NixOS – tak trochu jiná Linuxová distribuce

Michal Sojka

ms@2x.cz
@wentasah

ČVUT v Praze

17. 9. 2022

Outline I

Úvod

Nix

Nixpkgs

NixOS
    Konfigurace, Moduly

Závěr

Obsah

Úvod

Nix

Nixpkgs

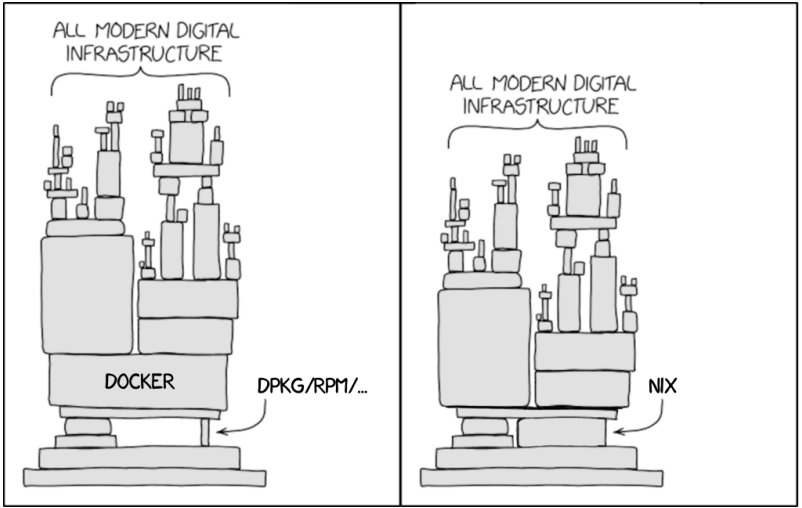
NixOS
- Konfigurace, Moduly

Závěr

O čem bude řeč?

- https://nixos.org/
- Linuxová distribuce
  - Existuje cca od roku 2003
  - Využívá správce balíků Nix (hlavní odlišnost od jiných distribucí)
  - Založená na systemd
  - V základu nekompatibilní s filesystem hierarchy standard (FHS)
- Nix otevírá uživatelům široké možnosti práce se systémem či jeho modifikace
  - Nevídané v jiných distribucích
  - Řeší velké množství problémů se závislostmi (není potřeba používat Docker/kontejnery :-)
  - Říká se, že Nix je složitý
    - je jiný, není to mainstream (zatím)

Nix vs. Docker



Obsah

Úvod

Nix

Nixpkgs

NixOS
Konfigurace, Moduly

Závěr

Nix

- Správce balíků založený na funkcionálním programovacím jazyku
  - Předpis (derivation), jak zkompilovat daný software
  - Jazyk je jednoduchý: JSON + Lambda funkce
  - Nebudeme zabíhat do detailů

Ukázka Nix výrazu pro derivaci

```
stdenv.mkDerivation rec {
  pname = "hello";
  version = "2.12.1";

  src = fetchurl {
    url = "mirror://gnu/hello/hello-${version}.tar.gz";
    sha256 = "sha256-jZkUKv2SV28wsM18tCqNxoCZmLxdYH2Idh9RLibH2yA=";
  };

  buildInputs = [ foo bar ];

  configureFlags = [ "--with-foo" "--with-bar" ];
  doCheck = true;

  meta = with lib; {
    description = "A program that produces a familiar, friendly greeting";
    homepage = "https://www.gnu.org/software/hello/manual/";
    license = licenses.gpl3Plus;
    maintainers = [ maintainers.eelco ];
    platforms = platforms.all;
  };
}
```

Hlavní vlastnosti

- Víc verzí jednoho balíku může být nainstalováno současně
  - Řeší spoustu problémů se závislostmi
- Je možné ho používat ve většině Linuxových distribucí + v MacOS (Darwin)
- Vše je uloženo v `/nix/store` (žádné jmenné kolize s ostatními distribucemi)
  - Většinu času je mountován jen pro čtení (ani root nemůže omylem rozbít systém)

```
/nix/store/4w9a5vcmb553j2s9wxpsj2x18rg80jnc-fmt-7.1.3
/nix/store/ncr3klpks49bkrb8hsgdypxr7mx285xc-fmt-7.1.3-dev
/nix/store/q9cqy0lq61zai0p8zi10yvdhxwfw4nv2-glibc-2.32-54-bin
/nix/store/90n0kdhgxhcig5imd3m44kcxdwz6903c-gmp-6.2.1
/nix/store/n2dprmax869lb76gwpc2ygaqwbdgq7r0-gnutar-1.34
/nix/store/hm66rnx7pm0czliwa42nhnk8vfwwn774-gzip-1.10
/nix/store/1v0pmb9f8jmn126nvbs2k2varzqfqczq-icu4c-69.1
/nix/store/k1chpw09lf8yi4sk2q2x0s4bg5mxm7hb-json-c-0.15
/nix/store/fvb3znd0x7ijk0bqg3m1rndbzpspamrg-keyutils-1.6.3-lib
```

Jak program najde knihovny?

- Cesty v programu/skripty jsou (většinou) absolutní
- Žádný program se nedokazuje na předem známá umístění (`/usr/lib`)
- Podobný princip, jako ukazatele/reference v programu – pokud nemám ukazatel (abs. cestu), nedostanu se k objektu (souboru)
- Automatická správa „paměti“ – garbage collector (`nix store gc`)

Další vlastnosti

Kompilace balíků v sandboxu

- Zápis povolen jen do $out
- Dostupné jen explicitně uvedené závislosti (ne celý /nix/store a ostatní adresáře).
- Přístup k síti není dovolen („fixed-output“ derivation je výjimka – např. zdrojové kódy)
- Instalace jednoho balíku nemůže rozbít ostatní balíky

Reprodukovatelnost

- Předpis (derivation) obsahuje:
  - odkazy na závislosti
  - zdrojový kód je také závislost (většinou „fixed-output“ derivation)
  - příkazy/parametry, jak program zkompilovat (vytvořit obsah v `$out`)
- Hash (sanitizovaného) předpisu je součást cesty v `/nix/store`
- Vykonáním předpisu (realizace) vznikne binární balík v `$out`
- Snaha, aby výstupu `$out` byl závislý pouze na vstupech
  - 99.77% instalačního CD je reprodukovatelné (https://r13y.com/)

Základní použití Nixu

- Dva režimy:
  - Klasický: `nix-build`, `nix-shell`, `nix-env`
  - Experimentální: `nix build`, `nix develop`, `nix profile`, musí se explicitně povolit:
    - `nix --experimental-features "nix-command flakes"`… nebo
    - `echo "experimental-features = nix-command flakes" >> ~/.config/nix/nix.conf`
- Správa balíků

apt	Nix classic	Nix experimental
apt install hello	nix-env -iA nixpkgs.hello	nix profile install nixpkgs#hello
dpkg -l	nix-env -q	nix profile list
apt remove hello	nix-env -e hello	nix profile remove …
apt upgrade hello	nix-env -uA nixpkgs.hello	nix profile upgrade …

Obsah

Úvod

Nix

Nixpkgs

NixOS
Konfigurace, Moduly

Závěr

Nixpkgs

- Celá distribuce NixOS „žije“ v repozitáři https://github.com/NixOS/nixpkgs/ (monorepo)
- `pkgs/` – jednotlivé balíky
  - Nejvíc balíků ze všech distribucí (73 tis.)
  - Nejvíc up-to-date balíků (viz https://repology.org/)
- `nixos/` – kompletní OS využívající `pkgs/`.
- Nixpkgs je možné bezproblémově využívat i v jiných distribucích
- NixOS používání trochu zjednodušuje (OpenGL, shell completion, automatické startování služeb, …)
  - Nix se snaží aby byl kompletně oddělený od hostované distribuce

Obsah

Úvod

Nix

Nixpkgs

NixOS
- Konfigurace, Moduly

Závěr

NixOS

- Co přináší NixOS oproti použití Nixu v jiné distribuci?
  - Deklarativní konfigurace celého systému
- Celý OS (rootfs) = balík (derivation)
  - závislosti = všechny nainstalované balíky
  - výstup = všechny konfigurační soubory
  - konfigurační soubory (např. init skripty) obsahují absolutní cesty k programům

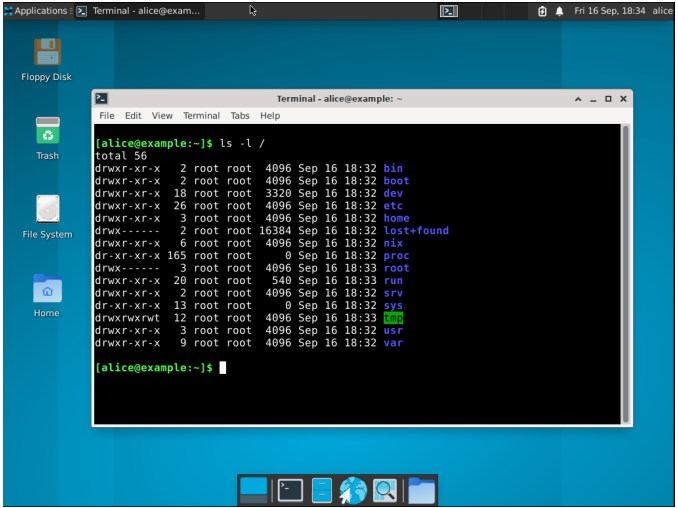
Ukázková konfigurace

```
{ config, pkgs, ... }:
{
  boot.loader.systemd-boot.enable = true;

  networking.hostName = "example";
  time.timeZone = "Europe/Prague";

  services.openssh.enable = true;

  services.xserver.enable = true;
  services.xserver.displayManager.lightdm.enable = true;
  services.xserver.desktopManager.xfce.enable = true;

  environment.systemPackages = with pkgs; [ emacs wget ];

  users.users.alice = {
    isNormalUser = true;
    extraGroups = [ "wheel" ];
    # secret
    hashedPassword = "$6$UeUi3mis.Na253Zz$l3RlyTEn8ZI0eFUgyJAwjvxr.TMx8rZc1sP/2qbKke1p4fxbSLx2iQ1mV5Qot3p9mofxWcPLxSnH/ciOYo3Pu0";
    packages = with pkgs; [ firefox thunderbird ];
  };
  documentation.nixos.enable = false;
  system.stateVersion = "22.11";
}
```

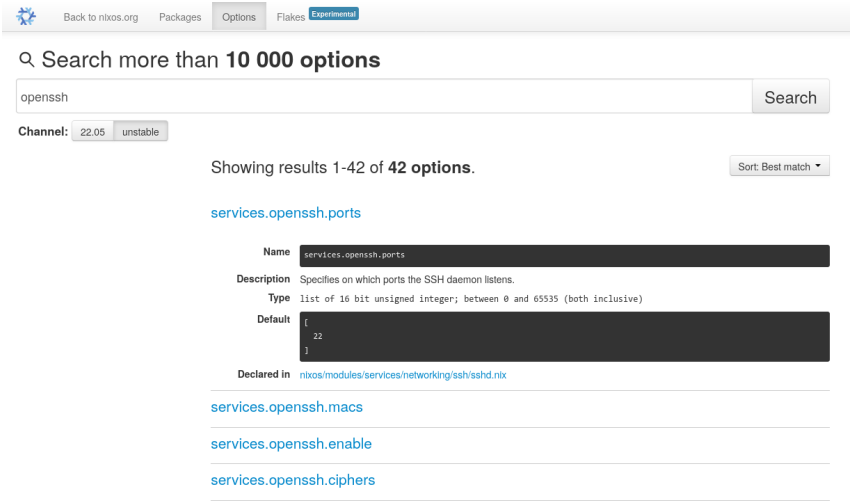
Vytvořit a spustit ve VM:

```
$ nixos-rebuild build-vm --no-flake -I nixos-config=$PWD/configuration.nix
$ ls -l ./result
./result -> /nix/store/kk9979hfqy7g345hkwj6ygjbiif14j21-nixos-vm
$ ./result/bin/run-example-vm
```

Výsledek – VM v qemu



search.nixos.org (`man configuration.nix`)



Skript nixos-rebuild

- Jednoduchý wrapper nad nixem

  `nix-build '<nixpkgs/nixos>' -A system`
- Podpříkazy:
  - `nixos-rebuild build` – pouze připraví (nový) systém
  - `nixos-rebuild switch` – připraví (nový) systém a „přepne“ do něj
  - `nixos-rebuild boot` – připraví (nový) systém a příští reboot do něj nabootuje

Jak to funguje?

./result/bin/run-example-vm:

```
exec /nix/store/izjm0lhwwwd8qw3mjsii1l86psqy1szd-qemu-host-cpu-only-7.1.0/bin/qemu-kvm -cpu max \
  -name example \
  -m 1024 \
  -virtfs local,path=/nix/store,security_model=none,mount_tag=nix-store \
  -drive cache=writeback,file="example.qcow2",id=drive1,if=none,index=1,werror=report -device virtio-blk-p
  … \
  -kernel ${NIXPKGS_QEMU_KERNEL_example:-/nix/store/2dkv6gq2q4hn9fvxm9q50j8l04sizi8b-nixos-system-example-
  -initrd /nix/store/2dkv6gq2q4hn9fvxm9q50j8l04sizi8b-nixos-system-example-22.11pre-git/initrd \
  -append "$(cat /nix/store/2dkv6gq2q4hn9fvxm9q50j8l04sizi8b-nixos-system-example-22.11pre-git/kernel-para
            init=/nix/store/2dkv6gq2q4hn9fvxm9q50j8l04sizi8b-nixos-system-example-22.11pre-git/init regInf
  $QEMU_OPTS \
  "$@"
```

Co je uvnitř?

```
$ ls -l /nix/store/2dkv6gq2q4hn9fvxm9q50j8l04sizi8b-nixos-system-example-22.11pre-git
Permissions Size User Group Date Modified Name
.r-xr-xr-x   14k root root   1 Jan  1970  activate
lrwxrwxrwx    91 root root   1 Jan  1970  append-initrd-secrets -> /nix/store/36jrskmfkkz5qkh1c0jahph3iyd635n9-appen
dr-xr-xr-x     - root root   1 Jan  1970  bin
.r--r--r--     0 root root   1 Jan  1970  configuration-name
.r-xr-xr-x  2.2k root root   1 Jan  1970  dry-activate
lrwxrwxrwx    51 root root   1 Jan  1970  etc -> /nix/store/hl8bixf8gnsdl57srywxaw85cikhja9y-etc/etc
.r--r--r--   133 root root   1 Jan  1970  extra-dependencies
lrwxrwxrwx    65 root root   1 Jan  1970  firmware -> /nix/store/5c962xh6icrwr4wxsivc0h2gn5ah7gic-firmware/lib/firmw
.r-xr-xr-x  4.5k root root   1 Jan  1970  init
.r--r--r--     9 root root   1 Jan  1970  init-interface-version
lrwxrwxrwx    71 root root   1 Jan  1970  initrd -> /nix/store/a4ihsxmx0q0ynrpmj1sy6jdfsj43qy9s-initrd-linux-5.15.67
lrwxrwxrwx    65 root root   1 Jan  1970  kernel -> /nix/store/si46agdxfw02zc37gjylbbq2pxrwjyah-linux-5.15.67/bzImag
lrwxrwxrwx    58 root root   1 Jan  1970  kernel-modules -> /nix/store/fhnhhaysdqxwibni6zjp6karp4cb4g93-kernel-modul
.r--r--r--    24 root root   1 Jan  1970  kernel-params
.r--r--r--    12 root root   1 Jan  1970  nixos-version
dr-xr-xr-x     - root root   1 Jan  1970  specialisation
lrwxrwxrwx    55 root root   1 Jan  1970  sw -> /nix/store/lyj37l6i5kgr7izrpyymxpyyhrii53bi-system-path
.r--r--r--    12 root root   1 Jan  1970  system
lrwxrwxrwx    57 root root   1 Jan  1970  systemd -> /nix/store/a026zm6yhqsil81q6f57jgs65ipjgfgz-systemd-251.4
```

Co je uvnitř – vysvětlení (zjednodušené)

- `init` (Stage 2 init) – základní inicializace systému (vytvoří `/etc`, `/tmp`) a spustí `activate` a pak `systemd`
- `activate` – skript vytvořen při `nixos-rebuild` na základě konfigurace
  1. vytvoří/maže uživatele
  2. nastaví symlinky do `/nix/store` v `/etc`
  3. vytvoří další potřebné adresáře (`/var/tmp`)
  4. …
  5. vytvoří symlink `/run/current-system` ukazující na aktuální systém
- `sw` – globální `/bin` (`$PATH`), `/lib`, atd., ale plné symlinků

nixos-rebuild switch

- Přepnutí na novou konfiguraci (profil) za běhu
- Jednotlivé verze (profily) se symlinkují do `/nix/var/nix/profiles/system-655-link`
- `…/bin/switch-to-configuration`
  1. Nainstaluje nebo aktualizuje bootloader
  2. Zavolá `…/activate`
  3. Najde rozdíly v aktuálně běžících a nových službách `systemd` a restartuje změněné, vypne zakázané, zapne nové

Přepínání konfigurací

- Teoreticky (a většinou i prakticky) lze přepnout z čehokoli na cokoli
  - upgrade, rollback, změna konfigurace
- Pomalý download upgradu ničemu nevadí (nic nekoliduje)
- Před přepnutím můžu spustit:

```
$ nix store diff-closures /nix/var/nix/profiles/system-{655,658}-link
acpi-call: 2020-04-07-5.8.16 → 2020-04-07-5.8.18
baloo-widgets: 20.08.1 → 20.08.2
bluez-qt: +12.6 KiB
dolphin: 20.08.1 → 20.08.2, +13.9 KiB
kdeconnect: 20.08.2 →  , -6597.8 KiB
kdeconnect-kde:  → 20.08.2, +6599.7 KiB
```

Jak se z konfigurace vytvoří celý systém?

- Konfigurační systém tvořen „moduly“
- Každý modul má „interface“ a „implementaci“
- Konfigurace všech modulů se sloučí dohromady (kolize se řeší prioritami)
- Implementace vytvoří vše potřebné na základě sloučené konfigurace
- Moduly se mohou includovat
- Uživatelská konfigurace je také modul
- NixOS tests

Příklad Modulu

Interface:

```
options = {
  services.throttled = {
    enable = mkEnableOption
      (lib.mdDoc "fix for Intel CPU throttling");

    extraConfig = mkOption {
      type = types.str;
      default = "";
      description = lib.mdDoc
        "Alternative configuration";
    };
  };
};
```

Implementace:

```
config = mkIf cfg.enable {
  systemd.packages = [ pkgs.throttled ];

  systemd.services.lenovo_fix.wantedBy = [ "multi-user.target" ];

  environment.etc."lenovo_fix.conf".source =
    if cfg.extraConfig != ""
    then pkgs.writeText "lenovo_fix.conf" cfg.extraConfig
    else "${pkgs.throttled}/etc/lenovo_fix.conf";

  boot.kernelParams =
    optional (versionAtLeast
      config.boot.kernelPackages.kernel.version "5.9")
      "msr.allow_writes=on";
};
```

Deployment tools

- Není nutné pouštět `nixos-rebuild` na cílovém stroji

```
$ nix build …configuration…
$ nix copy --to myserver ./result
$ ssh ssh://root@myserver $(readlink ./result)/bin/switch-to-configurat
```

- Nástroje, které to automatizují: NixOps, deploy-rs, cachix-deploy

Kontejnery

- Ukázali jsme si NixOS VM
- Stejně snadno se vytvářejí kontejnery
  - Příkaz `nixos-container`
  - Konfigurace `systemd.nspawn`…
  - Docker

```
pkgs.dockerTools.buildImage {  name = "gnuplot";
  config = { Cmd = [ "${pkgs.gnuplot}/bin/gnuplot" ]; };
}
```

Obsah

Úvod

Nix

Nixpkgs

NixOS
- Konfigurace, Moduly

Závěr

Shrnutí

Výhody Nixu/NixOSu

- Deklarativní, reprodukovatelný
- Snadné testování/CI SW s „bleeding edge“ závislostmi
- Snadná modifikace čehokoli v systému
- GitHub Action `install-nix-action` – faster than apt-get
- Vysoká míra automatizace (bootstrap všeho z kompilátoru C)

Nevýhody Nixu/NixOSu

- Náročnější na čas (`nixos-rebuild` po změně konfigurace)
  - Může se časem vrátit
- Ne všechno se dá najít na Stack Overflow
- Ne všechen software je zabalíčkovaný pro Nix
  - často je to snadné, ale SW je hodně
  - někdy složité
  - kontejnery, buildFhsEnv, …

Alternativy

- Samotný Nix na libovolné distribuci nebo na MacOS
- Deklarativní konfigurace (moduly) pro `$HOME`: home-manager
- Dual-boot (sdílený rootfs) s jinou distribucí