LEIDEN UNIVERSITY MEDICAL CENTER

Analysis projects skeleton

Jeroen F. J. Laros

Leiden Genome Technology Center

Department of Human Genetics

Center for Human and Clinical Genetics

Introduction

Shared projects

Most of us work on multiple projects with multiple people.

That is why is is convenient to:

- Have everything in one place.
  - Data.
  - Code.
  - Documentation.

Introduction

Shared projects

Most of us work on multiple projects with multiple people.

That is why is is convenient to:

- Have everything in one place.
  - Data.
  - Code.
  - Documentation.
- Have the same structure for all projects.

Starting a project

Project skeleton

Usage:

- Make a clone of the skeleton project.
- Rename the project.
- Create a new project on the server.
- Change the remote `origin` to your new project.

https://git.lumc.nl/lgtc-bioinformatics/project-skeleton

Starting a project

Project skeleton

Usage:

- Make a clone of the skeleton project.
- Rename the project.
- Create a new project on the server.
- Change the remote `origin` to your new project.

Configure your project.

- Choose to make your project public or not.
  - Private by default.
  - Public really means public.
- Add the people that work on this project.

https://git.lumc.nl/lgtc-bioinformatics/project-skeleton

Project structure

Global overview

Project layout:

- analysis
- data
- doc
- src

Ideally, every directory in the project has a `README.md` file.

Project structure

Markdown files

```
1  # Installation
2
3  To install [Git](http://www.git-scm.com/):
4
5      apt-get install git
6
7  Now you can do the following:
8
9  - Make a new repository with `git init`.
10 - Clone an existing repository with `git clone`.
```

Listing 1: Markdown snippet.

Project structure

Markdown files

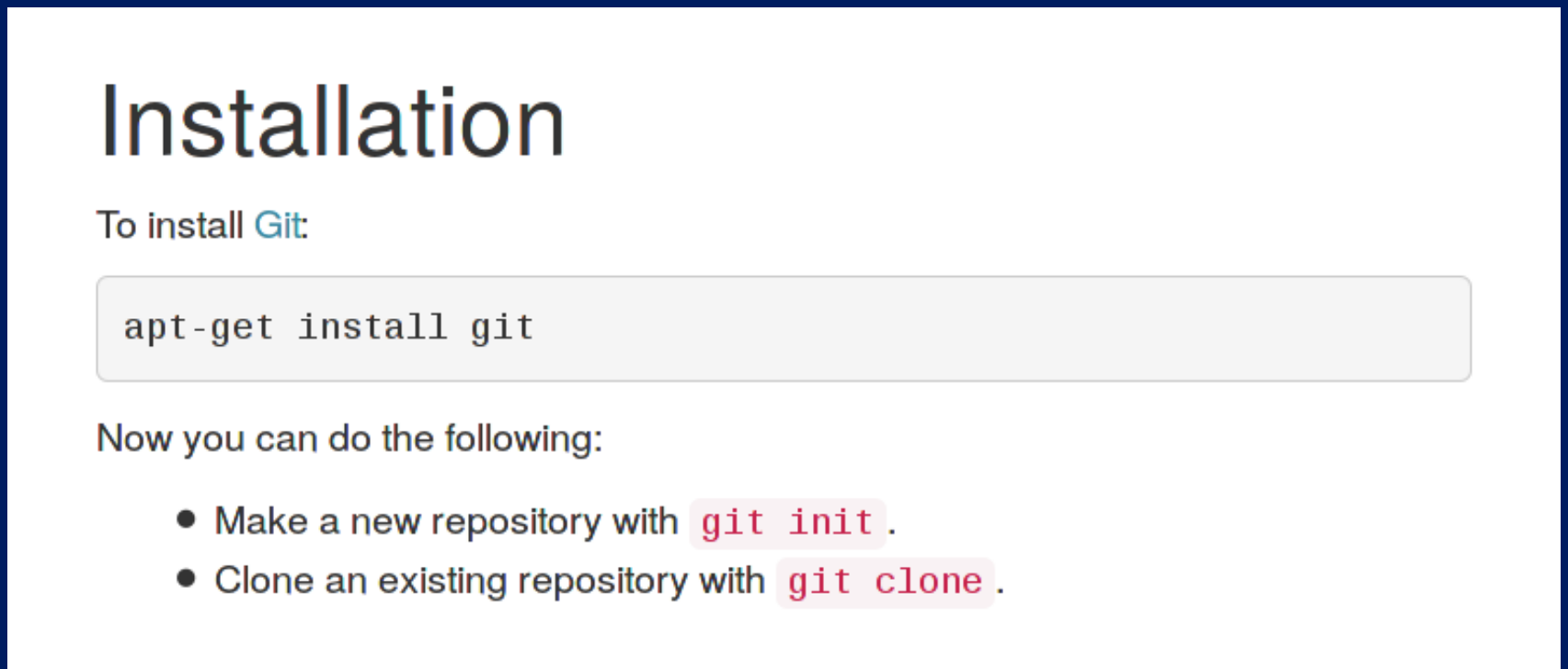

Figure 1: Rendered markdown page.

Project structure

The toplevel "README.md" file

This file contains general information about the project, for example:

- Who leads the project.
- Who participates in the project.
- The amount of hours people have spent on this project.

Project structure

The "doc" directory

Documentation on the project:

- Annotation of the data.
- Goal of the project.
- Related work and literature.
    - You may want to note who provided the documentation.

Project structure

The "data" directory

Used to store all raw data.

The `README.md` contains:

- Description of the delivered data.
  - Sequencing centre.
  - Platform.
  - Molecular type.
  - Owner.
  - Gatherer.
- Description of other data.
  - Perhaps you already got BAM files.
    - Who aligned it?
    - Which aligner?

Project structure

The "analysis" directory

All analysis related files are stored here:

- Symlinks to the actual data.
- Run scripts.
- Make files.
- Result files.

Try to separate self-contained parts of the analysis in their own subdirectories and document dependencies in a `README.md` file.

- Normal data analysis.
- *k*-mer analysis.

Project structure

The "src" directory

Any custom scripts and specific software versions for this project.

When these scripts are useful for other projects, move them to their own repository.

Working with large files

Git is not designed for massive files

Some problems with large files:

- Limited storage on the server.
- Checking out a repository would take a long time.

It also does not make much sense:

- These files are usually *static*.
- And probably *binary*.

http://git-annex.branchable.com/

Working with large files

Git is not designed for massive files

Some problems with large files:

- Limited storage on the server.
- Checking out a repository would take a long time.

It also does not make much sense:

- These files are usually *static*.
- And probably *binary*.

We do want to have some way to track our input and output data. This can be done with **`git-annex`**.

http://git-annex.branchable.com/

Working with large files

Git annex

Manage files with git, without checking their contents in.

- Manage large files without storing them.
- Store file checksums.
- Prevent files from being deleted accidentally.

Working with large files

Git annex

Manage files with git, without checking their contents in.

- Manage large files without storing them.
- Store file checksums.
- Prevent files from being deleted accidentally.

You first have to enable this for your repository.

```
1   $ git annex init "<name>"
```

Listing 2: Enable git-annex.

Working with large files

Adding big files

In our master repository, we annex a file.

```
1   $ git annex add <filename>
2   $ git commit
```

Listing 3: Adding files.

Working with large files

Adding big files

In our master repository, we annex a file.

```
1  $ git annex add <filename>
2  $ git commit
```

Listing 3: Adding files.

In a clone, this file will visible, but not really present.

```
1  $ file <filename>
2  <filename>: broken symbolic link to ...
3  $ git annex get <filename>
```

Listing 4: Make a file available.

Working with large files

Modifying files

Sometimes we need to change the content of a file.

```
1  $ git annex edit <filename>
2  unlock <filename> (copying ...) ok
```

Listing 5: Unlocking a file.

You can use **`git annex add`** when you are done.

Working with large files

Removing files

As long as there are enough copies available, you can remove files.

```
1  $ git annex drop <filename>
2  drop bigfile (unsafe)
3  git-annex: drop: 1 failed
```

Listing 6: A failing drop command.

Working with large files

Removing files

As long as there are enough copies available, you can remove files.

```
1  $ git annex drop <filename>
2  drop bigfile (unsafe)
3  git-annex: drop: 1 failed
```

Listing 6: A failing drop command.

It is actually quite well protected.

```
1  $ rm -rf <repository>
2  rm: cannot remove <repository>/.git/annex/objects/...
```

Listing 7: rm fails too.

Working with large files

Synchronise your results

Let the other repositories know what you have done.

```
1   $ git annex sync
```

Listing 8: Synchronise with all repositories.

Working with large files

Synchronise your results

Let the other repositories know what you have done.

```
1   $ git annex sync
```

Listing 8: Synchronise with all repositories.

You can choose to sync with a selection of repositories.

```
1   $ git annex sync origin
```

Listing 9: Synchronise with a selection.

Working with large files

Cleaning your repository

You can clean your repository with one command.

```
1   $ git clean -f -x
```

Listing 10: Remove untracked files.

option	description
-f	Force (really remove).
-x	Also remove *ignored* files.
-n	Do a *dry run*.

Table 1: Common options.

Working with large files

Working together on the same clone

Sometimes you need to work with other people on the same repository clone.

- Where the large files are stored.

Use the following command to give group access:

```
1   $ find -type d -exec chmod 775 {} \;
2   $ find -type f -exec chmod 664 {} \;
```

Listing 11: Make everyting group writable.

Questions?

Acknowledgements:

Martijn Vermaat
Wibowo Arindrarto
Zuotian Tatum

http://git-annex.branchable.com/

https://git.lumc.nl/lgtc-bioinformatics/project-skeleton