HPC Container Conformance

HPC3: Provide guidance on how to build and annotate HPC containers

Christian Kniep, 2023-02-05, FOSDEM'23

Christian Kniep
QNIB Solutions
Berlin area, Germany
info@qnib.org

Conformance What!?

What are we trying to achieve?

Guidance!

- Collect ways of building container images for HPC use cases
- Derive best-practices on how to build and annotate a container
- Use best-practices and annotation and take the SystemAdmin perspective

Expectation Management in terms of Portability/Performance

- Container images might be specific to a system or generic; how to we guide folks what to expect?

Application we start with: GROMACS, PyTorch (, WRF?)

Conformance What!?

What are we NOT trying to achieve?

- We are NOT going to boil the ocean in that we try to make everything work!
- Allow for generic images and also optimised images that only run in specific environments. Annotations will make the expectations clear
- We are going to focus on OCI image. Most likely build with a Dockerfile.
- Dockerfile might be derived with another artefact: spack.env / HPCCM recipe

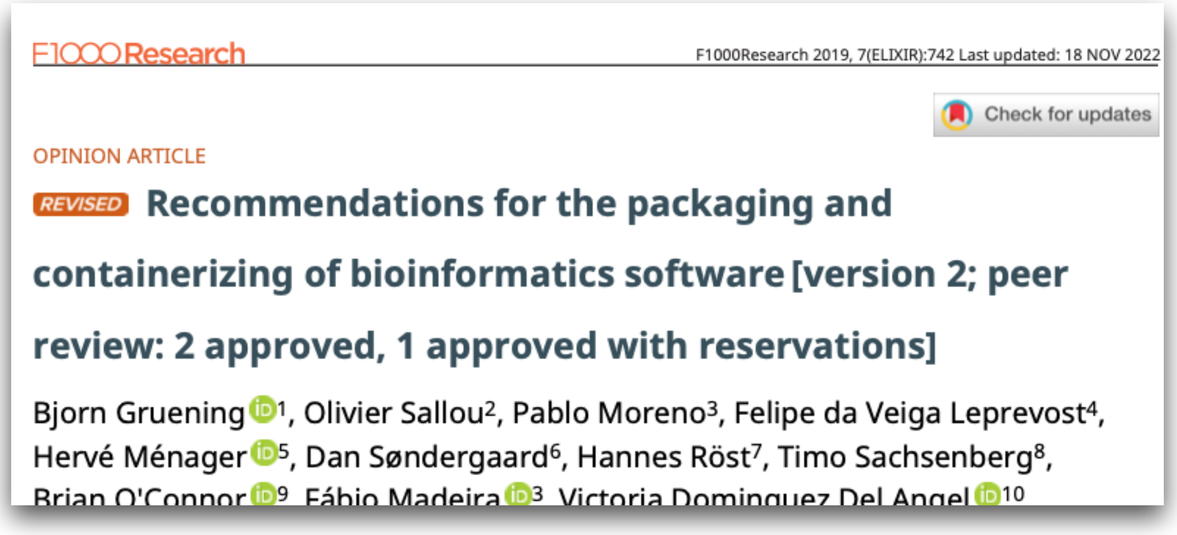
Biocontainer Community

Biocontainer Paper

Recommendations for the packaging and containerising of bioinformatics software

https://f1000research.com/articles/7-742/v2

1. A package first
2. One tool, one container
3. Tool and container versions should be explicit
4. Avoid using ENTRYPOINT
5. Reduce size as much as possible
6. Keep data outside of the container
7. Add functional testing logic
8. Check the license of the software
9. Make your package discoverable
10. Provide reproducible and documented builds
11. Provide helpful usage message



F1000Research F1000Research 2019, 7(ELIXIR):742 Last updated: 18 NOV 2022

Check for updates

OPINION ARTICLE

REVISED **Recommendations for the packaging and containerizing of bioinformatics software [version 2; peer review: 2 approved, 1 approved with reservations]**

Bjorn Gruening[1], Olivier Sallou[2], Pablo Moreno[3], Felipe da Veiga Leprevost[4], Hervé Ménager[5], Dan Søndergaard[6], Hannes Röst[7], Timo Sachsenberg[8], Brian O'Connor[9], Fábio Madeira[3], Victoria Dominguez Del Angel[10]

Expected Image Behaviour

Expected Image Bahavior

Login Container vs. Application Container

For HPC containers we expect to be dropped into a shell (most likely bash)

```
docker run -ti -v $(pwd):/data quay.io/cqnib/gromacs-2021.5_gcc-7.3.1:aarch64
bash-4.2#
```

The look and feel should be similar to logging into a compute node. The environment is prepared to have the application already at your fingertips.

```
$ docker run -ti -v $(pwd):/data -w /data \
     quay.io/cqnib/gromacs-2021.5_gcc-7.3.1:aarch64 gmx  mdrun -s benchRIB.tpr -resethway
                    :-) GROMACS - gmx mdrun, 2021.5-spack (-:
Using 1 MPI thread
Using 8 OpenMP threads
starting mdrun 'Protein'
10000 steps,     40.0 ps.
```

Expected Image Behavior

Login Container Image - ENTRYPOINT

The ENTRYPOINT **should be avoided or at least as small as possible** and setup the environment to tun the application.

- Source a bash profile to make the application and libraries available
- Apart from that as little runtime decisions as possible (if possible)
  (E.g. the upstream gromacs image has a gmx-choser to pick the best gmx binary - IMHO discouraged, even though it makes it portable)

Annotations

Annotations Base Ideas

Container Annotations

What for?

Annotations will serves two purposes

1. Describe the image: SysAdmins and end users know what to expect
   A. What user-land is provided by the image itself?
   B. In which ways can the image be tweaked to make the most out of the execution environment (CPU μArch, GPU, MPI)?
   C. Configure hooks, runtimes to tweak the container correctly
   D. Provide a smoke test to fail fast: “Container will SEGFAULT!”
2. Inform end users what to look out for an execution environment
   A. Look out for mpich variants with ABI version xyz…

Container Annotations

Mandatory Annotations / Optional once

The following slides just list groups of annotations, but at the end we need to define:

A. Mandatory annotations (which CPU architecture the container is compiled)
   -> Without those the container is not considered ‘HPC Container Compliant’

B. Optional annotations (CUDA version, complete SBOM)
   -> depending on how far you want to go

Annotation Groups

Hardware Annotations

CPU/GPU/...

Information about what the application in the containers user-land is compiled for.

- Will the application segfault due to architecture mismatch (beyond the platform specification ARM/x86)?
- What CUDA version and GPU architecture is the application build against?

org.supercontainers.hardware.cpu.optimized.mode	architecture, genericMicro, microarchitecture
org.supercontainers.hardware.cpu.optimized.version	x86_64 / x86_64_v4 / skylake / skylake_avx512
org.supercontainers.hardware.gpu.nvidia.driver.version	346.34
org.supercontainers.hardware.gpu.nvidia.cuda.version	12.1
org.supercontainers.hardware.gpu.nvidia.architecture	sm_35 (kepler), sm_86 (ampere)

MPI/Interconnect Annotations

Implementation/Framework/ABIs

Information about what the user-land is compiled for and what methods to tweak the container is the container designed for?

org.supercontainers.mpi.implementation	(openmpi,mpich,threadmpi)
org.supercontainers.communication.framework	(ucx, libfabrics)
org.supercontainers.openmpi.version	1.16.1
org.supercontainers.libfabric.abi.version	1.6
org.supercontainers.mpi.portability.optimization	stock, cray-xc-cnl10
org.supercontainers.mpi.portability.mode	mpi_replace, libfabric_inject, ucx_replace

System Annotations

What can the user expect

Scripting Environment: What does the container carry to support scripts?

org.supercontainers.libc.implementation	glibc,musl
org.supercontainers.glibc.version	2.35
org.supercontainers.python.version	3.10
org.supercontainers.shell.implementation	bash,sh,zsh
org.supercontainers.tools.includes	jq,wget,awscli
org.supercontainers.path.extra	/usr/local/bin (empty dir already in PATH)
org.supercontainers.kernel.version	5.1

What is expected from the host system

org.supercontainers.host.kernel.version.min	5.1
org.supercontainers.host.kernel.modules.expectation	user-namespaces

Documentation Annotations

Further information

Hello-world example as minimalistic as possible

Benchmark how-to with meaningful, representative result

How to use the container?

org.supercontainers.docs.quickstart.link	https://external.website.org/how-to-gromacs
org.supercontainers.docs.quickstart.base64	base64-encoded-markdown
org.supercontainers.docs.benchmark.link	https://external.website.org/how-to-bench-gromacs
org.supercontainers.docs.benchmark.base64	base64-encoded-markdown

How to reproduce/tweak the container build

org.supercontainers.docs.build.dockerfile	base64-encoded-dockerfile
org.supercontainers.docs.build.spack.env	base64-encoded-spack.env
org.supercontainers.docs.build.quickstart.link	https://external.website.org/how-to-build-gromacs
org.supercontainers.docs.build.quickstart.base64	base64-encoded-markdown

How to annotate?

Layered Approach

Annotations might be added in multiple stages

- The **base image** might provide some basic annotations about the
  - Operating system, tools already installed, libraries, etc.
- While **building a subsequent image** new annotations can be made:
  - Application version, additional dependencies
- **After an image** is build we might annotate more information
  - Using tools like crane
  - Collect annotation of image URIs (gromacs/gromacs:2021.5) without changing/republishing the image

Build Tools

Ideally tools like Spack/Easybuild/HPCCM have this build in

- HPCCM already provides a way to annotate the resulting image:
  `Stage0 += openmpi(version='3.1.4', annotate=True)`
- Spack might add annotations in the resulting image (make it Todds' problem)
- EasyBuild and other might do the same

Benefit

- By offloading (basic) annotations to build tools would make it easy to get annotations in, w/o the user even thinking about it.

External Curation of Annotations

Without access/control over images, we might just collect them

A curated list of HPC images can annotate without changing the image.
E.g. using MetaHub Collections:

```yaml
manifests:
  - name: gromacs/mpich
    tag: 2021.5
    manifests:
      - image: quay.io/cqnib/gromacs/gcc-7.3.1/2021.5/mpich:x86_64_v4
        platform:
          os: linux
          arch: amd64
        annotations:
          org.supercontainers.mpi.provider: mpich
          org.supercontainers.mpich.abi.version: 12.0
          org.supercontainers.mpi.portability.optimized: stock
          org.supercontainers.mpi.portability.mode: mpich_replace
          org.supercontainers.mpich.version: 3.4
```

System Fingerprint

System Fingerprint

archspec++

To match the annotations within the manifest and image index, we need to create a fingerprint of the system. Some ideas:

1. Hardware: CPU (archspec), GPU, Interconnect
2. OS: Kernel ABI
3. Software: glibc, maybe something like PMIx version?
4. Runtimes: What runtimes are installed, how are they configured?

Why do we need that again?

aka ReCAP

Discover the right Image

I want to run GROMACS 2021.5 on system XYZ!

Use the System Fingerprint to find a match within the annotations of an Image.

1. On g5.4xlarge / Amazonlinux2:
    a. HW: zen2 (x86_64_v3),
    b. SW: RHEL7 based con
2. Single node job: threadMP

```
manifests:
  - name: gromacs/tmpi
    tag: 2021.5
    manifests:
      - image: quay.io/cqnib/gromacs-2021.5_gcc-7.3.1:x86_64_v3-cuda-tmpi
        platform:
          os: linux
          arch: amd64
        annotations:
          org.supercontainers.hardware.cpu.optimization.mode: generic
          org.supercontainers.hardware.cpu.optimization.version: x86_64_v3
          org.supercontainers.mpi.provider: threadmpi
          org.supercontainers.cuda.version: 11.4
```

Fail fast when Scheduling

aka ‘before [sic!] you download 35GB of layers’

Use SystemFingerprint + annotations to fail fast.

1. On g5.4xlarge: zen2 (x86_64_v3)
2. Container Annotations:

org.supercontainers.hardware.cpu.optimized.mode	genericMicro
org.supercontainers.hardware.cpu.optimized.version	x86_64_v4

=> container will segfault (EC 132)

Help tweak the container

Inform hooks during the container lifecycle

Use SystemFingerprint + annotations to fail fast.

1. Container Annotations:

org.supercontainers.mpi.implementation	openmpi
org.supercontainers.communication.framework	libfabrics
org.supercontainers.openmpi.version	1.16.1
org.supercontainers.libfabric.abi.version	1.6
org.supercontainers.mpi.portability.optimization	stock
org.supercontainers.mpi.portability.mode	libfabric_inject

=> Inject local libfabrics provider into container

Thanks!

If you have questions or need consulting,
please reach out.

Christian Kniep
QNIB Solutions
Berlin area, Germany
info@qnib.org