Discussion: MetaHub

‚Container in HPC Workshop 101‘

Christian Kniep, NeIC’22

Goal

What do we want to achieve?

1. `module load` is setup locally and is executed at RUNTIME
2. Since a container (ideally) only has a single stack/target in them, we need to do it differently
3. MetaHub might help

MetaHub Registry

A smart Registry Proxy

Christian Kniep, ISC'22

Container Performance

Picking the right binary…

Without Containers

Picking the right binary for a given HW architecture usually comes down to:

```
$ module load gromacs:2021.5
```

It's set up and maintained by the System Admins and it's a **RUNTIME** decision made based on the current system.

With Containers

One way of doing it is to create a fat container with the same RUNTIME decision made within the container.
Copying the *module load* concept over to container land.

gromacs/gromacs:2021.5

entrypoint.sh

```
$ docker run -ti gromacs/gromacs:2021.5 cat /gromacs/bin/gmx
#!/bin/bash
FLAGS=`cat /proc/cpuinfo | grep ^flags | head -1`
ARCH="SSE2"
if echo $FLAGS | grep " avx512f " > /dev/null && test -d /gromacs/bin.AVX_512 \
   && echo `/gromacs/bin.AVX_512/identifyavx512fmaunits` | grep "2" > /dev/null; then
    ARCH="AVX_512"
elif echo $FLAGS | grep " avx2 " > /dev/null && test -d /gromacs/bin.AVX2_256; then
    ARCH="AVX2_256"
elif echo $FLAGS | grep " avx " > /dev/null && test -d /gromacs/bin.AVX_256; then
    ARCH="AVX_256"

/gromacs/bin.${ARCH}/gmx $@
```

Container Performance

Picking the right ~~binary~~ **container image**…

With Containers #2

Picking the right container image when submitting the job.

```
$ srun -N2 sarus run --mpi qnib/gromacs-2021.5_gcc-7.3.1:skylake_avx512 benchMEM.tpr
$ srun -N2 sarus run --mpi qnib/gromacs-2021.5_gcc-7.3.1:zen3 benchMEM.tpr
```

This implies that the submitter has clear expectations (/or constraints) regarding the hardware which is going to be used.

Putting bandage on the problem

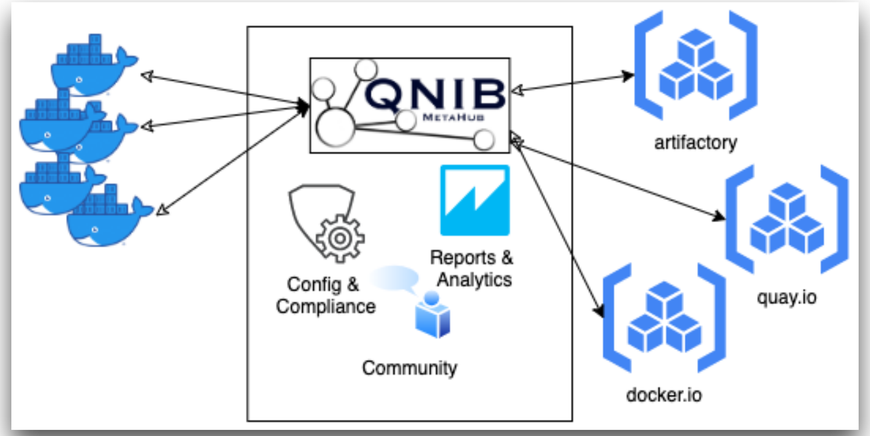
Make the decision at a different level:

- **RUNTIME**: on each node just before execution (configured for each runtime/node),
- **SCHEDULER**: when assigning the task (configured for each scheduler)

Container Performance

Pushing **module load** to the registry

Make the registry handle module load




```
$ docker login metahub-registry.org -u cascadelake
Login Succeeded
$ docker run metahub-registry.org/qnib/featuretest:latest
>> This container is optimized for: arch:cascadelake
```

```
$ docker login metahub-registry.org -u zen3
Login Succeeded
$ docker run metahub-registry.org/qnib/featuretest:latest
>> This container is optimized for: arch:zen3
```

```yaml
manifests:
  - name: gromacs
    tag: 2021.5
    manifests:
      - image: qnib/gromacs-2021.5_gcc-7.3.1:x86_64_v2
      - image: qnib/gromacs-2021.5_gcc-7.3.1:x86_64_v4
        platform:
          features: [arch:cascadelake]
      - image: qnib/gromacs-2021.5_gcc-11.3.0:zen3
        platform:
          features: [arch:zen3]
```

Users are associated with profiles. Profiles define filters…

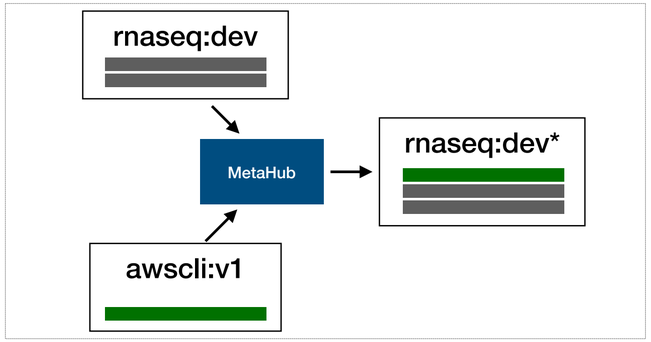
```yaml
profile:
  name: library/cascadelake:latest
  actions:
    - name: manifest-filter
      event: post-manifest-list-get
      args:
        select: .manifests .[] .platform .features
        value: arch:cascadelake
```

Dynamic Layer Addition

There is more you can do with this concept

Make the registry add layers dynamically

```
$ docker run metahub-registry.org/qnib/featuretest:latest aws –version

59bf1c3509f3: Pull complete

c1ac79912c87: Pull complete
exec: "aws": executable file not found in $PATH: unknown.
```



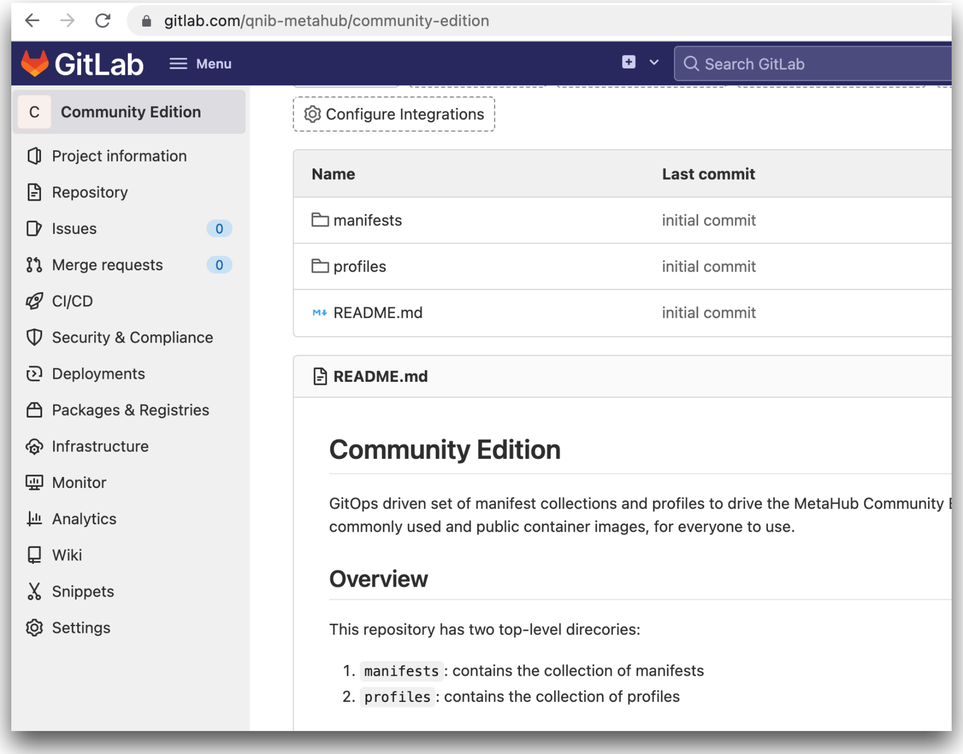

```
$ docker login metahub-registry.org --user aws/cli
Login Succeeded
$ docker run metahub-registry.org/qnib/featuretest:latest aws –version

59bf1c3509f3: Already exists

c1ac79912c87: Already exists

df9b9388f04a: Pull complete

d1ef575b3e16: Pull complete

03e1d868cf49: Pull complete
aws-cli/1.24.0 Python/3.9.7 Linux/5.10.109-104.500.amzn2.x86_64 botocore/1.26.0
```

```yaml
profile:
  name: aws/cli:latest
  actions:
    - name: manifest-add
      event: post-manifest-get
      args:
        layerRepository: qnib/layer-awscli
        # That should become a RegEx
        repository: qnib/featuretest
```

MetaHub Community Edition

Build a set of HPC manifest collection for the benefit of all

Inspired by nf-core, Spack, EasyBuild. Curate a collection for the HPC (and adjacent) communities, by the communities.



1. Public manifest collections for commonly used tools and applications (GROMACS, OpenFoam, etc.)
2. Public profiles for common architectures, use-cases (CPU, GPU, configuration)
3. Public profiles for common dynamic layers (entry points, awscli, …)

Conclusion

What did we learn

1. Preparing all possible permutations for targets/configuration
2. use (automatic) logins to have MetaHub pick the right container