

**GitCheat Sheet**

The essential Git commands every developer must know

**Creating Snapshot**

**Initializing a repository**

git init

**Staging files**

git add file1.js

git add file1.js file2.js git add \*.js

git add .

**Viewing the status** git status

git status -s

# Stages a single file

# Stages multiple files

# Stages with a pattern

# Stages the current directory and all its content

# Full status

# Short status

**Committing the staged files**

git commit -m “Message” # Commits with a one-line message git commit # Opens the default editor to type a long message

**Skipping the staging area**

git commit -am “Message”

**Removing files**

git rm file1.js

git rm --cached file1.js

**Renaming or moving files** git mv file1.js file1.txt

# Removes from working directory and staging area # Removes from staging area only

**Viewing the staged/unstaged changes**

git diff

git diff --staged git diff --cached

**Viewing the history** git log

git log --oneline git log --reverse

**Viewing a commit** git show 921a2ff git show HEAD

git show HEAD~2 git show HEAD:file.js

# Shows unstaged changes

# Shows staged changes

# Same as the above

# Full history

# Summary

# Lists the commits from the oldest to the newest

# Shows the given commit

# Shows the last commit

# Two steps before the last commit

# Shows the version of file.js stored in the last commit

**Unstaging files (undoing git add)**

git restore --staged file.js # Copies the last version of file.js from repo to index **Discarding local changes**

git restore file.js

git restore file1.js file2.js git restore .

git clean -fd

# Copies file.js from index to working directory # Restores multiple files in working directory # Discards all local changes (except untracked files) # Removes all untracked files

**Restoring an earlier version of a file** git restore --source=HEAD~2 file.js

**Browsing History Viewing the history**

git log --stat

git log --patch

**Filtering the history** git log -3

git log --author=“Mosh”

# Shows the list of modified files # Shows the actual changes (patches)

# Shows the last 3 entries

git log --before=“2020-08-17” git log --after=“one week ago”

git log --grep=“GUI” git log -S“GUI”

git log hash1..hash2 git log file.txt

**Formatting the log output**

# Commits with “GUI” in their message # Commits with “GUI” in their patches # Range of commits

# Commits that touched file.txt

git log --pretty=format:”%an committed %H”

**Creating an alias**

git config --global alias.lg “log --oneline"

**Viewing a commit**

git show HEAD~2

git show HEAD~2:file1.txt # Shows the version of file stored in this commit

**Comparing commits**

git diff HEAD~2 HEAD # Shows the changes between two commits git diff HEAD~2 HEAD file.txt # Changes to file.txt only

**Checking out a commit** git checkout dad47ed git checkout master

**Finding a bad commit** git bisect start

git bisect bad

git bisect good ca49180 git bisect reset

**Finding contributors** git shortlog

# Checks out the given commit

# Checks out the master branch

# Marks the current commit as a bad commit # Marks the given commit as a good commit # Terminates the bisect session

**Viewing the history of a file**

git log file.txt

git log --stat file.txt git log --patch file.txt

**Finding the author of lines**

# Shows the commits that touched file.txt # Shows statistics (the number of changes) for file.txt # Shows the patches (changes) applied to file.txt

git blame file.txt # Shows the author of each line in file.txt **Tagging**

git tag v1.0

git tag v1.0 5e7a828 git tag

git tag -d v1.0

# Tags the last commit as v1.0 # Tags an earlier commit # Lists all the tags

# Deletes the given tag

**Branching & Merging Managing branches**

git branch bugfix git checkout bugfix git switch bugfix

git switch -C bugfix git branch -d bugfix

**Comparing branches** git log master..bugfix git diff master..bugfix

**Stashing**

# Creates a new branch called bugfix

# Switches to the bugfix branch

# Same as the above

# Creates and switches

# Deletes the bugfix branch

# Lists the commits in the bugfix branch not in master # Shows the summary of changes

git stash push -m “New tax rules” git stash list

git stash show stash@{1}

git stash show 1

git stash apply 1

git stash drop 1

git stash clear

**Merging**

# Creates a new stash

# Lists all the stashes

# Shows the given stash

# shortcut for stash@{1}

# Applies the given stash to the working dir # Deletes the given stash

# Deletes all the stashes

git merge bugfix

git merge --no-ff bugfix

# Merges the bugfix branch into the current branch # Creates a merge commit even if FF is possible

git merge --squash bugfix # Performs a squash merge git merge --abort # Aborts the merge

**Viewing the merged branches**

git branch --merged git branch --no-merged

**Rebasing**

# Shows the merged branches # Shows the unmerged branches

git rebase master # Changes the base of the current branch

**Cherry picking**

git cherry-pick dad47ed # Applies the given commit on the current branch

**Collaboration**

**Cloning a repository**

git clone url

**Syncing with remotes** git fetch origin master git fetch origin

git fetch

git pull

git push origin master git push

**Sharing tags**

git push origin v1.0

git push origin —delete v1.0

**Sharing branches**

git branch -r

git branch -vv

git push -u origin bugfix git push -d origin bugfix

**Managing remotes**

git remote

git remote add upstream url git remote rm upstream

# Fetches master from origin

# Fetches all objects from origin # Shortcut for “git fetch origin”

# Fetch + merge

# Pushes master to origin

# Shortcut for “git push origin master” # Pushes tag v1.0 to origin

# Shows remote tracking branches # Shows local & remote tracking branches # Pushes bugfix to origin

# Removes bugfix from origin

# Shows remote repos

# Adds a new remote called upstream # Remotes upstream

**Rewriting History Undoing commits**

git reset --soft HEAD^ git reset --mixed HEAD^ git reset --hard HEAD^

**Reverting commits** git revert 72856ea git revert HEAD~3..

# Removes the last commit, keeps changed staged # Unstages the changes as well

# Discards local changes

# Reverts the given commit

# Reverts the last three commits

git revert --no-commit HEAD~3.. **Recovering lost commits**

git reflog

git reflog show bugfix

# Shows the history of HEAD # Shows the history of bugfix pointer

**Amending the last commit** git commit --amend

**Interactive rebasing** git rebase -i HEAD~5