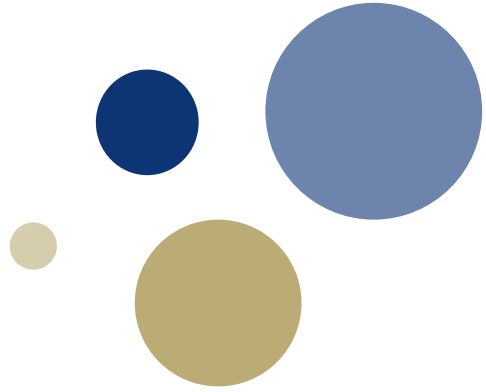


Version control for researchers

Sigurd Hofsmo Jakobsen

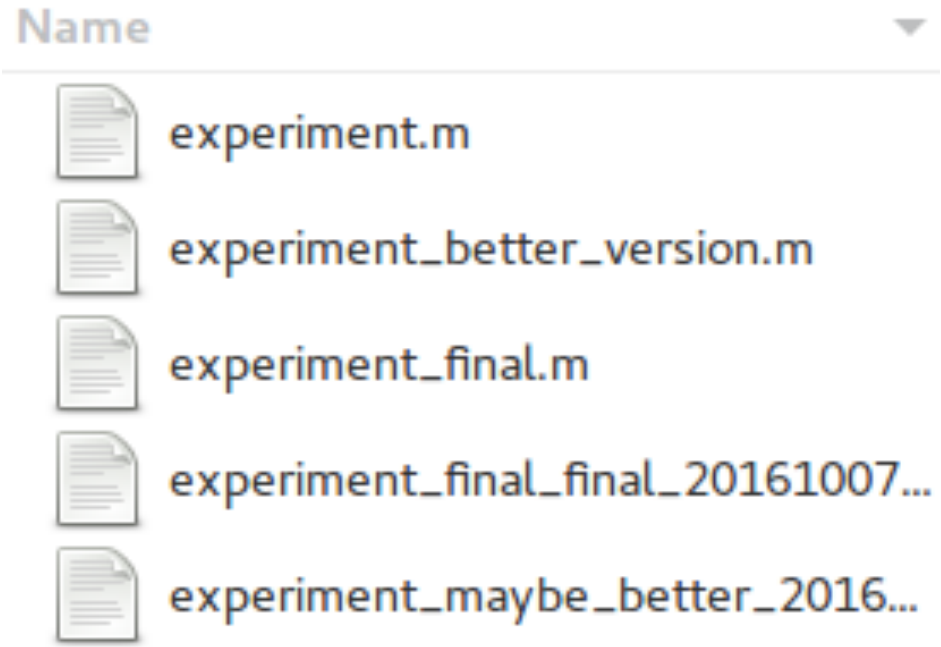
Department of electric power engineering

October 6, 2016

Outline

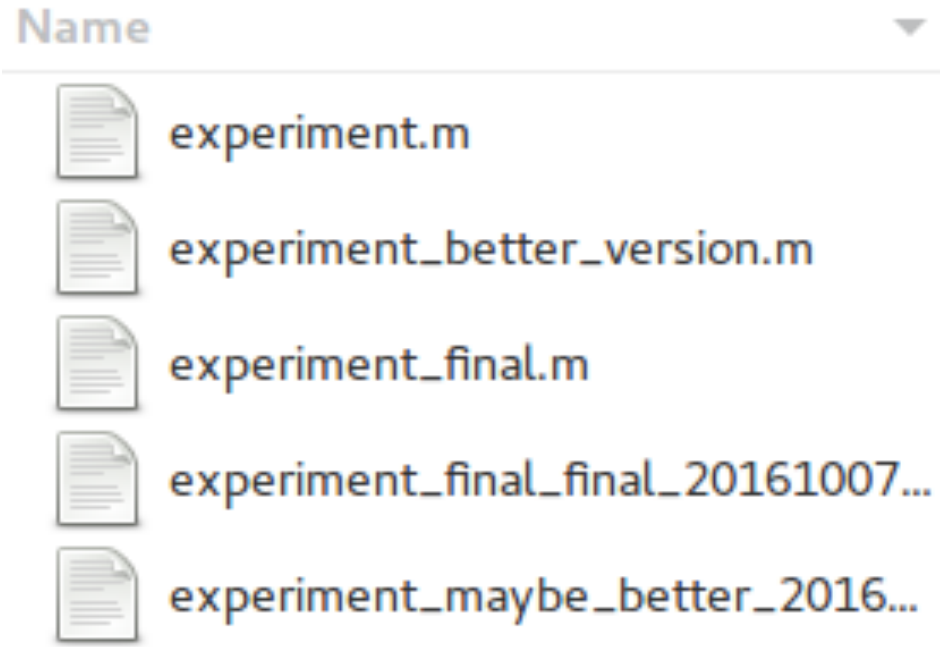
A basic example

— Only one person can work on a file at a time



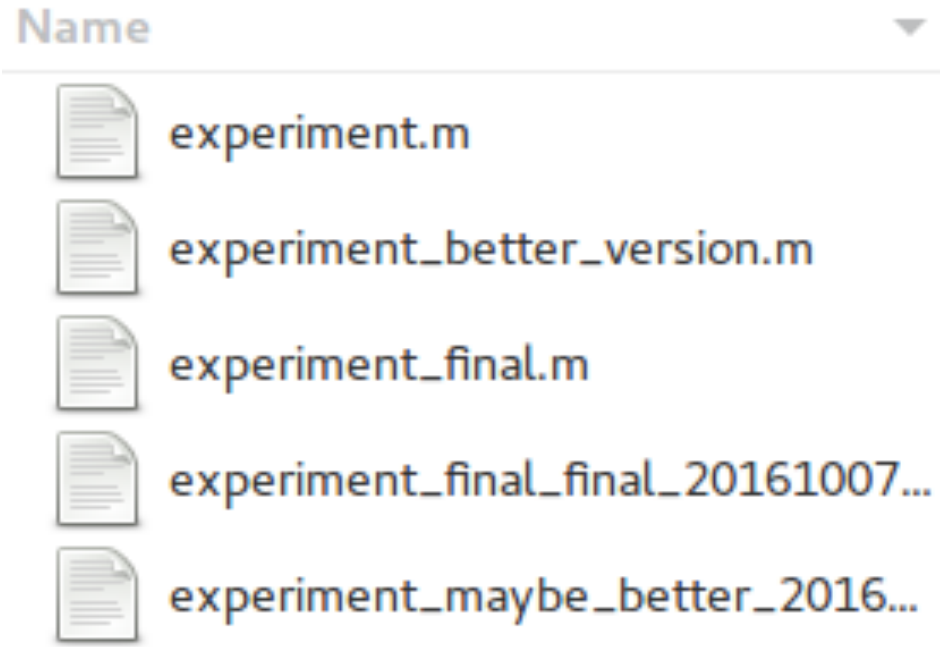
A basic example

- Only one person can work on a file at a time
- Difficult to keep track on which file to use



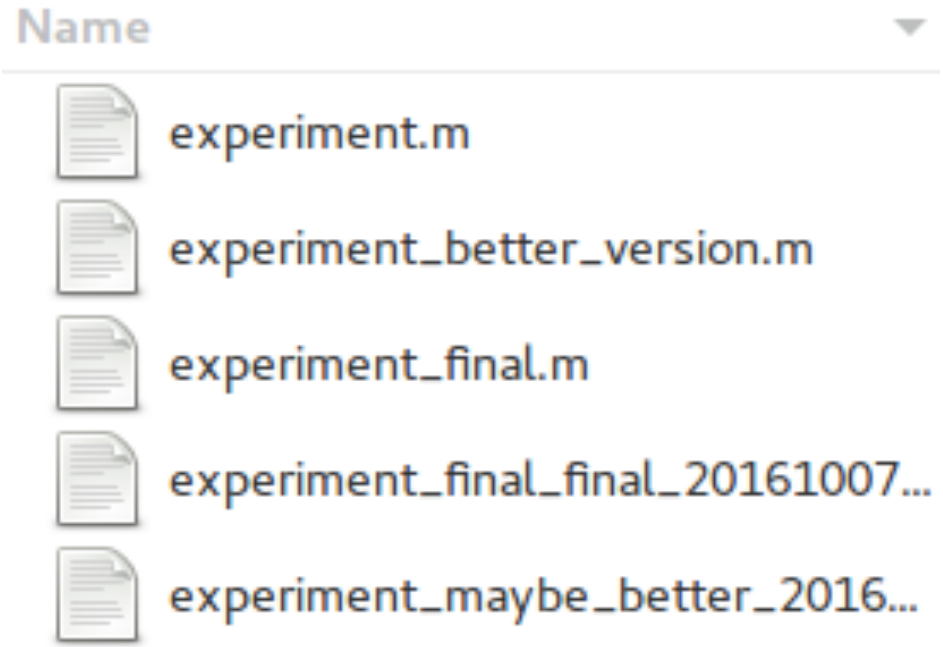
A basic example

- Only one person can work on a file at a time
- Difficult to keep track on which file to use
- Difficult to compare the files



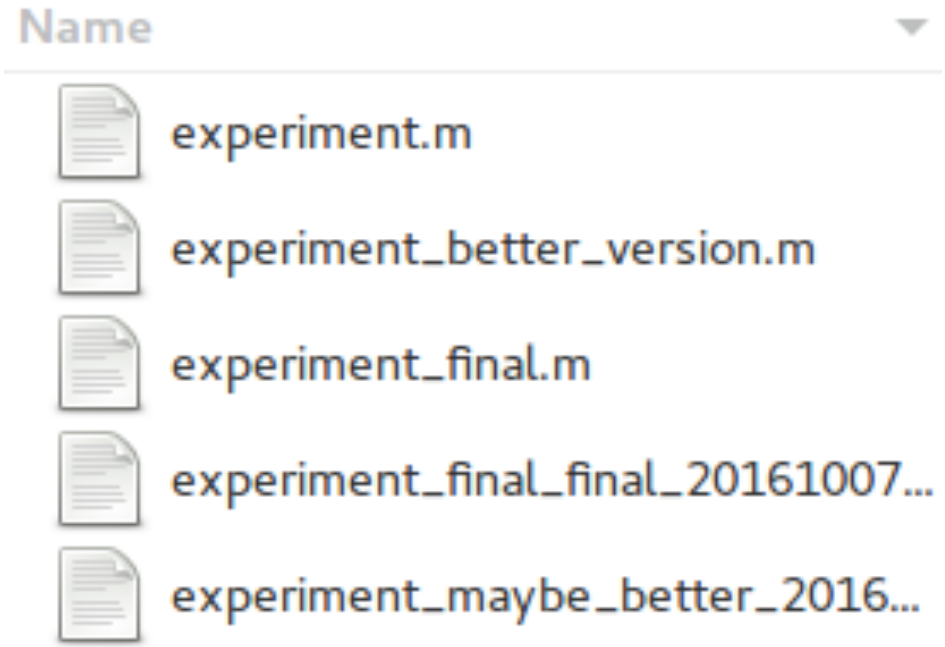
A basic example

- Only one person can work on a file at a time
- Difficult to keep track on which file to use
- Difficult to compare the files
- Do you have back up?



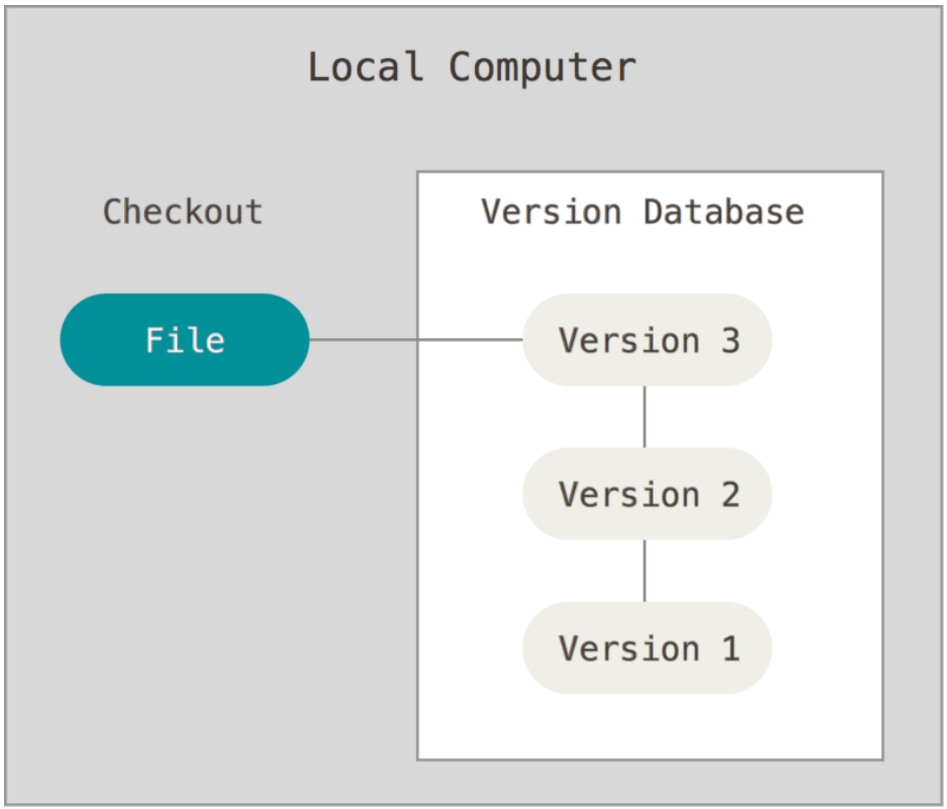
A basic example

- Only one person can work on a file at a time
- Difficult to keep track on which file to use
- Difficult to compare the files
- Do you have back up?
- Probably more issues



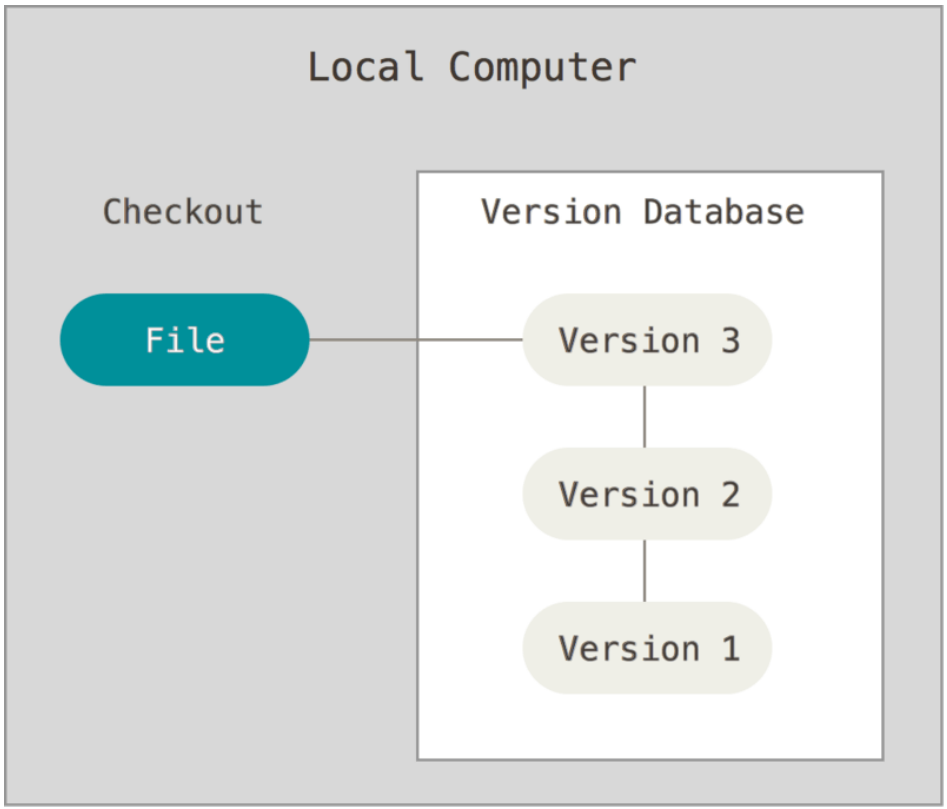
Smarter way of doing it

— Equivalent of storing your stuff on M



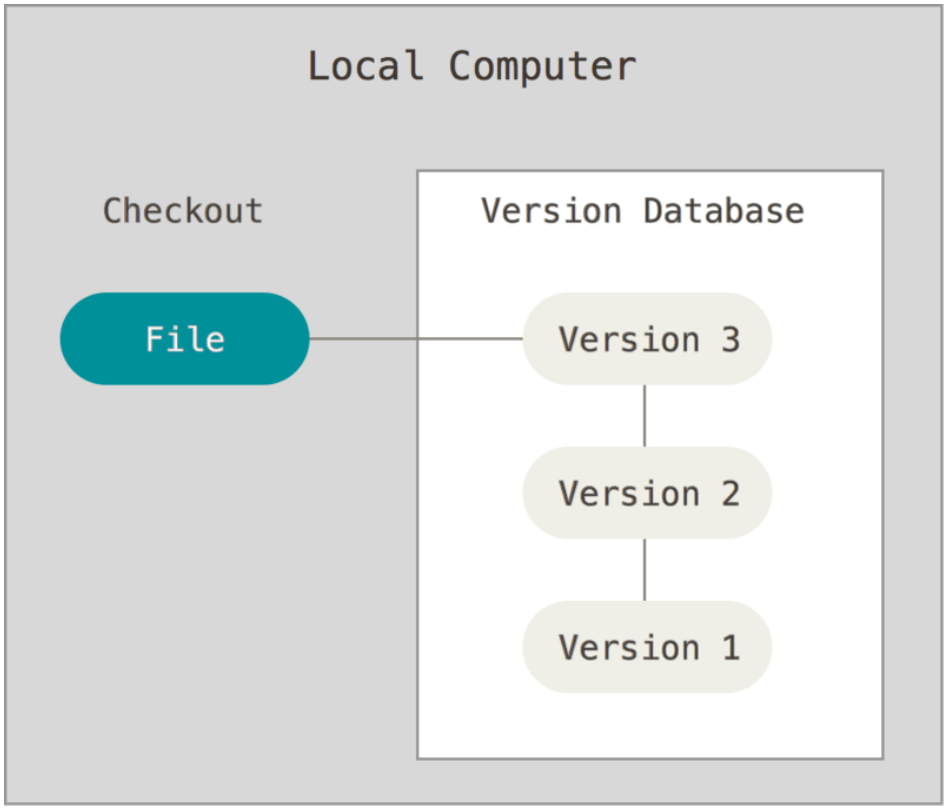
Smarter way of doing it

- Equivalent of storing your stuff on M
- Included in MAC



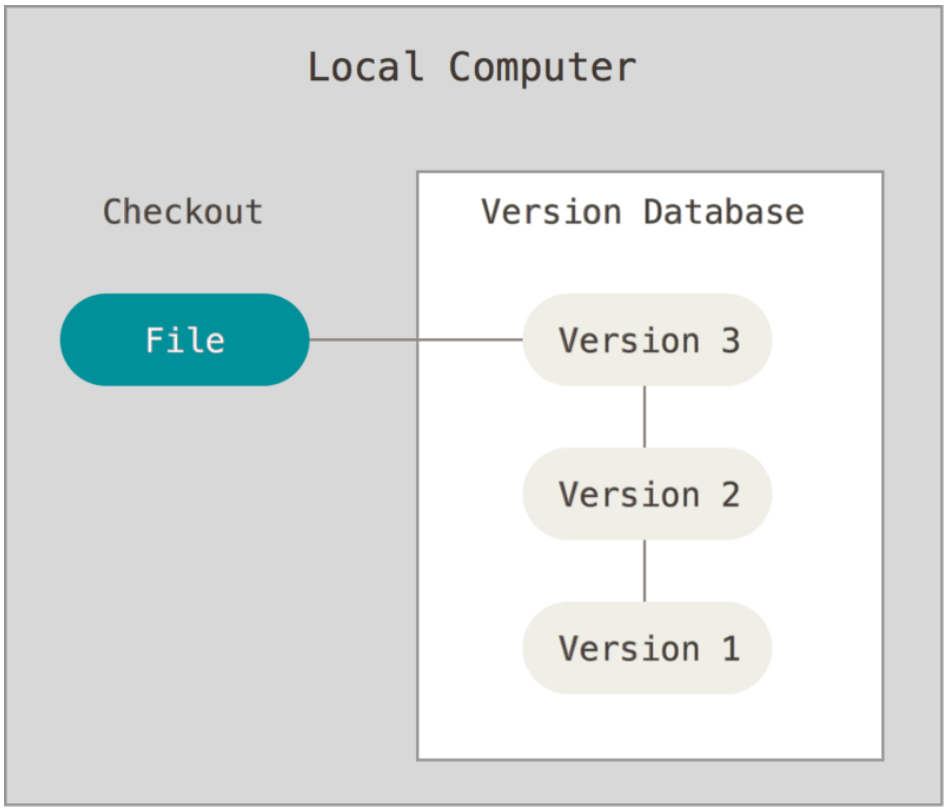
Smarter way of doing it

- Equivalent of storing your stuff on M
- Included in MAC
- Easy to set up



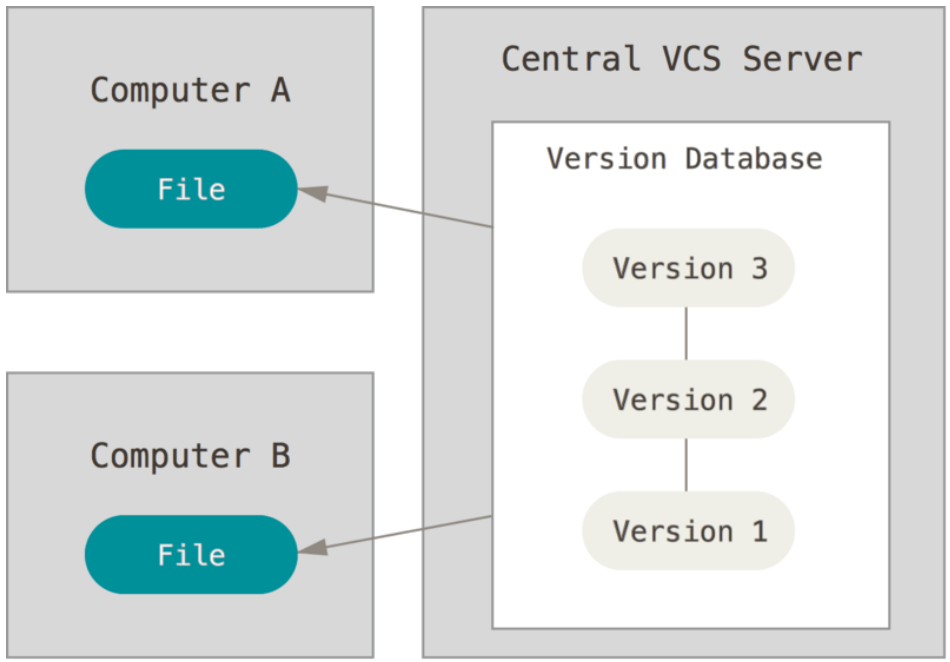
Smarter way of doing it

- Equivalent of storing your stuff on M
- Included in MAC
- Easy to set up
- Still difficult to collaborate



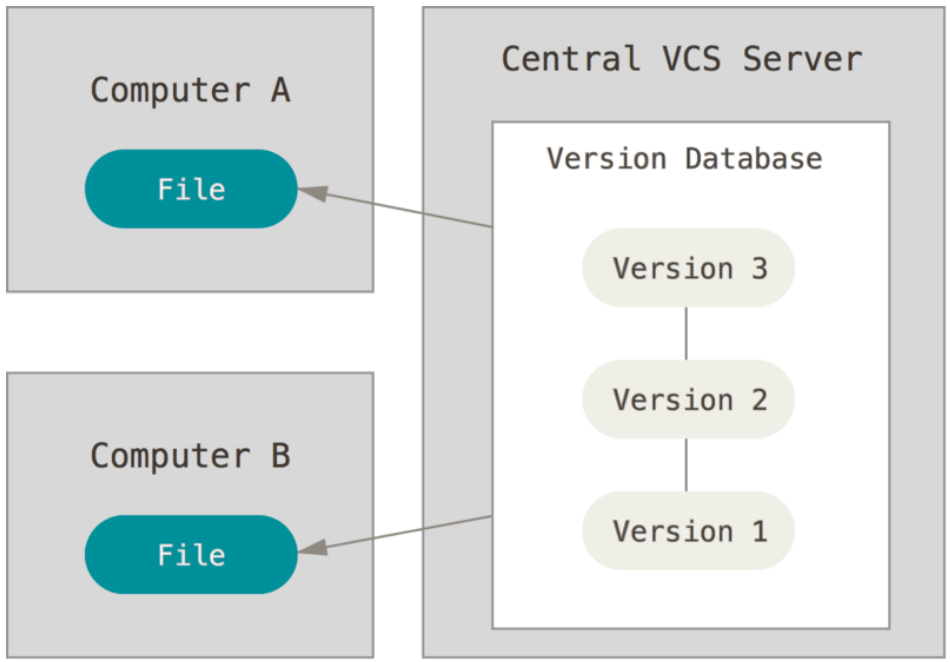
Centralized version control

— Examples:



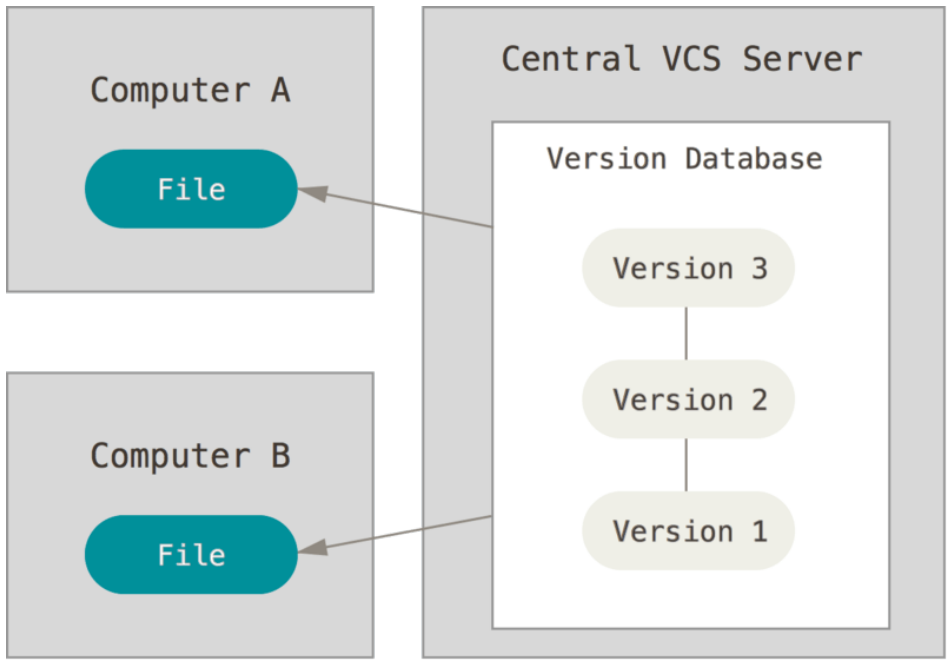
Centralized version control

— Examples:
  - CVS



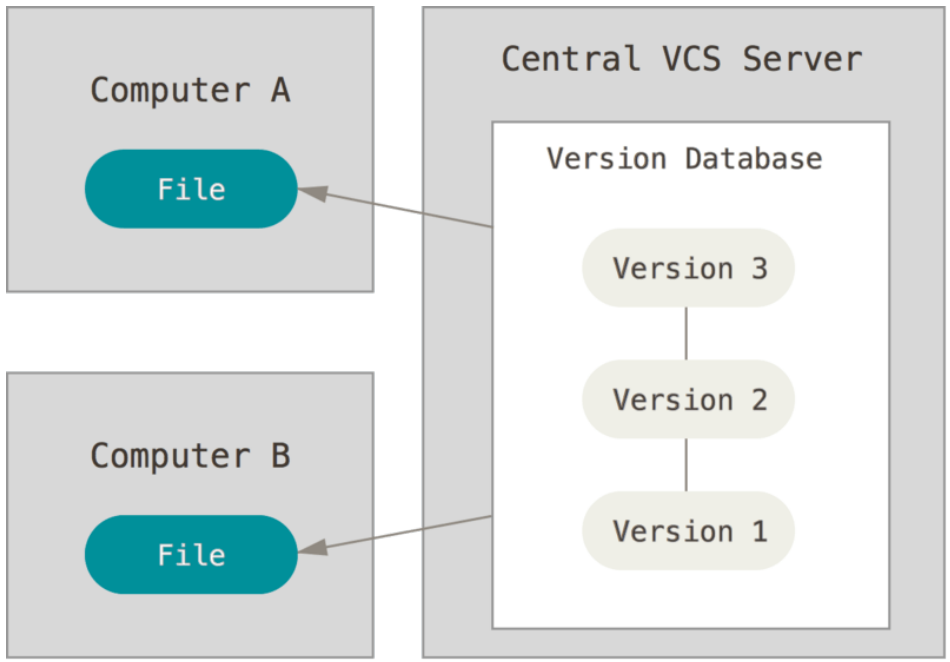
Centralized version control

— Examples:
- CVS
- subversion



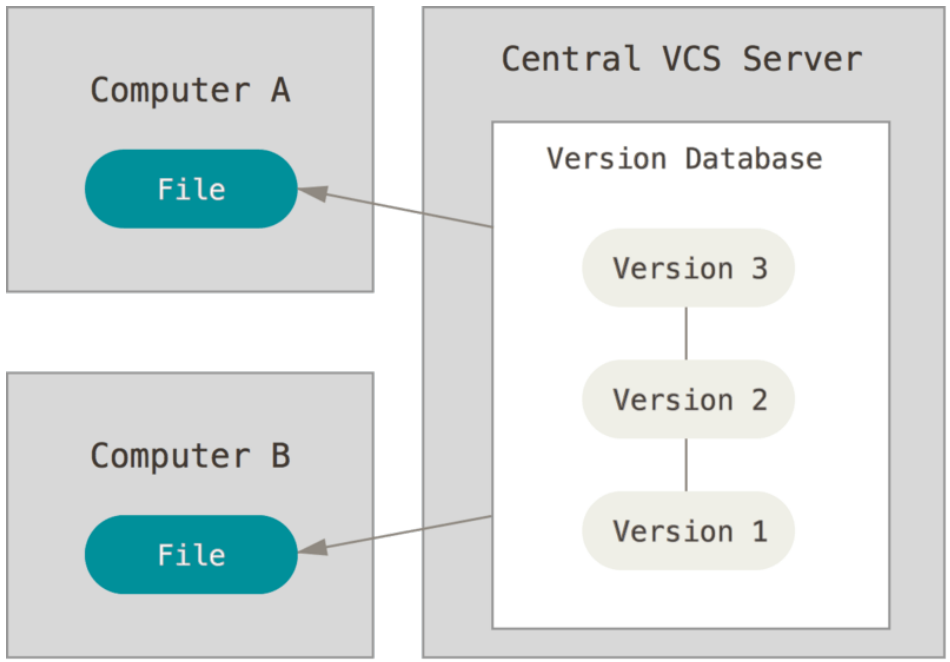
Centralized version control

— Examples:
- CVS
- subversion
- perforce



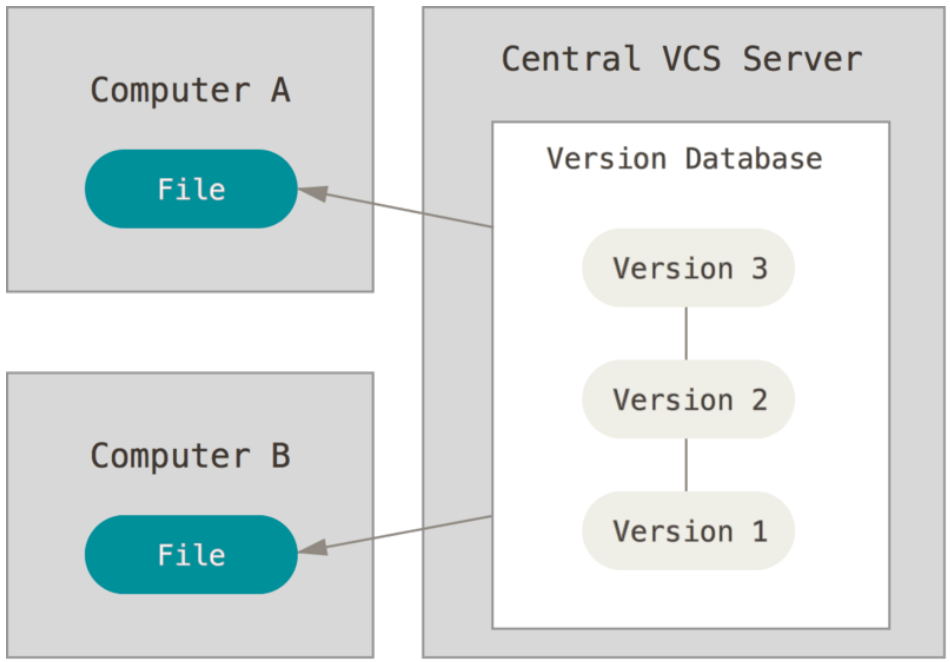
Centralized version control

- Examples:
  - CVS
  - subversion
  - perforce
- Easy to collaborate



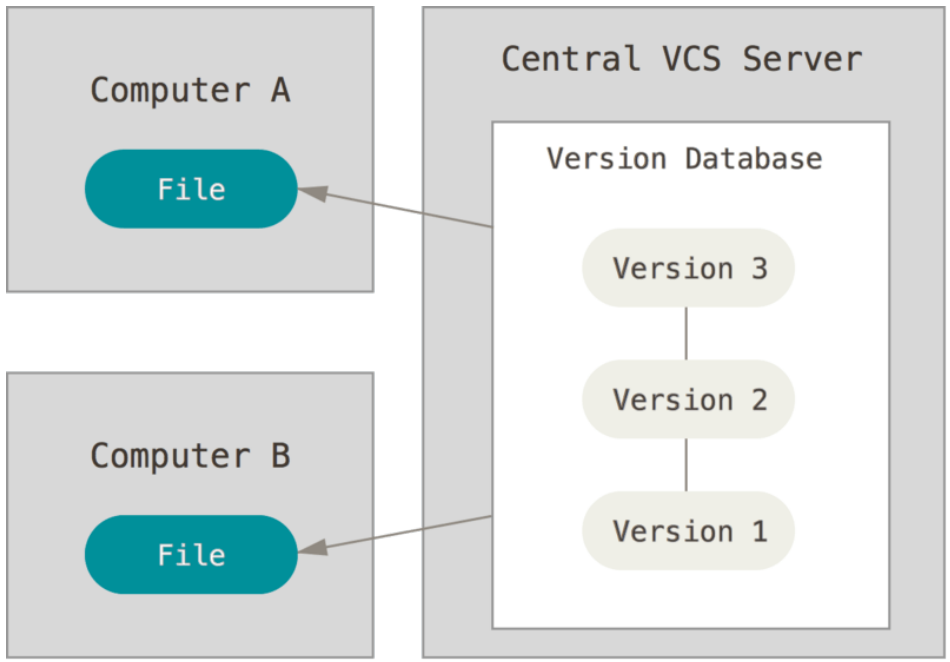
Centralized version control

- Examples:
  - CVS
  - subversion
  - perforce
- Easy to collaborate
- Check out specific versions



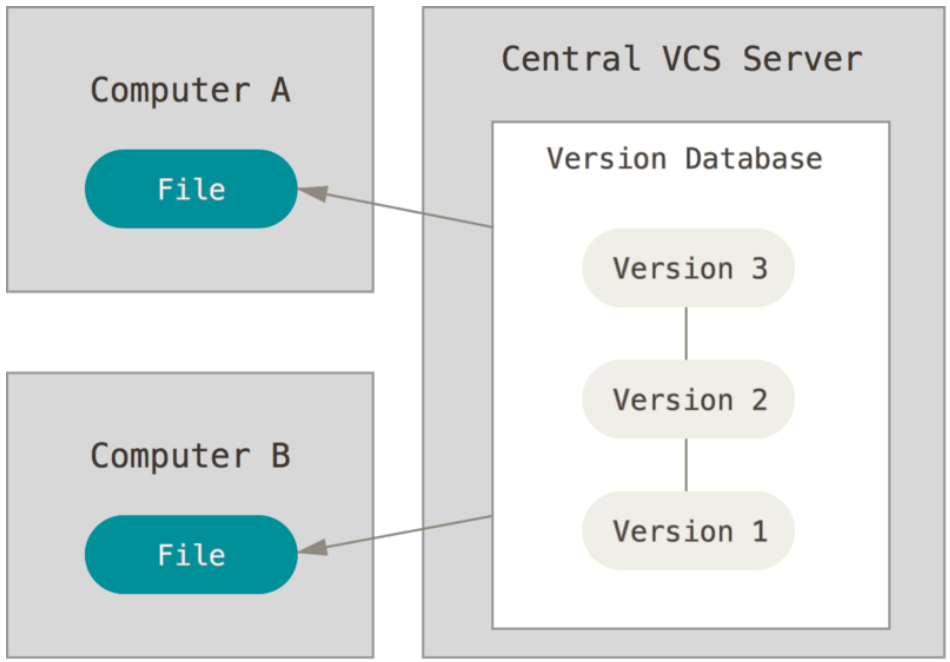
Centralized version control

- Examples:
  - CVS
  - subversion
  - perforce
- Easy to collaborate
- Check out specific versions
- Single point of failure (N-0)



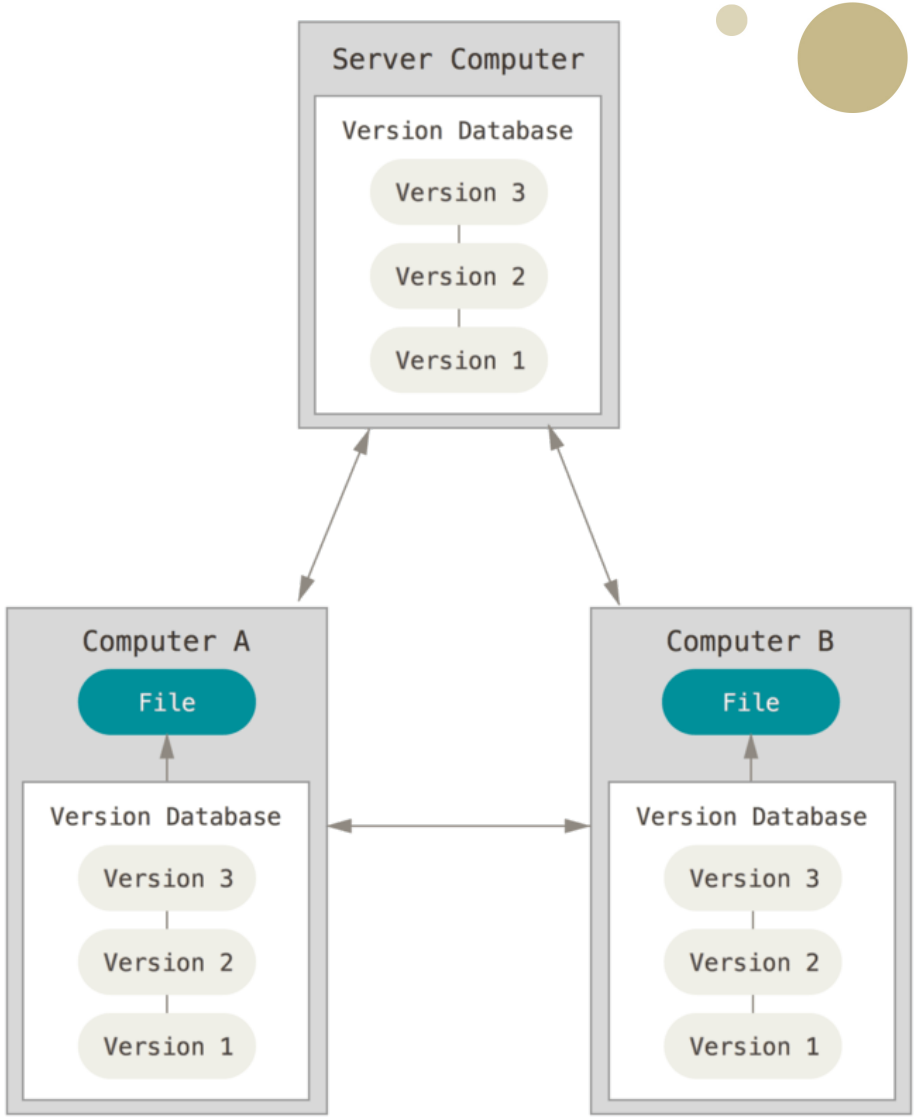
Centralized version control

- Examples:
  - CVS
  - subversion
  - perforce
- Easy to collaborate
- Check out specific versions
- Single point of failure (N-0)
- If the server dies only checked out versions can be saved



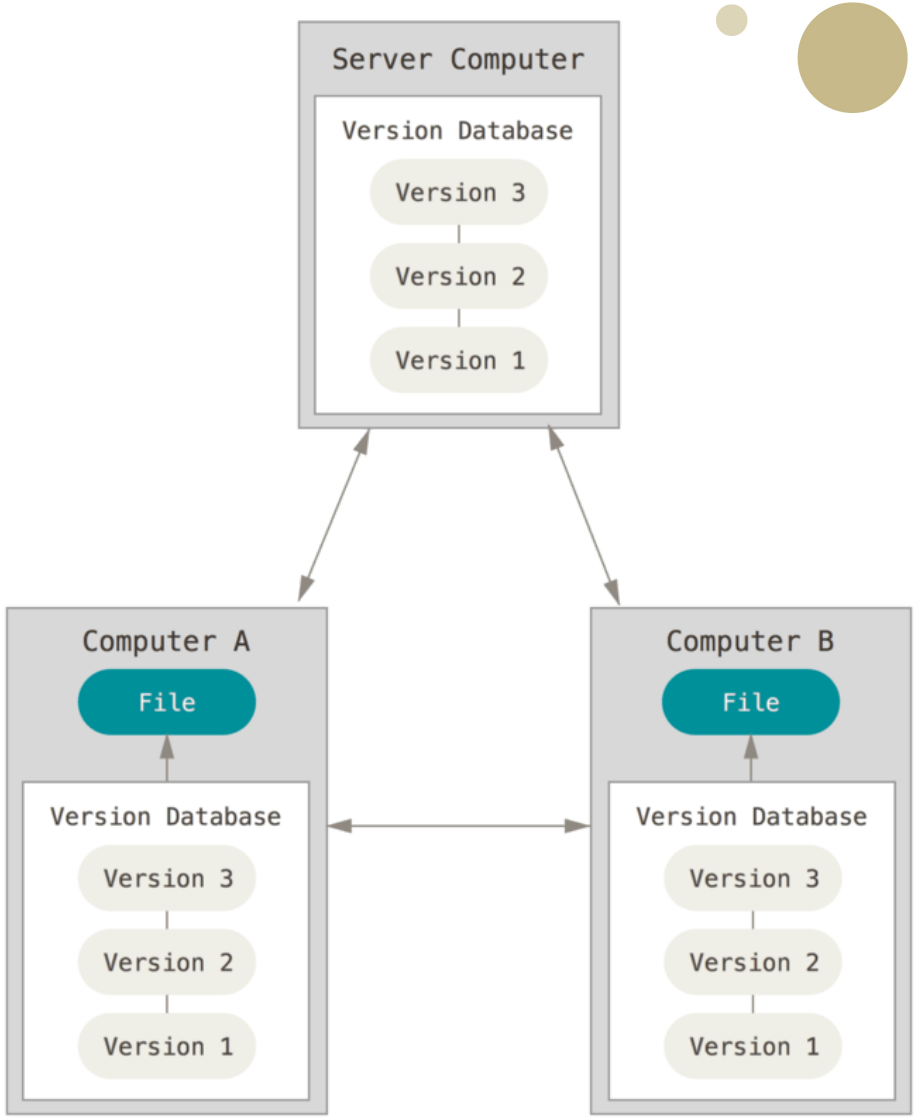
Distsributed version control

— Examples:



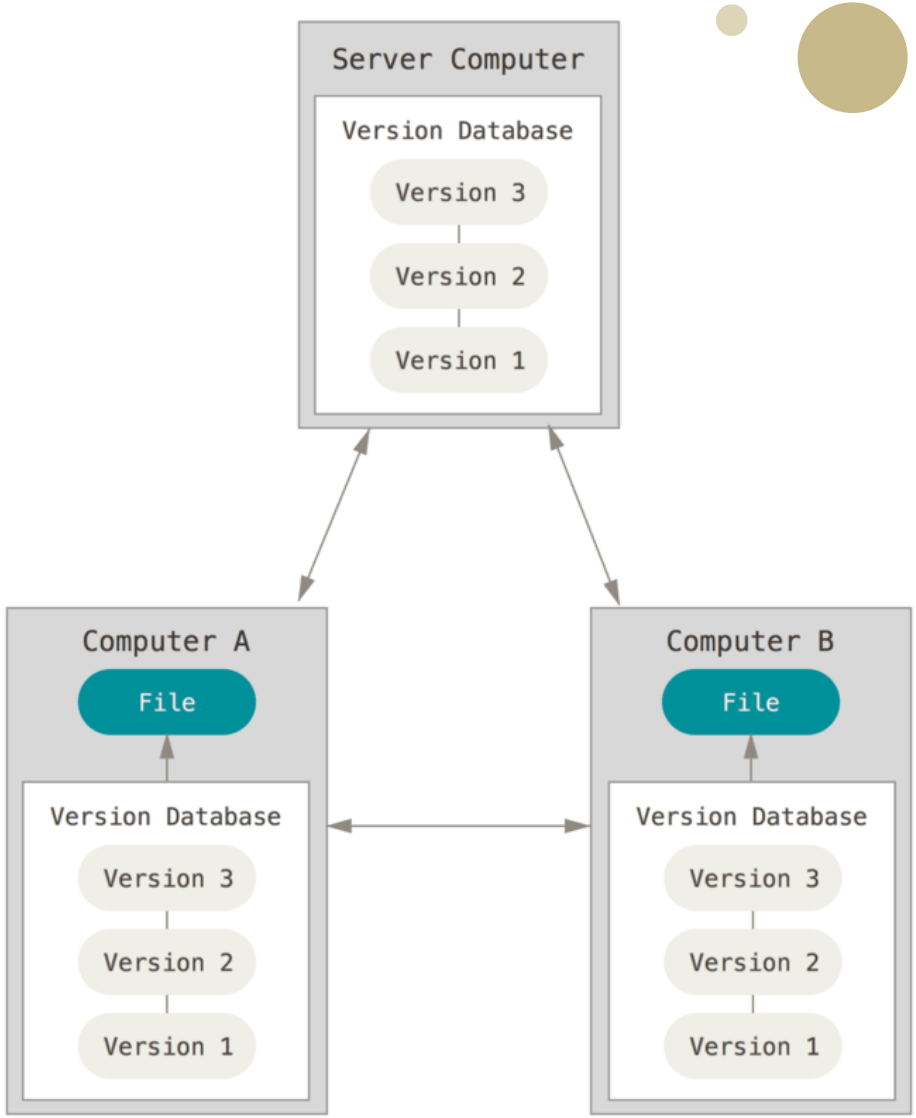
Distsributed version control

— Examples:
- Git



Distsributed version control

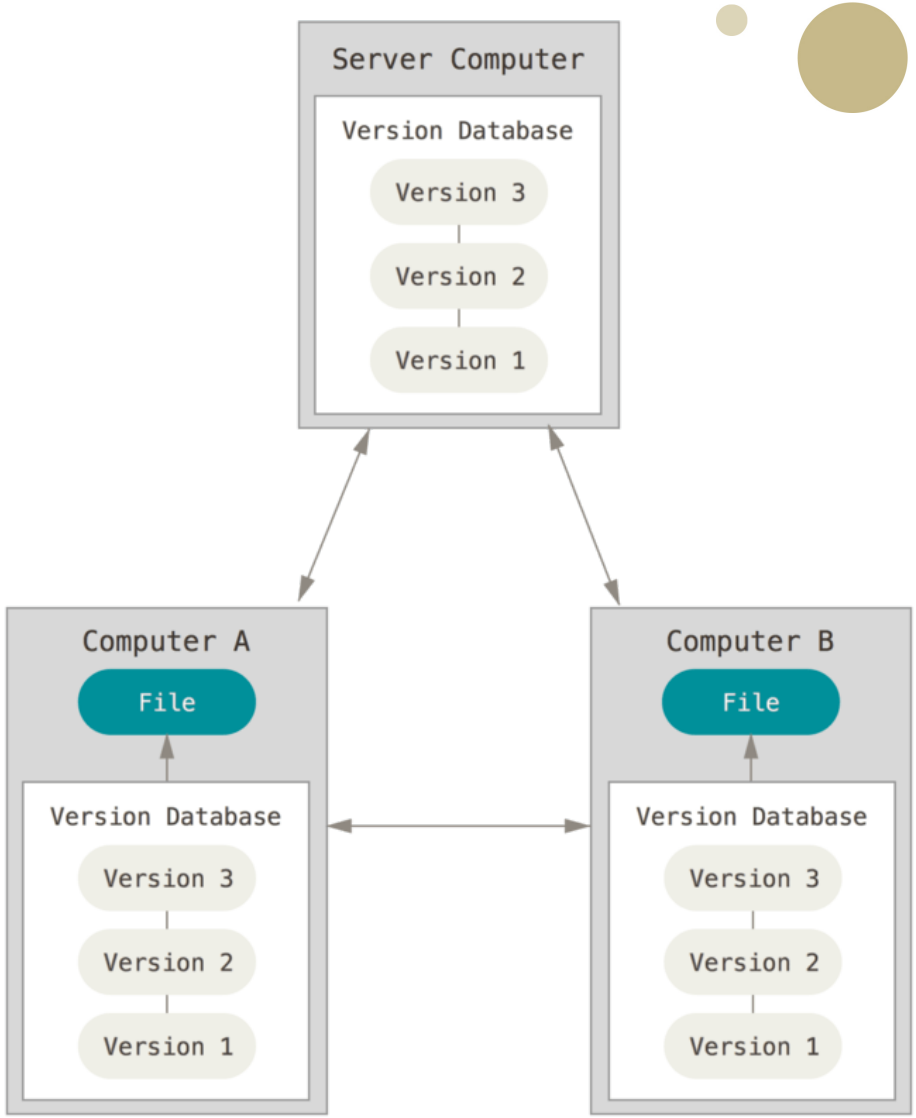
— Examples:
- Git
- Mercurial



Distsributed version control

— Examples:

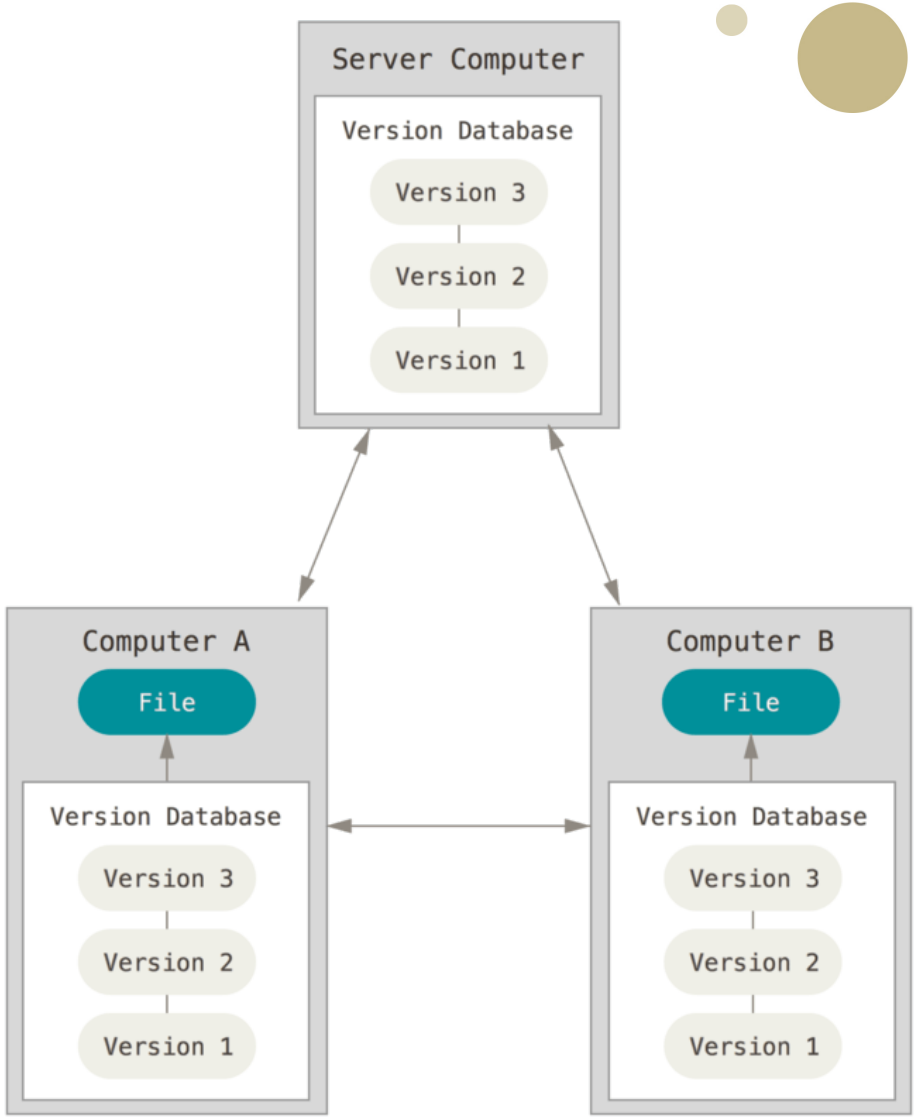
- Git
- Mercurial
- Bazaar



Distsributed version control

— Examples:

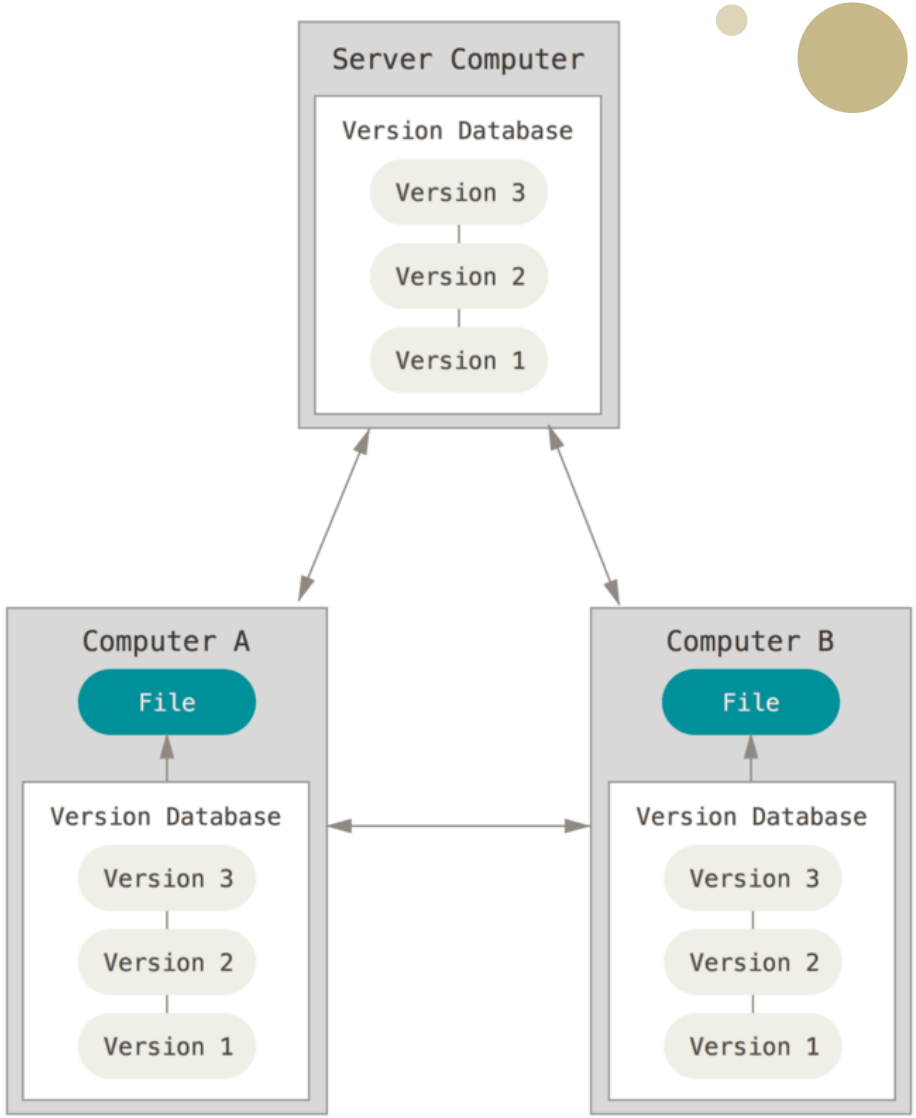
- Git
- Mercurial
- Bazaar
- Darcs



Distsributed version control

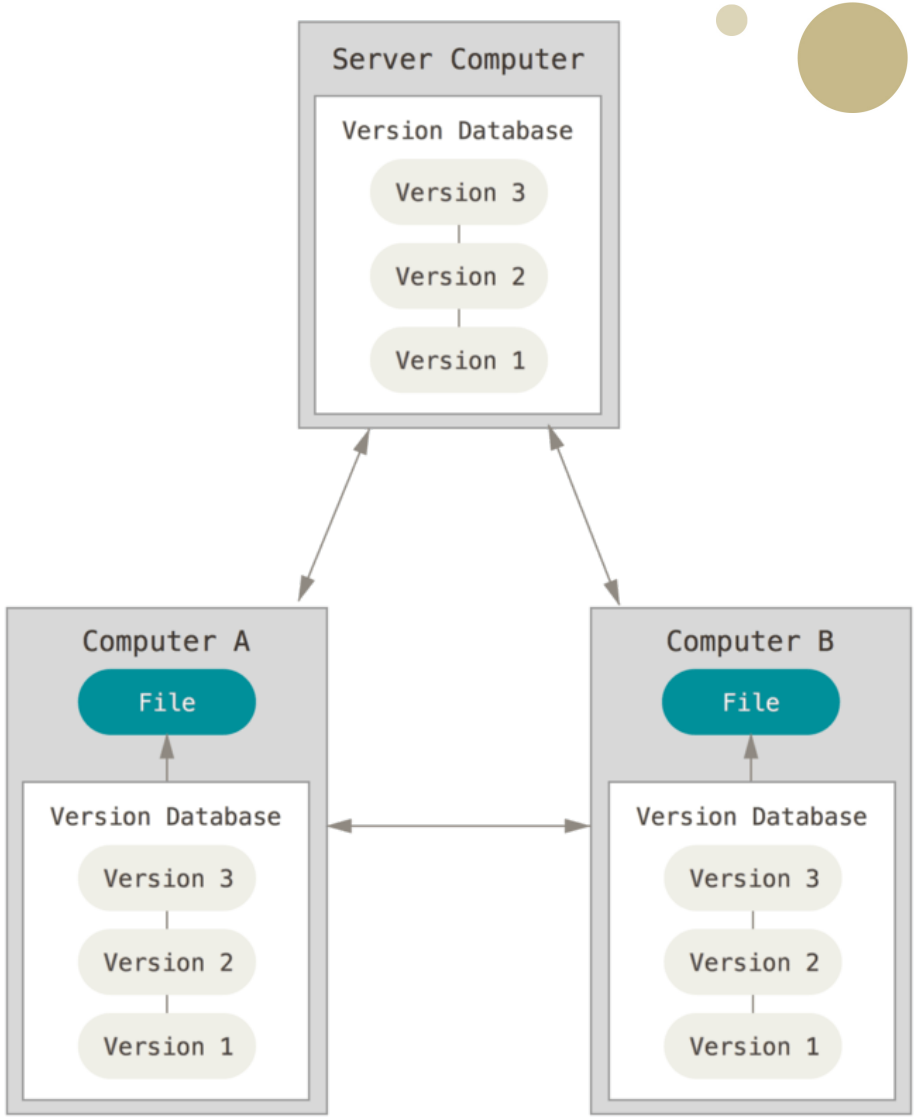
— Examples:
  - Git
  - Mercurial
  - Bazaar
  - Darcs

— Same advantages as centralized version control



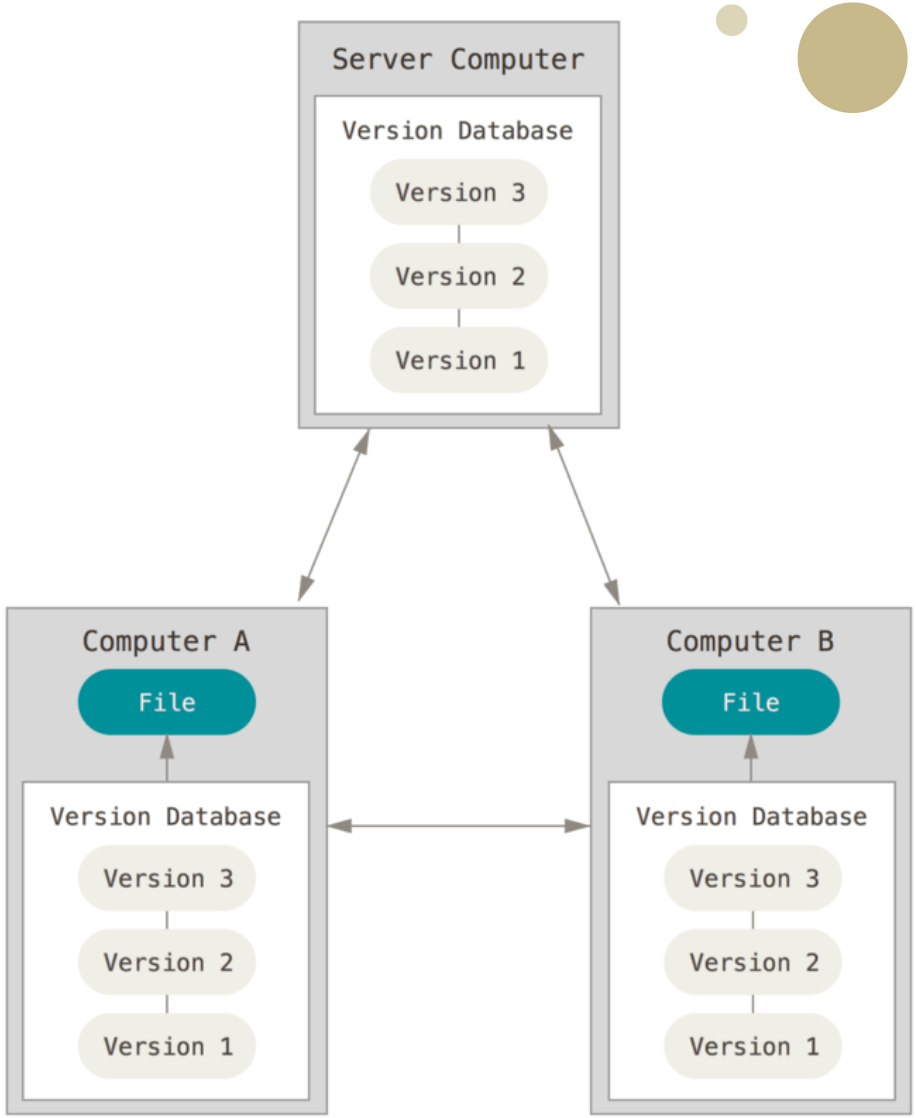
Distsributed version control

- Examples:
  - Git
  - Mercurial
  - Bazaar
  - Darcs
- Same advantages as centralized version control
- All users can reconstruct the project



Distsributed version control

- Examples:
  - Git
  - Mercurial
  - Bazaar
  - Darcs
- Same advantages as centralized version control
- All users can reconstruct the project
- Easy to work against multiple servers



Should you use version control?

— Do you write code?

Dont do this:

```
% Read data set 1
data=read_data('data1.csv
    ');
df = process_data(data,
    0.02, 1);
plot(df);

% Read data set 1
data=read_data('data2.csv
    ');
df = process_data(data,
    0.02, 1);
plot(df);
```

Should you use version control?

— Do you write code?
  - If yes: use version control

Dont do this:

```
% Read data set 1
data=read_data('data1.csv
    ');
df = process_data(data,
    0.02, 1);
plot(df);

% Read data set 1
data=read_data('data2.csv
    ');
df = process_data(data,
    0.02, 1);
plot(df);
```

Should you use version control?

- Do you write code?
  - If yes: use version control
- Never copy code -_-+

Dont do this:

```
% Read data set 1
data=read_data('data1.csv
    ');
df = process_data(data,
    0.02, 1);
plot(df);

% Read data set 1
data=read_data('data2.csv
    ');
df = process_data(data,
    0.02, 1);
plot(df);
```

Should you use version control?

- Do you write code?
  - If yes: use version control
- Never copy code -_-+
- Write functions and keep them version controlled

Dont do this:

```
% Read data set 1
data=read_data('data1.csv
    ');
df = process_data(data,
    0.02, 1);
plot(df);

% Read data set 1
data=read_data('data2.csv
    ');
df = process_data(data,
    0.02, 1);
plot(df);
```

Should you use version control?

- Do you write code?
  - If yes: use version control
- Never copy code -_-+
- Write functions and keep them version controlled
- Remember code is a virtual laboratory

Dont do this:

```
% Read data set 1
data=read_data('data1.csv
    ');
df = process_data(data,
    0.02, 1);
plot(df);

% Read data set 1
data=read_data('data2.csv
    ');
df = process_data(data,
    0.02, 1);
plot(df);
```

Should you use version control?

- Do you write code?
  - If yes: use version control
- Never copy code -_-+
- Write functions and keep them version controlled
- Remember code is a virtual laboratory
- Versions can for instance be tagged with name of publication(reproducability for review)

Dont do this:

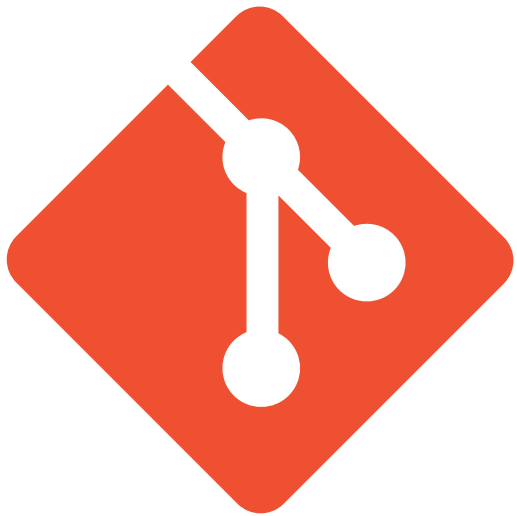
```
% Read data set 1
data=read_data('data1.csv
    ');
df = process_data(data,
    0.02, 1);
plot(df);

% Read data set 1
data=read_data('data2.csv
    ');
df = process_data(data,
    0.02, 1);
plot(df);
```

Outline

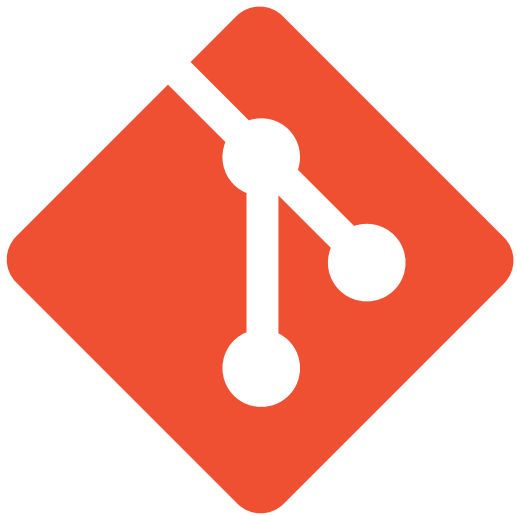
About git

— Developed by the team behind Linux



About git

— Developed by the team behind Linux
— used by companies such as:



About git

— Developed by the team behind Linux
— used by companies such as:
  - Linux

About git

— Developed by the team behind Linux
— used by companies such as:
  - Linux
  - Microsoft

About git

— Developed by the team behind Linux
— used by companies such as:
  - Linux
  - Microsoft
  - Google

About git

— Developed by the team behind Linux
— used by companies such as:
  - Linux
  - Microsoft
  - Google
  - Android

About git

— Developed by the team behind Linux
— used by companies such as:
  - Linux
  - Microsoft
  - Google
  - Android
  - Facebook

About git

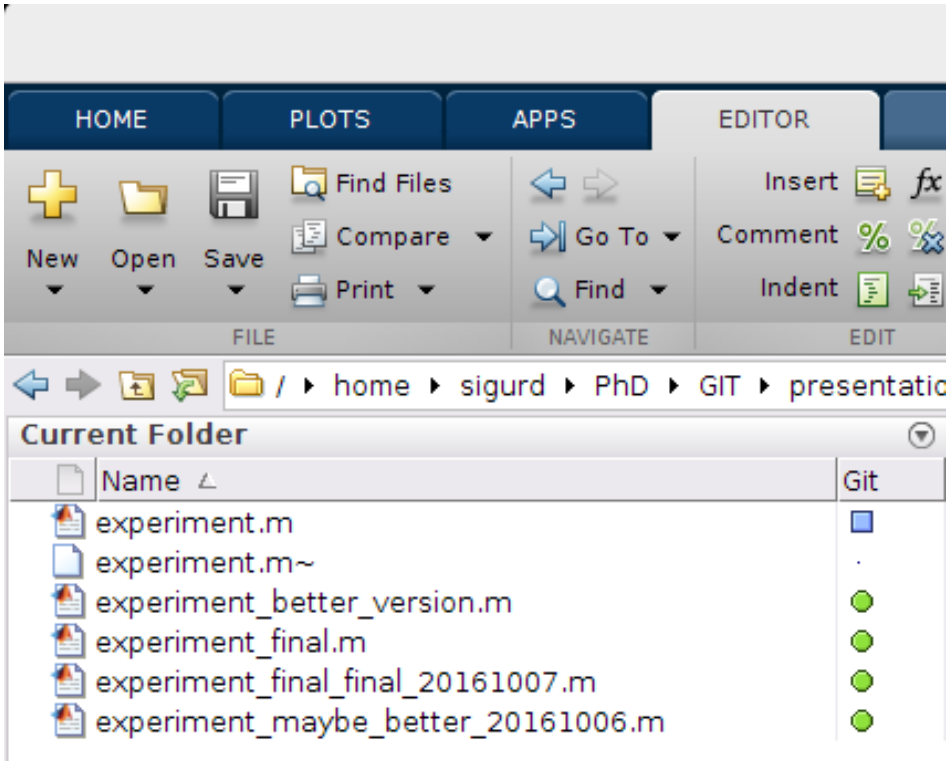
— Developed by the team behind Linux
— used by companies such as:
  - Linux
  - Microsoft
  - Google
  - Android
  - Facebook
  - Twitter

About git

— Developed by the team behind Linux
— used by companies such as:
  - Linux
  - Microsoft
  - Google
  - Android
  - Facebook
  - Twitter
  - Linkedin

Getting started with git

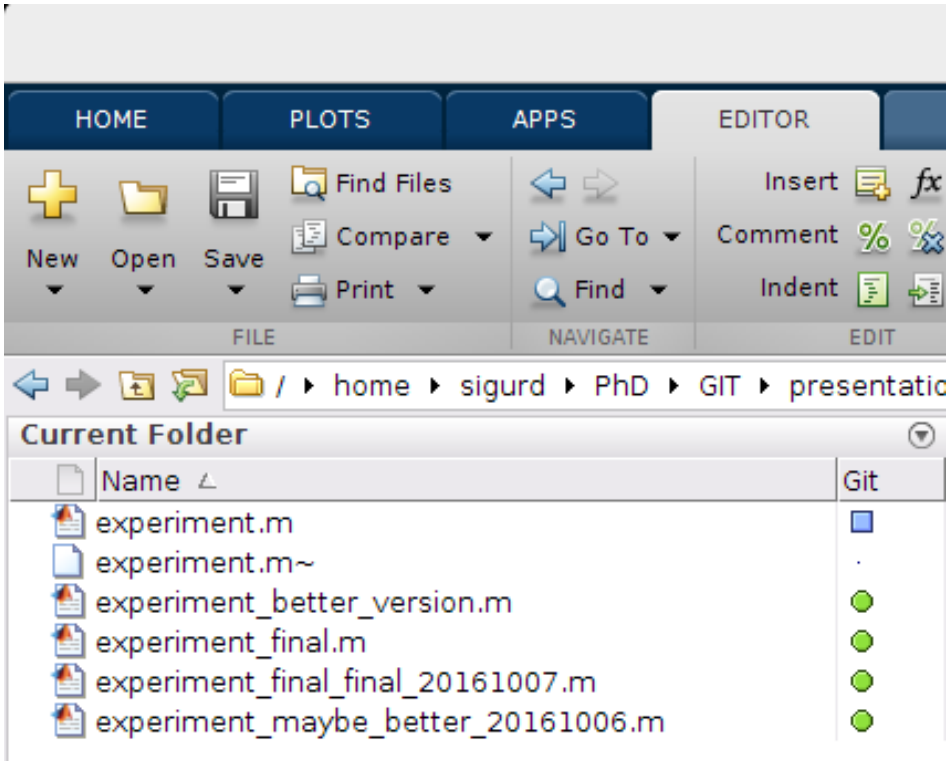
— Download git



Getting started with git

— Download git

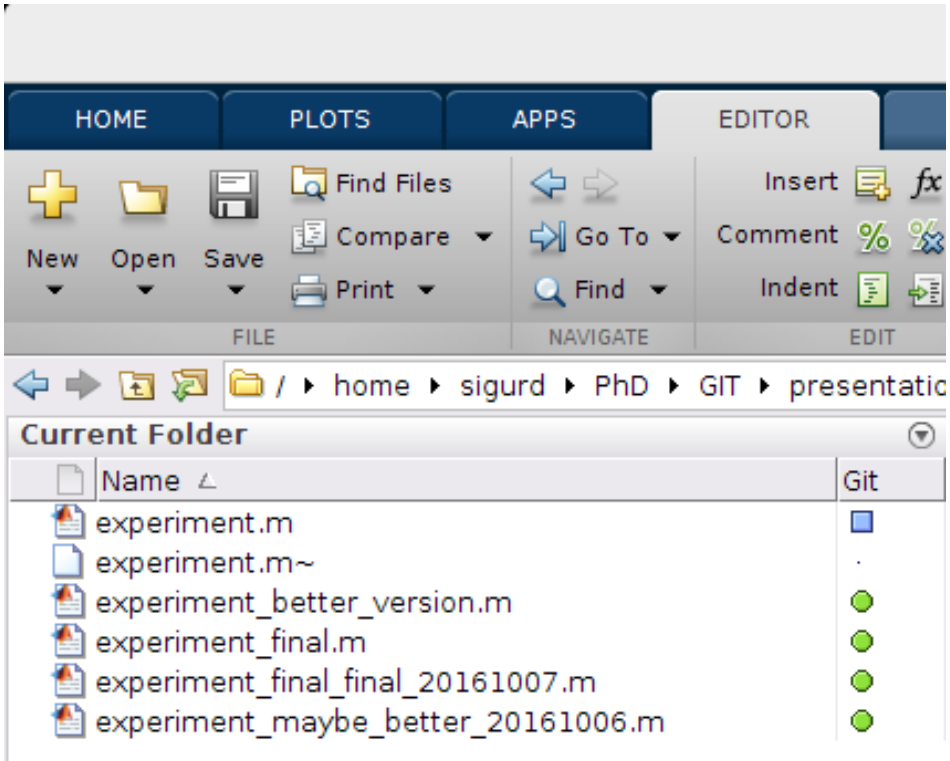
- `www.git-scm.com`



Getting started with git

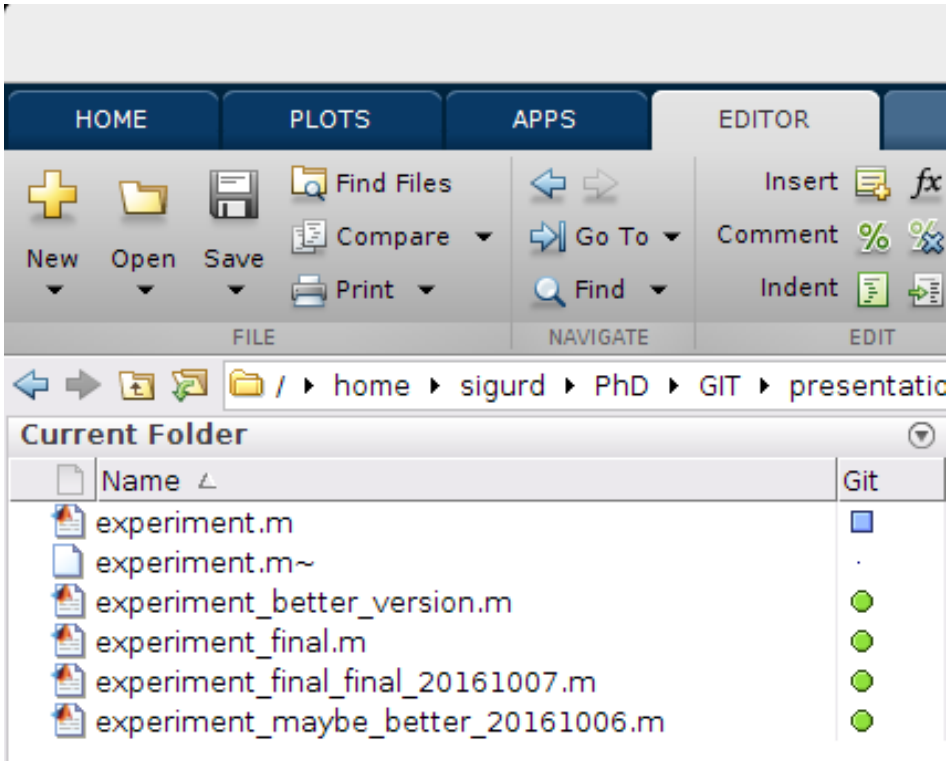
— Download git

- `www.git-scm.com`
- Your favourite package manager



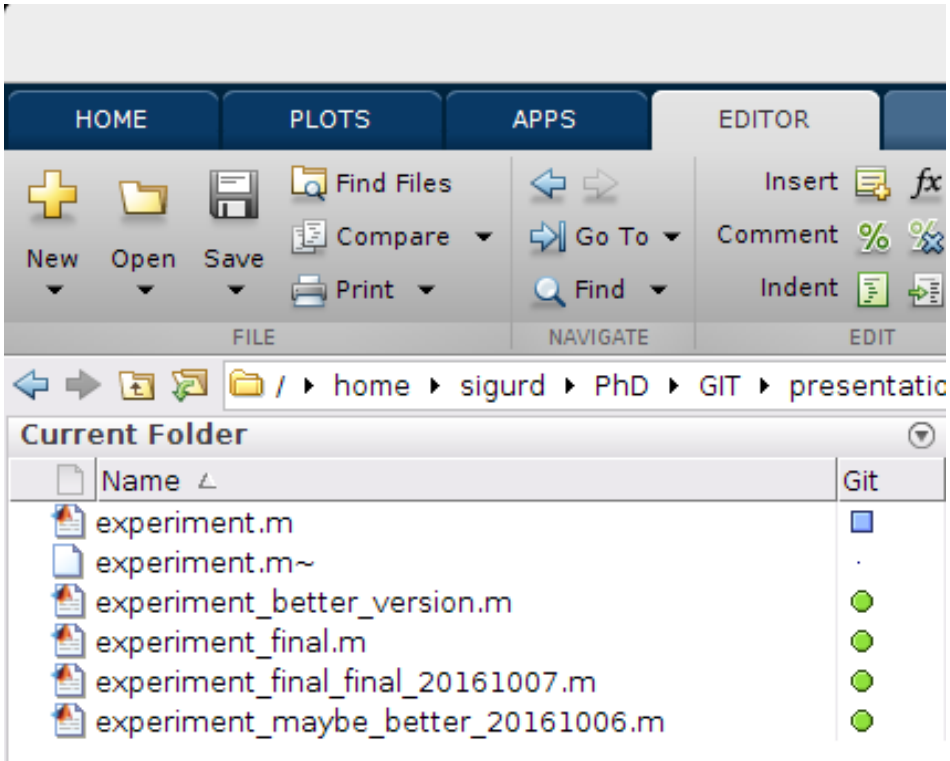
Getting started with git

- Download git
  - `www.git-scm.com`
  - Your favourite package manager
- Get a server:



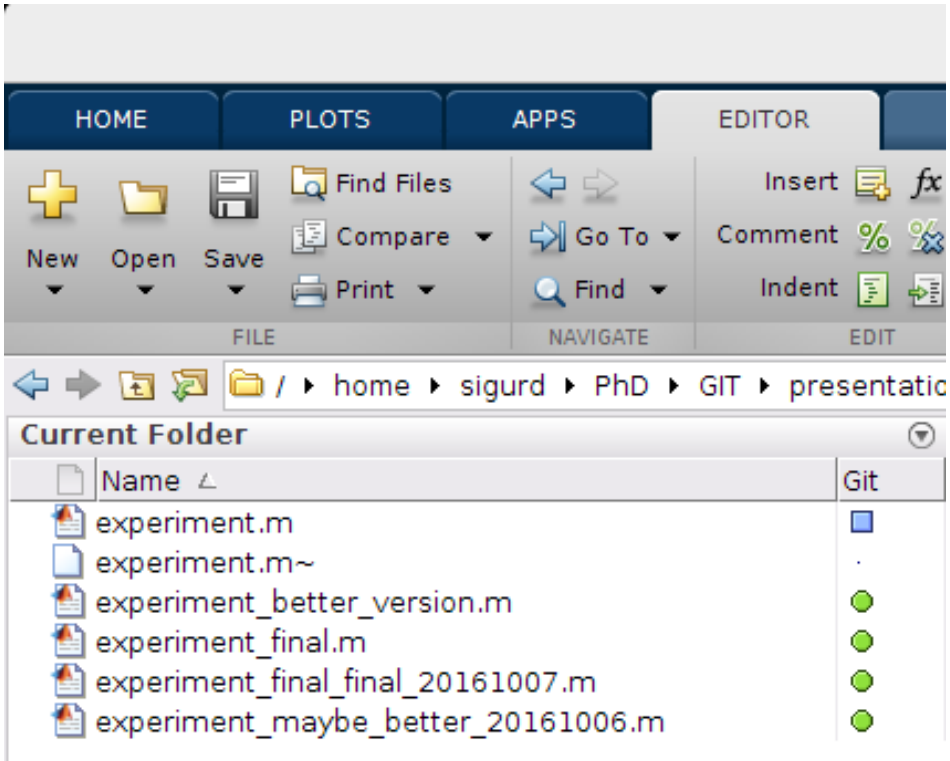
Getting started with git

- Download git
  - `www.git-scm.com`
  - Your favourite package manager
- Get a server:
  - Set up yourself



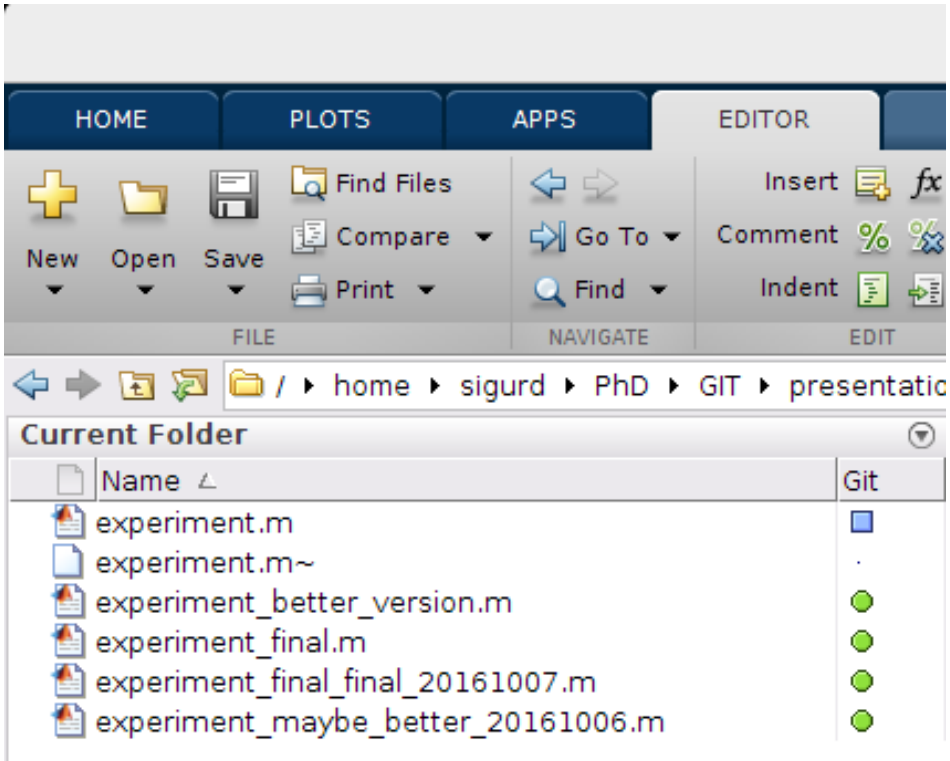
Getting started with git

- Download git
  - `www.git-scm.com`
  - Your favourite package manager
- Get a server:
  - Set up yourself
  - GitHub



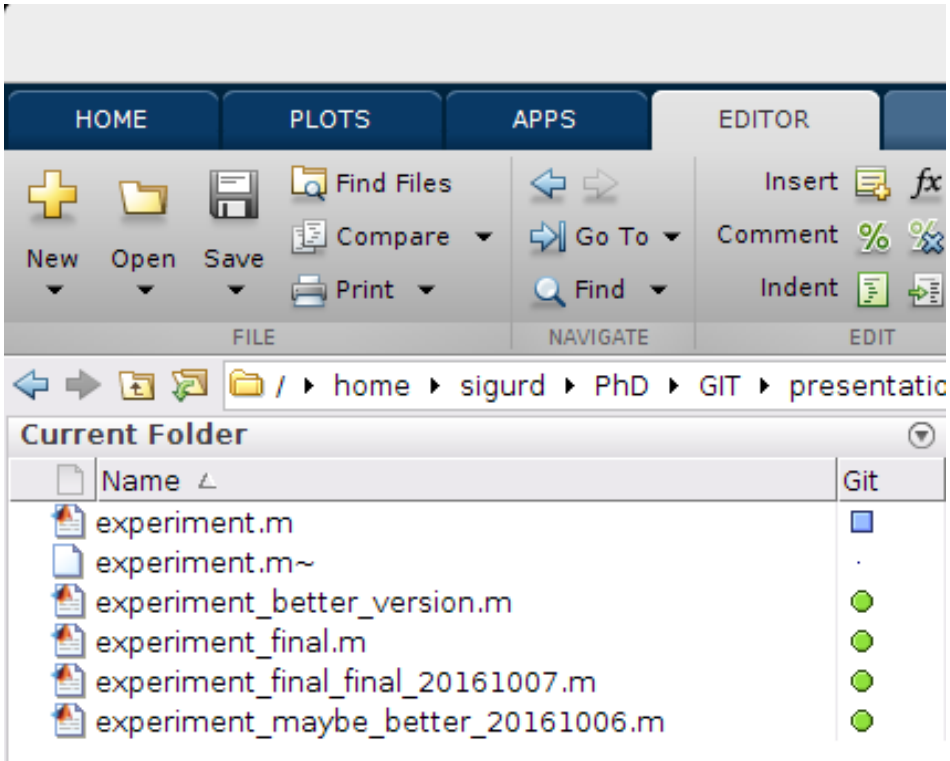
Getting started with git

- Download git
  - `www.git-scm.com`
  - Your favourite package manager
- Get a server:
  - Set up yourself
  - GitHub
  - BitBucket



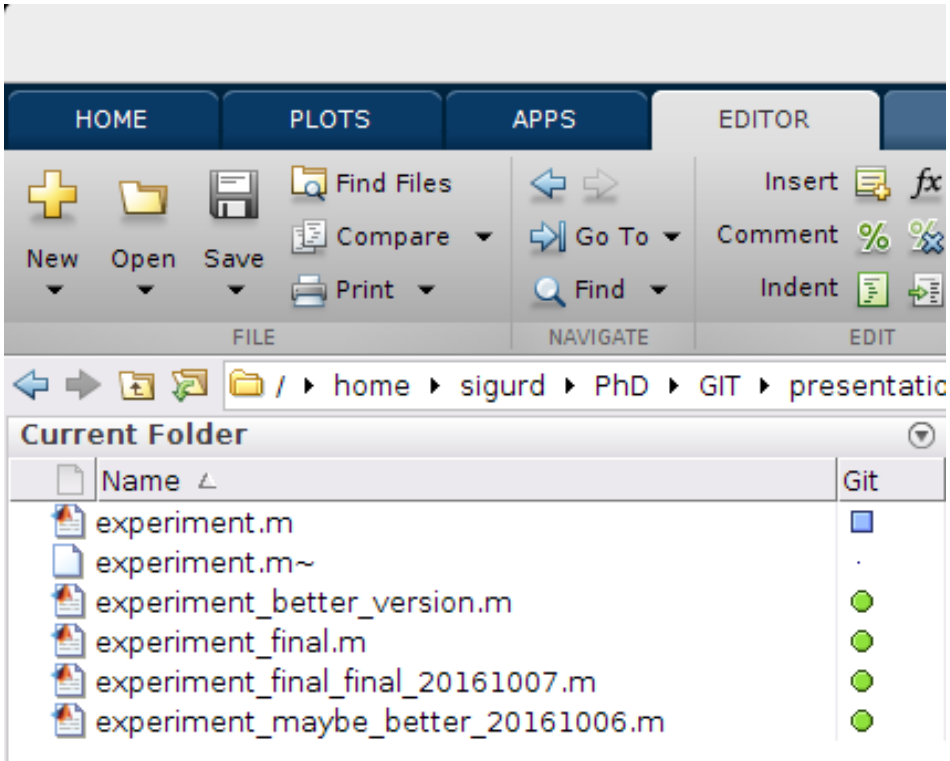
Getting started with git

- Download git
  - `www.git-scm.com`
  - Your favourite package manager
- Get a server:
  - Set up yourself
  - GitHub
  - BitBucket
- Get a client:



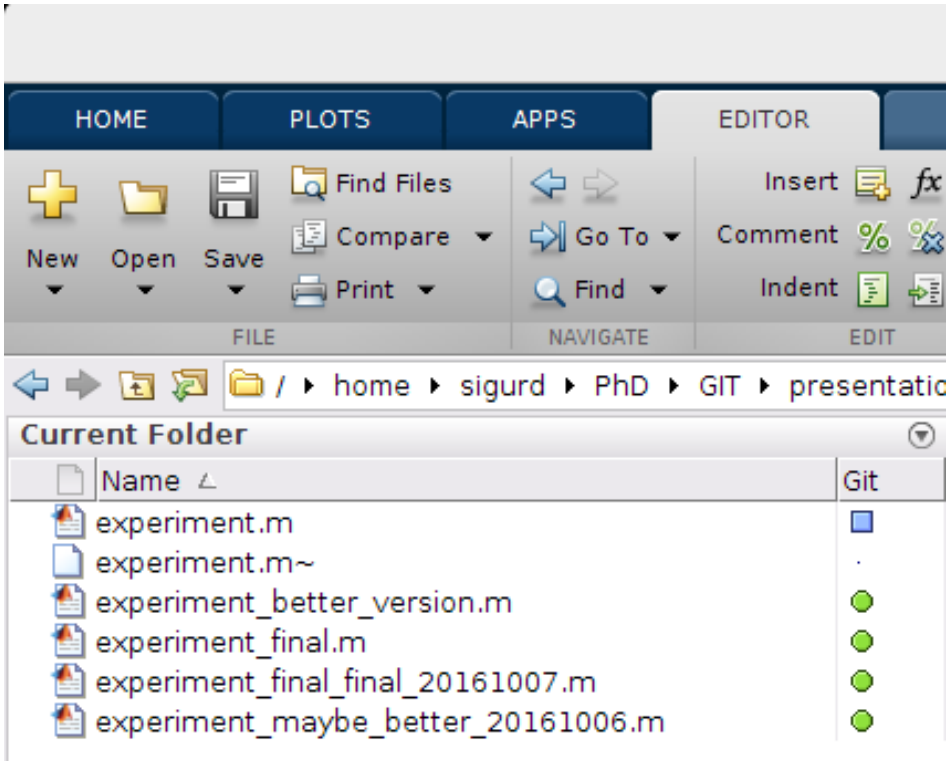
Getting started with git

- Download git
  - `www.git-scm.com`
  - Your favourite package manager
- Get a server:
  - Set up yourself
  - GitHub
  - BitBucket
- Get a client:
  - Command line is already included



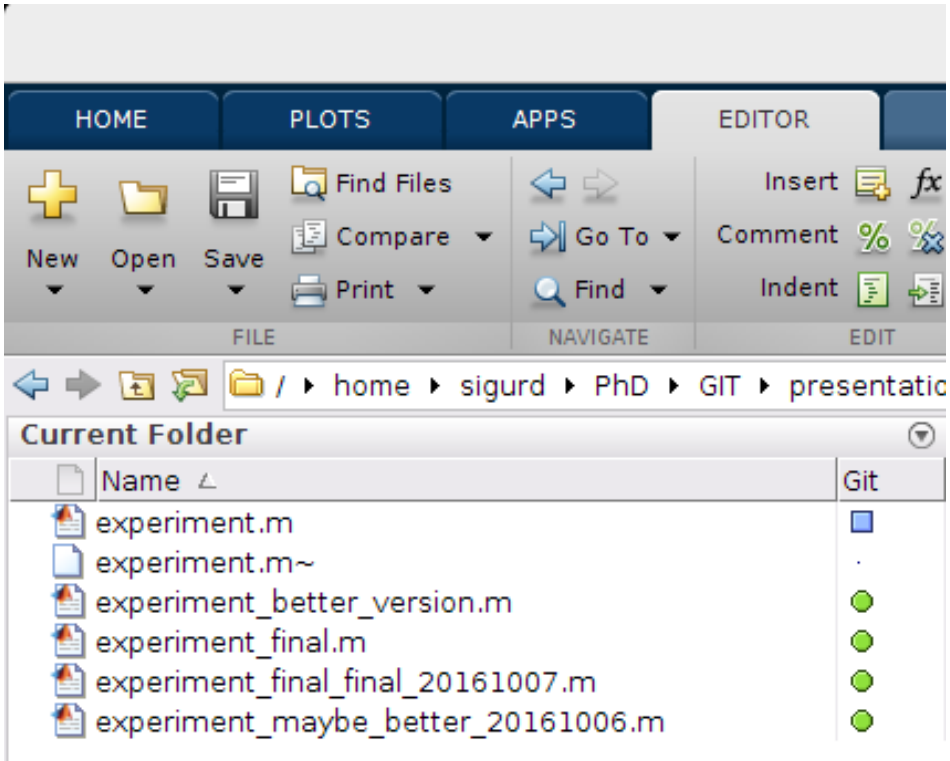
Getting started with git

- Download git
  - `www.git-scm.com`
  - Your favourite package manager
- Get a server:
  - Set up yourself
  - GitHub
  - BitBucket
- Get a client:
  - Command line is already included
  - GitKraken



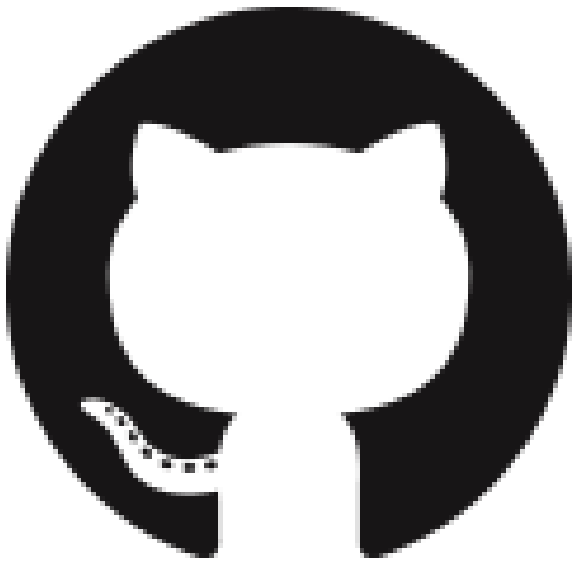
Getting started with git

- Download git
  - `www.git-scm.com`
  - Your favourite package manager
- Get a server:
  - Set up yourself
  - GitHub
  - BitBucket
- Get a client:
  - Command line is already included
  - GitKraken
  - Check out list at `www.git-scm.com/downloads/guis`



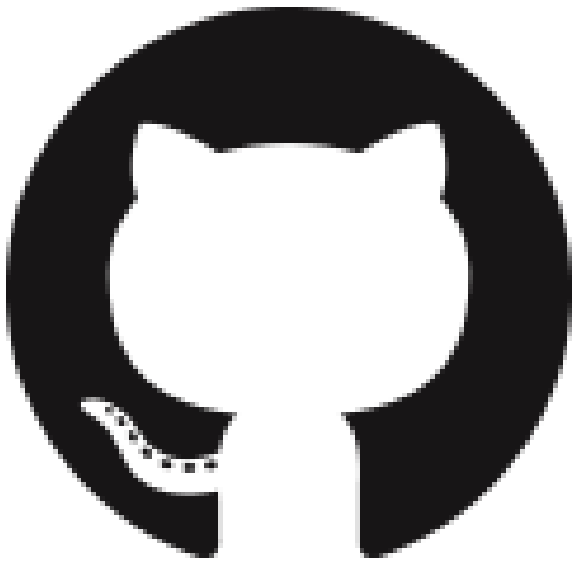
GitHub

— Largest host for git repositories



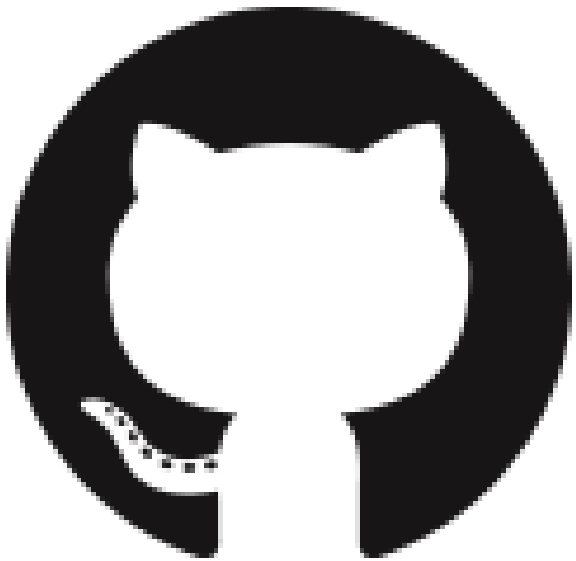
GitHub

- Largest host for git repositories
- For developers a git page is sometimes as important as a CV



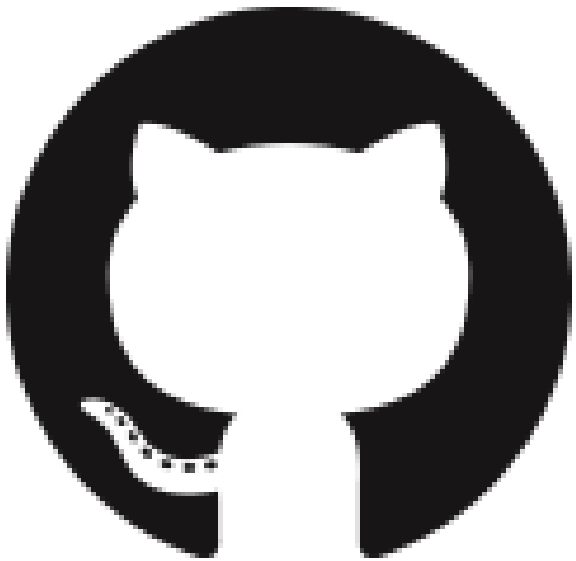
GitHub

- Largest host for git repositories
- For developers a git page is sometimes as important as a CV
- You can have as many open repositories you want free.



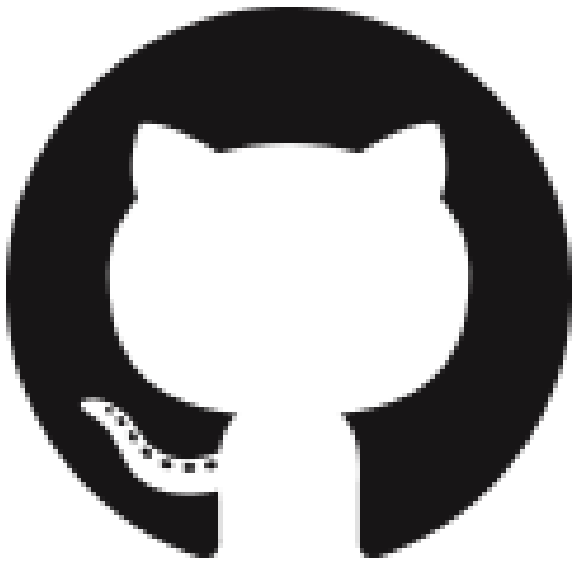
GitHub

- Largest host for git repositories
- For developers a git page is sometimes as important as a CV
- You can have as many open repositories you want free.
- Closed source projects cost money



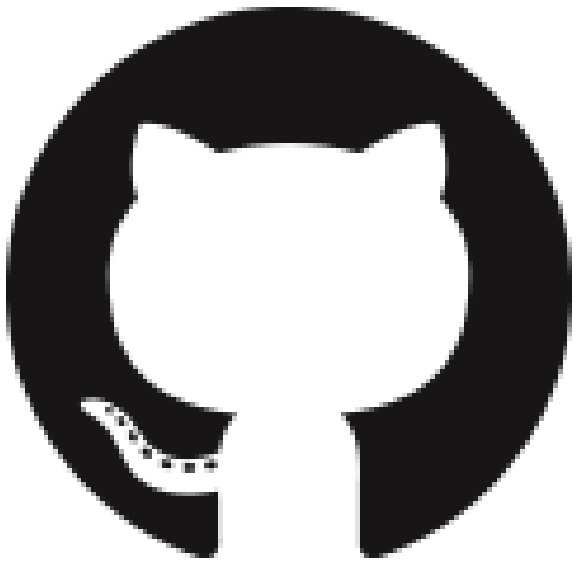
GitHub

- Largest host for git repositories
- For developers a git page is sometimes as important as a CV
- You can have as many open repositories you want free.
- Closed source projects cost money
- You can request an academic account at `https://education.github.com/`



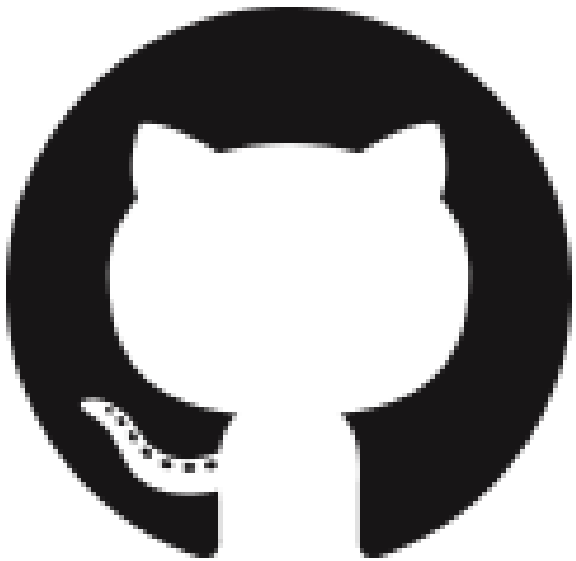
GitHub

- Largest host for git repositories
- For developers a git page is sometimes as important as a CV
- You can have as many open repositories you want free.
- Closed source projects cost money
- You can request an academic account at `https://education.github.com/`
  - You get five closed source projects for free



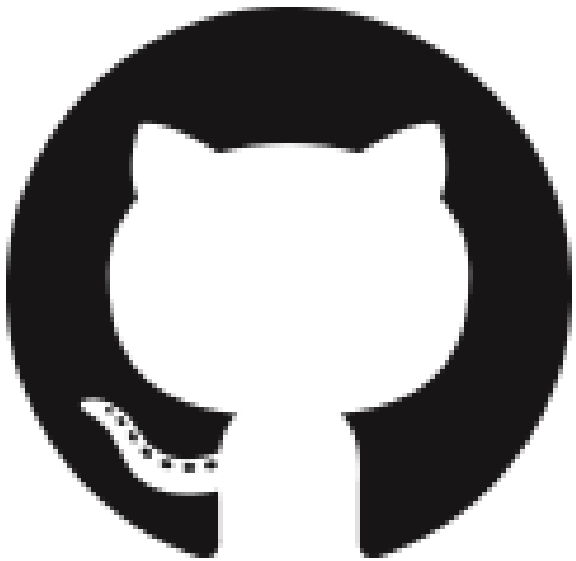
GitHub

- Largest host for git repositories
- For developers a git page is sometimes as important as a CV
- You can have as many open repositories you want free.
- Closed source projects cost money
- You can request an academic account at `https://education.github.com/`
  - You get five closed source projects for free
  - Maybe other stuff too



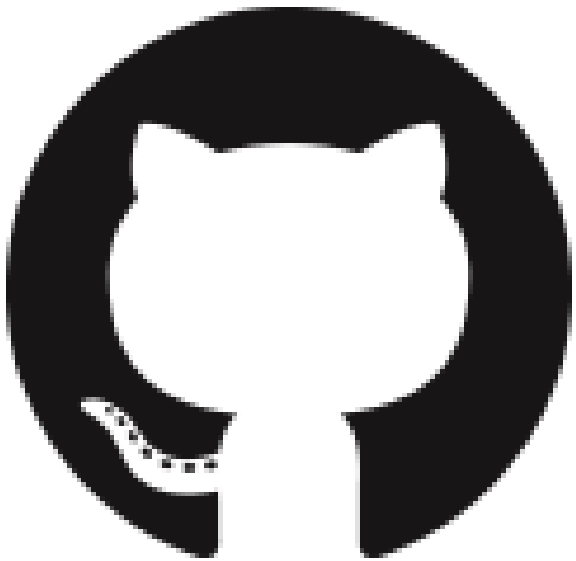
GitHub

- Largest host for git repositories
- For developers a git page is sometimes as important as a CV
- You can have as many open repositories you want free.
- Closed source projects cost money
- You can request an academic account at `https://education.github.com/`
  - You get five closed source projects for free
  - Maybe other stuff too
- Wiki for your project



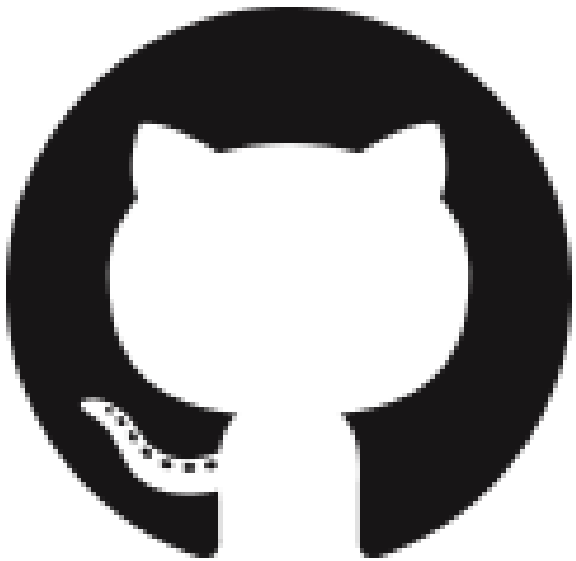
GitHub

- Largest host for git repositories
- For developers a git page is sometimes as important as a CV
- You can have as many open repositories you want free.
- Closed source projects cost money
- You can request an academic account at `https://education.github.com/`
  - You get five closed source projects for free
  - Maybe other stuff too
- Wiki for your project
- Place to discuss your project and fill bug reports



GitHub

- Largest host for git repositories
- For developers a git page is sometimes as important as a CV
- You can have as many open repositories you want free.
- Closed source projects cost money
- You can request an academic account at `https://education.github.com/`
  - You get five closed source projects for free
  - Maybe other stuff too
- Wiki for your project
- Place to discuss your project and fill bug reports
- Integrates with many cool services



BitBucket

— More or less same features as GitHub

BitBucket

— More or less same features as GitHub
— Wiki, bug reporting etc. different integrated services

BitBucket

— More or less same features as GitHub
— Wiki, bug reporting etc. different integrated services
— Free closed source repositories

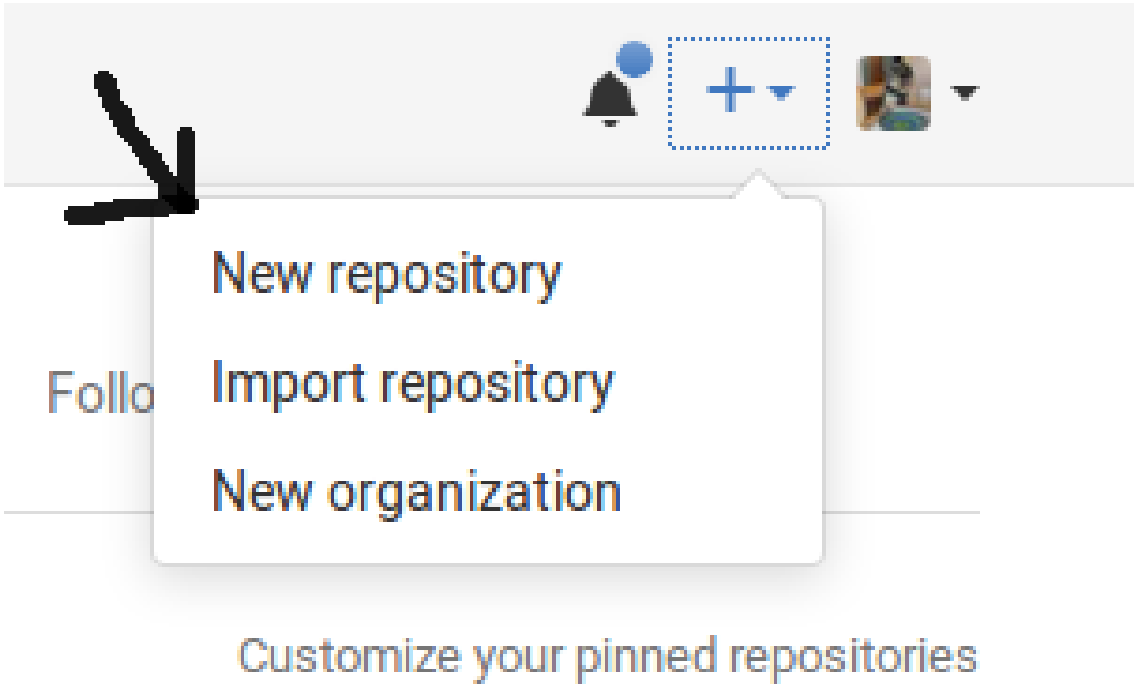
BitBucket

- More or less same features as GitHub
- Wiki, bug reporting etc. different integrated services
- Free closed source repositories
- With git it is easy to change the remote, try both GitHub and BitBucket

Outline

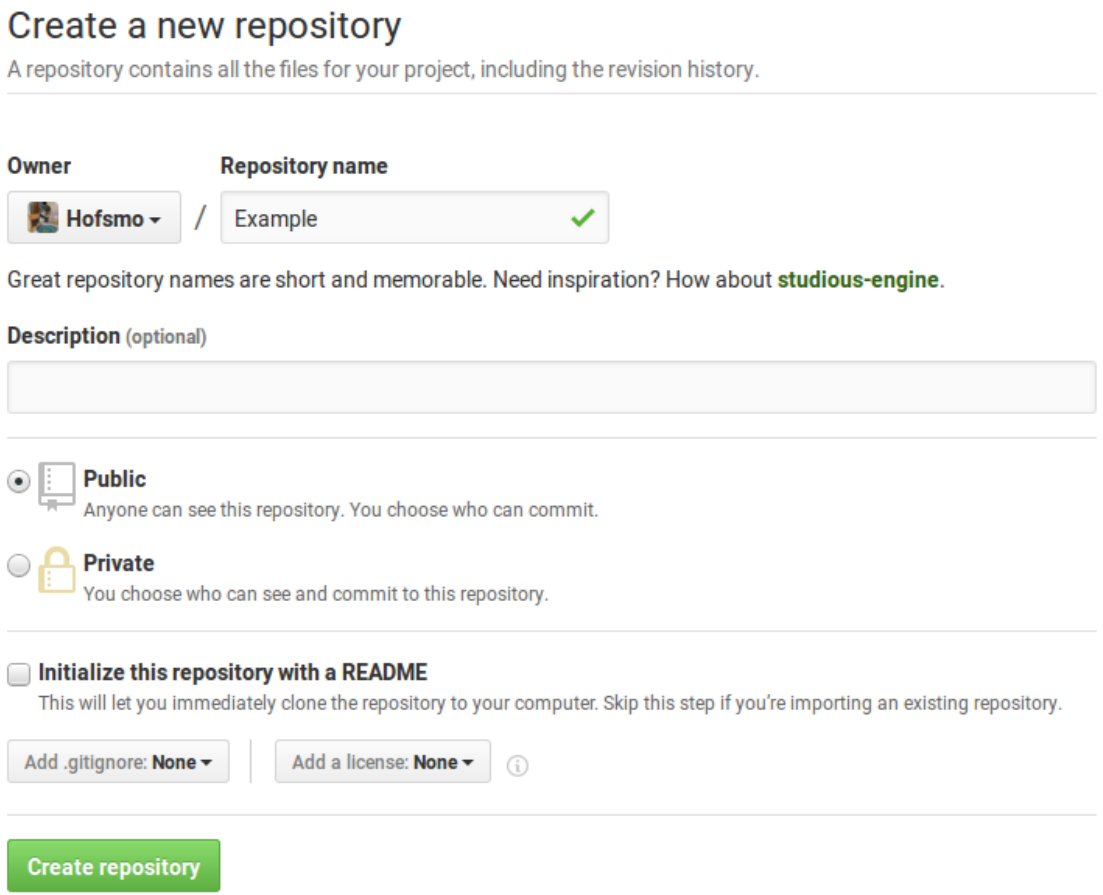
Creating a git repository

— New repository button



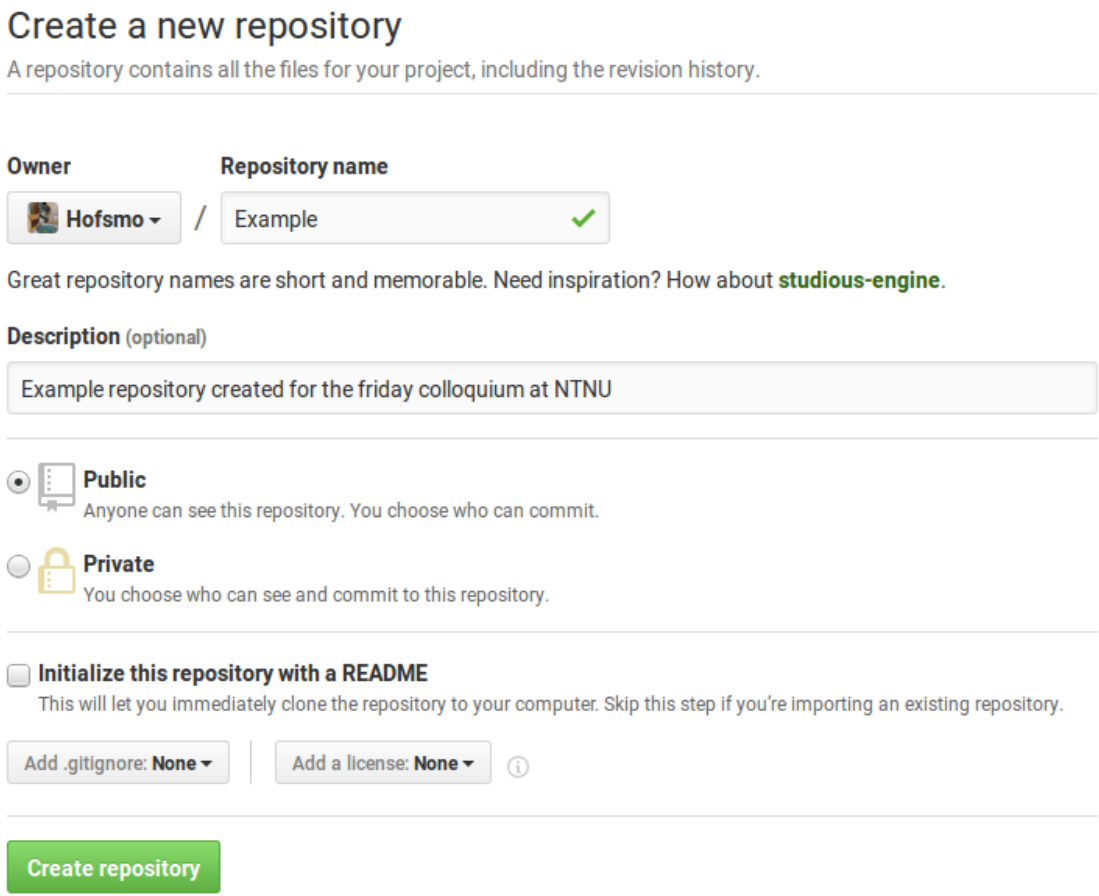
Creating a git repository

- New repository button
- Create the name



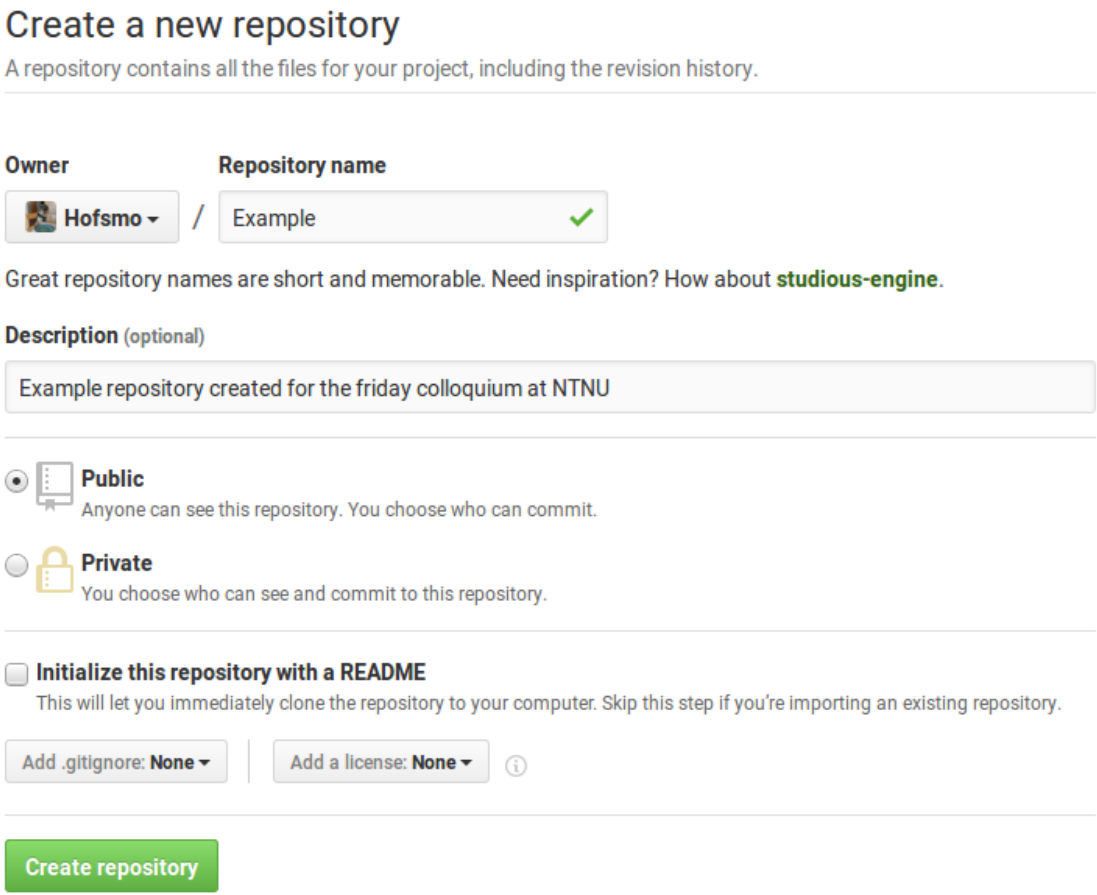
Creating a git repository

- New repository button
- Create the name
- Create a description



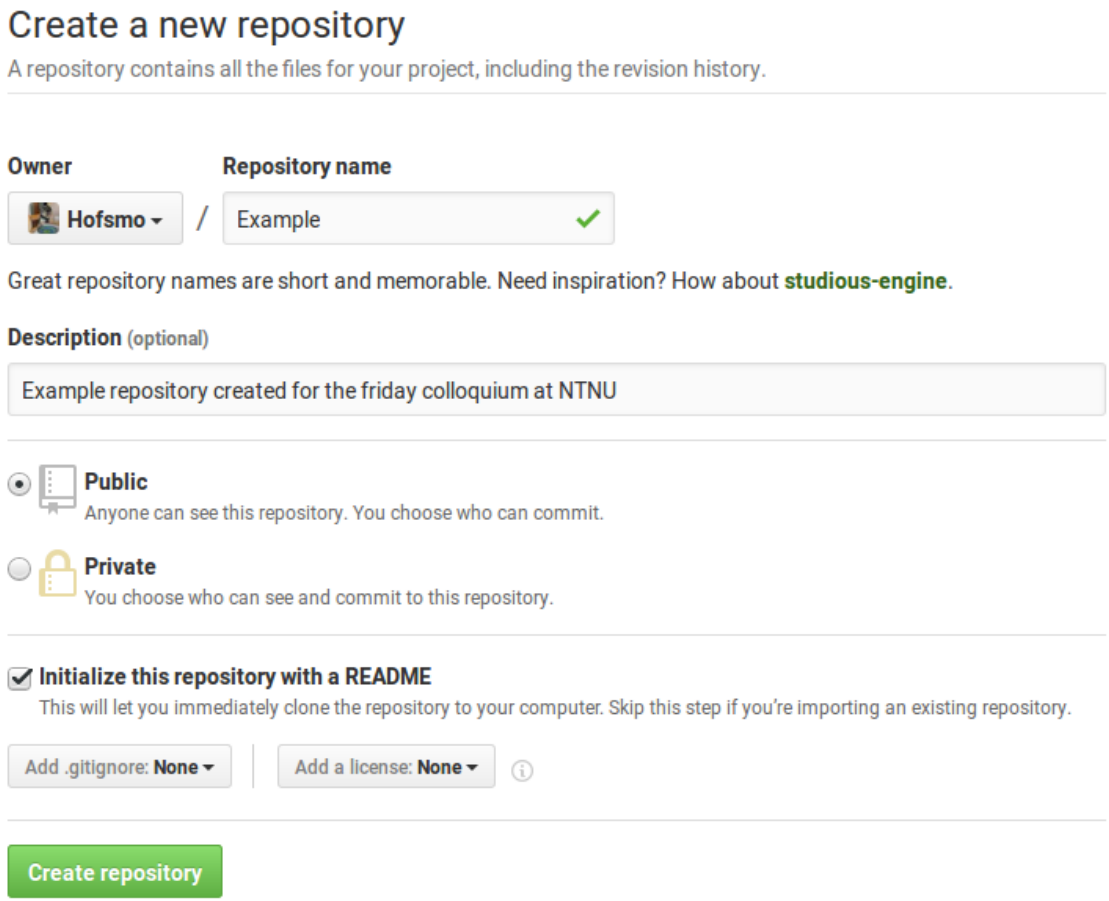
Creating a git repository

- New repository button
- Create the name
- Create a description
- Open or public?



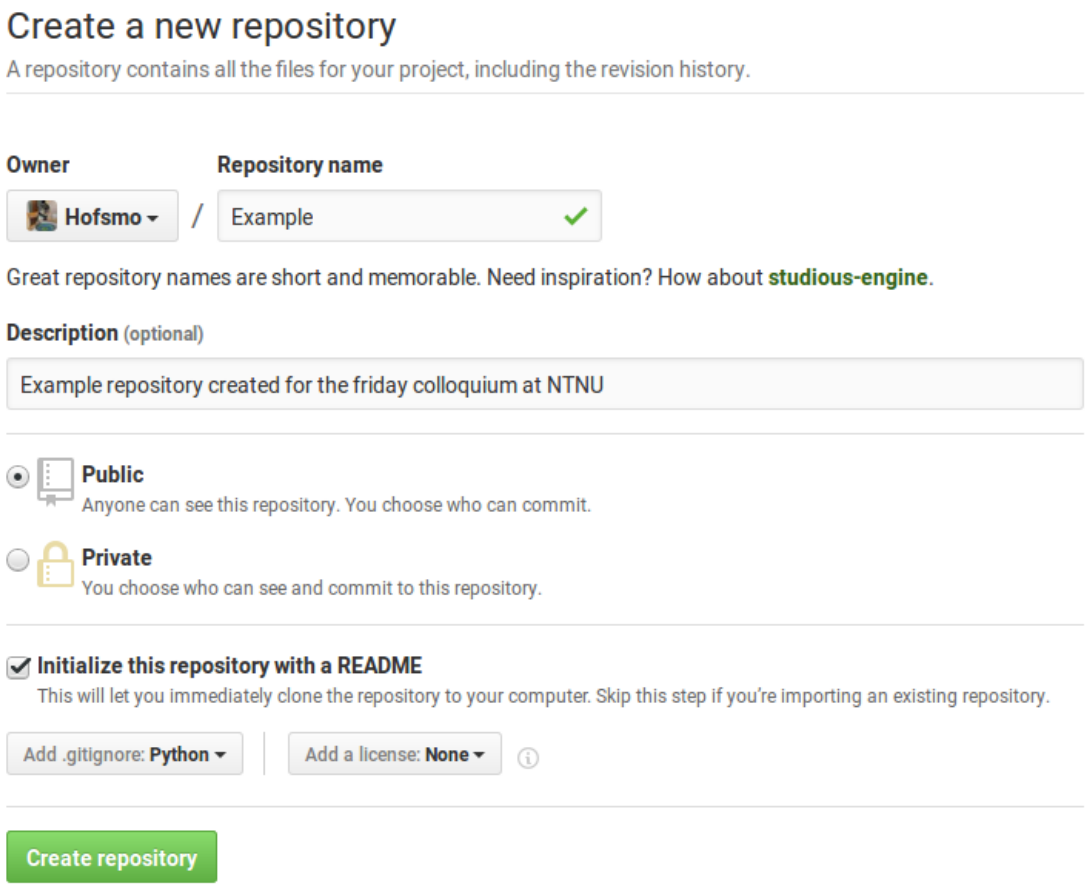
Creating a git repository

- New repository button
- Create the name
- Create a description
- Open or public?
- READMEs are cool



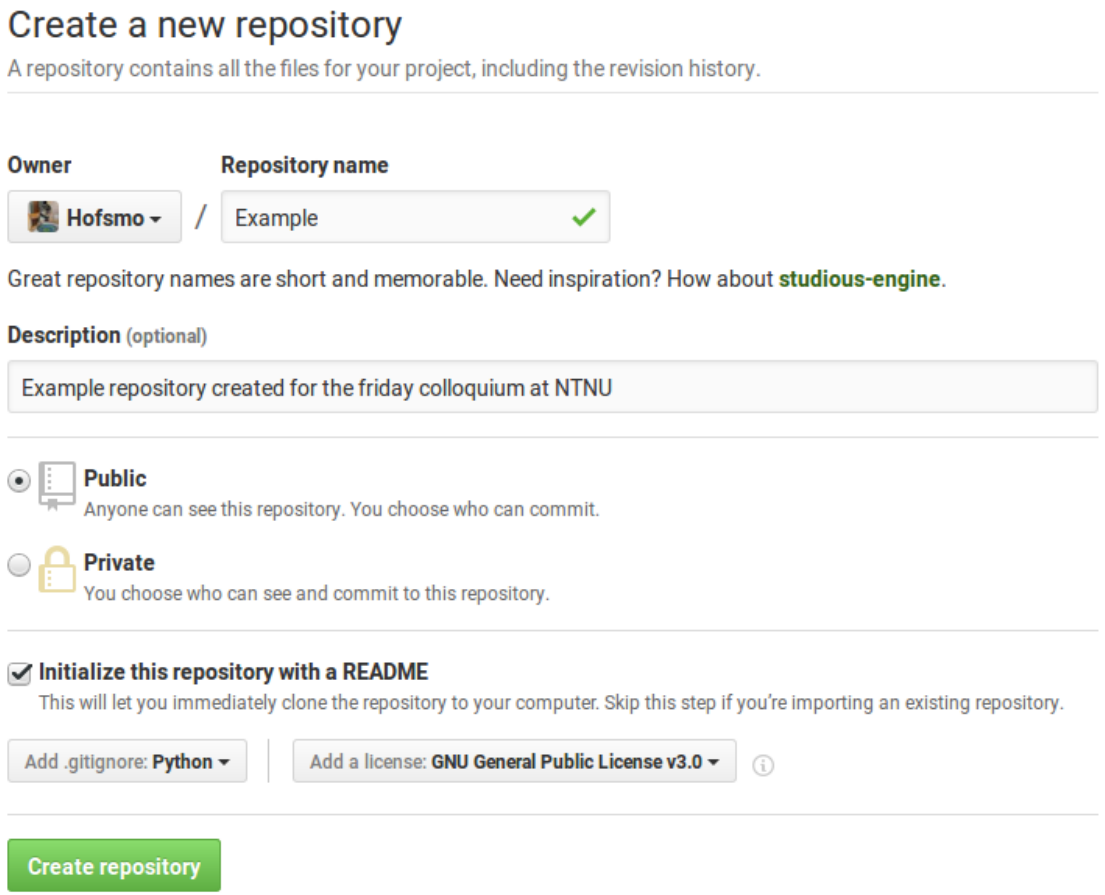
Creating a git repository

- New repository button
- Create the name
- Create a description
- Open or public?
- READMEs are cool
- Gitignore is useful

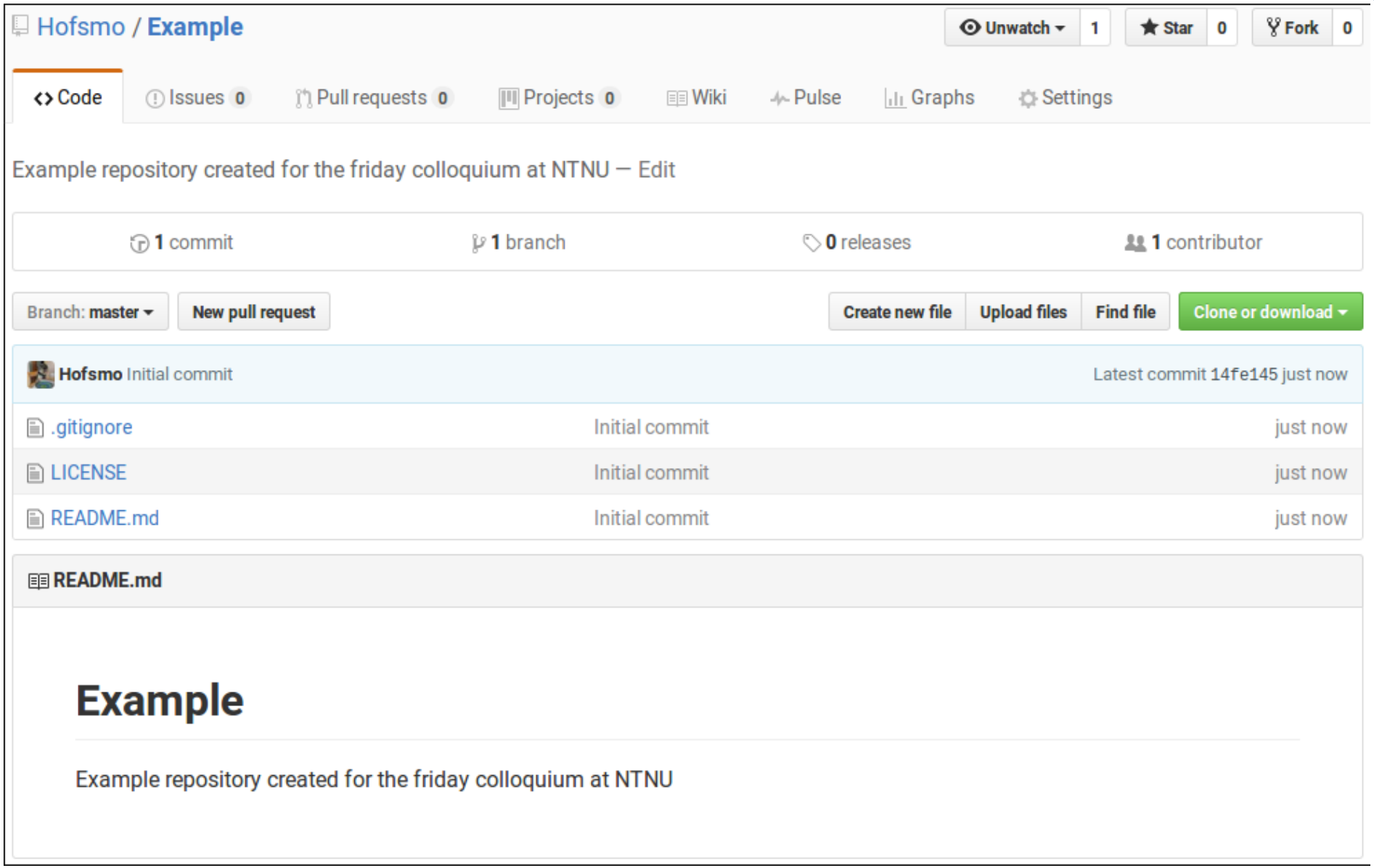


Creating a git repository

- New repository button
- Create the name
- Create a description
- Open or public?
- READMEs are cool
- Gitignore is useful
- Add license

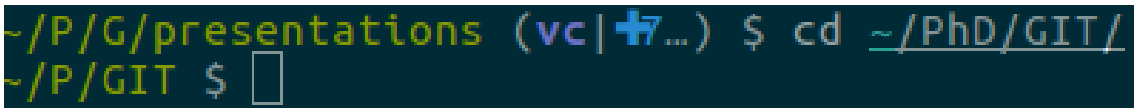


Resulting repository



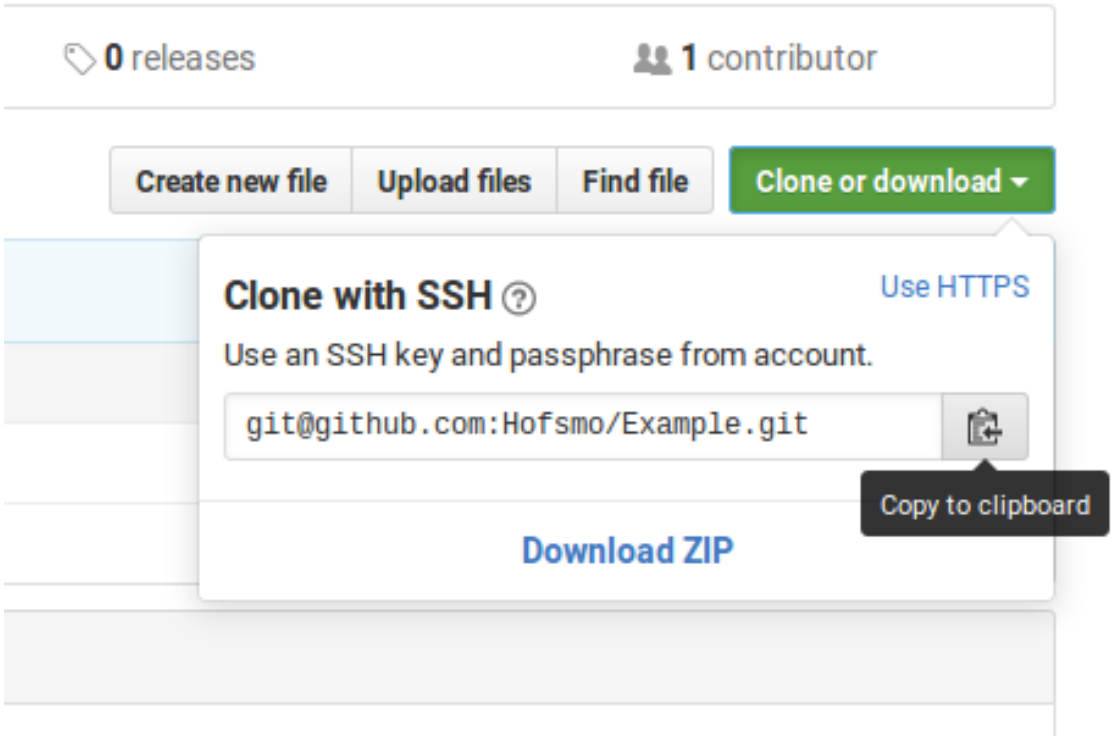
Cloing the repository

- using the terminal

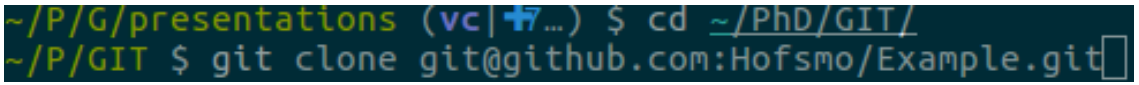


Cloing the repository

— using the terminal
  - Copy the url



Cloing the repository



— using the terminal
  - Copy the url
  - Run command

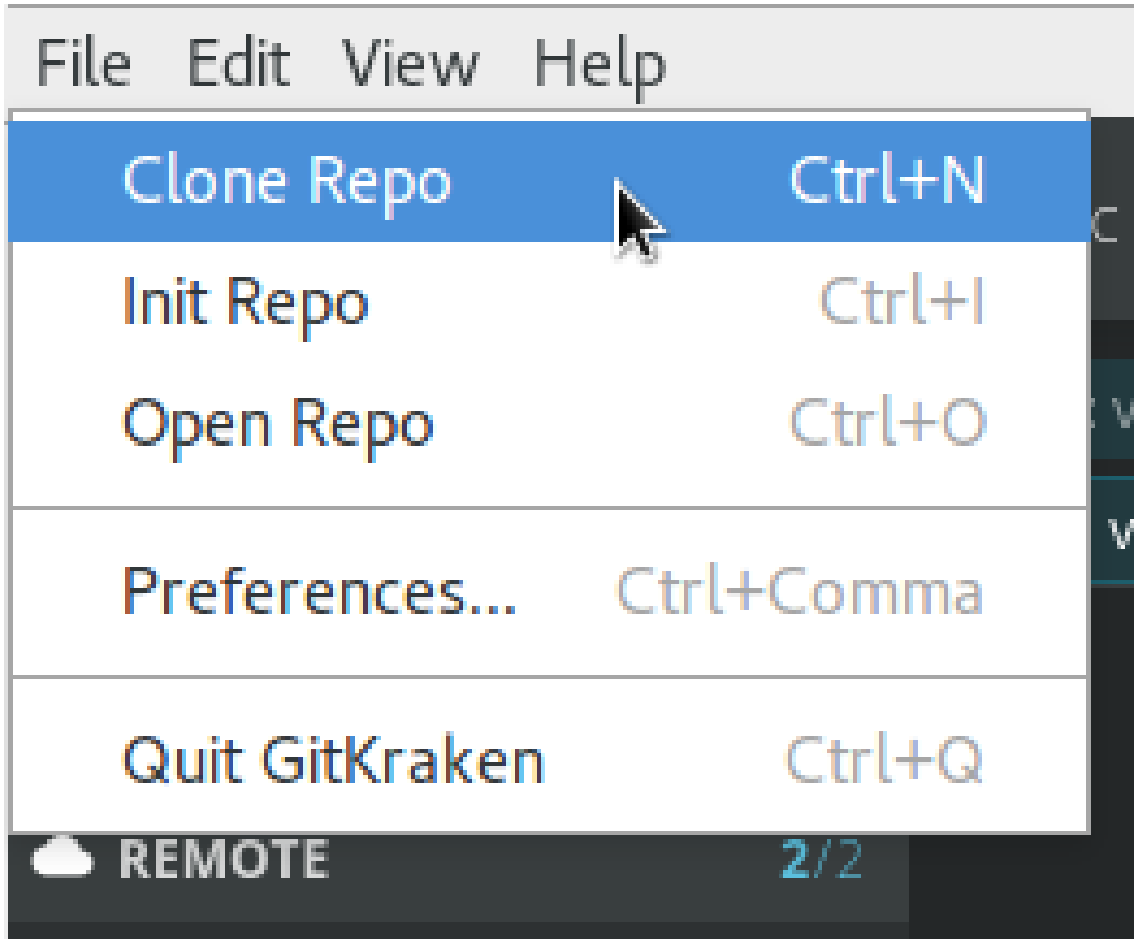
Cloing the repository

— using the terminal
  - Copy the url
  - Run command
  - Done

```
~/P/G/presentations (vc|✚7…) $ cd ~/PhD/GIT/
~/P/GIT $ git clone git@github.com:Hofsmo/Example.git
Cloning into 'Example'...
Enter passphrase for key '/home/sigurd/.ssh/id_rsa':
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), 12.78 KiB | 0 bytes/s, done.
Checking connectivity... done.
```

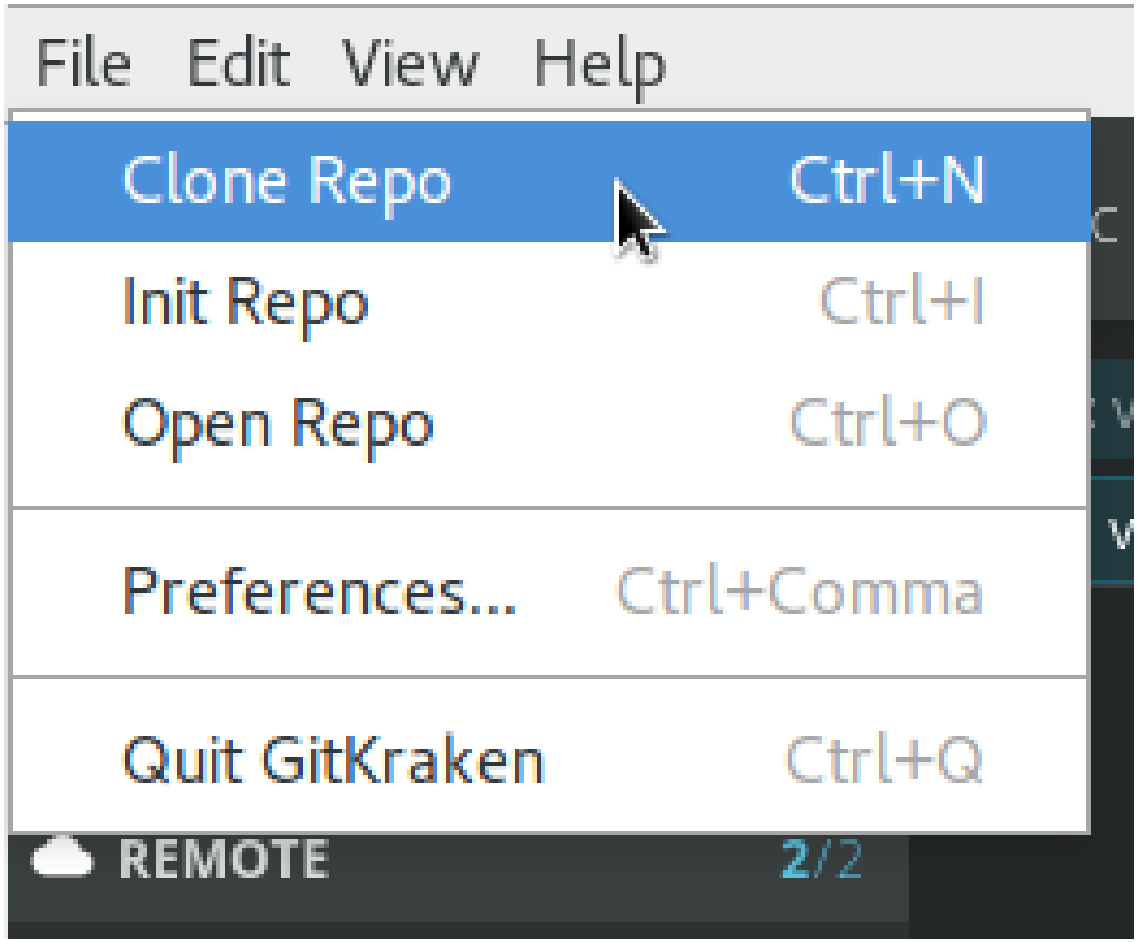
Cloing the repository

- using the terminal
  - Copy the url
  - Run command
  - Done
- Using a GUI(GitKraken in this case)



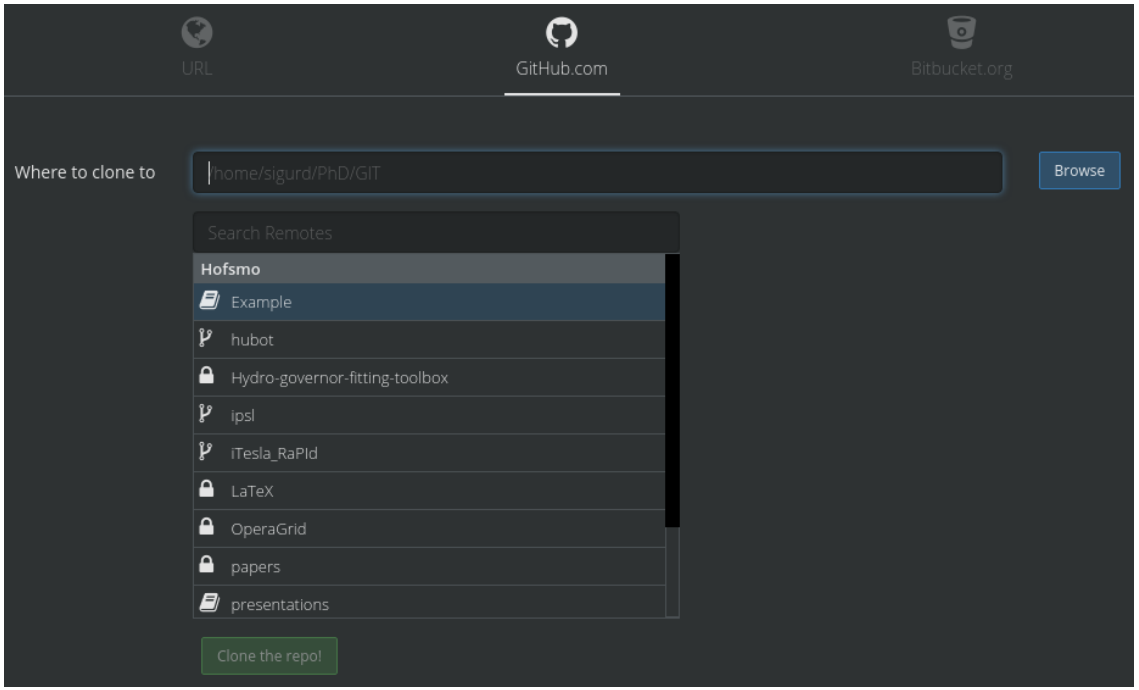
Cloing the repository

- using the terminal
  - Copy the url
  - Run command
  - Done
- Using a GUI(GitKraken in this case)
  - Find clone repo button



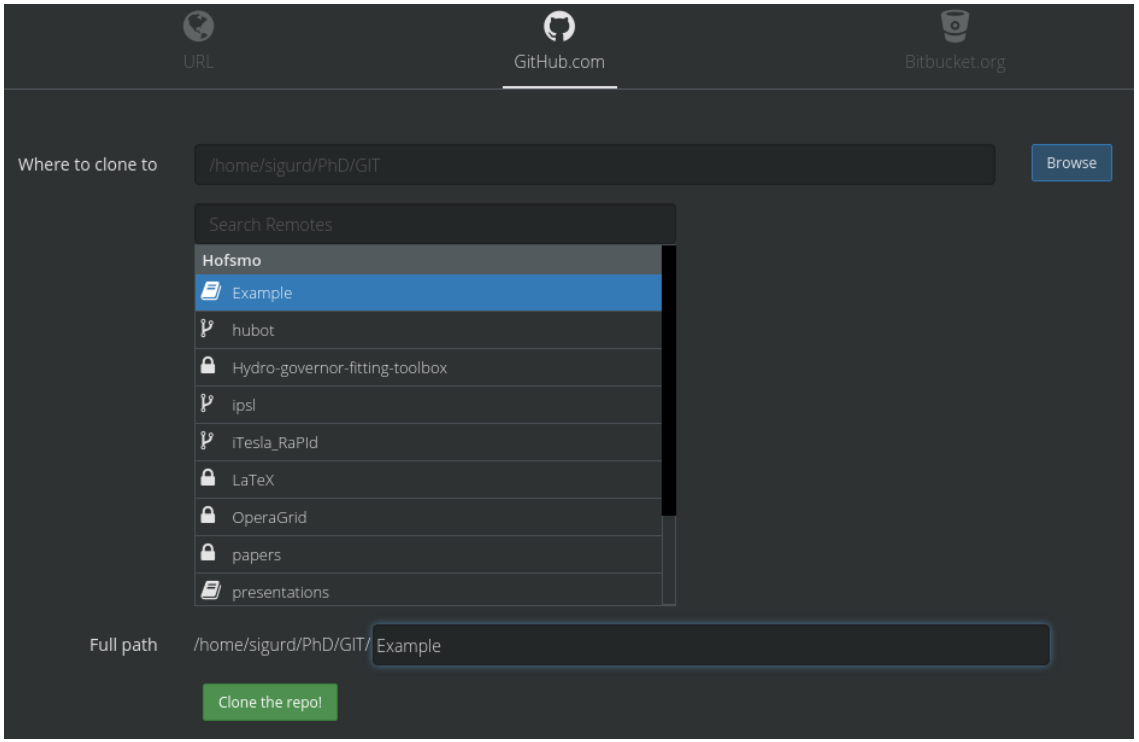
Cloing the repository

- using the terminal
  - Copy the url
  - Run command
  - Done
- Using a GUI(GitKraken in this case)
  - Find clone repo button
  - Select repository



Cloing the repository

- using the terminal
  - Copy the url
  - Run command
  - Done
- Using a GUI(GitKraken in this case)
  - Find clone repo button
  - Select repository
  - Decide where to put it and clone



Staging our first file

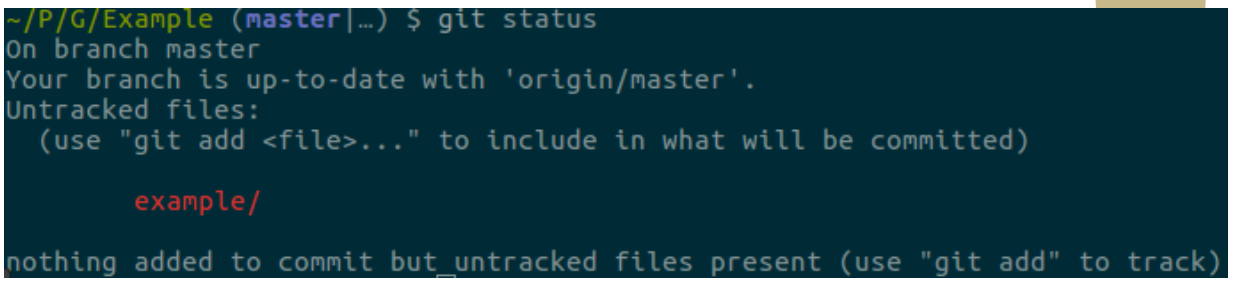
```
~/P/G/Example (master|…) $ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        example/

nothing added to commit but untracked files present (use "git add" to track)
```

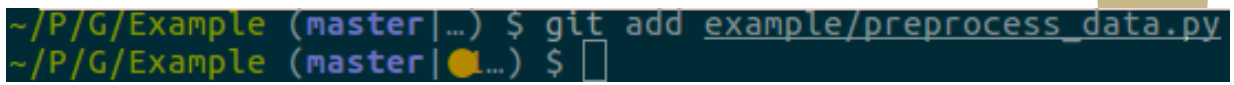
— In the terminal

Staging our first file



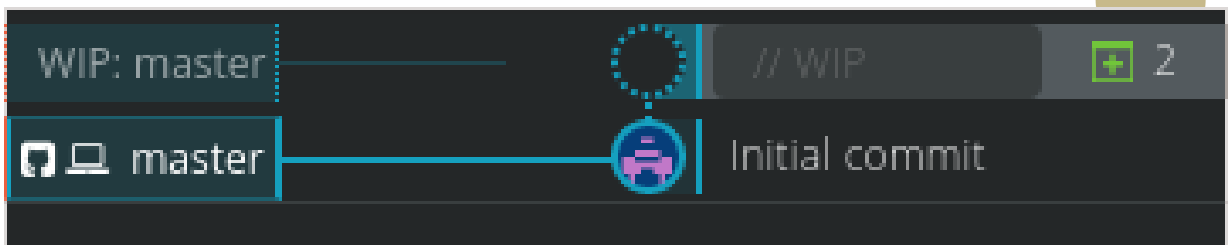
— In the terminal
  - git status

Staging our first file



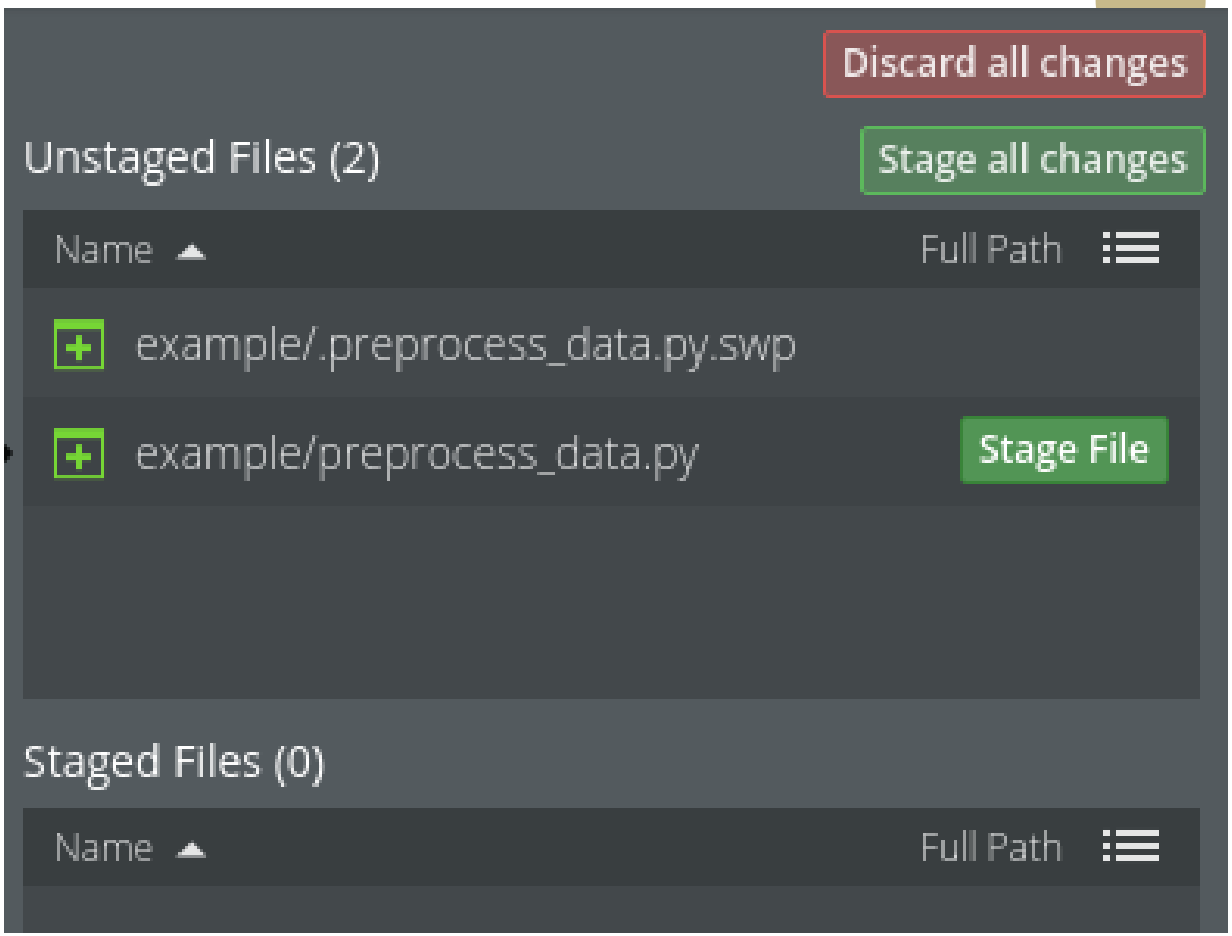
— In the terminal
  - git status
  - stage file using git add

Staging our first file



- In the terminal
  - git status
  - stage file using git add
- In the GUI

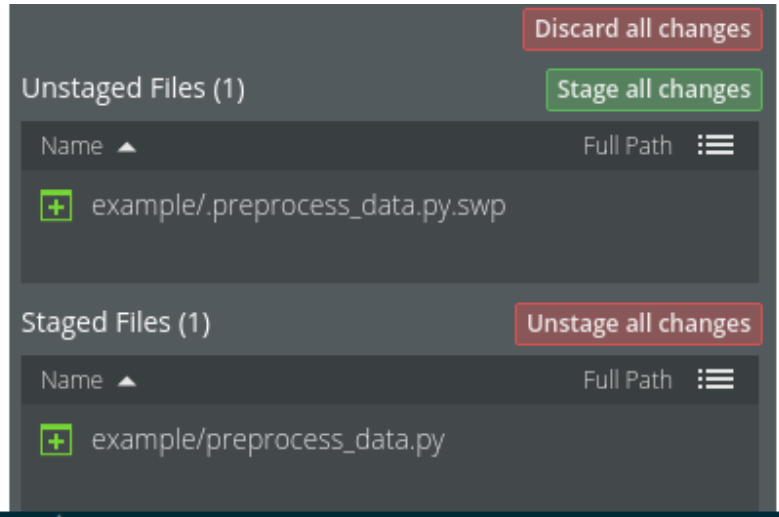
Staging our first file



- In the terminal
  - git status
  - stage file using git add
- In the GUI
  - Push "Stage File" button

Ignoring editor generated file

— Remember the second file?




```
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   example/preprocess_data.py

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        example/.preprocess_data.py.swp
```

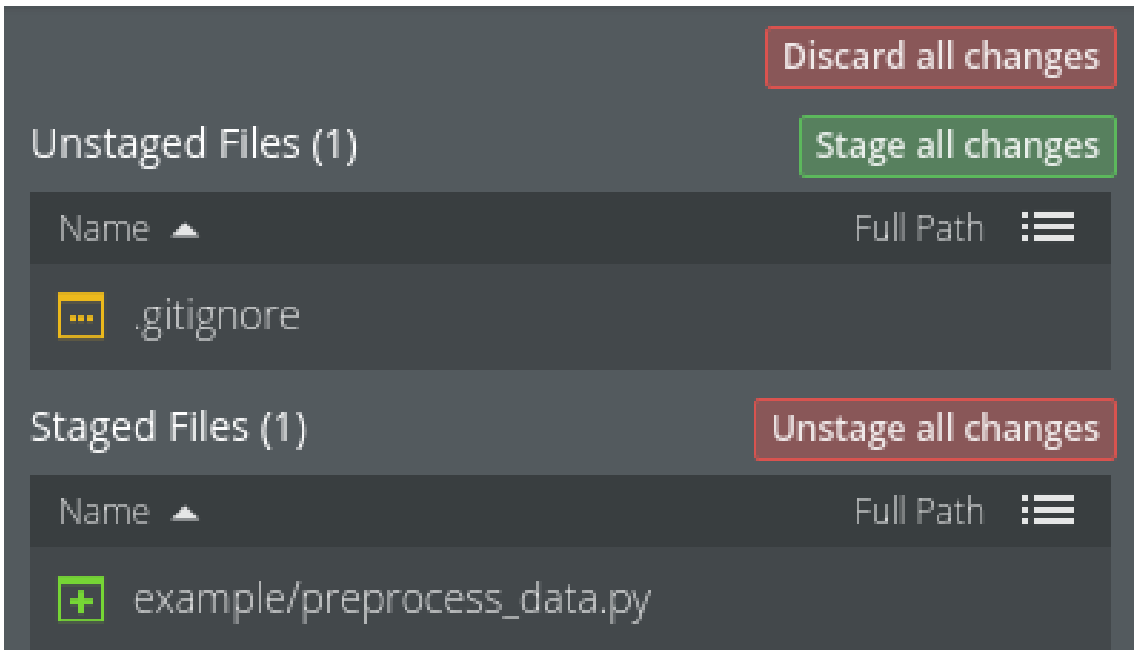
Ignoring editor generated file

- Remember the second file?
- Tell git to ignore it

```
 1
 2 # Spyder project settings
 3 .spyderproject
 4
 5 # Rope project settings
 6 .ropeproject
 7
 8 # Ignore vim stuff
 9 *.swo
10 *.swp
NORMAL > master > .gitignore
```

Ignoring editor generated file

- Remember the second file?
- Tell git to ignore it
- Git ignores it



Ignoring editor generated file

- Remember the second file?
- Tell git to ignore it
- Git ignores it
- Stage the modified .gitignore

Our first commit

— In terminal:

```
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   .gitignore
        new file:   example/preprocess_data.py
```

Our first commit

— In terminal:
  - Enter git commit

```
~/P/G/Example (master|●2) $ git commit
```

Our first commit

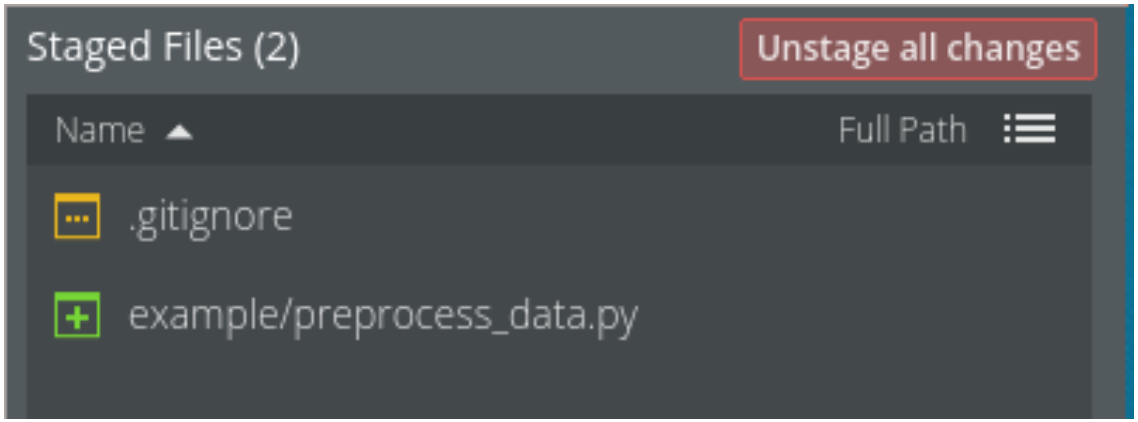
— In terminal:
  - Enter git commit
  - Write and save commit message

```
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
 On branch master
 Your branch is up-to-date with 'origin/master'.

 Changes to be committed:
        modified:   .gitignore
                Added swp and swo to gitignore
        new file:   example/preprocess_data.py
                First implementation of function
```

Our first commit

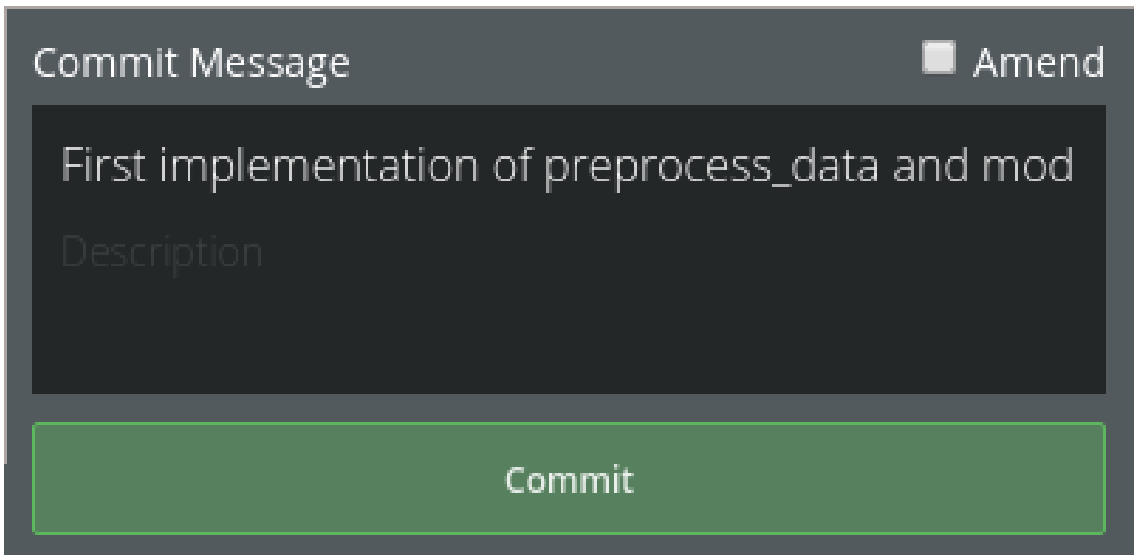
- In terminal:
  - Enter git commit
  - Write and save commit message
- In GUI:



Our first commit

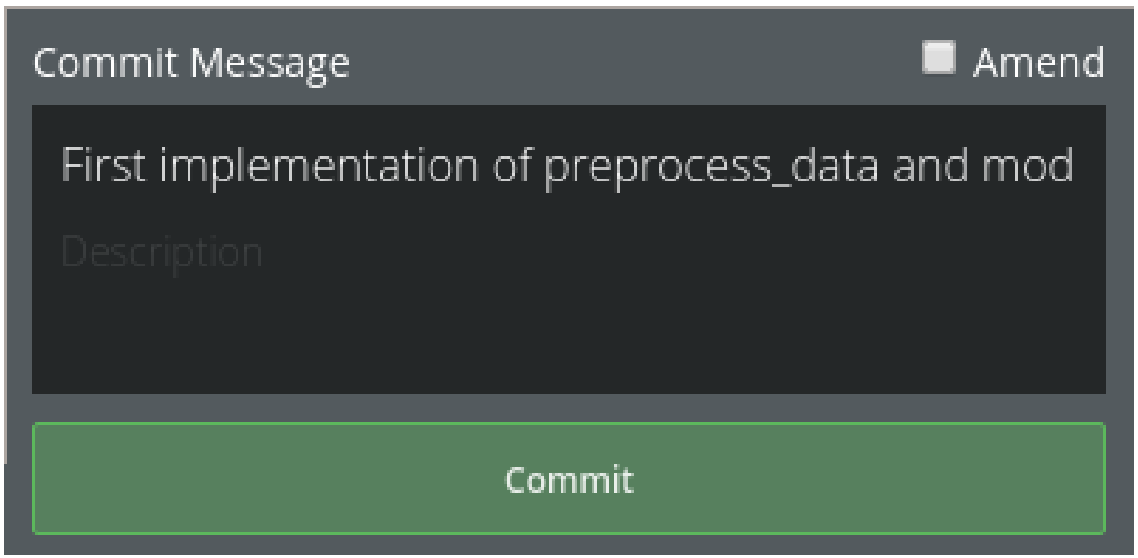
— In terminal:
  - Enter git commit
  - Write and save commit message

— In GUI:
  - Write commit message



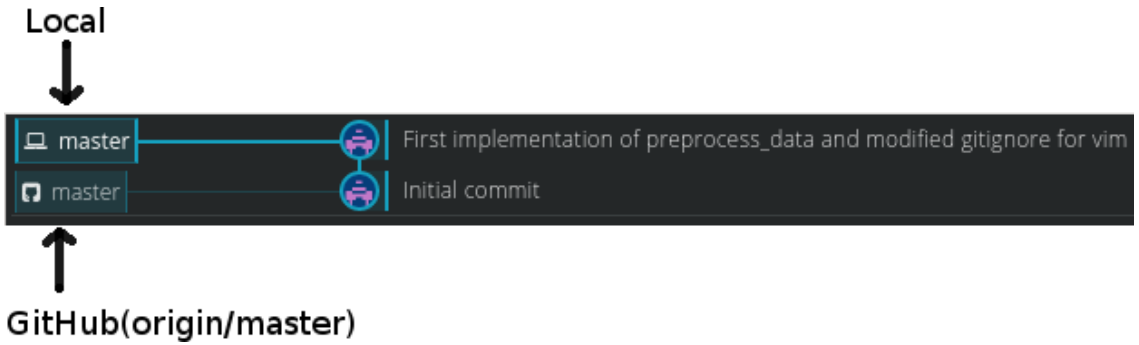
Our first commit

- In terminal:
  - Enter git commit
  - Write and save commit message
- In GUI:
  - Write commit message
  - Push commit button

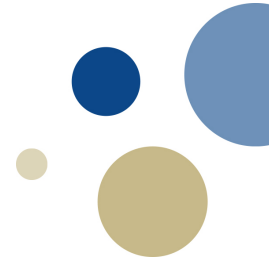


Push the changes to GitHub

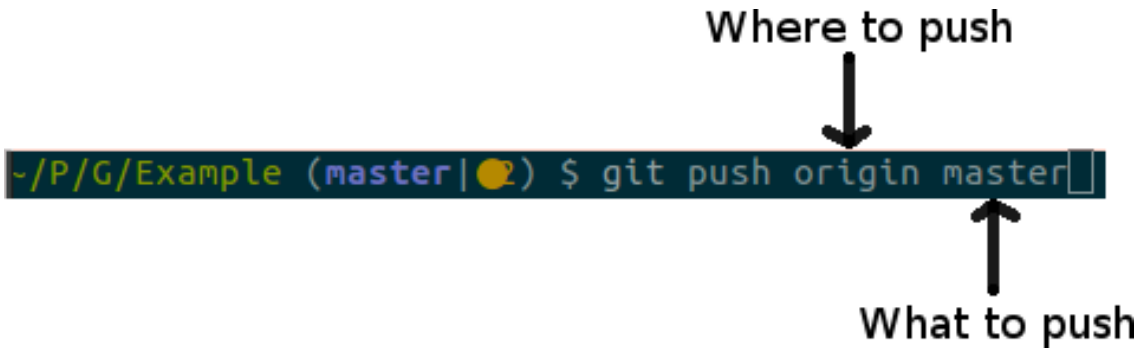
— GitHub (remote) is now behind our local copy



Push the changes to GitHub

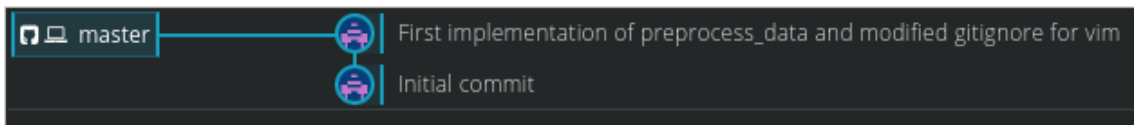


- GitHub (remote) is now behind our local copy
- Push changes to GitHub



Push the changes to GitHub

- GitHub (remote) is now behind our local copy
- Push changes to GitHub
- GitHub and local copy are now equal



Outline

Introducing a git work flow

— Working directly on the master branch is not reccommended

Introducing a git work flow

— Working directly on the master branch is not reccommended
— In this section I will present a git flow inspired working flow I use at SINTEF:

Introducing a git work flow

- Working directly on the master branch is not reccommended
- In this section I will present a git flow inspired working flow I use at SINTEF:
  1. Create issue

Introducing a git work flow

- Working directly on the master branch is not reccommended
- In this section I will present a git flow inspired working flow I use at SINTEF:
  1. Create issue
  2. Create issue branch

Introducing a git work flow

- Working directly on the master branch is not reccommended
- In this section I will present a git flow inspired working flow I use at SINTEF:
  1. Create issue
  2. Create issue branch
  3. Work on issue branch until it works

Introducing a git work flow

- Working directly on the master branch is not reccommended
- In this section I will present a git flow inspired working flow I use at SINTEF:
  1. Create issue
  2. Create issue branch
  3. Work on issue branch until it works
  4. Create a pull request

Introducing a git work flow

- Working directly on the master branch is not reccommended
- In this section I will present a git flow inspired working flow I use at SINTEF:
  1. Create issue
  2. Create issue branch
  3. Work on issue branch until it works
  4. Create a pull request
  5. Merge issue branch to develop

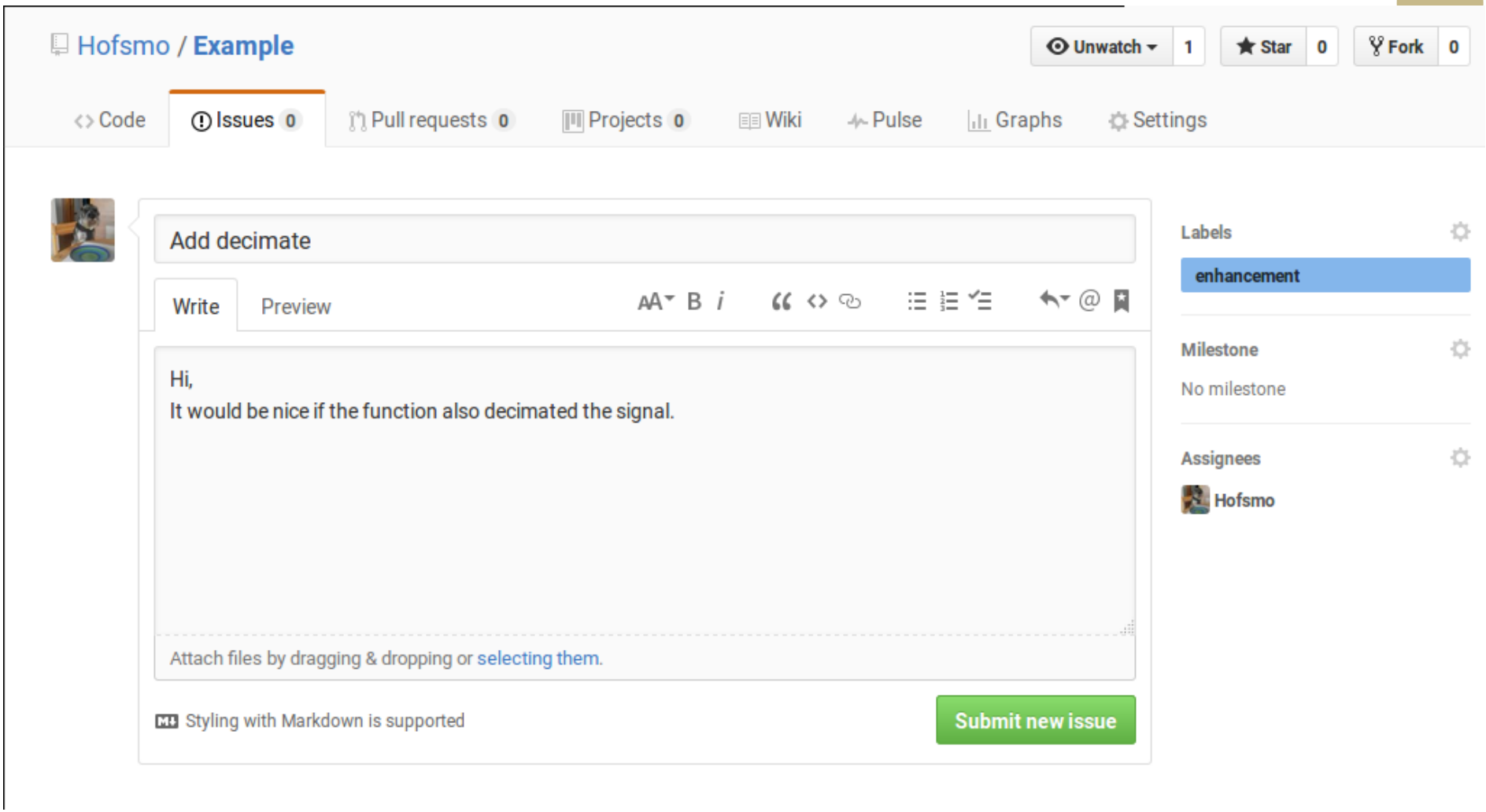
Introducing a git work flow

- Working directly on the master branch is not reccommended
- In this section I will present a git flow inspired working flow I use at SINTEF:
  1. Create issue
  2. Create issue branch
  3. Work on issue branch until it works
  4. Create a pull request
  5. Merge issue branch to develop
  6. When develop is stable merge into master

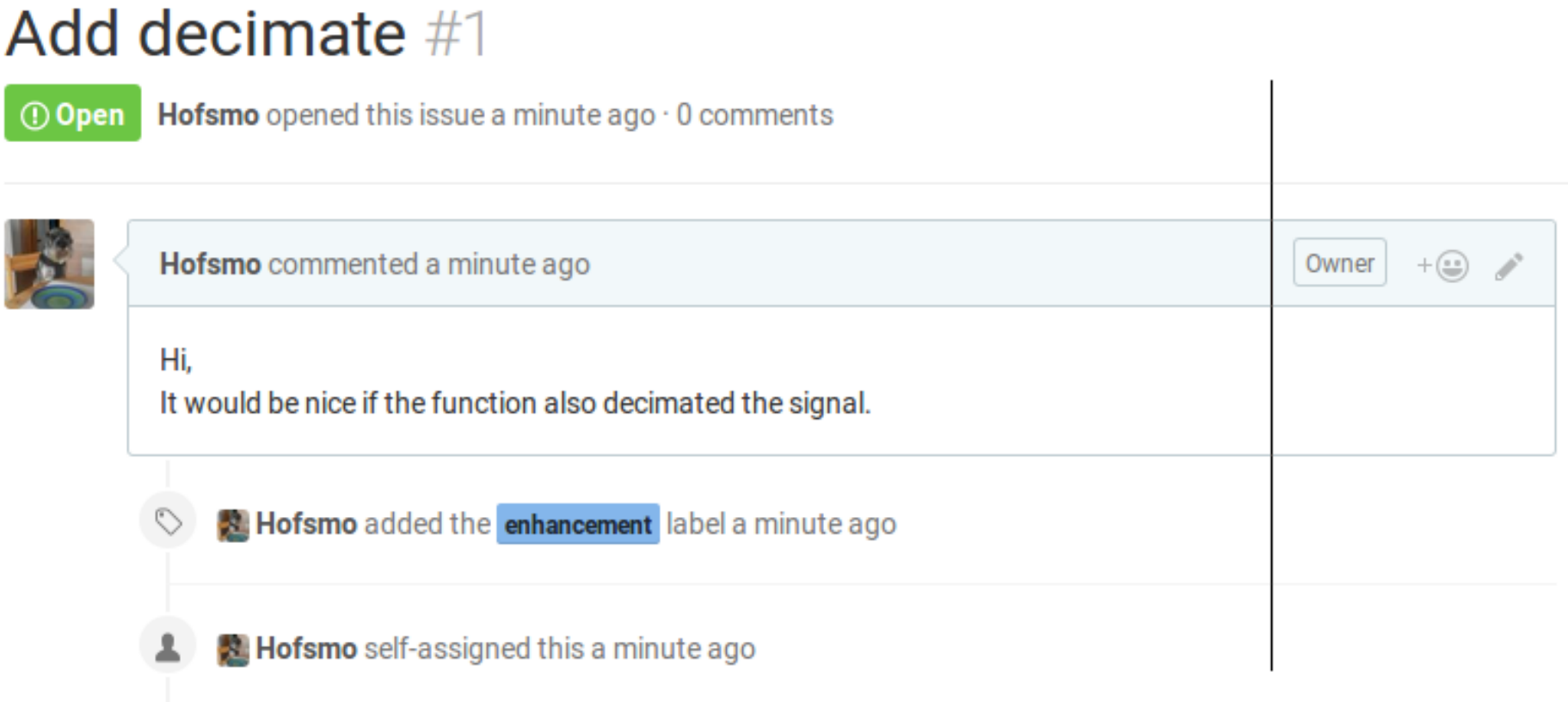
Introducing a git work flow

- Working directly on the master branch is not reccommended
- In this section I will present a git flow inspired working flow I use at SINTEF:
  1. Create issue
  2. Create issue branch
  3. Work on issue branch until it works
  4. Create a pull request
  5. Merge issue branch to develop
  6. When develop is stable merge into master
- Example will follow

Create an issue



Create an issue



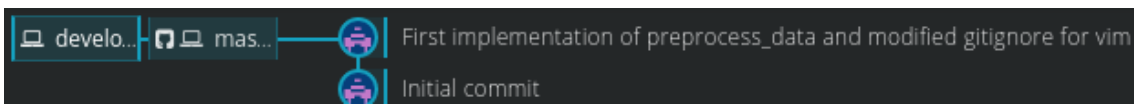
Create the feature branches

— Branch the develop branch from master

```
~/P/G/Example (master|✔) $ git checkout -b develop
Switched to a new branch 'develop'
~/P/G/Example (develop|✔) $
```

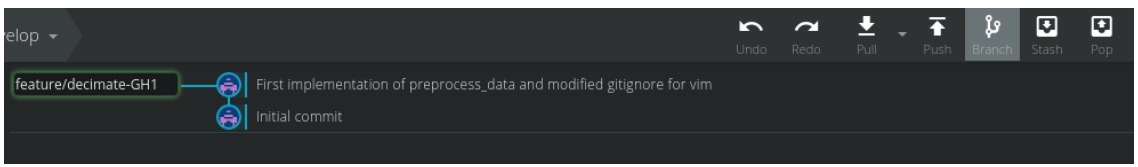
Create the feature branches

- Branch the develop branch from master
- Develop and master point to the same commit

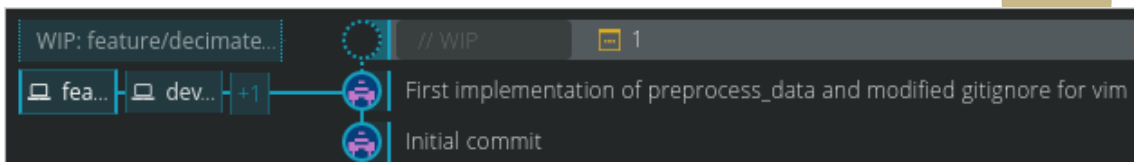


Create the feature branches

- Branch the develop branch from master
- Develop and master point to the same commit
- Branch the feature branch from develop



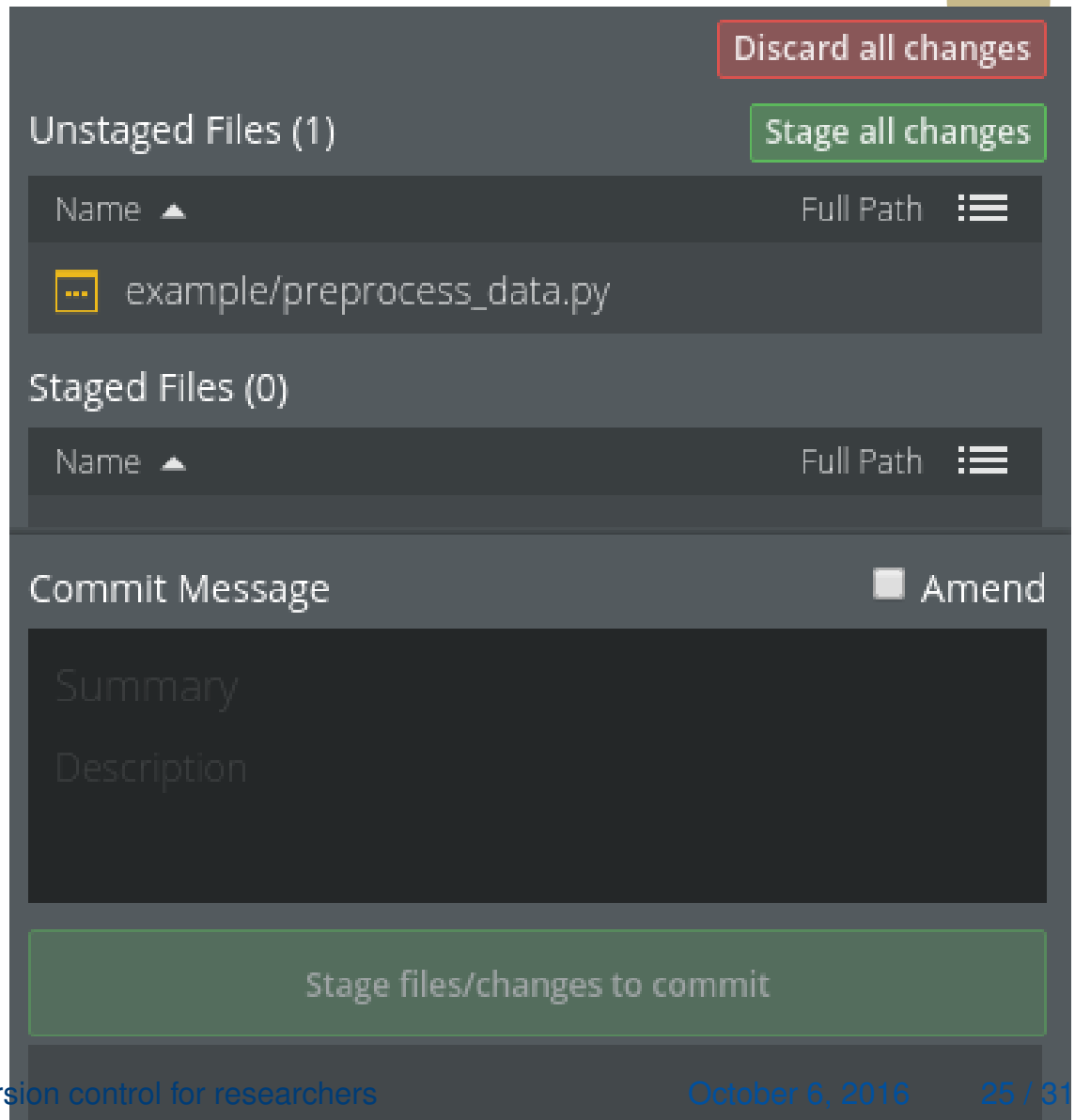
Create a pull request



— Now we can see that we have unstaged changes

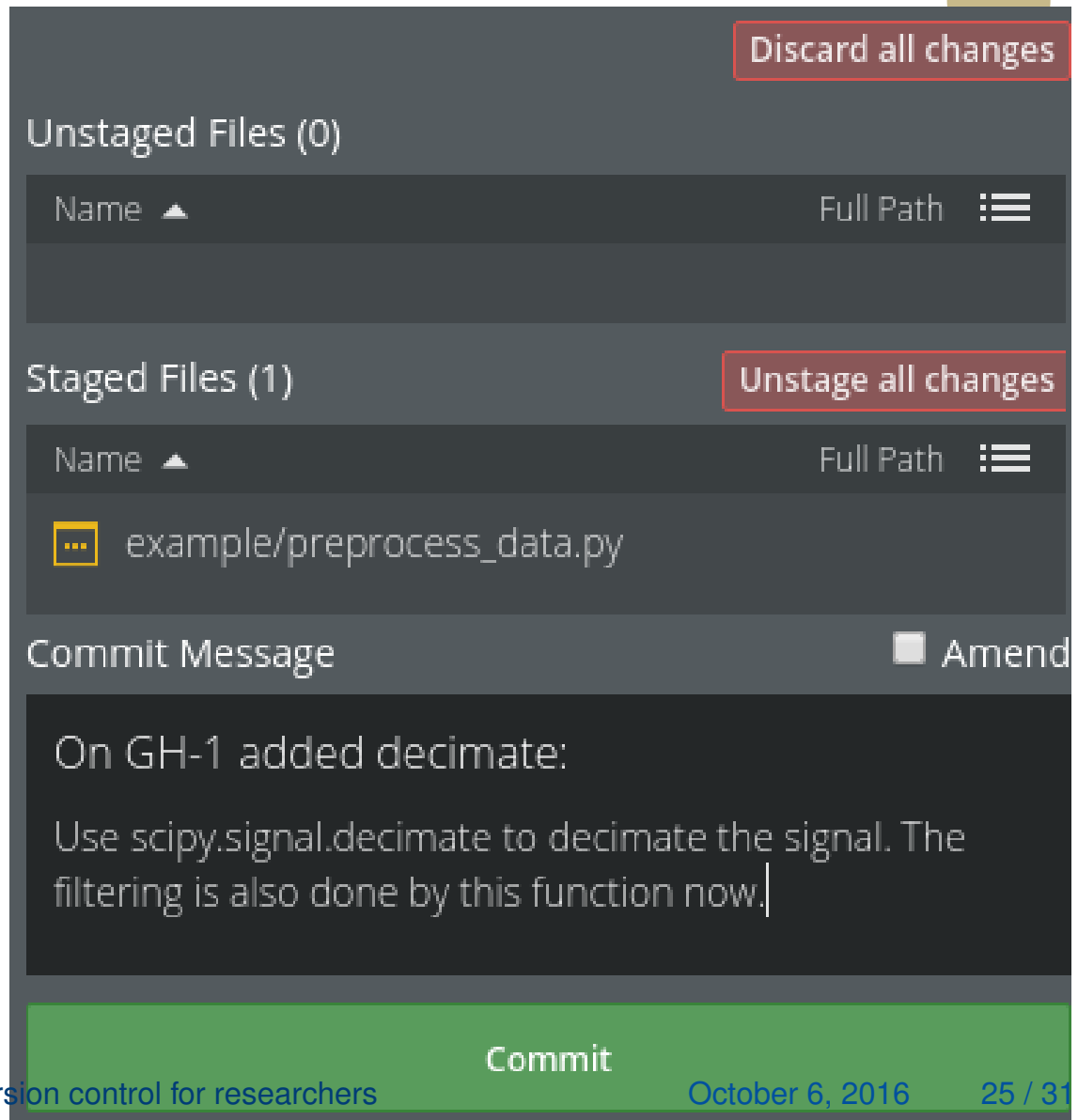
Create a pull request

- Now we can see that we have unstaged changes
- Stage the changes



Create a pull request

- Now we can see that we have unstaged changes
- Stage the changes
- Commit the changes

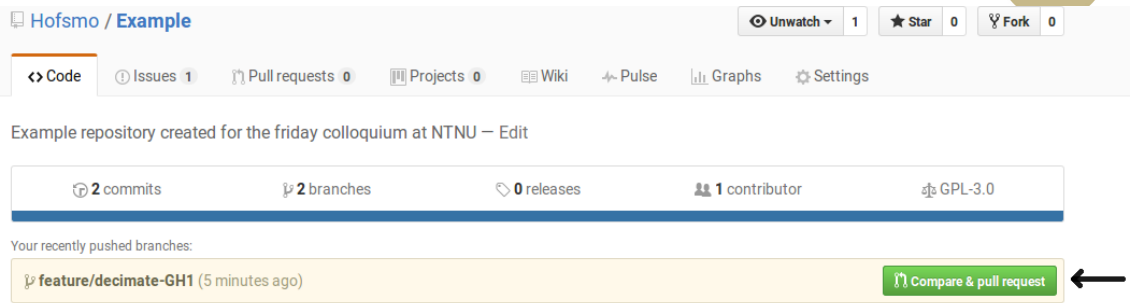


Create a pull request

- Now we can see that we have unstaged changes
- Stage the changes
- Commit the changes
- GH-1 is a reference to the issue

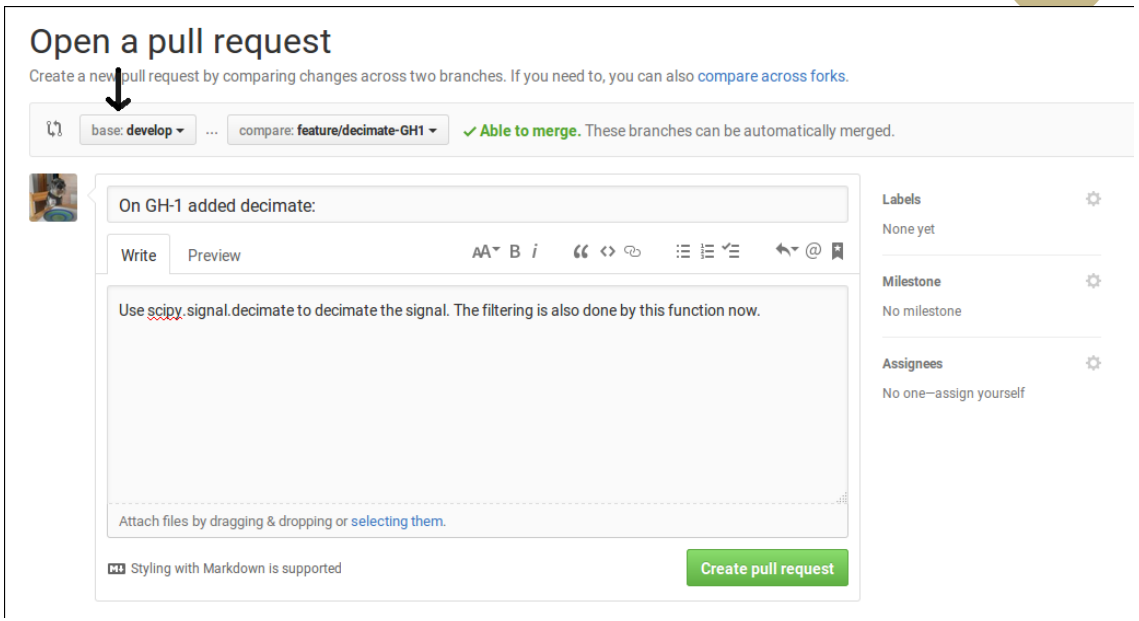
Create pull request

— Create a pull request



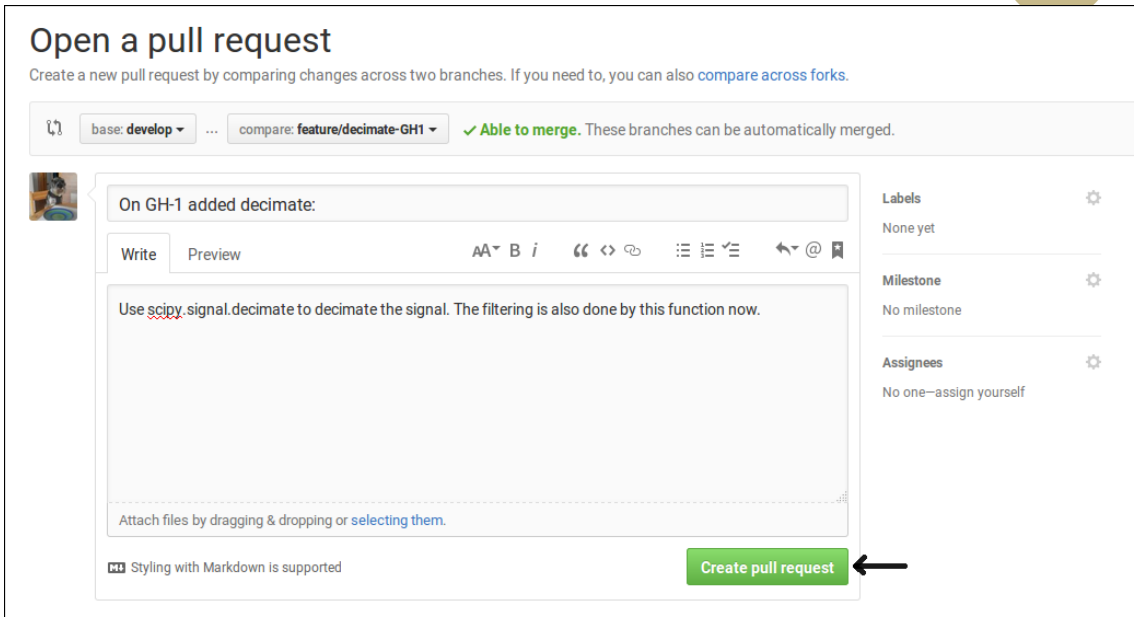
Create pull request

- Create a pull request
- Choose develop as base



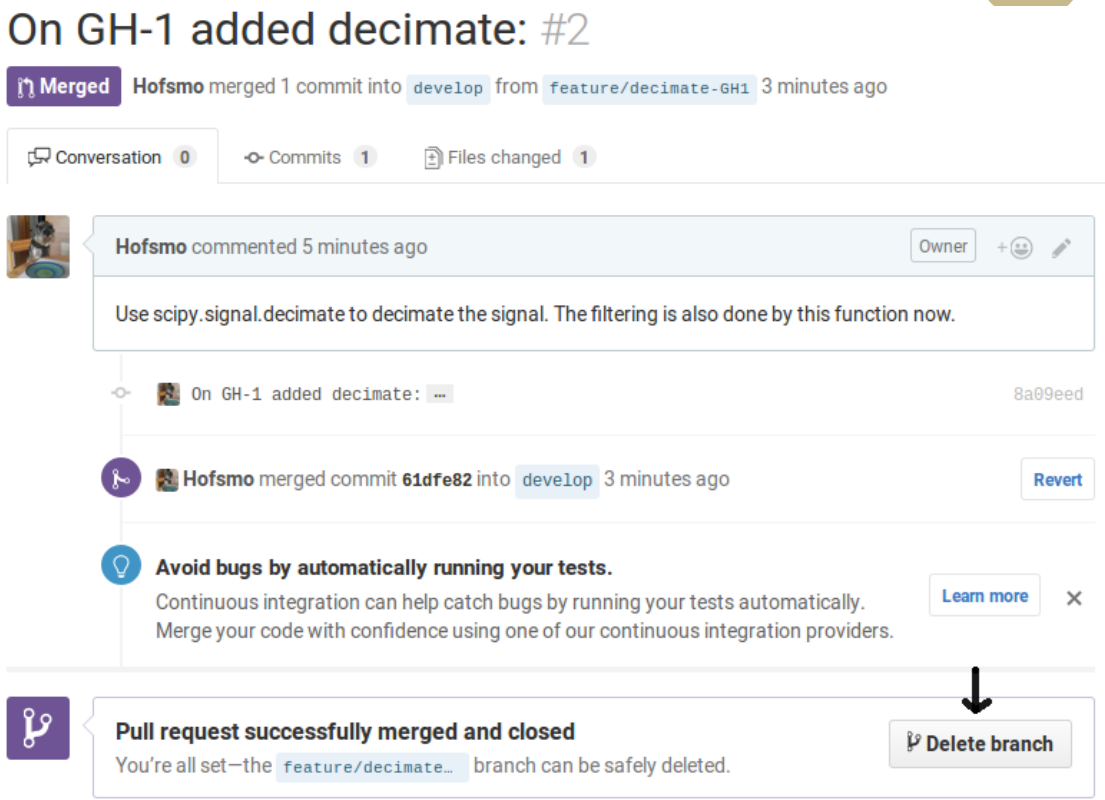
Create pull request

- Create a pull request
- Choose develop as base
- Create the pull request



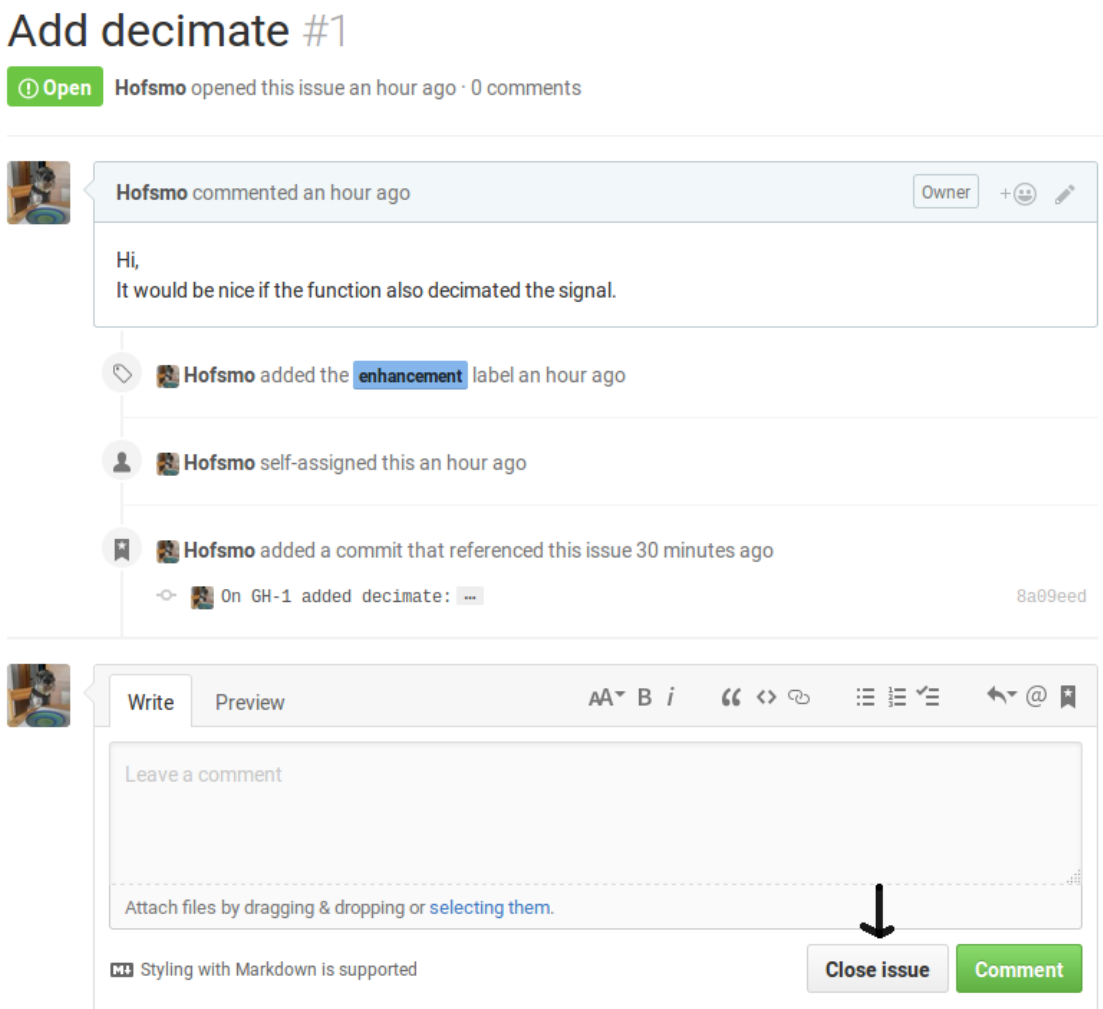
Create pull request

- Create a pull request
- Choose develop as base
- Create the pull request
- Delete the branch

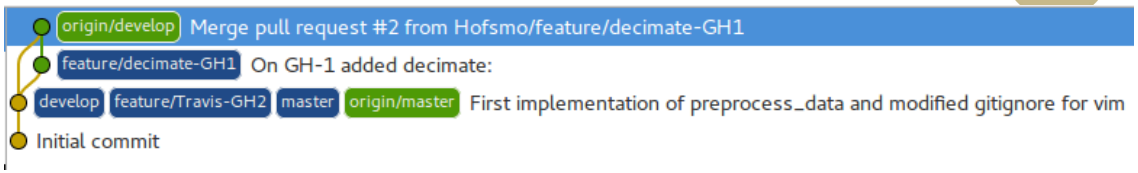


Create pull request

- Create a pull request
- Choose develop as base
- Create the pull request
- Delete the branch
- Close the issue (Could also have written "close GH-1" earlier to do this



Create pull request

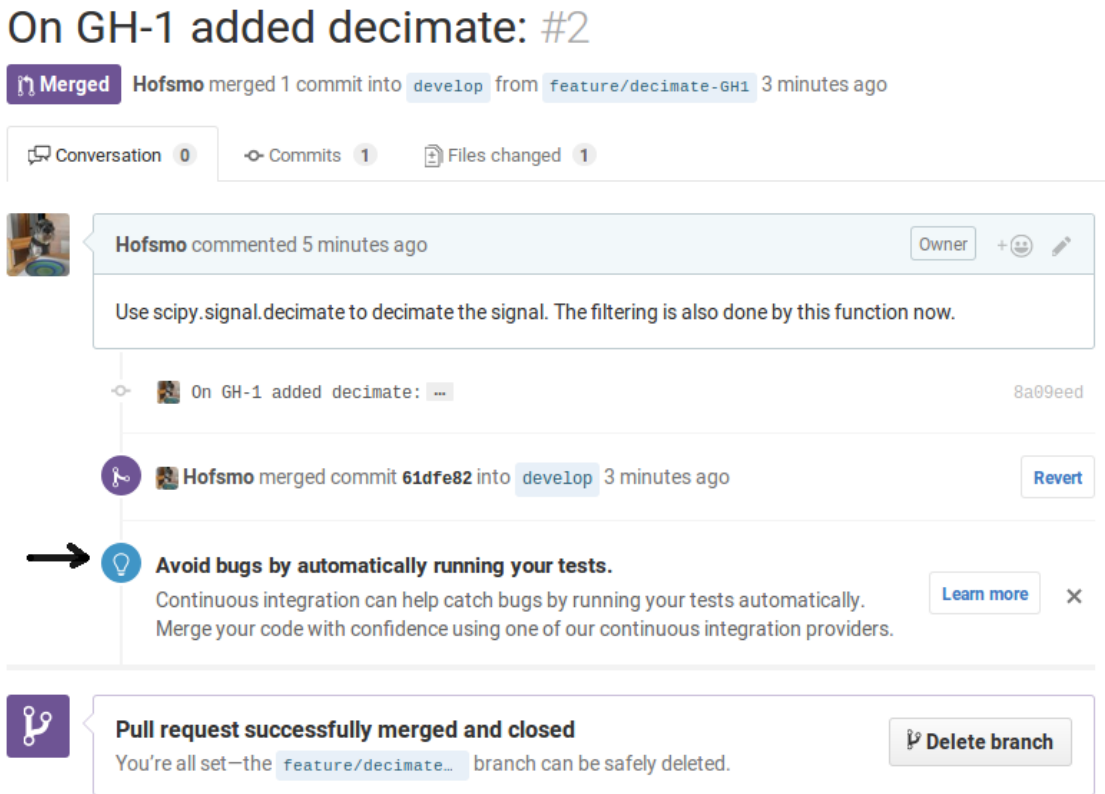


- Create a pull request
- Choose develop as base
- Create the pull request
- Delete the branch
- Close the issue (Could also have written "close GH-1" earlier to do this
- The git tree

Outline

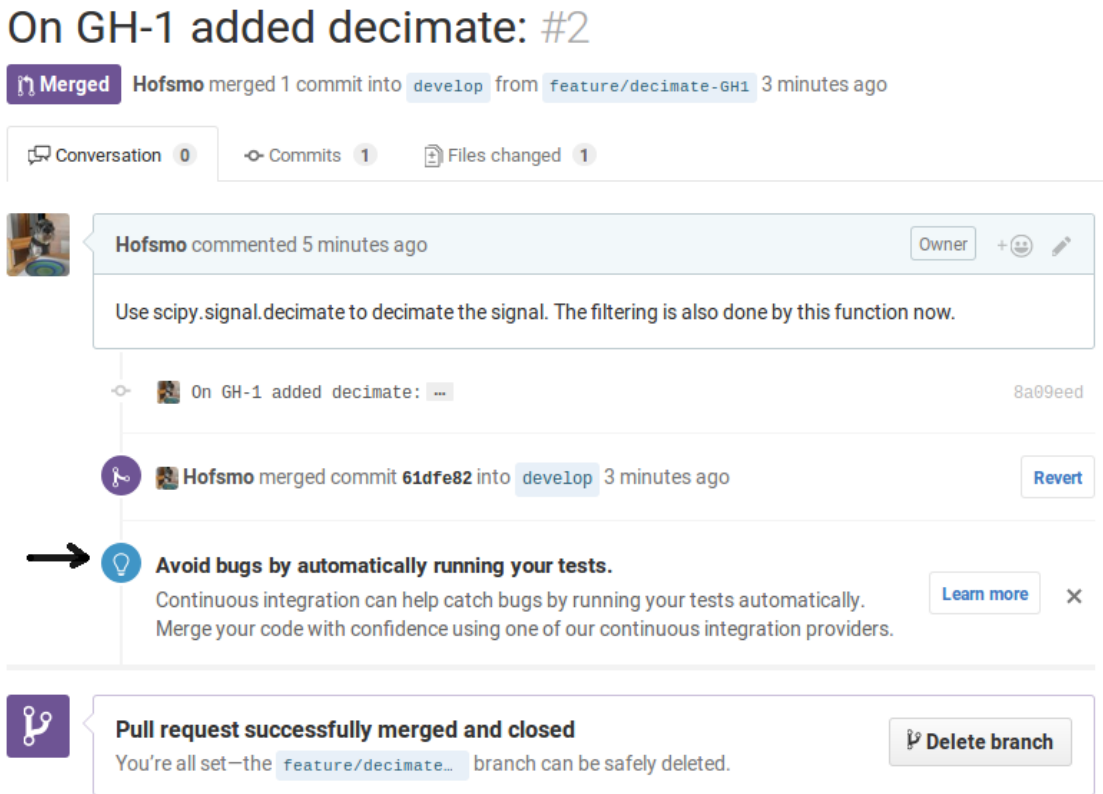
Continuous integration

— Did anyone notice?



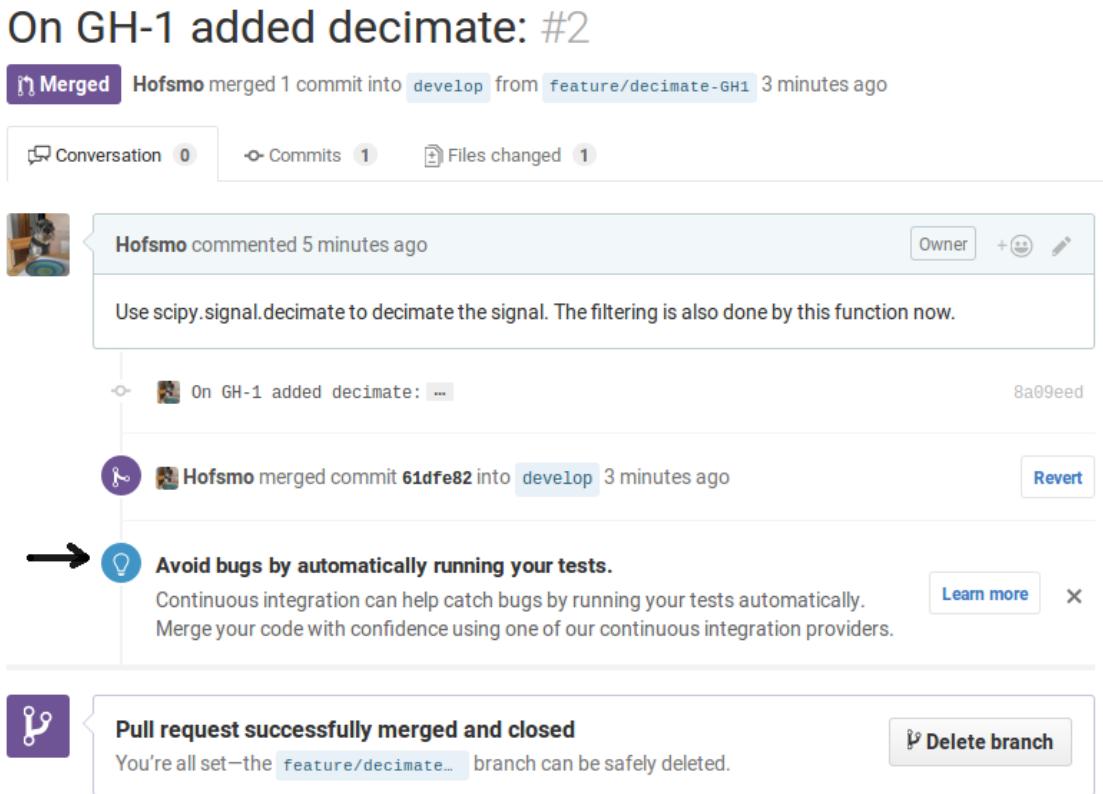
Continuous integration

- Did anyone notice?
- For GitHub travis is a good alternative



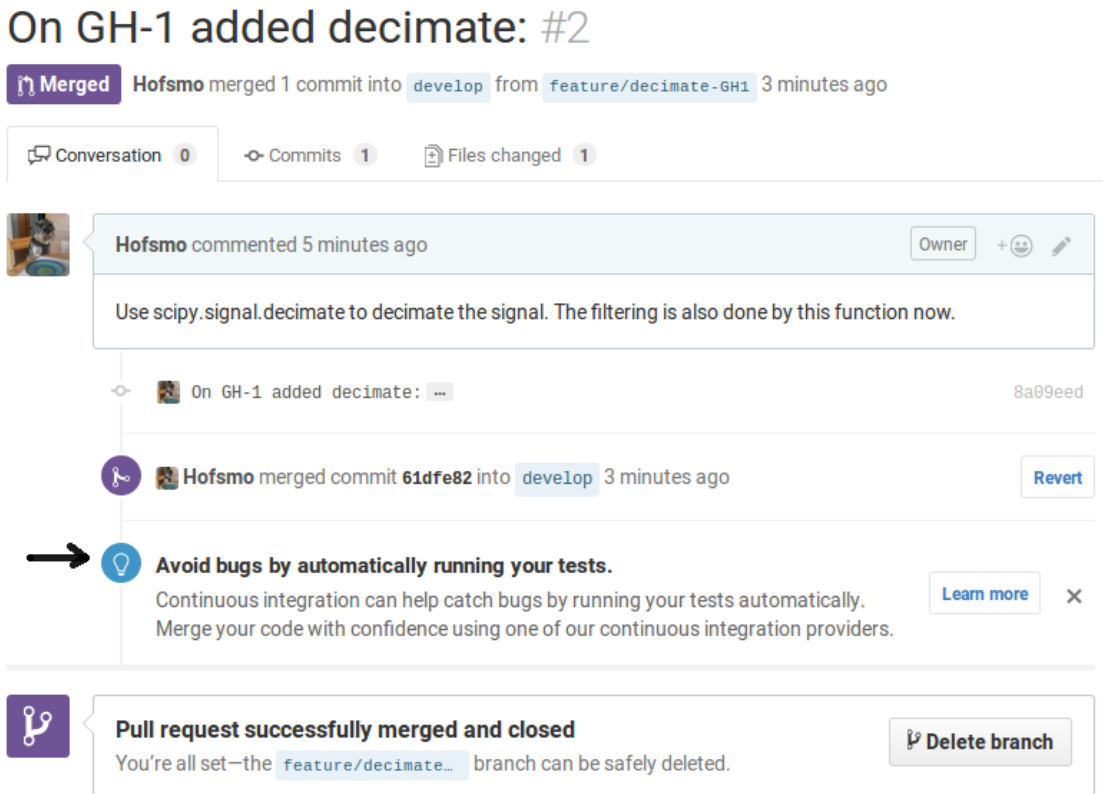
Continuous integration

- Did anyone notice?
- For GitHub travis is a good alternative
- Jenkins also works with GitHub and supports MATLAB unlike travis



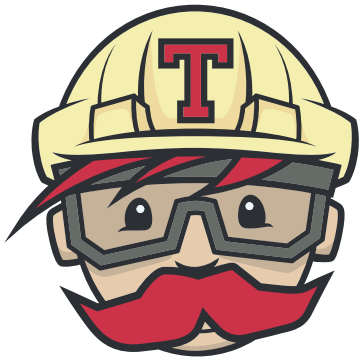
Continuous integration

- Did anyone notice?
- For GitHub travis is a good alternative
- Jenkins also works with GitHub and supports MATLAB unlike travis
- BitBucket has bamboo, but may cost money



Travis

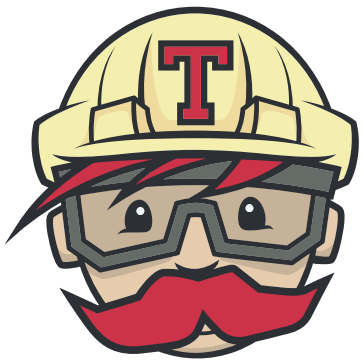
— Travis is free for open repositories



Travis CI

Travis

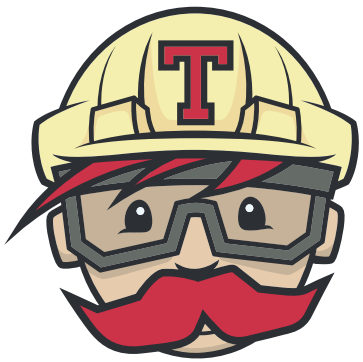
- Travis is free for open repositories
- It is controlled through a configuration file named .travis.yml



Travis CI

Travis

- Travis is free for open repositories
- It is controlled through a configuration file named .travis.yml
- Does not support MATLAB



Travis CI

Outline

References

- Most pictures and a lot of information from:
  `www.git-scm.com/book/en/v2/`
- `www.github.com`
- `www.bitbucket.org`
- This presentation:
  `https://github.com/Hofsmo/presentations/tree/vc`
- `https://github.com/Hofsmo/Example`