DÉCOUVERTE DE PIPENV

Introduction

Présentation

Bonjour je m'appelle Amer et actuellement, je suis analyste de production. Je travaille sur la partie data "collection et traitement des données". Dans notre équipe nous utilisons python car c'est un langage simple à l'utilisation et possède un très bon catalogue de librairies puissantes comme pandas, boulot. Nous utilisons notamment pipenv, et je souhaite à travers mon tout premier article dans ce blog de vous faire **découvrir l'utilité et la manipulation pipenv**.

C'est quoi la BI ?

Tout d'abord je souhaiterais vous décrire mon domaine d'activité qui est la BI "Business Intelligent" ou "informatique décisionnelle" en fr, mais en quoi ça consiste BI ? La BI présente un ensemble de méthodes, d'outils informatique, qui nous permet de piloter les stratégies des entreprise au travers de Dashboard (tableaux de bord) et de rapports de suivi transmis aux différents managers de l'entreprise. D'un point de vue méthodologique, la BI repose sur la collecte, la mise en forme et la restitution des données pour obtenir des informations utiles aux prises de décisions dans l'entreprise. Si j'en ai l'occasion, dans de futurs articles nous nous intéresserons particulièrement à la découverte des différentes librairies existantes du langage python qui nous aiderons par la suite à collecter et traiter des données.

Python

Dans ce chapitre, nous découvrirons **comment installer et configurer notre environnement Python**.

Nous présenterons aussi comment installer et gérer nos paquets en utilisant pipenv. >Pour ce faire il nous faut principalement une machine avec l'un de ces OS (Debian/Ubuntu, CENTOS ou Windows 10.

Installation de python sur CENTOS

D'abord on va se mettre en root :

```
sudo -i
```

Installe les paquets suivants :

```
yum groupinstall -y 'Development Tools'
yum install -y zlib-devel
```

Ensuite nous allons dans `/usr/src` :

```
cd /usr/src
```

On télécharge les repositories, ici j'ai choisi la version 3.8.2 mais vous pouvez choisir une autre version :

```
wget https://www.python.org/ftp/python/3.8.2/Python-3.8.2.tar.xz
```

Ensuite, on décompresse le fichier tar et on va dans le dossier Python-3.8.2 comme suit :

```
tar xf Python-3.8.2.tar.xz
cd Python-3.8.2
```

La prochaine étape consiste à **compiler nos sources python** avec les commandes suivantes :

```
./configure --enable-optimizations --with-ensurepip=install
make altinstall
```

Voici, ci-dessous une explication des différentes options utilisées :

- `--enable-optimizations` : optimise le temps de compilation.
- `make altinstall` : empêche le remplacement de l'exécutable python par default situé dans **/usr/bin/python** et donc par la même occasion cela permet de tester la nouvelle version sans désinstaller la version de python disponible par défaut sur votre OS.
- `--with-ensurepip=install` : s'assure que 'pip' est bien installé.

> **Information**
>
> `pip` est le gestionnaire de paquets utilisé pour installer et gérer des paquets programmées en Python.

Maintenant, il nous reste upgrader "pip" :

```
sudo python3.8 -m pip install --upgrade pip
```

Installation de python sur Debian

Comme pour CENTOS, on va d'abord on va se connecter en tant que root

```
sudo -i
```

On procède ensuite à l'installation des paquets suivantes :

```
apt update -y
apt install -y wget build-essential libffi-dev libgdbm-dev libc6-dev libssl-dev zlib
```

Ensuite, on se déplace sur le dossier **/usr/src/** pour déposer nos sources de python :

```
cd /usr/src
```

Puis, on télécharge les sources python. De la même façon que pour CENTOS, j'ai choisi la version 3.8.2 mais vous pouvez bien sûr choisir une autre version si vous le souhaitez, concernant cette version voici les sources à télécharger :

```
wget https://www.python.org/ftp/python/3.8.2/Python-3.8.2.tar.xz
```

On décompresse ensuite le fichier tar et on va dans le dossier Python-3.8.2 :

```
tar xf Python-3.8.2.tar.xz
cd Python-3.8.2
```

La commande de compilation ne change pas par rapport à CENTOS, vous lancerez donc la même de compilation (l'explication des différentes options et fournit dans la section d'installation sur CENTOS) :

```
./configure --enable-optimizations --with-ensurepip=install
make altinstall
```

Enfin vous aurez besoin d'upgrader votre gestionnaire de paquet python pip :

```
sudo python3.8 -m pip install --upgrade pip
```

Installation de python sur Windows

Pour Windows c'est encore plus simple ! Il faut télécharger et installer le paquet MSI. Il faut que vous ayez les droits d'administrateurs. **N'oubliez pas de cocher la case 'Add PATH'** car celà permet d'ajouter le chemin du binaire python dans votre variables d'environnement "PATH" :

PIPENV

Dans cette partie, je vous présenterai `pipenv` qui est une alternative de pip. Pipenv, ou le 'Python Development Workflow for Humans' a été créé par Kenneth Reitz, il y a un peu plus de 3 ans. Actuellement, il est devenu la ressource officielle recommandée par Python pour la gestion des dépendances des paquets. Mais il y a encore de la confusion sur les problèmes qu'il résout et sur la façon dont il est plus utile que le pip standard.

Pourquoi utiliser pipenv ?

La plupart des utilisateurs de Python connaissent pip qui nous permet d'installer et de désinstaller des packages. Cependant pip n'inclus pas un moyen d'isoler les packages les uns des autres. Nous pourrions avoir besoin de travailler sur des projets python qui utilisent différentes versions des mêmes bibliothèques. C'est là tout l'intérêt de la création des environnements virtuels, qui nous ont permis de créer de petits environnements isolés pour chaque projet python sur lequelle nous souhaitons travaillé.

L'apparition de l'outil Pipenv permet de faciliter la gestion des packages et la prise en charge des environnements virtuels, vous pouvez donc utiliser un seul outil pour installer, désinstaller, suivre et documenter vos dépendances et créer , utiliser et organiser vos environnements virtuels.

Comment utiliser pipenv ?

Tout d'abord il faut **installer pipenv** avec la commande suivante :

```
pip install --user pipenv
```

Voici la commande pour **mettre à jour pipenv** :

```
pip install --user --upgrade pipenv
```

Ensuite, il faut se déplacer dans le dossier de votre projet python, et vous spécifierez la version python, ça sera la version qui sera utilisée pour exécuter votre projet dans votre environnement virtuelle :

```
cd myproject
pipenv --python 3.6
pipenv install
```

La commande `pipenv install` permet de créer deux fichiers **Pipfile** et **Piplock** dans le dossier de votre projet. Ces deux fichiers sont nécessaires pour ainsi créer votre virtuel environnement. Voici un exemple de **Pipfile** :

```
[[source]]
url = 'https://pypi.python.org/simple'
verify_ssl = true
name = 'pypi'

[packages]
requests = '*'

[dev-packages]
pandas = '*'
```

Dans ce fichier vous remarquerez qu'il existe deux types de paquets. Le premier `[dev-packages]` pour l'environnement de dev et le second `[packages]` pour la prod.

Pour installer des paquets en prod, vous utiliserez la commande suivante :

```
pipenv install mon-paquet
```

Pour installer des paquets en dev, vous utiliserez cette fois-ci la commande :

```
pipenv install mon-paquet --dev
```

Enfin voici la commande pour **lancer un script dans votre environnement virtuel** :

```
pipenv run python main.py
```

Si vous souhaitez entrer dans votre environnement virtuel, sans nécessité de taper la commande `pipenv run` à chaque fois. Vous-pouvez alors lancer la commande suivante :

```
pipenv shell python -m main
```

Si vous avez un fichier **requirements.txt**, vous pouvez alors demander à pipenv d'importer automatiquement les packages de ce fichier et de créer le fichier **Pipfile** pour vous. Voici comment faire :

```
pipenv install -r path/to/requirements.txt
```

Il est possible aussi de spécifier la version de vos packages, soit en modifiant le Pipfile ou en utilisant la commande suivante :

```
pipenv install 'pandas>=1.4'   # spécifier une version égale ou supérieure à la versi
pipenv install 'pandas<=2.13'  # spécifier une version égale ou plus petite que 2.13
pipenv install 'pandas>2.19'   # spécifier une version plus grande que 2.19.1 mais pa
```

Voici la commande pour **désinstaller des packages à l'aide de pipenv** :

```
pipenv uninstall mon-paquet
```

Conclusion

Dans cet article, nous avons vu comment installer python sur les différents OS et à étudier et utiliser l'outil pipenv.