

Introduction to Git & GitHub

Tuur Muyldermans: tuur.muyldermans@vib.be

James Collier: james.collier@vib.be

Objectives

- Understand how Git & GitHub works and how they interact
- Create repository for a project (and best practices)
- Apply the routine usage to version control a project
- Make branches and merge them
- Collaborate on someone else's repository
- Understand what I'm saying

Schedule

- 9:30 – 10:30 : Introduction & configurations
- 10:30 – 11:45 : Routine usage
- 11:45 – 12:00 : History & status
- 13:00 – 14:00 : Working with branches
- 14:45 – 15:30 : Collaborating in GitHub (forks)
- 15:45 – 16:00 : Ignoring files
- 16:00 – 16:30 : Version controlling in practice (Rstudio & questions)

General remarks

- Questions? Shoot!
 - ▶ Related to what I'm explaining: live
 - ▶ Extra question: in chat
- Exercises:
 - ▶ Poll: yes or no
- Too fast or too slow? Let me know!
- Continuously build on the same exercise, let me know if you have an **error** so we can fix it.
 - ▶ **Most likely it is relevant for everyone!**

Materials

- Slides in your mailbox
- <https://material.bits.vib.be/>
- Repositories that we will/might use:
 - ▶ <https://github.com/vibbits/introduction-github>
 - ▶ <https://github.com/vibbits/fork-repository>
- Terminology: https://en.wikipedia.org/wiki/Version_control

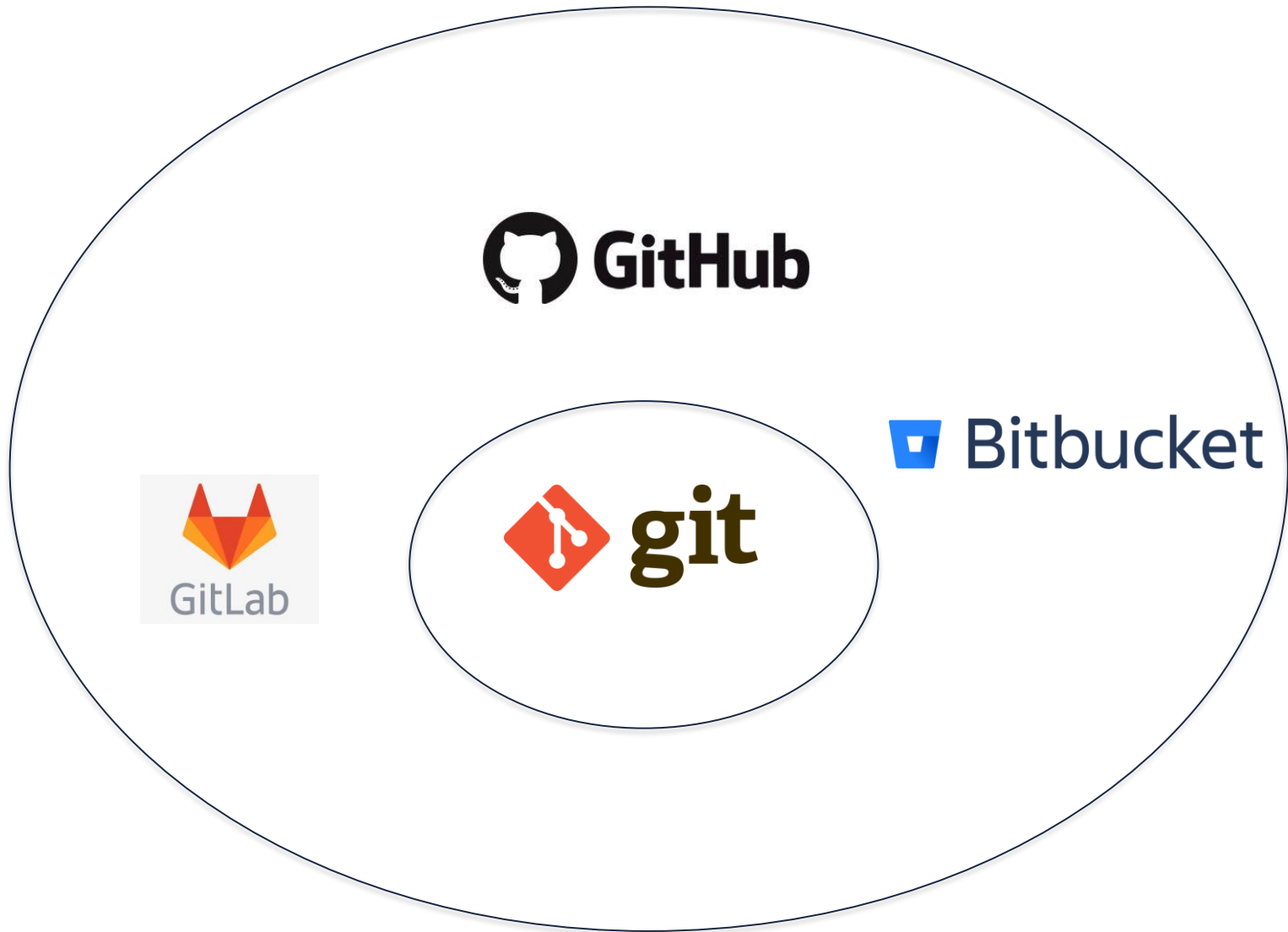
Installations

- Install Git: <https://git-scm.com/downloads>
- Make an account on [GitHub](#)
- Extra but not necessary: Rstudio at the end

1. Introduction

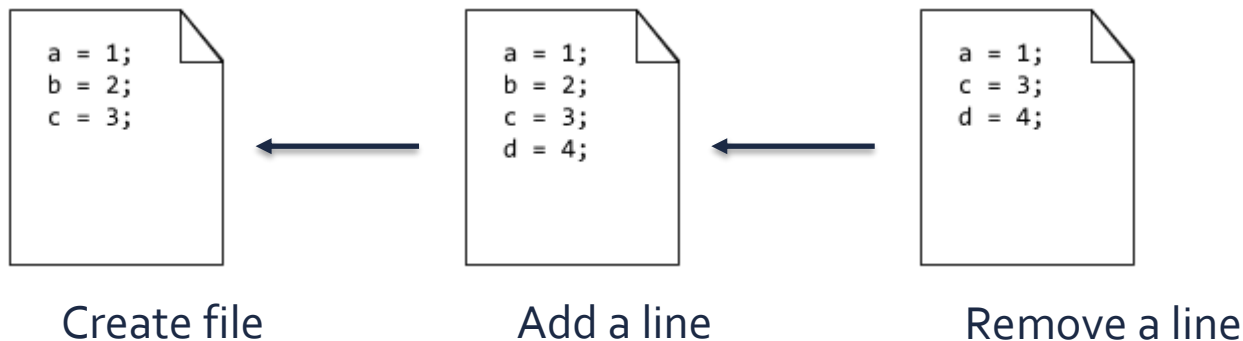
- What is Git & GitHub
- Why should I care about version controlling?
- Conceptual areas





1. Introduction

- What is Git used for?
 - ▶ Keep track of changes to your code

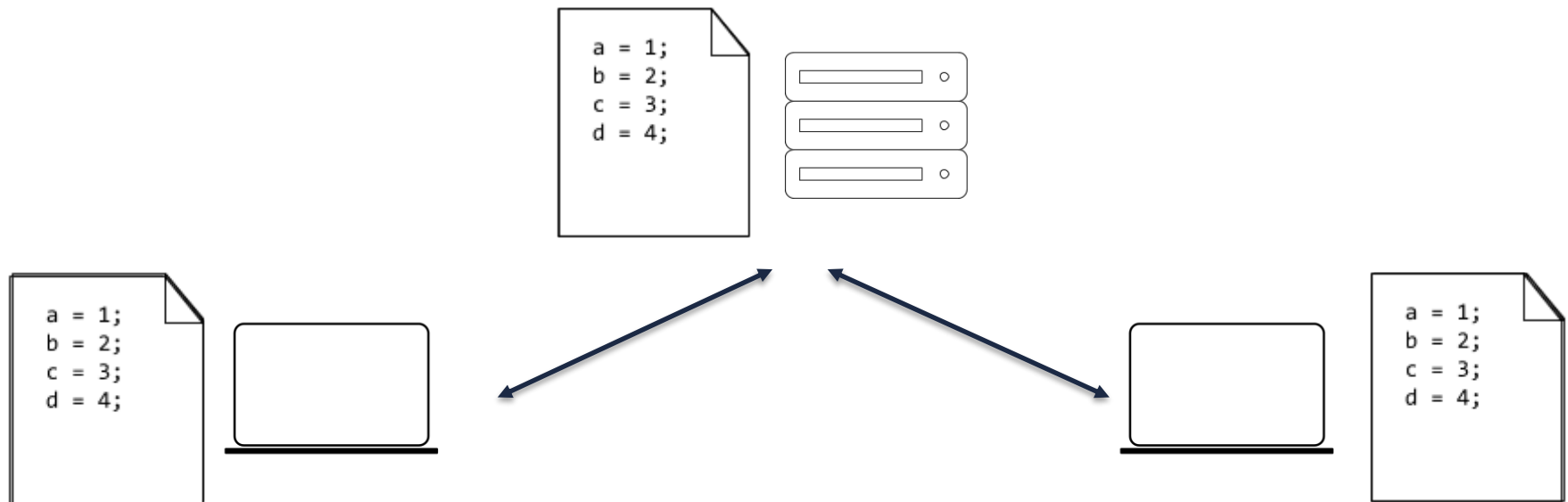


1. Introduction

- What is Git used for?
 - ▶ Keep track of changes to your code
 - ▶ Synchronize code between different people

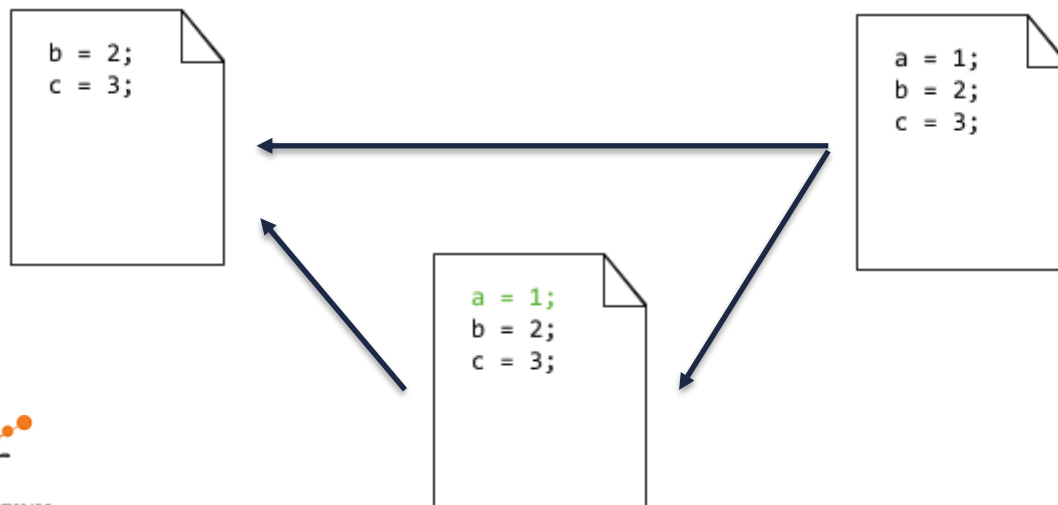
1. Introduction

- What is Git used for?
 - ▶ Keep track of changes to your code
 - ▶ Synchronize code between different people

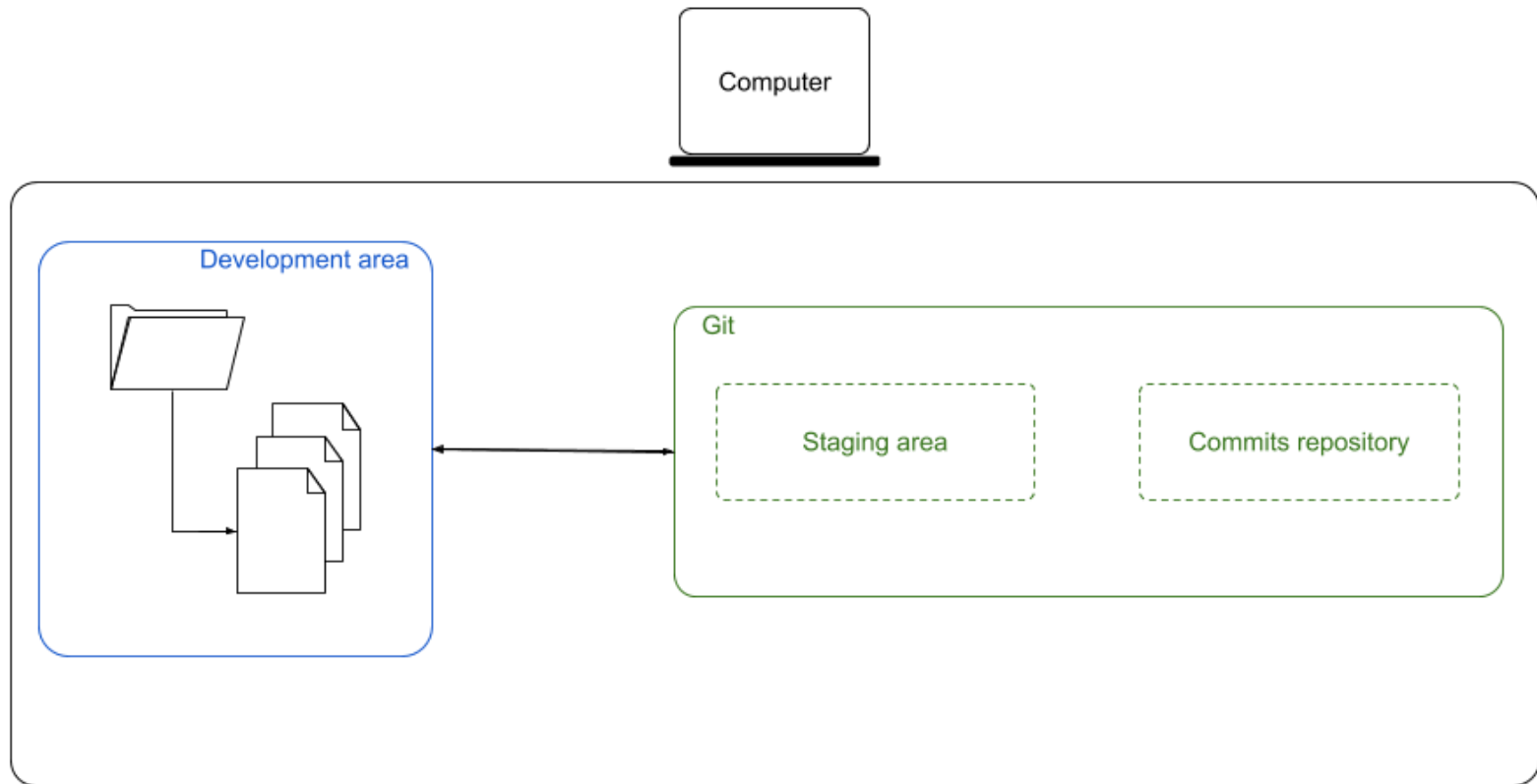


1. Introduction

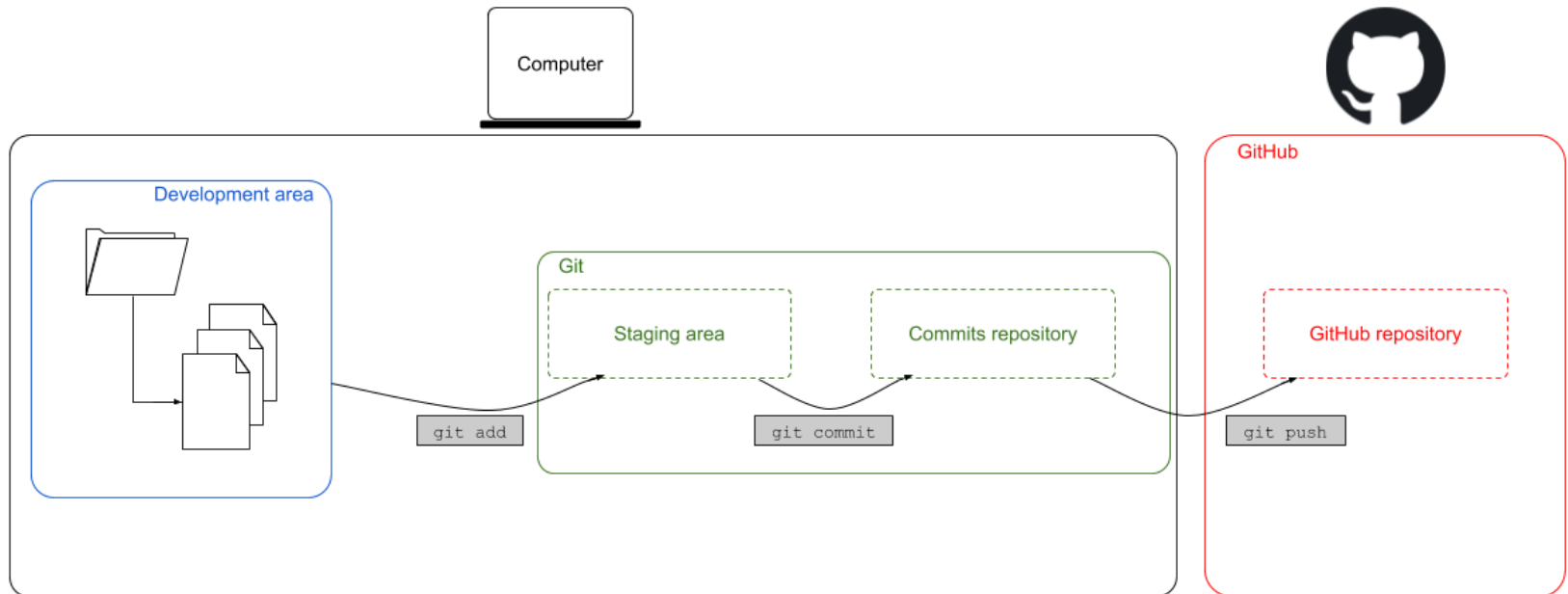
- What is Git used for?
 - ▶ Keep track of changes to your code
 - ▶ Synchronize code between different people
 - ▶ Testing new code



1. Introduction



1. Introduction



2. Configurations

- First use requires some personalisation in the configurations
- Three levels of configurations
 - ▶ Individual repository
 - ▶ User (global)
 - ▶ System

2. Configurations

- Windows: start 'Git Bash'
- Mac & Linux: start 'Terminal'
- Show our global config file with the following command

```
$ git config --global --list
```


2. Configurations

- Account information
 - ▶ Username & e-mail address

```
$ git config --global user.name "yourgithubusername"  
$ git config --global user.email "your_email@domain.com"
```

- ▶ Set SSH-keys: [link](#)

2. Configurations

- Editor
 - ▶ Git uses the editor to add a message or solve conflicts
 - ▶ Vim, Emacs, Notepad, Visual Studio Code, Atom, etc. ([link](#))

```
$ git config --global core.editor <editor>
```

- ! Editors are omitted as much as possible during this course

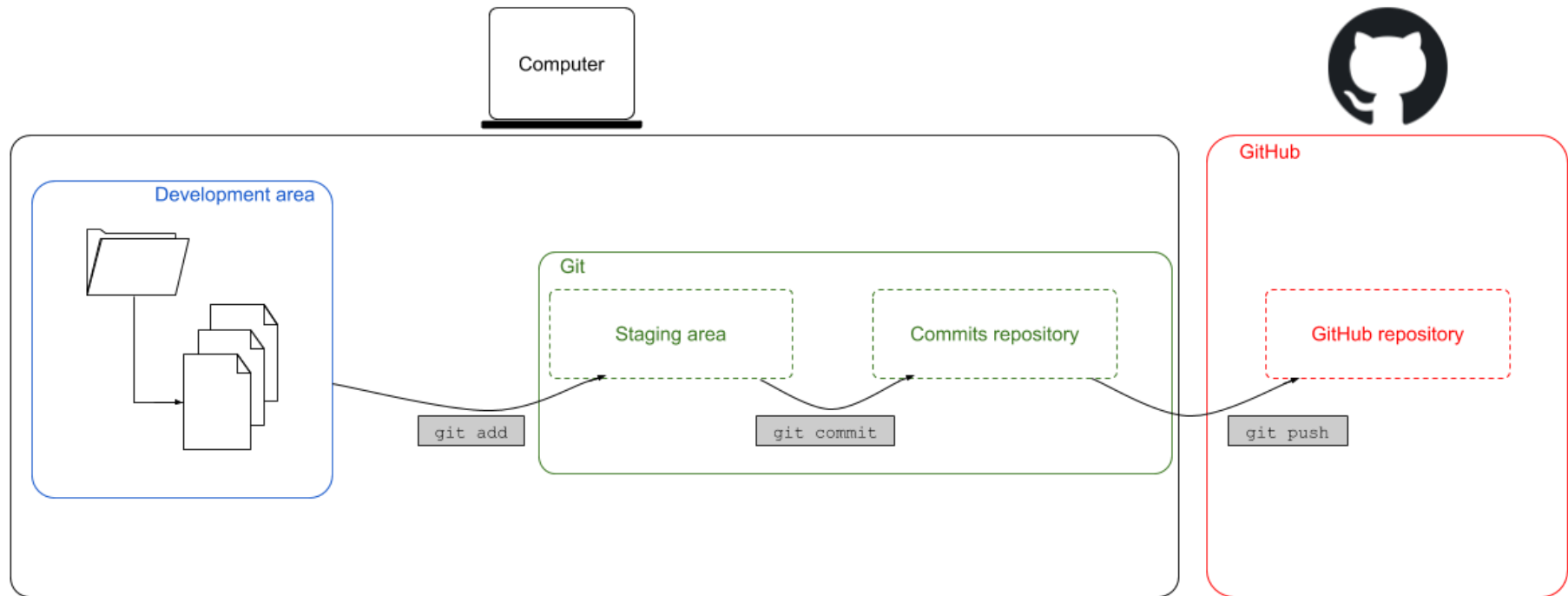
2. Configurations

- SSH keys
 - ▶ Connect to remote servers and services in a secure way.
 - ▶ Connect to GitHub without supplying your username or password at each visit
 - ▶ Instructions at [GitHub](#) or [our website](#).
- Aliases
 - ▶ A new command tailored to your wishes
 - ▶ Often consists of an existing Git command (e.g. *git log*) followed by a bunch of variables that are renamed to a new command
 - ▶ Omits typing long commands

3. First commit

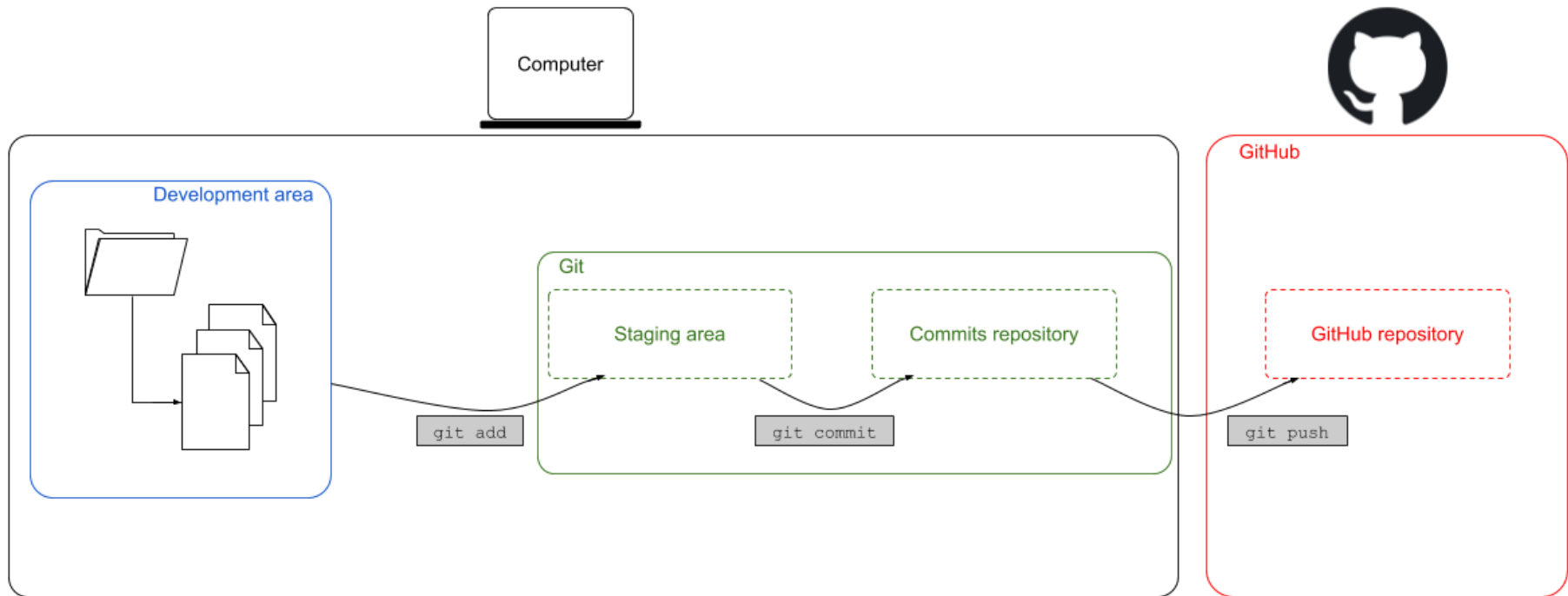
- 3.1 Routine usage & initialize a repository
- 3.2 Creating a repo from GitHub
- 3.3 Making our first commit
- 3.4 Pull
- 3.5 Creating a repo from your computer
- 3.6 Strength of staging area

3.1 Routine usage



```
$ git add <file>  
$ git commit -m "some text that explains what has changed"
```

3.1 Routine usage



```
$ git add <file>
$ git commit -m "some text that explains what has changed"
$ git push
```

3.1 Initialize a repository

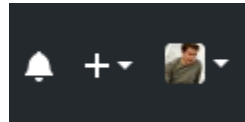
- Option 1:
 - ▶ Make a GitHub repository online
 - ▶ Clone it to your computer & start working
- Option 2:
 - ▶ Initialize Git on a folder on your computer
 - ▶ Create an empty GitHub repository online
 - ▶ Connect local folder with GitHub
- When option 1 and when option 2?

3.2 Create a repo from GitHub

- Let's start by opening GitHub first

3.2 Create a repo from GitHub

- Click the + icon in the upper right corner & select **New repository**



or





3.2 Create a repo from GitHub

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
[Import a repository.](#)


Owner Repository name *


 vibbits / introduction-github 

Great repository names are short and memorable. Need inspiration? How about **psychic-umbrella**?

Description (optional)


This is a test-repository for the GitHub tutorial.

☐  **Public**
Anyone can see this repository. You choose who can commit.

☒  **Private**
You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

☒ **Initialize this repository with a README**
This will let you immediately clone the repository to your computer.

Add .gitignore: **None** | Add a license: **None** 

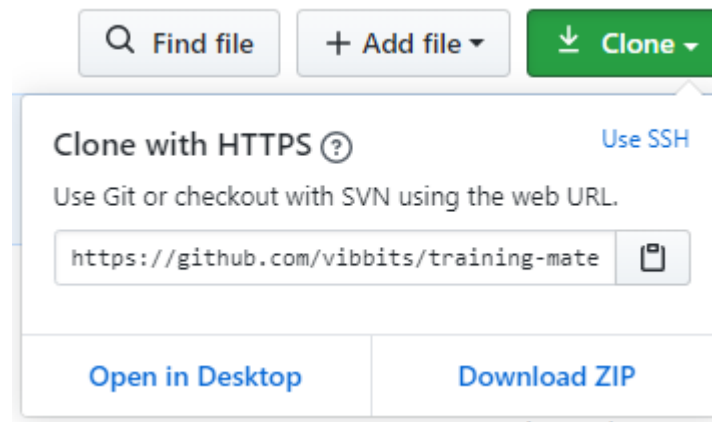
Create repository

3.2 Create a repo from GitHub

The screenshot shows a GitHub repository page for 'vibbits / introduction-github'. The repository is private and has 2 watchers, 0 stars, and 0 forks. The main navigation bar includes links for Code, Issues (0), Pull requests (0), Actions, Projects (0), Wiki, Security (0), Insights, and Settings. The repository description states: 'This is a test-repository for the GitHub tutorial.' Below this, there are statistics: 1 commit, 1 branch, 0 packages, 0 releases, and 1 contributor. The 'Branch: master' dropdown is set to 'master', and there is a 'New pull request' button. Action buttons include 'Create new file', 'Upload files', 'Find file', and a green 'Clone or download' button. The commit history shows 'tmuylder Initial commit' with the latest commit hash 'f955df3' and the time 'now'. The file list shows 'README.md' as the initial commit. The README content is displayed below, showing the repository name 'introduction-github' and the description 'This is a test-repository for the GitHub tutorial.'

3.2 Create a repo from GitHub

- Click on Clone & copy the link



- In Terminal or Git Bash, navigate to a location and:

```
$ git clone <link>
```

Exercise

- Create a repository on Github (with README file)
- Upload [this file](#) to the repository on GitHub
- Clone the repository to your computer
- Answer the following questions:
 - ▶ What is GitHub asking you to do when uploading the file
 - ▶ Which stage is omitted when uploading this file directly to GitHub
 - ▶ How many files are there in your local repository?

3.3 Our first commit

- 1. Staging
 - ▶ Download [this file](#) and add it in the folder on our local computer (not in GitHub)
 - ▶ First stage the file into the staging area
 - ▶

```
$ git add <file>
```

 will stage a specific file
 - ▶

```
$ git add .
```

 will stage all changed or new files

3.3 Our first commit

- 2. Committing
 - ▶ Commit message: informs our future selves of the changes that were done in this commit
 - ▶ Explanatory, but still brief
 - ▶ Best practices or guidelines: [here](#)

```
$ git commit -m "some message"
```

Question

- Which of the following commit messages would be most appropriate for a hypothetical commit made to our README.md file?
 - ▶ “Update README file”
 - ▶ “Added line ‘We use this repo as an example’ to README.md”
 - ▶ “Added purpose description to the README file”

Question

- What has happened after committing?
 - ▶ We saved a version of the file which is now visible on GitHub.com
 - ▶ We saved a version of the file which is now stored in our commit repository

3.3 Our first commit

- 3. Pushing
 - ▶ From commit repository locally to GitHub
 - ▶ Saving your files online

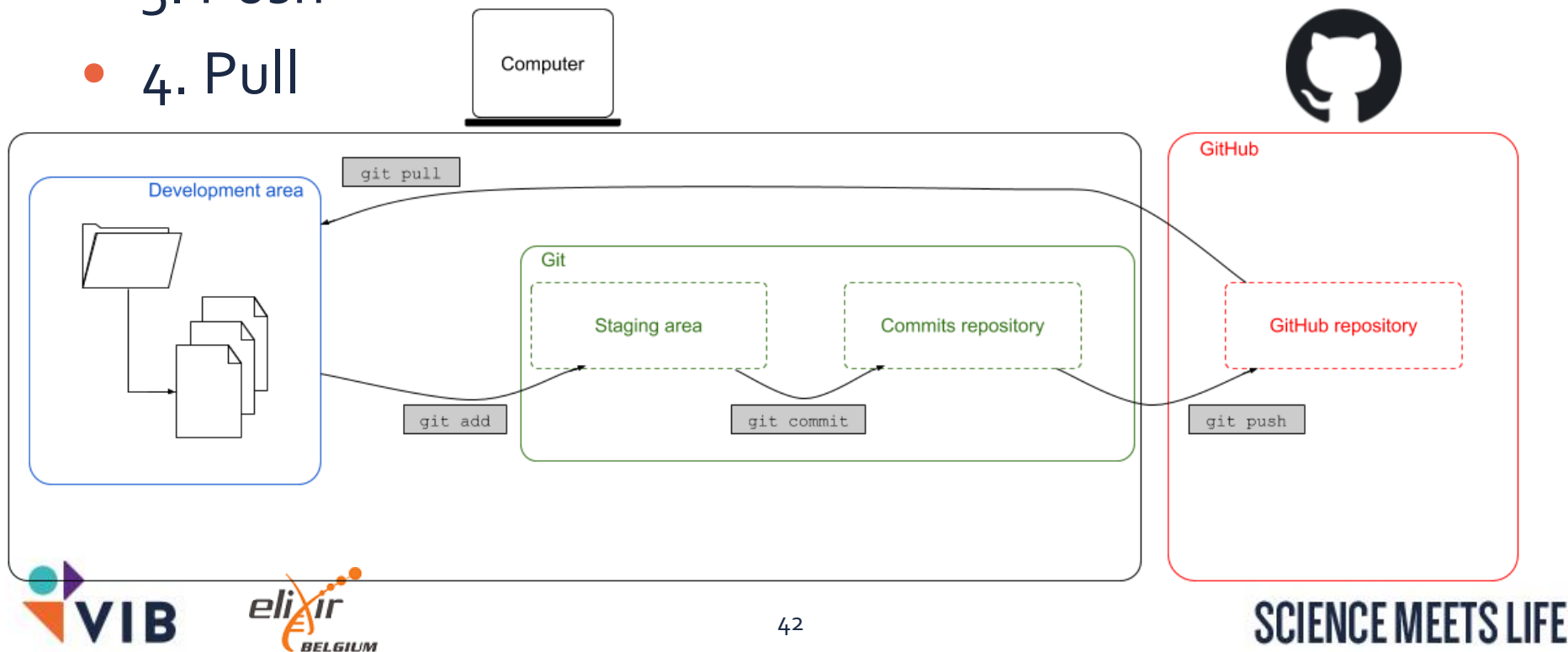
```
$ git push
```

Exercise

- Add a title to both files
 - ▶ `"# Title plot 1"`
 - ▶ `"# Title plot 2"`
- You can choose how you do this: e.g. open the files in a text editor and add the line on top of the file. Follow the steps that we just learned to push your changes to our GitHub repository.

3.4 Pull

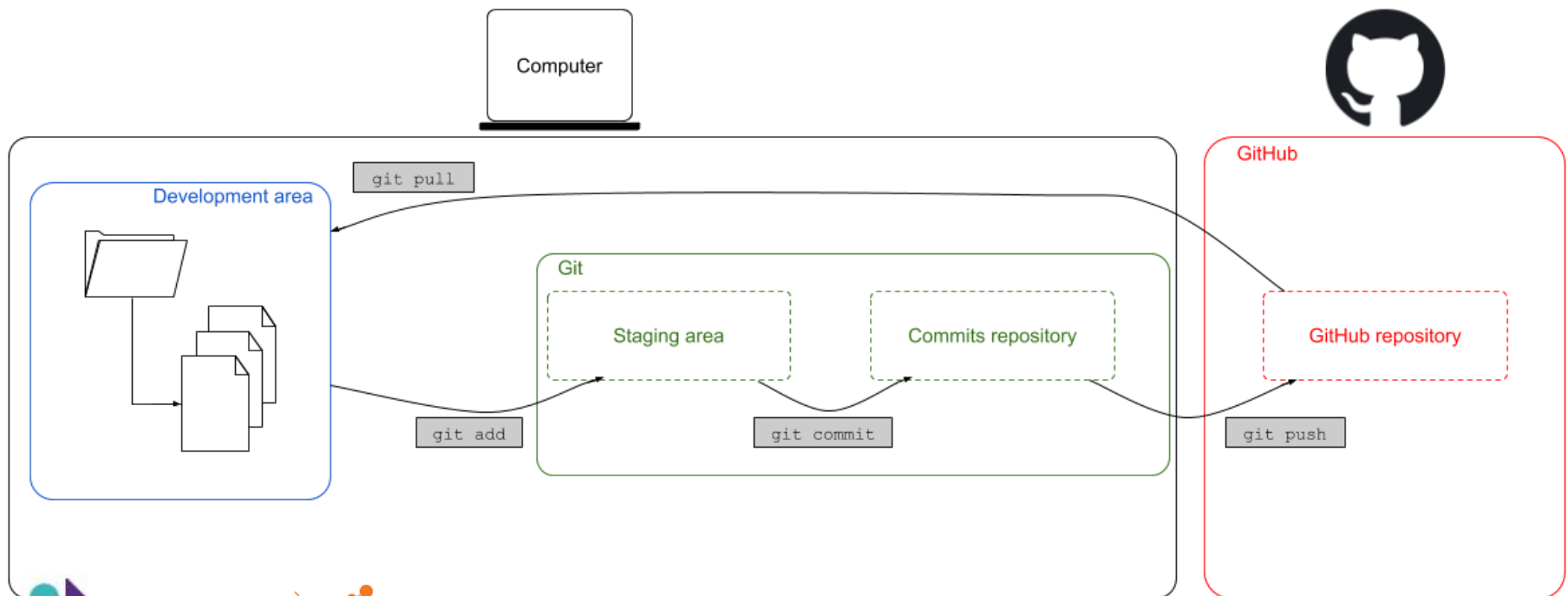
- 1. Stage
- 2. Commit
- 3. Push
- 4. Pull



3.4 Pull

- Incorporate changes from GitHub to your local repository

```
$ git pull
```



3.4 Pull (exercise)

- Make a change in the README file in the repository on GitHub
- Commit changes
- GitHub is now one commit ahead of our local repository.
- Pull this commit into our local repository by using the following command:

```
$ git pull
```

3.5 Create new repo locally

- Initiate Git on a folder on your computer

```
$ git init
```

- Open GitHub, create new repo and DO NOT initialize with README, gitignore or license
- In your Terminal or Git Bash follow the steps from GitHub:

```
$ git add .  
$ git commit -m "initial commit"  
$ git branch -M main  
$ git remote add origin git@github.com:<username>/<repository>.git  
$ git push -u origin main
```

Question

- What if we want to create a new folder inside the folder which we are using for version controlling? Do we need to initialize Git inside this subfolder as well?

Question

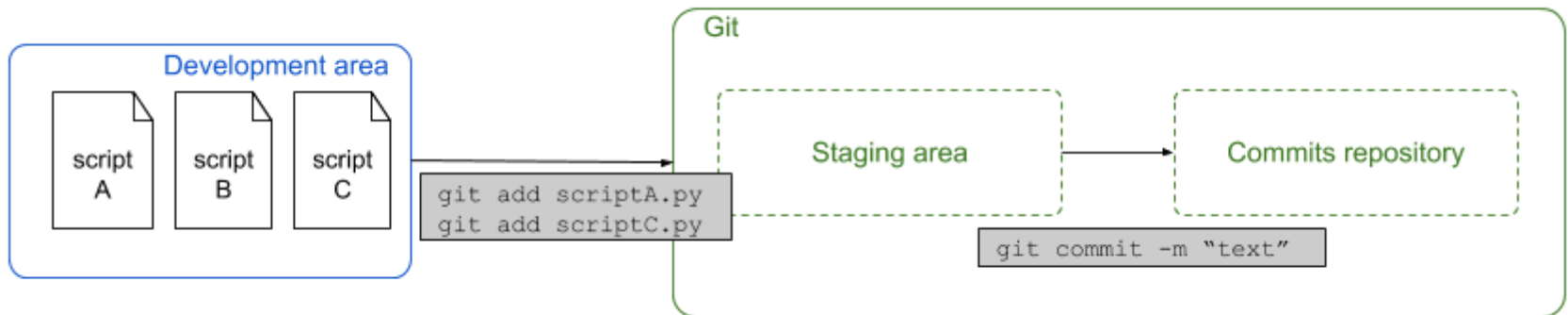
- How can we know whether a folder is already initialized with Git, meaning that we are already version controlling the project?

Exercise

- Find (or make) a folder on your computer with files that you want to version control.
- Initialize Git on that folder and make it (privately) available on GitHub.

3.6 Strength of the staging area

- Why is it useful to have that many steps to save a file?



- Separate commits depending on the relatedness of your changes

4. Status & history

- Get an overview of which files are in which status (staging area, committed)

```
$ git status
```

```
On branch main
```

```
Your branch is up to date with 'origin/main'.
```

```
nothing to commit, working tree clean
```

→ We're on main branch, the default branch in Git

→ Local repository and GitHub repository are exactly the same

→ Suggestion whether we should commit some files from the staging area

4. Status & history

- Let's make some changes to a file and check the status again

```
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be
  committed)
  (use "git restore <file>..." to discard changes in
  working directory)
        modified:   plot1.R

no changes added to commit (use "git add" and/or "git
commit -a")
```

First two lines remained the same

4. Status & history

- Add the file to the staging area


```
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   plot1.R
```

First two lines remained the same

4. Status & history

- List all your previous commits
- Long list of several blocks like this:

 Unique ID
commit e2d7e9a0b461426bee6b3ffd7583237bc5671dc1
Author: random <hello@gmail.com>
Date: Wed Jan 01 01:23:45 2020 +0200

The informative commit message



The commit message that you entered in the 'git commit' command

4. Status & history

- Many parameters can tweak how you see your history
 - ▶ `--oneline`
 - ▶ `--graph`
 - ▶ `--decorate`
- This is where aliases come in very handy

Question

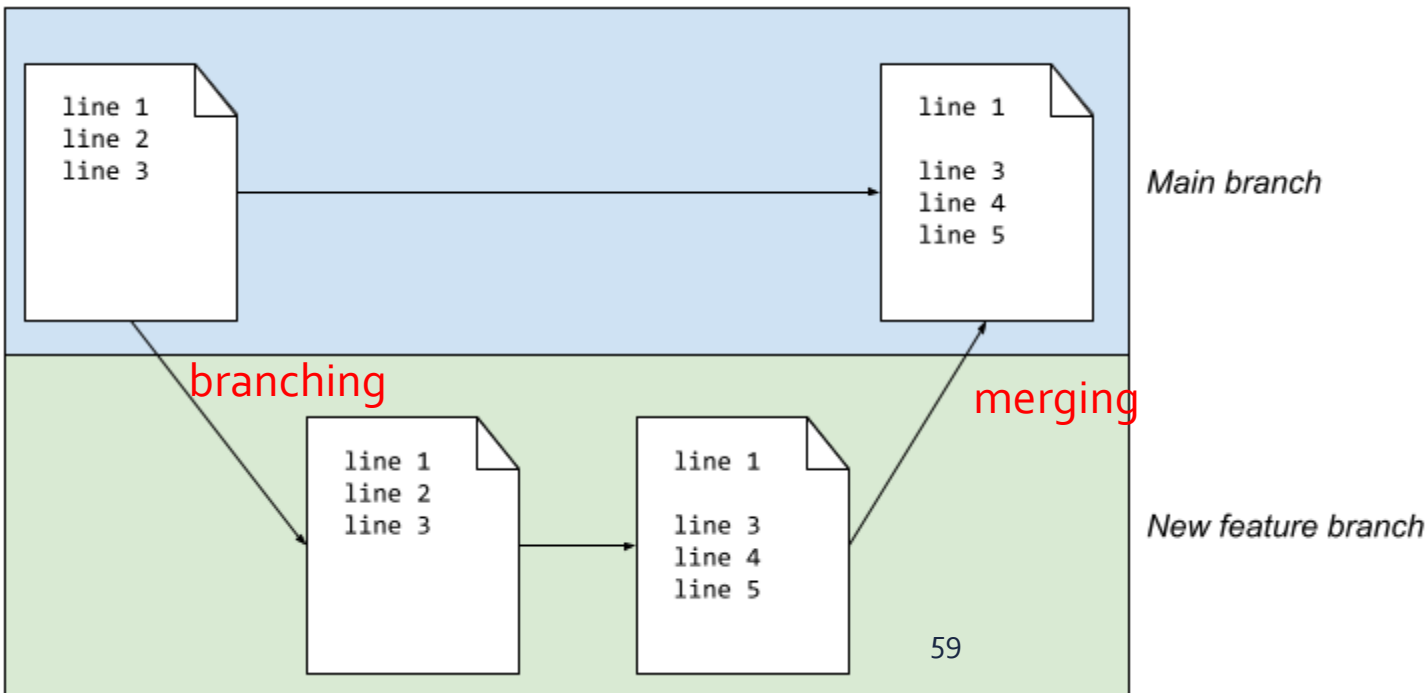
- Why is it useful to have the author's name and e-mail address in the history log?

5. Branching & merging

- 5.1 What's a branch?
- 5.2 Create a branch on GitHub
- 5.3 Merging branches on GitHub
- 5.4 Branching locally
- 5.5 Deleting branches

5.1 What's a branch?

- Testing new code while not messing up the original code.
- *main* branch by default



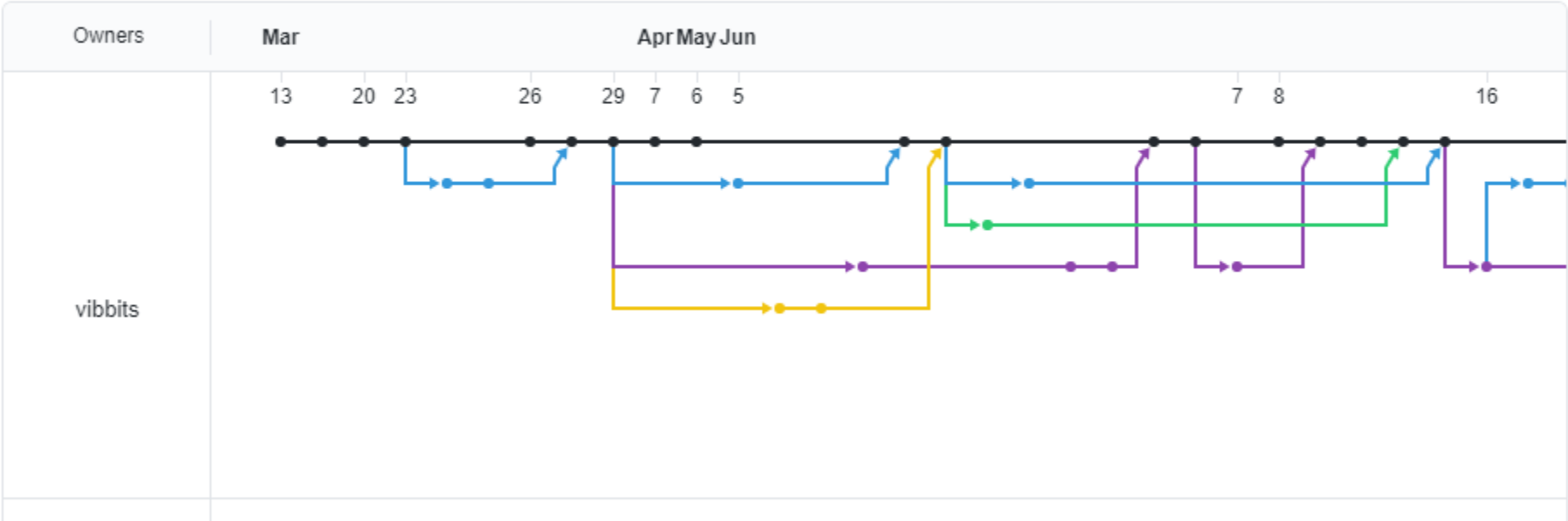
5.1 What's a branch?

- Testing new code while not messing up the original code.
- *main* branch by default
- As many new branches as you like
- Create branch on GitHub (6.2) or locally (6.4)
- Create branches for each new feature

branch early, and branch often

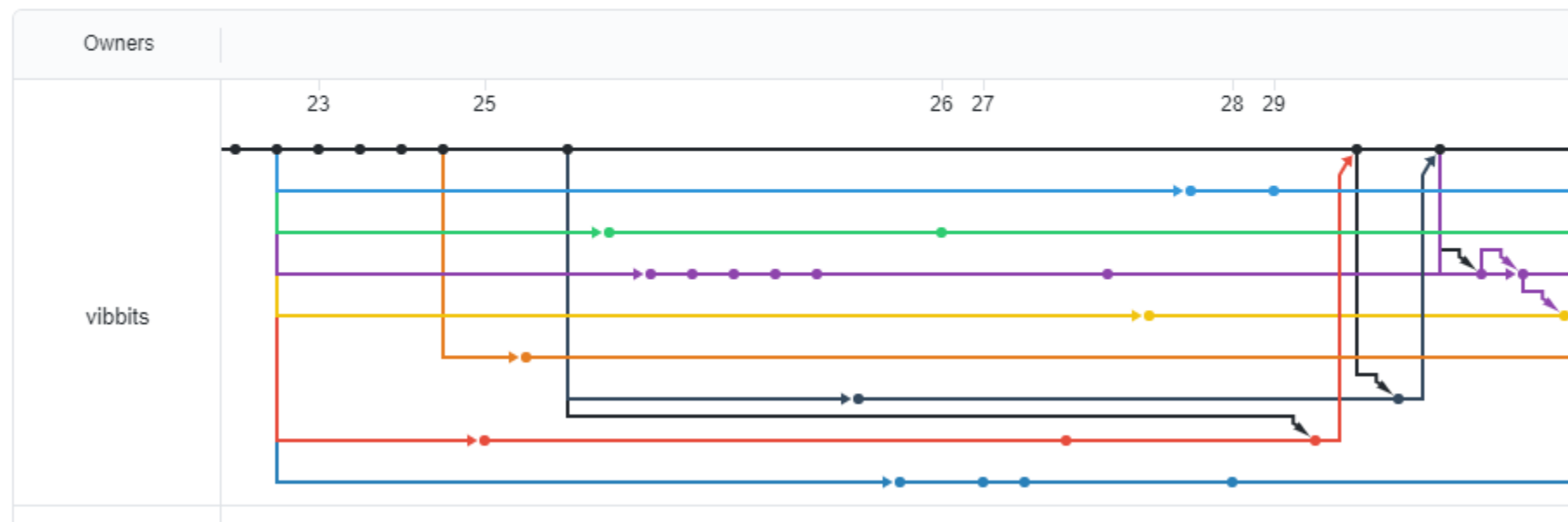
Network graph

Timeline of the most recent commits to this repository and its network ordered by most recently pushed to.



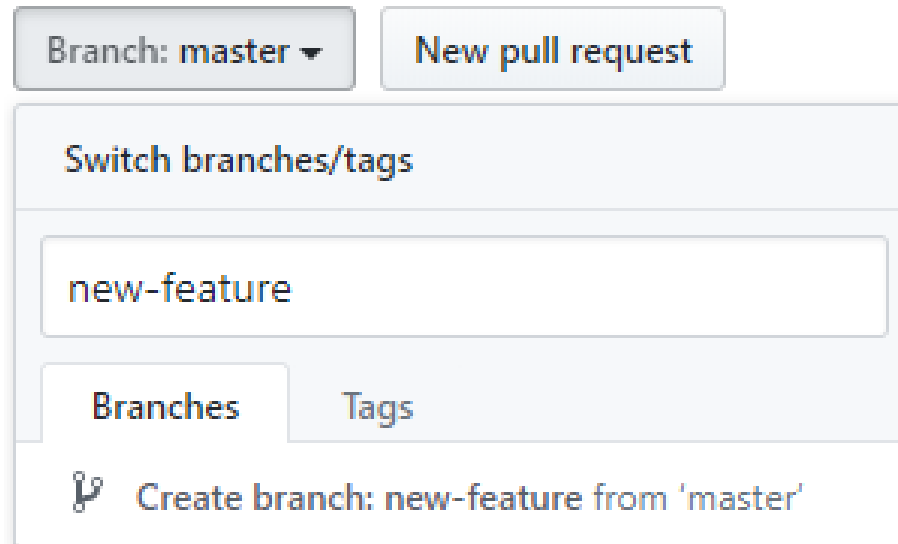
Network graph

Timeline of the most recent commits to this repository and its network ordered by most recently pushed to.



5.2 Branching on GitHub

1. Click the button: **'Branch: main'**
2. In **'Find or create a branch...'** type the name of your new branch, e.g. *new-feature*
3. Click **'Create branch'**: new-feature



Branch: master ▼ New pull request

Switch branches/tags

new-feature


Branches Tags


🔗 Create branch: new-feature from 'master'

5.3 Merging branches on GitHub

- Branches are merged into the main branch by making a **pull request**
- 1. Click on the green Compare & pull request button
- 2. On top of the repository, click Pull requests

Your recently pushed branches:

 new-feature (9 minutes ago)

 Compare & pull request


<> Code


! Issues 15

 Pull requests

▶ Actions

 Projects


 Wiki

 Security


 Insights

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

 base: master ← compare: new-feature ✓ **Able to merge.** These branches can be automatically merged.




Title 

Write

Preview

AA B i “ <> 🔗 ☰ ☷ ✓ @ ★ ↶

Leave a comment

Attach files by dragging & dropping, selecting or pasting them. 

Create pull request

▼

Reviewers 

No reviews

Assignees 

No one—assign yourself

Labels 

None yet

Projects 

None yet

Milestone 


No milestone

Linked issues 

Use [Closing keywords](#) in the description to automatically close issues

Helpful resources

[GitHub Community Guidelines](#)

 Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

5.3 Merging branches on GitHub

- The pull request should be interpreted as a request to pull the new branch and all of its changes into the main branch.
- main: The base where it would be pulled towards
- New-feature: Branch with changes
- GitHub checks the compatibility of the branches
- Descriptive title text and if appropriate some additional comment.

5.3 Merging branches on GitHub

- Underneath PR: summary of changes

1 commit

1 file changed

0 commit comments

1 contributor



Commits on Apr 28, 2020



tmuylder

Added 3D scatterplot wt, hp, mpg

Verified

07a2b7f

Showing 1 changed file with 7 additions and 0 deletions.

Unified

Split

7 plot2.R

```
@@ -2,3 +2,10 @@
2      2      # Basic Scatterplot Matrix
3      3      pairs(~mpg+disp+drat+wt,data=mtcars,
4      4      main="Simple Scatterplot Matrix")
5      +
6      + # Install requirements & plotting of 3D scatterplot: weight, dis
7      + install.packages("scatterplot3d")
8      + library(scatterplot3d)
9      + attach(mtcars)
10     + scatterplot3d(wt,hp,mpg, pch=16, highlight.3d=TRUE,
11     +               type="h", main="3D Scatterplot")
```


5.3 Merging branches on GitHub


- Each commit done in the branch *new-feature* (i.e. only added these 7 lines in this case)
- Display of the file and a visual representation of what changed in that commit.


5.3 Merging branches on GitHub


- Click on **Create Pull Request**


Merging 3D scatterplot #1


 **Open**


tmuyllder wants to merge 1 commit into `master` from `new-feature` 



 Conversation **0**

 Commits **1**



 Checks **0**

 Files changed **1**




tmuyllder commented now  


No description provided.


  Added 3D scatterplot wt, hp, mpg


Verified `07a2b7f`

Add more commits by pushing to the `new-feature` branch on [vibbits/introduction-github](#).



 **Continuous integration has not been set up**
[GitHub Actions](#) and [several other apps](#) can be used to automatically catch bugs and enforce style.

 **This branch has no conflicts with the base branch**
Merging can be performed automatically.

Merge pull request 

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

Exercise

- Make a new branch & edit the *plot2.R* file. Add the following lines that will make a new plot. Commit changes and merge them back into the *main* branch.

```
# Install requirements & plotting of 3D scatterplot
install.packages("scatterplot3d")
library(scatterplot3d)
attach(mtcars)
scatterplot3d(wt, hp, mpg, pch=16, highlight.3d=TRUE,
              type="h", main="3D Scatterplot")
```

- ▶ These lines will allow us to investigate the relation between the weight, horsepower and miles per gallon variables of mtcars dataset in R.

5.4 Branching locally

- Most important commands
 1. `git checkout -b <new-branch>`: will create a new branch and move into this branch.
 2. `git branch <new-branch>`: will create a new branch, but will remain in the current branch (i.e. the main branch in this case)
 3. `git checkout <branch>`: will switch from one branch to the other.
 4. `git branch -a` list all the existing branches

5.4 Branching locally

- List all existing branches

```
$ git branch -a
* main
remotes/origin/HEAD -> origin/main
remotes/origin/main
remotes/origin/new-feature
```

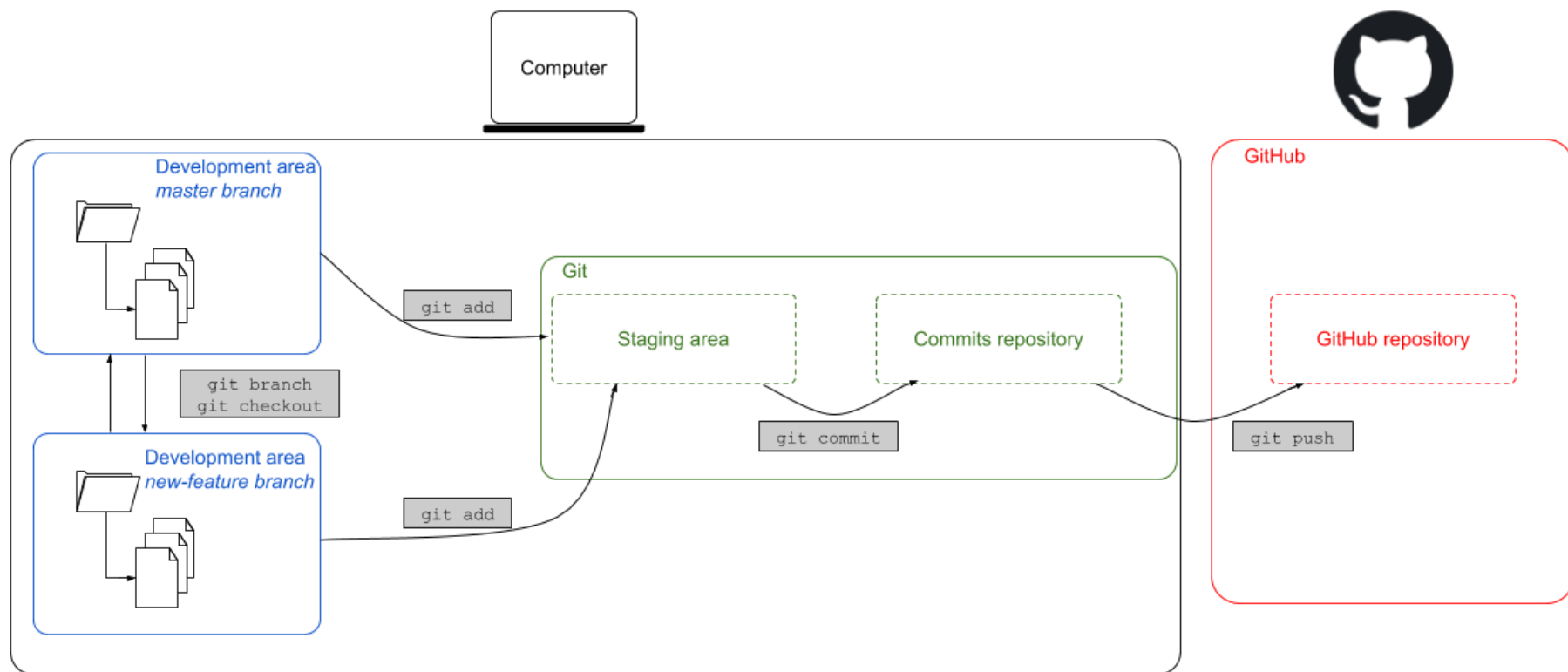
- Switch to another branch

```
$ git checkout new-feature
Switched to branch 'new-feature'
Your branch is up to date with 'origin/new-feature'.
```

- !!! Git checkout (-b)

5.4 Branching locally

- Example workflow



5.4 Branching locally

- Make a new branch
- Make necessary changes
- Stage changes
- Commit
- Push

```
$ git checkout -b <new-branch>  
$ git add .  
$ git commit -m "some useful commit message"  
$ git push --set-upstream origin <new-branch>
```

Exercise

- Make a new branch and make sure you're in the branch. Rewrite the README.md file so it contains the following text. Once the changes have been committed and pushed, create a pull request and merge the changes into the main branch.

```
# Downstream data-analysis R  
This repository contains all the scripts for the  
downstream data analysis of my project.
```

5.5 Deleting branches

- 1. in GitHub
 - ▶ Find all existing branches (two ways)

10 commits

3 branches

0 packages

0 releases

1 contributor

Branch: master ▾

Find file

+ Add file ▾

Clone ▾



tmuylder committed 0f75ae5 on May 13
Make repo a sample repo (update README)

11
commits

2
branches

- ▶ Delete branch

readme Updated 14 days ago by tmuylder

3 | 0

#2 Merged



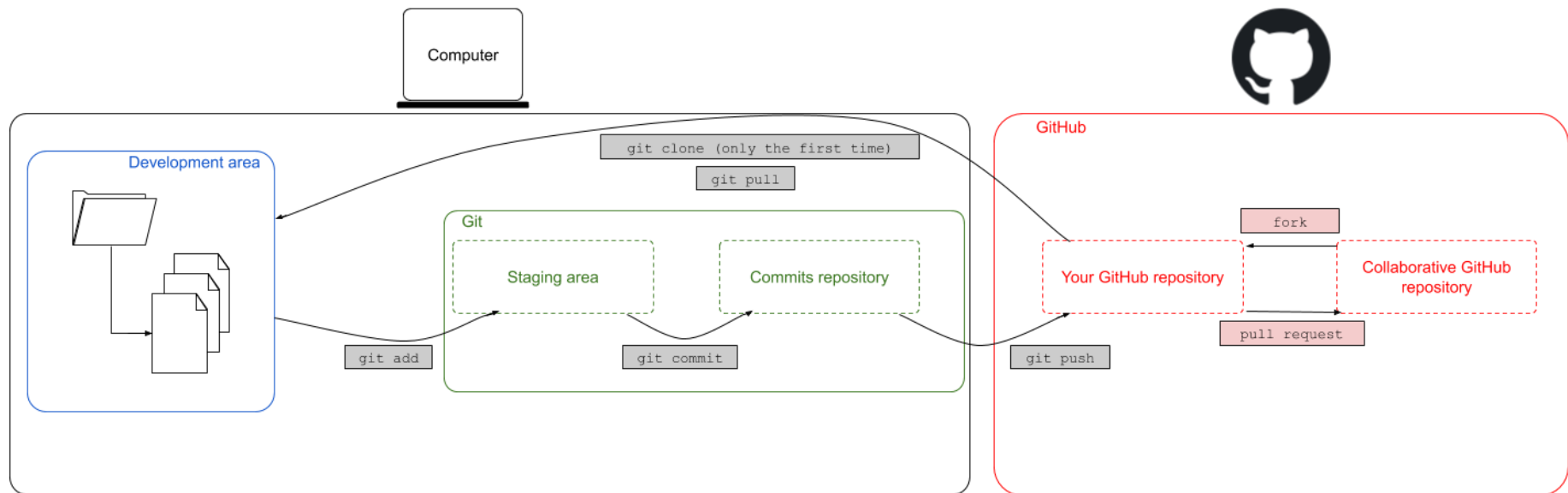
5.5 Deleting branches

- 2. Locally

```
$ git branch -d <name-of-the-branch>
```

6. Forking

- 6.1 Fork & Pull workflow
- 6.2 How to fork
- 6.3 Edit the fork
- 6.4 Pull requests
- 6.5 Forking overview



6.1 Fork & Pull workflow

- Strategy for collaborating on a project
- Fork & pull workflow:
 - ▶ Fork a central repository → personal repo
 - ▶ Clone it locally
 - ▶ Create a branch [question: why?]
 - ▶ Make changes (stage – commit – push)
 - ▶ (Merge upstream changes into your dev branch)
 - ▶ Make a pull request

6.1 Fork & Pull workflow

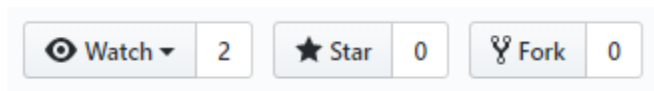
- Terminology:
 - ▶ *Upstream*: generally refers to the original repository that you have forked
 - ▶ *Origin*: your fork = your own repo on GitHub

6.2 How to fork (exercise)

- Find repository via search bar



- Find the *fork-repository* of vibbits & fork it



- Successful if the following appears



- Follow the steps as described in the repo ([link](#))

6.3 Edit the fork (exercise)

- Normally: clone it locally, make a branch, make edits
- Now: you can create a branch in GitHub & make changes directly

The screenshot shows the GitHub interface for a repository named 'fork-repository' by user 'tmuylder'. The repository is a fork of 'vibbits/fork-repository'. The top navigation bar includes links for 'Code', 'Pull requests' (0), 'Actions', 'Projects' (0), 'Wiki', 'Security' (0), 'Insights', and 'Settings'. Below the navigation bar, a message states 'This repository is made for teaching purposes.' with an 'Edit' button. A summary bar shows '5 commits', '1 branch', '0 packages', '0 releases', and '1 contributor'. Below this, there are buttons for 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find file', and a green 'Clone or download' button. The commit history section shows the current branch is '1 commit ahead of vibbits:master'. The latest commit is by 'tmuylder' with the message 'Added my name', committed 13 seconds ago. Below the commit list, there are two files: 'README.md' (with a message 'Add instructions to README file' from 13 minutes ago) and 'participants.txt' (with a message 'Added my name' from 12 seconds ago).

tmuylder / fork-repository
forked from vibbits/fork-repository

Watch 0 Star 0 Fork 1

Code Pull requests 0 Actions Projects 0 Wiki Security 0 Insights Settings

This repository is made for teaching purposes. [Edit](#)

[Manage topics](#)

5 commits 1 branch 0 packages 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

This branch is 1 commit ahead of vibbits:master. [Pull request](#) [Compare](#)

tmuylder Added my name Latest commit 8711fb0 13 seconds ago

README.md	Add instructions to README file	13 minutes ago
participants.txt	Added my name	12 seconds ago

6.3 Edit the fork (exercise)

- Upstream repository might have changed.
- Commits behind or ahead

This branch is 1 commit ahead, 1 commit behind vibbits:master.

 Pull request  Compare

6.4 Pull request (exercise)

- Repositories have diverged and need to be merged
- Exactly the same as merging branches: pull request
- Make a pull request to merge changes into vibbits:dev (!!! **dev**)

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).



base repository: vibbits/fork-repository ▾

base: dev ▾



head repository: tmuyllder/fork-repository ▾

compare: tuur ▾

✓ **Able to merge.** These branches can be automatically merged.



Update participants.txt



Write

Preview

H

B

I

≡

<>



≡

≡



Leave a comment

Attach files by dragging & dropping, selecting or pasting them.



☒ Allow edits by maintainers

Create pull request






Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).


6.4 Pull request (exercise)


- It compares the main branch of the forked repository with the upstream (base) repository
- It's able to merge these two branches with(out) any conflicting errors
- Summary of changes that have been done in the branch that will be merged into the upstream.


Added my name tmuylder #1

 **Open** tmuylder wants to merge 1 commit into `vibbits:master` from `tmuylder:master` 

 Conversation 0

 Commits 1

 Checks 0

 Files changed 1



tmuylder commented now

Member



No description provided.



Added my name

Verified

8711fb0

Add more commits by pushing to the `master` branch on `tmuylder/fork-repository`.



Continuous integration has not been set up

[GitHub Actions](#) and [several other apps](#) can be used to automatically catch bugs and enforce style.



This branch has no conflicts with the base branch

Merging can be performed automatically.

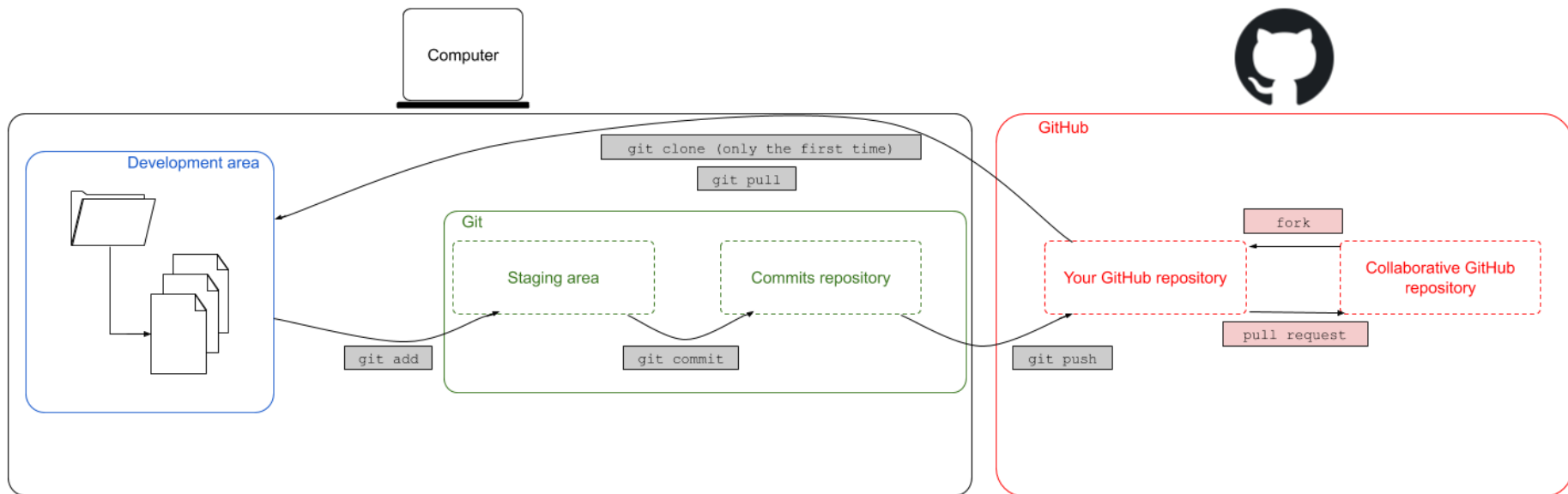
Merge pull request



You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

6.5 Forking overview

- *fork > edit > pull request (> merge)*
- *fork > clone (> branch) > edit-stage-commit-push > pull request (> merge)*



Question

- What if the upstream repository changed while you were working on your local repository and you want the upstream changes in your repository?

Further reading

- [Github – fork a repo](#)
- [Github – Creating a pull request from a fork](#)
- [Github – syncing a fork](#)

7. Gitignore

- 7.1 Introduction
- 7.2 Expressions
- 7.3 Standardized files

7.1 Introduction .gitignore

- Limit data upload to GitHub
- File size limit of 100Mb
- Sometimes your data is just in the same folder
- Or sometimes there are irrelevant files in the folder
- → .gitignore
 - ▶ List of expressions which define files it doesn't need to track

7.2 Expressions .gitignore

- Following folder structure

```
project-folder/  
|  
|- .ipynb_checkpoints  
|- .Rhistory  
|  
|- data/  
|   |- R1.fastq  
|   |- dataset.csv  
|  
...
```

7.2 Expressions .gitignore

- Ignore a file or folder

- ▶ Explicitly name them

```
data/R1.fastq
data/
.ipynb_checkpoints
```

- ▶ Wildcard *

```
*.csv
```

- ▶ Exception !

```
!dataset.csv
```

- ▶ Documentation line #

```
# Some information
```

```
project-folder/
|
|- .ipynb_checkpoints
|- .Rhistory
|
|- data/
|   |- R1.fastq
|   |- dataset.csv
|
...
```

7.3 Standardized .gitignore-files

- A set of predefined expressions based on the programming environment
- Ex.: Python, R, Ruby, Java, Perl, C++
- Can be added at the beginning when creating a repository or later on in the project

Other stuff

- Issues
 - ▶ <https://github.com/vibbits/training-material/issues/104>
 - ▶ <https://github.com/vibbits/training-material/issues/95>
- Pull request conversations/issues
 - ▶ <https://github.com/vibbits/gentle-hands-on-python/pull/7>
- Settings
- Project board
- Wiki

8. GitHub & RStudio

- There are three plausible scenarios:
 - ▶ You have a version controlled project on your computer which you want to integrate in Rstudio
 - ▶ You have a version controlled project on GitHub which you want to integrate in Rstudio locally
 - ▶ You have an Rstudio project that you now want to start version controlling

8.1 Starting a project

1. Integrating a version controlled project in Rstudio
 - ▶ File > New project...
 - ▶ **Existing Directory:** The preferred choice when a project folder already exists and which has previously been initialized with Git.
 - ▶ **Version Control:** Ideally for creating a new R project based on a repository in GitHub.

8.1 Starting a project

2. Initiating version controlling on an existing Rstudio project

- ▶ Tools > Version control > Project setup
- ▶ Select Git/SVN
- ▶ Version control system: Git (restart)
- ▶ Make GitHub repository (empty)
- ▶ New branch > add remote

8. GitHub & RStudio

- Full tutorial on <http://material.bits.vib.be>

Further reading

- <https://guides.github.com/>
- <https://help.github.com/>
- <https://learngitbranching.js.org/> (great visual exercises)

Extra questions

- How much money did Microsoft pay in June 2018 for the acquirement of GitHub?
 - ▶ \$1.5 million
 - ▶ \$7.5 million
 - ▶ \$1.5 billion
 - ▶ \$7.5 billion