dataport

anderScore
trust in competence

Tag 2: Vertiefung Git-Workflow, CI/CD & GitLab CI

18.06.2024, Daniel Krämer & Malte Fischer

Agenda

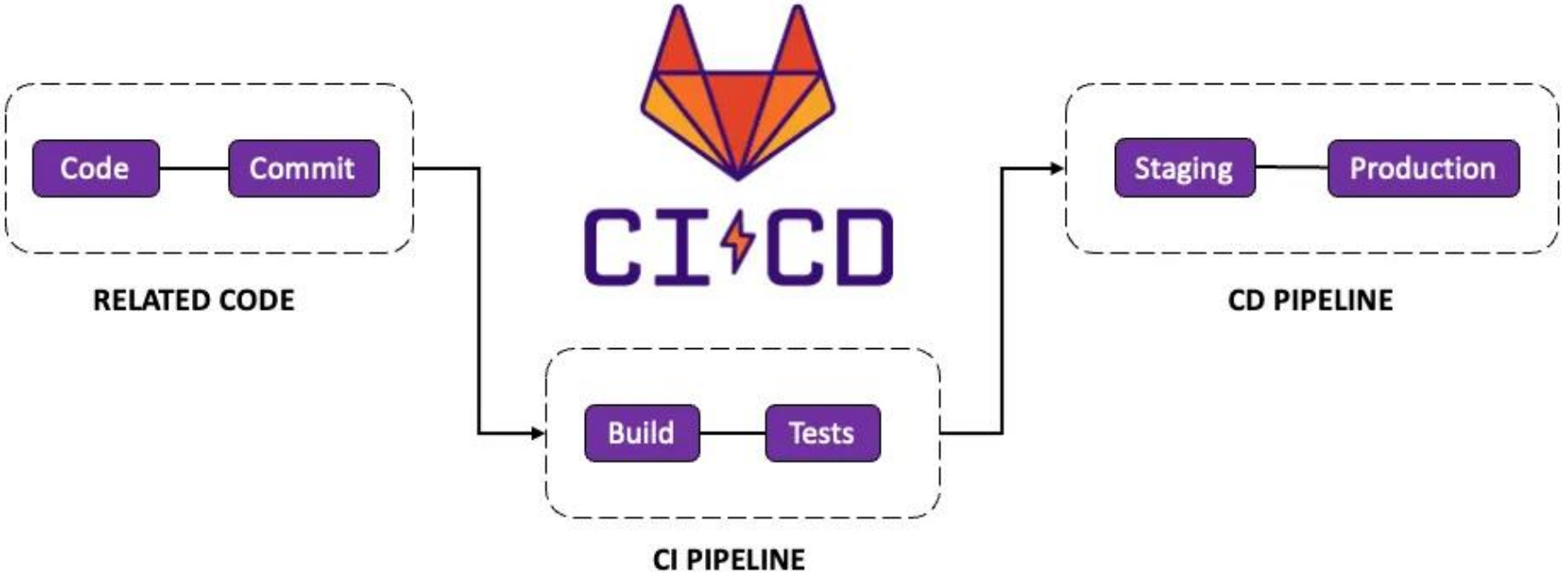
- **Tag 1 – Einführung in Git und GitLab, Git-Workflow im Team**
  - Einführung & Kursüberblick
  - Grundlagen von Git
  - Git Rebase und Merge-Strategien
  - Git Remote
  - Grundlagen von GitLab
  - Git-Workflow im Team
- **Tag 2 – Vertiefung Git-Workflow, CI/CD & GitLab CI**
  - Gitflow-Workflow
  - Tags, Releases & deren Verwaltung
  - GitLab-Runner
  - Einführung in GitLab CI/CD & gitlab.yml
- **Tag 3 – GitOps, Docker in der Entwicklung und Deployment-Strategien**
  - GitOps Grundlagen
  - Lokale Entwicklung mit Docker
  - Container/Docker-Registry
  - Erstellen von Release- und Tagged-Images
  - Möglichkeiten des Deployments & Verwaltung von Konfiguration
  - Abschlussübung & Diskussion

Agenda

- **Tag 1 – Einführung in Git und GitLab, Git-Workflow im Team**
  - Einführung & Kursüberblick
  - Grundlagen von Git
  - Git Rebase und Merge-Strategien
  - Git Remote
  - Grundlagen von GitLab
  - Git-Workflow im Team
- **Tag 2 – Vertiefung Git-Workflow, CI/CD & GitLab CI**
  - Gitflow-Workflow
  - Tags, Releases & deren Verwaltung
  - GitLab-Runner
  - Einführung in GitLab CI/CD & gitlab.yml
- **Tag 3 – GitOps, Docker in der Entwicklung und Deployment-Strategien**
  - GitOps Grundlagen
  - Lokale Entwicklung mit Docker
  - Container/Docker-Registry
  - Erstellen von Release- und Tagged-Images
  - Möglichkeiten des Deployments & Verwaltung von Konfiguration
  - Abschlussübung & Diskussion

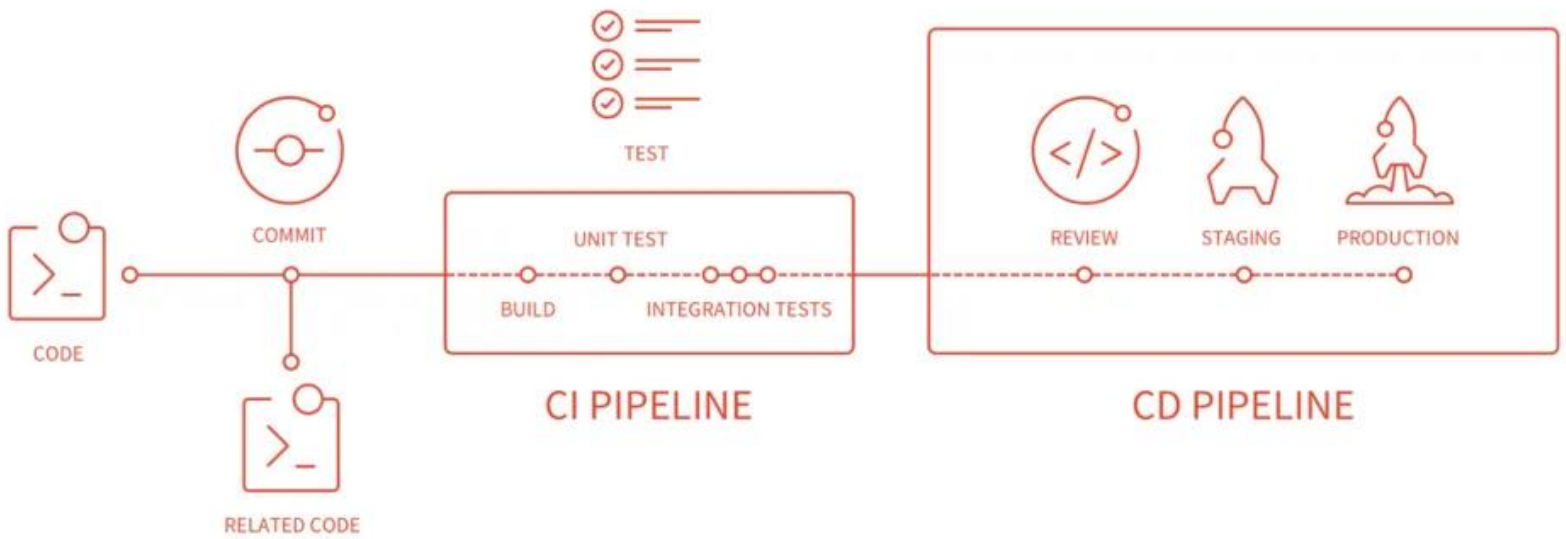
Grundlagen von

GitLab Runner



Quelle: https://medium.com/@truongbui95/exploring-gitlab-ci-cd-ce6a7ffb5746

GitLab CI



Quelle: https://medium.com/@mosiko1234/optimizing-gitlab-ci-cd-pipelines-for-high-efficiency-f2ebbc046a89

GitLab Runner

Basics

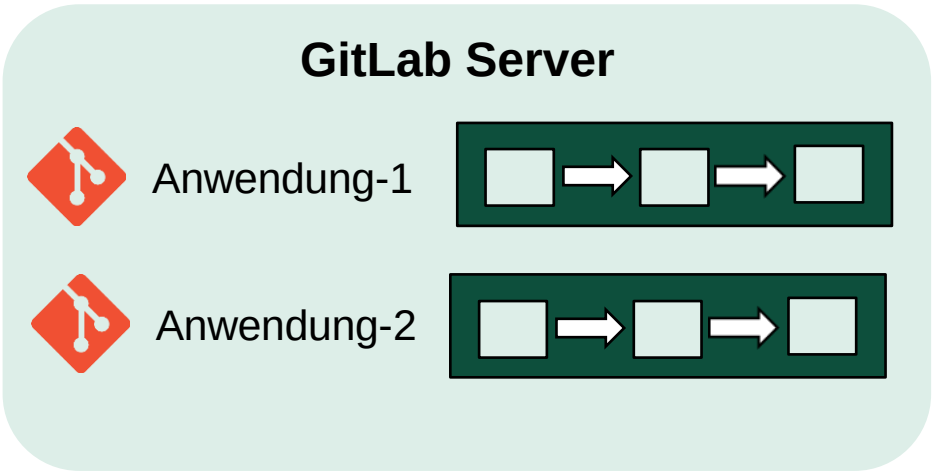
- GitLab Runner arbeiten mit GitLab CI/CD zusammen
  - … um Aufträge (engl. jobs) in einer Pipeline auszuführen
- Zwei Varianten
  1. GitLab-hosted Runners
  2. Self-managed Runners

- GitLab-hosted Runners
  - GitLab.com oder „GitLab Dedicated“* → verwaltet durch GitLab
  - Bei default für alle Projekte enabled
- **Self-managed Runners**
  - GitLab Runner auf Infrastruktur installieren
  - Im Anschluss im GitLab registrieren

*GitLab Enterprise DevSecOps Platform as a single-tenant SaaS deployment

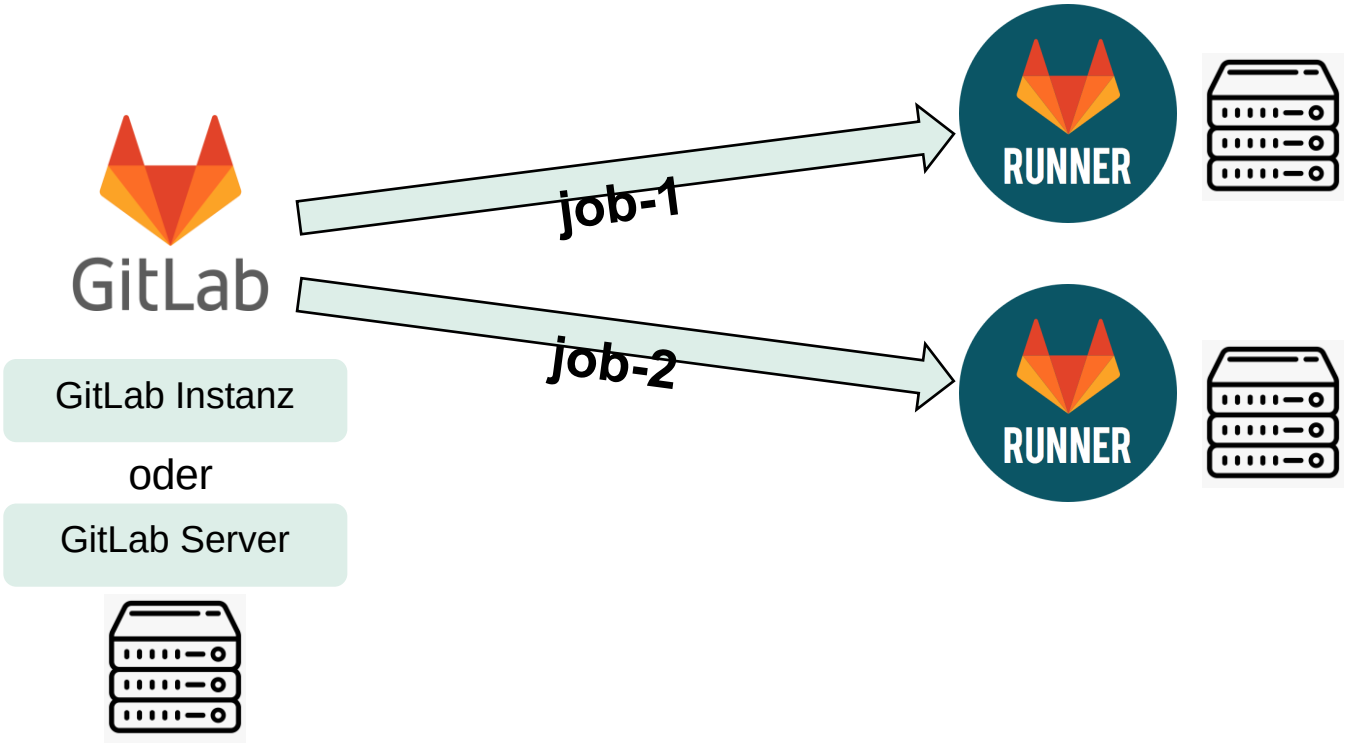
GitLab Architektur

- GitLab
  - Enthält Anwendungscode und Pipeline-Konfiguration
  - Weitere GitLab-Konfigurationen
  - Verwaltet die Pipeline-Ausführungen



GitLab Architektur

- GitLab-Runners
  - Agents führen CI/CD jobs aus
  - Pipeline jobs an Runner durch GitLab



Self-managed Runner

Self-managed Runner

- Eigenen Project Runner benutzen
- Verschiedene Runner verwalten
- Runner registrieren
- Executors
- Runner konfigurieren

GitLab und GitLab Runner Versionen

- MAJOR.MINOR sync: GitLab <-> GitLab Runner
- Rückwärtskompatibilität
  - Bei MINOR gegeben
  - aber neue GitLab-Features beachten!
- Falls GitLab.com genutzt wird
  - Runner immer updaten

Live-Demo: Eigenen Project Runner benutzen

- GitLab Runner installieren
- Neues Projekt erstellen
- Projekt-Pipeline erstellen
- Projekt-Runner erstellen und registrieren
- Pipeline triggern, um den Runner zu starten

GitLab Runner

Eigenen Project Runner benutzen

- GitLab Runner installieren
- Neues Projekt erstellen
- Projekt-Pipeline erstellen
- Projekt-Runner erstellen und registrieren
- Pipeline triggern, um den Runner zu starten

GitLab Runner installieren

- Erinnerung: GitLab Runner führen unsere CI/CD jobs aus
- Folgende Installationen sind möglich
  - Eigene Infrastruktur
  - In einem Docker-Container
  - Kubernetes-Cluster

- Hier: Infrastruktur → lokale Maschine
  - Linux oder Windows

GitLab Runner

Linux

1. Offizielles GitLab Repository hinzufügen
2. Aktuellstes GitLab Runner Paket installieren
3. Installation einer bestimmten GitLab Runner Version
4. GitLab Runner registrieren
5. GitLab Runner aktualisieren

Offizielles GitLab Repository hinzufügen

```
# Für Debian/Ubuntu/Mint (Achtung bei Debian: APT-Pinning!)
curl -L
https://packages.gitlab.com/install/repositories/runner/git
lab-runner/script.deb.sh | sudo bash

# Für RHEL/CentOS/Fedora
curl -L
https://packages.gitlab.com/install/repositories/runner/git
lab-runner/script.rpm.sh | sudo bash
```

Offizielle GitLab Repository hinzufügen

- Debian-Benutzer sollten APT-Pinning verwenden

```
cat <<EOF | sudo tee /etc/apt/preferences.d/pin-gitlab-runner.pref
Explanation: Prefer GitLab provided packages over the Debian native ones
Package: gitlab-runner
Pin: origin packages.gitlab.com
Pin-Priority: 1001
EOF
```

Aktuellstes GitLab Runner Paket installieren

```
# Für Debian/Ubuntu/Mint
sudo apt-get install gitlab-runner

# Für RHEL/CentOS/Fedora
sudo yum install gitlab-runner
```

Installation einer bestimmten GitLab Runner Version

```
# Für DEB basierte Systeme (Debian, Ubuntu, Mint..)
apt-cache madison gitlab-runner
sudo apt-get install gitlab-runner=10.0.0

# Für RPM basierte Systeme (RHEL, centOS, Fedora...)
yum list gitlab-runner --showduplicates | sort -r
sudo yum install gitlab-runner-10.0.0-1
```

GitLab Runner registrieren

1. Auf GitLab zu CI/CD Runner Einstellungen wechseln
   1. Projekt → Settings → CI/CD → Runners
2. Unter Project Runners auf „New project runner“ Button klicken
3. Tags und Konfiguration für den Runner eingeben (später über die GUI änderbar)
4. Runner über „Create runner“ erstellen
5. Plattform für Runner auswählen und gegeben Command im Terminal ausführen
6. In Konsole Konfiguration und Executor auswählen (Shell, Docker, …)
7. Falls Docker:
   - Standard Image angeben, welches genutzt werden soll, falls in .gitlab-ci.yml nichts definiert wurde
8. Optional: Runner mittels `gitlab-runner run` verifizieren

GitLab Runner aktualisieren

```
# Für Debian/Ubuntu/Mint
sudo apt-get update
sudo apt-get install gitlab-runner

# Für RHEL/CentOS/Fedora
sudo yum update
sudo yum install gitlab-runner
```

Windows

1. Systemordner erstellen, z.B.: C:\GitLab-Runner
2. Installationsdatei herunterladen und in den erstellten Ordner kopieren.
   1. Exe in gitlab-runner.exe umbenennen
3. Powershell als Admin starten
4. GitLab Runner registrieren
5. Den Runner als Service installieren über die Powershell:
   1. cd C:\Gitlab-Runner
   2. ./gitlab-runner.exe install
   3. ./gitlab-runner.exe start
6. Service läuft nun.
   Weitere Runner sindunter ./config.toml konfigurierbar

Eigenen Project Runner benutzen

- GitLab Runner installieren
- Neues Projekt erstellen
- Projekt-Pipeline erstellen
- Projekt-Runner erstellen und registrieren
- Pipeline triggern, um den Runner zu starten

Neues Projekt erstellen (optional, falls eins besteht)

1. Linke Sidebar oben
   → „**+**“ (Create new)
   → New project/repository
2. Create blank project
3. Projektdetails eingeben
   - Project name
   - Project slug
4. Abschließend: Create project

Eigenen Project Runner benutzen

- GitLab Runner installieren
- Neues Projekt erstellen
- Projekt-Pipeline erstellen
- Projekt-Runner erstellen und registrieren
- Pipeline triggern, um den Runner zu starten

Projekt-Pipeline erstellen

- .gitlab-ci.yml Datei im Projekt erstellen
- = YAML Datei für die CI/CD Pipeline Anweisungen

- In diese Datei gehört folgendes:
  - Die Struktur und Reihenfolge der jobs, welche durch den Runner ausgeführt (execute) werden
  - Die Entscheidungen, die der Runner bei bestimmten Bedingungen (conditions) treffen soll

Projekt-Pipeline erstellen

1. Linke Sidebar → „Search or go to“ → Projekt suchen
2. „Project overview“ auswählen
3. „**+**“ Icon in der Projektübersicht (nicht Sidebar!) auswählen → „New file“
4. „Filename“: .gitlab-ci.yml
5. Beispielkonfiguration:

```
stages:
        - build
        - test

job_build:
        stage: build
        script:
                - echo "Buildin the project"
job_test:
        stage: test
        script:
                - echo "Running tests"
```

6. „Commit changes“

Eigenen Project Runner benutzen

- GitLab Runner installieren
- Neues Projekt erstellen
- Projekt-Pipeline erstellen
- Projekt-Runner erstellen und registrieren
- Pipeline triggern, um den Runner zu starten

Projekt-Runner erstellen und registrieren

Voraussetzung: Maintainer-Rechte für das Projekt

1. Eignes Projekt auswählen
2. „Settings“ → „CI/CD“
3. Runners-Sektion aufklappen (Expand)
4. „Project runners“ → „New project runner“
5. „Tags“ → „Run untagged jobs“ auswählen
6. „Create runner“
7. Die on-screen Anweisungen befolgen für das OS, auf dem der Runner läuft (lokaler Rechner)
    1. Runner registrieren
    2. Executor auswählen (shell)
    3. Runner starten

Eigenen Project Runner benutzen

- GitLab Runner installieren
- Neues Projekt erstellen
- Projekt-Pipeline erstellen
- Projekt-Runner erstellen und registrieren
- Pipeline triggern, um den Runner zu starten

Pipeline triggern, um den Runner zu starten

1. Eignes Projekt auswählen
2. „Build“ → „Pipelines“
3. „Run pipeline“
4. Job selektieren → Man sieht die dazugehörigen Logs

Aufgabe 1: Eigene Pipeline mit einem Project Runner starten

1. **Ziel:** Verständnis über Runner schaffen
2. **Schritte:**
   - Installieren Sie lokal den GitLab Runner
   - Erstellen oder nutzen Sie Ihr eigenes Projekt
   - Erstellen Sie eine Pipeline für Ihr Projekt
   - Erstellen und registrieren Sie den Runner
   - Starten Sie Ihre Pipeline


FINALLY
BREAK TIME
memegenerator.net

GitLab Runner

anderScore trust in competence

Verschiedene Runner verwalten

- Wann ist welcher Runner sinnvoll?
  - Hängt davon ab, wer Zugriff haben soll!
- Instance Runners
  - Verfügbar für alle Gruppen und Projekte einer GitLab-Instanz
  - Adminrechte in GitLab unbedingt notwendig!
- Group Runners
  - Verfügbar für alle Projekte und Untergruppen einer Gruppe
  - Owner-Rolle für die Gruppe benötigt
- Project Runners
  - Können mit verschiedenen Projekten verknüpft sein
  - Normalerweise nur von einem Projekt genutzt
  - Verfügbar für das verknüpfte Projekt
  - Maintainer-Rolle für das Projekt benötigt
- Hier: Runner mit authentication token! (nicht registration token)

Live-Demo: Instance Runners

- Einen Instance Runner mit authentication token erstellen
- Anhalten und Fortsetzen eines Runners
- Einen Instance Runner löschen
- Instance Runners für ein Projekt aktivieren/deaktivieren
- Instance Runners für eine Gruppe aktivieren/deaktivieren

GitLab Runner

Instance Runners

- Einen Instance Runner mit authentication token erstellen
- Anhalten und Fortsetzen eines Runners
- Einen Instance Runner löschen
- Instance Runners für ein Projekt aktivieren/deaktivieren
- Instance Runners für eine Gruppe aktivieren/deaktivieren

Einen Instance Runner mit authentication token erstellen

Voraussetzung: Adminrechte in GitLab

1. Linke Sidebar → ganz unten → „Admin Area“
2. „CI/CD“ → „Runners“ auswählen
3. „New instance runner“ auswählen
4. Tags auswählen bzw. erstellen, falls keine Tags vorhanden, dann „Run untagged“ auswählen
5. Optional: Beschreibung ausfüllen
6. Optional: Konfiguration ausfüllen
7. „Create runner“ auswählen
8. Die Anweisungen von GitLab folgen, um den Runner zu registrieren

Instance Runners

- Einen Instance Runner mit authentication token erstellen
- Anhalten und Fortsetzen eines Runners
- Einen Instance Runner löschen
- Instance Runners für ein Projekt aktivieren/deaktivieren
- Instance Runners für eine Gruppe aktivieren/deaktivieren

Anhalten und Fortsetzen eines Runners

Voraussetzung: Adminrechte in GitLab

- Runner können pausiert werden, damit keine weiteren jobs angenommen werden

1. Linke Sidebar → ganz unten → „Admin Area“
2. „CI/CD“ → „Runners“ auswählen
3. Den entsprechenden Runner suchen
4. In der Liste von Runnern
   - Pause-Symbol zum Pausieren
   - Play-Symbol zum Fortsetzen

Instance Runners

- Einen Instance Runner mit authentication token erstellen
- Anhalten und Fortsetzen eines Runners
- Einen Instance Runner löschen
- Instance Runners für ein Projekt aktivieren/deaktivieren
- Instance Runners für eine Gruppe aktivieren/deaktivieren

Einen Instance Runner löschen

Voraussetzung: Adminrechte in GitLab

- Achtung: Der Runner wird permanent gelöscht!

1. Linke Sidebar → ganz unten → „Admin Area“
2. „CI/CD“ → „Runners“ auswählen
3. Den entsprechenden Runner suchen
4. Löschen des Instance Runners
   - Um einen einzelnen Runner zu löschen → „Delete runner“ (Lösch-Symbol)
   - Um mehrere Runner zu löschen → Checkbox selektieren neben dem Runner und „Delete selected“ auswählen
   - Um alle Runner zu löschen → Die Checkbox für alle Runner auswählen und „Delete selected“ auswählen
5. „Permanently delete runner“ auswählen

Instance Runners

- Einen Instance Runner mit authentication token erstellen
- Anhalten und Fortsetzen eines Runners
- Einen Instance Runner löschen
- Instance Runners für ein Projekt aktivieren/deaktivieren
- Instance Runners für eine Gruppe aktivieren/deaktivieren

Instance Runners für ein Projekt aktivieren/deaktivieren

- GitLab.com-Default: Für alle Projekte aktiviert
- Self-managed GitLab: Admin kann aktivieren/deaktivieren
- Für bestehende Projekte: Ein Admin muss diese installieren und registrieren

Um Instance Runners zu (de)aktivieren:

1. Gewünschtes Projekt auswählen
2. „Settings“ → „CI/CD“
3. „Runners“ ausklappen
    1. Aktivieren: „Enable instance runners for this project“ auswählen
    2. Deaktivieren: „ Enable instance runners for this project“ abwählen

In folgenden Fällen sind Instance Runners automatisch deaktiviert:

- Wenn Instance Runner für die Parent-Gruppe deaktiviert sind
- Wenn das Überschreiben dieser Einstellung auf Projektebene nicht erlaubt ist

Instance Runners

- Einen Instance Runner mit authentication token erstellen
- Anhalten und Fortsetzen eines Runners
- Einen Instance Runner löschen
- Instance Runners für ein Projekt aktivieren/deaktivieren
- Instance Runners für eine Gruppe aktivieren/deaktivieren

Instance Runners für eine Gruppe aktivieren/deaktivieren

1. Gewünschte Gruppe auswählen
2. „Settings“ → „CI/CD“
3. „Runners“ ausklappen
   1. Aktivieren: „Enable instance runners for this group“ auswählen
   2. Deaktivieren: „Enable instance runners for this group“ abwählen
4. Optional: Damit Instance Runners für individuelle Projekte oder Sub-Gruppen aktiviert werden können
   - „Allow projects and subgroups to override the group setting” auswählen

Aufgabe 2: Instance Runner kennenlernen

1. **Voraussetzungen:** Adminrechte in GitLab
2. **Ziel:** Verständnis über Runner schaffen
3. **Schritte:**
   - Erstellen Sie einen Instance Runner
   - Schauen Sie sich an, wie man einen Instance Runner starten und stoppen kann
   - Deaktivieren und aktivieren sie Ihren Instance Runner für Ihr Projekt
   - Machen Sie das Gleiche für eine Gruppe
   - Löschen Sie Ihren Instance Runner

Wie Instance Runners ihre Jobs auswählen

- Mittels Fair-Queuing (fair usage queue)
  - Verhindert alle Ressourcen an nur ein Projekt
- Jobs: auf Basis von Projekten zugeordnet
- Projekt: geringste Anzahl Runner → erhält Runner
- Beispiel-Queue
  - Job 1 für Project 1
  - Job 2 für Project 1
  - Job 3 für Project 1
  - Job 4 für Project 2
  - Job 5 für Project 2
  - Job 6 für Project 3

Wie Instance Runners ihre Jobs auswählen

CI/CD jobs parallel, dann folgende Reihenfolge

- Beispiel-Queue
  - Job 1 für Project 1
  - Job 2 für Project 1
  - Job 3 für Project 1
  - Job 4 für Project 2
  - Job 5 für Project 2
  - Job 6 für Project 3

1. Job 1, da niedrigste Job-Nummer von Projekten ohne laufenden Job
2. Job 4, da niedrigste Job-Nummer von Projekten ohne laufenden Job (Project 1 hat einen laufenden Job)
3. Job 6, da niedrigste Job-Nummer von Projekten ohne laufenden Job (Project 1 & 2 haben einen laufenden Job)
4. Job 2, von allen Projects mit der niedrigsten Job-Anzahl, Job 2 die niedrigste Job-Nummer hat
5. Job 5, Project 1 hat nun zwei Jobs und Job 5 die niedrigste verbleibende Job-Nummer zwischen Project 2 und 3
6. Job 3, da ist der letzte Job ist.

Wie Instance Runners ihre Jobs auswählen

- Beispiel-Queue
  - Job 1 für Project 1
  - Job 2 für Project 1
  - Job 3 für Project 1
  - Job 4 für Project 2
  - Job 5 für Project 2
  - Job 6 für Project 3

Wenn jeweils nur ein job läuft:

1. Job 1, da niedrigste Job-Nummer von Projekten ohne laufende Jobs.
2. Job 1 beendet.
3. Job 2, da Job 1 beendet und alle Projekte haben keine Jobs laufen und 2 ist die niedrigste verfügbare Job-Nummer.
4. Job 4, da für Project 1 bereits ein Job läuft und da die 2 die niedrigste Projektnummer ist von den Projekten, bei denen kein Job läuft
5. Job 4 beendet.
6. Job 5, da Job 4 beendet und Projekt 2 keine laufenden Jobs hat.
7. Job 6, da Projekt 3 das einzige Projekt ist, das noch keine laufenden Jobs hat.
8. Job 3, da ist der letzte Job ist.

I need a

**
**

Live-Demo: Group Runners

- Einen Group Runner mit authentication token erstellen
- Group Runners anzeigen lassen
- Anhalten und Fortsetzen eines Runners
- Einen Group Runner löschen
- Alte/“abgestandene“ Group Runners bereinigen

GitLab Runner

Group Runners

- Einen Group Runner mit authentication token erstellen
- Group Runners anzeigen lassen
- Anhalten und Fortsetzen eines Runners
- Einen Group Runner löschen
- Alte/“abgestandene“ Group Runners bereinigen

Einen Group Runner mit authentication token erstellen

Voraussetzung: Owner-Rechte für die Gruppe

1. Gewünschte Gruppe auswählen in GitLab
2. „Build“ → „Runners“ auswählen
3. „New group runner“ auswählen
4. Tags auswählen bzw. erstellen, falls keine Tags vorhanden, dann „Run untagged“ auswählen
5. Optional: Beschreibung ausfüllen
6. Optional: Konfiguration ausfüllen
7. „Create runner“ auswählen
8. Die Anweisungen von GitLab folgen, um den Runner zu registrieren

Group Runners

- Einen Group Runner mit authentication token erstellen
- Group Runners anzeigen lassen
- Anhalten und Fortsetzen eines Runners
- Einen Group Runner löschen
- Alte/“abgestandene“ Group Runners bereinigen

Group Runners anzeigen lassen

Voraussetzung: Maintainer- oder Owner-Rechte für die Gruppe

Alle Runner einer Gruppe und dessen Sub-Gruppen sowie Projekte kann man wie folgt einsehen:

1. Gewünschte Gruppe in GitLab auswählen
2. „Build“ → „Runners“ auswählen
3. Filter, um nur Sub-Gruppen zu sehen:
   - „Show only inherited“ toggeln

Group Runners

- Einen Group Runner mit authentication token erstellen
- Group Runners anzeigen lassen
- Anhalten und Fortsetzen eines Runners
- Einen Group Runner löschen
- Alte/“abgestandene“ Group Runners bereinigen

Anhalten und Fortsetzen eines Runners

Voraussetzung: Administrator oder Owner-Rechte für die Gruppe

- Group Runner anhalten, damit dieser keine Jobs mehr von Sub-Gruppen und Projekten annimmt
  - Bei Benutzung durch mehreren Projekten → für alle Projekte pausiert

1. Gewünschte Gruppe in GitLab auswählen
2. „Build“ → „Runners“ auswählen
3. Den gewünschten Runner suchen
4. In der Liste von Runnern
   - Pause-Symbol zum Pausieren
   - Play-Symbol zum Fortsetzen

Group Runners

- Einen Group Runner mit authentication token erstellen
- Group Runners anzeigen lassen
- Anhalten und Fortsetzen eines Runners
- Einen Group Runner löschen
- Alte/“abgestandene“ Group Runners bereinigen

Einen Group Runner löschen

Voraussetzung: Administrator oder Owner-Rechte für die Gruppe

- Achtung: Der Runner wird permanent gelöscht!

1. Gewünschte Gruppe auswählen
2. „CI/CD“ → „Runners“ auswählen
3. Den entsprechenden Runner suchen
4. Löschen des Group Runners
   - Um einen einzelnen Runner zu löschen → „Delete runner“ (Lösch-Symbol)
   - Um mehrere Runner zu löschen → Checkbox selektieren neben dem Runner und „Delete selected“ auswählen
   - Um alle Runner zu löschen → Die Checkbox für alle Runner auswählen und „Delete selected“ auswählen
5. „Permanently delete runner“ auswählen

Group Runners

- Einen Group Runner mit authentication token erstellen
- Group Runners anzeigen lassen
- Anhalten und Fortsetzen eines Runners
- Einen Group Runner löschen
- Alte/“abgestandene“ Group Runners bereinigen

Alte/“abgestandene“ Group Runners bereinigen

Voraussetzung: Owner-Rechte für die Gruppe

- Inaktive (> 3 Monate) Group Runner (= „stale“) können automatisch bereinigt werden
- Group Runners = erstellt auf Gruppenebene

1. Gewünschte Gruppe in GitLab auswählen
2. „Settings“ → „CI/CD“ auswählen
3. „Runners“ aufklappen
4. „Enable stale runner cleanup“ toggeln

Aufgabe 3: Group Runner kennenlernen

1. **Ziel:** Verständnis über Runner schaffen
2. **Schritte:**
   - Erstellen oder nutzen Sie eine Gruppe
   - Erstellen Sie einen Group Runner
   - Lassen Sie sich Ihren Group Runner anzeigen
   - Stoppen und Starten Sie einen Group Runner

Live-Demo: Project Runners

- Einen Project Runner mit authentication token erstellen (Zur Erinnerung ☺)
- Anhalten und Fortsetzen eines Runners
- Einen Project Runner löschen
- Project Runners für ein anderes Projekt aktivieren
- Project Runners für andere Projekte sperren

GitLab Runner

Project Runners

- Einen Project Runner mit authentication token erstellen
- Anhalten und Fortsetzen eines Runners
- Einen Project Runner löschen
- Project Runners für ein anderes Projekt aktivieren
- Project Runners für andere Projekte sperren

Einen Project Runner mit authentication token erstellen

Voraussetzung: Maintainer-Rechte für das Projekt

1. Eignes Projekt auswählen
2. „Settings“ → „CI/CD“
3. Runners-Sektion aufklappen (Expand)
4. „Project runners“ → „New project runner“
5. „Tags“ → „Run untagged jobs“ auswählen
6. „Create runner“
7. Die on-screen Anweisungen befolgen für das OS, auf dem der Runner läuft (lokaler Rechner)
    1. Runner registrieren
    2. Executor auswählen (shell)
    3. Runner starten

Project Runners

- Einen Project Runner mit authentication token erstellen
- Anhalten und Fortsetzen eines Runners
- Einen Project Runner löschen
- Project Runners für ein anderes Projekt aktivieren
- Project Runners für andere Projekte sperren

Anhalten und Fortsetzen eines Runners

Voraussetzung: Administrator oder Maintainer-Rechte für das Projekt

1. Gewünschtes Projekt in GitLab auswählen
2. „Settings“ → „CI/CD“
3. „Runners“ aufklappen
4. Unter „Assigned project runners“ den Runner finden
   - Pause-Symbol zum Pausieren
   - Play-Symbol zum Fortsetzen

Project Runners

- Einen Project Runner mit authentication token erstellen
- Anhalten und Fortsetzen eines Runners
- Einen Project Runner löschen
- Project Runners für ein anderes Projekt aktivieren
- Project Runners für andere Projekte sperren

Einen Project Runner löschen

Voraussetzung: Administrator oder Maintainer-Rechte für das Projekt

- Achtung: Der Runner wird permanent gelöscht!

1. Gewünschtes Projekt in GitLab auswählen
2. „Settings“ → „CI/CD“ auswählen
3. „Runners“ aufklappen
4. Unter „Assigned project runners“ den Runner finden
5. „Remove runner“ auswählen
6. Mit „Remove“ das Löschen bestätigen

Project Runners

- Einen Project Runner mit authentication token erstellen
- Anhalten und Fortsetzen eines Runners
- Einen Project Runner löschen
- Project Runners für ein anderes Projekt aktivieren
- Project Runners für andere Projekte sperren

Project Runners für ein anderes Projekt aktivieren

Voraussetzung: Mindestens die Maintainer-Rechte für

- Das Projekt, in dem der Runner bereits aktiviert ist
- Das Projekt, in dem der Runner aktiviert werden soll
- Der Project Runner darf nicht gesperrt sein

1. Gewünschtes Projekt auswählen
2. „Settings“ → „CI/CD“
3. „Runners“ aufklappen
4. Im Bereich „Project runners“
    1. Gewünschten Runner auswählen und
    2. „Enable for this project“ selektieren

Project Runners

- Einen Project Runner mit authentication token erstellen
- Anhalten und Fortsetzen eines Runners
- Einen Project Runner löschen
- Project Runners für ein anderes Projekt aktivieren
- Project Runners für andere Projekte sperren

Project Runners für andere Projekte sperren

- Project Runner können für andere Projekte gesperrt werden

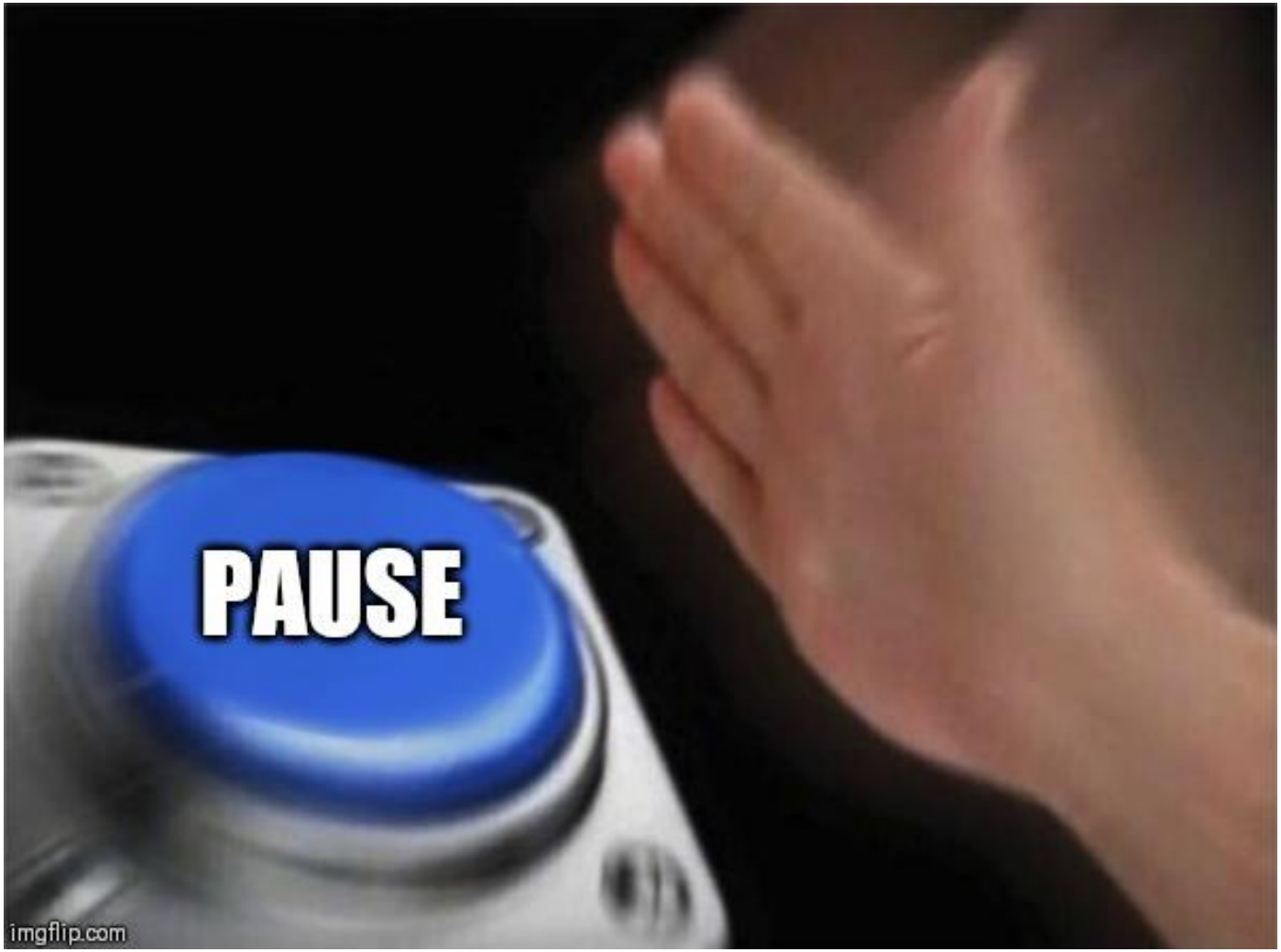
1. Gewünschtes Projekt auswählen
2. „Settings“ → „CI/CD“
3. „Runners“ aufklappen
4. Den zu (ent)sperrenden Project Runner auswählen
5. „Edit“ (Stift-Icon) anklicken
6. „Lock to current projects“ auswählen
7. Mit „Save changes“ bestätigen

GitLab Runner

anderScore trust in competence

Runner Status	Beschreibung
online	Der Runner hat sich innerhalb der letzten 2 Stunden mit GitLab in Verbindung gesetzt und ist für die Ausführung von Jobs verfügbar.
offline	Der Runner hat sich seit mehr als 2 Stunden nicht mehr mit GitLab in Verbindung gesetzt und ist nicht verfügbar, um Jobs auszuführen. Überprüfen Sie den Runner, um zu sehen, ob Sie ihn online bringen können.
stale	Der Runner hat seit mehr als 3 Monaten keinen Kontakt zu GitLab aufgenommen. Wenn der Läufer vor mehr als 3 Monaten erstellt wurde, aber nie mit der Instanz in Kontakt getreten ist, wird er ebenfalls als veraltet betrachtet.
never_contacted	Der Runner hat sich noch nie mit GitLab in Verbindung gesetzt. Um den Runner mit GitLab in Kontakt zu bringen, führen Sie gitlab-runner run aus.

GitLab Runner



GitLab Runner

TL;DR

- GitLab Job
  - Kleinste Komponente einer Pipeline
  - 1-n auszuführbaren Befehlen
- GitLab Runner
  - Agent, oft auf anderer Infrastruktur
  - GitLab Server gibt Anweisung über nächste Job-Ausführung
- Runner Executor
  - Jeder Runner hat mindestens einen Executor
  - Executor = Umgebung für die Ausführung des GitLab jobs

Executors

- Verschiedene Executors sind in GitLab verfügbar
  - Shell
  - SSH
  - VirtualBox
  - Parallels
  - Docker
    - Docker Machine
  - Kubernetes
- Executor ist abhängig vom Use Case!
  - Es gibt nicht den „besten“ Executor

Shell Executor

- Führt die Jobs dort aus, wo der Runner installiert wurde
  - Analog zu Jenkins oder anderen CI Servern
- Alle benötigten Dependencies müssen auf dem Server installiert sein
- Aber: Docker Images in der Job-Config werden ignoriert!
  - Selbst wenn Docker installiert ist
- Alles zur Laufzeit vorhanden
  - → Jobs werden sehr schnell ausgeführt

Shell Executor

- Use Case
  - Native Umgebung (spezifisches Betriebssystem oder Hardware)

- Zu beachten
  - Umgebung kann unzureichend dokumentiert sein
    - Welche Versionen werden verwendet?
    - Job auf andere Infrastruktur → Welche Dependencies in welcher Version werden benötigt?
  - „Left-overs“ aus vorherigen Jobs → keine saubere build Umgebung
  - Dependency-Management
    - Ein Projekt benötigt Node.js v20 ein Anderes v18
  - Muss den anderen Jobs vertrauen
    - Haben vollen Zugriff auf das Projekt + Secrets

SSH Executor

- Befehle über SSH an eine Maschine
- Funktionsweise ähnlich zum Shell Executor
- Funktioniert nur für Bash-Scripts!
- Höhere Sicherheit, da SSH
  - → Befehle haben nicht Zugriff auf das gesamte Dateisystem
- Nur zur Vollständigkeit bei GitLab selbst aufgeführt!
  - Hat den geringsten Support
- Wird von GitLab selbst nicht empfohlen!

SSH Executor

- Use Case
  - GitLab Server nur per SSH erreichbar? Das ist der Weg.
  - Es kann kein GitLab Runner auf einer anderen Maschine installiert werden
- Zu beachten (analog zu Shell)
  - Umgebung kann unzureichend dokumentiert sein
    - Welche Versionen werden verwendet?
    - Job auf andere Infrastruktur → Welche Dependencies in welcher Version werden benötigt?
  - „Left-overs“ aus vorherigen Jobs → keine saubere build Umgebung
  - Dependency-Management
    - Ein Projekt benötigt Node.js v20 ein Anderes v18
  - Muss den anderen Jobs vertrauen
    - Haben vollen Zugriff auf das Projekt + Secrets

Virtual Machine Executor (VirtualBox / Parallels)

- Eine bereits bestehende virtuelle Maschine wird genutzt
  - Wird gecloned und darauf läuft dann der Build
- Jeder Job startet dementsprechend in einer virtuellen Umgebung
  - Windows, Linux, macOS oder FreeBSD

Virtual Machine Executor (VirtualBox / Parallels)

- Use Case
  - Verschiedene Umgebungen durch Virtualisierung
    - Testen mit verschiedenen Betriebssystemen
  - Falls Docker nicht angenommen oder verstanden wird
  - Falls VirtualBox/Parallels bereits eingesetzt werden

- Zu beachten
  - Overhead → Betriebssystem starten
  - Debugging schwerer (bei Job-fail)
  - Verbindung läuft über SSH! (Fehlerquellenmöglichkeit)
  - Zusätzliche Config/Fehlersuche beim Hochladen von Job-Artefakten

Docker Executor

- Docker Container wird in der Pipeline definiert
- Simple Laufzeitumgebungen
- Mehrere Jobs gleichzeitig
  - Ohne Interferenzen (außer Systemlast/Performance)
- Docker Umgebung für die meisten Projekte sinnvoll
- Alle Abhängigkeiten im Docker Image definiert

Docker Executor

- Use Case
  - Saubere Umgebung für jeden Job
  - Projekte unabhängig voneinander

- Zu beachten
  - Overhead vom „Pulling“
    - Für jeden Job wird das Docker Image heruntergeladen

Docker Machine Executor

- Spezielle Version des Docker Executor
  - Mit Support für auto-scaling
- Funktioniert wie der Docker Executor
  - Aber mit Build Hosts
    - Werden bei Bedarf von der Docker Maschine erstellt
- Use Case
  - Skalierbare Build-Infrastruktur
- Zu beachten
  - … Kubernetes? ☺

Kubernetes Executor

- Bestehendes Kubernetes-Cluster wird verwendet
- Jobs werden auf dem Kubernetes-Cluster ausgeführt
- Executor ruft die API auf und erzeugt neuen Pod
  - Mit einem Build-Contrainer und Services-Containers
  - Für jeden GitLab CI Job

- Use Case
  - Skalierbare Build-Infrastruktur
  - Bei bestehendem Kubernetes-Cluster

Executor Chart

Executor	SSH	Shell	VirtualBox	Parallels	Docker	Kubernetes
Clean build environment for every build	×	×	✓	✓	✓	✓
Reuse previous clone if it exists	✓	✓	×	×	✓	×
Runner file system access protected (5)	✓	×	✓	✓	✓	✓
Migrate runner machine	×	×	partial	partial	✓	✓
Zero-configuration support for concurrent builds	×	× (1)	✓	✓	✓	✓
Complicated build environments	×	× (2)	✓ (3)	✓ (3)	✓	✓
Debugging build problems	easy	easy	hard	hard	medium	medium

(1) Möglich, aber problematisch, wenn der Build auf der Build Maschine installierte Services nutzt

(2) Erfordert manuelle Dependency Injection

(3) Beispielsweise mit Vagrant

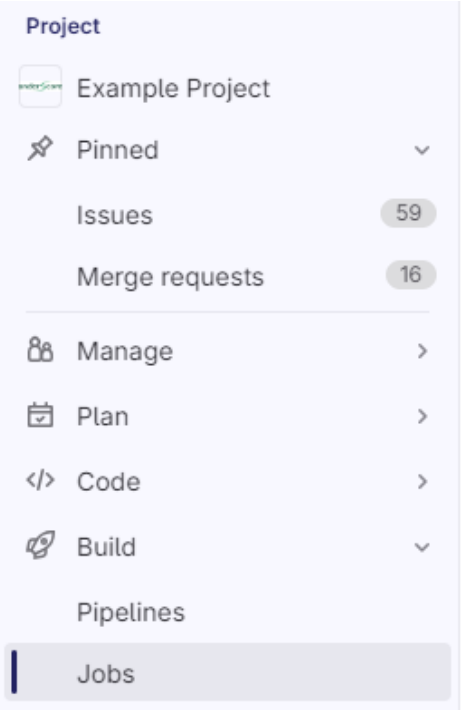
anderScore trust in competence

Compability Chart

Executor	SSH	Shell	VirtualBox	Parallels	Docker	Kubernetes	Custom
Secure Variables	✓	✓	✓	✓	✓	✓	✓
`.gitlab-ci.yml`: image	×	×	✓ (1)	✓ (1)	✓	✓	✓ (via `$CUSTOM_ENV_CI_JOB_IMAGE`)
`.gitlab-ci.yml`: services	×	×	×	×	✓	✓	✓
`.gitlab-ci.yml`: cache	✓	✓	✓	✓	✓	✓	✓
`.gitlab-ci.yml`: artifacts	✓	✓	✓	✓	✓	✓	✓
Passing artifacts between stages	✓	✓	✓	✓	✓	✓	✓
Use GitLab Container Registry private images	n/a	n/a	n/a	n/a	✓	✓	n/a
Interactive Web terminal	×	✓ (UNIX)	×	×	✓	✓	×

Welcher Executor wird genutzt?

- Steht in den Logs vom GitLab Job




```
1  Running with gitlab-runner 15.3.0 (bbcb5aba)
2    on docker-default HFitoxxJ
3  Preparing the "docker" executor
4  Using Docker executor with image gitlab.ads.anderscore.com:5006/example/project:latest ...
5  Authenticating with credentials from job payload (GitLab Registry)
6  Pulling docker image gitlab.ads.anderscore.com:5006/example/project:latest ...
7  Using docker image sha256:4b078e4f3a50b99dfbfc1e92c5736eb598a7662fd918f0ef83f3df2e79b5ba0e
   76778b0f8ba01b48962ad4ebb2657ed520c30ad6774f6a ...
8  Preparing environment
9  Running on runner-hfitoxxj-project-63-concurrent-0 via ed568276aa82...
10 Getting source from Git repository
11 Fetching changes...
12 Reinitialized existing Git repository in /builds/example/project/.git/
13 Checking out 3ff31f42 as master...
14 Removing Gemfile.lock
```