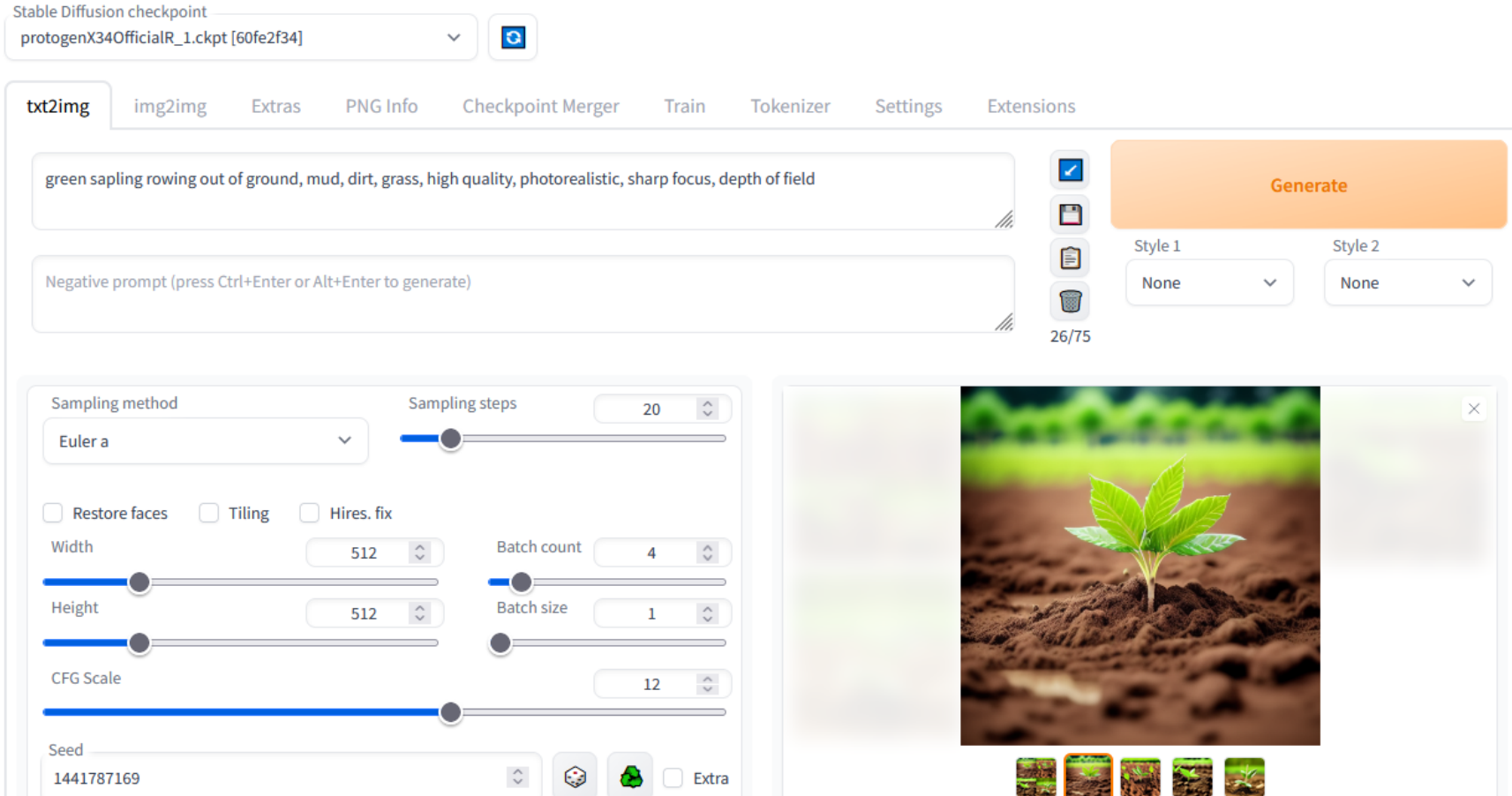
- interface en plusieurs colonnes
- paralléliser les entrées vers plusieurs modèles
- réagir à un event

Blocks - exemple

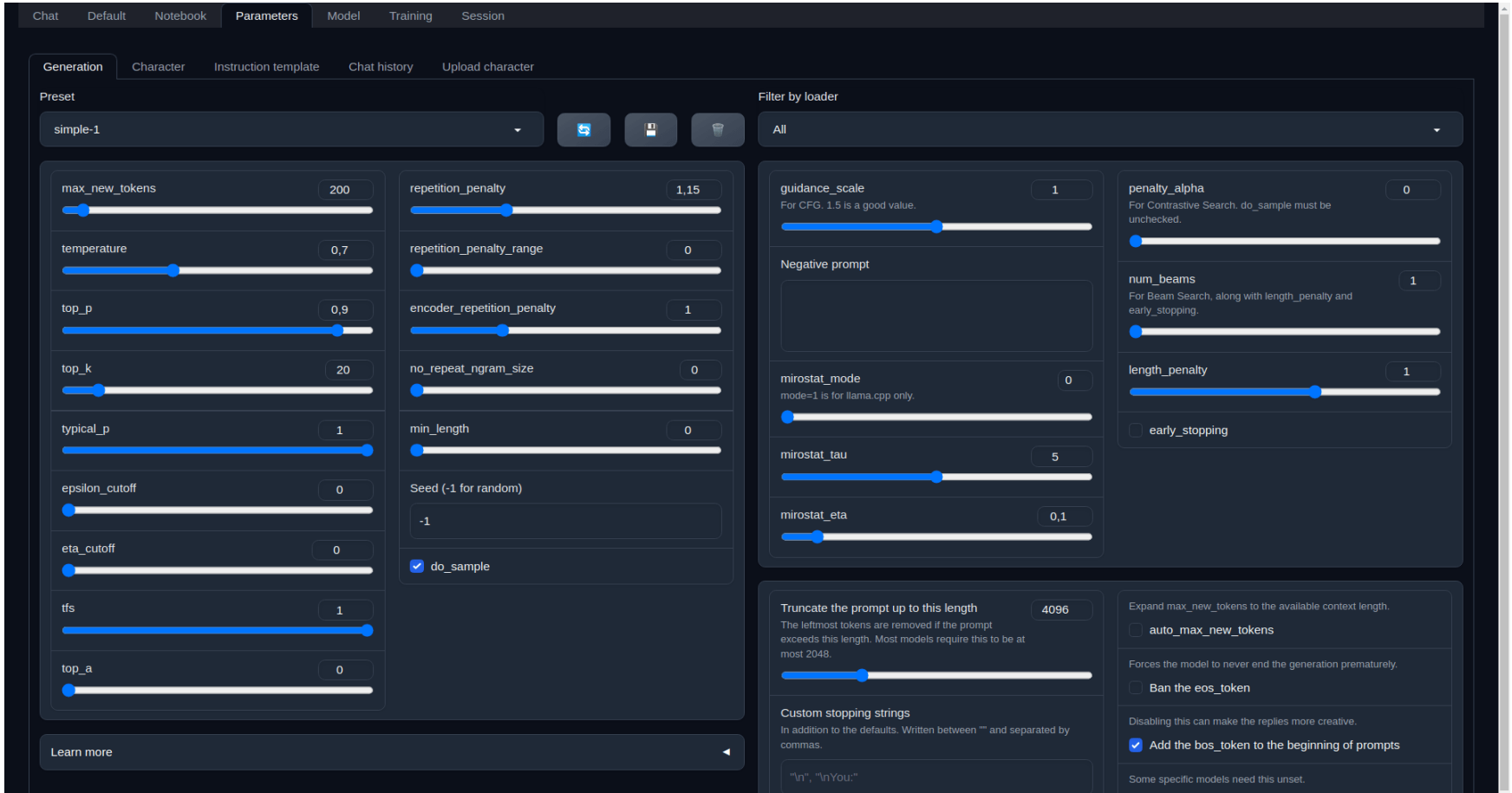
```
import gradio as gr

def greet(name):
    return "Hello " + name + "!"

with gr.Blocks() as demo:
    name = gr.Textbox(label="Name")
    output = gr.Textbox(label="Output Box")
    greet_btn = gr.Button("Greet")
    greet_btn.click(fn=greet, inputs=name, outputs=output)

demo.launch()
```

Exemples d’applications

Applications complexes



Applications complexes



Déploiement

En local

- Lancer un serveur local pour test/démo :

```
demo.launch()
```

Public

- lancer un serveur local + remote control sur une URL **publique** XXXXX.gradio.app valable 72h.

```
demo.launch(share=True)
```

- Héberger sur Spaces HuggingFace

Déploiement

Interne entreprise

- Déploiement Docker sur K8s
- Gradio Share Server (nouveau en 4.0) : permet de montrer son propre serveur de partage

```
import gradio as gr

app = gr.Interface(lambda x: x, "text", "text")
app.launch(share=True, share_server_address="mycorp-gradio-share.com:7000")
```

```
Running on public URL: http://07f56cd0f87061c8a1.mycorp-gradio-share.com
```

Ressources

Démarrer ici : https://www.gradio.app/quickstart/

Doc de référence : https://www.gradio.app/docs/

Playground : https://www.gradio.app/playground

A vous de jouer :)