Workshop git

Fabio Gratl
Technical University of Munich
Faculty of Informatics
Chair of Scientific Computing in Computer Science (SCCS)
TU Kaiserslautern, 12.10.2018
What is git?

• Officially:
  git - the stupid content tracker
• Distributed Version Control System (DVCS)
  ⇒ local + remote repository!
• git-scm.com
Getting help

- **Manpages:** `man git`, `man gittutorial`, `man giteveryday`
- **Built-in help:** `git help $COMMAND` *(same as `man git-$COMMAND`)*
- **GUI tools for all platforms exist (but not covered here).**
- **IDEs can handle git** (eclipse, intelliJ, CLion...)
A new repository (GitHub)

• Create a new repository on GitHub or look for one you like.
A new repository (GitLab)

- Create a new repository on GitLab or look for one you like.
Bring it on your machine (GitHub)

- Clone via **ssh** (requires public-key upload¹) or https.

Bring it on your machine (GitHub)

- Clone via **ssh** (requires public-key upload\(^1\)) or https.

Bring it on your machine (GitLab)

- Clone via **ssh** (requires public-key upload[^2]) or https.

[^2]: [https://docs.gitlab.com/ee/ssh/](https://docs.gitlab.com/ee/ssh/)
Bring it on your machine

`git clone <repository> [directory]`

- Copies remote repository to local machine.
- Fetches all branches.
- Requires target folder to be empty!

Example:

```
1 ~$ git clone git@gitlab.lrz.de:ga68cat/DemoRepo.git
2 Cloning into 'DemoRepo'...
3 ~$ cd DemoRepo
4 ~/DemoRepo$ ls -a
5 .. .git README.md
```
Gaining Overview

**git status**

- Show status of current working copy.
- List modifications, new files, deletions, merge conflicts...
- Always provides hints what to do! (not shown here)

Example:

```
~/DemoRepo$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean

~/DemoRepo$ git log -1
commit 139e2f8be08bb6fba96b27fd30f31008880584d4
Author: FG–TUM <FG–TUM@users.noreply.github.com>
Date: Fri Oct 5 11:52:47 2018 +0200
Initial commit
```
Ignoring Stuff

.gitignore

- List of files and paths to be ignored by git.
- Accepts "*" as wildcard.
- This file should be uploaded as any other.
- Useful for output or IDE files.

Example:

```
1 ~/DemoRepo$ touch a.foo b.foo
2 ~/DemoRepo$ echo "*.foo" >> .gitignore
3 ~/DemoRepo$ mkdir dir && touch dir/one dir/two dir/c.foo
4 ~/DemoRepo$ echo "dir" >> .gitignore
```
Small Changes

- **Modify** README.md
- **Create a new file** FileA.txt

```bash
~/DemoRepo$ echo "some useless text..." >> README.md
~/DemoRepo$ echo "Let there be text\nCommon line." >> FileA.txt
~/DemoRepo$ ls
FileA.txt README.md
~/DemoRepo$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes not staged for commit:
  modified: README.md
Untracked files:
  FileA.txt
no changes added to commit
```
Committing Changes

**git add**

- Track new files.
- Stage existing files.

**git commit** [--amend | --message | --signoff]

- Commit changes to local repository.
- Checkpoint to revert or compare to.

Example:

1. `~/DemoRepo$ git add FileA.txt README.md`
2. `~/DemoRepo$ git status`
3. On branch master
4. Your branch is up—to—date with ’origin/master’.
5. Changes to be committed:
   - new file: FileA.txt
   - modified: README.md
6. `~/DemoRepo$ git commit --message "meaningful message"
7. `~/DemoRepo$ git status`
8. On branch master
9. Your branch is ahead of ’origin/master’ by 1 commit.
Pushing Changes

git push

• Sends all committed changes to remote.

Example:

```
1 ~/DemoRepo$ git push
2 To git@gitlab.lrz.de:ga68cat/DemoRepo.git
3 139e2f8..014affb master → master
```
Pulling Changes

**git pull [--rebase]**

- Updates all files in the local branch.
- Updates information about other remote branches.
- `rebase`: Put local changes on top of remote instead of merging.

Example: (Suppose someone added a line break in README.md.)

```
1 ~/DemoRepo$ git pull
2 From git@gitlab.lrz.de:ga68cat/DemoRepo.git
3 014affb..d20ae23 master → origin/master
4 Updating 014affb..d20ae23
5 Fast-forward
6 README.md 1 +
7 1 file changed, 1 insertion(+)
```
Starting a new Feature

**git checkout** `<branch> | <File>`
- Switch to existing branch.
- OR reset `<File>` to last commit.

**git branch** `[<new_branch>]`
- Create a new branch.
- OR list all local branches.

Example:

```
~/DemoRepo$ git branch branchForAwesomeFeature
~/DemoRepo$ git branch
branchForAwesomeFeature
* master
~/DemoRepo$ git checkout branchForAwesomeFeature
Switched to branch 'branchForAwesomeFeature'
```
Merging branches

\texttt{git merge <branch>}

• Merges given branch in current branch.

Example: (Suppose we changed FileA.txt on the new branch and someone else on master.)

\begin{verbatim}
1 ~/DemoRepo$ git merge master
2 Auto-merging FileA.txt
3 CONFLICT (content): Merge conflict in FileA.txt
4 Automatic merge failed; fix conflicts and then commit the result.
5 ~/DemoRepo$ git status
6 On branch branchForAwesomeFeature
7 You have unmerged paths.
8 Unmerged paths:
9   both modified: FileA.txt
\end{verbatim}
Resolving conflicts

**git mergetool** (needs to be configured)

- vimdiff, Meld, SmartGit...
- IDEs

Example: Merge view in CLion:
Resolving conflicts

git mergetool (needs to be configured)

• vimdiff, Meld, SmartGit...
• IDEs

Example: Merge view in CLion:
Resolving conflicts

**git mergetool** (needs to be configured)

- vimdiff, Meld, SmartGit...
- IDEs

Example: Merge view in CLion:
Resolving conflicts

git mergetool (needs to be configured)

• vimdiff, Meld, SmartGit...
• IDEs

Example: Merge view in CLion:
Resolving conflicts

**git mergetool** (needs to be configured)

- vimdiff, Meld, SmartGit...
- IDEs

Example: Merge view in CLion:
Resolving conflicts

**git mergetool** (needs to be configured)

- vimdiff, Meld, SmartGit...
- IDEs
- After resolving all conflicts: `git add` and `git commit`.

Example: Merge view in CLion:
Merging back to master (GitHub)
Merging back to master (GitHub)
Merging back to master (GitLab)
Merging back to master (GitLab)
Summary

- git pull
- `git branch` branchForAwesomeFeature
- `git checkout` branchForAwesomeFeature
- Do what you must...
- `git commit`
- `git merge` master
- `git push`
- Create pull request

And anytime you feel lost:

- `git status`
Backup Content
Initialize an existing Repository

**git init**
- Create a new local repository (.git folder)

**git remote** add <name> <url>
- Manage remote tracking repositories.

---

Example: First create repository on GitLab³

```bash
~/DemoRepo$ git init
Initialized empty Git repository in ~/DemoRepo/.git/
~/DemoRepo$ git add . && git commit --m"First commit"
~/DemoRepo$ git remote add origin git@gitlab.lrz.de:ga68cat/DemoRepo.git
~/DemoRepo$ git remote -v
origin git@gitlab.lrz.de:ga68cat/DemoRepo.git (fetch)
origin git@gitlab.lrz.de:ga68cat/DemoRepo.git (push)
```

Undoing a Commit

**git revert**

```
[--soft | --hard] [<commit>]
```

- Undo commit (not pushed yet!).
- soft: preserve changes.
- hard: revert changes.

**Example:**

```bash
~/DemoRepo$ git reset --soft HEAD~1
~/DemoRepo$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  new file: FileA.txt
  modified: README.md
~/DemoRepo$ git reset HEAD README.md
Unstaged changes after reset:
M README.md
```
Who did that!?

```
who blane [-w] <file>
```

- Show author of last line change.
- `-w` Exclude whitespace-only changes

Example: (Suppose there is a second author)

```bash
~/AutoPas$ git blame README.md
f5344b80 README.md (FG-TUM 2018-07-31 15:53:56 +0200 1) # AutoPas
```

CLion: Annotations
Put stuff aside

**git stash** [save <message> | list | pop [<stash>]]

- Save all current changes with a message.
- List all stashes.
- Apply a stash and remove it from the list.

Example:

```
1 ~/DemoRepo$ git checkout branchForAwesomeFeature
2 error: Your local changes to the following files would be overwritten by checkout:
   3       README.md
4 Please commit your changes or stash them before you switch branches.
5 Aborting
6 ~/DemoRepo$ git stash save "my message"
7 Saved working directory and index state On master: my message
8 ~/DemoRepo$ git stash list
9 stash@{0}: On master: my message
10 ~/DemoRepo$ git stash pop
11 Changes not staged for commit:
12  README.md
```