Git Cheat Sheet

Local

Working directory | Staging | Remote branch tracking | Remote

git merge ← git fetch

git pull

git restore

Local Repository

git add → git commit → git push

Stash

git stash

git stash apply

Start a project

Create a local repo (omit <directory> to initialise the current directory as a git repo
$ git init <directory>

Download a remote repo
$ git clone <url>

Review your repo

List of modified files
$ git status

List of commit history
$ git log --oneline

Show changes to file
$ git diff

workdir | Staging | Local Repo

git diff

git diff --cached

git diff HEAD

Setup

Set the name and email that will be attached to your commits and tags

$ git config --global user.name "John Doe"
$ git config --global user.email "me@univ.ac"

Make a change

Add a file to staging
$ git add <file>

Stage all files
$ git add .

Choose interactively which line to stage in each modified file
$ git add -p

Commit all staged files
$ git commit -m "message"

Staged all tracked files and commit
$ git commit -am "message"

Basic Concepts

master: default development branch (can also be main)
origin: default upstream repo
HEAD: current branch
HEAD^: parent of HEAD
HEAD~2: grandparent of HEAD

Undoing mistake

Revert your file in working directory to one in staging area
$ git restore <file>

Revert a git add (un-stage) a file
$ git restore --staged <file>

Restore a file from an old commit/other branch/tag
$ git restore --source=<ID> <file>

Create a new commit reverting the change of an old commit
$ git revert <commit_ID>

Go back to a previous commit and delete all commits ahead of it. Does not change working directory (does it with --hard: careful no backup of local change)
$ git reset <commit_ID>

Recover recently lost commit
$ git reflog
$ git switch -c <name> <commit_ID>

Searching for bug

Start the search procedure
$ git bisect init

Set the current commit as "bad"
$ git bisect bad

Set the last known working commit as good
$ git bisect good <tag|commit>

Then check the propose commit and then run one of the command
$ git bisect good
$ git bisect bad

When finish/or to stop the search
$ git bisect reset

Branches

List all local branches. Add -r to show all remotes branches, -a for all branches
$ git branch
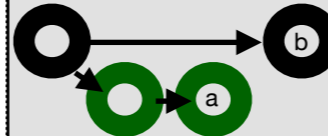
create a new branch and switch to it
$ git switch -c <new-branch>

Switch to a new branch (update the working directory)
$ git switch <branch>

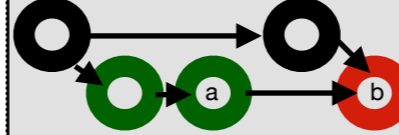Delete a merged branch (use -D if the branch is not merged)
$ git branch -d <branch>

Merging

Merge branch a into branch b
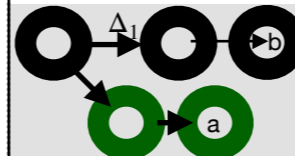$ git switch <b>
$ git merge <a>

Before:

After:

Merge and squash all commit to a single one
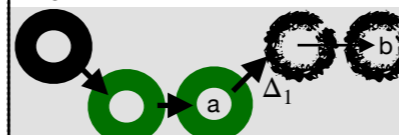$ git merge --squash <a>

Rebasing

Alternative to merge to include **independent** changes into another branch (keep local only)
$ git switch <b>
$ git rebase <a>

Before:

After:

Stash area

Store modified changes away (reset working directory and staging area to last commit). With or without comment
$ git stash
$ git stash save "comment"

Partial stash (interactive mode)
$ git stash -p

Re-apply the stash without deleting it
$ git stash apply

Re-apply the stash and delete it
$ git stash pop

List all stash
$ git stash list

clear all stash
$ git stash clear

Remote's management

Add a remote repo
$ git remote add <alias> <url>

View all remote
$ git remote -v

Remove a remote connection
$ git remote remove <alias>

Change url of remote
$ git remote set-url <alias> <url>

Synchronizing

Update (all) local status of remote branch
$ git fetch <alias>

Fetch the remote of the current branch and then merge
$ git pull

Mover your local change onto the top of the new changes coming from the remote repo
$ git pull --rebase

Basic Objects

Commit: a state of the code
Branch: *line of development*
TAG: symbolic names for a given *commit*.

Publish to GitHub

Upload local content to remote repo
$ git push

Push a new branch on a remote and setup git to follow the remote
$ git push --set-upstream <alias> <branchname>

Typical workflow

After debugging code. Select interactively what to keep and discard
$ git add -p

Stash all the rest (-k forbid to stash the staging area)
$ git stash -k

Run check that everything is fine (if not restore the stash) and commit the change
$ git commit -m "Message"

Publish the change on github/lab
$ git push

Clean the last entry of the stash
$ git stash drop

Tag

Tag current commit with a human readable name
$ git tag <tagname>

List all tags
$ git tag

Remove a tag
$ git tag -d <tagname>

Set worker as state of the tag
$ git switch --detach <tagname>

Start a new branch from a tag
$ git switch -c <newname> <tagname>

Working in wrong branch

This works with both change in the working directory and staging Area
$ git stash
$ git switch <correct branch>
$ git stash pop