

Introduction to Continuous-X services at NHR@KIT

Holger Obermaier | 10. November 2021



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 (continued)

Advantages using CX

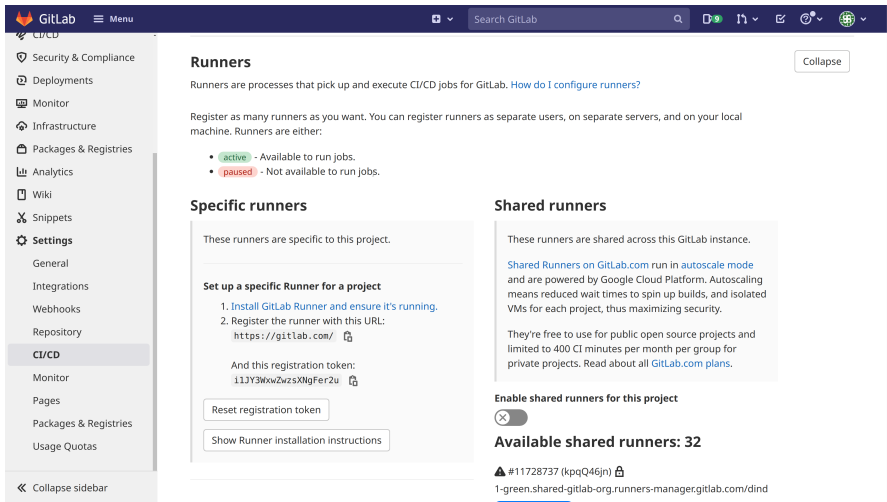
- Does it compile with different compilers, MPIs,...?
- Does it compile on different hardware?
- Do my unit tests pass?
- Do my unit tests cover most of the code?
- Linting
- Automated deployment / packaging (for varying compilers, MPIs, hardware)
- Standardized Environment in conjunction with containers

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



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. It explains that runners are processes for CI/CD jobs and lists two states: 'active' (available) and 'paused' (not available). Below this, there are two sections: 'Specific runners' and 'Shared runners'. The 'Specific runners' section provides instructions on how to set up a runner for a project, including a registration URL and a registration token. The 'Shared runners' section explains that they are shared across the instance and are powered by Google Cloud Platform. At the bottom, there is a toggle for 'Enable shared runners for this project' (which is turned off) and a list of available shared runners, showing 32 runners with details like 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

- Install [GitLab Runner](#) and ensure it's running.
- 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 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

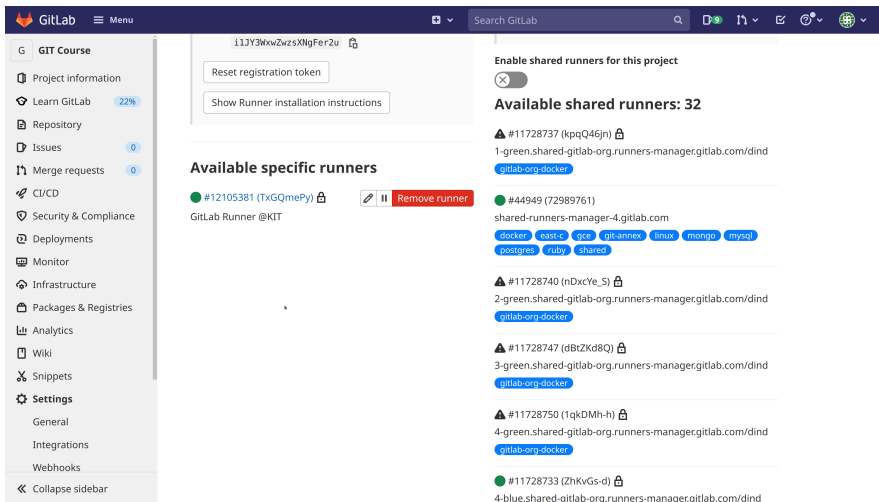
```
gitlab-runner run
```


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



The screenshot shows the GitLab web interface for configuring CI/CD runners. The left sidebar contains navigation options like 'GIT Course', 'Project information', 'Learn GitLab', 'Repository', 'Issues', 'Merge requests', 'CI/CD', 'Security & Compliance', 'Deployments', 'Monitor', 'Infrastructure', 'Packages & Registries', 'Analytics', 'Wiki', 'Snippets', and 'Settings'. The 'Settings' menu is expanded to show 'General', 'Integrations', 'Webhooks', and 'Collapse sidebar'. The main content area is titled 'Available specific runners' and shows a single runner: a green dot with ID #12105381 (TxGQmePy) and name 'GitLab Runner @KIT'. There are edit and pause icons, and a red 'Remove runner' button. Above this, there are buttons for 'Reset registration token' and 'Show Runner installation instructions'. To the right, the 'Enable shared runners for this project' toggle is turned off. Below that, it says 'Available shared runners: 32'. A list of shared runners follows, each with a warning icon, ID, name, and tags. The runners are: 1-green.shared-gitlab-org.runners-manager.gitlab.com/dind (tags: gitlab-org-docker), 44949 (72989761) shared-runners-manager-4.gitlab.com (tags: docker, east-e, gce, git-annex, linux, mongo, mysql, postgres, ruby, shared), 2-green.shared-gitlab-org.runners-manager.gitlab.com/dind (tags: gitlab-org-docker), 3-green.shared-gitlab-org.runners-manager.gitlab.com/dind (tags: gitlab-org-docker), 4-green.shared-gitlab-org.runners-manager.gitlab.com/dind (tags: gitlab-org-docker), and 11728733 (ZhKvGs-d) (tags: 4-blue.shared-gitlab-org.runners-manager.gitlab.com/dind).

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 (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:

```
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 ⇒ 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

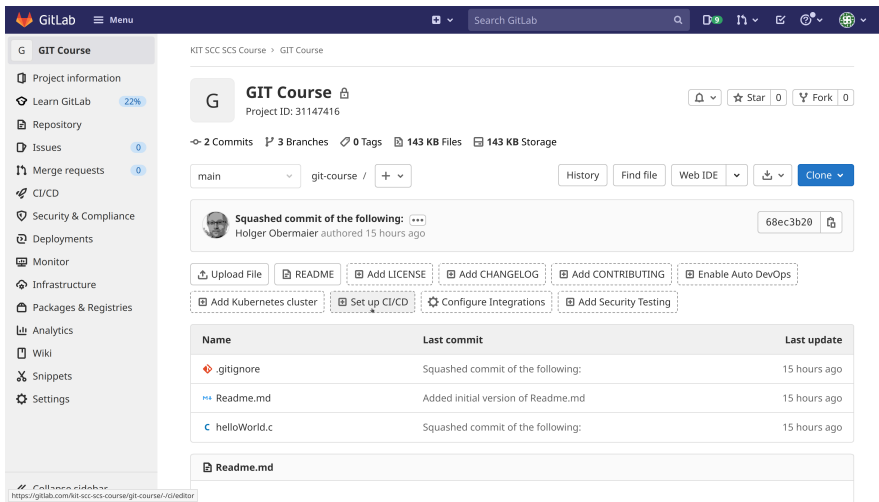
Continuous Integration using GitLab

- 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)



The screenshot shows the GitLab web interface for a project named "GIT Course". The left sidebar contains navigation options such as "Project information", "Learn GitLab", "Repository", "Issues", "Merge requests", "CI/CD", "Security & Compliance", "Deployments", "Monitor", "Infrastructure", "Packages & Registries", "Analytics", "Wiki", "Snippets", and "Settings". The main content area displays the project details, including the project ID (31147416), commit and branch statistics, and a list of files. A "Squashed commit of the following" section shows a commit by Holger Obermaier. Below this, there are buttons for "Upload File", "README", "Add LICENSE", "Add CHANGELOG", "Add CONTRIBUTING", "Enable Auto DevOps", "Add Kubernetes cluster", "Set up CI/CD", "Configure Integrations", and "Add Security Testing". A table lists the files in the repository, including ".gitignore", "Readme.md", and "helloWorld.c".

Project Information: GIT Course (Project ID: 31147416)

Statistics: 2 Commits, 3 Branches, 0 Tags, 143 KB Files, 143 KB Storage

Current Branch: main (Path: git-course / +)

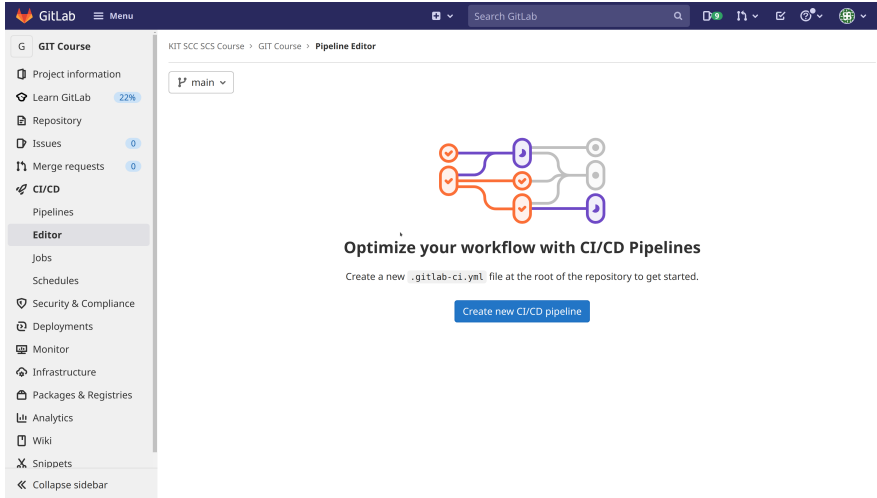
Recent Commit: Squashed commit of the following: 68ec3b20 (Holger Obermaier, 15 hours ago)

Actions: 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

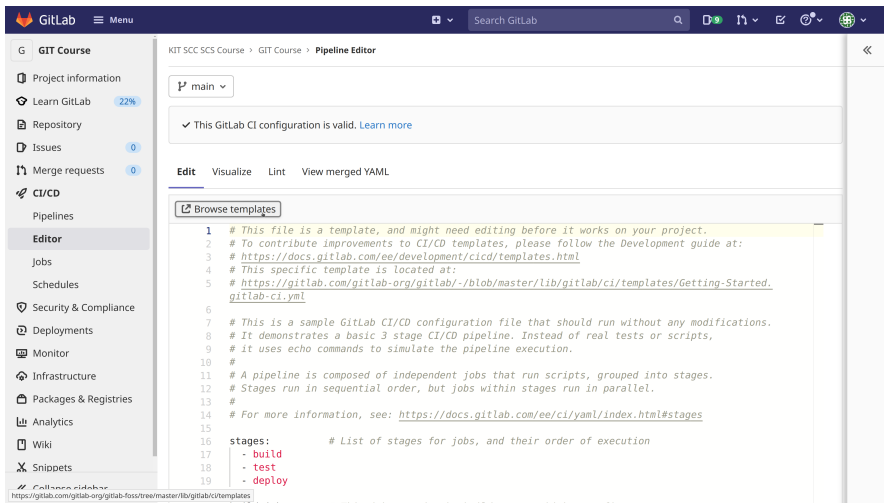
Files: [Readme.md](#)

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 project 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.

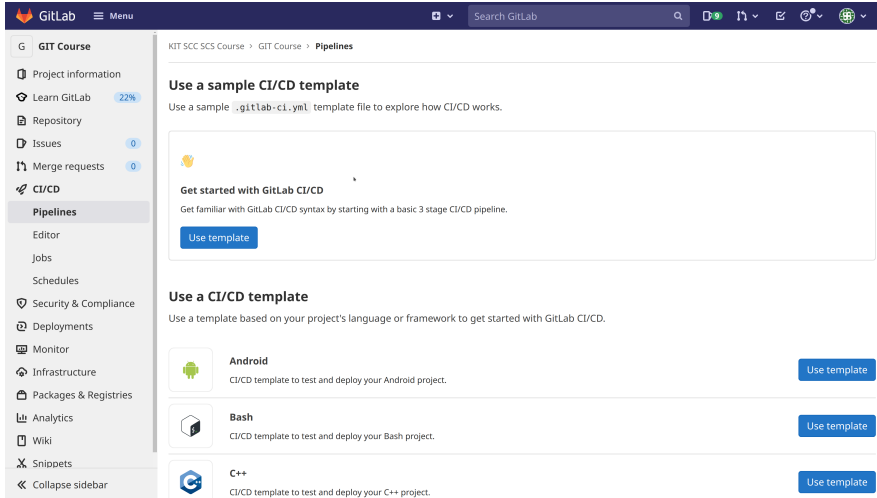
Continuous Integration using GitLab (Browse Templates)



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 the 'main' branch. 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 a code editor displays a sample CI configuration file with the following content:

```
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

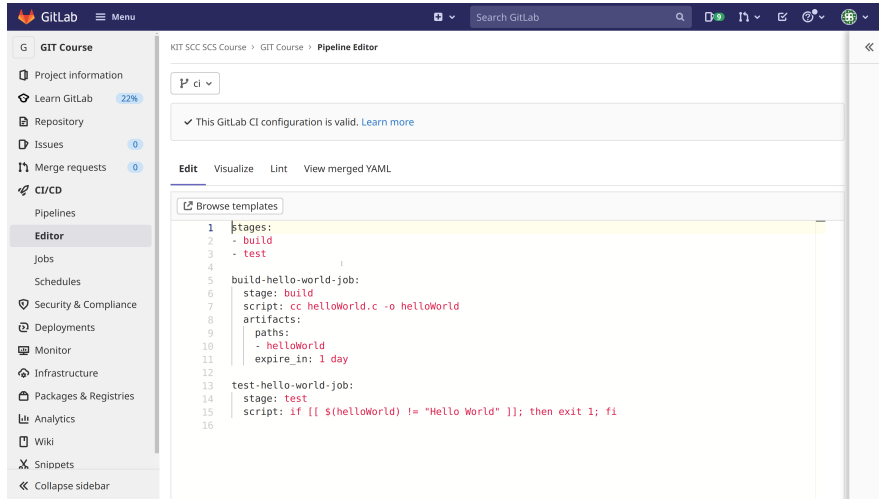
test-code-job1:
  stage: test
  script: if [[ $(helloWorld) != "Hello World" ]]; then exit 1; fi
```

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
```

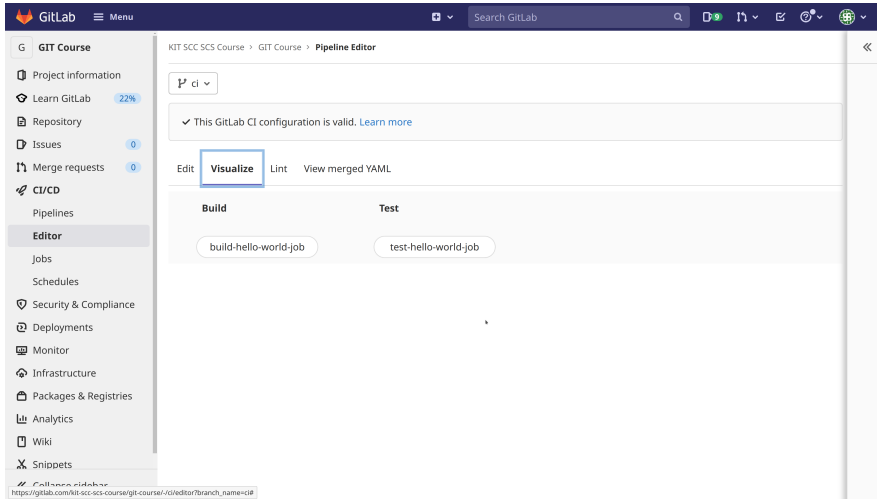
Continuous Integration using GitLab (Edit)



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 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. The main editor shows a YAML configuration for a pipeline with two stages: 'build' and 'test'. The 'build' stage includes a job named 'build-hello-world-job' with a script to compile a C program and generate an artifact. The 'test' stage includes a job named 'test-hello-world-job' with a script to verify the output of the build.

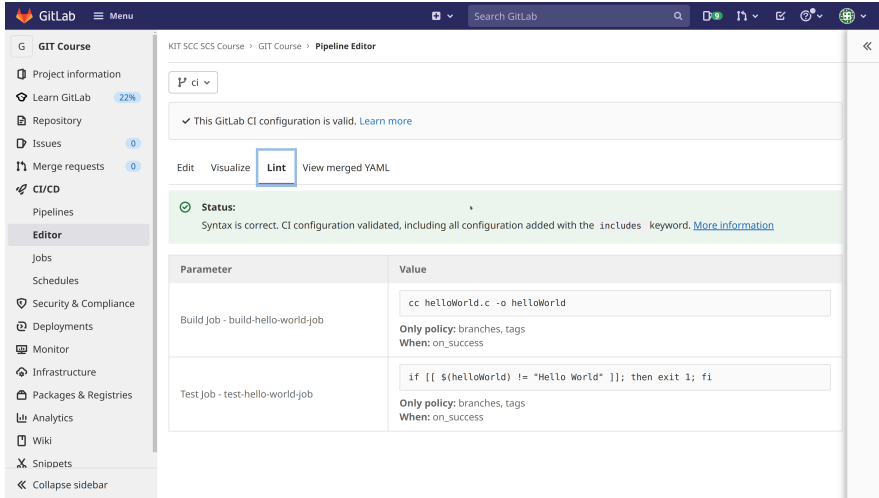
```
1 stages:
2   - build
3   - test
4
5 build-hello-world-job:
6   stage: build
7   script: cc helloWorld.c -o helloWorld
8   artifacts:
9     paths:
10      - helloWorld
11     expire_in: 1 day
12
13 test-hello-world-job:
14   stage: test
15   script: if [[ $(helloWorld) != "Hello World" ]]; then exit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

Continuous Integration using GitLab (Visualize)



The screenshot displays the GitLab Pipeline Editor interface. The top navigation bar includes the GitLab logo, a menu icon, and a search bar. The left sidebar shows the project navigation menu with 'CI/CD' selected, and 'Editor' highlighted under the 'Pipelines' section. The main content area shows the pipeline configuration for 'KIT SCC SCS Course > GIT Course > Pipeline Editor'. A message indicates that the GitLab CI configuration is valid. Below this, there are three tabs: 'Edit', 'Visualize', and 'Lint', with 'Visualize' selected. The 'Visualize' tab shows a visual representation of the pipeline with two jobs: 'build-hello-world-job' and 'test-hello-world-job'. The jobs are arranged in a sequence, with 'build-hello-world-job' preceding 'test-hello-world-job'. The URL at the bottom of the page is https://gitlab.com/kit-scc-scs-course/git-course/jci/editor?branch_name=cif#.

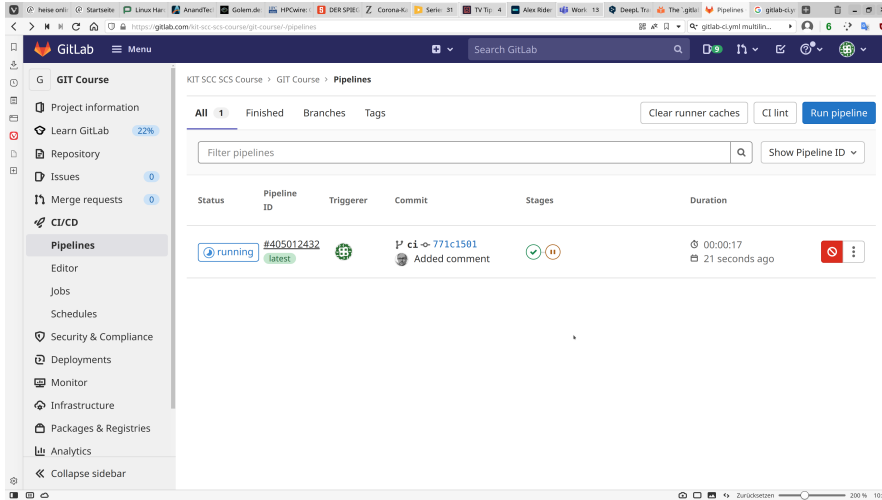
Continuous Integration using GitLab (Lint)



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 content area is titled 'KIT SCC SCS Course > GIT Course > Pipeline Editor'. It features a search bar, a dropdown menu, and a confirmation message: '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'. A green status bar indicates 'Status: Syntax is correct. CI configuration validated, including all configuration added with the includes keyword. More information'. At the bottom, a table lists pipeline jobs with their parameters and values.

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>

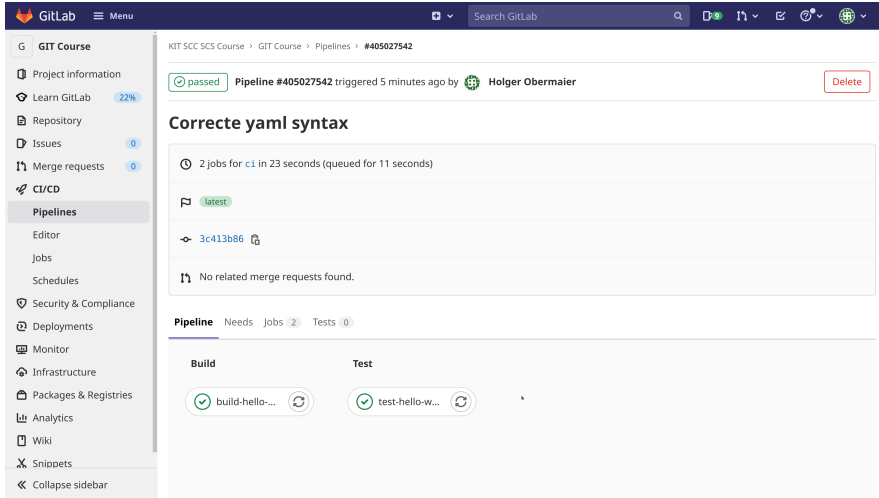
Continuous Integration using GitLab (CI/CD: Pipelines)



The screenshot shows the GitLab web interface for a project named 'GIT Course'. The left sidebar contains navigation options such as 'Project Information', 'Learn GitLab', 'Repository', 'Issues', 'Merge requests', 'CI/CD', and 'Pipelines'. The main content area displays the 'Pipelines' section for the 'GIT Course' project. It includes tabs for 'All', 'Finished', 'Branches', and 'Tags'. A search bar is present with the text 'Filter pipelines'. Below this, a table lists pipeline runs. The first entry is a pipeline with ID '#405012432' in the 'latest' branch, triggered by a commit 'ci -> 771c1501' with the message 'Added comment'. The pipeline status is 'running', and it has completed 2 stages. The duration is 00:00:17, and it was last updated 21 seconds ago. Action buttons for 'Clear runner caches', 'CI lint', and 'Run pipeline' are visible at the top right of the pipeline list.

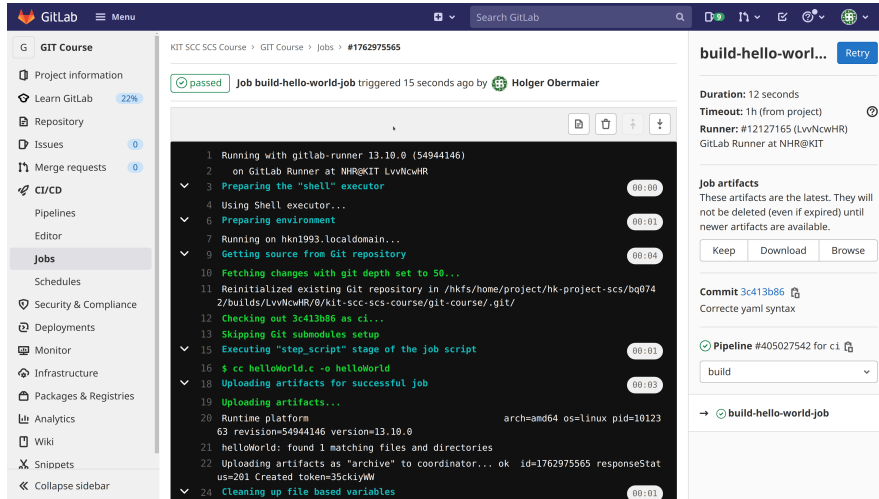
Status	Pipeline ID	Triggerer	Commit	Stages	Duration
running	#405012432 latest		ci -> 771c1501 Added comment		00:00:17 21 seconds ago

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' (22%), 'Repository', 'Issues' (0), 'Merge requests' (0), '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' tag is visible, along with the commit ID '3c413b86'. A message states 'No related merge requests found.' At the bottom, the pipeline summary shows 'Build' and 'Test' stages, both with 'passed' status and refresh icons.

Continuous Integration using GitLab (Build Job)



The screenshot displays the GitLab CI/CD interface for a project named "KIT SCC SCS Course". The main view shows a job named "build-hello-world-job" that has successfully completed, triggered 15 seconds ago by user "Holger Obermaier".


Job Details:


- Status: passed
- 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.

Buttons: Keep, Download, Browse

Commit: 3c413b86  Corrected yaml syntax

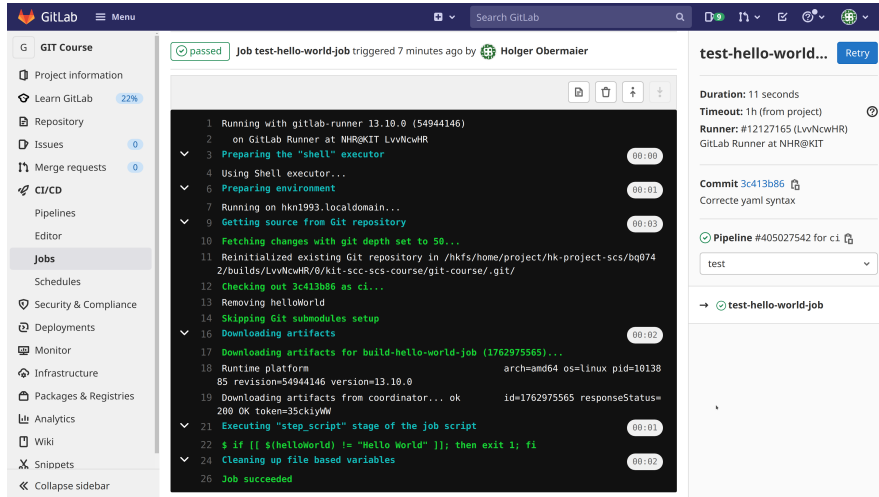
Pipeline: #405027542 for ci  build

Job Log:

```

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...
8
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
12 2/builds/LvNcwHR/0/kit-scc-scs-course/git-course/.git/
13 Checking out 3c413b86 as ci...
14 Skipping Git submodules setup
15 Executing "step_script" stage of the job script 00:01
16 $ cc helloWorld.c -o helloWorld
17
18 Uploading artifacts for successful job 00:03
19 Uploading artifacts...
20 Runtime platform arch=amd64 os=linux pid=10123
21 63 revision=54944146 version=13.10.0
22 helloWorld: found 1 matching files and directories
23 Uploading artifacts as "archive" to coordinator... ok id=1762975565 responseStat
24 us=201 Created token=35ckiywW
25
26 Cleaning up file based variables 00:01
  
```

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 "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".
- 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 with the message "Correcte yaml syntax"
 - Pipeline #405027542 for ci
 - Job name: test-hello-world-job