Guide ML Sandbox

Configuration et Outils

Container Docker pour le Machine Learning en Python

Repository : github.com/ogautier1980/sandbox-ml

Version 0.1 – Janvier 2026

Résumé

Ce guide présente l'installation et l'utilisation du container Docker **ML Sandbox**, un environnement complet pour le Machine Learning en Python. Il inclut PyTorch, TensorFlow, scikit-learn, et plus de 100 packages pour le data science, le traitement de documents, l'OCR, et la visualisation.

Composant	**Contenu**
Python	3.11
ML/Deep Learning	PyTorch, TensorFlow, scikit-learn, XGBoost
Data	NumPy, Pandas, Polars
Documents	LaTeX, LibreOffice, Pandoc
OCR	Tesseract (FR/EN), EasyOCR
Web/API	FastAPI, Streamlit, Gradio

Table des matières

Première partie

Guide de Configuration

1 Prérequis

Avant de commencer, assurez-vous d'avoir installé les logiciels suivants :

Logiciel	Téléchargement	Vérification
Docker Desktop	docker.com/products/docker-desktop	`docker -version`
VS Code	code.visualstudio.com	`code -version`
Git	git-scm.com	`git -version`

Mémoire Docker

Docker Desktop utilise par défaut 2 GB de RAM. Pour les packages ML, augmentez à **8 GB minimum** (16 GB recommandé) dans Docker Desktop → Settings → Resources.

2 Méthode 1 : VS Code Dev Container (Recommandée)

Cette méthode ouvre VS Code directement dans le container Docker avec toutes les extensions pré-configurées.

2.1 Étape 1 : Installer l'extension Dev Containers

1. Ouvrir VS Code
2. Aller dans Extensions (`Ctrl+Shift+X`)
3. Rechercher **"Dev Containers"** (Microsoft)
4. Cliquer sur **Installer**

2.2 Étape 2 : Cloner le repository

Option A – Via terminal :

```
git clone https://github.com/ogautier1980/sandbox-ml.git
cd sandbox-ml
code .
```

Option B – Via VS Code :

1. Ouvrir VS Code
2. `Ctrl+Shift+P` → taper "Git : Clone"
3. Coller l'URL : `https://github.com/ogautier1980/sandbox-ml.git`
4. Choisir un dossier de destination
5. Cliquer "Open" quand VS Code propose d'ouvrir le repository

2.3 Étape 3 : Ouvrir dans le container

1. VS Code détecte automatiquement le fichier `.devcontainer/devcontainer.json`
2. Une notification apparaît : *"Folder contains a Dev Container configuration file..."*
3. Cliquer sur **"Reopen in Container"**

Si la notification n'apparaît pas

`Ctrl+Shift+P` → taper "Dev Containers : Reopen in Container"

2.4 Étape 4 : Attendre le build

Le premier lancement prend **10-20 minutes** car Docker doit :

— Télécharger l'image Python 3.11
— Installer les packages système (LaTeX, LibreOffice, FFmpeg, etc.)
— Installer les packages Python (∼100 packages)
— Configurer les extensions VS Code

2.5 Étape 5 : Accéder à Jupyter Lab

Jupyter Lab démarre automatiquement sur le port 8888.

1. Ouvrir un navigateur
2. Aller à : `http://localhost:8888`
3. Jupyter Lab s'ouvre sans mot de passe

3 Méthode 2 : Docker Compose

Pour utiliser le container sans VS Code Dev Containers.

3.1 Démarrer le container

Windows :

```
git clone https://github.com/ogautier1980/sandbox-ml.git
cd sandbox-ml
start.bat
```

Linux / macOS :

```
git clone https://github.com/ogautier1980/sandbox-ml.git
cd sandbox-ml
chmod +x start.sh
./start.sh
```

Ou directement :

```
docker-compose up -d --build
```

3.2 Accéder aux services

Service	Port	URL
Jupyter Lab	8888	http://localhost:8888
TensorBoard	6006	http://localhost:6006
MLflow	5000	http://localhost:5000
Streamlit	8501	http://localhost:8501
Gradio	7860	http://localhost:7860

4 Vérification de l'installation

4.1 Python et packages ML

```python
import torch
import tensorflow as tf
import sklearn
import pandas as pd
import numpy as np

print(f"PyTorch: {torch.__version__}")
print(f"TensorFlow: {tf.__version__}")
print(f"scikit-learn: {sklearn.__version__}")
print(f"CUDA disponible: {torch.cuda.is_available()}")
```

4.2 Outils système

```
pdflatex --version      # LaTeX
libreoffice --version   # LibreOffice
tesseract --version     # OCR
convert --version       # ImageMagick
ffmpeg -version         # FFmpeg
pandoc --version        # Pandoc
```

5 Commandes utiles

5.1 Gestion du container

```
# Demarrer
docker-compose up -d

# Arreter
docker-compose down

# Redemarrer
docker-compose restart

# Voir les logs
```

```
docker-compose logs -f ml-sandbox

# Acceder au shell
docker exec -it ml-sandbox bash

# Reconstruire (apres modif Dockerfile)
docker-compose build --no-cache && docker-compose up -d
```

5.2 Services optionnels

```
# Demarrer avec TensorBoard
docker-compose --profile tensorboard up -d

# Demarrer avec MLflow
docker-compose --profile mlflow up -d
```

6 Résolution de problèmes

6.1 Port déjà utilisé

Modifier le port dans `docker-compose.yml` :

```
ports:
  - "8889:8888"  # Changer 8888 en 8889
```

6.2 Mémoire insuffisante

1. Ouvrir Docker Desktop → Settings → Resources
2. Augmenter Memory à **8 GB minimum**
3. Augmenter Swap à **4 GB**
4. Cliquer “Apply & Restart”

6.3 Permissions sur Linux/macOS

```
sudo chown -R $USER:$USER notebooks/ data/ models/ src/
```

Deuxième partie

Guide des Outils

7 Outils Système (Linux)

7.1 Compression / Archives

Outil	Description	Compression	Décompression
zip/unzip	Format ZIP standard	`zip -r a.zip dir/`	`unzip a.zip`
p7zip	Format 7z (haute compression)	`7z a a.7z dir/`	`7z x a.7z`
tar+gzip	Archive .tar.gz	`tar -czvf a.tar.gz dir/`	`tar -xzvf a.tar.gz`
tar+bzip2	Archive .tar.bz2	`tar -cjvf a.tar.bz2 dir/`	`tar -xjvf a.tar.bz2`
tar+xz	Archive .tar.xz	`tar -cJvf a.tar.xz dir/`	`tar -xJvf a.tar.xz`

7.2 ImageMagick

Manipulation d'images en ligne de commande.

```
# Conversion de format
convert image.png image.jpg

# Redimensionner
convert image.png -resize 800x600 output.png
convert image.png -resize 50% output.png

# Rotation
convert image.png -rotate 90 output.png

# Creer un PDF a partir d'images
convert *.jpg output.pdf

# Creer un GIF anime
convert -delay 100 -loop 0 frame*.png animation.gif

# Ajouter du texte
convert image.png -pointsize 36 -fill white -annotate +50+50 'Texte' output.png
```

7.3 FFmpeg

Manipulation audio/vidéo.

```
# Conversion de format
ffmpeg -i video.avi video.mp4

# Extraire l'audio d'une video
```

```
ffmpeg -i video.mp4 -vn audio.mp3

# Extraire une image d'une video
ffmpeg -i video.mp4 -ss 00:01:30 -frames:v 1 screenshot.png

# Couper une video
ffmpeg -i video.mp4 -ss 00:00:30 -to 00:01:00 -c copy extrait.mp4

# GIF a partir d'une video
ffmpeg -i video.mp4 -vf "fps=10,scale=320:-1" output.gif
```

7.4 PDF Tools (poppler-utils)

```
# PDF vers images
pdftoppm document.pdf output -png

# PDF vers texte
pdftotext document.pdf
pdftotext -layout document.pdf output.txt

# Informations sur un PDF
pdfinfo document.pdf

# Fusionner des PDFs
pdfunite file1.pdf file2.pdf merged.pdf

# Separer un PDF
pdfseparate document.pdf page_%d.pdf
```

7.5 LibreOffice (Mode Headless)

```
# Word -> PDF
libreoffice --headless --convert-to pdf document.docx

# PowerPoint -> PDF
libreoffice --headless --convert-to pdf presentation.pptx

# Excel -> CSV
libreoffice --headless --convert-to csv spreadsheet.xlsx

# Word -> Texte brut
libreoffice --headless --convert-to txt document.docx

# Conversion par lot
libreoffice --headless --convert-to pdf *.docx
```

7.6 Tesseract OCR

```
# OCR basique (anglais par defaut)
tesseract image.png output

# OCR en francais
tesseract image.png output -l fra

# OCR multilingue
tesseract image.png output -l fra+eng

# Sortie en PDF searchable
tesseract image.png output -l fra pdf
```

7.7 Sox (Audio)

```
# Informations sur un fichier audio
soxi audio.wav

# Conversion de format
sox audio.wav audio.mp3

# Modifier le volume
sox audio.wav output.wav vol 0.5

# Couper un extrait (de 10s, duree 30s)
sox audio.wav output.wav trim 10 30

# Concatener des fichiers
sox audio1.wav audio2.wav output.wav
```

8 Machine Learning & Deep Learning

8.1 scikit-learn

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# Preparation des donnees
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Entrainement
model = RandomForestClassifier(n_estimators=100)
model.fit(X_train, y_train)

# Prediction et evaluation
predictions = model.predict(X_test)
print(f"Accuracy: {accuracy_score(y_test, predictions):.2f}")
```

```
print(classification_report(y_test, predictions))
```

8.2 PyTorch

```
import torch
import torch.nn as nn
import torch.optim as optim

class SimpleNN(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super().__init__()
        self.fc1 = nn.Linear(input_size, hidden_size)
        self.relu = nn.ReLU()
        self.fc2 = nn.Linear(hidden_size, output_size)

    def forward(self, x):
        return self.fc2(self.relu(self.fc1(x)))

# Creer le modele
model = SimpleNN(784, 128, 10)
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

# GPU support
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model.to(device)
```

8.3 XGBoost

```
import xgboost as xgb

model = xgb.XGBClassifier(
    n_estimators=100,
    max_depth=6,
    learning_rate=0.1,
    objective='binary:logistic'
)
model.fit(X_train, y_train)

# Feature importance
xgb.plot_importance(model)
```

9 Data Processing

9.1 Pandas

```
import pandas as pd
```

```
# Lecture de fichiers
df = pd.read_csv('data.csv')
df = pd.read_excel('data.xlsx')
df = pd.read_parquet('data.parquet')

# Exploration
df.head()
df.info()
df.describe()

# Selection et filtrage
df['colonne']
df[df['age'] > 25]

# Agregation
df.groupby('ville')['age'].mean()

# Export
df.to_csv('output.csv', index=False)
df.to_parquet('output.parquet')
```

9.2 Polars (Alternative rapide)

```
import polars as pl

df = pl.read_csv('data.csv')

result = (
    df.lazy()
    .filter(pl.col('age') > 25)
    .select(['nom', 'age'])
    .with_columns([
        (pl.col('age') * 2).alias('age_double')
    ])
    .collect()
)
```

10 Visualisation

10.1 Matplotlib

```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
plt.plot(x, y, label='Donnees', color='blue')
plt.xlabel('X')
```

```
plt.ylabel('Y')
plt.title('Mon Graphique')
plt.legend()
plt.grid(True)
plt.savefig('graph.png', dpi=300)
plt.show()
```

10.2 Seaborn

```
import seaborn as sns

sns.set_theme(style="whitegrid")

# Distribution
sns.histplot(data=df, x='age', kde=True)

# Relations
sns.scatterplot(data=df, x='x', y='y', hue='category')

# Heatmap
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
```

10.3 Plotly (Interactif)

```
import plotly.express as px

fig = px.scatter(df, x='x', y='y', color='category',
                 hover_data=['name'], title='Mon Plot')
fig.show()
fig.write_html('graph.html')
```

11 NLP & LLM

11.1 Transformers (Hugging Face)

```
from transformers import pipeline

# Analyse de sentiment
classifier = pipeline("sentiment-analysis")
result = classifier("I love this product!")

# Generation de texte
generator = pipeline("text-generation", model="gpt2")
text = generator("Once upon a time", max_length=100)

# Question-Answering
qa = pipeline("question-answering")
```

```
result = qa(question="What is ML?", context="Machine learning is...")
```

11.2 spaCy

```
import spacy

nlp = spacy.load("fr_core_news_md")
doc = nlp("Apple cherche a acheter une startup.")

# Tokens
for token in doc:
    print(token.text, token.pos_, token.lemma_)

# Entites nommees
for ent in doc.ents:
    print(ent.text, ent.label_)
```

12 PDF & Documents

12.1 pypdf

```
from pypdf import PdfReader, PdfWriter, PdfMerger

# Lecture
reader = PdfReader("document.pdf")
for page in reader.pages:
    print(page.extract_text())

# Fusionner des PDFs
merger = PdfMerger()
merger.append("doc1.pdf")
merger.append("doc2.pdf")
merger.write("merged.pdf")
```

12.2 python-docx

```
from docx import Document

doc = Document()
doc.add_heading('Mon Document', 0)
doc.add_paragraph('Texte normal')
doc.add_paragraph('Item 1', style='List Bullet')
doc.save('output.docx')
```

12.3 pytesseract (OCR Python)

```python
import pytesseract
from PIL import Image

# OCR basique
text = pytesseract.image_to_string(Image.open('image.png'))

# Avec langue
text = pytesseract.image_to_string(Image.open('image.png'), lang='fra')

# PDF searchable
pdf = pytesseract.image_to_pdf_or_hocr(Image.open('image.png'), extension='pdf')
with open('output.pdf', 'wb') as f:
    f.write(pdf)
```

13 Web & API

13.1 FastAPI

```python
from fastapi import FastAPI
from pydantic import BaseModel

app = FastAPI()

class Item(BaseModel):
    name: str
    price: float

@app.get("/")
def read_root():
    return {"message": "Hello World"}

@app.post("/items/")
def create_item(item: Item):
    return {"name": item.name, "price": item.price}

# Lancer: uvicorn main:app --reload --host 0.0.0.0
```

13.2 Streamlit

```python
import streamlit as st

st.title("Mon Application ML")

name = st.text_input("Votre nom")
age = st.slider("Age", 0, 100, 25)

uploaded_file = st.file_uploader("Fichier CSV")
```

```
if uploaded_file:
    df = pd.read_csv(uploaded_file)
    st.dataframe(df)

# Lancer: streamlit run app.py
```

13.3 Gradio

```
import gradio as gr

def predict(text):
    return f"Prediction pour: {text}"

demo = gr.Interface(
    fn=predict,
    inputs=gr.Textbox(label="Texte"),
    outputs=gr.Textbox(label="Resultat"),
    title="Mon Modele ML"
)

demo.launch(server_name="0.0.0.0")
```

14 ML Utilities

14.1 MLflow

```
import mlflow

mlflow.set_experiment("mon_experiment")

with mlflow.start_run():
    mlflow.log_param("n_estimators", 100)
    model.fit(X_train, y_train)
    mlflow.log_metric("accuracy", accuracy)
    mlflow.sklearn.log_model(model, "model")
```

14.2 Optuna

```
import optuna

def objective(trial):
    n_estimators = trial.suggest_int('n_estimators', 50, 500)
    max_depth = trial.suggest_int('max_depth', 3, 15)

    model = XGBClassifier(n_estimators=n_estimators, max_depth=max_depth)
    scores = cross_val_score(model, X, y, cv=5)
    return scores.mean()
```

```
study = optuna.create_study(direction='maximize')
study.optimize(objective, n_trials=100)
print(study.best_params)
```

14.3 SHAP (Explainability)

```
import shap

explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(X_test)

shap.summary_plot(shap_values, X_test)
shap.force_plot(explainer.expected_value, shap_values[0], X_test.iloc[0])
```

A Récapitulatif des Ports

Port	Service	URL
8888	Jupyter Lab	http://localhost:8888
6006	TensorBoard	http://localhost:6006
5000	MLflow	http://localhost:5000
8000	FastAPI (uvicorn)	http://localhost:8000
8501	Streamlit	http://localhost:8501
7860	Gradio	http://localhost:7860

B Structure du Projet

```
sandbox-ml/
 .devcontainer/        # Config Dev Container
 .vscode/              # Config VS Code
 notebooks/            # Notebooks Jupyter
 data/                 # Datasets (gitignored)
 models/               # Modeles sauvegardes (gitignored)
 src/                  # Code source Python
 logs/                 # Logs TensorBoard (gitignored)
 mlruns/               # Runs MLflow (gitignored)
 docs/                 # Documentation PDF
 Dockerfile
 docker-compose.yml
 requirements.txt
 README.md
 claude.md
 tools.md
 config.md
```