SAND2024-03997C

Sandia National Laboratories

Exceptional service in the national interest

THE CASE FOR CONTAINERIZATION WITH HPC SCIENTIFIC SOFTWARE

Samuel E. Browne

ASC S3C at NLIT 24, Seattle, WA

April 8-11, 2024

U.S. DEPARTMENT OF ENERGY

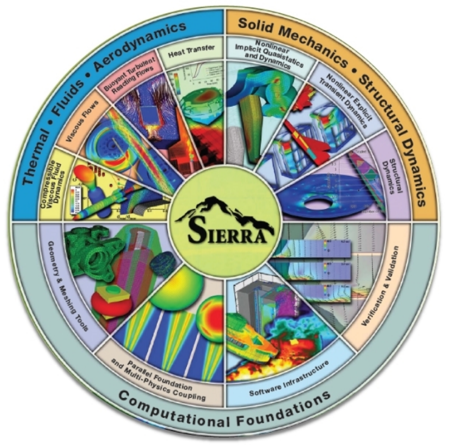
NNSA National Nuclear Security Administration

ACKNOWLEDGEMENTS

- SIERRA DevOps Team
  - Mark Hamilton, Jake Healy, Andrew Kimler, Justin Lamb, Matt Mosby, Tony Nguyen

- Trilinos Framework Team
  - Anderson Chauphan, Joe Frye, Justin LaPre

CODES BACKGROUND


Thermal • Fluids • Aerodynamics
Solid Mechanics • Structural Dynamics
Heat Transfer
Nonlinear Implicit Quasistatics and Dynamics
Nonlinear Explicit Transient Dynamics
Structural Dynamics
Verification & Validation
Software Infrastructure
Parallel Foundation and Multi-Physics Coupling
Geometry & Meshing Tools
Compressible Viscous Fluid Dynamics
Viscous Flows
Buoyant Turbulent Reacting Flows
SIERRA
Computational Foundations

SIERRA is Sandia's engineering mechanics code suite. It is a large C++ code under development for more than 30 years, and is widely used across many government HPC customers. It leverages the Trilinos software stack.

Trilinos is an object-oriented software framework for the solution of large-scale, complex multi-physics engineering and scientific problems. It is primarily developed at Sandia, but is open-source and hosted on GitHub, with many developers from the open-source community.

BACKGROUND

- I have been a member of the SIERRA DevOps team since 2016, primarily working on areas surrounding the build system
- I wrote a custom TPL builder for the SPARC team in 2018 to support the ATS2 ‘Sierra’ platform, which was then extended to all other HPCs supported by SPARC (ATS1, CTS1, Vanguard ‘Astra’, etc.)
- Since 2022, I have led the Trilinos Framework team which is responsible for the CI build/test infrastructure (including environment management)

INDUSTRY’S PREFERRED ENCAPSULATION METHOD

Containers are **packages of software that contain all of the necessary elements to run in any environment**. In this way, containers virtualize the operating system and run anywhere, from a private data center to the public cloud or even on a developer's personal laptop.
– Google Cloud

>53% of software development companies report using containers*

* Survey responses were all over the map, so the lowest found percentage was used

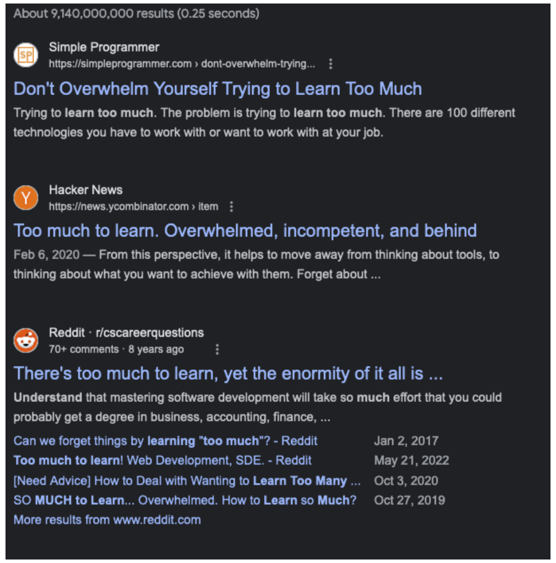
So why don't more HPC development teams use containers for environment management?

THE BIGGEST PROBLEM

- Codes are often developed by physicists/hard-science engineers rather than career software engineers, so workflows must be *as simple as is reasonably possible*, freeing the developers to focus on the complex physics

- The “too much to learn” issue:

Amusing note: of the top twelve Google results for this string, six are about software engineering



About 9,140,000,000 results (0.25 seconds)

Simple Programmer
https://simpleprogrammer.com › dont-overwhelm-trying...

Don't Overwhelm Yourself Trying to Learn Too Much

Trying to **learn too much**. The problem is trying to **learn too much**. There are 100 different technologies you have to work with or want to work with at your job.

Hacker News
https://news.ycombinator.com › item

Too much to learn. Overwhelmed, incompetent, and behind

Feb 6, 2020 — From this perspective, it helps to move away from thinking about tools, to thinking about what you want to achieve with them. Forget about ...

Reddit · r/cscareerquestions
70+ comments · 8 years ago

There's too much to learn, yet the enormity of it all is ...

Understand that mastering software development will take so **much** effort that you could probably get a degree in business, accounting, finance, ...

Can we forget things by **learning "too much"**? - Reddit	Jan 2, 2017
Too much to learn! Web Development, SDE. - Reddit	May 21, 2022
[Need Advice] How to Deal with Wanting to **Learn Too Many** ...	Oct 3, 2020
SO **MUCH to Learn**... Overwhelmed. How to **Learn** so **Much**?	Oct 27, 2019

More results from www.reddit.com

CHANGING WORKFLOWS

Current developer workflow

```
module load gcc
module load openmpi
module load hdf5
…
```

Proposed new workflow

```
podman run <containerid>
…
```

These are only the setup commands, not the entire workflow.

The setup may be simpler, but **what about other aspects of the workflow**?

HPC DEVELOPMENT REQUIREMENTS

- Scientific software development workflows have different requirements than “normal” software
  - Large code suites (e.g. SIERRA base code repository ~25GB)
  - LARGE test suites (e.g. SIERRA test repository ~155GB)
  - Deep and complex third-party library (TPL) dependency trees
    - No universal C++ package manager ala npm for JavaScript or pip for Python
  - MPI parallelism
  - GPU/other device accelerator support
  - ”Real” runtime environment is not a container (yet), so concerns about reproducibility

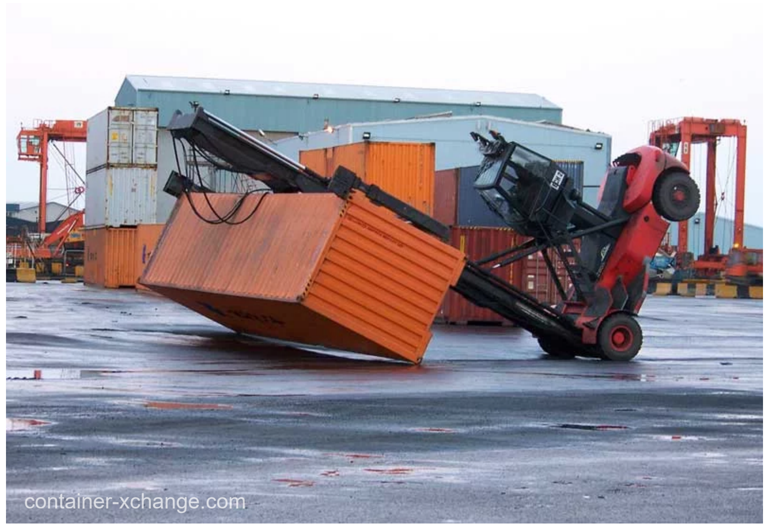
Are containers incapable of meeting any of the above requirements?

LARGE CODE/TESTS

LOCAL DISKS INTRODUCE COMPLEXITY

- The large code/tests issue from earlier appears!

- Cloning ~200GB of code each time a container is started is not feasible



Overweight dev container accident

- Enter mounting complexity

```
podman run --mount type=bind,src=/path/to/my/code,dst=/code <containerid>
```

TPL COMPLEXITY

CONTAINERS REDUCE NEEDED CONTEXT

Spack – a flexible package manager for HPCs

- Using Spack is demonstrably cheaper than maintaining builds of TPL stacks manually
- Using Spack is complicated, because Spack solves a **very complicated problem**
  - Example: Multiple parallel installs of different compiler-based software stacks

If we want to use Spack to solve our problem, but as simply as possible, we must reduce the complexity of the problem to be as simple as possible

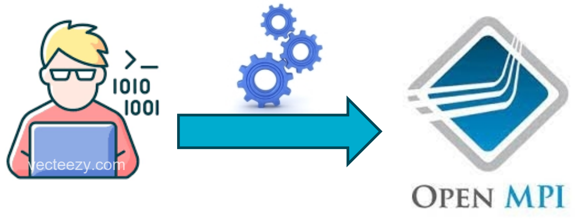
- Containers make using Spack at least an order of magnitude easier
- *Programming environment teams can now provide Spack for downstream developers*
  - They can also pre-build TPLs for developers who don't want/need to build them

MPI

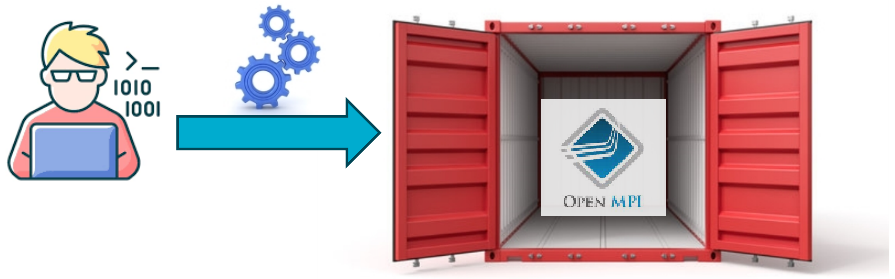
IS IT REALLY ANY DIFFERENT?

- Most HPC scientific software is NOT developed on HPC-style machines
- Developers have large-ish (~16-32 core) workstations that can run MPI software (albeit single-node)
- How does the MPI get there?

On-system



Containerized



This is the same from the user's perspective, and as with other TPL dependencies, much more robust than on-hardware builds

Note: In other scenarios with vendor-specialized MPIs, this may be increased complexity, but still has solutions

GPUS

IS IT REALLY ANY DIFFERENT?

- Key Concept: Most code development does not currently happen on machines with GPUs
  - However, this may change with the evolving hardware landscape

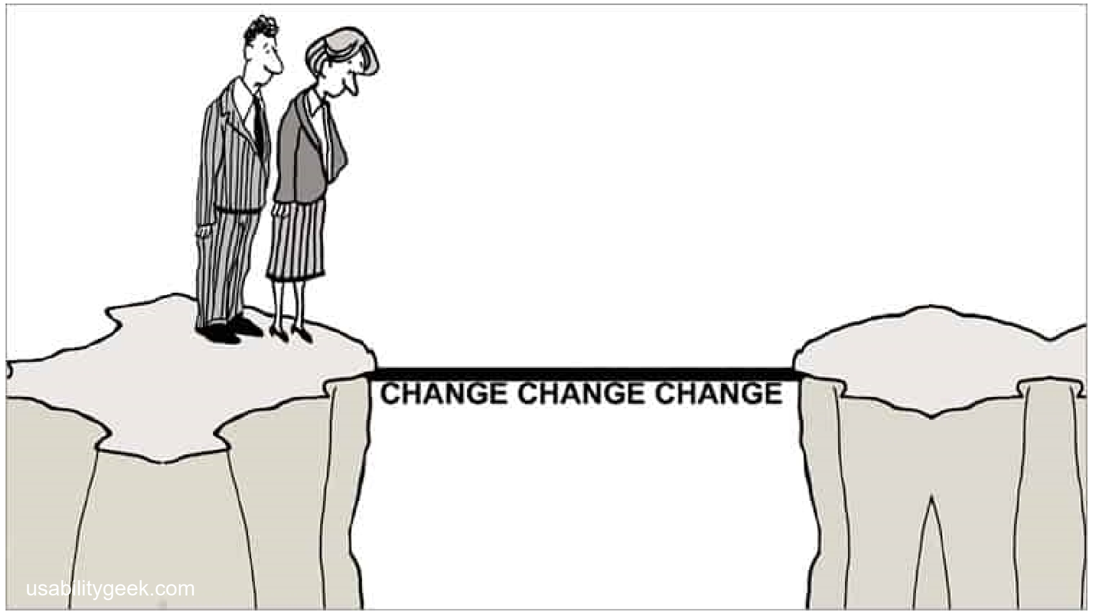
- GPU hardware/drivers **can** be accessed from within containers
  - See work with ECP/E4S and ParaTools testing large stacks of software on a continuous basis using containers and many different GPU architectures

Differences in CPU-only vs. GPU development workflows are likely, but technical paths forwards exist

RUNTIME ENVIRONMENT REPRODUCIBIL ITY

MY CUSTOMERS DON'T WANT TO USE A CONTAINER

- Nobody wants to change their workflows at all, ever, and this include shifting to using environments that are managed via containers instead of on-system environment modules



Taking a step back, is it a problem for development to use containerized environments if the end-user does not want to use one?

CASE STUDY: SIERRA DEVOPS CODE TESTING

- SIERRA has a set of Python-based tools that wrap building the code, as well as a specialized distributed test harness that can interface with HPC queues
- The team desired to move to GitLab CI to manage their testing for integration and coverage visibility reasons
  - Restriction: Can not run as GitLab user, must run as user who pushed
  - Jacamar or containers?
  - Manufactured a RHEL UBI container that contained what we knew to be our runtime dependencies and added it to testing ~02/2023

Concern: We are no longer testing in the environment that our users are using

CASE STUDY: SIERRA DEVOPS CODE TESTING

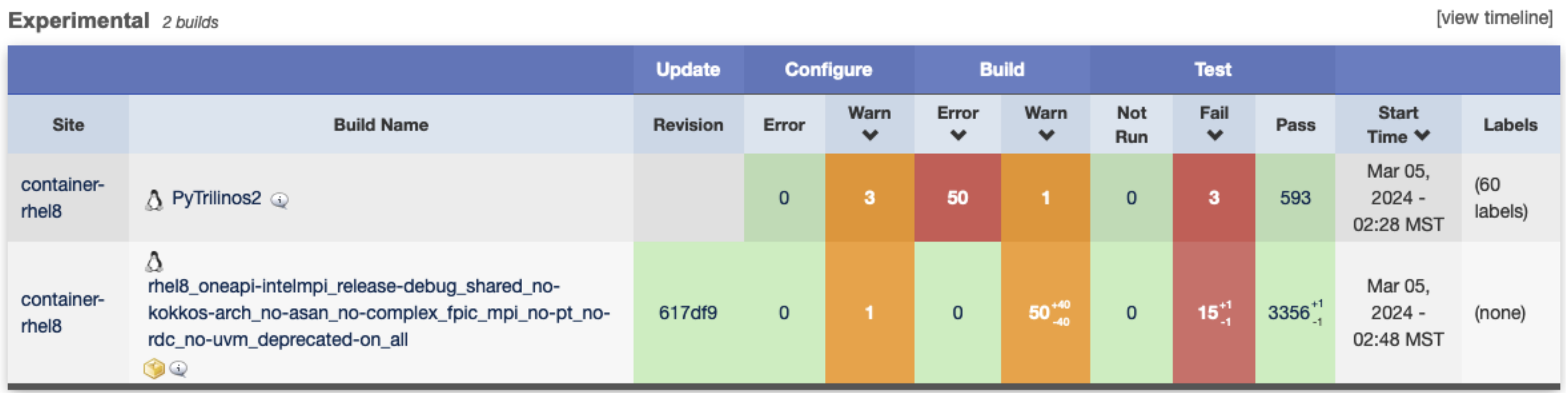
- Bugs in our code allowed through testing due to environment differences to-date: 0*

- It has repeatedly allowed us to distinguish issues with the runtime environment vs. our code
  - This allowed us to pass relevant debugging information back to the developers of those environments

- * Once, when attempting to upgrade the version of Git in the container, we exposed that our code would not work with the new Git, *prior to deploying that new Git to our customers!*

A NEW WAY TO MANAGE CI ENVIRONMENTS

- Contributors to the Trilinos project have historically had problems reproducing the environments represented by CI testing
- Containers will allow us to share testing environments
  - Externally AND internally!
  - Developers can help drive TPL selection/upgrade/configuration process
- Results already matching (and with increased stability) on-hardware testing

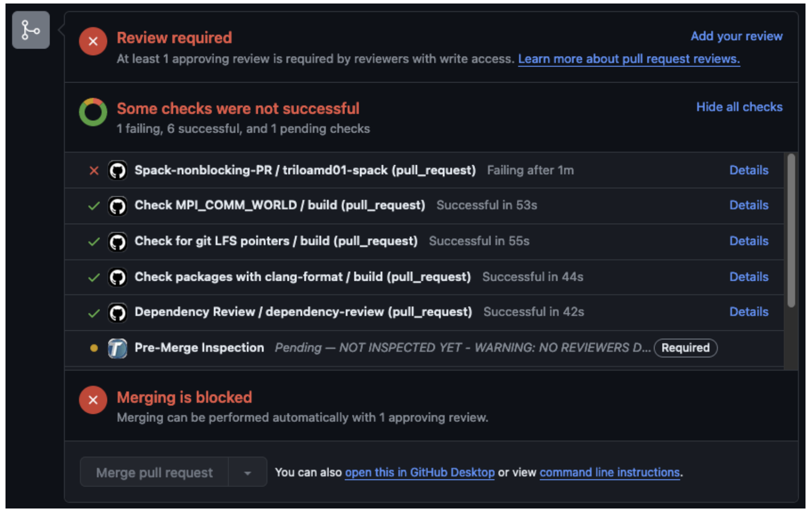


Experimental *2 builds* [view timeline]

		Update	Configure		Build		Test				
Site	Build Name	Revision	Error	Warn	Error	Warn	Not Run	Fail	Pass	Start Time	Labels
container-rhel8	PyTrilinos2		0	3	50	1	0	3	593	Mar 05, 2024 - 02:28 MST	(60 labels)
container-rhel8	rhel8_oneapi-intelmpi_release-debug_shared_no-kokkos-arch_no-asan_no-complex_fpic_mpi_no-pt_no-rdc_no-uvm_deprecated-on_all	617df9	0	1	0	50^{+40}_{-40}	0	15^{+1}_{-1}	3356^{+1}_{-1}	Mar 05, 2024 - 02:48 MST	(none)

A NEW WAY TO MANAGE CI ENVIRONMENTS – BETTER WORKFLOW

- As with SIERRA DevOps, Trilinos wants better integration with its CI processes and its repository manager (GitHub)
- Using containers will allow for secure use of self-hosted runners and GitHub Actions



Review required — Add your review

At least 1 approving review is required by reviewers with write access. Learn more about pull request reviews.

Some checks were not successful — Hide all checks

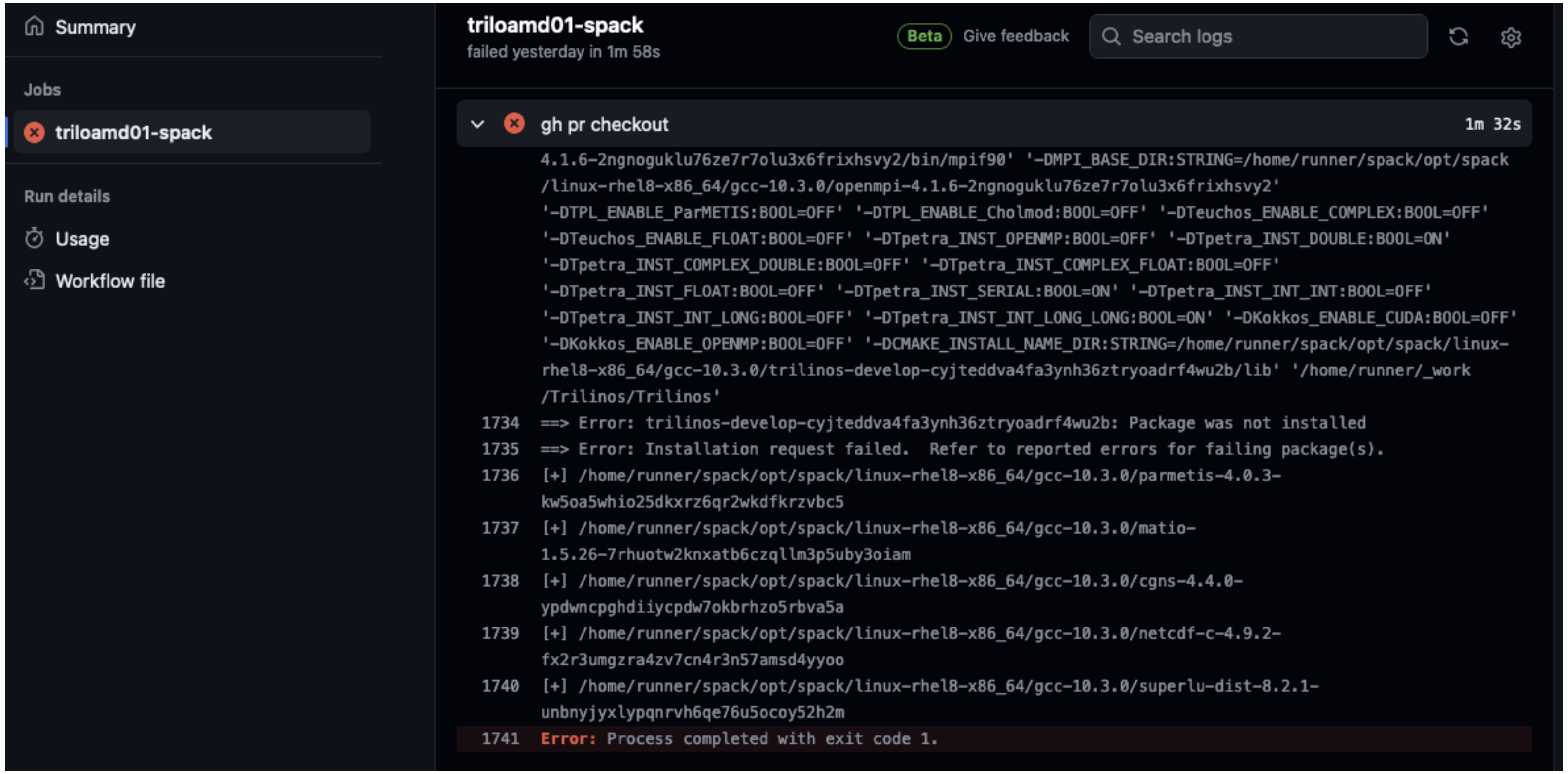
1 failing, 6 successful, and 1 pending checks

- ✗ Spack-nonblocking-PR / triloamd01-spack (pull_request) — Failing after 1m — Details
- ✓ Check MPI_COMM_WORLD / build (pull_request) — Successful in 53s — Details
- ✓ Check for git LFS pointers / build (pull_request) — Successful in 55s — Details
- ✓ Check packages with clang-format / build (pull_request) — Successful in 44s — Details
- ✓ Dependency Review / dependency-review (pull_request) — Successful in 42s — Details
- ● Pre-Merge Inspection — *Pending — NOT INSPECTED YET - WARNING: NO REVIEWERS D...* Required

Merging is blocked

Merging can be performed automatically with 1 approving review.

Merge pull request — You can also open this in GitHub Desktop or view command line instructions.



triloamd01-spack

failed yesterday in 1m 58s

gh pr checkout — 1m 32s

```
4.1.6-2ngnoguklu76ze7r7olu3x6frixhsvy2/bin/mpif90' '-DMPI_BASE_DIR:STRING=/home/runner/spack/opt/spack
/linux-rhel8-x86_64/gcc-10.3.0/openmpi-4.1.6-2ngnoguklu76ze7r7olu3x6frixhsvy2'
'-DTPL_ENABLE_ParMETIS:BOOL=OFF' '-DTPL_ENABLE_Cholmod:BOOL=OFF' '-DTeuchos_ENABLE_COMPLEX:BOOL=OFF'
'-DTeuchos_ENABLE_FLOAT:BOOL=OFF' '-DTpetra_INST_OPENMP:BOOL=OFF' '-DTpetra_INST_DOUBLE:BOOL=ON'
'-DTpetra_INST_COMPLEX_DOUBLE:BOOL=OFF' '-DTpetra_INST_COMPLEX_FLOAT:BOOL=OFF'
'-DTpetra_INST_FLOAT:BOOL=OFF' '-DTpetra_INST_SERIAL:BOOL=ON' '-DTpetra_INST_INT_INT:BOOL=OFF'
'-DTpetra_INST_INT_LONG:BOOL=OFF' '-DTpetra_INST_INT_LONG_LONG:BOOL=ON' '-DKokkos_ENABLE_CUDA:BOOL=OFF'
'-DKokkos_ENABLE_OPENMP:BOOL=OFF' '-DCMAKE_INSTALL_NAME_DIR:STRING=/home/runner/spack/opt/spack/linux-
rhel8-x86_64/gcc-10.3.0/trilinos-develop-cyjteddva4fa3ynh36ztryoadrf4wu2b/lib' '/home/runner/_work
/Trilinos/Trilinos'
==> Error: trilinos-develop-cyjteddva4fa3ynh36ztryoadrf4wu2b: Package was not installed
==> Error: Installation request failed.  Refer to reported errors for failing package(s).
[+] /home/runner/spack/opt/spack/linux-rhel8-x86_64/gcc-10.3.0/parmetis-4.0.3-
kw5oa5whio25dkxrz6qr2wkdfkrzvbc5
[+] /home/runner/spack/opt/spack/linux-rhel8-x86_64/gcc-10.3.0/matio-
1.5.26-7rhuotw2knxatb6czqllm3p5uby3oiam
[+] /home/runner/spack/opt/spack/linux-rhel8-x86_64/gcc-10.3.0/cgns-4.4.0-
ypdwncpghdiiycpdw7okbrhzo5rbva5a
[+] /home/runner/spack/opt/spack/linux-rhel8-x86_64/gcc-10.3.0/netcdf-c-4.9.2-
fx2r3umgzra4zv7cn4r3n57amsd4yyoo
[+] /home/runner/spack/opt/spack/linux-rhel8-x86_64/gcc-10.3.0/superlu-dist-8.2.1-
unbnyjyxlypqnrvh6qe76u5ocoy52h2m
Error: Process completed with exit code 1.
```

CONCLUSION: CONTAINERS SUIT THE TECHNICAL NEED

But remember, the real issue is that workflows need to be as simple as reasonably possible

- Can a containerized development workflow be as simple as what is done today?
  - MAYBE – there is always complexity involved with solving a complex problem
  - Can the complexity be shifted, or hidden where appropriate (without making it impossible to expose!) to simplify the development process for HPC scientific software?