Introduction to GitLab

Basics and Continuous Integration

Christian Stankowic

www.stankowic-development.net

Free and Open Source software Conference

20.08.2016

whoami

Christian Stankowic

Messer Information Services GmbH

Linux & vSphere administrator

Blogger & book author

AGENDA

Agenda

Motivation

Installation

Crash course

Continuous Integration

MOTIVATION

Why Git?

simple, distributed version control system

suitable even for non-developers

high security against code corruption by hashing

secure data transfers using SSH



Hello Git world!

```
$ git clone http://server/repo.git
$ cd repo ; vi myfile.c
$ git add *
$ git commit -m 'added feature'
$ git push
```

Listing 1: Clone repo and upload changes



Git variety

Standalone, local data

Web interface using **GitWeb**

Hosted (extern and on-premise):

GitHub

BitBucket

GitLab

etc...



What is GitLab?

web-based interface for version control with git

includes additional management and documentation functionality

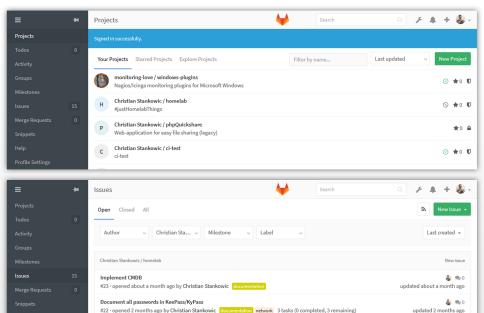
free and paid models available

monthly release cycle



Features

Intuitive responsive web interface file manager integrated 'ticket tool' and wiki automatic code compilation (CI) and much more...



Implement web-based IP management software

#20 · opened 2 months ago by Christian Stankowic documentation enhancement network

updated 2 months ago

Edition comparison

Feature	CE	EE
file manager, issues, wiki	\checkmark	\checkmark
Online code changes	\checkmark	\checkmark
GitHub import	\checkmark	\checkmark
LDAP/AD authentication	\checkmark	\checkmark
CI and Docker support	\checkmark	\checkmark
Support	×	\checkmark
Kerberos authentication	×	\checkmark
Merge Request Approvals	×	\checkmark
Issues/Merge Requests templates	X	\checkmark

Detailed comparison: about.gitlab.com/features



Git clients

git, command line

GitX-dev (macOS)

SourceTree Free (Windows, macOS)

GitG / qGit (Linux)

Git jargon variety

Description	GitLab	GitHub
Merge code	Merge Request	Pull Request
Code snippet	Snippet	Gist
Project	Project	Repository
Team/Group	Group	Organization

Source: about.gitlab.com

INSTALLATION

Installation

GitLab CE and EE can be downloaded as **Omnibus** package (*distro-specific, Chef*)

available for:

Debian 7/8

Raspbian

Ubuntu 12.04 / 14.04 / 16.04

Enterprise Linux (RHEL, CentOS, SL) 6 / 7

see also: about.gitlab.com/downloads

Requirements

at least 2 Cores

minimum 2 GB RAM

more than 500 users: graded by user amount

see also: doc.gitlab.com/install

Preparation

Required services and packages:

SSH server

Curl

Postfix

Ruby 2.1

Import software source, install gitlab-ce package

Configure using: gitlab-ctl reconfigure



CRASH COURSE

Projects, users,...

Source code is located in projects

Users (in groups) have access

Code versions vary between branches (e.g. master, nightly,...)

Changes between branches are transfered using Merge Requests

Users and SSH keys

Users can be part of groups (teams)

Each user profile contains SSH client keys

Code changes are transfered securely using SSH

The first project

Creating a project

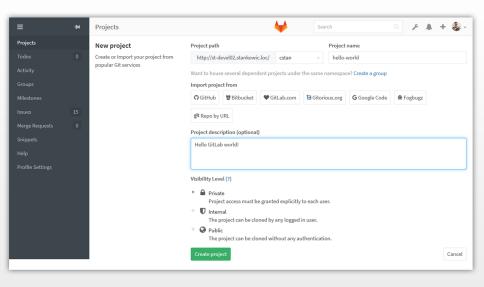
Creating a README, LICENSE

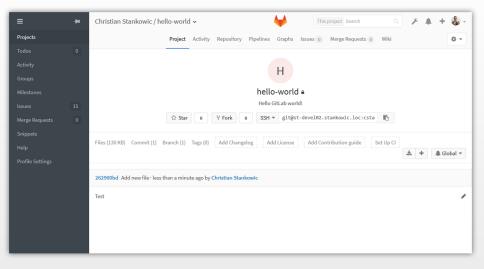
Adding your SSH key

Cloning the project

Changing the code

Uploading changes





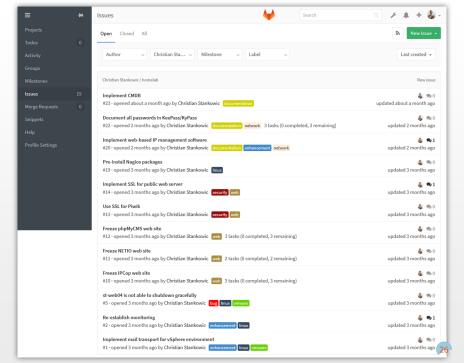
Issues: bug tracking and more

Documenting bugs, ideas, support requests,...

Grouped by tags (e.g. bug, enhancement,...)

Can be mapped to users and milestones







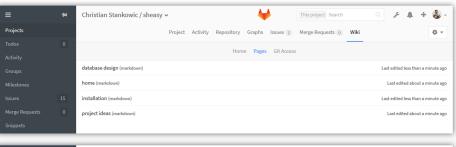
Full-featured wiki per project

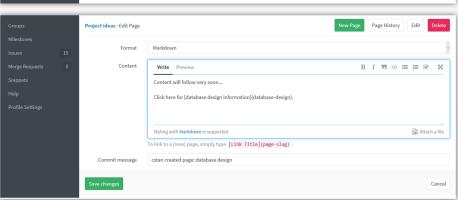
Format using Markdown, RDoc or AsciiDoc

Good way to implement big documentations

Wiki is managed using a dedicated Git repository







CONTINUOUS

Why CI?

Automating development sub-steps, e.g.:

Compiling sub-programs
Compound the product
Testing sub-programs with (*test units*)

Available since version 8.0



Functionality

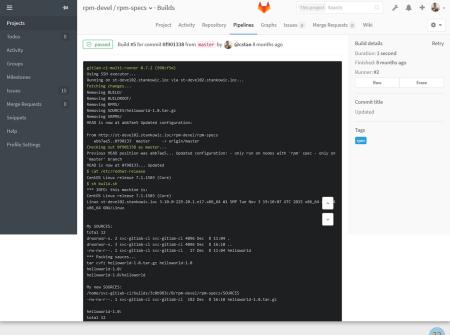
Code changes are applied

GitLab CI selects systems for automation (*Runner*)

Runners retrieve code changes

Pre-defined scripts are executed





GitLab runner

Agent for Linux, Windows, macOS and BSD written in **Go**

Supports Bash, Batch and PowerShell

Also available as Docker service

Encrypted communication between CI and Runner

Builds can be seen within the GUI



CI configuration

Configuration using YAML file within the project (.gitlab-ci.yml)

Defines:

Scripts pre/after/during compilation Branches and Runner (e.g. custon Runner for **master**)

Artefacts



Excursus: Executor types

Executor controls code compilation:

Shell - simple, local host/Windows system

Docker - for complex projects with dependencies (*e.g. databases*)

VirtualBox / **Parallels** - VMs can be cloned during life-time

SSH - 'letzte Wahl', remote commands

see also: gitlab.com/gitlab-org/...



Artefacts

Selection of files after running a build (e.g. binary files)

Can be downloaded from the GUI

Artefacts can be handed to web hooks (content delivery)

Only using **shell** and **Docker** executors!



Example: Python plugins

Icinga plugins for Linux and Windows

automatic packaging using pyInstaller for Windows

Result: EXE file containing Python interpreter + script

.gitlab-ci.yml

```
before_script:

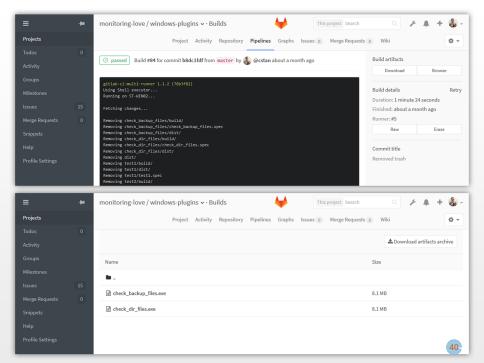
    hostname

  build:
    script:
       - "powershell.exe -File
       build_binaries.ps1"
    artifacts:
       paths:
7
         - dist
```









QUESTIONS?

Links

GitLab website: gitlab.org

Omnibus package downloads

GitLab documentation

GitLab CI documentation

GitLab Runner repository

GitLab CI tutorial on my blog

Thanks for your attention!

http://www.stankowic-development.net







Christian Stankowic

stankowicdevel