Les dérivations

Les dérivations dans Nix sont la base des builds des applications. Elle se basent sur la fonction `derivation`, cependant vous n'utiliserez probablement jamais cette fonction directement car elle est bien trop basique. C'est pour cela que des fonctions tel que `mkDerivation` du package `stdenv` sont là, ce sont des fonctions basée sur `derivation` mais qui rendent les choses plus simple et plus sûres pour les mainteneurs de paquets.

Nous allons voir ici comment la fonction de base `derivation` fonctionne, ce qu'elle fait et comment les builds fonctionnent dans Nix.

La syntaxe de `derivation`

Cette fonction prends comme argument un set d'attribut avec les attributs suivant :

- `system`, qui précise l'architecture du paquet
- `name`, qui précise le nom du paquet
- `builder` qui est le programme utilisé pour construire le paquet (cela peut être une autre dérivation, ou un chemin de fichier)
- (optionnel) `args` qui est une liste des arguments à passer au *builder*
- (optionnel) `outputs` qui spécifie la liste symbolique des outputs (par exemple `outputs = [ "lib" "doc" ]`. Par défault, Nix crée une variable d'environnement `$out` qui spécifie la localisation du dossier dans le Nix store, mais on peut très bien en ajouter d'autres tel que `$lib` ou `$doc`

Ce qui se passe lors d'un build d'une dérivation

Dans cet exemple regardons ce qu'il se passe si on build la dérivation suivante :

```
nix-repl> :l <nixpkgs>
nix-repl> my-derivation = derivation { name = "foo"; builder = "${bash}/bin/bash"; args = [
./builder.sh ]; system = builtins.currentSystem; }
nix-repl> :b my-derivation
```

1. Déplacement de tous les fichiers nécessaires dans le store

Tout d'abord Nix va détecter que `./builder.sh` est un chemin de fichier dans les arguments et va le passer dans le Nix Store. Pour cela il va le hash d'une certaine manière et y ajouter le nom du fichier à la fin.

Il en va de même si on précise un dossier de source, etc.

2. Création d'un `.drv`

Ensuite Nix va créer un fichier `.drv` dans le nix store.

Si on utilise la commande `nix show-derivation` sur ce fichier, on peut le lire, ce qui nous donne ceci :

```
[snowcode@snowcode ~]$ nix show-derivation /nix/store/yg5ps2gvx39l38w90iyxss0xalcqka0y-foo.drv
{
  "/nix/store/yg5ps2gvx39l38w90iyxss0xalcqka0y-foo.drv": {
    "args": [
      "/nix/store/qfs6b6gcq9xi6gpf11vribfir8xn2nln-builder.sh"
    ],
    "builder": "/nix/store/561wgc73s0x1250hrgp7jm22hhv7yfln-bash-5.2-p15/bin/bash",
    "env": {
      "builder": "/nix/store/561wgc73s0x1250hrgp7jm22hhv7yfln-bash-5.2-p15/bin/bash",
      "name": "foo",
      "out": "/nix/store/vk7mk50mqswblpi88l86lf40bmrlffs1-foo",
      "system": "x86_64-linux"
    },
    "inputDrvs": {
      "/nix/store/0hnjp6s8k71xm62157v37zg3qzwvl8lx-bash-5.2-p15.drv": [
        "out"
      ]
    },
    "inputSrcs": [
      "/nix/store/qfs6b6gcq9xi6gpf11vribfir8xn2nln-builder.sh"
    ],
    "outputs": {
      "out": {
        "path": "/nix/store/vk7mk50mqswblpi88l86lf40bmrlffs1-foo"
```

```
      }
    },
    "system": "x86_64-linux"
  }
}
```

Ce fichier contient toutes les informations pour build un programme, sans tout les machins de Nix. Il contient les choses suivantes :

- Les "out paths", comme on en a parlé avant par défault il n'y en a qu'un c'est le `$out`. Le hash est calculé en faisant un hash spécifique du .drv ayant un out path vide.
- Les dérivations en input et les fichiers de source, ici nous avons bash (car nous y avons fait référence dans la dérivation) et le builder.sh (car nous avons mis son chemin). Nix va donc récupérer leur chemin dans le Nix store
- Le système, l'exécutable (builder) et ses arguments
- Les variables d'environnement qui sont passées au builder

Tous les chemins fonctionnant avec des hash des différentes dérivations et fichiers, cela assure donc qu'une même dérivation donnera toujours le même résultat quoi qu'il arrive.

3. Le build de la dérivation

Ensuite grâce au fichier intermédiaire `.drv`, Nix n'a plus qu'a ajouter les variables d'environnement et exécuter le builder avec les bons arguments (et uniquement celles là).

Une fois cela fait, un raccourcis vers le dossier de la dérivation dans le Nix store est créé sous le nom de "result" dans le dossier courrant.

Revision #2
Created 8 July 2023 11:48:34 by SnowCode
Updated 10 July 2023 18:01:46 by SnowCode