Version Control with Git

Course outline

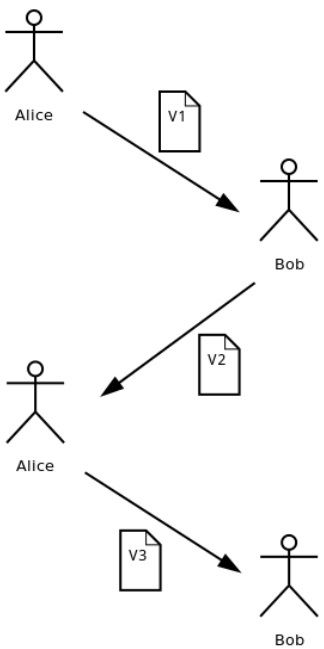
- What is version control? Why use it?
- Configuring
- Basic Git commands (setting up a repository, tracking changes to files, recovering old versions, etc)
- Collaboration
- Dealing with conflicts

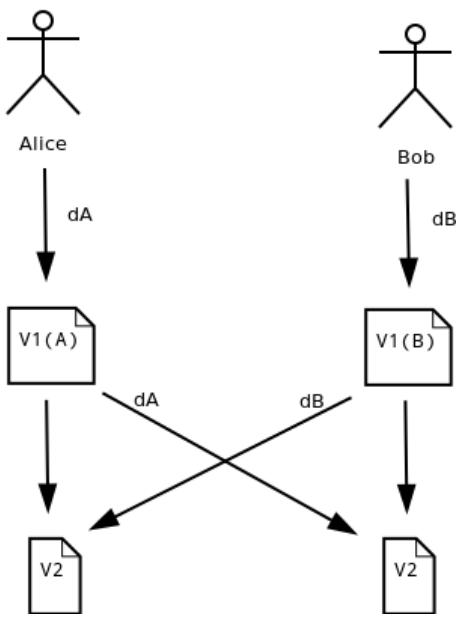
- Based on material by Software Carpentry:
  `http://software-carpentry.org/v5/novice/git/index.html`

What is version control?

The problem

- Imagine you want to work on a document with a colleague.
  - If you take turns: each one will spend a lot of time waiting for the other to finish.
  - If you work on the document at the same time and email changes back and forth: things will be lost, overwritten, or duplicated.





What is version control?

The problem

The filename extension is the portion of a file's name that comes after the final "." character. By convention this identifies the file's type: .txt means "text file", .png means "Portable Network Graphics file", and so on. These conventions are not enforced by most operating systems: it is perfectly possible to name an MP3 sound file homepage.html.

The filename extension is the portion of a file's name that comes after the final "." character. By convention this identifies the file's type: e.g. .txt means "text file". These conventions are not enforced by most operating systems: it is perfectly possible to name an MP3 sound file homepage.html. Since many applications use filename extensions to identify the MIME type of the file, misnaming files may cause those applications to fail.

What is version control?

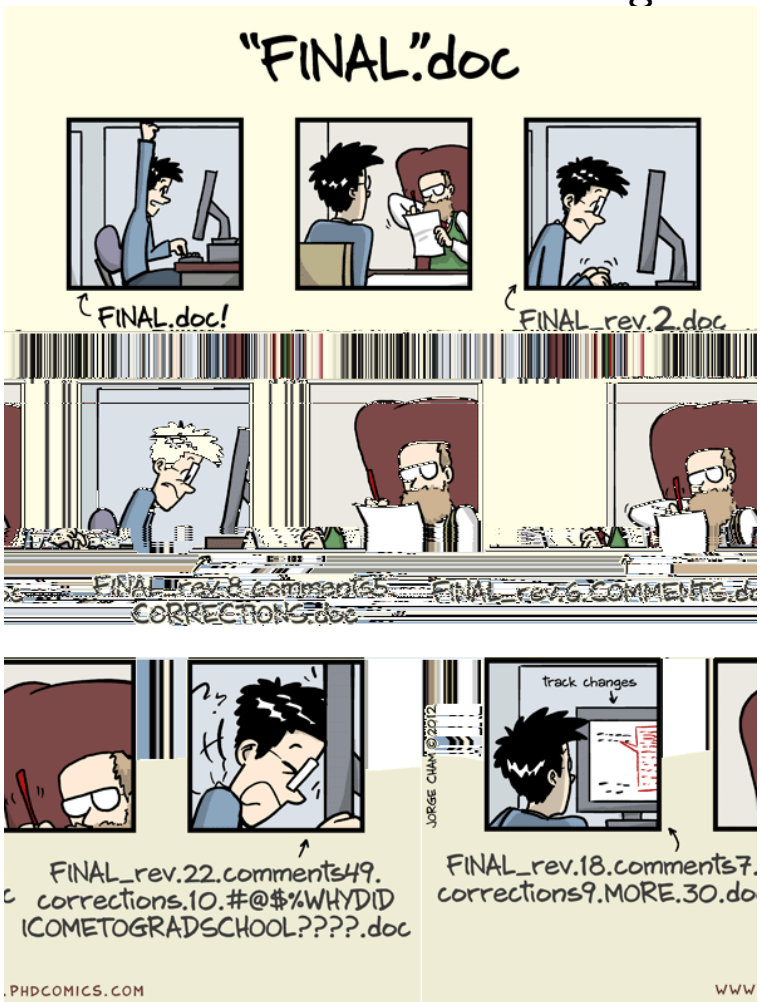
The solution

- **Version control**: a tool for managing changes to a set of files.
- All old versions of files are saved - no changes are ever lost. Always possible to go back in time.
- It keeps a record of who made what changes when.
- It's harder to accidentally overlook or overwrite someone's changes.

What is version control?

A better kind of backup

- Many people choose to keep track of the various changes to a file by simply backing up its contents into a new file each time a significant change is made. However, version control is much better at tracking changes.



"Piled Higher and Deeper" by Jorge Cham, http://www.phdcomics.com

What is version control?

A better kind of backup

sudoku-backup.py	30.9 kB	Python script	Sat 26 Jan 2013 00:11:58 GMT
sudoku-backup-2.py	18.7 kB	Python script	Sat 26 Jan 2013 00:11:58 GMT
sudoku-genetic-algorithm.py	23.1 kB	Python script	Sat 26 Jan 2013 00:11:58 GMT
sudoku-genetic-algorithm-1.py	18.9 kB	Python script	Sat 26 Jan 2013 00:11:58 GMT
sudoku-genetic-algorithm-2.py	21.5 kB	Python script	Sat 26 Jan 2013 00:11:58 GMT
sudoku-genetic-algorithm-2 - Copy.py	24.2 kB	Python script	Sat 26 Jan 2013 00:11:58 GMT
sudoku-genetic-algorithm-2 - Copy (2).py	20.9 kB	Python script	Sat 26 Jan 2013 00:11:58 GMT
sudoku-genetic-algorithm-2 - Copy (3).py	32.3 kB	Python script	Sat 26 Jan 2013 00:11:58 GMT
sudoku-genetic-algorithm-2 - Copy (4).py	33.5 kB	Python script	Sat 26 Jan 2013 00:11:58 GMT
sudoku-genetic-algorithm-2 - Copy (5).py	39.6 kB	Python script	Sat 26 Jan 2013 00:11:58 GMT
sudoku-genetic-algorithm-2 - current - 16Feb.py	21.6 kB	Python script	Sat 26 Jan 2013 00:11:58 GMT
sudoku-genetic-algorithm-2-old.py	32.1 kB	Python script	Sat 26 Jan 2013 00:11:58 GMT
sudoku-genetic-algorithm-3.py	36.0 kB	Python script	Sat 26 Jan 2013 00:11:58 GMT
sudoku-genetic-algorithm - Copy.py	10.4 kB	Python script	Sat 26 Jan 2013 00:11:58 GMT
sudoku-genetic-algorithm - Copy (2).py	11.7 kB	Python script	Sat 26 Jan 2013 00:11:58 GMT
sudoku-genetic-algorithm - Copy (3).py	17.2 kB	Python script	Sat 26 Jan 2013 00:11:58 GMT
sudoku-genetic-algorithm - Copy (4).py	23.6 kB	Python script	Sat 26 Jan 2013 00:11:58 GMT
sudoku-genetic-algorithm - Copy (5).py	27.9 kB	Python script	Sat 26 Jan 2013 00:11:58 GMT
sudoku-genetic-algorithm - Copy (6).py	23.7 kB	Python script	Sat 26 Jan 2013 00:11:58 GMT
sudoku-genetic-algorithm - old.py	20.5 kB	Python script	Sat 26 Jan 2013 00:11:58 GMT

What is version control?

Some more terminology

- **Version control**: a tool for managing changes to a set of files.
  - **Git** is just a particular implementation of a version control system. There are others, e.g. Subversion, Bazaar, Mercurial.
- **Repository**: a storage area containing files whose changes are being tracked.
- **Revision**: a recorded state of the repository at a point in time. A revision is created each time a set of changes are applied.
- The revision log facilitates the reliable recovery of old revisions.
- Helps manage conflicting changes made by different users.

8 / 29

What is version control?

Revision log

- When recording/committing/checking-in changes, the differences between the files you are committing and the files from the previous version are recorded in the revision log.

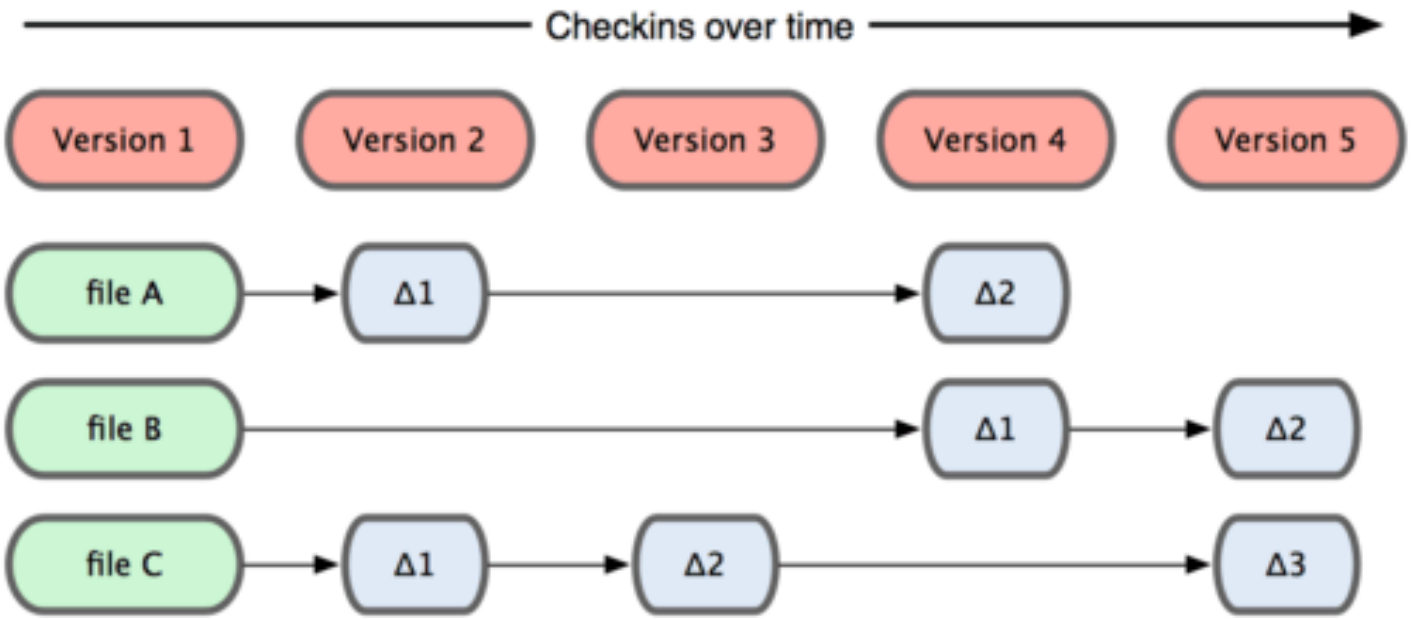


Image by Scott Chacon, used under the Attribution-NonCommercial-ShareAlike 3.0 Unported license. Note that Git works a bit differently to this: each version is essentially a snapshot of the whole repository rather than just a list of changes.

Con guring Git

- To affiliate each change to a particular person, Git needs to know some user information:

```
git config --global user.name "Christian Jacobs"
git config --global user.email "c.jacobs10@imperial.ac.uk"
```

- This information only has to be entered once. Once Git is configured, we can start using it.

Basic Git commands

Creating a repository

- A repository can be created from an existing folder/directory on your computer. Let's create a directory for our work:

```
mkdir planets
cd planets
```

- Now we can tell Git to make it a repository - a place where Git can store old versions of our files:

```
git init
```

- We can check that everything is set up correctly by asking Git to tell us the status of our project:

```
git status
```

Basic Git commands

Tracking changes to files

- Let's create a file called `mars.txt` that contains some notes.

  ```
  nano mars.txt
  ```

  Type the text below into the `mars.txt` file and save it:

  ```
  Cold and dry, but everything is my favorite color
  ```

- If we check the status of our project again using `git status`, Git tells us that it's noticed the new file.

- The "untracked files" message means that there's a file in the directory that Git isn't keeping track of. We can tell Git to track the changes to the file using:

  ```
  git add mars.txt
  ```

Basic Git commands

Tracking changes to files

- Git now knows that it's supposed to keep track of `mars.txt`, but it hasn't yet recorded any changes. To get it to do that, we need to run one more command:

```
git commit -m "Starting to think about Mars"
```

- When we run `git commit`, Git takes everything we have told it to save by using `git add` and stores a permanent copy inside the special `.git` directory. This permanent copy is called a revision and is given a short identifier.

- Use `git log` to list all the revisions in the repository.

Basic Git commands

Changing a file

- Let's open the file mars.txt using

  ```
  nano mars.txt
  ```

  and add another line:

  ```
  The two moons may be a problem for Wolfman
  ```

- When we run `git status` now, it tells us that a file it already knows about has been modified.

- We can use `git diff` to see exactly what has been changed.

Basic Git commands

Changing a file

- Let's commit our change:

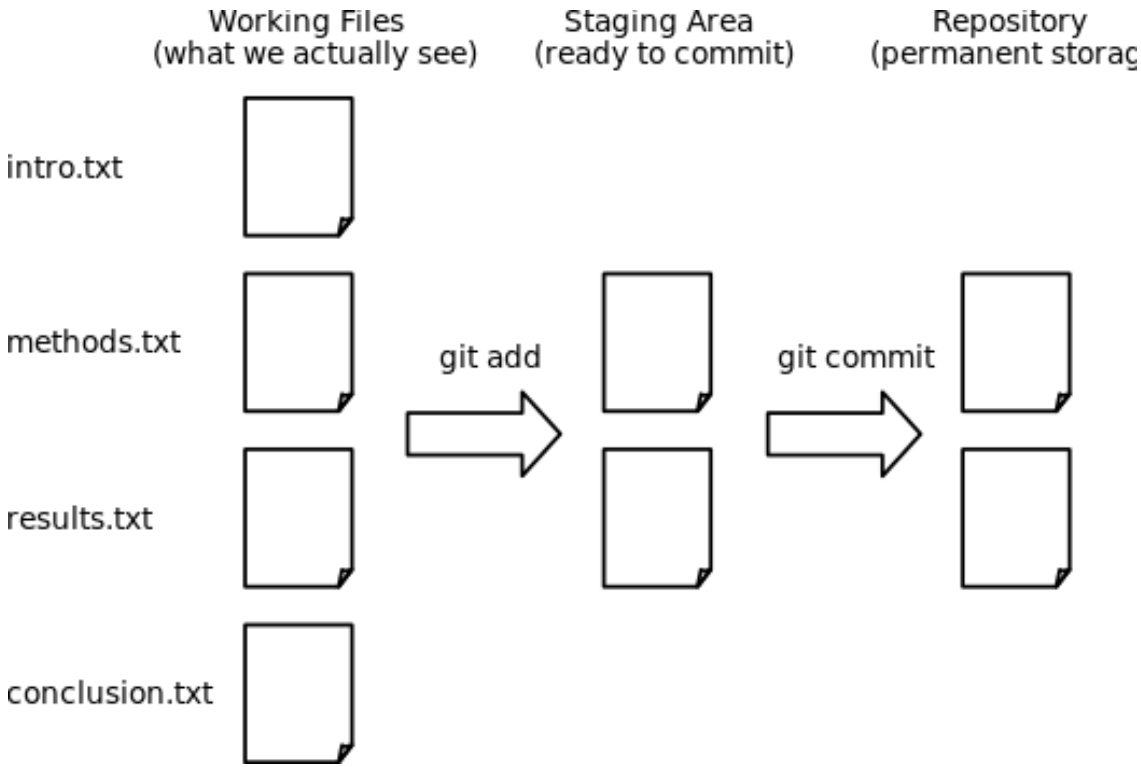
  ```
  git commit -m "Concerns about Mars's moons on my furry friend"
  ```

- Git insists that we add files to the set we want to commit before actually committing anything because we may not want to commit everything at once.

- To allow for this, Git has a special staging area where it keeps track of things that have been added to the current change set but not yet committed:
  - `git add` puts things in this area
  - `git commit` then copies them to long-term storage

Basic Git commands

Changing a file

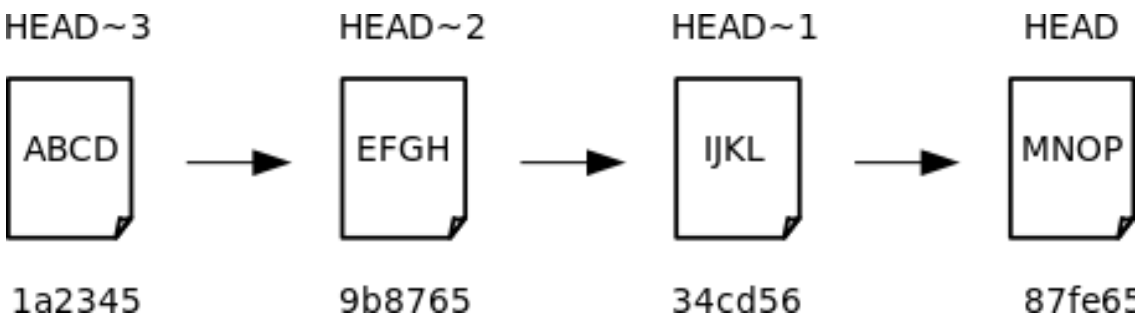


Basic Git commands

Exploring history

- If we want to see what we changed when, we use `git diff` again, but refer to old versions using the notation `HEAD 1`, `HEAD 2`, and so on:

```
git diff HEAD 1 mars.txt
```



- Alternatively, we can use a the commit's identifier, e.g.:

```
git diff f22b25e mars.txt
```

Basic Git commands

Recovering old versions

- Let's modify the `mars.txt` file again. After modifying the file, we can put things back the way they were by using `git checkout`:

  ```
  git checkout HEAD mars.txt
  ```

  As you might guess from its name, `git checkout` checks out (i.e., restores) an old version of a file.

Basic Git commands

Ignoring things

- We can tell Git to ignore certain files by creating a file called `.gitignore` and placing the names of those files inside:

```
nano .gitignore

git add .gitignore

git commit -m "Add the ignore file"
```

Collaboration

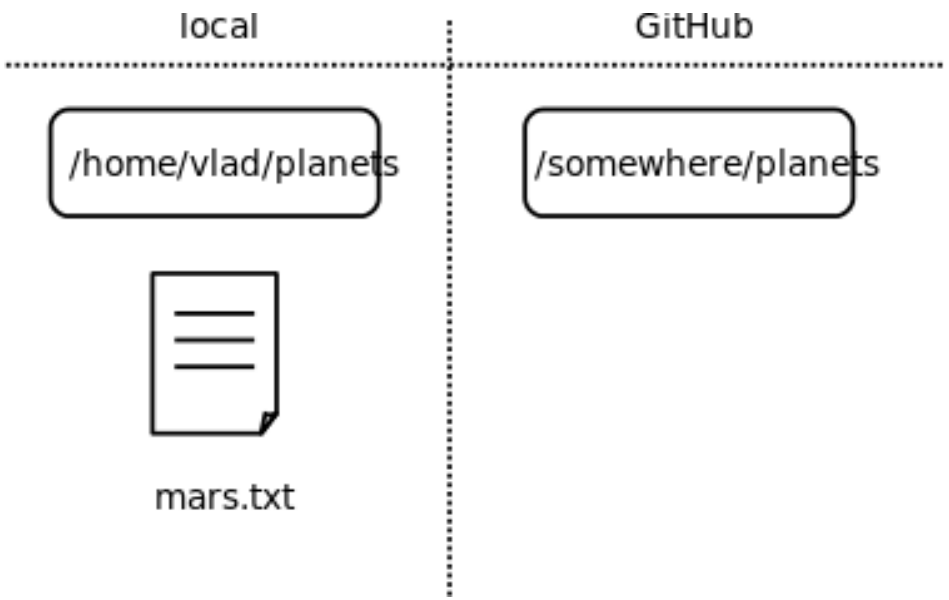
GitHub

- Version control is even more useful when collaborating with other people.
- You can create a version controlled repository online which everyone can commit to.
- GitHub is one of the more popular repository hosting services.

Collaboration

GitHub

- Set up an account on GitHub.
- Create a remote repository called `planets`.



Collaboration

Connecting repositories

- The next step is to connect your local repository with the one on GitHub (known as the remote repository).

```
git remote add origin https://github.com/vlad/planets
```

- The name `origin` is just a nickname for your online GitHub repository.

- We can check that the command has worked using:

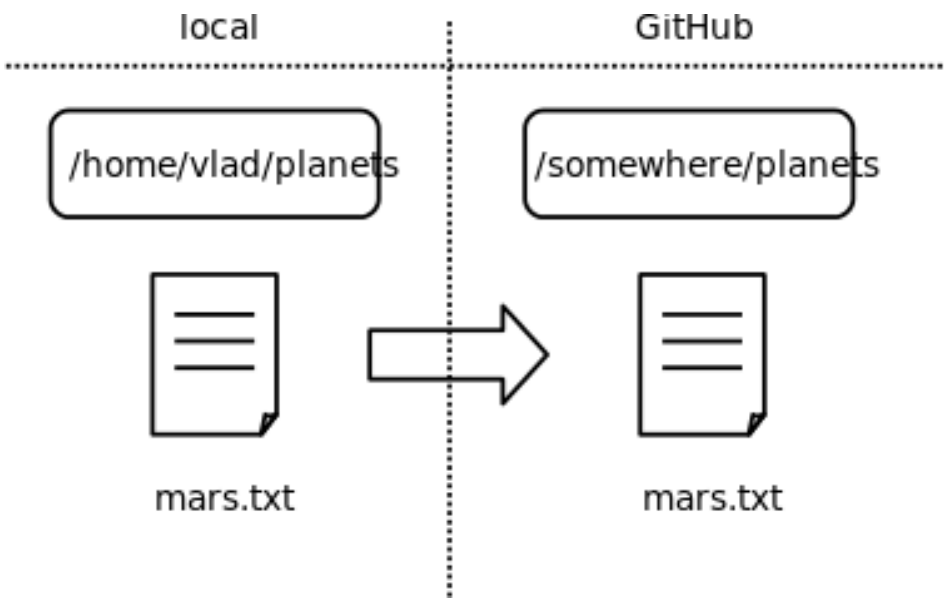
```
git remote -v
```

Collaboration

Pushing/uploading commits to GitHub

- Once the nickname `origin` is set up, we can push (upload) the changes from our local repository to the repository on GitHub:

```
git push origin master
```



Collaboration

Pulling/downloading commits to your local repository

- We can also go the other way. Imagine your colleague has 'pushed'/uploaded some commits to the remote repository. We can 'pull'/download those changes to our local repository using:

```
git pull origin master
```

Collaboration

Cloning a repository

- Imagine you had a local copy of the GitHub repository on the campus computers, but not on your laptop computer.
- ...or imagine that a colleague sets up a new repository on GitHub and you want to have a local copy of it.
- Rather than copying across all the files from computer to computer, you can use `git clone` to obtain a fresh local copy of the GitHub repository:

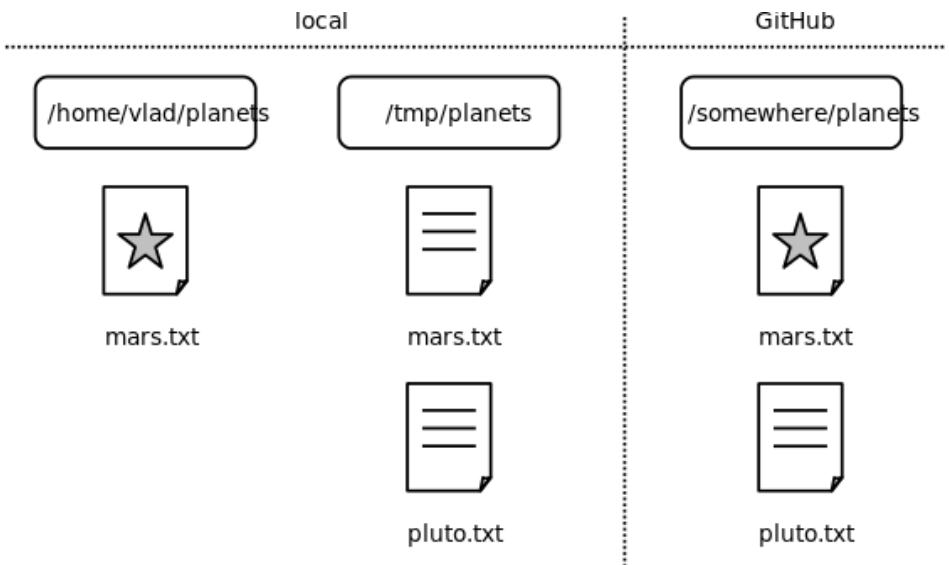
```
git clone https://github.com/vlad/planets.git
```

Resolving con icts

- As soon as people can work in parallel, someone's going to step on someone else's toes.
- **Conflict**: A change made by one user of a version control system that is incompatible with changes made by other users.
- Version control helps us manage these conflicts by giving us tools to resolve overlapping changes.

Resolving con icts

- Let's make some changes to the `mars.txt` file in the `/home/vlad/planets` repository.
- Then commit and push them to GitHub to get:

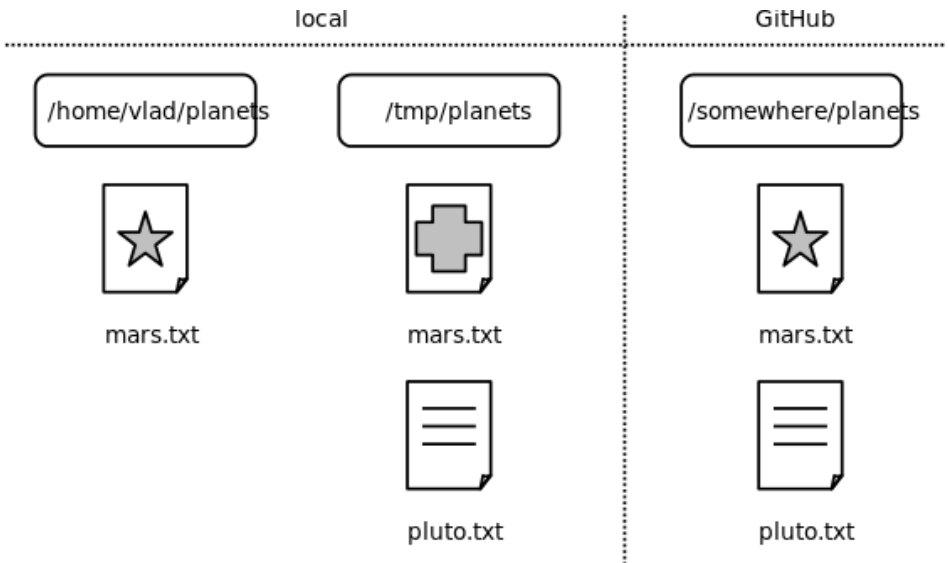


Resolving con icts

- Now let's make some changes to the same text in the `mars.txt` file in the `/tmp/planets` repository.
- Trying to push the changes to GitHub results in an error, because our local repository is out-of-date with the one on GitHub. So, we must do a `git pull`.
- Here, Git encounters a conflict, because we have made two different changes to the same line.
- We need to resolve the conflict by editing the relevant part of the file ourselves.

Resolving con icts

- Once resolved, our repositories will look something like this:



- We can now commit the merged version of the file and push it to GitHub.