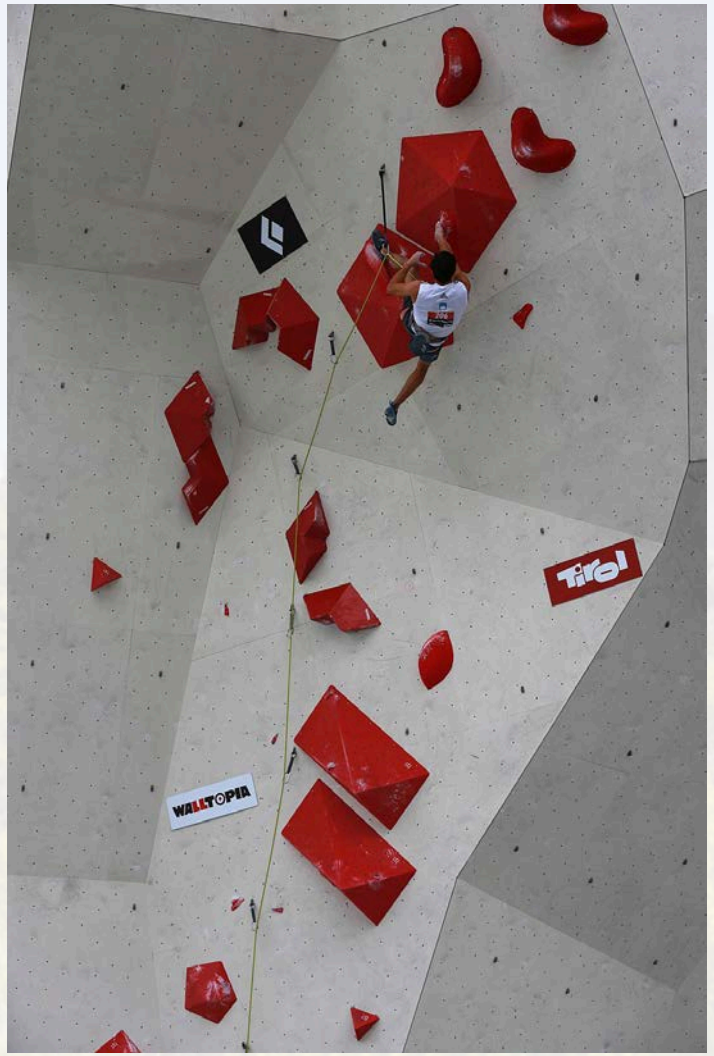
Version Control and You

Carl Schaffer - April 2020

Intro



Topics

- Git: What can it do, how can you profit?
- GitLab at KIS: Platform for software projects
- What does git have to do with rock climbing?

Keep your hand raised if you've ever:

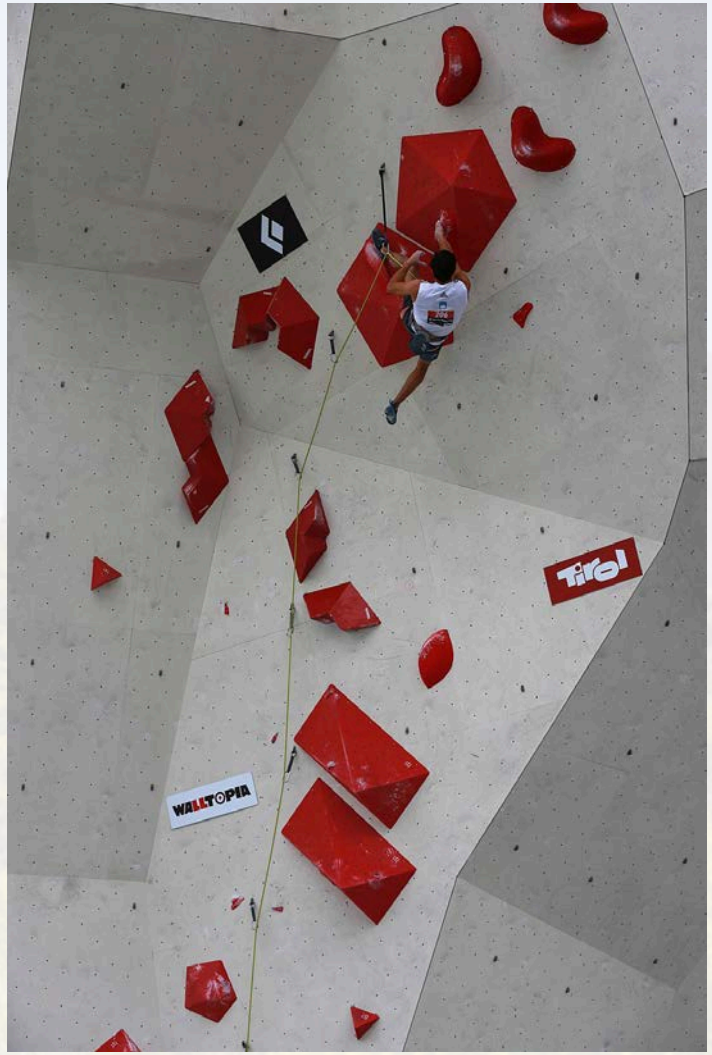
- Used a Version Control System
- Used git
- Created a git commit
- Worked on a branch other than master
- Created a Pull request
- Forked a repository
- Rewritten history of a git project

Why Software Development is like Rock Climbing

- Strenuous
- Correct route unclear
- Potentially fatal errors
- Safety net needed

How does a VCS help?

- Lock in stable versions (like anchor points)
- Provide history of (like rope)
- Backup our path on remote servers



VCS Intro

What is a version control system?

- Version control tracks changes of a (collection of) document (s)
- This can be basically anything: Source code, legal documents, documentation, scientific publication
- A snapshot of such a collection is a **revision**
- All revisions together are the full **history** of our project

Why use it?

- Allows us to o back to arbitrary revisions
- Show difference between revisions
- Enable collaboration
- Acts as backup if combined with remote server

- Created by Linus Torvalds in 2005 for the Linux Kernel
- Currently the standard tool for version control, especially in free and open source software
- Available as command line tool for all mayor operating systems, integrated into modern IDEs, various platforms or graphical interfaces built on top (GitHub, Bitbucket, GitLab)

Makes answering the following questions easy:

What? What changed from revision a to revision b?

Who? Who made a change? Who contributed?

Why? Git encourages or even forces adding explanations to changes.

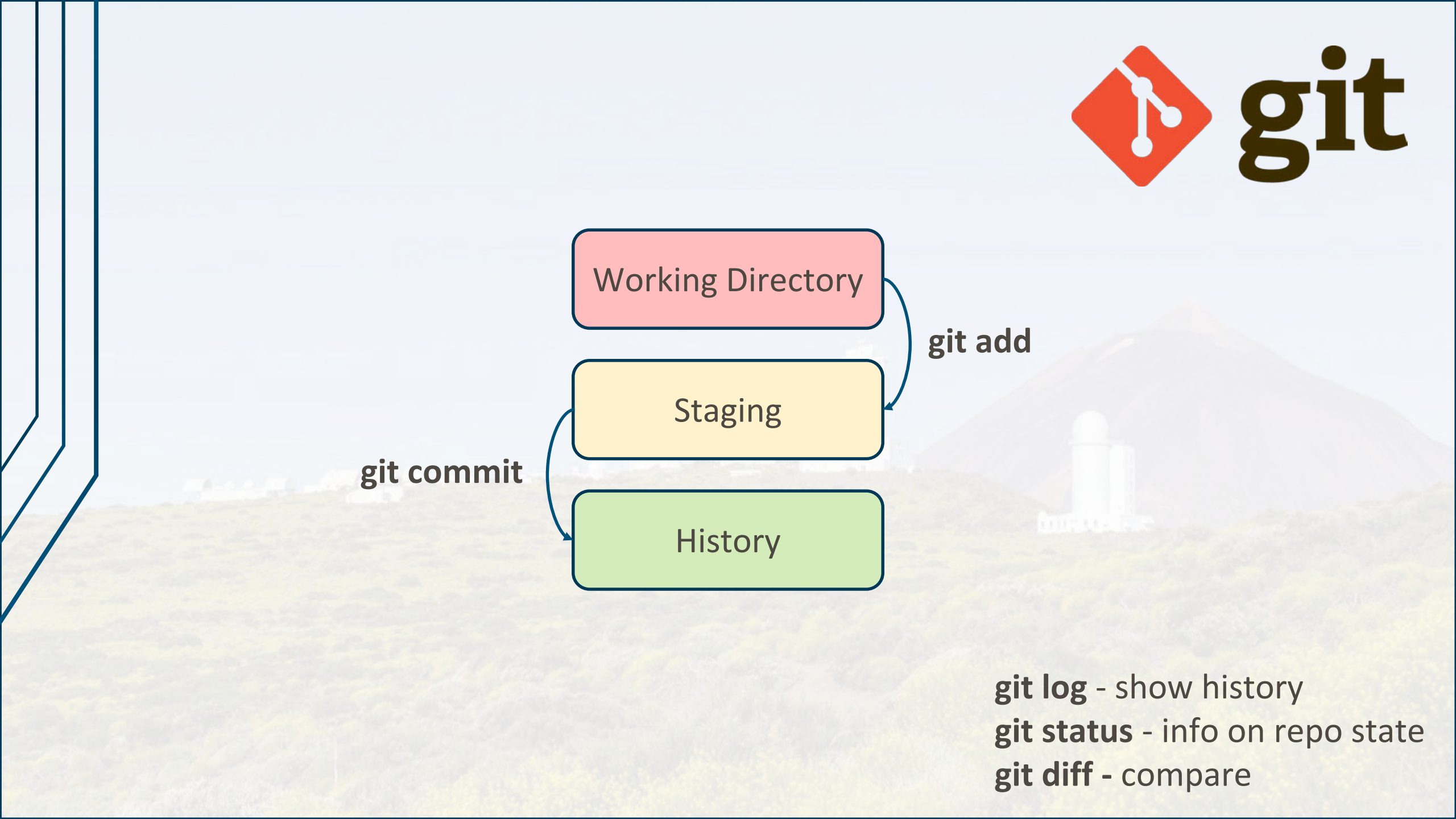
When? In which revision did something change?

Where are changes stored in git?

Working Directory	What actually is on disk in the current working directory.
Staging	Changes that are saved to go into the next commit.
History	The history of the project. All changes ever made. A directed Acyclic Graph of commits.

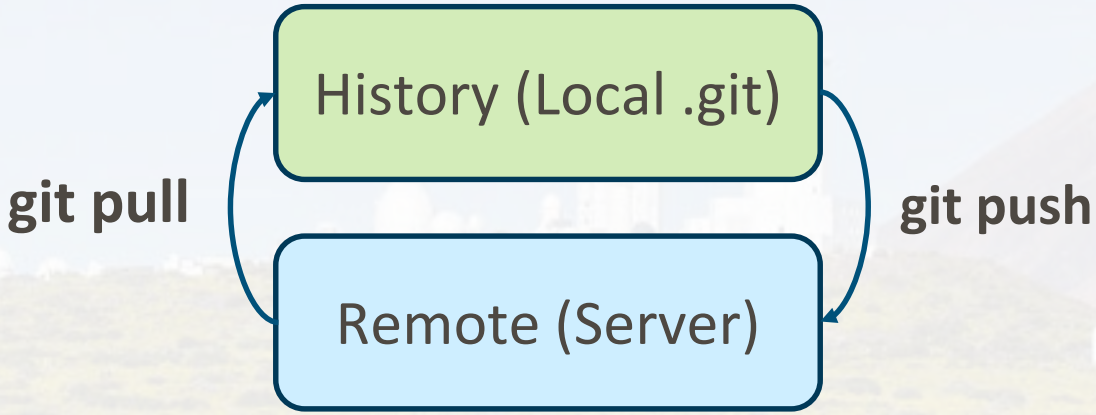
git init creates a git repository in the current working directory

→ All git data is stored in the **.git** directory

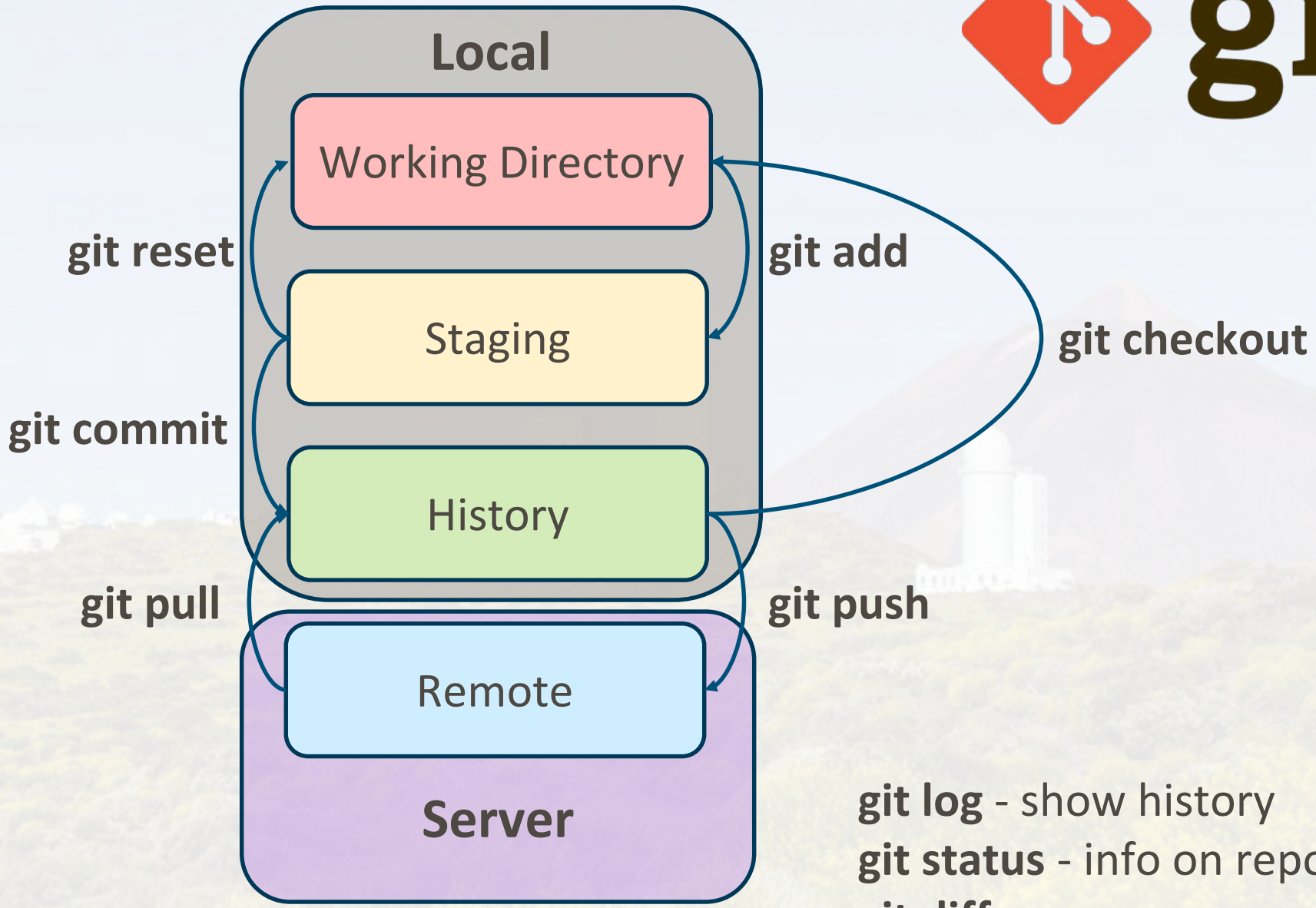


git log - show history
git status - info on repo state
git diff - compare

Remotes are central places, e.g. servers, where repositories can be saved and which can be used to synchronize different clients.



Adding a remote:
git remote add <name> <url>
Example: **git remote add origin** https://gitlab.leibniz-kis.de/sdc/kis_tools
The main remote is canonically named origin, automatically when using **git clone**



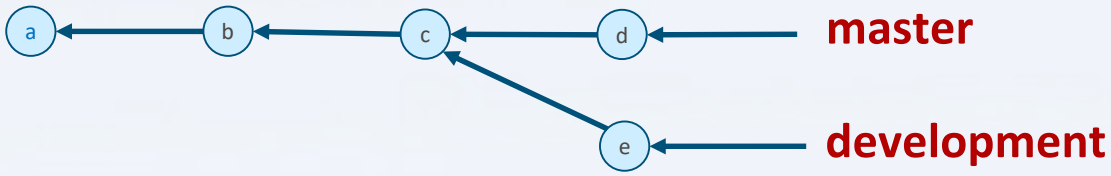
git log - show history
git status - info on repo state
git diff - compare

History

Commit: State/Content at a given time

- Contains a commit message to describe the changes
- Commits always point to their parent(s)
- Commits are identified by a hash of the content, message, author(s), parent(s)

History



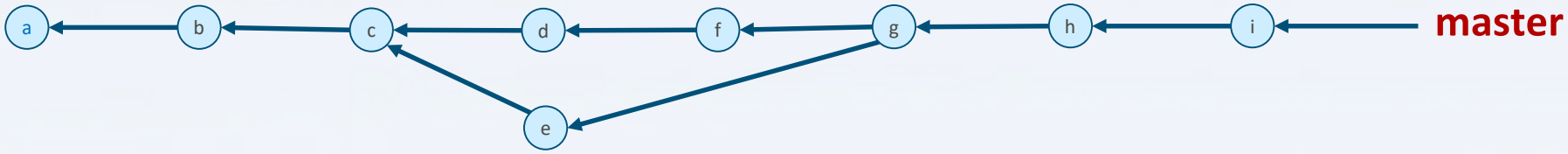
Commit: State/Content at a given time

- Contains a commit message to describe the changes
- Commits always point to their parent(s)
- Commits are identified by a hash of the content, message, author(s), parent(s)

Branch: A named pointer to a commit

- Development branches
- Main branch: master
- Moves to the next child if a commit is added

History



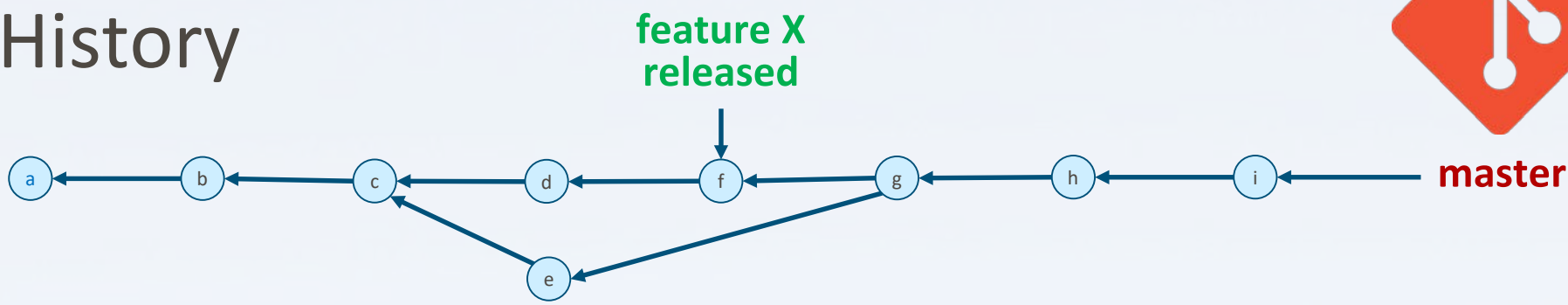
Commit: State/Content at a given time

- Contains a commit message to describe the changes
- Commits always point to their parent(s)
- Commits are identified by a hash of the content, message, author(s), parent(s)

Branch: A named pointer to a commit

- Development branches
- Main branch: master
- Moves to the next child if a commit is added

History



Commit: State/Content at a given time

- Contains a commit message to describe the changes
- Commits always point to their parent(s)
- Commits are identified by a hash of the content, message, author(s), parent(s)

Branch: A named pointer to a commit

- Development branches
- Main branch: master
- Moves to the next child if a commit is added

Tag: Fixed, named pointer to a commit

- For important revisions, e.g. release versions or version used for a certain paper

Typical single-branch Workflow

If new Create or clone repository : git init, git clone <url>

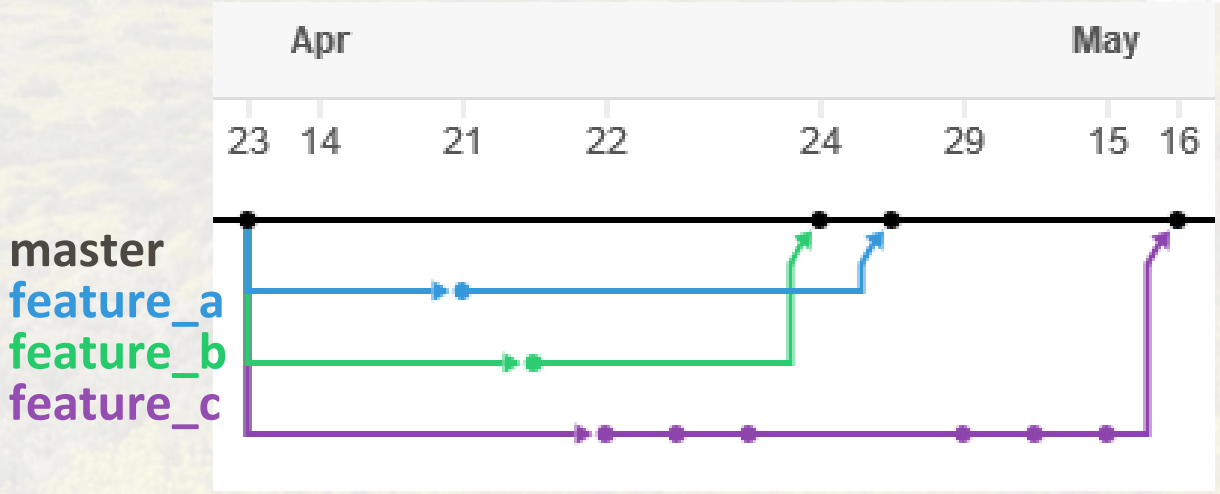
If exists git pull

1. Work
    - 1.1 Edit files and build/test
    - 1.2 Add changes to the next commit: git add
    - 1.3 Save added changes in the history as *commit*: git commit
2. Download commits that happened in the meantime: git pull
3. Upload your own: git push

Typical multi-branch Workflow

There are multiple models of working together with git using branches
Simplest and most popular: “GitHub-Workflow”

- Nobody directly commits into the master branch
- For each new feature / change / bugfix a new branch is created
- Branches should be rather short-lived
- Merge into master as soon as possible, then delete branch
- Master should always contain a working version



GitLab Showcase

GitLab Infrastructure at KIS : gitlab.leibniz-kis.de

Git-based software development platform

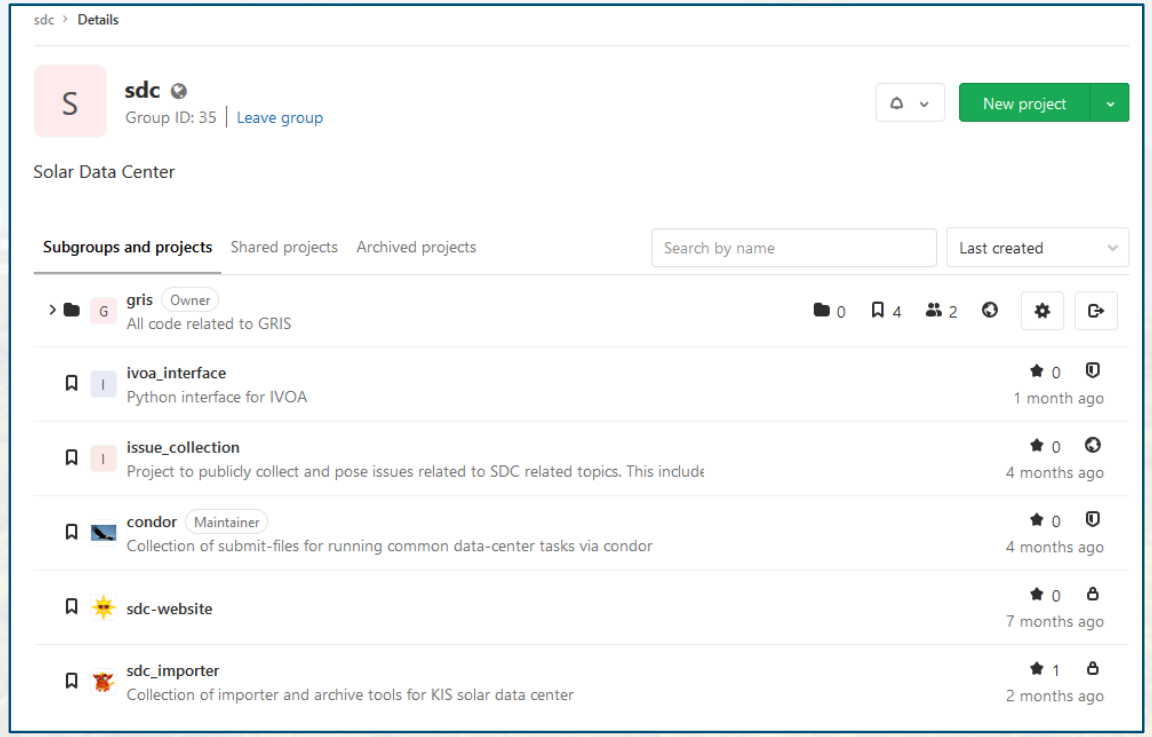
"GitLab started as an open source project to help teams collaborate on software development. [ ] GitLab now provides a single application for the entire software development and operations lifecycle."

Competitors: GitHub, Bitbucket

Useful features (not complete):

- Self-hosted git server (Remote)
- Perform git actions via browser
- Communicate with collaborators
- Display content
  - Documentation: Markdown
  - Results: Jupyter notebooks
- Share code
- Track Issues for your project

Example: [chroquest]



Conclusion

- Git is currently **THE** industry standard, the vast majority of modern software development uses git for version control
- If used correctly git can help you:
  - Document changes to code: What? When? Who? Why?
  - Ensure reproducibility in your results
  - Collaborate with other developers
- Our GitLab Server makes this tool even more powerful and useful

Further Resources:

- In depth git tutorial on DataCamp
- Atlassian git tutorials
- SDC wiki on setup at KIS

Be smart, be safe, use a rope!

Thanks!