Keskesé

Il y a pas mal de choses dans l'écosystème Nix et on va découvrir les détails des composants plus en détail plus tard dans ce guide.

Le nom **Nix** fait référence à deux choses : language et un gestionnaire de paquets. **NixOS** en revanche fait référence au système d'exploitation utilisant Nix (le langage et le gestinonaire de paquets)

Fun facts sur Nix

- Nix est le plus gros gestionnaire de paquet en terme de nombre de paquets à jour ET de nombre de paquet tout cours avec plus de **80 000** paquets et plus de **55 000** paquets en dernière version (en comparaison Ubuntu a ~35000 paquets dont ~18000 en dernière version)
- Le cache de Nix (qui contient tous les builds des différents paquets) fait plus de **400TiB** (presque un demi petabyte) et le maintient de toute l'infrastructure du serveur de cache coûte **9000€/mois**
- Nix existe depuis **2003** (en comparaison *pacman* existe depuis 2002, *apt* depuis 1998 et *apk* (de alpine) depuis 2005)

Nix (le gestionnaire de paquets)

Le gestionnaire de paquet Nix est très différents des autres gestionnaires de paquets :

- Il est **multi-version** et très **fiable**, ce qui signifie que l'on peut avoir simultanément plusieurs version d'un même logiciel. Cela empèche donc tout problème d'*enfer des dépendences*.
- Tous les paquets ont des **dépendences complètes**, tous les paquets étant installés dans un dossier et étant isolé du reste lors du build, on a l'assurance que chaque paquet a la liste complète de ses dépendences spécifiée)
- Nix supporte le **multi-utilisateur** ce qui signifie qu'il n'y a pas besoin d'un accès admin pour y installer des choses, également le cache reste commun, ce qui signifie que si utilisateur A demande le paquet X puis que utilisatrice B demande le même paquet alors celui-ci ne sera pas de nouveau téléchargé/build.
- Les changements sont **atomiques**, ce qui signifie que l'on peut tout à fait mettre à jour uniquement un logiciel mais pas les autres sans créer de conflit.
- Un **historique** est gardé de tous les changements, ainsi il est très facile de retourner à état précédent en cas de problème

- Nix possède aussi un **garbage collector**, ce qui signifie que lorsque l'on supprime un paquet, celui-ci n'est que déréférencé mais le fichier est toujours présent sur le système ce qui permet de développer très facilement. Cependant quand il faut faire de la place le garbage collector va supprimer la totalité des logiciels qui ne sont plus référencé nulle part.
- Nix est **déterministe** et **reproductible**, une certaine configuration et une certaine dérivation doit toujours arriver à l'exact même résultat, ce qui assure donc que si quelquechose marche sur une machine, elle fonctionnera sur toutes les machines de la même architecture.
- Nix supporte à la fois l'installation **précompilée** et l'installation **depuis la source**. Un cache est disponible pour télécharger tous les logiciels (c'est la méthode par défault), cependant il est également possible d'installer tout depuis la source (comme dans Gentoo).
- Nix permet aussi **l'installation temporaire** de paquets, c'est à dire le développement d'environment dans lequel on peut spécifier la liste des logiciels à utiliser sans pour autant affecter le reste du système.
- Nix est **multi-distribution et multi-plateforme**, il fonctionne sur toutes les distributions Linux ainsi que sur macOS. Il a également des ports pour BSD et Windows (avec WSL ou Cygwin)

Nix (le langage)

Nix est le langage utilisé pour créer les paquets (appelés *dérivations* mais on verra ça plus tard) mais également pour gérer les configruations ou même créer des outils permettant de créer des dérivations plus facilement.

- Nix est **spécifique à un domaine d'utilisation**, c'est un language pour le gestionnaire de paquet et rien d'autre, ce n'est pas un langage multi-usage.
- Nix est un langage **déclaratif**, Il n'y a pas de notion d'exécution d'étapes séquentielles. Les dépendances entre les opérations ne sont établies qu'à partir des données.
- Nix est **pure**, toutes les valeurs sont immuables
- Nix est un langage **fonctionnel**, les fonctions sont des valeurs comme les autres et peuvent elle même être retournées par d'autres fonctions
- Nix est **paresseux**, les expressions sont seulement calculées quand elles sont demandées (ainsi il on a des milliers d'expressions et que l'on en demande qu'une, cela ne ralentis rien du tout car seule celle que l'on va demander sera calculée)
- Nix fonctionne avec des **types dynamiques**, il n'y a donc pas besoin de préciser les types dans le code

NixOS

- NixOS utilise **Nix** comme gestionnaire de paquet mais aussi comme langage de configuration de l'entièreté du système

- Il est possible de choisir différentes configurations et de passer d'une à l'autre très facilement dès le début du démarrage
- Il possède tous les autres avantages de Nix.

Revision #3
Created 7 July 2023 18:33:28 by SnowCode
Updated 7 July 2023 20:21:53 by SnowCode