git cheat sheet

Configure tooling

Set user name attached to your commits
```
$ git config --global user.name "[name]"
```

Set email address attached to your commits
```
$ git config --global user.email "[email]"
```

Enable helpful colourisation of command line output
```
$ git config --global color.ui auto
```

Create repositories

Create a new local repository
```
$ git init [project-name]
```

Download a project and its entire version history
```
$ git clone [ssh://user@domain.com/repo.git]
```

Make changes

List all new or modified files to be committed
```
$ git status
```

Show file differences not yet staged
```
$ git diff
```

Snapshot the file in preparation for versioning
```
$ git add [file]
```

Add all changes to the staging area
```
$ git add .
```

Show file differences between staging and the last file version
```
$ git diff --staged
```

Record file snapshots permanently in version history
```
$ git commit -m"[descriptive message]"
```

Branches & Tags

List all local branches in the current repository
```
$ git branch -av
```

Create a new branch
```
$ git branch [branch-name]
```

Switch to the specified branch and updates working directory
```
$ git checkout [branch-name]
```

Delete the specified branch
```
$ git branch -d [branch-name]
```

Create new tracking branch based on a remote branch
```
$ git checkout --track <remote/branch>
```

Mark the current commit with a tag
```
$ git tag
```

Merge & Rebase

Merge <branch> into your current HEAD
```
$ git merge <branch>
```

Rebase your current HEAD onto <branch>
Don't rebase published commits!
```
$ git rebase <branch>
```

Abort a rebase
```
$ git rebase --abort
```

Continue a rebase after resolving conflicts
```
$ git rebase --continue
```

Use your configured merge tool to solve conflicts
```
$ git mergetool
```

Use your editor to manually solve conflicts and (after resolving) mark file as resolved
```
$ git add <resolved-file>
$ git rm <resolved-file>
```

Refactor file names

Deletes the file from the working directory and stages the deletion
```
$ git rm [file]
```

Removes the file from version control but preserves the file locally
```
$ git rm --cached [file]
```

Changes the file name and prepare it for commit
```
$ git mv [file-original] [file-renamed]
```

Save fragments

Temporarily store all modified tracked files
```
$ git stash
```

Restore the most recently stashed files
```
$ git stash pop
```

List all stashed change sets
```
$ git stash list
```

Discard the most recently stashed change set
```
$ git stash drop
```

git cheat sheet

Review history

List version history for the current branch

```
$ git log
```

List version history for the file, including renames

```
$ git log --follow [file]
```

Show content differences between two branches

```
$ git diff [first-branch] [second-branch]
```

Show what changed between commits ID1 and ID2

```
$ git diff [ID1] [ID2]
```

Show who changed what and when in a file

```
$ git blame [file]
```

Output metadata and content changes of the specified commit

```
$ git show [commit]
```

Redo/Undo Commits

Reset your HEAD pointer to a previous commit …and discard all changes since then (cannot be undone!!!)

```
$ git reset --hard <commit>
```

…and preserve all changes as unstaged changes

```
$ git reset <commit>
```

…and preserve uncommitted local changes

```
$ git reset --keep <commit>
```

Fix the last commit

```
$ git commit -a --amend
```

Un-stage the file, but preserves its contents

```
$ git reset [file]
```

Return back to the last commit (cannot be undone!!!)

```
$ git reset --hard
```

Discard local changes in a specific file

```
$ git checkout HEAD <file>
```

Revert a commit (by producing a new commit with contrary changes)

```
$ git revert <commit>
```

Working with remote

List all currently configured remotes

```
$ git remote -v
```

Show information about a remote

```
$ git remote show
```

Add new remote repository

```
$ git remote add [my-remote-repo]
```

Download all history from the remote repository, but don't integrate into HEAD

```
$ git fetch [remote]
```

Download bookmark history and directly merge/integrate into HEAD

```
$ git pull [remote]
```

Combine the remote branch into the current local branch

```
$ git merge [remote]/[branch]
```

Upload all local branch commits to remote repository

```
$ git push [remote] [branch]
```

Publish your tags

```
$ git push --tags
```

Patches

Create a patch file for whole branch

```
$ git format-patch [branch] -stdout > [patch-file.patch]
```

See what is in the patch file

```
$ git apply -stat [patch-file.patch]
```

Check the patch file before applying

```
$ git apply -check [patch-file.patch]
```

Apply a patch file to repository

```
$ git am -signoff < [patch-file.patch]
```

Ignore tracking

A text file named `.gitignore` suppresses accidental versioning of files and paths matching the specified patterns

```
*.log
build/
temp-*
```

List all ignored files in this project

```
$ git ls-files --others --ignored --exclude-standard
```