

Gitting Started

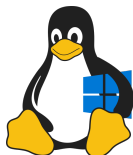
An Introduction to Git

K. Müller¹ J. Verzijden²

¹MasterCLASS
E.T.S.V. Scintilla

²Scintilla Operator Team
E.T.S.V. Scintilla

June 27, 2023



Let's imagine...

Alice



Bob



About us



Kasper Müller

kasperm@scintilla.utwente.nl



Johan Verzijden

johanv@scintilla.utwente.nl

Contents

- 1 Theoretical
 - Why Git?
 - History
 - Concepts of Git
 - Principles of Git
- 2 Installation
- 3 Break
- 4 Practical

Why Git?

- Solves the problems from the intro.
- It's efficient and fast
- It's the most used VCS^a in the world

^aVersion Control System

Version control systems used by responding developers:

Name	2015	2017	2018	2022
Git	69.3%	69.2%	87.2%	93.9%
Subversion	36.9%	9.1%	16.1%	5.2%
TFVC	12.2%	7.3%	10.9%	[ii]
Mercurial	7.9%	1.9%	3.6%	1.1%
CVS	4.2%	[ii]	[ii]	[ii]
Perforce	3.3%	[ii]	[ii]	[ii]
VSS	[ii]	0.6%	[ii]	[ii]
ClearCase	[ii]	0.4%	[ii]	[ii]
Zip file backups	[ii]	2.0%	7.9%	[ii]
Raw network sharing	[ii]	1.7%	7.9%	[ii]
Other	5.8%	3.0%	[ii]	[ii]
None	9.3%	4.8%	4.8%	4.3%

The origin story



- Linus Torvalds^a
- 2005
- VCS for the Linux kernel
- *The stupid content tracker*

^aSource of the photo: [WikiMedia](#)

Concepts of Git

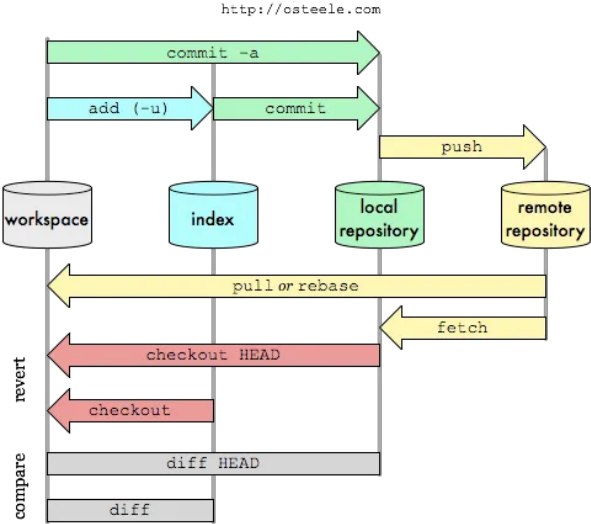
- Repository (repo)
 - A location for storage¹
 - In this case, storage of files
 - Just a folder on your computer, e.g. `/home/johanv/my-repo`
- Commit
 - Description of changes
 - Metadata: author, timestamp, message, parent
 - Hash: Its unique identifier
- Branch
 - A list of commits belonging together
 - A tag points to the head of the branch (the last commit)
- Remote
 - The same repository in a different location
 - Really remote (`https://github.com/instructure/canvas-lms`)
 - Locally remote (`/home/kasperm/some-repo/`)

¹*repository* on Wiktionary

Principles of Git

- Recording small sets of changes, not versions
- Meant for text files, not for binary files
- Meant for source files, not compiled files
- Make a new branch for each feature
- Branch names should be descriptive
 - ✓ *create-home-page*
 - ✗ *branch15*
- Commits should contain small, logical changes
- Commit messages should be descriptive
 - ✓ *Fix styling of logout button*
 - ✗ *fix*
 - ✗ *Meep*
- Commit messages should be in imperative tense
 - ✓ *Add page layout*
 - ✗ *Added page layout*
 - ✗ *Adding page layout*

Git Workflow



Let's take another look (at the situation)

Let's see how Git actually solves the problem in the introduction.

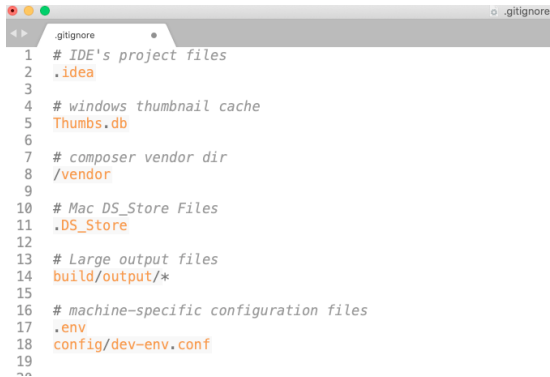


Tracking source files - Gitignore

In git you only want to track source files. Binaries or compiled files don't work well with version changes and can be different on other pcs.

Solution: don't add the changed source files, permanent solution:

`.gitignore`

A screenshot of a code editor window titled ".gitignore". The editor shows a list of files and directories to be ignored, with line numbers 1 through 19 visible. The content is as follows:

```
1 # IDE's project files
2 .idea
3
4 # windows thumbnail cache
5 Thumbs.db
6
7 # composer vendor dir
8 /vendor
9
10 # Mac DS_Store Files
11 .DS_Store
12
13 # Large output files
14 build/output/*
15
16 # machine-specific configuration files
17 .env
18 config/dev-env.conf
19
20
```

Installation - GUI

Some options:

- Pure Git GUI² clients:
 - git-gui & gitk (comes with the Git CLI)
 - Github Desktop (only for Mac & Windows)
 - gitg (only for Linux & Windows)
 - See <https://git-scm.com/downloads/guis> for more
- Most IDEs have a Git GUI built-in

Nice and all, but we will use the CLI³.

²Graphical User Interface

³Command Line Interface

Installation - CLI

- Windows

- Using *winget*:

```
winget install --id Git.Git -e --source winget
```

- Using an installer or portable version:

<https://git-scm.com/downloads/win>

- Linux

- Don't, you already have it (-:
- Through your package manager:

- `apt install git`
- `dnf install git`
- `pacman -S git`

- MacOS

- Using Homebrew:
 - `brew install git`
- Install Xcode, that ships with Git

Installation successful?

```
johanv@my-machine: ~$ git
usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]
  [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
  [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
  [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
  [--super-prefix=<path>] [--config-env=<name>=<envvar>]
  <command> [<args>]
```

These are common Git commands used in various situations:

start a working area (see also: `git help tutorial`)

clone	Clone a repository into a new directory
init	Create an empty Git repository or reinitialize an existing one

work on the current change (see also: `git help everyday`)

add	Add file contents to the index
mv	Move or rename a file, a directory, or a symlink
restore	Restore working tree files
rm	Remove files from the working tree and from the index

examine the history and state (see also: `git help revisions`)

bisect	Use binary search to find the commit that introduced a bug
diff	Show changes between commits, commit and working tree, etc
grep	Print lines matching a pattern
log	Show commit logs

Install Git!

Get out your laptop and install Git on it if you haven't already, we will start using it directly after the break

Practical - Goals

- Show the typical flow of working with git.
- Show you most important commands and their uses in a short time

Configuration

How:

- Using an editor:
`git config --global --edit`
- Using commands:
`git config --global user.name 'Johan Verzijden'`

What:

- Name (*user.name*)
- Email address (*user.email*)
- Default branch name (*init.defaultBranch*)
- Fast-forward (*pull.ff*)

Read `git help config` or `man git config` for all configuration options

Practical - Documentation


Feel free to work on your own or in pairs. If there are many questions on a certain topic we will do a class-wide explanation.



Starting a new project


```
~$ mkdir my-portfolio  
~$ cd my-portfolio  
~/my-portfolio$ git init
```

Make some changes, do your first commits




```
# Create index.html and style.css files
# Add standard HTML layout
~/my-portfolio$ git add index.html style.
css
~/my-portfolio$ git commit -m 'Initial
commit'
# Add heading and welcome text
~/my-portfolio$ git commit -am 'Add
heading and welcome text'
~/my-portfolio$ git log
```

Create a new branch and switch to it



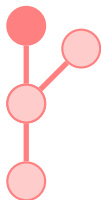
```
~/my-portfolio$ git switch -c add-git-  
project-info  
#### Alternative  
~/my-portfolio$ git branch add-git-  
project-info  
~/my-portfolio$ git checkout add-git-  
project-info
```

Commit and switch back



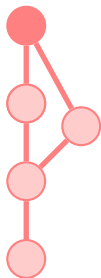
```
# Add info about your latest project
~/my-portfolio$ git commit -am 'Add info
  about latest project'
# Add info about another project
~/my-portfolio$ git commit -am 'Add info
  about another project'
# Oh wait, I don't want this in this
  branch. Let's go back
~/my-portfolio$ git log
~/my-portfolio$ git reset <previous
  commit hash>
~/my-portfolio$ git switch main
```

Commit on main branch



```
# Add contact details
~/my-portfolio$ git commit -am 'Add
    contact details'
```

Merge the two branches



```
~/my-portfolio$ git merge add-git-project  
-info  
# Merge conflict!
```


Resolve the merge conflict

```
# Go into the file and resolve the conflict
~/my-portfolio$ git add index.html
~/my-portfolio$ git merge --continue
```

Change the text colour and commit



```
# Make the text and background colour the  
    same  
# (This is on purpose, will become clear  
    later)  
~/my-portfolio$ git commit -am 'Give the  
    text a great colour'
```

Create repository on Gitlab

Your work

- Projects
- Groups
- Issues
- Merge requests
- To-Do List
- Milestones
- Snippets
- Activity
- Environments Dashboard
- Operations Dashboard
- Security

← Collapse sidebar

Your work > Projects

Welcome to GitLab

Faster releases. Better code. Less pain.

Create a project

Projects are where you store your code, access issues, wiki and other features of GitLab.

Create a group

Groups are the best way to manage projects and members.

Explore public projects

Public projects are an easy way to allow everyone to have read-only access.

Learn more about GitLab

Take a look at the documentation to discover all of GitLab's capabilities.

Create repository on Gitlab

The screenshot shows the GitLab interface for creating a new project. The left sidebar contains navigation options: Your work, Projects, Groups, Issues, Merge requests, To-Do List, Milestones, Snippets, Activity, Environments Dashboard, Operations Dashboard, and Security. The main content area is titled 'Create new project' and features four options:

- Create blank project**: Create a blank project to store your files, plan your work, and collaborate on code, among other things. This option is highlighted with a red box.
- 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.

At the bottom, a note states: "You can also create a project from the command line. [Show command](#)"

Create repository on Gitlab

The screenshot shows the GitLab interface for creating a new project. The left sidebar contains navigation options like 'Projects', 'Groups', 'Issues', 'Merge requests', 'To-Do List', 'Milestones', 'Snippets', 'Activity', 'Environments Dashboard', 'Operations Dashboard', and 'Security'. The main content area is titled 'Create blank project' and includes a sub-header 'Create a blank project to store your files, plan your work, and collaborate on code, among other things.'

Three red circles with numbers 1, 2, and 3 are overlaid on the form to highlight specific fields:

- 1** Points to the 'Project name' field, which contains the text 'My awesome project'. Below it is a note: 'Must start with a lowercase or uppercase letter, digit, emoji, or underscore. Can also contain dots, pluses, dashes, or spaces.'
- 2** Points to the 'Visibility Level' section. It has three radio button options: 'Private' (selected), 'Internal', and 'Public'. The 'Private' option has a sub-note: '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.'
- 3** Points to the 'Project Configuration' section. It contains two checkboxes: 'Initialize repository with a README' (checked) and 'Enable Static Application Security Testing (SAST)' (unchecked). The SAST option has a sub-note: 'Analyze your source code for known security vulnerabilities. Learn more.'

At the bottom of the form are two buttons: 'Create project' (in blue) and 'Cancel'.

Create repository on Gitlab

The screenshot displays the GitLab web interface. At the top, a dark blue navigation bar contains the GitLab logo, a search bar, and various utility icons. On the left, a sidebar lists navigation options for the project, including Project information, Issues, Merge requests, CI/CD, Security and Compliance, Deployments, Packages and registries, Infrastructure, Monitor, Analytics, Wiki, Snippets, and Settings. The main content area shows the project page for 'My awesome project', which has been successfully created. A notification banner at the top of the main area states 'Project 'My awesome project' was successfully created.' Below this, the project name 'My awesome project' is displayed with its ID '9405'. The 'Invite your team' section includes a button to 'Invite members'. The 'The repository for this project is empty' section provides options to start the project, such as cloning the repository or uploading files. A 'Command line instructions' section provides a 'Git global setup' and a 'Create a new repository' script. The 'Push an existing folder' section is also visible at the bottom.

Project 'My awesome project' was successfully created.

My awesome project

Project ID: 9405

Invite your team

Add members to this project and start collaborating with your team.

[Invite members](#)

The repository for this project is empty

You can get started by cloning the repository or start adding files to it with one of the following options.

[Clone](#) [Upload File](#) [New file](#) [Add README](#) [Add LICENSE](#) [Add CHANGELOG](#) [Add CONTRIBUTING](#) [Add Wiki](#) [Configure Integrations](#)

Command line instructions

You can also upload existing files from your computer using the instructions below.

Git global setup

```
git config --global user.name "John Doe"
git config --global user.email "john@doe@example.com"
```

Create a new repository

```
git clone https://gitlab.utwente.nl/<number>/my-awesome-project.git
cd my-awesome-project
git switch -c main
touch README.md
git add README.md
git commit -m "add README"
git push -u origin main
```

Push an existing folder

Set up remote tracking locally

```
~/my-portfolio$ git remote add origin https  
://gitlab.utwente.nl/<s-number>/<project-  
name>.git
```

Push to remote

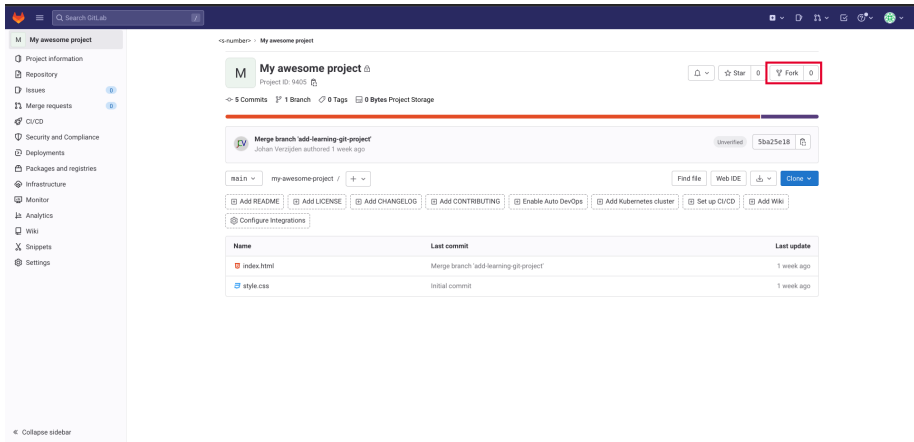
```
~/my-portfolio$ git push -u origin main
```


Look at repository on Github/Gitlab

The screenshot shows the GitLab interface for a repository named "My awesome project". The left sidebar contains navigation options: Project information, Repository, Issues, Merge requests, CI/CD, Security and Compliance, Deployments, Packages and registries, Infrastructure, Monitor, Analytics, Wiki, Snippets, and Settings. The main content area displays the repository details, including the project name, ID (1405), and statistics (5 Commits, 1 Branch, 0 Tags, 0 Bytes Project Storage). A recent commit is highlighted: "Merge branch 'add-learning-git-project'" by Johan Verzijden, authored 1 week ago, with commit hash 5ba25e18. Below this, there are buttons for "Add README", "Add LICENSE", "Add CHANGELOG", "Add CONTRIBUTING", "Enable Auto DevOps", "Add Kubernetes cluster", "Set up CI/CD", and "Add Wiki". A table lists the files in the repository:

Name	Last commit	Last update
index.html	Merge branch 'add-learning-git-project'	1 week ago
style.css	Initial commit	1 week ago

Fork from neighbour



The screenshot shows the GitLab web interface for a project named "My awesome project". The interface includes a sidebar on the left with navigation options like "Project information", "Repository", "Issues", "Merge requests", "CI/CD", "Security and Compliance", "Deployments", "Packages and registries", "Infrastructure", "Monitor", "Analytics", "Wiki", "Snippets", and "Settings". The main content area displays the project name, ID (1405), and statistics (5 Commits, 1 Branch, 0 Tags, 0 Bytes Project Storage). A "Fork" button is highlighted with a red box. Below the project name, there is a section for a recent merge request titled "Merge branch 'add-learning-git-project'" by Johan Verzijden, authored 1 week ago. The merge request details show the source branch "main" and the target branch "my-awesome-project". A table of files is visible, listing "index.html" and "style.css" with their last commit and update dates.

My awesome project

Project ID: 1405

5 Commits 1 Branch 0 Tags 0 Bytes Project Storage

Merge branch 'add-learning-git-project'

Johan Verzijden authored 1 week ago

main my-awesome-project

Find file Web IDE Clone

Add README Add LICENSE Add CHANGELOG Add CONTRIBUTING Enable Auto DevOps Add Kubernetes cluster Set up CI/CD Add Wiki

Configure Integrations

Name	Last commit	Last update
index.html	Merge branch 'add-learning-git-project'	1 week ago
style.css	Initial commit	1 week ago

Fork from neighbour

My awesome project

My awesome project

Fork project

A fork is a copy of a project. Forking a repository allows you to make changes without affecting the original project.

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.utwente.nl/ **Select a namespace**

Project slug

my-awesome-project

Want to organize several dependent projects under the same namespace? [Create a group](#)

Project description (optional)

Visibility level

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

Internal
The project can be accessed by any logged in user.

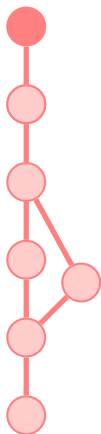
Public
The project can be accessed without any authentication.

Fork project **Cancel**

Clone fork to local


```
~$ git clone https://gitlab.utwente.nl/<number>/<forked-project-name>  
~$ cd <forked-project-name>
```

Revert last commit to fix the text colour



```
~/<forked-project-name>$ git revert <  
commit hash>
```

Do a different commit, push to remote again



```
# Change text to a different colour
~/<forked-project-name>$ git add .
~/<forked-project-name>$ git commit -m '
    Change the text colour to something
    amazing'
~/<forked-project-name>$ git push
```

Open merge request

The screenshot shows a GitLab merge request page. The left sidebar contains navigation options: Project information, Repository, Issues, Merge requests, CI/CD, Security and Compliance, Deployments, Packages and registries, Infrastructure, Monitor, Analytics, Wiki, Snippets, and Settings. The main content area displays the merge request details for 'My awesome project fork' (Project ID: 9407). It shows a commit titled 'Change the text colour to something amazing' by Johan Verzijden, authored 1 minute ago. The merge request is currently 'Unverified' with a SHA of 40fccf90. The source branch is 'main' in the 'my-awesome-project-fork' repository. A 'Create merge request' button is highlighted with a red box. Below the commit information, there are checkboxes for 'Auto DevOps enabled', 'Add README', 'Add LICENSE', 'Add CHANGELOG', 'Add CONTRIBUTING', 'Add Kubernetes cluster', and 'Add Wiki'. A table lists the files included in the merge request:

Name	Last commit	Last update
index.html	Merge branch 'add-learning-gli-project'	1 week ago
style.css	Change the text colour to something amazing	1 minute ago

Open merge request

My awesome project fork > Merge requests > New

New merge request

From `/my-awesome-project-fork:main` into `/my-awesome-project:main` [Change branches](#)

Title (required)

Change the text colour to something amazing

Mark as draft
Drafts cannot be merged until marked ready.

Description

Write Preview

Describe the goal of the changes and what reviewers should be aware of.

Supports [Markdown](#). For quick actions, type `/`.

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

Assignees

Unassigned [Assign to me](#)

Reviewers

Unassigned

Approvals are optional.
[Approval rules](#)

Milestone

Select milestone

Labels

◀ Collapse sidebar

Open merge request

The screenshot shows the GitLab web interface for a project named "My awesome project fork". The left sidebar contains navigation links for Project information, Repository, Issues, Merge requests, CI/CD, Security and Compliance, Deployments, Packages and registries, Infrastructure, Monitor, Analytics, Wiki, Snippets, and Settings. The main content area is titled "Creating merge request" and includes sections for "Approvals are optional", "Milestone" (with a "Select milestone" dropdown), "Labels" (with a "Labels" dropdown), and "Merge request dependencies" (with a text input field for URLs and a note that references should be in the form of path/to/project/merge_request_id). Below this is the "Merge options" section with a checkbox for "Squash commits when merge request is accepted". The "Contribution" section has a checkbox for "Allow commits from members who can merge to the target branch" and a note "Not available for protected branches". At the bottom of the form, the "Create merge request" button is highlighted with a red box, next to a "Cancel" button. Below the form, there is a "Commits" section showing "20 Jun, 2023 1 commit" and a commit entry: "Change the text colour to something amazing" by Johan Verzijden, authored 6 days ago, with a commit hash of 48fccf88.

Review and accept merge request

The screenshot shows a GitLab Merge Request interface. The title is "Change the text colour to something amazing". The status is "Ready to merge!". The "Changes" tab is selected and highlighted with a red box. The "Merge" button is also highlighted with a red box. The "Activity" section is visible at the bottom, with a text area for comments and a "Close merge request" button.

My awesome project

Change the text colour to something amazing

requested to merge /my-awesome-project into main 6 days ago

Overview Commits Pipelines **Changes**

Approval is optional

Ready to merge!

Squash commits Edit commit message

The source branch is 2 commits behind the target branch. 1 commit and 1 merge commit will be added to main.

Activity

Write Preview

Write a comment or drag your files here...

Supports Markdown. For quick actions, type /

Comment Close merge request

Mark as done

0 Assignees

0 Reviewers

Labels

Milestone

Time tracking

Lock merge request

Notifications

1 Participant

Reference: s2235854/my-awesom...
Source branch: main

Look at Blame

```
~/my-repo $ git blame index.html
```

Advanced topics

- Stash
- Rebase
- Cherry-pick

Common Git Commands - Cheat sheet

git ...

- `init`
- `add [<filenames>]`
- `commit [-m '<message>']`
- `switch [-c] <branch name>`
- `log [--graph --oneline]`
- `push`
- `pull`
- `fetch`
- `clone <path/URL>`
- `blame <file> [--color-by-age]`
- `revert <commit>`
- `reset [--hard] <commit>`

Sources for future reference

- `git help <command>`
- `man gittutorial`
- gitimmersion.com

Think of a question later on? Feel free to reach out to us!

MasterCLASS

masterclass@scintilla.utwente.nl

Kasper Müller

kasperm@scintilla.utwente.nl

Johan Verzijden

johanv@scintilla.utwente.nl