LEIDEN UNIVERSITY MEDICAL CENTER

Git Basics

Jeroen F. J. Laros

Leiden Genome Technology Center

Department of Human Genetics

Center for Human and Clinical Genetics

Initialisation

Starting a project.

Creating a new repository is easy. You do not need a server, no registration, etc.

```
1  $ git init
2  Initialized empty Git repository in <path>/.git/
```

Listing 1 : Make a new repository.

Initialisation

Starting a project.

Creating a new repository is easy. You do not need a server, no registration, etc.

```
1  $ git init
2  Initialized empty Git repository in <path>/.git/
```

Listing 1 : Make a new repository.

Or you can “clone” an existing repository.

```
1  $ git clone <path-to-repository>
```

Listing 2 : Clone an existing repository.

Initialisation

Starting a project.

You can see a hidden directory in a Git repository.

```
1   $ ls -a
2   .   ..   .git
```

Listing 3 : A hidden directory is added.

This directory contains almost everything that Git stores and manipulates.

Manipulation

Local operations.

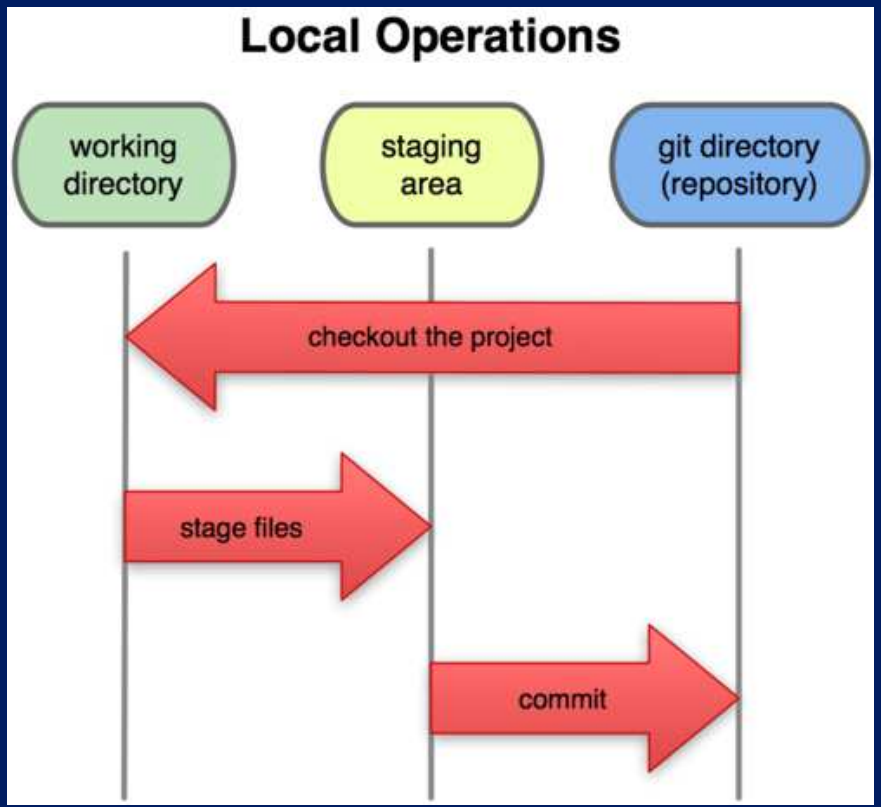

Figure 1 : Working directory, staging area, and Git directory.

Manipulation

Checking the status of your files.

```
1   $ git status
2   nothing to commit (working directory clean)
```

Listing 4 : Check status.

"**`git status`**" can tell you whether your files are:

- Untracked.
- Unmodified.
- Modified.
- Staged.

Manipulation

Checking the status of your files.

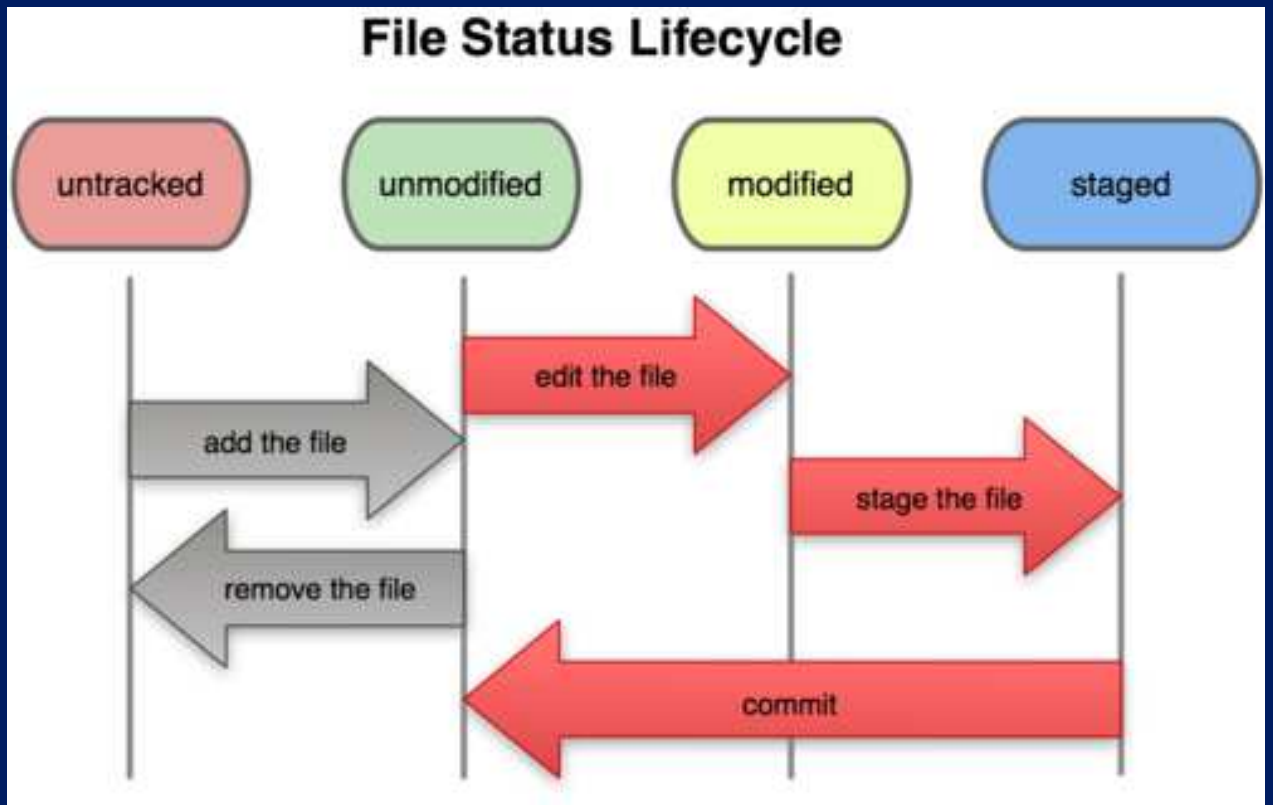

Figure 2 : The lifecycle of the status of your files.

Manipulation

Adding, removing, renaming.

```
1   $ echo First version. > README
2   $ git status
3   # Untracked files:
4   #       README
```

Listing 5 : This file is not tracked.

Manipulation

Adding, removing, renaming.

```
1   $ echo First version. > README
2   $ git status
3   # Untracked files:
4   #       README
```

Listing 5 : This file is not tracked.

```
1   $ git add README
2   $ git status
3   # Changes to be committed:
4   #       new file:   README
```

Listing 6 : Now the file will be tracked.

To rename a file, use “**`git mv`**”, to remove, use “**`git rm`**”.

Working with versions

commit

Once you have modified your files and added them to the *staging area*, you can commit them (add them to the repository).

```
1   $ git commit
```

Listing 7 : Commit a new version.

This will open an editor, give a short description (50 characters) and a list of the changes (72 column format).

- Useful when you want to search.

Working with versions

commit

Once you have modified your files and added them to the *staging area*, you can commit them (add them to the repository).

```
1   $ git commit
```

Listing 7 : Commit a new version.

This will open an editor, give a short description (50 characters) and a list of the changes (72 column format).

- Useful when you want to search.

```
1   $ git commit -m "Solved the counting bug."
```

Listing 8 : Commit a new version.

Working with versions

Undoing changes.

Sometimes you will accidentally add a file.

```
1  $ echo Second version. > README
2  $ git add README
3  $ git status
4  # Changes to be committed:
5  #       modified:   README
```

Listing 9 : Adding a new version of a file.

Working with versions

Undoing changes.

Sometimes you will accidentally add a file.

```
1  $ echo Second version. > README
2  $ git add README
3  $ git status
4  # Changes to be committed:
5  #       modified:   README
```

Listing 9 : Adding a new version of a file.

If you want to undo this, you can use "**`git reset`**".

```
1  $ git reset README
2  $ git status
3  # Changes not staged for commit:
4  #       modified:   README
```

Listing 10 : Unstage a file.

Working with versions

Viewing the history.

To see the history of your project, use "`git log`".

```
$ git add README
$ git commit
$ git log
commit cc61ee7cd72590f3bebcc9e1ff3e9435c7f7dd28
Author: J.F.J. Laros <j.f.j.laros@lumc.nl>
Date:   Fri Oct 11 14:18:13 2013 +0200

    Second version.

commit 8e10be812fd78f69a5e3bac7670c62438161b6b0
Author: J.F.J. Laros <j.f.j.laros@lumc.nl>
Date:   Fri Oct 11 14:17:51 2013 +0200

    First version.
```

Listing 11 : The log of our project.

Working with versions

Undo a commit.

Sometimes we want to undo an entire commit. This is done with "**`git revert`**".

```
1  $ git revert cc61ee7cd72590f3bebcc9e1ff3e9435c7f7dd28
2  $ cat README
3  First version.
```

Listing 12 : Revert a commit.

The hash can be found with "**`git log`**".

Working with versions

Undo a commit.

Sometimes we want to undo an entire commit. This is done with "**`git revert`**".

```
1  $ git revert cc61ee7cd72590f3bebcc9e1ff3e9435c7f7dd28
2  $ cat README
3  First version.
```

Listing 12 : Revert a commit.

The hash can be found with "**`git log`**".

You can also use an unique prefix of this hash, usually six characters is enough.

Working with versions

Viewing changes in detail.

If you want to see what has changed, use "**`git diff`**".

```
1  $ echo Third version. > README
2  $ git diff
3  --- a/README
4  +++ b/README
5  @@ -1 +1 @@
6  -First version.
7  +Third version.
```

Listing 13 : Difference between the working copy and the staging area.

Working with versions

Viewing changes in detail.

If you want to see what has changed, use "**`git diff`**".

```
1  $ echo Third version. > README
2  $ git diff
3  --- a/README
4  +++ b/README
5  @@ -1 +1 @@
6  -First version.
7  +Third version.
```

Listing 13 : Difference between the working copy and the staging area.

For staged files, use the "**`--cached`**" or "**`--staged`**" option.

```
1  $ git diff --cached
```

Listing 14 : Difference between the staging area and the last commit.

Working with versions

Viewing changes in detail.

You can see the differences between any two versions.

```
1  $ git diff cc61ee
2  --- a/README
3  +++ b/README
4  @@ -1 +1 @@
5  -Second version.
6  +Third version.
```

Listing 15 : Difference between the working copy and an other commit.

Working with versions

Viewing changes in detail.

You can see the differences between any two versions.

```
1  $ git diff cc61ee
2  --- a/README
3  +++ b/README
4  @@ -1 +1 @@
5  -Second version.
6  +Third version.
```

Listing 15 : Difference between the working copy and an other commit.

```
1  $ git diff 8e10be cc61ee
```

Listing 16 : Difference between two committed versions.

LUMC

Extras

Explicit no tracking.

Sometimes you do not want to track certain files:

- Executables.
- `pdf` files (if you still have the LaTeX source).
- Python bytecode (`.pyc`) files.
- Files containing passwords.

Extras

Explicit no tracking.

Sometimes you do not want to track certain files:

- Executables.
- **`pdf`** files (if you still have the LaTeX source).
- Python bytecode (**`.pyc`**) files.
- Files containing passwords.

Use the special "**`.gitignore`**" file.

```
1  $ touch notrack.txt
2  $ echo notrack.txt > .gitignore
3  $ git add .gitignore
4  $ git commit
5  $ git status
6  nothing to commit, working directory clean
```

Listing 17 : Ignoring certain files.

Extras

Help.

With the "**`help`**" command get the manual of a particular subcommand.

```
1   $ git help <command>
```

Listing 18 : Get the full manual.

Extras

Help.

With the "**`help`**" command get the manual of a particular subcommand.

```
1   $ git help <command>
```

Listing 18 : Get the full manual.

Example.

```
1   $ git help diff
```

Listing 19 : Get the manual for the diff subcommand.

Questions?

Acknowledgements:

Martijn Vermaat
Zuotian Tatum

http://git-scm.com/book