# Nix & kontejnery

Jakub Vokoun
jakub.vokoun@wftech.cz

## `whoami(1)`

- linuxák skoro již 20 let
- "liný DevOpsák, který automatizuje, co potká"

# Motivace

- deklarativní spouštění kontejnerů na NixOS
- deklarativní buildy
- deklarativní[1] řešení pro `docker compose`

[1] Ano, slovo deklarativní tu dnes bude slyšet často

# Nix 101

## Nix

- jazyk
- CLI nástroj
- operační systém

# Nix 101

## Jazyk Nix

- dynamicky typovaný
- funkcionální
- lazy evaluation
- DSL (Domain Specific Language)

# Proč Nix?

## Co znamená "deklarativní"

- **Reprodukovatelnost -** stejný výsledek na každém stroji
- **Immutabilita -** buildy se nemění, jen se vytváří nové
- **Rollback -** jednoduché vrácení na předchozí verzi
- **Izolace -** žádné konflikty závislostí

# Kontejnery 101

- Docker
- Podman
- LXC (Linux Containers), Incus
- `systemd-nspawn(1)`

# Kontejnery *tradičně*

```
docker run image:tag command
```

```
docker run --detach image command
```

```
docker run --rm --interactive --tty image command
```

# Kontejnery *tradičně*

```
# Dockerfile
FROM python:3.13
WORKDIR /usr/local/app

# Install the application dependencies
COPY requirements.txt ./
RUN pip install --no-cache-dir -r requirements.txt

# Copy in the source code
COPY src ./src
EXPOSE 8080

# Setup an app user so the container doesn't run as the root user
RUN useradd app
USER app

CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port", "8080"]
```

# Kontejnery *tradičně*

```
docker build -t local/my-python-app .
```

```
docker run -p 8080:8080 local/my-python-app:latest
```

# Docker vs Nix

| Aspekt | Docker | Nix |
|---|---|---|
| **Reprodukovatelnost** | Závisí na čase buildu | Garantovaná |
| **Velikost image** | Obsahuje OS layer | Jen potřebné závislosti |
| **Cache** | Layer-based | Content-addressed |
| **Rollback** | Ruční tag management | Automatický |
| **Dev prostředí** | Dockerfile + compose | Jedna konfigurace |

# Kontejnery *the Nix way*

## Incus - základní konfigurace

```
{ config, pkgs, lib, ... }:

{
  virtualisation.incus = {
    enable = true;
    ui = { enable = true; };
  };
  networking.nftables.enable = true;
  users.extraGroups.incus-admin.members = [ "jakub" ];
}
```

```
sudo nixos-rebuild --switch
sudo incus admin init --minimal
```

# Kontejnery *the Nix way*

## Incus - custom images

```
nixosConfigurations = {
  container = inputs.nixpkgs.lib.nixosSystem {
    system = "x86_64-linux";
    modules = [
      "${inputs.nixpkgs}/nixos/modules/virtualisation/lxc-container.nix"
      (
        { pkgs, ... }:
        {
          environment.systemPackages = [ pkgs.vim ];
        }
      )
    ];
  };
};
```

# Kontejnery *the Nix way*

## Incus - custom images

```
nixosConfigurations = {
  vm = inputs.nixpkgs.lib.nixosSystem {
    system = "x86_64-linux";
    modules = [
      "${inputs.nixpkgs}/nixos/modules/virtualisation/lxd-virtual-machine.nix"
      (
        { pkgs, ... }:
        {
          environment.systemPackages = [ pkgs.vim ];
        }
      )
    ];
  };
};
```

# Kontejnery *the Nix way*

## Incus - custom images

```
incus image import --alias nixos/custom/vm \
    $(nix build .#nixosConfigurations.vm.config.system.build.metadata --print-out-paths)/tarball/nixos-system-x86_64-linux.tar.xz \
    $(nix build .#nixosConfigurations.vm.config.system.build.qemuImage --print-out-paths)/nixos.qcow2
```

```
incus image import --alias nixos/custom/container \
    $(nix build .#nixosConfigurations.container.config.system.build.metadata --print-out-paths)/tarball/nixos-system-x86_64-linux.tar.xz \
    $(nix build .#nixosConfigurations.container.config.system.build.squashfs --print-out-paths)
```

# Kontejnery *the Nix way*

## Docker - základní konfigurace

```
{ config, pkgs, ... }: {
  virtualisation.docker.enable = true;
  users.extraGroups.docker.members = [ "jakub" ];
}
```

# Kontejnery *the Nix way*

## Docker - pokročilá konfigurace

```
{ config, pkgs, ... }: {
  virtualisation.docker = {
    enable = true;
    daemon.settings = {
      dns = [ "1.1.1.1" "8.8.8.8" ];
      log-driver = "journald";
      registry-mirrors = [ "https://mirror.gcr.io" ];
      storage-driver = "overlay2";
    };
    rootless = {
      enable = true;
      setSocketVariable = true;
    };
  };
}
```

# Kontejnery *the Nix way*

## Podman

```
{ config, pkgs, ... }: {
  virtualisation.containers.enable = true;
  virtualisation = {
    podman = {
      enable = true;
      # Docker alias
      dockerCompat = true;
      # Required for `podman-compose`
      defaultNetwork.settings.dns_enabled = true;
    };
  };

  # Additional packages
  environment.systemPackages = with pkgs; [ docker-compose podman-compose ];
}
```

# Kontejnery *the Nix way*

## Spuštění kontejneru

```
{ config, pkgs, ... }: {
  # Use Podman
  virtualisation.oci-containers.backend = "podman";

  # my-python-app container
  virtualisation.oci-containers.containers = {
    my-python-app = {
      image = "local/my-python-app:latest";
      ports = [ "8080:8080" ];
      volumes = [ ];
      cmd = [ "uvicorn" "app.main:app" "--host" "0.0.0.0" "--port" "8080" ];
    };
  };
}
```

# Kontejnery *the Nix way*

## Build

```
{ pkgs ? import <nixpkgs> { }
, pkgsLinux ? import <nixpkgs> { system = "x86_64-linux"; } }:

pkgs.dockerTools.buildImage {
  name = "hello-docker";
  config = { Cmd = [ "${pkgsLinux.hello}/bin/hello" ]; };
}
```

# Kontejnery *the Nix way*

## Build

```
nix-build hello-docker.nix
/nix/store/15iq02c94d0r3ha2kd7rhz7z8035v8hc-docker-image-hello-docker.tar.gz
```

```
tree
.
├── hello-docker.nix
└── result -> /nix/store/15iq02c94d0r3ha2kd7rhz7z8035v8hc-docker-image-hello-docker.tar.gz
```

```
docker load < result
```

# Kontejnery *the Nix way*

## `compose2nix`

```
nix run github:aksiksi/compose2nix -- -h
```

```
{ config, pkgs, ... }: {
  # Add `compose2nix` package
  environment.systemPackages = [ pkgs.compose2nix ];
}
```

# Kontejnery *the Nix way*

## `compose2nix`

```
compose2nix -project=my-app --inputs=compose.yml --output=compose.nix --runtime=docker --env_files=.env
```

```
compose2nix -project=my-app --inputs=compose.yml --output=compose.nix --runtime=podman --env_files=.env
```

# Kontejnery *the Nix way*

## `compose2nix` - labely

```
services:
  my-service:
    labels:
      # Enable
      - "compose2nix.settings.autoStart=true"
```

## `compose2nix` - limitace

- nemá 1:1 podporu pro `depends_on`

# Demo

## 1. Nix image build - webový server

```
nix-build demo/buid/default.nix
docker load < result
docker run -p 8080:8080 nix-web-server:latest
# curl localhost:8080
```

## 2. compose2nix - Gitea + PostgreSQL

```
# Ukázka původního compose.yml
# Ukázka vygenerovaného compose.nix
nixos-rebuild switch
systemctl status docker-gitea-server
```

# Reference

- https://github.com/the-nix-way/nix-docker-examples/
- https://github.com/aksiksi/compose2nix