Dev environments: use the nix, Luke!

Speakers : Clément Delafargue (FretLink), Hussein Ait-Lahcen (FretLink)
Format : Tools in action
Date : 18 avril 2019

L'objectif de ce tools in action consiste à présenter comment mettre à disposition des environnements de développement.
Clément est un gros hispter. Comme tout le monde, il s'est mis à apprendre le Deep Learning. Un pré-requis était d'installer Python et Anaconda. Une alternative était d'utiliser Miniconda. Il y'a une image Docker. Mais le DL demande à utiliser des GPU, ce que ne propose pas (encore) Docker.
C'est facile de crier sur Python. Dans l'univers Python, virtualenv permet de créer des environnements virtuels Python isolés.

Échaudé, Clément a testé **Nix**.
Chaque langage dispose de son package manager (ex : Maven sur Java).
On peut demander à Nix de basculer dans un contexte avec Keras et tensorFLow pour lancer un script python. Une ligne de commande est suffisante.

Sous le capot, Nix s'appuie sur le Nix store. Nix utilise des hash pour indexer les contenus (comme Git). Les builds sont reproductibles et on ne peut pas casser son environnement.

Pour créer un paquet, Nix demande à ce que toutes les dépendances soient déclarées. Tout est recherché dans le $PATH. Nix patche les binaires pour aller chercher ses librairies dans le store.

Cas de Node qui évolue plus rapidement que Python. NVM permet d'avoir plusieurs versions de NPM en parallèle. Fonctionne bien jusqu'à ce qu'on pointe sur des dépendances systèmes.
Nix permet de déclarer les dépendances systèmes comme toute dépendance.
Dans le fichier shell.nix, on voit que nodejs-11_x est déclaré.
Exemple d'un script go.js qui fait appel à Git (appel système).

La commande « Nix-shell --pure » permet de s'assurer qu'on a bien toutes les dépendances.

Comment packager des paquets NPM sous Nix ?
Il existe plusieurs outils (comme node2mix) qui permet de créer un package Nix.

Peut-on tout packager en Nix ?
Non car il y'a beaucoup de chose à faire. Problématique non résolue : lorsque NPM télécharge des binaires.

D'après le dernier sondage 2019 StackOverflow, Rust est pour la quatrième année de suite, le langage préféré des développeurs.
Rust est très proche du C et s'appuie beaucoup sur des librairies systèmes.

Exemple d'utilisation de rust, openssl et pkgconfig.
Mozilla fournit un overlay permettant de récupérer Rust via Nix.

Pourquoi ne pas utiliser Docker à la place de nix ?
Le cache de Docker prend plein de place. Qui plus est, cela arrive à Docker de retélécharger tous les layers.
La reproductibilité d'un build Dockerfile n'est pas garantie.

Sur de grosses applications Angular, il y'a des dévs qui ont écrit un docker-compose. Mais c'est lent à démarrer.

Nix permet de ne pas pourrir sa machine, sans avoir à utiliser de la contenerisation.

Installation de PosgreSQL avec Postgis.
Le fichier de conf est bien plus verbeux.

Nix peut faire bien plus que setter son environnement de dév.
En effet, Nix permet de générer des images Docker. Nix se charge alors d'agencer les layers.
La création de l'image Docker avec Nix est reproductible.

nixos permet de revenir à la version précédente du système.

Mise en garde : l'utilisation de Nix demande un peu de bricolage et la doc n'est pas forcément très claire