SUMMARY OF UDEMY COURSE

GIT: BECOME AN EXPERT IN GIT AND GITHUB (9 SECTIONS)

RATING: 5/5

SECTION 1 – INTRODUCTION (5 UNITS)

UNIT 1: Intro to Course

- Course overview
- You have to be able to use Terminal to use Git properly
- Branches an integral feature in Git
- Source Tree: a Git GUI

UNIT 2: What is GIT

- Source control/ Version control: is a way of tracking your files progress over time
- SC allows us to:
    1. Prevent data loss/ damage by creating backup snapshots
    2. Manage Project Structures
- Linux one of the biggest project on GitHub
- Git is a source control software:
    1. Most popular
    2. Lot of documentation and support
    3. Integration with other applications (Source Tree, Heroku (hosting), GitHub)

UNIT 3: Git vs GitHub

- GitHub: an application allowing you to store remote repositories
- Interact with your GitHub repositories through push/ pull on your Local Machine
- GitHub is used primarily to allow other people to add to the Project (Open Source)
- Git is the bone and flesh of Source control, while GitHub gives you the Platform to work with your repository

UNIT 4: Installing Git

- Install Git: Google “Git”
- Open GitBash > “git __version” > to test Git
- To get Ubuntu Shell > Control Panel > Programs > Turn Windows Features On/ Off > Windows Subsystem for Linux
- Microsoft Store > “Ubuntu” > install Ubuntu

QUIZ 1: Section Quiz

SECTION 2 – THE TERMINAL (6 UNITS)

UNIT 1: Update about Terminal Section (Article)

UNIT 2: Section Introduction

- Section focus on learning the Terminal
- Look at the different parts of the Terminal and what it can do
- Basic commands you can use in the Terminal
- Basic navigation commands

UNIT 3: Introduction to Terminal

- What is the Terminal: a way to operate with OS directly
- "Commad Prompt": Windows Terminal
- Source Tree gives a more User Friendly way of using Git
- Ubuntu Shell, shell and terminal are virtually same thing
- Ubuntu Shell has different commands compared to CMD
- All terminals have a home directory
- "clear": to clear commands
- "cd -": to go back to Home Directory

UNIT 4: Moving Between Directories

- "cd": Change Directory command
- "cd + name of path/name of folder"
- It is case sensitive, tab auto-complete
- "cd ..": takes you back a level (directory)
- Right click to paste into the Ubuntu shell
- "cd mnt" > "cd c" (enter C drive) > "cd Users" > and so on

UNIT 5: Working with Files and Directories

- "ls": to list all the files in your current directory
- Directory means Folder
- "mk dir + name of folder": make directory command
- "rm –rf +name of folder": to remove folder
- "git --version": to check git version
- "touch +filename with extension": create files
- "rm +filename": to delete file
- "touch" does not work in Windows Powershell
- Powershell does not do a good job of mimicking the Linux shell

QUIZ 2: Section Quiz

SECTION 3 – GIT BASICS (11 UNITS)

UNIT 1: Section Introduction

- Bit of theory behind the Git workflow and how it connects together
- Learn different git commands

UNIT 2: Git Cheat sheet (downloads)

- You don't need to memorize the commands, it is important to understand what they do and how they work, rather than memorizing the syntax
- https://github.com/joshnh/Git-Commands

UNIT 3: Git Workflow

- Before Git tracks a change, it goes through a long chain of operations and tasks
- Repositories store the Full History and Source control of a Project
- Any Repos stored somewhere other than locally is called a "remote repository"
- Repos are timelines of the entire project including all previous changes
- Directories or Working Directories are projects at their current state in time
- Commit: a similar to taking a snapshot of the current state of the project, then storing it on a timeline
- Working Directory < > Staging Area < > Local Repository < > Remote Repository

UNIT 4: Creating a New Repository

- Create a folder > Turn it to a Git Repos
- "git init": to initialize a git repository, creates a ".git" folder, be sure you are in the right folder

UNIT 5: Adding and Removing Files

- Enclose folder/ path names with spaces in > " * "
- "git status": to check the status of the Staging Area
- Git is not exclusive to Tech Stuffs only it can be used for anything (e.g. Essay writing)
- "git add +filename": to add file to Staging Area
- "git add .": to add all files to Staging Area
- "git rm -f": to force remove from Staging Area, completely wipes it out
- "git rm –cached": to untrack a file, remove it from the staging area

UNIT 6: Your First Commit

- Commit allows you to take a snapshot of all the changes you have done at the time and store it to a tree
- "git commit": has different flags
- "git commit --help": sort of a a documentation, better to use Google
- "git commit –m "**" ": to add a message to the Commit

- “git commit –a –m ”: to commit only the tracked files
- “git log”: brings out all of our commits

UNIT 7: Git Checkout

- Source control helps to track history
- Commit ID, messages in commits are very useful
- “git checkout commitID”: to checkout a particular commit
- Never create a repository directly on your C: drive
- “git checkout master”: to go back to current state

UNIT 8: Git Revert & Reset

- To go back in time PERMANENTLY
- Git Revert safer than Git Reset
- “git revert commitID”: undo the changes of the commit and creates a revert copy
- You can revert a revert (nested shii)
- Git reset goes all the way back and stays there
- Soft git reset: like a checkout
- Mixed git reset, Hard git reset
- “git log --oneline”: shows commits in one line each
- “git reset --hard”:

UNIT 9: Types of Git Reset

- “--mixed” (confusing, don’t grab): useful primarily when you want to remove a file you accidentally committed
- “--hard”: most commonly used

UNIT 10: Creating a “ .gitignore”

- “gitignore”: a list of files or directories you don’t want to track
- “create a file “.gitignore” (touch .gitignore) ” > Add files to ignore
- Add gitignore at the start of the project
- “git rm –r –cached .”: to remove all the cached files
- “folder/*”: all the files in folder
- You should use gitignore in all your projects

QUIZ 3: Section Quiz

SECTION 4 – GIT BRANCHES (6 UNITS)

UNIT 1: Section Introduction

- Focused on branches in Git
- Branches can get complex in the Terminal

UNIT 2: What are Branches & why should you use them?

- Git branches are a way to create separate development paths without overriding or creating copies of your project
- Branches can be added, deleted, and merged just like regular commits
- 4/ 5 branches max

UNIT 3: Working with Branches

- ”git checkout –b +branchname”: to create a branch and switch to it
- “git checkout master”: to switch back to master branch
- “git branch +branchname”: to just create a branch without switching to it
- “git branch -a”: list all the available branches
- “git checkout +branchname”: to switch to a particular branch
- “git branch –d +branchname”: to delete a branch

UNIT 4: Editing Branches

- Adding commits to branches
- Switch to branch > Add > Commit
- Git helps to organize the complexity of your works
- Be careful about what you commit

UNIT 5: Merging Branches

- Without merging Branches are kind of useless
- “git merge +branchname”: to merge branchname into current branch

QUIZ 4: Section Quiz

- Branch is a chain of commits that are on a separate timeline from the master, therefore, they do not have conflicting timelines

SECTION 5 – GITHUB (7 UNITS)

UNIT 1: Section Introduction

- How to use GitHub, forking, pull request
- History behind GitHub, create and upkeep repositories

UNIT 2: What is GitHub?

- GitHub is an application that allows you to store remote repositories on their servers
- GitHub allows you to:
    1. Have a portfolio of your best work
    2. Open opportunities you might not have with a local repos
    3. Easy access from any device in any location

4. Industry standard for hosting Git repos

UNIT 3: Creating a GitHub Account

- “github.com”

UNIT 4: Creating our First GitHub Repository

- I know this already
- License (MIT license), “.gitignore” templates
- Issue: listing of issues
- Pull requests: anyone looking for permission to pull repo to local repos
- Wiki: to layout a kinda summary, documentation
- Insight: More features about your repos, similar to Google Analytics
- Settings:
- Collaborators
- Branches
- Webhooks, Deploy Keys (when using Heroku in order to deploy)

UNIT 5: Viewing Other Repositories

- “watching”: notifies you of updates on the Repos
- “fork”: makes a copy and put directly in your account
- “cloning”: taking a link using it to take and make a copy of it on your local machine
- He recommends forking

UNIT 6: Download GitHub Repos

- Downloading: does not include the git source/ history files

QUIZ 5: Section Quiz

SECTION 6 – USING GIT REMOTELY (6 UNITS)

UNIT 1: Section Introduction

- We will use our Git and GitHub skill together to bring our project to the world
- Different commands we can use from terminal to connect to GitHub

UNIT 2: Creating a new Remote Repository

- Create in GitHub (blank repo) > Connect it to local machine
    1. Create a “git init”, initialize git on Local Machine
    2. Use “git remote add origin +HTTPLinkfrom GitHub”
- “git remote -v”: to check if it is connected (brings fetch/push)

UNIT 3: The Push and Pull System

- “fetch”: mean “pull”, pulling from Remote Repo to Local Repo
- “push”: pushing into Remote Repo from Local Repo
- Push and Pull cycle

UNIT 4: Pushing and Pulling to and from a GitHub Repos

- Google search in case of Conflicts/ Errors
- “git pull origin master”: to track the master branch
- “git pull”: after tracking the branch
- “ls”: list of files in directory
- “rm –rf +Foldername”: to remove folder
- “git push –u origin master”: to push a branch and remember the branch
- “git push”: to push to the remembered branch

UNIT 5: Deleting Remote Branches

- We will usually create branch to fix error/revisions and then merge it back
- You have to push each branch individually
- “git push origin +branchname”: to push a particular branch
- “git push origin –delete +branchname”: to delete a branch (pushes a delete protocol)
- Check out the Git Cheat Sheet

QUIZ 6: Section Quiz

SECTION 7 – GIT GUI w/ SOURCE TREE (11 UNITS)

UNIT 1: Section Introduction

- How GUI are better than Terminals
- Features of Source Tree, Difference between GUI and Terminal
- Remote Features, bring Git and GitHub together in Source Tree

UNIT 2: What is SourceTree

- GUI: Graphic User Interface, allows you to perform task through a graphical interface
- SourceTree: allows you to interact with Git through a graphical interface
- It executed through icons and buttons instead of terminal commands

UNIT 3: Installing SourceTree

- Google “SourceTree” and Download > Create Account

UNIT 4: Setting up a new Repository

- Create > Select Directory > Select Git

UNIT 5: Introduction to the SourceTree Environment

- Play with the Software
- Icons and Buttons instead of Terminal Commands

UNIT 6: Stage and Commit

- Play with SourceTree

UNIT 7: Interaction in SourceTree

- Checkouts, Revert
- Latest commit is called the Tips
- “Reverse Commit”: same as Revert
- “Reset Current Batch to commit”: same as Reset

UNIT 8: Create and Remove Branches

- Create Branch, Delete Branch

UNIT 9: Merge Branches

- Be “Checkedout” on the branch you want to merge into

UNIT 10: Push/Pull Requests

- Create a brand new Repo > “git remote add origin +url” on Terminal > Push/Pull
- “origin/branchname”: refers to remote repos

QUIZ 7: Section Quiz

SECTION 8 – GOODBYE (2 UNITS)

UNIT 1: Goodbye

- Git is not the only source control you can use
- Continue to learn

UNIT 2: Continue your Learning

- Sign up for VideoLab Digest https://www.videolab.ae/digest/

SECTION 9 – BONUS LECTURES (2 UNITS)

UNIT 1: Note about Bonus Lectures

UNIT 2: Git Rebase

- Merging: Create one new commit that combine all the changes of the branches
- Rebase: Moving the changes to the tip of the branch you are merging into

- Move a branch unto the tip of another branch
- “git rebase master”: rebase from a branch to the master branch
- He recommends sticking with “merge”