GIT CHEAT SHEET

Git Config

git config — global user.name <name>	Defines the author name to be used by the current user for all commits.
git config — global user.email <email>	Defines the author email to be used for all commits by the current user.
git config —global —edit	Opens a text editor for editing the username and email address.

Git Basics (Local Git)

git —version	It shows the git version installed.
git init <directory>	Creates an empty git repository in the specified directory.
git add . / git add <directory>	Adds all changes. / Appends all changes in the directory for the next commit.
git commit -m “message”	Commits the changes added to the index with a message.
git status	Indicates which files in the directory have been modified, added, or not added.
git log	Shows the past changes (such as commit, merge, branch) made in the directory.
git diff	Shows the differences between the two directories.

Git Essentials (Filesystem interactions)

git mv <old_file_name> <new_file_name>	Used to rename or move a file.
git rm	Used to remove a file.
git rm -n (veya —dry-run)	Used to see a simulated git rm run without actually removing the requested file.
git reset / git reset --mixed	It resets the state of all files in the working directory to match the last commit, but is used to leave the working directory unchanged.
git reset —hard	Indicates that it wants both the directory staging area and the working directory to be reset to the state of the previous commit from this branch.
git reset --soft	It neither resets the directory staging area nor the working directory, it just changes the HEAD pointer to point to the previous commit.
git reset <commit_id>	It is used to reset the status of all files in the working directory down to <commit_id>.
git reset —hard <commit_id>	It deletes all uncommited changes and resets the state of the working directory up to <commit id>.
git clean —force	It is used to delete untracked files.
git ls-files	It is used to view the tracked files in the Git working directory.
git ls-files -others (-o)	It is used to view untracked files in the Git working directory.
git clean —force -X	Used to delete ignored files in git directory.
git clean -x	It is used to remove both ignored and untracked files.
git clean -xdf	Removes all untracked, ignored files (-x) and directories (-d).
git revert <commit_id>	Undoes all changes made to <commit_id> and applies them to the current branch.
git stash	Temporarily stores current changes.
git stash list	Lists the stored files.
git stash pop	Retrieves stored files, restores them to the git directory.
git stash clear	Clears stored files.
git ls-files -v	The default is used to list unmodified files.

Git Basics (Remote Git)

git remote add <name> <url>	Creates a new connection to the remote repository. After adding the link, <name> (example: origin) is used as a shortcut for <url> in other commands.
git remote —verbose	It is used to see the connection created to the remote repository.
git remote show	Provides information about the information contained in the remote repository.
git push --set-upstream origin <branch name>	It is used to send the changes made in the local repository to the remote repository.
git clone <repo_url>	Clones the repository located at <repo_url> to the local machine.
git pull	It downloads new commits from another repository and merges the remote branch with the local branch.
git fetch	It is used to download new commits in the remote repository without changing the changes in the local repository.
git branch <branch name>	Creates a new branch named <branch_name> from the current branch.
git branch	Lists the branches in the repository.
git checkout <branch name>	Switches between branches.
git checkout -b <branch name>	Creates a new branch named <branch_name> from the current branch and switches to that branch.
git branch -a	Lists local branches and remote tracking branches.
git branch -r	Lists remote branches.
git branch -vv	Lists local tracking branches.
git branch --track <track_branch_name> origin/<branch_name>	Creates a local tracking branch for origin/<branch_name> named <track_branch_name>.
git merge <branch_name>	Merges <branch_name> into master branch
git rebase <base>	It adds the additional commits in the <base> to the end of the current branch, but does not move them, it creates the commits again with the new commit_id.
git cherry-pick <commit_id>	It takes the changes from the commit with <commit_id> and moves them to the current branch.
git push --delete origin <branch name>	Deletes the <branch_name> in the remote repository.
git branch --delete <branch name>	Deletes the <branch_name> located in the local repository.

Git Log (Advanced)

git log —author	Searches for commits for a specific author.
git log —grep “<pattern>”	Searches for commit messages that match <pattern> in it.
git log —max-count (or -n or -<limit>)	Limits the number of commits (git log -5).
git log —reverse	It sorts the commits with the oldest commit first.
git log —after “<date>”	Shows commits after <date>.

git log —before “<date>”	Shows commits before <date>.
git show HEAD^	Shows the last commit. Equal to git log —max-count=1 —patch HEAD^.
git log —patch (or -p)	It shows the exact difference of each commit.
git log —format <parameter>	<parameter> can be parameters such as email, medium, oneline, short, full, fuller, and raw.
git log --format="%ar %an did: %s"	(format string) uses placeholders to populate various attributes per commit.

git shortlog	It shows commits grouped by author, with a commit subject per line.
git log —graph —decorate	The —graph parameter draws a text-based commit graph on the left side of the commit message. —decorate adds branches or tags of the commits shown.
git log —stat	It shows which files have been modified and the corresponding number of lines added or deleted on each.
git log — <file>	It only shows commits for the specified <file>.
git blame —date=short <file_name>	Shows who last modified each line of a file.
git reflog	Shows the log of changes in the local repository.