The Abdus Salam
International Centre
for Theoretical Physics

Git for Open Science

Werner Florian

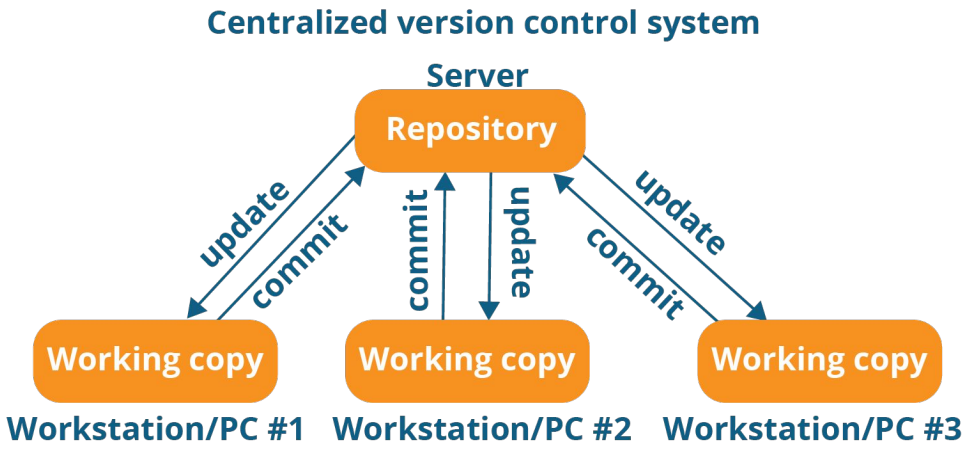
Overview

- Open Science
- What are Version Control Systems?
- What is Git?
- How is the Git Workflow structured?
- How to configure Git in my pc?
- How to use Git to manage my Vivado projects?
- How to troubleshoot my repository?


OPEN SCIENCE
Open Source
Open Access
Open Infrastructures
Open Hardware
Open Educational Resources
Open Evaluation
Open Data
Citizen Science
Open Labs
Open Innovation
Open Notebook
Crowd-Funding

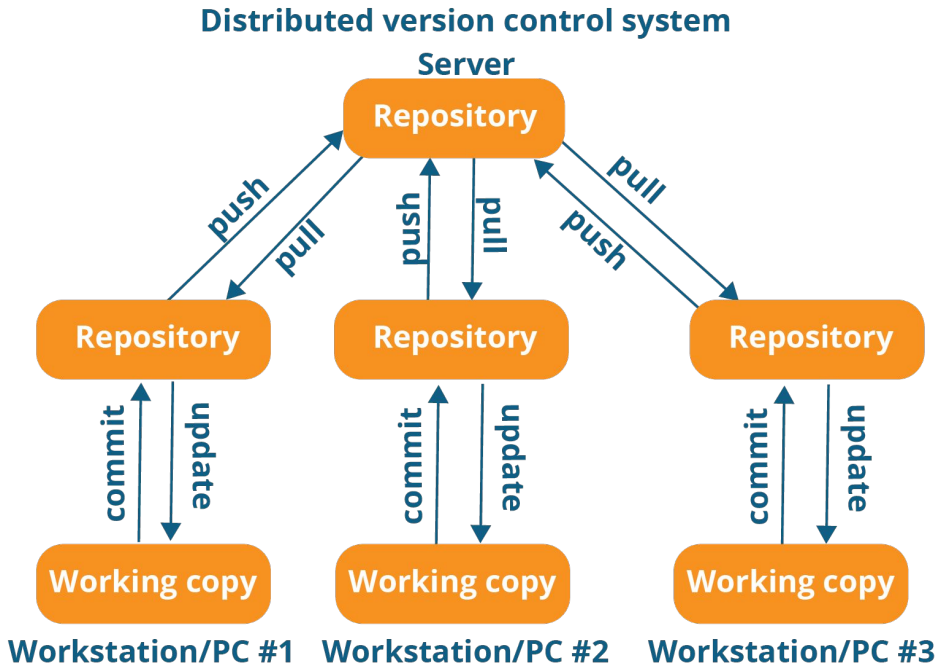
Centralized Version Control Systems

- Only one copy of the files is “official”
- Collaboration is hard
- Always connected to the remote source or local changes without versions
- Example Dropbox, Google docs, etc.



Distributed Version Control Systems

- All users clone a complete copy of the project and pull to update local
- Changes can be tested on local
- Local changes are versioned
- Examples Git, Mercurial, etc.

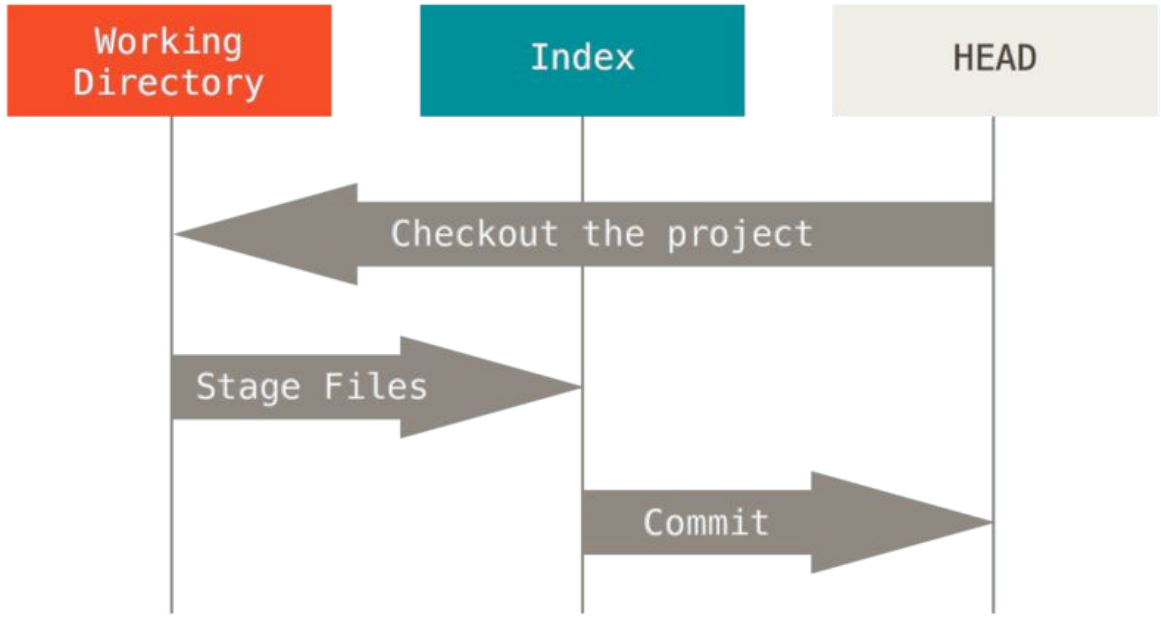


About Git

- Most popular solution
- Free and open source
- Fully Distributed
- Created by Linus Torvald

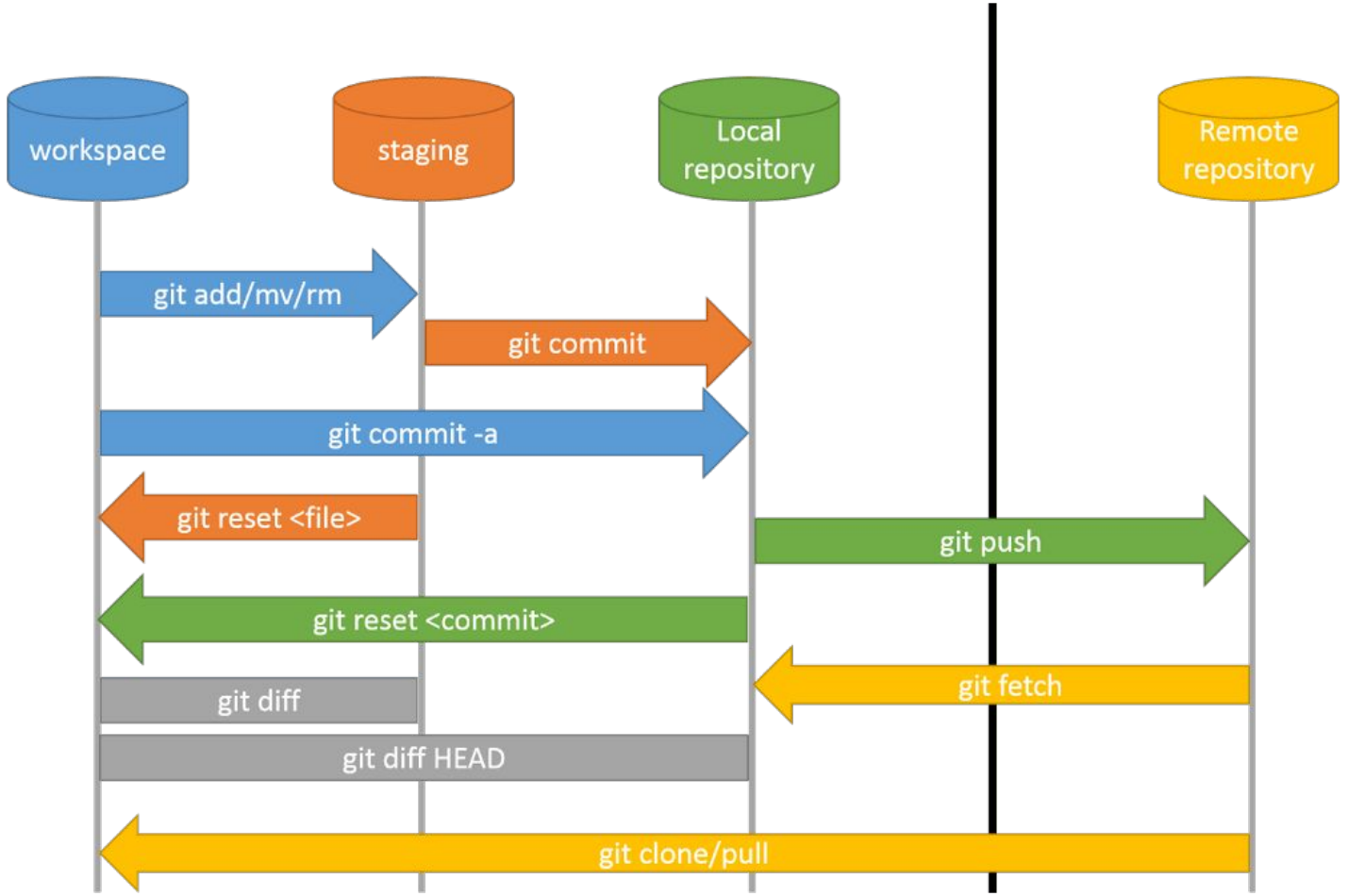
The Three Trees of Git

- HEAD
  - Pointer to the current branch reference (last commit)
- Index / Staging
  - Proposed next commit
- Workspace, working directory, working tree
  - Where files are edited (sandbox)



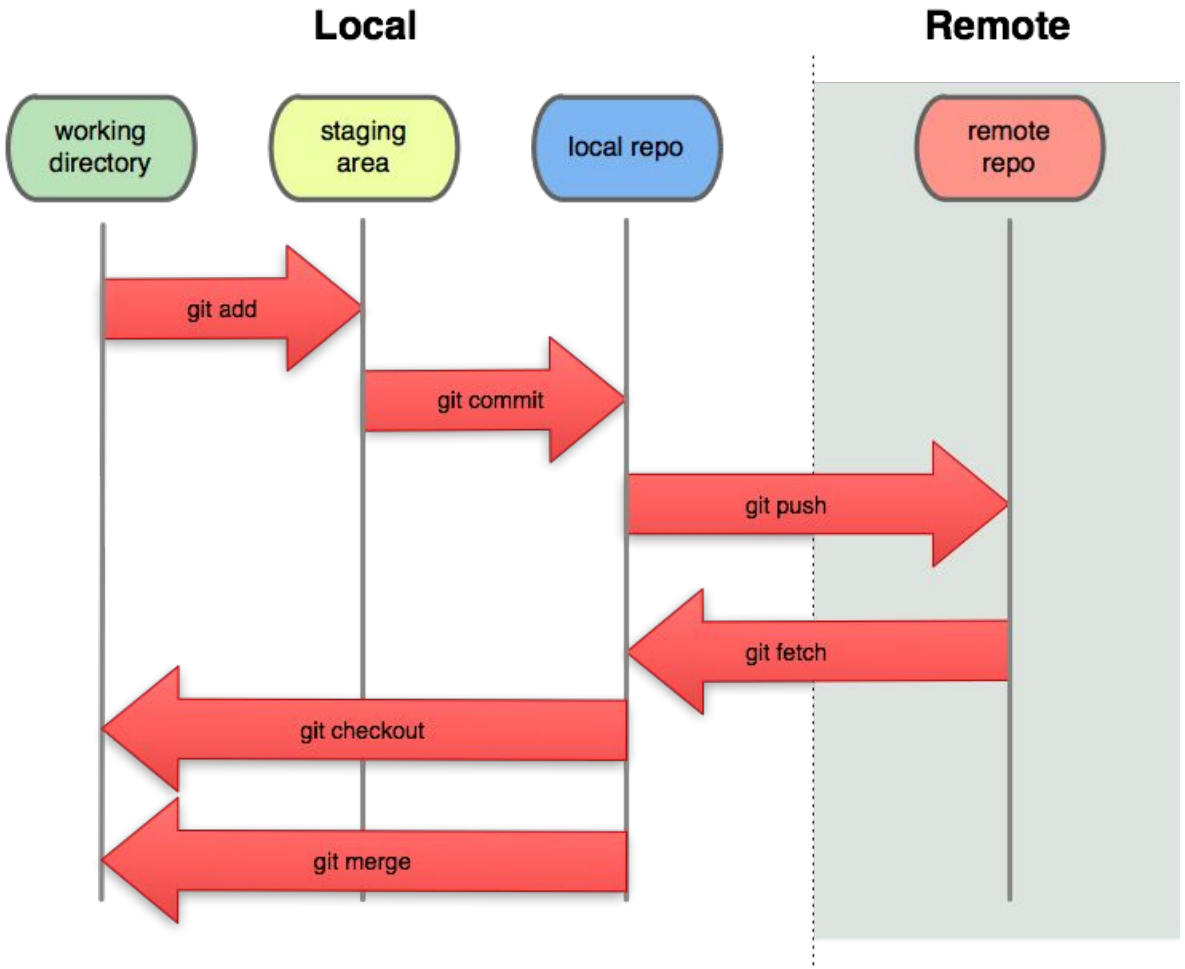
Git Workflow

- Remote:
  - Clone
  - Fetch
  - Pull
  - Push
- Local:
  - Add
  - Commit
  - Checkout
  - Stash
  - Merge



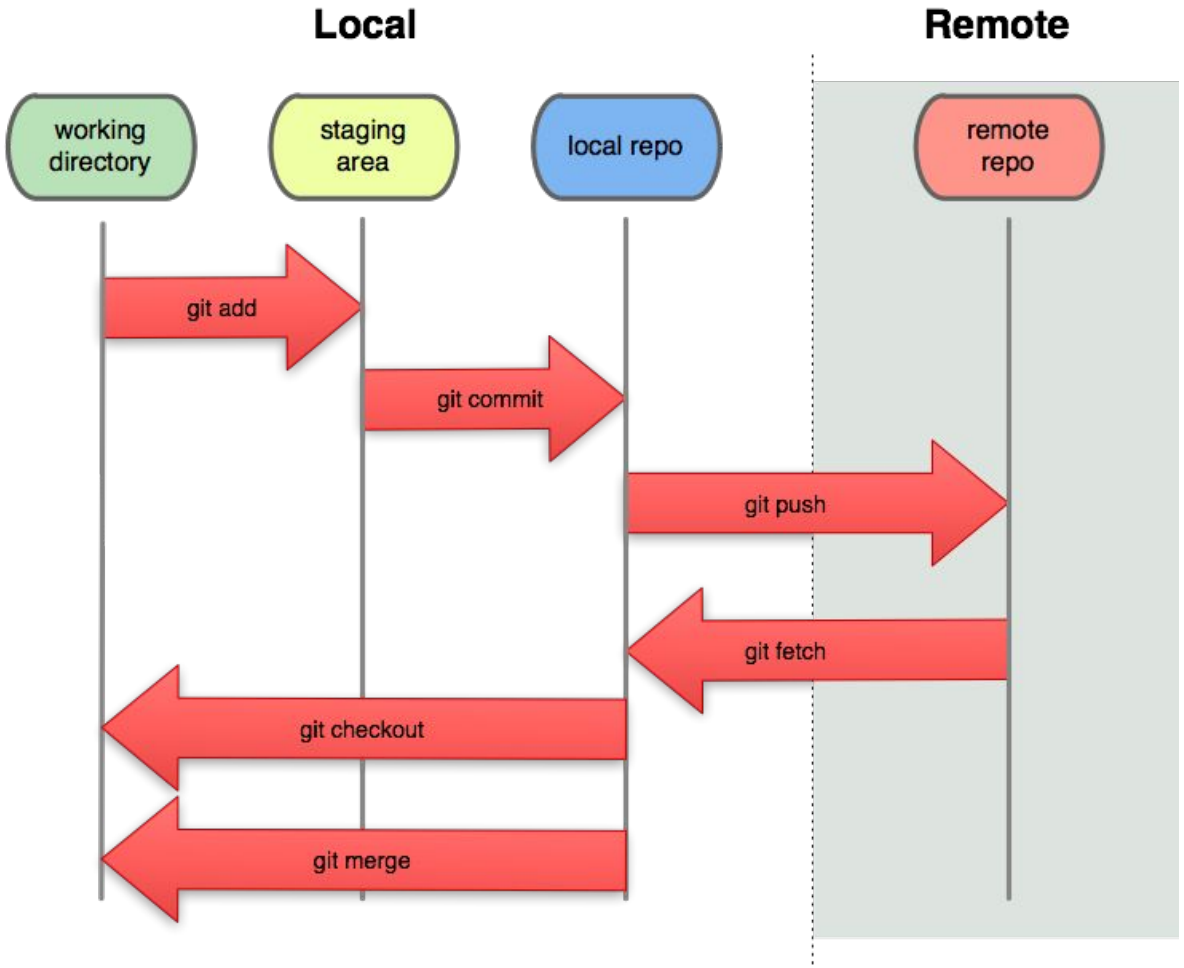
Git Local Workflow

- Init: initialize current directory as a git repo
- Add, rm: interact with single files in working branch
- Commit: takes a snapshot from files and includes the changes in the Git repo
- Branch: create a new branch
- Checkout: change branch
- Merge: join branches or working directories

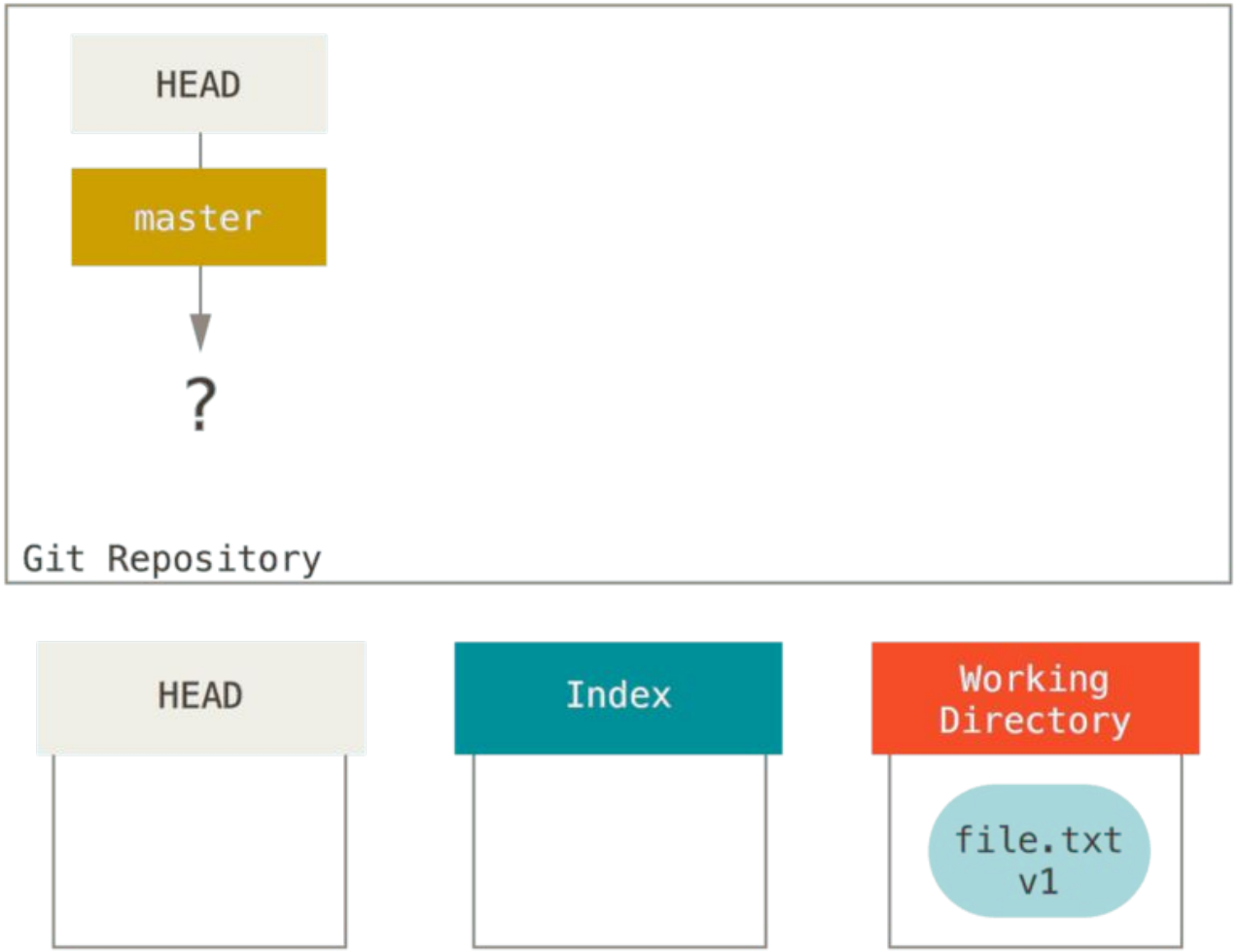


Git Remote Workflow

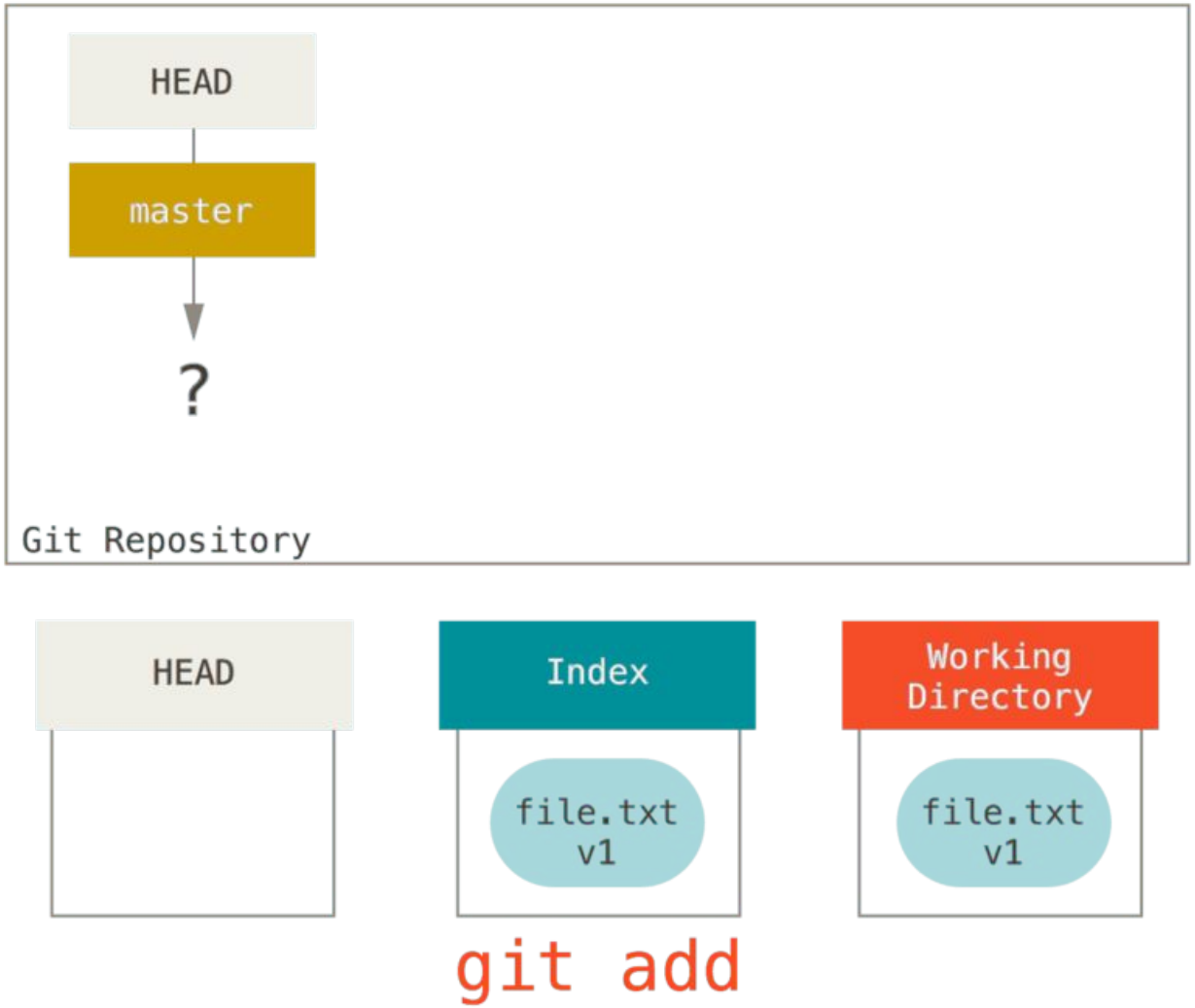
- Clone: copy remote repository to current directory
- Fetch: download all changes from remote repo to local
- Pull: implement local repo incoming changes into current branch
- Push: publish local repository changes to remote



Git Three Trees Commit Story #1



Git Three Trees Commit Story #2

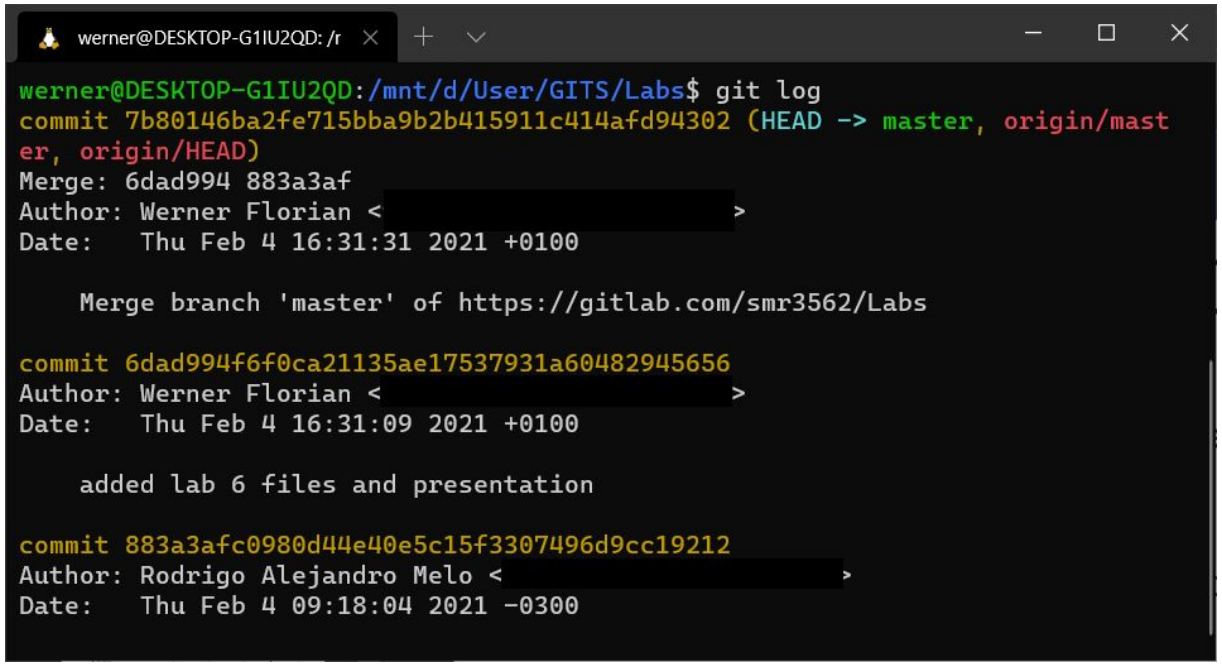


Git Three Trees Commit Story #3

git commit

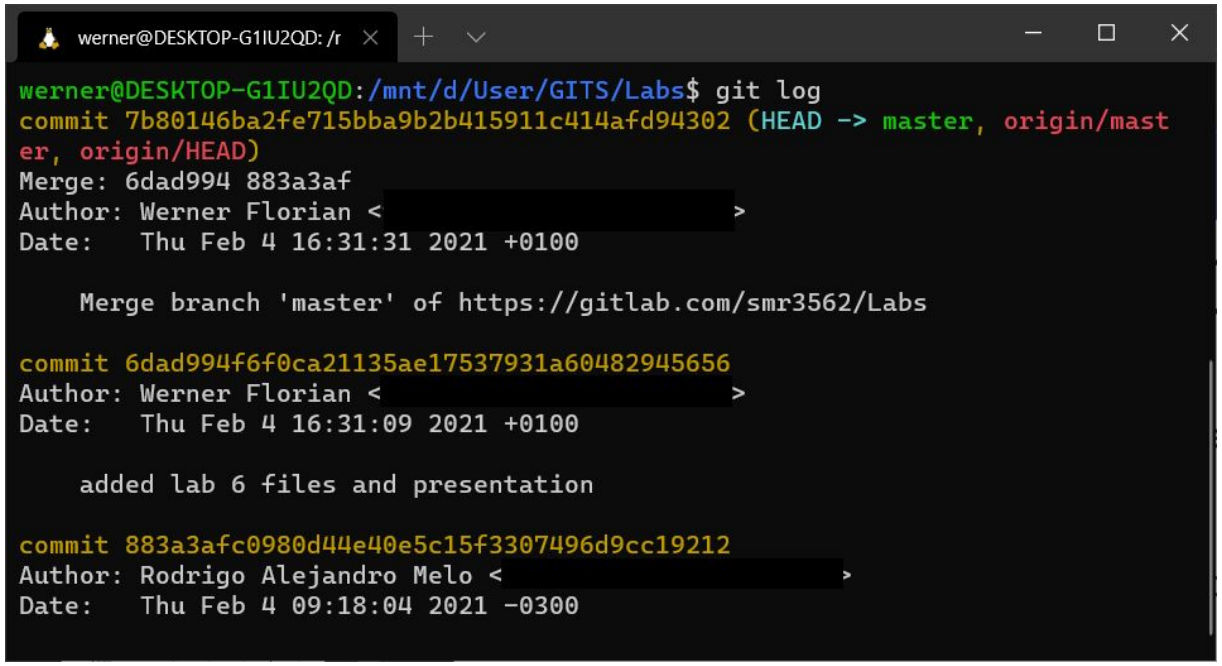
Git Management Commands

- Config: allows loading settings like username and password
- Remote: connects a local repository to a remote repository
- Status: outputs the currents working branch information
- Log: show the chronological commit history for the current branch
- Reflog: show history of all branches



Git Commit Checksum and Metadata

- SHA-1 hash to identify each commit
- Distributed updates
- Name and mail of the author



Git Initial Configuration

- Set the name, email and password to identify the author when committing changes

```
git config --global user.name “Foo Bar”
git config --global user.email foo@bar.com
```

- To check if the changes were applied run

```
git config --list
```

Creating Git Repositories

There are 2 ways you can create a repository:

1. Initialize a local repository and connect to a remote later

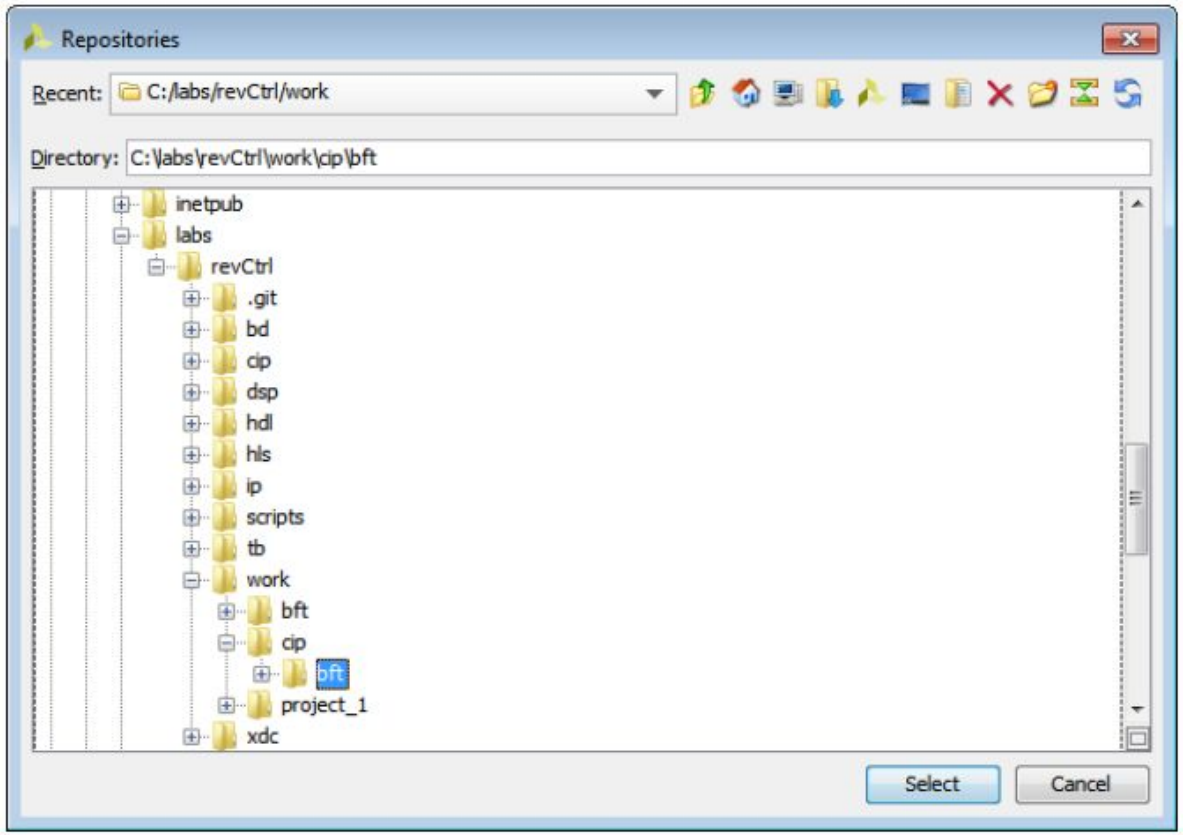
```
git init
git add filename
git commit –m “message”
git remote add name-of-local url-of-remote
```

2. Clone a remote repository

```
git clone url-of-remote
```

Git for Vivado

- Xilinx repository recommended structure
  - Bd for block design
  - hdl: Verilog HDL including the top-level design
  - xdc: Xilinx Design Constraints for the top-level design
  - hls: C++ design
  - dsp: SysGen design
  - scripts: tcl and make files
  - work: vivado workspace

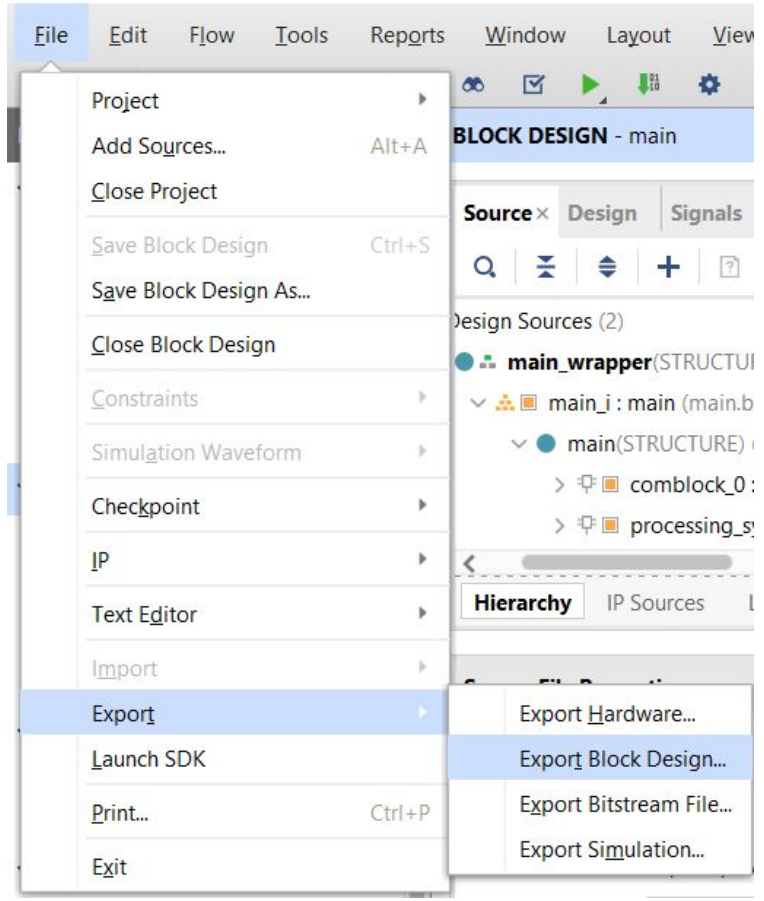


Vivado Project Automation

- Automate whole project creation, requires all files and sources to be inside the repository.

  write_project_tcl output-file

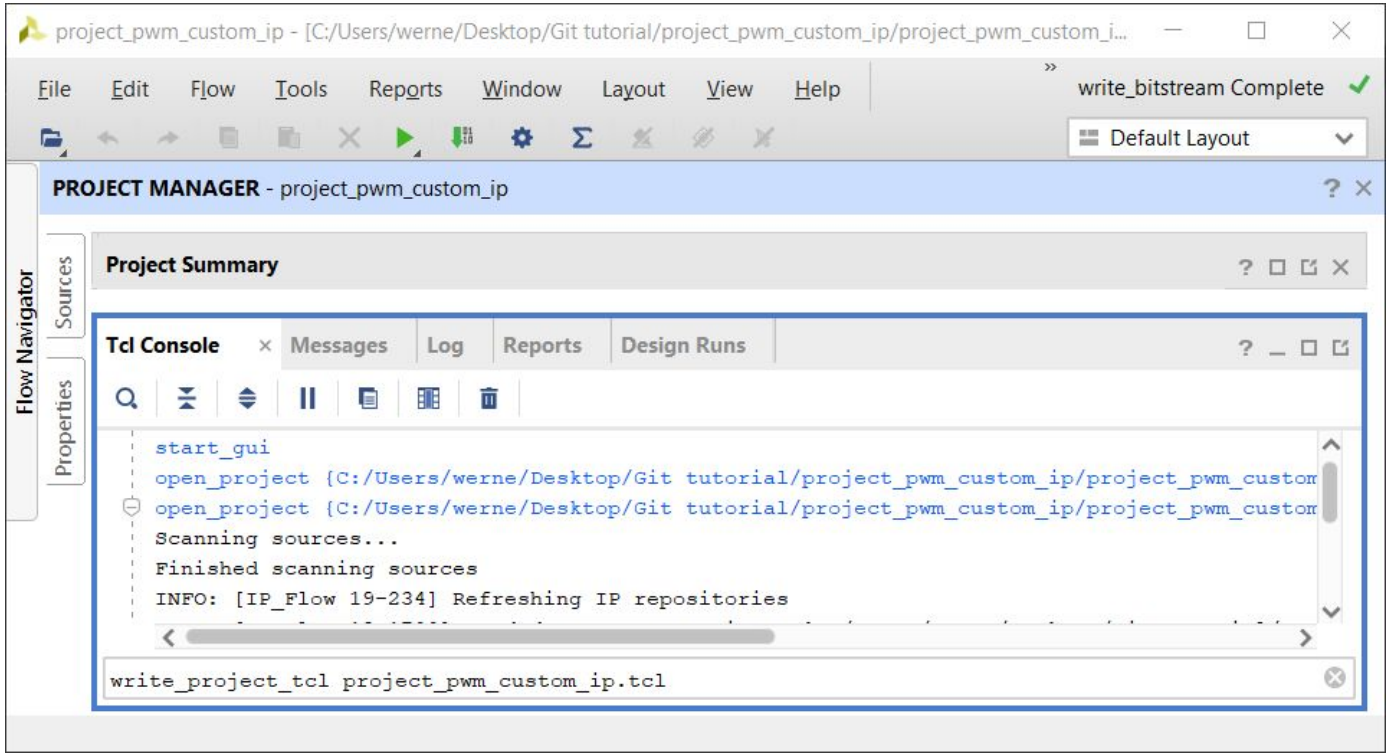
- Automate design block, useful when sources or ips are outside the repository and may change place (user must add manually the sources and ips to the catalog beforehand).
  1. Open project
  2. Open Block Design
  3. File -> Export -> Export Block Design



Project TCL File Generation

This is the best way to share functional projects

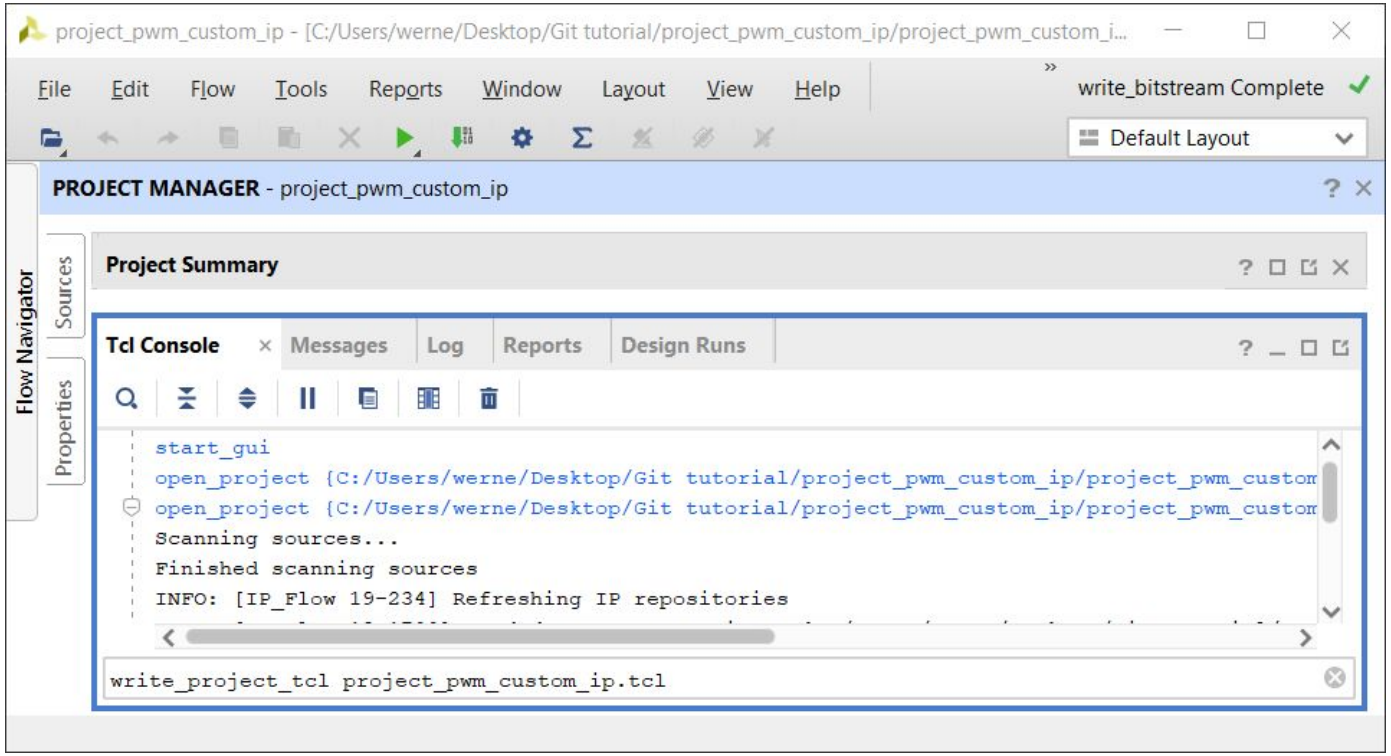
- Self contained
- Automates Vivado workflow
- Bigger repository size
- Manual update of sources/ips by copying and pasting
- Xilinx recommended



Required Sources

The following are required always

- Custom Ip-repo
- Custom VHDL
- Custom C code
- VHDL Wrapper
- Project tcl
- Block design tcl (nice to have)



How to extract VHDL wrapper?

1. Open Vivado generated wrapper in a text editor.
2. Copy the whole content to a new file (ex. top.vhdl).
3. Add the file to the VHDL sources in the repo.
4. Add the new wrapper into the Vivado project.
5. Delete the automatically generated wrapper.
6. Generate project tcl

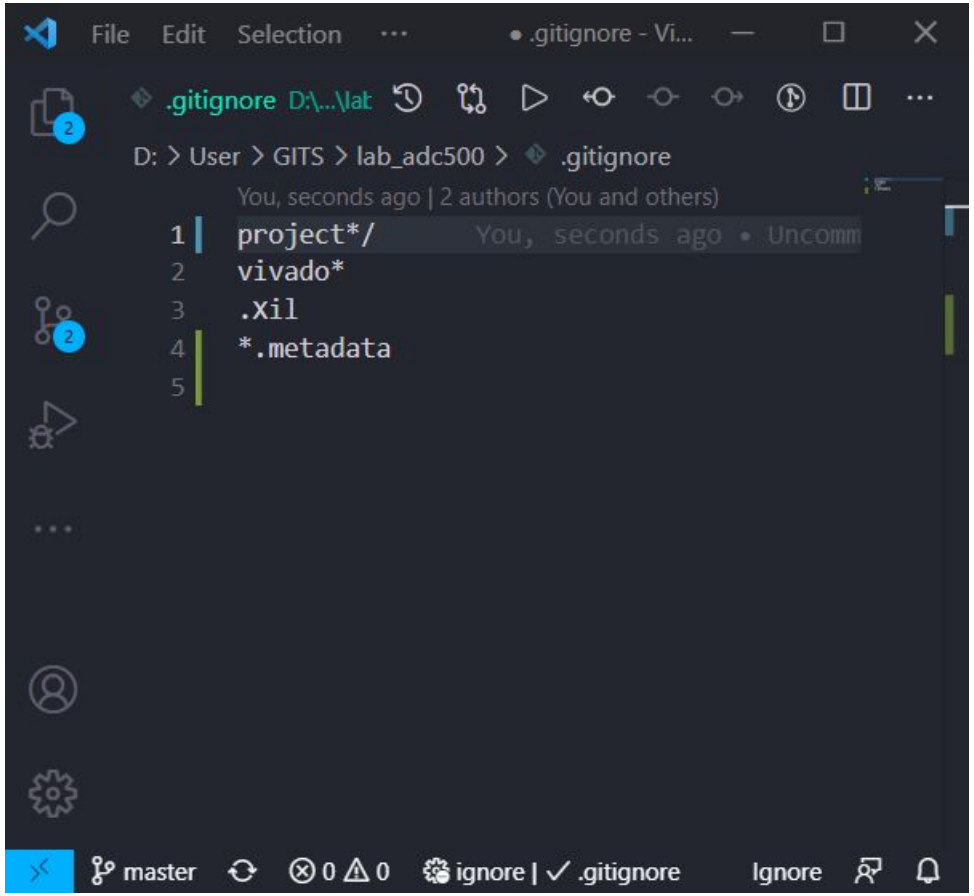
Project TCL File Structure

- set origin_dir
- create_project
- create_bd_design
- current_run

.GITIGNORE for Vivado

There are several files we do not want to share.

- Useless metadata
- Logs
- Object files
- Side products
- Workspace



Troubleshooting

Fixing committed changes

- git reset
- git revert
- git rebase

Fixing staged changes

- git restore
  - git stash
  - git clean

Compare file versions

- git diff

Gitlab, Github. Bitbucket which one?

Pros of Bitbucket

▲ 903	Free private repos
▲ 397	Simple setup
▲ 345	Nice ui and tools

Pros of GitHub

▲ 1.8K	Open source friendly
▲ 1.5K	Easy source control
▲ 1.2K	Nice UI

Pros of GitLab

▲ 488	Self hosted
▲ 415	Free
▲ 331	Has community edition

Cons of Bitbucket

▲ 19	Not much community activity
▲ 17	Difficult to review prs because of confusing ui
▲ 14	Quite buggy
▲ 10	Managed by enterprise Java company
▲ 8	CI tool is not free of charge

Cons of GitHub

▲ 40	Owned by micrcosoft
▲ 36	Expensive for lone developers that want private repos
▲ 15	Relatively slow product/feature release cadence
▲ 9	API scoping could be better
▲ 8	Only 3 collaborators for private repos

Cons of GitLab

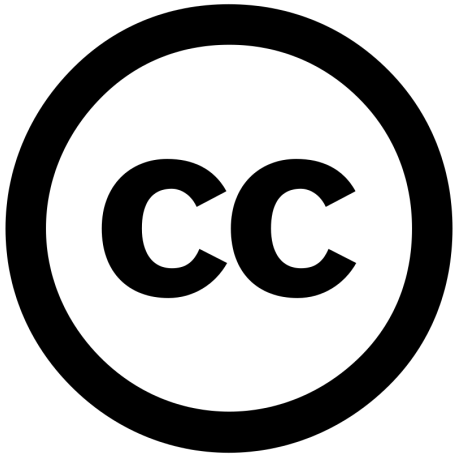
▲ 25	Slow ui performance
▲ 6	Introduce breaking bugs every release
▲ 5	Insecure (no published IP list for whitelisting)
▲ 0	Review Apps feature
▲ 0	Built-in Docker Registry

Licensing

Different for creative work and hardware!!

Creative Work:

- Firmware
- Software



Hardware:

- CAD-design, schematics
- Digital design files for physical objects

Licensing for Open Source Software

Public domain: anyone is free to use and modify

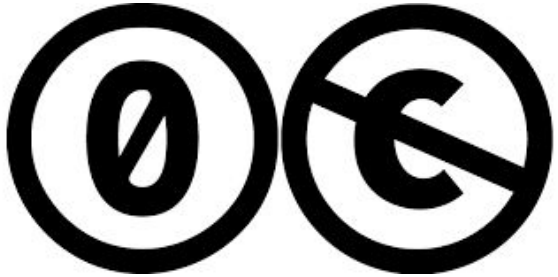
- CC0: creative commons zero

Permissive: reuse, modification, and distribution with some restrictions

- Apache
- BSD
- MIT License

Copyleft: enforce openness

- Mozilla GPL
- GNU GPLv3

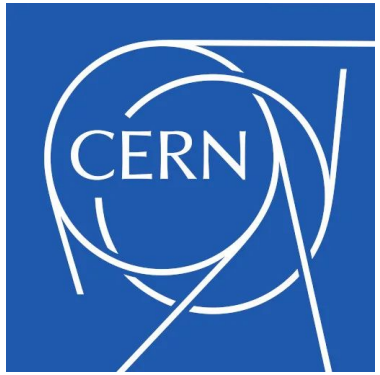


What if I don’t want a license?

- Work is under exclusive copyright by default.
- Nobody else can copy, distribute, or modify your work without being at risk of takedowns or litigation.
- You don’t have to do anything to not offer a license.

Licensing for Open Source Hardware

- Copyleft licenses (GPL) and the Creative Commons Attribution-ShareAlike license.
- CERN Open Hardware License (OHL)
- TAPR Open Hardware License (OHL).
- Permissive licenses
  - FreeBSD
  - MIT license
  - Creative Commons Attribution
  - Solderpad Hardware



Thanks for your attention

Werner Florian
Joint ICTP-UniTS PhD student
wflorian@ictp.it