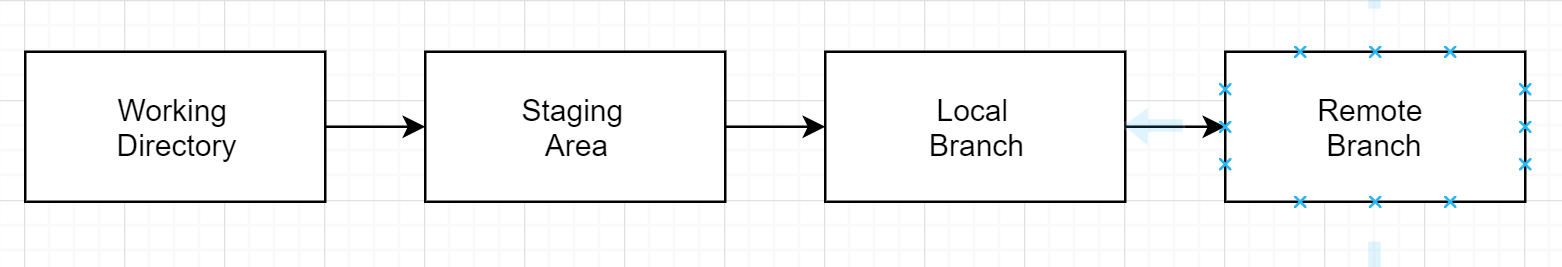
**Git Commands /Scenarios**

**Git Repository view**



|  |  |
| --- | --- |
| Commands | Purpose/ Description |
| git config --list | List the config details |
| git config --global user.name "First Last"  git config --global user.email [you@email.com](mailto:you@email.com)  clear the git credentials  git config --global --unset credential.helper | Add the git config  --local  --global  --system |
| git init | Initialize the git repository in the local machine. |
| git add . | Staging the files. |
| git commit -m “<message>”  git commit --amend | Commit the staged files local branch.  Amend your last commit |
| git log -n  git log –oneline  git reflog  git log --oneline origin/master  git log --oneline <from-branch>..origin<to-branch> | Shows the history from local branch .  Shows the history info in oneline  Shows the detailed log.  Show the logs from origin master  List the additional logs to catch up from <to-branch> |
| git status | Shows the status . |
| git reset --hard <commit-id>  --soft – only reset the local repo and leave the staging and working directly changes | -Reset all the changes occurred after this commit -Move the MASTER and HEAD references to this commit  -Discard the staging area and working directly changes |
| git remote add orgin https://github.com/senrepo/git-learn.git | Add remote repo to your local git repo |
| git remote -v | List the remote repo url |
| git log –oneline  git cherry -v | See all unpushed commit  d81d908 (HEAD -> master) Panel 4 is added in the index.html  9f84cba (origin/master, origin/HEAD) Panel 3 added in the index.html |
| git reset --hard HEAD | Discard the changes and go back HEAD |
| git revert --strategy resolve <commit-id> | Revert a particular commit, but its doesn’t do the job well sometimes |
| git rebase --abort | Abort the rebase operation |
| git diff <commit-id> <commit-id> | Compare two commits |
| git stash save <message>  git reflog show stash  git stash pop <stash-id>  git stash -p  git reset –hard  git add . | Create the stash  Show all stashes  Reappy a stash  Cherry pick the changes for stash  Discard the local changes will help applying stash  Stage the current changes will help applying stash |
| git branch -a  git branch -r  git checkout -b <branch-name>  git checkout <branch-name>  git branch -d <branch-name> | List all branches available.  List all remote branches.  Create a branch from current working directory.  Switching the branch.  Delete a branch. |
| git remote -v  git remote rm <name>  git remote add <name: origin> url | List the remote connections.  Remove a remote connection.  Add a remote connection. |

**How do we switch back to older commit view and then move back to current commit view?**

git checkout <commit-id>

git checkout <branch>

**What is difference between Revert and Reset?**

* Revert command is more ideal way working in collaborative repos, because it won’t remove the existing commits, a new commit will be created with undo changes

git revert <commit-id>

* Reset command will remove all existing commits occurred after the resetting commit. So you will loose all commit details.

git reset <commit-id>

**How do undo specific commits?**

Number Hash Commit Message Author

1 2c6a45b (HEAD) Adding public method to access protected method Tom

2 ae45fab Updates to database interface Contractor 1

3 77b9b82 Improving database interface Contractor 2

4 3c9093c Merged develop branch into master Tom

5 b3d92c5 Adding new Event CMS Module Paul

6 feddbb Adding CMS class and files Tom

7 a809379 Adding project to Git Tom

***Using Cherry Pick***

Step 1: Find the commit before the commit you want to remove git log

Step 2: Checkout that commit git checkout <commit hash>

Step 3: Make a new branch using your current checkout commit git checkout -b <new branch>

Step 4: Now you need to add the commit after the removed commit git cherry-pick <commit hash>

Step 5: Now repeat Step 4 for all other commits you want to keep.

Step 6: Once all commits have been added to your new branch and have been commited. Check that everything is in the correct state and working as intended. Double check everything has been commited: git status

Step 7: Switch to your broken branch git checkout <broken branch>

Step 8: Now perform a hard reset on the broken branch to the commit prior to the one your want to remove git reset --hard <commit hash>

Step 9: Merge your fixed branch into this branch git merge <branch name>

Step 10: Push the merged changes back to origin. WARNING: This will overwrite the remote repo! git push --force origin <branch name>

You can do the process without creating a new branch by replacing Step 2 & 3 with Step 8 then not carry out Step 7 & 9.

Example

Say we want to remove commits 2 & 4 from the repo.

git checkout b3d92c5 Checkout the last usable commit.

git checkout -b repair Create a new branch to work on.

git cherry-pick 77b9b82 Run through commit 3.

git cherry-pick 2c6a45b Run through commit 1.

git checkout master Checkout master.

git reset --hard b3d92c5 Reset master to last usable commit.

git merge repair Merge our new branch onto master.

git push --force Push master to the remote repo.

**What is fetch command does?**

Fetch brings the changes to local machine and then you can merge it

* Example – Fetch the origin master and merge to local master
* git fetch origin master - fetch the origin master to local machine
* git checkout origin/master – switch to origin master and view the changes
* git checkout master – switch to master branch
* git merge origin/master – merge the origin/master to local master