Build and Development Environments for Microservices with Nix

Christine Koppelt
Senior Consultant @ INNOQ
microxchg 2018

The Problem

Build & Development Environments

- Require native tools
  - Build tools, Compilers, Test tools, Runtimes, …
- Should be reproducible & changeable
- Want: Identical build environments with fixed versions everywhere
  - Developer machines
  - CI Server

(Many) Microservices: (Many) Environments

- Developer may want to switch between environments of multiple services
- Environment setup for new developers should happen fast
- Tools can be provided only for a single project

A possible solution: Nix

What is Nix?

- Package Manager
- Contains a broad range of tools
  - ~13.000 packages
  - Own packages can be added
- Own configuration language
- Works on MacOS and Linux
- Immutable package store, multi-version support

Loading tools on the fly

nix-shell -p a_package

```
ck@ck-innoq:~/microxchg$ java -version
openjdk version "1.8.0_131"
ck@ck-innoq:~/microxchg$ nix-shell -p openjdk9 maven
[nix-shell:~/microxchg]$ java -version
openjdk version "9.0.4-internal"
```

What happens

- Downloads packages
- Stores them at /nix/store

  Example:

```
/nix/store/2fiavk609lgb9wsr560lkjf6wyx7d9a3-apache-maven-3.5.2
```

- Sets Links

```
[nix-shell:~/Dokumente/microxchg]$ which mvn
/nix/store/2fiavk609lgb9wsr560lkjf6wyx7d9a3-apache-
maven-3.5.2/bin/mvn
```

Write a default.nix script

```
with import <nixpkgs>{};
stdenv.mkDerivation {
  name = "my-service";
  buildInputs = [openjdk9 maven];
}
```

Loading configuration

```
nix-shell

nix-shell --run "your-test-command"
```

Version Pinning

```
let
  hostPkgs = import <nixpkgs> {};
  nixpkgs = (hostPkgs.fetchFromGitHub {
    owner = "NixOS";
    repo = "nixpkgs-channels";
    rev = "9c31c72cafe536e0c21238b2d47a23bfe7d1b033";
    sha256 = "0pn142js99ncn7f53bw7hcp99ldjzb2m7xhjrax00xp72zswzv2n";
  });
in
with import nixpkgs {};
stdenv.mkDerivation {...}
```

Configure Tools

```
with import <nixpkgs>{};

let curl = pkgs.curl.override {
  zlibSupport   = true;
  sslSupport    = true;
  http2Support  = false;
};
in
stdenv.mkDerivation {
  name = "my-service";
  buildInputs =  [ openjdk9 maven curl ];
}
```

Define new package

```
a_new_package = pkgs.stdenv.mkDerivation rec {
   name = "a-new-package-${version}";
   version = "2.7.1";
   src = fetchurl { url = "http://..."; sha256 = "1lppzd...";};
   phases = [ "installPhase" ];
   buildInputs = [ pkgs.unzip ];
   installPhase = ''
      mkdir -p $out/new-package
      unzip $src -d $out/new-package
    '';
};
```

Add it to buildInputs

```
stdenv.mkDerivation {
 name = "my-service";
 buildInputs =
   [openjdk9 maven a_new_package];
}
```

Extension

- Use nix for building the project
  - Wrapper for a lot of build systems
- Using NixOS
  - Operating System based on Nix and systemd
  - Declarative configuration for everything
  - Rollbacks, Versioning
  - Testing Framework

Benefits

- Nix
  - Makes it possible to create environments which are: Scripted, versioned, immutable, reproducible
- NixOS
  - Extends the concept for system configuration & services

Caveats

- Steep learning curve
- Documentation is not beginner friendly

Questions?

Christine.Koppelt@innoq.com