KALAM DREAM LABS

GLOBAL TECHNOLOGY FOR LOCAL NEEDS

GIT Cheat Sheet

Commands segregation

MOST USED COMMANDS IN GIT:

```
1. git add .                                  # Stages the current directory and all its content
2. git status                                 # Full status
3. git commit –m “message”                    # Commits with a one-line message
4. git commit                                 # Opens the default editor to type a long message
5. git rm <file name .extension>              # Removes from working directory and staging area
6. git rm --cached <file name.extension>      # Removes from staging area only
7. git log                                    # Full history
8. git show <commit ID>                       # Shows the given commit
9. git show Head                              # Shows the last commit
10. git restore --staged <file name.extension>         # Copies the last version of file.js from repo to index
11. git restore <file name .extension>         # Copies file.js from index to working directory
12. git restore .                              # Discards all local changes (except untracked files)
13. git log –author=”Mosh”
14. git checkout master                        # Checks out the master branch
15. git log <file name .extension>             # Shows the commits that touched file.txt
16. git blame <file name.extension>            # Shows the author of each line in file.txt
17. git branch <branch_name>                   # Creates a new branch called new branch
18. git checkout <branch_name>                 # Switches to the new branch
19. git switch <branch_name>                   # Same as the above
20. git switch –C<branch_name>                 # Creates and switches
21. git branch –d<branch_name>                 # Deletes the bugfix branch
22. git stash push –m”msg”                     # Creates a new stash
23. git stash list                             # Lists all the stashes
24. git stash apply 1                          # Applies the given stash to the working dir
25. git stash drop 1                           # Deletes the given stash
26. git stash clear                            # Deletes all the stashes
29. git rebase master                          # Changes the base of the current branch
30. git clone url
31. git fetch origin master      # Fetches master from origin
32. git fetch origin             # Fetches all objects from origin
33. git fetch                    # Shortcut for “git fetch origin”
34. git pull                     # Fetch + merge
35. git push origin master     # Pushes master to origin
36. git push                   # Shortcut for “git push origin master”
```

MEDIUM USED COMMANDS IN GIT:

```
1. git inti
2. git status –s                              # Short status
3. git commit --am “msg”
4. git mv file1.js file2.js
5. git diff                                   # Shows unstaged changes
6. git diff --staged                          # Shows staged changes
```

7. git log --oneline # Summary
8. git log --reverse # Lists the commits from the oldest to the newest
9. git show HEAD~2 # Two steps before the last commit
10. git restore --staged <file name.ext> # Copies the last version of file.js from repo to index
11. git restore --source=HEAD~2 <file name.ext>
12. git log --patch # Shows the actual changes (patches)
13. git log --before="date"
14. git log --after="one week ago"
15. git log hash1..hash2 # Range of commits
16. git log <filename .ext> # Commits that touched file.txt
17. git config --global alias.lg"log --oneline"
18. git diff HEAD~2# Shows the changes between two commits
19. git diff HEAD~2HEAD<file name .ext> # Changes to file.txt only
20. git shortlog
21. git log --stat <file name.ext> # Shows statistics (the number of changes) for file.txt
22. git tag v1.0 # Tags the last commit as v1.0
23. git log master..<branch_name> # Lists the commits in the bugfix branch not in master
24. git diff master..<branch_name> # Shows the summary of changes
25. git stash show stash@{1} # Shows the given stash
26. git stash show 1 # shortcut for stash@{1}
27. git merge<branch_name> # Merges the bugfix branch into the current branch
28. git branch --no –ff<branch _name> # Creates a merge commit even if FF is possible
29. git merge --abort # Aborts the merge
30. git branch --merged # Shows the merged branches
31. git branch --no-merged # Shows the unmerged branches
32. git cherry –pick <commit ID> # Applies the given commit on the current branch
33. git push origin v1.0 # Pushes tag v1.0 to origin
34. git branch –r # Shows remote tracking branches
35. git branch –vv # Shows local & remote tracking branches
36. git push –u origin <branch_Name> # Pushes bugfix to origin
37. git remote add upstream url # Adds a new remote called upstream
38. git reset --soft HEAD^ # Removes the last commit, keeps changed staged
39. git reset --mixed HEAD^ # Unstages the changes as well
40. git reset --hard HEAD^ # Discards local changes
41. git reflog # Shows the history of HEAD
42. git reflog show <branch_name> # Shows the history of bugfix pointer
43. git commit --amend

LEAST USED COMMANDS IN GIT:

1. git add<file name.ext> # Stages a single file
2. git add<file name.ext><file name .ext> # Stages multiple files
3. git add *.ext # Stages with a pattern
4. git show HEAD:<file name.ext> # Shows the version of file.js stored in the last commit
5. git clean –fd # Removes all untracked files
6. git log --grep="GUI" # Commits with "GUI" in their message

```
7. git log –S"GUI"                          # Commits with "GUI" in their patches
8. git log --pretty =format:"%an committed %H"
9. git show HEAD~2
10. git show HEAD~2:<file name.ext>       # Shows the version of file stored in this commit
11. git bisect start
12. git bisect bad                          # Marks the current commit as a bad commit
13. git bisect good <commit ID>             # Marks the given commit as a good commit
14. git bisect reset                        # Terminates the bisect session
15. git log --patch<file name.ext>          # Shows the patches (changes) applied to file.txt
16. git tag                                 # Lists all the tags
17. git tag v1.0 <commit ID>                # Tags an earlier commit
18. git tag –d v1.0                         # Deletes the given tag
19. git merge --squash <branch_name># Performs a squash merge
20. git push origin –delete v1.0
21. git push –d origin <branch_name># Removes bugfix from origin
22. git remote                              # Shows remote repos
23. git remote rm upstream                  # Remotes upstream
24. git revert <commit ID>                  # Reverts the given commit
25. git revert HEAD~3..                     # Reverts the last three commits
26. git revert --no-commit HEAD~3
27. git rebash – i HEAD~5
```