## Git

Git is a version control  system that helps managing files within projects. It keeps track of the entire history of things that you are working on. It tracks all changes in the projects and allows us to revert back to a previous version. It also allows tp work with several people on the same project, without disturbing each other. Team members can work on different features and easily merge the changes.

## Github

GitHub is a Web Service for version control using Git.

## Basic Terminology

**Working Directory:** The directory you are working in and in which you add, modify and remove files. The changes need to be staged and commited to Git in order to track the project history.

**Git Repository**: A Git Repository is a folder that you've told Git to help you track file changes.

**Remote Repository**: a copy of your local Git repo that is saved on a server.

**Branch**: A branch is an independent line of development (like a new copy of the repo in which you can experiment without any risk)

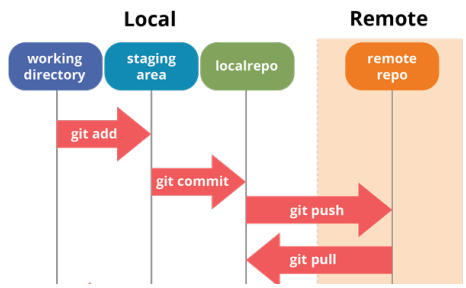
**Master**: The default development branch. Whenever you create a git repo, a branch named "master" is created which becomes the default active branch.

**Fork**: Copy the repo of another user to your account. This is done in order to contribute to a project of another user (to which you would not have writing permission).

**Clone**: Download a repository from GitHub

**Commit**: save changes permanently on local Git repository

**Gitignore**: a .gitignore file specifies all folders/files that should not be staged and commited




## Installation on Ubuntu

```
sudo apt-get update
```

Update the package lists from a server (needed inorder to install the latest version of git)

```
sudo apt-get install git
```

Install git

## Setup and Configuration

```
which git
```

Get location of git

```
git --version
```

Get the version of Git

```
git config --global user.name "[github-username]"
```

Assign username to your commits

```
git config --global user.email "[github-mail]"
```

Assign email to your commits

## Create new Repository / Download Repository

First create a Github Repository (with Readme file) and copy the url.

```
git clone [url]
```

Clone (download a repository from GitHub)

## Create Repository from existing Folder

First create an empty Github Repository (without Readme file) and copy the url.

```
git init
```

Turn current directory into a local git repository.

```
git remote add origin [url]
```

Connect local repository with remote repository (github repository). By default the remote repo is called origin.

```
git add .
```

Move files from the working directory to the staging are. Files within the staging area will be commited to the local git repo with the next commit.

```
git commit -m "initial commit"
```

Commit the files that are staged to your local git repo (-m for adding a comment). This adds the staged files to the project history

```
git push origin master
```

Push the code to the master branch of the remote repository (called origin). This saves the changes on the GitHub server.

## Branches

Branches can be used to isolate work without affecting other branches. In the new branch you can work in isolation from changes that other people are making to the repository. Branches are used to develop features, fix bugs and to safely experiment with new ideas.

```
git branch
```

List all existing branches (a " * "  will appear next to the currently active branch)

```
git branch [branch_name]
```

Create a new branch (eg. in order to work on a new feature)

```
git checkout [branch_name]
```

Switch to the specified branch.

## Delete failed Branches

```
git branch -D [branch_name]
```

To delete the local copy of your branch

```
git push origin –delete [branch_name]
```

To delete the remote branch

## Merge Branches

Before merging make sure that both branches are up-to-data with the latest remote commits.

```
git checkout [branch_name]
```

Switch to the branch that should be updated (receiving branch)

```
git fetch origin
```

downloads latest changes from the remote repository named "origin", but does not integrate any of this new data into your working files. It is a very safe way to get a view on the things that happened on the remote repo.

```
git pull origin master
```

Updates the current branch with the latest changes from the remote server. Thus, pull downloads and integrates data from the remote into your local repo. You should only pull if you do not have any uncomitted local changes.

```
git merge [branch_name]
```

Merges (combines) the specified branch with the current branch (the current branch is the one that gets updated). Brings two histories back together again.

```
git merge --abort
```

Stop merging. To return to the state before you started the merge

```
git reset --hard
```

Undo a completed merge (eg. in case you have made a mistake while resolving a conflict).

```
git branch -d [branch_name]
```

Delete the specified branch after successful merging. Only works if the branch has been merged to the root branch (otherwise you get an error message)

## Merge Conflicts

If the two branches you are trying to merge both changed the same part of the same file, Git won't be able to figure out which version to use. The merge will stop right before the merge commit so that you can resolve the conflicts manually.

```
git status
```

Shows which files need to be resolved

```
git merge --abort
```

Stop merging. To return to the state before you started the merge

```
Some content not affected by the conflicts
<<<<<< master
this is conflicted text from master
======
this is conflicted text from feature branch
```

The affected file contains visual indicators that mark both of the conflicted content. The content before the ===== marker is the receifvng branch and the part after this is the merging branch (the branch in which you developed a new feature). Resolve the conflict by making the necessary changes.

```
git add [conflicted_file]
git commit [conflicted_file]
```

Git add tells git that the conflict has been resolved and git commit generate the merge commit.

## Save changes temporarily

You can temporarly get rid of changes (without losing them) and continue working on them later. This is for example useful if an important bug report comes in which you should fix. Stashing can be useful before checking out a different branch, before pulling remote changes or before merging a branch.

```
git stash
```

Puts away the local changes so that you have a clean working copy. Discard all local changes, but saves them for later

```
git stash list
```

Get an overview of your current stashes.

```
git stash pop
```

This applies the newest stash and removes it from your clipboard

```
git stash apply [stash_name]
```

Will apply the specified stash, but it will remain saved

```
git stash drop [stash_name]
```

Delete the specified stash.

## Handling Files

Newly created files first need to be added to the staging area before they are commited to the local git repo. The files that are in the staging will be commited to the local git repo with the next commit. Files that have been commited can be pushed to the remote repo (github).

```
git branch [branch_name]
```

Create a new branch

```
git status
```

Show modified files in your working directory that are staged for your next commit. Also shows if there are any merge conflicts (unmerged paths).

```
git add
```

Adds changes in the working directory to the staging area.

```
git reset [file_name]
```

Unstage a file while retaing the changes in the working directory

```
git rm [file_name]
git commit -m "remove file_name"
git push origin master
```

Delete the file from the Git repository **AND** from local file system and push the changes to the remote repo

```
git rm --cached [file_name]
git commit -m "remove file_name"
git push origin master
```

Remove file from the Git repo and **NOT** delete it from the local file system

```
git commit -m [message]
```

Commit staged files to the local git repository

```
git log --oneline
```

Shows all the commits in one line each

```
git pull
```

Pull remote repo

```
git push origin master
```

Push changes to the commited files

```
git pull
```

Pull remote repo