

GIT CHEATSHEET

Concepts (see also: *man gitglossary*)

- Repository** = A database that can be used by git to produce a checkout of any point in the project timeline.
- Checkout** = The action of updating your filesystem to reflect a particular commit in the repository.
- Commit** = A node in the graph, representing a snapshot in history. Similar to revisions in SVN, but instead of being specific to particular files, they span the whole repository.
- Branch** = A pointer to a particular commit. The current branch is "pulled along" with new commits, whereas other branches behave like tags.
- Tag** = A fixed pointer to a particular commit, used to mark versions or other important points in time.
- master** = The name of the default development branch
- origin** = The name of the default upstream repository
- HEAD** = A pointer to the currently checked out commit

Common commands

Setting it all up:

- `$ git config --global user.name "Max Mustermann"`
- `$ git config --global user.email "maxine@musterfrau.de"`

Getting started

- Starting from scratch:
 - `$ cd project/`
 - `$ git init`
- Cloning another repository
 - `$ git clone URL`

Typical workflow

1. Add, remove or change some files
2. Stage the changes. You can decide which changes will be included in the next commit by putting them into the so-called **staging area** with
 - `$ git add <FILE>`
 - `$ git rm <FILE>`
 - `$ git add --interactive` (or `-i` for short)You can reset the staging area with:
 - `$ git reset HEAD`
3. Commit the staged changes
 - `$ git commit`You are asked to input a commit message which should ideally follow these formatting rules:

Structure:

Example:

Short summary <empty line> Optional explanation

Deleted file X The file was not needed anymore, it used to be part of an obsolete feature Y
--

4. Optionally, push the changes to a remote server where your collaborators can see and review your changes.
 - `$ git push <remote> <branch>`Often "master" is the relevant branch and "origin" is the remote repository you work on, so this will often suffice:
 - `$ git push origin master`

Display information

- "git status" is the main command to view the status of the files in your repository
- "git log" displays the list of commits in the current branch. The "--graph" option adds a visualization of branches and the "--all" option includes other, diverged branches.
- There are advanced tools like "gitk" and "qgit" (with a graphical user interface) and "tig" (for the console) that provide tools to view the history and find information.

Branching workflow

1. `$ git branch featureX` (creates a new branch)
2. `$ git checkout featureX` (checkouts that branch)
3. implement the feature
4. `$ git add --interactive; git commit` (create the commits)
5. `$ git checkout master` (go back to master)
6. `$ git merge featureX` (merge featureX into master)
7. `$ git branch -d featureX` (delete branch featureX)

If you decide to stop any time, just commit your changes and checkout master. To continue working on featureX, checkout back into the featureX branch.


Remotes

A "remote" describes a place that contains a git repository, from which you can clone, pull, fetch and push.

Remotes can be:

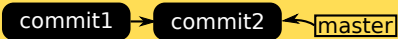
- A local directory, like `/home/me/repos/myproject`
- A SSH path: `ssh://user@host/home/.../myproject`
- GitHub (pull-only): `https://github.com/user/project.git`
- GitHub (push+pull): `git@github.com:user/project.git`
- Welfenlab-GitLab: `git@git.gdv.uni-hannover.de:user/project.git`

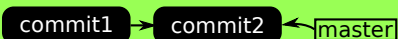
A brief example

origin:  note: whenever "origin/master" is left out, it points to the same commit as "master"!

Local: nothing.

`$ git clone <URL>`

origin: 

Local: 

`do changes → $ git add [...] → $ git commit`


origin: 


Local: 

You decide to push the changes with "git push origin master", but your commit is **rejected** because a collaborator already changed something else! To fix this,


1. fetch the remote branch
2. merge your branch with the remote branch
3. deal with possible merge conflicts
4. try pushing again.

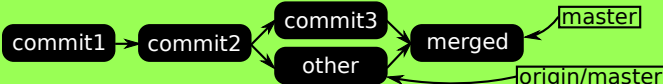
`$ git fetch origin master`

origin: 

Local: 

`$ git merge origin/master`

origin: 

Local: 

`$ git push origin master`

origin: 

Local: 

Alternatively: `$ git rebase origin/master → $ git push origin/master`

Both: 

WHEN SOMETHING GOES WRONG

- When you lose a commit (e.g. by deleting a branch), try "**git reflog**", a list of all recent actions, or "**git fsck --unreachable**". You can "**git checkout <hash>**" or "**git show <hash>**" to access the lost commits.
- To reset your local branch to the state of the remote branch, run "**git reset --hard <remote>/<branch>**".
- If you need to modify the commits in your history, there are a few solutions. However, it is **strongly** advised to avoid rewriting the history in a public remote repository - it will cause problems for your collaborators if they already pulled your old changes!
 - "**git commit --amend**" lets you edit the last commit. Combine this with "git add" or "git rm" to change the content of the last commit.
 - "**git rebase --interactive <hash>**" lets you cherry-pick, reword or squash individual commits.

BETTER SAFE THAN SORRY

Before you do something you don't quite understand, especially if the command begins with "git reset" or "git rebase", create a branch at your current state with "git branch <name>". If anything fails, you can always "git checkout <name>" to restore the old state.