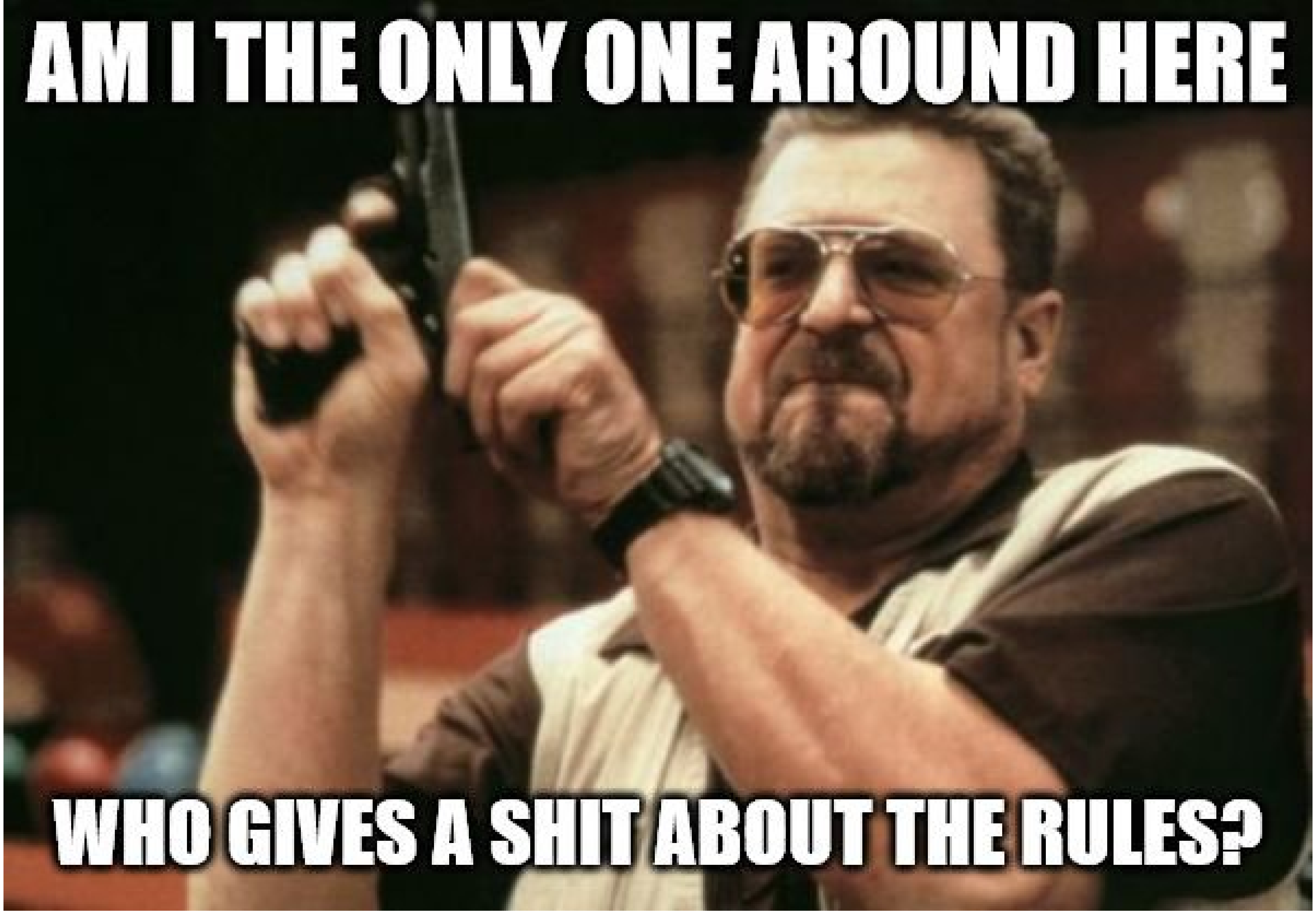
Bessere TYPO3 Projekte durch Linting und Code Analyse mit GitLab CI

Felix Heller

Linting und Code Analyse

Ziele

- Einhaltung von Coding Standards (z. B. PSR-2 für PHP), dadurch bessere Lesbarkeit und Einheitlichkeit des Codes
- Höhere Qualität des Quellcodes:
  - Weniger (Flüchtigkeits-)Fehler
  - Besserer Aufbau des Quellcodes (z. B. keine "Gott-Klassen")
  - Korrektes Verhalten über Tests mit PHPUnit prüfbar
- Automatische Prüfung und automatisches Genörgel 🤬


AM I THE ONLY ONE AROUND HERE
WHO GIVES A SHIT ABOUT THE RULES?

GitLab CI (Continuous Integration)

Ersteinrichtung

- Installation von mindestens einer GitLab-Runner-Instanz (Zeitaufwand ca. 10 Min. / https://docs.gitlab.com/runner/install/)

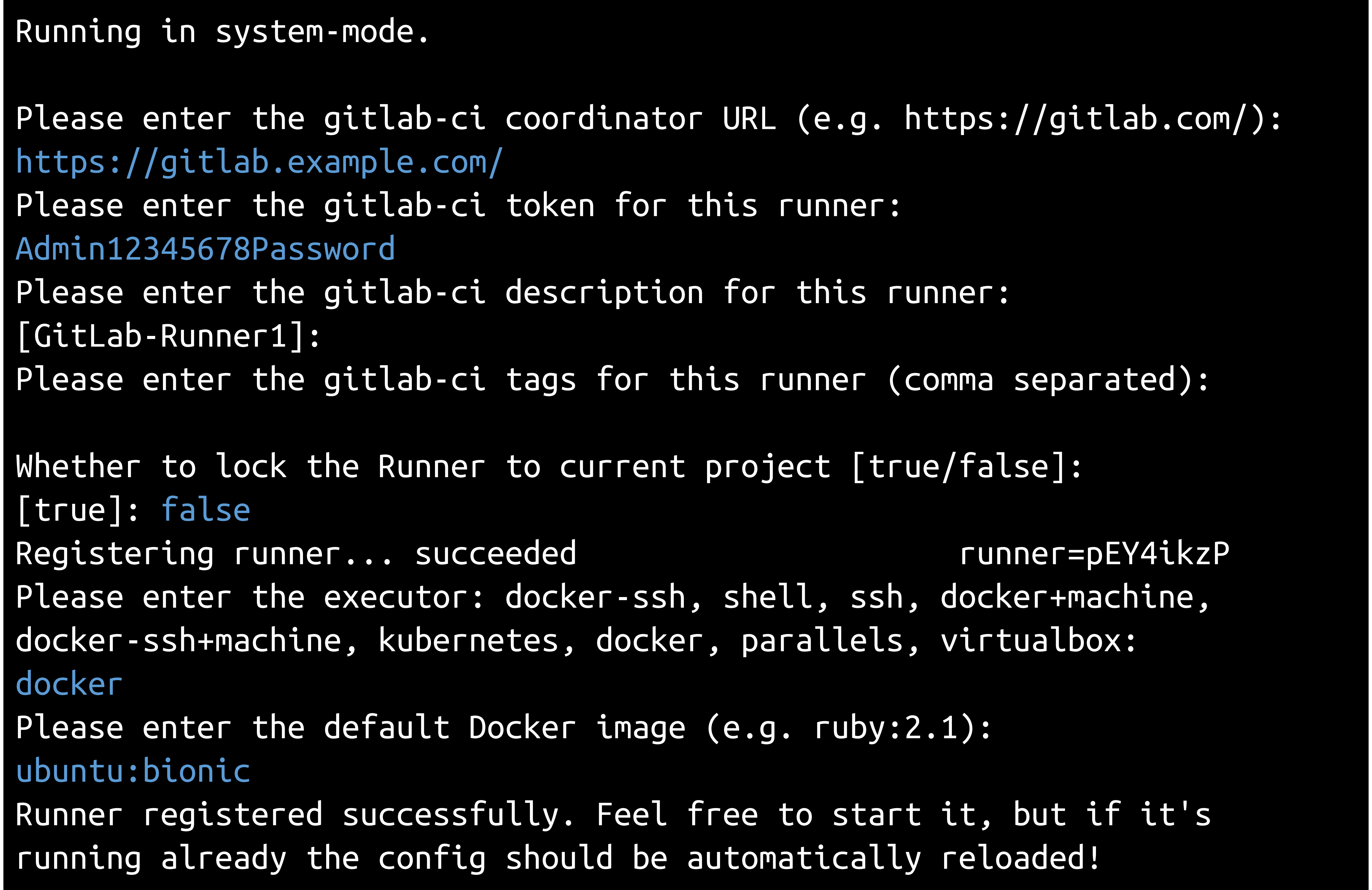
```
curl -L
https://packages.gitlab.com/install/repositories/runner/gitlab-
runner/script.deb.sh | sudo bash
apt-get install gitlab-runner

apt install docker.io
```

GitLab CI (Continuous Integration)

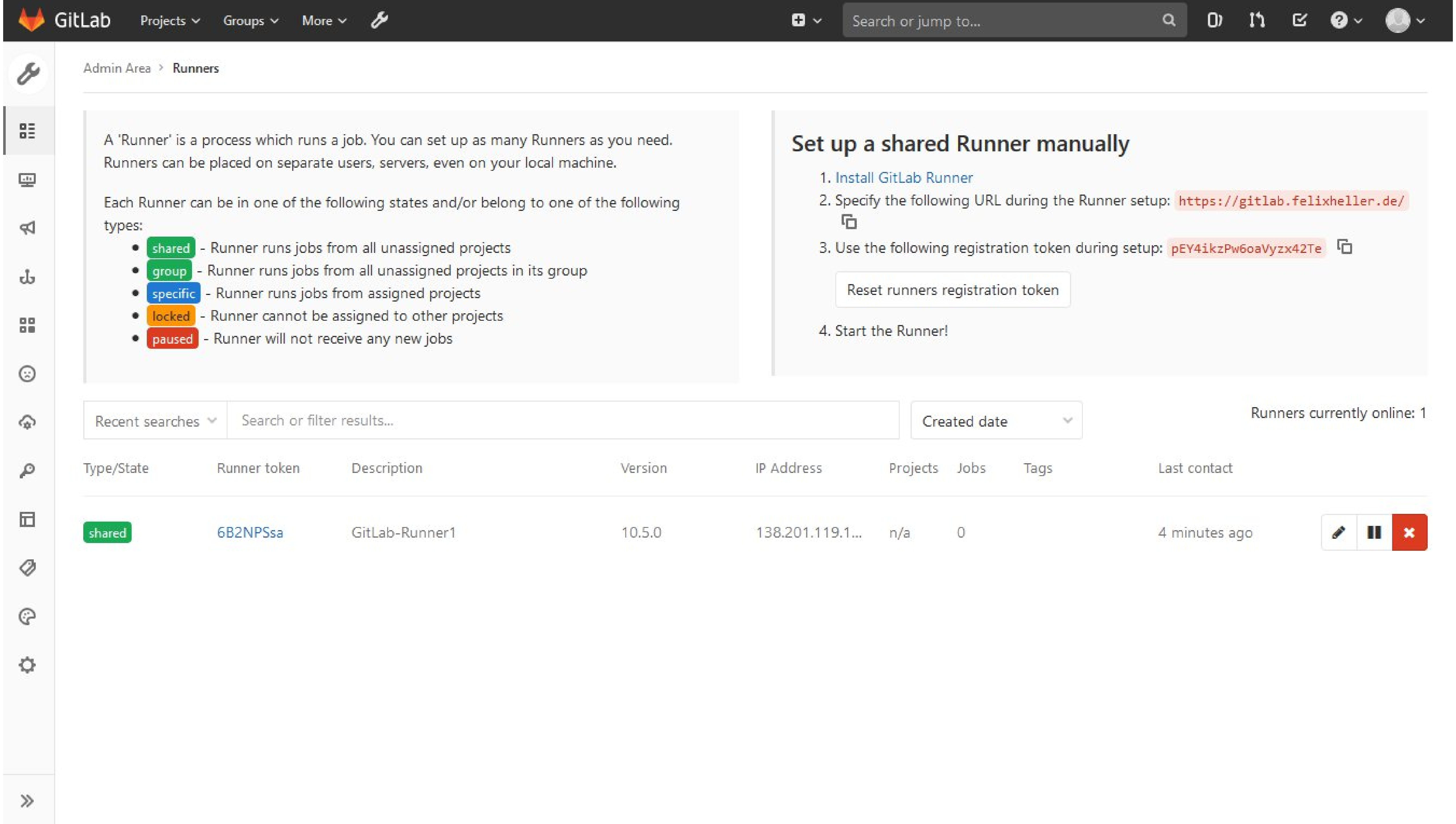
Ersteinrichtung

- Verbindung mit GitLab, d. h. Registrierung des GitLab-Runner (https://docs.gitlab.com/runner/register/)

```
gitlab-runner register
```



```
Running in system-mode.

Please enter the gitlab-ci coordinator URL (e.g. https://gitlab.com/):
https://gitlab.example.com/
Please enter the gitlab-ci token for this runner:
Admin12345678Password
Please enter the gitlab-ci description for this runner:
[GitLab-Runner1]:
Please enter the gitlab-ci tags for this runner (comma separated):

Whether to lock the Runner to current project [true/false]:
[true]: false
Registering runner... succeeded                     runner=pEY4ikzP
Please enter the executor: docker-ssh, shell, ssh, docker+machine,
docker-ssh+machine, kubernetes, docker, parallels, virtualbox:
docker
Please enter the default Docker image (e.g. ruby:2.1):
ubuntu:bionic
Runner registered successfully. Feel free to start it, but if it's
running already the config should be automatically reloaded!
```



Admin Area › Runners

A 'Runner' is a process which runs a job. You can set up as many Runners as you need. Runners can be placed on separate users, servers, even on your local machine.

Each Runner can be in one of the following states and/or belong to one of the following types:

- shared - Runner runs jobs from all unassigned projects
- group - Runner runs jobs from all unassigned projects in its group
- specific - Runner runs jobs from assigned projects
- locked - Runner cannot be assigned to other projects
- paused - Runner will not receive any new jobs

Set up a shared Runner manually

1. Install GitLab Runner
2. Specify the following URL during the Runner setup: `https://gitlab.felixheller.de/`
3. Use the following registration token during setup: `pEY4ikzPw6oaVyzx42Te`

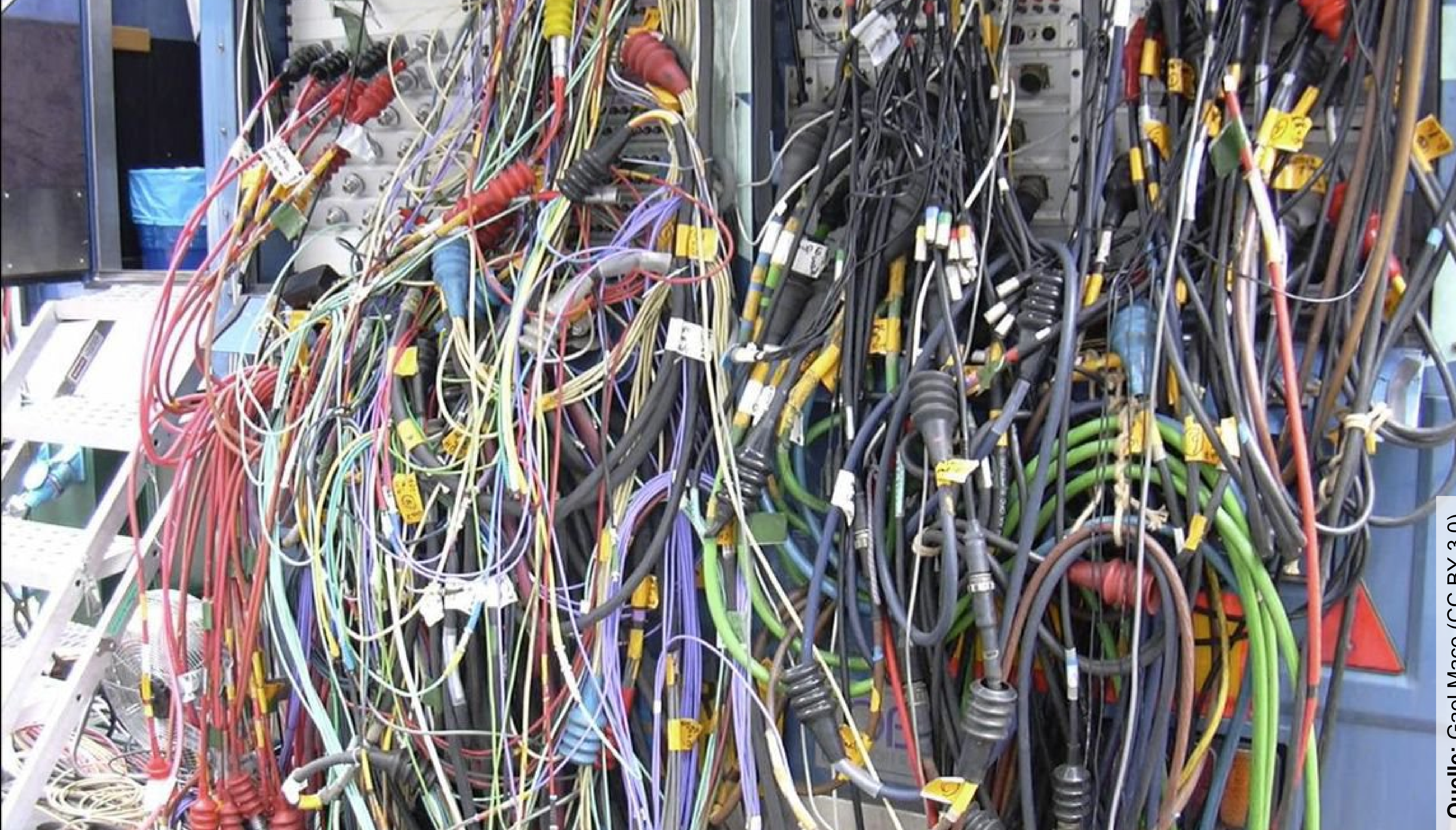
   Reset runners registration token

4. Start the Runner!

Recent searches | Search or filter results... | Created date

Runners currently online: 1

Type/State	Runner token	Description	Version	IP Address	Projects	Jobs	Tags	Last contact
shared	6B2NPSsa	GitLab-Runner1	10.5.0	138.201.119.1...	n/a	0		4 minutes ago



Quelle: Gael Mace (CC BY 3.0)

GitLab CI (Continuous Integration)

Verwendung

- Konfiguration über eine simple YAML-Datei `.gitlab-ci.yml`
- Automatisches Ausführen von Shell-Skripten
  - ...nach jedem Commit Push
  - ...nach Änderung bestimmter Dateien (`packages/**/*.php`)
  - ...in bestimmten Branches (`release/*`)
- Shell-Skripte laufen in Docker-Umgebung (o. ä.)
- Beliebige Linux-Befehle ausführbar (`php, node, rsync, ...`)

Linting und Code Analyse

Beispiel TypoScript Linter (Konfiguration `typoscript-lint.yml`)

```
composer require --dev helmich/typo3-typoscript-lint
```

```
paths:
  - packages/
filePatterns:
  - "*.typoscript"
sniffs:
  - class: DeadCode
    disabled: true
  - class: RepeatingRValue
    disabled: true
```

Linting und Code Analyse

Beispiel PHP Static Analysis (Konfiguration `phpstan.neon`)

```
composer require --dev phpstan/phpstan
```

```
parameters:
      level: 8
      ignoreErrors:
            - '/Constant TYPO3_MODE not found./'
            - '/Undefined variable: \$_EXTKEY/'
            - '/Call to an undefined( static)? method
(.*?)Repository::countBy(.*?)\(\)./'
            - '/Call to an undefined( static)? method
(.*?)Repository::find(One)?By(.*?)\(\)./'
      reportUnmatchedIgnoredErrors: false
```

Linting und Code Analyse

Beispiel TypeScript Linter (Konfiguration: `tslint.json`)

```
npm install --dev eslint @typescript-eslint/eslint-plugin
@typescript-eslint/parser                              # image: "node:12.24"
```

```
{
    "extends": "tslint:recommended",
    "rules": {
        "no-console": {
            "severity": "warning"
        },
        "no-var-requires": false
    }
}
```

Linting und Code Analyse

Beispiel YAML Linter (ohne extra Konfiguration)

```
pip install yamllint                                   # image: "python:2.7"
yamllint -d relaxed .
```

Dank an Sanjay Chauhan (NITSAN)
für den Vortrag auf den TYPO3 Developer Days 2019

GitLab CI (Continuous Integration)

```yaml
stages:
  - lint
  - check

phpstan:
  image: "gitlab.example.com:5050/dockerfiles/runner-php"     # "php:7.2"
  stage: check
  only:
    changes:
      - packages/**/*.php
  script:
    - composer install
    - composer require --dev phpstan/phpstan
    - ./vendor/bin/phpstan analyze ./packages/
```

$this->startLiveDemonstration();

https://bit.ly/2SXJGmE