|  |  |
| --- | --- |
|  |  |
|  |  |

ENCORE: ENhancing COmputational REproducibility

A Bioinformatics Laboratory initiative

|  |  |
| --- | --- |
| **Date:** 30 March 2023  **Version:** 11 | **Prof. dr. A.H.C. van Kampen**  Bioinformatics Laboratory  Amsterdam UMC  Amsterdam, the Netherlands  [a.h.vankampen@amsterdamumc.nl](mailto:a.h.vankampen@amsterdamumc.nl)  <https://www.bioinformaticslaboratory.eu> |

## Contents

[Contents 2](#_Toc131053034)

[1 Introduction 3](#_Toc131053035)

[2 Access rights 3](#_Toc131053036)

[3 Organizing your project: the recipe 3](#_Toc131053037)

[4 Appendix 1. Using GitHub and Git 7](#_Toc131053038)

[4.1 GitHub 7](#_Toc131053039)

[4.2 The Git workflow 7](#_Toc131053040)

[4.3 Install git bash 8](#_Toc131053041)

[4.4 Git documentation 8](#_Toc131053042)

[4.5 GitHub and Git from scratch 8](#_Toc131053043)

[4.6 Further Git/GitHub notes 10](#_Toc131053044)

[4.6.1 Git pull vs Git fetch 10](#_Toc131053045)

[4.6.2 Use of branches 11](#_Toc131053046)

[4.6.3 Authorization 12](#_Toc131053047)

[4.6.4 Problems with ‘merging’ 13](#_Toc131053048)

[4.6.5 Remove all files in a github repository 13](#_Toc131053049)

[4.6.6 How to use a GitHub repo with Overleaf (an online collaborative LaTeX editor) 13](#_Toc131053050)

[4.6.7 How to use a GitHub repo with RStudio? 13](#_Toc131053051)

[4.6.8 Automatic creation of repository from within R/Rstudio 13](#_Toc131053052)

[4.7 Software Versioning 14](#_Toc131053053)

[4.8 VERSION HISTORY 14](#_Toc131053054)

[4.9 ACKNOWLEDGMENTS 14](#_Toc131053055)

# Introduction

The Bioinformatics Laboratory is in the process of changing its way of working to achieve a high level of organisation and reproducibility for all research and support projects. The use of GitHub and the File System Structure (FSS) are two interlinked approaches that we use in this context.

# Access rights

Every repository should be readable by all members from the Bioinformatics Laboratory. If you cannot read a repository then contact the owner of the repository. In specific cases, and after approval of the Bioinformatics Laboratory staff, we can give access to external researchers, reviewers, etc. to specific repositories.

**Providing access for external collaborators**

1. Go to the repository
2. Go to Settings
3. Go to Manage access
4. Button "Invite teams or people"

# Organizing your project: the recipe

1. **If you are new to GitHub then first read Appendix 1.**
   * The recipe below will take you step by step towards the creation of a new git repository and FSS directory for your project. There are other ways of doing this, but this will get you going.
2. **Create your project repository (repo) directly on the GitHub website**
   * <https://github.com/EDS-Bioinformatics-Laboratory>

* Click on the ‘New’ button
* Choose a repository name
  + - Use an informative name
    - Don’t put the date in the name
    - Don’t put a literature reference in the name
* Add a description. Note, the description should start with the name(s) of the repository owner
  + - Example: Antoine van Kampen, Barbera van Schaik --- pipeline for repertoire analyses
    - *Rationale:* The number of repositories may quickly grow. By starting with the owner name in the 'About' field we can directly see to who a project belongs.
* Make the repository Private
  + - *Note*: at a later stage we can decide to make a repository Public.
* Do not add a README file
* Do not add .gitignore
* Choose the GNU General Public License v3.0
  + - *Rationale*: this open-source license allows other people to use, extend, and modify our software. These changes will again become open source.
    - *Note*: although our software is open source this does not imply that we make our repositories public by default.
* Now ‘Create repository’

You should now see your new repository with only a single file (LICENSE) and a single branch (main).

* Click on the ‘wheel’ right of About and add keywords (see other repositories for examples to keep it consistent).
  + - Some standard keywords are: research, support, education.
    - Look at other repositories for used keywords to keep consistency
    - *Rationale:* having keywords helps to retrieve specific repositories.

Now you are ready to use this repository from your own computer/laptop using e.g., git bash.

1. **Create a directory ‘Reproducibility’ on your computer**
2. **Clone the GitHub repository ‘Reproducibility’ in this directory**
   * Start git bash in the Reproducibility directory just created
   * Clone the complete Reproducibility repository using the next command in git bash
     + git clone https://github.com/EDS-Bioinformatics-Laboratory/Reproducibility
   * The template FSS resides in the GitHub repository ‘Reproducibility’, which also contains some other useful files and information

* I encourage you to have a look at the latest version of the PowerPoint ReproducibleResearch\_vx.pptx.
* Remove the (hidden) directory .git in \Reproducibility\Reproducibility.
* The template FSS is found in \Reproducibility\Reproducibility\\_LATEST-ENCORE-TEMPLATE:
  + yyyymmdd\_ProjectName

1. **Initialize FSS on your computer**
   * Copy the FSS template yyyymmdd\_ProjectName to another location on your computer.
   * Rename the folder to have a descriptive name for your project (e.g., **20221019\_NetworkReconstruction**).
   * In the root directory of your FSS (e.g., 20221019\_NetworkReconstruction). You find the markdown file HELP\_FileSystemStructure.md. Read this file carefully. For reading/editing markdown files you can use, for example, Typora (www.typora.io; Windows, Mac) or Notepad++ (Windows).
2. **Start your FSS project**
   * Enter information in the **0\_PROJECT.md** file in the FSS root directory
     + This file should provide some basic details about the project as defined in this file.
     + This file is the entry point of a project (FSS)
     + Only for the *Contact* actual contact information (e.g., email, phone number) should be provided. For the other *Participants in the project* only their name, affiliation, and role should be given.
     + Use First and Last name and titles (e.g., prof, dr).
     + It contains a *Navigation Index* that by default is initialized as the FSS structure. This can be removed/neglected for the time being.
   * Note: the extension ‘md’ refers to markdown, which allows formatting of the information your provide. See <https://guides.github.com/features/mastering-markdown/>
   * Enter information in the **20221019\_NetworkReconstruction\Processing\README.md**.
     + This **README.md** file is the default README file that is used by GitHub (and will be synchronized with the repository in one of the next steps)
     + You can edit the README.md file to contain different information as from the root 0\_PROJECT.md file. For example, you probably want the README.md file to contain information about the content of the (sub)directories in your repository (i.e., Processing directory).
   * Edit gihub.txt
     + **20221019\_NetworkReconstruction\Processing\github.txt**
     + In this file enter at least the name of the github repository corresponding to your FSS. For example, **https://github.com/EDS-Bioinformatics-Laboratory/NetworkReconstruction**)
   * Copy and edit gitignore-template.txt
     + **20221019\_NetworkReconstruction\Processing\github.txt**
     + Copy the file gitignore-template.txt to .gitignore
     + This file is used by git to exclude files/directories to be synchronized with git.
     + Edit this file to ensure that no data and results are synchronized to your github repository. Only code, notebooks, code documentation, etc should be synchronized with the repository.
   * The directory **\Processing\0\_SoftwareEnvironment** contains general and project specific information about the software environment needed to run your code. Update this whenever needed.
   * Start your lab journal
     + **20221019\_NetworkReconstruction\ProjectDocumentation\labjournal.docx**
     + Enter project title, your name and starting date. Keep this labjournal up-to-date.
   * Run the script **fss2md.R** in the root of the FSS. This will generate an FSS navigation markdown and html file in the root directory with the content of the FSS. You can re-run this script at anytime to update the navigation file.
3. **Synchronize your FSS project with your GitHub repository**
   * Go to **20221019\_NetworkReconstruction\Processing**
   * Start git bash in this directory (in Windows: right mouse click, then select ‘Git bash here’)
   * Enter the following git commands (after each command you can use git status to check):
     + git init –initial-branch=main
     + git remote add origin [URL of repo]
       - URL of repo: as entered in github.txt, e.g.,
       - https://github.com/EDS-Bioinformatics-Laboratory/NetworkReconstruction]
     + git pull origin main
     + git add .
     + git commit -m "First sync" -m "First sync with github after setting up FSS"
     + git push -u origin main
   * Note: the command ‘git init’ has created the (hidden) directory .git in your Processing directory.
4. **Populate and document your FSS**

* Congratulations! You have now setup your GitHub repository and FSS. You have linked the FSS to the repository and performed a first synchronization.
* Now it is time to populate the remainder of your FSS
  + See **HELP\_FileSystemStructure.md** for further instructions
  + Add ProjectDocumentation (e.g., literature, powerpoint presentations, etc
  + Add Data to any of the three data locations in the FSS
  + Add code/notebooks in the processing directory
* You can remove files/directories that you don’t use. For example, the Sharing directory, two of the Data directories, LINK.txt, etc. You can always put them back when needed. However, the overall FSS structure should stay intact.
* If you add files or directories then also provide documentation in one of the README.md files (which are in principle present in each sub-directory), unless the content is self-explanatory.

1. **Keep your FSS and GitHub repository up-to-date**

From now on you can use the following git commands to keep your project directory and GitHub synchronized (preferably, you do this on a daily/weekly basis).

1. Go to the Processing directory
2. git pull <https://github.com/EDS-Bioinformatics-Laboratory/RNAseq>
   1. only if there were changes on the GitHub repo that are not yet in your local directory
3. git add .
4. git commit -m "short description" -m "long description"
5. git push

# Appendix 1. Using GitHub and Git

## GitHub

GitHub is used to manage your software. You have been invited to the EDS GitHub Repository by your supervisor. This is an **organization repository** on GitHub.com. Once you accept the invitation you will have access to all software currently being developed by the Bioinformatics Laboratory. Most of the software repositories are private (not public) and you are not allowed to redistribute any of the repositories since they may contain confidential information.

Location: <https://github.com/EDS-Bioinformatics-Laboratory>

## The Git workflow



## Install git bash

1. Download git bash: <https://git-scm.com/downloads>
2. Optionally you can download one of the GUI clients but using git bash (command line) will get you a better understanding of git.
   * GUI client: <https://desktop.github.com/>

Git bash allows you to access GitHub from your own computer/laptop thus you can start using git repositories (repo’s)

## Git documentation

There is a lot of documentation and tutorials on the internet.

* <https://guides.github.com/>
* <https://docs.github.com/en>
* Learn Github in 20 Minutes
  + Video: [https://www.youtube.com/watch?v=nhNq2kIvi9s](https://eur04.safelinks.protection.outlook.com/?url=https%3A%2F%2Fwww.youtube.com%2Fwatch%3Fv%3DnhNq2kIvi9s&data=04%7C01%7Ca.h.vankampen%40amsterdamumc.nl%7Cd96ee4a61ab84549ec0908d8c13bf687%7C68dfab1a11bb4cc6beb528d756984fb6%7C0%7C0%7C637471810587099881%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C1000&sdata=6kKHK%2BEG3ndg9pdaQ9gMKsJwVzxWnynI85mcO1PSR1M%3D&reserved=0)
  + Notes and more information: [https://www.notion.so/Introduction-to-GitHub-202af6f64bbd4299b15f238dcd09d2a7](https://eur04.safelinks.protection.outlook.com/?url=https%3A%2F%2Fwww.notion.so%2FIntroduction-to-GitHub-202af6f64bbd4299b15f238dcd09d2a7&data=04%7C01%7Ca.h.vankampen%40amsterdamumc.nl%7Cd96ee4a61ab84549ec0908d8c13bf687%7C68dfab1a11bb4cc6beb528d756984fb6%7C0%7C0%7C637471810587109836%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C1000&sdata=CAA2GHpC8NpWmAV7GWFALPcMK5dzVeZQX7W%2Bl5nssvE%3D&reserved=0)
* Learn Git In 15 Minutes
  + Video: [https://www.youtube.com/watch?v=USjZcfj8yxE](https://eur04.safelinks.protection.outlook.com/?url=https%3A%2F%2Fwww.youtube.com%2Fwatch%3Fv%3DUSjZcfj8yxE&data=04%7C01%7Ca.h.vankampen%40amsterdamumc.nl%7Cd96ee4a61ab84549ec0908d8c13bf687%7C68dfab1a11bb4cc6beb528d756984fb6%7C0%7C0%7C637471810587089933%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C1000&sdata=c2AXLVG0D%2FW0iGK%2FwabzBQCEAtqYCnBENl1ppQAIPkM%3D&reserved=0)
  + Notes and more information: [https://www.notion.so/Introduction-to-Git-ac396a0697704709a12b6a0e545db049](https://eur04.safelinks.protection.outlook.com/?url=https%3A%2F%2Fwww.notion.so%2FIntroduction-to-Git-ac396a0697704709a12b6a0e545db049&data=04%7C01%7Ca.h.vankampen%40amsterdamumc.nl%7Cd96ee4a61ab84549ec0908d8c13bf687%7C68dfab1a11bb4cc6beb528d756984fb6%7C0%7C0%7C637471810587099881%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C1000&sdata=aeaNib9RHxxoSG7Xa%2FrkbOgywT8u8QUak%2FTQAIh1SMI%3D&reserved=0)
* GitHub cheat sheet: <https://education.github.com/git-cheat-sheet-education.pdf>
* .gitignore
  + <https://git-scm.com/docs/gitignore>
  + <https://github.com/github/gitignore> (templates)

For some more information see also the following files in the Reproducibility repo (<https://github.com/EDS-Bioinformatics-Laboratory/Reproducibility>):

* GitHub\_Help.md
* GitHub\_BioLab-BestPractices.md
* GitHub\_Branch.md
* GitHub\_OpenIssues.md

## GitHub and Git from scratch

If you are completely new to Github and Git then you can follow the next steps to create your first repository.

1. Go to <https://github.com/> to create your own account
2. At a certain stage you will need a Fine-Grained personal access token to access your repositories. Read all about it: <https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token>
3. Create a new public or private repository on GitHub. Give it a name and short description but do not add any files (e.g., README.md, LICENSE, .gitignore) to it.
4. Once you created the repository you will see the Quick Setup screen that also shows the name of the repository that you created. This looks something like: [**https://github.com/YourUserName/test.git**](https://github.com/YourUserName/test.git)
5. Next create a directory on your computer that should mirror your repository locally, and step into it, and add a markdown file README.md.
6. Start Git bash in this directory
7. Create your local repository

* Git init –initial-branch main

1. Add Git credentials for this specific repository (or use the -global option to do this for all current/future repositories)
   * git config credential.helper manager-core
   * git config user.name YourUserName
   * git config user.email YourEmail
   * The next time you will commit/push to the repository for which you added the credentials, Git will ask you for the credentials for that particular remote server if it is unable to find the username and password already stored.
2. Next give the following commands
   * git remote add origin [**https://github.com/YourUserName/test.git**](https://github.com/YourUserName/test.git)
   * git remote -v #check, or use git remote set-ulr [ulr.git] to change
   * git add .
   * git commit -m 'First sync' -m ‘This is the first syncrhonization of my local repositories’
   * git push --set-upstream origin main
3. Note: if you get a ‘fatal error’ in step 9e then it is likely that something went wrong with the authentication.
4. Go back to GibHub in your webbrowser and select your Repository. You will see that the README.md file is added and its contents is shown by default on the main page.
5. Click on the ‘Cog’ icon to change your description, add topics (keywords), and add your (personal) website.
6. Next, select ‘Add file’ and ‘Create new file’. Type ‘LICENSE’ as the file name. This will activate a button on the right part of the screen where you can select a License template. Select one, Review and Submit, and (don’t forget) to Commit at the bottom of the screen (select ‘commit directly to the main branch’).
7. Now we need to synchronize these changes with your local repository:
   * git pull
8. This is basically it. The next time you add files to your local directory you only have to give the following commands to update your remote github repository
   * git add .
   * git commit -m ‘Made a change’
   * git push

## Further Git/GitHub notes

### Git pull vs Git fetch



**Git fetch vs Git pull**.

git fetch is similar to git pull but doesn't merge. i.e. it fetches remote updates (refs and objects) but your local stays the same (i.e. origin/master gets updated but master stays the same). git pull pulls down from a remote and instantly merges.

To check for differences between your remote repository and the local working copy:

git fetch

git diff main origin/main

If you are happy with the changes then you can merge with git merge or just do a git pull

See also: <https://stackoverflow.com/questions/292357/what-is-the-difference-between-git-pull-and-git-fetch>

### Use of branches

Note: the use of the Master branch is discouraged. For some more background about this see

* <https://medium.datadriveninvestor.com/why-githubs-change-from-master-to-main-is-not-the-solution-a3ac38cc48dd>
* <https://stevenmortimer.com/5-steps-to-change-github-default-branch-from-master-to-main/>

The default branch in git is the "main" branch. Common practice is that this is the stable code. When you would like to develop a new feature, fix a bug, etc. you can make use of branches. This code will live next to your stable version and once you are satisfied with the changes you have made (and when you have stable code again) you can merge the newly developed feature into your master branch.

#### Overview of the branches on your workstation

git branch - will show you the list of branches. There is an asterix in front of the active branch on your workstation.

#### Create a new branch

When you have several branches you probably would like to create a new branch with the 'main' branch as a starting point most of the time. First switch to the 'main' branch and then create a new branch.

git checkout main - switch to the main branch

git branch devel-some-feature - a new branch will be created with the name 'devel-some-feature'

Note that you are not in this branch yet when you create a new one. Switch to this branch with:

git checkout devel-some-feature - go to the new branch

You can start developing. Even when you are not ready yet you can commit and push your code to the repository on Github because this branch is separate from your stable main branch.

git commit -m 'some description' - make a snapshot of your changes, it will be logged with a git version

git push origin devel-some-feature - send your snapshot to Github

#### Merge the new code into the master branch

In the case that you are satisfied with the changes/new feature and you have tested whether the code works as it should you can merge it in the master branch. The order is as follows: first you switch to the main branch, then you merge the new developments into the master branch.

git checkout main - switch to the main branch

git merge devel-some-feature - merge the new feature into your main branch

Git will check whether there is conflicting code and notify you when this is the case. When that happens the merging process will stop and you will get the opportunity to resolve this. As soon as you are happy and everything works you can commit and push all the changes in the main branch.

git commit -m 'describe the changes' - make a snapshot of your changes on your workstation

git push origin main - synchronize with Github

In the case that there are no conflicts git will just merge the changes. It is still good to test your code again and then push it to Github.

Resolving conflicts: [www.atlassian.com/git/tutorials/using-branches](http://www.atlassian.com/git/tutorials/using-branches)

Tips for collaboration and best practices: [www.atlassian.com/git/tutorials](http://www.atlassian.com/git/tutorials)

#### Using branches for support

I tested the branch function for a support project. Normally I run stable code and, in that case, I only need to store the version of the stable code and the parameters that were used. However, I also use Python notebooks where I make changes in the code. E.g., a file can be comma-delimited instead of tab-delimited. Column names might be different, so then I need to change the code as well.

I created a new branch, from the master branch (well... actually... I made a mistake. I started with one of my devel branches instead of the stable main branch), with the name "runXXX-20201005-maria-reseda20201006". After I did all the analysis, I committed all the changes and pushed it to GitHub. I will probably never use this branch again, but now I have a record of the changes that I made. For that reason, I will not merge this branch into the master branch! I also included a "README-analysis.md" file in this branch where I tried to record as much as possible what I did. A sort of lab journal basically.

Link to this example: https://github.com/EDS-Bioinformatics-Laboratory/reseda/tree/runXXX-20201005-maria-reseda20201006

#### Moving master to main

<https://www.r-bloggers.com/2020/07/5-steps-to-change-github-default-branch-from-master-to-main/>

git branch -m master main

git push -u origin main

Sometime there might be a ‘mixup’ of branch names (main vs master). If so, these can be resolved with the following commands:

* git rm –cached -r .
* git branch main
* git checkout main
* git merge master
* git push origin main
* if necessary: git push origin --delete master

### Authorization

<https://github.blog/2020-12-15-token-authentication-requirements-for-git-operations/>

<https://docs.github.com/en/github/authenticating-to-github/creating-a-personal-access-token>

### Problems with ‘merging’

git mergetools

### Remove all files in a github repository

cd /tmp #make temporary directory

git clone /your/local/rep # make a temp copy

cd rep

git rm -r \* # delete everything

git status # everything but those copied will be removed

git commit -a -m 'deleting stuff'

git push

### How to use a GitHub repo with Overleaf (an online collaborative LaTeX editor)

See <https://www.overleaf.com/learn/how-to/How_do_I_connect_an_Overleaf_project_with_a_repo_on_GitHub,_GitLab_or_BitBucket%3F>.

* Remark 1: For the moment I (Perry Moerland) only tried this with my personal GitHub account, since Overleaf asks for permission to read/write \_all\_ GitHub repositories even the private ones, see https://www.scivision.dev/overleaf-with-github/. Probably this is something we may not want for the entire Bioinformatics Laboratory organization.
* Remark 2: This works with my (Perry Moerland) Overleaf account but it seems that this feature is not included in more recent Overleaf accounts, see <https://www.overleaf.com/learn/how-to/Working_Offline_in_Overleaf>. Curious to know if that is indeed the case.

### How to use a GitHub repo with RStudio?

See, for example, <https://happygitwithr.com/rstudio-git-github.html>. Many other useful tips regarding R, RStudio and GitHub can be found here as well

### Automatic creation of repository from within R/Rstudio

Aldo Jongejan has a R script that creates a new repository for you when working from R/RStudio. You wouldn’t have to go to the GitHub website first to create a new repository.

https://github.com/aldojongejan/R\_tests.git - zie getFSS.r

Example in R:

# Start new analysis

source("getFSS.r")

startNewRepo("20210119\_FleurPeters\_CLL\_Dasatinib", analysisDir = "Dasatinib")

## Software Versioning

To keep track of different versions (and releases) of the software you can use a versioning scheme. For more information see:

* Versioning: https://en.wikipedia.org/wiki/Software\_versioning
* Semantic versioning: https://semver.org/
* How to manage version numbers in git: https://stackoverflow.com/questions/37814286/how-to-manage-the-version-number-in-git

Rationale: In a later stage we can use these version numbers to connect to data, results, and documentation that are not part of the git repository.

## DOCUMENT VERSION HISTORY

**8 March 2023**

* Added the section ‘VERSION HISTORY’
* Added the section “GitHub and Git from scratch”.

**29 March 2023**

* Added section ‘Git fetch vs Git pull
* Changed of title page (using ENCORE for the time being)
* Started further changes/improvements to the document to accompany the publication about this initiative

## ACKNOWLEDGMENTS

ENCORE is an initiative of the Bioinformatics Laboratory ([www.bioinformaticslaboratory.eu](http://www.bioinformaticslaboratory.eu)) with contributions from many group members:

* Prof. dr. Antoine van Kampen
* Dr. Perry Moerland
* Dr. Aldo Jongejan
* Dr. Adrie Dane
* Barbera van Schaik
* Eric Wever
* Dasha Balashova
* Rodrigo Garcia Valiente
* Danial Lashgari
* Utkarsh Mahamune