# TheNewBoston Tutorials

Toturials are from: <https://www.youtube.com/watch?v=cEGIFZDyszA&list=PL6gx4Cwl9DGAKWClAD_iKpNC0bGHxGhcx>

If tutorials are skipped, it’s because there’s no important info or I already know it/completed the things.

## Tutorial 2 – Config Username and Email

* “*git --version"*
* “*git help*”
* “*git help <command>”*
  + For help on a specific command
* *“git config --global user.name “Matthieu Capuano””*
* *“git config --global user.email “capuanomat@gmail.com”*
  + Both of these are to change global settings
* “*git config --list"*
  + To view all your global settings
* “*git config <global setting>”*
  + To view a specific global setting from the list above

## Tutorial 3 – Creating Repository

* “*git init”*
* *“ls -la*”
  + To show hidden items as well

## Tutorial 4 – Commit

* *“git add .*”
  + To add everything to staging environment?
* “*git commit -m “<message>”*

## Tutorial 5 – Adding Files and the Commit Log

* “*git log”*
  + View history of commits
* “*git log –author=“<username>”*
  + To view changes made by a specific person
* “*git status”*
  + Tracks changes, will tell you if you have any that aren’t committed
  + If the file existed in the past, it will say “Changes not staged for commit”
  + If you just created the file, it will say “Untracked files”

## Tutorials 6/7 – Git Workflow / How to Edit Files

* “*git add <file>”*
  + Will add the file to the staging area, meaning that these will be committed
* “*git add .”*
  + Will add all the files that have been modified to staging area
* You need to add files to your staging environment before committing them

## Tutorials 8/9 – Viewing Changes You Made / Comparing Staging Area with Repository

* “*git diff”*
  + Will show you the difference between original and changed file
* “*git diff --staged"*
  + Will show difference between staging area and repository

## Tutorials 10/11 – How to Delete Files

* “*git rm <file>”*
  + Will delete that file from working directory, **you still need to commit though**
* “*git mv <oldname> <newname>”*
* If you rename a file in the GUI instead of git, it will see it as a deleted file and a new one
  + You have to git add the “new” one and git rm the “deleted” one for git to see that you just renamed it
* “*git mv <oldname> <newlocation/newname>”*
  + You can use this to rename and/or move a file to a new directory

## Tutorial 13/14 – How to Commit Directly to the Repository / Checkout this Video (Git it?)

* “*git commit -am “<Message>”*
  + Commits tracked changes directly to repository without staging environment
  + WARNING: This will commit changes for ALL tracked files
  + USAGE: Only use it when making small edits to files, NOT when deleting/adding files
* “*git checkout -- <file>”*
  + Makes that file the working copy, basically undoes the changes since last commit for that file

## Tutorial 15 – Unstage Files

* “*git reset HEAD <file>”*
  + Will remove a file from the staging environment

## Tutorial 16 – Getting Old Versions from the Repository

* “*git checkout <hashcodeOfPreviousCommit> -- <file>”*
  + This will recover a specific file from a previous commit, you then need to commit

## Tutorials 17/18 – Github / Pushing to a Github Repository

* **“*git remote add <nickname> <URL>”***
  + This is to push content on your computer into an existing online repository
  + The “nickname” is what you’ll use to refer to this online repo instead of the full URL
* “*git remote”*
  + Will show you the nickname
* **“*git push -u <nickname> master”***

## Tutorial 19/20 – gitignore and Github Desktop / Committing Changes to GitHub

* You can use the Desktop version to download and track changes from any of your repos
* If you want it to ignore some files create a new file called “.gitignore”
  + In this file, just write the directory or name of files you want to ignore
* You commit changes and publish them (sync them) from the desktop version…

## Tutorial 21 – Branches

* You can start a branch online on github
* A “pull request” asks to merge that branch with the main one

# Lynda.com Tutorial

* git --version

**Initializing, adding, committing, and status**

* Navigate to the directory and do: “*git init*”
  + If it has a hidden .git folder, then it’s been initialized
  + Check by typing “ls -la”
  + It’s where Git keeps track of things going on in your project
* Git has different places where you can store the state of files
  + Staging environment: Add things with “*git add index.html*”
    - “*git add .”* will add everything in current directory
  + You can check which are being tracked with “*git status*”
* Coming with “git commit -m “add your message””
* “*git log*” allows you to look at commits that have been made

**Working with the Staging environment**

* Doing “git checkout <filename>” brings you back to the previous version of the file (the last committed version
* If you git add the files, then git status will show it as ready to be committed because it’s in your environment (it’s like saving to your computer, printing would be committing)
  + If you make some more changes after, it will tell you that it’s not ready to be committed
* “*git reset HEAD <filename>”* will remove the file from the staging environment, but it won’t revert the changes you’ve made
  + It’s useful if you added the changed file to staging environment, and then made more changes so you don’t want those first changes anymore

**Deleting files**

* If you deleted a file, then do “git status” and it says that it’s deleted, you can:
  + Get the file back with: “*git checkout <filename>*”
  + Commit the deletion by adding to the staging environment “*git add <filename>”*
    - Again you can unstage with “*git reset HEAD <filename>*”
* You can delete and move it into staging with “*git rm <filename>*”
* Doing “*git add –all”* will just move everything to the staging environment (useful if you’ve added, edited, deleted lots of files)

**Managing your log**

* You can go back to an old version of your file and make changes from there with “*git checkout <hashcode of previous commit>*”
  + If you make changes and save them, they will be saved as a branch
  + You do that the same way though (git add, git commit)
* “*git branch”* - to look at all branches
  + An \* next to a branch means that’s the one you’re in right now
* “*git checkout master*” – will allow you to go back to the original root branch, it will also give you the option to save all those commits as actual branches
  + “*git branch <branch name> <hascode from checkout master>*” – will allow you to save that alternate branch
* “*git checkout <branch name>”* – allows you to go to a branch

**Controlling state with branches**

* “*git branch <new branch name>”* – creates a new branch, you stay in your current one though
  + This is useful to save the state of everything you’ve done so far
  + Also useful if you want to test out a new feature without adding it to the main branch
* “*git merge <branch name>”* – You must do this from the m aster branch, it will merge the branch name with the master branch
* “*git branch -m <old branch name> <new branch name>”* – will change the name of a branch
* “*git branch -D <branch name>*” – will delete a branch

**Cloning individual branches**

* “git clone <url>”
  + “*git clone -b base\_project <url>*” will clone the “base\_project” branch
* “*git branch -a”* – will show you all the actual branches, you only created a local copy of the main one
* “*git checkout -b <branch name> <path on github>”* – will copy that branch onto your local directory
* “*git clone --mirror <url> .git*” – will grab the entire .git folder
  + Then go into .git do *“git config –bool core.bare false*”
  + “*git reset --hard”* – grabs all the branches, so your local directory is like the online one
* If the github project has some automation, you can use “gulp” to run it
  + There’s also “npm install”??

**Using GitHub Branch as a Template**

* If you download a repository or a branch of a repository and want to use that as a starting point, you need to delete the .git file
  + “*rm -dfr .git*”
  + “*git init*” – you need to re-initialize it then