Karlsruhe Institute of Technology

SCC
Steinbuch Centre for Computing

# Introduction to Continous-X services at NHR@KIT

Holger Obermaier | 10. November 2021

# Table of Contents

# Motivation

## What is the X in Continous-X (CX)

- Continues Integration
- Continues Testing
- Continues Benchmarking
- Continues Deployment

# Motivation (continued)

## Advantages using CX

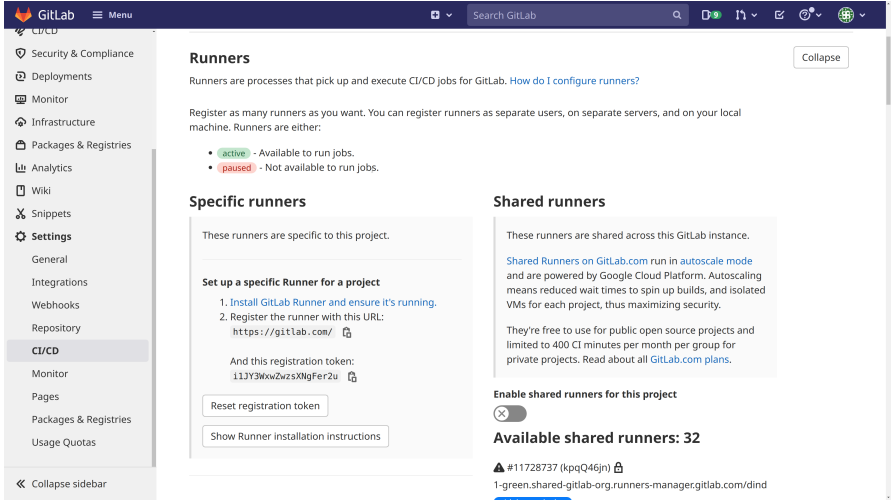
- Does it compile with different compilers, MPIs,...?
- Does it compile on different hardware?
- Do my unit tests pass?
- Do my unit tests cover most of the code?
- Linting
- Automated deployment / packaging (for varying compilers, MPIs, hardware)
- Standardized Environment in conjunction with containers

# Continuous-X Service Levels at NHR@KIT

| Level | Number jobs/day | Runner on | Self-managed Runner | Permanent Runner | Full Hardware Access (e.g. GPUs) | Container support | Jobs with timing constraints |
|---|---|---|---|---|---|---|---|
| 1 | Low | Compute Node | ✔ | ✖ | ✔ | ✔ | ✖ |
| 2 | Medium | Dedicated Login Node | ✔ | ✔ | ✔ | ✔ | medium workloads |
| 3 | High | Dedicated Hardware | ✖ | ✔ | ✔ | ✔ | ✔ |

See: NHR@KIT User Documentation: Continuous Integration: Overview

# GitLab: Settings: CI/CD: Runners



# Register a GitLab Runner on HoreKa Cluster

- Login to dedicated CI node

```
ssh hk-ci-controller.scc.kit.edu
```

```
(bq0742@hk-ci-controller.scc.kit.edu) Your OTP: <OTP>
(bq0742@hk-ci-controller.scc.kit.edu) Password: <Password>
...
Last login: Mon Oct 25 16:15:10 2021 from 2a00:1398:4:1801::810d:3bc6
 [ ] [bq0742@hkn1993] [Mon. 2021-11-08 15:07:19]
 [~] $
```

# Register a GitLab Runner on HoreKa Cluster (continued)

- Register and configure GitLab Runner

```
gitlab-runner register
```

  - Enter the GitLab instance URL: e.g. `https://gitlab.com/`
  - Enter the registration token: <Token from GitLab: Settings: CI/CD: Runners>
  - Enter a description for the runner: e.g. GitLab Runner at NHR@KIT
  - Enter an executor: shell
- Configuration is written to `${HOME}/.gitlab-runner/config.toml`
- Execute GitLab Runner

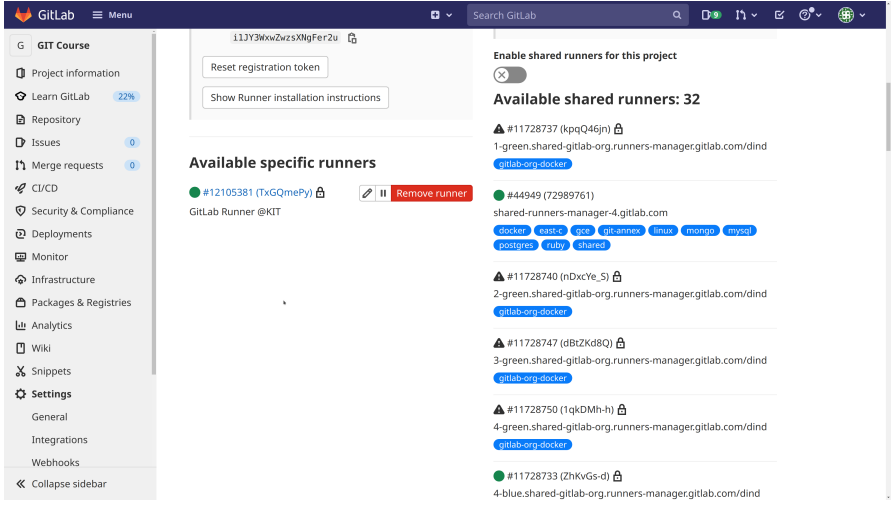
```
gitlab-runner run
```

# Register a GitLab Runner on HoreKa Cluster (Containers)

## Using Containers

- NHR@KIT User Documentation: Using Containers
- Custom executor instead of shell executor
- No native Docker support (security constraints)
- Enroot: Root-less execution of Docker images
- Template folder: `/usr/share/gitlab-runner/custom-executor-enroot`
- Template GitLab Runner config: `config.toml`
- GitLab Runner config uses prepared scripts: `config.sh`, `prepare.sh`, `run.sh`, `cleanup.sh`
- `.gitlab-ci.yml` uses keyword `image` to configure Docker image

# GitLab Settings: CI/CD: Runners



# CI Level 1 at NHR@KIT

## CI Level 1

- GitLab Runner is executed by a *batch job*
- GitLab Server is contacted, when the batch job starts
- Requires access from compute node to GitLab Server
- GitLab Runner quits when all waiting CI jobs are executed
- Problem: Start of the GitLab Runner job is unknown in advance
- For repeating GitLab Runner jobs use `scrontab`
- $\Rightarrow$ Best suited for *nightly builds* and *nightly integration* tests

# CI Level 1 at NHR@KIT (continued)

- Prerequisite: Register a GitLab Runner on HoreKa Cluster
- In CI Level 2 the GitLab Runner is executed as a batch job

```
sbatch \
    --wrap="gitlab-runner run" \
    --time="00:30:00" \
    --partition="dev_cpuonly"
```

# CI Level 1 at NHR@KIT (continued)

- Alternative method: Create a batch script for submission:

```
#!/usr/bin/bash
#SBATCH --partition dev_cpuonly
#SBATCH --time 00:30:00

# Prepare your environment
module add compiler/intel mpi/impi numlib/mkl
module list

# Start GitLab Runner
gitlab-runner run
```

# CI Level 1 at NHR@KIT (continued)

- Use `scrontab -e` to set up regular GitLab Runner jobs:

```
#SCRON -p dev_accelerated
#SCRON -t 00:30:00
@midnight gitlab-runner run
```

- Jobs are not guaranteed to execute at the preferred time!
- Jobs are regularly queued in the batch system:

```
squeue
```

```
  JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
1503545 dev_accel gitlab-r   bq0742 PD       0:00      1 (BeginTime)
```

# CI Level 2 at NHR@KIT

- Prerequisite: Register a GitLab Runner on HoreKa Cluster
- In CI Level 2 the GitLab Runner
  - is executed as systemd user service
  - runs on dedicated login node (e.g. `hk-ci-controller.scc.kit.edu`)
    - Limited / shared resources ⇒ Runtime variations
    - Only suited for medium workloads
    - No access to special hardware
  - is self-managed
    - Systemd management
    - CI jobs can start immediately

# CI Level 2 at NHR@KIT (continued)

- Start GitLab Runner Service

```
systemctl --user start gitlab-runner.service
```

- Get status of GitLab Runner Service

```
systemctl --user status gitlab-runner.service
```

```
 gitlab-runner.service - GitLab Runner for bq0742
  Loaded: loaded (/etc/systemd/user/gitlab-runner.service; disabled; vendor
  ↪  preset: enabled)
  Active: active (running) since Tue 2021-11-09 11:45:41 CET; 3s ago
Main PID: 1167130 (gitlab-runner)
  CGroup:
  ↪  /user.slice/user-8946.slice/user@8946.service/gitlab-runner.service
          1167130
```

# CI Level 2 at NHR@KIT (continued)

- Read log output of GitLab Runner Service

```
journalctl --user --unit gitlab-runner.service
```

```
WARNING: Running in user-mode.
WARNING: Use sudo for system-mode:
WARNING: $ sudo gitlab-runner...

Configuration loaded                                    builds=0
listen_address not defined, metrics & debug endpoints disabled  builds=0
[session_server].listen_address not defined, session endpoints disabled
↪  builds=0
```

# CI Level 2 at NHR@KIT (continued)

- Stop GitLab Runner Service

```
systemctl --user stop gitlab-runner.service
```

# CI Level 3 at NHR@KIT

- Dedicated hardware
- For projects that
  - generate many CI jobs per day
  - need predictable runtimes and performance
  - require privileged access to special resources
- Available platforms:
  - Intel: Broadwell, Cascade Lake, Ice Lake
  - NVIDIA: V100, A100
  - AMD: Rome, Milan, MI100
  - Fujitsu: ARM64FX
- Get in contact with CI Operations Team

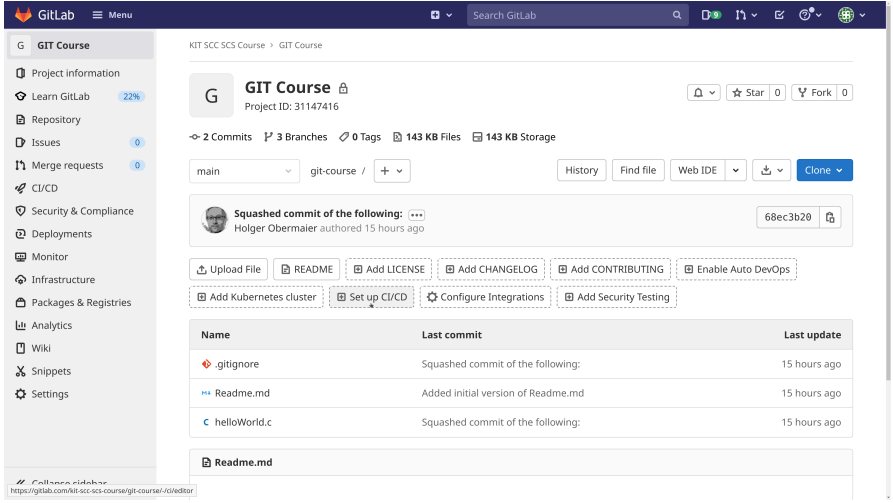
# Continuous Integration using GitLab

- GitLab Continuous Integration Documentation:
  - Get started with GitLab CI/CD
  - Keyword reference for the .gitlab-ci.yml file
  - GitLab CI templates
- YAML-config file `gitlab-ci.yml`
- GitLab offers integrated CI-editor with visualization and linting

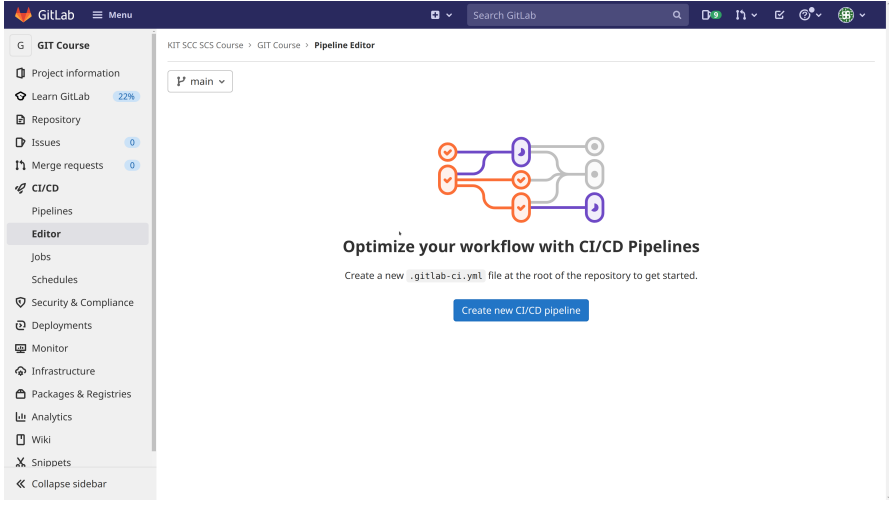
# Continuous Integration using GitLab (continued)

- Each commit triggers a new pipeline
- Each pipeline consist of stages
  - Predefined stages: `.pre`, `build`, `test`, `deploy`, `.post`
  - Stages run in sequence
  - Working tree is cleaned up between stages: Use artifacts to keep files
- Each stage consist of jobs
  - Jobs can run in parallel
- When a job fails the stage fails, all remaining stages are skipped
- When a stage fails the pipeline fails

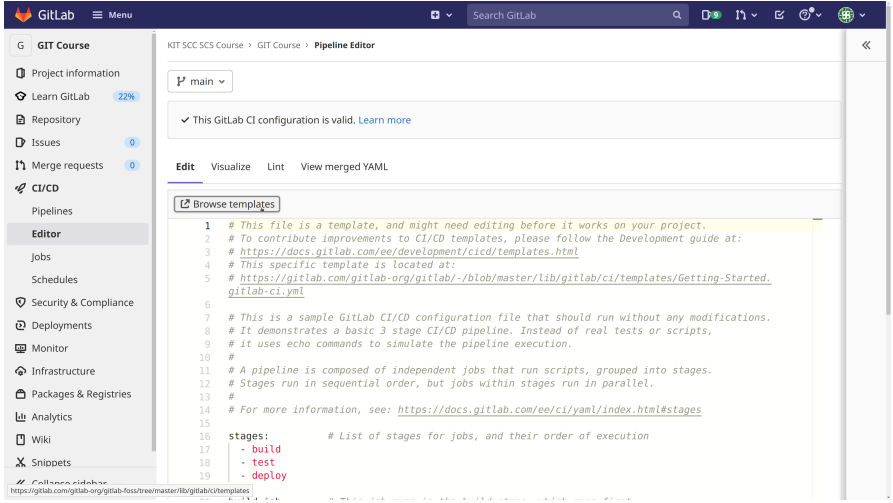
# Continuous Integration using GitLab (Set up CI/CD)



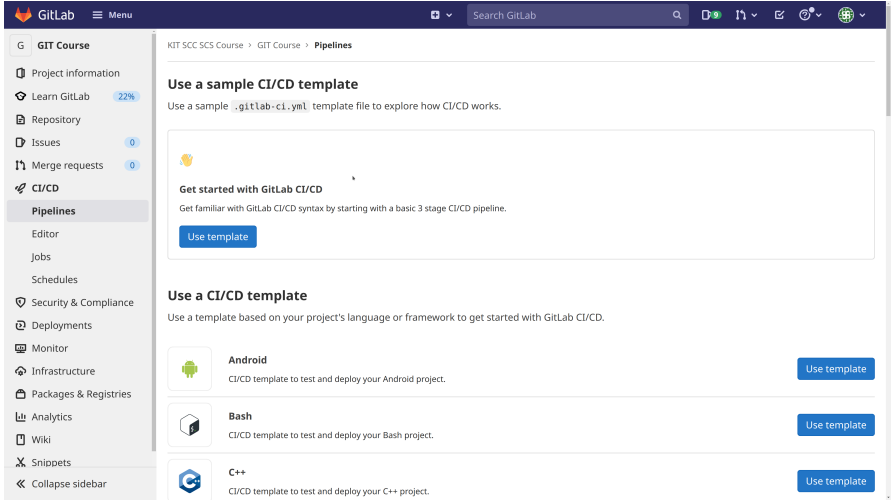
# Continuous Integration using GitLab (CI/CD Editor)



# Continuous Integration using GitLab (Browse Templates)



# Continuous Integration using GitLab (CI/CD Templates)



# Continuous Integration using GitLab (`.gitlab-ci.yml`)

```
stages:
- build
- test

build-hello-world-job:
  stage: build
  script: cc helloWorld.c -o helloWorld
  artifacts:
    paths:
    - helloWorld
    expire_in: 1 day

test-code-job1:
  stage: test
  script: if [[ $(helloWorld) != "Hello World" ]]; then exit 1; fi
```

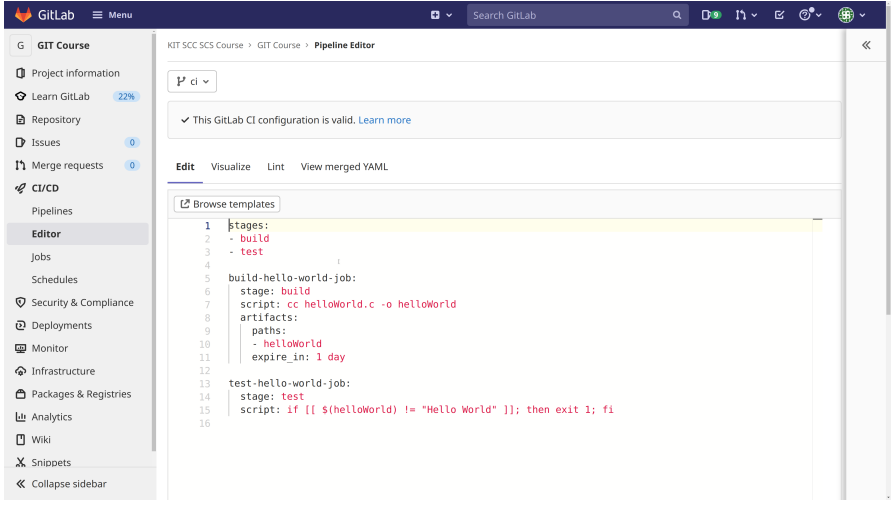
# Continuous Integration using GitLab (`.gitlab-ci.yml`)

- Test stage can use compute resources

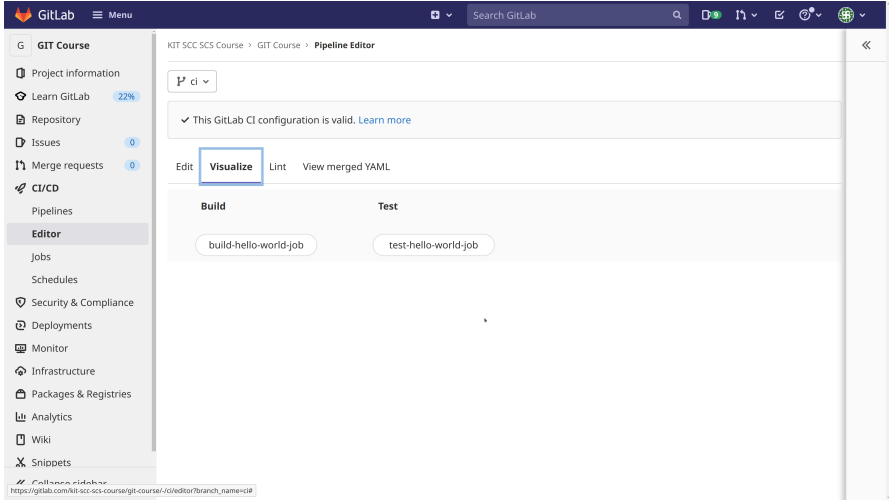
```
...

test-code-job2:
  stage: test
  script:
  - srun -p dev_cpuonly     -t 20 -N 1 -n 76              CPU_Test.sh
  - srun -p dev_accelerated -t 20 -N 1 -n 76 --gres=gpu:4 GPU_Test.sh
```

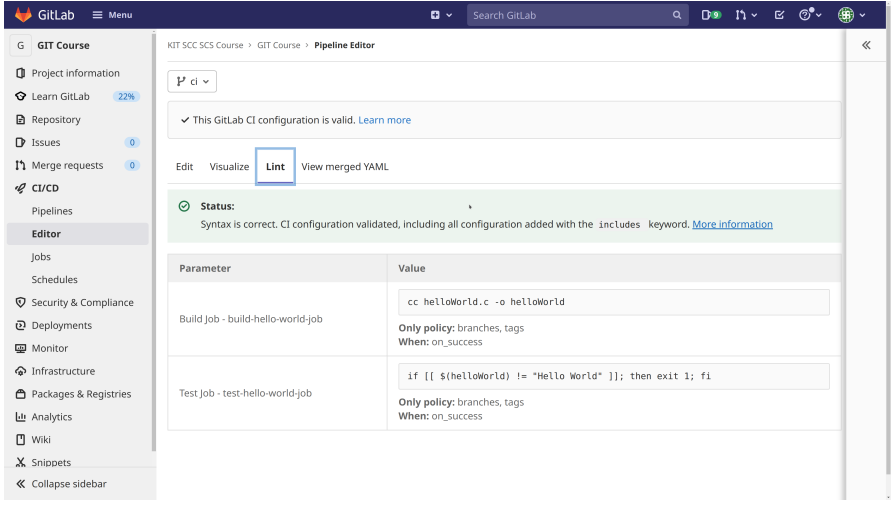
# Continuous Integration using GitLab (Edit)



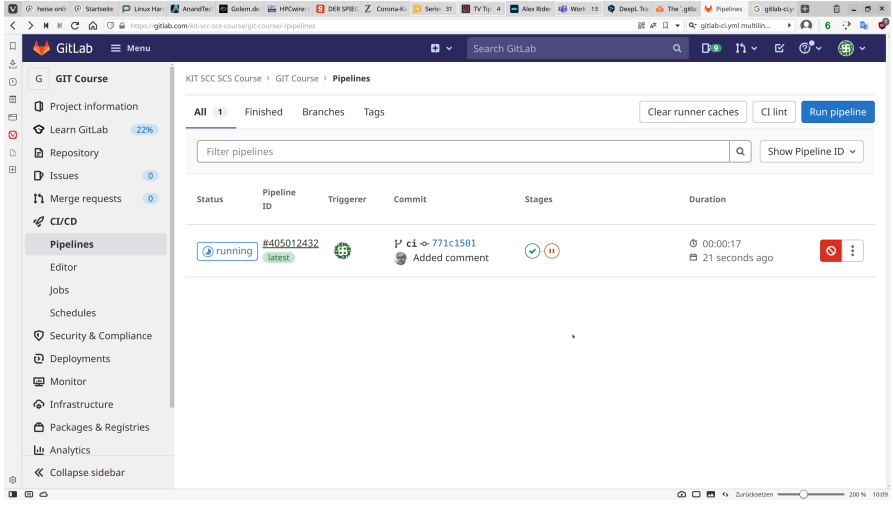
# Continuous Integration using GitLab (Visualize)



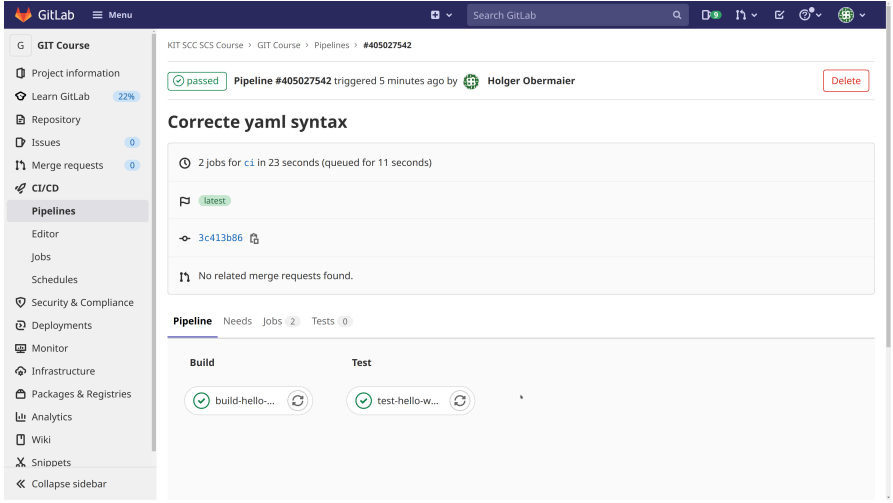
# Continuous Integration using GitLab (Lint)



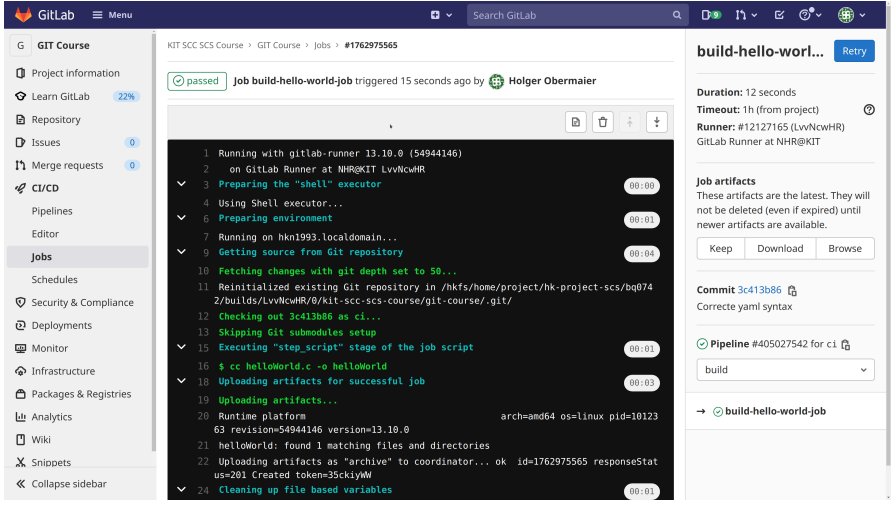
# Continuous Integration using GitLab (CI/CD: Pipelines)



# Continuous Integration using GitLab (Pipeline passed)



# Continuous Integration using GitLab (Build Job)



# Continuous Integration using GitLab (Test Job)