Technische
Hochschule
Augsburg

VLSI Design

Introduction to Version Management

J Färber

SS2026

Contents

Introduction to Version Management

Learning Objectives

After completing this unit, you will be able to:

- Understand why using Version Management
- Create, clone, and synchronise a Git repository
- Commit changes and mark milestones with tags

Why Version Management?

Hardware design is iterative. A circuit that simulates correctly today may break after tomorrow's extension. Without version management you risk:

- **Losing working intermediate states** - your last known-good netlist is gone after one bad edit
- **Uncontrolled file duplicates** - `de1_mux2to1_v3_final_FINAL2.vhd` is not a version strategy
- **No traceability when debugging** - you cannot answer *"what exactly changed between the run that worked and the one that doesn't?"*
- **No safe experimentation** - without branches, every change is a gamble on your only copy

With Git you get:

- **Snapshots at any milestone** - tag a commit when simulation passes, another when synthesis meets timing
- **Instant rollback** - one command to restore any previous working state
- **Diff at any level** - compare two versions of a VHDL entity line by line
- **Parallel experiments** - try a different architecture in a branch without touching your working design

In short: Git is your lab notebook, your undo history, and your backup - all in one.

References

- Gitlab-Server at our University of Applied Sciences Augsburg- UAS
  - GitHub-Server
  - GitLab-Server
- [ChaStr] - Chacon, S, Straub, B.: Pro Git Book by Scott Chacon and Ben Straub

Distributed Version Control System

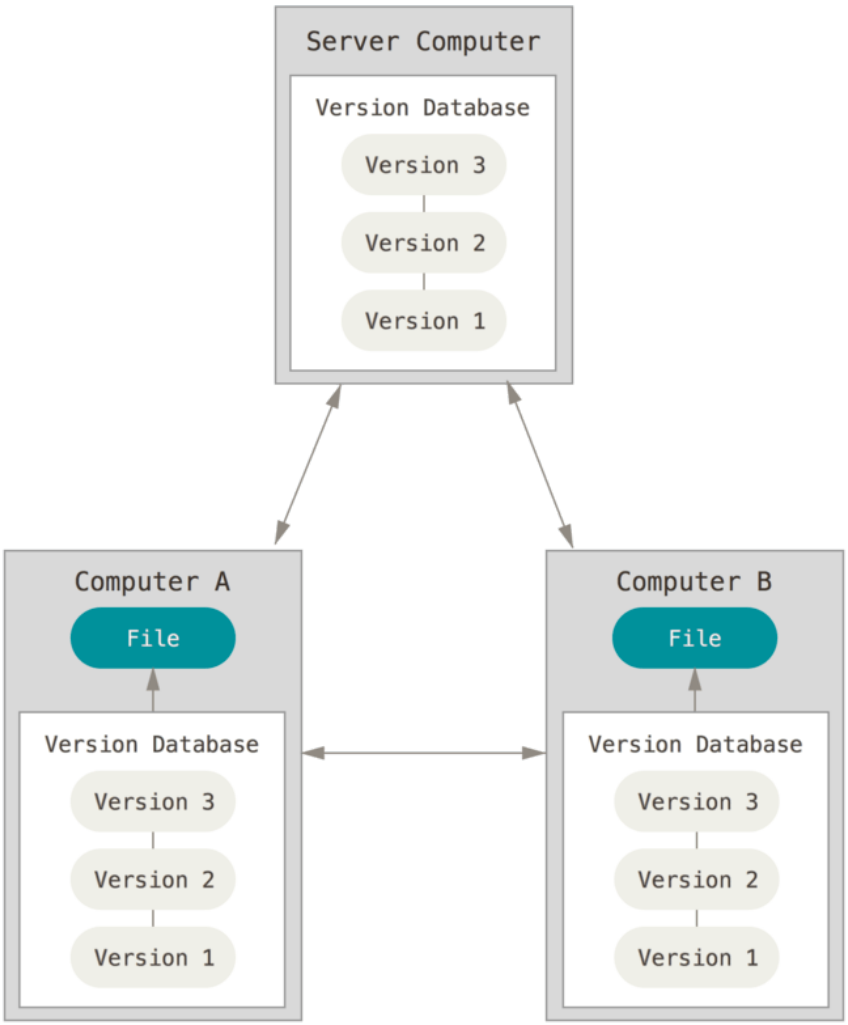

Figure 1: Distributed Version Control

Storage Areas in Git

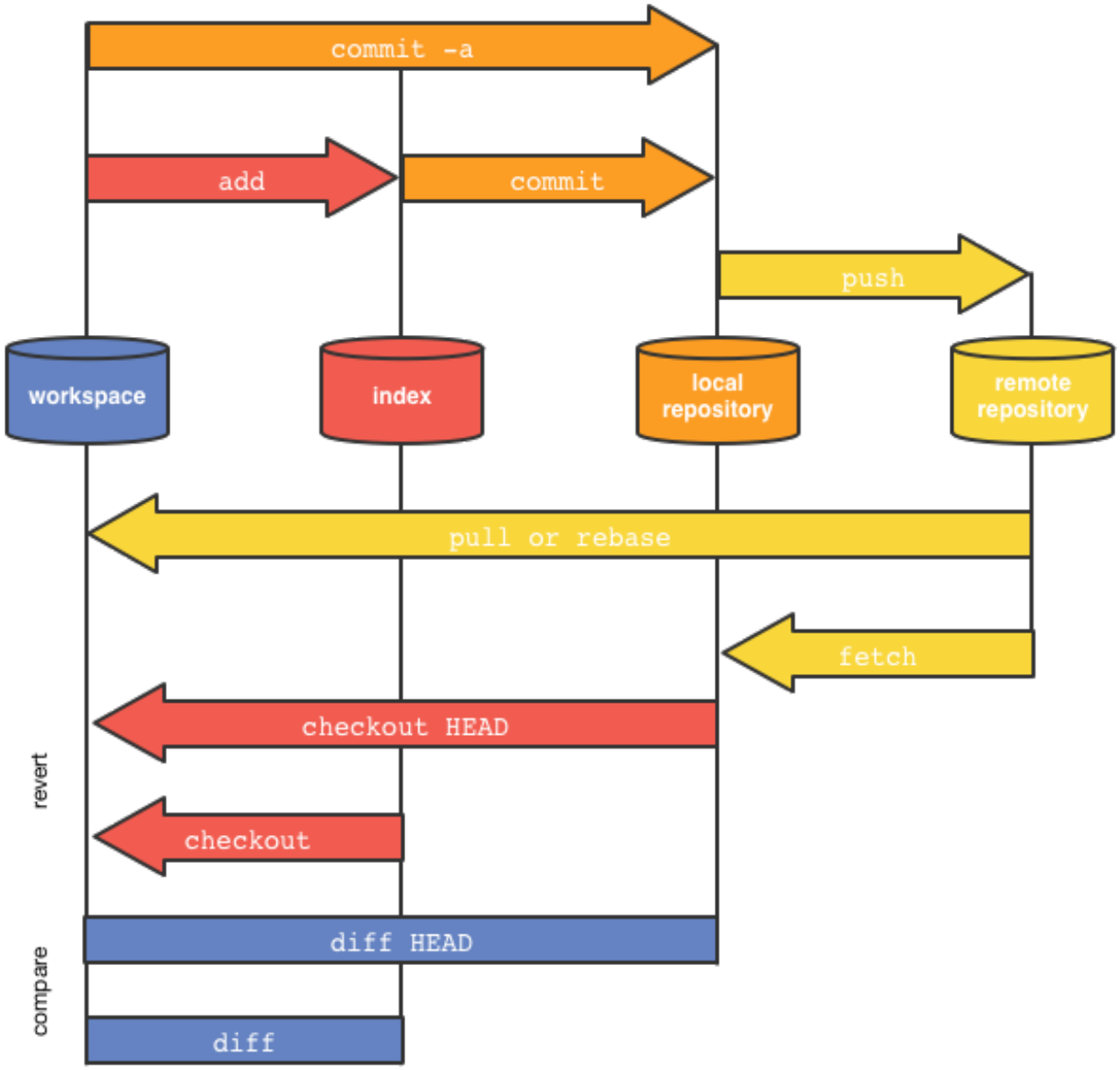

Figure 2: Data flow from and to remote and local repository

Area	Description
Working Tree / workspace	The files you are currently editing
Staging Area / index	Changes staged for the next commit (`git add`)
Local Repository	Complete version history stored locally (`git commit`)
Remote Repository	Central server - GitLab at HS Augsburg (`git push / pull`)

Git Basic Steps

Referring to [ChaStr] - chap. Git Basics

Git Global Setup

- The git global setup has to be created only once in your local file system, e.g.

```
git config --global user.name "Johann Faerber"
git config --global user.email "johann.faerber@hs-augsburg.de"
git config --global push.default simple
git config --global http.postBuffer 524288000
```

Verify the configuration:

```
cat ~/.gitconfig
```

Create a Remote Repository

- In the WebInterface of Gitlab-Server-UAS select the '+' Icon in the top right corner to create a blank repository with your lastname and firstname in the format vlsi-fpga-lab-2026_lastname_firstname, e.g.

```
vlsi-fpga-lab-2026_lastname_firstname
```

Unselect 'Initialize repository with a README.md

- The server responds with a message, e.g.

```
Project 'vlsi-fpga-lab-2026_lastname_firstname' was successfully created.
```

- Additionally, further command line instructions are suggested.

Clone Remote Repository

- Clone your remote repository from the server into the documents directory of your local machine, e.g.

```
cd ~/Documents
git clone https://gitlab.elektrotechnik.hs-augsburg.de/haf/vlsi-fpga-lab-
↪  2026_lastname_firstname.git
```

Initialise your Repository with Lab Exercises

```
# 1. Clone your personal repository
git clone
 ↪  https://gitlab.elektrotechnik.hs-augsburg.de/haf/vlsi-fpga-lab-2026_lastname_firstname.git
cd vlsi-fpga-lab-2026_lastname_firstname

# 2. One-time global Git configuration
git config --global user.name  "Firstname Lastname"
git config --global user.email "firstname.lastname@hs-augsburg.de"

# 3. Clone master repo, copy script, remove master repo
git clone --depth=1 \
  "https://gitlab.elektrotechnik.hs-augsburg.de/haf/2026-vlsi-fpga-flow-lab.git" \
  /tmp/vlsi-master-init
cp /tmp/vlsi-master-init/update_repo.sh .
rm -rf /tmp/vlsi-master-init

# 4. Run the script
sudo apt install rsync
chmod +x update_repo.sh
./update_repo.sh lab#0
```

> **Your existing work is never overwritten.** The script only adds files that do not yet exist in your repository.

Checking the Status

- With the working directory of the repoistory, get a short status

  ```
  git status -s
  ```

- Full comprehensive status

  ```
  git status
  ```

Add and Commit Changes to a Local Repository

- Perform a local git configuration

  ```
  cd vlsi-fpga-lab-2026_lastname_firstname
  git config --local core.filemode false
  ```

- Add and commit your changes to your local repository, e.g.

  ```
  git add *
  git add .gitignore
  git commit -m "Added design project"
  ```

Synchronise Remote Repository

- Push your local repository to your remote repository on our server, e.g.

```
git push -u origin main
```

Remove a File from Tracking

- Remove a file from working directory and from tracking

```
git rm README.md
```

- Commit changes

```
git commit -m "Removed README.md"
```

Add a Tag

Tags mark important milestones in the project history. **All lab submissions are handed in as an annotated Git tag.**

- Tag it with **lab#1_mux2to1**, e.g.

```
git tag -a lab#1_mux2to1 -m 'tag lab#1_mux2to1 created'
git push --tags
```

> **Convention:** Every lab exercise defines a tag of the form `lab#N_name`. You will find the required tag name in the lab exercise sheet.
>
> Example tags: `lab#0_setup`, `lab#1_and2gate`, `lab#2_or2gate`

Daily Tasks

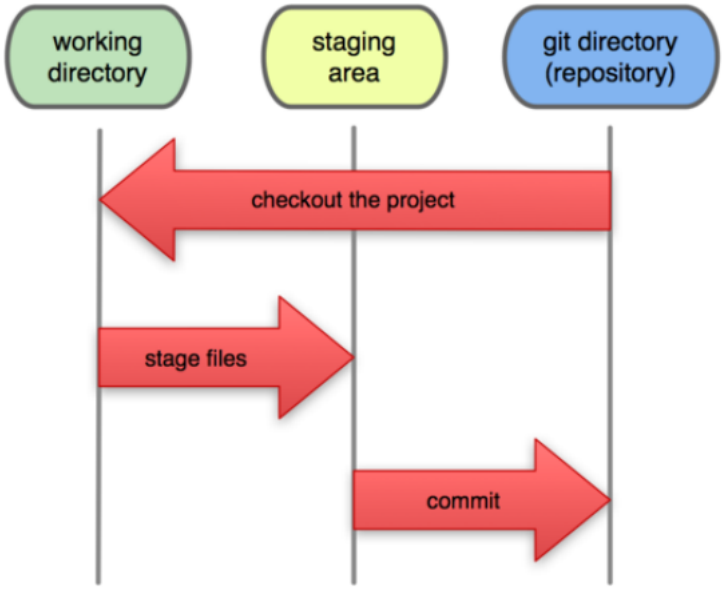

Figure 3: Local Operations

- Checking the Status
- Viewing the Difference of Modified Files
- Add Files and Commit Changes
- Reload Missing Files
- Add Tags
- Synchronise with Remote Repository

Features of GitLab Web Interface

GitLab User Documentation

- Add Members to Remote Repository
- Bug Tracking
- Code Review
- Workflow Management
- User Management
- Multiple Project Managment

Summary

- Why using Version Management
- Create a Remote Repository
- Clone a Remote Repository
- Commit changes and mark milestones with tags
- Invite Team Members