






integrata  
cegos

# GitLab

Einführung




## A full DevOps toolchain.\*

GitLab is a single application for the entire software development lifecycle. From project planning and source code management to CI/CD, monitoring and security.

*\*No assembly required.*

[Try GitLab for Free](#)

[▶ Watch a demo](#)



Quick answers to common problems

## GitLab Cookbook

Over 60 hands-on recipes to efficiently self-host your own Git repository using GitLab

Jeroen van Baarsen

**[PACKT]** open source\*  
PUBLISHING community experience distilled

- Die in diesem Seminar verwendete Werkzeuge und Frameworks sind Open Source
  - LPGL Lizenzmodell
- Dies ist ein Programmier-Seminar
  - Damit werden die Inhalte durch Übungen vertieft und verinnerlicht
  - Musterbeispiele werden zur Verfügung gestellt
  - Diese können am Ende des Seminars als ZIP-Datei kopiert werden
    - USB-Stick oder ähnliches
- Dokumentation und Ressourcen stehen auch im Internet zur Verfügung
- Konventionen
  - Befehle werden in `Courier-Schriftart` dargestellt
  - Dateinamen werden in *`Courier-Schriftart`* dargestellt
  - Links werden in `unterstrichener Courier-Schriftart` dargestellt

© Javacream

Javacream

Dr. Rainer Sawitzki

Alois-Gilg-Weg 6

81373 München

**Alle Rechte, einschließlich derjenigen des auszugsweisen Abdrucks, der fotomechanischen und elektronischen Wiedergabe vorbehalten.**

Einführung in die GitLab Administration	6
Git	29
Projektorganisation	40
GitLab Konfiguration	50
Betriebliche Aspekte	58
Continuous Integration	69
Weitere Themen	75

1

# EINFÜHRUNG IN DIE GITLAB ADMINISTRATION

1.1

# **ALLGEMEINES VORGEHEN**

- GitLab ist ein komplexes Produkt, welches sich nicht theoretisch vermitteln lässt
- Die Web Konsole sowie die darin integrierte Konsole bieten eine hervorragende Möglichkeit, GitLab-Funktionen direkt in Aktion zu sehen
  - Und damit nachhaltig zu lernen



1.2

## **FUNKTIONSUMFANG**

- Ein Server zum Zugriff auf Git-Repositories
  - Inklusive Authentifizierung, Berechtigungen und Verschlüsselung
- Eine Implementierung von Workflows, die über reine Git-Funktionen hinaus gehen
- Ein Build-Server für Continuous Integration/Delivery/Deployment
- Ein Issue-Tracking-System

- Übersicht
  - Core
  - Starter
  - Premium
  - Ultimate
- Die Core-Edition ist frei
  - Die Lizenz-pflichtigen Distributionen ergänzen weitere Plugins
  - <https://about.gitlab.com/pricing/self-managed/feature-comparison/>

1.3

## **INSTALLATION**

- Eine native GitLab -Installation wird nur für Linux unterstützt
- Bitnami stellt fertig konfigurierte Images für VMWare Player und Virtual VBox zur Verfügung
  - <https://bitnami.com/stack/gitlab/virtual-machine>
- Ein Docker-Image ist auf DockerHub vorhanden
  - <https://hub.docker.com/r/gitlab/gitlab-ee/>
  - Auch die Docker-Variante wird aktuell nicht für einen Windows Host empfohlen

- **Standalone**

- <https://gitlab.com/gitlab-org/omnibus-gitlab/blob/master/doc/README.md>

- **Docker**


- <https://gitlab.com/gitlab-org/omnibus-gitlab/blob/master/doc/docker/README.md>








- Die bevorzugte Distribution
- Eine zentrale Installation aller beteiligten Komponenten
  - Web Server
  - Ruby-Interpreter
  - Datenbank
- Vereinheitlichte Konfiguration
  - `gitlab-ctl <Options>`
    - Gültige Optionen mit `gitlab-ctl --help`

## ■ Container-Definition

```
sudo docker run --detach \  
  --hostname gitlab.example.com \  
  --publish 9443:443 \  
  --publish 9080:80 \  
  --publish 9022:22 \  
  --name gitlab \  
  --restart always \  
  --volume /srv/gitlab/config:/etc/gitlab \  
  --volume /srv/gitlab/logs:/var/log/gitlab \  
  --volume /srv/gitlab/data:/var/opt/gitlab \  
  gitlab/gitlab-ee:latest
```





Projects ▾ Groups ▾ More ▾  ▾ Search or jump to...      ▾  ▾



## Projects

New project

**Your projects** Starred projects Explore projects 

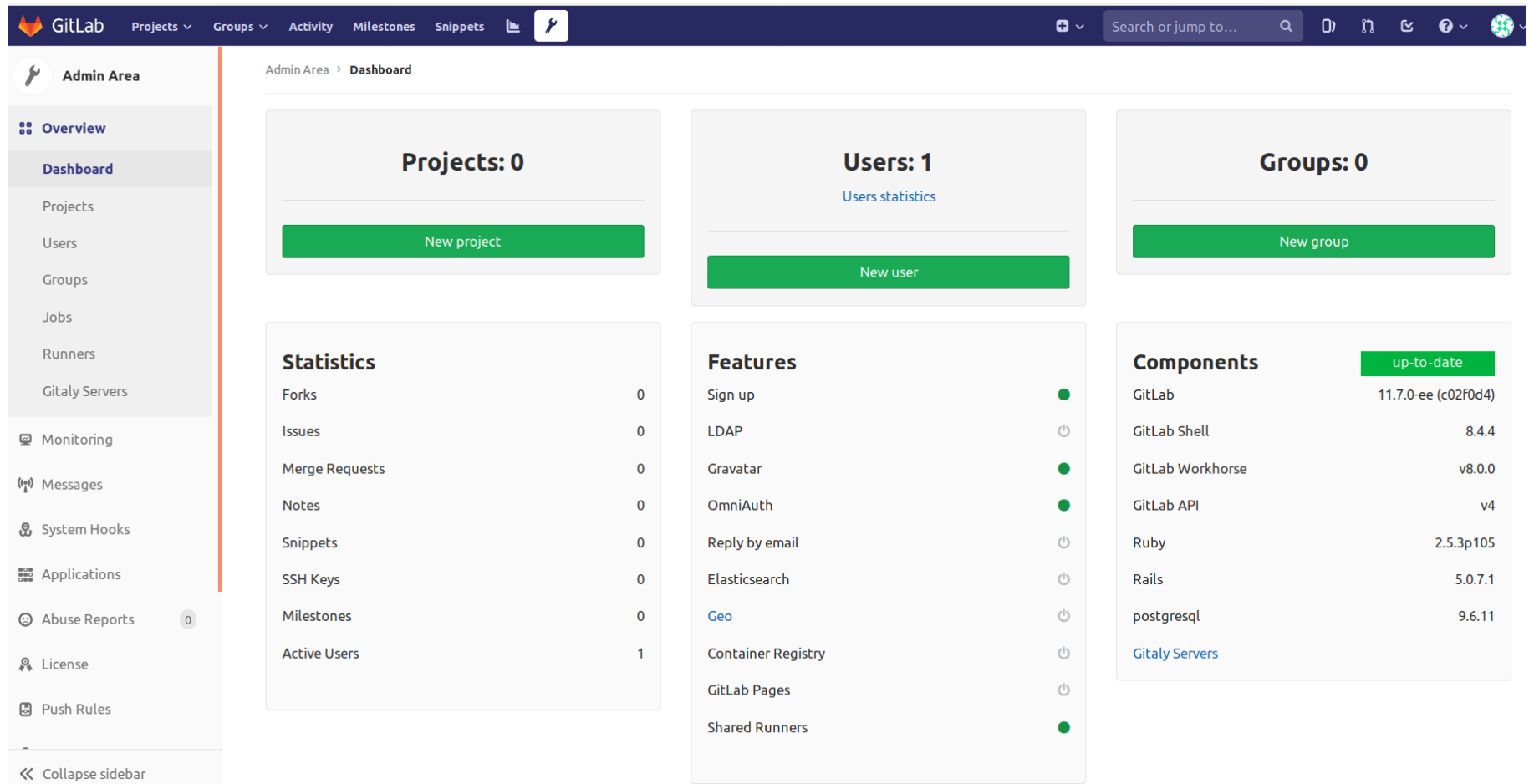
Last updated ▾

**All** Personal

 D Administrator / **Demo Project1**  Maintainer

★ 0  
Updated 4 days ago

# Der GitLab-Administrationsbereich



The screenshot shows the GitLab Admin Area Dashboard. The top navigation bar includes links for Projects, Groups, Activity, Milestones, Snippets, and a search bar. The left sidebar contains the Admin Area menu with options like Overview, Dashboard, Projects, Users, Groups, Jobs, Runners, GitLab Servers, Monitoring, Messages, System Hooks, Applications, Abuse Reports, License, Push Rules, and a Collapse sidebar button. The main content area displays the Admin Area Dashboard with three summary cards: Projects: 0 (New project button), Users: 1 (Users statistics link, New user button), and Groups: 0 (New group button). Below these are three sections: Statistics (Forks: 0, Issues: 0, Merge Requests: 0, Notes: 0, Snippets: 0, SSH Keys: 0, Milestones: 0, Active Users: 1), Features (Sign up, LDAP, Gravatar, OmniAuth, Reply by email, Elasticsearch, Geo, Container Registry, GitLab Pages, Shared Runners), and Components (GitLab 11.7.0-ee (c02f0d4), GitLab Shell 8.4.4, GitLab Workhorse v8.0.0, GitLab API v4, Ruby 2.5.3p105, Rails 5.0.7.1, postgresql 9.6.11, GitLab Servers link). A green 'up-to-date' badge is next to the Components section header.

**GitLab** Projects Groups Activity Milestones Snippets

Admin Area

**Admin Area**

- Overview
- Dashboard**
- Projects
- Users
- Groups
- Jobs
- Runners
- GitLab Servers

Monitoring

Messages

System Hooks

Applications

Abuse Reports 0

License

Push Rules

Collapse sidebar

Admin Area > Dashboard

**Projects: 0**

New project

**Users: 1**

Users statistics

New user

**Groups: 0**

New group

**Statistics**

Forks	0
Issues	0
Merge Requests	0
Notes	0
Snippets	0
SSH Keys	0
Milestones	0
Active Users	1

**Features**

Sign up	●
LDAP	⏻
Gravatar	●
OmniAuth	●
Reply by email	⏻
Elasticsearch	⏻
Geo	⏻
Container Registry	⏻
GitLab Pages	⏻
Shared Runners	●

**Components** up-to-date

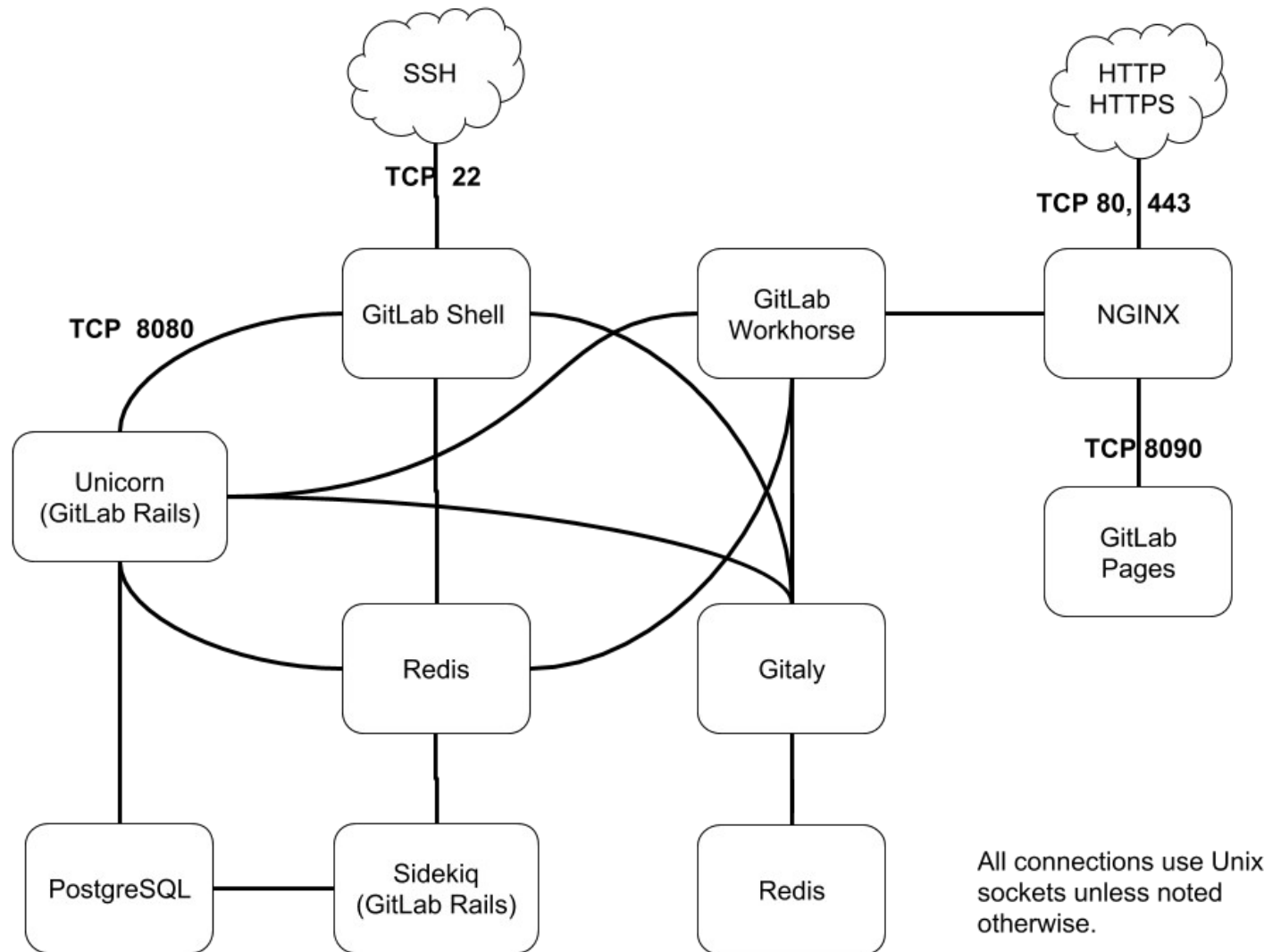
GitLab	11.7.0-ee (c02f0d4)
GitLab Shell	8.4.4
GitLab Workhorse	v8.0.0
GitLab API	v4
Ruby	2.5.3p105
Rails	5.0.7.1
postgresql	9.6.11
GitLab Servers	

1.4

## **ARCHITEKTUR**

- GitLab ist als Ruby-Anwendung realisiert
  - Damit wird auch die Konfiguration über Ruby-Dateien durchgeführt
- Die Ablage der einzelnen Git-Repositories erfolgt im Dateisystem
  - Eine Sicherung bzw. Spiegelung der Dateien ist mit GitLab-Mitteln vorgesehen
- Zusätzlich wird eine relationale Datenbank benötigt
  - MySQL oder PostgreSQL sind hier empfohlen
- Das Web Frontend wird durch einen separaten Web Server geliefert
  - nginx ist hier der Standard
- Als Cache wird eine Redis-Datenbank benutzt
  - Ein Key-Value-Store aus dem Produkt-Katalog der NoSQL-Umgebung

# GitLab Application Architecture



1.5

## **WERKZEUGE**

- Zentrales Skript zur Kontrolle des GitLab-Servers und aller seiner Bestandteile
- Zugriff über die Linux-Konsole des GitLab-Hosts
  - Direkt
  - SSH
- `gitlab-ctl -help`
  - Ausgabe der verfügbaren Kommandos
- `gitlab-ctl reconfigure`
  - Rekonfiguration der Komponenten bei Änderungen der Konfigurationsdateien

- Remote Zugriff auf Git-Befehle
- Die internen Anwendungen benutzen alle gitaly
  - gitaly wird bis auf wenige Ausnahmefälle nicht direkt benutzt



- Die Web-Oberfläche des GitLab-Servers
  - Realisiert mit Ruby on Rails
- Zugriff erfolgt von Außen gekapselt über nginx

- Direkter Aufruf von GitLab-Kommandos
  - Steuerung und Konfiguration des Servers

- Ruby-Bibliothek zur Parallelisierung von Prozessen und Jobs
- Damit bildet Sidekiq das Rückgrat sämtlicher Prozesse in GitLab

2

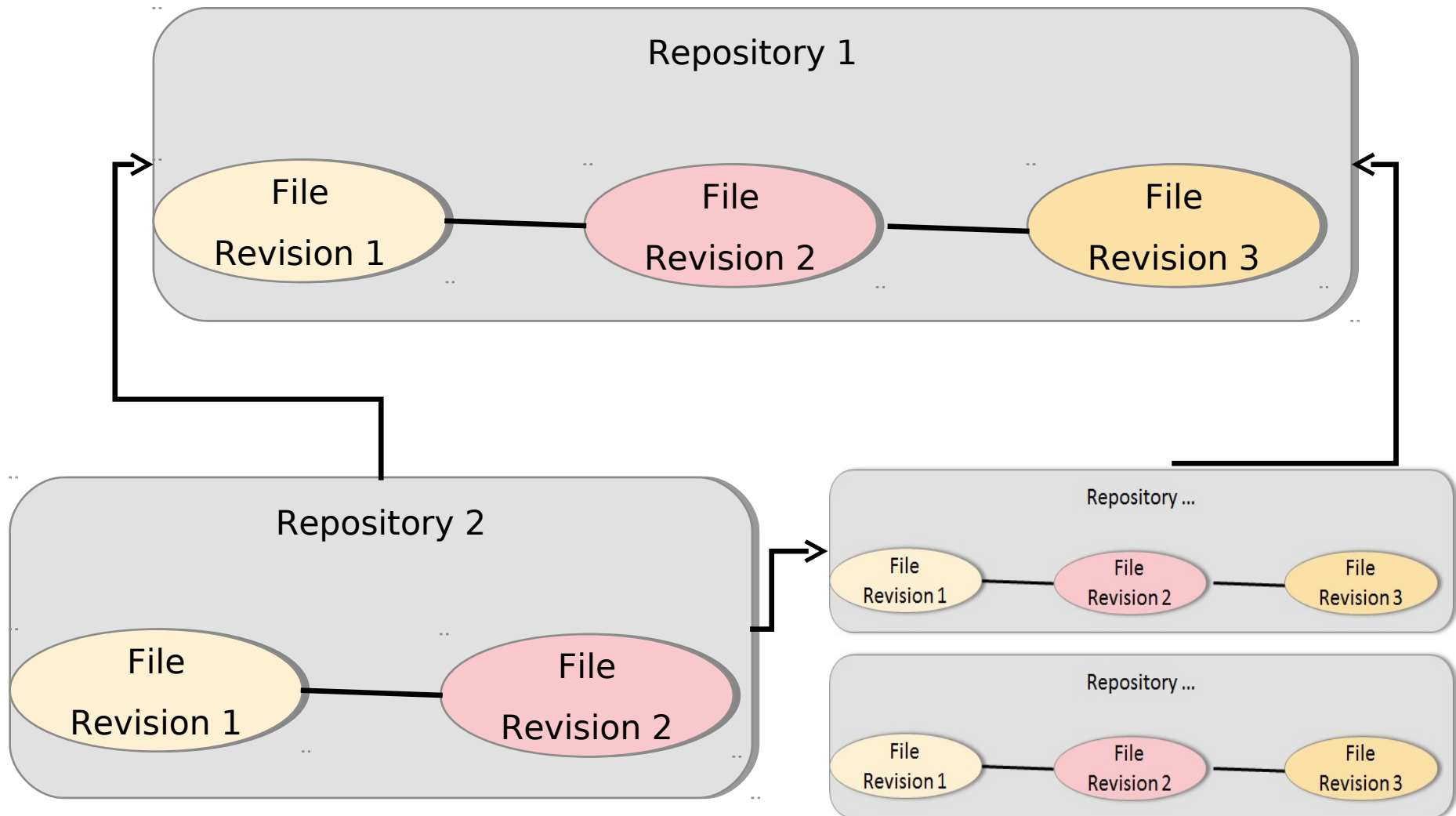
# **GIT**

2.1

## **KURZEINFÜHRUNG**

- Dateiablage in gleichberechtigten Repositories
- Jedes Repository bietet die kompletten Funktionen eines Source Code Management Systems an
  - commit
  - checkout
  - Historie
  - Diffs
- Ständen können mit verschiedene Repositories synchronisiert werden
  - clone
  - pull
  - push

# Zusammenspiel von Repositories



- Branches sind Alias-Namen auf Revisionsstände
- Aktive Branches werden bei jedem Commit weiterbewegt
  - Und bilden damit eine fortlaufende Projektentwicklung ab
- Ein Software-Projekt wird aus verschiedenen Branches aufgebaut sein
  - Best Practices werden als „Git Flows“ bezeichnet
- Branches sind häufig korreliert mit dem Issue-Management
  - Sie bilden ab, welche Aktivitäten im Projekt durchgeführt werden
- Namen damit
  - Issues
    - z.B. Ticket-Nummern
  - Verben
    - Sprechende Namen für Aufgaben



- Tags sind ebenfalls Alias-Namen auf Revisionsstände
- Tags bewegen sind jedoch fix und bewegen sich nicht mehr
  - Und bilden damit eine festen Stand
- Tags sind häufig korreliert mit dem Release-Management
  - Sie bilden ab, welche fertigen Stände im Projekt erreicht sind
- Namen damit
  - Releases
    - z.B. Versionsnummern
  - Substantive
    - Sprechende Namen für Stände

2.2

## **GIT SERVER**

- Git alleine stellt für die Team-Arbeit nur wenige Hilfsmittel zur Verfügung
- Zentrale Server-Lösungen sind sinnvoll
  - Atlassian BitBucket
  - GitHub
  - GitLab
- Funktionen
  - Zentraler Zugriff über Netzwerk
  - Web Frontend
  - Authentifizierung und Autorisierung
    - Insbesondere Schützen von Branches vor unzulässigen Änderungen

2.3

## **ANBINDEN VON GIT-CLIENTS**

- Die Kommandozeilen-Befehle sind für ein technisches Verständnis der Abläufe sehr interessant
- In der Praxis werden jedoch häufig Git-Clients mit grafischer Unterstützung verwendet
- Standalone-Programme
  - Tortoise
  - SourceTree
- Integration in Entwickler-Werkzeuge
  - Eclipse
  - XCode
  - Visual Studio
  - Atom
  - ...

- http-basiert mit Authentifizierung über Credentials
- SSH mit Schlüssel/Zertifikat

3

# PROJEKTORGANISATION

3.1

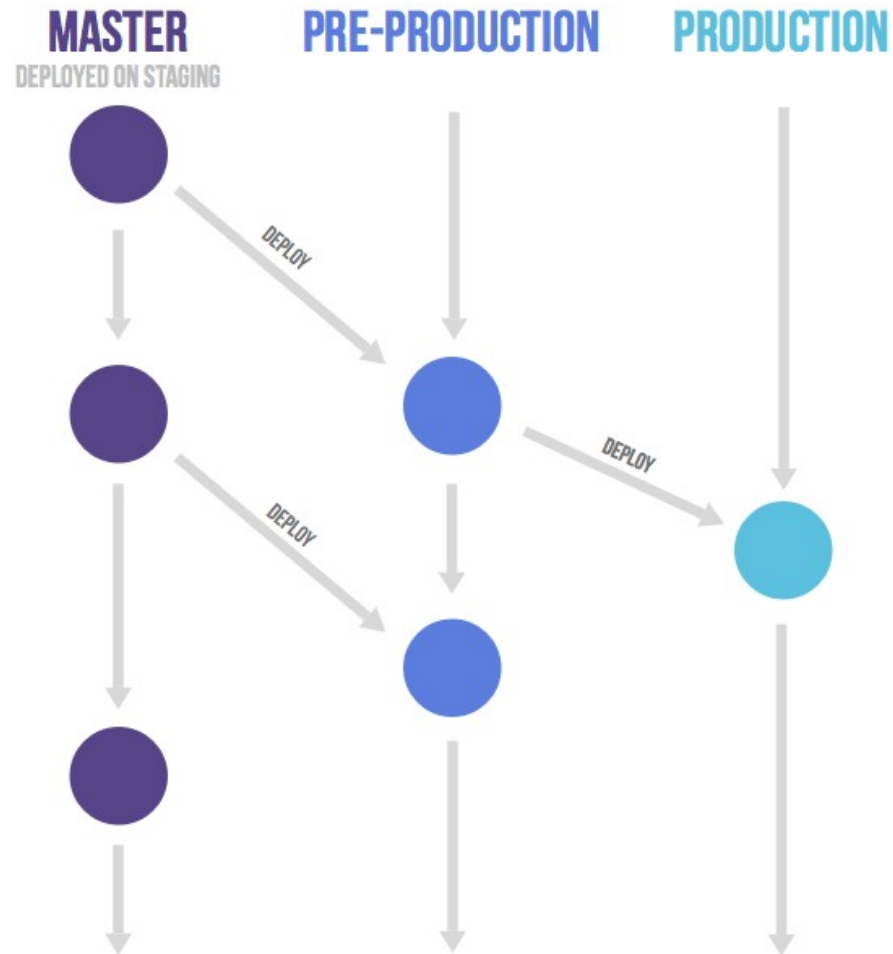
## **GITLAB FLOW**



- Git selber ist nur ein Revisionsverwaltungssystem
  - und stellt damit Basis-Befehle zur Verfügung
- Wie genau Git am Besten benutzt werden kann wird durch einen (Work) Flow beschrieben
  - Im Wesentlichen eine Vorgehensweise, die aus den Erfahrungen vieler Projekte gewonnen wurden
  - Damit eine "Best Practice"
- Beispiele
  - Git Flow
    - Vorgestellt und dokumentiert von Atlassian
  - GitHub Flow
  - GitLab Flow

- Basiert auf Merge Requests
  - Es existieren somit geschützte Branches, die nur von speziellen Rollen benutzt werden dürfen
- Die geschützten Branches definieren ein Environment bestehend aus verschiedenen Stages
  - Test und QS
  - Preproduction
  - Production
  - ...
- Auch Releases können damit verwaltet werden

# Beispiel für den GitLab Flow



3.2






## **ISSUE MANAGEMENT**

- GitLab enthält ein integriertes Issue Management
- Auch eine Anbindung an externe Systeme ist möglich
  - Jira
  - Bugzilla
  - ...
- Damit können Aufgaben erfasst und verwaltet werden

3.3

## **ANALYTICS**

- Die Zustandswechsel eines Issues sowie die Aktionen auf den Environment-Banches werden erfasst
  - Neue Issues
  - Commits
  - Deployments
- Damit ist eine Analyse des Fortschritts eines Projekts möglich

Pipeline Health			
15 New Issues	24 Commits	5 Deploys	Last 30 days ▾
Stage ?	Median ?	Related Issues ?	Total Time ?
Issue	7 days	Time before an issue gets scheduled	
Plan	8 days	 <b>Cycle Analytics: Consequatur dolor dolore voluptat...</b> #15 · Opened 34 minutes ago by Dr. Catherine Schultz	7 days 12 hr
Code	8 days	 <b>Cycle Analytics: Ut omnis suscipit optio animi nesciu...</b> #14 · Opened 34 minutes ago by Administrator	7 days
Test	about 5 hours	 <b>Cycle Analytics: Quaerat eos eius quo dolor ut et mol...</b> #13 · Opened 34 minutes ago by Administrator	6 days 12 hr
Review	16 days	 <b>Cycle Analytics: Quia assumenda temporibus ut omn...</b> #12 · Opened 34 minutes ago by Jamal Hirthe	6 days
Staging	7 days	 <b>Cycle Analytics: Veniam ex mollitia saepe nobis qui a...</b> #11 · Opened 34 minutes ago by Curtis Franecki	5 days 12 hr
Production	about 1 month		



# 4

## **GITLAB KONFIGURATION**

## 4.1

# OPTIONEN DER KONFIGURATION

- Eine zentrale Datei für alle Einstellungen
  - Im Original sind exemplarische Einstellungen kommentiert vorhanden
  - Ebenso Verweise auf die Seiten der GitLab-Dokumentation
- gitlab.rb in /etc/gitlab
- Editieren
  - mit Text-Editor
  - über gitlab-ctr
- Übernehmen der Änderungen durch Rekonfiguration des Servers

- Einstieg über die Admin-Oberfläche
- Darin dann das settings-Menü
- Vorsicht: Bei weitem nicht alle Einstellungen der Konfigurationsdatei können über die Web-Oberfläche durchgeführt werden

- <https://docs.gitlab.com/omnibus/README.html#configuring>

4.2

## **BEREICHE**

# Das allgemeine settings-Menü



- <http://localhost:9080/admin/appearance>
- <https://docs.gitlab.com/ee/user/profile/preferences.html>



5

## **BETRIEBLICHE ASPEKTE**

5.1

## **MONITORING**

- Kategorien
  - CPU und Speicher sowie I/O
    - natürlich auch über Betriebssystem-Überwachung möglich
  - Hintergrundprozesse
  - Server-Kommunikation
- Zugriff über
  - Web Anwendung
  - Log-Dateien
  - REST-API
    - http-Endpoints
    - Zugriff nur durch Verwendung eines Access Tokens

## Monitoring

System Info

Background Jobs

## Logs

Health Check

Requests Profiles

JSON	Rohdaten	Kopfzeilen
Speichern	Kopieren	Alle einklappen
Alle ausklappen		
▼ db_check:		
status:		"ok"
▼ redis_check:		
status:		"ok"
▼ cache_check:		
status:		"ok"
▼ queues_check:		
status:		"ok"
▼ shared_state_check:		
status:		"ok"
▼ gitaly_check:		
status:		"ok"

5.2

## UPGRADING/DOWNGRADING

- <https://docs.gitlab.com/omnibus/update/>

5.3

## **GITLAB API**



- RESTful API zum Zugriff auf praktisch alle relevanten Server-Informationen
- <https://docs.gitlab.com/ee/api/>

5.4

## **TROUBLESHOOTING**

- <https://docs.gitlab.com/ee/administration/troubleshooting/debug.html>
- [https://docs.gitlab.com/ee/topics/git/troubleshooting\\_git.html](https://docs.gitlab.com/ee/topics/git/troubleshooting_git.html)
- <https://forum.gitlab.com/t/troubleshooting-guide-wiki/31>

6

# CONTINUOUS INTEGRATION

6.1

## **BEGRIFFE UND ARBEITSWEISE**

- Ein Build-Prozess erzeugt ein Artefakt
  - Ein Artefakt wird in eine Laufzeitumgebung installiert und wird darin als Anwendung ausgeführt
  - Beispiel
    - Java-Source werden vom Compiler in Bytecode verwandelt und in ein JAR-Archiv gepackt
- Dieser Build-Prozess wird bei der Continuous Integration automatisch ausgeführt, sobald im Versionsverwaltungssystem neue Sourcen eintreffen
  - Damit werden neue Artefakt-Versionen erzeugt
- Bei der Continuous Delivery werden diese Artefakte zusätzlich automatisch in die Laufzeitumgebung installiert

- Das Konzept der Runner ermöglicht das Ausführen eines Build-Skriptes aus GitLab heraus
- Der GitLab-Runner kann auf einer externen Maschine installiert werden
  - Registrierung bei GitLab über ein einem Administrator bekanntem Token
- Projekte können einen Runner zugeordnet bekommen
  - Dieser führt das dem Projekt beigefügte Build-Skript an
    - `.gitlab-ci.yml`
  - Näheres unter
    - <https://docs.gitlab.com/ee/ci/pipelines.html>
    - <http://<YourGitLabHost>/help/ci/yaml/README.md>

# Beispiel: Ein Maven-Projekt

```
image: maven:latest
variables:
  MAVEN_OPTS: "-Dmaven.repo.local=.m2/repository"
cache:
  paths:
    - .m2/repository/
    - target/
build:
  stage: build
  script:
    - mvn compile
test:
  stage: test
  script:
    - mvn test
deploy:
  stage: deploy
  script:
    - mvn deploy
only:
  - master
```



- Das Build-Skript definiert als zentrales Element ein Docker-Image
  - Dieses stellt die Build-Umgebung bereit
  - Damit ist eine maximale Flexibilität erreicht, da durch die Images auch unterschiedliche Umgebungen in unterschiedlichen Versionen bereitgestellt werden können
- Dieses Basis-Image wird auf der Maschine des GitLab-Runners installiert
  - Dazu wird von Docker ein lokales Repository als Cache benutzt
- Das Build-Skript wird dann in einem für diesen Build definierten Container ausgeführt

7

## **WEITERE THEMEN**

7.1

## WEB HOOKS

- Mit GitLab Hooks wird der Repository-Server mit anderen Produkten verbunden
  - Haben nichts mit Git-Hooks zu tun!
- Bei bestimmten Aktionen werden Http-Requests abgesetzt
  - Ziel-URL ist definierbar
  - Die übermittelten Daten werden von GitLab festgelegt und sind nicht veränderbar

## Web hooks

Web hooks can be used for binding events when something is happening within the project.

**URL**

`http://example.com/trigger-ci.json`

**Trigger**



**Push events**

This url will be triggered by a push to the repository



**Tag push events**

This url will be triggered when a new tag is pushed to the repository



**Comments**

This url will be triggered when someone adds a comment



**Issues events**

This url will be triggered when an issue is created



**Merge Request events**

This url will be triggered when a merge request is created

7.2

## **CONTAINER REGISTRY**


- Eine Registry für Docker-Images
  - Dies sind Artefakte
- [https://docs.gitlab.com/ee/administration/container\\_registry.html](https://docs.gitlab.com/ee/administration/container_registry.html)

7.3

## **SNIPPETS**



- Mit Snippets können Informationen zwischen Benutzern geteilt werden
  - Code-Fragmente
  - Texte
  - Dokumentationen und Bilder
- Rollenkonzept zum Zugriff ist vorhanden

Snippet \$13017 authored a year ago by  Marcia Ramos

[Edit](#)[Delete](#)[New snippet](#)

## Quick index.html page

Edited 11 months ago

 **index.html** 200 Bytes 

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>Hello World</title>
6 </head>
7 <body style="background-color: black; color: white">
8     <h1>HELLO WORLD!</h1>
9 </body>
10 </html>
```

 0  0 



**Write** Preview

**B** *I* “ ” </> ☰ ☷ ☑ ✕

Write a comment or drag your files here...

Markdown is supported

 Attach a file

[Comment](#) ▼