



GitLab als alternative Entwicklungsplattform zu Github.com

Chemnitzer Linux-Tage 2015

21. März 2015



Ralf Lang
Linux Consultant/Developer
B1 Systems GmbH
lang@b1-systems.de

GitLab – Open-Source-Unterstützung beim gemeinsamen Coden



Warum Versionsverwaltung?

- einfaches Rechtemanagement
- integrierte Feedbackoptionen
- Reviews erzwingen
- Zeit für's Coden gewinnen – kein umständliches Management von Tools und Benutzern

Warum ein eigenes VCS?

- strikte Zugriffskontrolle auf den Source Code
- niemand Drittes involviert
- dedizierte Ressourcen
- einfach einzufügen in Umgebungen mit speziellen Tools und Reports
- sensible Daten bleiben im Haus

Was ist GitLab?

- ... eine Open-Source-Software zum gemeinsamen Entwickeln von Code
- ... ein Abkömmling von Git, dem am weitesten verbreiteten VCS für Softwareentwicklung
- ... ein Qualitätsgarant durch sein umfangreiches Reviewsystem

⇒ GitLab vereinfacht verteiltes Arbeiten an Projekten mit einem zentralen Server.

GitLab Features im Überblick

- Code Review
- Bug Tracking
- persönliche und private Branches
- GitLab kann mehrere Git Repositories verwalten
- 25.000 Benutzer auf einem Server
- Hochverfügbarkeit im active/active Cluster möglich
- Code Snippets
- Access Control
- Issue Tracking
- Web Hooks
- Wiki

GitLab ist kollaborativ, weil . . .

- . . . es eine unbegrenzte Anzahl öffentlicher oder privater Repositories unterstützt
- . . . es eine unbegrenzte Anzahl öffentlicher oder privater Projektmitglieder ermöglicht
- . . . es LDAP-Integration bietet
- . . . es sich in externe Ticketsysteme integrieren lässt (z. B. Redmine)
- . . . das Omnibus Paket die Konfiguration externer Datenbanken wie *PostgreSQL* oder *MySQL* unterstützt
- . . . mit *JIRA* zum Issue Tracking benutzt werden kann
- . . . es den Merge Request Status für Builds auf *Jenkins CI* anzeigen kann (nur Enterprise Edition)

Wer verwendet GitLab?

- Mehr als 100.000 Organisationen, wie z. B.:
 - AT&T
 - Bell
 - CERN
 - Fraunhofer
 - Interpol
 - Jülich Research Center
 - NASA
 - O'Reilly Media
 - Red Hat
 - ...

Das GitLab-Projekt

- aktiv seit September 2011
- über 700 aktive Entwickler
- verwaltet von GitLab.com
- Open Core Business Model:
 - GitLab EE (Enterprise Edition):
Enterprise-Version unter proprietärer Lizenz mit Support durch GitLab B.V.
 - GitLab CE (Community Edition):
Open-Source-Version unter der MIT License

Herkömmlicher Git-Workflow

- 1 Repository klonen
- 2 Branch anlegen
- 3 Source Code ändern
- 4 Einchecken
- 5 Patch anlegen oder Änderungen Upstream pushen

Nachteile des herkömmlichen Git-Workflows

- Jeder Mitarbeiter braucht für seine Commits Schreibzugriff auf die Projekte.
⇒ beabsichtigte Workflows können umgangen werden (fehlende Absicherung durch Rechte)
- Patchbearbeitung (*format-patch*):
Jeder Mitarbeiter muss nach Einreichung seiner Patches auf eine manuelle Bearbeitung durch einen Reviewer warten.
⇒ immer noch ein arbeits- und zeitintensiver sowie fehleranfälliger Prozess

Der GitLab Workflow

- 1 Fork des Repositories in den eigenen Namensraum
- 2 Vollzugriff auf die eigene geforkte Kopie erteilen
- 3 Online im Browser oder im lokalem Checkout editieren
- 4 Merge-Request einreichen: \Rightarrow Die Änderungen gehen online und können diskutiert werden; der Reviewer kommentiert Diffs.
- 5 Automatisierter Prozess zur Re-Integration von Forks ins Mainstream-Repository

\Rightarrow keine Notwendigkeit, Zugriffsrechte zu geben oder zu entziehen

\Rightarrow keine endlosen Threads mit Patchmails

\Rightarrow entspricht den Paradigmen für Code-Reviews:

- Unabhängigkeit der Entwicklung von einzelnen Teammitgliedern
- leichte Integration neuer Teammitglieder
- Code-Reviews helfen Bugs aufzudecken

Internes Issue/Review System

- APIs für externes Ticketing
 - Zugriff auf Redmine Tickets über Git Commit Messages
 - alternativ: internes Ticketing
- API für Gitlab CI
 - *Continuous Integration*:
 - automatisierte Builds
 - durch Commits getriggerte Testsuite
 - *Dead Code Detectors* (DCD) entfernen überflüssigen Code.
 - *Code Coverage* (CC) Tools helfen die Testabdeckung aufzuzeigen (funktionale Tests, Sicherheitstests).
⇒ bessere Qualität der Software

Zugriffskontrolle

- GitLab bietet Zugriffskontrolle für Benutzer und Gruppen basierend auf unterschiedlichen Rechte-Levels.
- Befugnisse eines Benutzers hängen ab von seinem Zugriffslevel auf ein bestimmtes Projekt oder eine Gruppe.
- Gehört ein Benutzer gleichzeitig einer Projektgruppe und dem Projekt selbst an, wird das höhere Rechte-Level angewandt.
- Der GitLab Administrator erhält alle Rechte.

GitLab Continuous Integration

- fügt sich in die GitLab-Installation ein, um Tests für Projekte laufen zu lassen
- Login mit GitLab-Account
- neue Projekte mit einem Klick hinzufügen
- Installation auf eigener Hardware („On-Premises“)
- kann auf allen beliebigen Linux-Servern installiert werden

GitLab Dashboard – Projektüberblick

Dashboard


Projects
Issues 0
Merge Requests 0
Help







My Projects


All projects you have access to are listed here. Public projects are not included here unless you are a member


All	32	🔒 development / b1 [redacted]	Last activity: about 5 hours ago
Personal	0		
Joined	32	🔒 development / b [redacted]	
Owned	0		Last activity: 6 days ago
Groups			
		🔒 development / B1 Open Build Service	
		[redacted] - OUR open build service related stuff.	
📁 development	32		Last activity: about 1 month ago









GitLab Dashboard – Commits

 development / raspi-video
Search in this project

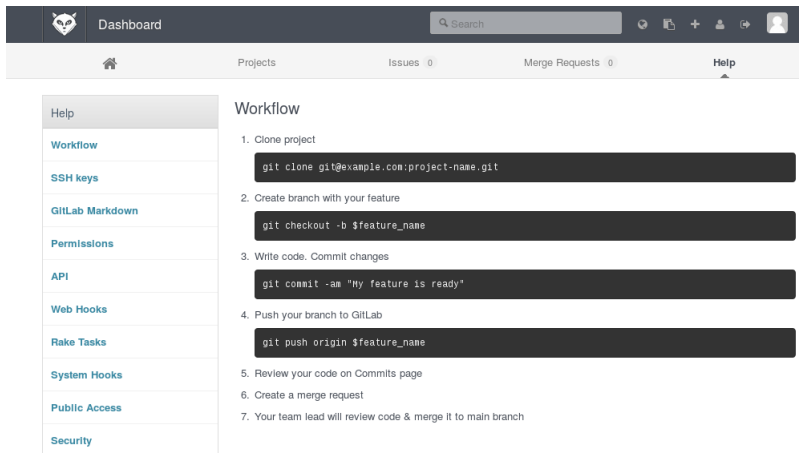








Files
Commits
Network
Graphs
Issues 0
Merge Requests 0

master ▾
Commits
Compare
Branches 1
Tags 0
Stats


 25 Jul, 2014 2 commits	<div style="border-bottom: 1px solid #ccc; padding-bottom: 5px;"> 339412c8b add ffmpeg patch for completeness  Stefan Seyfried </div> <div style="border-bottom: 1px solid #ccc; padding-bottom: 5px;"> 4ad8d74db stream-receiver.sh: add newly created -ftgs option  Stefan Seyfried </div>	Browse Code » 2 months ago
 10 Jun, 2014 1 commit	<div style="border-bottom: 1px solid #ccc; padding-bottom: 5px;"> 7c165546b add hacked hello_video version, built as hello_video2.bin  Stefan Seyfried </div>	Browse Code » 4 months ago
 08 Jun, 2014 2 commits	<div style="border-bottom: 1px solid #ccc; padding-bottom: 5px;"> cda6812dc raspi: add Makefile for easy build (cross and native)  Stefan Seyfried </div> <div style="background-color: #e6f2ff; padding-bottom: 5px;"> fe90555db import ilclient and hello_video from raspi userland  Stefan Seyfried </div>	Browse Code » 4 months ago Browse Code » 4 months ago

GitLab Dashboard – Workflow



The screenshot shows the GitLab Dashboard interface. At the top, there is a dark navigation bar with the GitLab logo, the word "Dashboard", a search bar, and several utility icons. Below this is a light-colored secondary navigation bar with links for "Home", "Projects", "Issues 0", "Merge Requests 0", and "Help". On the left side, there is a vertical sidebar menu with the following items: "Help", "Workflow", "SSH keys", "GitLab Markdown", "Permissions", "API", "Web Hooks", "Rake Tasks", "System Hooks", "Public Access", and "Security". The main content area is titled "Workflow" and contains a numbered list of seven steps, each with a corresponding terminal command in a dark box:

1. Clone project








```
git clone git@example.com:project-name.git
```
2. Create branch with your feature

```
git checkout -b $feature_name
```
3. Write code. Commit changes

```
git commit -am "My feature is ready"
```
4. Push your branch to GitLab

```
git push origin $feature_name
```
5. Review your code on Commits page
6. Create a merge request
7. Your team lead will review code & merge it to main branch

GitLab Dashboard – Neues Projekt

 New Project      

Project name is [Customize repository name?](#)

Repository name

[Import existing repository?](#)

Description (optional)

Public project Make project visible to everyone

GitLab Dashboard – Projekte

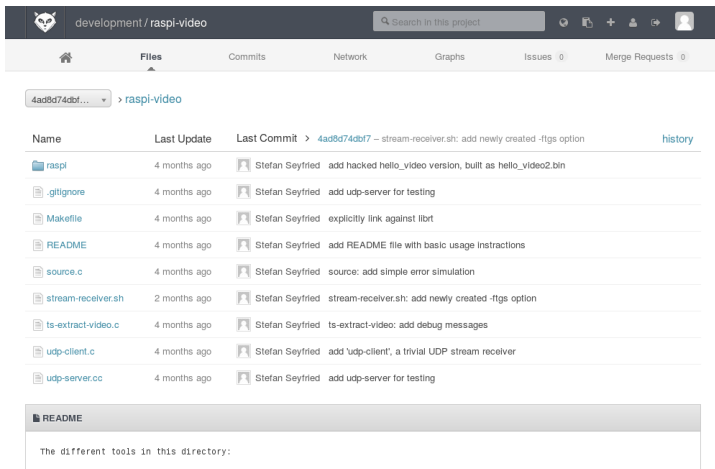
The screenshot shows the GitLab interface for a project named 'development / raspi-video'. At the top, there is a search bar and navigation icons. Below that, a horizontal menu contains 'Files', 'Commits', 'Network', 'Graphs', 'Issues 0', and 'Merge Requests 0'. A secondary bar shows 'SSH' and 'HTTPS' links, a 'private' status indicator, and 'Fork' and 'Download' buttons. The main content area features a list of recent activity:

- [Stefan Seyfried](#) pushed new branch **master** at development / raspi-video 2 months ago.
- [Stefan Seyfried](#) joined project at development / raspi-video 3 months ago.
- [Karsten Keil](#) joined project at development / raspi-video 3 months ago.

On the right side, a summary box for 'raspi-video' provides the following details:

- Repo size is 0.79 MB
- Created at Jul 04, 2014
- Owned by [development Group](#)
- 22 commits
- 1 branch
- 0 tags

GitLab Dashboard – Projektdateien



The screenshot shows the GitLab interface for a project named 'development / raspi-video'. The top navigation bar includes a search box and icons for home, files, commits, network, graphs, issues, and merge requests. Below the navigation bar, the current commit hash '4ad8d74dbf...' is displayed, followed by the project name 'raspi-video'. A table lists the project's files and their commit history:

Name	Last Update	Last Commit	Commit Message	History
raspi	4 months ago	Stefan Seyfried	add hacked hello_video version, built as hello_video2.bin	history
.gitignore	4 months ago	Stefan Seyfried	add udp-server for testing	
Makefile	4 months ago	Stefan Seyfried	explicitly link against librt	
README	4 months ago	Stefan Seyfried	add README file with basic usage instructions	
source.c	4 months ago	Stefan Seyfried	source: add simple error simulation	
stream-receiver.sh	2 months ago	Stefan Seyfried	stream-receiver.sh: add newly created -flgs option	
ts-extract-video.c	4 months ago	Stefan Seyfried	ts-extract-video: add debug messages	
udp-client.c	4 months ago	Stefan Seyfried	add 'udp-client', a trivial UDP stream receiver	
udp-server.cc	4 months ago	Stefan Seyfried	add udp-server for testing	

Below the table, there is a section for the 'README' file, which contains the following text:

```
The different tools in this directory:
```

Mehr Informationen zu GitLab ...

- GitLab.com:
`GitLab.com`
- GitLab Continuous Integration (CI):
`https://about.gitlab.com/gitlab-ci/`
- Die offizielle GitLab Dokumentation:
`http://doc.gitlab.com/ce/`

Vielen Dank für eure Aufmerksamkeit!

Bei weiteren Fragen wendet euch bitte an info@b1-systems.de oder
+49 (0)8457 - 931096