What are we going to see in this session?

- What is Head ?
- Git Restore
- Git Checkout
- Git Reset
- Ignoring files by Git

What is Head ?

- You can see the “pointer” head from command “git log”
- This actually helps us to understand which commit we are using currently.
- By default head → points to the last commit.
- If in-case we go back to any of the previous state of project then head will point to the relevant commit.
- You can also learn more details about of head using command “git show HEAD”
- This command is going to provide more visibility about your commit and as well as commit difference (which we can see at later point)
- Note : You can also view this output by providing head ID or array number.

  “git show 0d196fa”

  “git show HEAD~1”

Git Restore

- Let's say for example we are modifying the project file. (Adding some line into it)
- As a result of modification you can see file status as modified in "git status" command.
- Now if in-case you need to restore the change back you can simply use command "git restore filename"
- If in-case you need to restore multiple file changes you can use commands "git restore . (or) git restore *

Git Checkout

- Let see how to undo things using “git checkout”
- Using this command we can go back in time & check different state of our files.
- If you want to move previous state just use command “git checkout <head-id>”
- This command is going to move your head to previous state, but not going to delete the files. You can check in which state you are using command “git status”
- You can revert back to the current state using command “git checkout master”
- Note : Its very safe to use git checkout to go back in time because its read only.
- This can proved by going back to previous state file and modify something and make a commit. However this commit is not going to result anywhere.
  - ✓ git checkout <head-id>
  - ✓ Modify the file
  - ✓ Stage the file & make test commit
  - ✓ “git status & git log” command is going to show you the commit.
  - ✓ Go back to current state “git checkout master” your new test commit will be no longer exist.
- Checkout also allows us to move between different branches. (which we can see at later point)

Git Reset

- Git reset is going to help us to delete the commits which we made.
- Git reset has 3 different flags. (soft, mixed, hard)

  - ✓ Create sample file and do stage & commit the file.
  - ✓ Lets try “git reset --mixed” which is default flag, if incase you are not specifying anything.
  - ✓ git reset --mixed <head-id> this just going to un stage the files and also remove the commit.
  - ✓ git reset --soft <head-id> this just going to remove the files from commit and keep it in staging area.
  - ✓ git reset --hard <head-id> this command is going to be bit dangerous which is going to delete the file permanently from workspace.

Git ignore

- You might be in an situation to maintain some files not tracked by Git. To solve this kind of situation git has an solution called ".gitignore"
  - ✓ Create file called .gitignore under your project directory.
  - ✓ Create some sample files which need to be ignored & update those file details in .gitignore
- Note : Same procedure might not work if incase you are trying with existing files which is already managed by git. Because those files are exist before we create .gitignore file.
  - ✓ Modify the existing file which is tracked by git & try update that file details in .gitignore.
- Ignore complete folder
  - ✓ Create folder with couple files underneath.
  - ✓ One way is providing the absolute details in .gitignore like below

```
# Auto generated files
auto-generated-files/a.txt
auto-generated-files/b.txt
```

Else, you use * to notify all the files under the folder

```
auto-generated-files/*
```

End of this topic !

Any questions?