SVN and GitHub

Davide Falessi

What is a version Control system (VCS)

- System that keeps track of changes applied to files.
- It allows to go back to a previous (better) version of a file.

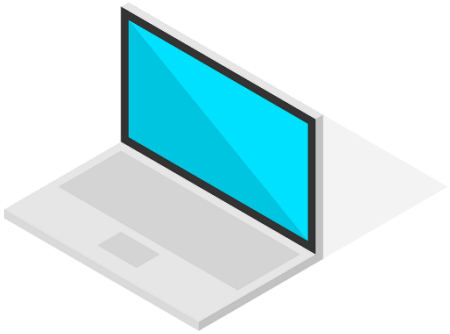
- Centralized (SVN)
  - The shared repository resides in ONE server
  - The users of the repository have only a working copy that needs to be synchronized with the server
    - This is one version of the branch
- Distributed (Git)
  - The full shared repository resides locally

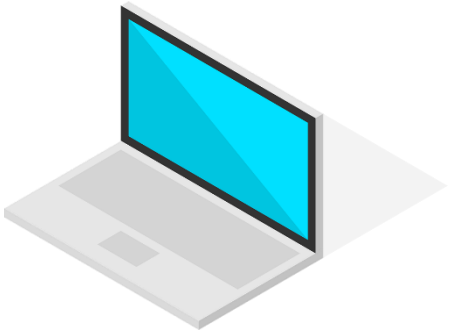
SVN

- Apache Subversion (often abbreviated SVN, after the command name svn) is a software versioning and revision control system distributed as free software under the Apache License.
- Software developers use Subversion to maintain current and historical versions of files such as source code, web pages, and documentation.
- Its goal is to be a mostly compatible successor to the widely used Concurrent Versions System (CVS).

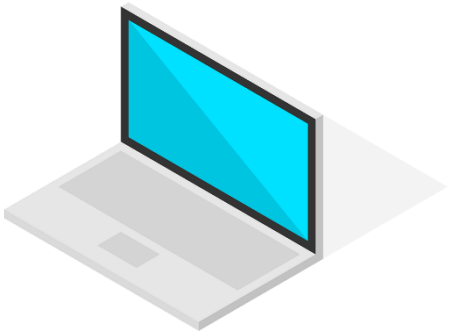
Important commands

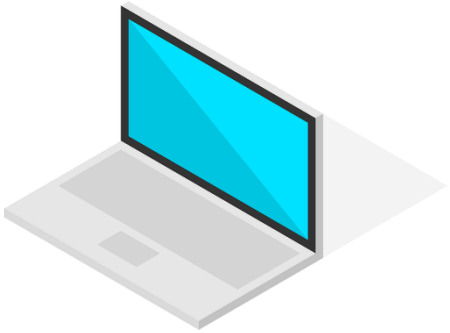
- *svnadmin create:* Create the shared resource.
- *svn checkout:* Create a working copy of the shared resource.
- *svn commit*. This command recursively sends your changes to the SVN server.
- *svn update:* Update the shared resource.
- *svn add*: When you are creating a new file or directory, you need to tell the SVN server about it. This command does that. Note that the file won't appear in the repository until you do an *svn commit* (see below).
- *svn delete*: When you do an **svn commit** the file will be deleted from your local sand box immediately as well as from the repository after committing.

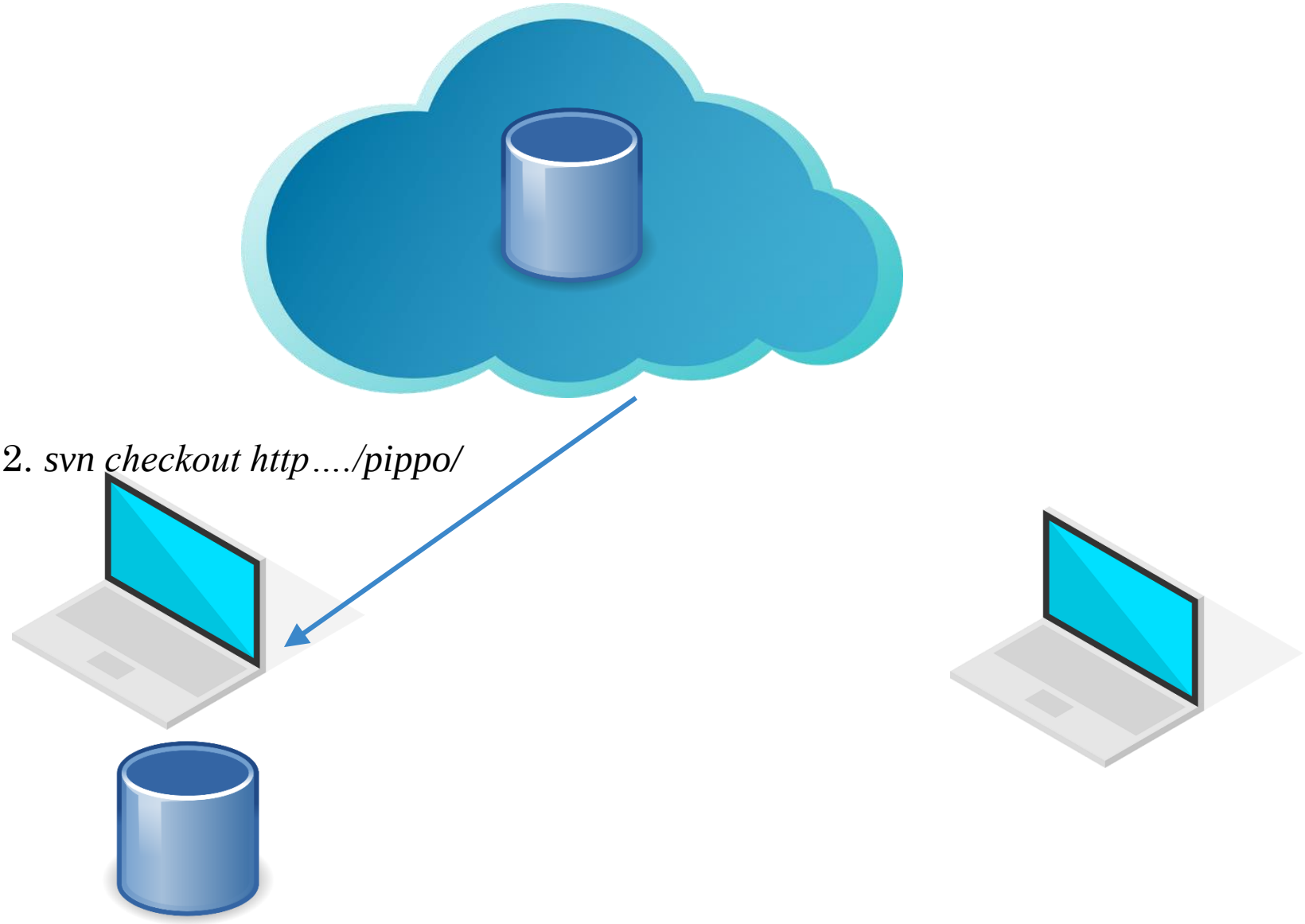




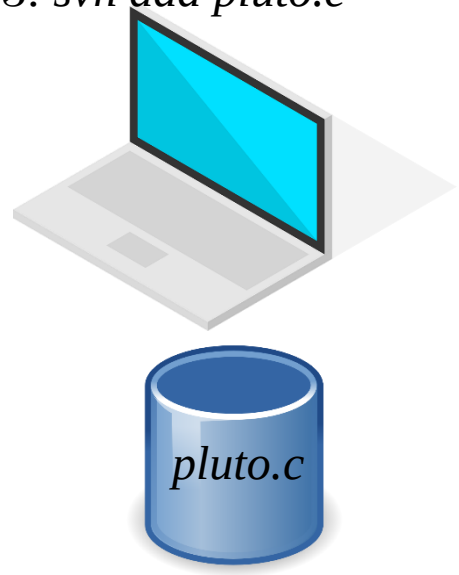
1. *svnadmin create Pippo*

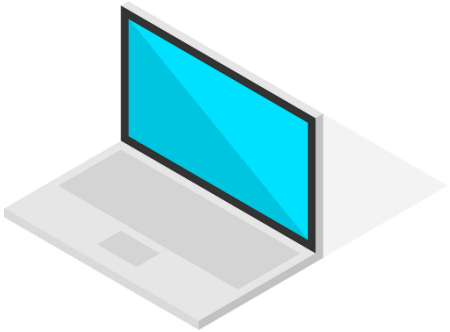


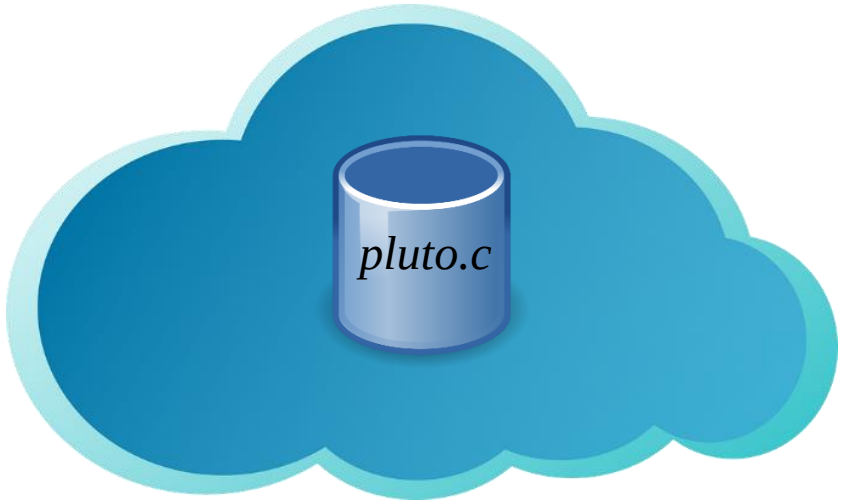



2. svn checkout http…./pippo/

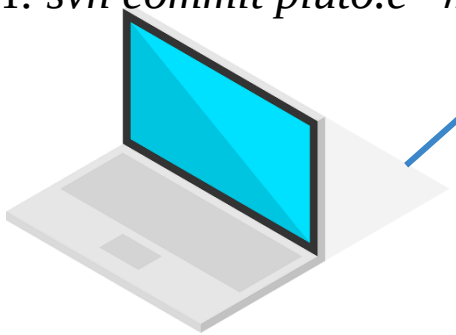
3. *svn add pluto.c*

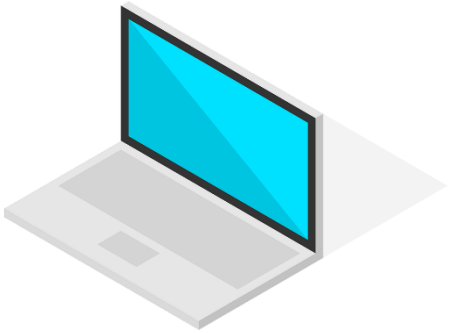


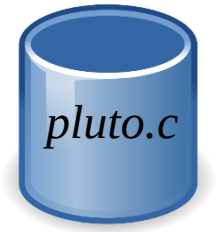


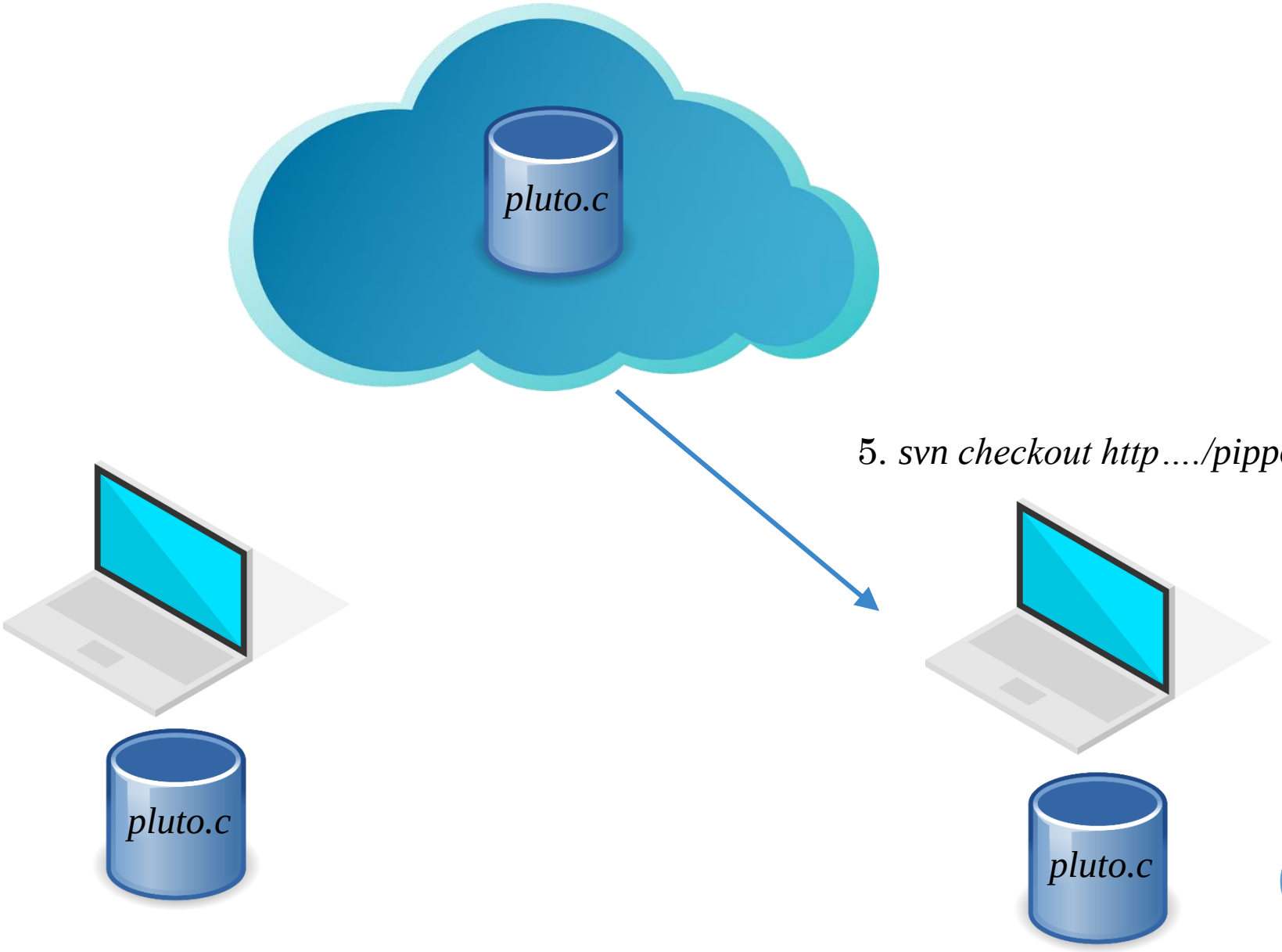


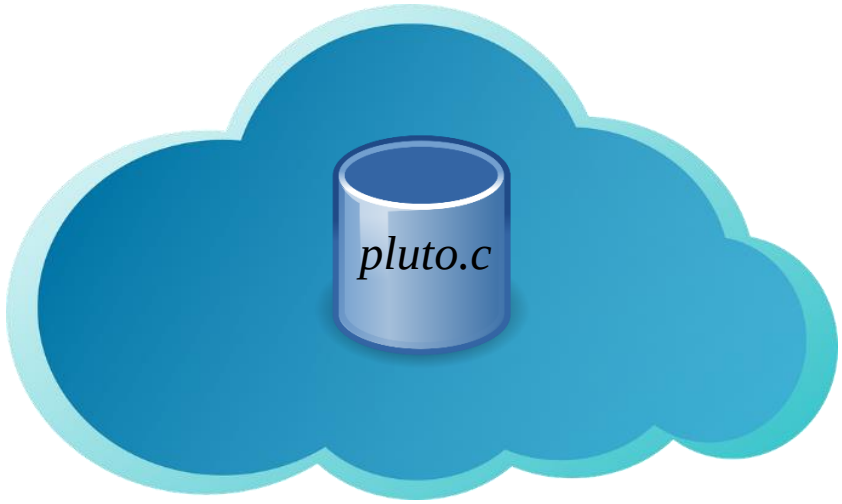
4. *svn commit pluto.c –m “Esempio perfetto”*



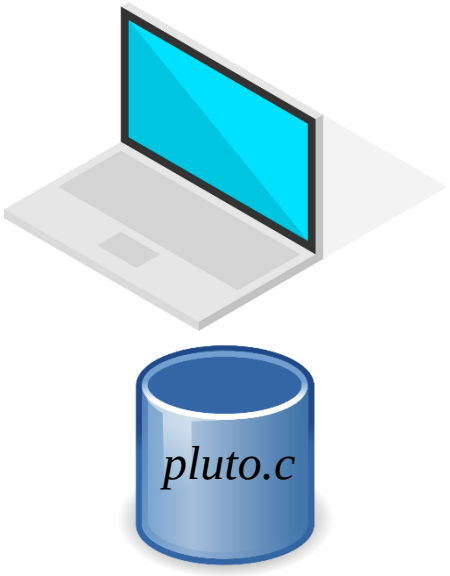


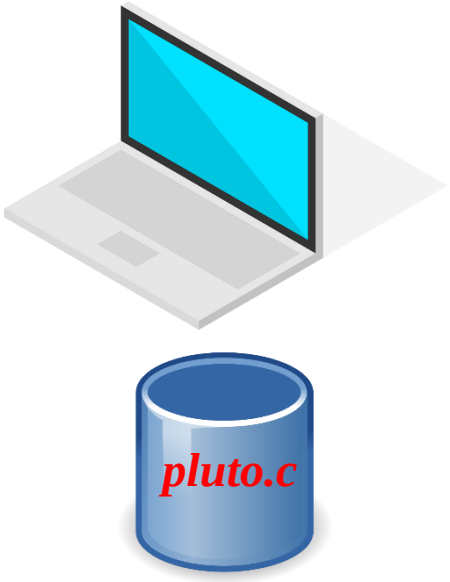


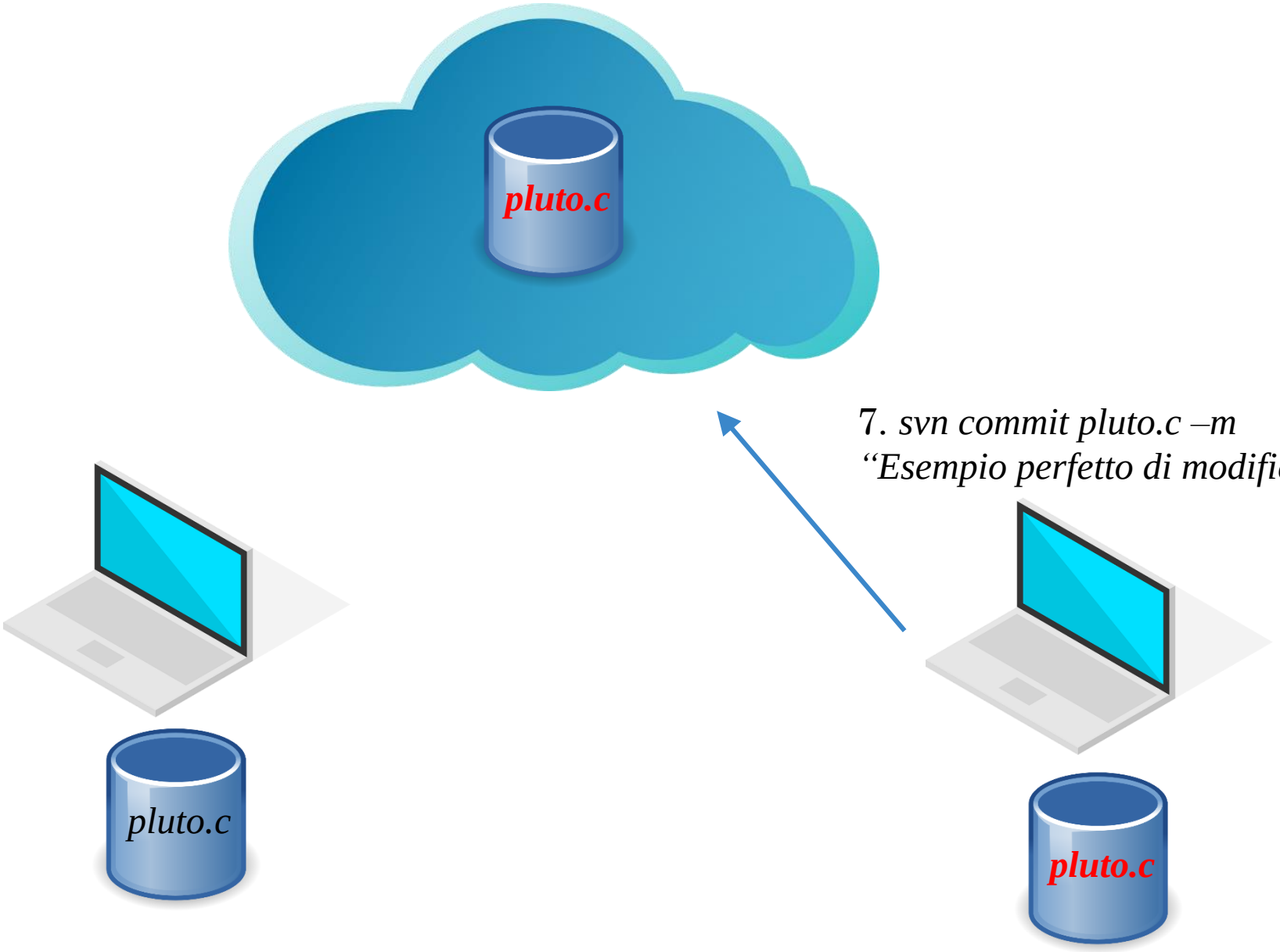

pluto.c
5. svn checkout http…./pippo/
pluto.c
pluto.c

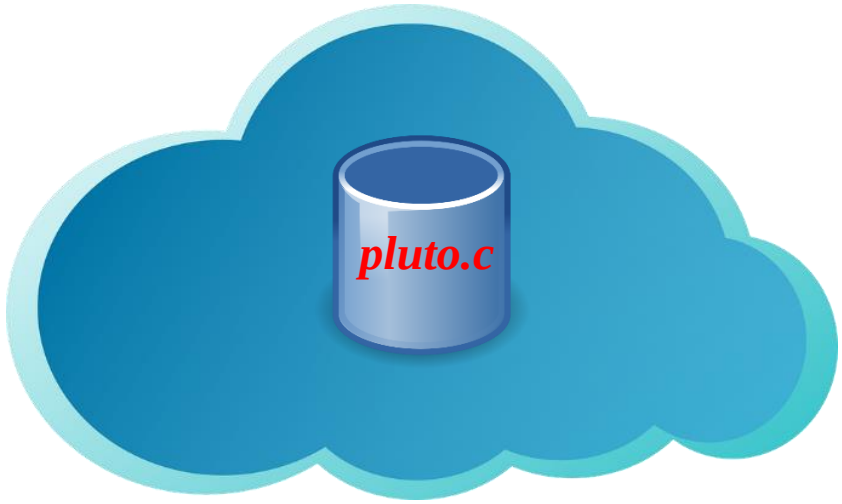


6. … *modifiche su pluto.c* …

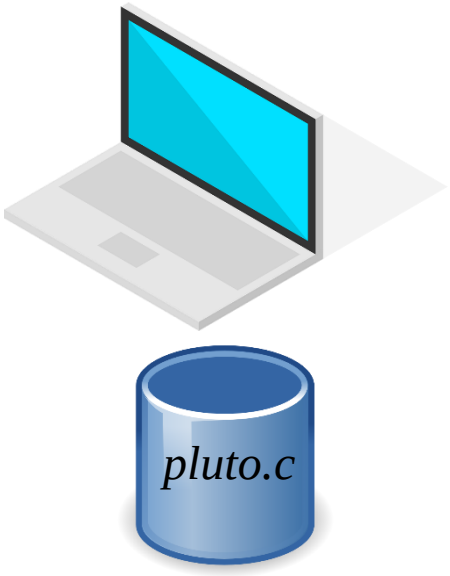


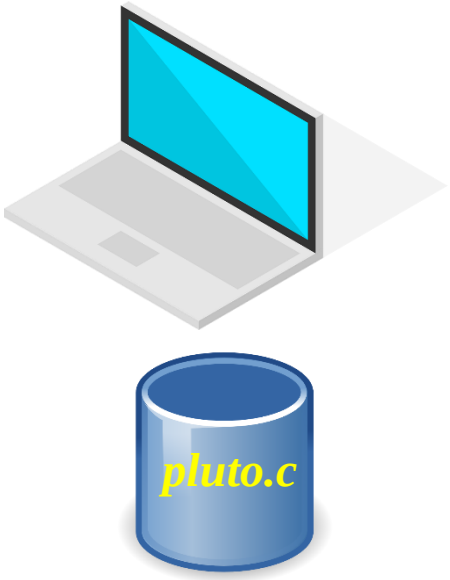


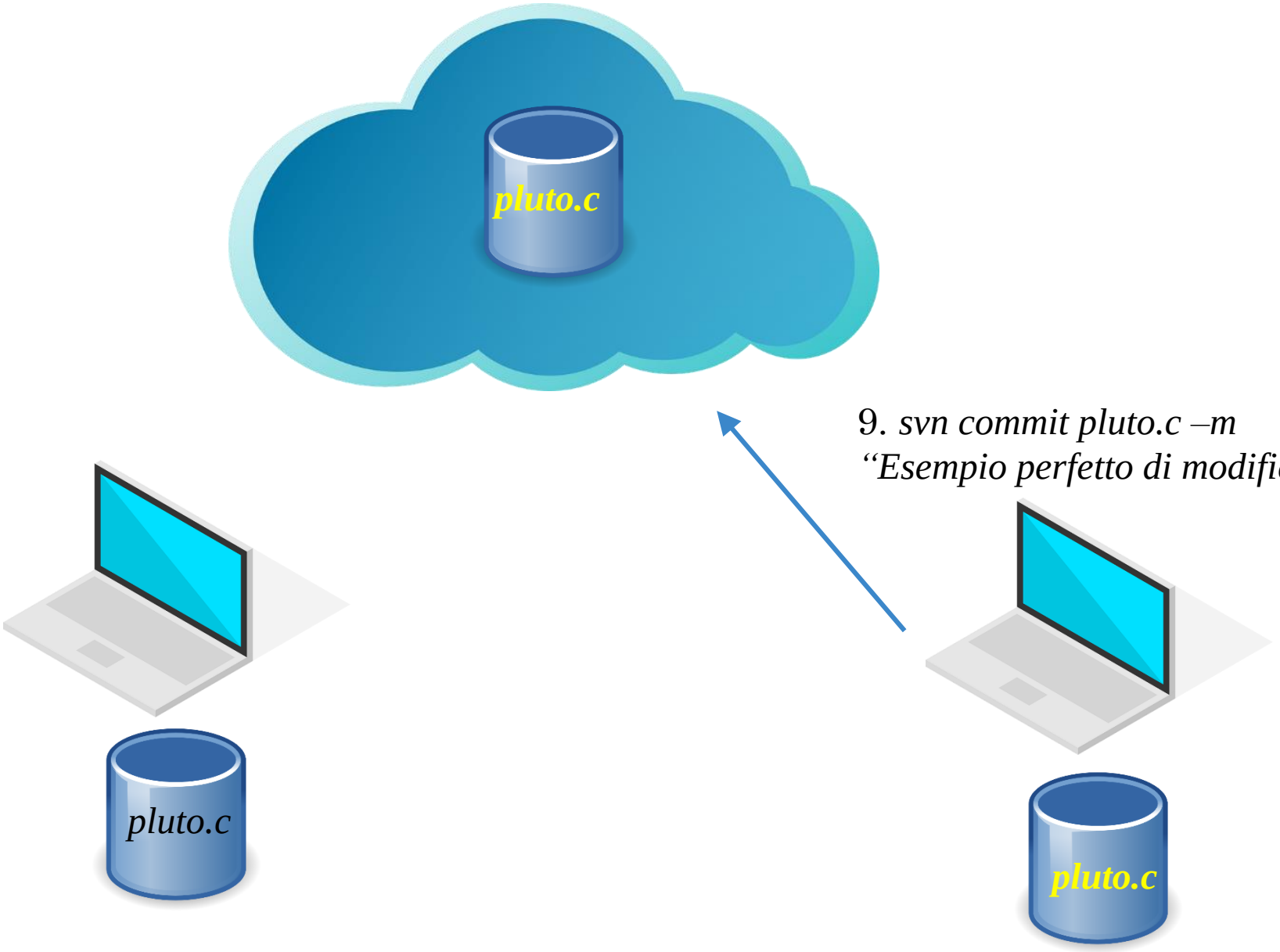

pluto.c
7. svn commit pluto.c –m
“Esempio perfetto di modifica”
pluto.c
pluto.c



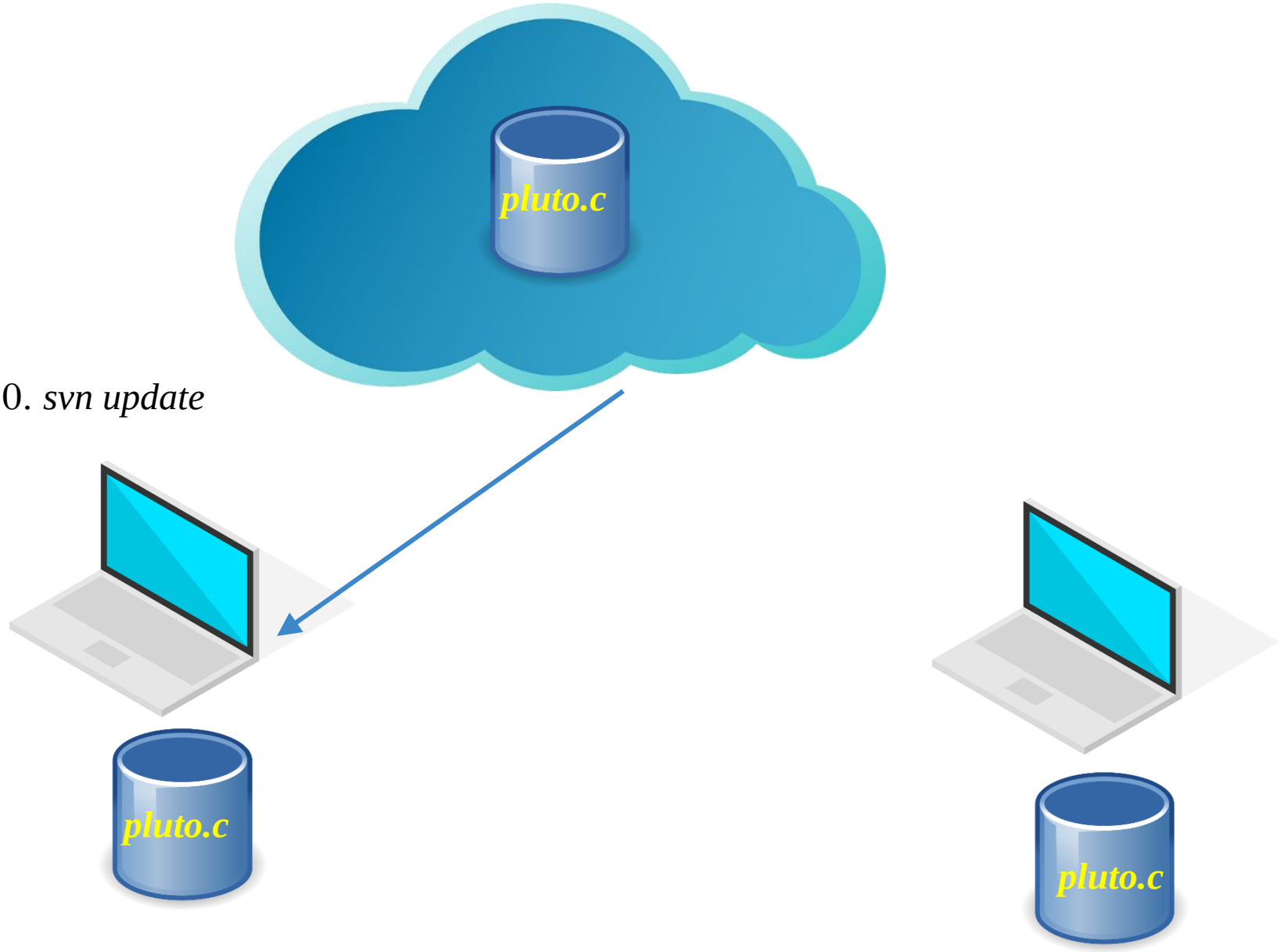
8. *… modifiche su pluto.c …*






pluto.c
9. svn commit pluto.c –m
“Esempio perfetto di modifica”
pluto.c
pluto.c

10. *svn update*



Creating a sharable repository

- In order to share the data, a sharable resource must be created.
- Only one member per project should create the sharable repository.

svnadmin create ProjectName

Creating a Working copy

- All members should have a working copy of the shared resource.
- A working copy is created by "checking out" the shared resource.
- When in the server…
  - Create a folder that will contain your working directory
  - Create the working directory:

$ svn checkout URL

Checked out revision 1.

Updating the repository

- If you have a directory you would like to make it sharable, you should move such a directory into your shared repository.
- *The "import" command* doesn't require a working copy, and your files are immediately committed to the repository.

Working the copy

- Once you have a working copy it is possible to add files or modify the content.
- The paradigm suggests that:
  1. You apply the modifications on your local “working copy”.
  2. You share your changes via “commit”.
  3. A new revision has now been created.
  4. The changes will be visible to the others only after they update the directory.

```
$ type null > button.c
$ svn add button.c
$ svn commit button.c -m "Fixed a typo in button.c."
Sending        button.c
Transmitting file data .
Committed revision 2.
$
```

Updating your copy

- Unlike Dropbox or GoogleDrive, SVN does NOT update your directly automatically when a change is committed by your colleagues.

- In SVN, the working directory must be updated manually via “update”.
- In case you just worked on file button.c and you want to synchronize this change to the rest of the team

$ svn update

U button.c

Updated to revision 57.

$

Common mistakes

- Working on a file without having updated it (before)
- Directly working on the shared resource rather than on than on the working directory.
- GitHub != Git
- GitHub supports different technologies including SVN. You are encouraged to add a Readme file when creating the repo.

SVN vs. Git

- Similar size
- Similar branches activity
- Most of open source projects uses SVN
- Git: no access control, full copy of repository on every computer, no exclusive files locks and so on.
- SVN scales better (there is no limit on the repo size)
- Git history is not reliable

Source: https://svnvsgit.com/

More details

- http://svnbook.red-bean.com/en/1.6/svn.basic.html
- http://svnbook.red-bean.com/en/1.7/svn.branchmerge.using.html

SVN Laboratory

- Each group should have one SVN shared repository “name of the project” in GitHub.
- Each group shall invite me: falessiinguniroma2it
- Each member should create a working copy of it.
- Each member should create 2 file, one file named “Lastname-FR.txt” and “Lastname-US.txt” each reporting 3 functional and 3 user stories respectively.
- Each member should commit these 2 files to the base folder of the repository.

What is an Issue Tracking System?

- It is a system that allows developers (in the large) to create, assign, and track issues (aka. tickets).
- An issue can be:
  - A feature to be developed
  - A bug to be fixed
  - A task to be completed
  - Something similar to the above
- Each issue has:
  - A name (e.g., requirement)
  - A set of fields (e.g., creation date, summary,
  - A workflow (e.g., created, assigned, developed, tested, approved)
  - Set of restrictions: who can create what, set of roles assigned to users, etc.
- Examples
  - JIRA, Redmine, GitHub

GitHub as an ITS

- Very easy to set up
- Additional set-ups
  - Create two new labels: User Story, Functional Requirement.

ITS Lab

- Each member should create one issue for each User Story and Functional Requirement.