Friedrich-Alexander-Universität
Technische Fakultät

Efficient Docker Container Management and CI Strategies for Research Software Engineering

Dominik Thönnes, Harald Köstler

deRSE24 Würzburg

High Performance Simulation Software

waLBerla

widely applicable Lattice Boltzmann from Erlangen

- Open source Finite Element Framework with a focus on Multigrid solvers
- https://www.terraneo.fau.de/
- https://i10git.cs.fau.de/hyteg/hyteg

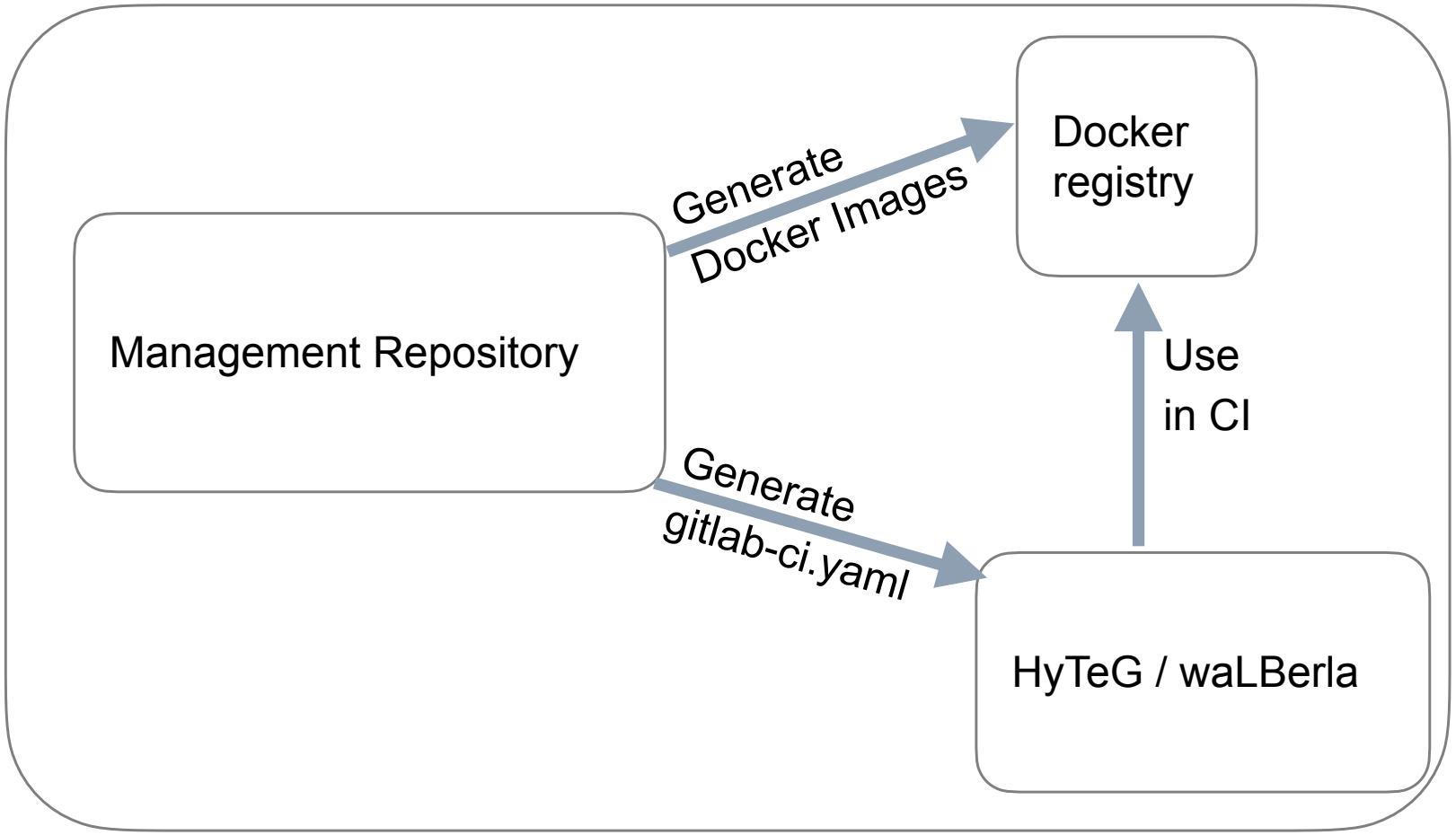
- A massively parallel open source framework for multi physics applications.
- https://www.walberla.net
- https://i10git.cs.fau.de/walberla/walberla

Requirements for CI

- Compiling library, tests, applications without warnings
- Running tests
- Different compilers (gcc, clang, intel, …) and versions
- Build Options
  - MPI,OpenMP
  - Debug, Release
  - External libs

CI @ LSS

Software / GitLab

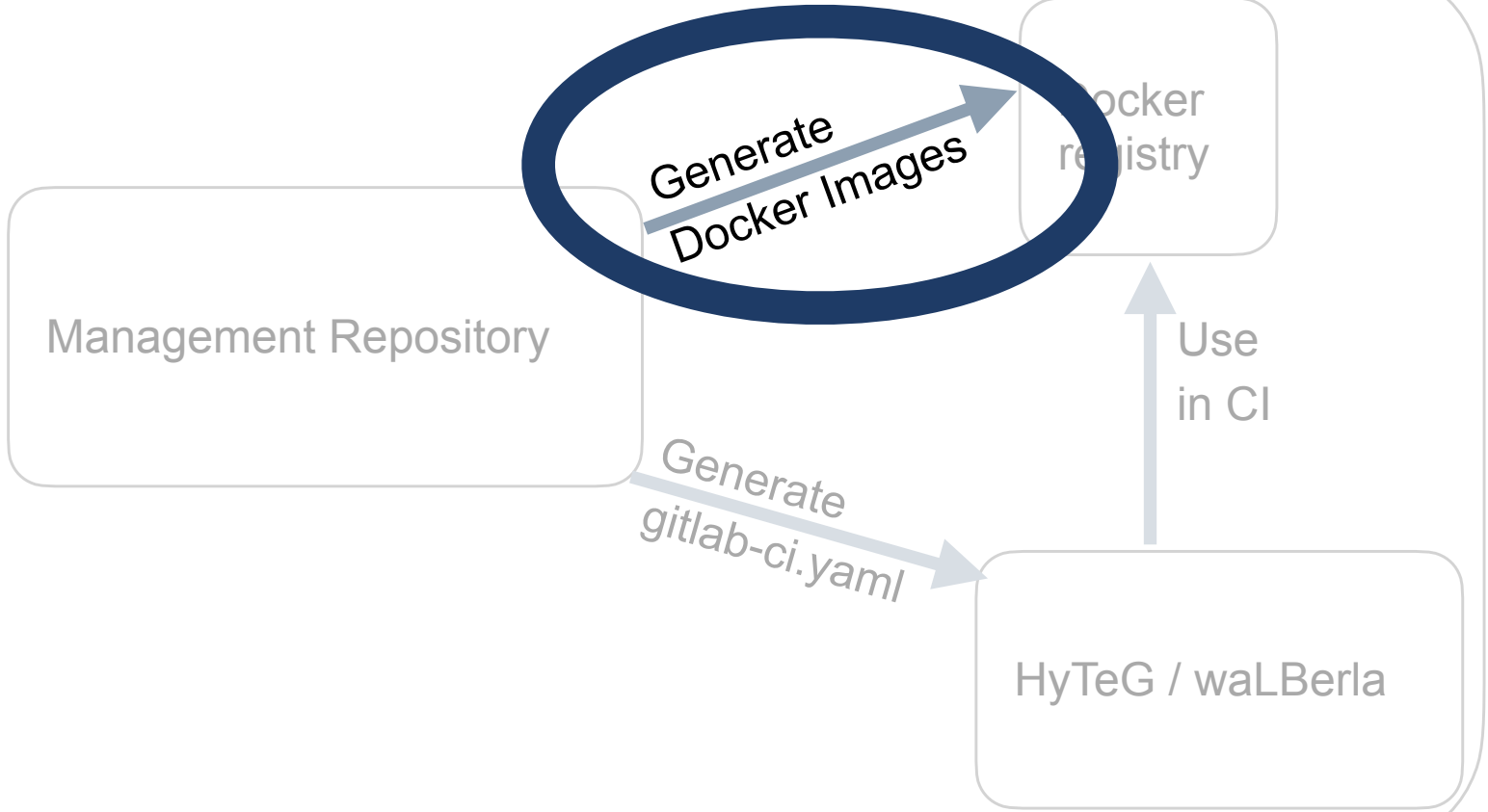


Hardware

- 14 Servers
- Various CPU architectures
- With and without GPU

CI @ LSS

Software / GitLab



Hardware

- 14 Servers
- Various CPU architectures
- With and without GPU

Docker container creation

- Goal: Docker container for each compiler with all required software
- Past: “Manual” Dockerfiles with install scripts for software
- Now: Use spack environments
- Jinja2 template for **`spack.yaml`** files
- Much smaller + cleaner codebase
- Somewhat restricted -> some aspects are harder to customise

Docker container creation

- nvidia/cuda images as a base
- Create custom image with specific spack version
- Acts as base for all compiler images

```
FROM nvidia/cuda:11.8.0-devel-ubuntu22.04

ENV DEBIAN_FRONTEND=noninteractive LANGUAGE=en_US.UTF-8 LANG=en_US.UTF-8 LC_ALL=en_US.UTF-8 SPACK_ROOT=/opt/spack
RUN apt-get -yqq update  && apt-get -yqq upgrade  && apt-get -yqq install --no-install-recommends build-essential ca-certificates  \
    curl file  git gnupg2 iproute2 locales make emacs-nox unzip wget \
    python3 python3-pip python3-setuptools python3-dev  \
    g++ gcc gfortran libc6-dev libc6-dev-i386 gcc-multilib g++-multilib \
    && locale-gen en_US.UTF-8  && pip3 install boto3  && rm -rf /var/lib/apt/lists/*
RUN mkdir $SPACK_ROOT && cd $SPACK_ROOT && git clone --depth 1 --branch=releases/v0.21 https://github.com/spack/spack.git /opt/spack
RUN ln -s $SPACK_ROOT/share/spack/docker/entrypoint.bash /usr/local/bin/docker-shell  && ln -s $SPACK_ROOT/share/spack/docker/entrypoint.bash /
usr/local/bin/interactive-shell  && ln -s $SPACK_ROOT/share/spack/docker/entrypoint.bash /usr/local/bin/spack-env
SHELL ["docker-shell"]
RUN spack bootstrap now
RUN spack mirror add lss http://i10apt.informatik.uni-erlangen.de/spack-mirror
RUN spack buildcache keys -i -t
ENTRYPOINT ["spack-env"]
CMD ["interactive-shell"]
```

Dockerfile generation

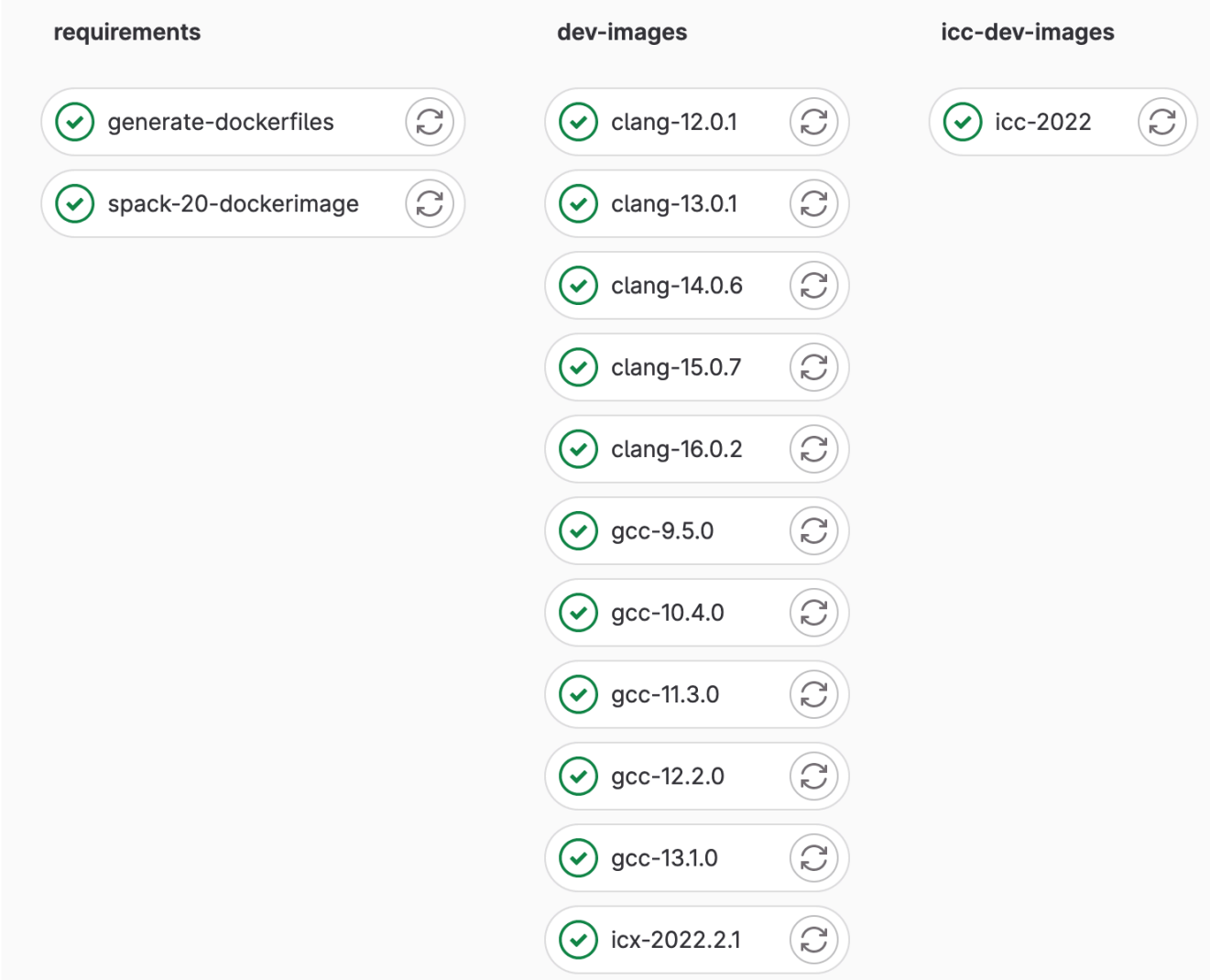
- Jinja2 Template for all compilers and versions

```
spack:
   specs:
{%- if not llvm_package %}
      - {{base_compiler}} target=x86_64
{%- endif %}
      - cmake{{build_compiler}}
      - openmpi{{build_compiler}}
      - ccache{{build_compiler}}
      - fftw+pfft_patches{{build_compiler}}
      - pfft{{build_compiler}}
      - likwid{{build_compiler}}
      - metis+int64+real64{{build_compiler}} build_type=Debug
      - parmetis+int64{{build_compiler}} build_type=Debug
      - trilinos@14.4.0+mumps~kokkos{{build_compiler}} build_type=Debug
      - petsc+int64~metis+mumps{{build_compiler}}
      - doxygen{{build_compiler}}
      - adios2+hdf5{{build_compiler}}
      - mpfr{{build_compiler}}
{%- if llvm_package %}
      - {{llvm_package}}
{%- elif oneapi_package %}
      - {{oneapi_package}}
{%- endif %}
```

```
spack:
…
   packages:
      all:
         compiler: [ {{base_compiler}} ]
         target: [ x86_64 ]
      cuda:
         buildable: False
         externals:
            - spec: "cuda@11.8.0%{{base_compiler}} arch=linux-ubuntu22-x86_64"
              prefix: /usr/local/cuda
   concretizer:
      unify: true
   config:
      install_missing_compilers: true
      template_dirs:
      - ../../templates
   mirrors:
      lss: http://i10apt.informatik.uni-erlangen.de/spack-mirror
   container:
      strip: false
      images:
         build: i10git.cs.fau.de:5005/walberla/buildenvs/{{spack_base_image}}
         final: i10git.cs.fau.de:5005/walberla/buildenvs/{{spack_base_image}}
{%- if llvm_package %}
      template: spack-dev-dockerfile-clang
{% elif oneapi_package %}
      template: spack-dev-dockerfile-icx
{%- else %}
      template: spack-dev-dockerfile-gcc
{%- endif }
```

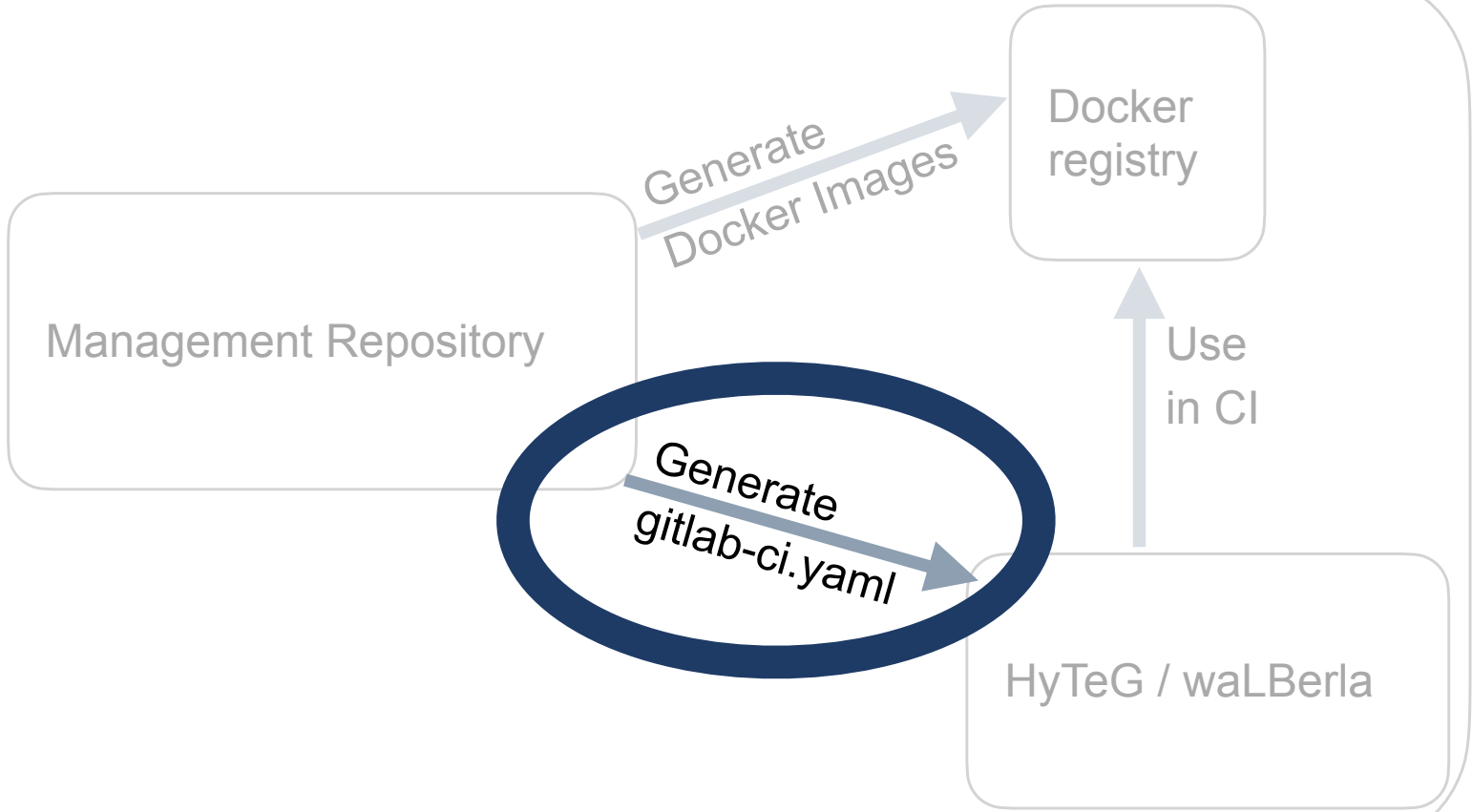
Docker Image Build

1. Generate gitlab-ci.yml
2. Generate Dockerfiles
3. Build spack base Docker image
4. Build all compiler Docker images
5. Push Docker images to registry



CI @ LSS

Software / GitLab



Hardware

- 14 Servers
- Various CPU architectures
- With and without GPU

Generation of gitlab-ci.yml

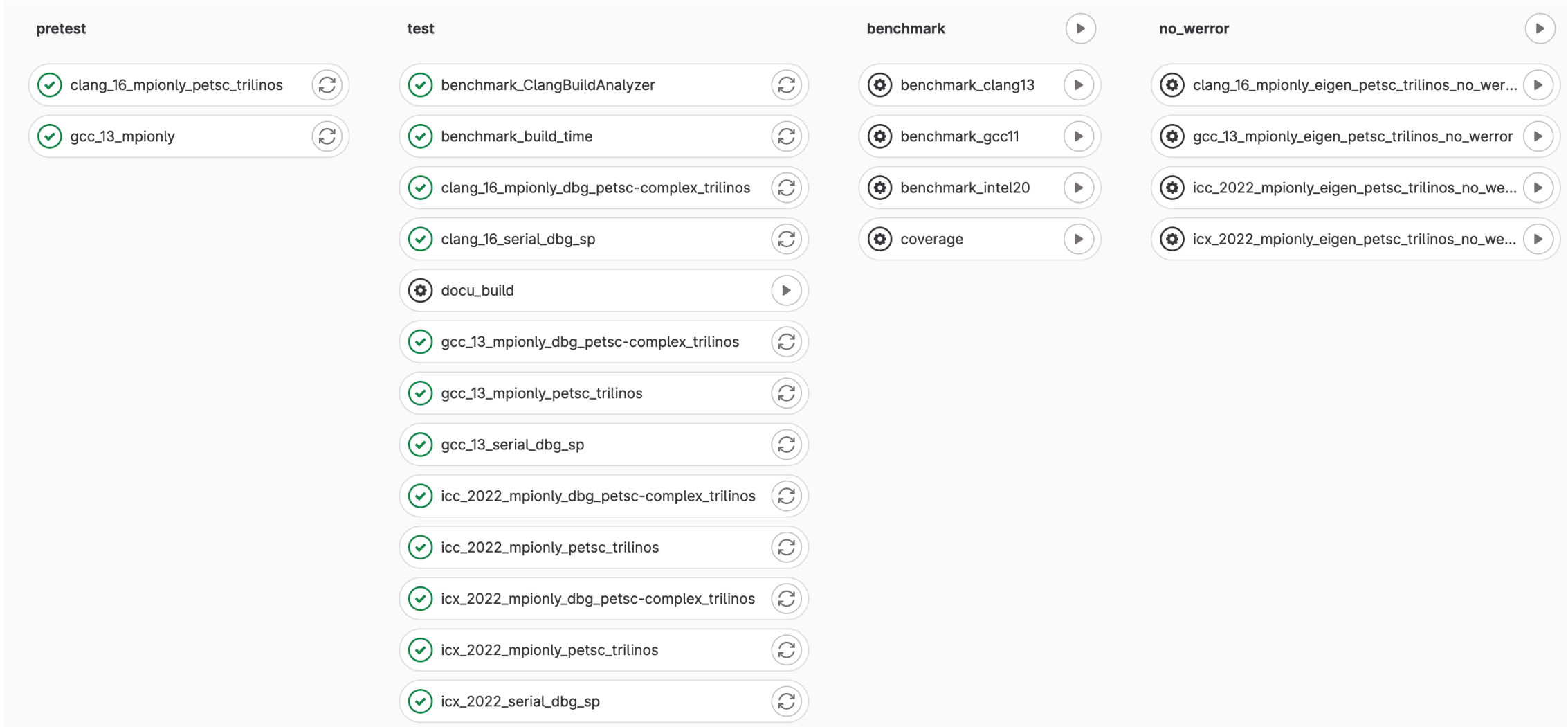
- Jinja2 templates for gitlab-ci.yml in HyTeG / waLBerla
- Python script to generate all combinations
  - Compiler Version
  - CMake Options
- Commit generated gitlab-ci.yml
- Prevent manual changes
  - Dedicated CI job

Generation of gitlab-ci.yml

```
.build_template:
  script:
   - cmake ..
     -DCMAKE_CXX_FLAGS=$CMAKE_CXX_FLAGS
     -DWALBERLA_BUILD_TESTS=ON
     -DWALBERLA_BUILD_WITH_MPI=$WALBERLA_BUILD_WITH_MPI
     -DWALBERLA_BUILD_WITH_CUDA=$WALBERLA_BUILD_WITH_CUDA
     -DWALBERLA_BUILD_WITH_PYTHON=$WALBERLA_BUILD_WITH_PYTHON
     -DWALBERLA_BUILD_WITH_OPENMP=$WALBERLA_BUILD_WITH_OPENMP
     -DWALBERLA_DOUBLE_ACCURACY=$WALBERLA_DOUBLE_ACCURACY
    - make -j $NUM_BUILD_CORES -l $NUM_CORES
    - ctest …
  tags:
    - docker
  variables:
    WALBERLA_BUILD_WITH_MPI: "ON"
    WALBERLA_BUILD_WITH_OPENMP: "ON"
    CMAKE_BUILD_TYPE: "Release"
    WALBERLA_BUFFER_DEBUG: "OFF"
    WALBERLA_DOUBLE_ACCURACY: "ON"
```

```
{% for buildJob in buildJobs %}
{{buildJob.name}}:
   extends: .build_template
   image: {{buildJob.imageName}}
   {%- if buildJob.stage %}
   stage: {{buildJob.stage}}
   {%- endif %}
   {%- if buildJob.before_script %}
   before_script:
   {%- for var in buildJob.before_script %}
      - {{var}}
   {%- endfor -%}
   {%- endif %}
   variables:
   {%- for var in buildJob.variables %}
      {{var}}
   {%- endfor %}
   {%- if buildJob.nightly %}
   only:
      variables:
         - $ENABLE_NIGHTLY_BUILDS
   {%- endif -%}
   {%- for name, elements in buildJob.properties.items()|sort %}
   {{name}}:
   {%- for element in elements|sort %}
      - {{element}}
   {%- endfor -%}
   {% endfor -%}
{% endfor %}
```

Generated pipeline



Lessons learned

- Use ccache with a global cache
- Local Docker Image registry
- Introduce pre-test stage
  - If one job fails -> likely all jobs fail
- spack build cache
- Version tags for Docker Images
  - Prevent regression with silent Image updates

CI apart from testing

- Build time - especially interesting over time
- ClangBuildAnalyzer - identify expensive headers / source files
- Documentation - automatic website updates
- Coverage - require testing for new code
- Clang Tidy - ensure coding standards
- Clang Format - ensure formatting standards

Thank you for your Attention!

Questions? Suggestions?

Dominik Thönnes (dominik.thoennes@fau.de)