Sandia National Laboratories

Exceptional service in the national interest

DEVELOPING TRILINOS WITHIN CONTAINERS

Samuel E. Browne, Anderson Chauphan, Joseph R. Frye, Caleb L. Jackson, Justin M. LaPre

Trilinos User-Developer Group Meeting 11/02/2023

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

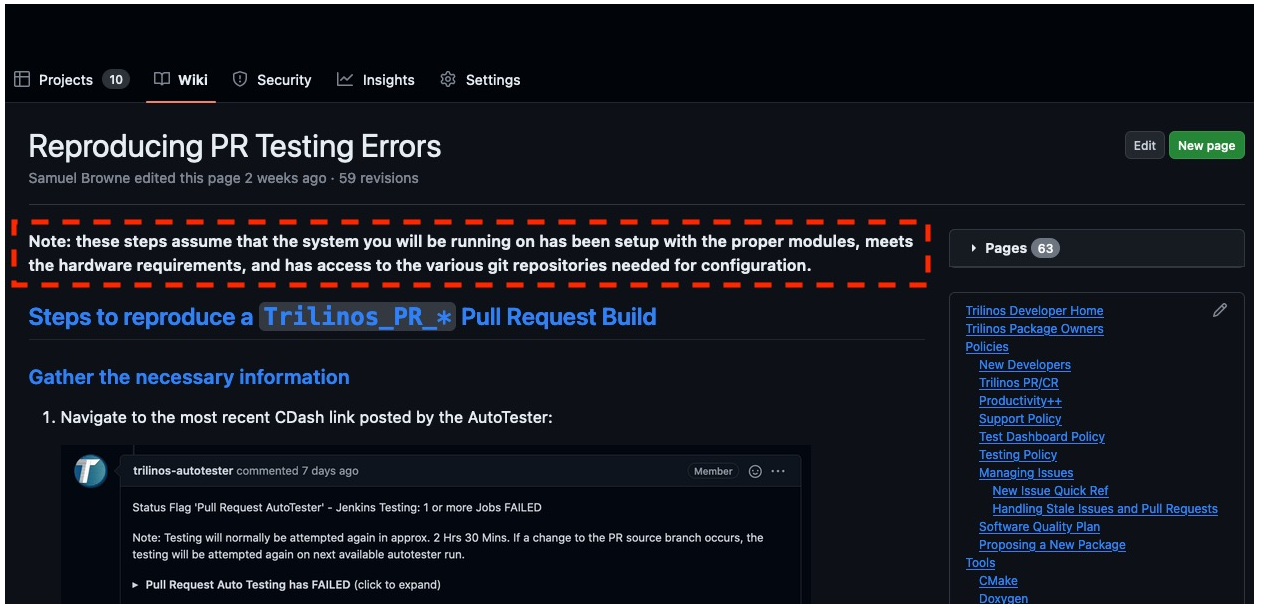
SAND2023-11770PE

U.S. DEPARTMENT OF ENERGY

NNSA

MANAGING TRILINOS BUILD ENVIRONMENTS

- Currently use a home-grown system called ‘GenConfig’
- Paired with third-party library modules that are maintained on our internal systems
- How do individual developers replicate pull request builds/tests?



EXTERNALLY-UNAVAILABLE REQUIREMENTS

- 5/7 GenConfig-related repositories
- TPLs on hardware
- Hardware itself (note that we have no control over this aspect)

How can we provide the configuration tool and a software environment (TPLs) that work together to external partners?

MAKING GENCONFIG AVAILABLE TO THE COMMUNITY

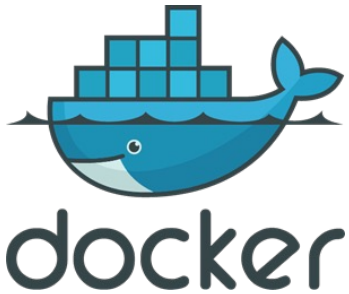
- Continuing to use GenConfig (and related tooling) will require open-sourcing to make available to the broader community
- This should not be an issue for most or all of the code developed as part of the GenConfig tool
- May have some time delays, but doable

MAKING TPLS AVAILABLE TO THE COMMUNITY

- Currently a team within Sandia deploys third-party libraries, compilers, and MPIs to select systems that are used for automated testing
- Unreleasable to external partners for technical reasons
- Also unavailable to internal systems outside the scope of the support agreement

CONTAINERS AS A MECHANISM FOR DISTRIBUTING TPLS

- Containers handily solve the third-party software problem
- There are limitations of reproducing novel software environments (e.g. DOE ATS systems), but these environments are not currently in pull request testing, and are outside the scope of this effort
- Containers *greatly* simplify the act of setting up build environments
  - Complexity is still there, but is largely handled within the Dockerfile that describes how to build the container image
  - Complexity is removed from user workflow



HOW TO RUN A CONTAINER

```
# Authenticate if needed
docker login your-registry.yourdomain.com

# Pull the image that you want to use from the registry
docker pull your-registry.yourdomain.com/yourimage

# Run the image
#   Remove container once it exits
#   Run interactively and attach tty
#   Run bash as the container entrypoint
docker run --rm –it --entrypoint bash yourimage
```

EXAMPLE

```
[sebrown@triloamd01 ~]$ podman run --rm -it --entrypoint bash registry-ex.sandia.gov/trilinos-project/t
rilinos-containers/rhel8/trilinos-test-env:gcc-8.5.0
[root@b73633b60c9d /]# module list
Currently Loaded Modulefiles:
 1) gcc/8.5.0      7) git/2.40.0           13) netlib-lapack/3.11.0    19) scotch/7.0.3
 2) boost/1.82.0   8) hdf5/1.14.1-2        14) ninja/1.11.1            20) suite-sparse/5.13.0
 3) ccache/4.8     9) metis/5.1.0          15) openmpi/4.1.5           21) superlu-dist/5.4.0
 4) cgns/4.3.0    10) netcdf-c/4.9.2       16) parallel-netcdf/1.12.3  22) superlu/5.3.0
 5) cmake/3.26.3  11) netcdf-cxx/4.2       17) parmetis/4.0.3          23) yaml-cpp/0.6.2
 6) cuda/11.8.0   12) netcdf-fortran/4.6.0  18) python/3.10.10         24) zlib/1.2.13
[root@b73633b60c9d /]# type cmake
cmake is /spack-installs/cmake/3.26.3/gcc/8.5.0/base/l4oiivv/bin/cmake
[root@b73633b60c9d /]# type ncdump
ncdump is /spack-installs/netcdf-c/4.9.2/gcc/8.5.0/openmpi/4.1.5/td6fe5a/bin/ncdump
[root@b73633b60c9d /]#
```

Note that all TPLs are “Just There”, with no module load, source, etc. All (ish) of the complexity is baked into the container recipe itself.

Can now clone Trilinos, or any other code you wish to develop.

HOW TO MOUNT YOUR LOCAL CODE INTO A CONTAINER

```
docker run --rm -it --entrypoint bash \
--mount type=bind,src=/path/on/your/machine,dst=/path/in/container \
yourimage
```

Allows you to get data in/out of container through the mounted directory

Depends on host filesystem (e.g. can have some issues when mounting a Windows directory into a Linux container)

Extension: It is possible to point VSCode at a container image and have it boot said image, mount your code project for you, and then place your terminal in the running container.

CONTAINERS HELP ENSURE CONSISTENCY

Cons

- There is overhead in learning to use containerized development environments

Pros

- Near-perfect reproducibility between container runs
- Ability to easily share development environments between developers
- Anybody can create a new container on any machine with compatible architecture
- Can take container used for “validation” runs (PR testing) and run locally on developer machines

DISCUSSION