Git versioning

1. Version Control – Git allows you to track changes to files, see who made what changes, and revert to previous versions if needed.
2. Collaboration – Multiple people can work on the same file simultaneously using branches and merging changes.
3. Branching and Multiple Versions – You can maintain multiple versions of a project using branches, making it easy to develop features or fixes in parallel.
4. Data Safety – Changes are never truly lost because Git stores a history of all commits, allowing you to revert to any previous state.
5. Push – to merge the local changes into our git repository.
6. Fetch – to retrieve changes from the remote to local

Commands:   
  
1. git remote -v (displays the URLs of the remote repositories associated with your local Git repository.)  
Commit – set of saved repository changes. (never lost)  
commit has a hash key – unique identifier. (is same for all the environments).   
  
2. git add . (git add . stages all the changes (new, modified, or deleted files) in the current directory and its subdirectories for the next commit, this is like a holding area, where you can review and make any changes if needed if you them permanent part of your repo)  
(. -> to stage all the changes)  
(you can specify the files as well if you want)  
  
3. git commit -m “XYZ” (allows us to commit the changes)

4. git log (gives the commits of the repository)  
5. git status - The git status command shows the current state of your working directory and staging area. It helps you understand which changes are staged, unstaged, or untracked before committing.

6. git show <hash> - gives in detail commit info  
7. git show –name-only <hash>  
8. gives shorthand commit info  
9. git reflog - gives info about all commits on all branches   
10. git branch <branchName> - for creating a new branch.   
11. git push -u origin featureBranch  
12. git switch <branchName>  
13. git clone - clone a remote repository to a local machine.   
14. git pull  
15. git revert <hash>   
:q!

16. git reset <hash> - (moves the current branch to a specific commit, effectively undoing commits after it).  
a. –soft -(moves head to the specific commit but keeps changes staged).   
b. –hard - (Moves HEAD to the commit and deletes all changes after it.)

17. git merge featureBranch ;   
This command merges changes from featureBranch into the current branch (usually main or develop).  
  
18. git push -f do?

Normally, when you push changes (git push), Git ensures that the remote branch is updated **only if** your changes are based on the latest version of that branch.

But git push -f (**force push**) **overwrites** the remote branch with your local branch, even if the remote branch has different commits.

That’s what git push -f does! It **forces** your changes, **erasing** what was there before. (without updating)

19. git init -h  
Displays the help documentation for git init  
  
20. what is git init (affects only the local machine)  
(for initializing a git repo in a directory) (.git file will get added in that directory, Git is now ready to track change)  
21. git remote add origin <https://github.com/your-username/your-repo.git> (to link the local repo to your github repo) (Adds a new remote repository reference to your local Git repository)  
  
22. git diff sourceBranch targetBranch  
Compares the differences between two branches (sourceBranch and targetBranch).