Software Configuration Management With Git

Rohan Elukurthy

Number one question we should ask before using any library/software

https://github.com/27tang/HackathonCDK2015/network

Clone

```
git clone https://github.com/THEhEROLocker/git-workshop
```

Status

git status

- Displays paths that have differences between the index file and the current HEAD commit
- If in doubt, run git status. It will point you in the right direction

Other commands

Commit	git commit -m <message>
Branch	git branch
Switch Branch	git checkout <branchname>
Merge	git merge <branchname>

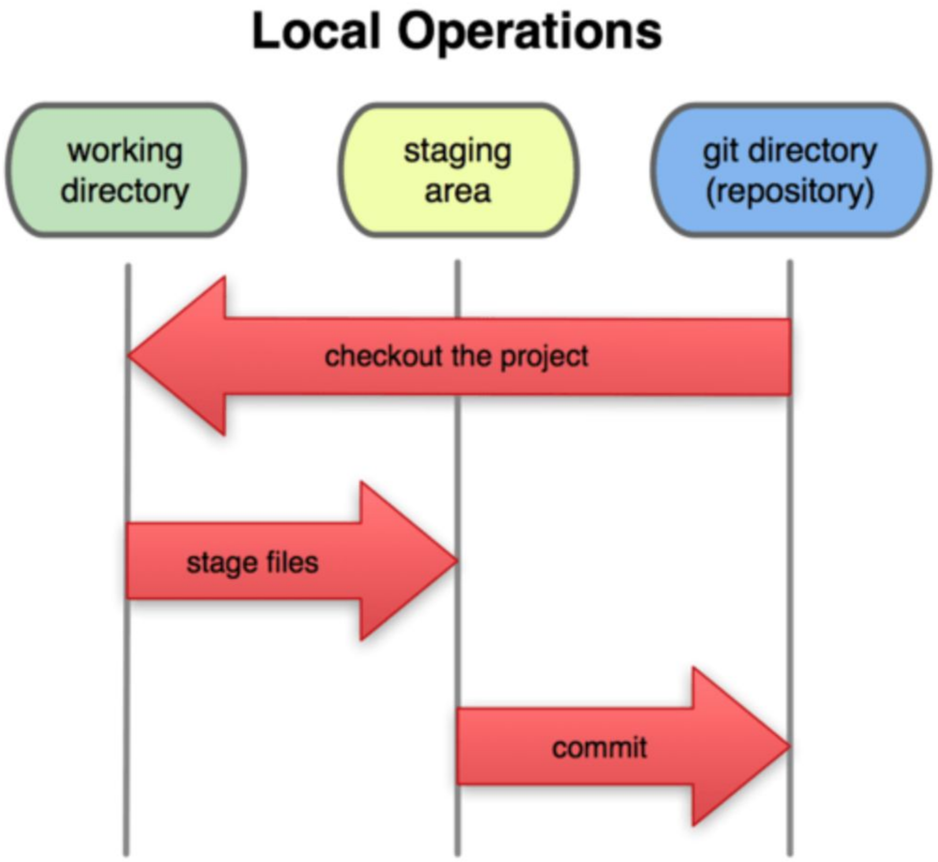

Illustration of the main three states your Git versioned file's lifecycle

Task 1: Basic Merge

1. Create a new branch. Name the branch as **new_feature**
2. Make sure you are on the new branch (run **git branch)**
3. Navigate to the Code Directory (called Task1)
4. Insert code that makes the new feature work
5. Test the code by running the program
6. Add and Commit the new changes
7. Run git Diff
8. Now Merge with master

https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging

Issues that could arise with merge

Rebase

- Rebase is recreating your work of one branch onto another
- For every commit in feature branch, a new commit will be created, on top of where the branch is being rebased to.

http://gitready.com/intermediate/2009/01/31/intro-to-rebase.html
https://randyfay.com/content/rebase-workflow-git

Task 2: Merge with rebase to avoid merge commit

1. Create branch with <branch1> . Start solving Interview question 1 here.
2. Let’s say you got stuck halfway into solving the problem.
   For the sake of the presentation let’s assume **you get stuck!** :D
3. Switch branch to Master and start a new branch, <branch2> to solve Interview question 2
4. Solve Interview question 2 and follow steps in Task 1 to merge it onto master
5. Switch to <branch1> and finish coding it. Add and Commit the changes
6. Note: Master is now ahead of when we branched off to create <branch1>
7. Perform rebase
8. Merge with master
9. Look at git log

Git workflow

- Branch Often
- Commit Often

Task 3: Squashing Commits with Rebase

1. Create <branch1>. Switch to Task3 directory
2. Finish Line 1 in file . Add + Commit
3. Finish Line 2 in file . Add + Commit
4. Finish Line 3 in file. Add + Commit
5. Check the log
6. Squash the three commit on this branch, into one commit
7. Switch branch to master
8. Rebase with <branch1>
9. Check history

http://gitready.com/advanced/2009/02/10/squashing-commits-with-rebase.html

Some Useful Commands

1. **git grep <regex>** // look for stuff through all files that are being tracked by git
2. **git checkout -f** // Throw away all local changes since last commit
3. **git reset** // Resets the index (i.e., removes all changes staged for commit)
4. **git checkout -** // quickly reset to previous branch
5. **git diff --cached** // Show changes staged for a commit
6. **git checkout -- <filename>** // Abort changes of of file
7. **git revert HEAD^** // undo last commit

https://github.com/git-tips/tips