Git Cheat Sheet (by Shamim)

What is Git?

Git is a distributed version control system (DVCS) that is widely used for tracking changes in source code during software development. It allows multiple developers to collaborate on a project, keeping track of modifications, and managing different versions of the codebase. Developed by Linus Torvalds in 2005, Git has become the de facto standard for version control in the software development industry.

Download Git For All Platforms

Download Git for your operating system(mac, windows, linux)

https://git-scm.com

Setup

Configuring user information used across all local repositories

```
# set a name that is identifiable for credit when review version history
git config --global user.name "John Doe"

# set an email address that will be associated with each history marker
git config --global user.email "example@gmail.com"
```

Initialize | Clone a repository from remote location

```
# initialize an existing directory as a Git repository
git init

# clone a repository from remote location  (url => remote repository location. like github repository url.)
git clone [url]
```

Stage

```
# Note: [file => file path. like index.js]

# add a file as it looks now to your next commit (stage)
git add [file]

# add all file as it looks now to your next commit (stage)
git add .

# show modified files in working directory (including staged and not staged file for commit)
git status

# to discard changes in working directory that are not in staged area
```

```
git restore [file]

# unstage a file from stage area
git reset [file]
# or
git restore --staged [file]
```

Snapshot

```
# difference of what is changed but not staged
git diff

# difference of what is staged but not yet commited
git diff --staged

# commit your staged content as a new commit snapshot
git commit -m "[descriptive message]"
```

Branch

```
# show all branches(* will appear next to the currently active branch)
git branch

# create a new branch at the current commit
git branch [your-branch-name]

# switch to the another branch
git checkout [branch-name]


# create and switch to a new branch
git checkout -b [new-branch-name]
```

Merge a Branch

```
# merge a branch's history (local branch) into the currently active branch
git merge [branch-name]
```

Inspect & Compare

```
# Note: [SHA => unique commit-id that are appears on (git log) / (git log --oneline)]

# show all commits in the current branch's history
git log

# short version of log
```

```
git log --oneline

# show the commits on branch1 that are not on branch2
git log branch2..branch1

# show the diff of what is in branch1 that is not on branch2
git diff branch2..branch1

# details of a individual commit(by commit id)
git show [SHA]

# history of a file (details of every line of a file)
git blame [file]
```

Tracking path changes

```
# remove a file from the project directory (permanently) and stage for the next commit
git rm [file]

# force fully remove a file from the project directory (permanently) and stage for the next commit
git rm [file] --f

# untrack the file and keep it in the project directory
git rm [file] --cached

# change an existing filepath and stage the move (move one folder to another folder)
git mv [existing-file-path] [new file path]

# show all commit logs with indication of any paths that moved
 git log --stat -M
```

Ignore file/folders

create a .gitignore file to the project root (Preventing unintentional staging or commiting of files)

```
node_modules
logs/
.env.local
pattern*/
```

Share and Update Your Code

```
# Note: [alias => name of your remote address]

# add a remote git url(github repository url) as an alias
git remote add [alias] [remote-server-url]

# show remote alias
git remote
```

```
# show remote location
git remote -v

# transmit local branch commits to the remote repository branch
git push [alias] [branch-name]

# create new branch and push to the remote server
git push --set-upstream [alias] [main]


# fetch all branches from remote repository
git fetch [alias]

# merge a remote repository branch into currently active local branch
git merge [alias]/[branch]


# fetch & merge any commits from the tracking remote branch
git pull
```

Git History Re-writing

```
# bring all commits from another branch and apply into the current branch
git rebase [branch]


# back to the previous commit and clear staging area (by commit id)
git reset --hard [SHA]


# revert back to the previous commit(by commit id)
git revert [SHA]
```

Temporary commits

```
# save changes temporary and stage changes
git stash

# apply temporary changes from top of stash stack
git stash apply

# list stashed file changes in stack-order
git stash list

# write working from top of stash stack
git stash pop

# drop stash from top of the stash stack
git stash drop
```

Thank you for using this cheat sheet to get the help you needed.