⚙️ Configuration de Windows pour la masterclass Data

Ce document vous guide dans la configuration d'un PC Windows pour la masterclass Data avec **Python**, **pyenv**, **TensorFlow**, **pandas**, **WSL (Windows Subsystem for Linux)**, et **Jupyter**.

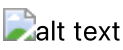
---

1. 🛠️ Vérification préliminaire: La virtualisation de windows

WSL2, ou Windows Subsystem for Linux 2, est une fonctionnalité qui permet d'exécuter un environnement Linux directement sur Windows. Le but est simple, utiliser les outils et les commandes Linux 🐧 , sans jamais quitter votre ordinateur Windows 🖥️ .

Pour utiliser WSL 2, la virtualisation doit être activée sur votre machine. Voici comment vérifier si c'est le cas.

- Tapez `Windows` + `R`
- Ecrivez `taskmgr`
- Tapez `Enter`
- Cliquer sur `Performance`
- Cliquez sur `CPU`

 alt text

Si vous voyez Enabled comme sur l'image précedente → ✅ Sinon il va falloir l'**activer** pour continuer.

Pour ce faire activier la virtualisation sur Windows 11 ou 10:

1. **Redémarrez votre ordinateur et accédez au BIOS/UEFI** :
    - Pendant le démarrage, appuyez sur la touche indiquée pour accéder au BIOS/UEFI. Les touches courantes sont `Esc`, `F2`, `F10`, `F12`, ou `Del`. La touche exacte peut varier selon le fabricant de votre ordinateur.
2. **Naviguez vers les paramètres de virtualisation** :
    - Une fois dans le BIOS/UEFI, utilisez les touches fléchées pour naviguer dans les menus. Cherchez une section nommée `Advanced`, `Configuration`, `CPU Configuration`, ou similaire.
    - Recherchez une option nommée `Intel VT-x`, `Intel Virtualization Technology`, `AMD-V`, ou similaire.
3. **Activer la virtualisation** :
    - Changez l'option de `Disabled` à `Enabled`.
4. **Sauvegardez et quittez** :
    - Sauvegardez les modifications et quittez le BIOS/UEFI. Cela se fait généralement en appuyant sur `F10`, mais cela peut varier selon le fabricant. Assurez-vous de confirmer les modifications si vous y êtes invité.

5. **Redémarrez votre ordinateur** :
    - L'ordinateur redémarrera avec la virtualisation activée.

Par la suite vérifie de nouveau dans ton gestionnaire des tâches si l'option **enabled** est activé.

2. 🚀 Installation de WSL2 en quelques étapes :

1. **Vérifiez que votre Windows est à jour** 📅 :
    - Vous devez avoir **Windows 10** version 2004 ou plus récent, ou **Windows 11**.
    - Pour vérifier votre version, tapez `winver` dans la barre de recherche Windows.
2. **Activez WSL** 🛠️ :
    - Ouvrez **PowerShell** en tant qu'administrateur (clique droit et "Exécuter en tant qu'administrateur") et tapez la commande suivante :

      ```
      wsl --install
      ```

    - Cette commande installe **WSL** et une distribution Linux par défaut (Ubuntu).
3. **Redémarrez votre PC** 🔄 :
    - Une fois l'installation terminée, redémarrez votre ordinateur pour finaliser la configuration.
4. **Vérifiez la version de WSL** 📋 :
    - Après redémarrage, tapez la commande suivante dans PowerShell pour vous assurer que **WSL2** est bien installé :

      ```
      wsl -l -v
      ```

    - Assurez-vous que votre distribution Linux utilise la version 2.
    - Si ce n'est pas le cas:

      ```
      wsl --set-default-version 2
      ```

5. **Installez d'autres distributions Linux** 🐧 :
    - Ouvrez le **Microsoft Store**, recherchez **Ubuntu** ou une autre distribution Linux, et installez celle que vous préférez.
6. **Lancez Linux** 🎉 :
    - Tapez `wsl` dans la barre de recherche Windows pour lancer votre environnement Linux !

Et voilà ! 🎊 Vous avez maintenant **WSL2** installé, prêt à utiliser Linux directement sur votre machine Windows sans machine virtuelle 🖥️ .

Dans le cas ou tu as une ancienne version de linux, regarde ce lien: → WSL installation

Un lien très utile qui vous sera nécessaire, expliquant l'utilisation de visual studio code avec wsl: → Wsl et vscode

2. ⚙️ Installation des outils en ligne de commande

Git

Pour installer Git, utilisez la commande suivante dans votre terminal WSL :

```
sudo apt update
sudo apt install git
```

curl, jq et d'autres outils

Installez les outils requis :

```
sudo apt install -y curl jq
```

---

3. 🐍 Installation de Python avec pyenv

Installation de pyenv

Pyenv permet de gérer différentes versions de Python sur votre machine, pratique pour maintenir des environnements isolés.

1. Lance la commande suivante:

```
sudo apt-get update;
sudo apt-get install make build-essential libssl-dev zlib1g-dev \
libbz2-dev libreadline-dev libxml2-dev libsqlite3-dev llvm \
libncursesw5-dev xz-utils tk-dev  libxmlsec1-dev libffi-dev liblzma-dev \
python3-dev
```

2. Installez `pyenv` via les commandes suivantes :

```
git clone https://github.com/pyenv/pyenv.git ~/.pyenv
```

3. Configurez votre terminal pour pyenv en ajoutant ces lignes à votre fichier `.bashrc` ou `.zshrc` (selon la distribution que vous avez installée) :

```
echo 'export PYENV_ROOT="$HOME/.pyenv"' >> ~/.bashrc
echo 'command -v pyenv >/dev/null || export PATH="$PYENV_ROOT/bin:$PATH"'
>> ~/.bashrc
echo 'eval "$(pyenv init -)"' >> ~/.bashrc
```

4. Rechargez le terminal :

```
exec "$SHELL"
```

Installer la bonne version de Python avec Pyenv

Maintenant nous allons créer notre environnement de travail.

Avec pyenv tu peux maintenant lister l'ensemble des versions disponibles au telechargements :

```
pyenv install --list
```

Pour notre cours nous allons utiliser la versions 3.12.3

```
pyenv install 3.12.3
```

POur être sûr que tu peux continuer, vérifie que ton installation c'est bien passé:

- Lance la commande

```
pyenv versions
```

 alt text

Si tu vois bien la version dans la reponse du terminal tu peux continuer.

4. 📦 Créer un environnement virtuel

On va créer maintenant un environnement virtuel à partir de cette version de python. Créez un environnement virtuel pour isoler les dépendances du projet :

1. Installez le plugin pyenv-virtualenv :

```
git clone https://github.com/pyenv/pyenv-virtualenv.git $(pyenv
root)/plugins/pyenv-virtualenv
exec bashrc
```

2. Créez et activez un environnement virtuel :

```
pyenv install 3.12.3  # Assurez-vous d'installer la bonne version de
Python
pyenv virtualenv 3.12.3 masterclassData
pyenv activate masterclassData
```

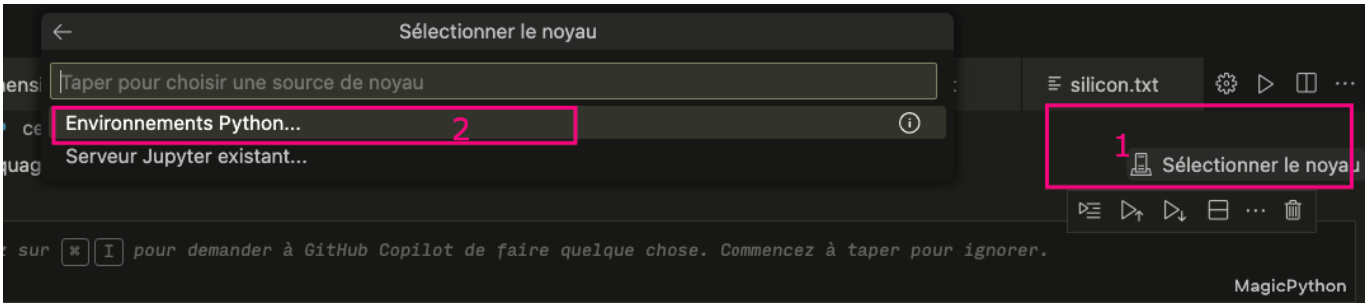
3. Installez les packages nécessaires :

```
pip install -r intel.txt
```

Good Job 👌 Ta machine est prête maintenant pour commencer. Tu peux dès a présent dans vsCode créer un fichier .**ipynb** et choisir le bon kernel afin de te familiariser avec Jupyter.

5. 🫀 📓 Configurer le noyau Jupyter Notebook dans VS Code

Dans vsCode, tu vas devoir selectionner le Kernel que tu souhaites sutiliser. C'est à dire l'environnement de python avec lequel ton notebook doit lancer ses cellules.



Comme sur l'image précedente, tu dois cliquer sur :

1. Selectionner un noyau
2. Environnements Python
3. Et enfin choisir l'environnements que tu as créé avec Pyenv

6. 📚 Des ressources à préparer

- Utilisation de Jupyter
- Les bases de Python
- Choisir le Kernel de VsCode