

GIT - Globar Information Tracker

Gergely Fekete

2022.06.01 - Lab Seminar

Features - What git can do?

- security copies
- synchronisation between machines
- work together
- version tracking

merged_operetta_layout_20200215.csv
merged_operetta_layout_20200210.csv
merged_operetta_layout_20200204.csv
merged_operetta_layout_20200115.csv
merged_operetta_layout_20200113-tmp.xls
merged_operetta_layout_20200113.csv
merged_operetta_layout_20191217-tmp.xlsx
merged_operetta_layout_20191217.csv
merged_operetta_layout_20191216.csv
merged_operetta_layout_20191213.csv
merged_operetta_layout_20191129.csv

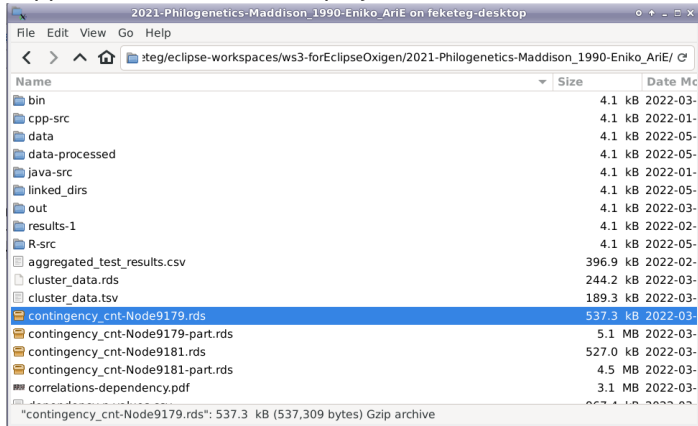




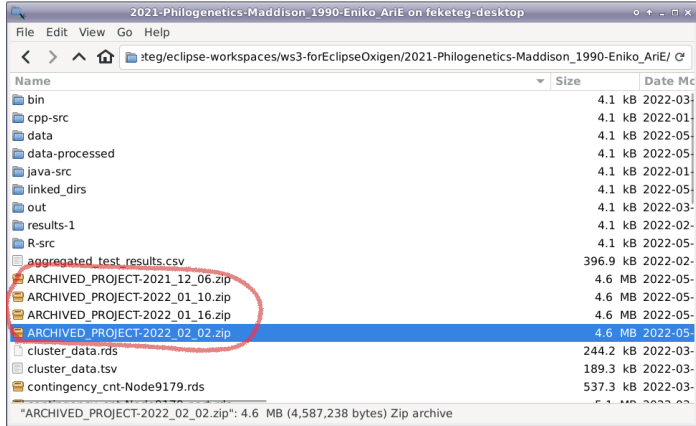
IDEA

Let's desing a version tracker

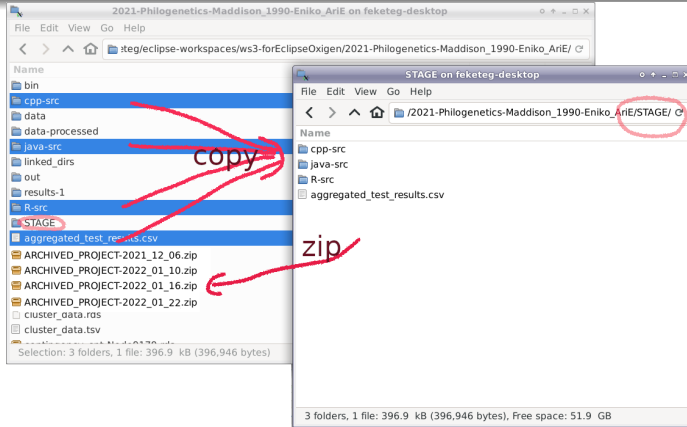
suppose that ... we have a project folder and we want to desing a version tracking workflow



The Snapshots

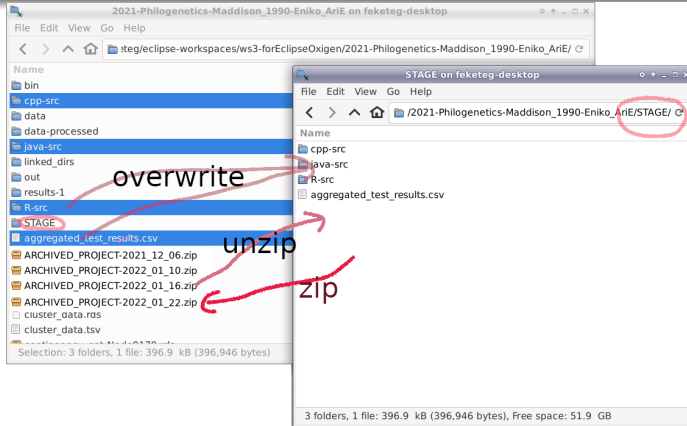


The Stage



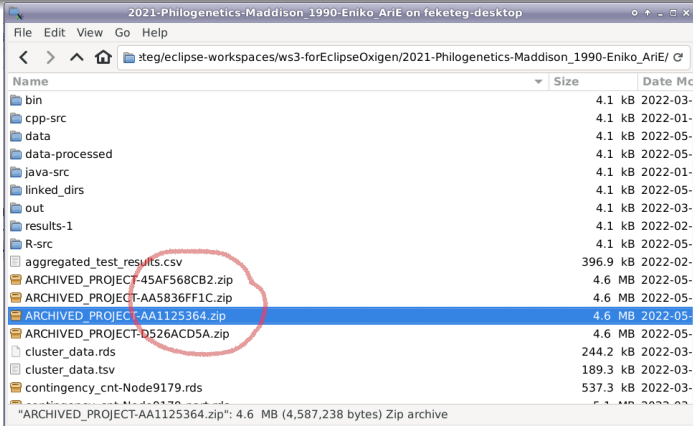
create an empty stage -> copy what you need to the stage -> zip the stage -> delete the stage

The Stage



create an empty stage -> extract there the prevois zip -> overwrite some files from the actual project folder -> zip the stage -> delete the stage

The Hash



Trivial namig method causes name conflicts.
Hash is a unique hexadecimal number.

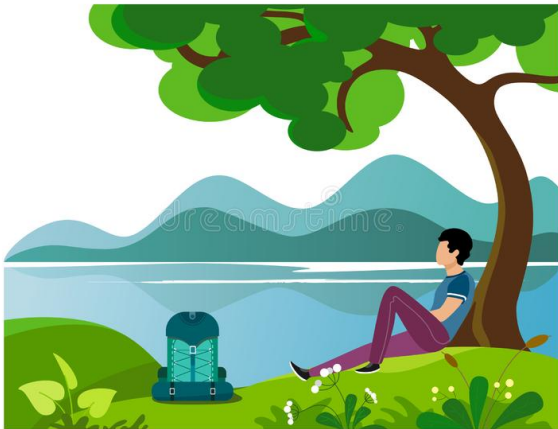
The Log and the Commit Message and The Commit

hash	author	date	message
4649520675e14bea	Fekete Gergo <fekger@gmail.com>	Tue Feb 25 18:55:34 2020 +0100	20190717 no operetta info
280dbd863f1d5274	Fekete Gergo <fekger@gmail.com>	Tue Feb 25 18:53:18 2020 +0100	new plates added + fix
01fc0c3f0eb2583d	Fekete Gergo <fekger@gmail.com>	Tue Feb 25 18:49:16 2020 +0100	fix in WP-bakers-plate1
5f3541ba26ced7d5	Fekete Gergo <fekger@gmail.com>	Tue Feb 25 17:57:12 2020 +0100	mol new columns clade,clinical isolate,control
07d502c30a444b68	Fekete Gergo <fekger@gmail.com>	Tue Feb 25 17:53:10 2020 +0100	introduce merged operetta layout file
f190947bc82aa557	Fekete Gergo <fekger@gmail.com>	Tue Feb 25 15:15:24 2020 +0100	fix formatting again
54a51ca076f664c1	Fekete Gergo <fekger@gmail.com>	Tue Feb 25 15:11:11 2020 +0100	fix formatting error
b8bfb3b9388b910e	Fekete Gergo <gergo@desktop1>	Tue Feb 25 15:03:48 2020 +0100	first files

The Log and the Commit Message and The Commit

A **commit** means creating a (1) snapshot and (2) note it to the log table and (3) a reference to the parent(s)

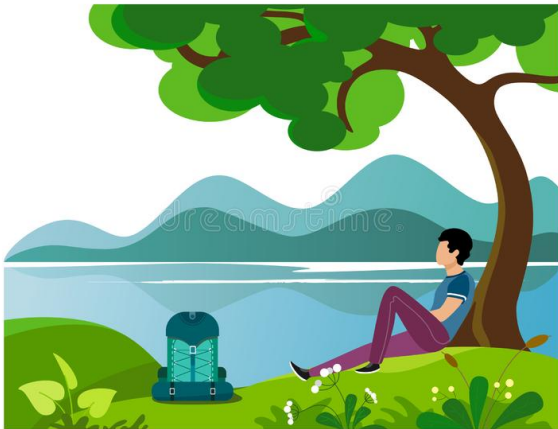
Summarise



- stage
- snapshot
- commit
- hash
- log
- **repository**

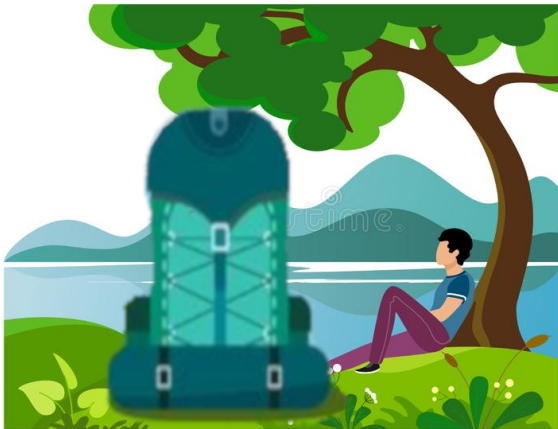
- `git status`
- `git add`
- `git commit`
- `git log`

Summarise



- stage
- snapshot
- commit
- hash
- log
- **repository**
- `git status`
- `git add`
- `git commit`
- `git log`

Resource Requirements



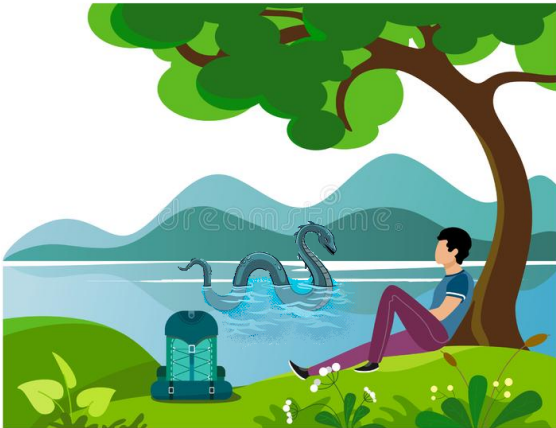
Does the lot of snapshots need a lot of space?

No

Is it fast?

Yes

The scary part



- how to upload on a server?
- how to download ?
- how to roll back ?
- how to manage the simultaneous work of more users ?
- branching?
- which is the current snapshot?

The not so scary part

how to roll back to an older version?

- `git checkout` extract the zip you want

how to upload on a server?

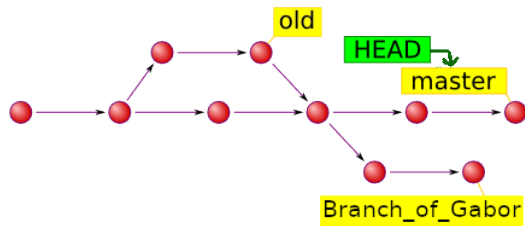
- `git push` simple upload all zip-s
- some of the zip-s has a copy on the server yet. The hash based name helps.
- commit first, then push

how to download from a server?

- simple download zip-s you don't have yet
- `git fetch` only download zips
- `git pull` download zips, and extract the **current**



That is the question:
which is the current snapshot?



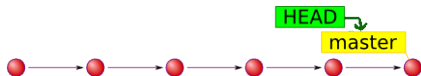
- red: commit objects : snapshots + ...
- yellow: Branches = pointers to commit objects
- green: the HEAD = pointer to pointer

That is the question:

which is the current snapshot?

- Different snapshot is current for you and for the co-workers.
- the HEAD points to the current *branch* and it points to the current snapshot
- how is it managed ?

Situation 1 : The simplest case



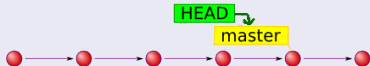
- red: commit objects
- yellow: Branches
- green: the HEAD

- If you work alone on only one machine with no rollbacks , no branches
- master branch allways points to the current commit
- master steps automaticly by commit
- HEAD always points to the master

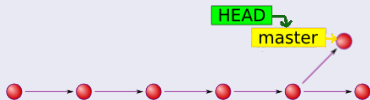
Situation 2 : Rollback



Something is wrong, we step back



The next commit creates a bifurcation; The bad commit turns to a dead end



Situation 3 : Working together

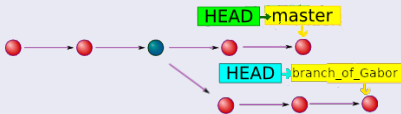
Gabor want to join to my work. We create a virtual copy of all files: a branch



We both are working on two separate machines, separate working copies

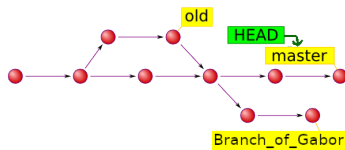


We synchronise : push, push, fetch, fetch

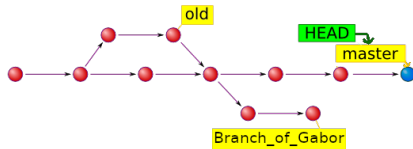


There is only one head per machine!

Branches and the HEAD pointer



Effect of `git commit`



- default branch name is master (can be renamed)
- branches are moveing pointers
- current branch steps automaticly by commit
- HEAD points to the current branch
- the new commit object will know which one is its parent

GIT

GIT is a tool to save folders , and handle versions

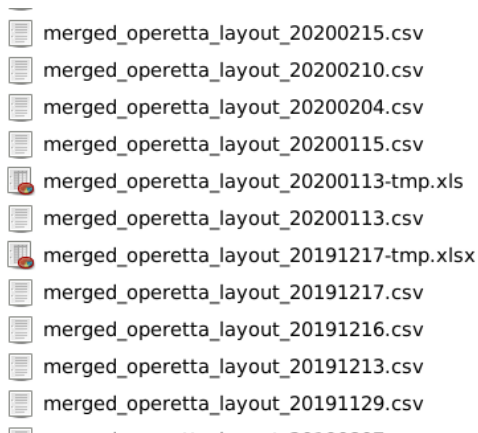
GitHub

GitHub is a webserver where you can save

- GIT works without GitHub
- GitHub is only one of the many git servers
- Git servers adds the abilty to share your staff

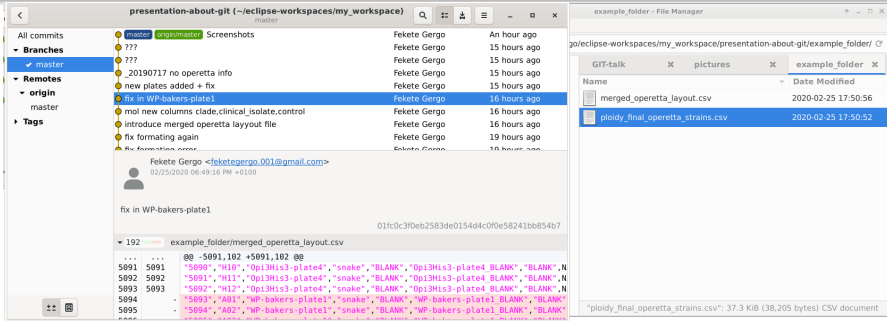
What are versions?

- Sometimes we edit a file continuously and want to keep its earlier versions

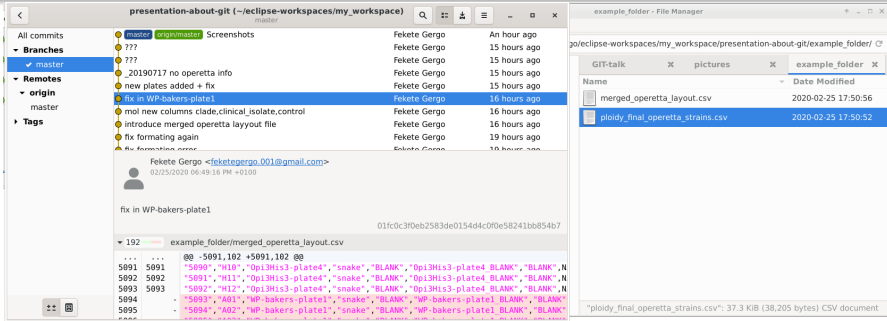


- the state of the art solution
 - have one file in the working directory
 - store the old versions 'hidden' in a **repository**
- What is a **repository**?
 - a simple subfolder
 - The folder name is `'.git'`.
 - It is a hidden folder
 - You have to start git to see the content

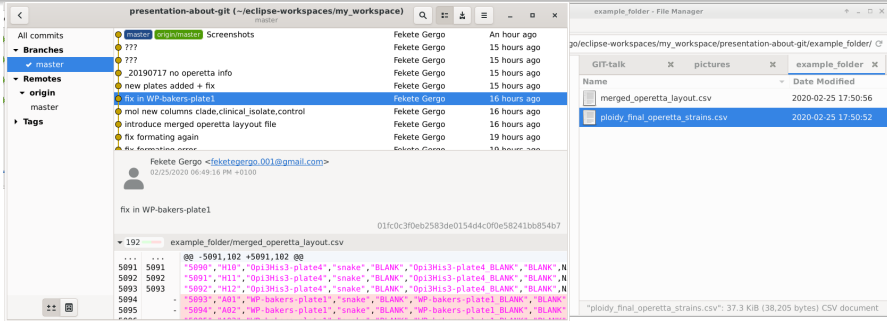
- the state of the art solution
 - have one file in the working directory
 - store the old versions 'hidden' in a **repository**
- What is a **repository**?
 - a simple subfolder
 - The folder name is `'.git'`.
 - It is a hidden folder
 - You have to start git to see the content



- normally you see only the 2 important files
- If you need the old versions you can turn on the repository browser.
- Each ball represents a previous version
- the term of the 'balls' is commit/revision/version
- You can delete files from the working directory. The repo keeps it.



- normally you see only the 2 important files
- If you need the old versions you can turn on the repository browser.
- Each ball represents a previous version
- the term of the 'balls' is commit/revision/version
- You can delete files from the working directory. The repo keeps it.



- normally you see only the 2 important files
- If you need the old versions you can turn on the repository browser.
- Each ball represents a previous version
- the term of the 'balls' is commit/revision/version
- You can delete files from the working directory. The repo keeps it.

Name	Size	Date Modified
ploddy_final_operetta_strains_190807_mod2-verGer-temp.csv	36.8 KiB	2019-12-20 22:45:22
ploddy_final_operetta_strains_190807_mod2.csv	37.0 KiB	2019-10-23 16:35:17
ploddy_final_operetta_strains_190807_mod.csv	37.3 KiB	2019-08-07 16:40:33
merged_operetta_layout_20200224.csv	3.7 MiB	2020-02-24 16:53:11
merged_operetta_layout_20200215.csv	3.7 MiB	2020-02-15 20:21:35
merged_operetta_layout_20200210.csv	3.6 MiB	2020-02-10 13:24:00
merged_operetta_layout_20200204.csv	3.6 MiB	2020-02-04 16:04:49
merged_operetta_layout_20200115.csv	3.2 MiB	2020-01-15 15:09:45
merged_operetta_layout_20200113-temp.xls	4.0 MiB	2020-01-14 09:14:32
merged_operetta_layout_20200113.csv	2.9 MiB	2020-01-13 18:07:00
merged_operetta_layout_20191217-temp.xls	1018.0 KiB	2019-12-18 10:53:08
merged_operetta_layout_20191217.csv	2.4 MiB	2019-12-17 12:02:21
merged_operetta_layout_20191216.csv	2.4 MiB	2019-12-16 13:11:23
merged_operetta_layout_20191213.csv	2.4 MiB	2019-12-13 13:30:20
merged_operetta_layout_20191129.csv	2.4 MiB	2019-11-29 16:36:46
merged_operetta_layout_20190807.csv	2.4 MiB	2019-08-07 17:49:06
merged_operetta_layout_20190805.csv	2.4 MiB	2019-08-05 13:36:59
merged_operetta_layout_20190803.csv	2.4 MiB	2019-08-03 14:56:55
merged_operetta_layout_20190724.csv	2.0 MiB	2019-07-24 14:36:08
merged_operetta_layout_20190717_no_operetta_info.csv	691.8 KiB	2019-07-17 14:18:48
merged_operetta_layout_20190215.csv	1.6 MiB	2019-02-15 17:50:50
merged_operetta_layout_20181122.csv	1.6 MiB	2018-11-22 17:44:42
merged_operetta_layout_20181108.csv	1.6 MiB	2018-11-10 10:49:19
merged_operetta_layout_20180911.csv	1.3 MiB	2018-09-11 13:30:37
cellnum_perplate_genotype_170616_v5_withK015.csv	56.7 KiB	2018-11-19 10:08:57
all_strains_morphology_ploidy.csv	18.4 MiB	2019-01-16 16:33:18
all_strains_morphology2.csv	16.0 MiB	2018-10-11 14:35:53
all_strains_morphology.csv	8.3 MiB	2018-08-28 13:42:32

29 items: 94.3 MiB (98,844,291 bytes). Free space: 14.1 TiB

- working with messy folders is slower and confusing
- it causes errors
- It is waste of time and money.

Back to the top

GIT

GIT is a tool to save folders , and handle versions

- now we know what are versions
- Let's see why to save forlders instead of files

Why to save folders?

Belive me! It is a result of 35 years of evoluton and desig.

Why to save forlders?

Imagine a project where are

- experimental layout file
- result files form a microscopoe

They belong together. It is nice to connect them.

- actually it does not save full forlder. You can select some files to save together.
- The principal concept is 'comit together what belongs together'

Why to save forlders?

Imagine a project where are

- experimental layout file
- result files form a microscopoe

They belong together. It is nice to connect them.

- actually it does not save full forlder. You can select some files to save together.
- The principal concept is 'comit together what belongs together'

Why to save folders?

Imagine a project where are

- experimental layout file
- result files form a microscope

They belong together. It is nice to connect them.

- actually it does not save full folder. You can select some files to save together.
- The principal concept is 'commit together what belongs together'

Back to the top again

GIT

GIT is a tool to save folders , and handle versions

- now we know what are versions
- we undarstand that commit many files together is clever
- What is the GIT tool?

Back to the top again

GIT

GIT is a tool to save folders , and handle versions

- now we know what are versions
- we undarstand that commit many files together is clever
- What is the GIT tool?

What is the GIT tool?

- actually git is not one tool: it is a protocol/standard
- There arae a lot of git program you can install.
- Linux and Mac have preinstalled git
- Rstudio contains a git client
- every IDE contains a git client (C , JAVA, pyton editors ...)
- gitg (grafikal UI - linux, windows, mac)
- git SMC (Window git client)
- Git Bash (Window git terminal)
- every IDE contains a git client (C , JAVA, pyton editors ...)

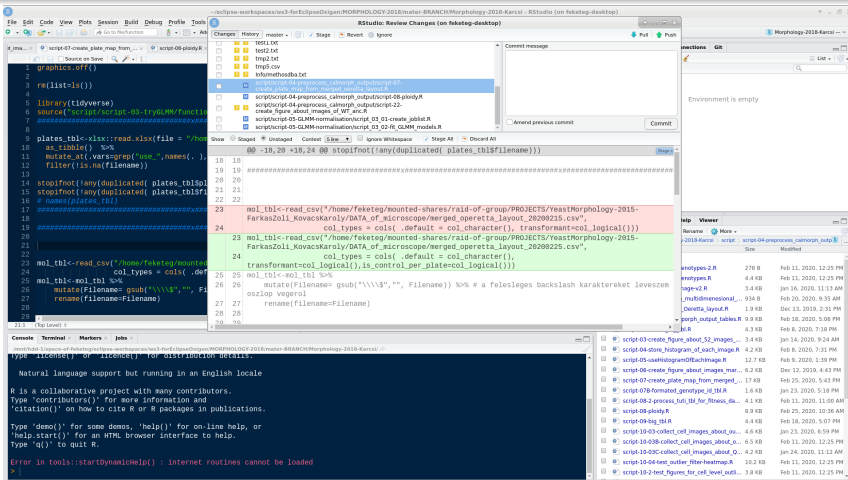
What is the GIT tool?

- actually git is not one tool: it is a protocol/standard
- There arae a lot of git program you can install.
- Linux and Mac have preinstalled git
- Rstudio contains a git client
- every IDE contains a git client (C , JAVA, pyton editors ...)
- gitg (grafikal UI - linux, windows, mac)
- git SMC (Window git client)
- Git Bash (Window git terminal)
- every IDE contains a git client (C , JAVA, pyton editors ...)

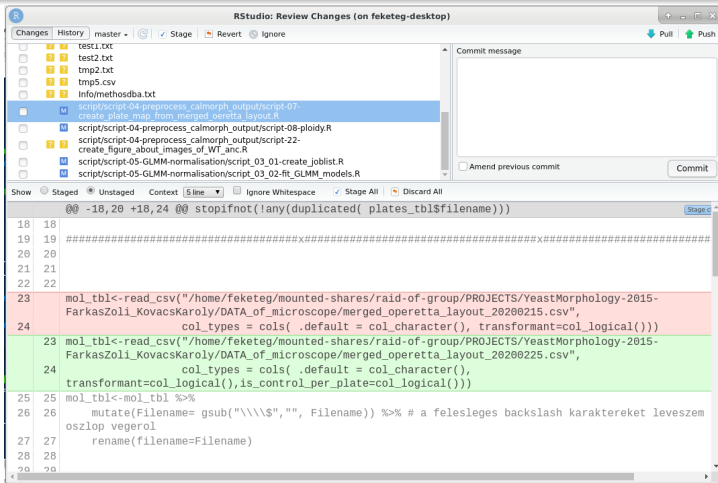
What is the GIT tool?

- actually git is not one tool: it is a protocol/standard
- There arae a lot of git program you can install.
- Linux and Mac have preinstalled git
- Rstudio contains a git client
- every IDE contains a git client (C , JAVA, pyton editors ...)
- gitg (grafikal UI - linux, windows, mac)
- git SMC (Window git client)
- Git Bash (Window git terminal)
- every IDE contains a git client (C , JAVA, pyton editors ...)

Let's see how to use it



Let's see how to use it



The screenshot shows the RStudio 'Review Changes' interface. The top panel lists files to be committed, including test files and R scripts. The bottom panel shows a diff view of a script file, with changes highlighted in red and green. The commit message field is empty, and the 'Commit' button is visible.

Changes: test1.txt, test2.txt, tmp2.txt, tmp5.csv, Info/methods.txt, script/script-04-preprocess_calmorph_output/script-07-create_plate_map_from_merged_operetta_layout.R, script/script-04-preprocess_calmorph_output/script-08-ploidy.R, script/script-04-preprocess_calmorph_output/script-22-create_figure_about_images_of_WT_anc.R, script/script-05-GLMM-normalisation/script_03_01-create_joblist.R, script/script-05-GLMM-normalisation/script_03_02-fit_GLMM_models.R

Commit message

@@ -18,20 +18,24 @@ stopifnot(!any(duplicated(plates_tbl\$filename)))

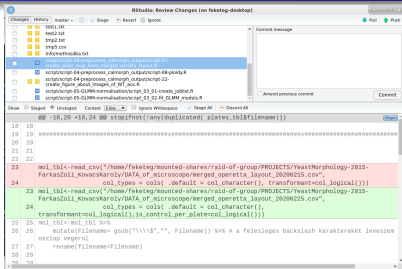
18 18
19 19 #####x#####x#####
20 20
21 21
22 22

23 mol_tbl<-read_csv("/home/feketeg/mounted-shares/raid-of-group/PROJECTS/YeastMorphology-2015-FarkasZoli_KovacsKaroly/DATA_of_microscope/merged_operetta_layout_20200215.csv",
24 col_types = cols(.default = col_character(), transformant=col_logical()))

23 mol_tbl<-read_csv("/home/feketeg/mounted-shares/raid-of-group/PROJECTS/YeastMorphology-2015-FarkasZoli_KovacsKaroly/DATA_of_microscope/merged_operetta_layout_20200225.csv",
24 col_types = cols(.default = col_character(),
transformant=col_logical(),is_control_per_plate=col_logical()))

25 25 mol_tbl<-mol_tbl %>%
26 26 mutate(Filename= gsub("\\\\", "", Filename)) %>% # a felesleges backslash karaktereket leveszem
oszlop vezerol
27 27 rename(filename=Filename)
28 28
29 29

Let's see how to use it



- select files to the stage
- unfulfilled/fulfilled files
- diff-s
- commit msg + button
- push/pull button

Terminology

- commit = save it (to the local repository)
- stage = files selected for save
- push = upload to the server
- pull = download from the server

Let's see how to use it - History

The screenshot displays the RStudio interface with the History tab active, showing a list of commits. The Environment tab is also visible, showing a list of objects.

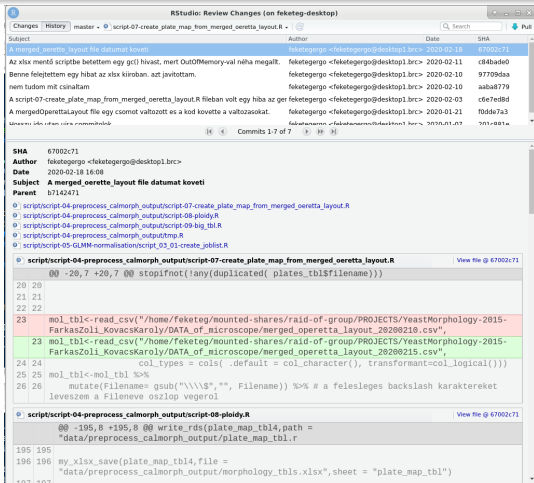
History Tab:

SHA	Author	Date	SHA
6700c71	feketegergo <feketegergo@desktop1.brc>	2020-02-18 16:08	6700c71
094ba50	feketegergo <feketegergo@desktop1.brc>	2020-02-11 16:08	094ba50
97709da	feketegergo <feketegergo@desktop1.brc>	2020-02-10 16:08	97709da
aab4779	feketegergo <feketegergo@desktop1.brc>	2020-02-10 16:08	aab4779
c6e7e8d	feketegergo <feketegergo@desktop1.brc>	2020-02-10 16:08	c6e7e8d
0f6de7a	feketegergo <feketegergo@desktop1.brc>	2020-02-10 16:08	0f6de7a
701e8b1	feketegergo <feketegergo@desktop1.brc>	2020-02-10 16:08	701e8b1

Environment Tab:

Name	Size	Modified
find_median_cell_of_genotypes-2.R	278 B	Feb 11, 2020, 12:25 PM
find_median_cell_of_genotypes.R	4.4 KB	Feb 11, 2020, 12:25 PM
function-find_x_cel_image-v2.R	3.4 KB	Jan 16, 2020, 11:13 AM
function-p_value_from_multidimensional...	934 B	Feb 20, 2020, 9:35 AM
kuka-script07-Merged_Oeretta_layout.R	1.9 KB	Dec 15, 2019, 2:31 PM
script-01-collect_calmorph_output_tables.R	9.9 KB	Feb 18, 2020, 5:06 PM
script-02-create_big_tbl.R	4.3 KB	Feb 8, 2020, 7:18 PM
script-03-create_figure_about_52_images...	3.4 KB	Jan 14, 2020, 9:24 AM
script-04-store_histogram_of_each_image.R	4.2 KB	Feb 8, 2020, 7:31 PM
script-05-useristogramOfEachImage.R	12.7 KB	Feb 9, 2020, 1:39 PM
script-06-create_figure_about_images_ma...	6.2 KB	Dec 12, 2019, 4:43 PM
script-07-create_plate_map_from_merged...	17 KB	Feb 25, 2020, 5:43 PM
script-07B-formatted_genotype_id_tbl.R	1.6 KB	Jan 23, 2020, 5:18 PM
script-08-2-process_tutl_tbl_for_fitness_da...	4.1 KB	Feb 11, 2020, 11:00 AM
script-08-plot.R	8.9 KB	Feb 25, 2020, 10:36 AM
script-09-big_tbl.R	4.4 KB	Feb 18, 2020, 5:07 PM
script-10-03-collect_cel_images_about...	4.6 KB	Jan 23, 2020, 6:59 PM
script-10-03B-collect_cel_images_about...	6.5 KB	Feb 11, 2020, 12:25 PM
script-10-03C-collect_cel_images_about...	4.2 KB	Jan 24, 2020, 11:12 AM
script-10-04-test_outlier_filter-heatmap.R	10.2 KB	Feb 11, 2020, 12:25 PM
script-10-2-test_figures_for_cel_level_outl...	3.8 KB	Feb 11, 2020, 12:25 PM

Let's see how to use it- History



The screenshot shows the RStudio 'Review Changes' window for the file 'script-07-create_plate_map_from_merged_oeretta_layout.R'. The window is divided into three main sections: a commit history table at the top, a commit details section in the middle, and a file diff view at the bottom.

Subject	Author	Date	SHA
A merged_oeretta_layout file datumat koveti	feketegergo <feketegergo@desktop1.brc>	2020-02-18	67002c71
Az xlsx mentő scriptbe betettem egy gc() hívast, mert OutOfMemory-val néha megállt.	feketegergo <feketegergo@desktop1.brc>	2020-02-11	c84bade0
Benne felejtettem egy hibát az xlsx kiíróban. azt javítottam.	feketegergo <feketegergo@desktop1.brc>	2020-02-10	97709d3aa
nem tudom mit csináltam	feketegergo <feketegergo@desktop1.brc>	2020-02-10	aaba8779
A script-07-create_plate_map_from_merged_oeretta_layout.R fileban volt egy hiba az ger feketegergo <feketegergo@desktop1.brc>	feketegergo <feketegergo@desktop1.brc>	2020-02-03	c6e7ed8d
A mergedOerettaLayout file egy csomót változott és a kod kovette a változasokat.	feketegergo <feketegergo@desktop1.brc>	2020-01-21	f0dde7a3

SHA: 67002c71
Author: feketegergo <feketegergo@desktop1.brc>
Date: 2020-02-18 16:08
Subject: A merged_oeretta_layout file datumat koveti
Parent: b7142471

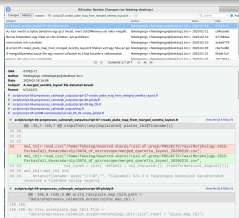
Diff view for 'script-04-preprocess_calmorph_output/script-07-create_plate_map_from_merged_oeretta_layout.R':

```
@@ -20,7 +20,7 @@ stopifnot(!any(duplicated( plates_tbl$filename)))
20 20
21 21
22 22
23 mol_tbl<-read_csv("~/home/feketeg/mounted-shares/raid-of-group/PROJETS/YeastMorphology-2015-
FarkasZoli_KovacsKaroly/DATA_of_microscope/merged_operetta_layout_20200218.csv",
23 mol_tbl<-read_csv("~/home/feketeg/mounted-shares/raid-of-group/PROJETS/YeastMorphology-2015-
FarkasZoli_KovacsKaroly/DATA_of_microscope/merged_operetta_layout_20200215.csv",
24 24
25 col_types = cols( .default = col_character(), transformant=col_logical())
25 25
26 mutate(Filename= gsub("\\\\\\\\$", "", Filename)) %>% # a felesleges backslash karaktereket
leveszem a Fileneve oszlop vegerol
```

Diff view for 'script-04-preprocess_calmorph_output/script-08-ploidy.R':

```
@@ -195,8 +195,8 @@ write_rds(plate_map_tbl4,path =
"data/preprocess_calmorph_output/plate_map_tbl.r
195 195
196 196 my_xlsx_save(plate_map_tbl4,file =
"data/preprocess_calmorph_output/morphology_tbls.xlsx",sheet = "plate_map_tbl")
```

Let's see how to use it- History



The screenshot shows the 'History' window in a code editor. It displays a list of commits, each with a commit ID, author, date, and a brief description. Below the list, the 'Diff' for the selected commit is shown, highlighting the changes made in the code. The diff shows changes to a file named 'src/main/java/org/example/HelloWorld.java', with lines added and removed. The commit message is 'Initial commit'.

- each row is a commit with
 - date
 - author
 - comment
 - commit ID
- list of files modified in the selected commit
- diffs : for each file it shows what is modified

Principles

- If you want to roll back
 - You have to commit first
 - git will replace the actual files with the old ones
 - You can not rollback only one file.
- You can go back to an old version and then return to the latest version
- If you want to go somewhere you have to tell the ID of the version
- the last commit called **HEAD**
- the 'go to' command is **checkout**

Example

```
git checkout 67002c71
```

```
git checkout HEAD
```


Principles

bad news

sometimes you need to type commands

GitHub

GitHub

GitHub is a webserver where you can save

- GIT saves things to the local repository on your machine
- GIT can upload everything to a remote git-server
- GitHub is one of the git servers

GitHub

GitHub is a webserver where you can save

- GIT saves things to the local repository on your machine
 - local repository needs to exist
- GIT can upload everything to a remote git-server
 - does not upload the actual files.
 - uploads the repository: all versions
- GitHub is one of the git servers
 - Bitbucket
 - Gitlab
 - We can have our own git server

GitHub

GitHub is a webserver where you can save

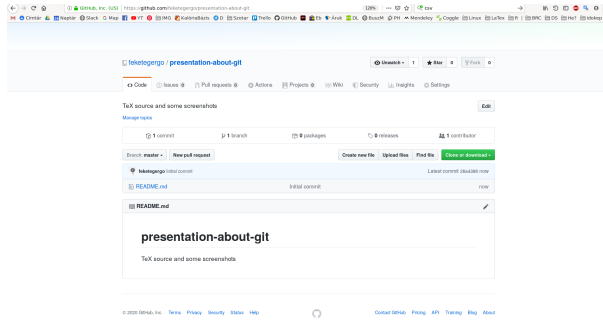
- GIT saves things to the local repository on your machine
 - local repository needs to exist
- GIT can upload everything to a remote git-server
 - does not upload the actual files.
 - uploads the repository: all versions
- GitHub is one of the git servers
 - Bitbucket
 - Gitlab
 - We can have our own git server

GitHub

GitHub is a webserver where you can save

- GIT saves things to the local repository on your machine
 - local repository needs to exist
- GIT can upload everything to a remote git-server
 - does not upload the actual files.
 - uploads the repository: all versions
- GitHub is one of the git servers
 - Bitbucket
 - Gitlab
 - We can hawe our own git server

- basic git servers just stores the repository
- GitHub provides additional Web interface



Start a GitHub Project

- go to github.com , register a user
- create a new project. Tick the 'initialised' checkbox.
- copy paste the url of the project
- start a terminal, go to the parent folder.
- use 'git clone <url>' command
- now you have an initialised local repository in the folder
- the **clone** command automaticly connected it to the GitHub repo.
- you can commit files.
- if you press the 'push' button or give the 'git push' command, then everything will be uploaded.
- If another user modified the files on the server the 'git pull' command download it

Start a GitHub Project

- go to github.com , register a user
- create a new project. Tick the 'initialised' checkbox.
- copy paste the url of the project
- start a terminal, go to the parent folder.
- use 'git clone <url>' command
- now you have an initialised local repository in the folder
- the **clone** command automaticly connected it to the GitHub repo.
- you can commit files.
- if you press the 'push' button or give the 'git push' command, then everything will be uploaded.
- If another user modified the files on the server the 'git pull' command download it

Tricky things start here

- If you try to upload a file, what is modified by another user. . . .
- It is called 'conflict'
- The operation 'merge' can fix the problem
- normally git merge it automatically
- pull first then push

Tricky things start here

- If you try to upload a file, what is modified by another user. . . .
- It is called 'conflict'
- The operation 'merge' can fix the problem
- normally git merge it automatically
- pull first then push

Git Data Transport Commands

<http://osteele.com>

