NixOS

un sistema operativo riproducibile, dichiarativo e affidabile

Obiettivi di questa presentazione

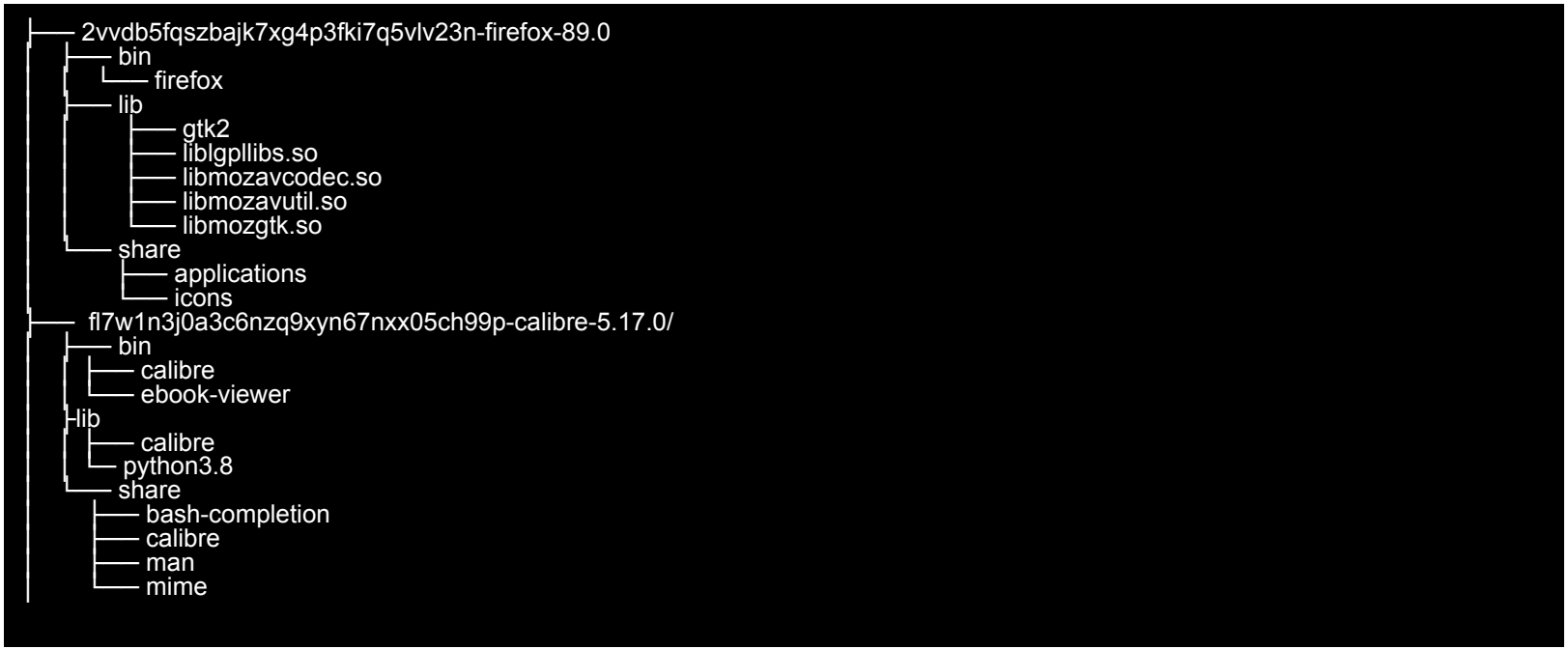
- Scoprire cosa sono Nix e NixOs
- Perchè usarlo sulla propria macchina?
- Perchè usarlo per gestire gli ambienti di sviluppo?
- Ambienti di deploy
- Risorse

Nix

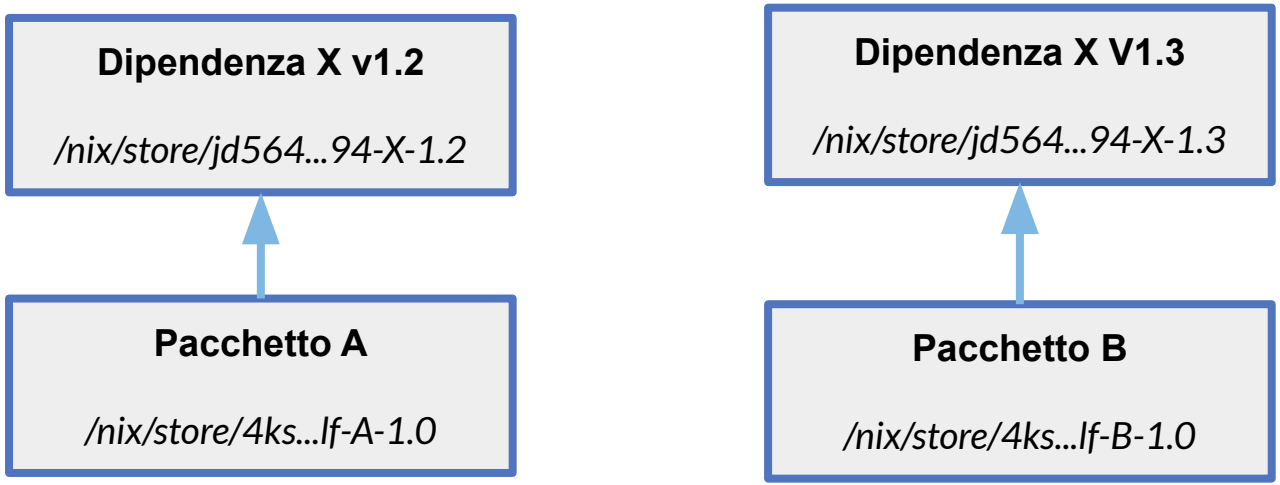
Cos’è Nix?

- Un gestore di pacchetti “purely functional”
- I suoi pacchetti sono descritti da funzioni scritte nel linguaggio Nix
- Crea e gestisce solo pacchetti “completi”
- Permette di installare pacchetti senza permessi di root
- Aggiornamenti atomici
- Permette di installare più versioni dello stesso pacchetto

/nix/store


2vvdb5fqszbajk7xg4p3fki7q5vlv23n-firefox-89.0
bin
firefox
lib
gtk2
liblgpllibs.so
libmozavcodec.so
libmozavutil.so
libmozgtk.so
share
applications
icons
fl7w1n3j0a3c6nzq9xyn67nxx05ch99p-calibre-5.17.0/
bin
calibre
ebook-viewer
lib
calibre
python3.8
share
bash-completion
calibre
man
mime

Due versioni dello stesso pacchetto



Hello Expression

```
let
 pkgs = import <nixpkgs> {} ;
in
pkgs.stdenv.mkDerivation {
 pname = "hello";
 version = "2.10";

 src = pkgs.fetchurl {
   url = "mirror://gnu/hello/hello-2.10.tar.gz";
   sha256 = "0ssi1wpaf7plaswqqjwigppsg5fyh99vdlb9kzl7c9lng89ndq1i";
 };

 buildPhase = "make";
 installPhase = "make install";
}
```

Comandi nix

- nix-env: manipola o consulta il nix environment dell’utente
- nix-build: costruisce una derivazione partendo da un espressione
- nix-store: manipola o consulta il nix store
- nix-collect-garbage: rimuove pacchetti non utilizzati
- nix-shell: avvia una shell partendo da un espressione
- nix-channel: lista e seleziona canale

Pinning

Possiamo specificare la versione di un pacchetto “pinnando” per specificare quali versioni dei pacchetti vogliamo utilizzare

```
let
 pkgs-pinned = import (fetchTarball
                        “https://github.com/NixOS/nixpkgs/archive/12...323.tar.gz") {} ;
in
   ...
```

Ambienti di sviluppo

- Utilizzando nix-shell
- Utilizzando direnv+lorri

NixOs

Quali sono i suoi vantaggi?

- Descrivere la propria macchina in un unico file
- Pacchetti installati-rimossi in modo “giusto”
- Non serve installare tutto globalmente
- Rollback

Quali sono gli svantaggi?

- Learning curve ripida
- Tutte le personalizzazioni devono essere scritte in Nix
- Difficile cambiare alcune cose, es: systemd

Ambienti di deploy

Docker

```
{ pkgs ? import <nixpkgs> {} }:
pkgs.dockerTools.buildLayeredImage {
   name = "nix-redis";
   tag = "latest";
   contents = [pkgs.redis];
}
```

Risorse

- nixos.org
- search.nixos.org
- Nix, NixOs e NixOps manual
- NixPills
- nixos.wiki: NixosWiki (non ufficiale)
- Esempi docker