Git and Github

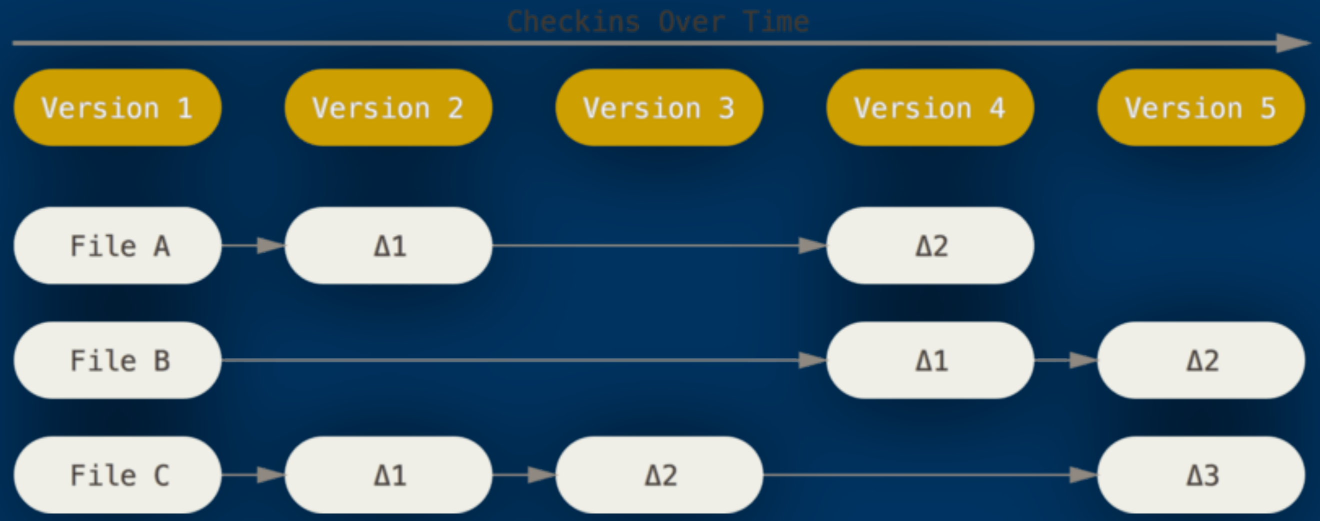
The most popular version control system

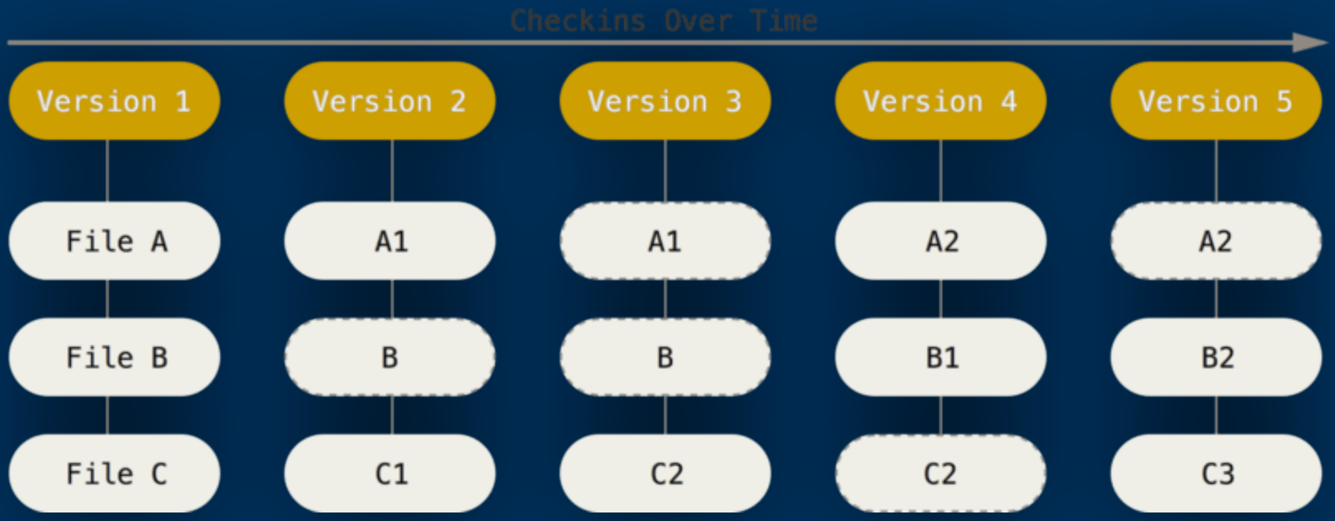
Frank Würthner

Introduction

The Principle

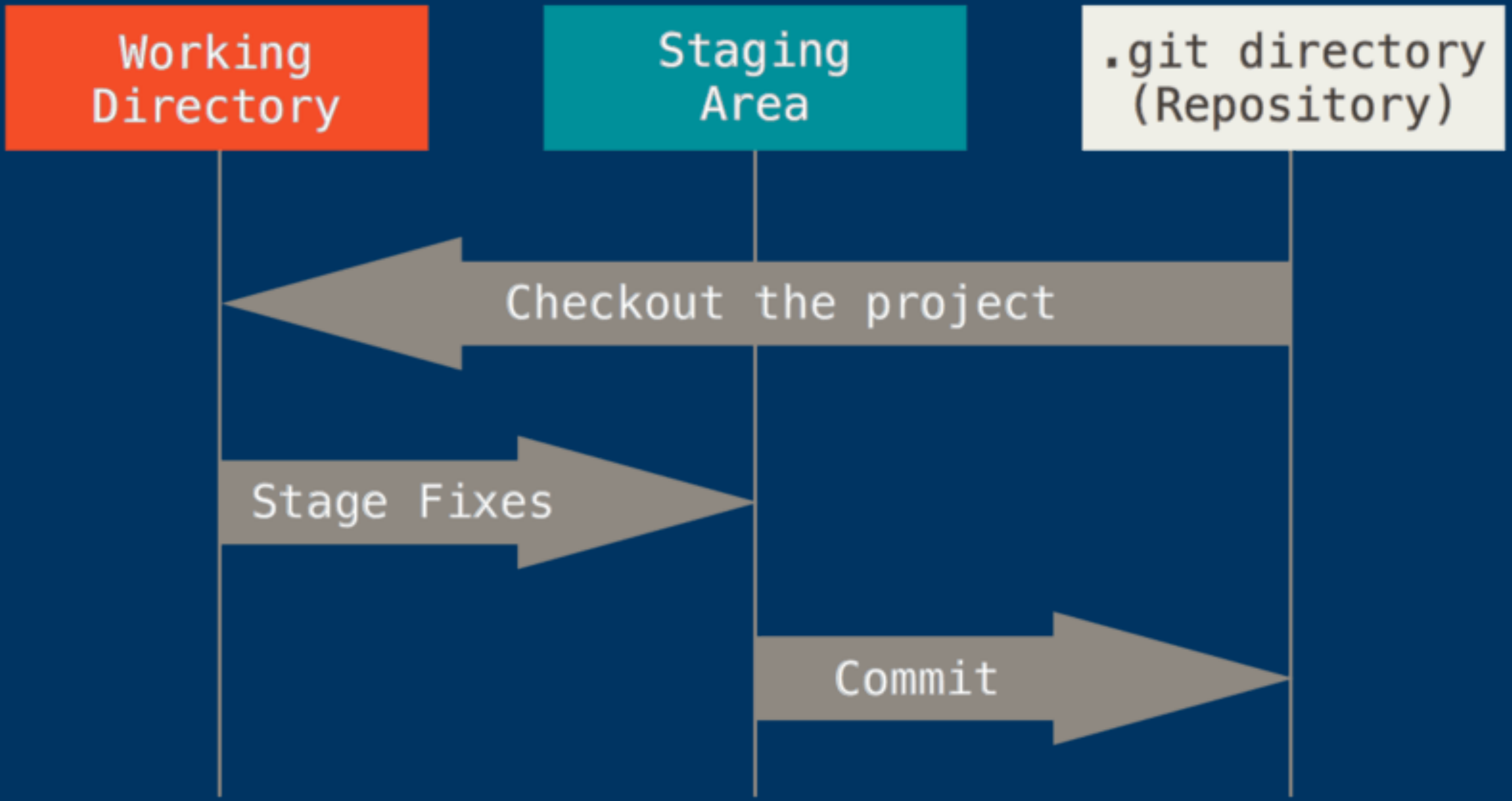
- Track changes in your project
- Revert files or the entire project to a previous state
- See who wrote which line of code: *git blame*
- Git does not track changes, but **snapshots** of a *"miniature filesystem"*





Git tracks versions through **commits**

Three States



- **Modified:** Changed file(s) but not committed to database
- **Staged:** Marked to be included in next commit snapshot
- **Committed:** Data stored in database
- State of files can be seen with *git status*

Basics

- **Initialising** a directory with git: *git init*
    - Metadata is stored in *.git*
- Add files for git to **track** / add files for the next commit: *git add* (*—patch*) *<filename>*
- Save changes: *git commit -m “..”*
- Viewing history: *git log* (*—oneline —graph*)
- Changing previous commit: *git commit —amend* (*-m “..”*)
- Unstaging changes: *git restore —staged*
- **Resetting** back to previous commits: *git reset —soft/—hard*
- **Revert** previous change by creating new commit: *git revert*

Branching

Make changes to a copy of your project without messing up with the main code.

- **List** branches: *git branch*
- Create new branch: *git branch <branch-name>*
- Create and switch to branch: *git checkout -b <branch>*
- Delete branch: *git branch -d <branch>*
- Merge branch into current: *git merge <branch>*
- **Switch** branch: *git switch (-c / -)*
- Stashing changes: *git stash (list / pop / drop)*

Forking and Pull Requests

- Fork: Makes a copy of the repository
- Collaborate on projects where you don’t have access
- Code review from other members before accepting changes
- Works based on branches, **not** individual commits
  - You propose to merge your branch into the project