git

*

Presentation by Moritz Witt

jan.witt@stadium.uni-hamburg.de

* https://git-scm.com/

Structure

- What is Version Control?
- Git basics
- Branching
- Remote Repository
- GitHub
- Workflows
- Switch to Git

Version Control System (VCS)

- Record changes of a file/set of files
- Possible to nearly any type of file (code, images, layouts,…)

- Local VCS
  - Copy of files in another directory
  - Error-prone; mix up directories

Centralized VCS

- One server that contains all file versions
- Enables Collaboration
- Everyone knows what the collaborator are doing
- Admin has access control
- No access if the server is down
- If central database is corrupted, everything is lost

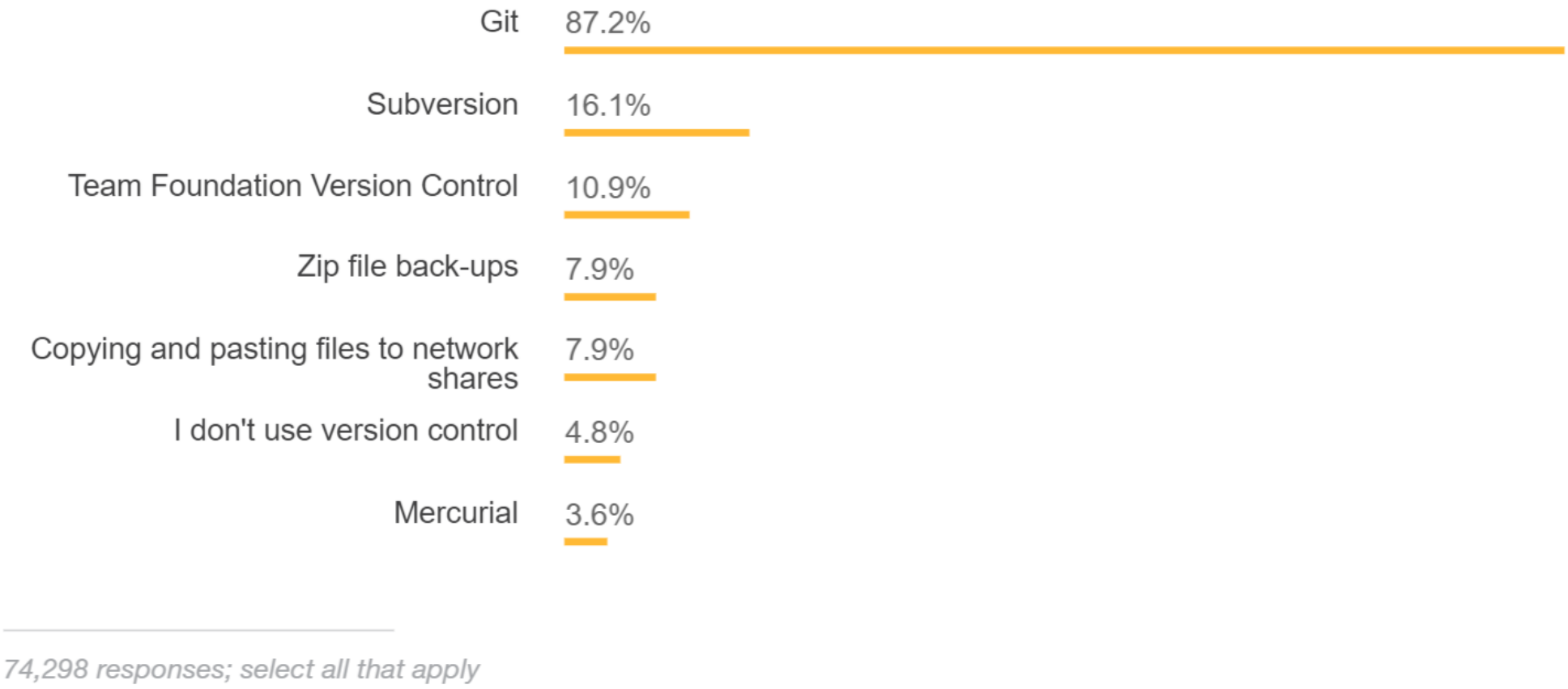
- Concurrent Versions System (CVS), Subversion (SVN) and Team Foundation Version Control (TFVC)

Distributed VCS

- Collaborator clones full repository onto local computer
- Deals with several remote repositories
- Simultaneous work of many people on the same project
- Offline work possible
- Mostly delta-based version control

- Git, Mercurial, Bazaar

VCS used in 2018



*

* https://insights.stackoverflow.com/survey/2018

Git

- From Linux development community
- Goals:
  - Speed
  - Simple design
  - Strong support for non-linear development (branches)
  - Fully distributed
  - Able to handle large projects

- Developed in 2005

Properties

- Entire repository as well as history gets saved on local PC
  - No network latency overheat
  - Offline work possible
- Stores as series of snapshots of current state
  - Every further state easy to revert
- Every commit gets checksummed and stored by referring to it
  - No undetected data loss

Speed Comparison

Operation		Git	SVN	
Commit Files (A)	Add, commit and push 113 modified files (2164+, 2259-)	0.64	2.60	4x
Commit Images (B)	Add, commit and push a thousand 1 kB images	1.53	24.70	16x
Diff Current	Diff 187 changed files (1664+, 4859-) against last commit	0.25	1.09	4x
Diff Recent	Diff against 4 commits back (269 changed/3609+,6898-)	0.25	3.99	16x
Diff Tags	Diff two tags against each other (v1.9.1.0/v1.9.3.0)	1.17	83.57	71x
Log (50)	Log of the last 50 commits (19 kB of output)	0.01	0.38	31x
Log (All)	Log of all commits (26,056 commits – 9.4 MB of output)	0.52	169.20	325x
Log (File)	Log of the history of a single file (array.c – 483 revs)	0.60	82.84	138x
Update	Pull of Commit A scenario (113 files changed, 2164+, 2259-)	0.90	2.82	3x
Blame	Line annotation of a single file (array.c)	1.91	3.04	1x

*

Operation		Git*	Git	SVN
Clone	Clone and shallow clone(*) in Git vs checkout in SVN	21.0	107.5	14.0
Size (MB)	Size of total client side data and files after clone/checkout (in MB)		181.0	132.0

*

* https://git-scm.com/

Starting a Git Repository

```
Moritz@PC-Moritz MINGW64 ~/Desktop/SiW
$ ls
calculation.py  hello_world.py  README.md

Moritz@PC-Moritz MINGW64 ~/Desktop/SiW
$ git init
Initialized empty Git repository in C:/Users/Moritz/Desktop/SiW/.git/

Moritz@PC-Moritz MINGW64 ~/Desktop/SiW (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        README.md
        calculation.py
        hello_world.py

nothing added to commit but untracked files present (use "git add" to track)

Moritz@PC-Moritz MINGW64 ~/Desktop/SiW (master)
$
```

Git status

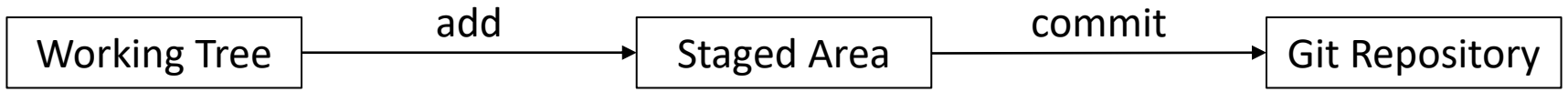
```
Moritz@PC-Moritz MINGW64 ~/Desktop/SiW (master)
$ git add *.py

Moritz@PC-Moritz MINGW64 ~/Desktop/SiW (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   calculation.py
        new file:   hello_world.py

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        README.md


Moritz@PC-Moritz MINGW64 ~/Desktop/SiW (master)
$
```

Git commit

```
Moritz@PC-Moritz MINGW64 ~/Desktop/SiW (master)
$ git commit
[master (root-commit) 4e339be] Initial commit
 2 files changed, 4 insertions(+)
 create mode 100644 calculation.py
 create mode 100644 hello_world.py

Moritz@PC-Moritz MINGW64 ~/Desktop/SiW (master)
$ git log
commit 4e339be144fbb1263274141ba843e60fc5b3d4fd (HEAD -> master)
Author: mwitt95 <moritz.witt@outlook.com>
Date:   Mon May 4 21:03:35 2020 +0200

    Initial commit

    Commit python files into Git repository
    Leaves two files untracked, i.e. not saved in Git repository

Moritz@PC-Moritz MINGW64 ~/Desktop/SiW (master)
$
```

The Stages of a File

- Modified: File is changed but not staged yet
- Staged: File will get saved in next commit
- Committed: File is saved in Git repository



Branching

- Diverging from main line of development
- Branch = Pointer to one commit state
- “Killer feature”
  - Lightweight
  - Nearly instantaneous
  - Switching between branches nearly instantaneous
- git branch *<name>*
- git checkout *<branch_name>*

Merge Branches

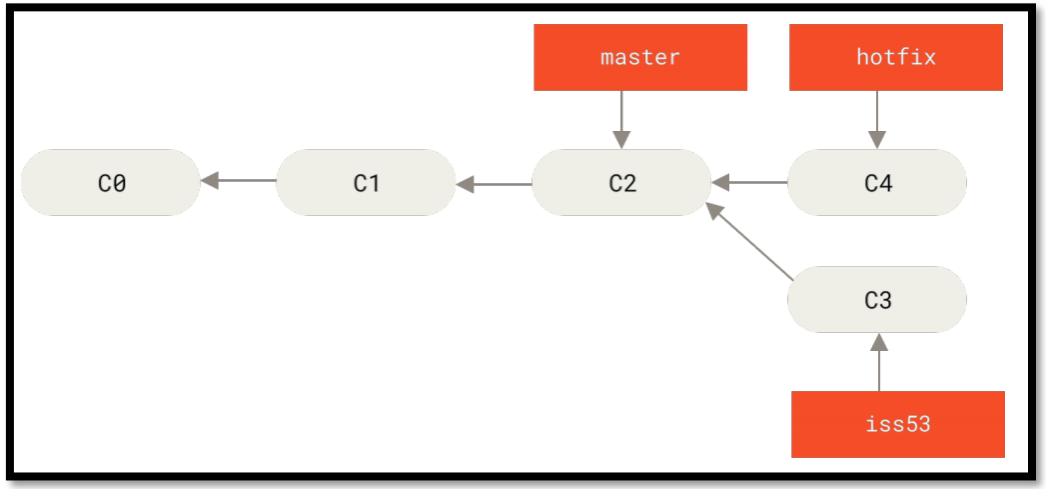
- Merge content of two branches
  - git merge <*branch_to_merge*>
- “Fast-forward”
- “Merge made by recursive strategy”
- CONFLICT

- Workflows:
  - Long-Running Branches
  - Topic Branches

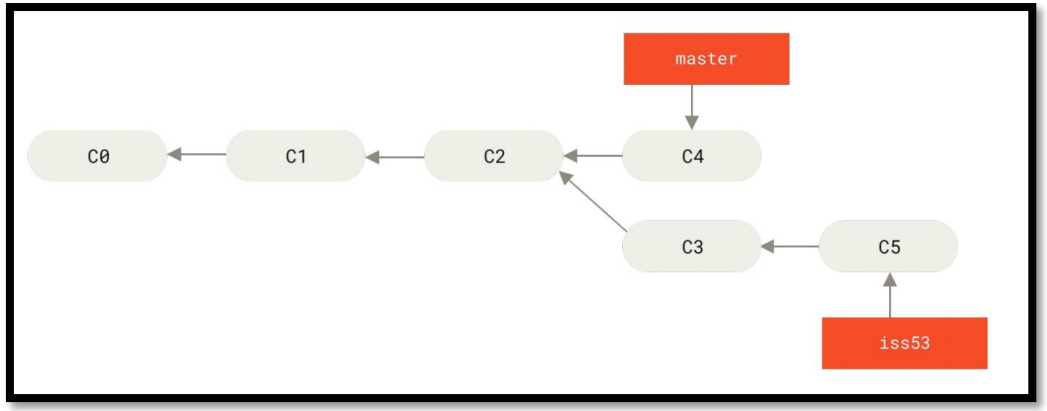
Merge Branches

- Merge content of two branches
  - git merge *<branch_to_merge>*
- "Fast-forward"
- "Merge made by recursive strategy"
- CONFLICT

- Workflows:
  - Long-Running Branches
  - Topic Branches



*

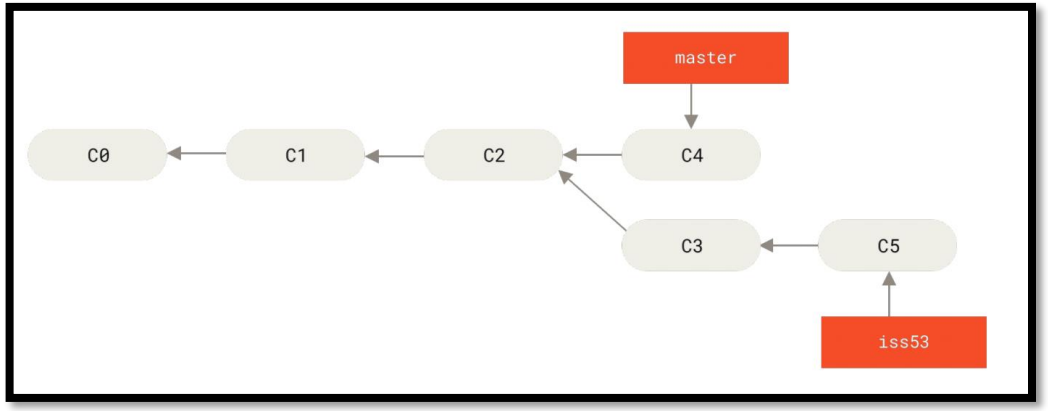


*

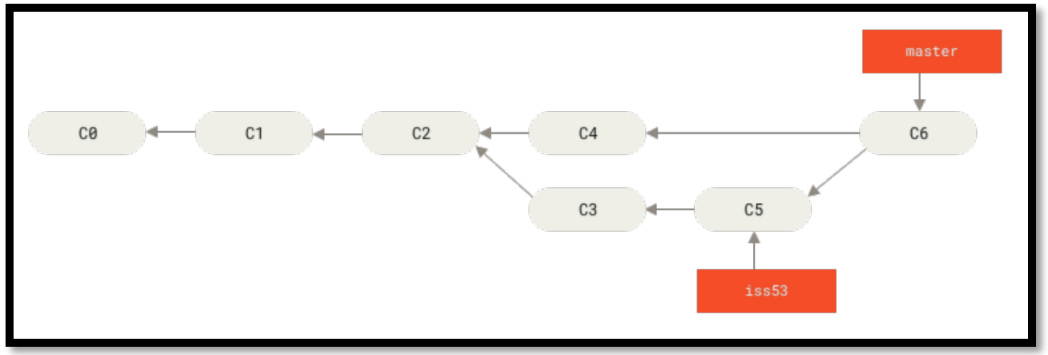
* Pro Git

Merge Branches

- Merge content of two branches
  - git merge *<branch_to_merge>*
- “Fast-forward”
- “Merge made by recursive strategy”
- CONFLICT



*



*

- Workflows:
  - Long-Running Branches
  - Topic Branches

* Pro Git

Merge Branches

- Merge content of two branches
  - git merge <*branch_to_merge*>
- “Fast-forward”
- “Merge made by recursive strategy”
- CONFLICT

- Workflows:
  - Long-Running Branches
  - Topic Branches

Remote repositories

- Project that is hosted somewhere else
- Necessary for collaboration
- git clone *<URL>*
- git remote add *<URL>*
- git fetch *<Remote>*
- git merge *<Remote>*

Git Protocols

- Local
  + Access already exists
  + Easy grabbing from others
  - Difficult to set up
  - Chance of accidental damage
- Git
  + Fastest transfer
  - No authentication
- HTTP
  + One URL for authentication and encryption
  - Sometimes more difficult to set up
- SSH
  + Easy to set up, safe
  - Collaborators need SSH access to the machine, no anonymous access

Git Servers

- Several open source options
- Some explicitly for Git
- Different in
  - Amount of data space per repository
  - Number of collaborators
  - Number of free private and/or open source repositories

- E.g. Bitbucket (only 5 collaborators),
  Visual Studio Team Services (only private repositories) *

* https://git.wiki.kernel.org/index.php/GitHosting

GitLab *

- Databased-backed web application
- Free to use
- Unlimited repositories and collaborators
- Private and public projects

- Different options of access permissions
- Push access or merge requests

* https://about.gitlab.com/press/press-kit/

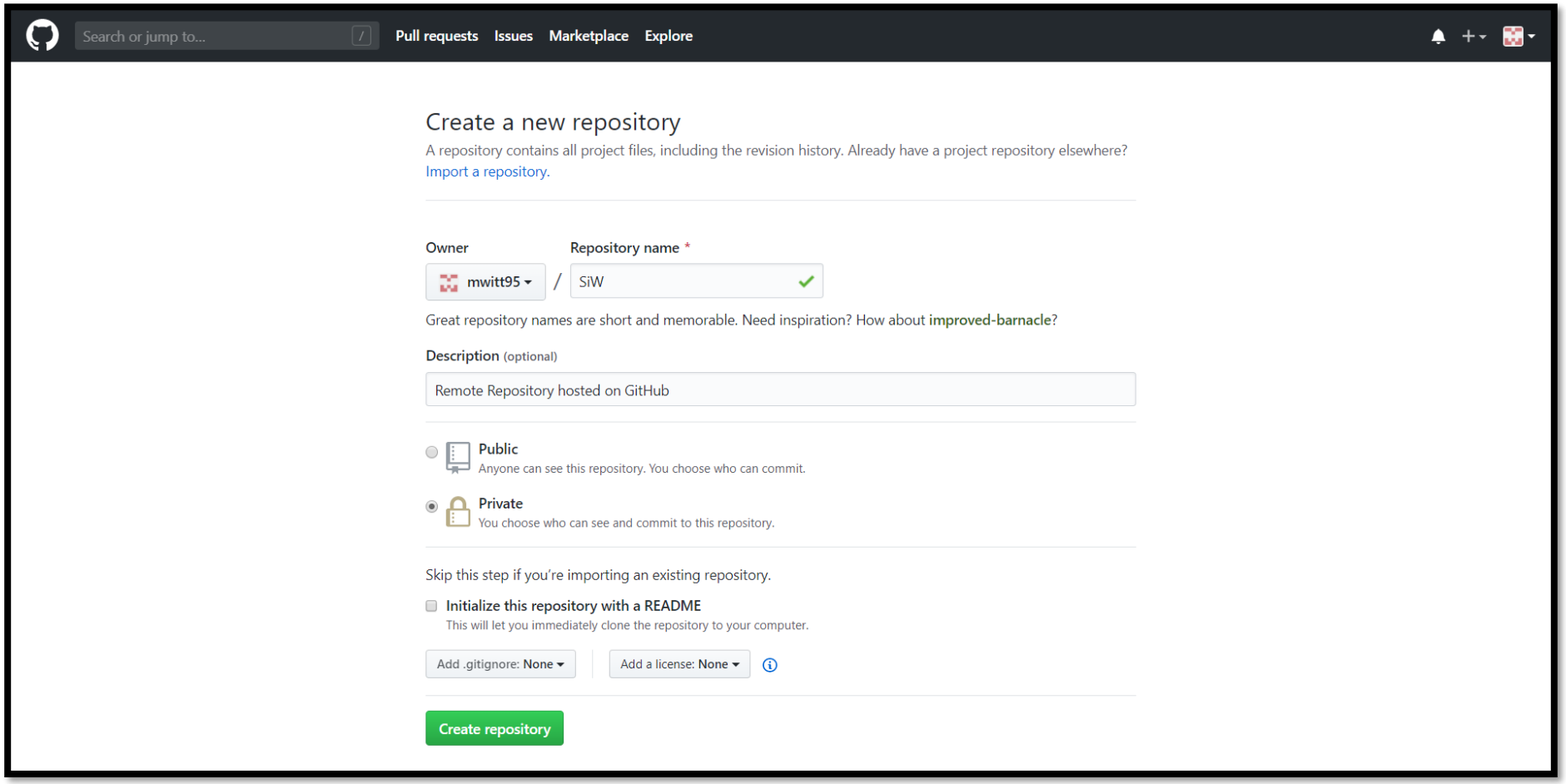
GitHub*

- Largest host for Git repositories
- Unlimited private and open source repositories
- Max three collaborators per repository
- Repositories are deleted if original author leaves GitHub
- Possible two-factor authentication

- GitHub flavored Markdown
- Personalizing GitHub

* https://github.com/logos

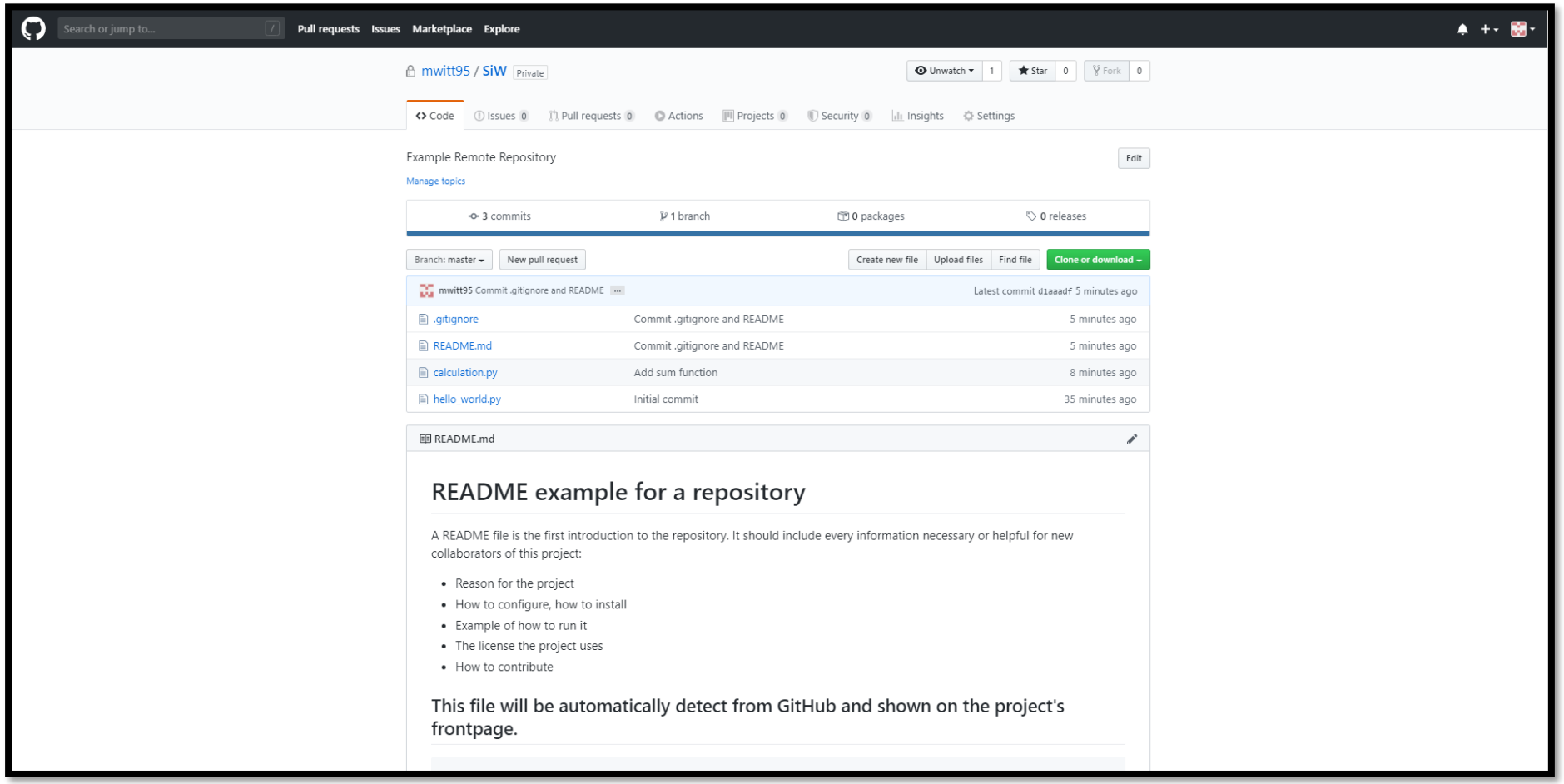
GitHub New Repository



Add Remote from Existing Repository

```
Moritz@PC-Moritz MINGW64 ~/Desktop/SiW (master)
$ git remote add origin https://github.com/mwitt95/SiW.git

Moritz@PC-Moritz MINGW64 ~/Desktop/SiW (master)
$ git remote -v
origin  https://github.com/mwitt95/SiW.git (fetch)
origin  https://github.com/mwitt95/SiW.git (push)

Moritz@PC-Moritz MINGW64 ~/Desktop/SiW (master)
$ git push origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 386 bytes | 386.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/mwitt95/SiW.git
 * [new branch]      master -> master

Moritz@PC-Moritz MINGW64 ~/Desktop/SiW (master)
$
```

GitHub Repository

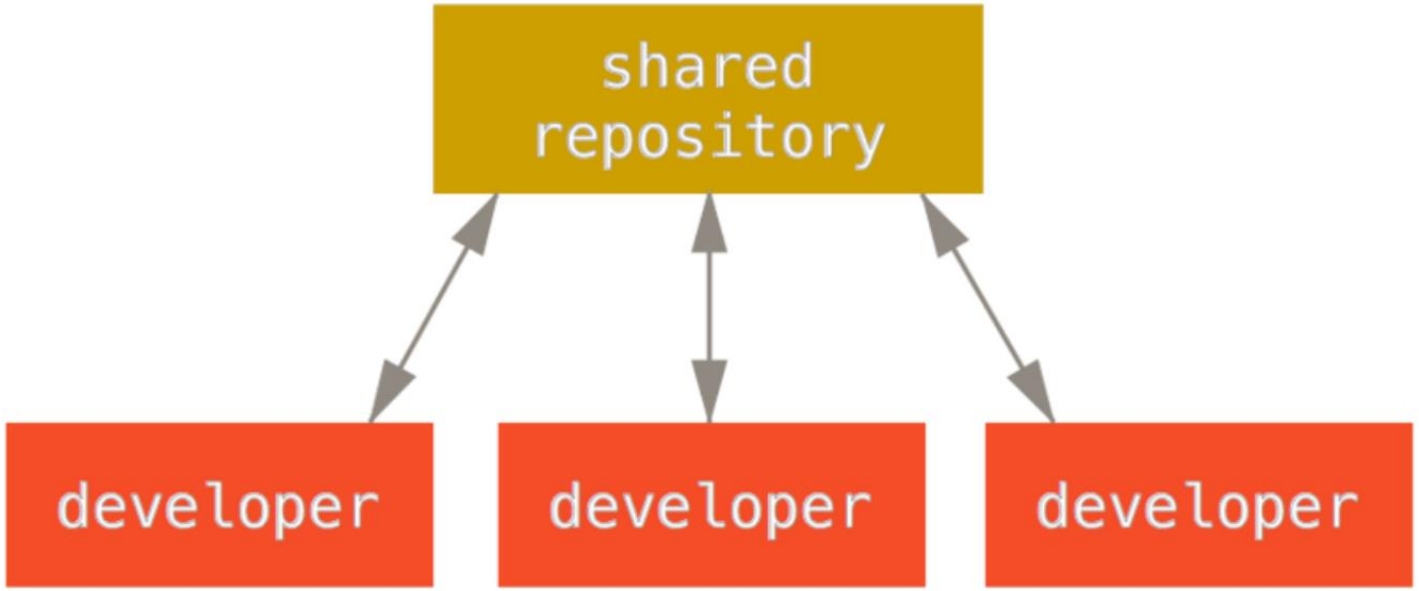


GitHub Workflow

1. Fork the project
2. Create a topic branch
3. Make some commits
4. Push to own GitHub project
5. Open Pull Request
6. Discuss, work in some comments
7. Product owner merges/closes Pull Request
8. Pull updated Master back to fork

Git Workflows

- Centralized



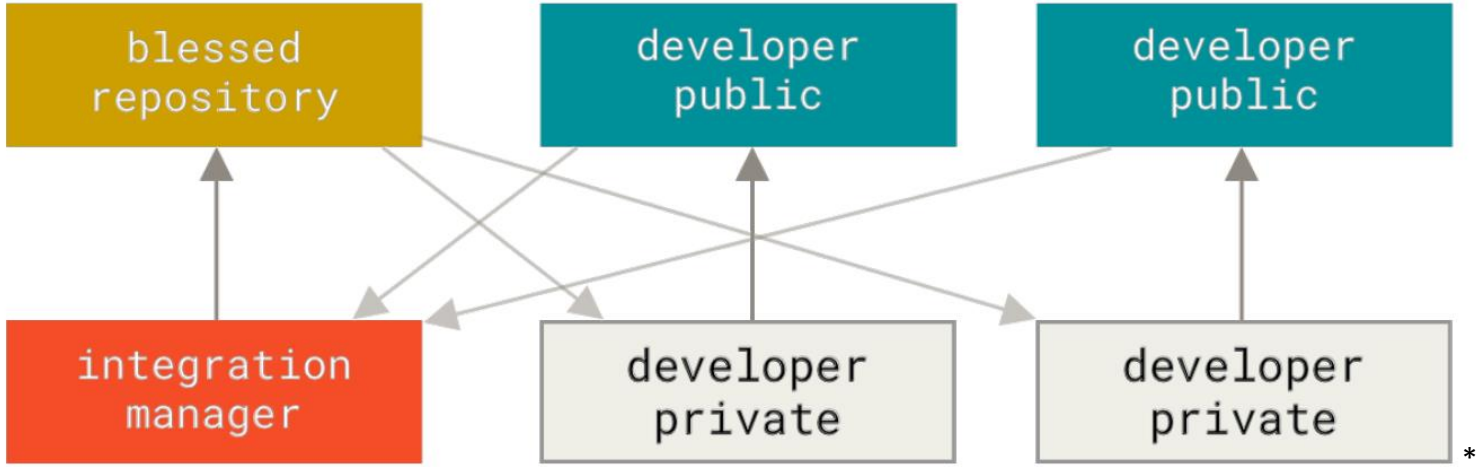
*

- Integration-Manager
- Dictator

* Pro Git

Git Workflows

- Centralized
- Integration-Manager
- Dictator

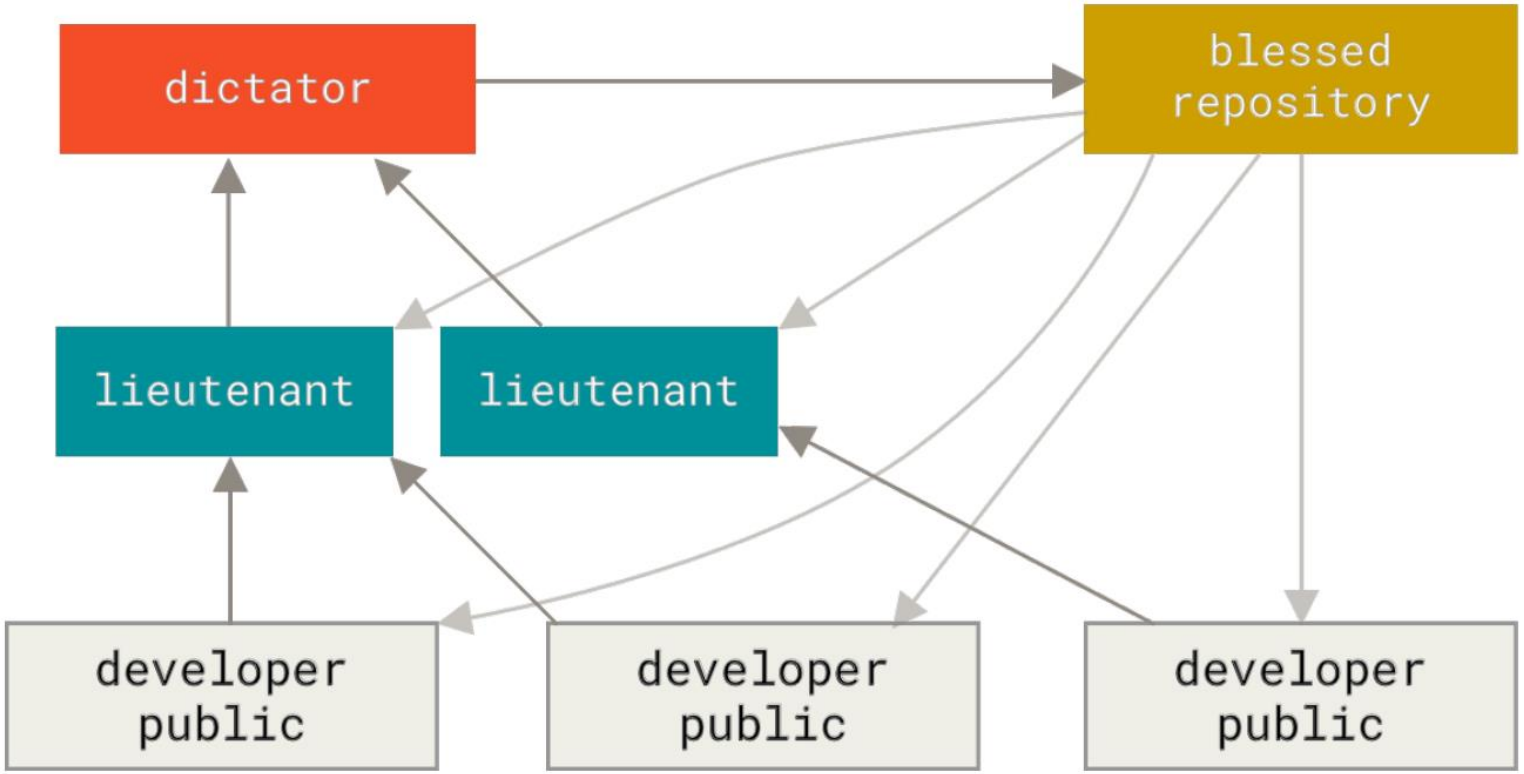


*

* Pro Git

Git Workflows

- Centralized
- Integration-Manager
- Dictator



*

* Pro Git

Commit Guidelines

- Makes collaboration a lot easier
- One commit per issue
- Whitespace errors: `git diff --check`

- Comment rules:
  - Capitalized, short summary
  - More detailed explanation (72 characters)
  - Write in imperative
  - Further paragraphs (next steps) or bullet points

Git with Subversion

- Bidirectional bridge: “git svn clone”
  - Needs to check every commit individually, needs very long!!
- Uses git as valid client, all git features are available
- Linear history, rebase before pushing
- Migration:
  - Push clone to Git server
  - Author information needed
  - Post-import clean-up

Git with Mercurial

- Bridge as remote helper: git-remote-hg
- git clone *<shortname>* *<URL>*
- Usual Git client, all features available
- git push
- Remote helper translates between different name assignment
- Migration:
  - Straightforward due to same structure
  - “hg-fast-export” tool
  - Create author mapping

Summary

*

- Git is the largest Version Control System (VCS)
- A commit is a snapshot of the current state
- Every clone includes the entire history
- Branches are lightweight pointers at specific commits
- Add remote repositories for collaboration
- Several open source Hosts
- Suited for any number of collaborators

*https://git-scm.com/

References

- Pro Git by Chacon and Staub, The Expert Voice; 2nd Edition
- https://git-scm.com/
- https://guides.github.com/
- https://product.hubspot.com/blog/git-and-github-tutorial-for-beginners
- https://insights.stackoverflow.com/survey/2018 - 25.04.20
- https://git.wiki.kernel.org/index.php/GitHosting - 01.05.20
- https://help.github.com/en/github/getting-started-with-github/git-and-github-learning-resources
- https://about.gitlab.com/press/press-kit/
- https://github.com/logos
- https://help.github.com/en/github/importing-your-projects-to-github/adding-an-existing-project-to-github-using-the-command-line
- https://biz30.timedoctor.com/git-mecurial-and-cvs-comparison-of-svn-software/

Git and TFS

- Git-tfs: .NET project, only runs on Windows
  - git tfs clone –with-branches: maps TFVC branches to Git branches
  - Setting of Git configuration necessary
  - Features for branches that aren’t represented in TFVC are complicated
  - git rebase / git merge
- Git-tf: Java projects, not able to have branches
  - git tf clone : shallow copy (only latest version) of repository
  - git tf pull –rebase
- Migration for Git-tfs:
  - Map usernames and format it
  - Full clone of repository
  - Clean got-tfs-id section