

GitLab as a Collaborative Working Environment

Introductory Course

Mihai Lefter

Department of Human Genetics



Outline

Time	Activity	Contents
09:00	Lecture 1	Introduction, version control.
09:30	Practical 1	GitLab: groups, projects, and file handling.
09:50	Break	
10:00	Lecture 2	GitLab: issues, labels, milestones,
10:20	Practical 2	and the issue board.
10:40	Break	
10:50	Lecture 3	GitLab: wiki, markdown, extras.
11:20	Practical 3	
11:50	Final questions	
12:00	Closing	

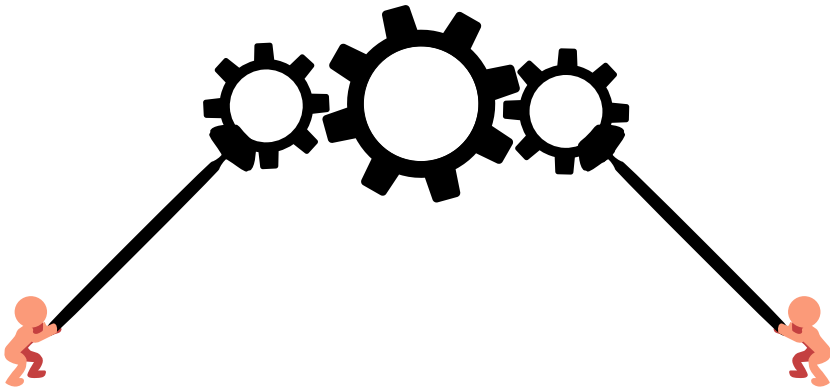
<https://git.lumc.nl/courses/gitlab-intro-course>

Introduction

What is a collaborative working environment

A collaborative working environment supports people, such as e-professionals, in their individual and cooperative work.

— Wikipedia



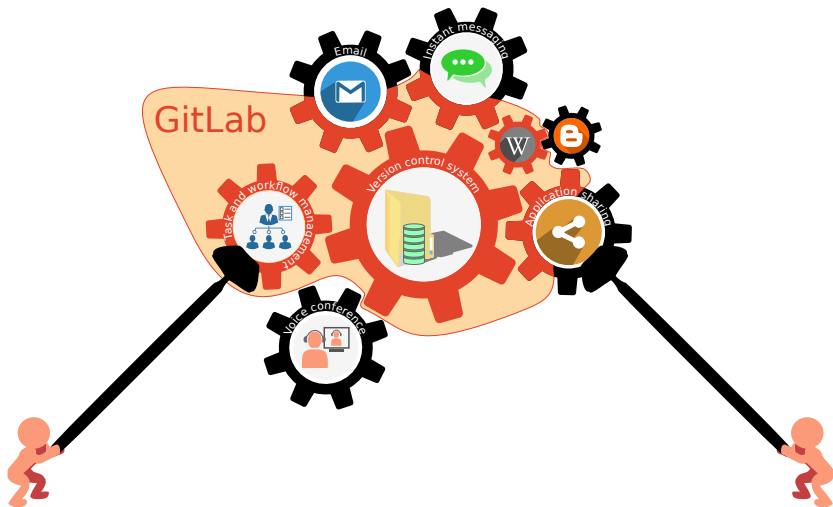
Introduction

Collaborative working environment elements



Introduction

Collaborative working environment elements



What is version control

*The **management of changes** to documents, computer programs, large web sites, and other collections of information.*

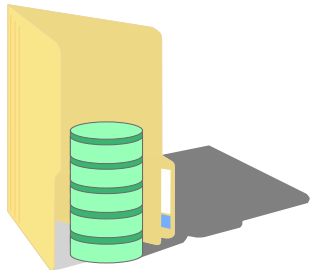
— Wikipedia

*A system that **records changes** to a file or set of files over time so that you can recall specific versions later.*

— <https://git-scm.com/>

General features:

- Keep track of your files in an orderly manner.
- Enables collaboration.



Why should I use it?

For a single user:

- Revert files to a previous state.
- Revert the entire project back to a previous state.
- Review changes made over time.
- Backup.

Why should I use it?

For a single user:

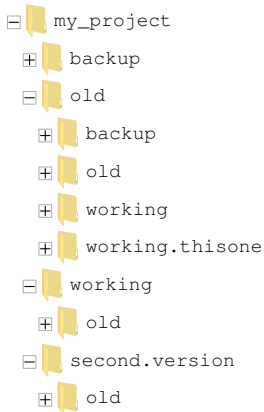
- Revert files to a previous state.
- Revert the entire project back to a previous state.
- Review changes made over time.
- Backup.

For multiple users:

- Reliable way to share files.
- Concurrent working on the same project.
- Conflict resolution.
- See who made which changes at which time.

Why should I not use version control?

- I have my own system.



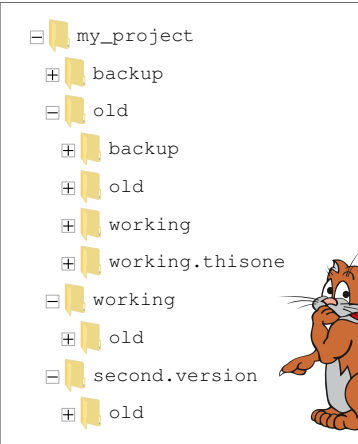
```
graph TD; my_project[my_project] --> backup1[backup]; my_project --> old1[old]; old1 --> backup2[backup]; old1 --> old2[old]; old2 --> working1[working]; working1 --> working_thisone[working.thisone]; working1 --> working[working]; working --> old3[old]; old3 --> second_version[second.version]; second_version --> old4[old];
```

The diagram illustrates a directory structure for a project named 'my_project'. It shows a hierarchy of folders and subfolders, with some folders containing further subfolders. The structure is as follows:

- my_project
 - backup
 - old
 - backup
 - old
 - working
 - working.thisone
 - working
 - old
 - second.version
 - old

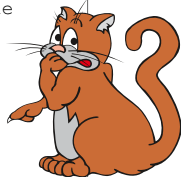
Why should I not use version control?

- I have my own system.



A file explorer window showing a directory structure with the following folders:

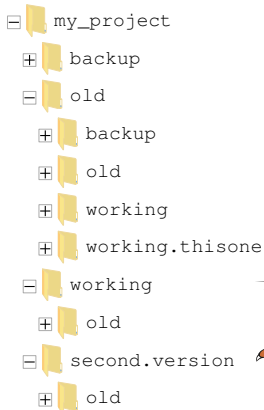
- my_project
 - backup
 - old
 - backup
 - old
 - working
 - working.thisone
- working
 - old
- second.version
 - old



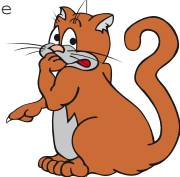
Why should I not use version control?

A list of common excuses:

- I have my own system.
- It is too much work.
- I am the only one working on this project.
- This code will not be used by anyone else.
- The bugs can be tracked forever.
- ...



```
my_project
├── backup
├── old
│   ├── backup
│   ├── old
│   ├── working
│   └── working.thisone
├── working
│   ├── old
│   └── second.version
└── old
```



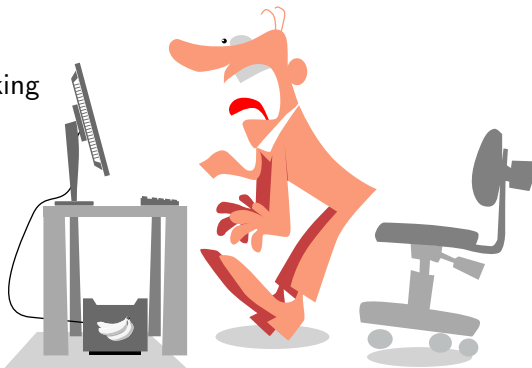
Why should I not use version control?

A list of common excuses:

- I have my own system.
- It is too much work.
- I am the only one working on this project.
- This code will not be used by anyone else.
- The bugs can be tracked forever.
- ...

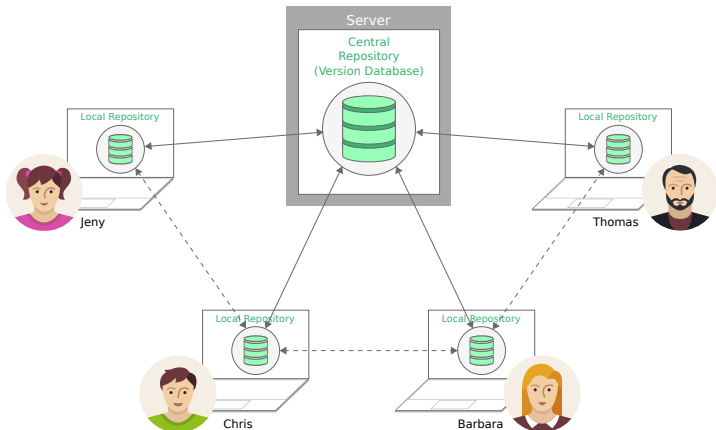
Eventually leading to:

- I'm too busy rewriting the code I accidentally deleted.



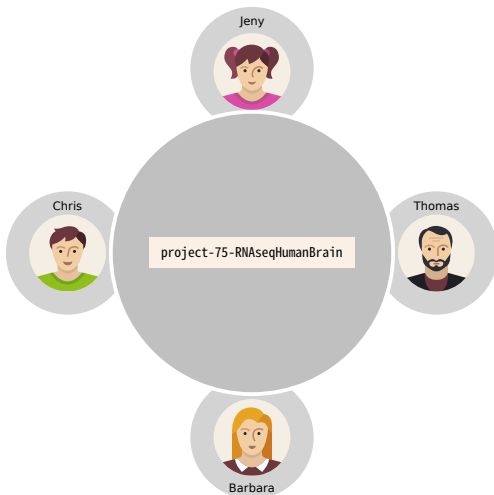
Version Control

Overview (git based)



Version Control

Example: new project meeting



Version Control

Example: new project creation

project-75-RNAseqHumanBrain

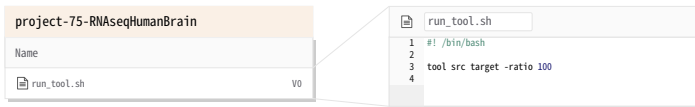
Name



Starts working and creates the project.

Version Control

Example: new file



Creates and adds a script to the project.

Version Control

Example: first results

project-75-RNAseqHumanBrain

Name

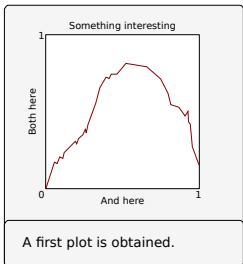
run_tool.sh V0

run_tool.sh

```
1 #!/bin/bash
2
3 tool src target -ratio 100
4
```

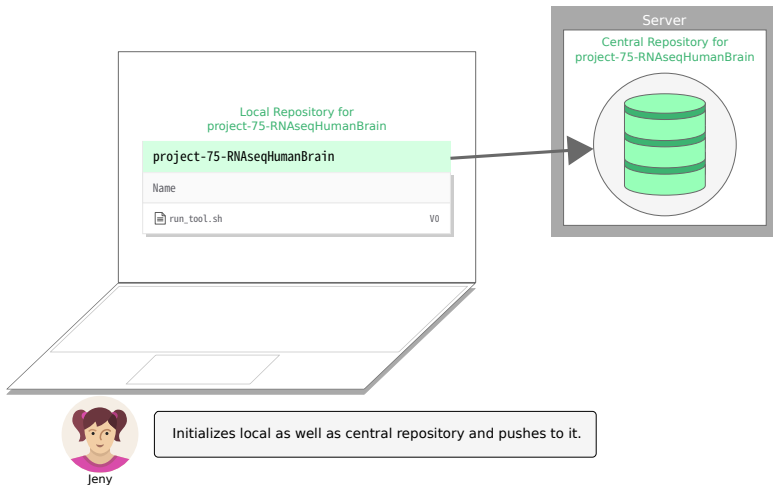


Jeny



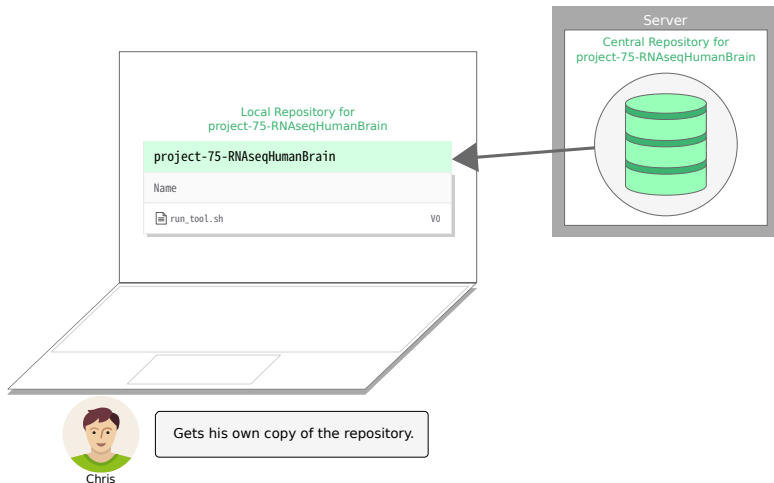
Version Control

Example: create and push to central repository



Version Control

Example: clone central repository



Version Control

Example: edit a file

project-75-RNAseqHumanBrain

Name

run_tool.sh V0

run_tool.sh

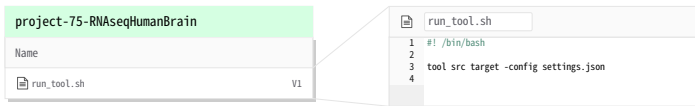
```
1 #!/bin/bash
2
3 tool src target -ratio-100 -config settings.json
4
```



Makes some changes to the script.

Version Control

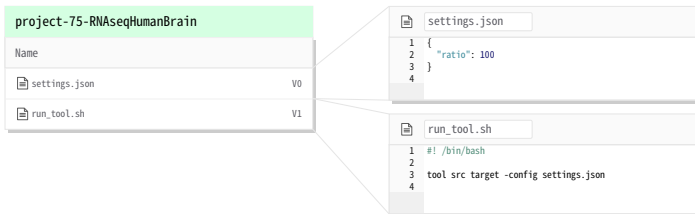
Example: save file to local repository



Saves the new version of the file.

Version Control

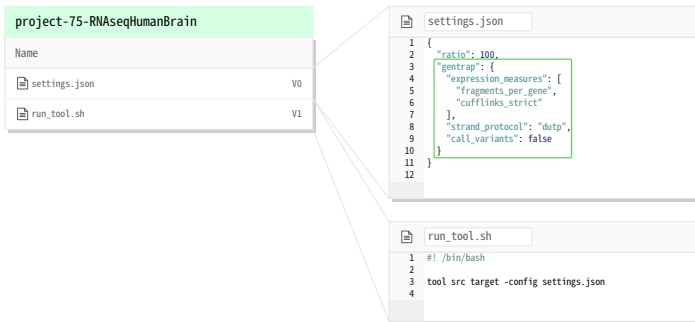
Example: add another file to the local repository



Creates and saves a new file.

Version Control

Example: edit second file

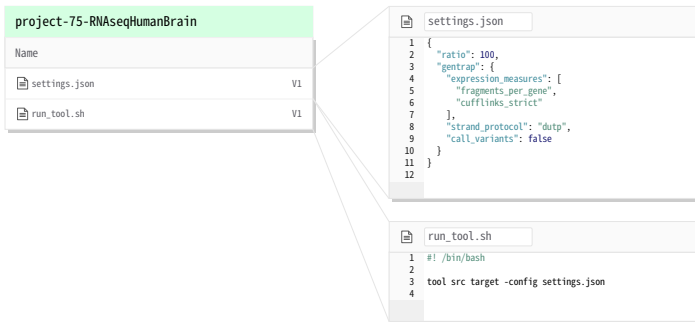


Chris

Makes changes to the new file.

Version Control

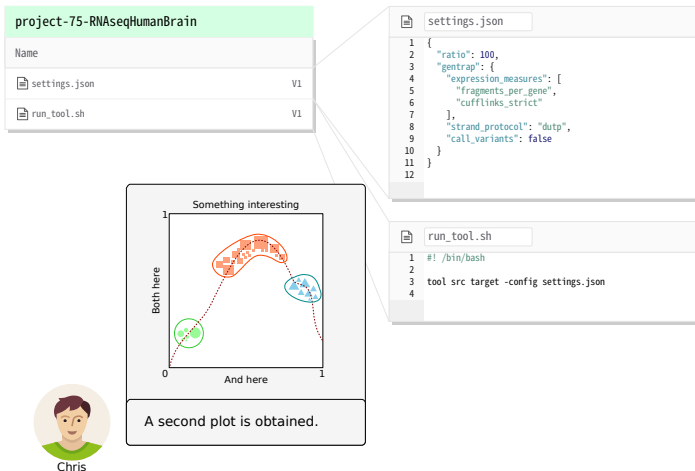
Example: save the second file to the local repository



Saves the new version of the second file.

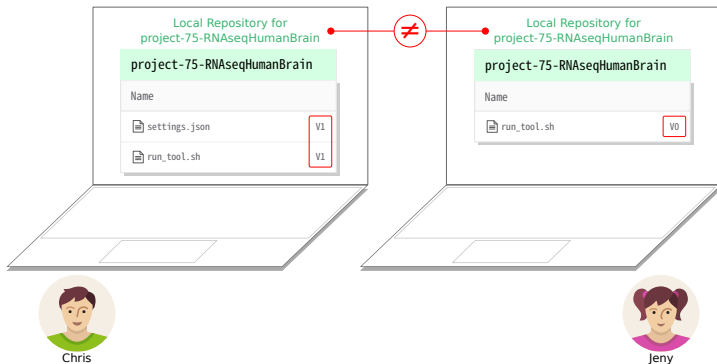
Version Control

Example: more results are obtained



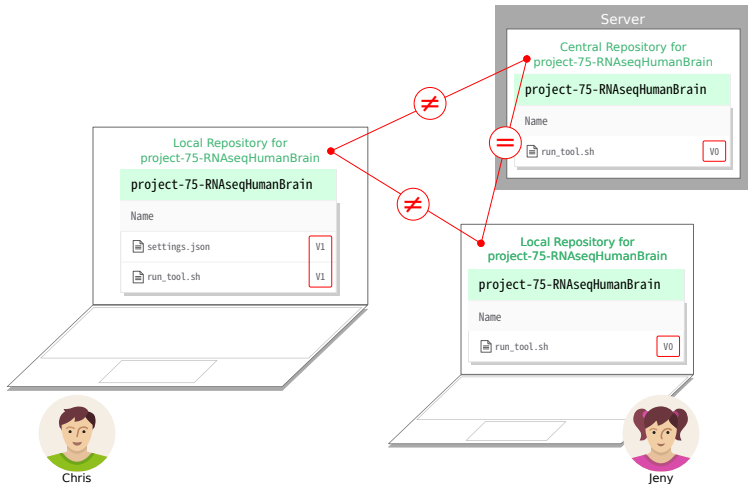
Version Control

Example: local repositories are not synchronized



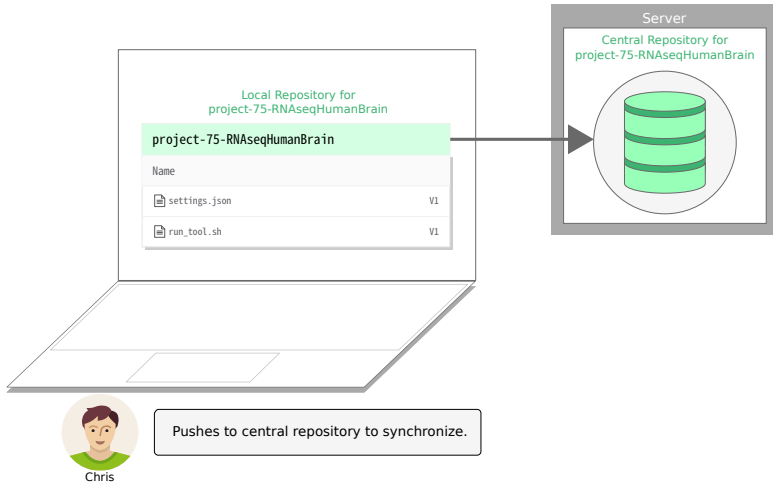
Version Control

Example: what about the server?



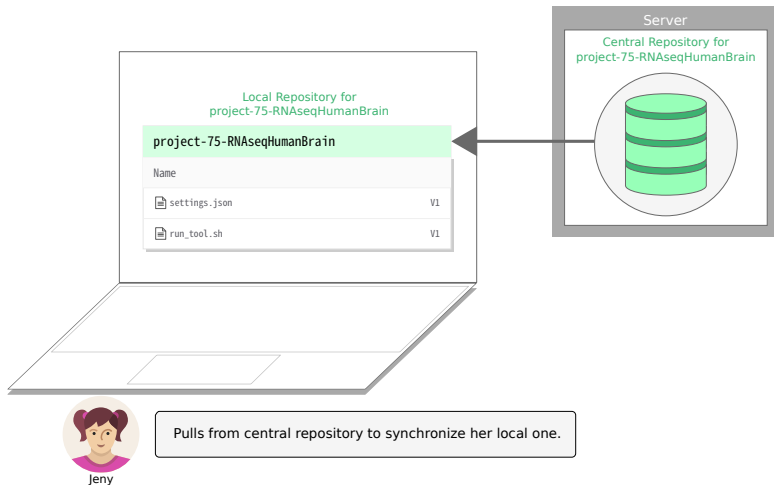
Version Control

Example: update central repository



Version Control

Example: update local repository



Version Control

Example: edit again the second file

project-75-RNAseqHumanBrain

Name	
settings.json	V1
run_tool.sh	V1

settings.json

```
1 {
2   "ratio": 100,
3   "gentrap": {
4     "expression_measures": [
5       "fragments_per_gene",
6       "cufflinks_strict"
7     ],
8     "strand_protocol": "dntp",
9     "call_variants": false,
10    "aligner": "gsnap",
11    "gsnap": {
12      "exe": "/usr/local/bin/gsnap",
13      "npaths": 1,
14      "threads": 8
15    }
16  }
17 }
18
```

run_tool.sh

```
1 #!/bin/bash
2
3 tool src target -config settings.json
4
```



Makes more changes to the configuration.

Version Control

Example: save again the second file to the repository

project-75-RNAseqHumanBrain

Name	
settings.json	V2
run_tool.sh	V1

settings.json

```
1 {
2   "ratio": 100,
3   "gentrap": {
4     "expression_measures": [
5       "fragments_per_gene",
6       "cufflinks_strict"
7     ],
8     "call_variants": false,
9     "aligner": "gsnap",
10    "gsnap": {
11      "exe": "/usr/local/bin/gsnap",
12      "npaths": 1,
13      "threads": 8
14    }
15  }
16 }
17
```

run_tool.sh

```
1 #!/bin/bash
2
3 tool src target -config settings.json
4
```



Saves the new version of the configuration file.

Version Control

Example: more results

project-75-RNAseqHumanBrain

Name	
settings.json	V2
run_tool.sh	V1

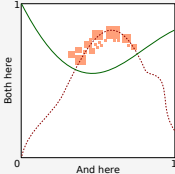
settings.json

```
1 {
2   "ratio": 100,
3   "gentrap": {
4     "expression_measures": [
5       "fragments_per_gene",
6       "cufflinks_strict"
7     ],
8     "call_variants": false,
9     "aligner": "gsnap",
10    "gsnap": {
11      "exe": "/usr/local/bin/gsnap",
12      "npaths": 1,
13      "threads": 8
14    }
15  }
16 }
17
```

run_tool.sh

```
1 #!/bin/bash
2
3 tool src target -config settings.json
4
```


Something interesting



Both here

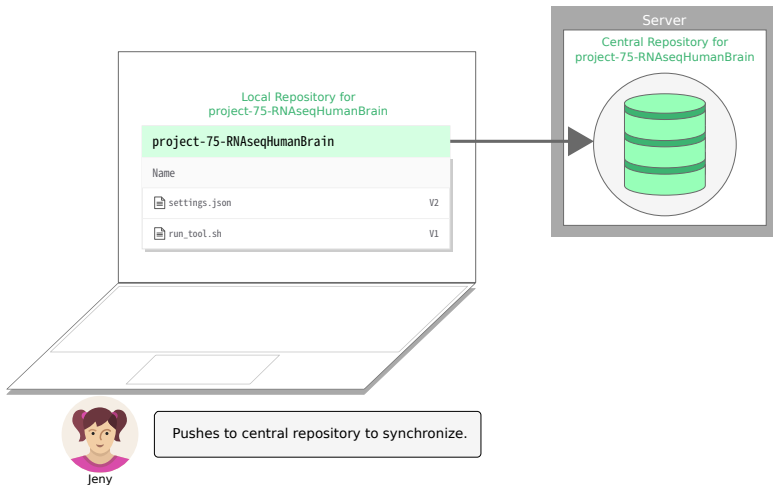
And here

The plot is updated.

 Jeny

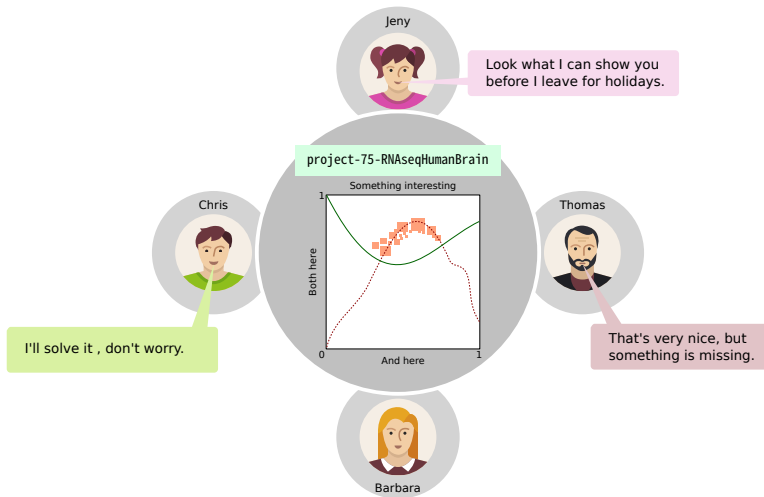
Version Control

Example: update central repository again



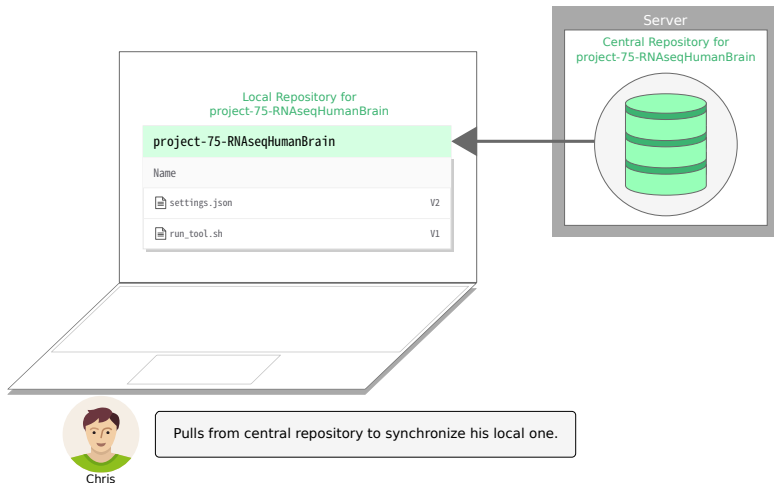
Version Control

Example: intermediate results project discussion



Version Control

Example: update local repository again



Version Control

Example: check the differences with an older version

VS

settings.json V1

```
1 {
2   "ratio": 100,
3   "gentrap": {
4     "expression_measures": [
5       "fragments_per_gene",
6       "cufflinks_strict"
7     ],
8     "strand_protocol": "dutp",
9     "call_variants": false
10  }
11 }
12
```

settings.json V2


```
1 {
2   "ratio": 100,
3   "gentrap": {
4     "expression_measures": [
5       "fragments_per_gene",
6       "cufflinks_strict"
7     ],
8     "call_variants": false,
9     "aligner": "gsnap",
10    "gsnap": {
11      "exe": "/usr/local/bin/gsnap",
12      "npaths": 1,
13      "threads": 8
14    }
15  }
16 }
17
```

Finds deleted line.

settings.json

...	...	
5	5	"fragments_per_gene",
6	6	"cufflinks_strict"
7	7],
8	-	"strand_protocol": "dutp",
9	-	"call_variants": false
10	8	"call_variants": false,
11	9	"aligner": "gsnap",
12	10	"gsnap": {
13	11	"exe": "/usr/local/bin/gsnap",
14	12	"npaths": 1,
15	13	"threads": 8
16	14	}
17	15	}
18	16	}
...	...	

7 additions and 2 deletions

 Chris

Version Control

Example: make changes

project-75-RNAseqHumanBrain

Name	
settings.json	V2
run_tool.sh	V1

settings.json

```
1 {
2   "ratio": 100,
3   "gentrap": {
4     "expression_measures": [
5       "fragments_per_gene",
6       "cufflinks_strict"
7     ],
8     "strand_protocol": "dutp",
9     "call_variants": false,
10    "aligner": "gsnap",
11    "gsnap": {
12      "exe": "/usr/local/bin/gsnap",
13      "npaths": 1,
14      "threads": 8
15    }
16  }
17 }
18
```

run_tool.sh

```
1 #!/bin/bash
2
3 tool src target -config settings.json
4
```



Repairs the configuration.

Version Control

Example: first results draft

project-75-RNAseqHumanBrain

Name	
settings.json	V3
run_tool.sh	V1

Something interesting

Real results.

Chris

settings.json

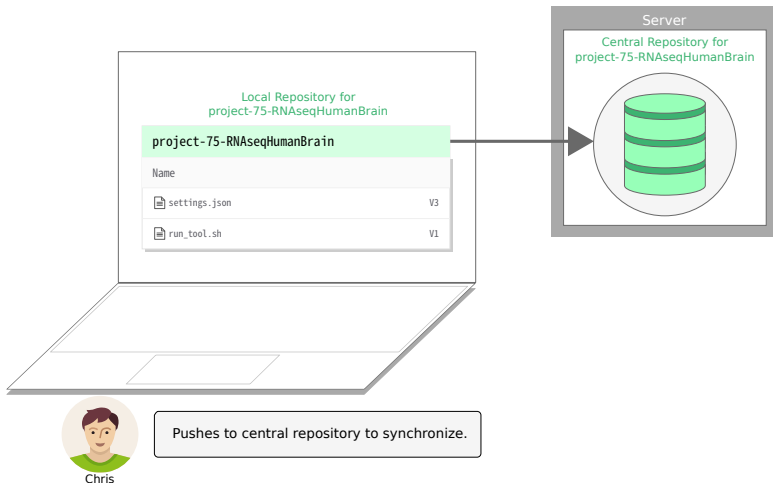
```
1 {
2   "ratio": 100,
3   "gentrap": {
4     "expression_measures": [
5       "fragments_per_gene",
6       "cufflinks_strict"
7     ],
8     "strand_protocol": "dutp",
9     "call_variants": false,
10    "aligner": "gsnap",
11    "gsnap": {
12      "exe": "/usr/local/bin/gsnap",
13      "npaths": 1,
14      "threads": 8
15    }
16  }
17 }
18
```

run_tool.sh

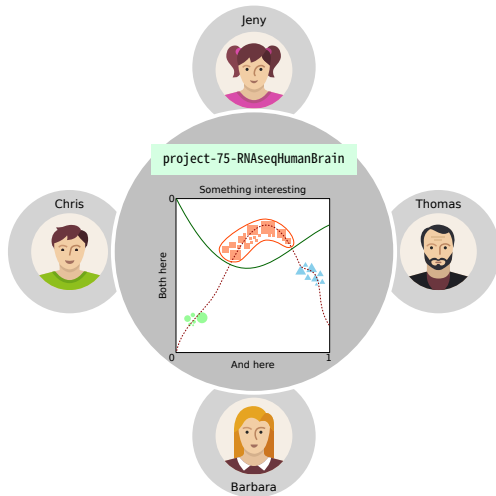
```
1 #!/bin/bash
2
3 tool src target -config settings.json
4
```

Version Control

Example: final central repository update



Example: results discussion meeting





What is GitLab?

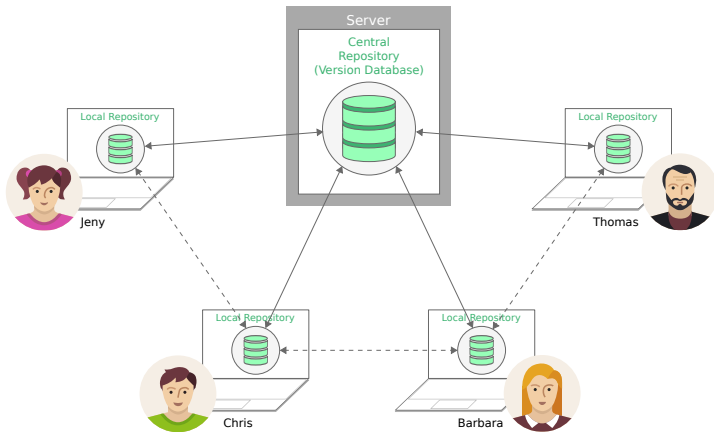
GitLab is a fully integrated software development platform that enables you and your team to work:

- *cohesively,*
- *faster,*
- *transparently,*
- *and effectively*

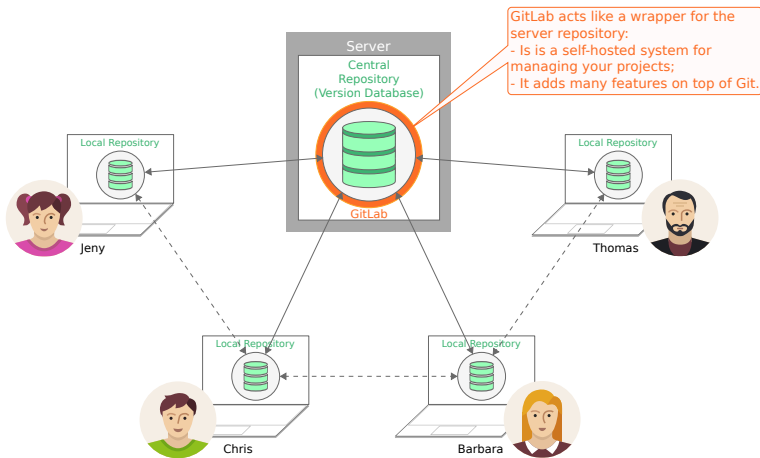
since the discussion of a new idea until taking that idea to production all the way through, from within the same platform.

— <https://docs.gitlab.com/ce/user/index.html>

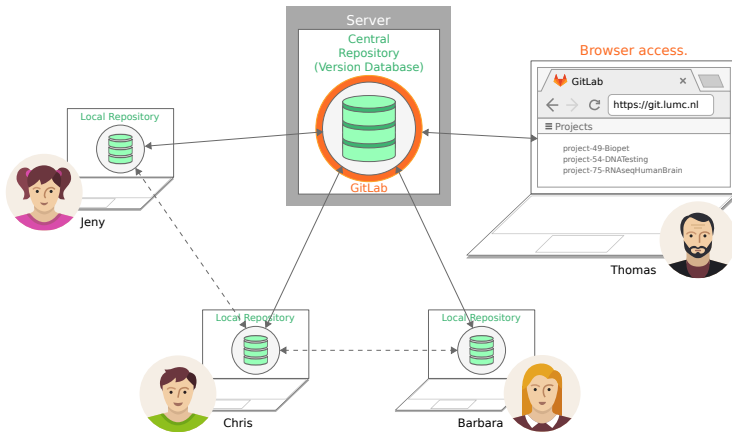
Overview: without GitLab



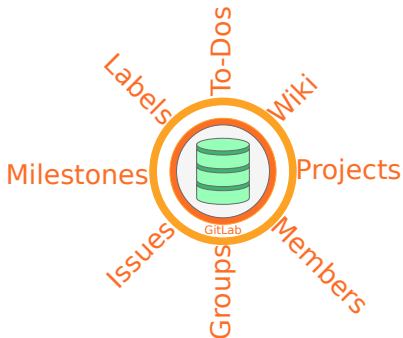
Overview: with GitLab I



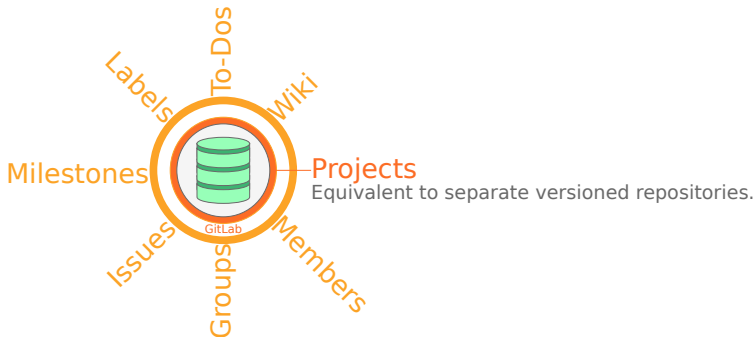
Overview: with GitLab II



Collaborative Environment



Collaborative Environment



Collaborative Environment



Collaborative Environment



Collaborative Environment



Issues
A manner through which changes are requested.

Collaborative Environment



Collaborative Environment

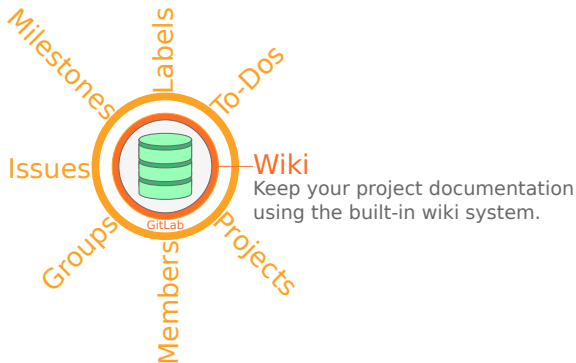


Collaborative Environment



To-Dos
See where you should spend your time,
where your team members need help,
where you need to take some action,
or what you need to keep an eye on.

Collaborative Environment



Alternatives and some differences

GitHub:

- Free for public repositories. Paid for private ones.
- Hosting provided for free.

GitLab:

- Requires self hosting - own installation.
- Basic features for free. Paid advanced ones.

More details, including *Bitbucket* and *Coding*, [here](#).

Projects and groups

- A GitLab **project** corresponds to a single Git repository.
- You can **group** projects and give users access to several projects at once.
- Every project has a **visibility level**:
 - A way of controlling who has **read** access to that project.
- 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.

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.

Users

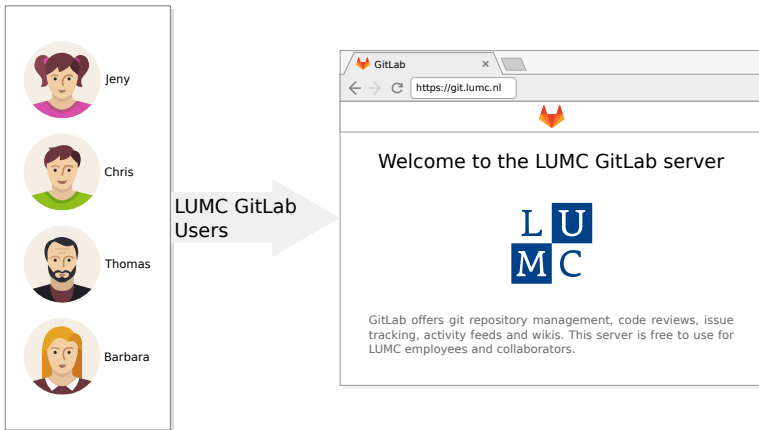
Actions:

- Create / import / manage projects.
- Create / manage groups.

Roles per project/group:

- Guest.
- Reporter.
- Developer.
- Maintainer.
- Owner.

Example

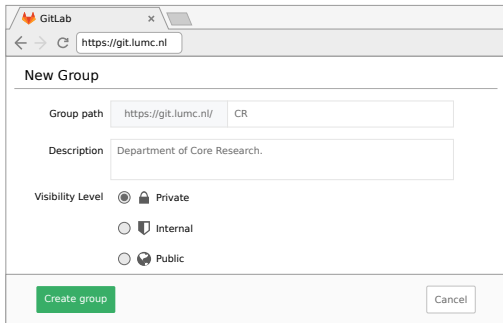


Disclaimer



In the following, not actual screenshots.

Example: group creation



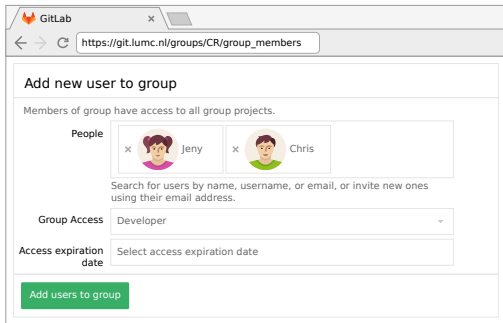
A screenshot of a web browser window showing the GitLab 'New Group' form. The browser's address bar displays 'https://git.lumc.nl'. The form is titled 'New Group' and contains the following fields and options:

- Group path:** A text input field containing 'https://git.lumc.nl/' and a dropdown menu set to 'CR'.
- Description:** A text input field containing 'Department of Core Research.'
- Visibility Level:** Three radio button options: 'Private' (selected), 'Internal', and 'Public'.
- Buttons:** A green 'Create group' button and a 'Cancel' button.



Creates department group.

Example: add users to group

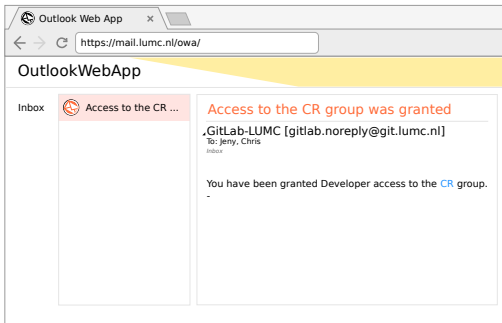


The screenshot shows a web browser window with the GitLab logo in the top left and the URL `https://git.lumc.nl/groups/CR/group_members` in the address bar. The main heading is "Add new user to group". Below this, a sub-header states "Members of group have access to all group projects." The "People" section contains two user cards, each with a small 'x' icon, a profile picture, and the name "Jeny" and "Chris" respectively. Below the cards is a search instruction: "Search for users by name, username, or email, or invite new ones using their email address." The "Group Access" section has a dropdown menu currently set to "Developer". The "Access expiration date" section has a text input field with the placeholder "Select access expiration date". At the bottom is a green button labeled "Add users to group".



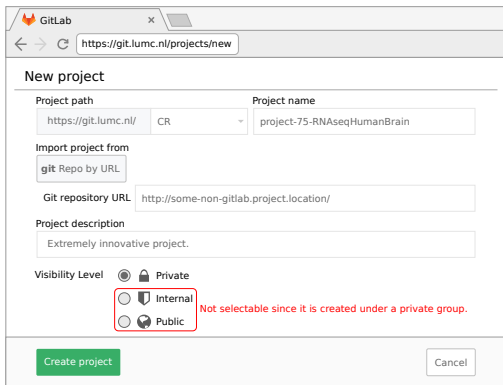
Adds colleagues to the department group.

Example: group access email confirmation



Receive confirmation emails.

Example: project creation







The screenshot shows the 'New project' form in the GitLab web interface. The browser address bar shows 'https://git.lumc.nl/projects/new'. The form includes fields for 'Project path' (https://git.lumc.nl/), 'Project name' (project-75-RNAseqHumanBrain), 'Import project from' (git Repo by URL), 'Git repository URL' (http://some-non-gitlab.project.location/), and 'Project description' (Extremely innovative project.). Under 'Visibility Level', the 'Private' radio button is selected. The 'Internal' and 'Public' radio buttons are disabled, indicated by a red box and a red text note: 'Not selectable since it is created under a private group.' At the bottom, there are 'Create project' and 'Cancel' buttons.



Creates GitLab project for "project-75-RNAseqHumanBrain".

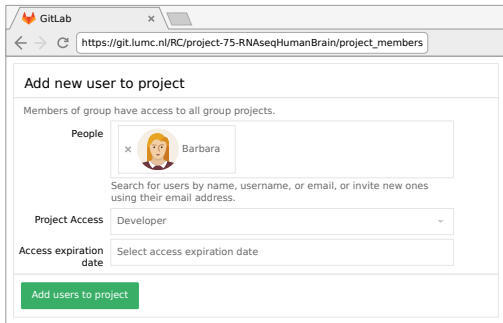
Example: current roles

	Group CR	Project project-75-RNAseqHumanBrain
 Thomas	Owner	Owner
 Jeny	Developer	Developer
 Chris	Developer	Developer
 Barbara	-	-

Inherited from the group

Neither group nor project access for Barbara.

Example: add users to project



The screenshot shows a web browser window with the GitLab logo in the top left. The address bar contains the URL `https://git.lumc.nl/RC/project-75-RNAseqHumanBrain/project_members`. The main content area is titled "Add new user to project". Below the title, it says "Members of group have access to all group projects." There is a section labeled "People" with a search bar containing "x" and "Barbara". Below this, it says "Search for users by name, username, or email, or invite new ones using their email address." There are two more fields: "Project Access" with a dropdown menu set to "Developer", and "Access expiration date" with a text input field containing "Select access expiration date". At the bottom, there is a green button labeled "Add users to project".



Adds Barbara to the project.

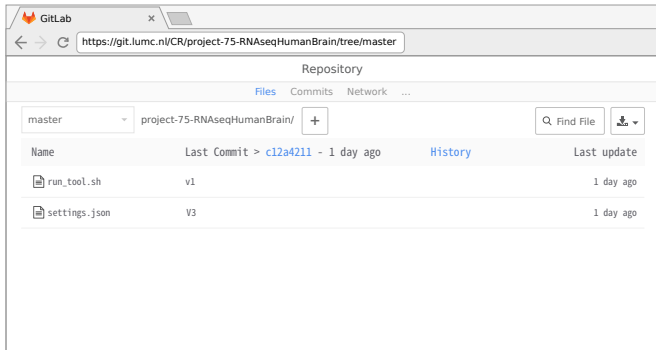
Example: roles updated

	Group CR	Project project-75-RNAseqHumanBrain
 Thomas	Owner	Owner
 Jeny	Developer	Developer
 Chris	Developer	Developer
 Barbara	-	Receives also an email with this information. Developer

Users and groups extras

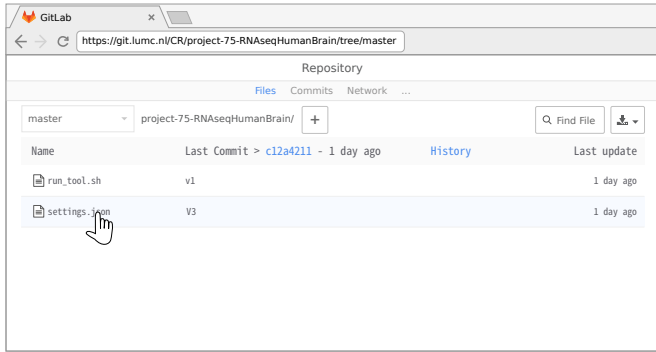
- You can import another project's users into your own project.
- You can request access to a project/group (if enabled).
- You can share your project with other groups.
- You can manage group/project members roles at any time.

Example: view project files on GitLab



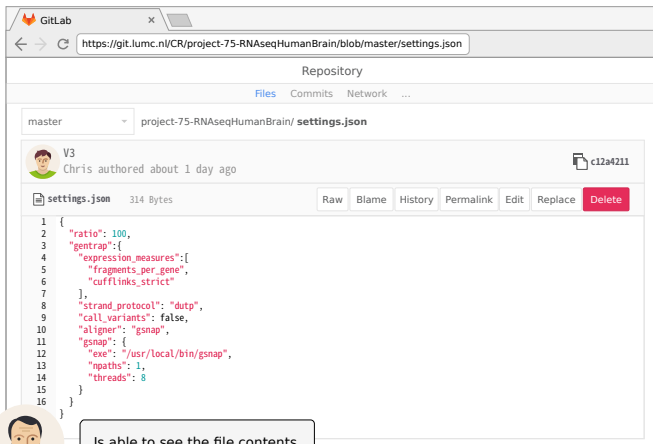
Browses the project files.

Example: view project files on GitLab

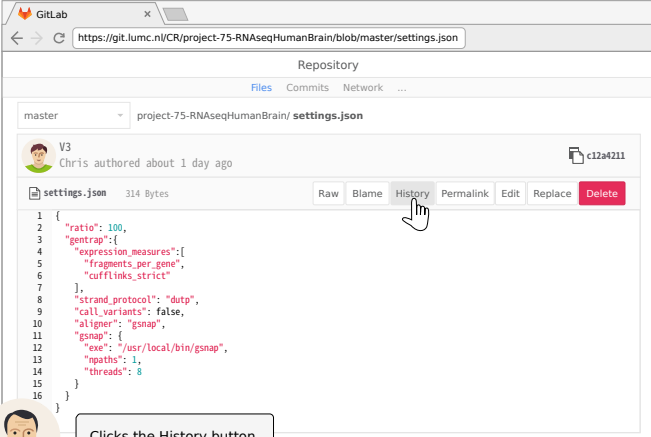


Clicks on a file.

Example: view project files on GitLab



Example: see file history



The screenshot shows the GitLab web interface for a repository named "project-75-RNAseqHumanBrain". The file "settings.json" is selected, and the "History" button is highlighted with a mouse cursor. The file content is displayed in a code editor, showing a JSON object with various configuration parameters. A callout box points to the "History" button with the text "Clicks the History button.".

Repository

Files Commits Network ...

master project-75-RNAseqHumanBrain/ settings.json

V3
Chris authored about 1 day ago

settings.json 314 Bytes

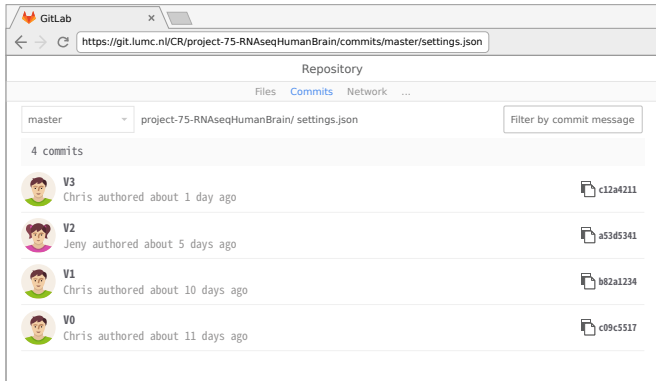
Raw Blame History Permalink Edit Replace Delete

```
1 {  
2   "ratio": 100,  
3   "gentrap": {  
4     "expression_measures": [  
5       "fragments_per_gene",  
6       "cufflinks_strict"  
7     ]  
8   },  
9   "strand_protocol": "dutp",  
10  "call_variants": false,  
11  "aligner": "gsnap",  
12  "gsnap": {  
13    "exe": "/usr/local/bin/gsnap",  
14    "npaths": 1,  
15    "threads": 8  
16  }  
}
```

Clicks the History button.

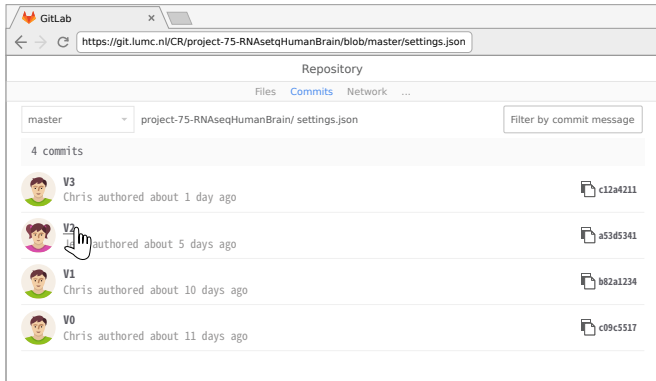
Thomas

Example: see file history



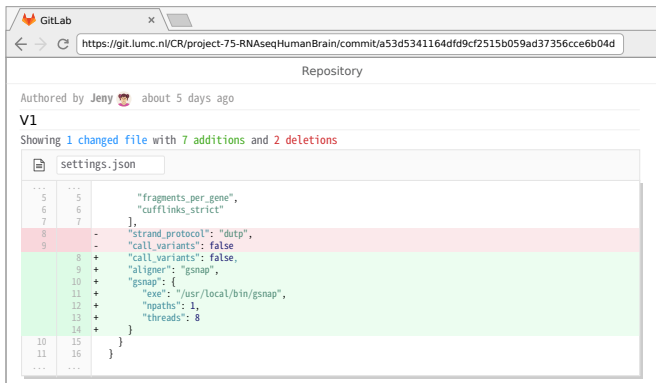
Clicks the History button.

Example: see differences



Clicks on the commit message.

Example: see differences



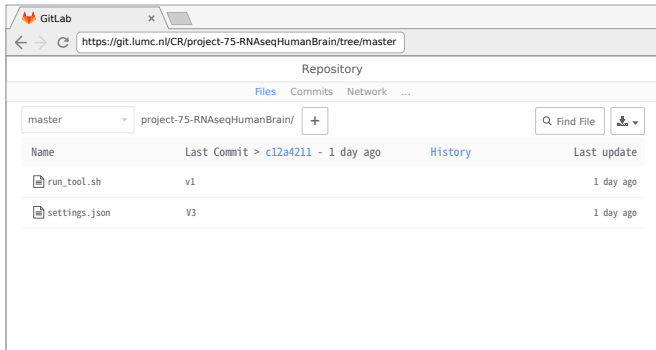
The screenshot shows a web browser window with the GitLab logo and a search bar. The URL is <https://git.lumc.nl/CR/project-75-RNAseqHumanBrain/commit/a53d5341164dfd9cf2515b059ad37356cce6b04d>. The page title is "Repository". Below the title, it says "Authored by Jeny about 5 days ago". The commit message is "V1". Below the message, it says "Showing 1 changed file with 7 additions and 2 deletions". The file "settings.json" is selected. The diff shows changes to the "gsnap" configuration. The "strand_protocol" and "call_variants" are unchanged. The "aligner" is changed from "gatk" to "gsnap". The "threads" are changed from 1 to 8.

```
... 5      "fragments_per_gene",
6      "cufflinks_strict"
7      },
8      - "strand_protocol": "dntp",
9      - "call_variants": false
10     + "call_variants": false,
11     + "aligner": "gsnap",
12     + "gsnap": {
13     +   "exe": "/usr/local/bin/gsnap",
14     +   "npaths": 1,
15     +   "threads": 8
16     + }
17   }
```



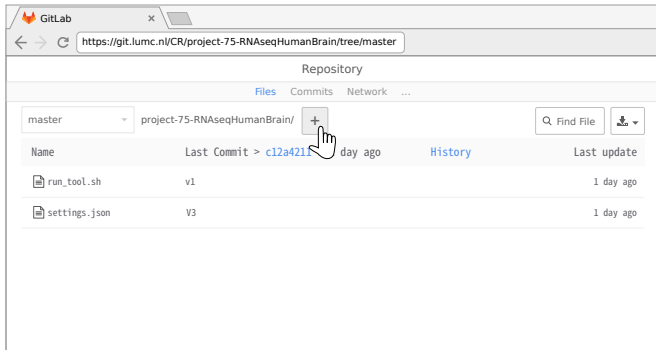
Is able to see the differences.

Example: upload existing file



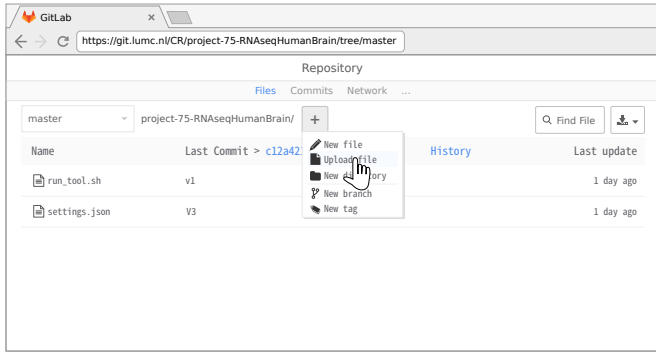
Browses the project files.

Example: upload existing file



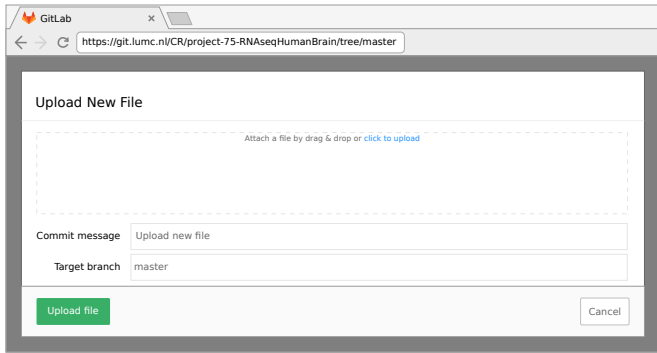
Starts the uploading file process.

Example: upload existing file



Starts the uploading file process.

Example: upload existing file

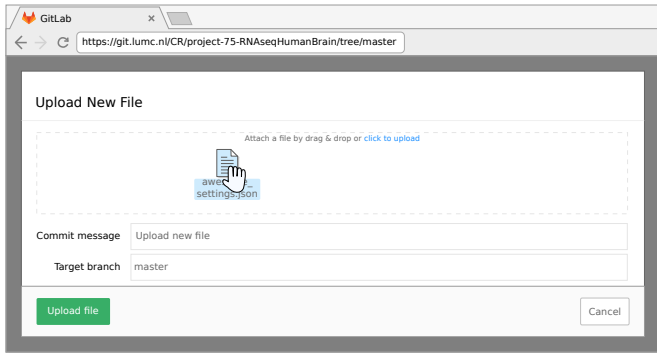


The screenshot shows a web browser window with the GitLab logo in the top left corner. The address bar displays the URL `https://git.lumc.nl/CR/project-75-RNAseqHumanBrain/tree/master`. The main content area is titled "Upload New File". Below the title is a large dashed rectangular box with the text "Attach a file by drag & drop or [click to upload](#)". Underneath this box are two input fields: "Commit message" with the text "Upload new file" and "Target branch" with the text "master". At the bottom left of the form is a green button labeled "Upload file", and at the bottom right is a button labeled "Cancel".



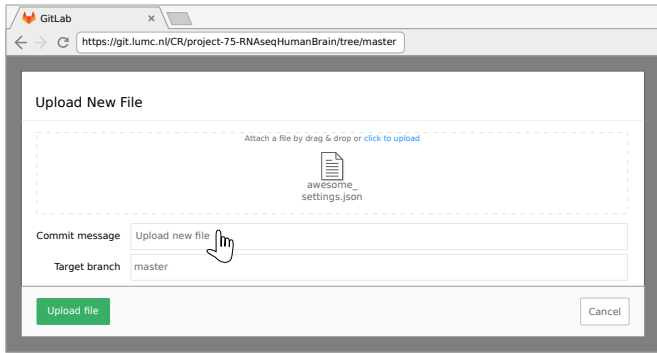
Starts the uploading file process.

Example: upload existing file



Drags the file.

Example: upload existing file



GitLab

https://git.lumc.nl/CR/project-75-RNAseqHumanBrain/tree/master

Upload New File

Attach a file by drag & drop or [click to upload](#)

awesome_settings.json

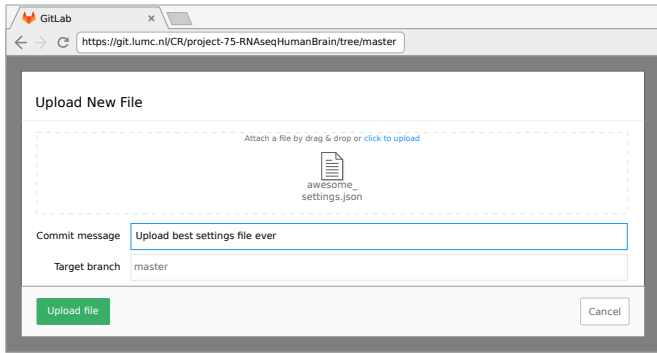
Commit message

Target branch



Changes the commit message.

Example: upload existing file



GitLab

https://git.lumc.nl/CR/project-75-RNAseqHumanBrain/tree/master

Upload New File

Attach a file by drag & drop or [click to upload](#)

awesome_settings.json

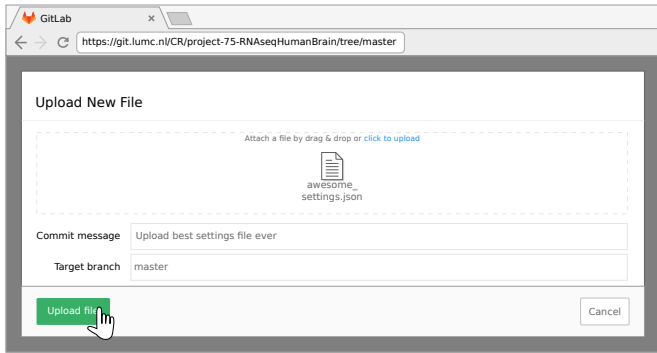
Commit message

Target branch



Changes the commit message.

Example: upload existing file



GitLab

https://git.lumc.nl/CR/project-75-RNAseqHumanBrain/tree/master

Upload New File

Attach a file by drag & drop or [click to upload](#)

awesome_settings.json

Commit message: Upload best settings file ever

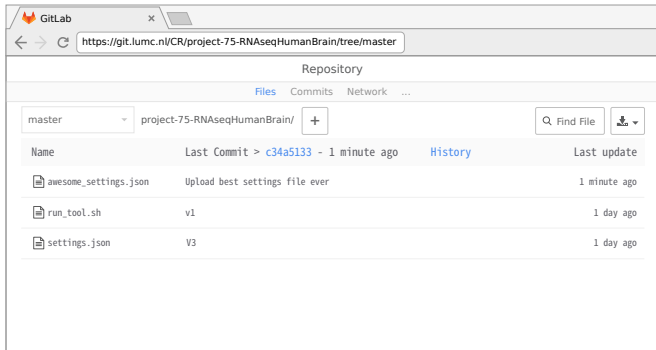
Target branch: master

Upload file Cancel



Presses the Upload file button.

Example: upload existing file



The new file appears in the repository.

Practical 1

Groups, Projects, and File Handling