**vcs : vcs is a s/w that helps s/w developers to work together and maintain a complete history of thier work and versions.**

**vcs advantages :**

**- all the team members are allowed to work freely on any file at any time.**

**- does not allow overwriting each others changes.**

**- maintain history of every version means it can move forward or backward of the previous file.**

**- passing condition (branches) in which can create branches or where we can create branches.**

**ex: google drive in we cannot create banch.**

**why we use vcs :**

**1. it is initially part of CI/CD pipeline or devops cycle.**

**2.Didnt run developers even operation team has to write some scripts like ( playbooks in ansibe) which can also maintain in versions.**

**3. if git server is config.or built in cloud .even cloud admin has to known the git.**

**whe to use vcs :**

**1.made a change to code ,realized it was a mistake and wanted to revert back...?**

**2. lost code or had a backup that was too old...?**

**3. had to maintain multiple versions of a product**

**4. wanted to prove that a particular change broke or fixed a piece of code .**

**5. wanted to review the history of some code**

**6.wanted to submit a change to someone else"s code**

**7.wanted to share your code or let other people work on your code.**

**8. wanted to see how much work is being done. and where , when and by whom ..?**

**why use git:**

**- it is a fast**

**- you dont need access to a server**

**- amazingly good atmerging simultaneous changes.**

**-everyones using gt**

**git branch :**

**branches are highly important in the git .branch represents an independent line of development.**

**- by using branches ,severl developers are able to work in parallel on the same projects simantaneously. we can use the git branch cmds for creating,deleting, listing, rename branches.**

**creating a new branch :**

**# git branch <branch name>**

**this cmd will create a branch locally .to push the new branch into the remote repo. you need to use the cmd**

**# git push -u <remote url> <branch name>**

**viewing branches or list all the local branches in the current repo :**

**# git branch or git branch --list**

**deleting a branch : (specify branch )**

**# git branch -d <branch name >**

**forcely delete the specify branch (permantely):**

**# git branch -D <branch name>**

**swiching one to another branch :**

**# git checkout <name of the banch>**

**we can acreate a branch and switch or newto a branch at the same time :**

**# git checkout -b <name of the branch>**

**git fetch : developer get changes from remote to local without merging with local repo.**

**or**

**fetch all the changes in the remote repo and move the origin /master pointer to HEAD.MEAN WHILE YOUR LOCAL BRANCH MASTER WILL KEEP pointing to where it has.**

**# git fetch url or**

**# git fetch origin master**

**here origin represents our remote repo.location.**

**git pull :developer get changes from remote to local repo.**

**or**

**pulls all the changes new commits from remote machine and master branch and it is also does git merge. git merges all changes to our local repo.git pull basicall does two things.it does git fetch insense it gets all new commits remote to local local server followed by it performs git merge . it automaticlly merges changes in remote repo. with local repo. use the cmd**

**# git pull =git fetch+ git merge**

**git auto merge using fast forwarded machanism.**

**# git pull origin master**

**the check to see all details of remote repo usethe cmd**

**# git remote -v**

**run this cmd .this gives info.about our origin.origin is a gives a names poiniting our remote server .in stop pointing this url evertime push and pull its better to give a short name which is "origin ".**

**git reset :**

**moving the changes from w.area to s.area . i have added some changes to s.area to w.area we want to use this case.we consider that as add this gives mostly used to two cmds add,commit to removed.bcoz added by mistake , i dont wan that, which means that added into s.area bring backinto w.area. thi operation is called as " reset"**

**in this 1st**

**# git checkout master**

**# git staus**

**# git add filename (existingfile)**

**# git status**

**# git reset --hard**

**git reset types" :**

**1.git reset hard**

**2. git reset soft**

**3.git reset mixed**

**#git tag : is used to give tag to the specified commit.**

**#git log : is usedto list the version history for the current branch**

**#git log --oneline -display only oneline**

**# git log --follow "filename"**

**- lists version history for a file ,including the remaining of files also.**

**# git show < commit id> - shows the metadata and content changes of the specified commit.**

**git rebase :this is also a way of combining the work b/w diff. branches .rebasing takes a set of commits ,copies them and stores them outside our repo.**

**the advantage of rebasing is that it can be used to make linear sequence of commits.**

**the commit log or history of the repo. stays clean if rebasing is done**

**# git rebase master**

**git revert : some times we need to undo the changes that we have made. there are various ways to undo our changes locally or remotely ( depends on what we need). butwe must use these cmnds to avoid unwanted deletions.**

**# git log --oneline**

**#git revert commit id (7 characters is required here )**

**# git revert 3321844**

**# git log --oneline**