Developer Days
Network Services Orchestrator

NSO in Docker

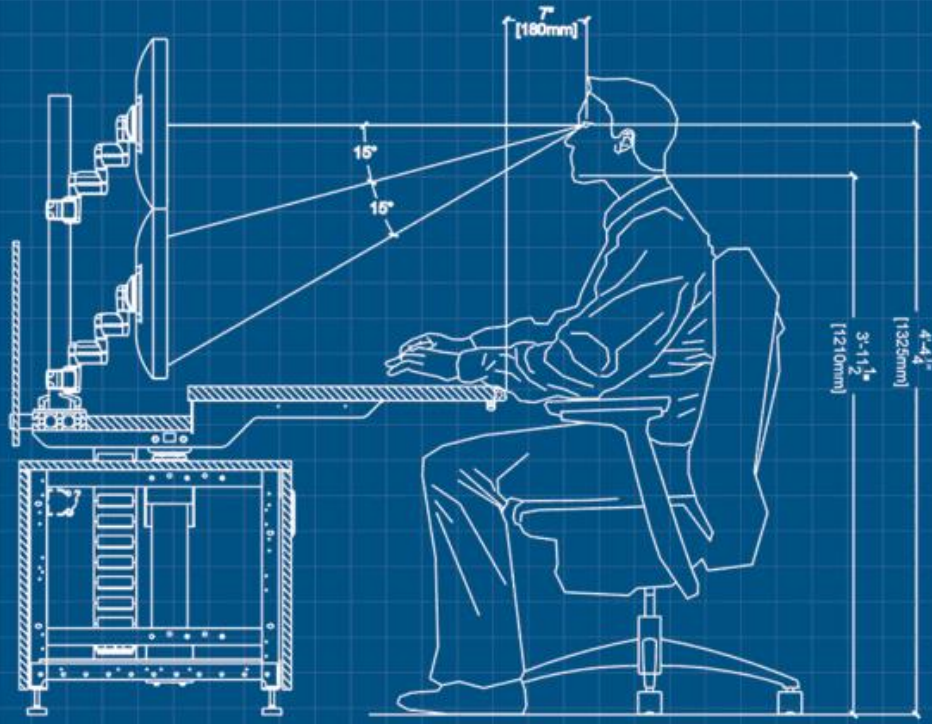
Building a better development environment

Kristian Larsson
Architect
2020-04-29

Why Docker?

- More lightweight than a VM
- More isolation than standard UNIX processes

ERGONOMICS



NID (NSO in Docker) world

- Best practice for way of working
  - With the code to support it
- Encourage composition
  - Test in isolation
- ”Standard form” for repositories
  - Skeletons for NED, package, NSO system
- Uniform & standardized interface
- Shareable work (test & dev) environments
- One-click CI

Code repo interaction

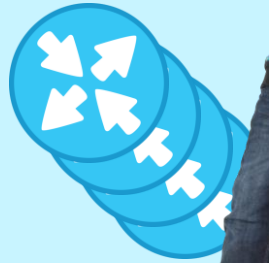
- Edit files
  - Mostly out of scope
  - Plenty of good editors etc
- Compile / build
  - `make build`
- Run / test
  - `make test`

Test environments

- For testing NSO we need not just NSO
  - What is **around** NSO?
- Orchestration system must have devices to orchestrate
  - Netsim, virtual routers, physical routers
- Ensure consistency for testing
  - Codify the environment!
  - Share in team!
- **`testenv`** – a shareable work environment

Test environment

Test environment



Demo

- cisco-ios CLI NED
- make build
- make test
  - Break down testenv-start
- docker ps – see started containers
  - NSO – test NSO instance that have the NED loaded
  - netsim – netsim compiled version of the NED
- Simple test suite
- make testenv-cli
- make testenv-shell

Moar Demo

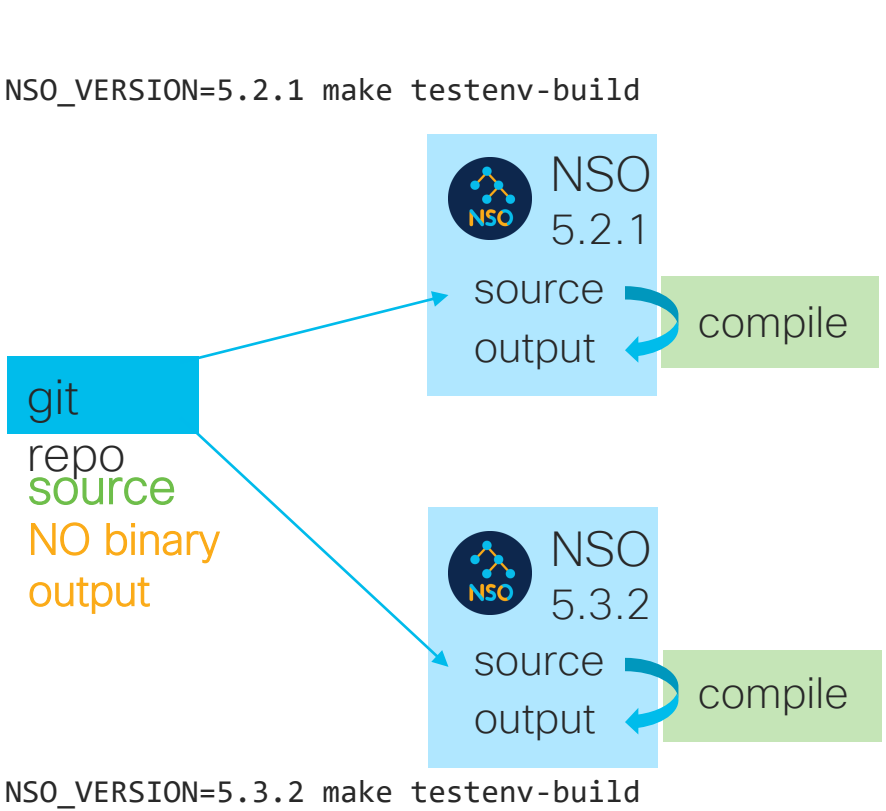
- Unique testenv names
  - Local – username
  - CI – pipeline ID
- NSO version
  - Parallel testenv for multiple NSO versions
  - Cheap & simple to test -> upgrade often

Demo package bgworker

- Bgworker
- testenv-build

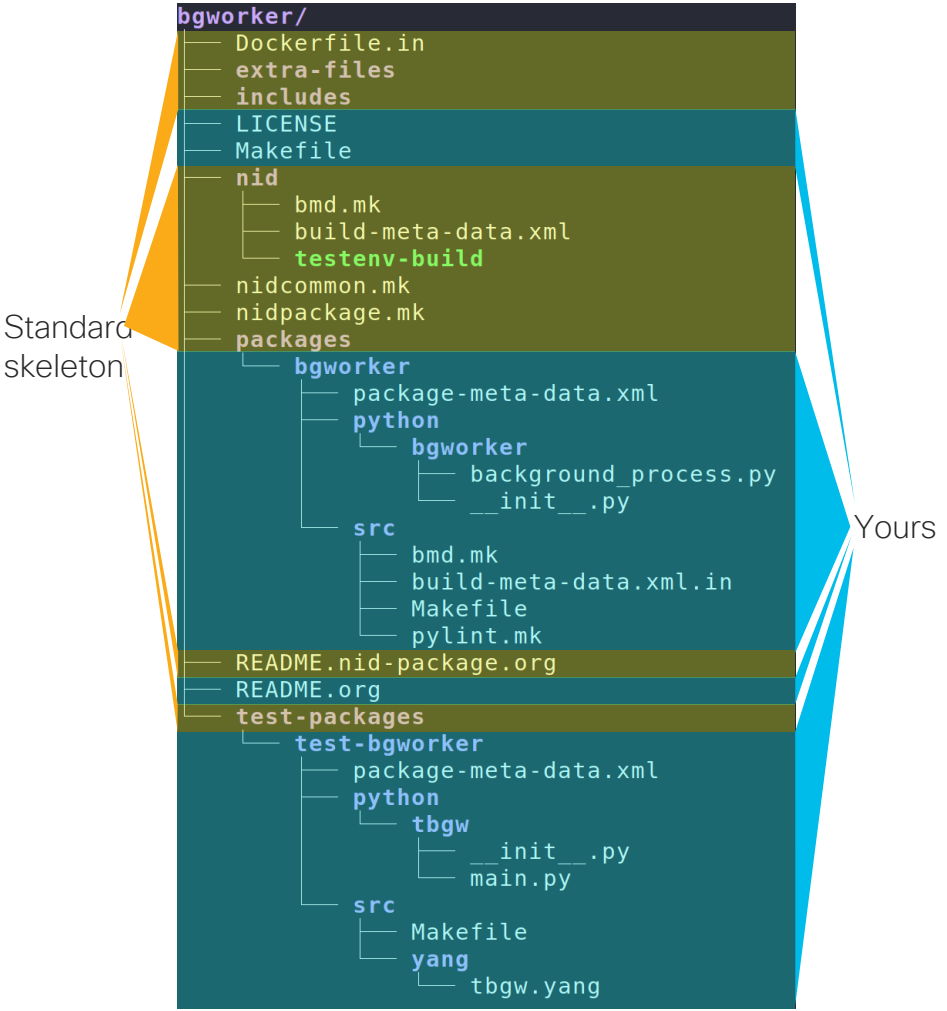
testenv-build

- Loads code into running NSO
  - Source is copied to container
  - Compile happens in helper container
  - Efficient reload/redeploy in NSO container
- Does not pollute work dir with build output



Skeleton Anatomy

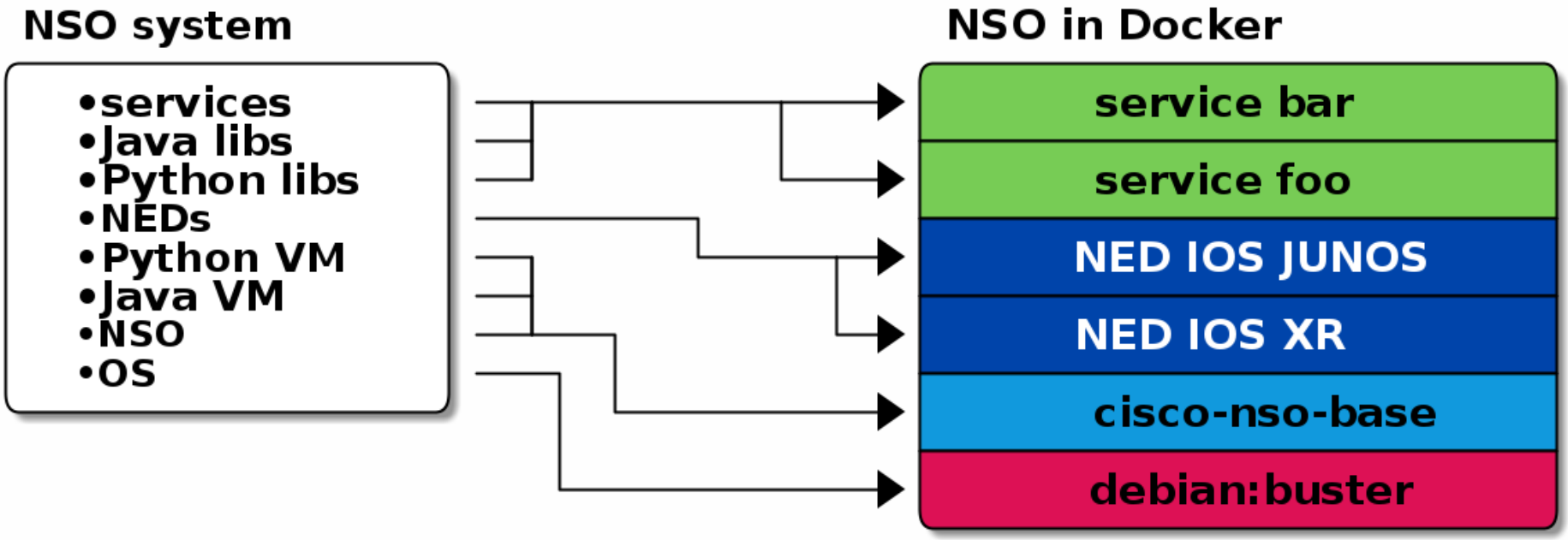
- packages/
- Makefile
- test-packages/



NID skeleton demo; NED for Nokia SR (alu-sr)

- How to use the skeletons?
- Create repo from scratch using NID skeleton for NED
- ./do.sh

What did we just build?

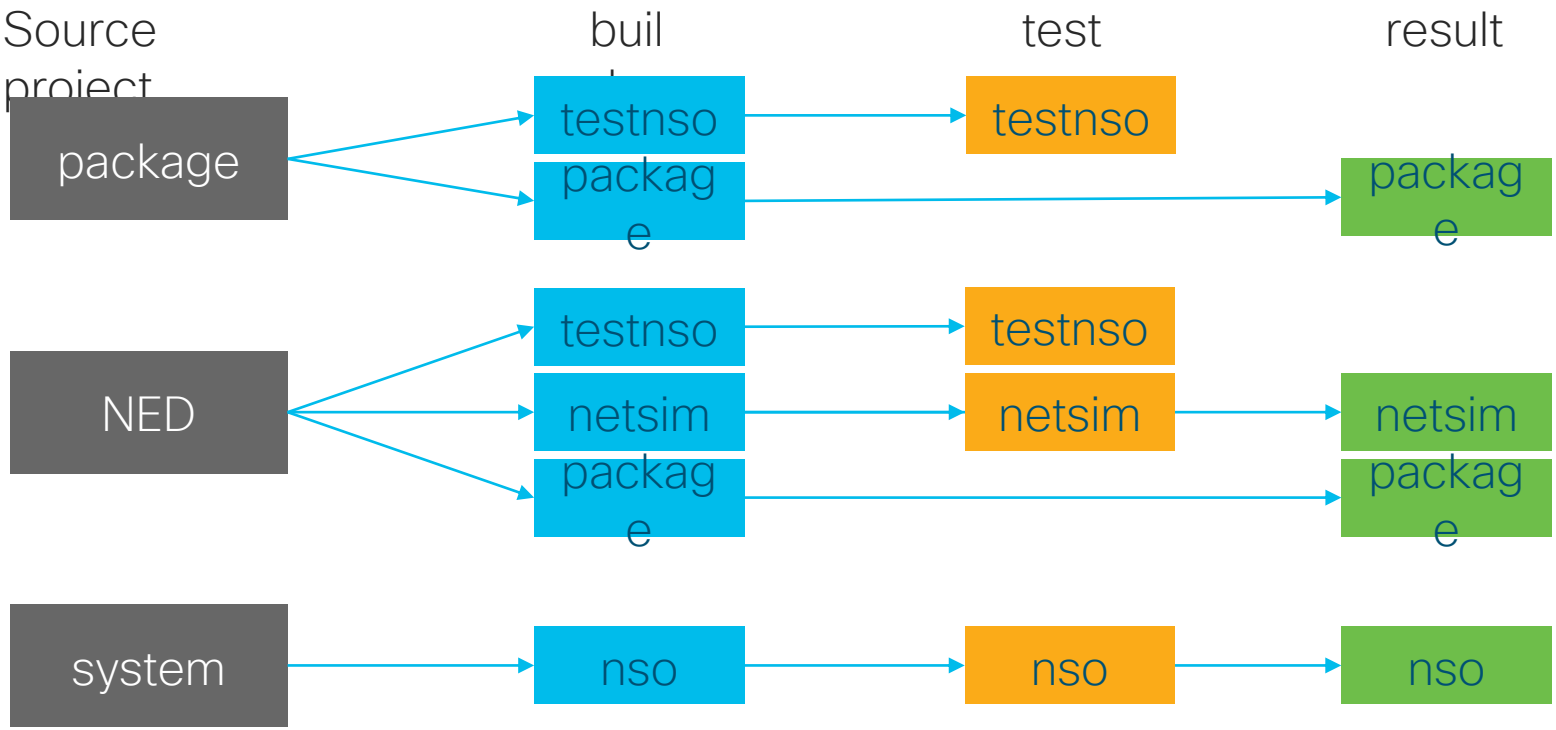


cisco-nso-base

- Docker native
- Thought through file layout
- Config customization via env
- SSH key & TLS handling
- NSO 4 -> 5 upgrade help
- Healtcheck
- Also cisco-nso-dev

Well tested in CI

```
ARG NSO_IMAGE_PATH
ARG NSO_VERSION

# DEP_START
# DEP_END

# Compile local packages in the build stage
FROM ${NSO_IMAGE_PATH}cisco-nso-dev:${NSO_VERSION} AS build
ARG PKG_FILE

COPY / /src
RUN for PKG in $(find /src/packages /src/test-packages -mindepth 1 -maxdepth 1 -type d); do \
  make -C ${PKG}/src || exit 1; \
  make -f /src/nid/bmd.mk -C ${PKG} build-meta-data.xml; \
  done


# produce an NSO image that comes loaded with our NED - perfect for our testing,
# but probably not anything beyond that since you typically want more NSO
# packages for a production environment
FROM ${NSO_IMAGE_PATH}cisco-nso-base:${NSO_VERSION} AS testnso

# DEP_INC_START
# DEP_INC_END

COPY --from=build /src/packages/ /var/opt/ncs/packages/
COPY --from=build /src/test-packages/ /var/opt/ncs/packages/

# Copy in extra files as an overlay, for example additions to
# /etc/ncs/pre-ncs-start.d/
COPY extra-files /


# build a minimal image that only contains the package itself - perfect way to
# distribute the compiled package by relying on Docker package registry
# infrastructure
FROM scratch AS package
COPY --from=build /src/packages/ /var/opt/ncs/packages/
```



Inclusion

- Place include declaration in includes/ directory
  - Name it after package
  - bgworker/package:${NSO_VERSION}
- Make takes Dockerfile.in + includes to produce Dockerfile
- Copy in package from included docker image

includes/bgworker

```
${PKG_PATH}bgworker/package:${NSO_VERSION}
```

Dockerfile.in

```
# DEP_START
# DEP_END
```

```
# DEP_INC_START
# DEP_INC_END
```

Dockerfile

```
# DEP_START
FROM bgworker/package:5.3 AS bgworker
# DEP_END
```

```
# DEP_INC_START
COPY --from=bgworker /var/opt/ncs/packages/ /includes/
# DEP_INC_END
```

Why compose?

- Build many small things
- Compose them together into bigger whole

- ”Everything” is easier on a small thing
  - Development, testing, running, understanding etc.

- NEDs move slowly
  - Upgrade once a month?
  - XR, JUNOS take ~5 min to compile
- Your service packages move fast
  - Commits every day
  - Compile <1 minut
- Avoid recompiling unchanged things

Uniform interface & One-Click CI

- NID skeleton testenv
  - Use on your local laptop for development
- Build / Testenv interface is standardized & uniform
  - make build
  - make testenv-start testenv-test testenv-stop
- Gitlab CI config wraps Make targets
  - No extra action required to enable CI testing
- Test against multiple versions of NSO in parallel

vrnetlab

- netsim is good for simple use
  - ”Free” – automatically built with NSO in Docker
- No real operational state on netsim – you have to mock
  - Configuring BGP neighbor does not result in any state
- To get closer to real device, use virtual router
- vrnetlab ”plugs right in” – get a virtual router instead of netsim
  - VM(s) running inside of container

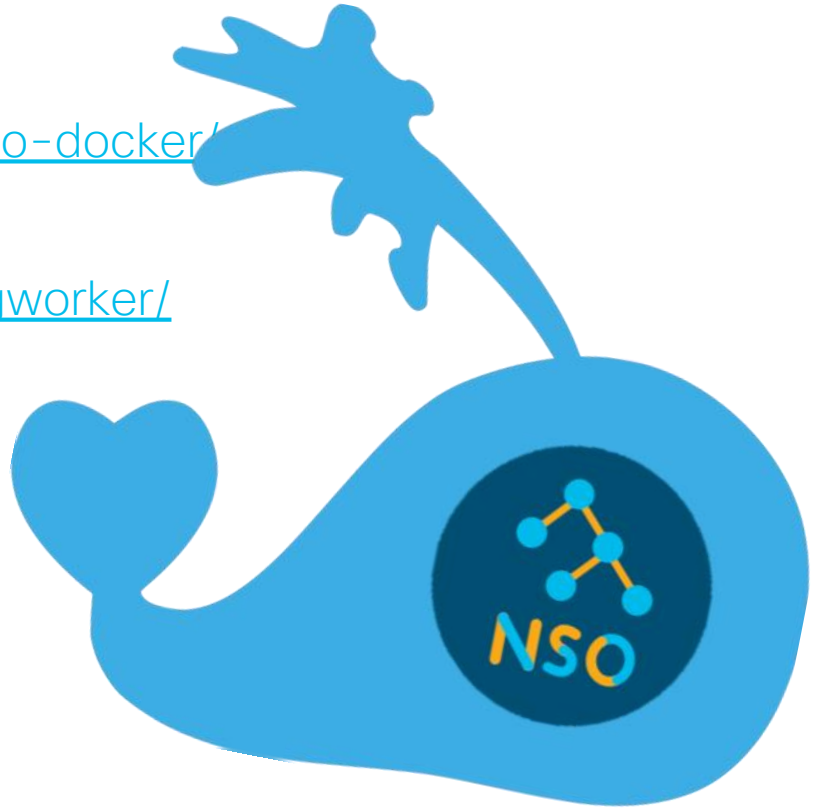
NSO in Docker on Linux on Mac

- Most (all?) production NSOs run on Linux
  - Your production system is likely Linux
- NSO in Docker runs on Linux
  - Same code, same build output – works everywhere
- Docker on Mac OS X actually runs on Linux
  - Using xhyve hypervisor on OS X to run Linux VM
  - Docker runs on Linux VM
- Run exact same NSO Docker images in prod as on Laptop!
  - Works everywhere!

That's all folks

- Main NSO in Docker repo:
  - https://gitlab.com/nso-developer/nso-docker/
- Bgworker
  - https://gitlab.com/nso-developer/bgworker/

- Twitter: @plajjan
- Email: krlarsso@cisco.com




CISCO