git best practices

# setting up a repository

# linking to remote source

# discarding local changes

git reset --hard

# make commits feature-wise

It is important that each commit houses one change to the project. This is useful if you want to only merge certain changes to another branch later on, or see exactly when a bug was born and in many more cases. Very often, in the later stages of the project, programmers use tools to split one commit into several to separate them functionally.

For example, since the last commit if I have made two changes: tweaking UI elements on a form, and implementing a PHP best practice across all files, these two changes have to be saved as different commits. To do this, 1) see all changes to a project 2) see what changed in each of these files 3) stage only related changes and 4) commit them one after another.

git status // see all files that were changed since the last commit   
git diff file.php // see the difference in each file from that list  
git diff file.html // see the difference in each file from that list  
git add file1.txt // add only specific files to the stage  
git commit –m “msg” // commit the files in the stage

Tired of spaces coming up on difference?

git diff -–ignore-all-space file.php

# managing remote server

To collaborate with other people on a repository, it is necessary to host your files on a common server. These servers are called ‘remote’ in Git language. When you clone a repository a remote is automatically added, and named ‘origin’. If you initialize a repository, you would have added a remote branch. Here are some common commands to manage a server.

git remote // see all remote servers   
git remote show origin // see all branch-wise mapping to remote server named ‘origin’

# working on a new computer

# setting up Git on a live server

The first step is to determine the operating system that is running on the server.