**Branching**

Jab hum commit krty hein hmara har commit aik hash generate krta hy. Hr commit reference krta hy atleast aik parent commit ko jo us se pehle kia jaata hy.

Jab bht sare contexts me kaam ho raha ho kisi bhi project par tab hum linear commits k sath ni kr skty humy wahan use krna prta hy branches ko.

Jab ap aik branh par kaam krogy to ap koi doosri branch ni khraab krogy. Ye faida hoga branching krne ka jab ap aik branch me change krogy doosri branch ko uska nip ta hoga.

**Is it master or main:**

Hum Git me kaam aik branch me hi kr rhy hoty hein jisko master branch kehty hein. Lekin ye hoti aese hi hy jese koi bhi aam branch hoti hy git me.

**Head 🡪 Master:**

Ye term hy Git ki specifically ye point krta hy hmari current location ko repository ke andr.

Forexample aik book ki example lein jisme 3 nishaniya lgi hui hein different jgho pr wo 3 nishania aese smjho jese 3 branches hein ab aik wqt me ap aik hi nishani wala page khol skty ho. Aik wqt me aik hi page khul skta hy to yehi kaam HEAD krta hy wo refer krta hy aapki active branch kit rf.

**Viewing Branches:**

Git branch 🡪 ye command likhne sari existing branches show ho jati hein.

Switch Branches:

Git switch 🡪 ye new command aayi hy branches k drmian switch krne k liye is se pehle git checkout use hoti thi.

* Git branch -v : aese krne se thori information or mil jati hy branches k baare me.

**Git checkout vs Git Switch:**

Git checkout branches switch krne ke ilawa or bhi bht se kaam krti hy.

**Create and switching in one Go:**

Git switch -c <branh-name>

Git checkout -b <branch-name>

**Switching Branches with unstaged changes:**

Agr changes stage kiye bgair branch switch ki jyegi to changes lost ho jiengi.

Ksii doosri branch par agr koi new file bnai jye or phr usko stage or commit kiye bgair doosri branch pr switch kia to wo file apki doosri branch pr b aa jyegi. Ku k wo bs aik branch pr create hui thi. Is liye jis bhi branch me jaogy file apko follow krygi. Jb tk kisi aik branch se usko stage or commit ni kr dety.

**Deleting and Renaming Branches:**

Jis branch me ap hoty ho us me rehty hue ap delete ni kr skty apko kisi doosri branch me switch krna prta hy.

* Git branch -d <branch-name>

Agr fully merged hy to hogi delete wrna -D lgana pry ga jo ke force ke equal hy.

Rename krna delete ke opposite hy isme apko us branch me switch krna pryga jisko rename krna hy.

* Git branch -m <new-name>

**How Git Stores HEAD And Branches:**

.git folder k andr HEAD Hota hy

**Merging Branches:**

Git par ap different contexts par kaam kr skty ho. Kuch changes krty ho lekin at the end apko kuch to aesa hoga jo incorporate krna pryga uske liye use hota hy ye concept merging ka.

Hum branches merge krty hein koi specific commit ni.

Hum hmesha branch merge krty hein based on the current HEAD Branch.

Hum merge krty hein is hisaab se k hum kahan hein or HEAD Kahan hein.

**Fast Forward Merge:**

Merging ki simplest form hy fast forward merge.

STEPS:

1. SWITCH kro destination branch mein
2. Phr merge ki command chalao

Command : 🡪 git merge branchname

**Merge Commits:**

Har commit fast forward ni hota kbhi aesa b hota hy k new branch kisi aik branch se bna kr ap kaam shuru kr dety ho or jis branch se bnai hoti hy whn new commits hoty hein jo ap k pass ni hoty whn phr concept ata hy merge commits ka.

Commits can have more than one parent.

Git apke liye new commit bnata hy jb a fast forward merge ni kr rhy hoty.

**Merge Conflicts:**

Kuch changes ap krty ho aesi jo git automatically merge ni krta wo apko manually krni prti hein jiski wjha hoti hy merge conflict.

Merge Conflict ata kese hy?

For example koi file kiisi ny modify ki woi same file apne modify ki new branch me jb ap branch ko merge krne lgogy to git confuse hojyega k kon c changes rkhni hein dono mese.

<<<<<<HEAD

Drmian me jo show ho raha hy wo is branch ka content hy jisme ap currently ho.

=======equalto==

Equal to k neeche jo content show ho raha hy wo content wo hy jo merge hone wali branch ka hy

Content

>>>>>>>merge-hone-wali-branch.

* Merge conflict tab ata hy jab commits overlap hoty hein