Workshop git + Github

Tammo Jan Dijkema & Thomas Jürges

22 November 2018

github.com/tammojan/gitcourse

If that doesn't fix it, git.txt contains the phone number of a friend of mine who understands git. Just wait through a few minutes of 'It's really pretty simple, just think of branches as...' and eventually you'll learn the commands that will fix everything.

Plan for today

- Introduction / why revision control
- Create a temp git repository locally
- Create commits
- Work with branches / tags / merge
- Create a public github repository of your cool python script
- Merge conflicts
- Improve someone elses project through a pull request

ASTRON
Netherlands Institute for Radio Astronomy

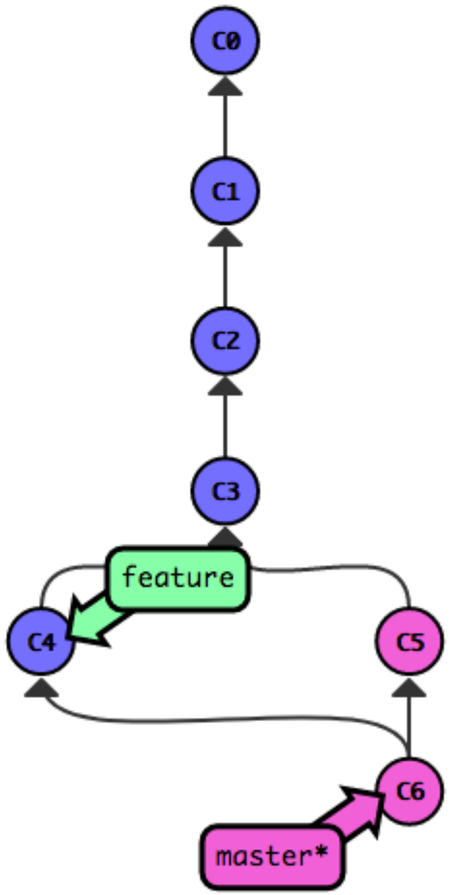
Why revision control

- Keep track of changes: who changed what, when, how, why
- Allow for collaborative development
- Go back to a previous state of the code, undo changes

- Bonus for github: publish / share your code

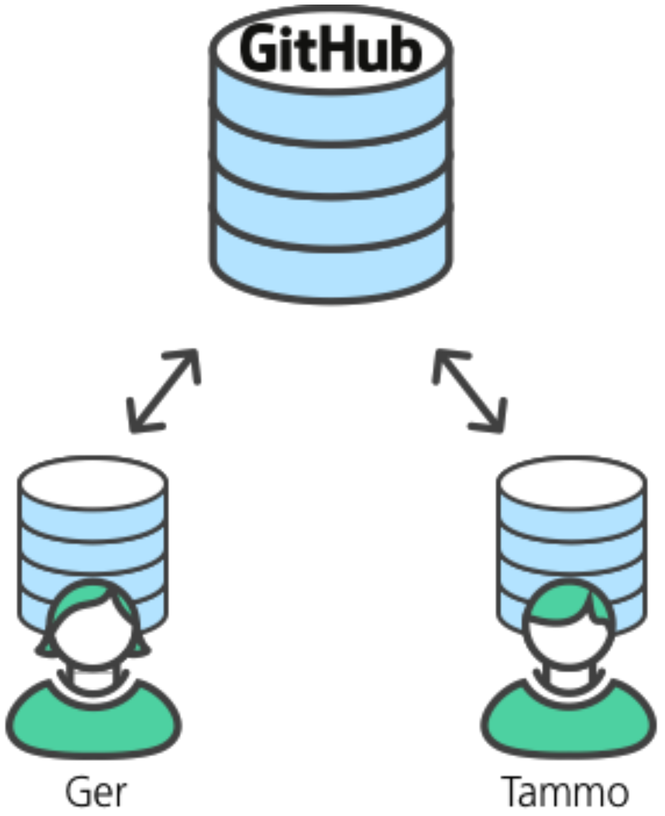
Example repository

- A *commit* represents a state of the repository (project)
- A commit consists of:
  - A pointer to the previous commit
  - Changes w.r.t. the previous commit
  - A commit message (more later)
- The history of the project is kept as a graph



Git is a distributed system

- Every cloned repository has all history
- There is no ‘central’ server necessary
  (although github acts as one)
- Getting repositories in sync requires action
- So you can make and use a repository offline



Tell git who you are

- Check if you've set a username before:
  ```
  git config --list
  ```

- If you get no output for these commands, tell git who you are:
  ```
  git config --global user.name "Tammo Jan Dijkema"
  git config --global user.email "dijkema@astron.nl"
  ```

- Guideline: use your work e-mail for work projects.

- While we're at it: tell git your favorite editor:
  ```
  git config --global core.editor vim
  ```

Create a git repository

- Github repositories do not need a git server

- On your laptop, create a temporary directory:
  ```
  mkdir ~/gittemp
  ```

  ```
  mkdir ~/gittemp
  ```

  ```
  cd ~/gittemp
  ```

- Initialize a new git repository:
  ```
  git init
  ```

- All history of this repository is now tracked in `~/gittemp/.git`

Making your first commit (1)

- In `~/gittemp` make a new file (`touch dummy.txt`)

- See what git thinks of the current state of the repository:
  `git status`

```
On branch master

Initial commit

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

    dummy.txt

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

Making your first commit (2)

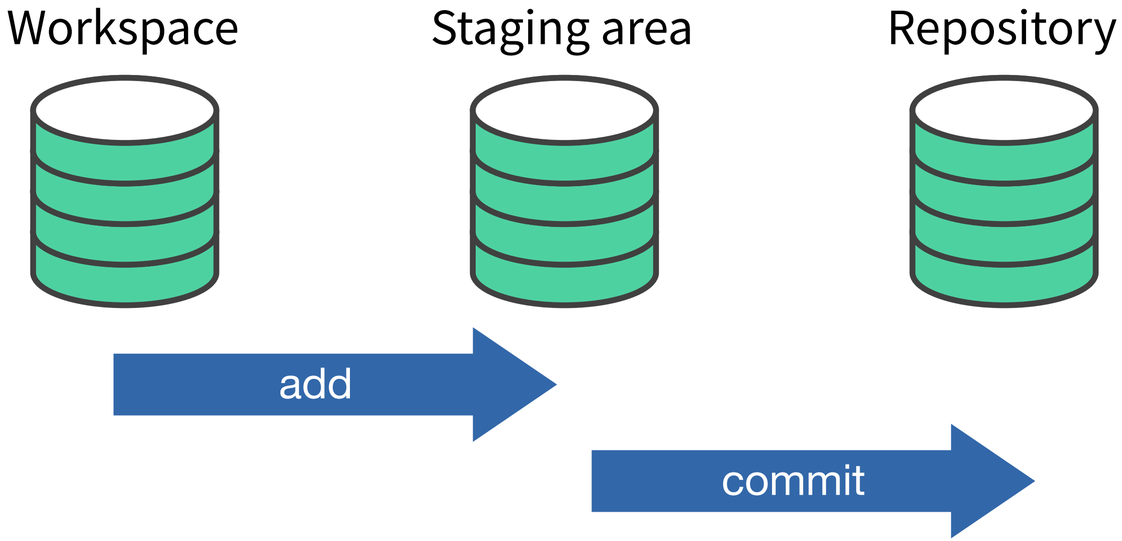
- Git says an untracked file is present, let's tell git to track it:
  `git add dummy.txt`

- See what git thinks of the current state of the repository:
  `git status`

```
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

    new file:   dummy.txt
```

- Nothing has been committed yet!

The staging area



ASTRON
Netherlands Institute for Radio Astronomy

Making your first commit (3)

- Commit the changes (this will commit what git status says will be committed):
  `git commit`

- This gets you your favorite editor to type a commit message. Make one up (e.g. `add dummy.txt`) and type `:wq` to exit your favorite editor.

- Congratulations on your first commit. What is the status now?
  `git status`

```
On branch master
nothing to commit, working tree clean
```

ASTRON
Netherlands Institute for Radio Astronomy

How to write a commit message

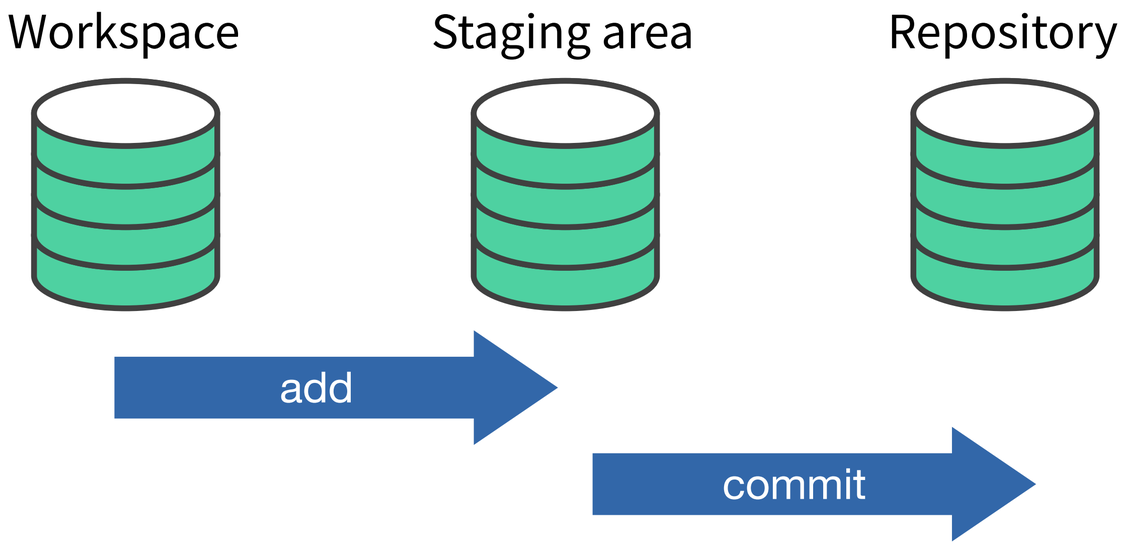
- Reader will be ‘future you’!
- Subject line < 50 characters
- If necessary: blank line, then long message (wrapped at 72)
- Subject line: capital, no period
- In long message, explain why, not how

ASTRON
Netherlands Institute for Radio Astronomy

The staging area

- The staging area shows what will be committed when you do `git commit`

  (This prevents accidental commits of stuff in your working directory.)



Learn this!

- `git status`

  show the staging area, which branch you're on, etc
- `git add`

  add a change to the staging area
- `git commit`

  commit the changes in the staging area

Practice in ~/gittemp

- Change a file, what does it do to `git status`?
- Read the output of `git status` and undo the change.
- Change the file again, and `add` it to the staging area.
- Make some more changes, `add` them to the staging area.
- Commit your changes.
- Check `git log`

Learn this!

- `git status`

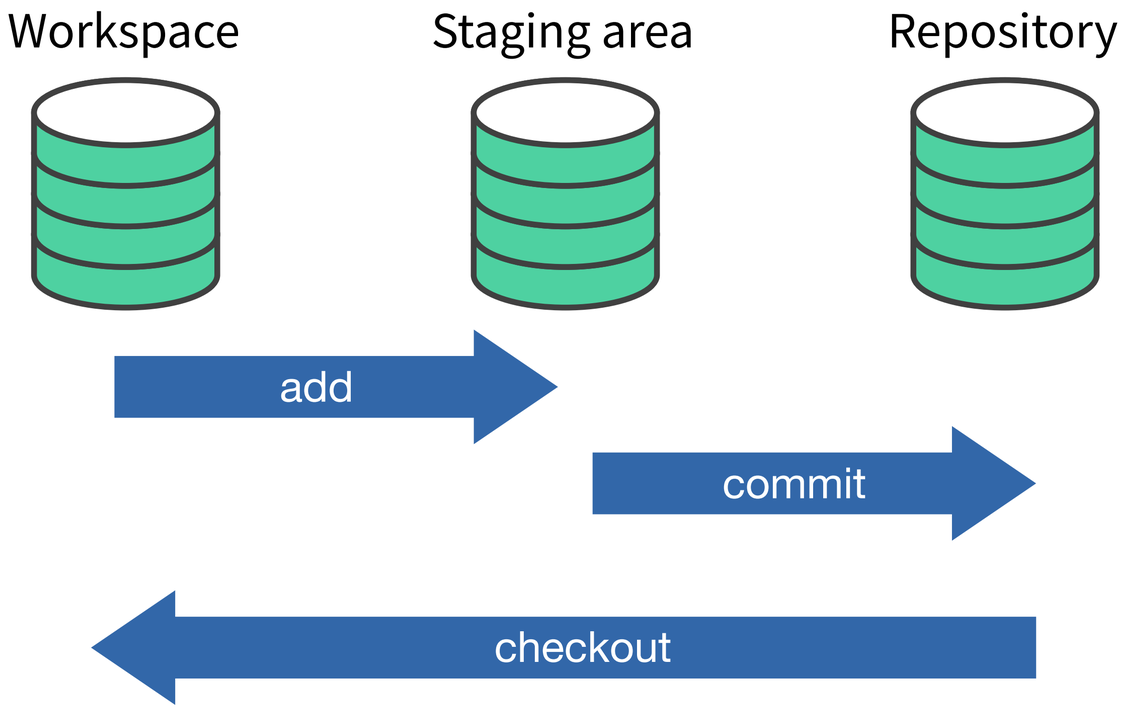
  show the staging area, which branch you're on, etc
- `git add`

  add a change to the staging area
- `git commit`

  commit the changes in the staging area
- `git log`

  see the history (try `--graph` or `--oneline -n 3`)

Navigating around



Branches

- Make a branch (for a single feature!) if:
  - You’re not sure you’ll use it
  - It breaks the existing functionality temporarily

- Branches should be merged to the master as soon as possible

- Demo: https://learngitbranching.js.org/?NODEMO

Learn this!

- `git status`

  show the staging area, which branch you're on, etc
- `git add`

  add a change to the staging area
- `git commit`

  commit the changes in the staging area
- `git log`

  see the history (try `--graph` or `--oneline -n 3`)
- `git checkout`

  navigate to another commit
- `git checkout -b`

  create a new branch from the commit you're at now

Github 

- Github contains a copy of your repository.

- On github, people can ‘fork’ your project and work on their own version of it.

- Challenge is to keep these copies in sync with your local copy.
  (But that’s what git was made for.)

- Many extra features:
  - Pull requests
  - Issue tracker
  - Wiki
  - Web pages
  - Continuous integration

Github

- Create a new github repository for the script you brought
  Initialize with a README

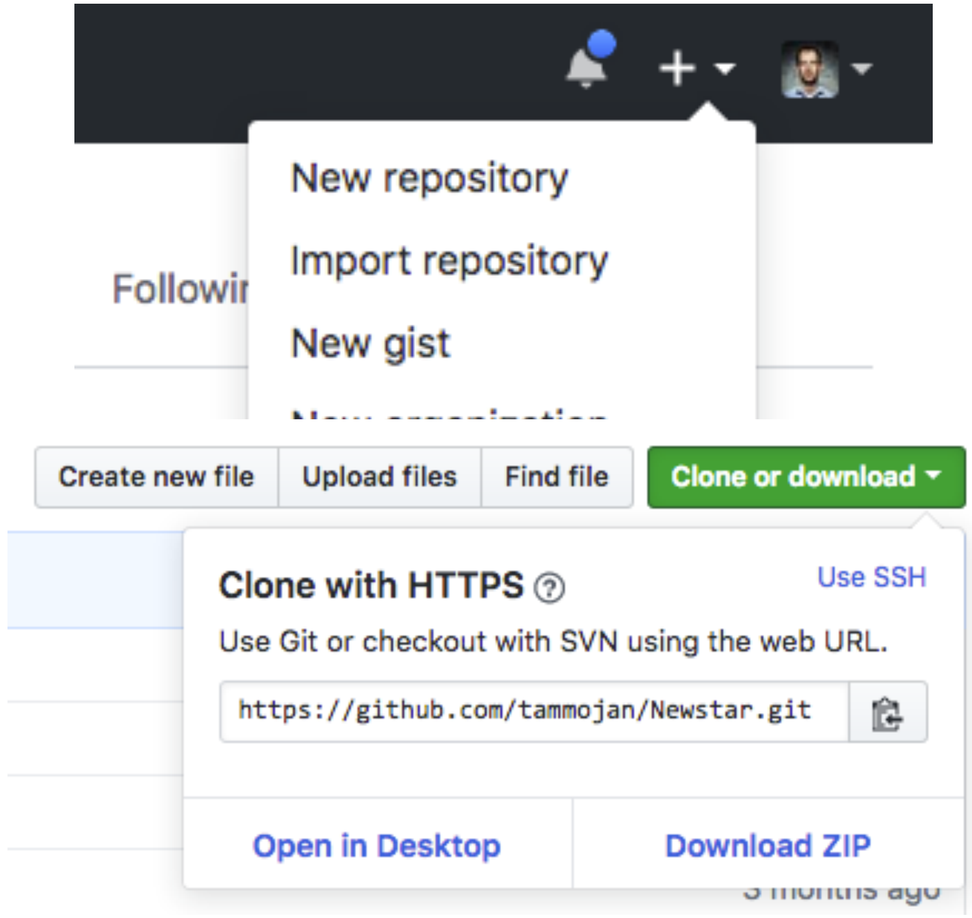
- Clone the repository on your laptop:
  `git clone https://github.com/…`

- Copy your script to the repository directory

- `git status`

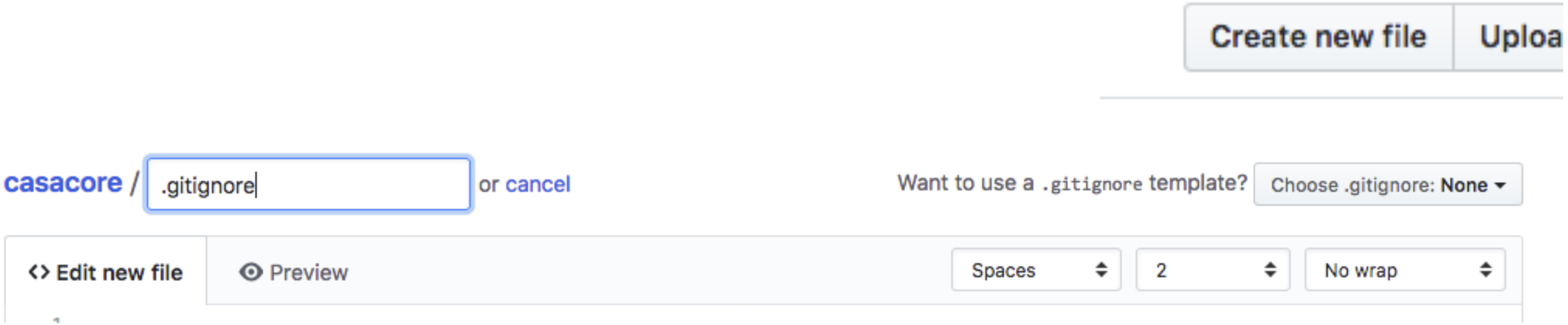
- Add your script to the repository, commit it

- Bring the new commit to github:
  `git push origin master`



Github

- Add a LICENSE and .gitignore to your project through the web interface

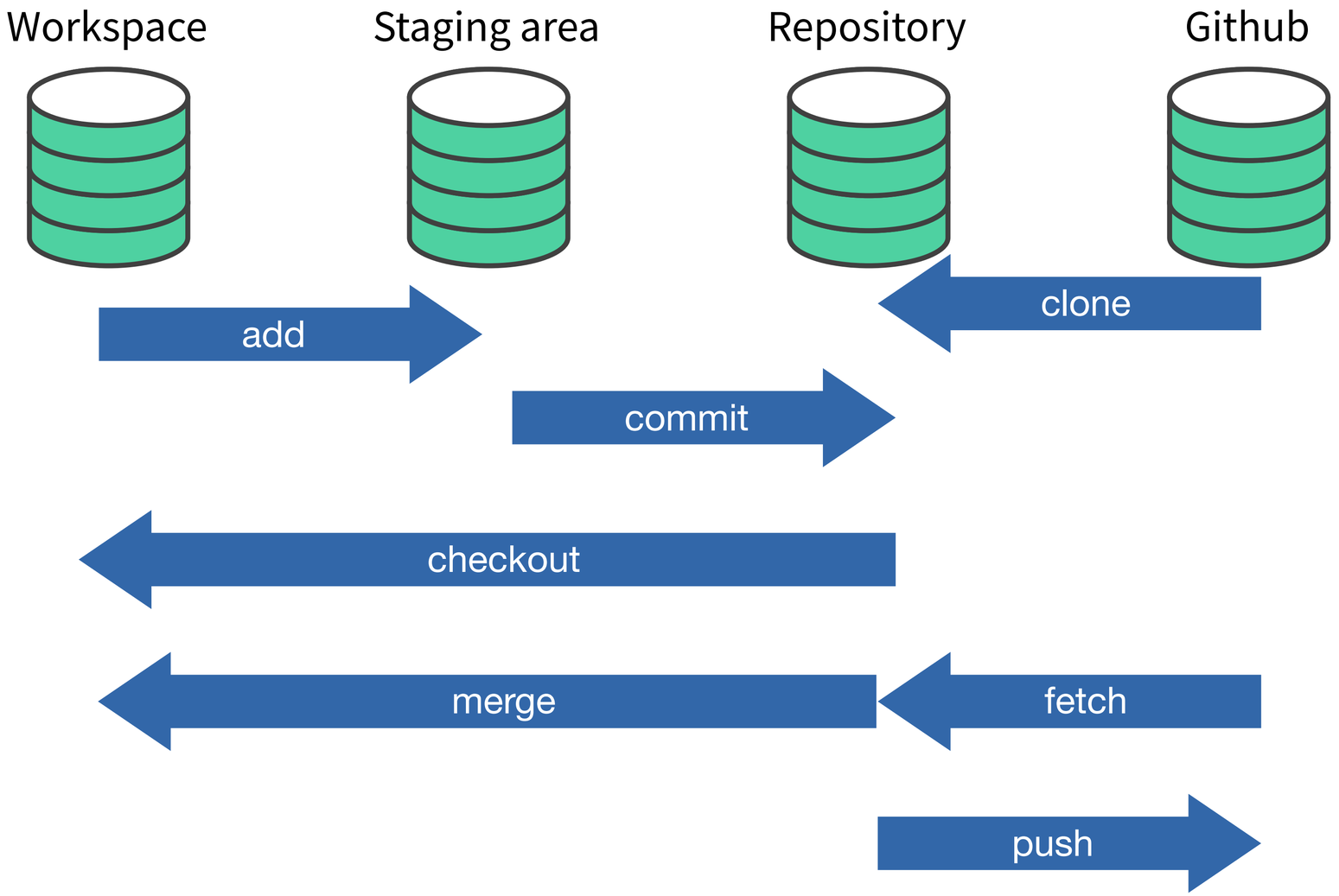


- Get the online changes back to your computer:

  `git fetch origin`

  This only updates the branch origin/master, merge this branch:
  `git merge origin/master`

Working with remote repositories



Learn this!

- `git status`
  show the staging area, which branch you're on, etc
- `git add`
  add a change to the staging area
- `git commit`
  commit the changes in the staging area
- `git log`
  see the history (try `--graph` or `--oneline -n 3`)
- `git checkout`
  navigate to another commit
- `git checkout -b`
  create a new branch from the commit you're at now
- `git clone`
  clone a remote repository to your laptop
- `git merge`
  merge a branch
- `git fetch`
  fetch changes from a remote repository
- `git push`
  push local changes to a remote repository

ASTRON
Netherlands Institute for Radio Astronomy