Leiden University
Medical Center

Code management with Git

Git and remote repositories

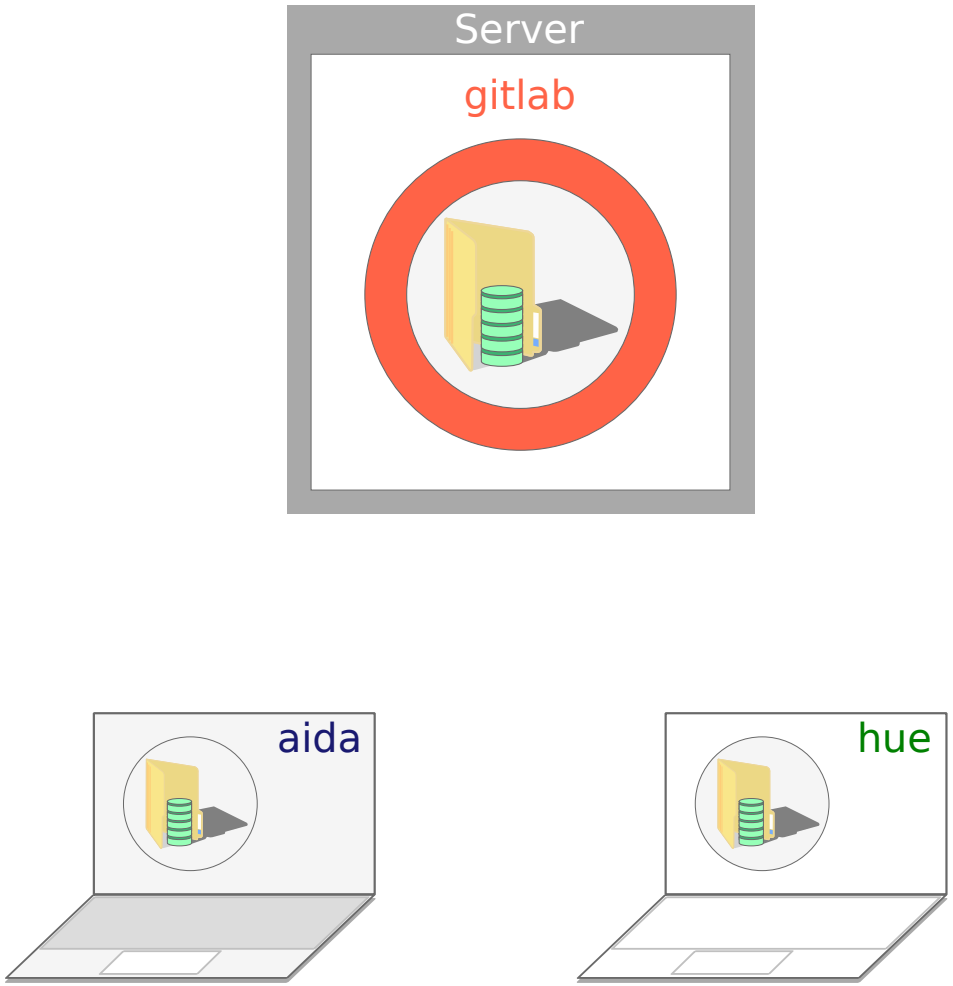
Table of contents

Distributed Git

Repositories can reference each other:

- A repository can be on a server, your desktop, your coworker's laptop, etc.
- Technically, no repository is 'special'.
- We call a reference to another repository a `remote`.

Distributed Git

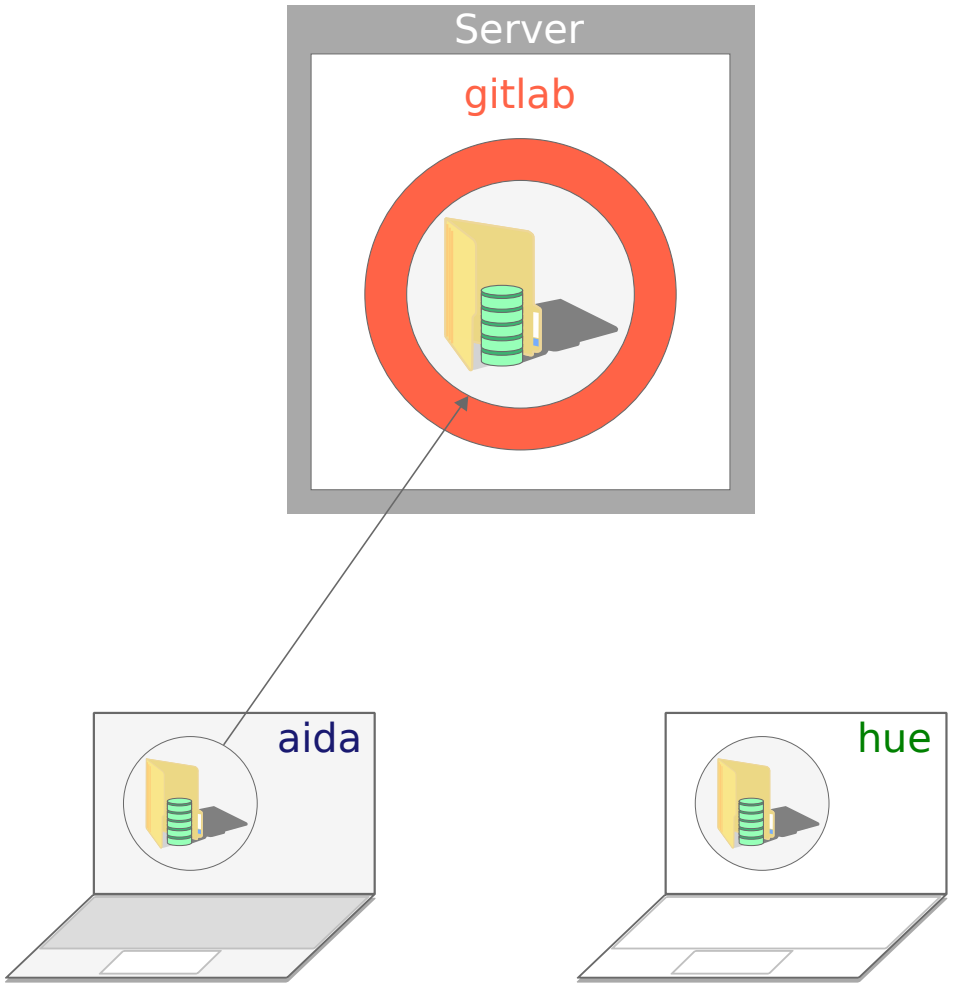


Listing remotes

```
$ git remote
gitlab
```

We are on `aida` and have one remote, `gitlab`, defined.

Listing remotes



Listing remotes

```
$ git remote -v
gitlab https://example.com/zorro/tv-series.git (fetch)
gitlab https://example.com/zorro/tv-series.git (push)
```

`-v`: Include remote location.

We see that communication with `gitlab` is over HTTPS.

Remote repositories

Adding a remote: `git remote add`

```
$ git remote add hue 192.168.0.8:docs/tv-series
```

This adds a reference to the repository on the remote machine with name `hue`.

Adding a remote: `git remote add`

```
$ git remote add hue 192.168.0.8:docs/tv-series
```

This adds a reference to the repository on the remote machine with name hue.

```
$ git remote -v
gitlab https://example.com/zorro/tv-series.git (fetch)
gitlab https://example.com/zorro/tv-series.git (push)
hue 192.168.0.8:docs/tv-series (fetch)
hue 192.168.0.8:docs/tv-series (push)
```

Remote repositories

Adding a remote: `git remote add`

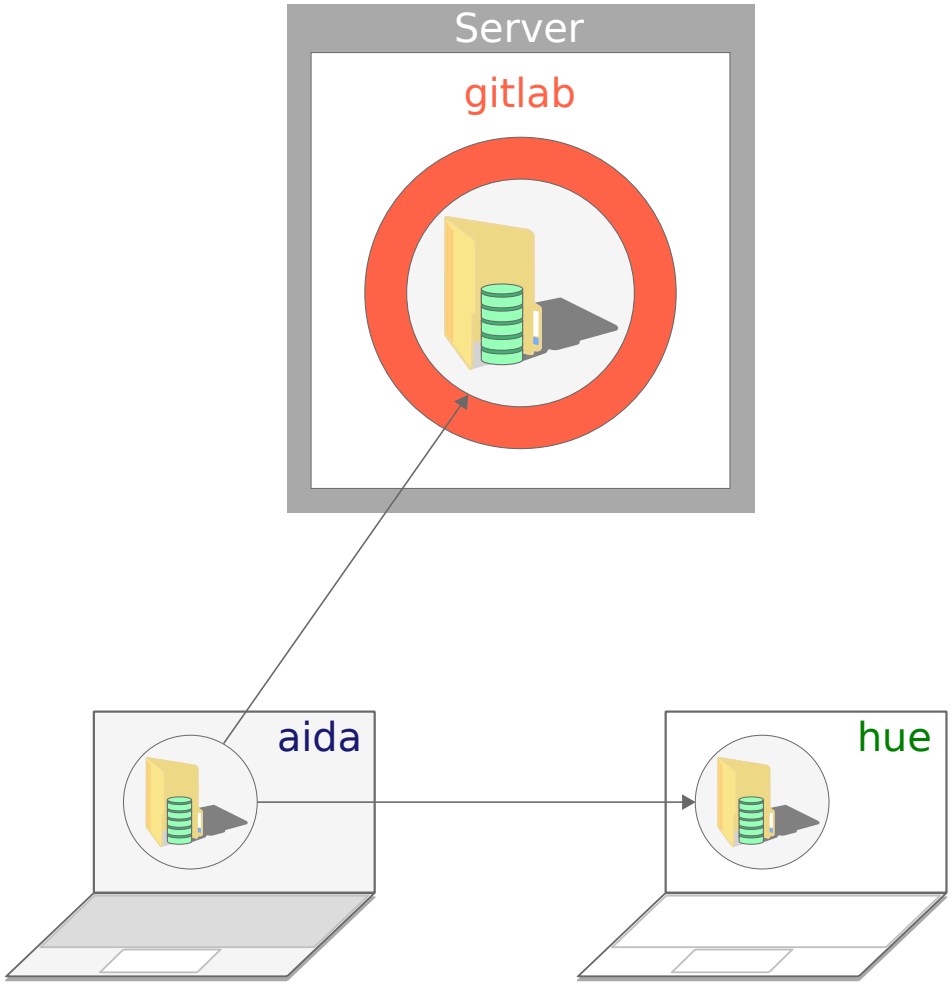


Table of contents

Transferring commits between repositories

Fetching, merging, and pushing

There are three main commands to work with a remote:

- `git fetch` to update our knowledge of the remote.
- `git merge` to use the remote commits.
- `git push` to send our local commits to the remote.

(There's a shortcut for the first two: `git pull`)

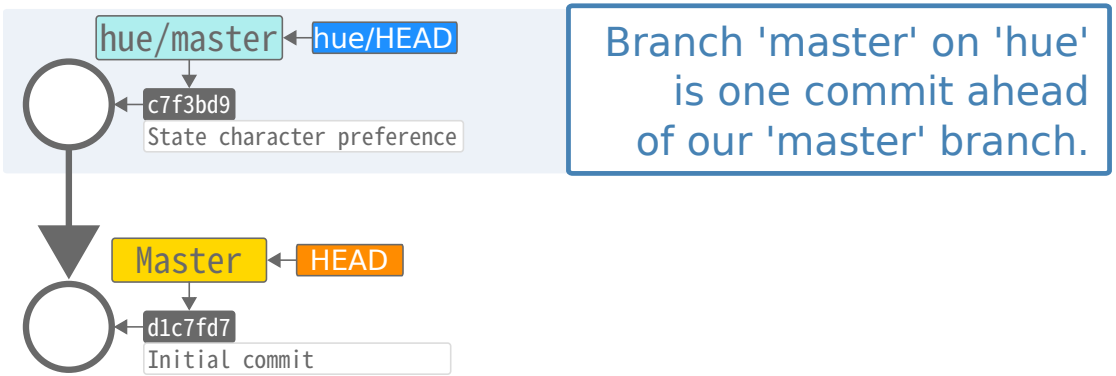
Fetching, merging, and pushing

Updating remote commits: `git fetch`

```
$ git fetch hue
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From 192.168.0.8:docs/tv-series
* [new branch] master -> hue/master
```

Updating remote commits: `git fetch`

```
$ git fetch hue
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From 192.168.0.8:docs/tv-series
* [new branch] master -> hue/master
```



Transferring commits between repositories

Merging remote information: `git merge`

We can `merge` the commits from a remote into our own `master`.

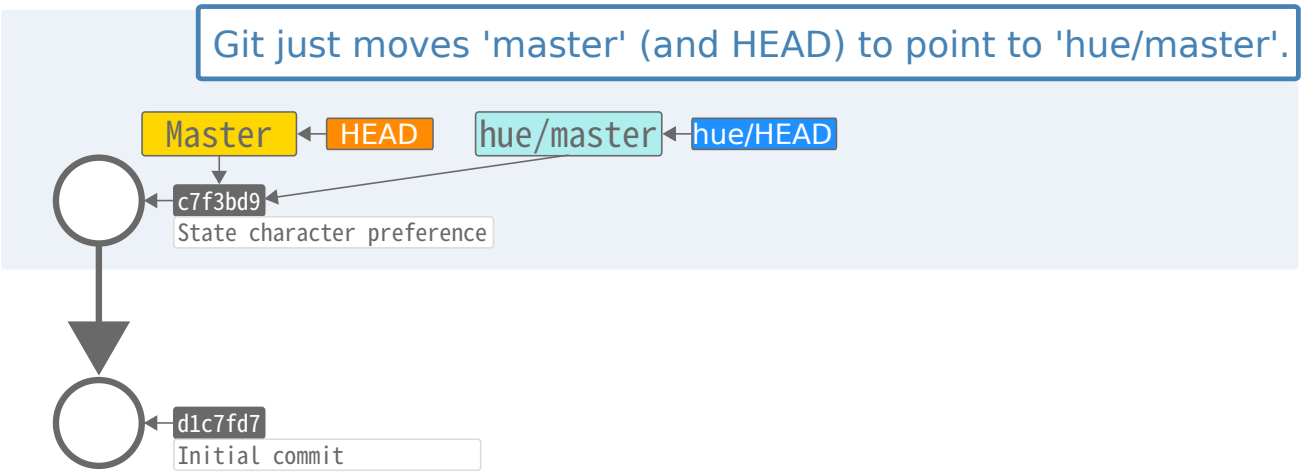
```
$ git merge hue/master
Updating c7f3bd9..251a51b
Fast-forward
testlib.py | 2 +
1 file changed, 2 insertions(+)
```

Transferring commits between repositories

Merging remote information: `git merge`

We can `merge` the commits from a remote into our own `master`.

```
$ git merge hue/master
Updating c7f3bd9..251a51b
Fast-forward
testlib.py | 2 +
1 file changed, 2 insertions(+)
```



Continue adding commits locally

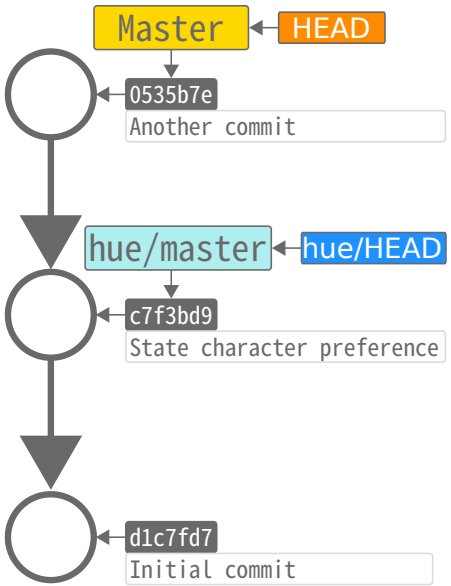
If we add some more commits, our local repository gets ahead of the remote repository.

```
$ ...
$ git commit
```

Continue adding commits locally

If we add some more commits, our local repository gets ahead of the remote repository.

```
$ ...
$ git commit
```



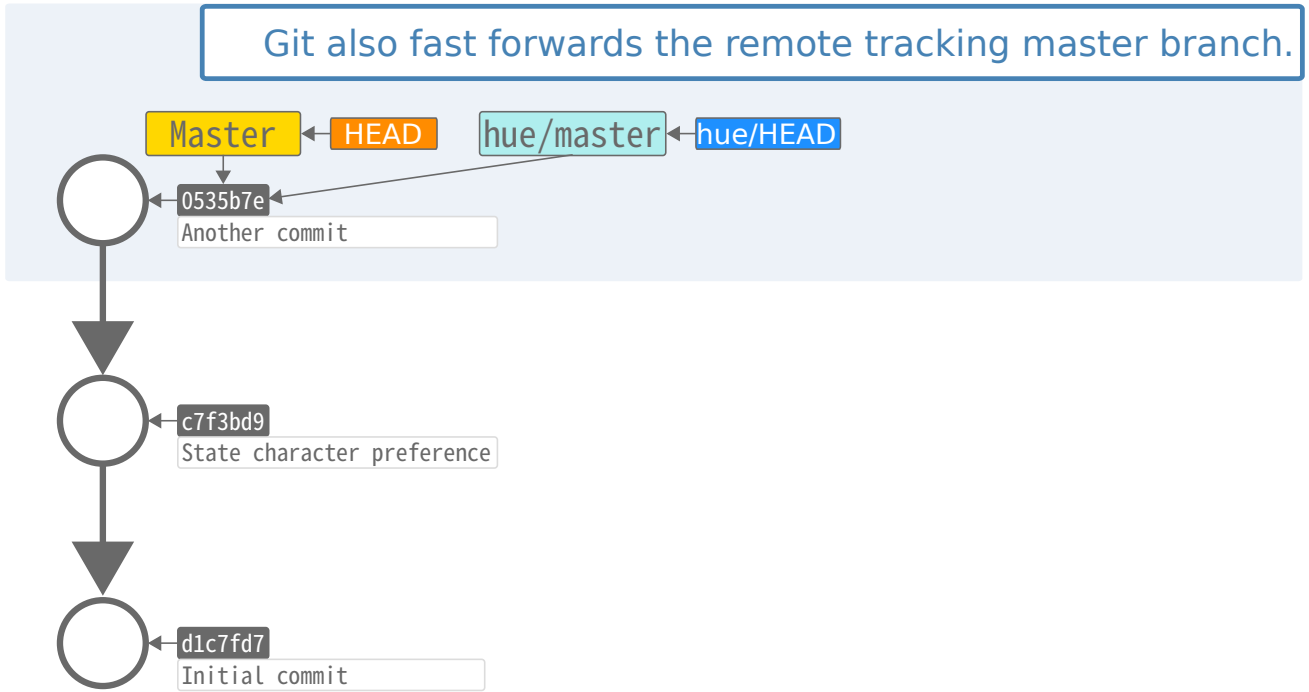
Transferring commits between repositories

Pushing changes to a remote: `git push`

```
$ git push hue master
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 303 bytes, done.
Total 3 (delta 1), reused 0 (delta 0)
To hue.remote:docs/tv-series
0535b7e..0676334 simpsons -> simpsons
```

Transferring commits between repositories

Pushing changes to a remote: `git push`



Cloning an existing repository

Instead of creating repositories using `git init`, you can create a local `clone` of an existing (remote) repository.

```
$ git clone https://example.com/zorro/tv-series.git
Cloning into 'tv-series'...
remote: Counting objects: 6, done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 1), reused 0 (delta 0)
Unpacking objects: 100% (6/6), done.
```

Cloning an existing repository

A remote called `origin` is added for the original repository automatically.

```
$ cd tv-series/
$ git remote -v
origin https://example.com/zorro/tv-series.git (fetch)
origin https://example.com/zorro/tv-series.git (push)
```

Shortcuts for pulling and pushing

The full forms of `git push`/`fetch`/`merge` get boring quickly, so there are some shortcuts.

Shortcuts for pulling and pushing

The full forms of `git push`/`fetch`/`merge` get boring quickly, so there are some shortcuts.

For example, if our remote is called `origin`:

```
$ git push
- instead of -
$ git push origin master
```

```
$ git pull
- instead of -
$ git fetch origin
$ git merge origin/master
```

Shortcuts for pulling and pushing (setup)

The full forms of `git push/fetch/merge` get boring quickly, so there are some shortcuts.

This needs configuration by using `git push` with `-u` once:

```
$ git push origin master -u
```

If your repository was created by cloning, this is already done.

Table of contents

Remote protocols

Git can use two major protocols to transfer data:

- HTTP(S)
- SSH

The HTTP(S) protocol

Most popular protocol when the remote repository is on a server.

```
$ git clone https://example.com/gitproject.git
```

For pushing (or fetching if the repository is private), this asks for your username and password everytime.

The SSH protocol

Most convenient protocol when the remote repository is private or you are a regular contributor.

```
$ git clone user@example.com:gitproject.git
```

This usually requires public/private key authentication.

Table of contents

Using a central server

Git can be used by a team completely decentralized.

However, often a central server is used:

- It can be easier to communicate via the server.
- It can be convenient to have a canonical repository.
- Services such as *GitLab* and *GitHub* add many features on top of Git.

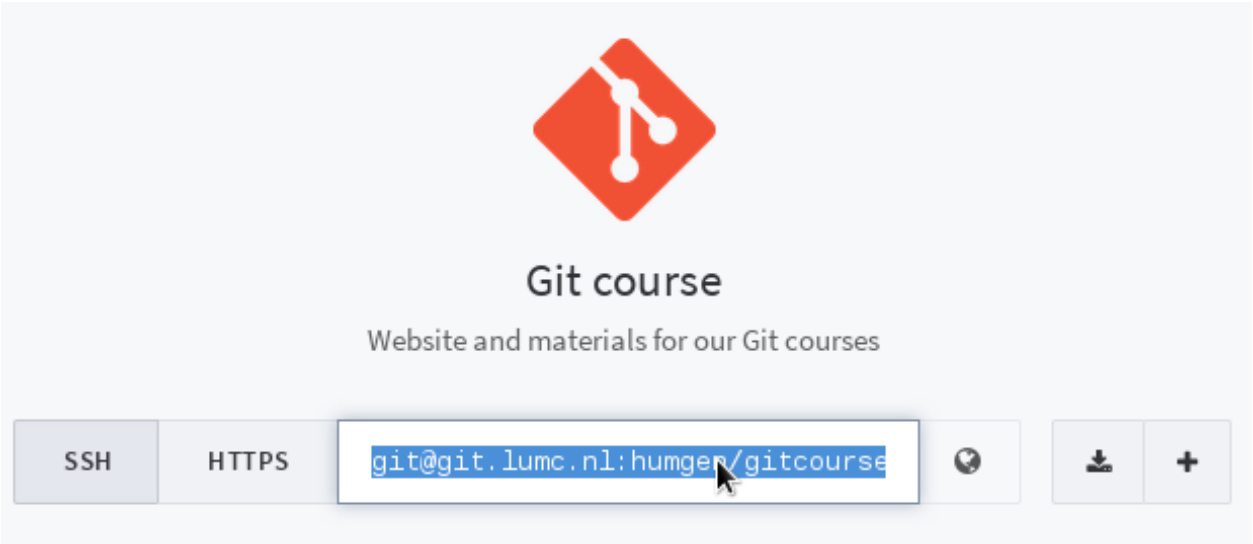
GitLab

Our GitLab server is at `https://git.lumc.nl/`

- Coupled to your LUMC account.
- All users can create projects.
- Browse repositories and edit files online.
- Control access for other users.
- Track bugs/issues/tickets.
- Create merge requests and do code reviews.

GitLab clone URLs

To clone a repository from GitLab, you need its clone URL.



You can choose to use the HTTPS or the SSH protocol.

GitLab Projects

Every project belongs to a single namespace, either a:

- User:
  - The project owner has direct control over the project.
- Group:
  - The group's user-level permissions will take effect.

Every project has a visibility level:

- A way of controling who has **read** access to that project.
- Note that this controls both git "fetch" access as well as access to the web UI for that project.

Project visibility levels

- Private projects:
  - The project owner must explicitly grant access to specific users.
  - Are not listed on the public access directory.
- Internal projects:
  - Can be cloned by any logged in user.
  - Are listed on the public access directory for logged in users.
  - Logged in users have Guest permissions on the repository.
- Public projects:
  - Can be cloned without any authentication.
  - Are listed on the public access directory.
  - Logged in users have Guest permissions on the repository.

Leiden University
Medical Center

Acknowledgements

Martijn Vermaat
Wibowo Arindrarto
Szymon Kiełbasa
Jeroen Laros

`http://git-scm.com/book`
`https://www.atlassian.com/git`