Further Git features

Bálint Aradi

Course: Scientific Programming / Wissenchaftliches Programmieren (Python)

https://www.bccms.uni-bremen.de/people/b-aradi/wissen-progr/python/2023

Outline

- Further Git features
- Hosting Git-repositories

Rename files

- **Rename a file** under version control:

```
git mv README README.txt
git status
On branch main
Changes to be committed:

renamed: README -> README.txt

git commit -m "Add .txt extension to readme file"
```

- Corresponding file in working directory will be renamed immediately
- The name change must be committed like any other change

Delete files

- **Delete (remove)** a file under version control

```
git rm unnecessary_file
git status
On branch main
Changes to be committed:

deleted:     unnecesary_file

git commit -m "Delete unnecessary file"
```

- Corresponding file in the working directory will be deleted immediately
- The removal must be committed like any other change
- The file will be not present in **future revisions**, but stays part of the previous commits!

Investigating changes

- Changes between **working copy and last checked in / staged version**

```
git diff README.txt
diff --git a/README.txt b/README.txt
index 8eab0a7..770eee5 100644
--- a/README.txt
+++ b/README.txt
@@ -1,5 +1,5 @@
-**********
-Linsolvers
-**********
+*********
+Linsolver
+*********
```

Lines removed

Lines added

- If no file name is specified, all changes in all version controlled files are shown

```
git diff
```

- Changes between **two committed revisions** using revision hashes

```
git diff 04d386 2a3186
-- README.txt
```

Optional, if missing changes in all files shown

- The **difftool** sub-command calls the default diff-viewer (kdiff3, meld) to **visualize changes**

```
git difftool
Viewing (1/1): 'README.rst'
Launch 'kdiff3' [Y/n]?
```

Discard changes in working copy

- **Set working directory back** to last committed / staged version:

```
git status
[...]

   modified:   README.txt

no changes added to commit (use "git add" and/or "git commit -a")

git restore README.txt
git status
[...]
nothing to commit, working tree clean
```

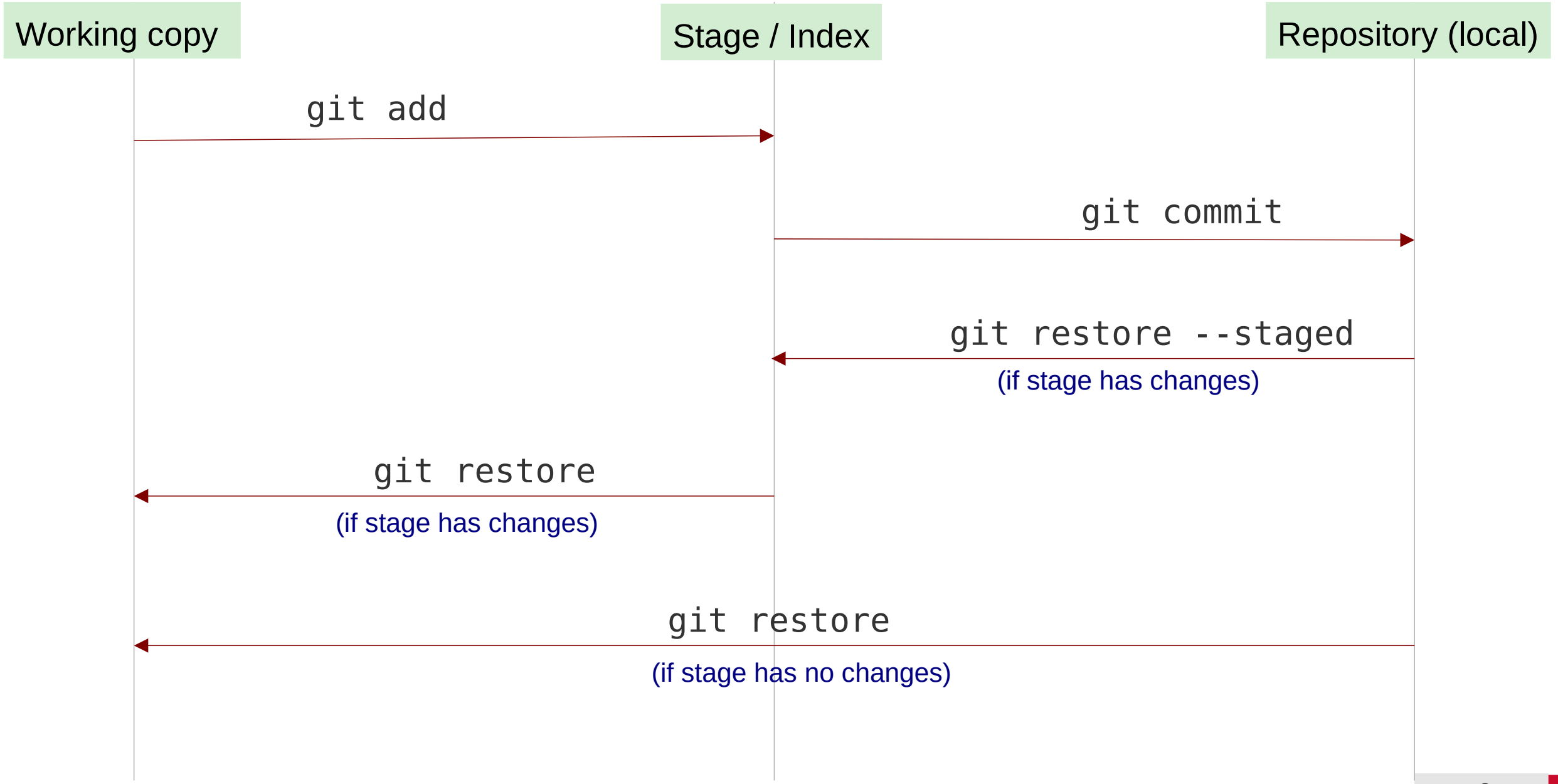
Overwrites working copy!

Unstage files

- Staged files can be **unstaged**, if they should not be part of the next commit
- Corresponding file in the work directory is not affected by the opertation

```
git status
On branch main
Changes to be committed:

   modified:   README.txt

git restore --staged README.txt

git status
On branch main
Changes not staged for commit:

   modified:   README.txt
```

Overview: Git file transfer commands



Switch to an earlier version

- **Switching to earlier commit** (by specifying its hash value)

```
git switch --detach 2a31862
HEAD is now at 2a31862 Add readme file

git status
HEAD detached at 2a31862
```

- You have to change back to the current version (or to create a branch) to commit any changes

```
git switch main
Switched to branch 'main'
```

HEAD	Points to last commit in the repository, next commit will be attached to this commit
Detached HEAD	Points to an earlier commit, no new commit can be appended (unless a new branch is created from this point)

Tagging versions

- **Commits with special importance** (e.g. release) can be tagged
- Annotated tags are commited with a log-message
- By default the last checked in commit is tagged

```
git log --oneline
7d8cf66 (HEAD -> main, origin/main) Add readme and .gitignore
ddd085a Start linsolver project with stub files

git tag -a v0.1

git log --oneline
7d8cf66 (HEAD -> main, tag: v0.1, origin/main) Add readme and .gitignore
ddd085a Start linsolver project with stub files
```

- Tag names can be used instead of revision hashes in git commands

```
git diff 04d3866 v0.1
```

Git aliases

- Aliases help to **abbreviate often used git commands and options**

```
git config --global alias.ci commit
git config --global alias.sw switch
git config --global alias.swd "switch --detach"
git config --global alias.st status
git config --global alias.gdiff difftool
```

Please create these aliases for your account, since the following examples will make use of them!

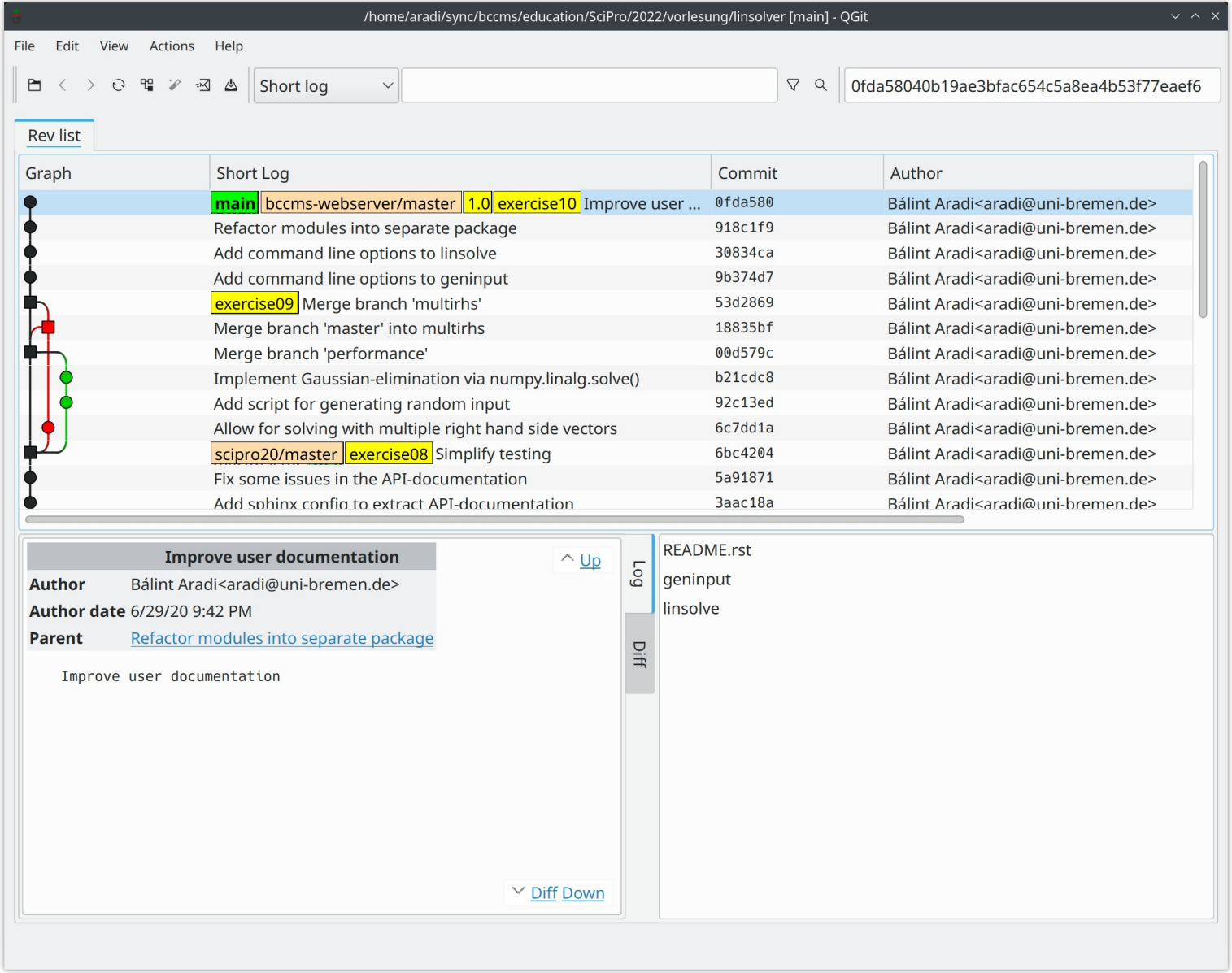
- If an alias is used, the corresponding command / options will be substituted

```
git ci -m "Add quick changes"
git swd 2a31862
git st
git gdiff README.rst
```

Graphical git-viewers

- Several graphical git-clients exist to visualize development history:

```
qgit &
```



Tracking remote repositories

- Remote repository must be cloned first to create a local git repository

```
git clone https://github.com/SciProBA/linsolver.git
Cloning into 'linsolver'...
```

- Remote repository will be associated with the new local repository (under then name “origin”)

```
git remote -v
origin  https://github.com/SciProBA/linsolver.git (fetch)
origin  https://github.com/SciProBA/linsolver.git (push)
git status
On branch main
Your branch is up to date with 'origin/main'.
```

- Recent changes in the remote repository can be pulled

```
git pull
Updating ddd085a..7d8cf66
```

Note: This only works without side-effects, if the local repository was not modifed apart of “git pull” calls.

Some further git-notes

- **Read the manual** for detailed git options
- You should **commit after each non-trivial change** of the project.
  Rule of thumb: It should be easy for other developers to follow and understand the changes of a commit.
- One commit should always contains **logically related changes**.
- Version history is stored in the .git sub-directory. If it is copied with the project, the version history is copied as well.
- Git commands must be executed in the project directory or in a subdirectory of it.
- If no files are specified, git commands have the entire project (the files which are already under version control) as target
- Revision hashes are global: They represent the status of all files in the project to a given time.

Hosting git repositories

- Raw Git repositories can be easily hosted on any webserver
- Hosting Git repositories with user friendly web interfaces is also possible, but more complex (e.g. running a GitLab server)
- Several companies offer free of charge git repository hosting (with some constraints):
  - GitHub (Microsoft): https://github.com
  - GitLab (GitLab Inc.): https://gitlab.com
  - Bitbucket (Atlassian): https://bitbucket.org

See https://github.com/SciProBA/linsolver for an example …

Have fun!