Cheat sheet: Git

Vocabulary

Repository (local) Name given to the folder containing your project managed by Git.

Remote repository Distant repository accessible to every collaborators. You can put and retrieve states of the project to and from this remote.

Commit State of the project.

SHA1 Unique identifier/version number given to a commit. Example: *5500bd8e09749*[...]*cde9c34001650b*

Push The action of propagating your local commits to the remote. Thus, making them accessible to your collaborators.

HEAD The name given to the current revision of your project.

Working copy/workspace The current state of the files on your hard drive.

Tags Tags allow you to name a state of your project with a human-readable name. They are used to mark important states of a project, like releases.

Staged Files in the staging area are waiting to be commited. Staging is a step before the commit process in Git.

Tracked Tracked files are files that Git knows about. They can be modified, staged or unmodified. We can also talk about untracked files which are everything else.

Commands

Git configuration

`git config --global user.name "Your Name"` Set the name that will be attached to your commits and tags.

`git config --global user.email "Your Email"` Set the email that will be attached to your commits and tags.

Start a Project

`git help` Display a *non-exhaustive* list of the available git commands.

`git init <dir>` Create an empty repository or reinitialize an existing one.

`git clone <remote_url>` Copy the current state of the remote into a new repository on your computer.

Make changes

`git add <file>` Register changes made to a file for the next commit. Git tracks files that have been *added* to a commit.

`git rm <file>` Remove a file from the repository. Git will no longer track this file.

`git mv <source> <dest>` Move files inside the repository. Git sees this move as a rename.

`git commit` Create a new state of the project containing the changes and files that have previously been *added*.

Work with a clean repository

`.gitignore` A `gitignore` file specifies untracked files that Git should ignore. Does **not** affect files already tracked by Git.

`git clean <path>` Clean the working tree by recursively removing files that are not under version control.

Collaboration

`git push` Propagate commits stored locally to the remote.

`git fetch` Update the local repository.

`git pull` Progagate the remote version of the project onto your local repository. Retrieve the changes made by your collaborators.

`git tag -a <tag> -m "tag decription"` Mark the last commit with the specified tag.

`git push --tags` Propagate tags to the remote.

Commit and work review

`git status` Display the state of the files in your workspace.

`git log` A tool that displays the commit history of the project. Some useful options for `git log`: *oneline*, *decorate*, *stat*.

`git diff` Show changes between working directory and staging area.

`git diff <file>` Allow you to see the modifications made on a file since the last commit.

`git diff <commitId> <commitId>` Show changes between two commits.

`git shortlog` Summarize git log output by grouping commit by author and title.

`git grep` Print lines matching a pattern.

`git show <SHA1>` Show any object in Git in human-readable format.

`tig` A powerful tool that allows you to see your commit history and the changes made in each commits.

Using save states

`git checkout <file>` Allow to reset the state of the specified file to its last committed state.

`git checkout <SHA1>` Set your working directory to the state of the specified commit.

`git checkout <SHA1> <files>` Set the files to the state of the specified commit.

Temporary commits

`git stash` Temporarily store your modifications so you can work on something else and then re-appply them later on. It is really handy if you need to quickly switch context.

`git stash list` List stashed file changes.

`git stash pop` Retrieve the commits from the stash stack.

`git stash drop` Remove the commits from the stash stack.

Branch & Merge

`git branch` List all the branches.

`git branch <branch>` Create a new branch.

`git merge <branch>` Merge a branch into the current one.

Work with remote

`git remote add <alias> <url>` Create a remote with an alias from a Git url.

`git merge <alias>/<branch>` Merge a remote branch into the current one.

`git push <alias> <branch>` Propagate any commits from the tracking remote branch to the local one.

Frequent manipulation

Intranet submission

```
git add <files>

git commit -m <message>

git tag -a <tag> -m <message>

git push --follow-tags
```

Merge a branch into master

```
git checkout master

git merge <branch>
```