Git Commands

```
//Cloning
$ git clone git@github.com:Ritesh-Goyal/devops.git

//Set global name
$ git config --global user.name "Ritesh G"

//Set email id
$ git config --global user.email "ritesh.goyal590@gmail.com"

// Check all git configs
$ cat ~/.gitconfig

// List all global config

$ git config --global --list

// Add alias for status and log command
$ git config --global alias.s "status -s"
$ git config --global alias.lg "log --oneline --all --graph --decorate"

$ git log --oneline

// For checking the git status
$ git status -s
$ git s  //this is as per alias which we created

//create a file
$ touch index.html
$ git status

//add to git
$ git add login.html
$ git add .  // it will add all content
$ git add -A // it will add all content including deleted

$ git status -s
A  aboutus.html //stagged file
 M shop.html    // modified file
?? contact.html // untracked file
D  aboutus.html // delete file
```

R index.html -> home.html // Rename file

// command to commit
$ git commit -m "sample commit"

//below method will add as well as commit in git. It will do only for modified files.
$ git commit -am "commit"

//Branching- allows to create independent line of work
 //gives local branches
$ git branch
 * master

//gives local & remote branches
$ git branch -a

$ git branch develop // create uat branch
$ git checkout develop //Switched to branch 'uat'
$ git checkout -b uat // creates and switches to new branch

// for deleting uat branch
$ git branch -D uat

//check remote details
$ git remote -v
origin git@github.com:Ritesh-Goyal/devops.git (fetch)
origin git@github.com:Ritesh-Goyal/devops.git (push)

// Command to push code remotely
//for remote dev branch
$ git push -u origin develop

//for remote main branch
$ git push -u origin main

// git pull for git version > 2.0
$ git pull // will only push from particular branch by default

//pull changes from remote origin main branch to local dev branch
// Pull does Fetch+merge
$ git pull origin main

// will fetch+rebase // clean pull with all logs of other commits
$ git pull --rebase

// Fetch
$ git fetch //will get copy of remote origin/master to local

// gitignore for ignoring type of files and specific names
$ vi .gitignore

//not ignore command inside .gitignore file
!index.log

//diff command
$ git diff --staged
$ git diff HEAD

//tags
Annotated Tags
Annotated tags are tags that store extra Metadata like developer name, email, date, and more. They are stored as a bundle of objects in the Git database.

$ git tag -m “message” <version>
$ git show <version>

Light-Weighted Tag:
Git supports one more type of tag; it is called as Light-weighted tag. The motive of both tags is the same as marking a point in the repository. Usually, it is a commit stored in a file. It does not store unnecessary information to keep it light-weight.

$ git tag <tag name>
$ git tag projectv1.0

$ git push origin <tagname>

// git reset
$ git reset commit-id
$ git reset current~2

$ git reset --hard dd90d0f

// The net effect of the git revert command is similar to reset, but its approach is different. Where the reset command moves the branch pointer back in the chain (typically) to "undo" changes, the revert command adds a new commit at the end of the chain to "cancel" changes.

$ git revert HEAD

Name: Ritesh Goyal
Contact: 9960930111
Email-id: ritesh.goyal590@gmail.com / ritesh.devopstrainer@gmail.com