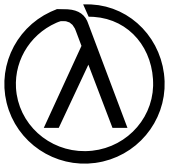

Présentation de Nix (et de son univers)

Nix est un gestionnaire de paquets logiciels. Encore un.

Qu'ils soient liés à un système d'exploitation (APT pour Debian, RPM pour Red Hat, Pacman pour Archlinux…) ou à une technologie de développement logiciel (NPM pour Node.js, repris ensuite par d'autres), ces gestionnaires de paquets sont légion. Ils permettent d'installer des logiciels désirés **et leurs dépendances** à partir de dépôts en ligne (idéalement) de confiance.

Chacun y va de sa spécificité technique… ou pas, quand la seule distinction d'un gestionnaire de paquet est d'être celui choisi officiellement par telle ou telle organisation (typiquement, une distribution Linux).

Alors pourquoi Nix ? A t'il quelque chose de concret à apporter ?

Oui !

La spécificité de Nix est son approche fonctionnelle, c'est à dire inspirée par le modèle de la programmation fonctionnelle.

1. Un Gestionnaire de paquet fonctionnel

Nix s'inspire de la programmation fonctionnelle et des types de données fonctionnels. Cela induit plusieurs caractéristiques :

- Tout comme dans la programmation fonctionnelle, l'installation de paquets par Nix prend une forme **déclarative**.

  Des gestionnaires de paquets plus conventionnels, comme APT ou RPM, peuvent être considérés comme des gestionnaires de paquets impératifs. Avec ces gestionnaires, on lance, les unes après les autres, des **commandes** d'installation, qui sont autant d'instructions données au gestionnaire, lui ordonnant de récupérer et d'installer séquentiellement des ensembles de paquets (pourvu qu'ils n'entrent pas en conflit les uns avec les autres).

  Avec Nix, la définition des paquets à installer prend la forme d'un fichier de configuration qui exprime ce qui doit constituer le système. Tout le contenu de ce fichier est une **expression** décrivant le système, composée de sous expressions décrivant ses différentes caractéristiques, parmi lesquelles les paquets à installer. Le gestionnaire se charge ensuite de faire en sorte que le système ainsi décrit soit mis en place.

  Cet aspect déclaratif fait qu'il est facile de stocker la configuration dans un fichier et de la récupérer pour l'appliquer sur une autre machine, telle quelle ou après quelques modifications.

- Dans le soucis de limiter les effets de bords, au premier rang desquels on trouve dans le domaine de la gestion de paquets les conflits de versions et de dépendances, Nix installe **chaque paquet dans un répertoire qui lui est propre**.

  Il est ainsi possible d'avoir sur un même système des versions différentes d'un même logiciel et d'activer l'une ou l'autre dans un environnement configuré. Ceci est particulièrement utile pour les programmeurs, ce dont nous traiterons dans une section ultérieure du présent article.

- Cette approche permet également une chose qui est resté longtemps un rêve pour de nombreux administrateurs système (et l'est encore pour ceux qui ignore l'existence de Nix ou ne sont pas autorisés par leurs employeurs à l'utiliser) : **la possibilité de revenir en arrière** (notion de « rollback ») après une installation que l'on regrette.

  La mise à jour d'un système peut en effet parfois mal se passer. Et parfois, la situation qui en résulte n'est pas récupérable, du moins pas facilement.

  La possibilité de revenir à la situation d'avant l'installation calamiteuse est donc un luxe dont on n'a pas envie de se passer quand il est disponible.

2. L'univers autour de Nix

Dans le monde Linux, quand on a un gestionnaire de paquets, il sert inévitablement de fondation à tout un univers bâti autour : distributions logiciels (c'est à dire systèmes d'exploitations), dépôts logiciels divers et variés...

2.1. Le système d'exploitation : NixOS

Dans le monde des distributions Linux, il est courant qu'un gestionnaire de paquet soit fait uniquement pour constituer l'élément central d'une distribution donnée (même s'il peut ensuite être récupéré par d'autres distributions qui seront alors dites « dérivées »).

Nix, quant à lui, est un gestionnaire de paquets disponible sur de multiples distributions. Sur ces distributions, il fait donc figure de gestionnaire de paquets secondaire par rapport au gestionnaire officiel de la distribution.

NixOS est une distribution Linux faite autour de Nix, comme la distribution Debian est faite autour d'APT ou la distribution Red Hat autour de RPM.

L'intérêt d'utiliser NixOS, plutôt que simplement Nix sur une autre distribution, est de pouvoir bénéficier de la puissance de Nix sur l'intégralité du système installé (jusqu'au noyau) plutôt que seulement pour les applications installées par dessus le système.

2.2. Dépôts logiciels

Nixpkgs, la collection standard de paquets Nix, contient plus de 80 000 paquets, ce qui est considérable. Cela fait de NixOS la distribution la plus à jour en logiciels et la fait systématiquement figurer dans le top 3 des distributions les plus fournies en paquets.

Ces paquets sont organisés en canaux, qui peuvent correspondre à des versions successives de NixOS ou à des dépôts thématiques.

2.3. Nix pour le développement logiciel

Nix n'est pas seulement un gestionnaire de paquets destinés à être installés sur un système d'exploitation, comme APT ou RPM, mais également un gestionnaire de paquets de développement, comme Maven dans le monde Java ou NPM dans le monde Javascript.

Mais à la différence de ces derniers, Nix n'est pas dédié à un langage de programmation particulier.

C'est peut être dans la communauté des développeurs Haskell, qui (logiquement) se sentent des affinités avec son approche fonctionnelle, qu'il est le plus utilisé. Mais je l'ai également vu utilisé par des développeurs C++ et Python.

Là aussi, l'approche fonctionnelle fait des merveilles. Il est ainsi possible d'installer des dépendances de développement correspondant à différents projets dans des environnements parfaitement isolés les uns des autres. On a donc la possibilité de travailler, sur un même poste de développement, sur des projets qui nécessitent des dépendances antinomiques (par exemple des versions différentes d'une même dépendance).