VSB TECHNICAL UNIVERSITY OF OSTRAVA | IT4INNOVATIONS NATIONAL SUPERCOMPUTING CENTER

INTRODUCTION TO HIGH PERFORMANCE COMPUTING

PART 3
HPC @ IT4INNOVATIONS
BUILDING CODE ON THE CLUSTER

Jakub Beránek

EUROPEAN UNION
European Structural and Investment Funds
Operational Programme Research, Development and Education

MINISTRY OF EDUCATION, YOUTH AND SPORTS

OUTLINE

- Remote development
- Modules
- Available software on clusters
- Common toolchains (C/C++, Python)
- Containerization
- Gitlab, CI

LOCAL VS REMOTE DEVELOPMENT

- Local code development is much easier

- **Remote IDE** - VSCode/Clion/... offer remote development over SSH
  - The server compiles code on the cluster, but the UI runs on your PC
- **sshfs** - treat the remote filesystem as a local one
- **git** – synchronize code through a repository

BUILDING CODE AND DEPENDENCIES

- You must build your program and its **dependencies** for your target cluster
  - e.g. Karolina runs on Rocky Linux 8
- You DO NOT have admin privileges on the cluster
  - Standard package managers (like `yum/apt`) cannot be used

You have several options how to compile your code and its dependencies

- Use the available pre-installed modules
- Compile your code and all its dependencies from scratch
- Use Apptainer containers
- Use Spack (HPC package manager)
- ...

< > All further command examples will assume execution on a login node

ⓘ You can find more information here

USING LMOD MODULES

- Each IT4I cluster has its own set of pre-installed modules available for immediate use
- Module
  - Is a set of binaries, libraries, header files, …
  - Has a set of modules that it depends on
  - Might have several available versions (`Python/2.7.9` vs `Python/3.6.1`)
  - Might have a specific toolchain (GCC vs Intel toolchain)
- To use a module, you have to load it
  - Loading a module modifies environment variables (`PATH`, `LD_LIBRARY_PATH`)
  - This enables executing module binaries and linking to module libraries
- Lmod is used to load modules
- You can also create your own modules or ask support to install new modules for you
  - Modules are defined using EasyBuild
- If you find a module that is not working, contact support

AVAILABLE MODULES

- Language toolchains (Python, Java, C#, …)
- C/C++ compilers (GCC, Clang, Intel C++ compiler, CUDA nvcc, …)
- Communication libraries (MPI, GPI-2, ...)
- Parallel debuggers and profilers (Allinea Forge, VTune, PAPI, Scalasca, Score-P, Vampir, …)
- Parallelized libraries (FFTW, PETSc, Trilinos, Octave, ...)
- Specialized software for chemistry, bioinformatics, physics, visualization, 3D rendering, …
    - GROMACS, Gaussian, Molpro, NWChem, Orca, Phono3py, OpenFOAM, ParaView, ...

ⓘ Full list of modules available at IT4I clusters is located here

USING LMOD

```
# show available modules
$ ml av

# load a module with its dependencies
$ module load Python/3.6.8

# list loaded modules
$ module list
Currently loaded modules:
1) GCC/6.3.0 2) Python/3.6.8
$ python --version
Python 3.6.8

# unload all loaded modules
$ ml purge
$ python --version
Python 2.7.5
```

Useful hints

- Always load specific versions of modules to avoid surprises
  - `ml GCC/6.3.0` ✓
  - `ml GCC` ✗
- Module load order matters (because of conflicting dependencies)
  - `ml A B` might produce different results than `ml B A`
- Save module combinations that you commonly use into *collections*

```
$ ml purge
$ ml GCC/6.3.0 Python/3.6.8 MPICH/3.2.1-GCC-6.3.0-2.27
$ ml save mpienv1 # save current modules under name mpienv1
# ... later
$ ml restore mpienv1 # restore modules from collection mpienv1
```

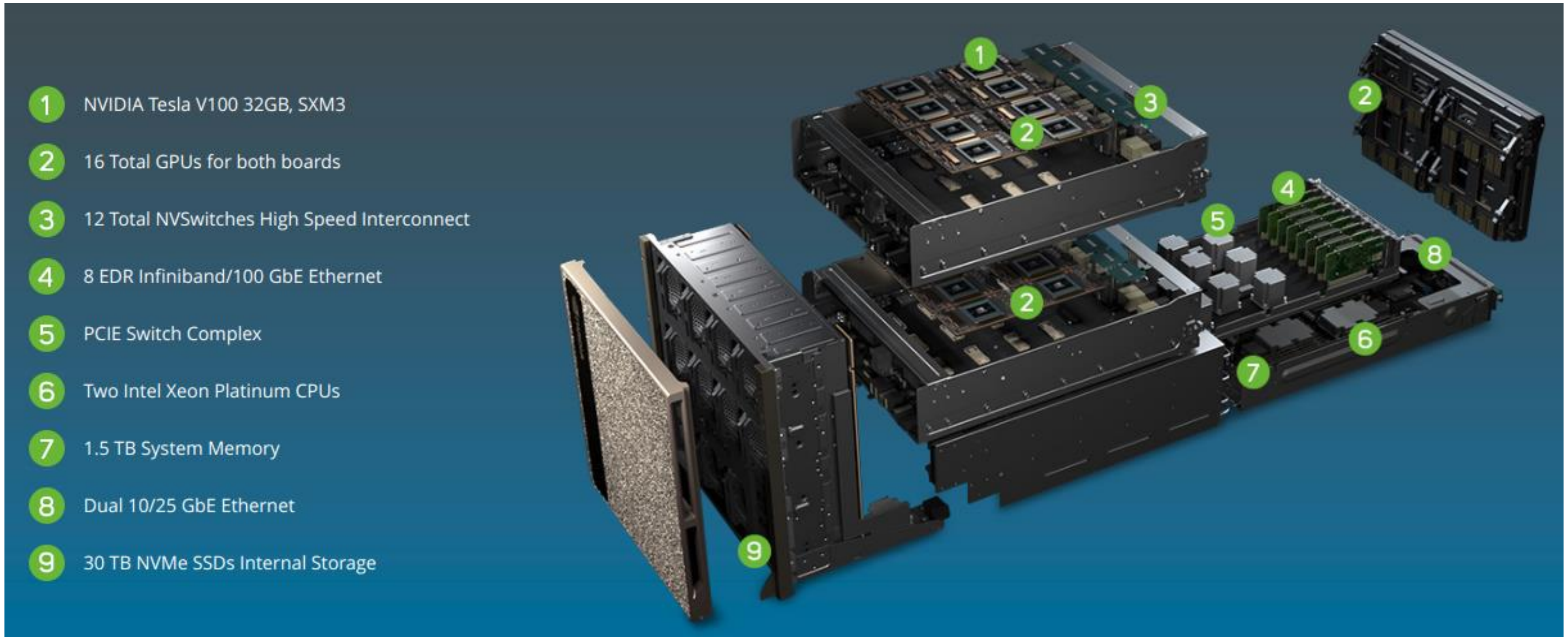
- Filtering modules
  - `$ ml spider <package>`
  - `ml` command also provides tab completion
- `ml` command is case sensitive
- Match module toolchains (GCC vs Intel)
- Do not forget to load correct modules in your Slurm job script!

USING GPUS

- Available clusters with GPUs:
  - Karolina: 72 nodes, 8 A100 (40 GiB) GPUs per node
  - Barbora: 8 nodes, 4 V100 (16 GiB) GPUs per node
  - DGX: 16 V100 (32 GiB) GPUs
- By default, Karolina jobs will allocate a single GPU
  - Check if your tool has support for Multi-GPU setups
- Use prepared modules:
  - `$ ml CUDA/12.2.0`
  - `$ ml av nvhpc`
- When loading multiple GPU modules, match their versions!
  - `$ ml CUDA/12.2.0 cuDNN/8.9.2.26-CUDA-12.2.0`

DGX-2

- Has a dedicated PBS queue
  - Accessible from Barbora (qdgx queue)
- Check if your tool has direct support for it
  - [Dask-DGX]



1 NVIDIA Tesla V100 32GB, SXM3
2 16 Total GPUs for both boards
3 12 Total NVSwitches High Speed Interconnect
4 8 EDR Infiniband/100 GbE Ethernet
5 PCIE Switch Complex
6 Two Intel Xeon Platinum CPUs
7 1.5 TB System Memory
8 Dual 10/25 GbE Ethernet
9 30 TB NVMe SSDs Internal Storage

COMPILING C/C++ PROGRAMS

1. Load necessary modules
   - Compiler (e.g. `GCC/6.3.0`)
   - Dependencies (e.g. `MPICH/3.2.1-GCC-6.3.0-2.27`)
   - Build system (e.g. CMake/3.16.2)
2. Build your program on a login node
   - Once your binary is built, it can be accessed by all cluster nodes using the shared filesystem
3. Adjust your PATH/LD_LIBRARY_PATH environment variables
   - PATH – directories where binaries are located
   - LD_LIBRARY_PATH – directories where shared dynamic libraries are located

ⓘ If a dependency is not available as a module, you must compile it yourself

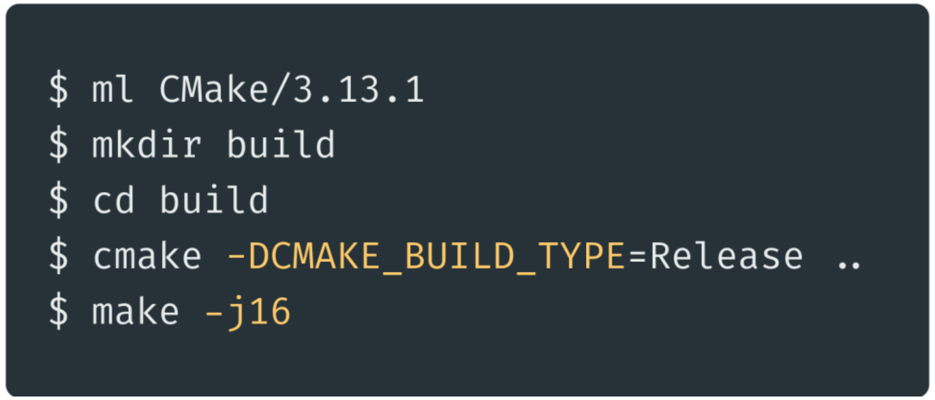
C/C++ COMPILATION FLAGS AND TIPS (GCC)

- Make use of optimizations and available instruction sets
  - Karolina has AVX2 (256-bit vectorization)
  - Barbora has AVX-512 (512-bit vectorization)
- Useful flags
  - Optimizations: `-O2`, `-O3`
    - Benchmark what works best for your code
  - Use native instruction set: `-march=native`
  - Fast floating point math (at the cost of precision): `-ffast-math`
  - Link-time optimization: `-flto`
  - Profile-guided optimization: `-fprofile-generate`, `-fprofile-use`
  - Enable OpenMP: `-fopenmp`

- Tip: you can check generated assembly at godbolt.org

COMMON C/C++ BUILD SYSTEMS

- Makefile
  - Simply run `make` in the project directory
- CMake
  1. Load CMake module
  2. Create build files inside a build directory
  3. Invoke Make (or other build systém, e.g. Ninja) to build the project
- CMake tip: use `–DCMAKE_INSTALL_PREFIX=<dir>` so that you can use `make install`



```
$ ml CMake/3.13.1
$ mkdir build
$ cd build
$ cmake -DCMAKE_BUILD_TYPE=Release ..
$ make -j16
```

MPI

- Choose desired MPI implementation and module
  - MPICH, OpenMPI, Intel MPI (`impi`)
  - Keep the same impl. and version for compilation and execution
- Compile using `mpicc` or `mpicxx`
- Run your program
  - `$ mpirun -n 2 <program>`
- More information about MPI in a later section

ⓘ There is also MPI4Py for Python

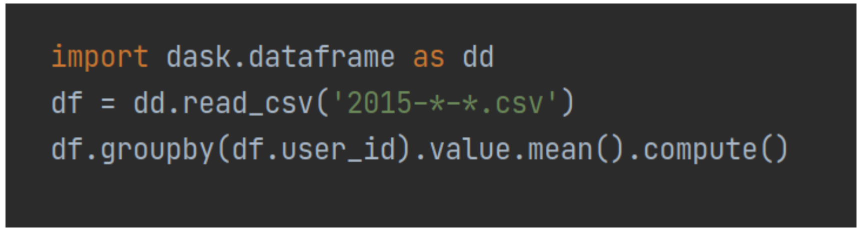
PYTHON

- Works mostly out-of-the-box on all clusters
- Make sure to load the same Python version module
  - When setting up your environment
  - Inside Slurm jobs
- Avoid using system/user Python, use virtual environments instead
  - Puts all your dependencies inside a single directory
  - venv usage example

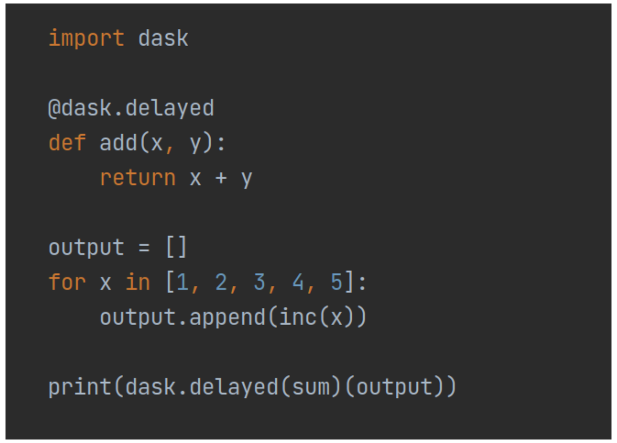
```
$ python3 -m venv venv
$ source venv/bin/activate
(venv) $ pip install -U pip setuptools wheel
(venv) $ pip install <my-package>
```

PYTHON (PERFORMANCE)

- Many useful cluster/HPC frameworks exist
  - Parallelize computation or put it on GPU with a few lines of codes
  - Distributing computation: Dask, Ray, PySpark, HyperLoom
  - GPU-acceleration: RAPIDS (cuDF, cupy), numba



```
import dask.dataframe as dd
df = dd.read_csv('2015-*-*.csv')
df.groupby(df.user_id).value.mean().compute()
```

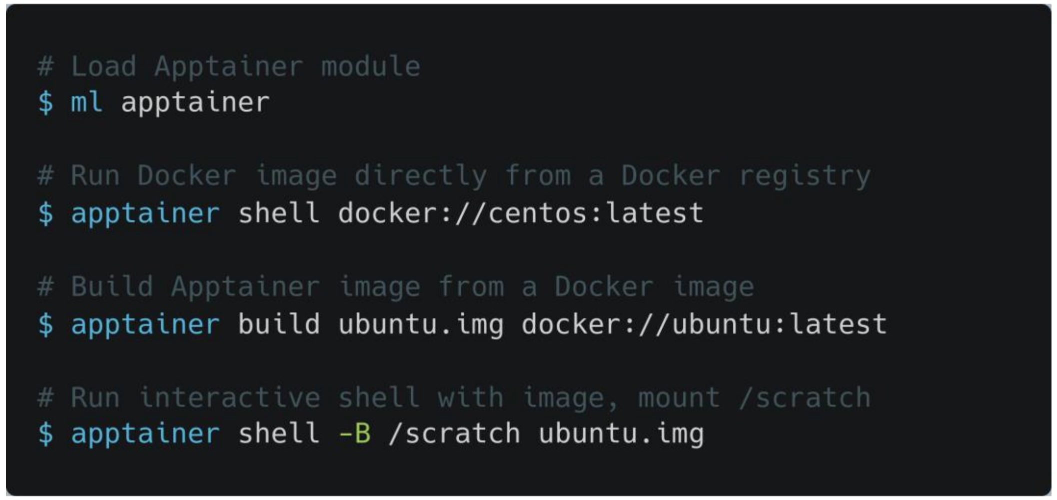


```
import dask

@dask.delayed
def add(x, y):
    return x + y

output = []
for x in [1, 2, 3, 4, 5]:
    output.append(inc(x))

print(dask.delayed(sum)(output))
```

- Python compute bound programs can be accelerated by PyPy or Cython
- Profile performance using py-spy or Scalene

CONTAINERIZATION USING APPTAINER

- Containers allow you to
  - Prepare your code and all dependencies
  - Distribute them easily in the form of an archive (image)
  - Execute them in a sandboxed environment
- Popular container solution is Docker
  - It cannot be used on IT4I clusters directly because of security issues
- You can use Apptainer instead
  - Preferred deployment method on DGX-2
  - Nvidia containers available at NGC

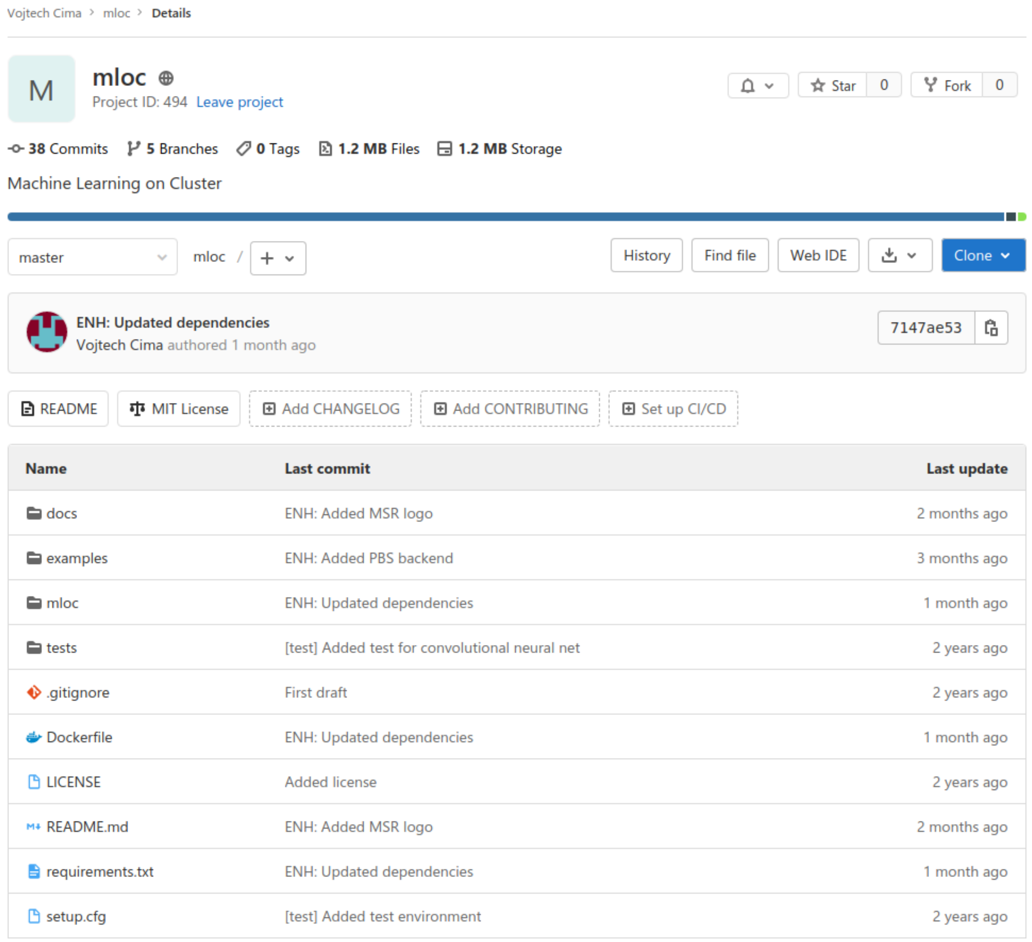


```
# Load Apptainer module
$ ml apptainer

# Run Docker image directly from a Docker registry
$ apptainer shell docker://centos:latest

# Build Apptainer image from a Docker image
$ apptainer build ubuntu.img docker://ubuntu:latest

# Run interactive shell with image, mount /scratch
$ apptainer shell -B /scratch ubuntu.img
```

ⓘ IT4I helper Apptainer wrappers (https://docs.it4i.cz/software/tools/singularity-it4i/)

GITLAB

- IT4I hosts a GitLab instance at https://code.it4i.cz
- Code storage, sharing and review (repositories, pull requests)
- Project management (issue tracker, wiki)
- Container repository
- Continuous integration



Vojtech Cima › mloc › Details

mloc
Project ID: 494 Leave project

Star 0 Fork 0

38 Commits 5 Branches 0 Tags 1.2 MB Files 1.2 MB Storage

Machine Learning on Cluster

master mloc / + History Find file Web IDE Clone

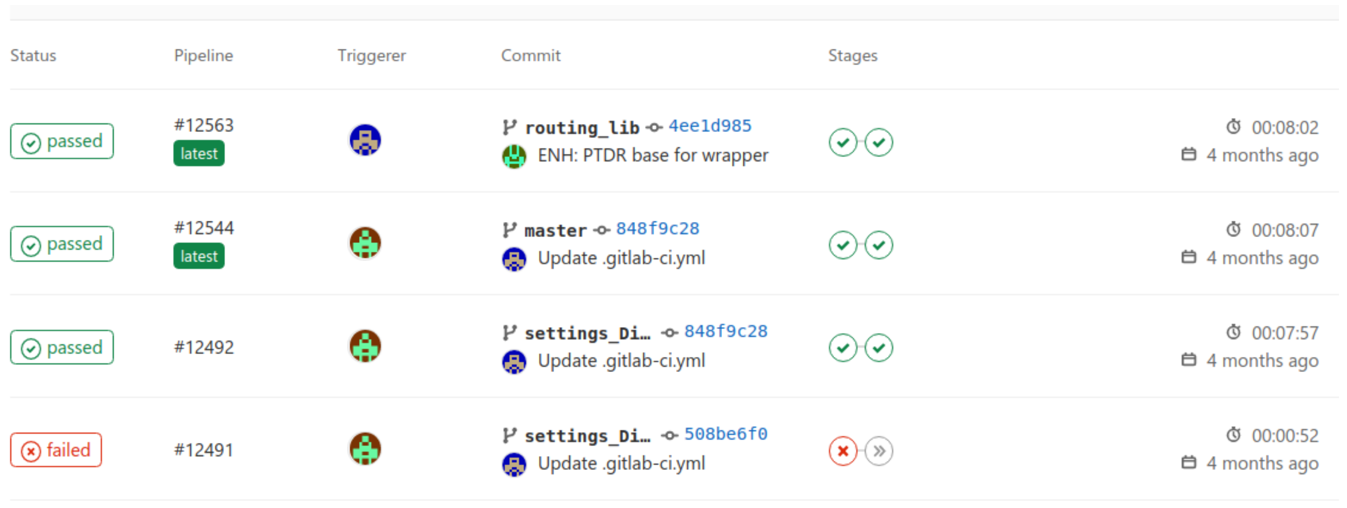
ENH: Updated dependencies
Vojtech Cima authored 1 month ago 7147ae53

README MIT License Add CHANGELOG Add CONTRIBUTING Set up CI/CD

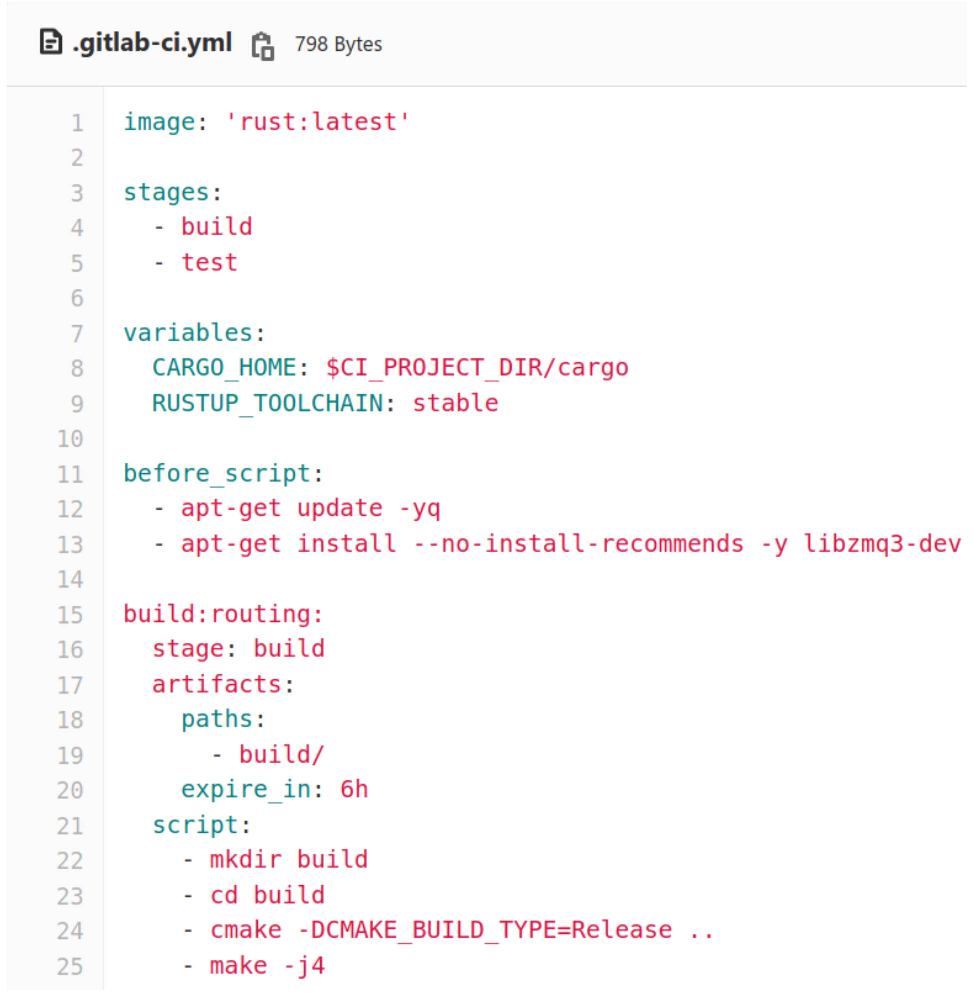
Name	Last commit	Last update
docs	ENH: Added MSR logo	2 months ago
examples	ENH: Added PBS backend	3 months ago
mloc	ENH: Updated dependencies	1 month ago
tests	[test] Added test for convolutional neural net	2 years ago
.gitignore	First draft	2 years ago
Dockerfile	ENH: Updated dependencies	1 month ago
LICENSE	Added license	2 years ago
README.md	ENH: Added MSR logo	2 months ago
requirements.txt	ENH: Updated dependencies	1 month ago
setup.cfg	[test] Added test environment	2 years ago

GITLAB CI (CONTINUOUS INTEGRATION)

- Pipelines = scripts executed after a push to a repository
  - IT4I has 5 shared runners that can run pipelines
- Check that your code was not broken by a commit
  - Correctness (unit tests)
  - Performance (benchmarks)
  - Code style, lints, …
- Deploy built artifacts
- Configured with `.gitlab-ci.yml`



Status	Pipeline	Triggerer	Commit	Stages	
passed	#12563 latest		routing_lib 4ee1d985 ENH: PTDR base for wrapper	✓ ✓	00:08:02 4 months ago
passed	#12544 latest		master 848f9c28 Update .gitlab-ci.yml	✓ ✓	00:08:07 4 months ago
passed	#12492		settings_Di… 848f9c28 Update .gitlab-ci.yml	✓ ✓	00:07:57 4 months ago
failed	#12491		settings_Di… 508be6f0 Update .gitlab-ci.yml	✗ »	00:00:52 4 months ago



.gitlab-ci.yml 798 Bytes

```yaml
image: 'rust:latest'

stages:
  - build
  - test

variables:
  CARGO_HOME: $CI_PROJECT_DIR/cargo
  RUSTUP_TOOLCHAIN: stable

before_script:
  - apt-get update -yq
  - apt-get install --no-install-recommends -y libzmq3-dev

build:routing:
  stage: build
  artifacts:
    paths:
      - build/
    expire_in: 6h
  script:
    - mkdir build
    - cd build
    - cmake -DCMAKE_BUILD_TYPE=Release ..
    - make -j4
```

Gitlab CI documentation can be found [here]()

FURTHER READING

- Productivity tools workshop
    - Git
    - EasyBuild
    - Gitlab CI
    - Singularity
    - Lmod
    - kvm

Jakub Beránek
jakub.beranek@vsb.cz

IT4Innovations National Supercomputing Center
VSB – Technical University of Ostrava
Studentská 6231/1B
708 00 Ostrava-Poruba, Czech Republic
www.it4i.cz

VSB TECHNICAL UNIVERSITY OF OSTRAVA | IT4INNOVATIONS NATIONAL SUPERCOMPUTING CENTER

EUROPEAN UNION
European Structural and Investment Funds
Operational Programme Research, Development and Education

MINISTRY OF EDUCATION, YOUTH AND SPORTS