Workflows in Environmental Research with Containers

26.03.2021 International Symposium on Grids and Clouds
Viktoria Pauw

Content of this talk

Environmental Computing at Leibniz Rechenzentrum (LRZ)

- LRZ is one of 3 large scientific computation center in Germany
- Located in Garching near Munich

- First Project: GeoKW - An Approach to simulating effects on groundwater by geothermal heating facilities in the urban area

- Second Project: CoCoReCS - Testing und Usability Support for SeisSol (Earthquake Simulator)

- In both projects we aim to improve workflows with Containers

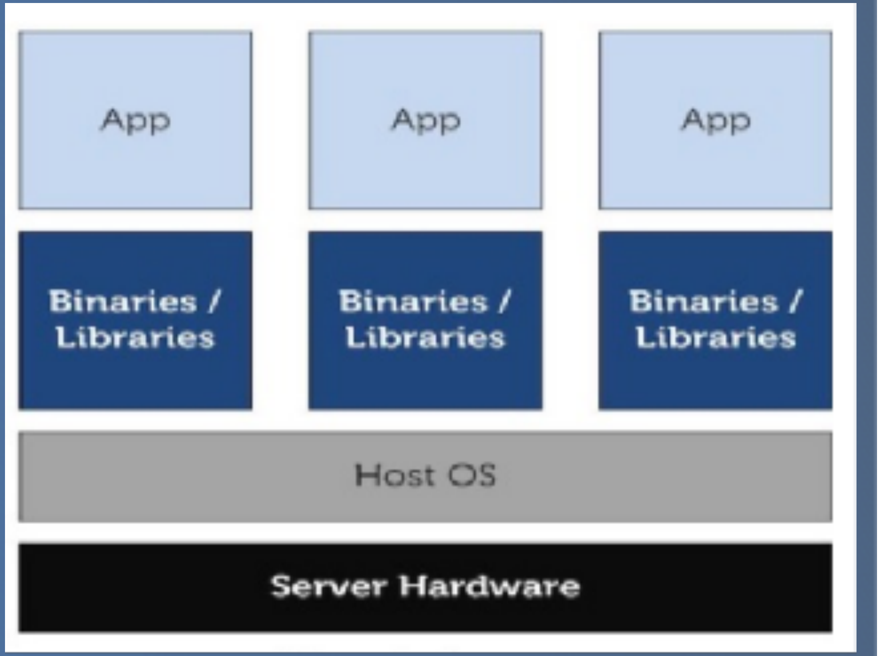
SuperMUC-NG

SuperMUC-NG consists of

- Intel Xeon Skylake
- 6,336 thin nodes with 48 cores and 96 GB memory
- 144 Fat nodes 48 cores and 768 GB memory per node
- 8 islands
- 19.5 PetaFLOP/s
- 100 Gbit/s OmniPath
- 15th place der TOP500

What are Containers?

- Method of OS-level virtualization
- Several OS' share a kernel
- No hardware emulation overhead like a Virtual Machine: uses the normal systems call interface
- Can be supplied via repository or tar-files
- Contains pre-build applications, dependencies, data…



Some Containerization Programs for Scientific Computing

Docker
- Not suitable for HPC (Runs a root daemon)
- De-facto standard format in industry

Singularity
- Growing eco-system
- uses setuid - not considered safe by all HPC admins
- Integration with Spack
- Can use overlayfs (avoids fixed bind mounts)
- SIF files can be signed and verified

Charliecloud
- Based on Docker
- Developed by LANL
- Small Code
- only user namespace

Example for Containers on HPC systems

- Use a Docker image with compiled newest code and dependencies
- Build a Charliecloud file:

  ch-builder2tar SeisSol-docker /dir/SeisSol-docker
- Copy the tar.gz file to the HPC system/Cloud
- Run appliance in your own namespace (no daemon)

Or:

- Build an image and upload to a registry (docker, singularity)

Container Workflow

On Local Machine

Dockefile → Container → Test → Package

UDSS

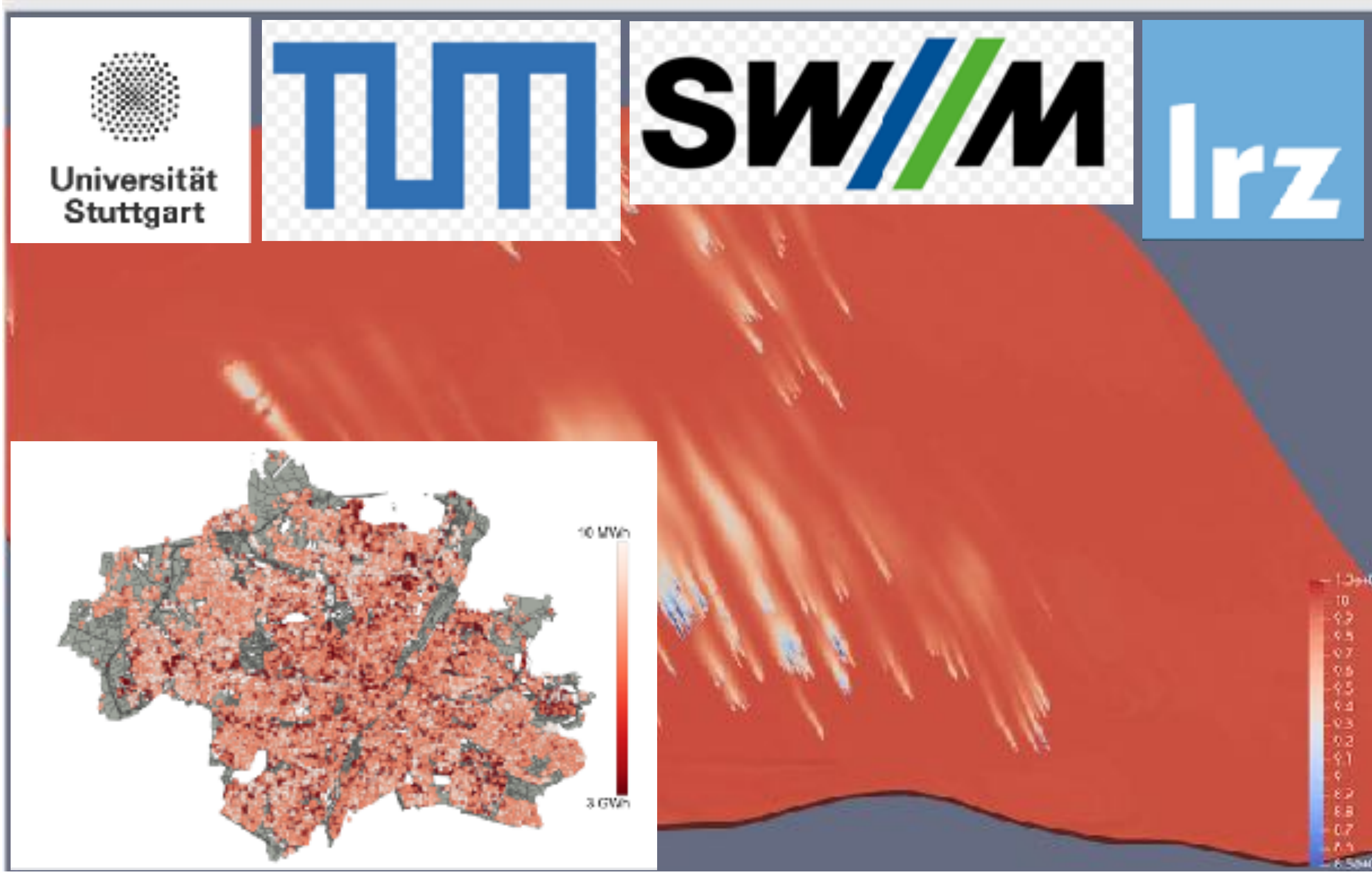
On User Machine/HPC system

Unpack → (Bind System Dirs → Rebuild →) Run

GEO.KW

- Calibration of Groundwater Flow in the City of Munich
- Impact of Geothermal Heat Input on Environment
- Optimizing Placement of Heat Pumps
- Computing Budget 30 Mio cpuh on SuperMUC-NG

Partners: TUM, Universität Stuttgart, Leibniz Rechenzentrum, Energy Supplier and Water Management Office of Munich



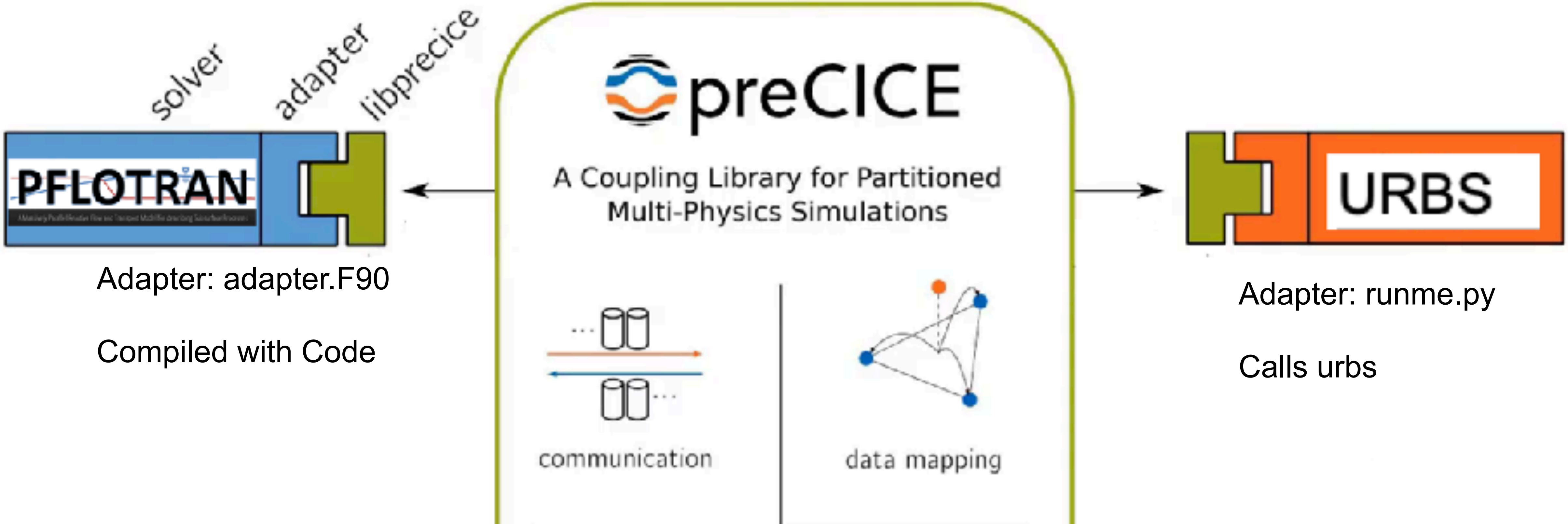
Optimising Well Locations for Geothermal Applications in Munich

- Plume length estimation
- Avoid negative interference by selecting favorable placement
- Strongly interacting locations are grouped together
- Urbs optimization runs in alternation with groundwater modellation

Set-Up

preCICE

- Pflotran + Urbs run separately and exchange temperatures and flow-rates via preCICE
- Need self-written Adapters that are compiled with the code + config files
- Need prerequisites like Parmetis (Mesh decomposition) and PETSc (Matrix operations)



Adapter: adapter.F90

Compiled with Code

Adapter: runme.py

Calls urbs

Complex software stack can be distributed with a Container

- preCICE v2.2.0
- Pflotran+Adapter
- MPI
- Python Environment
  - preCICE-python bindings…
- Urbs+Adapter
- Data on wells
- PEST
- Parmetis

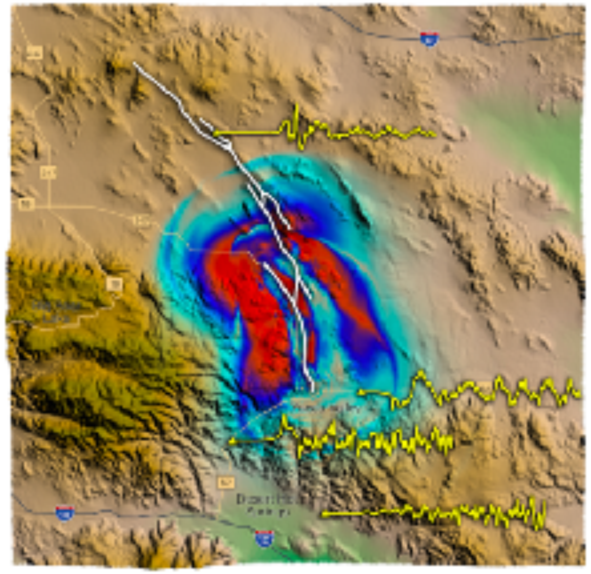
Benefit of containers

- Many users want to experiment with the workflow (on cloud machine, VM, HPC system)
- Containers can be used for sharing the complex software stack + dependencies + test data + config files
- Readily compiled and able to run on different machines (linux)

CoCoReCS - **Co**mmunity **Co**de for **Re**producible **C**omputational **S**eismology

Improving SeisSol Workflows and User experience with CI/CD

- Help developers build clean code -> Automatic testing
- Save manual tasks. -> Create build pipelines
- Reduce entry barrier for new users -> Ready to use images (containers)
- Regular parallel performance and convergence tests



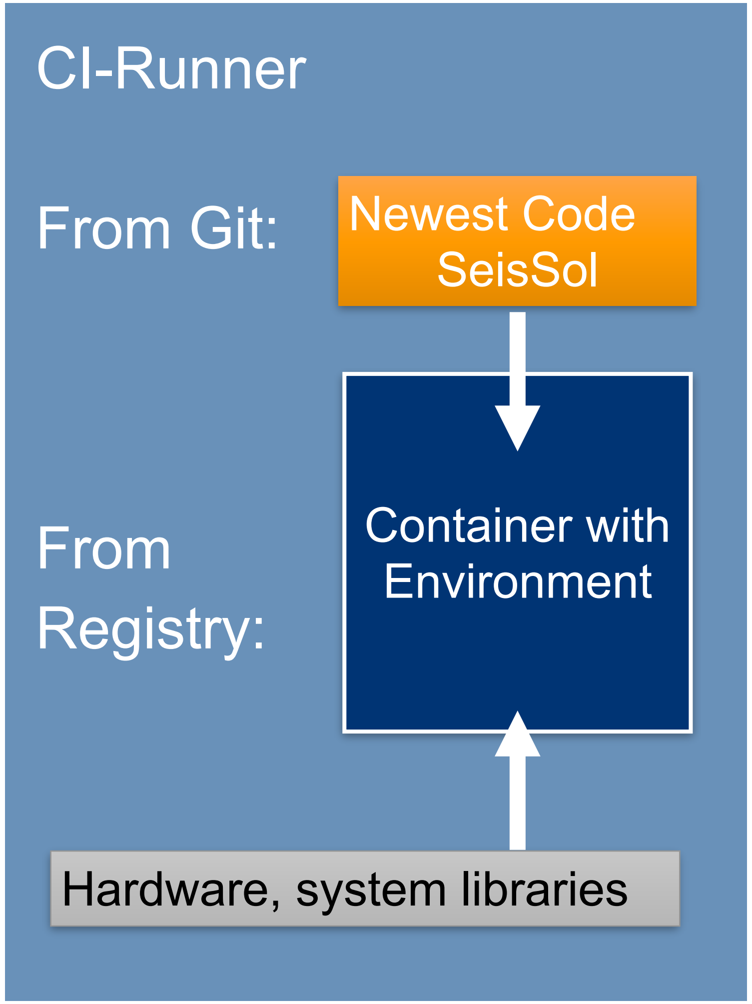
What is CI/CD?

„Agile development“

- CI - continuous integration
  - tested, cleanly buildable code even after small incremental changes
  - every change is build and tested *automatically*
  - warning, if a build/test fails due to new commit
  - avoid complex conflicts that may arise when merging after a longer time
- CD - continuous deployment/delivery
  - use the pipeline to build a deployable binary/image
  - supply each incremental improvement to production

- Containers can be both helpful tool and product of CI/CD

CI-Tests with Singularity Base Container

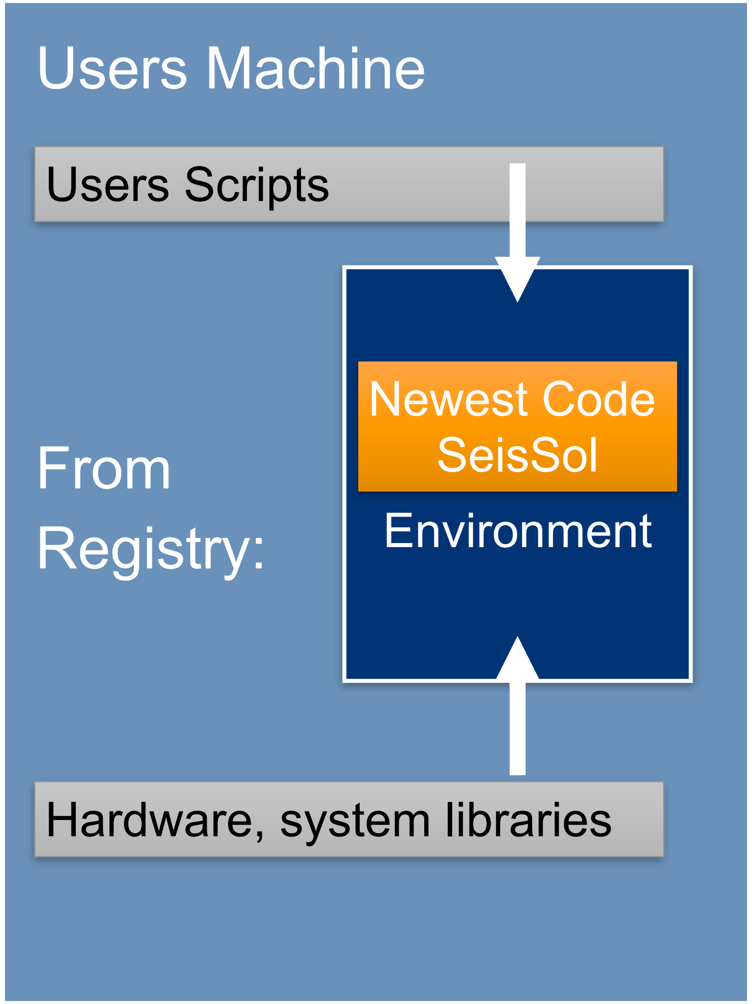


- Container environments for x86, amd, powerPC
- Pull sif image from registry
- Bind system directories in the container
- Build only new code and test

```
singularity run --nv -B $PWD:/home/$(whoami)
               --env GPU_VENDOR=nvidia
               --env GPU_MODEL=sm_61
               --env HOST=snb ./seissol-base.sif
               ./test_singularity.sh
```

- Faster: 10 min instead of 30 min
- Reproducible

SeisSol „ToGo“ with Singularity with SeisSol inside



- Goal:
  - Give users one file with a ready to use code
  - usable on many different systems
  - registry with newest version and older releases

- Challenges:
  - Many possibilities to compile (different functionalities)
  - Many different versions, legacy code?
  - Binding host system (libraries, specific hardware), requires customization on specialized systems (HPC)

Building a Container for SuperMUC-NG

To make MPI find the fabric, you need to install the necessary libs

```
FROM ubuntu:bionic
RUN apt-get update -qq && apt-get install wget software-properties-common -y --no-instal
l-recommends && add-apt-repository -y ppa:ubuntu-toolchain-r/test && apt-get update -qq
&& apt-get install -y -qq g++ gfortran python3 python3-pip m4 unzip less vim git python
pkg-config cxxtest cpio libpapi5 libpfm4 libfabric1 libpsm-infinipath1 --no-install-reco
mmends && rm -rf /var/lib/apt/lists/*
```

And MPI

```
RUN wget http://registrationcenter-download.intel.com/akdlm/irc_nas/tec/16120/
l_mpi_2019.6.166.tgz && tar xf l_mpi_2019.6.166.tgz && cd l_mpi_2019.6.166 &&
./install.sh --silent silent.cfg --accept_eula
```

Make mount points for system directories,

get your scripts+data

```
RUN mkdir -p /lrz/sys
RUN mkdir -p /usr/lib64
COPY mpi_test_sng.sh .
```

User software stack

get and compile source with wget and make

```
RUN mkdir /opt/precice
WORKDIR /opt/precice
ENV PRECICE_PREFIX=/opt/precice/precice-2.2.0
ENV LD_LIBRARY_PATH=$PRECICE_PREFIX/lib:$LD_LIBRARY_PATH
ENV CPATH=$PRECICE_PREFIX/include:$CPATH
# Enable detection with pkg-config and CMake
ENV PKG_CONFIG_PATH=$PRECICE_PREFIX/lib/pkgconfig:$PKG_CONFIG_PATH
ENV CMAKE_PREFIX_PATH=$PRECICE_PREFIX:$CMAKE_PREFIX_PATH
RUN wget https://github.com/precice/precice/archive/v2.2.0.tar.gz && tar -xzvf v2.2.0.tar.gz && cd precice-2.2.
& mkdir build && cd build && cmake -DBUILD_SHARED_LIBS=ON -DCMAKE_BUILD_TYPE=Release -DCMAKE_INSTALL_PREFIX=/op
recice/precice-2.2.0 -DPRECICE_PETScMapping=OFF -DPRECICE_PythonActions=OFF .. && make -j $(nproc)
```

or git, or pip…

```
RUN git clone https://bitbucket.org/pflotran/pflotran && cd pflotran/src/pflotran
WORKDIR pflotran/src/pflotran
RUN git checkout v3.0

RUN make pflotran PREFIX=/opt/bin/pflotran
ENV PATH=/opt/bin/pflotran/bin:$PATH
```

CharlieCloud Test on SuperMUC-NG

- Unpack the image into a folder:

```
ch-tar2dir prepfl_test.tar.gz .
```

- Run an application (here shell) in the Container:

```
ch-run -w prepfl_test -- /bin/bash
```

- Use the software stack installed inside:

```
ri46rob2@login01:/opt/precice/precice-2.2.0/build$ /cmake/bin/ctest
```

```
      Start 24: precice.solverdummy.run.cpp-c
24/26 Test #24: precice.solverdummy.run.cpp-c .............   Passed    0.32 sec
      Start 25: precice.solverdummy.run.cpp-fortran
25/26 Test #25: precice.solverdummy.run.cpp-fortran .......   Passed    0.32 sec
      Start 26: precice.solverdummy.run.c-fortran
26/26 Test #26: precice.solverdummy.run.c-fortran .........   Passed    0.33 sec

100% tests passed, 0 tests failed out of 26

Label Time Summary:
Solverdummy    =  11.50 sec*proc (9 tests)

Total Test time (real) =  29.79 sec
```

Take away

- Containers make SeisSol-CI pipelines faster and simpler
- Containers are a promising solution for sharing a complex software stacks

Documentation/Further reading

- http://www.geo-kw.de/
- https://www.pflotran.org
- https://urbs.readthedocs.io/en/latest/

- https://hpc.github.io/charliecloud/tutorial.html
- http://www.seissol.com/

- https://doku.lrz.de/display/PUBLIC/SuperMUC-NG