Git Commands

* Initial setup commands

Need to be done only once at the time of first startup for global settings or can be done for each git repository initialized

* git config --global user.name “name”
* git config –global user.email ”[something@mail.com](mailto:something@mail.com)”

Both above commands are for telling the system about the user. These details are shown each time a commit is made whether in local or remote repo for traceability.

* Working with repo
* git init : Navigate to location where the repo is needed to be created and run this command, an empty repo will be created.
* git add . / git add \* : Stage the changes done for commit in local repo.
* git add filename : Stage a specific file
* git \*.ext : Stage specific set of files with given extension.
* git commit –m “msg” : push the staged changes in the local repo. Message is necessary for every commit done as it tells about the changes done in the specific commit.
* git commit –a –m “msg” : push the changes directly into repo without staging.
* Git status: shows the status of tracked, untracked or modified files in the directory.
* Git status –s: shows short status for the files.
* For ignoring specific files in the working directory from staging or committing, create a .gitignore and add the files or directory name in it and commit it.
* Git diff: compares what’s in the working directory with the staged files.
* Git diff –staged: compares the staged changes with last commit.
* Git diff –cached: similar to staged
* rm filename: removes a file from the working directory and from tracked files. If committed afterwards, the file is completely removed from the repo.
* Git rm filename: stages the removal of file from the repo.
* Git rm –cached filename: removes the file from tracked but keeps in the working directory.
* Git rm –f filename: to forcefully delete a indexed file i.e., forcefully a staged file.
* Git mv file\_from file\_to: renames a file.
* Git log: view the commit history.
* Git log –p -2: -p view the changes introduced in each commit -2 limit the output to two commits.
* Git log --stat: see abbreviated stat of each commit.
* Git log –pretty=oneline / git log –oneline: shows all the commit hash with one line commit message.
* Git log –pretty=format: “%h -%a, %ar : %s” : shows the commit log in the specified format hereit shows the commit in hash author time and message sequence.
* Git log –graph: adds a ASCII graph showing branch and merge history along with commits.
* Git commit --amend: changes the last commit, adds the staged files in last commit. If there are no files in staged area then it just changes the commit message.
* Git reset HEAD filename: unstages a staged file.
* Git checkout -- filename: discards any changes staged in working directory
* Working with remote repo
* Git clone url : clones the data from the specified url, creates a directory and places a .git in the directory.
* Git clone url name: clones the code from specified url and places the code in directory named name with .git in it.
* Git remote: shows all the remote urls saved for current repo
* Git remote –v: shows the name for the remote along with url and command executed for the same.
* Git remote add name url: adds the given url for given name
* Git fetch name: fetches the code from the url saved as name.
* Git push name branch: pushes the data of specified branch in the remote name url.
* Git remote show name: displays information about particular repo.
* Git remote rename name name2: renames the remote ref name to name2.
* Git remote rm name: removes the remote ref.
* Git push origin --delete branchname: deletes the remote server branch.
* Extra git working
* Git tag: Tags are like bookmarks in git, used to tag specific points in the history. This command shows the list of available tags in alphabetical order.
* Git tag –l ‘v.\*’: search for a specific pattern in the tags.
* Git tag –a name –m “msg”: adds a tag with specified name along with message.
* Git show name: shows the details of a specific tag. That shows the tagger information, the date the commit was tagged, and the annotation message before showing the commit information.
* Git tag name –lw: adds a lightweight tag for specific commit which doesn’t stores any additional information just the author, date of tag, message and commit. No need to specify –a, -s or –m options.
* Git tag –a name hash: tags a commit of specified hash.
* Git push origin tagname: pushes the specified tag to remote as by default the tags are not shared.
* Git push origin --tags: pushes all the tags at once.
* Git aliases: git doesn’t automatically infer your command. If the user doesn’t want to type the whole command then they can create aliases for the same and those can be used later. Eg.: git config --global alias.co checkout. The same command can be used for aliases for some commands like git config --global alias.unstage ‘reset HEAD --’.
* Git branch: displays the list of branches in the repo with the position of head.
* Git branch name: creates a new branch with all the data till its previous commits.
* Git branch –d branchname: deletes the specified branch.
* Git branch –D branchname: forcefully deletes specified branch without merging.
* Git branch –v: displays all the branches with their last commit.
* Git branch –merged: displays the branches that are already merged.
* Git branch --no-merged: displays all the non-merged branches.
* Git checkout branchname: to change the current working branch. After checkout the head points to the branch. After this all the commits done will be reflected in the branch only and not the master branch.
* Git checkout –b branchname: creates a branch and switches to it at the same time.
* Git merge branchname: merges the specified branch into current branch.
* Git rebase base: moving a branch to new base commit. It replaces the old commits to new one.
* Git rebase -i base: starts an interactive rebasing session. Gives complete control over what the project history will look like.
* Git rebase basebranch topicbranch: checksout the topic branch and replays it on the server branch if not checked out.