Dive in to Git and GitHub

CompiCampus

2019-02-28

Roman Plessl

Agenda and Goals for Today

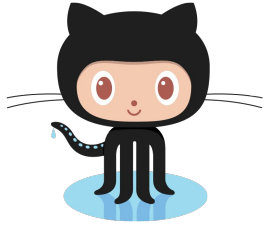
1. Git - Introduction to version control systems, History of Git
2. Install Git / Use Git at ETH
3. Basic Git Workflow, How to use local Git with GitHub
4. How to use GitHub (I):
   a. Create a GitHub Account, create new Repositories
   b. Local Git Checkouts and GitHub
   c. Interact with each other's and other coders:
      Git Branching, Code Changes and Commits, Pull Requests, Merging, Comments
5. Host your personal, organization, and project sites with GitHub Pages.
6. How to use GitHub (II): Issues Tracking and Feature Requests, Task Management and Wiki, Insights and Settings
7. Further Reading and Exercises

Git

- **Git** is a distributed version-control system for tracking changes in source code.
- **Git** was created by Linus Torvalds in 2005 for development of the Linux kernel, with other kernel developers contributing to its initial development.
- **Git** is maintained since 2005 by @gitster - Junio C Hamano.
- Over 70% of all Code Projects use Git.

GitHub

- **GitHub** is a web-based hosting service for version control using Git. It is mostly used for computer code.
- **GitHub** offers all of the distributed version control and source code management functionality of Git as well as adding its own features (bug tracking, feature requests, task management, and wikis for every project)
- **GitHub** was bought 2018 by Microsoft.

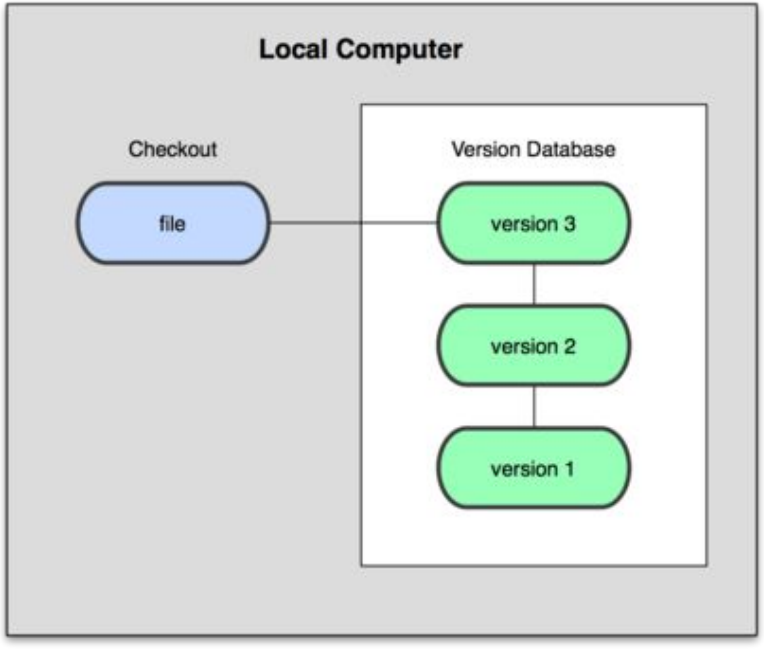


Introduction Version Control System

What is a version control system?

Version control system (VCS) in general

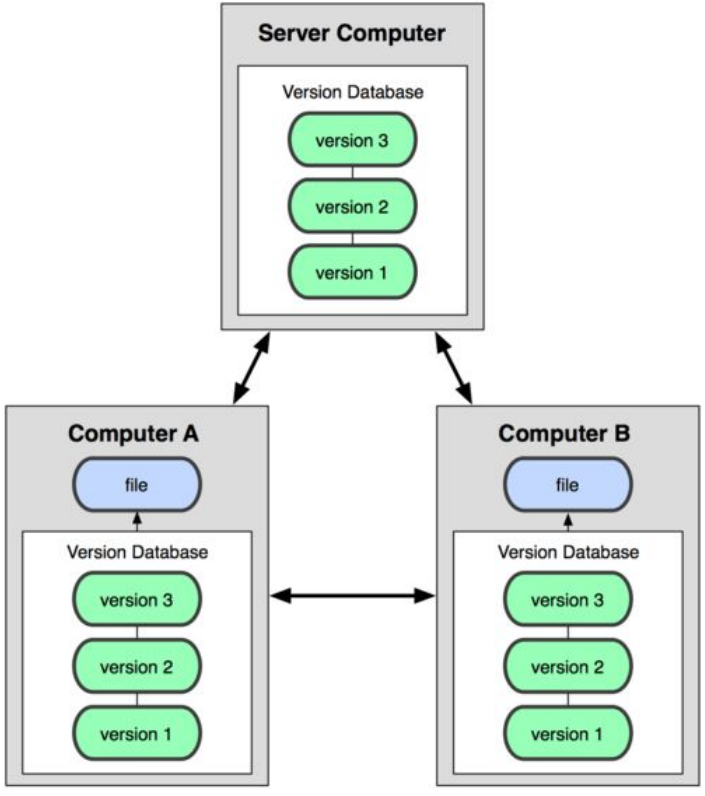
- A system that keeps records of your changes
- Allow for collaborative development
- Allows you to know who made what change and when
- **Allows you to revert any changes and go back to previous state**



Introduction to Git

What is distributed version control?

- Distributed version control
- Users keep entire code and history on their location machines
- Users can make any changes without internet access
- (Except pushing and pulling changes from a remote server)



What is Git?

Git is a **software** that helps you **keep track of changes to files** in a folder on your PC. After making some changes to files in this folder, you can **“commit”** the changes to **Git** for safe-keeping. These changes could be creating, renaming, deleting a file or subfolder; or editing the content of a file.

You can also share this folder you’re “git-tracking” with others (via a local computer network, for example) and they can also make changes to files in the folder and **“commit”** their changes into the mighty hands of Git for safe-keeping.

Then, you can turn back the hands of time whenever you need to and view the content of those files *as they were* at that time in the past. God forbid one of the friends you shared your files with messed things up for you! Oh wait, it was you and not your friend that messed things up! Oh well, we have Git, don’t we?

History about Git

History about Git

- Git started in 2005 by Linus Torvalds (Linux)
- Created by Linus to aid the Linux Kernel development. To help the developer to manage their code change patches (till 2002) and as a replacement for the proprietary and free-of-charge software BitKeeper (till 2005).

History about Git

Design Goals:

- Speed
- Simple design
- Strong support for non-linear development (thousands of parallel branches)
- Fully distributed
- Able to handle large projects like the Linux kernel efficiently (speed and data size)

Daten und Zahlen zu den jüngsten Versionen des Linux-Kernels

Kernel-Version	Anzahl Dateien[1]	Zeilen Quelltext (Ohne Doku)[2]	Entwick-lungs-zeitraum	Commits (Ohne Merges)[3]	Diffstat[4]
Linux 4.14	61.290	25.041.284 (23.050.486)	70 Tage	14.659 (13.452)	23.388 files changed, 719.862 insertions(+), 445.585 deletions(-)
Linux 4.15	62.303	25.364.802 (23.329.451)	77 Tage	16.223 (14.866)	13.265 files changed, 643.912 insertions(+), 320.289 deletions(-)
Linux 4.16	62.915	25.558.805 (23.495.643)	63 Tage	14.896 (13.630)	12.239 files changed, 1.133.069 insertions(+), 939.066 deletions(-)
Linux 4.17	61.362	25.379.564 (23.314.368)	63 Tage	14.745 (13.541)	14.504 files changed, 777.301 insertions(+), 956.941 deletions(-)
Linux 4.18	61.003	25.280.872 (23.183.236)	70 Tage	14.432 (13.283)	13.141 files changed, 583.336 insertions(+), 682.028 deletions(-)
Linux 4.19	61.734	25.588.455 (23.449.221)	70 Tage	15.204 (14.043)	11,693 files changed, 552.223 insertions(+), 244.235 deletions(-)
Linux 4.20	62.481	25.955.520 (23.776.585)	63 Tage	14.995 (13.844)	11.402 files changed, 685.027 insertions(+), 317.959 deletions(-)
Linux 5.0-rc (Stand: 19.02.19)	63.134	26.202.026 (23.932.125)	n. n.	13.561 (12.518)	12.000 files changed, 576.767 insertions(+), 330.262 deletions(-)

[1] git ls-tree -r --name-only HEAD | wc -l
[2] find . -type f -not -regex '\./\.git/.*' | xargs cat | wc -l; echo "($(find . -name *.[hcS] -not -regex '\./\.git/.*' | xargs cat | wc -l))"
[3] git-log --pretty=oneline *vx.(y-1)..vx.(y)* | wc -l; echo "($(git-log --pretty=oneline --no-merges vx.(y-1)..vx.(y) | wc -l))"
[4] git diff --shortstat *vx.(y-1)..vx.(y)*

Facts and Figures for the latest Kernel Changes

Install Git locally

Install Git locally

mac OS:

- https://git-scm.com/download/mac

Windows:

- https://git-scm.com/download/win

Linux:

- Debian / Ubuntu: `sudo apt install git`
- Fedora / RHEL: `sudo yum install git`

Use Git at ETH (Windows)

- Either open a command prompt (cmd) and check if the 'git' command is part of known commands (PATH):

  ```
  git status
  ```

  or if that is not working:

- Open in the File Explorer

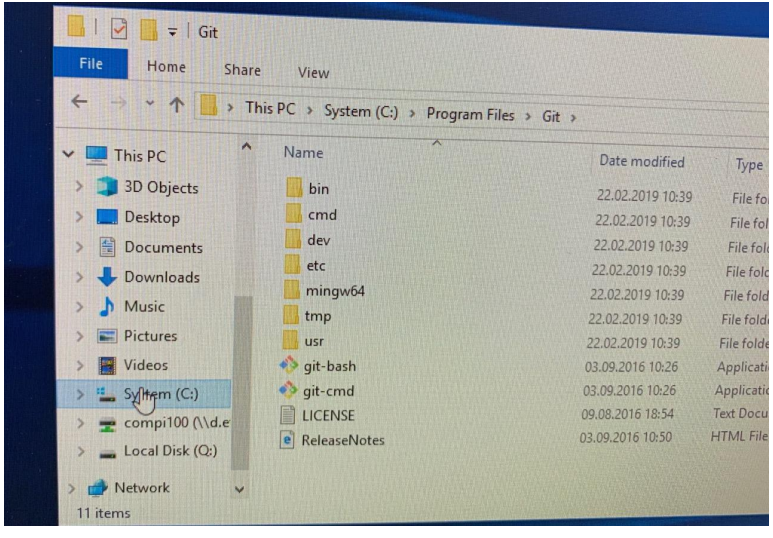
  This PC → System (C:) → Program Files → Git

  and start

  ```
  git-bash
  ```

  or

  ```
  git-cmd
  ```



Basic Git Workflow

Basic Git Workflow (I)

Tell Git to track **that folder** that houses your project files.

```
cd <path-to-project-folder>
```

Use the following command to convert your project folder into a local git repository:

```
git init
```

(Important: do **not execute** git init more than once in a project folder.)

Basic Git Workflow (II)

You can tell Git to mark certain files for saving. The git add command is responsible for this. At the end of the command, enter the path to the files you want to mark (relative to the project folder), separating each file's relative path with a space.

The command below tells git to MARK 2 files for saving (changed_file_1 and changed_file_2 in the subfolder named folder1):

```
git add changed_file_1 folder1/changed_file_2
```

You can also tell Git to **MARK ALL CHANGED/EDITED** files for saving. The code below does just that:

```
git add .
```

There are times when you want to use the git add . command to MARK ALL edited files. **Actually, most times that's what you should do.** But there may be some files and folders that have no business whatsoever in a git repository.

Basic Git Workflow (III)

If you want to exclude files, create a file on the top level with the name **“.gitignore”**, with the file names to exclude.

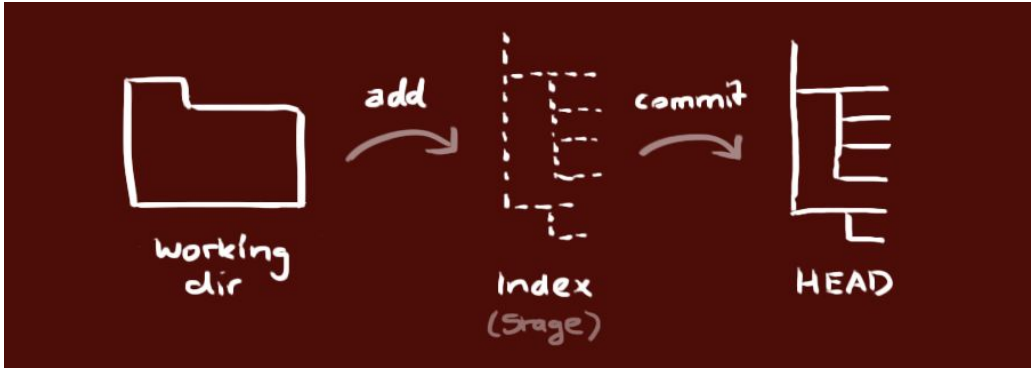
To finally save the MARKED files to the local repository, enter the following code in your terminal / command prompt:

```
git commit -m "short descriptive message"
```

Basic Git Workflow (IV)

Your local repository consists of three "trees" maintained by git.

- the first one is your `Working Directory` which holds the actual files.
- the second one is the `Index` which acts as a staging area
- and finally the HEAD which points to the last commit you've made.



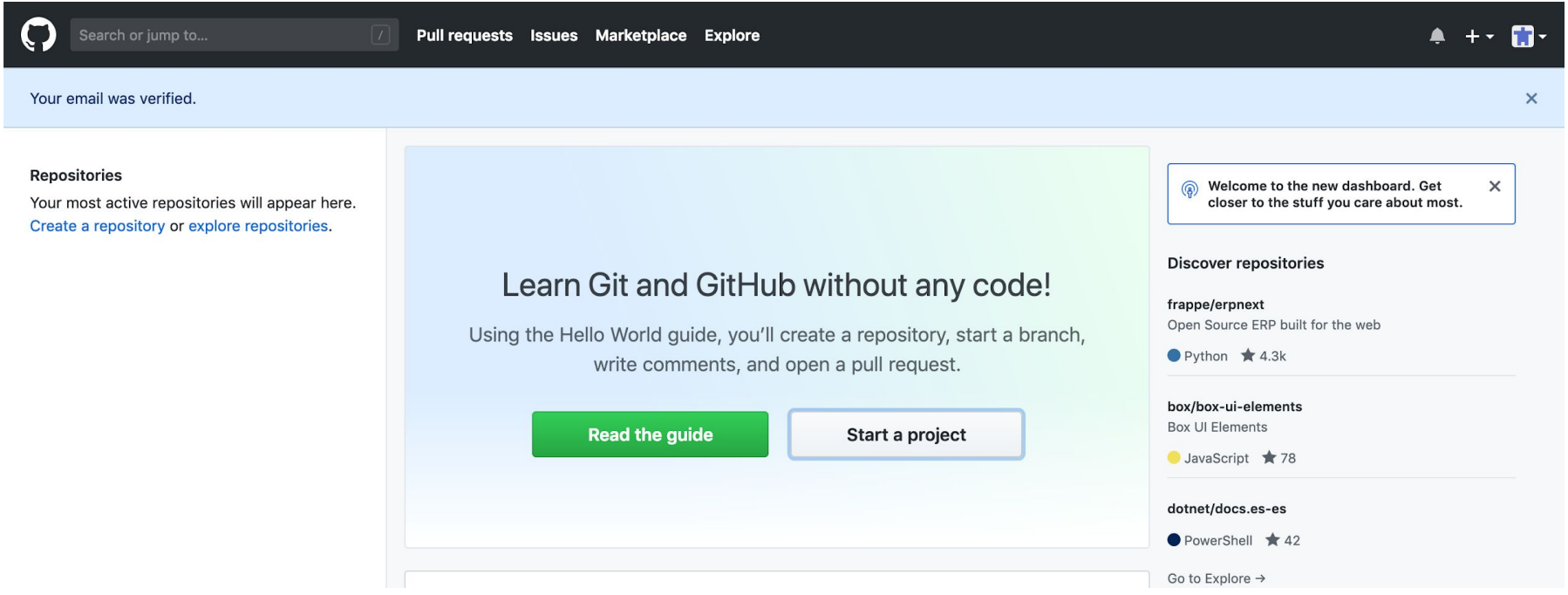
How to use GitHub (I)

How to use GitHub (I): Create an GitHub Account, create new repositories

If you haven't already a github account create one:

- Login to https://github.com/join
- Choose a not already taken username for the account and fill in your personal data in “Create your personal account”. To complete, solve the riddle.
- “Choose your plan”: Select the “free” plan
- Enter your experience in “Tailor your experience”
- Verify your Email Address in your Email Account.

Create your first “hello-world” example (I)



Create your first “hello-world” example (II)

- Open the “Read the Guide” Website … :
  https://guides.github.com/activities/hello-world/
- … AND Open the “Start a project” Website
  https://github.com/new
- follow the “hello world” example of GitHub,
  but name the repository “compicampus-git-intro” instead of “hello world”

Create your first “hello-world” example (III)

You should have

- edited your Readme.md file and enhanced the content with Text
- created a new branch
- created a pull request
- merged the branch to the master branch

Enhance your README.md with Markdown

- Open your README.md in your “compicampus-git-intro” Repository and enhance this with more Markdown

- https://guides.github.com/features/mastering-markdown/
- https://github.com/ikatyang/emoji-cheat-sheet/blob/master/README.md

- Build Documentation parts with
  - Code
  - Emojis
  - Tables
  - Usernames (ask your neighbor for his/her GitHub username)

Global Set of local installed Git

Really recommended step after installation of local installed Git:

After the installation of Git, open terminal (if you're using Linux or Mac) or command prompt (if using Windows). Please keep your terminal open, we'll be using it for a while in the next exercises.

Now let's tell git things about you—your name and email really. Git needs those info so it can confidently tie your identity to any commits you make; and then when someone comes asking, *who made those changes?*, Git can say, it was this guy or that lady.

Punch the following **TWO** instructions into your terminal/command prompt.
You would only have to do it this one time:

```
git config --global user.name "Your Name Comes Here"
git config --global user.email "you@yourdomain.example.com" (same as on GitHub)
```

How to use local Git with GitHub

Important Git Commands

- `git init` initializes a brand new Git repository and begins tracking an existing directory. It adds a hidden subfolder within the existing directory that houses the internal data structure required for version control.

- `git clone` creates a local copy of a project that already exists remotely. The clone includes all the project's files, history, and branches.

- `git add` stages a change. Git tracks changes to a developer's codebase, but it's necessary to stage and take a snapshot of the changes to include them in the project's history. This command performs staging, the first part of that two-step process. Any changes that are staged will become a part of the next snapshot and a part of the project's history. Staging and committing separately gives developers complete control over the history of their project without changing how they code and work.

Important Git Commands

- `git commit` saves the snapshot to the project history and completes the change-tracking process. In short, a commit functions like taking a photo. Anything that's been staged with git add will become a part of the snapshot with git commit.
- `git status` shows the status of changes as untracked, modified, or staged.
- `git branch` shows the branches being worked on locally.
- `git merge` merges lines of development together. This command is typically used to combine changes made on two distinct branches. For example, a developer would merge when they want to combine changes from a feature branch into the master branch for deployment.

Important Git Commands

- `git pull` updates the local line of development with updates from its remote counterpart. Developers use this command if a teammate has made commits to a branch on a remote, and they would like to reflect those changes in their local environment.

- `git push` updates the remote repository with any commits made locally to a branch.

For practical examples of these commands have a look at

https://git-scm.com/book/en/v2/Git-Basics-Recording-Changes-to-the-Repository

Git Cheat Sheet

https://services.github.com/on-demand/resources/cheatsheets/

GitHub
GIT CHEAT SHEET
V1.0.1

Git is the open source distributed version control system that facilitates GitHub activities on your laptop or desktop. This cheat sheet summarizes commonly used Git command line instructions for quick reference.

INSTALL GIT
GitHub provides desktop clients that include a graphical user interface for the most common repository actions and an automatically updating command line edition of Git for advanced scenarios.

GitHub for Windows
https://windows.github.com

GitHub for Mac
https://mac.github.com

Git distributions for Linux and POSIX systems are available on the official Git SCM web site.

Git for All Platforms
http://git-scm.com

CONFIGURE TOOLING
Configure user information for all local repositories

`$ git config --global user.name "[name]"`
Sets the name you want attached to your commit transactions
`$ git config --global user.email "[email address]"`
Sets the email you want attached to your commit transactions
`$ git config --global color.ui auto`
Enables helpful colorization of command line output

CREATE REPOSITORIES
Start a new repository or obtain one from an existing URL

`$ git init [project-name]`
Creates a new local repository with the specified name
`$ git clone [url]`
Downloads a project and its entire version history

MAKE CHANGES
Review edits and craft a commit transaction

`$ git status`
Lists all new or modified files to be committed
`$ git diff`
Shows file differences not yet staged
`$ git add [file]`
Snapshots the file in preparation for versioning
`$ git diff --staged`
Shows file differences between staging and the last file version
`$ git reset [file]`
Unstages the file, but preserve its contents
`$ git commit -m "[descriptive message]"`
Records file snapshots permanently in version history

GROUP CHANGES
Name a series of commits and combine completed efforts

`$ git branch`
Lists all local branches in the current repository
`$ git branch [branch-name]`
Creates a new branch
`$ git checkout [branch-name]`
Switches to the specified branch and updates the working directory
`$ git merge [branch]`
Combines the specified branch's history into the current branch
`$ git branch -d [branch-name]`
Deletes the specified branch

How to use local Git with GitHub (I): Cloning using HTTPS

- Create a new Repository on GitHub including your GitHub Username "compicampus-USERNAME-https" (e.g. compicampus-rplessl-https)
- New user / re-open the shell or prompt with the git command line interface:
- Clone your new GitHub Repository locally with HTTPS as documented here: https://help.github.com/en/articles/cloning-a-repository
- (You can delete the repository and checkout, we will use SSH for the next steps):
- Delete the Repository on GitHub as documented here: https://help.github.com/en/articles/deleting-a-repository
- Delete your local checkout

How to use local Git with GitHub (II): Cloning using SSH

Example: Contribute to an existing repository

```
# switch to that branch (line of development)
git checkout my-branch

# make changes, for example, edit `file1.md` and `file2.md` using the
text editor

# stage the changed files
git add file1.md file2.md

# take a snapshot of the staging area (anything that's been added)
git commit -m "my snapshot"

# push changes to github
git push --set-upstream origin my-branch
```

How to use local Git with GitHub (III): Cloning using SSH

Excercise: Contribute to an existing repository

- Create a new Repository on GitHub including your GitHub Username "compicampus-USERNAME" (e.g. compicampus-rplessl)
- Add your / a SSH Key in your GitHub Account as documented here: https://help.github.com/en/articles/adding-a-new-ssh-key-to-your-github-account
- **Contribute to an existing repository**

```
# download a repository on GitHub.com to our machine
git clone ssh://git@github.com/<USERNAME>/compicampus-USERNAME.git

# change into the `repo` directory
cd repo

# create a new branch to store any new changes
git branch my-branch
```

How to use local Git with GitHub (IV): New Content using SSH

Exercise: Start a new repository and publish it to GitHub

- **Start a new repository and publish it to GitHub**

  First, you will need to create a new repository on GitHub. You can learn how to create a new repository in our Hello World guide. Do not initialize the repository with a README, .gitignore or License. This empty repository will await your code.

```
# create a new directory, and initialize it with git-specific functions
git init my-repo

# change into the `my-repo` directory
cd my-repo

# create the first file in the project
touch README.md
```

How to use local Git with GitHub (V): Initialisation using SSH

Exercise: Start a new repository and publish it to GitHub

```
# git isn't aware of the file, stage it
git add README.md

# take a snapshot of the staging area
git commit -m "add README to initial commit"

# provide the path for the repository you created on github
git remote add origin git@github.com:YOUR-USERNAME/YOUR-REPOSITORY.git

# push changes to github
git push --set-upstream origin master
```

Interact with each other and other coders

Interact with each others and other coders

Forking: After using Git and GitHub by yourself for a while, you may find yourself wanting to contribute to someone else's project. This process is known as ***forking***.

- Follow this tutorial with the additional informations below:
  https://guides.github.com/activities/forking/

- Clone the forked repository to your local machine:
  `git clone git git@github.com:<username>/Spoon-Knife.git`

- Open the Atom Editor on the machine (or whatever your prefered Editor is)
- Change files locally
- `git commit` and `git push` them to your Repository on Github

- Create a compare and pull request as described in the Guide

Interact with each others and other coders

Pull Requests, Comments on Pull Requests, Merging:
Now we are working together on our own repositories and work together

1. Build Groups of 2
2. Ask your colleague for his github username and for a repository-name
3. Fork the Repository of your colleague on GitHub
4. Clone it to your local machine
5. Change some code and text parts in the Repository
6. Commit your changes
7. Push it to GitHub back
8. Create a pull request (https://help.github.com/en/articles/creating-a-pull-request)
9. And comment parts of the code changes in the pull request to you (https://help.github.com/en/articles/commenting-on-a-pull-request)

GitHub Pages
Host your static website

GitHub Page Website

GitHub Pages are public webpages hosted and easily published through GitHub

- GitHub Pages are free of charge static websites

- The GitHub Page Website is build with the Jekyll Framework
- Websites are themeable

- Content and Style can be modified remotely via the GitHub Website or locally in the checkout on your computer

- Website Content is given in Markdown

- A custom Website Address (Domain) can be chosen (incl. SSL Certificate from Letsencrypt) https://help.github.com/en/articles/using-a-custom-domain-with-github-pages

Create your personal GitHub Page Website

- Login and go to your GitHub Dashboard

- Create a repository called '<username>.github.io'

- Add a Repository Description
- Select the 'Jekyll' Template from the .gitignore Select-Box
- Select a License (Apache License 2.0)

- And follow this guide to create your public https://<username>.github.io Website
  https://guides.github.com/features/pages/

- Additional: If you have a unused spare DNS Domain try to setup the a custom domain website:
  https://help.github.com/en/articles/using-a-custom-domain-with-github-pages
  and activate force HTTPS afterwards

Issues Tracking and Feature Requests, Task Management and Wiki, Insights and Settings

Collaborative Features included in GitHub

GitHub includes several collaborative features for handling, enhancing and reporting of the code / of the **project**.

- Issue and Feature Request Tracker

  https://github.com/USERNAME/REPOSITORY/issues

  See Guide https://guides.github.com/features/issues

- Project Board (handling 1 or more repositories)

  https://github.com/USERNAME/REPOSITORY/projects

  See Guides: https://help.github.com/en/articles/about-project-boards and https://github.com/features/project-management/

Collaborative Features included in GitHub

- Package and Release Code

  https://github.com/features#code-hosting

- All Features in one big page

  https://github.com/features#code-hosting

Further Reading and Exercises

Further Exercises and Reading

Book Recommendation:

- The complete “Pro Git” book is available at: https://git-scm.com/book/en/v2/

Especially read and test the stuff written in the branching chapter:

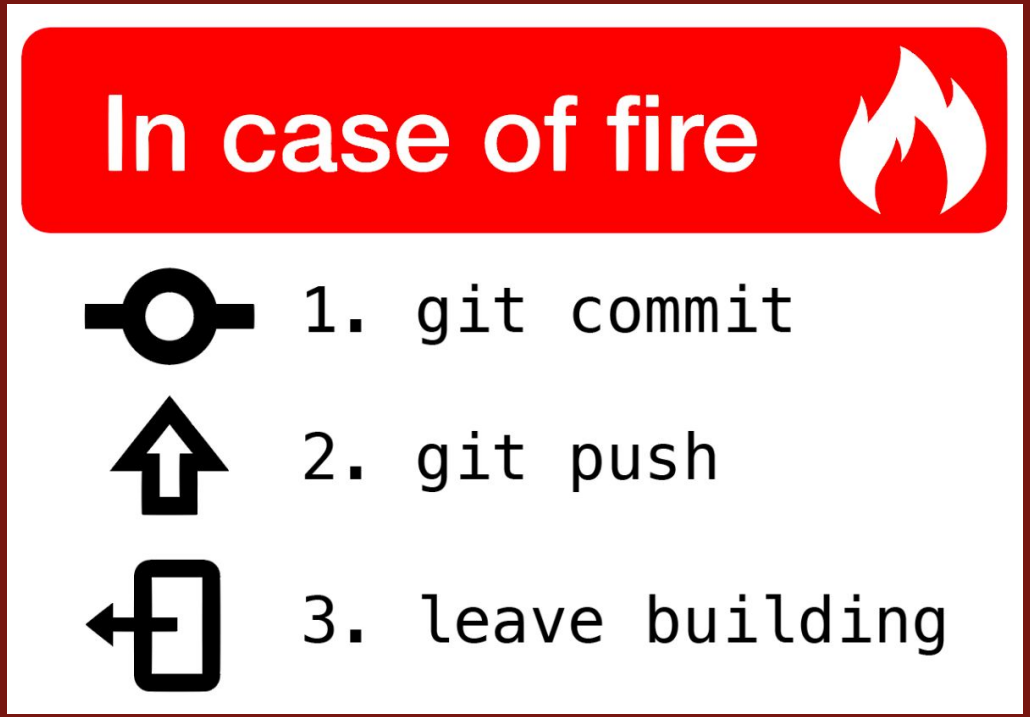
- https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell

Further Exercises and Reading

Further Exercises:

- Nearly always - after forking a repository, a re-synchronisation is necessary (https://help.github.com/en/articles/syncing-a-fork)

- Also nearly always - resolving merge conflicts (https://help.github.com/en/articles/resolving-a-merge-conflict-using-the-command-line) (https://help.github.com/en/articles/resolving-a-merge-conflict-on-github)


In case of fire
1. git commit
2. git push
3. leave building

Thank you for being part of this class!

Roman Plessl

rplessl@prunux.ch

https://github.com/rplessl