Linux

Durée

2 à 3h

Objectifs

- [ ] WSL
- [ ] Arborescence système
- [ ] Installer des logiciels
- [ ] lignes de commande, bash
- [ ] Scripts
    - [ ] boucles
    - [ ] chercher un fichier
    - [ ] grep
    - [ ] sed
    - [ ] écrire un fichier de script exécutable

WSL2

Il ne s'agit pas à proprement parler de Linux. Mais d'une façon de faire tourner un système linux dans Windows. Enfin, à ce que j'ai compris, plutôt une émulation, une sorte de machine virtuelle. On n'a pas tout, loin de là. Mais c'est assez bien intégré dans Windows. Et c'est très pratique.

Windows depuis la version 10 permet d'installer un noyeau linux, dans son système Windows, ce qui permet de disposer d'un système linux en ligne de commande (pas d'interface graphique, par défaut) qui peut interagir avec le système Windows. Ca s'appelle WSL (Windows Subsystem Linux).

Installer WSL2

Sur les ordis IDGeo, ça devrait déjà être fait.

Mais sinon, suivre les instructions : https://learn.microsoft.com/fr-fr/windows/wsl/install

Comme le dit la doc, par défaut je crois qu'il installe une ubuntu. Si on veut autre chose, il faut le spécifier.

Installer une distribution linux

Une fois WSL activé, par défaut je crois qu'il installe une ubuntu. Si on veut installer une distrib additionnelle par exemple, on peut passer par le Windows Store. Si si.

Spécificités de WSL

Je vais sûrement en rater, mais en vrac :

- pas d'interface graphique linux, uniquement ligne de commande
- depuis linux, le système de fichiers Windows s'accède via un chemin typiquement linuxien, dans /mnt. Ex. : C:/ se voit depuis le système linux à /mnt/c/

- depuis Windows, le système de fichier linux se trouve dans `\\wsl$\`. Par exemple, mon compte utilisateur dans ma debian se trouvera à `\\wsl$\Debian/home\jean`

Arborescence système Linux

Ca dépend un peu de la distribution utilisée. On va parler ici de Debian/Ubuntu. TODO

Installer des logiciels

Ca dépend un peu de la distribution utilisée. On va parler ici de Debian/Ubuntu.

Linux utilise un système de "stores" depuis bien avant que ça devienne la mode. La plupart des logiciels et librairies supportées par une distribution sont listées dans un grand magasin d'applications, auquel on accède de manière unifiée. On peut ajouter des dépôts pour agrandir la liste si besoin. Ensuite, un outil unique permet de chercher des entrées dans ce magasin, d'installer des logiciels, de faire des mises à jour, etc.

Par exemple, si je veux utiliser la librairie GDAL/OGR dans linux, je vais juste demander à l'installer, et elle sera directement disponible sans complication, variable d'environnement ou quoi :

```
# sudo passe en mode admin le temps de la commande.
# apt est l'outil de gestion du "store".
# update : recharge la liste des applications
sudo apt update

# Si on ne connait pas le nom exact, on peut faire une recherche
sudo apt-cache search gdal

# Et si on le connait, c'est aussi simple que ça :
sudo apt install gdal-bin

# On vérifie que ça marche. Affichons l'aide d'ogr2ogr
ogr2ogr --help
```

Bash

Bash est le shell le plus répandu dans l'environnement linux. Enfin, c'est faux, c'est sh mais il est moins riche.

Vous allez retrouver à peu près toutes les commandes listées pour le powershell, puisqu'elles viennent de bash. Avec plus d'options et de puissance, vous aurez : pwd, cd, ls, mkdir, cp, rm.

Pour afficher le contenu d'un fichier, ça sera

- `head log.txt` pour les 10 premières lignes
- `tail log.txt` pour les 10 dernières

Pour créer un fichier : `touch mon_fichier.md`.

Pour envoyer le résultat d'une commande dans un fichier : `head -n 50 users.csv > users_sample.csv` va créer un fichier limité aux 50 premières lignes.

Télécharger un fichier ? Utilisez `wget` ou `curl`.

Editer un fichier ? `nano` pour les utilisateurs occasionnels, `vim` ou `emacs` pour les ambitieux.

Chercher

- `grep` pour chercher dans un fichier :
  - `grep jean users.csv`
  - `grep -R "jean" .` : cherche dans tous les fichiers à partir du dossier courant
- `sed` pour remplacer :
  - `sed -i "s/jean/jpommier/" users.csv` : chercher/remplacer jean par jpommier
- `find` pour trouver un fichier
  - `find . -name user*.csv` : chercher tous les fichiers user*.csv à partir du dossier courant

Boucles

Par rapport au Powershell, la syntaxe est un peu plus simple :

```
for i in {1..5}
do
    echo The value of i is: $i
done
```

Ou pour une boucle sur une liste de chaines de caractères :

```
employees="Bijay Bhawana Padmini Lakshmi"
for emp in $employees
do
  echo $emp;
done
```

Et enfin, pour reprendre l'exemple précédemment donné :

```
for f in $(ls -Sr midi-pyrenees-osm/*.shp)
do
    ogr2ogr $(basename -- ${f%.*})_09.shp -clipsrc ../departement-
ariege.shp -lco ENCODING=UTF-8 $f ;
done
```

Ecrire un fichier de script exécutable

De base, on tape ces commandes et on les enchaîne depuis le terminal.

Mais ensuite, si on veut reproduire ces actions, il est préférable de les organiser dans un fichier de script. Un fichier de script peut vite devenir sa propre documentation

Un fichier bash doit :

- commencer par la ligne `#!/bin/bash`
- être exécutable

On peut écrire des commentaires en les débutant par `#`

On peut définir des variables (Ex. `ma_var=quelquechose`) et l'appeler avec $ (ex. `echo $ma_var`)

On peut écrire des fonctions pour les opérations répétitives, ou pour structurer le code

Tâches répétitives

Si vous avez les commandes à exécuter à intervalles régulier, vous pouvez utiliser `cron`.

Bash -- exercices

Ecrire un script

Supposons que dans le cadre d'un projet lié au prix du foncier, nous voulons produire un fichier recensant les prix du foncier pour un lot de communes autour de Colomiers. Un fichier par commune.

Notre donnée source sera https://www.data.gouv.fr/fr/datasets/demandes-de-valeurs-foncieres-geolocalisees/

La liste des codes insee des communes : 31032, 31056, 31069, 31088, 31149, 31150, 31157, 31291, 31351, 31417, 31424, 31526, 31555, 31557

Première approche

En première approche, on va se simplifier la vie : le service fournit des jeux de données par commune. Vous allez créer un script qui

- crée un dossier principal et dans ce dossier, un sous-dossier par commune
- télécharge les fichiers pour chaque commune, les stocke dans le sous-dossier correspondant et le décompresse
- Utilise ogr2ogr pour transformer le fichier CSV en fichier geopackage

Deuxième approche

Supposons que le service ne fournisse pas des extractions par commune et qu'on doive faire avec un plus gros fichier. Vous allez créer un script qui

- crée un dossier principal et dans ce dossier, un sous-dossier par commune
- télécharge le fichier CSV du département Haute Garonne (31) et de dézippe
- copie le template VRT fourni ici dans chacun des dossiers de commune
- remplace la chaine de caractère REMPLACER_PAR_CODE_INSEE_COMMUNE dans chacune des copies du VRT, afin de l'adapter à chaque commune
- Utilise ogr2ogr pour transformer la source VRT en fichier geopackage

Note 1 : VRT est un format inveté par GDAL/OGR, qui permet de définir une donnée virtuelle à partir d'une donnée source et d'un fichier .vrt qui définit une transformation à appliquer à cette donnée source. Vous

aurez un cours dédié à ça durant votre cursus. On ne va donc pas rentrer trop dans les détails. Mais pour les impatients : https://gdal.org/drivers/vector/vrt.html. *Note 2* : bien sûr, on pourrait faire plus simple. Mais on essaie d'appliquer ce qu'on a vu en cours.

Linux et les serveurs informatiques

Dans l'ensemble, on peut définir un serveur comme un ordinateur connecté au réseau, fournissant des services distants et auquel on accède sans interface graphique.

Attention, Linux peut fournir une interface graphique. Je dirais même qu'il est meilleur que Windows pour ça. en tous cas il offre plus de choix. Mais là où il se démarque vraiment, c'est quand on n'a *pas* d'interface graphique. Il nous reste alors la ligne de commande. Nous y sommes.

Connexion distante

Un outil omniprésent lorsqu'on parle de connexion distante : `ssh` (Secure Shell). Il permet de se connecter en ligne de commande à une machine distante, de façon totalement sécurisée. On y est alors comme chez soi... en ligne de commande

Tunnel SSH

On peut même faire un pas de plus, et jouer à saute-mouton : ssh permet beaucoup de choses, notamment d'ouvrir des *tunnels*. Un tunnel permet de passer par un serveur distant pour se connecter à une autre machine à laquelle seul ce serveur distant pouvait se connecter. Sauf qu'avec le tunnel, une fois établi, c'est comme si cette autre machine était en local !

Note : on peut aussi faire du ssh et même du tunneling depuis Windows, certains outils permettent ça, notamment le très connu `Putty`. Mais bon... comment faire mal, en 20 clics ce qu'une seule ligne de commande fait bien et proprement ? Ca s'eppelle en 5 lettres, ça commence par 'P'.

Sujet suivant : bases de données