Git cheat sheet

Git installation

For GNU/Linux distributions, Git should be available in the standard system repository. For example, in Debian/ Ubuntu please type

```
$ sudo apt install git
$ sudo apt install bash-completion
```

Git config

```
$ git config --global user.name "NAME"
$ git config --global user.email "EMAIL-ADDRESS"
```

specifies the name and email address that should be linked to your commits and tags.

```
$ git config --global color.ui auto
```

allows you to set the colouring of the Git output.

.gitignore

Some files should not normally be tracked by Git. They are written to a special file called .gitignore. Helpful templates can be generated on the website gitignore.io.

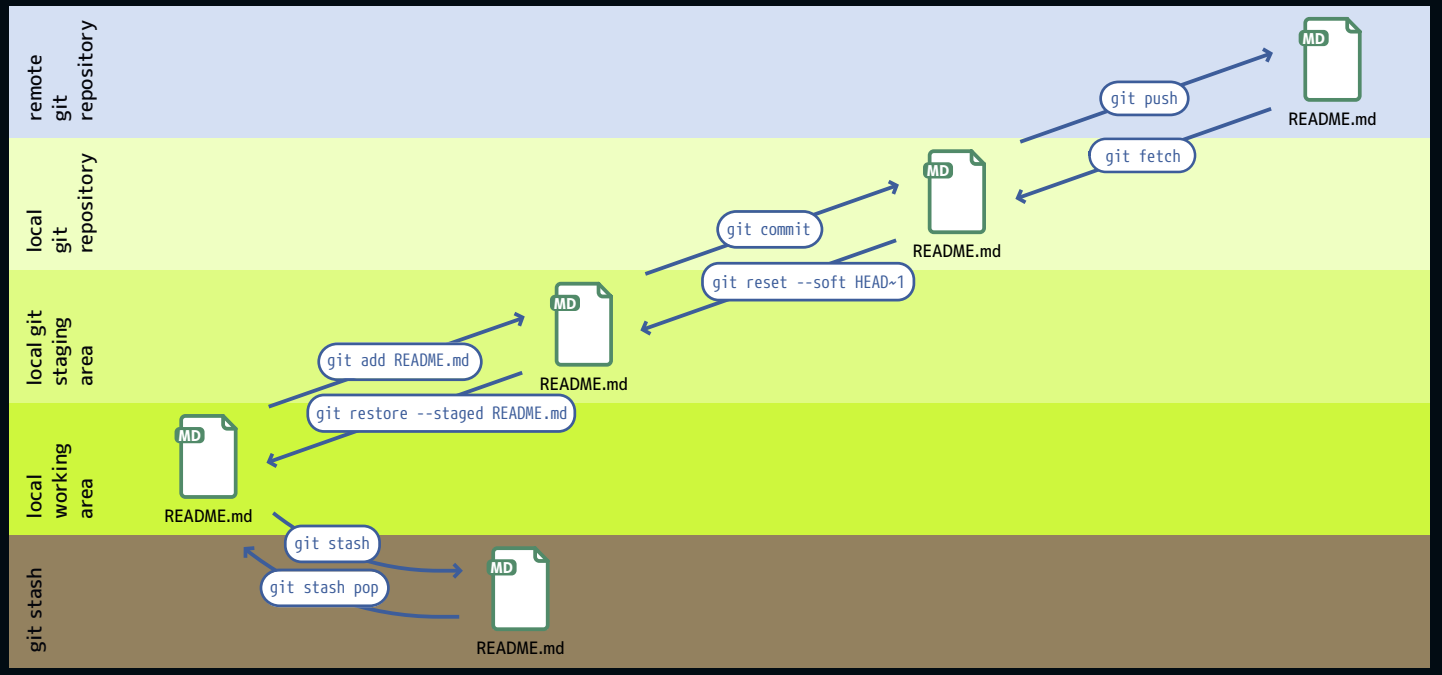
Start a project

```
$ git init [PROJECT]
```

creates a new local repository. If *PROJECT* is given, Git creates a new directory and initializes it. If it is not specified, the current directory is initialised.

```
$ git clone PROJECT_URL
```

downloads a project with all branches and the entire history from the remote repository.



Make a commit

```
$ git add PATH
```

adds one or more files to the stage area.

```
$ git add -p PATH
```

adds parts of one or more files to the stage area.

```
$ git rm PATH
```

removes a file from the work and stage areas.

```
$ git commit -m 'COMMIT MESSAGE'
```

writes a commit message directly in the command line.

Show changes

```
$ git status
```

shows the status of the working directory with new, provided and changed files for the current branch.

```
$ git diff [PATH]
```

shows differences between work and stage areas.

```
$ git diff --staged [PATH]
```

shows differences between the stage area and the repository.

```
$ git show COMMIT_SHA
```

shows the difference between a commit and its parent.

Discard your changes

```
$ git clean
```

deletes untracked files.

```
$ git restore
```

changes files in the working directory to a state that was previously known to Git.

```
$ git restore --staged PATH/TO/FILE
```

undoes the addition of files.

```
$ git revert COMMIT_SHA
```

creates a new commit and reverts the changes of the specified commit.

Stashes

`$ git stash`

moves the current changes from the workspace to a stash.

`$ git stash list`

lists the various stashes.

`$ git stash show`

shows the changes in the stashed files.

`$ git stash pop`

transfers the changes from the stash to the workspace and empties the stash.

`$ git stash drop`

empties a specific stash.

Branches

`$ git branch`

shows all local branches in a repository.

`$ git branch -a`

shows also the remote branches.

`$ git switch BRANCH_NAME`

switches between branches.

`$ git switch -c BRANCH_NAME`

creates the branch to switch to.

`$ git merge [FROM_BRANCH_NAME]`

connects the specified branch with the branch you are currently in.

`$ git merge –squash [FROM_BRANCH_NAME]`

combines the commits in a single commit.

`$ git branch -d [BRANCH_NAME]`

deletes the selected branch if it has already been transferred to another.

`-D` instead of `-d` forcing the deletion.

Review

`git log [-n COUNT]`

lists the commit history of the current branch.
-n limits the number of commits to the specified number.

`$ git log --oneline --decorate --graph --all`

display the history graph for all branches, one commit per line.

`$ git log MAIN..FEATURE`

shows changes in FEATURE that are not contained in MAIN.

`$ git log FEATURE..MAIN`

shows changes in MAIN that are not contained in FEATURE.

`$ git reflog`

displays the reference log, a record of all commits made.

Tagging

`$ git tag`

lists the tags of your repo.

`$ git tag TAGNAME`

creates a tag for the current commit.

`$ git tag -d TAGNAME`

deletes a tag.

Change the history

`$ git reset HEAD~`

moves the current branch back by one commit.

`$ git reset --keep '@{u}‘`

undoes all local changes to a branch.

`$ git reset --keep main`

undoes all changes in the current branch.

`$ git reset --soft $(git merge-base @ main)`

undoes all commits in the current branch.

`$ git filter-repo --invert-paths --path PATH/SOMEFILE`

removes a file from the history.

Synchronising repositories

`$ git remote add origin REMOTE_URL`

links to a remote repository.

`$ git fetch`

fetches the changes from the remote repository, but don’t change any of your local branches.

`$ git pull`

fetches the changes from the remote repository and merges the current branch with the upstream branch.

`$ git push [--tags]`

transfers local changes to the remote repository.
Use --tags to transfer tags.

`$ git push -u origin [BRANCH]`

pushes the local branch to the remote repository.

`$ git push --force-with-lease`

protects all remote branches if they do not match the locally existing remote tracking branches.

Text and design by cusy GmbH

https://cusy.io/en/seminars
https://cusy.io/de/seminare

>cusy_