# Parallel & Distributed Computing: Lecture 2

Alberto Paoluzzi

October 1, 2019

1. Markdown
2. VCS (Version Control System): Git
3. References

# Markdown

# Data Markup Languages

- Markup languages are designed for the processing, definition and presentation of text.
- The language specifies code for formatting, both the layout and style, within a text file.
- The code used to specify the formatting are called tags.
- HTML is a an example of a widely known and used markup language.

# Data Markup Language Examples

1. XML
2. JSON
3. Document markup languages

# Data Markup: XML (Extensible Markup Language)

Metalanguage which allows users to define their own customized markup languages

```
<!DOCTYPE glossary PUBLIC "-//OASIS//DTD DocBook V3.1//EN">
 <glossary><title>example glossary</title>
  <GlossDiv><title>S</title>
   <GlossList>
    <GlossEntry ID="SGML" SortAs="SGML">
     <GlossTerm>Standard Generalized Markup Language</GlossTerm>
     <Acronym>SGML</Acronym>
     <Abbrev>ISO 8879:1986</Abbrev>
     <GlossDef>
      <para>A meta-markup language, used to create markup
languages such as DocBook.</para>
      <GlossSeeAlso OtherTerm="GML">
      <GlossSeeAlso OtherTerm="XML">
     </GlossDef>
     <GlossSee OtherTerm="markup">
    </GlossEntry>
   </GlossList>
  </GlossDiv>
 </glossary>
```

# Data Markup: JSON (JavaScript Object Notation)

Is often used for serializing and transmitting structured data over a network connection.

```
"glossary": {
    "title": "example glossary",
    "GlossDiv": {
        "title": "S",
        "GlossList": {
            "GlossEntry": {
                "ID": "SGML",
                "SortAs": "SGML",
                "GlossTerm": "Standard Generalized Markup Language",
                "Acronym": "SGML",
                "Abbrev": "ISO 8879:1986",
                "GlossDef": {
                    "para": "A meta-markup language, used to create markup language
                    "GlossSeeAlso": ["GML", "XML"]
                },
                "GlossSee": "markup"
            }
        }
    }
```

# Markdown

- Markdown (Wikipedia)
- Statement by creator John Gruber
- Mastering Markdown

# Headers

```
# This is an <h1> tag
## This is an <h2> tag
###### This is an <h6> tag
```

# This is an <h1> tag

## This is an <h2> tag

### This is an <h6> tag

Figure 1: **

# Emphasis

```
*This text will be italic*
_This will also be italic_

**This text will be bold**
__This will also be bold__

_You **can** combine them_
```

*This will also be italic*

**This text will be bold**

**This will also be bold**

*You **can** combine them*

Figure 2: Emphasis

# Lists

```
### Unordered

    * Item 1
    * Item 2
      * Item 2a
      * Item 2b

### Ordered

    1. Item 1
    1. Item 2
    1. Item 3
       1. Item 3a
       1. Item 3b
```

### Unordered

- Item 1
- Item 2
  - Item 2a
  - Item 2b

### Ordered

1. Item 1
2. Item 2
3. Item 3
   1. Item 3a
   2. Item 3b

Figure 3: **

# Images

```
![GitHub Logo](figs/julia
Format: ![Alt Text](figs/
```


Figure 4: GitHub Logo

# Links

- http://github.com - automatic!
- `[GitHub](http://github.com)`

GitHub

# Blockquotes

```
As Kanye West said:

> We're living the future so
> the present is our past.
```

As Kanye West said:

> We're living the future so the present is our past.

Figure 5: Blockquotes

# Inline code

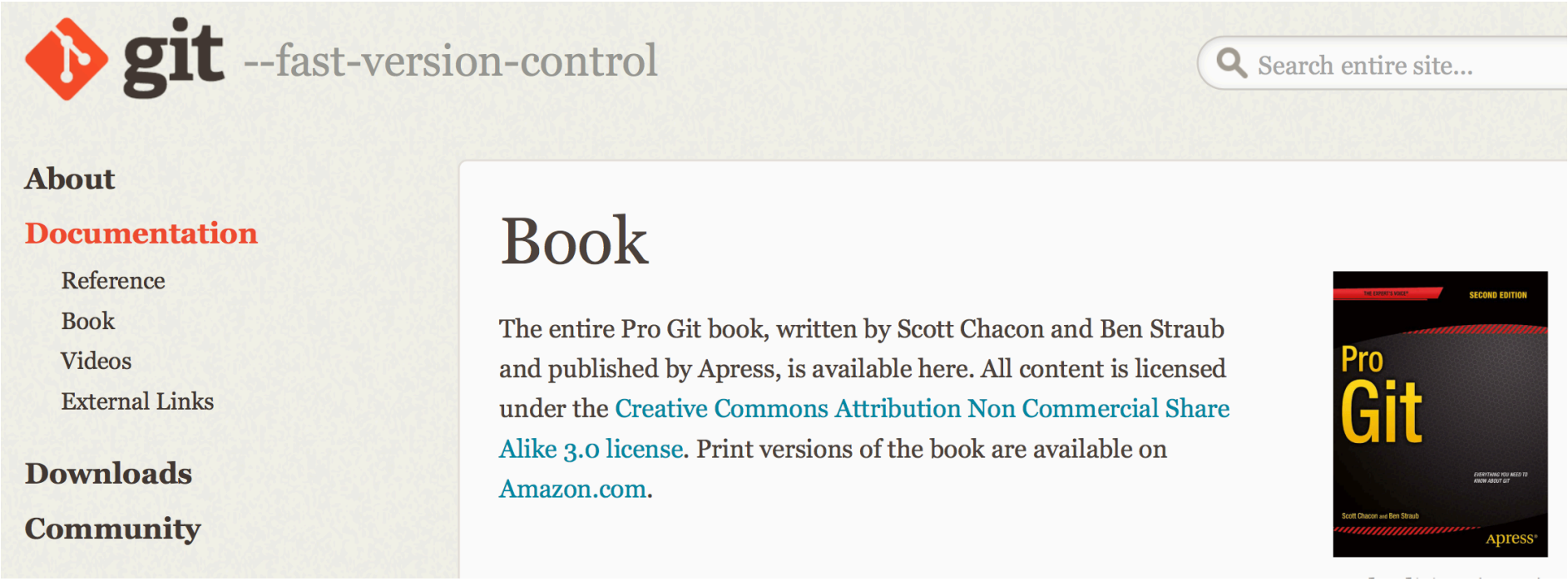
```
I think you should use an
`<addr>` element here instead.
```

I think you should use an `<addr>` element here instead.

Figure 6: **

# VCS (Version Control System): Git

# Pro Git book




The entire Pro Git book, written by Scott Chacon and Ben Straub and published by Apress, is available here.

All content is licensed under the Creative Commons Attribution Non Commercial Share Alike 3.0 license.

Print versions of the book are available on Amazon.com.

# A Short History of Git

Tn 2005 the Linux development community (and in particular Linus Torvalds, the creator of Linux) to develop their own VCS (Version Control System) tool.

Some of the goals of the new system were:

- Speed
- Simple design
- Strong support for non-linear development (thousands of parallel branches)
- Fully distributed
- Able to handle large projects like the Linux kernel efficiently (speed and data size)

# Getting Started - Git Basics

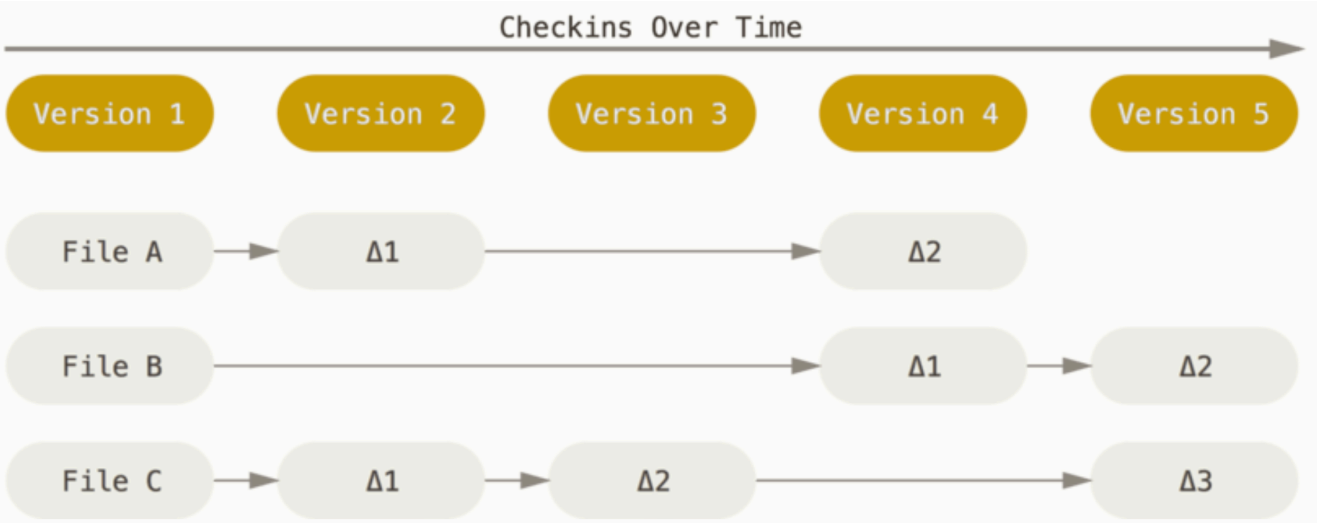



Figure 4. Storing data as changes to a base version of each file.

# Getting Started - Git Basics

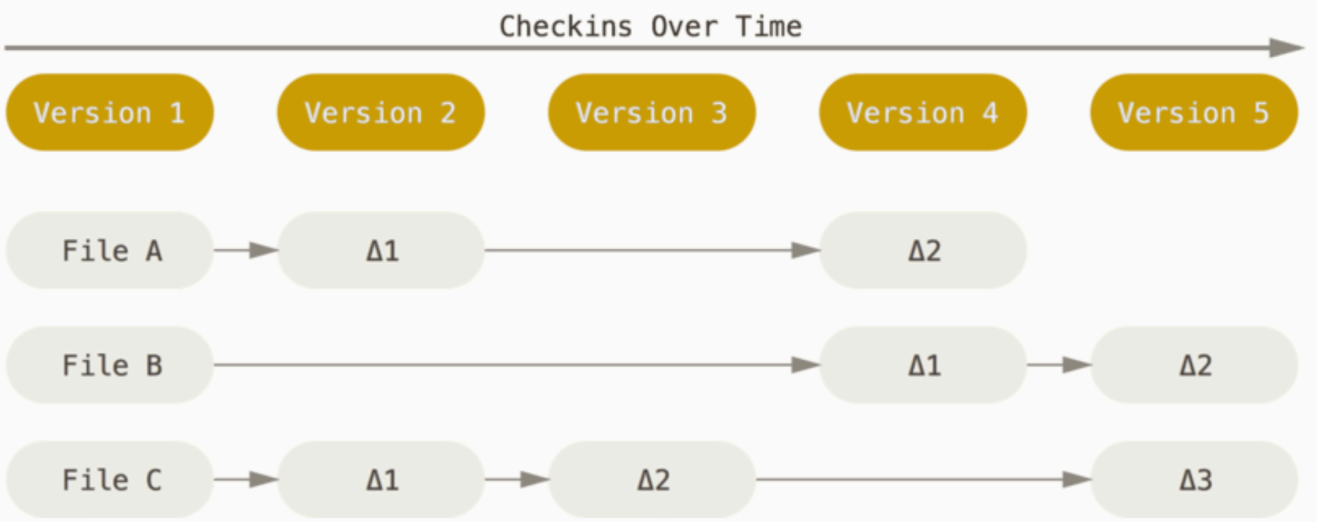



Figure 4. Storing data as changes to a base version of each file.

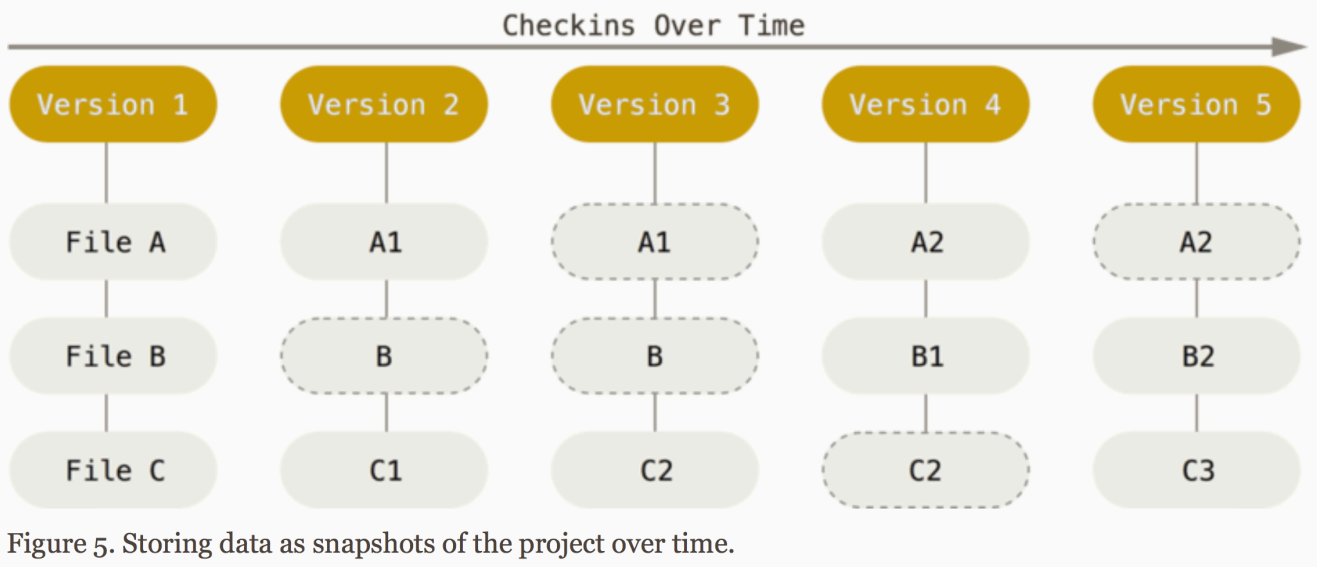



Figure 5. Storing data as snapshots of the project over time.

Git thinks of its data more like a series of snapshots of a miniature filesystem. With Git, every time you commit, you save the state of your project

This makes Git more like a mini filesystem with some incredibly powerful tools built on top of it, rather than simply a VCS

# Git Objects

The core of Git is a simple key-value data store (object database)
Therefore you can insert any kind of content into a Git repository, for which Git will hand you back a unique key you can use later to retrieve that content.
Git would take the content you handed to it and merely return the unique key that would be used to store it as a {blob} in your Git database.
Git normally creates a {tree} by taking the state of your staging area or index and writing a series of tree objects from it.

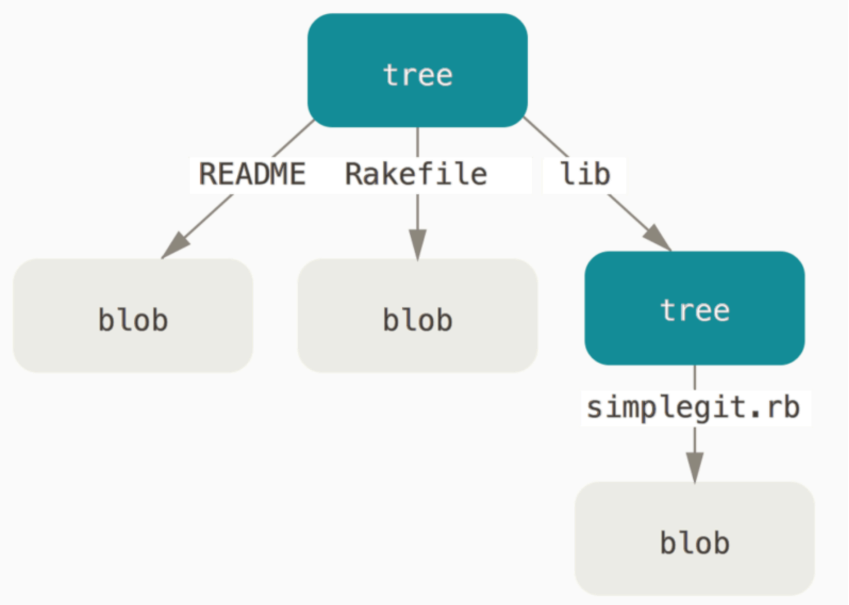



Figure 7: Simple version of the Git data model

# Getting Started - Installing Git

# First-Time Git Setup

### Your Identity

```
$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com
```

### Your Editor

```
$ git config --global core.editor emacs
```

### Checking Your Settings

```
$ git config --list
user.name=John Doe
user.email=johndoe@example.com
color.status=auto
color.branch=auto
color.interactive=auto
color.diff=auto
...
```

# Git basic workflow

1. You modify files in your working tree.
2. You selectively stage just the changes you want to be part of your next commit, which adds only those changes to the staging area.
3. You do a commit, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory.

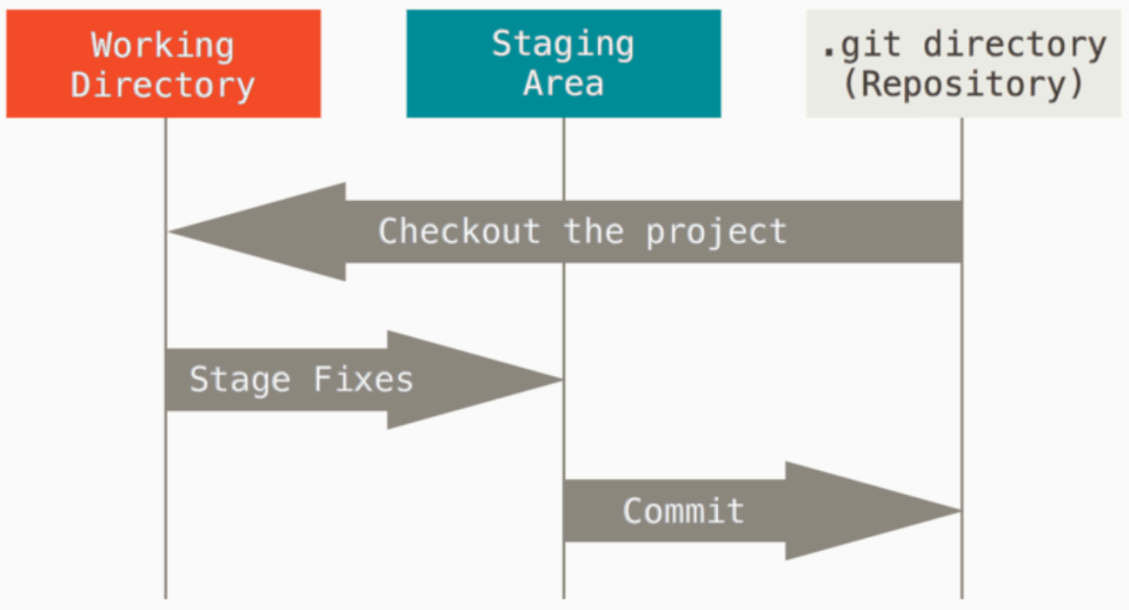



Figure 8: Git data structures

# Git basic workflow

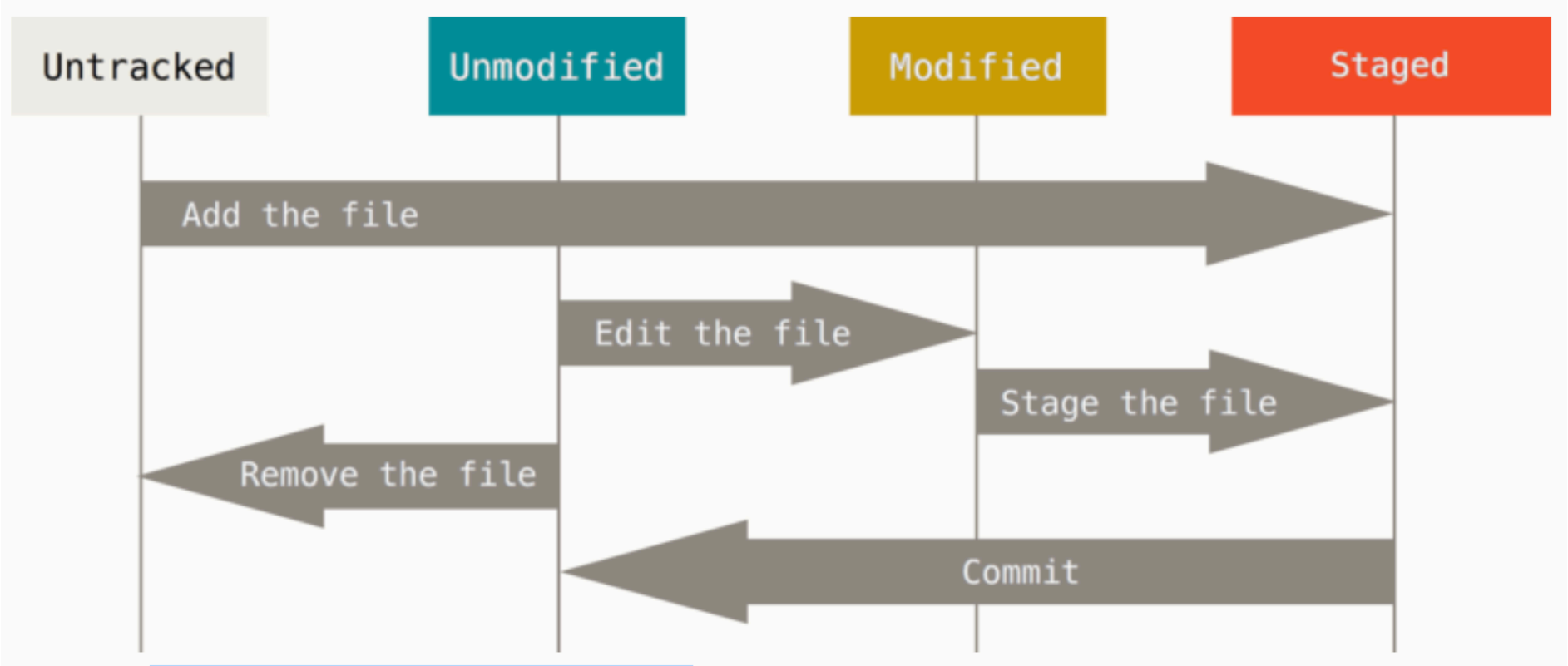



Figure 9: The lifecycle of the status of your files

# Interactive creation of a git repo

```
✔ ~/Documents/DID/DIDATTICA_2019/CPD/LEZIONI-2018/01-julia1-git1/local-repo
16:35 $ ls
✔ ~/Documents/DID/DIDATTICA_2019/CPD/LEZIONI-2018/01-julia1-git1/local-repo
16:35 $ ls -al
total 0
drwxr-xr-x  2 paoluzzi  staff   64 Sep 24 16:35 .
drwxr-xr-x  9 paoluzzi  staff  288 Sep 24 16:35 ..
✔ ~/Documents/DID/DIDATTICA_2019/CPD/LEZIONI-2018/01-julia1-git1/local-repo
16:35 $ git init
Initialized empty Git repository in /Users/paoluzzi/Documents/DID/DIDATTICA_2019/CPD/LEZIONI-2018/01
-julia1-git1/local-repo/.git/
✔ ~/Documents/DID/DIDATTICA_2019/CPD/LEZIONI-2018/01-julia1-git1/local-repo [master L|✔]
16:36 $ ls -al
total 0
drwxr-xr-x  3 paoluzzi  staff   96 Sep 24 16:36 .
drwxr-xr-x  9 paoluzzi  staff  288 Sep 24 16:36 ..
drwxr-xr-x  9 paoluzzi  staff  288 Sep 24 16:36 .git
✔ ~/Documents/DID/DIDATTICA_2019/CPD/LEZIONI-2018/01-julia1-git1/local-repo [master L|✔]
16:36 $ git --version
git version 2.11.0
✔ ~/Documents/DID/DIDATTICA_2019/CPD/LEZIONI-2018/01-julia1-git1/local-repo [master L|✔]
16:45 $ mkdir myproject
✔ ~/Documents/DID/DIDATTICA_2019/CPD/LEZIONI-2018/01-julia1-git1/local-repo [master L|…2]
16:57 $ cd myproject
✔ ~/Documents/DID/DIDATTICA_2019/CPD/LEZIONI-2018/01-julia1-git1/local-repo/myproject [master L|…2]
16:57 $ mkdir src
```

Figure 10: `git init`

# Interactive creation of a git repo

```
✔ ~/Documents/DID/DIDATTICA_2019/CPD/LEZIONI-2018/01-julia1-git1/local-repo/myproject [master L|…2]
16:57 $ cat myfile.txt
cat: myfile.txt: No such file or directory
✘-1 ~/Documents/DID/DIDATTICA_2019/CPD/LEZIONI-2018/01-julia1-git1/local-repo/myproject [master L|…2
]
17:00 $ touch myfile.txt
✔ ~/Documents/DID/DIDATTICA_2019/CPD/LEZIONI-2018/01-julia1-git1/local-repo/myproject [master L|…3]
17:01 $ ls -al
total 0
drwxr-xr-x  4 paoluzzi  staff  128 Sep 24 17:01 .
drwxr-xr-x  6 paoluzzi  staff  192 Sep 24 17:01 ..
-rw-r--r--  1 paoluzzi  staff    0 Sep 24 17:01 myfile.txt
drwxr-xr-x  2 paoluzzi  staff   64 Sep 24 16:57 src
✔ ~/Documents/DID/DIDATTICA_2019/CPD/LEZIONI-2018/01-julia1-git1/local-repo/myproject [master L|…3]
17:02 $ mv myfile.txt src/
✔ ~/Documents/DID/DIDATTICA_2019/CPD/LEZIONI-2018/01-julia1-git1/local-repo/myproject [master L|…3]
17:03 $ nano src/myfile.txt
✔ ~/Documents/DID/DIDATTICA_2019/CPD/LEZIONI-2018/01-julia1-git1/local-repo/myproject [master L|…3]
17:04 $ cat src/myfile.txt
Questo e` il mio primo file sotto git
```

Figure 11: populate the sorking directory

# Interactive creation of a git repo

```
✔ ~/Documents/DID/DIDATTICA_2019/CPD/LEZIONI-2018/01-julia1-git1/local-repo [master L|…3]
17:06 $ ls -al
total 16
drwxr-xr-x  6 paoluzzi  staff   192 Sep 24 17:04 .
drwxr-xr-x  9 paoluzzi  staff   288 Sep 24 17:04 ..
-rw-r--r--@ 1 paoluzzi  staff  6148 Sep 24 16:56 .DS_Store
drwxr-xr-x  9 paoluzzi  staff   288 Sep 24 16:36 .git
drwxr-xr-x  4 paoluzzi  staff   128 Sep 24 16:56 latex-project
drwxr-xr-x  3 paoluzzi  staff    96 Sep 24 17:04 myproject
✔ ~/Documents/DID/DIDATTICA_2019/CPD/LEZIONI-2018/01-julia1-git1/local-repo [master L|…3]
17:07 $ git add . -A
✔ ~/Documents/DID/DIDATTICA_2019/CPD/LEZIONI-2018/01-julia1-git1/local-repo [master L|•3]
17:07 $ git status
On branch master

Initial commit

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

        new file:   .DS_Store
        new file:   latex-project/.DS_Store
        new file:   myproject/src/myfile.txt
```

Figure 12: git add files on stage

# Interactive creation of a git repo

```
✔ ~/Documents/DID/DIDATTICA_2019/CPD/LEZIONI-2018/01-julia1-git1/local-repo [master L|•3]
17:08 $ git commit -m "Initial commit"
[master (root-commit) 5886b42] Initial commit
 3 files changed, 3 insertions(+)
 create mode 100644 .DS_Store
 create mode 100644 latex-project/.DS_Store
 create mode 100644 myproject/src/myfile.txt
✔ ~/Documents/DID/DIDATTICA_2019/CPD/LEZIONI-2018/01-julia1-git1/local-repo [master L|✔]
17:10 $ git status
On branch master
nothing to commit, working tree clean
```

Figure 13: commit of repository status

# Git Tutorial (by Marino & Spini)

webcrumbs / **git-crumbs**

Watch 1 | Star 8 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Insights

Git crumbs for beginners

**26** commits | **1** branch | **0** releases | **2** contributors | MIT

Branch: **master** | New pull request | Create new file | Upload files | Find file | Clone or download

**federicospini** Update Readme.md — Latest commit bbdf8a6 on Mar 5, 2015

| | | |
|---|---|---|
| git | Update Readme.md | 4 years ago |
| .gitignore | added ignoring files | 7 years ago |
| .npmignore | added ignoring files | 7 years ago |
| LICENSE | license | 7 years ago |
| Readme.md | Update Readme.md | 7 years ago |
| package.json | license | 7 years ago |

# Simple daily git workflow

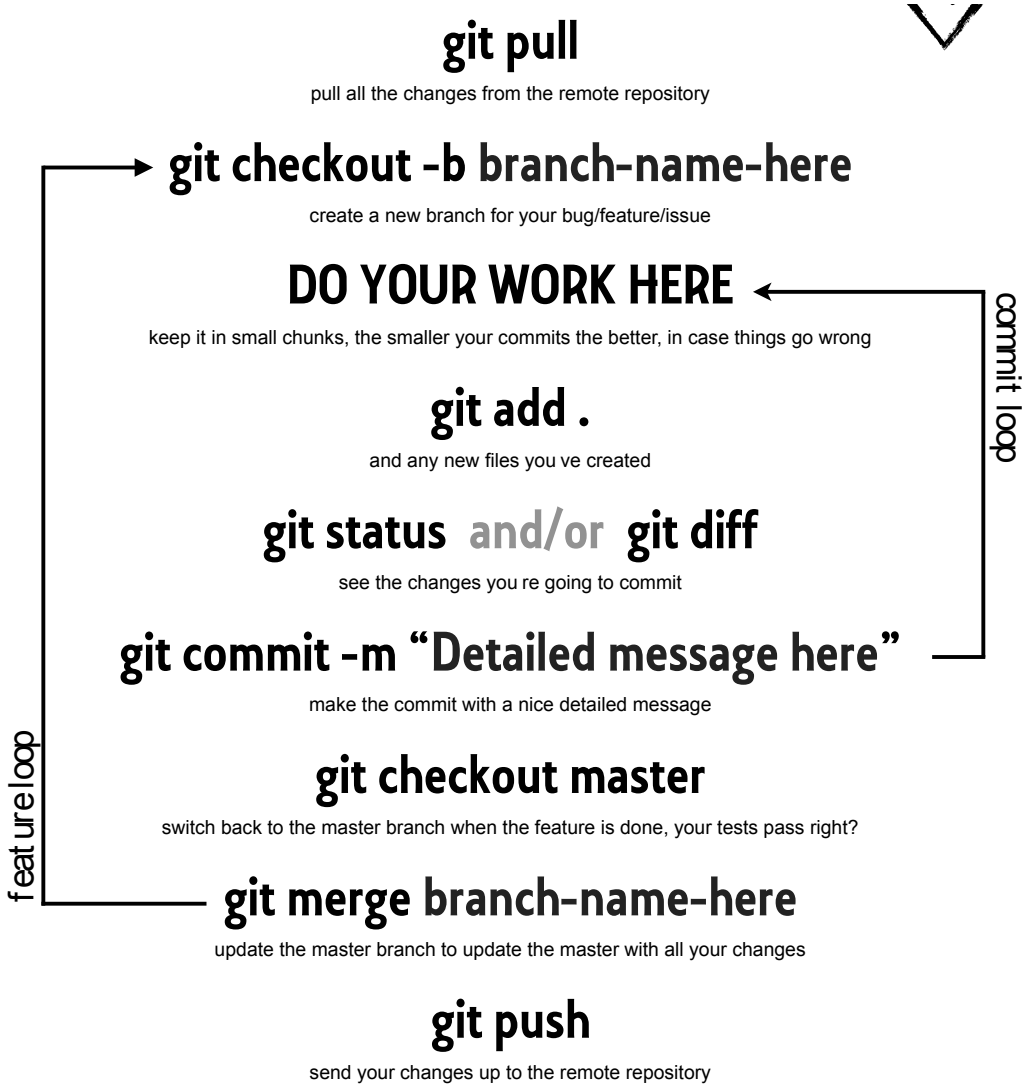



Figure 14:

# References

# References

```
@book{Chacon:2014:PG:2695634,
 author = {Chacon, Scott and Straub, Ben},
 title = {Pro Git},
 year = {2014},
 isbn = {1484200772, 9781484200773},
 edition = {2nd},
 publisher = {Apress},
 address = {Berkely, CA, USA},
}
```