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 Committs Schreibzugriff auf die Projekte.
  $\Rightarrow$ 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.
  $\Rightarrow$ 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).
    $\Rightarrow$ bessere Qualität der Software

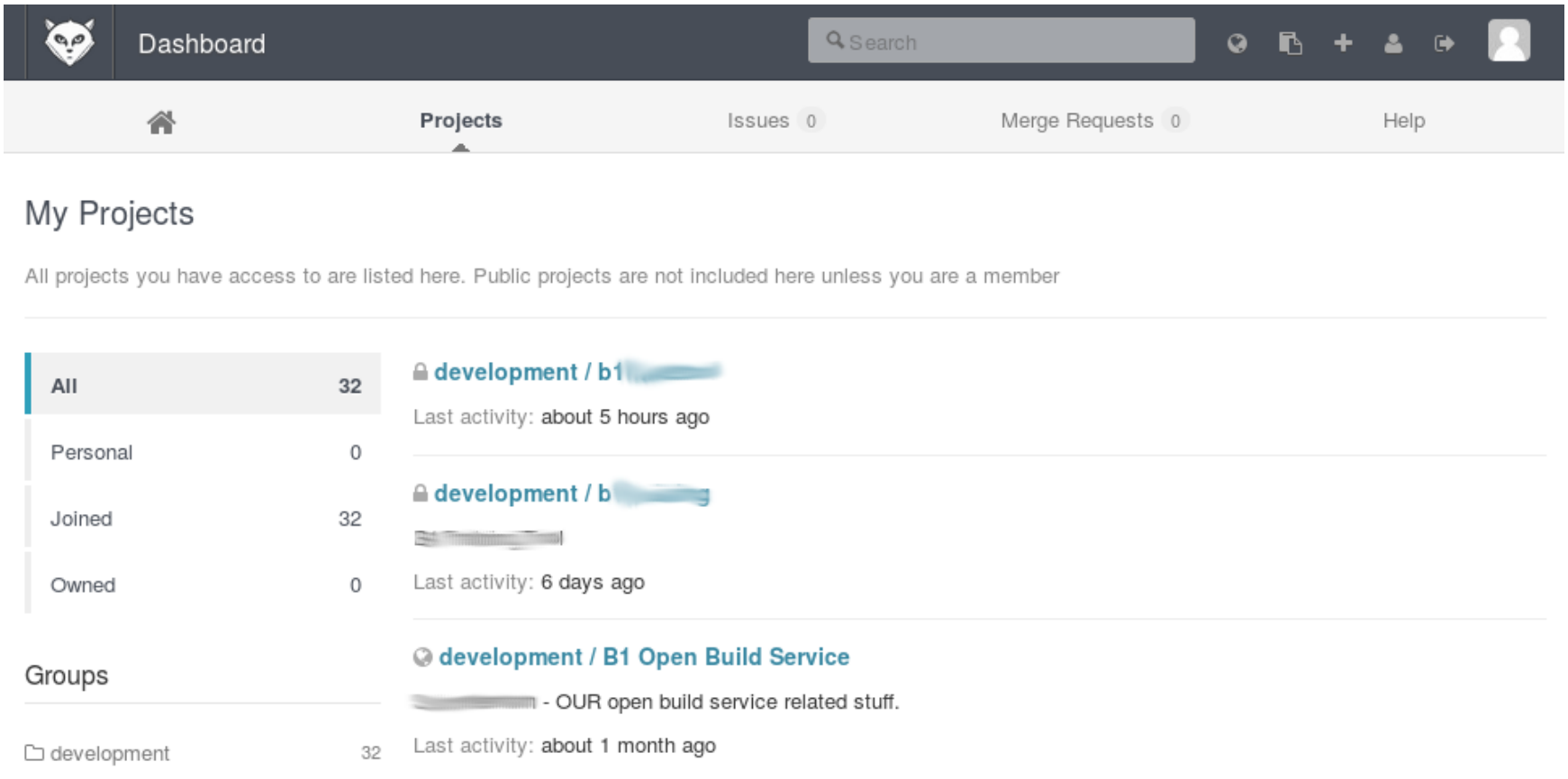
Zugriffskontrolle

- 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.

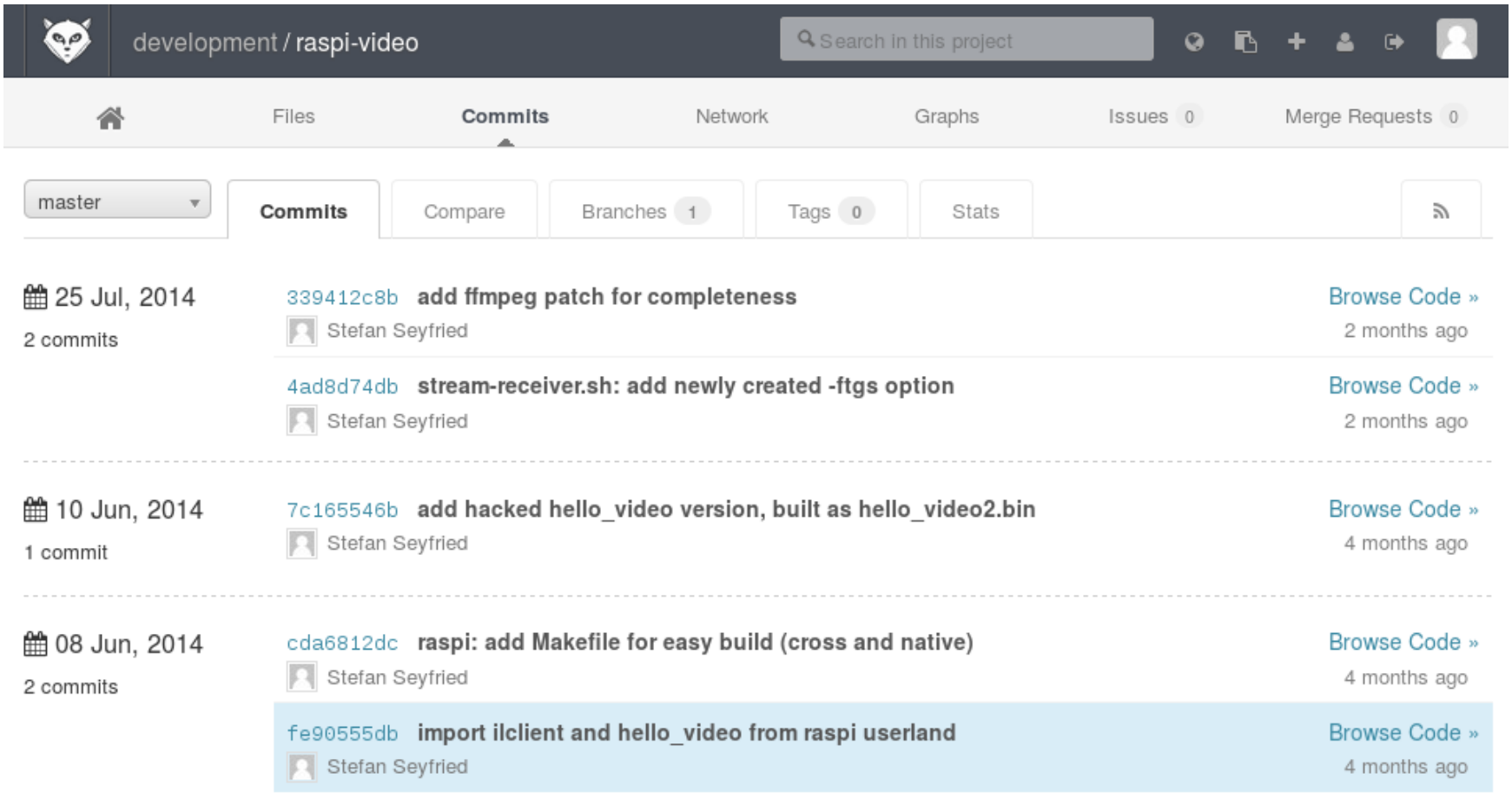
GitLab Continuous Integration

- 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

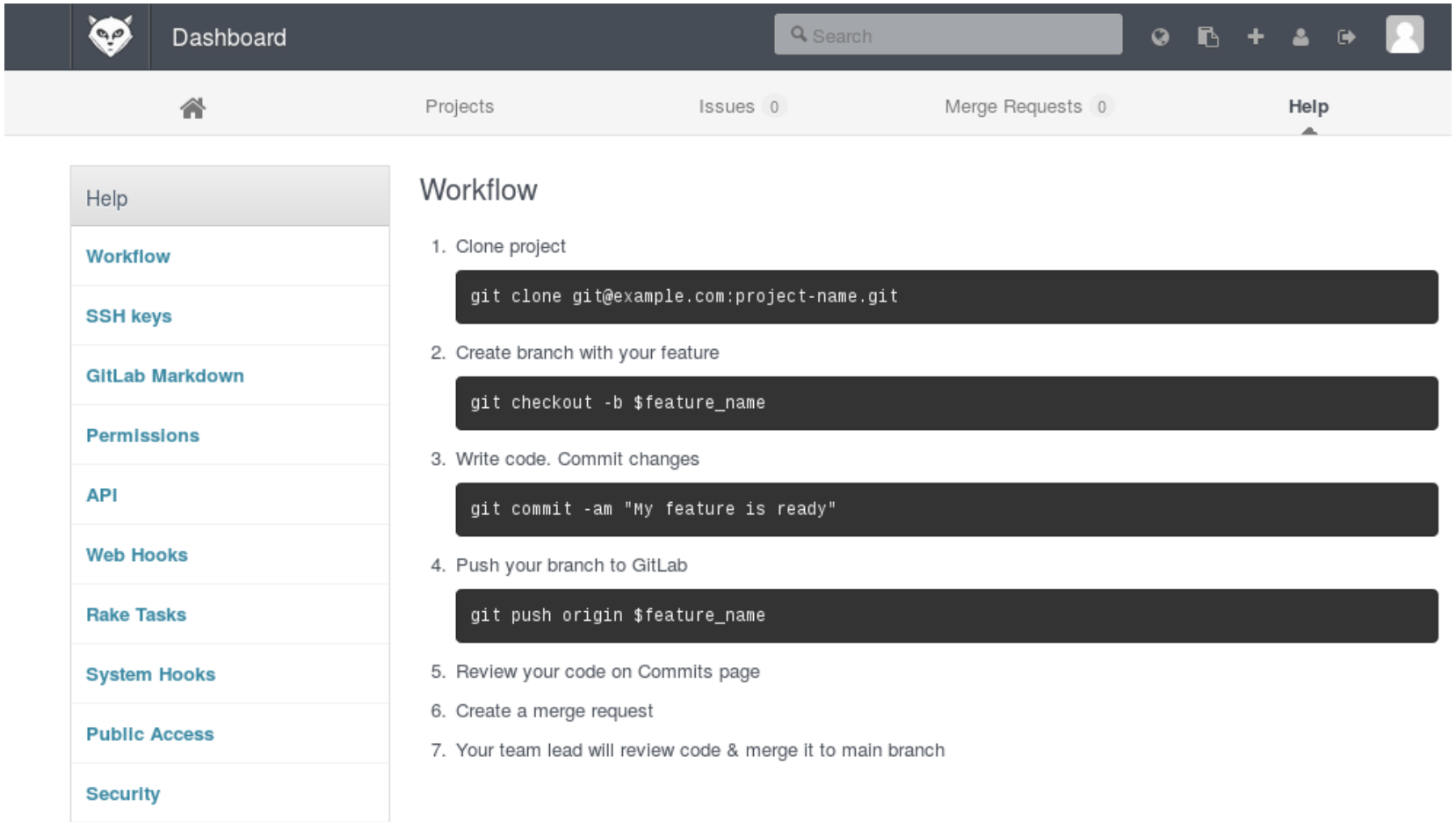
GitLab Dashboard – Projektüberblick



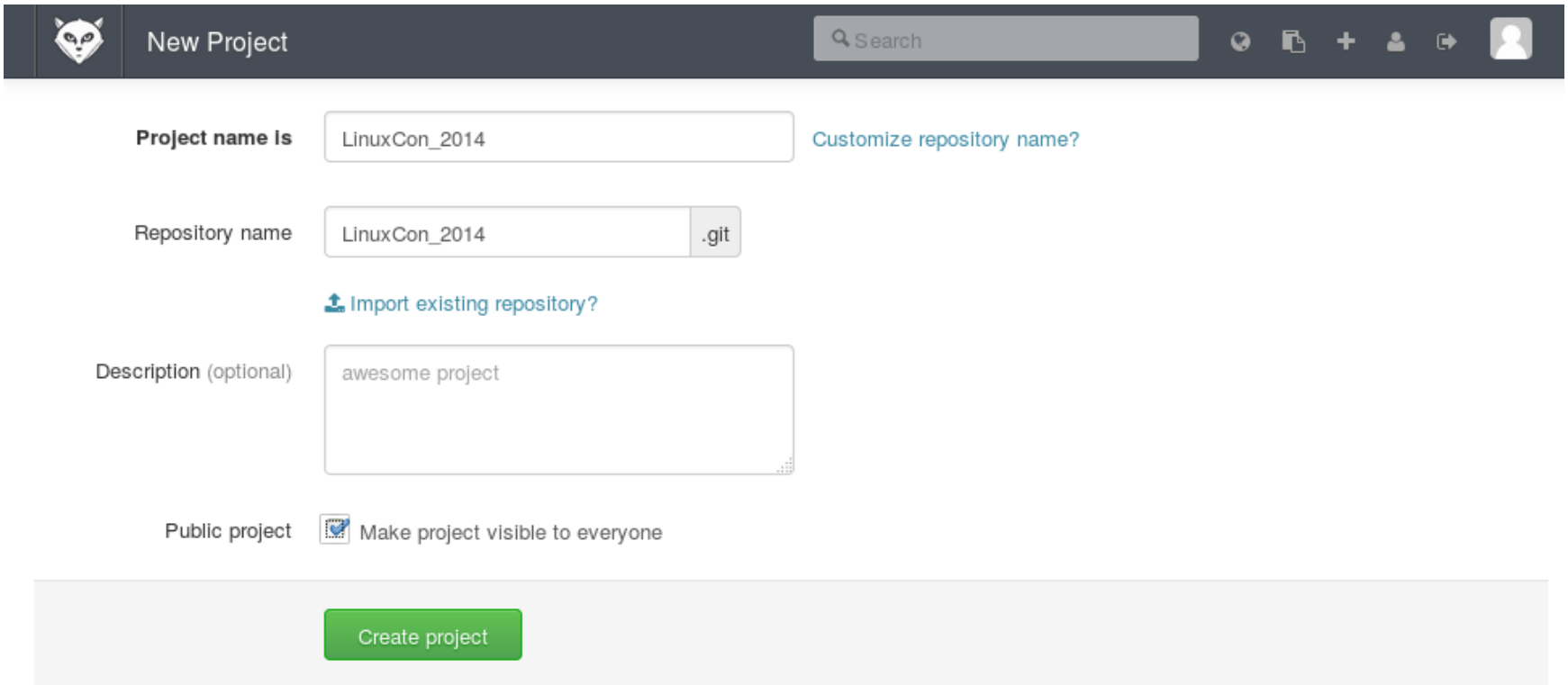
GitLab Dashboard – Commits



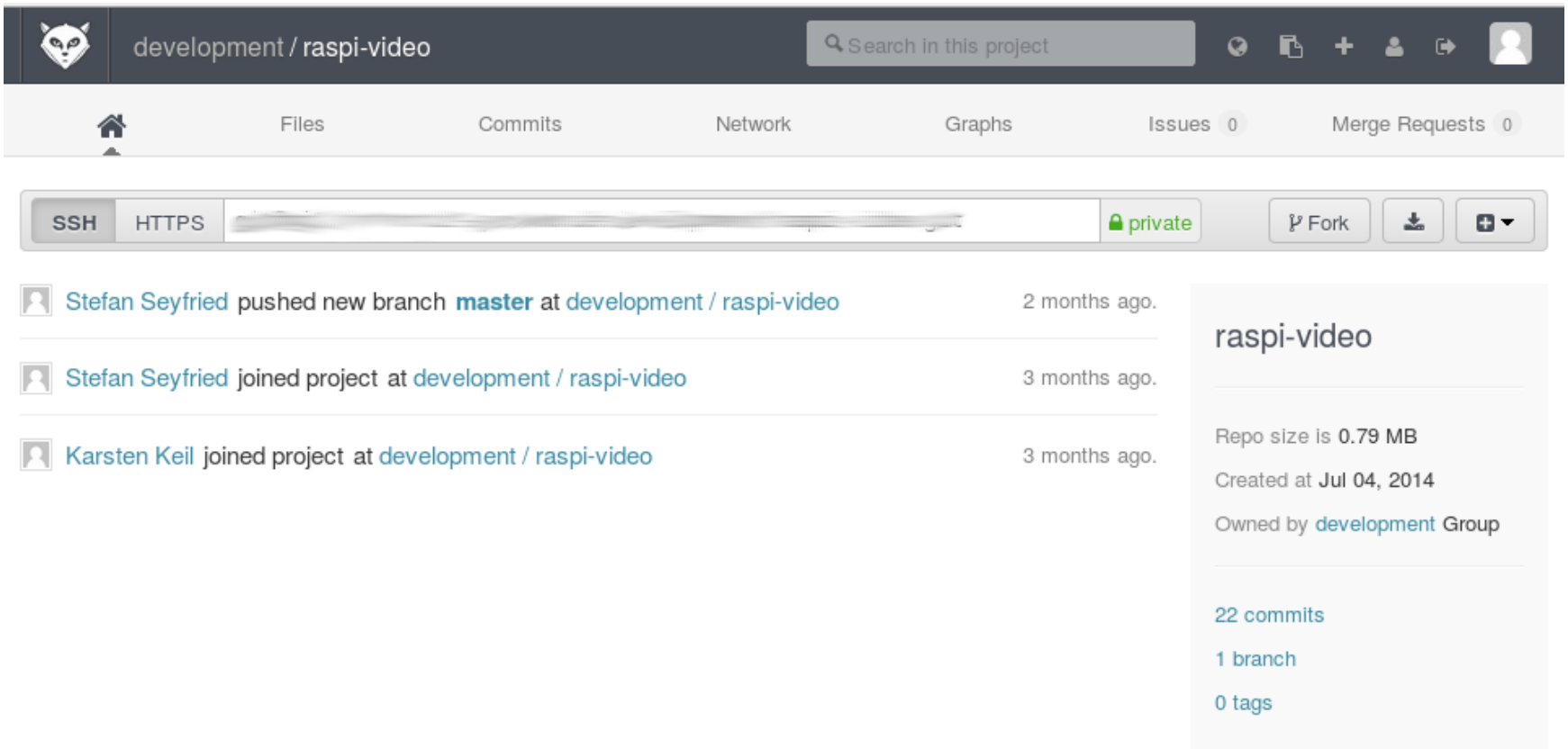
GitLab Dashboard – Workflow



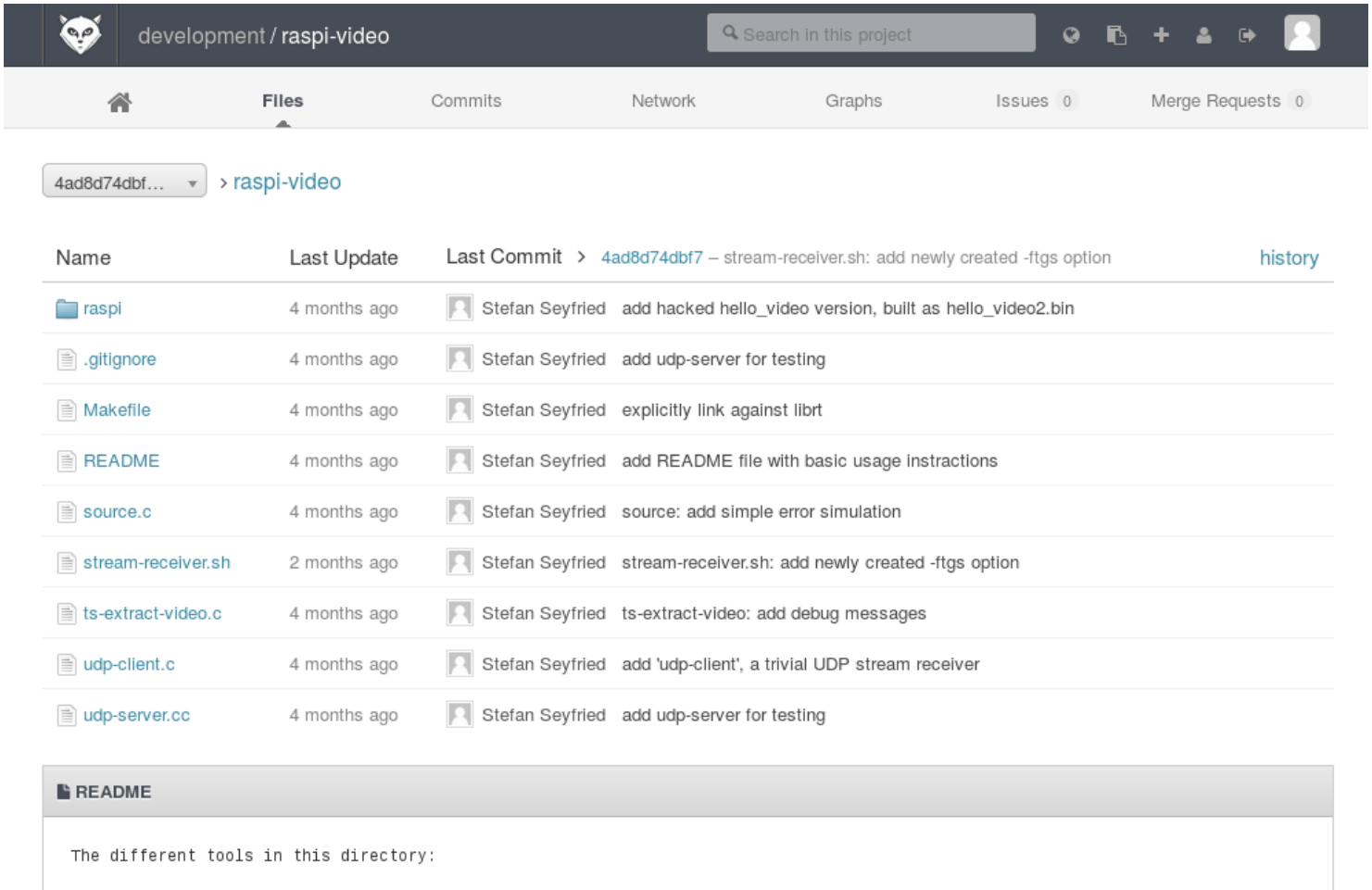
GitLab Dashboard – Neues Projekt



GitLab Dashboard – Projekte



GitLab Dashboard – Projektdateien



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