Hochschulgruppe für Freie Software und Freies Wissen
`https://fsfw-dresden.de`

FSFW Git Workshop Cheat Sheet

Note: Parts of commands in [square brackets] are optional.

Configuring git

```
$ git config [--global] [option]
```
with `--global` it is stored in `~/.gitconfig`

Information about the user
```
$ git config user.name NAME
$ git config user.email EMAIL@HOST.TLD
```

Enabling coloured output
```
$ git config color.ui auto
```

Improve interactive user-experience
```
$ git config interactive.singlekey true
```

Creating or cloning repositories

From existing data
```
$ cd  my_project_dir
$ git init
$ git add .
$ git commit -m 'initial commit'
```

From existing repo
```
$ git clone path/to/existing/repo path/to/new/repo
$ git clone you@host.de:dir/project.git
$ git clone http://[USER@]host.de/project.git
```

Getting information

Files changed in working directory
```
$ git status
```

Changes of tracked files
```
$ git diff
```

Changes between ID1 and ID2
```
$ git diff <ID1> <ID2>
```

History of changes
```
$ git log
$ gitk
$ gitk --all
```

Working with the Index and committing changes

Add all changes in a file or directory to the index
```
$ git add path/to/file_or_dir
$ git add .
```

Interactively select changes for addition
```
$ git add -p [path/to/preselect_some_files]
```

Commit changes added to the index
```
$ git commit
```

Add and commit all local changes
```
$ git commit -a
```

Commit changes without editing the message
```
$ git commit -m "<message>"
```

Working with Branches

List all branches
```
$ git branch
$ git branch -a
```
list remote branches as well

Switch to a branch
```
$ git checkout <branch>
```

Merge Branch B1 into B2
```
$ git checkout <B2>
$ git merge <B1>
```
or
```
$ git merge --no-ff <B1>
```
generates commit even if fast-forwarding is possible

Create Branch based on HEAD and checkout
```
$ git checkout -b <branch>
```

Delete a branch
```
$ git branch -d <branch>
$ git branch -D <branch>
```
Delete a branch that is not merged to the default branch

Get changes from upstream

Fetch changes from a remote
```
$ git fetch [remote]
```
`remote` defaults to `origin`

Get changes
```
$ git pull [remote] [refspec]
```
`remote` defaults to `origin`

Publishing changes

Push changes to a remote
```
$ git push [origin] [branch]
$ git push
```

Create tags
```
$ git tag [-s] <tag name>
```
with `-s` the tag is signed with GPG

Prepare a patch
```
$ git format-patch origin
```

Reverting changes

Return to last committed state
```
$ git reset --hard
```

Revert specific commit
```
$ git revert <ID>
```
this is safe for published commits

fix/change last commit
```
$ git commit --amend
```
never do this on a published commit
unless you know what you are doing

Misc

Documentation/help
```
$ git help [command]
$ man git-<command>
```

Delete branch (locally and remote)
```
$ git branch -d <branch>
$ git push <origin> :<branch>
```

Beautify non-pushed history interactively
```
$ git rebase -i <ID>
```