Commits

? git commit

? makes a new commit

? git commit amend

? updates or edits the last commit

Branching

? git branch <branchname>

? creates a new branch

? git checkout <branchname>

? checks out an existing branch

? git checkout -b <branchname>

? creats a new branch and cheks it out

Merging

? git checkout bugFix; git merge master;

? checks out bugFix branch adn merges all changes of master branch into bugFix branch

? git commit amend

updates or edits the last commit

rebase :- helps to have a linear sequence of commits, instead of parallel ones

? git checkout -b bugFix; git commit; git checkout master; git commit; git checkout bugFix; git rebase master;

? combines the parallel branches commits to linear line and rebases master into bugFix branch

? master ? bugFix(\*)

Detach yo HEAD

? Detaching HEAD just means attaching it to a commit instead of a branch

? git checkout C1

will detach head from the branch and goes to C1 commit

Relative Refs (^) and (~)

? Moving upwards one commit at a time with Caret (^) operator

? git checkout master^ or git checkout HEAD^

equivalent to the first parent of master

? git checkout master^^ or git checkout HEAD^^

equivalent grandparent of master or second generation ancestor of master

? Moving upwards a number of times with tilde(~) operator ~<num>

? git checkout master~4 or git checkout HEAD~4

goes back 4 commits backward to the first parent of master

? git checkout master^^ or git checkout HEAD^^

equivalent grandparent of master or second generation ancestor of master

? Branch Forcing

? git branch -f master HEAD~3

moves by force the master branch to three parents behind HEAD

? git branch -f master commit-name

will move the master branch to specific commit-name

Reversing Changes

? Git Reset:- will move branch backwards as if the commit had never happen. Works great for LOCAL BRANCHES on your own machine

? git reset HEAD~1

goes 1 commit back in a state that the commit had never happen

? Git Revert:- in order to reverse changes(pushed or remote branches) and share it with others

? git revert HEAD

creates new commit C for the reverted commit c and you can work with c and push out your changes to share it with others

Cherry-pick

? git cherry-pick <commit1> <commit2>

? i would like to copy these series of commits in my current location(HEAD)

Interactive Rebase

? Is like cherry-pick but used when you dont know what commits you want?

? It is a best way to review a series of commits that you are about to rebase

? When interactive UI opens you have the ability to do 3 things

? reorder commits by dragging them up and down

? choose or omit commits by pick pick off

? squash commits:- combine commits

? git rebase -i HEAD~4

? will open the 4 commits before the HEAD and it is ur decision to pick,omit and rearrange their order.

Locally Stacked Commits

? After fixing a bug, if we fast-forwarded the master, it will copy all the commits after that but we need to tell master to copy only specific commits made after it by using

? git rebase i

? git cherry-pick

? git commit amend

updates or edits the last commit

Juggling commits(using git rebase -i), Picking the changes made after master and rearrange

? git rebase -i HEAD~<num>

? rearrange the commits put the commit that you wanna make change

? git commit amend

make your changes on the commit that uve put on the top

? git rebase -i HEAD~<num>

rearrange the commits the way you want them

? git checkout master

? checkout the master

? git branch -f master <commit>

? move the master to the branch on the top

Juggling commits(using git cherry-pick), Picking the changes made after master and rearrange

? git checkout master

? be on the master branch

? git cherry-pick commit1 commit2

pick the commits and copy them into master and work on the top commit

Git Tags

? they(somewhat) permanently mark certain commits as milestones then you can reference like a branch

? you cant checkout tags and complete work on that tag, tags exist as anchors in the commit tree that designates certain spots.

? Gitd tag v1 C1

? creates tag v1 on commit C1

? git commit amend

updates or edits the last commit

Git Describe

? git describe

? git tells yout where you are relative to the closest Anchor or Tag using

? <tag>\_<commits>\_g<hash> :- RESPONSE of git describe <optional branch name>

<tag> :- name of the tag

<commits> number of commits away this tag

<hash>:- the hash of the commit being described

Rebasing over 9000 times

? Answer to the question from learngitbranching.js.org

? git rebase master bugFix

? git rebase butFix side

? git rebase side another

? git rebase another master

To get only the remote url of ur projects

? git config --get remote.origin.url

? will display the url of remote orign

Instead of removing and re-adding, you can do this:

git remote set-url origin git://new.url.here

To remove remote use this:

git remote remove origin

To create a new remote git url for project

? If new project with no remote url yet

? git commit -m "first commit"

? git remote add origin <github repo url>

? git push -u origin master

?

? if the project has already a remote url and u want to change it

? git remote set-url origin <url>

will assign the given url as a remote repository

Commits

? git commit

? makes a new commit

? git commit amend

updates or edits the last commit

- to see the all the mvn dependency tree

mvn dependency:tree

- to clean the local repositry .m2

mvn dependency:purge-local-repository