



# Working with Git and Github

## - Cheatsheet I - Local Repositories -



## Git in a nutshell

Git is a Source Code Management (SCM) system. It lays the foundation for code trust by providing a secure way to review and monitor code changes. Each project will be represented by a **repository**. They are represented by a hidden folder in your project named **.git**. Git uses repositories to keep track of all changes made to files in the project's folder. Each change is called a **commit**. Collaboration is supported by online Git services, such as Github or Gitlab. Over these, many developers can independently work on issues, features and bug fixes.

## Overview of a local Git development workflow



1) **Create** a new repository with '**git init**'



2) Add new files and **create an initial commit** with '**git add <filename>**' and '**git commit -m "<message>"**'



3) **Create a new branch** with '**git branch <branchname>**' and change to it with '**git checkout <branchname>**'



4) Make changes, **stage** with '**git stage <filename>**' and **commit** them with '**git commit -m "<message>"**'



5) **Merge** the changes into the initial branch with '**git checkout master**' and '**git merge <branchname>**'



6) **Check** what changes were made with '**git log**'. You can also check a single file's history using '**git blame <filename>**'



See local changes: '**git diff [<filename>]**'



Revert local changes: '**git checkout -- .**'



Revert a commit '**git revert <commit-hash>**' (get commit # with '**git log**')



Remove a tracked file: '**git rm --cached <filename>**' ('**git reset**' if not committed yet)



Delete a local branch: '**git branch -d <branchname>**'



Stash changes and re-apply \*: '**git stash**' + '**git stash pop**'

\* Stashing means: Temporarily store away changes made without committing them. This is useful when pulling remote changes that conflict with local, uncommitted changes