Unix

Python

LaTeX

C/C++

bash

make

git

Statistics

ssh

Introduction to git

Sabrina Einecke

HEAG Introduction Course

Working Together

Often you work together in a group on a report, some code, exercises etc. How do you organise?

- Email?
- Dropbox?

Probably you encountered some problems (examples):

- 2 people made changes in the same file, and it was quite some work to merge them.
- 2 people made changes to a program. With the changes of person A, the changes of person B are not working.
- You lost changes.
- You had to wait for other people to finish their work, before you were able to proceed.
- You want to test something but don't want to bother the others.

→ **That's quite inefficient and annoying!**

Working On Your Own

- If you are develping code, it often happens that it stops working at some point and you are not able to revert the changes and make it work again.
- If you are developing code or writing research reports etc, it might be useful to receive reviews from others.
- You want a review from someone, or you want to review the work of someone.
- You work on different computers and want to have the same version of files, without copying everything everytime.

Rules of good scientific practice:

- Documentation of research
- Openness to criticism and doubt expressed by other scientists
- Proper review process
- Open access of research results wherever possible

About git

- Version control system
- Change tracker / history of changes
- Issue tracker /review system
- Backup
- Documentation / wiki
- (Automatic) Merge of files
- Parallel development versions

Documentation

README.rst

ctapipe

build passing | codecov 65% | Install with conda

Low-level data processing pipeline software for CTA (the Cherenkov Telescope Array)

This is code is a prototype data processing framework and is under rapid development. It is not recommended for production use unless you are an expert or developer!

- Code: https://github.com/cta-observatory/ctapipe
- Docs: https://cta-observatory.github.io/ctapipe/
- Example notebooks: https://github.com/cta-observatory/ctapipe/tree/master/examples/notebooks

Installation for Users

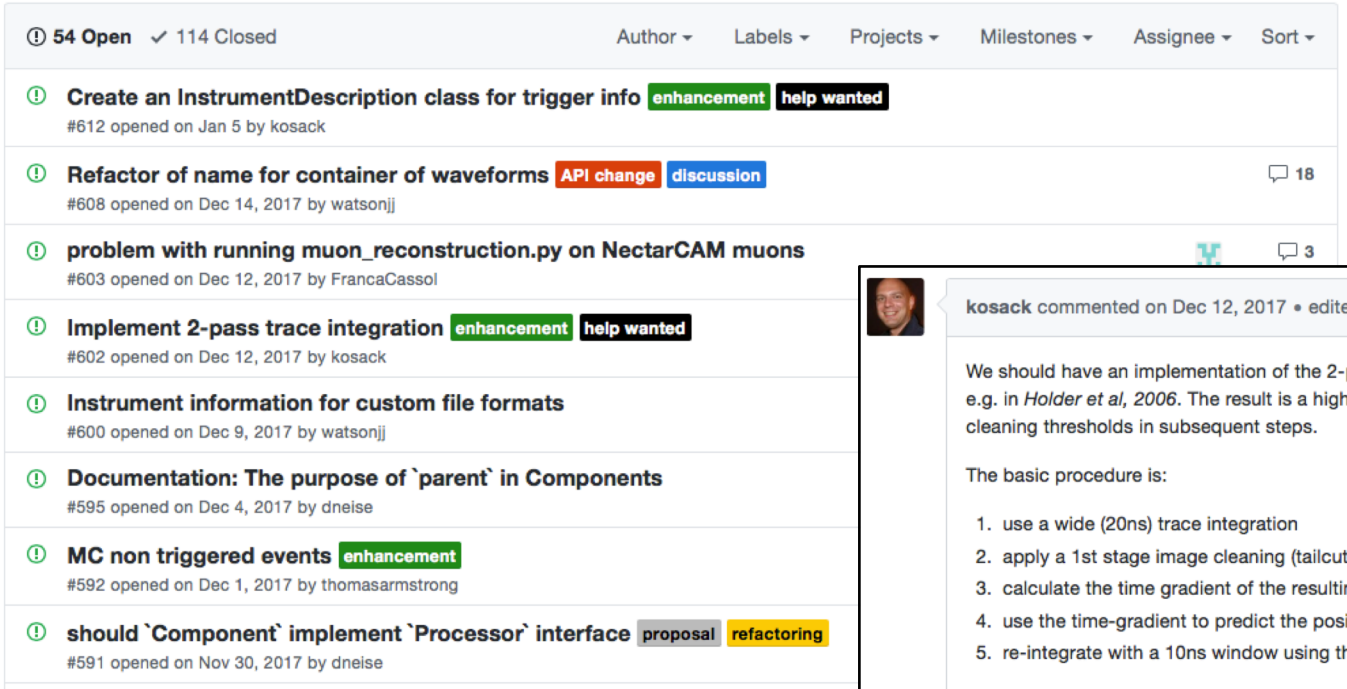
ctapipe and its dependencies may be installed using the *Anaconda* or *Miniconda* package system. We recommend creating a conda virtual environment first, to isolate the installed version and dependencies from your master environment (this is optional).

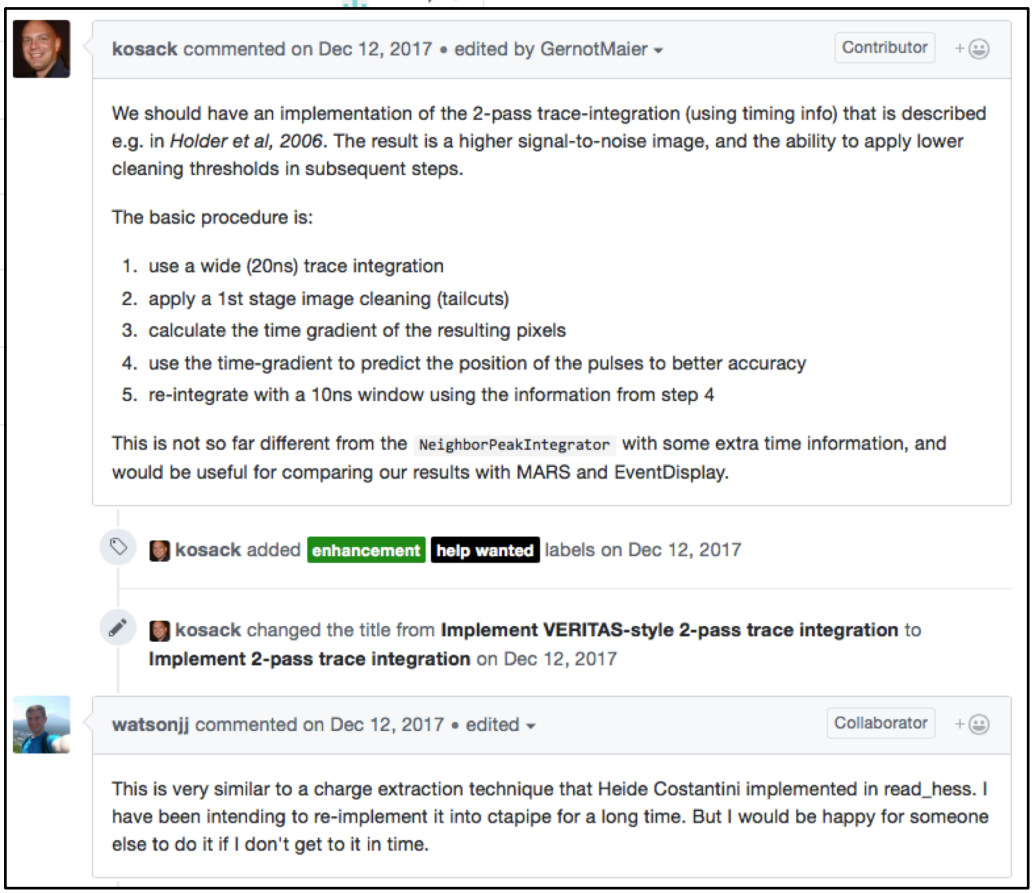
The following command will set up a conda virtual environment, add the necessary package channels, and download ctapipe and its dependencies. The file *environment.yml* can be found in this repo. Note this is *pre-alpha* software and is not yet stable enough for end-users (expect large API changes until the first stable 1.0 release).

```
conda env create -n cta -f environment.yml
source activate cta
conda install -c cta-observatory ctapipe
```

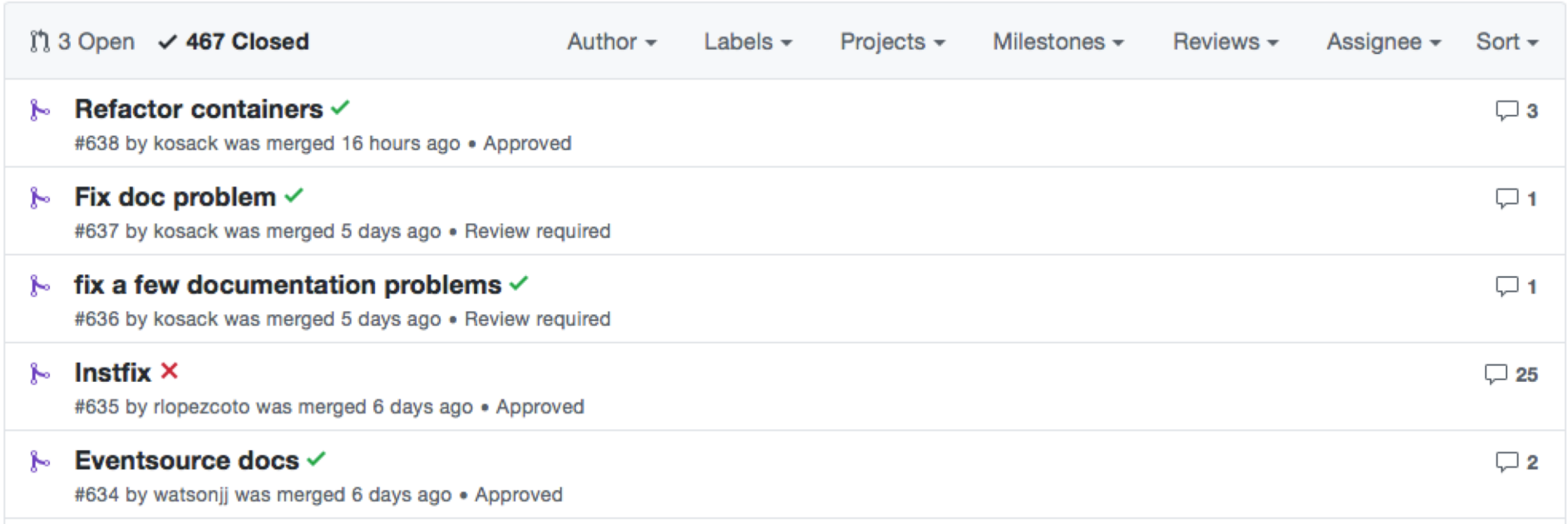
Developers should follow the development install instructions found in the documentation.

Issues


54 Open
114 Closed
Author
Labels
Projects
Milestones
Assignee
Sort
Create an InstrumentDescription class for trigger info enhancement help wanted
#612 opened on Jan 5 by kosack
Refactor of name for container of waveforms API change discussion
18
#608 opened on Dec 14, 2017 by watsonjj
problem with running muon_reconstruction.py on NectarCAM muons
3
#603 opened on Dec 12, 2017 by FrancaCassol
Implement 2-pass trace integration enhancement help wanted
#602 opened on Dec 12, 2017 by kosack
Instrument information for custom file formats
#600 opened on Dec 9, 2017 by watsonjj
Documentation: The purpose of `parent` in Components
#595 opened on Dec 4, 2017 by dneise
MC non triggered events enhancement
#592 opened on Dec 1, 2017 by thomasarmstrong
should `Component` implement `Processor` interface proposal refactoring
#591 opened on Nov 30, 2017 by dneise


kosack commented on Dec 12, 2017 • edited by GernotMaier
Contributor
We should have an implementation of the 2-pass trace-integration (using timing info) that is described e.g. in *Holder et al, 2006*. The result is a higher signal-to-noise image, and the ability to apply lower cleaning thresholds in subsequent steps.
The basic procedure is:
1. use a wide (20ns) trace integration
2. apply a 1st stage image cleaning (tailcuts)
3. calculate the time gradient of the resulting pixels
4. use the time-gradient to predict the position of the pulses to better accuracy
5. re-integrate with a 10ns window using the information from step 4
This is not so far different from the NeighborPeakIntegrator with some extra time information, and would be useful for comparing our results with MARS and EventDisplay.
kosack added enhancement help wanted labels on Dec 12, 2017
kosack changed the title from Implement VERITAS-style 2-pass trace integration to Implement 2-pass trace integration on Dec 12, 2017
watsonjj commented on Dec 12, 2017 • edited
Collaborator
This is very similar to a charge extraction technique that Heide Costantini implemented in read_hess. I have been intending to re-implement it into ctapipe for a long time. But I would be happy for someone else to do it if I don't get to it in time.

Review Requests

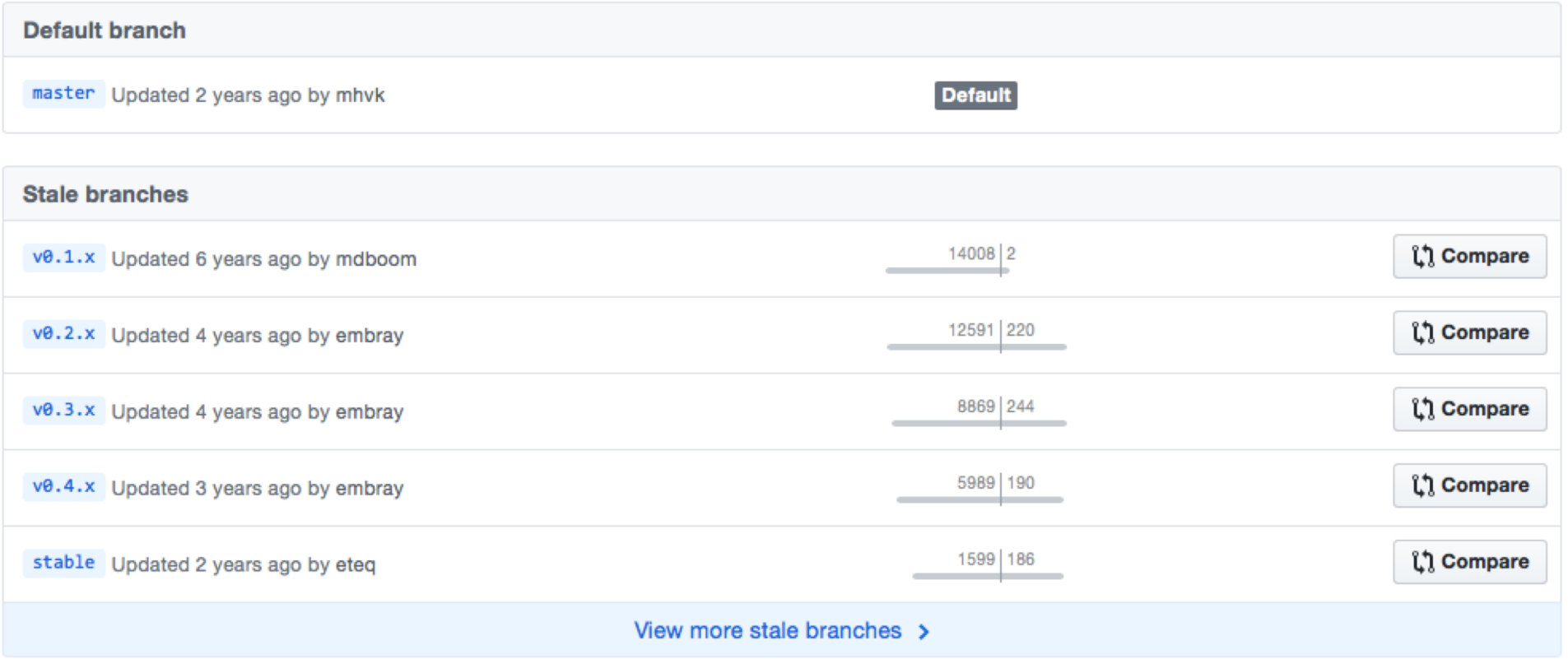


Change Tracker

10 ctapipe/calib/camera/dl0.py — View

```
@@ -58,9 +58,9 @@ def check_r1_exists(self, event, telid):
58  58          Returns
59  59          -------
60  60          bool
61      -           True if r1.tel[telid].pe_samples is not None, else false.
    61  +           True if r1.tel[telid].waveform is not None, else false.
62  62          """
63      -       r1 = event.r1.tel[telid].pe_samples
    63  +       r1 = event.r1.tel[telid].waveform
64  64          if r1 is not None:
65  65              return True
66  66          else:
@@ -82,10 +82,10 @@ def reduce(self, event):
82  82          """
83  83          tels = event.r1.tels_with_data
84  84          for telid in tels:
85      -           r1 = event.r1.tel[telid].pe_samples
    85  +           r1 = event.r1.tel[telid].waveform
86  86              if self.check_r1_exists(event, telid):
87  87                  if self._reductor is None:
88      -                   event.dl0.tel[telid].pe_samples = r1
    88  +                   event.dl0.tel[telid].waveform = r1
```

Parallel Versions



Git Hosts

- GitHub
  - Largest host
  - Many open-source projects (astropy, CTA, etc.)
  - Private repositories for students for free

- Bitbucket
  - Private repositories restricted to 5 people

- GitLab
  - Self hosted options

GitHub

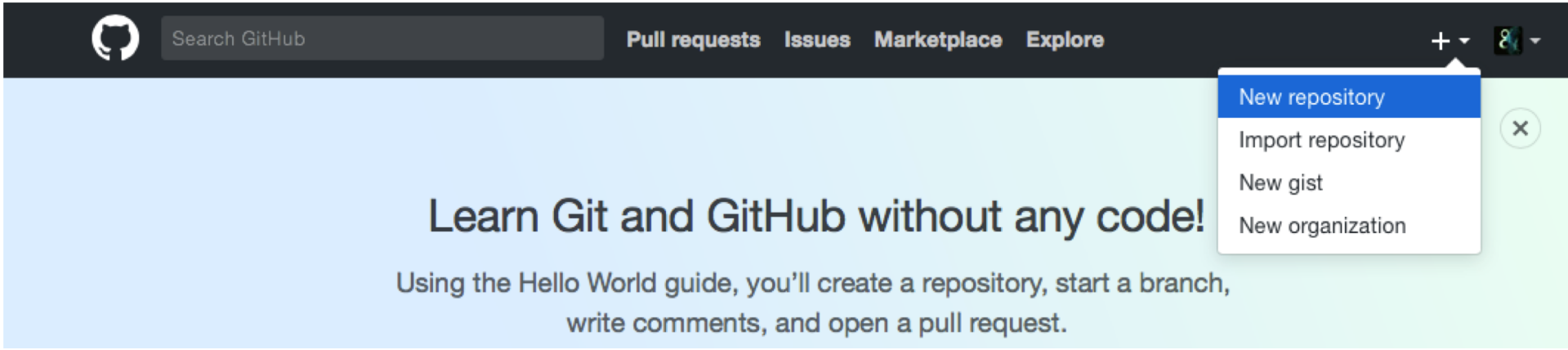
Get a free account with unlimited private repositories:

- https://education.github.com
- Choose 'student' and 'individual account'
- Use your edu mail address
- How do you want to use git?
  - Student, starting honours project
  - Backup and track research
  - Collaborate with other researchers

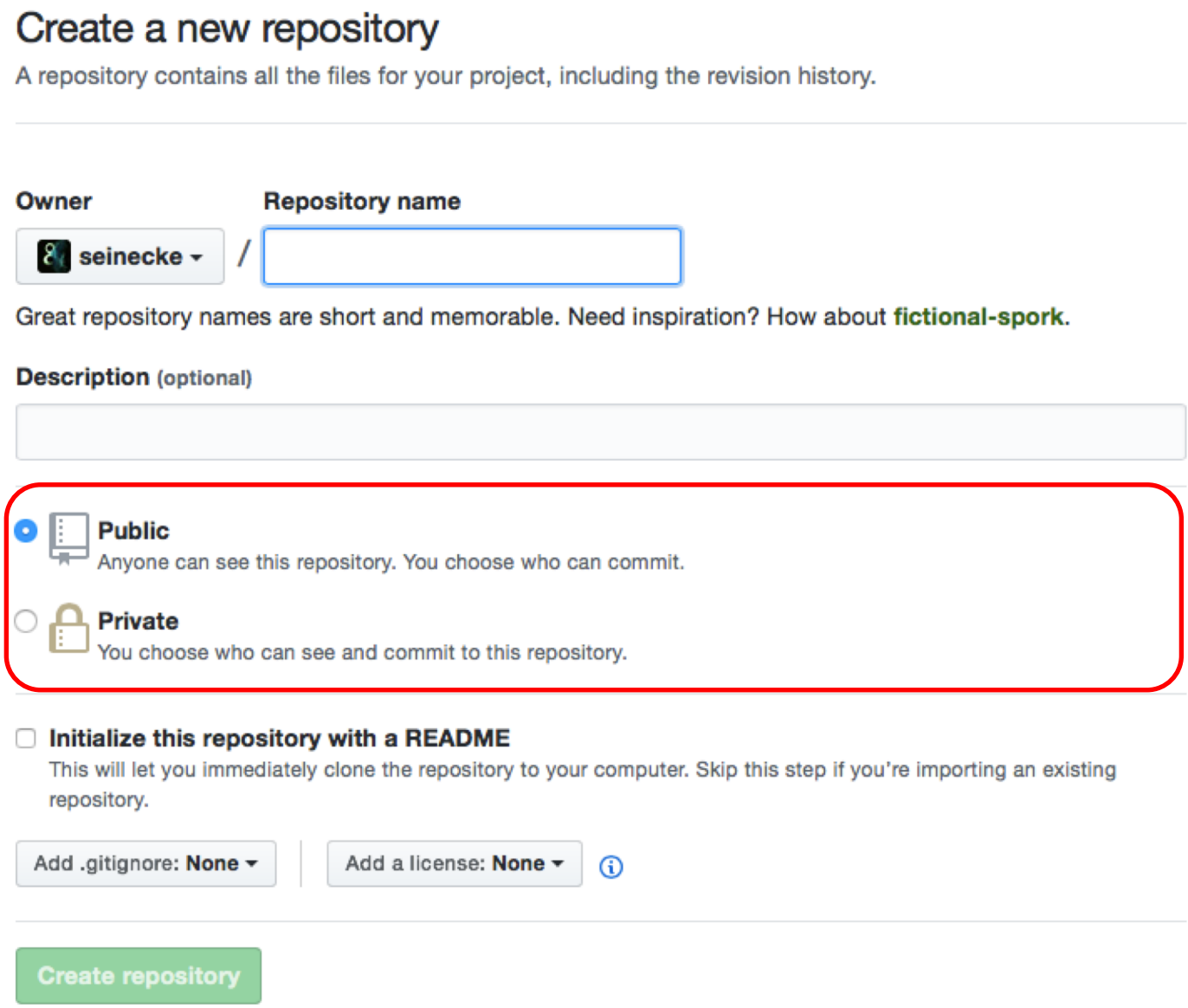
GitLab

- Uni Adelaide is hosting one here: https://gitlab.adelaide.edu.au
- You can use your Uni account to sign in
- You can create public (visible to everyone), private (visible to granted users only) and internal repositories (visible to UofA users)

Create New Repository



Create New Repository



Choose private, if confidential information (internal collaboration code etc.) is to be uploaded

Initialise New Repository

- Create a folder where you store your project:

```
mkdir my_project
cd my_project
```

- Create a file with the description of your project (will also appear on the website):

```
vi README.md
i
# My Project
Description of my project.
:wq
```

- Initialize:

```
git init
```

Initialise New Repository

- Add and commit the file:

```
git add README.md
git commit -m 'inital commit'
```

- Create a new repository on github.com called `my_project`

- Provide information about the host of the repository:

```
git remote add origin https://github.com/YOUR_USERNAME/my_project.git
```

Or for gitlab:

```
git remote add origin https://gitlab.adelaide.edu.au/YOUR_USERNAME/my_project.git
```

- Push the commit to the repository:

```
git push -u origin master
```

Getting Existing Repository

- Go to the folder you want to store the project:

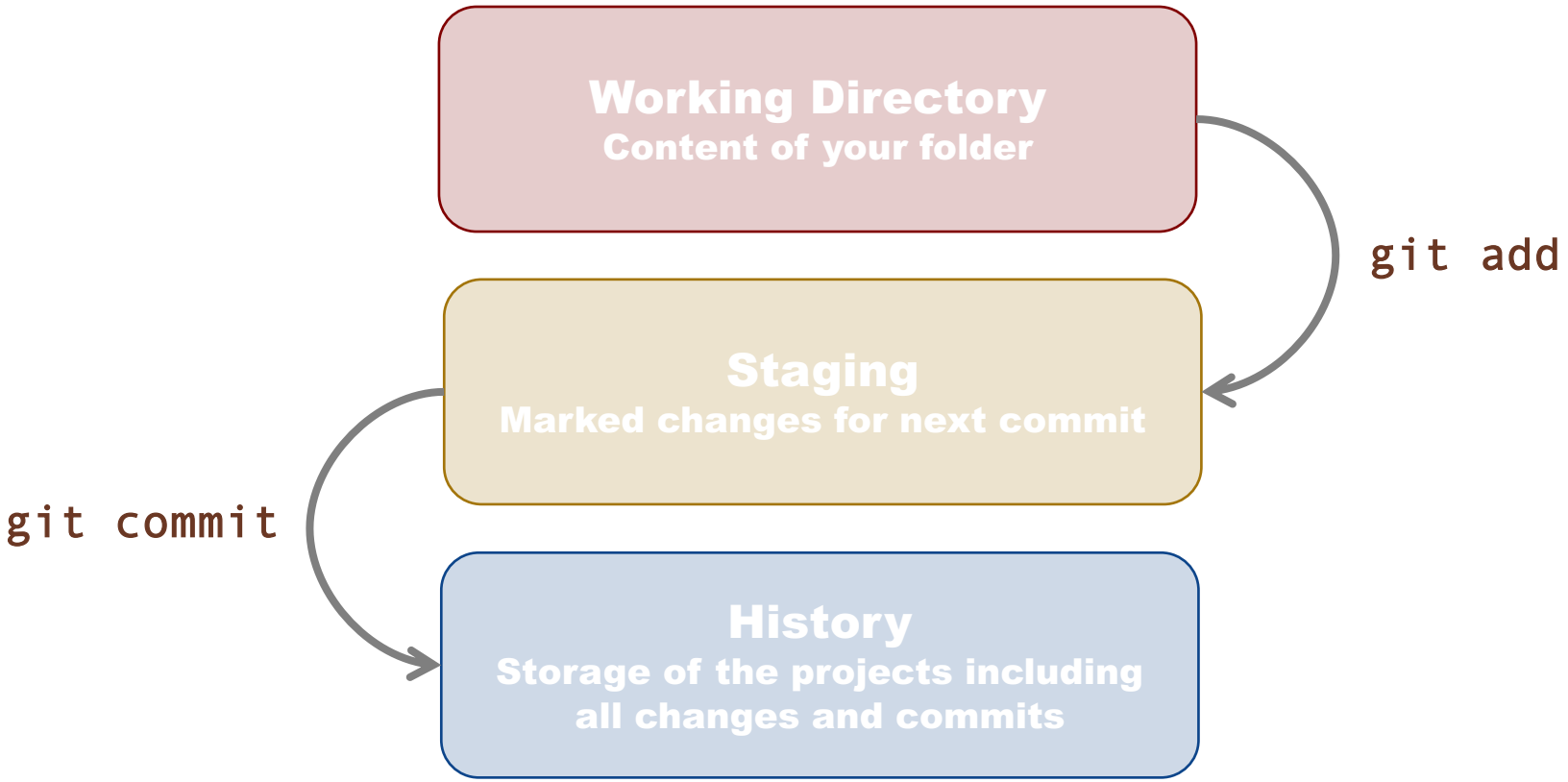
```
cd folder_for_project
```

- Get a clone of a project of a specific user :

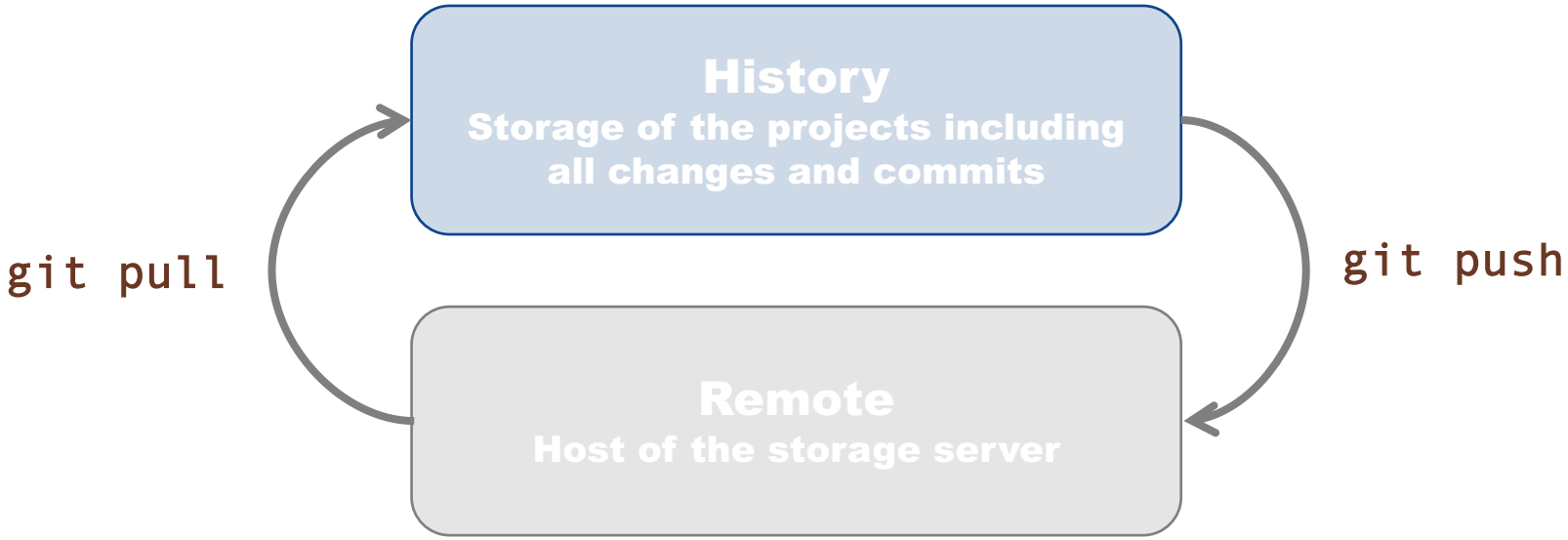
  (This will download the entire project folder to the current folder on your computer.)

```
git clone https://github.com/USERNAME/project.git
```

Working Flow



Working Flow



Further Commands

- `git add` *`file`* adds file to staging
- `git commit` creates commit from staging. and opens editor for commit message → write down the changes the commit includes

- `git status` shows the status of the repository, new and deleted files etc.
- `git log` lists all commits
- `git reset` *`file`* deletes file from staging

- `git diff` shows differences between staging and working directory
- `git diff --staged` shows differences between last commit and staging
- `git diff` *`commit1 commit2`* shows differences between two commits

Merge Conflicts

It might happen that 2 people made changes of the same line → automatic merge not possible

- Open the affected file. Both versions of the line are included:

  (They are marked with the conflict markers `<<<<<<<`, `=======`, `>>>>>>>`)

```
<<<<<<< HEAD
version 1 of the line
=======
version 2 of the line
>>>>>> branch-a
```

- Make the changes, delete the conflict markers

- Add your changes and commit

```
git add affected_file
git rebase --continue
```

.gitignore

There might be files which you don't want to track / should not be uploaded (examples):

- **Large data files:** They are large and they probably do not change often. There are better ways to backup them.
- **Compiled code:** You can just compile again.
- **Build output:** You can just run and output again.
- **System files:** Not necessary to redo your work.

In general, the file is uploaded once and from there on only the changes are uploaded and stored. This saves memory.

But this is not possible for all kind of files, so try to reduce commits of file types like .ppt

.gitignore

.gitignore is a simple text file, including all files and folders, which should not be tracked by git

vi .gitignore

build/

*.pdf

output/

*.o

.DS_store