Container Build (without root)

Tobias Poschwatta and Tobias Watermann

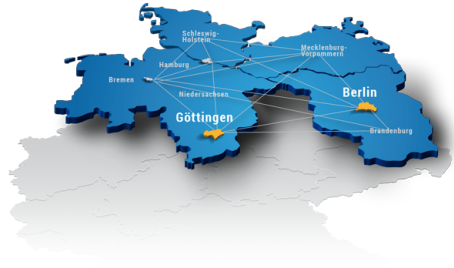
Zuse Institute Berlin

Berlin, Dec 2021

Background

HLRN Systems in NHR@Göttingen (Emmy) and NHR@ZIB (Lise)

- >1k nodes each
- Omnipath network



Users and software from a variety of fields

- Physics
- Chemistry
- Earth sciences
- Engineering

Container usage:

- Singularity
- few users/support cases
- built externally

Building Containers

How to get/build containers

- container provided (repo, software creator)
- from base image via recipe

Image building usually requires

- root on external device
- build service
- VM provided by compute center

Easy way to get started directly on HPC system:

- build from rootfs
- customize in sandbox
- only squashfstools are needed

Livedemo

Nix

- Nix is a different kind of package manager, with many unique features
- It puts each package into a separate location in the *store*

```
$ ls /nix/store/
jsp3h3wpzc842j0rz61m5ly71ak6qgdn-glibc-2.32-54
3365c8gpdcw0vzyk2n4zdzmmns3z6p8x-bash-4.4-p23
bn2zlzs294rkxxjvx7153akgbwxpzi96-hello-2.10
cpx9hzqcxci468qjwx1yk9k8rwmhlqxl-git-2.31.1
...
```

- Different versions and variants of a package can coexist in the store
- RUNPATH is used to link programs and libaries
- Symlinks and environment variables are used to create user environments
- We'll use nix to build Hello World containers for Singularity and Docker.

Hello World Container

- Add `hello` to the store, this also adds all dependencies automatically

```
$ nix-build '<nixpkgs>' -A hello
/nix/store/bn2zlzs294rkxxjvx7153akgbwxpzi96-hello-2.10
```

- The `hello` program is inside the returned store path, we can test it now

```
$ /nix/store/bn2zlzs294rkxxjvx7153akgbwxpzi96-hello-2.10/bin/hello
Hello, world!
```

- For Singularity, we also need a shell

```
$ nix-build '<nixpkgs>' -A busybox
/nix/store/rk29fnf9024l94grw82dsyqrbxr1kb9d-busybox-1.32.1
```

The Hello Closure

- To build a container on the cluster, we need a copy of *hello* and **all** dependencies
- We can ask nix-store for the full list

```
$ nix-store -qR $hello $busybox
/nix/store/5d821pjgzb90lw4zbg6xwxs7llm335wr-libunistring-0.9.10
/nix/store/ckb0qa2yrxrpp0piffgjq9id38gc5z9v-libidn2-2.3.1
/nix/store/jsp3h3wpzc842j0rz61m5ly71ak6qgdn-glibc-2.32-54
/nix/store/bn2zlzs294rkxxjvx7153akgbwxpzi96-hello-2.10
/nix/store/rk29fnf9024l94grw82dsyqrbxr1kb9d-busybox-1.32.1
```

- and create a tarball

```
$ tar czf closure.tgz $(nix-store -qR $hello $busybox)
```

Hello Singularity

- Only few steps are required to make it work with Singularity

```
$ tar xf closure.tgz
```

- Directories and files expected by Singularity

```
$ mkdir etc bin dev home proc sys
$ touch etc/{group,passwd,resolv.conf}
```

- Convenient symlinks in /bin to all binaries

```
$ ( cd bin; ln -s ../nix/store/*/bin/* . )
```

- Build and run

```
$ singularity build hello.sif root
$ singularity exec hello.sif /bin/hello
Hello, world!
```

Hello Docker - Nix Expression

- Nix comes with a complete programming language
- This calls **buildImage** from the standard library to create a Docker container with name **hello-docker**
  and a command to run **hello** from the store.

```
{ pkgs ? import nixpkgs {} }:
pkgs.dockerTools.buildImage {
  name = "hello-docker";
  config = {
    Cmd = "${pkgs.hello}/bin/hello";
  };
}
```

Hello Docker

- nix-build creates a tarball in the store that can be loaded by Docker

```
$ nix-build hello-docker.nix
/nix/store/m8i70rx141vp782hq2afjh052vyszjvs-docker-image-hello-docker.tar.gz
```

- It also creates a convenient *result* symlink

```
$ docker load <result
Loaded image: hello-docker:m8i70rx141vp782hq2afjh052vyszjvs
```

- That's it

```
$ docker run hello-docker:m8i70rx141vp782hq2afjh052vyszjvs
Hello, world!
```