Introduction to *git*

Vincent Oberhauser

Universität Zürich

November 4, 2021

Slides

Please Visit `https://github.com/UpperDweller/git_intro` and download the session_git.pdf.

Introduction

Git is an Distributed Version Control System.
This means it:

- Tracks the content of stored data.
- Maintains a history of the changes applied to the data.
- Functions as a local repository in an indivudal computer as well as a remote repository on a server.

Use Cases

Possible use cases are:

- Backing up data.
- Keeping track of the changes of code allowing for reverting to an earlier state or branching out new features.
- Working on a project with multiple collaborators and keeping everybody up to date with the newest version.

Installation

First we need to install *git*:

- On Windows visit `https://git-scm.com/download/win`, download the correct version for your system and install it by running the .exe file and proceeding through all pages with default options.
- On Mac visit `https://sourceforge.net/projects/git-osx-installer/`, download the correct version for your system and install it by running the .dmg file.

Navigating to the desired directory

Next we open a terminal:

- Create a directory named **git_demo**
- Right click on the directory and choose **Git Bash Here** (Windows)/**Services** $\rightarrow$ **New Terminal at Folder**(Mac)
- Type **git --version** to check if *git* is installed

Add your credentials to git. This helps to identify contributors of a project:

- **git config --global user.email "you@example.com"**
- **git config --global user.name "Your Name"**

Initializing local repository

Now we can create a local repository at the current location:

- **git init** - This command prepares the directory to function as a repository

Staging and Comitting

Now we can add files to the repository:

- In the directory create a textfile, enter "some text" to it and save it as **test.txt**.
- **git add test.txt** - Before files can be added to a repository they need to be added to the staging area. This is done using the former command. (git add . moves all changed files to the staging area)
- **git commit -m "commit message"** - A commit is a snapshot of the current version of a project. Use the commit message to describe the features of the version comitted.

Log and Status

- Now modify the test.txt file adding "more text" and saving it.
- **git status** - Use this command to find out what files are modified and what files are in the staging area.
- Now stage and commit the changes to the repository
- **git log** - This command prints out all the commits which have been done up until now.
- **git reset --hard <commit>** - With this command you can reset your repository to a previous commit. The commit to reset to may be entered using the first 9 letters of the commit or with HEAD~n; where n indicates the number of previous commits to reset. (We will make use of this later)

Branch and Merge

Usually you are working on the *master branch*. However, working on a project with multiple collaborators or testing new features without endangering the working code may require creating new branches:

- **git branch branch_name** - The branch command creates a new branch with the chosen name copying the current state of the master branch.
- **git checkout branch_name** - The new branch is accessed. All changes and commits are now only performed on this branch
- Add "another line" to the text file and save it, then stage and commit it.
- Check out back to the masterbranch. Check with **git log** if the changes applied to it.
- **git merge branch_name** - The merge command brings together the changes made on the branch and the status of the master branch. You may check this out using **git log**.

Using github as remote repository

While a local repository serves as a means of version control and backup facilitation a remote repository allows for sharing of projects and collaboration:

- Visit `https://github.com/` and create an account.
- Go to **Settings** → **Developer Settings** → **Personal access tokens** and generate a new token with the Scope **repo**.
- Create a new github repository.

Add remote and push

Now you can add your github repository as a remote to your local repository and move your project to it:

- Copy the URL you find in your GitHub repository to the clipboard
- **git remote add <remote_name> URL** - Using a name for the remote and the copied url you may now add your remote repository.
- **git push remote_name master** - Transmit the current commit of your master branch to the remote repository. Use your GitHub-username and the generated access token as login credentials.

Pull and Clone

A remote may not only serve as a place to save a current commit but also functions as a place to gather commits ahead of the local version:

- Reset your local version back one commit by entering **git reset --hard HEAD 1**
- **git pull remote_name master** - The pull command changes the local repository to the state of the remote repository if it contains more recent commits. This could be due to working on several machines or with several users.
- **git clone https://github.com/UpperDweller/git_intro** - This command clones an existing remote repository into your computer. Entering the command will put the slides of this presentation into your repository.

Thank you for your attention!