

Introduction to Continuous-X services at NHR@KIT

Holger Obermaier | 25. May 2022



Table of Contents

1. Motivation

2. Continuous-X at NHR@KIT

3. Continuous Integration using GitLab

What is the X in Continuous-X (CX)

- Continuous Integration
- Continuous Testing
- Continuous Benchmarking
- Continuous Deployment


Motivation...

Advantages using CX

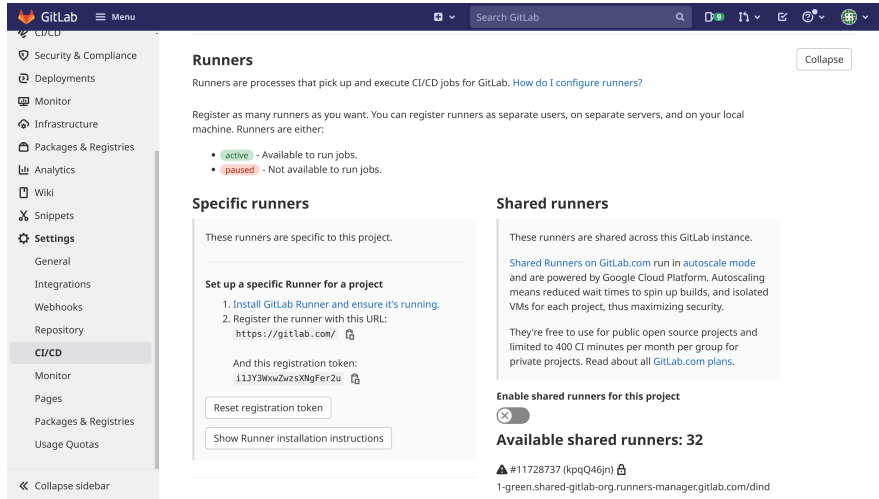
- Does it compile with different compilers, MPIs,...?
- Does it compile on different hardware?
- Do my unit tests pass?
- Do my unit tests cover most of the code?
- Linting
- Automated deployment / packaging (for varying compilers, MPIs, hardware)
- Standardized Environment in conjunction with containers

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



The screenshot shows the GitLab web interface for the 'Runners' settings page. The left sidebar contains navigation options: Security & Compliance, Deployments, Monitor, Infrastructure, Packages & Registries, Analytics, Wiki, Snippets, Settings (selected), General, Integrations, Webhooks, Repository, CI/CD (selected), Monitor, Pages, Packages & Registries, Usage Quotas, and Collapse sidebar. The main content area is titled 'Runners' and includes a 'Collapse' button. The text explains that runners are processes for CI/CD jobs and lists their states: 'active' (available) and 'paused' (not available). It is divided into 'Specific runners' (project-specific) and 'Shared runners' (instance-wide). The 'Specific runners' section provides instructions to set up a runner, including a registration URL and a token. The 'Shared runners' section shows that 32 runners are available for the project and lists one runner with its ID and URL.

Runners Collapse

Runners are processes that pick up and execute CI/CD jobs for GitLab. [How do I configure runners?](#)

Register as many runners as you want. You can register runners as separate users, on separate servers, and on your local machine. Runners are either:

- **active** - Available to run jobs.
- **paused** - Not available to run jobs.

Specific runners

These runners are specific to this project.

Set up a specific Runner for a project

1. [Install GitLab Runner and ensure it's running.](#)
2. Register the runner with this URL:
`https://gitlab.com/`

And this registration token:
`i1JY3WxwZwzsXNgFer2u`

Shared runners


These runners are shared across this GitLab instance.

[Shared Runners on GitLab.com](#) run in **autoscale mode** and are powered by Google Cloud Platform. Autoscaling means reduced wait times to spin up builds, and isolated VMs for each project, thus maximizing security.

They're free to use for public open source projects and limited to 400 CI minutes per month per group for private projects. Read about all [GitLab.com plans](#).

Enable shared runners for this project

Available shared runners: 32

#11728737 (kppQ46jn) 
[1-green.shared-gitlab-org.runners-manager.gitlab.com/dind](#)

Register a GitLab Runner on HoreKa Cluster

- Login to dedicated CI node

```
ssh bq0742@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: Thu May 5 11:41:16 2022 from 2a00:1398:4:0:5a8e:d6ff:8366:d96e  
[bq0742@hkn1993 ~]$
```

Register a GitLab Runner on HoreKa Cluster...

- 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 Custom Runner on HoreKa Cluster

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

Register a Custom Runner on HoreKa Cluster...

- Register and configure GitLab Runner

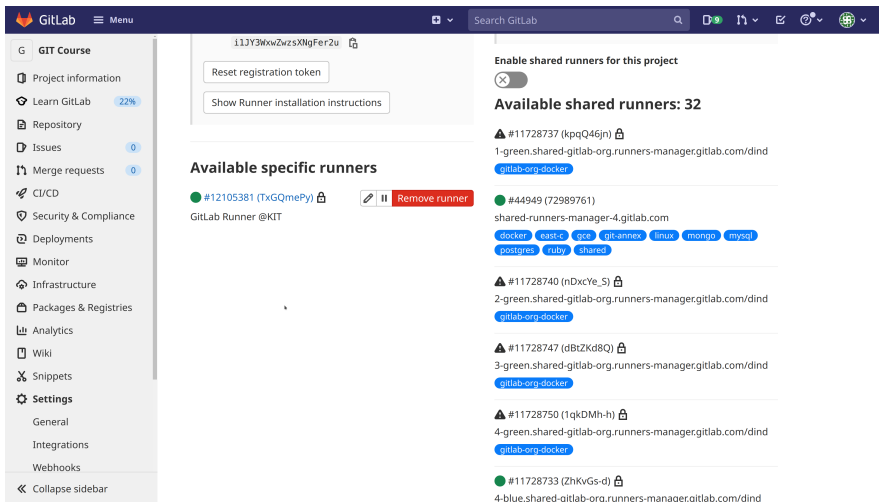
```
gitlab-runner register
```

- 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. Custom Runner for Enroot at NHR@KIT
- Enter an executor: custom
- Use template `/usr/share/gitlab-runner/custom-executor-enroot/config.toml`
- Use config options `url` and `token` from `${HOME}/.gitlab-runner/config.toml` to modify template

Register a Custom Runner on HoreKa Cluster...

```
[[runners]]
...
builds_dir = "/home/hk-project-scs/bq0742/gitlab-runner/builds/"
cache_dir = "/home/hk-project-scs/bq0742/gitlab-runner/cache/"
[runners.custom]
  config_exec = "/usr/share/gitlab-runner/custom-executor-enroot/config.sh"
  config_exec_timeout = 200
  prepare_exec = "/usr/share/gitlab-runner/custom-executor-enroot/prepare.sh"
  prepare_exec_timeout = 200
  run_exec = "/usr/share/gitlab-runner/custom-executor-enroot/run.sh"
  cleanup_exec = "/usr/share/gitlab-runner/custom-executor-enroot/cleanup.sh"
  cleanup_exec_timeout = 200
  graceful_kill_timeout = 200
  force_kill_timeout = 200
```

GitLab Settings: CI/CD: Runners



The screenshot shows the GitLab interface for configuring CI/CD runners. The left sidebar contains navigation options: GIT Course, Project information, Learn GitLab (22%), Repository, Issues (0), Merge requests (0), CI/CD, Security & Compliance, Deployments, Monitor, Infrastructure, Packages & Registries, Analytics, Wiki, Snippets, and Settings (General, Integrations, Webhooks, Collapse sidebar). The main content area is titled "Available specific runners" and shows one runner: #12105381 (TxGQmePy) with a "Remove runner" button. The right sidebar is titled "Enable shared runners for this project" (disabled) and "Available shared runners: 32". It lists five shared runners with their IDs, hostnames, and tags (e.g., docker, east-e, gce, git-annex, linux, mongo, mysql, postgres, ruby, shared).

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`
- ⇒ Best suited for *nightly builds* and *nightly integration tests*

CI Level 1 at NHR@KIT...

- Prerequisite: Register a GitLab Runner on HoreKa Cluster
- In CI Level 1 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...

- 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...

- 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:

```
queue
```

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 \Rightarrow 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...

- 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...

- 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...

- 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: Milan, MI100
 - Fujitsu: ARM64FX
- Get in contact with CI Operations Team

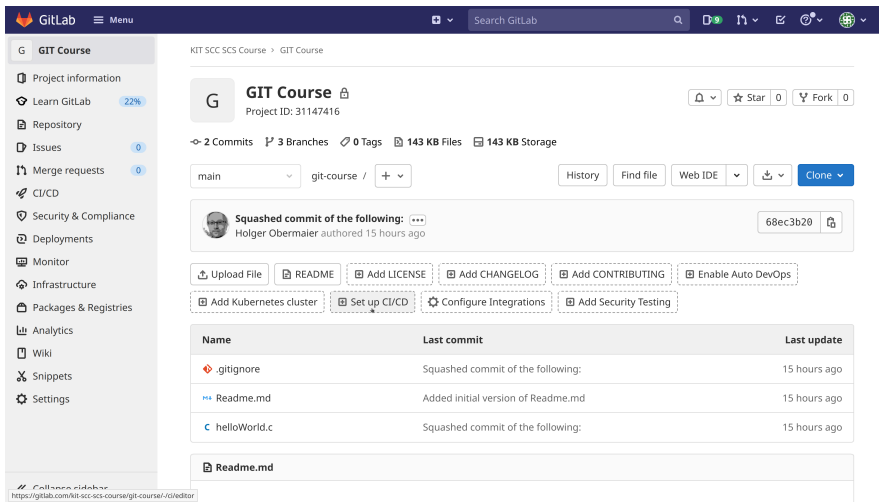
Continuous Integration using GitLab

- 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...


- 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)



GitLab Menu

KIT SCC SCS Course > GIT Course

GIT Course  Project ID: 31147416




2 Commits 3 Branches 0 Tags 143 KB Files 143 KB Storage

main git-course / + History Find file Web IDE Clone

Squashed commit of the following: 68ec3b20
Holger Obermaier authored 15 hours ago

Upload File README Add LICENSE Add CHANGELOG Add CONTRIBUTING Enable Auto DevOps

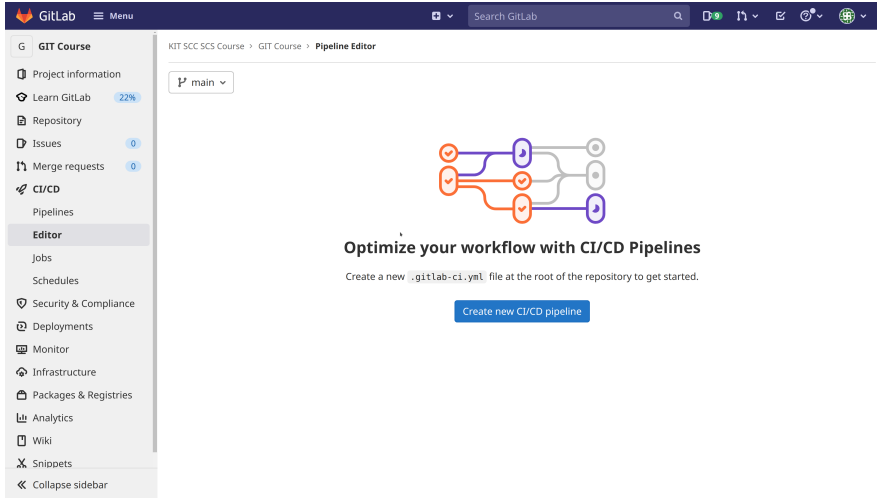
Add Kubernetes cluster Set up CI/CD Configure Integrations Add Security Testing

Name	Last commit	Last update
 .gitignore	Squashed commit of the following:	15 hours ago
 README.md	Added initial version of README.md	15 hours ago
 helloWorld.c	Squashed commit of the following:	15 hours ago

Readme.md

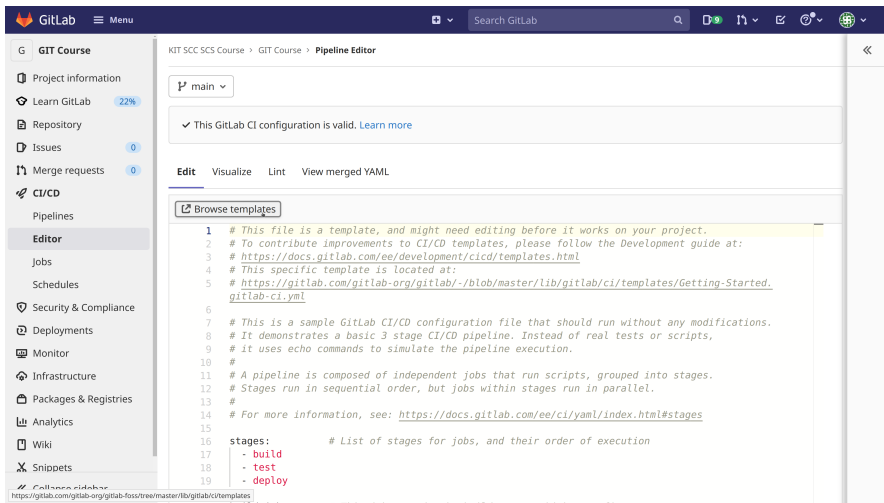
Call me: +49 7141 147-4100
<https://gitlab.com/kit-scc-scs-course/git-course/ci/editor>

Continuous Integration using GitLab (CI/CD Editor)



The screenshot shows the GitLab Pipeline Editor interface. The top navigation bar includes the GitLab logo, a menu icon, and a search bar. The left sidebar contains a navigation menu with categories like 'GIT Course', 'CI/CD', and 'Editor'. The main content area displays the 'Pipeline Editor' for a repository named 'KIT SCC SCS Course'. It features a dropdown menu for the 'main' branch and a central diagram of a CI/CD pipeline with nodes and connections. Below the diagram, the text reads 'Optimize your workflow with CI/CD Pipelines' and 'Create a new .gitlab-ci.yml file at the root of the repository to get started.' A blue button labeled 'Create new CI/CD pipeline' is positioned below the text.

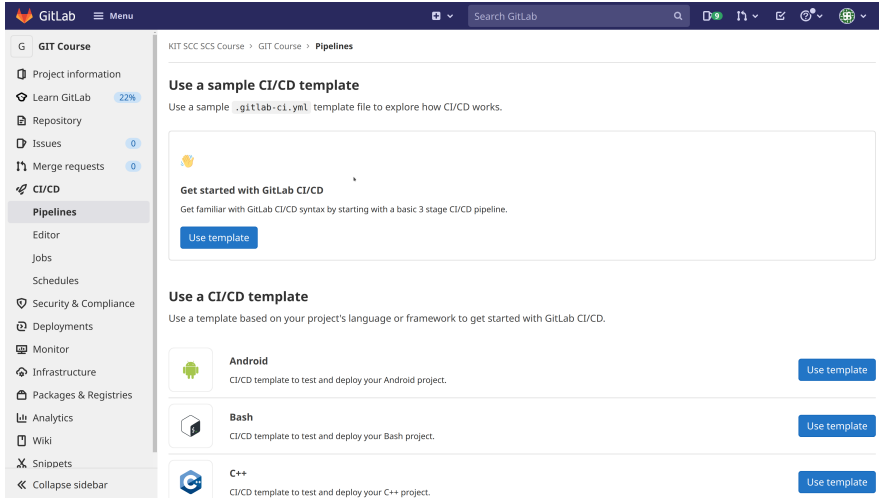
Continuous Integration using GitLab (Browse Templates)



The screenshot shows the GitLab Pipeline Editor interface. The top navigation bar includes the GitLab logo, a menu icon, and a search bar. The left sidebar contains a navigation menu with categories like 'GIT Course', 'CI/CD', and 'Security & Compliance'. The main content area is titled 'Pipeline Editor' and shows a dropdown menu for 'main'. A message indicates that the GitLab CI configuration is valid. Below this, there are tabs for 'Edit', 'Visualize', 'Lint', and 'View merged YAML'. A 'Browse templates' button is visible, and the main area displays a sample CI configuration file with comments and a 'stages' section.

```
1 # This file is a template, and might need editing before it works on your project.
2 # To contribute improvements to CI/CD templates, please follow the Development guide at:
3 # https://docs.gitlab.com/ee/development/cicd/templates.html
4 # This specific template is located at:
5 # https://gitlab.com/gitlab-org/gitlab/-/blob/master/lib/gitlab/ci/templates/Getting-Started.
   gitlab-ci.yml
6
7 # This is a sample GitLab CI/CD configuration file that should run without any modifications.
8 # It demonstrates a basic 3 stage CI/CD pipeline. Instead of real tests or scripts,
9 # it uses echo commands to simulate the pipeline execution.
10 #
11 # A pipeline is composed of independent jobs that run scripts, grouped into stages.
12 # Stages run in sequential order, but jobs within stages run in parallel.
13 #
14 # For more information, see: https://docs.gitlab.com/ee/ci/yaml/index.html#stages
15
16 stages:           # List of stages for jobs, and their order of execution
17   - build
18   - test
19   - deploy
```

Continuous Integration using GitLab (CI/CD Templates)



The screenshot shows the GitLab web interface. The top navigation bar includes the GitLab logo, a menu icon, and a search bar. The left sidebar contains a navigation menu with categories like 'GIT Course', 'Project information', 'Learn GitLab', 'Repository', 'Issues', 'Merge requests', 'CI/CD', 'Pipelines', 'Editor', 'Jobs', 'Schedules', 'Security & Compliance', 'Deployments', 'Monitor', 'Infrastructure', 'Packages & Registries', 'Analytics', 'Wiki', 'Snippets', and 'Collapse sidebar'. The main content area is titled 'KIT SCC SCS Course > GIT Course > Pipelines'. It features a section 'Use a sample CI/CD template' with a sub-section 'Get started with GitLab CI/CD' and a 'Use template' button. Below this, there is another section 'Use a CI/CD template' with three options: 'Android', 'Bash', and 'C++', each with a 'Use template' button.

GitLab Menu

KIT SCC SCS Course > GIT Course > Pipelines

Use a sample CI/CD template

Use a sample `.gitlab-ci.yml` template file to explore how CI/CD works.




Get started with GitLab CI/CD

Get familiar with GitLab CI/CD syntax by starting with a basic 3 stage CI/CD pipeline.

[Use template](#)

Use a CI/CD template

Use a template based on your project's language or framework to get started with GitLab CI/CD.

-  **Android**
CI/CD template to test and deploy your Android project. [Use template](#)
-  **Bash**
CI/CD template to test and deploy your Bash project. [Use template](#)
-  **C++**
CI/CD template to test and deploy your C++ project. [Use template](#)

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

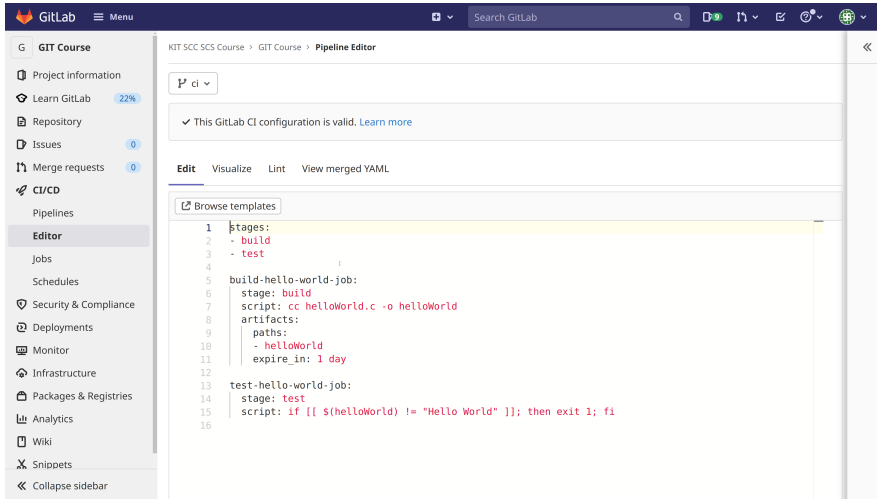
```
stages:  
- build  
- test  
  
build-hello-world-job:  
  stage: build  
  script: cc helloWorld.c -o helloWorld  
  artifacts:  
    paths:  
    - helloWorld  
    expire_in: 1 day  
  
test-code-job1:  
  stage: test  
  script: if [[ $(helloWorld) != "Hello World" ]]; then exit 1; fi
```

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
```

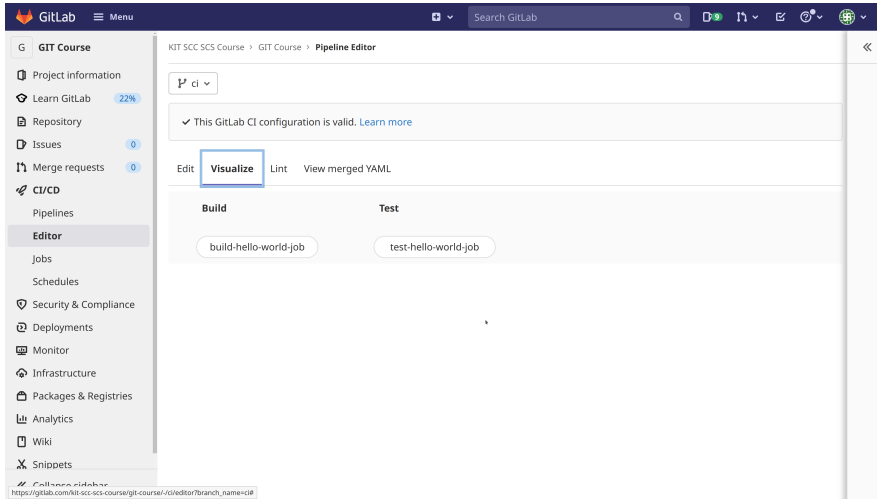
Continuous Integration using GitLab (Edit)



The screenshot shows the GitLab Pipeline Editor interface. The top navigation bar includes the GitLab logo, a menu icon, and a search bar. The left sidebar contains a navigation menu with categories like 'GIT Course', 'CI/CD', and 'Editor'. The main content area displays the pipeline configuration for 'KIT SCC SCS Course > GIT Course > Pipeline Editor'. A confirmation message states 'This GitLab CI configuration is valid. Learn more'. Below this, there are tabs for 'Edit', 'Visualize', 'Lint', and 'View merged YAML'. A 'Browse templates' button is visible. The main editor shows a YAML configuration for a pipeline with two stages: 'build' and 'test'. The 'build' stage includes a job 'build-hello-world-job' with a script to compile a C program and an artifact named 'helloWorld'. The 'test' stage includes a job 'test-hello-world-job' with a script that checks if the artifact contains the string 'Hello World'.

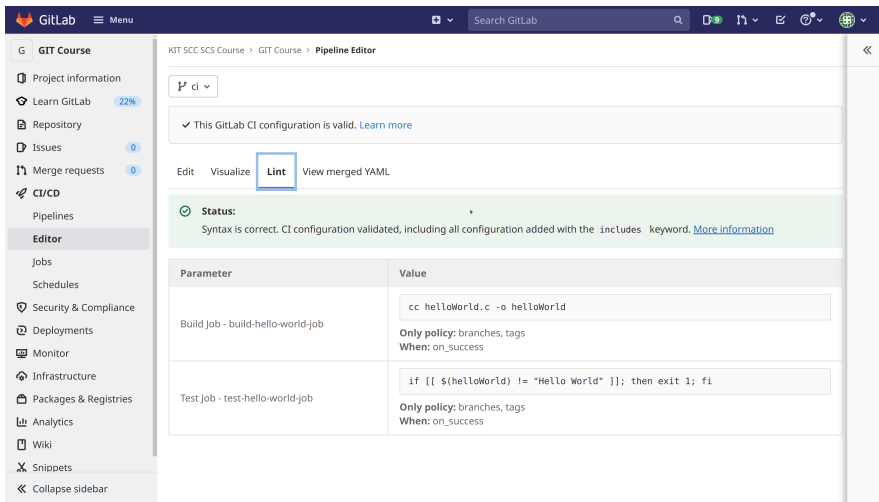
```
1 stages:
2   - build
3   - test
4
5 build-hello-world-job:
6   stage: build
7   script: cc helloWorld.c -o helloWorld
8   artifacts:
9     paths:
10      - helloWorld
11     expire_in: 1 day
12
13 test-hello-world-job:
14   stage: test
15   script: if [[ $(helloWorld) != "Hello World" ]]; then exit 1; fi
16
```

Continuous Integration using GitLab (Visualize)



The screenshot shows the GitLab Pipeline Editor interface. The top navigation bar includes the GitLab logo, a menu icon, and a search bar. The left sidebar contains a navigation menu with the following items: GIT Course, Project information, Learn GitLab (22%), Repository, Issues (0), Merge requests (0), CI/CD, Pipelines, Editor (highlighted), Jobs, Schedules, Security & Compliance, Deployments, Monitor, Infrastructure, Packages & Registries, Analytics, Wiki, and Snippets. The main content area displays the Pipeline Editor for the 'KIT SCC SCS Course' project. It shows a dropdown menu for the pipeline type (set to 'ci'), a confirmation message 'This GitLab CI configuration is valid. Learn more', and three tabs: 'Edit', 'Visualize' (highlighted with a blue box), and 'Lint'. Below the tabs, the pipeline is visualized as a graph with two jobs: 'build-hello-world-job' under the 'Build' stage and 'test-hello-world-job' under the 'Test' stage. The URL at the bottom of the page is https://gitlab.com/kit-scc-scs-course/git-course/ci/editor/branch_name=ci#.

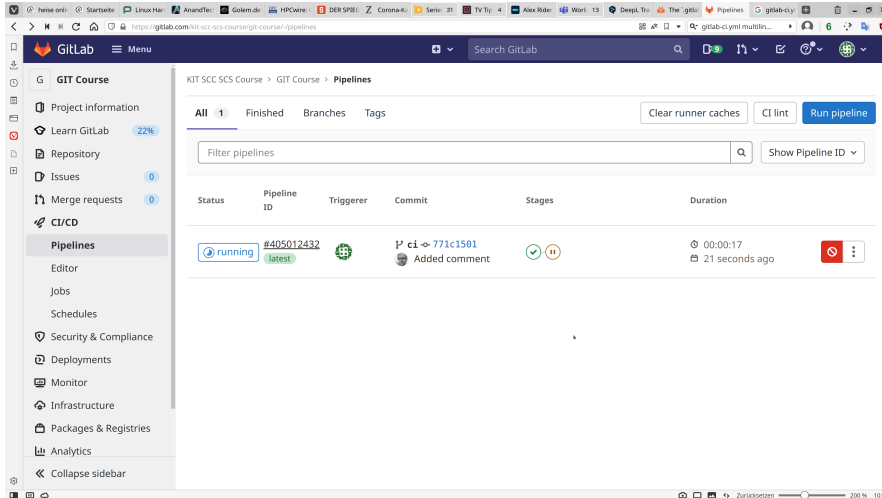
Continuous Integration using GitLab (Lint)



The screenshot shows the GitLab Pipeline Editor interface. The left sidebar contains navigation options like 'GIT Course', 'Project information', 'Learn GitLab', 'Repository', 'Issues', 'Merge requests', 'CI/CD', 'Editor', 'Jobs', 'Schedules', 'Security & Compliance', 'Deployments', 'Monitor', 'Infrastructure', 'Packages & Registries', 'Analytics', 'Wiki', 'Snippets', and 'Collapse sidebar'. The main area displays the pipeline configuration for 'KIT SCC SCS Course > GIT Course > Pipeline Editor'. A dropdown menu shows 'ci v'. A message states: 'This GitLab CI configuration is valid. Learn more'. Below this, there are tabs for 'Edit', 'Visualize', 'Lint' (which is highlighted with a blue box), and 'View merged YAML'. The 'Lint' status is shown as a green checkmark with the text: 'Status: Syntax is correct. CI configuration validated, including all configuration added with the includes keyword. More information'. A table below shows the lint results for two jobs.

Parameter	Value
Build Job - build-hello-world-job	<pre>cc helloWorld.c -o helloWorld</pre> <p>Only policy: branches, tags When: on_success</p>
Test Job - test-hello-world-job	<pre>if [[\$(helloWorld) != "Hello World"]]; then exit 1; fi</pre> <p>Only policy: branches, tags When: on_success</p>

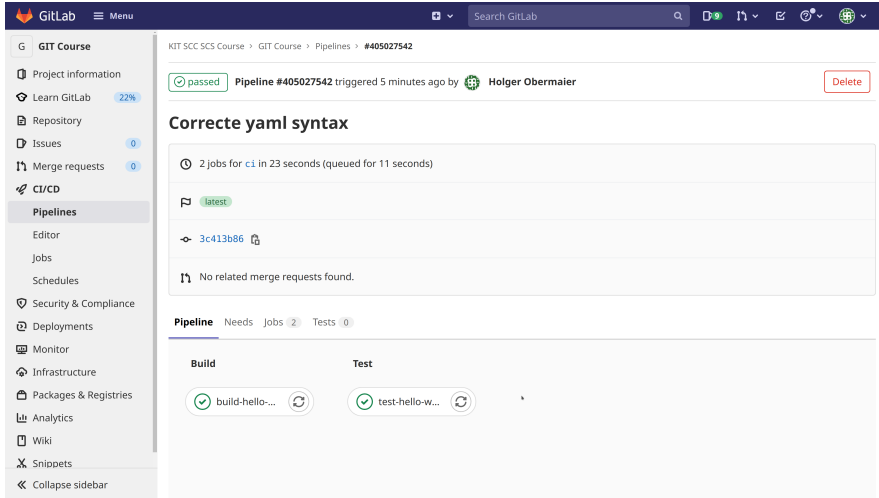
Continuous Integration using GitLab (CI/CD: Pipelines)



The screenshot shows the GitLab web interface for a project named 'GIT Course'. The left sidebar contains navigation options: Project Information, Learn GitLab (22%), Repository, Issues (0), Merge requests (0), CI/CD, Pipelines (selected), Editor, Jobs, Schedules, Security & Compliance, Deployments, Monitor, Infrastructure, Packages & Registries, Analytics, and Collapse sidebar. The main content area displays the 'Pipelines' page for the 'GIT Course' project. It features a filter bar with 'All' (1), 'Finished', 'Branches', and 'Tags' tabs, along with buttons for 'Clear runner caches', 'CI lint', and 'Run pipeline'. Below the filter bar is a search input 'Filter pipelines' and a 'Show Pipeline ID' dropdown. A table lists the pipeline details:

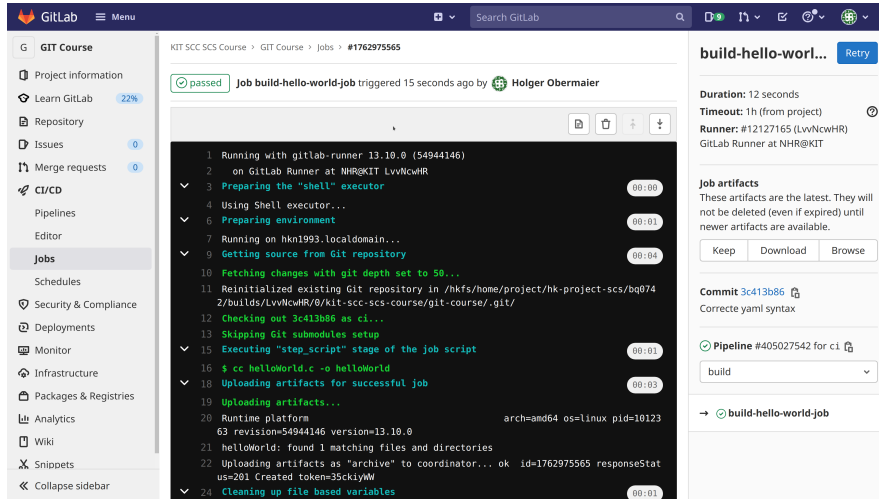
Status	Pipeline ID	Triggerer	Commit	Stages	Duration	
running	#405012432 latest		ci -> 771c1501 Added comment		00:00:17 21 seconds ago	

Continuous Integration using GitLab (Pipeline passed)



The screenshot displays the GitLab web interface for a pipeline. The top navigation bar includes the GitLab logo, a menu, and a search bar. The left sidebar contains navigation options such as 'GIT Course', 'Project information', 'Learn GitLab', 'Repository', 'Issues', 'Merge requests', 'CI/CD', 'Pipelines', 'Editor', 'Jobs', 'Schedules', 'Security & Compliance', 'Deployments', 'Monitor', 'Infrastructure', 'Packages & Registries', 'Analytics', 'Wiki', 'Snippets', and 'Collapse sidebar'. The main content area shows the pipeline details for 'KIT SCC SCS Course > GIT Course > Pipelines > #405027542'. The pipeline status is 'passed', triggered 5 minutes ago by 'Holger Obermaier'. The pipeline title is 'Correcte yaml syntax'. Below the title, it shows '2 jobs for ci in 23 seconds (queued for 11 seconds)'. The 'latest' version is '3c413b86'. There are no related merge requests found. The pipeline summary shows 'Build' and 'Test' stages, both of which are completed successfully, indicated by green checkmarks and circular progress indicators.

Continuous Integration using GitLab (Build Job)



The screenshot displays the GitLab CI/CD interface for a job named "build-hello-world-job". The job status is "passed", triggered 15 seconds ago by Holger Obermaier. The job log shows the following steps:

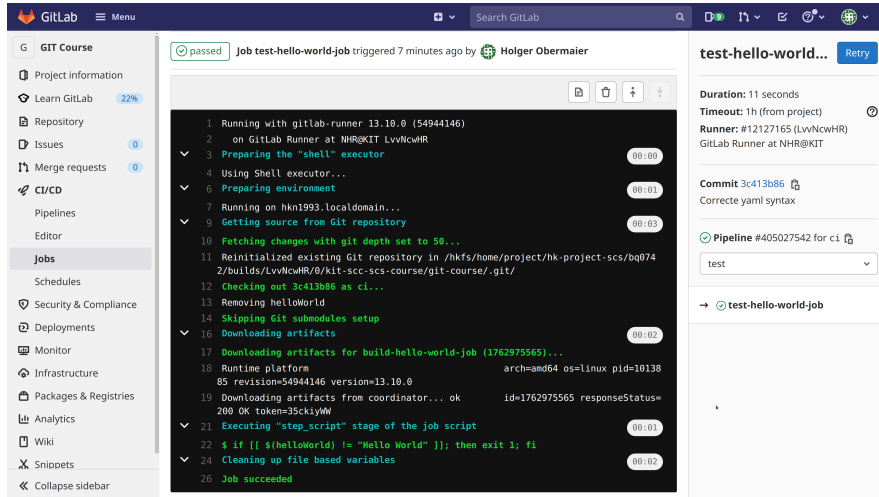
```

1 Running with gitlab-runner 13.10.0 (54944146)
2 on GitLab Runner at NHR@KIT LvNcwHR
3 Preparing the "shell" executor 00:00
4 Using Shell executor...
5
6 Preparing environment 00:01
7 Running on hkn1993.localdomain...
9 Getting source from Git repository 00:04
10 Fetching changes with git depth set to 50...
11 Reinitialized existing Git repository in /hks/home/project/hk-project-scs/bq074
    2/builds/LvNcwHR/0/kit-scc-scs-course/git-course/.git/
12 Checking out 3c413b86 as ci...
13 Skipping Git submodules setup
15 Executing "step_script" stage of the job script 00:01
16 $ cc helloWorld.c -o helloWorld
18 Uploading artifacts for successful job 00:03
19 Uploading artifacts...
20 Runtime platform arch=amd64 os=linux pid=10123
    63 revision=54944146 version=13.10.0
21 helloWorld: found 1 matching files and directories
22 Uploading artifacts as "archive" to coordinator... ok id=1762975565 responseStat
    us=201 Created token=35ckiywW
24 Cleaning up file based variables 00:01
  
```

On the right side, the job details are shown:

- build-hello-worl...** [Retry](#)
- Duration: 12 seconds
- Timeout: 1h (from project)
- Runner: #12127165 (LvNcwHR)
- GitLab Runner at NHR@KIT
- Job artifacts**: These artifacts are the latest. They will not be deleted (even if expired) until newer artifacts are available.
 - [Keep](#) [Download](#) [Browse](#)
- Commit: [3c413b86](#) [🔗](#)
- Corrected yaml syntax
- [🟢 Pipeline #405027542 for ci](#) [🔗](#)
- build [▼](#)
- [🟢 build-hello-world-job](#)

Continuous Integration using GitLab (Test Job)



The screenshot displays the GitLab CI/CD interface for a job named "test-hello-world-job". The job status is "passed" and was triggered 7 minutes ago by Holger Obermaier. The interface is divided into three main sections:

- Left Sidebar:** A navigation menu with options like "Project information", "Learn GitLab", "Repository", "Issues", "Merge requests", "CI/CD", "Pipelines", "Editor", "Jobs", "Schedules", "Security & Compliance", "Deployments", "Monitor", "Infrastructure", "Packages & Registries", "Analytics", "Wiki", "Snippets", and "Collapse sidebar".
- Central Panel:** A log viewer showing the execution steps of the job. The steps include:
 - Running with gitlab-runner 13.10.0 (54944146)
 - on GitLab Runner at NHR@KIT LvNcwHR
 - Preparing the "shell" executor (00:00)
 - Using Shell executor...
 - Preparing environment (00:01)
 - Running on hkn1993.localdomain...
 - Getting source from Git repository (00:03)
 - Fetching changes with git depth set to 50...
 - Reinitialized existing Git repository in /hkfs/home/project/hk-project-scs/bq0742/builds/LvNcwHR/0/kit-scc-scs-course/git-course/.git/
 - Checking out 3c413b86 as ci...
 - Removing helloWorld
 - Skipping Git submodules setup
 - Downloading artifacts (00:02)
 - Downloading artifacts for build-hello-world-job (1762975565)...
 - Runtime platform: arch=amd64 os=linux pid=10138 85 revision=54944146 version=13.10.0
 - Downloading artifacts from coordinator... ok id=1762975565 responseStatus=200 OK token=35ckiywW
 - Executing "step_script" stage of the job script (00:01)
 - Shell script execution: `$ if [[$(helloWorld) != "Hello World"]]; then exit 1; fi`
 - Cleaning up file based variables (00:02)
 - Job succeeded
- Right Panel:** Job details including:
 - Duration: 11 seconds
 - Timeout: 1h (from project)
 - Runner: #12127165 (LvNcwHR) GitLab Runner at NHR@KIT
 - Commit: 3c413b86 (Corrected yaml syntax)
 - Pipeline: #405027542 for ci
 - Job name: test-hello-world-job