Environnements virtuels conda :

intérêt – installation – utilisation - partage

Youssef MOURCHID

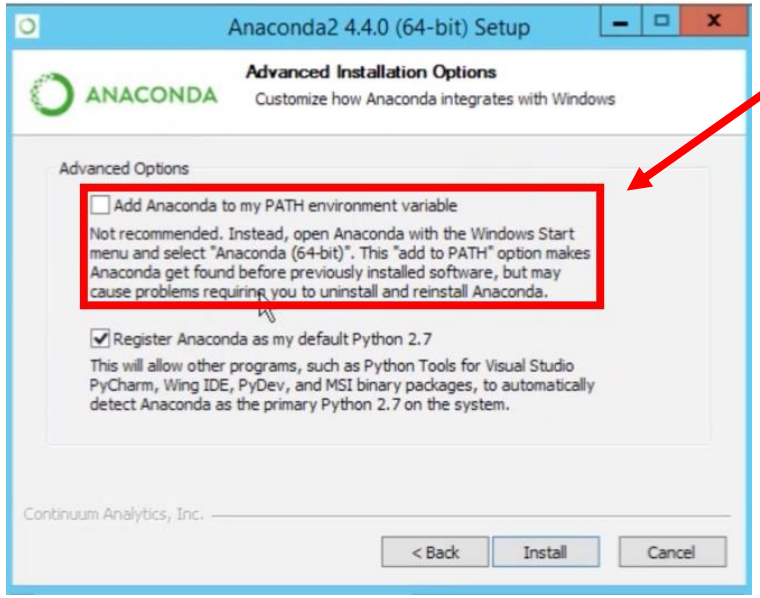
youssef.mourchid@isen-ouest.yncrea.fr

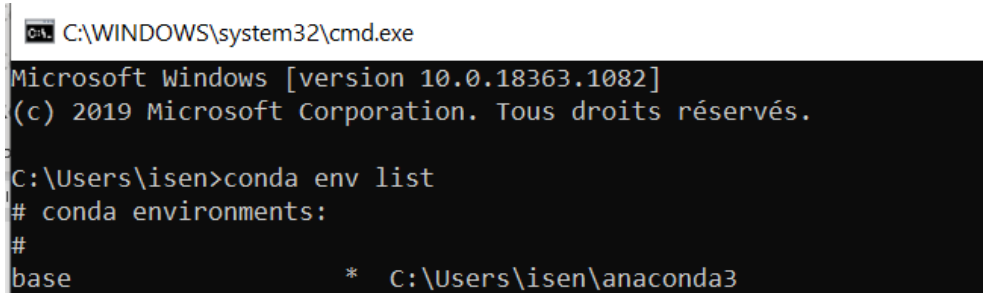
Prérequis

Voici quelques prérequis pour préparer au mieux la formation :

1. **Installer Anaconda :** https://www.anaconda.com/products/individual

Remarque : lors de l'installation et si demandée, activer l'option





2. **Protocole de test :**
   a) Ouvrir un terminal
   b) Lancer la commande conda env list qui doit retourner

Déroulement

1. Introduction à CONDA
   a) Contexte et avantages
   b) Démonstration

2. Les incontournables
   a) Création manuelle : création d'un environnement pour soi
   b) Création automatique : à partir d'un YAML (duplicable)

3. Diffusion : communiquer un environnement virtuel
   a) Créer son propre YAML
   b) Mettre à jour un YAML

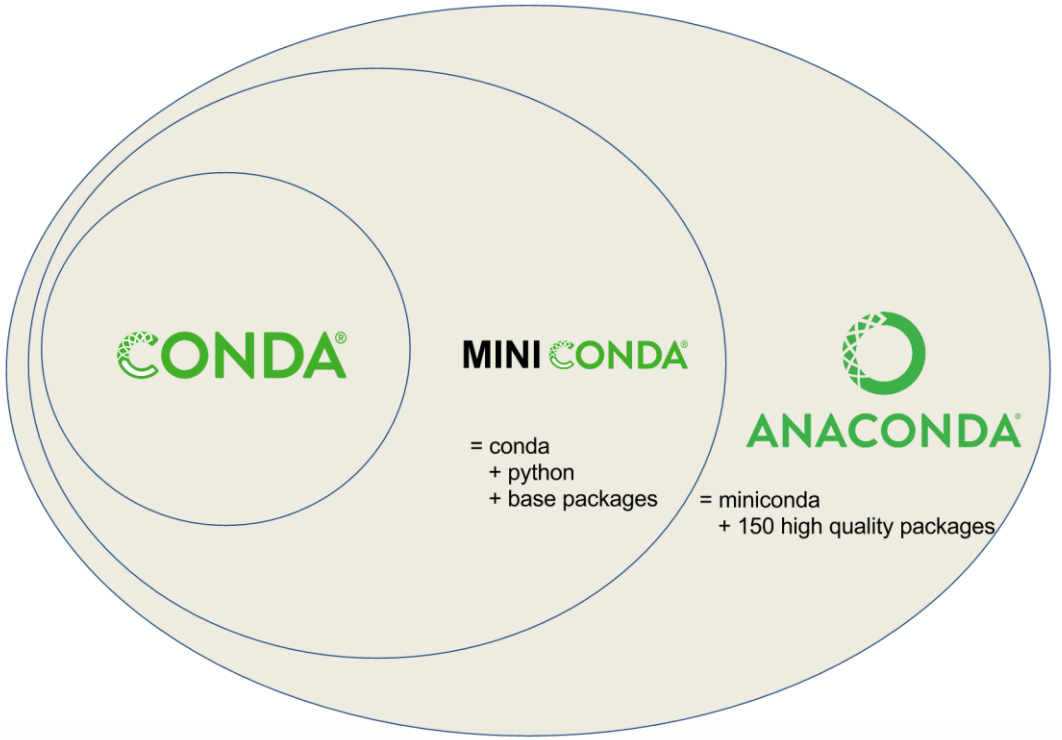
4. Pour aller plus loin
   a) Arborescence d'installation de CONDA
   b) CONDA révisions

Introduction à CONDA

Pourquoi CONDA ?

Environnements conda

CONDA	ENVIRONNEMENTS VIRTUELS
Installer Python	Créer des environnements de développements spécifiques à des langages (Python, R, Ruby, Julia...)
Installer les bibliothèques	Personnaliser les environnements en fonction du projet
Installer un environnement de développement (IDE-spyder, jupyter...)	Isoler un environnement de développement
Gérer les environnements virtuels	Pouvoir partager et reproduire un environnement à l'identique
	Gérer les dépendances entre librairies (selon machine, OS)



- Les environnements virtuels permettent :
  - stabilité dans le temps : assurent une reproductibilité des calculs
  - collaboratif: un environnement est exportable (simple fichier texte) à transmettre à un collaborateur,
  - autonomie par rapport aux administrateurs systèmes de la plateforme (pas besoin d'accès administrateur)
- Inconvénient : taille disque : une distribution de Python (ou R, Ruby) peut atteindre plusieurs Go

Démonstration

Grâce à la formation CONDA on va voir comment y arriver…

Lancement de VS code depuis l'environnement « **microsoft_env** » :

- Ouverture d'un terminal
- conda env list
- conda activate microsoft_env
- code

Les incontournables

Créer et utiliser un environnement virtuel

Création manuelle : création d'un environnement pour soi

On ouvre un terminal puis :

Théorie

1. conda env list permet de lister les environnements installés.
2. conda create permet de créer un environnement.
   i. L'option –n définit le nom de l'environnement (ex : -n microsoft_env).
   ii. Cet environnement se situe en général ici : C:\Users\isen\anaconda3\envs\. On peut le voir avec la commande conda env list citée plus haut.

Mise en pratique

Exécution de la commande conda create –n microsoft_env python=3.7 numpy=1.18.5 qui créé un environnement formation_env incluant python 3.7 et numpy 1.18.5.

Protocole de test

1. conda env list
2. Activer l'environnement avec conda activate microsoft_env
3. conda list pour lister les modules de l'environnement
4. python
5. import numpy => ok
6. import pandas => No module named « pandas » => normal pas installé

Mise à jour manuelle

On veut inclure matplotlib (e.g. 3.2.1) : conda install matplotlib=3.2.1

Création automatique : à partir d'un YAML (duplicable)

On ouvre un terminal puis :

Théorie

conda env create permet de créer un environnement.

L'option –f définit le nom du YAML (ex : -f microsoft _env_auto.yml).

Mise en pratique

Exécution de la commande conda env create –n microsoft _env_auto –f microsoft _env_auto.yml qui créé un environnement microsoft_env_auto à partir du YAML microsoft_env_auto.yml

Protocole de test

1. conda env list
2. Activer l'environnement avec conda activate microsoft_env_auto
3. conda list pour lister les modules de l'environnement
4. code
   a) conda env list et !where python (!which python sous linux) depuis le terminal VS Code pour vérifier l'environnement
   b) import streamlit
5. jupyter notebook
   a) conda env list et !where python (!which python sous linux) depuis la console pour vérifier l'environnement
   b) import streamlit

Diffusion

Communiquer l’environnement virtuel de son projet

Reconstruire l’environnement d’un projet téléchargé

Créer son propre YAML

On ouvre un terminal puis :

Théorie

1. Méthode 1 : Aujourd'hui on privilégie une création à la main à partir d'un fichier existant
2. Méthode 2 : Mais on peut utiliser `conda env export`
   i. L'option `-n` définit le nom de l'environnement dont on veut exporter le contenu (ex : `-n microsoft_env`).
   ii. `>` permet d'envoyer la sortie de la commande vers un fichier YAML (ex : `> microsoft_env.yml`)

Mise en pratique

- Exécution de la commande

```
conda env export -n microsoft_env_auto > microsoft_env_auto_export.yml
```

Mettre à jour son YAML puis son environnement

On ouvre un terminal puis :

Théorie

1. Méthode 1 : Si l'environnement contient des dépendances PIP
   i. On supprime l'environnement avec un `conda env remove`
      L'option `-n` définit le nom de l'environnement (ex : `-n microsoft_env_auto`)
   ii. On créé un nouvel environnement à partir du nouveau YAML avec un `conda env create`
2. Méthode 2 : Si l'environnement ne contient pas de dépendances PIP
   On met à jour l'environnement avec un `conda env update`
   i. L'option `-n` définit le nom de l'environnement (ex : `-n microsoft_env_auto`)
   ii. L'option `-f` définit le nom du YAML (ex : `-f microsoft_env_auto_squelette.yml`)
   iii. L'option `--prune` supprime les dépendances qui ne sont plus requises

Mise en pratique

1. Créer un environnement à partir du YAML **`microsoft_env_auto_squelette.yml`**
2. Est-ce qu'il contient des dépendances PIP ?
3. Ajouter matplotlib 3.2.1 au YAML **`microsoft_env_auto_squelette.yml`**
4. Exécution de la commande

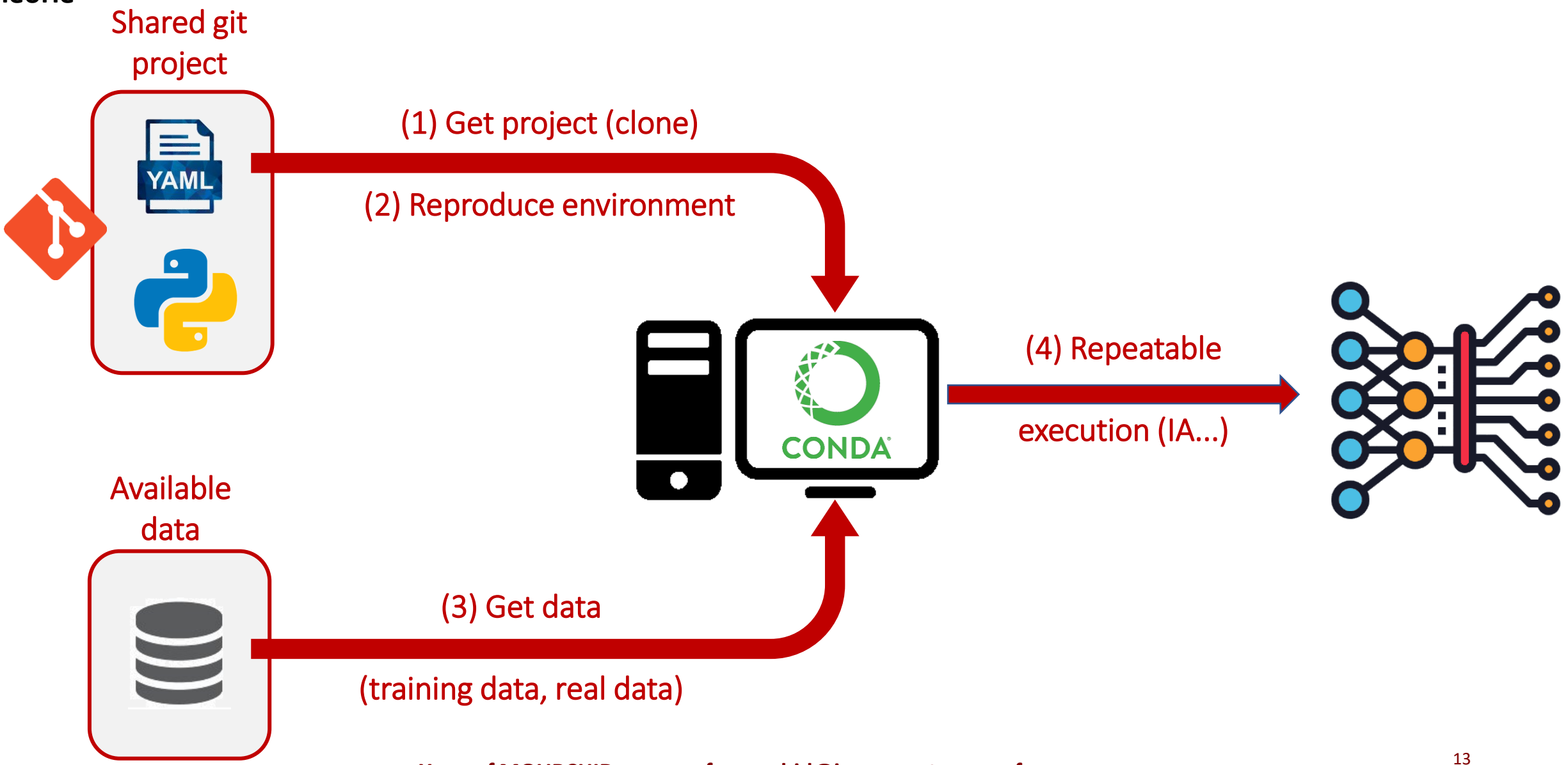
```
conda env update -n microsoft_env_auto_squelette -f microsoft_env_auto_squelette.yml --prune
```

Protocole de test

1. `conda env list`
2. Activer l'environnement avec `conda activate microsoft_env_auto_squelette`
3. `conda list matplotlib` pour lister les modules dont le nom contient matplotlib
4. Vérifier que matplotlib 3.2.1 est installé
5. Faire un plot

Reconstruire l'environnement d'un projet Python

Théorie



Cas d'application : un projet .ZIP

Mise en pratique

1. Créer un répertoire NuageMots
2. Récupérer le projet project_wordcloud.zip, puis décompresser l'archive dans le répertoire NuageMots
3. Repérer le fichier YAML, et créer un environnement **`env_WORD`** associé
4. Activer l'environnement **`env_WORD`**

Protocole de test

a. `conda env list`

b. `conda list word` pour lister les modules dont le nom contient word

c. Vérifier que wordcloud est installé, ainsi que matplotlib : la version correspond-elle à celle du YAML ?

5. Lancer VS code:

   `code`

6. Se positionner dans le répertoire de travail ***`<chemin vers NuageMots>/project_wordcloud/code`***

1. Lancer le programme python **`tracer_nuage.py`**

Protocole de test

a. `Un graphe des 10 mots clés les plus représentés dans le fichier` ***`<chemin vers NuageMots>`***`/project_wordcloud/data/projectLSL.csv`

Quelques conseils

- Fichier texte contenant les **commandes conda usuelles**

```
...

> /ENV_NAME/conda-meta/history
    historique des commandes conda
> conda clean --all
    remove temporary files
> conda search -f numpy
    recherche des package conda numpy dispo.
> conda env export -n ENV_NAME -f ENV_description.yml
    automatic creation of YAML file
> conda env remove -n ENV_NAME
    remove an environment

...
```