EasyBuild @ HPC2N

- History
- Today
- A little bit of details
- Future

History

- EasyBuild was first taken into production 2016
  - On our, then new, cluster Kebnekaise
    - Two different CPU archs (broadwell and KNL) + K80 GPUs
    - Later (2018) extended with skylake and V100 GPUs
  - And on our older cluster Abisko
    - AMD Bulldozer/Piledriver cores
- We went for HMNS from the start
  - Reduce the number of modules directly visible
  - Make sure users can’t mix and match incompatible modules.

History

- Originally we placed the software stack on our Lustre file system
  - Worked well in the beginning, then users started using it heavily
- Builds where done by staff using their own account
  - Rather quickly caused permission problems
- Building directly on the nodes caused lots of interference from OS installed develop packages
  - And they tend to get updated, sometimes causing more problems

Today

- Software stack is distributed via CVMFS (started in June 2019)
  - Much better responsiveness to users jobs compared to Lustre
- Builds are done with a specific user
  - No more permission problems
- Builds are isolated from the OS on the nodes using a singularity container
  - We get a software stack that is as unaffected by OS packages as we can possibly make it

Architechture

- Layout of CVMFS tree is based on base CPU architecture (amd64, aarch64, etc), OSdistro, and CPU hwspec
  - /cvmfs/ebsw.hpc2n.umu.se/amd64_ubuntu2204_zen3 for example
  - We can fully utilize EasyBuilds optarch when building
- Top level directory for software stack is ‘/hpc2n/eb’
  - ‘eb’ is a node specific link into the CVMFS tree
- There is also a ‘common’ directory with hooks, local easyconfigs and easyblocks etc

Building software

- The singularity container consists of a minimal set of OS packages
  - Includes Infiniband dev packages
- The actual builds are done against a NFS based master tree
  - Which is bind mounted over /cvmfs/ebsw.hpc2n.umu.se
- Bind mount /hpc2n so the running container points to the correct architecture tree
- On GPU enabled nodes Nvidia and CUDA driver libraries are bind mounted in
- The result is a clean environment for building the software stack

Building software

- We have an ‘eb_builder’ module to set things up
  - It has an ‘eb’ wrapper that determines which container to use depending on the build nodes OS distro and runs the container
    - We also bind mount our Slurm version (which is not an OS package) and anything that it depends on
- The runscript in the container is yet another wrapper script sitting outside of the container, which
  - sets up the Lmod environment
  - loads the right EasyBuild version
    - which is can be the development version if requested
  - builds the requested packages using the actual EasyBuild ‘eb’ command
  - updates the Lmod spider cache for that specific OS/arch combo if builds finish successfully

Using --job and the batch system

- We have simple submit files
  - targeted to the various CPU types
  - they load the eb_builder module
  - If --job is specified and --from-pr is used --tmpdir is set to a shared directory
    - and --disable-cleanup-tmpdir is added
      - To keep anything downloaded due to --from-pr around for the subsequent sub jobs

--job works by accident

- Due to pure luck --job works without changing anything
  - Since the ‘eb_builder’ environment is still active, the EasyBuild generated submit files will call our ‘eb’ wrapper for the container… and it all works out just fine.

What we have

- Since changing to CVMFS
  - 19466 modules in total since 2019-06-01
    - >10 builds/day, 365 days/year
    - 8 hwarch (with 5 GPU types)
    - 5 OS distro/version combos
    - 1014 different SW packages
      - ~1 per workday
    - 3580 different versions of those packages (35 of these are Easybuild)
  - 9388 currently active modules
    - 6 hwarch (with 4 GPU types)
    - 3 OS distro/version combos
    - 843 different packages
    - 2235 different versions of those packages (21 of these are EasyBuild)

Where can I find this?

- https://github.com/hpc2n/EasyBuilding_in_Singularity

Future plans

Since EESSI is now in production we have started to look into using it.

Initially just as another software stack, but possibly as the base for our own stack, building and installing things like VASP, MATLAB separately, but also things not yet available in EESSI.

The future is already here

Thanks to EESSI PR#371 (EESSI-extend module) there is really nothing to it (once it is actually merged).

- Enable EESSI host_injection
- Load EESSI-extend module
- Build software

EESSI host_injection

We use /cvmfs/eessi.hpc2n.umu.se/versions as the base for our site EESSI extensions

- /cvmfs/software.eessi.io/host_injections is a CVMFS variant symlink
- Add EESSI_HOST_INJECTIONS=/cvmfs/eessi.hpc2n.umu.se/versions to the cvmfs config for the software.eessi.io repository
- Now /cvmfs/software.eessi.io/host_injections points to our cvmfs - tree

EESSI-extend

- export EESSI_SITE_INSTALL=1
- Load the EESSI-extend module
- Build software
- Installpath is /cvmfs/software.eessi.io/host_injections/…

Due to our use of cvmfs for our site EESSI stack we use containters when building just as we do for our pure EasyBuild stack.

User/project specific extensions

Set EESSI_PROJECT/USER_INSTALL=/path...

- Load EESSI-extend module
- Build software

Install target will be the path from above

- or $HOME/eessi/versions/… if not set

The EESSI-extend module must be loaded to get the modulepath set

EESSI and HMNS

We use HierarchialMNS for our normal EasyBuild softwarestack, can we do that for EESSI too?

- EESSI uses EasyBuildMNS (flat naming scheme)
- But EasyBuild can do –module-only –installpath-modules /some-other-path-than-normal

EESSI and HMNS

- The upstream EESSI stack is read-only
- --module-only runs the sanity-check step
- We need to create HMNS modules even for the software pre-installed by EESSI

Magic (well almost)

Creating HMNS modules for the pre-installed EESSI software:

```
eb --module-only
  --module-naming-scheme HierarchicalMNS
  --installpath $EESSI_SOFTWARE_PATH
  --installpath-modules our-path-to/hmns-modules
  --locks-dir /tmp/ake_eessi_lock
  --stop module
  --easystack .../eessi-2023.06-eb-4.9.0-2022b.yml
  -r
```

Questions?