Nix: une approche déclarative et reproductible des configurations

Rudy Perraudin, Satya Tamby

10 novembre 2026

1. Introduction : un gestionnaire de paquets reproductible
2. Nixos : *System as code*
3. Extension : *Infrastructure as code*

Nix : gestionnaire de paquets

Dependency hell :

- Problèmes liés au *packaging* : plusieurs paquets dépendent de différentes versions de la même dépendance (par ex : glibc)

Nix : gestionnaire de paquets

Dependency hell :

- Problèmes liés au *packaging* : plusieurs paquets dépendent de différentes versions de la même dépendance (par ex : glibc)
- Chaque paquet réfère à glibc via /usr/lib/glibc.so

Nix : gestionnaire de paquets

Dependency hell :

- Problèmes liés au *packaging* : plusieurs paquets dépendent de différentes versions de la même dépendance (par ex : glibc)
- Chaque paquet réfère à glibc via /usr/lib/glibc.so
- Deux versions ne peuvent pas co-exister

Nix : gestionnaire de paquets

Dependency hell :

- Problèmes liés au *packaging* : plusieurs paquets dépendent de différentes versions de la même dépendance (par ex : glibc)
- Chaque paquet réfère à glibc via /usr/lib/glibc.so
- Deux versions ne peuvent pas co-exister
- Problème standard en developpement. Ex en python : virtualenv, conda, uv, ...

Nix : Résolution du dependency hell

- Un paquet est défini par sa configuration

Nix : Résolution du dependency hell

- Un paquet est défini par sa configuration
- Il est installé dans /nix/store/<hash de la configuration>

Nix : Résolution du dependency hell

- Un paquet est défini par sa configuration
- Il est installé dans /nix/store/<hash de la configuration>
- Deux versions du même paquet sont donc installées dans deux endroits différents

Nix : Résolution du dependency hell

- Un paquet est défini par sa configuration
- Il est installé dans /nix/store/<hash de la configuration>
- Deux versions du même paquet sont donc installées dans deux endroits différents
- Plusieurs versions du même paquet peuvent donc coéxiser

Nix : résolution des dépendances

Lors de l'installation d'un paquet, Nix remplace les dépendances par leur chemin dans /nix/store

- Ces résolutions sont *reproductible*

Nix : résolution des dépendances

Lors de l'installation d'un paquet, Nix remplace les dépendances par leur chemin dans /nix/store

- Ces résolutions sont *reproductible*
- Seul problème : les paquets exposés à l'utilisateur

Nix : résolution des dépendances

Lors de l'installation d'un paquet, Nix remplace les dépendances par leur chemin dans /nix/store

- Ces résolutions sont *reproductible*
- Seul problème : les paquets exposés à l'utilisateur
- Solution : chaque utilisateur a un profile, exposant uniquement les paquets qu'il a installé

Nix : résolution des dépendances

Lors de l'installation d'un paquet, Nix remplace les dépendances par leur chemin dans /nix/store

- Ces résolutions sont *reproductible*
- Seul problème : les paquets exposés à l'utilisateur
- Solution : chaque utilisateur a un profile, exposant uniquement les paquets qu'il a installé
- Les paquets exposés n'ont qu'une seule version

Nixos : *System as a code*

- Une configuration système peut être vue comme un paquet

Nixos : *System as a code*

- Une configuration système peut être vue comme un paquet

$\Longrightarrow$

- Plusieurs configurations peuvent coéxister simultanément

Nixos : *System as a code*

- Une configuration système peut être vue comme un paquet

$\Longrightarrow$

- Plusieurs configurations peuvent coéxister simultanément
- La configuration active peut être remplacée par une configuration anterieure

De plus : Nix est déclaratif

- Au lieu d'écrire les étapes de configurations, il suffit d'écrire l'état désiré

De plus : Nix est déclaratif

- Au lieu d'écrire les étapes de configurations, il suffit d'écrire l'état désiré
- Nix se charge de générer les étapes de configurations

Exemple : ajout de openssh

Sur Debian :

- Installer le paquet : *apt install openssh*

Exemple : ajout de openssh

Sur Debian :

- Installer le paquet : *apt install openssh*
- Ajuster le fichier de configuration par défaut (port non-standard : 5022)

Exemple : ajout de openssh

Sur Debian :

- Installer le paquet : *apt install openssh*
- Ajuster le fichier de configuration par défaut (port non-standard : 5022)
- Activer le service

Exemple : ajout de openssh

Sur Debian :

- Installer le paquet : *apt install openssh*
- Ajuster le fichier de configuration par défaut (port non-standard : 5022)
- Activer le service
- Configurer le firewall

Exemple : ajout de openssh

Sur Debian :

- Installer le paquet : *apt install openssh*
- Ajuster le fichier de configuration par défaut (port non-standard : 5022)
- Activer le service
- Configurer le firewall

Risques : la configuration doit être cohérente.

Exemple : ajout de openssh

Inconvénients :

- Ces tâches sont techniques et répétitives

Exemple : ajout de openssh

Inconvénients :

- Ces tâches sont techniques et répétitives
- Une erreur/Un oubli peut avoir des conséquences pénibles

Exemple : ajout de openssh

Inconvénients :

- Ces tâches sont techniques et répétitives
- Une erreur/Un oubli peut avoir des conséquences pénibles
- Faire une configuration "minimale" demande une bonne compréhension du service

Exemple : ajout de openssh

Inconvénients :

- Ces tâches sont techniques et répétitives
- Une erreur/Un oubli peut avoir des conséquences pénibles
- Faire une configuration "minimale" demande une bonne compréhension du service
- Est-il possible de les automatiser en fournissant le minimum d'informations ?

Exemple : ajout de openssh

Sur NixOS :

```
# Enable the OpenSSH daemon.
services.openssh = {
  enable = true;
  ports = [5022];
  openFirewall = true;
};
```

Exemple : ajout de openssh

Sur NixOS :

```
# Enable the OpenSSH daemon.
services.openssh = {
  enable = true;
  ports = [5022];
  openFirewall = true;
};
```

Nix sait que le port est non-standard et peut configurer le firewall en fonction.

Configuration de NixOS

- Un formalisme unifié pour la configuration de tout le système (services, les modules noyaux, KVM...)

Configuration de NixOS

- Un formalisme unifié pour la configuration de tout le système (services, les modules noyaux, KVM...)
- La configuration est centralisée (/etc/nixos/configuration.nix)

Avantages

- Moins de tâches techniques

Avantages

- Moins de tâches techniques
- Limitation de la surface d'erreur

Avantages

- Moins de tâches techniques
- Limitation de la surface d'erreur
- Versionable

Avantages

- Moins de tâches techniques
- Limitation de la surface d'erreur
- Versionable
- C'est un paquet Nix donc reproductible (indépendant de la machine et possibilité de rollbacks).

Nix est programmable

On peut étendre le formalisme pour résoudre des tâches plus complexes.

Nix est programmable

On peut étendre le formalisme pour résoudre des tâches plus complexes.
Ex : configurer plusieurs machines simultanément.

Extension : Infrastructure as code

Exemple : Configuration de Nextcloud avec une BDD Postgres centralisée

Extension : Infrastructure as code

Exemple : Configuration de Nextcloud avec une BDD Postgres centralisée
Méthode impérative :

- Créer la base de données sur le serveur Postgres

Extension : Infrastructure as code

Exemple : Configuration de Nextcloud avec une BDD Postgres centralisée
Méthode impérative :

- Créer la base de données sur le serveur Postgres
- Générer les accès

Extension : Infrastructure as code

Exemple : Configuration de Nextcloud avec une BDD Postgres centralisée
Méthode impérative :

- Créer la base de données sur le serveur Postgres
- Générer les accès
- Envoyer les accès sur les machines Postgres et Nextcloud

Extension : Infrastructure as code

Exemple : Configuration de Nextcloud avec une BDD Postgres centralisée
Méthode impérative :

- Créer la base de données sur le serveur Postgres
- Générer les accès
- Envoyer les accès sur les machines Postgres et Nextcloud
- Configurer Nextcloud pourqu'il utilise la base de données

Extension : Infrastructure as code

Exemple : Configuration de Nextcloud avec une BDD Postgres centralisée
Méthode impérative :

- Créer la base de données sur le serveur Postgres
- Générer les accès
- Envoyer les accès sur les machines Postgres et Nextcloud
- Configurer Nextcloud pourqu'il utilise la base de données
- Configurer les firewalls, générer les certificats TLS, ...

Extension

Configuration minimale pour déduire toutes ces étapes :

```
vm-postgres= {
  host = "cpuhost1";
  vcpu = 4;
  memory = 8000;
  ip = "10.0.0.2";
  services = ["postgres"];
};
vm-nextcloud = {
  host = "cpuhost1";
  vcpu = 8;
  memory = 16000;
  ip = "10.0.0.3";
  containers = ["nextcloud"];
};
```

On pourrait faire encore plus :

- Créer les VM avec terraform
- Génèrer les secrets
- Déployer les configurations (Containers, routes, ...)

Limite la surface de panne et d'attaque car :

- Configuration statique minimale déduite de la déclaration (ex : pas besoin de DNS...)
- Vérification de la compatibilité AVANT le déploiement

NAPS : Infrastructure as code

Projet actuel développé par le LPHI :
NAPS : Nix-based Automated Plumbing System

NAPS : Infrastructure as code

Projet actuel développé par le LPHI :
NAPS : Nix-based Automated Plumbing System

