



Tag 1: Einführung in Git und GitLab, Git-Workflow im Team

17.06.2024, Daniel Krämer & Malte Fischer

© Copyright 2024 anderScore GmbH

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

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

Einführung in **GitLab**

Inhalt

- Einführung
 - Was ist GitLab
 - Git vs GitLab
 - Versionen
- Live-Demo GitLab
 - Projekte erstellen
 - Gruppen
 - Arbeiten im Projekt
 - Planing Tools
 - Branches und Merge-Requests
- Übung zu GitLab

Was ist GitLab?

- 2011 von Dimitri Saporoschez und Valery Sizov entwickelt
- Service zur Softwareentwicklung und Versionsverwaltung
- Hosting von Git Remote Repositories als Projekte
- Zusätzliche Funktionen im Bereich:
 - Issue-Tracking: Verwaltung/Verfolgung von Aufgaben
 - Dokumentation
 - CI/CD (Continuous Integration/Continuous Deployment)



Git vs GitLab

Git

- Verteiltes Versionskontrollsystem zur lokalen Verwaltung von Dateien
- Verfolgung und Speicherung von Änderungen an Dateien

GitLab

- Umfassende DevOps-Plattform
- Erweitert Git um Tools und Funktionen für Projektmanagement, Zusammenarbeit und Automatisierung
- Zentrale Plattform zur Verwaltung von Projekten

Unterschiedliche GitLab Versionen

- Free
 - Kostenlos und Open Source
 - Konzipiert für persönliche Projekte
 - 400 compute minutes für CI/CD
 - Support nur über GitLab Forum
- Premium
 - Für mittlere bis große Teams
 - Fortgeschrittene Features im Bereich Projektmanagement, Code Reviews sowie CI/CD
 - 10.000 compute minutes für CI/CD
 - Support bei Problemen

- Ultimate
 - Kostenlos und Open Source
 - Für Unternehmen und große Organisationen
 - Umfangreiche Statistiken zur Analyse
 - Erweiterte Funktionen insbesondere im Bereich Security und Automatisierung
 - 50,000 compute minutes für CI/CD
- Versionen können als SaaS auf GitLab.com oder selbst gehostete Instanz im eigenen Netzwerk betrieben werden
- Zusätzliche Add-ons mit Fokus auf AI-Unterstützung für Premium und Ultimate verfügbar

GitLab

Startseite und Projekte

Startseite & Projekte

The screenshot shows the GitLab interface with the following details:

- Left Sidebar:** Shows navigation links for "Your work" including Projects (selected), Groups, Issues (5), Merge requests, To-Do List (6), Milestones, Snippets, Activity, Workspaces, Environments, Operations, Security, and Help.
- Header:** Shows "Your work / Projects".
- Top Bar:** Shows project counts (5 issues, 1 merge request, 6 to-do items), a search bar ("Search or go to..."), and user profile icons.
- Section Headers:** "Projects", "Yours 2", "Starred 0", "Pending deletion".
- Filter Options:** "Filter by name", "Language", "Name".
- Project Listings:**
 - P** GitLabUser123 / own_project (Owner) - Last updated 5 months ago.
 - G** Trainings / Gitlab (Developer) - Last updated 2 weeks ago.

Your work / Projects / New project

Create new project



Create blank project

Create a blank project to store your files, plan your work, and collaborate on code, among other things.



Create from template

Create a project pre-populated with the necessary files to get you started quickly.



Import project

Migrate your data from an external source like GitHub, Bitbucket, or another instance of GitLab.



Run CI/CD for external repository

Connect your external repository to GitLab CI/CD.

You can also create a project from the command line. [Show command](#)



Create blank project

Create a blank project to store your files, plan your work, and collaborate on code, among other things.

Project name

Must start with a lowercase or uppercase letter, digit, emoji, or underscore. Can also contain dots, pluses, dashes, or spaces.

Project URL

Project slug

Visibility Level

 Private

Project access must be granted explicitly to each user. If this project is part of a group, access is granted to members of the group.

 Internal

The project can be accessed by any logged in user except external users.

 Public

The project can be accessed without any authentication.

Project Configuration

Initialize repository with a README

Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.

Enable Static Application Security Testing (SAST)

Analyze your source code for known security vulnerabilities. [Learn more](#).

Neues Projekt erstellen

- URL
 - Project URL + Project Slug bestimmen die URL, unter der das neue Projekt aufgerufen werden kann
- Visibility Level
 - Gibt an, welche User Zugriff auf das Projekt erhalten
 - Internal als zusätzliche Option bei eigenen GitLab Instanzen für angemeldete User und Guest Accounts
- Project Configuration
 - Mit einer README wird eine erste Datei und ein initialer Commit angelegt
 - Sinnvoll wenn man zuerst das Remote Repository anlegt um direkt clonen zu können
 - Bei bestehendem lokalen Repository sollte man ein leeres Remote Repository anlegen

GitLab

Groups

Groups

The screenshot shows the GitLab interface with the 'Groups' page selected. The left sidebar includes links for Projects, Groups (selected), Issues, Merge requests, To-Do List, Milestones, Snippets, Activity, Workspaces, Environments, Operations, and Security. The main content area displays two groups: 'example-group' (with a lock icon) and 'Trainings' (with a lock icon). A search bar at the top right allows users to search by name or filter by 'Last created'.

- Projects
- Groups
- Issues 5
- Merge requests >
- To-Do List 6
- Milestones
- Snippets
- Activity
- Workspaces
- Environments
- Operations
- Security >

Your work / Groups

Groups

Explore groups New group

Search by name Last created

Group Name	Description	Last Activity
example-group	Example group description	1
Trainings		6

- Gruppen können zur Verwaltung von einem oder mehreren Projekten gleichzeitig verwendet werden
 - Projekte können innerhalb von Gruppen angelegt werden
 - Dadurch befinden diese sich im **Namespace** der Gruppe, bspw. https://gitlab.example.de/my_group/my_project
 - Verwaltung von Rechten auf Gruppenebene
 - Mitglieder erhalten Zugriff auf alle Projekte innerhalb einer Gruppe
- Können zur Kommunikation verwendet werden
- Abrufen von gruppenspezifischen Statistiken und Aktivitäten
- Können in Untergruppen aufgeteilt werden
- Gruppeneinteilungen und Strukturen auf individuelle Bedürfnisse anpassbar

Groups

Your work / Groups / New group / Create group



Create group

Groups allow you to manage and collaborate across multiple projects. Members of a group have access to all of its projects.

Groups can also be nested by creating subgroups.

Group name

Must start with letter, digit, emoji, or underscore. Can also contain periods, dashes, spaces, and parentheses.

⚠ Your group name must not contain a period if you intend to use SCIM integration, as it can lead to errors.

Group URL

Visibility level

Who will be able to see this group? [View the documentation](#)

Private

The group and its projects can only be viewed by members.

Internal

The group and any internal projects can be viewed by any logged in user except external users.

Public

The group and any public projects can be viewed without any authentication.

Now, personalize your GitLab experience

We'll use this to help surface the right features and information to you.

Role

Who will be using this group?

My company or team

Just me

What will you use this group for?

Invite Members (optional)

Invited users will be added with developer level permissions. [View the documentation](#) to see how to change this later.

Email 1

+ Invite another member

Create group

Cancel

Groups

- Anlegen eines Projektes innerhalb einer Gruppe über Button „New Project“ auf Startseite/Gruppenseite

Your work / Projects / New project / Create blank project

Create blank project

Create a blank project to store your files, plan your work, and collaborate on code, among other things.

Project name
My awesome project

Must start with a lowercase or uppercase letter, digit, emoji, or underscore. Can also contain dots, pluses, dashes, or spaces.

Project URL
https://gitlab.com/ Example_Group

Project slug
/ my-awesome-project

Project deployment target
Select the deployment target for this project. This will determine where the project is accessible from and how it is versioned.
Groups
Example_Group

Visibility Level
 Private
Project access must be granted by a member of the group.
 Public
The project can be accessed without any authentication.

Users
Example_User

this project is part of a group, access is granted to members of the group.

Project Configuration

Initialize repository with a README
Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.

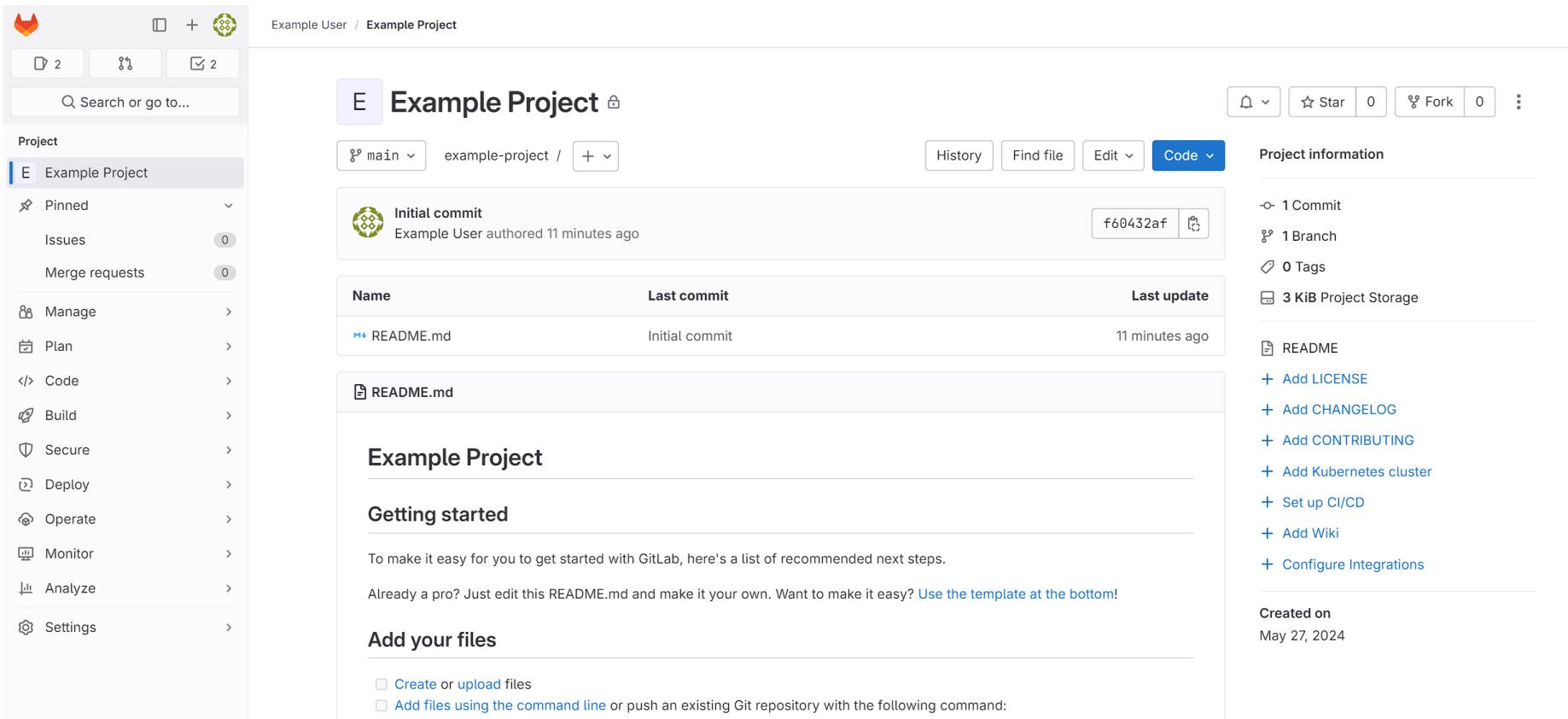
Enable Static Application Security Testing (SAST)
Analyze your source code for known security vulnerabilities. [Learn more](#).

Create project **Cancel**

GitLab

Clonen von Projekten

Clonen von Projekten



The screenshot shows a GitLab project interface for 'Example Project'. The left sidebar includes pinned items, issues (0), merge requests (0), and various management sections like Plan, Code, Build, Secure, Deploy, Operate, Monitor, Analyze, and Settings. The main content area displays the repository structure under 'Code', showing a single commit from 'Example User' and a file list. The right sidebar provides project information such as 1 commit, 1 branch, 0 tags, 3 KiB of storage, and links for README, Add LICENSE, Add CHANGELOG, Add CONTRIBUTING, Add Kubernetes cluster, Set up CI/CD, Add Wiki, and Configure Integrations. It also shows the project was created on May 27, 2024.

Project information

- > 1 Commit
- < 1 Branch
- < 0 Tags
- 3 KiB Project Storage

README

+ Add LICENSE

+ Add CHANGELOG

+ Add CONTRIBUTING

+ Add Kubernetes cluster

+ Set up CI/CD

+ Add Wiki

+ Configure Integrations

Created on

May 27, 2024

Clonen von Projekten

- In Projektübersicht **Code** auswählen

Example User / Example Project

The screenshot shows a GitLab project page for 'Example Project'. The top navigation bar includes 'History', 'Find file', 'Edit', 'Code' (which is currently selected), and 'Project information'. A dropdown menu titled 'Clone with SSH' displays the URL 'git@gitlab.example.de:exampleuse'. Below it, another dropdown menu titled 'Clone with HTTPS' also displays the URL 'git@gitlab.example.de:exampleuse'. The main content area shows a commit history with one entry: 'Initial commit' by 'Example User' a few hours ago. A table lists files with their last commits. A 'README.md' file is shown with its content: 'Example Project', 'Getting started' (with a note about recommended tools), and 'Add your files'. The bottom right corner shows the creation date: 'Created on May 07, 2024'.

Name	Last commit
README.md	Initial commit

Clone with SSH
git@gitlab.example.de:exampleuse

Clone with HTTPS
git@gitlab.example.de:exampleuse

Open in your IDE
Visual Studio Code (SSH)
Visual Studio Code (HTTPS)
IntelliJ IDEA (SSH)
IntelliJ IDEA (HTTPS)

Download source code
zip
tar.gz
tar.bz2
tar

Created on
May 07, 2024

Clonen von Projekten

- Clonen mittels SSH Key oder über HTTPS möglich
- Im gewünschten Zielverzeichnis den Befehl
`git clone` in Kombination mit der kopierten URL ausführen
- Zum Clonen über SSH muss ein öffentlicher SSH Schlüssel hinterlegt werden

SSH Keys

- Können zur verschlüsselten Kommunikation zwischen lokalem Repository und GitLab Server verwendet werden
- Sinnvolle Alternative zur wiederholten Verwendung von Username und Passwort
- SSH Keys besteht aus einem Schlüsselpaar
 - Öffentlicher Schlüssel wird auf den GitLab Server hinterlegt
 - Privater Schlüssel verbleibt geschützt auf Endgerät und wird nicht geteilt
- Unterstützte SSH Key Formate:
 - ED25519 (bevorzugt)
 - ECDSA
 - RSA (mind. 2048-bit)
 - DSA (nicht mehr empfohlen)

Clonen von Projekten

- SSH Schlüsselpaar generieren (ED25519)
 - Folgenden Befehl in Terminal ausführen:
`ssh-keygen -t ed25519 -C "<comment>"`
 - Dateiname und Speicherort angeben
 - Passphrase für Schlüssel einrichten (optional)
- Öffentlichen Schlüssel auf GitLab hochladen
 - Inhalt der erstellten .pub Datei kopieren
 - Navigieren **Startseite** → **Avatar** → **Edit profile** → **SSH Keys** (Sidebar)
 - **Add new key** auswählen
 - Öffentlichen Schlüssel in die Textbox kopieren
 - Optional **Usage Type** und **Expiration Date** festlegen
 - **Add key** auswählen

Clonen von Projekten

User Settings / SSH Keys

Search settings

SSH Keys

SSH keys allow you to establish a secure connection between your computer and GitLab. SSH fingerprints verify that the client is connecting to the correct host. Check the [current instance configuration](#).

Your SSH keys 3

Add an SSH key

Add an SSH key for secure access to GitLab. [Learn more](#).

Key

```
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIGkcePbmlo3PYxYHsrLZNFvYa97ZavFWhmuSAfrt79EF gitlab
```

Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'.

Title

Key titles are publicly visible.

Usage type

Expiration date

 X 

Optional but recommended. If set, key becomes invalid on the specified date.

Add key **Cancel**

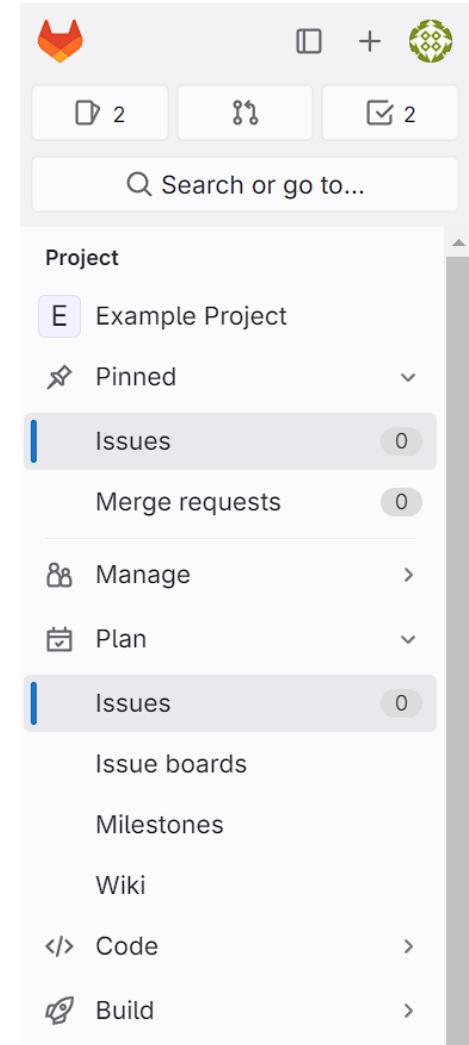
Title	Key	Usage type	Created	Last used	Expires	Actions
-------	-----	------------	---------	-----------	---------	---------

GitLab

Planing Tools

Issues

- In GitLab können innerhalb eines Projektes Issues für verschiedene Aufgaben angelegt werden
- Issues werden in der Menübar links sowohl als Shortcut als auch unter **Plan → Issues** angezeigt
- Issues können mit Labels versehen werden
 - Eigene Labels können beliebig unter **Manage → Labels** hinzugefügt werden
 - Default Set kann ebenfalls unter **Manage → Labels** generiert werden
 - Im Board können einzelne Listen für verschiedene Labels angelegt werden
 - Labels können priorisiert werden



Labels

Example User / Example Project / Labels

All Subscribed

Search Name New label

Labels can be applied to issues and merge requests. Star a label to make it a priority label.

Prioritized labels

Drag to reorder prioritized labels and change their relative priority.

No prioritized labels yet!

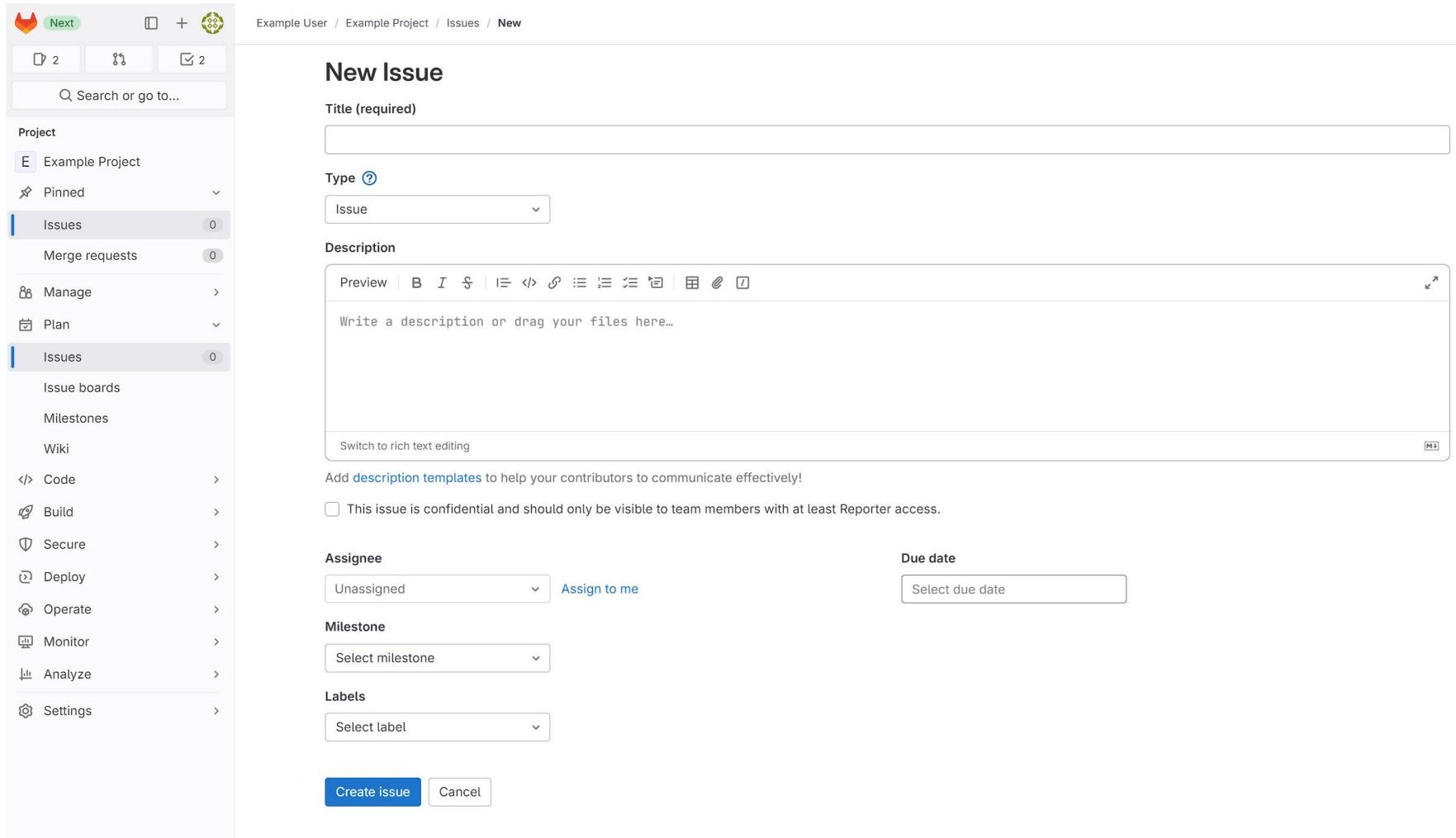
Star labels to start sorting by priority.

Other labels

Label	Count	Issues	Merge requests	Subscribe	⋮
bug	0	0	0	Subscribe	⋮
confirmed	0	0	0	Subscribe	⋮
critical	0	0	0	Subscribe	⋮
discussion	0	0	0	Subscribe	⋮
documentation	0	0	0	Subscribe	⋮
enhancement	0	0	0	Subscribe	⋮
suggestion	0	0	0	Subscribe	⋮
support	0	0	0	Subscribe	⋮

Help

Neues Issue anlegen



The screenshot shows the 'New Issue' creation interface in a web browser. The left sidebar contains navigation links for 'Example User / Example Project / Issues / New'. The main area is titled 'New Issue' and includes fields for 'Title (required)', 'Type (Issue selected)', 'Description' (with rich text editor preview), and optional 'Confidential' status. Below these are sections for 'Assignee' (Unassigned, Assign to me), 'Due date', 'Milestone' (Select milestone), and 'Labels' (Select label). At the bottom are 'Create issue' and 'Cancel' buttons.

New Issue

Title (required)

Type (Issue)

Description

Write a description or drag your files here...

Switch to rich text editing

This issue is confidential and should only be visible to team members with at least Reporter access.

Assignee

Due date

Milestone

Labels

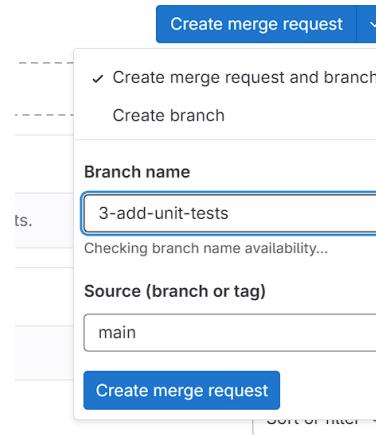
Create issue Cancel

Issues

The screenshot shows a GitLab interface for creating a new issue. The title is "Add unit tests". The top navigation bar includes "Example User / Example Project / Issues / #3". On the left, there's a sidebar with project navigation (Pinned, Issues, Merge requests, Manage, Plan, Issues), issue boards, milestones, and various operational tabs (Code, Build, Secure, Deploy, Operate, Monitor, Analyze, Settings). The main area has sections for "Child items" (0) and "Linked items" (0). Below these are activity logs showing recent changes: "Example User added testing label 7 minutes ago" and "Example User assigned to @Example_User 7 minutes ago". A rich text editor is available for comments, with options to "Switch to rich text editing", "Make this an internal note", "Comment", and "Close issue". To the right, there are fields for "Assignee" (set to "Example User"), "Labels" (with "testing" selected), "Milestone" (None), "Weight" (locked), "Due date" (None), "Time tracking" (No estimate or time spent), "Confidentiality" (Not confidential), and "Participants" (1 participant listed). A "Move issue" button is also present.

Issues

- Issues können an Personen „assigned“ werden, also zugewiesen werden
- Milestones, Fälligkeitsdatum und Time Tracking können ebenfalls hinzugefügt werden
- GitLab bietet die Möglichkeit, direkt aus Issues Merge-Requests oder Branches anzulegen



- Issues werden nach Erledigung geschlossen und damit in die „closed“ Sektion verschoben
- Templates für Issues können als Datei unter .gitlab/issue_templates mittels Markdown in eine .md Datei gespeichert werden

Issues

Example User / Example Project / Issues / #3

Add unit tests

Open Issue created 7 minutes ago by Example User

Edit ⋮ Mark as done »

Assignee Example User Edit

Labels testing x Edit

Milestone None Edit

Weight This feature is locked. [Learn more](#)

Due date None Edit

Time tracking ⌚ + No estimate or time spent Edit

Confidentiality ⓘ Not confidential Edit

1 Participant Move issue

Project

E Example Project

Pinned

Issues 3

Merge requests 0

Manage >

Plan >

Issues 3

Issue boards

Milestones

Wiki

Code >

Build >

Secure >

Deploy >

Operate >

Monitor >

Analyze >

Settings >

Child items 0 Show labels Add ⋮

Drag your designs here or [click to upload.](#)

No child items are currently assigned. Use child items to break down this issue into smaller parts.

Linked items 0 Add ⋮

Link issues together to show that they're related. [Learn more](#).

Activity

Sort or filter ⋮

- Example User added testing label 7 minutes ago
- Example User assigned to @Example_User 7 minutes ago

Preview | B I ♂ | ≡ ↔ ♂ ≡ ≡ ≡ ≡ | 田 ⌚ ⓘ ✉

Write a comment or drag your files here...

Switch to rich text editing Ⓜ

Make this an internal note ⓘ

Comment ⋮ Close issue

Issues

Issue-Liste

The screenshot shows the 'Issues' page of a GitLab project. The left sidebar includes navigation links for 'Project' (Example Project), 'Merge requests' (0), 'Manage', 'Plan', 'Issues' (3), 'Issue boards', 'Milestones', 'Wiki', and 'Code'. The main area displays three open issues:

- Add unit tests**: #3 · created just now by Example User **testing**
- Fix redirect after login**: #2 · created 2 minutes ago by Example User **bug critical**
- Redesign landing page**: #1 · created 2 minutes ago by Example User ◊ Release-v1.0.0 **enhancement**

At the bottom right, there is a link to "Email a new issue to this project".

- GitLab besitzt mit dem Issue Board ein Software Management Tool
- Kann zur Planung, Organisation und Visualisierung genutzt werden
 - Nutzung als Kanban oder Scrum Board
 - Effektive Aufgabenerfassung und Aufgabenverteilung
- Für Aufgaben werden einzelne Issues angelegt
 - Grundlegend in offene und geschlossen Labels aufteilt
- Board kann nach Labels strukturiert werden

Issue-Board

The screenshot shows a GitLab interface for an 'Example Project'. On the left, a sidebar navigation includes 'Project' (with 'Example Project' selected), 'Issues' (3 pinned), 'Merge requests' (0), 'Manage' (with 'Plan' and 'Issues' dropdowns), 'Issue boards' (selected), 'Milestones', 'Wiki', 'Code', 'Build', 'Secure', 'Deploy', 'Operate', 'Monitor', 'Analyze', and 'Settings'. At the bottom left is a 'Help' link. The main area displays an 'Issue Boards' view with three columns: 'Open' (containing three items: 'Redesign landing page', 'Fix redirect after login', and 'Add unit tests'), 'Closed' (empty), and a 'New list' button.

Development

Search

+ New list

Open

Redesign landing page

enhancement

#1 Release-v1.0.0

Fix redirect after login

bug critical

#2

Add unit tests

testing

#3

Closed

0

+ New list

Help

Issue-Board

Neue Liste anlegen

The screenshot shows a GitLab Issue Board interface with the following structure:

- Open** column (red box highlights the header):
 - Redesign landing page (enhancement)
 - Fix redirect after login (bug, critical)
 - Add unit tests (testing)
- Critical** column (red box highlights the header):
 - Fix redirect after login (bug, critical)
- Closed** column (red box highlights the header):
 - #2

At the top right, there is a red box highlighting the "New list" button.

Milestones

- Milestones können als Ziel angelegt werden
- Issues und Merge-Requests können an Milestones gekoppelt und dadurch gruppiert werden
- Besitzen optionalen Start- und Endpunkt

Neuen Milestone anlegen

The screenshot shows the 'New Milestone' creation interface. On the left, there's a sidebar with navigation links like 'Example User / Example Project / Milestones / New'. The main area has fields for 'Title', 'Start Date', 'Due Date', and a 'Description' rich text editor. Buttons at the bottom are 'Create milestone' and 'Cancel'.

Project

- E Example Project
- Pinned
- Issues 0
- Merge requests 0

Manage

- Plan
- Issues 0

Issue boards

- Milestones
- Wiki
- Code >
- Build >
- Secure >
- Deploy >
- Operate >
- Monitor >
- Analyze >
- Settings >

New Milestone

Title

Start Date

Due Date

Description

Preview | B I S | I E <> S P M H | Create milestone | Cancel

Write milestone description...

Switch to rich text editing

- Ermöglicht Dokumentation innerhalb des Projekts
- Wird in einem separaten Git-Repository gespeichert
 - Keine direkte Versionierung zusammen mit dem Code
- Ist in einzelne Seiten aufgeteilt, welche in der Sidebar rechts angezeigt werden
- Ermöglicht Strukturierung in Unterseiten

GitLab

Branches, Commits & Merging

- Beim Erstellen eines Projektes legt GitLab eine Default Branch an
 - Einstellungen bezüglich Default Branch können auf Projekt, (Sub-)Gruppen oder Instanz Level vorgenommen werden
- Default Branch wird als *protected* bezeichnet und kann nicht gelöscht werden
 - Mit Standardeinstellungen ist ebenfalls das force pushen auf einen Default Branch nicht möglich
 - Einstellungen hierzu können unter **Settings → Repository** feingranular vorgenommen werden
- Neue Branches können angelegt werden über
 - Projekt-Startseite
 - Unterpunkt Code → Branches
 - Über Issues
 - Lokal und per Push in GitLab übertragen
- Ist ein Branch abgeschlossen, wird üblicherweise ein Merge-Request (MR) gestellt, um Änderungen ins Projekt zu integrieren

Branches

The screenshot shows the 'Branches' page in the GitLab interface. On the left, there's a sidebar with project navigation (Example Users, Example Project, etc.) and repository details (Commits, Tags, etc.). The 'Branches' tab is selected. The main area has tabs for Overview, Active, Stale, and All, with 'Active' selected. A search bar and a 'View branch rules' button are at the top right. A modal window titled 'See all branch-related settings together with branch rules' is open, containing a message and 'View branch rules' and 'Dismiss' buttons. Below the modal, the 'Active branches' section lists three branches: '3-add-unit-tests' (protected), 'main' (default, protected), and 'feature/my_feature'. Each branch entry includes a commit history, a 'New' button, and a more options menu.

Project

E Example Project

Pinned

Issues (3)

Merge requests (1)

Manage >

Plan >

Code >

Merge requests (1)

Repository

Branches

Commits

Tags

Repository graph

Compare revisions

Snippets

Build >

Example Users / Example Project / Branches

Overview Active Stale All

Filter by branch name View branch rules New branch

① See all branch-related settings together with branch rules

You can now find an overview of settings for protected branches, merge request approvals, status checks, and security approvals conveniently in one spot.

[View branch rules](#) [Dismiss](#)

Active branches

Branch	Protected	Last Commit	Actions
3-add-unit-tests	Protected	66b511a1 · Add conflict file tests · 5 minutes ago	View Edit Delete More
main	Default Protected	212e031b · Add conflict file main · 5 minutes ago	View Edit Delete More
feature/my_feature	Protected	db15c779 · Add new file · 28 minutes ago	View Edit Delete More

Commits

- Commits bietet die Möglichkeit, alle Commits einer ausgewählten Branches und deren Änderungen anzuzeigen

The screenshot shows the GitLab interface for the 'example-project' repository. The left sidebar is open, showing various project management and code review sections like Pinned, Issues, Merge requests, Plan, and Code. The 'Commits' section is currently selected. At the top, there's a search bar and filters for 'Author' and 'View open merge request'. The main area displays a list of commits for the '3-add-unit-tests' branch. The commits are:

Commit Message	Author	Date	SHA
Add conflict file tests	Christopher Keutner	6 minutes ago	66b511a1
Add new file	Christopher Keutner	28 minutes ago	28094c25
Add new file	Christopher Keutner	30 minutes ago	db15c779
Initial commit	Christopher Keutner	5 hours ago	f60432af

Merge-Requests

- Merge-Requests sind eine Anfrage, um neue Änderungen in das Projekt einzubringen
- Git-Merge von Branches als Grundkonzept
- Erweitert einfachen Merge um folgende Features
 - Beschreibung des Merge-Request
 - Anzeige der Codeänderungen für Reviews
 - Auflistung der einzelnen Commits des Requests
 - Verknüpfungen mit Issues oder Milestones
 - Kommentarsektion zur Diskussion
- Bietet Option den Source Branch nach dem Merge zu löschen
- Ermöglicht Squashing von Commits
 - Zielbranch enthält dann nicht alle einzelnen Commits des Source Branches und bleibt ggf. übersichtlicher

Merge-Requests

Example User / Example Project / Merge requests / New

New merge request

Source branch

Example_User/example-project Select source branch

Select a branch to compare

Target branch

Example_User/example-project main

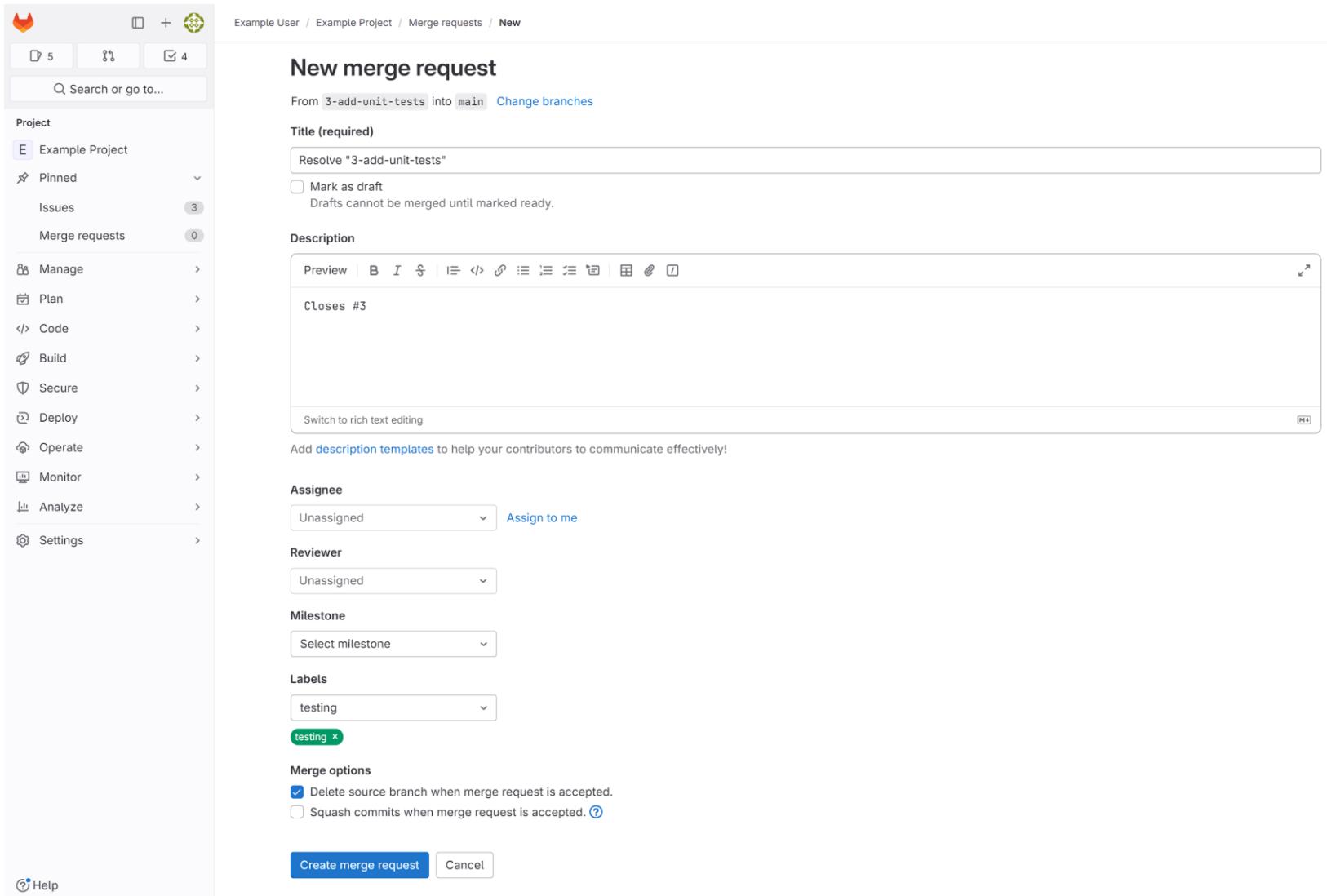
Add new file Example User authored May 27, 2024 db15c779

Compare branches and continue

Project

- E Example Project
- Pinned
- Issues 3
- Merge requests 0
- Manage >
- Plan >
- Code >
- Build >
- Secure >
- Deploy >
- Operate >
- Monitor >
- Analyze >
- Settings >

Merge-Requests



The screenshot shows a 'New merge request' form in a GitLab interface. The left sidebar lists project sections like Project, Issues, Merge requests, and various operational tabs. The main form includes fields for Title (required), Description (with rich text editor), Assignee (Unassigned), Reviewer (Unassigned), Milestone (Select milestone), Labels (testing), and Merge options (Delete source branch when merge request is accepted, Squash commits when merge request is accepted). Buttons at the bottom are 'Create merge request' and 'Cancel'.

Example User / Example Project / Merge requests / New

New merge request

From 3-add-unit-tests into main [Change branches](#)

Title (required)

Resolve "3-add-unit-tests"

Mark as draft
Drafts cannot be merged until marked ready.

Description

Closes #3

Switch to rich text editing

Add [description templates](#) to help your contributors to communicate effectively!

Assignee

Unassigned [Assign to me](#)

Reviewer

Unassigned

Milestone

Select milestone

Labels

testing

Merge options

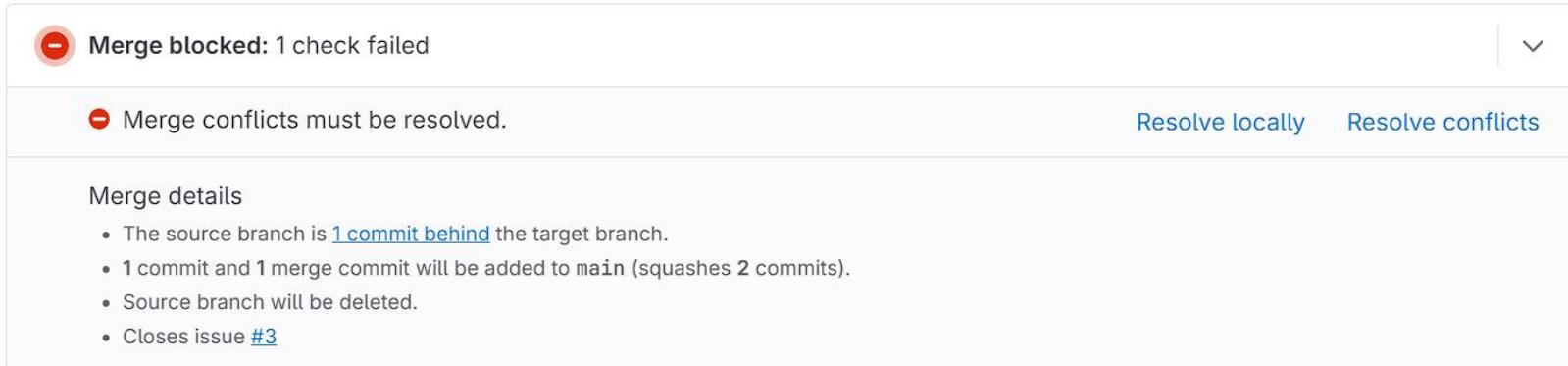
Delete source branch when merge request is accepted.
 Squash commits when merge request is accepted. [?](#)

[Create merge request](#) [Cancel](#)

- **Draft** – Merge Request ist für andere Personen einsehbar, kann aber noch nicht gemerged werden
- **Assignee** – Zugewiesene Person, die für den Merge-Request bzw. für das Einflegen der Änderungen zuständig ist
- **Reviewer** – Person(en), die die Änderungen reviewen und durch einen Approve bestätigen können. Approves können als verpflichtend sowie als Optional eingestellt werden
- **Milestone** – Zugehöriger Milestone
- **Labels** – Dienen zur Kategorisierung und Einordnung des MR
- **Merge Options**
 - Delete Source Branch → Source Branch wird nach Durchführung des MR gelöscht
 - Squash Commits → Commits des Source Branches werden gesquashed zum Target-Branch hinzugefügt

Merge-Requests

- GitLab versucht durch Templates den MR bereits mit Informationen anzureichern
 - So führt die 3 beim Branch 3-add-unit-tests im vorherigen Beispiel dazu, dass in die Beschreibung ein Closes #3 eingefügt wird und die Labels des Issues übernommen werden
 - Generelle Templates können als Datei unter .gitlab/merge_request_templates mittels Markdown in eine .md Datei gespeichert werden
- Merge-Konflikte werden in der Übersicht angezeigt



The screenshot shows a merge request interface with the following details:

- Merge blocked: 1 check failed**
- Merge conflicts must be resolved.** (with **Resolve locally** and **Resolve conflicts** buttons)
- Merge details:**
 - The source branch is [1 commit behind](#) the target branch.
 - 1 commit and 1 merge commit will be added to `main` (squashes 2 commits).
 - Source branch will be deleted.
 - Closes issue [#3](#)

Merge-Requests

- Auch bei Merge-Requests können, analog zu lokalen Konflikten, ebenfalls Merge-Konflikte auftreten
- Kleinere Konflikte können über GitLab aufgelöst werden, größere müssen jedoch lokal behoben werden

The screenshot shows a GitLab interface for resolving a merge conflict. On the left, the sidebar shows the user is in the 'Example Project' under 'Merge requests'. The main area is titled 'Resolve "3-add-unit-tests"' and shows a conflict in 'conflict_file.txt'. The conflict is between 'HEAD//our changes' (green) and 'main' (blue). The 'Interactive mode' section contains buttons for 'Use ours' (green) and 'Use theirs' (blue). Below this, there's a 'Resolve conflicts on source branch' section with instructions and a 'Commit message' field containing the command 'Merge branch \'main\' into \'3-add-unit-tests\''. At the bottom are 'Commit to source branch' and 'Cancel' buttons.

GitLab Übung

Aufgabe 1: Neues Projekt erstellen

1. (Optional) Beim Arbeiten mit GitLab ist es sinnvoll mittels SSH Keys auf Remote Repositories zuzugreifen, statt immer wieder Username und Passwort zu verwenden.

Falls Sie noch kein SSH Key in GitLab hinterlegt haben, so können Sie entweder die Anleitung aus dem GitLab nutzen oder dem Tutorial unter <https://docs.gitlab.com/ee/user/ssh.html> folgen.

2. Legen Sie ein neues privates Projekt **Playground** in Ihrem eigenen Bereich an, sodass nur Sie selbst Zugriff darauf haben. Achten Sie darauf, dass Sie das Repository mit einer Readme initialisieren, um dieses direkt clonen zu können.
3. Clonen Sie das Repository in ein lokales Verzeichnis auf Ihrem Rechner.

Hinweis

Man kann grundsätzlich viele Änderungen am Projekt sowohl lokal als auch in der GitLab Weboberfläche vornehmen. Für gewöhnlich bearbeitet man beispielweise eher selten Dateien in der GitLab Weboberfläche, sondern nimmt Änderungen lokal in einer IDE vor und pusht diese danach ins Remote Repository.

Um das Arbeiten mit mehreren Entwicklern im Projekt etwas nachzustellen, werden wir im Folgenden auch Änderungen über die Weboberfläche vornehmen, um damit Änderungen von anderen Entwicklern zu simulieren.

Dadurch lassen sich verschiedene Szenarien erzeugen, deren Behandlung wir an unserem lokalen Repository üben werden.

Aufgabe 2: Erste Schritte im Repository

1. Navigieren Sie in einem Terminal in das neue geklonte Projekt auf Ihrem Rechner.
2. Mit dem Befehl `git status` können Sie Ihren aktuellen Stand abfragen.
Sie sollten sich auf dem Branch **main** befinden, der sich auf demselben Stand wie **origin/main** befindet.
3. `git status` vergleicht nur den lokalen Remote Tracking Bereich mit Ihrem aktuellen Branch. Es gibt keinen Aufschluss darüber, ob Remote neue Änderungen verfügbar sind.
Nutzen Sie `git fetch`, um ihren lokalen Remote Tracking Bereich mit dem GitLab Repository zu synchronisieren.
Es sollten keine neuen Änderungen verfügbar sein.

Aufgabe 2: Erste Datei hinzufügen

1. Legen Sie eine Datei **random_numbers.sh** an und füllen Sie diese mit dem folgenden Inhalt

```
#!/bin/bash

random_number=$(( RANDOM % 10 + 1 ))

if [ $random_number -lt 11 ]; then
    echo "Number is less than 11"
fi
```

2. Committen Sie Ihre Änderungen auf dem **main** Branch.
3. Pushen Sie Ihre Änderungen ins Remote Repository. Vergewissern Sie sich über die Web-GUI, dass Ihr Push erfolgreich war.

Aufgabe 3: Issue erstellen

1. Über das Issue Board in GitLab können Aufgaben definiert und an bestimmte Personen zugewiesen werden.
Legen Sie im Issue Board ein neues Issue **Add output if number greater than 5** an.
Diese Aufgabe sollen Sie nun übernehmen, tragen Sie sich daher als Assignee ein.
2. GitLab bietet die Möglichkeit aus Issues direkt Merge-Requests und Branches zu erstellen.
Klicken Sie auf das Dropdown beim Button „Create merge request and branch“ und wählen Sie nur „Create branch“ aus.
Erstellen Sie nun eine Branch, um die Änderungen für das Feature vorzunehmen.

Aufgabe 4: Lokale Änderungen

1. Führen Sie in Ihrem lokalen Repository den Befehl `git status` sowie `git branch -a` aus.
Ihnen sollte der neu angelegte Branch aus GitLab nicht angezeigt werden, da Sie diesen erst vom Remote abrufen müssen.
2. Fetchen Sie alle Änderungen vom Remote.
`git branch -a` sollte nun den neuen Branch anzeigen.
3. Wechseln Sie in den Feature Branch.
4. Öffnen Sie die `random_numbers.sh` Datei und nehmen Sie die geforderten Änderungen vor, indem Sie die Datei um ein weiteres `if` ergänzen, indem geprüft wird, ob die Zahl größer als 5 ist.
5. Committen Sie Ihre Änderungen und pushen Sie diese ins Repository.

Aufgabe 5: Merge-Request

Nun wollen wir unsere Änderungen des Features in den main Branch übernehmen. Lokal würde man den Feature Branch in den main Branch mergen. Der Ablauf bei GitLab ist funktional sehr ähnlich, erweitert jedoch das reine Merging um einige Aspekte.

1. Navigieren Sie über die Sidebar in der GitLab Web-GUI auf den Punkt Merge-Requests.
2. Erstellen Sie einen neuen Merge-Request, um Ihren Feature Branch in den **main** Branch zu mergen.
Weisen Sie sich als Assignee und Reviewer zu (normalerweise würde man logischerweise jemand anderes als Reviewer eintragen).
3. Im Merge-Request haben Sie oben die Option, sich Commits und Changes anzuschauen. Reviewen Sie Ihre Änderungen und approven Sie den Merge-Request.

Aufgabe 5: Merge-Request

4. Mergen Sie nun den Merge-Request mit der Option „Delete source branch“ und verifizieren Sie, dass die Änderungen auf dem **main** Branch vorhanden sind.
5. Löschen Sie Ihren lokalen Branch, da dieser abgeschlossen ist und keine Verwendung mehr besitzt.

Aufgabe 6: Rebase von lokalen Branches

Nun wollen wir ein Rebase nutzen, um Remote Änderungen in unseren lokalen Branch zu integrieren.

1. Erstellen Sie ein neues Issue **Add output if number is maximum**. Dabei soll eine Ausgabe erfolgen, wenn die Random Nummer die größtmögliche Zahl ist.
2. Man muss nicht zwingend den Branch aus einem Issue heraus erstellen. Legen Sie dieses Mal lokal einen Branch **2-add-output-if-max** an und führen Sie die notwendigen Änderungen durch.
3. Committen Sie Ihre Änderungen und pushen Sie diese zum Remote Repository.

Aufgabe 6: Rebase von lokalen Branches

In der Zwischenzeit hat ein Kollege die Aufgabe bekommen, den Zahlenbereich von 10 auf 1000 zu erhöhen.

Um die Änderungen von Ihm zu simulieren, führen wir diese über die Web-GUI durch. Ein zugehöriges Issue bzw. einen Branch überspringen wir zur Simplifizierung.

Hinweis Ein gemeinsames Arbeiten über eine Datei sollte im Optimalfall vermieden werden, um Konflikten zu vermeiden.

4. Ändern Sie über GitLab den Random Number Bereich auf 1000 und committen Sie die Datei direkt aus dem Editor auf den **main** Branch.
5. Um im lokalen Repository die Änderungen zu übernehmen, müssen Sie den **main** Branch updaten. Wechseln Sie dazu in diesen und verwenden Sie `git pull`, um diese in Ihren lokalen **main** Branch zu übernehmen.

Aufgabe 6: Rebase von lokalen Branches

7. Um die Änderungen nun auch in den Feature Branch zu übernehmen, müssen Sie in den Feature Branch wechseln und dort auf den **main** Branch rebasen.
8. Rebasen Sie Ihren Feature Branch auf den **main** Branch. Dabei sollten keine Konflikte auftreten, da sich die betreffenden Zeilen bei Ihnen nicht geändert haben.
9. Passen Sie Ihre Ausgabe bezüglich der Random Number an und committen Sie Ihre Änderungen.
10. Versuchen Sie Ihren Branch zum Remote zu pushen. GitLab wird hierbei den Push wegen nicht zueinander passenden Commit-Historien ablehnen. Pushen Sie daher Ihren Branch mit der Option **--force**.

Aufgabe 6: Rebase von lokalen Branches

Wichtig --force bewirkt, dass der Remote Branch durch die lokale Version überschrieben wird.

Falls andere Entwickler auf diesem Branch arbeiten würden, so könnten Sie weder push noch pull auf diesem ausführen, da die Commit-Historien nicht vereinbar sind. Ihre Branches sind damit kaputt und müssen komplett neu vom Remote abgerufen werden.

Daher niemals auf Public-Banches rebasen!

Aufgabe 7: Merge-Konflikte

In dieser Aufgabe wollen wir uns das Auflösen von Merge-Konflikten anschauen.

Ein Kollege von Ihnen hat ebenfalls das Feature bezüglich Ausgabe bei maximaler Zahl umgesetzt und dieses schon in den **main** Branch integriert.

1. Simulieren Sie die Änderungen des Kollegen, indem Sie auch hier wieder über die Web-GUI direkt auf dem **main** Branch arbeiten.
Kopieren Sie Ihre lokalen Änderungen bezüglich der Ausgabe und fügen sie diese über die Web-GUI in die **random_numbers.sh** Datei ein.
Verändern Sie dabei die Konsolenausgabe, sodass sich die Änderungen unterscheiden.
2. Committen Sie Ihre Änderungen direkt auf **main**.

Aufgabe 7: Merge-Konflikte

3. Stellen Sie für Ihren Feature Branch ein Merge-Request. Nach der Erstellung sollte GitLab anzeigen, dass es Konflikte beim Mergen gibt.
4. Kleinere Konflikte lassen sich über die Web-GUI auflösen, größere Konflikte lassen sich nur lokal auflösen.
Zur Übung lösen wir den Konflikt lokal auf.
5. Updaten Sie den **main** Branch auf den neusten Stand aus dem Remote Repository.
6. Lösen Sie die Konflikte nun, indem Sie entweder den **main** Branch in Ihren Feature Branch mergen oder indem Sie den Feature Branch auf den aktuellen Stand des **main** rebasen. Da Sie alleine auf Ihrem Branch arbeiten, bietet sich auch hier ein Rebbase an.
7. Lösen Sie den Konflikt, indem Sie sich für Ihre Ausgabe entscheiden.

Aufgabe 7: Merge-Konflikte

8. Pushen Sie nach Auflösen des Konflikts Ihre Änderungen ins GitLab.
9. Der Merge-Request sollte nun ohne Konflikte umsetzbar sein. Mergen Sie Ihr Feature in den **main** Branch, um die Änderungen abzuschließen.